OpenGamma Quantitative Research

Algorithmic Differentiation in Finance: Root Finding and Least Square Calibration

Marc Henrard
marc@opengamma.com

Abstract

Algorithmic Differentiation (AD) is an efficient way to compute derivatives of a value with respect to the data inputs. In finance the model calibration to market data can be an important part of the valuation process. In presence of calibration, when obtained through exact equation solving or optimisation, very efficient implementation can be done using the implicit function theorem with the standard AD approach. Previous results discussed the exact case are here extended to the case of calibration obtained by a least-square approach.

Contents

1 Introduction **1**

2 Adjoint method and implicit function theorem **3**

3 Perfect Calibration Technique **4**

4 Perfect Calibration Examples **5**
4.1 Cash swaptions in the Hull-White model 5
4.2 Amortised swaptions in LMM 6

5 Least Square Calibration technique **8**

6 Least Square Examples **11**
6.1 Reference 11
6.2 Tenors 12
6.3 Strikes 12

7 Conclusion **13**

1 Introduction

In quantitative finance, computing the present value of financial instruments is only part of the game; most of the (computer) time is spent in calculating the derivatives of the present value with respect to the different financial inputs, the so-called *greeks*.

For the computation of present value derivatives, one efficient technique is *Algorithmic Differentiation* (AD) and in particular its *Adjoint mode* (AAD). The method is efficient in two senses: it is fast and provides results with machine-precision accuracy.

The adjoint method was popularised in finance by Giles and Glasserman (2006). Since then, the technique has been applied in different contexts in finance, in particular for the Libor Market Model in Denson and Joshi (2009a) and Denson and Joshi (2009b), Monte Carlo-based calibration in Kaebe et al. (2009), correlation risk in credit models in Capriotti and Giles (2010), and Monte Carlo credit risk in Capriotti et al. (2011). We also refer to Capriotti (2011) and the references therein for the general technique and its use for derivatives computations in finance.

In theory, applying the adjoint method to compute a single price (P) and all its derivatives (D) should lead to a relative time cost (Griewank and Walther, 2008, Section 4.6)

$$\frac{\text{Cost}(P+D)}{\text{Cost}(P)} \leq \omega_A$$

with $\omega_A \in [3,4]$. This relative cost can be compared to the *finite difference* (also called *divided difference* or *bump and recompute*) method, another popular technique for derivative computation. With the finite difference technique, the relative cost depends on the number of derivatives and is

$$\frac{\text{Cost}(P+D)}{\text{Cost}(P)} \simeq \text{number of derivatives} + 1.$$

Even for simple financial instruments, the number of derivatives can be large. For a simple vanilla 10-year swap in EUR, the number of derivatives is 41 in a standard multi-curves framework. The advantage of the AAD in term of computational cost is obvious.

In practice, for simple functions, it is in general possible to do better than the theoretical upper bound. For example the Black function option price and its derivatives (w.r.t. the forward, the volatility and the strike) can be computed with a ratio of 1.1[1], i.e. the three derivatives can be computed by increasing the computation time by only 10%. Part of the saving comes from using domain specific knowledge. In the Black formula, the derivative of the price with respect to the exercise boundary is zero as the boundary is the optimal one. This type of simplification is not infrequent in finance; it appears in European swaption pricing in the Hull-White model, Bermudan swaption pricing and in many other places.

In practice, the AD method is of greater interest when considering complex processes. One frequent part of such a process for exotic option pricing in finance is *model calibration*. The process is as follows:

- the price of an *exotic instrument* is related to a specific basket of *vanilla instruments*;
- the price of these vanilla instruments is computed in a given *base model*;

[0] First version: 7 September 2012; this version: 9 January 2013.

[1] All figures related to actual implementations have been computed with the OpenGamma OG-Analytics library. The OpenGamma analytic library is open source and available at http://developers.opengamma.com.

- the complex model parameters are calibrated to fit the vanilla option prices from the base model. This step is usually done through a generic numerical process (root finding or least square); and
- the exotic instrument is then priced with the calibrated complex model.

In the algorithmic differentiation process, we suppose that the pricing algorithm and its derivatives are implemented. We want to differentiate the exotic price with respect to the parameters of the base model. In the bump and recompute approach, this corresponds to computing the risks with model recalibration. As the calibration can represent a major part of the total computation time of the procedure listed above, an alternative method is highly desirable. The subject of this note is the calibration process part. It is intended to be complementary to the literature mentioned above. The first part of the note was already described in Henrard (2012); the second part on least square calibration is new.

In general, the calibration process is not explicit, it is done through a numerical equation solving or least square approach. We have only one set of parameters of the calibrated model; the set corresponding to a specific set of curves and base model parameters. There is no explicit algorithm that provides the calibrated model parameters as function of the base model, or its the adjoint algorithmic differentiation.

What is demonstrated in this note is that the derivative of the calibration process is not required; indeed the calculation is performed more quickly without it when the structure of the problem is used.

