VIBRATO AND AUTOMATIC DIFFERENTIATION FOR HIGH ORDER DERIVATIVES AND SENSITIVITIES OF FINANCIAL OPTIONS

Gilles Pagès, Olivier Pironneau, Guillaume Sall

VIBRATO AND AUTOMATIC DIFFERENTIATION FOR HIGH ORDER DERIVATIVES AND SENSITIVITIES OF FINANCIAL OPTIONS

GILLES PAGÈS *, OLIVIER PIRONNEAU †, AND GUILLAUME SALL ‡

Abstract. This paper deals with the computation of second or higher order greeks of financial securities. It combines two methods, Vibrato and automatic differentiation and compares with other methods. We show that this combined technique is faster than standard finite difference, more stable than automatic differentiation of second order derivatives and more general than Malliavin Calculus. We present a generic framework to compute any greeks and present several applications on different types of financial contracts: European and American options, multidimensional Basket Call and stochastic volatility models such as Heston's model. We give also an algorithm to compute derivatives for the Longstaff-Schwartz Monte Carlo method for American options. We also extend automatic differentiation for second order derivatives of options with non-twice differentiable payoff.

Key words. Financial securities, risk assessment, greeks, Monte-Carlo, automatic differentiation, vibrato.

AMS subject classifications. 37M25, 65N99

1. Introduction. Due to BASEL III regulations, banks are requested to evaluate the sensitivities of their portfolios every day (risk assessment). Some of these portfolios are huge and sensitivities are time consuming to compute accurately. Faced with the problem of building a software for this task and distrusting automatic differentiation for non-differentiable functions, we turned to an idea developed by Mike Giles called Vibrato.

Vibrato at core is a differentiation of a combination of likelihood ratio method and pathwise evaluation. In Giles [12], [13], it is shown that the computing time, stability and precision are enhanced compared with numerical differentiation of the full Monte Carlo path.

In many cases, double sensitivities, i.e. second derivatives with respect to parameters, are needed (e.g. gamma hedging).

Finite difference approximation of sensitivities is a very simple method but its precision is hard to control because it relies on the appropriate choice of the increment. Automatic differentiation of computer programs bypass the difficulty and its computing cost is similar to finite difference, if not cheaper. But in finance the payoff is never twice differentiable and so generalized derivatives have to be used requiring approximations of Dirac functions of which the precision is also doubtful.

The purpose of this paper is to investigate the feasibility of Vibrato for second and higher derivatives. We will first compare Vibrato applied twice with the analytic differentiation of Vibrato and show that it is equivalent; as the second is easier we propose the best compromise for second derivatives: Automatic Differentiation of Vibrato.

In [8], Capriotti has recently investigated the coupling of different mathematical methods – namely pathwise and likelihood ratio methods – with an Automatic differ-

*Laboratoire de Probabilités et Modèles Aléatoires, UMR 7599, UPMC, Case 188, 4 pl. de Jussieu, F-75252 Paris Cedex 5, France, gilles.pages@upmc.fr.

†Laboratoire Jacques Louis Lions, UMR 7598, Case 187, 4 pl. de Jussieu, F-75252 Paris Cedex 5, France, olivier.pironneau@upmc.fr.

‡Laboratoire de Probabilités et Modèles Aléatoires, UMR 7599, UPMC, Case 188, 4 pl. de Jussieu, F-75252 Paris Cedex 5, France, guillaume.sall@upmc.fr.

entiation technique for the computation of the second order greeks; here we follow the same idea but with Vibrato and also for the computation of higher order derivatives.

Automatic Differentiation (AD) of computer program as described by Greiwank in [19], [20], Naumann in [33] and Hascoet in [22] can be used in direct or reverse mode. In direct mode the computing cost is similar to finite difference but with no roundoff errors on the results: the method is exact because every line of the computer program which implements the financial option is differentiated exactly. The computing cost of a first derivative is similar to running the program twice.

Unfortunately, for many financial products the first or the second sensitivities do not exist at some point, such is the case for the standard Digital option at $x = K$; even the payofff of the a plain vanilla European option is not twice differentiatble at $x = K$, yet the Gamma is well defined due to the regularizing effect of the Brownian motion (or the heat kernel) which gives sense to the expectation of a Dirac as a pointwise value of a probability density; in short the end result is well defined but the intermediate steps of AD are not.

