OpenGamma Quantitative Research

Adjoint Algorithmic Differentiation: Calibration and Implicit Function Theorem

Marc Henrard
marc@opengamma.com

Abstract

Adjoint Algorithmic Differentiation is an efficient way to obtain financial instrument price derivatives with respect to the data inputs. Often the differentiation does not cover the full pricing process when a model calibration is performed. Thanks to the implicit function theorem, the differentiation of the solver embedded in the calibration is not required to differentiate to full pricing process. An efficient approach to the full process differentiation is described.

Contents

1 Introduction

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

For the computation of financial instrument price 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 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 that 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 10Y swap in EUR, the number of derivatives is 42. 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[1] to the price time, i.e. the three derivatives can be computed by increasing the computation time by only 10%. Part of the saving comes from using *inside information* on the problem. In the Black formula, the derivative of the price with respect to the exercise boundary is zero. This type of simplification is not infrequent in finance; it appears in European swaption pricing in the Hull-White model and Bermudan swaption pricing.

In practice, this method is primarily of interest when considering complex processes. One frequent part of such a process for exotic option pricing in finance is a *complex 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*;
- the complex model parameters are calibrated to fit the the vanilla option prices from the base model. This step is usually done through a generic numerical equation solver; and

[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 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 fraction 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 intended to be complementary to the literature mentioned above.

An independent similar approach, using an Automatic Differentiation tool (ADOL-C), is proposed in Schlenkrich (2011). The model used in that paper is the Hull-White one factor model. The problem analysed is the calibration to European swaptions and pricing of Bermudan swaptions.

In general, the calibration process is not explicit, but done through a numerical equation solving. We have only one set of parameters of the calibrated model; the set that matches the prices from 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.

To show this, *implicit function* theorem is used. Under mild regularity conditions, the theorem asserts that for a root finding problem

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

if there is a solution (x_0, y_0), and if $D_y f(x_0, y_0)$ is invertible, then, near to x_0, there is a (implicit) function g such that $f(x, g(x)) = 0$ that 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. 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 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).

The idea of combining ADD and the implicit function theorem has been present in AD for some time (Christianson (1998)). In addition to calculating the derivative through equation solving, this 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.

We describe the improved efficiency of the calculation of the derivatives below, and provide several examples and show that the efficiency described in theory can be obtained in practice. In both examples, the price 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 bound).

2 Adjoint method and implicit function theorem

The method is presented using a simple example, allowing simplified notation. Given 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 equation that must 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 adjoint version of the functions g_i ($1 \leq i \leq 3$) is 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 as there is no requirement to have adjoint version of the solver, only the adjoint version of the function g_2. The adjoint 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 is

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

3 Calibration Technique description

The method is applied to the interest rate model calibration case. 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 procedure is

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

For perfect calibration, the function is simply

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

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 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; if the model has 40 parameters, 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 and the curves:

$$\Phi = \Phi(C, \Theta).$$

The function is obtained through the equation solving procedure; there is no explicit solution or even explicit code that produces that function 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 adjoint 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 of interest are

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

Through composition we have

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

and

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

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

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

and

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

In the perfect calibration case

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

and

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

4 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 adjoint versions have been implemented[2].

4.1 Cash swaptions in the Hull-White model

In the first example our *exotic* instrument is a cash-settled swaption and our vanilla basket is composed of a unique physical delivery swaption. The base model is a SABR model on the swap rate. The curve framework is multi-curve (one discounting curve and one forward curve). The model parameters Θ are the SABR parameters α, ρ and ν (β is set to 0.50). The complex 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).

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%.

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.

To partly compare with the results of Schlenkrich (2011), we also report figures for a Hull-White one factor model with piecewise constant volatility. The set-up is different but the underlying model

[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.

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.

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 is in line with Schlenkrich (2011) figures (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 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 swaption is the efficient approximation described in Henrard (2010b).

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.

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 only 20% to the computation time with respect to the calibration and price computation .

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

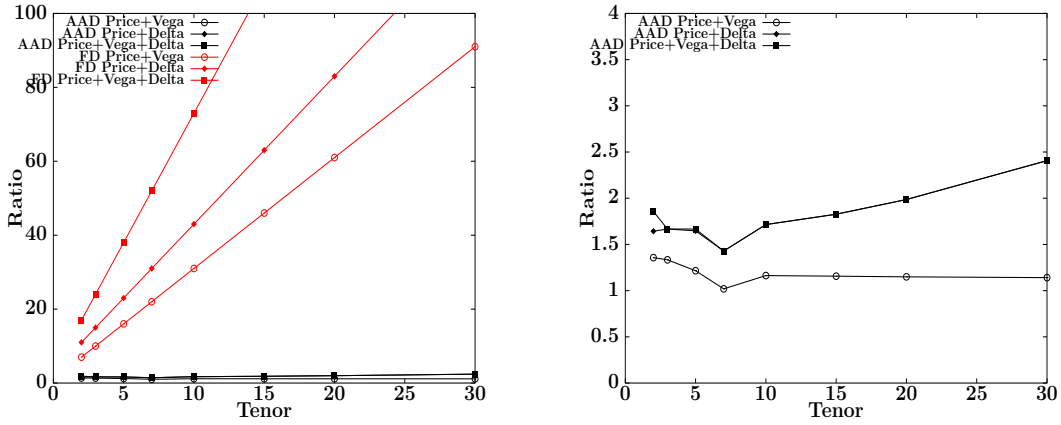
Similar results for amortised swaptions of different maturities and with different numbers of calibrating instruments are reported in Figure 1. The ratios between the price and sensitivities time and the price time are reported for the finite difference and Adjoint Algorithmic Differenti-

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.

ation using the implicit function method described in the previous section. The implicit function AAD method ratio is almost independent of the number of sensitivities. In all cases but the 30Y swaptions (where 212 sensitivities are calculated), the ratio is below two.



(a) Full picture

(b) AAD picture

Figure 1: 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 to vanilla swaptions in SABR.

5 Conclusion

With the algorithmic differentiation technique, one can, in general, obtain all the derivatives of the output (price) with respect to the inputs (in this case, curves and model parameters) at the

computation cost of less than a fixed constant (between three and four in theory) times the cost of one price.

This note analyses the equation solving (calibration) part of the algorithmic differentiation. In equation solving, the inner function defining the equation is usually computed numerous times. The implicit function theorem approach requires the adjoint method for the function, not for the equation solver. The derivatives of the global algorithm are computed at a cost which is a small multiple of the inner function cost, not the solver cost. If calibration is a significant part of the algorithm, this makes a large difference. In the complex pricing process of the two examples, the ratio for the price and derivatives computation time to the price only computation time is below two.

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. 2

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

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. 5

Henrard, M. (2010a). Cash-settled swaptions: How wrong are we? Technical report, OpenGamma. Available at SSRN: http://ssrn.com/abstract=1703846. 5

Henrard, M. (2010b). Swaptions in Libor Market Model with local volatility. *Wilmott Journal*, 2(3):135–154. 6

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

Schlenkrich, S. (2011). Efficient calibration of the Hull-White model. *Optimal Control Applications and Methods*. To appear (published online April 2011). 2, 5, 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.

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