To show this, *implicit function* theorem is used to analyse equations of the form

$$f(x, y) = 0.$$

Theorem 1 *Le $f : \mathbb{R}^{n+m} \to \mathbb{R}^m$ be continuously differentiable. If (x_0, y_0) is such that*

$$f(x_0, y_0) = 0$$

and if $D_y f(x_0, y_0)$ is invertible, then, in a neighbourhood $X \times Y$ of (x_0, y_0), there is a (implicit) function g such that $f(x, g(x)) = 0$ for $x \in X$, $\{(x, g(x)) : x \in X\} = \{(x, y) \in X \times Y : f(x, y) = 0\}$, g is differentiable in x_0 and

$$D_x g(x_0) = -\left(D_y f(x_0, y_0)\right)^{-1} D_x f(x_0, y_0).$$

In general, the numerical resolution of the equation $f(x, y) = 0$ requires several evaluations of f. This is the case for iterative Newton-like methods. According to ADD, the derivatives of f are computed in a time not greater than the time to perform four evaluation of f (in the one calibration instrument case). Thus, the derivatives of f can be computed faster than the calibrating value y_0. The derivatives of the implicit function, through the above formula, can be obtained in a time which is less than the pricing time (subject to the price being computed already).

An independent related approach, using an Automatic Differentiation tool (ADOL-C), is proposed in Schlenkirch (2012). The model used in their paper is the Hull-White one factor model in a textbook single-curves framework. The problem analysed is the calibration to European swaptions and pricing of Bermudan swaptions.

The idea of combining ADD and the implicit function theorem has been present in AD for some time (see for example Christianson (1998)). In addition to calculating the derivative through equation solving, the referenced paper also analyses error estimates for the function computed.

Similar ideas are used in other applied mathematics fields. In particular, Giles and Pierce (2000) used it in engineering design; their formula for objective function derivatives is similar to the derivatives of the exotic instrument price with respect to the curves in the base model presented in Section 3. To our knowledge, the approach has not yet been used in the financial calibration context. The least sure calibration approach appears also to be new.

We describe the improved efficiency of the calculation of the derivatives below. We provide several examples and show that the efficiency described in theory can be obtained in practice. In the examples, a present value with calibration and its 40 to 70 derivatives are obtained in a time which is below twice the pricing time (and so well below the theoretical upper bound). In the least square case, the ratio is bearly above 1. The computation of the derivatives is almost free.

2 Adjoint method and implicit function theorem

The method is first presented using a simple example, allowing simplified notation. Suppose we have a a function $f : \mathbb{R}^{p_a} \to \mathbb{R}^{p_z}$

$$z = f(a).$$

Within the algorithm to compute f, there is an equation to solve. The algorithm is decomposed into

$$\begin{array}{lcl} b & = & g_1(a) \\ c & \text{s. t.} & g_2(b,c) = 0 \\ z & = & g_3(c) \end{array}$$

with $g_1 : \mathbb{R}^{p_a} \to \mathbb{R}^{p_b}$, $g_2 : \mathbb{R}^{p_b} \times \mathbb{R}^{p_c} \to \mathbb{R}^{p_c}$ and $g_3 : \mathbb{R}^{p_c} \to \mathbb{R}^{p_z}$. The second part of the algorithm is a multi-dimensional root-finding problem that need to be solved.

It is assumed that all functions are differentiable. The derivative of $f : \mathbb{R}^{p_a} \to \mathbb{R}^{p_z}; a \mapsto z = f(a)$ at the point a is denoted $Df(a)$ or $D_a f(a)$ if we want to emphasise the variable with respect to which the derivative is taken. The elements of $\mathbb{R}^p$ are represented by column vectors. The derivative $Df(a) \in \mathcal{L}(\mathbb{R}^{p_a}, \mathbb{R}^{p_z})$ is represented by a $p_z \times p_a$ matrix (p_z rows, p_a columns).

Suppose that the AD versions of the functions g_i ($1 \leq i \leq 3$) are known but the adjoint version for the solver is unknown, i.e. the derivatives of the function that computes c from b is unknown. The implicit function theorem ensures (under certain conditions) that the process that produces c as function of b is actually differentiable and links its derivative to that of g_2. Defining g_4 as the implicit (and unknown) function associating c to b, i.e. $g_4(b) = c$, the derivative of g_4 is given by

$$D_b g_4(b) = -\left(D_c g_2(b,c)\right)^{-1} D_b g_2(b,c).$$

Solving the equation is usually much more time-consuming than simply computing one value of g_2. In the implicit function theorem approach, using the adjoint version there is no need to solve the equation again and there is no requirement to have AD version of the solver, only the adjoint version of the function g_2 is required. The AD method used in this way will give better results than the normal approach as there is no need to solve the equation for g_2 again. The time required to compute the price and all of its derivatives will be usually less than twice the time taken to calculate one price.

The standard notation in ADD is to denote the derivative of the final value z with respect to an intermediate value x by $\bar{x}$. The literature uses different notations with regards to the transposition of $\bar{x}$; here we use

$$\bar{x} = \left(D_x z(x)\right)^T,$$

i.e. the *bar* variables are column vectors if z is of dimension one, or matrices of size $p_x \times p_z$ otherwise. In our instrument price examples, the dimension of z is one.

The adjoint version of the algorithm can be written as

$$\begin{aligned}
\bar{z} &= I \quad \text{(with } I \text{ the } p_z \times p_z \text{ identity)} \\
\bar{c} &= (D_c g_3(c))^T \bar{z} \\
\bar{b} &= (D_b g_4(b))^T \bar{c} = -\left((D_c g_2(b,c))^{-1} D_b g_2(b,c) \right)^T \bar{c} \\
\bar{a} &= (D_a g_1(a))^T \bar{b}.
\end{aligned}$$

3 Perfect Calibration Technique

The method is applied with the interest rate model calibration case in mind. If the *yield curve* C is replaced by any other asset class data, like FX rates, equity prices or credit curves, the results are still valid.