We tested `ADOL-C` [21] and tried to compute the Hessian matrix for a standard European Call option in the Black-Scholes model but the results were wrong. So we adapted our AD library based on operator overloading by including approximations of Dirac functions and obtained decent results; this is the second conclusion of the paper: AD for second sensitivities can be made to work; it is simpler than Vibrato+AD (VAD) but it is risky and slightly more computer intensive.

More details on AD can be found in Giles et al. [11], Pironneau [35], Capriotti [7], Homescu [26] and the references therein.

An important constraint when designing costly software for risk assessment is to be compatible with the history of the company which contracts the software; most of the time, this rules out the use of partial differential equations (see [1]) as most quant companies use Monte Carlo algorithms for pricing their portfolios.

For security derivatives computed by a Monte Carlo method, the computation of their sensitivities with respect to a parameter is most easily approximated by finite difference (also known as the *shock method*) thus requiring the reevaluation of the security with an incremented parameter. There are two problems with this method: it is imprecise when generalized to higher order derivatives and expensive for multidimensional problems with multiple parameters. The n^{th} derivative of a security with p parameters requires $(n + 1)p$ evaluations; furthermore the choice of the perturbation parameter is tricky.

From a semi-analytical standpoint the most natural way to compute a sensitivity is the pathwise method described in Glasserman [15] which amounts to compute the derivative of the payoff for each simulation path. Unfortunately, this technique happens to be inefficient for certain types of payoffs including some often used in quantitative finance like Digitals or Barrier options. For instance, as it is not possible to obtain the Delta of a Digital Call that way (the derivative of the expectation of a Digital payoff is not equal to the expectation of the derivative of the Digital payoff, which in fact does not exist as a function), the pathwise method cannot evaluate the Gamma of a Call option in a standard Black-Scholes model. The pathwise derivative estimation is also called *infinitesimal perturbation* and there is a extensive literature on this subject; see for example Ho et al. [24], in Suri et al. [39] and in L'Ecuyer [28]. A general framework for some applications to option pricing is given in Glasserman [14].

There are also two well known mathematical methods to obtain sensibilities, the

so-called log-likelihood ratio method and the Malliavin calculus. However, like the pathwise method, both have their own advantage and drawback. For the former, the method consists in differentiating the probability density of the underlying and clearly, it is not possible to compute greeks if the probability density of the underlying is not known. Yet, the method has a great advantage in that the probability densities are generally smooth functions of their parameters, even when payoff functions are not. This method has been developed primarily in Glynn [17], Reiman et al. [36], Rubinstein [37] and some financial applications in Broadie et al. [5] and Glasserman et al. [16].

As for the Malliavin calculus, the computation of the greeks consists in writing the expectation of the orignal payoff function times a specific factor i.e. the Malliavin weight which is a Skorohod integral, the adjoint operator of the Malliavin derivative. The main problem of this method is that the computation of the Malliavin weight can be complex and/or computationally costly for a high dimensional problem. Several articles deal with the computation of greeks via Malliavin calculus, Fournié et al. [10], Benhamou [2] and Gobet et al. [18] to cite a few. The precision of the Malliavin formulae also degenerates for short maturities, especially for the Δ-hedge.

Both the likelihood ratio and the Malliavin calculus are generally faster than the pathwise or finite difference method because, once the terms in front of the payoff function (the weight is computed analytically), the approximation of a greek in a one-dimensional case is almost equivalent to the cost of the evaluation of the pricing function. One systematic drawback is the implementation of these method in the financial industry is limited by the specific analysis required by each new payoff.

The paper is organized as follows; in section 2 we begin by recalling the Vibrato method for first order derivatives as in Giles [12] for the univariate and the multivariate case. We then generalize the method for the second and higher order derivatives with respect to one or several parameters and we describe the coupling to an analytical or Automatic differentiation method to obtain an additional order of differentiation.

In section 3, we recall briefly the different methods of Automatic differentiation. We describe the direct and the adjoint or reverse mode to differentiate a computer program. We also explain how it can be extended to some non differentiable functions.