Let $\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}$ be the prices of the vanilla financial instruments used for calibration in the base model. The data required for the pricing are the yield curves (denoted C) and market volatility parameters (e.g. SABR parameters or Black volatilities) for the base model (denoted Θ). The exotic model can price the same vanilla options with the same curves but using different parameters (denoted Φ). The pricing function for the vanilla options in the calibrated complex model is denoted $\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}$. The calibration equation to solve is

$$f(C, \Theta, \Phi) = 0. \tag{1}$$

For perfect calibration, the function is simply

$$f(C, \Theta, \Phi) = \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(C, \Theta) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(C, \Phi). \tag{2}$$

Equation (1) will be multi-dimensional when there are several calibrating instruments. We suppose that there are as many calibration instruments as parameters to be calibrated in Φ.

In practice, some models may have more free parameters than calibrating instruments. In this case the model parameters are constrained in such a way that there are the same number of degrees of freedom as the number of calibrating instruments. The second example in the next section calibrates a two-factor LMM with many parameters. We calibrate them by adding the constraint that for each yearly period, the parameters are multiples of a initially-given structure. For a 10 year coverage the model has 40 parameters and the actual number of degrees of freedom is 10, one for each calibrating instrument. The parameters Φ used here are those degrees of freedom, not the original model parameters.

With the calibration procedure, we obtain calibrated model parameters from the original model parameters Θ_0 and the curves C_0:

$$\Phi = \Phi(C_0, \Theta_0).$$

The parameters are obtained through the equation solving procedure; there is no explicit solution or even explicit code that produces those parameters directly.

The exotic option is priced from the calibrated model through the pricing $\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C, \Phi)$. With the implicit function above we can define

$$\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C, \Theta) = \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C, \Phi(C, \Theta))$$

We are interested in the derivative of the exotic option with respect to the curves and the base model parameters Θ.

With the AD versions of $\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}$, we can compute the derivatives

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}} \text{ and } D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}.$$

The quantities we would like to compute are

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}} \text{ and } D_\Theta \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}.$$

Through composition we have

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C_0, \Theta_0) = D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) + D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) D_C \Phi(C_0, \Theta_0),$$

and

$$D_\Theta \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C_0, \Theta_0) = D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) D_\Theta \Phi(C_0, \Theta_0).$$

Where $D_C\Phi$ and $D_\Theta\Phi$ are yet unknown. Using the implicit function theorem, the function Φ is differentiable and its derivatives can be computed from the derivatives of f:

$$D_\Theta \Phi(C_0, \Theta_0) = -\left(D_\Phi f(C_0, \Theta_0, \Phi(C_0, \Theta_0))\right)^{-1} D_\Theta f(C_0, \Theta_0, \Phi(C_0, \Theta_0))$$

and

$$D_C \Phi(C_0, \Theta_0) = -\left(D_\Phi f(C_0, \Theta_0, \Phi(C_0, \Theta_0))\right)^{-1} D_C f(C_0, \Theta_0, \Phi(C_0, \Theta_0)).$$

In the perfect calibration case

$$D_\Theta \Phi(C_0, \Theta_0) = \left(D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0))\right)^{-1} D_\Theta \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(C_0, \Theta_0)$$

and

$$\begin{aligned} D_C \Phi(C_0, \Theta_0) &= \left(D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0))\right)^{-1} \\ &\quad \left(D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(C_0, \Theta_0) - D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0))\right). \end{aligned}$$

4 Perfect Calibration Examples

In line with the above technique, we would like to price and compute the sensitivities of exotic swaptions in a physical delivery SABR framework. For all the required pricing algorithms the AAD versions have been implemented[2].

4.1 Cash swaptions in the Hull-White model

In the first example we chose the simplest case, with only one calibrating instrument and one parameter, to illustrate the approach. Our *exotic* instrument is a cash-settled swaption and our vanilla basket is composed of a unique physical delivery swaption. The curve framework is the standard multi-curves framework as described in Henrard (2010b) with deterministic spread. The

[2]The implementations used for the performance figures are those in the OpenGamma analytics library. The computations are done on a Mac Pro 3.2 GHz Quad-core. The test code is available from the author.

base model is a SABR model on the swap rate. The model parameters Θ are the SABR parameters α, ρ and ν (β is set to 0.50). The formula used for swaption in the SABR framework is the one from Hagan et al. (2002). The calibrated model is a Hull-White one factor (extended Vasicek) model with constant volatility. The parameter of the Hull-White model to calibrate is the constant volatility. The pricing algorithm in the Hull-White model for the physical delivery swaption is described in Henrard (2003), and the pricing algorithm used for the cash-settled swaption is the efficient approximation described in Henrard (2010a). The pricing algorithms are fast and the calibration is an important part of the comptation time.

For this example, we use a 1Y×9Y swaption on an annual vs 6m Euribor swap. There are three SABR sensitivities (α, ρ, and ν) and 38 rate sensitivities (19 on each curve). The performance results are provided in Table 1. The computation of the three SABR derivatives add less than 30% to the pricing computation time in this approach; a non-symmetrical finite difference computation would add 300%.

Risk type	Approach	Price time	Risks time	Total
SABR	Finite difference	1.00	3×1.00	4.00
SABR	AAD and implicit function	1.00	0.28	1.28
Curve	Finite difference	1.00	38×1.00	39.00
Curve	AAD and implicit function	1.00	0.56	1.56
Curve and SABR	Finite difference	1.00	41×1.00	42.00
Curve and SABR	AAD and implicit function	1.00	0.83	1.83
Times relative to the pricing time. The pricing time is 0.45 second for 1000 swaptions.				

Table 1: Performance for different approaches to derivatives computations: cash settled swaption in Hull-White one factor model.

The same comparison was performed for the interest rate sensitivities. The finite difference requires 39 price time (3900%). The proposed approach adds only 0.55 price time (55%). In total, the proposed algorithm is around 23 times faster than a finite difference approach and is numerically more stable. Note also that the total ratio 1.83 is well below the theoretical upper bound $\omega_A \in [3, 4]$.

To partly compare with the results of Schlenkirch (2012), we also report figures for a Hull-White one factor model with piecewise constant volatility. The set-up of the above article is different but the underlying model is similar. The computation time for the finite difference and adjoint evaluations of the Jacobian with respect to the piecewise constant volatility for a 30Y and 100Y swap (annual volatility dates) is provided. The Jacobian is the derivative of all European swaption prices with respect to all volatilities in the model. The 30Y Jacobian computation requires 0.110 seconds (s) by finite difference and less than 0.015 s by algorithmic differentiation. This is approximately 14% of the runtime and lower but of the same order of magnitude that the result of Schlenkirch (2012) (20%). The corresponding figures for the 100Y case are 20.2 s and 0.42 s (2%).

4.2 Amortised swaptions in LMM

In this example, the exotic instrument is an amortised European swaption (i.e. a swaption with decreasing notional), and the vanilla basket is composed of vanilla European swaptions with same expiry and increasing maturities. The amortised swaption has a 10Y maturity and yearly amortisation. The calibrating instruments are ten vanilla swaptions with yearly maturities between 1Y

and 10Y and same strike as the amortised swaption.

The base model is a SABR model on each vanilla swaption. The pricing in the SABR framework is done as in the previous example. The complex model is a two-factor LMM with displaced diffusion and Libor period of six months. The pricing method for the vanilla and the amortised swaptions is the efficient approximation described in Henrard (2010c).

The calibration is performed as follows: for each yearly period the weights of the different parameters (four in each year) are fixed. The calibration is done by multiplying those weights by a common factor. The parameter Φ in the previous section are the multiplicative factors (10 in total), even if in practice the derivatives with all the model parameters (40 in total) are computed as an intermediary step.

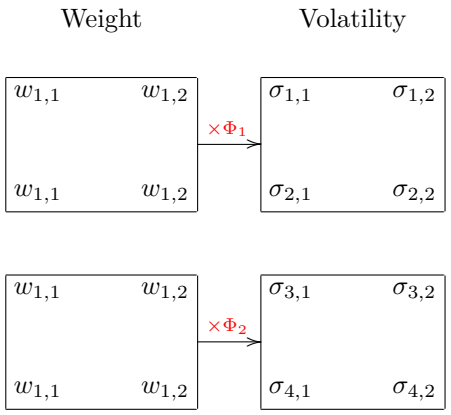


Figure 1: Representation of LMM calibration for a two years amortised swaption. Each yearly block is multiplied by a common multiplicative factor.

The results for the SABR and curve sensitivities are reported in Table 2. There are 30 SABR sensitivities (α, ρ, ν for 10 vanilla swaptions). In the described approach, the 30 sensitivities add less than 20% to the computation time with respect to the calibration and price computation.

Risk type	Approach	Price time	Risks time	Total
SABR	Finite difference	1.00	30×1.00	31.00
SABR	AAD and implicit function	1.00	0.18	1.18
Curve	Finite difference	1.00	42×1.00	43.00
Curve	AAD and implicit function	1.00	0.74	1.74
Curve and SABR	Finite difference	1.00	72×1.00	73.00
Curve and SABR	AAD and implicit function	1.00	0.75	1.75

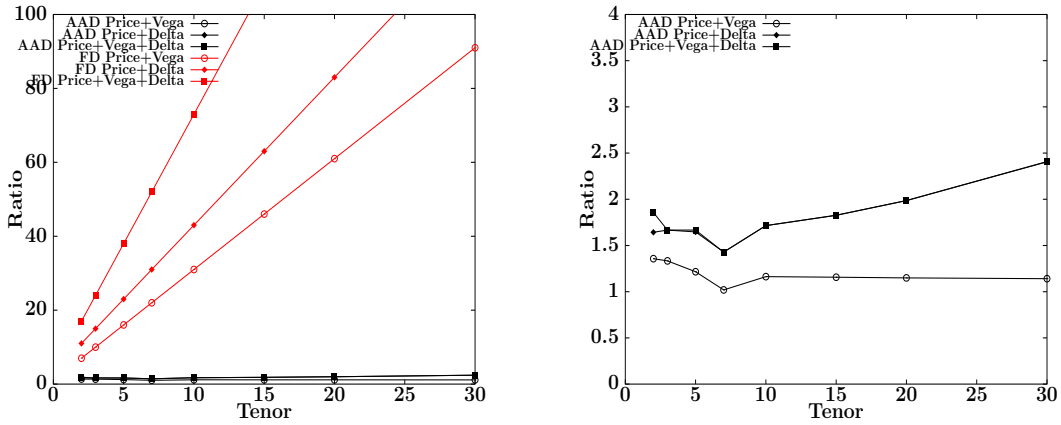
Times relative to the pricing time. The pricing time is 0.425 second for 250 swaptions.

Table 2: Performance for different approaches to derivatives computations: amortised swaption in the LMM.

There are 42 curve sensitivities (two curves, semi-annual payments over 10 years). The computation of the 42 sensitivities takes less than 75% of the price time. In total, the AAD approach

takes 2.5% of the time required by finite difference. Note that computing the curve and SABR sensitivities take approximately the same amount of time as computing the curve sensitivities as most of the computation are common. The total ration of 1.75 is well below the theoretical upper bound of $\omega_A \in [3, 4]$.

Similar results for amortised swaptions of different maturities and with different numbers of calibrating instruments are reported in Figure 2. The ratios between the price and sensitivities time and the price time are reported for the finite difference and Adjoint Algorithmic Differentiation using the implicit function method described in the previous section. The implicit function AAD method ratios are almost independent of the number of sensitivities. In all cases but the 30Y swaptions (where 212 sensitivities are calculated), the ratios are below two.



(a) Full picture

(b) AAD picture

Figure 2: Computation time ratios (price and sensitivities time to price time) for the finite difference and AAD methods. The *vega* represents the derivatives with respect to the SABR parameters; the *delta* represents the derivatives with respect to the interest rate curves. The AAD method uses the implicit function theorem approach. Figures for annually amortised swaptions in a two-factor LMM calibrated to vanilla swaptions in SABR.

5 Least Square Calibration technique

In this section we develop techniques similar to the one developed in Section 3 but for the case where the calibration is not a root-finding calibration but a least square calibration.

Here we consider the case were the parameters Φ are obtained through a (weighted) least square process. Suppose that there are n calibrated parameters in Φ and $m \geq n$ instruments for the calibration process. The weights associated to each instrument are $(w_i)_{i=1,\ldots,m}$. The calibration

parameters are defined as

$$\begin{aligned}\Phi_0 &= \arg\min_{\Phi\in\mathbb{R}^n} h(C_0,\Theta_0,\Phi)\\ &= \arg\min_{\Phi} \sum_{i=1,\dots,m} w_i \left(\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i,C_0,\Theta_0) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i,C_0,\Phi)\right)^2. \qquad (3)\end{aligned}$$

At the minimum Φ_0, the derivatives with respect to Φ are 0:

$$f(C_0,\Theta_0,\Phi_0) = D_\Phi h(C_0,\Theta_0,\Phi_0) = 0.$$

The minimum satisfies Equation (1) with the function f defined above. This last equation is a n unknown and n equation system. We use the following convention for derivatives: $D_X y$ is a vector (matrix) with one column for each element in X and as many line as the dimension of y. In particular $D_\Phi h$ is a line vector (n element). The above derivatives can be computed explicitly as

$$D_\Phi h(C,\Theta,\Phi) = -2\sum_{i=1,\dots,m} w_i \left(\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i,C,\Theta) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i,C,\Phi)\right) D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i,C,\Phi).$$

With the calibration procedure, we obtain calibrated model parameters from the original model parameters and the curves:

$$\Phi_0 = \Phi(C_0,\Theta_0).$$

The parameters are obtained through the optimization procedure; there is no explicit solution or even explicit code that produces those parameters directly.

The implicit function theorem states that there exists a function $\Phi(C,\Theta)$ such that

$$f(C,\Theta,\Phi(C,\Theta)) = 0$$

for (C,Θ) close to (C_0,Θ_0) and there are no other solution in a neighbourhood. We still need to prove that the function $\Phi(C,\Theta)$ gives a minimum of the original problem (3) and not only a point with 0 derivatives (like a saddle point).

Let m_0 denote the minimum value of (3) at Φ_0 i.e. $m_0 = h(C_0,\Theta_0,\Phi_0)$. As Φ_0 is a minimum with $D_\Phi f(C_0,\Theta_0,\Phi_0)$ invertible, there exists a $\epsilon > 0$ and a sphere around Φ_0 such that $h(C_0,\Theta_0,\Phi) > m_0 + 3\epsilon$ for Φ on the sphere. As f is continuous, for (C,Θ) close enough to (C_0,Φ_0), $h(C,\Theta,\Phi) > m_0 + 2\epsilon$ for Φ on the sphere and $h(C,\Theta,\Phi(C,\Theta)) < m_0 + \epsilon$ for (C,Θ) close to (C_0,Φ_0). This proves that h has a minimum in the interior of the disk, thus the minimum has zero derivatives. From the result of the theorem, $\Phi(C,\Theta)$ is the only zero. This proves that the implicit function $\Phi(C,\Theta)$ is not only a zero of the derivative but also a minimum of the least square problem.

The exotic option is priced from the calibrated model through the pricing $\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C,\Phi)$. With the implicit function above we can define

$$\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C,\Theta) = \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C,\Phi(C,\Theta))$$

We are interested in the derivative of the exotic option with respect to the curves and the base model parameters Θ.

The quantities of interest are

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}} \text{ and } D_\Theta \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}.$$

With the AD versions of $\mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}$, we can compute the derivatives

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}} \text{ and } D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}.$$

Through composition we have

$$D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C_0, \Theta_0) = D_C \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) + D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) D_C \Phi(C_0, \Theta_0),$$

and

$$D_\Theta \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Base}}(C_0, \Theta_0) = D_\Phi \mathrm{NPV}^{\mathrm{Exotic}}_{\mathrm{Calibrated}}(C_0, \Phi(C_0, \Theta_0)) D_\Theta \Phi(C_0, \Theta_0).$$

Where $D_C\Phi$ and $D_\Theta\Phi$ are yet unknown. Using the implicit function theorem, the function Φ is differentiable and its derivatives can be computed from the derivative of f:

$$D_\Theta \Phi(C_0, \Theta_0) = -\left(D_\Phi f(C_0, \Theta_0, \Phi(C_0, \Theta_0))\right)^{-1} D_\Theta f(C_0, \Theta_0, \Phi(C_0, \Theta_0))$$

and

$$D_C \Phi(C_0, \Theta_0) = -\left(D_\Phi f(C_0, \Theta_0, \Phi(C_0, \Theta_0))\right)^{-1} D_C f(C_0, \Theta_0, \Phi(C_0, \Theta_0)).$$

We need to describe $D_X f$.

$$D_\Theta f(C, \Theta, \Phi) = D_\Theta D_\Phi h(C, \Theta, \Phi) = -2 \sum_{i=1,\dots,n} w_i D_\Phi^T \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) D_\Theta \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta).$$

$$\begin{aligned} & D_C f(C, \Theta, \Phi) \\ = \; & -2 \sum_{i=1,\dots,n} w_i D_\Phi^T \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \left(D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta) - D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \right) \\ & -2 \sum_{i=1,\dots,n} w_i \left(\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \right) D_C D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \end{aligned}$$

$$\begin{aligned} & D_\Phi f(C, \Theta, \Phi) \\ = \; & D_\Phi D_\Phi h(C, \Theta, \Phi) = 2 \sum_{i=1,\dots,n} w_i D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) D_\Phi^T \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \\ & -2 \sum_{i=1,\dots,n} w_i \left(\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \right) D_\Phi^2 \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi). \end{aligned}$$

The *annoying* parts are the second order parts. Usually the first order derivatives are implemented in AD frameworks but not the second order one. Fortunately in the above formula, the second order derivatives are multiplied by $\mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta) - \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi)$ which is small when the calibrated model can match the base prices well enough. Based on that, we can use the following approximations:

$$\begin{aligned} & D_C f(C, \Theta, \Phi) \\ \simeq \; & -2 \sum_{i=1,\dots,n} w_i D_\Phi^T \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \left(D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Base}}(i, C, \Theta) - D_C \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) \right). \end{aligned}$$

$$D_\Phi f(C, \Theta, \Phi) \simeq 2 \sum_{i=1,\dots,n} w_i D_\Phi^T \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi) D_\Phi \mathrm{NPV}^{\mathrm{Vanilla}}_{\mathrm{Calibrated}}(i, C, \Phi)$$

This is very similar to the approximation done in computing Hessian as described in (Press et al., 1988, Section 14.4).

6 Least Square Examples

In this section we analyse an example similar to the second one of the previous example section.

The model is a Libor Market Model with displaced diffusion. We calibrate two parameters for each maturity: the volatility and the displacement parameters. The volatility parameter guide the general level of the smile while the displacement parameter command the skew of the smile. We calibrate the two parameters by a least square approach on the price of swaptions with several strikes. In the tests we use between 2 and 6 strikes.

The calibration is done for each yearly block on a multiplicative factor to given weights to obtain volatilities and on a shared displacement.

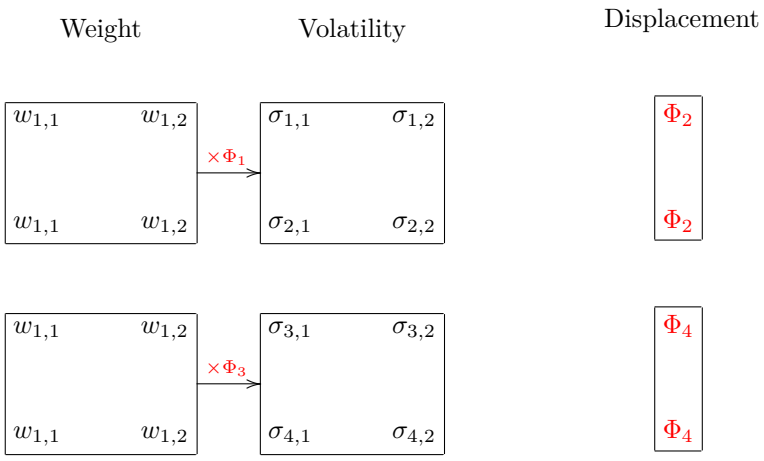


Figure 3: Representation of LMM calibration for a two years amortised swaption. Each yearly volatility block is multiplied by a common multiplicative factor and each yearly displacement block contains the same number.

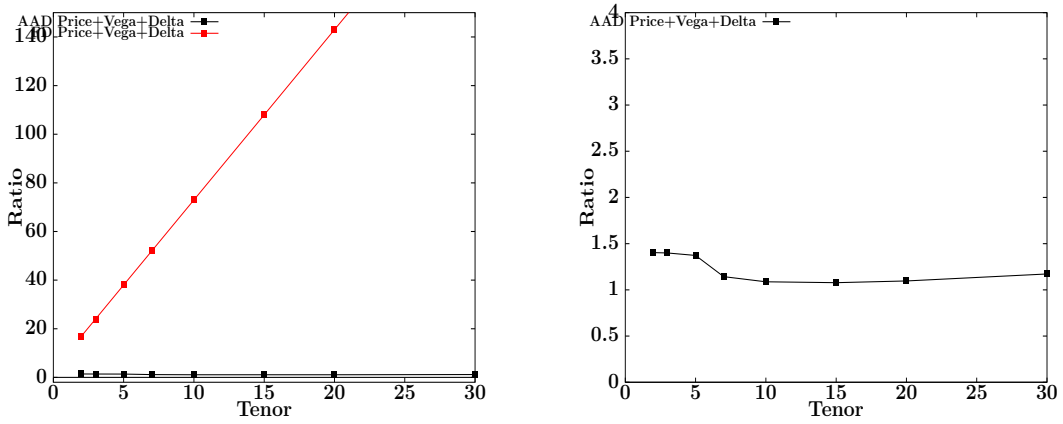
The more difficult the calibration is, the better the results of AD with implicit function will be on a relative basis. The algorithmic differentiation with implicit theorem method is using the already computed calibration in the sensitivity.

6.1 Reference

The reference example has a tenor of five years and there are five annual calibrations. The calibration on each tenor is done on three strikes (-100, 0, +100) bps from ATM. In the finite difference approach, the ratio is roughly 3 (SABR) + 4 (semi-annual payments with 2 curve) for each years. For a five years tenor, the finite difference ratio is around 36. The ratio obtained in practice in this example through AD with implicit function is 1.40.

6.2 Tenors

We run the same test with several tenors, between 2 years and 30 years. The calibration is similarly annual on three swaptions for each calibration date. In a finite difference the ratios would increase roughly linearly with the tenor. Figure 4 reports the results.



(a) Full picture

(b) AAD picture

Figure 4: Computation time ratios (price and sensitivities time to price time) for the finite difference and AAD methods. The *vega* represents the derivatives with respect to the SABR parameters; the *delta* represents the derivatives with respect to the curves. The AAD method uses the implicit function approach. Figures for annually amortised swaptions in a LMM calibrated yearly to three vanilla swaptions in SABR.

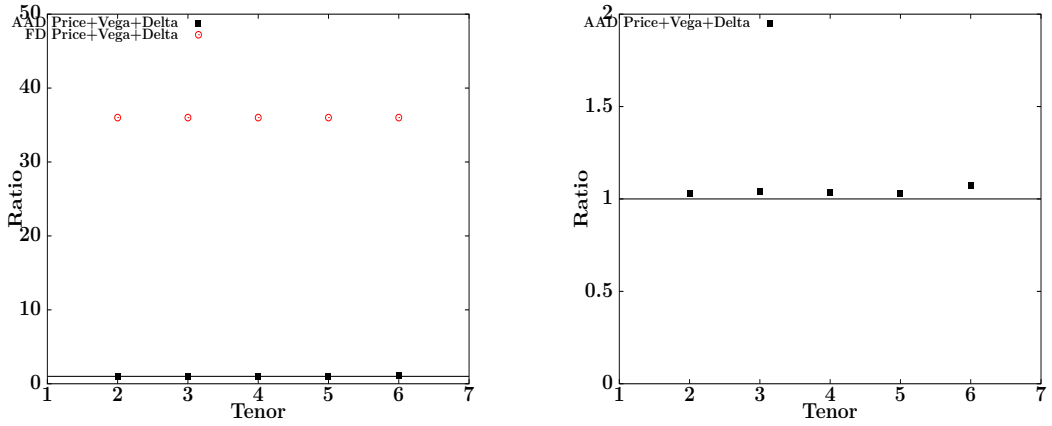
As in the previous examples, the metric to analyse the efficiency is the ratio between price and derivatives time and price time. The derivatives are composed of the SABR and curve sensitivities.

The linear increase of the ratios with the tenors is obvious for the finite difference method. The AD with implicit function method achieves a relatively constant ratio which is barely above 1 and well below 1.5. This is well below the theoretical upper bound of $\omega_A \in [3, 4]$. In the case of the 20-year swaption, the gain between the finite difference and optimised AD with implicit function is around 100. In practice this is reducing the computation time from 1 hour 40 minutes to 1 minute.

6.3 Strikes

In this part of the analysis, we go back to a 5 year amortised swaption with annual calibration. We run similar tests with calibrations sets for each periods with 2 to 6 strikes. Figure 5 reports the results.

As can be seen, the ratios are independent of the number of calibrating strikes for the finite difference and the AD versions. The ratio obtained in the AD with implicit function case is around 1.1, which is well below the theoretical upper bound of $\omega_A \in [3, 4]$.



(a) Full picture

(b) AAD picture

Figure 5: Computation time ratios (price and sensitivities time to price time) for the finite difference and AAD methods. The *vega* represents the derivatives with respect to the SABR parameters; the *delta* represents the derivatives with respect to the curves. The AAD method uses the implicit function approach. Figures for annually amortised swaptions in a LMM calibrated yearly to the given number of vanilla swaptions with different strikes in SABR.

7 Conclusion

With the algorithmic differentiation technique, one can, in general, obtain all the derivatives of the output with respect to the inputs at the computation cost of less than a fixed constant (between three and four in theory) times the cost of one price.

This note focuses on a efficient implementation in quantitative finance when a model calibration is part of the pricing process. The calibration process is generally done through a numerical procedure requiring the calibrating function to be computed numerous times. Using the implicit function theorem, we are able to compute the derivatives of the total process without requiring the derivative of the calibrating process itself.

By bypassing the calibration algorithm derivatives, a substantial performance gain is achieved. The gain is possible only using the domain specific knowledge of the pricing process; it can not be achieved using automatic differentiation. The gain is larger when the calibration process is an important part of the full pricing process. In all the examples presented the cost ratio obtained in practice is below 2.0. In the most complex case, the cost ratio obtain is practice is very close to 1.

References

Capriotti, L. (2011). Fast Greeks by algorithmic differentiation. *The Journal of Computational Finance*, 14(3):3–35. 1

Capriotti, L. and Giles, M. (2010). Fast correlation Greeks by adjoint algorithmic differentiation. *Risk*, 23(3):79–83. 1

Capriotti, L., Lee, J., and Peacock, M. (2011). Real time counterparty credit risk management in Monte Carlo. *Risk*, pages 86–90. 1

Christianson, B. (1998). Reverse accumulation and implicit functions. *Optimisation Methods and Software*, 9(4):307–322. 2

Denson, N. and Joshi, M. (2009a). Fast and accurate Greeks for the Libor Market Model. *Journal of Computational Finance.*, 14(4):115–140. 1

Denson, N. and Joshi, M. (2009b). Flaming logs. *Wilmott Journal*, 1:5–6. 1

Giles, M. and Glasserman, P. (2006). Smoking adjoints: fast Monte Carlo greeks. *Risk*, 19:88–92. 1

Giles, M. B. and Pierce, N. A. (2000). An introduction to adjoint approach to design. *Flow, Turbulence and Combustion*, 65:393–415. 3

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation.* SIAM, second edition. 1

Hagan, P., Kumar, D., Lesniewski, A., and Woodward, D. (2002). Managing smile risk. *Wilmott Magazine*, Sep:84–108. 6

Henrard, M. (2003). Explicit bond option and swaption formula in Heath-Jarrow-Morton one-factor model. *International Journal of Theoretical and Applied Finance*, 6(1):57–72. 6

Henrard, M. (2010a). Cash-settled swaptions: How wrong are we? Technical report, OpenGamma. Available at SSRN: http://ssrn.com/abstract=1703846. 6

Henrard, M. (2010b). The irony in the derivatives discounting - Part II: the crisis. *Wilmott Journal*, 2(6):301–316. 5

Henrard, M. (2010c). Swaptions in Libor Market Model with local volatility. *Wilmott Journal*, 2(3):135–154. 7

Henrard, M. (2012). Adjoint algorithmic differentiation: Calibration and implicit function theorem. *Journal of Computational Finance*, to appear. Available at http://docs.opengamma.com/display/DOC/Quantitative+Research. 2

Kaebe, C., Maruhn, J., and Sachs, E. (2009). Adjoint based Monte Carlo calibration of financial market models. *Finance and Stochastics*, 13(3):351–379. 1

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1988). *Numerical Recipes in C.* Cambridge University Press. 10

Schlenkirch, S. (2012). Efficient calibration of the Hull-White model. *Optimal Control Applicatios and Methods*, 33(3):352–362. 2, 6

OpenGamma Quantitative Research

1. Marc Henrard. Adjoint Algorithmic Differentiation: Calibration and implicit function theorem. November 2011.
2. Richard White. Local Volatility. January 2012.
3. Marc Henrard. My future is not convex. May 2012.
4. Richard White. Equity Variance Swap with Dividends. May 2012.
5. Marc Henrard. Deliverable Interest Rate Swap Futures: Pricing in Gaussian HJM Model. September 2012.
6. Marc Henrard. Multi-Curves: Variations on a Theme. October 2012.
7. Marc Henrard. Algorithmic Differentiation in Finance: Root Finding and Least Square Calibration. January 2013.

About OpenGamma

OpenGamma helps financial services firms unify their calculation of analytics across the traditional trading and risk management boundaries.

The company's flagship product, the OpenGamma Platform, is a transparent system for front-office and risk calculations for financial services firms. It combines data management, a declarative calculation engine, and analytics in one comprehensive solution. OpenGamma also develops a modern, independently-written quantitative finance library that can be used either as part of the Platform, or separately in its own right.

Released under the open source Apache License 2.0, the OpenGamma Platform covers a range of asset classes and provides a comprehensive set of analytic measures and numerical techniques.

Find out more about OpenGamma ____ **Download the OpenGamma Platform**

Europe
OpenGamma
185 Park Street
London SE1 9BL
United Kingdom

North America
OpenGamma
230 Park Avenue South
New York, NY 10003
United States of America