Section 4 deals with several applications to different derivative securities. We show some results of second order derivatives (Gamma and Vanna) and third order derivatives in the case of a standard European Call option: the sensitivity of the Gamma with respect to changes in the underlying asset and a cross-derivatives with respect to the underlying asset, the volatility and the interest rate. Also, we compare different technique of Automatic differentiation and we give some details about our computer implementations.

In section 5 we study some path-dependent products; we apply the combined Vibrato plus Automatic differentiation method to the computation of the Gamma for an American Put option computed with the Longstaff Schwartz algorithm [31]. We also illustrate the method on a multidimensional Basket option (section 4) and on a European Call with Heston's model in section 6. In section 7, we study the computing time for the evaluation of the Hessian matrix of a standard European Call Option in the Black-Scholes model. Finally, in section 8 we compare VADs to Malliavin's and to the likelihood ratio method in the context of short maturities.

2. Vibrato. Vibrato was introduced by Giles in [12]; it is based on a reformulation of the payoff which is better suited to differentiation. The Monte Carlo path is

split into the last time step and its past. Let us explain the method on a plain vanilla multi-dimensional option.

First, let us recall the likelihood ratio method for derivatives.
Let the parameter set Θ be a subset of $\mathbb{R}^p$. Let $b : \Theta \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \Theta \times \mathbb{R}^d \to \mathbb{R}^{d\times q}$ be continuous functions, locally Lipstchitz in the space variable, with linear growth, both uniformly in $\theta \in \Theta$. We omit time as variable in both b and σ only for simplicity. And let $(W_t)_{t\geq 0}$ be a q-dimensional standard Brownian motion defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$.

LEMMA 2.1. (Log-likelihood ratio)

Let $p(\theta, \cdot)$ be the probability density of a random variable $X(\theta)$, which is function of θ; consider

$$\mathbb{E}[V(X(\theta))] = \int_{\mathbb{R}^d} V(y)p(\theta, y)dy. \tag{2.1}$$

If $\theta \mapsto p(\theta, \cdot)$ is differentiable at $\theta^0 \in \Theta$ for all y, then, under a standard domination or a uniform integrability assumption one can interchange differentiation and integration : for $i = 1, .., p$,

$$\frac{\partial}{\partial\theta_i}\Big[\mathbb{E}[V(X(\theta))]\Big]_{|\theta^0} = \int_{\mathbb{R}^d} V(y)\frac{\partial \log p}{\partial\theta_i}(\theta^0, y)p(\theta^0, y)dy = \mathbb{E}\left[V(X(\theta))\frac{\partial \log p}{\partial\theta_i}(\theta, X(\theta))\right]_{|\theta^0}. \tag{2.2}$$

2.1. Vibrato for a European Contract. Let $X = (X_t)_{t\in[0,T]}$ be a diffusion process, the strong solution of the following Stochastic Differential Equation (SDE)

$$dX_t = b\,(\theta, X_t)\,dt + \sigma(\theta, X_t)dW_t, \quad X_0 = x. \tag{2.3}$$

For simplicity and without loss of generality, we assume that $q = d$; so σ is a square matrix. Obviously, X_t depends on θ; for clarity, we write $X_t(\theta)$ when the context requires it.

Given an integer $n > 0$, the Euler scheme with constant step $h = \frac{T}{n}$, defined below in (2.3), approximates X_t at time $t_k^n = kh$, i.e. $\bar{X}_k^n \approx X_{kh}$, and it is recursively defined by

$$\bar{X}_k^n = \bar{X}_{k-1}^n + b(\theta, \bar{X}_{k-1}^n)h + \sigma(\theta, \bar{X}_{k-1}^n)\sqrt{h}Z_k, \quad \bar{X}_0^n = x, \quad k = 1, \ldots, n, \tag{2.4}$$

where $\{Z_k\}_{k=1,..,n}$ are independent random Gaussian $\mathcal{N}(0, I_d)$ vectors. The relation between W and Z is

$$W_{t_k^n} - W_{t_{k-1}^n} = \sqrt{h}Z_k. \tag{2.5}$$

Note that $\bar{X}_n^n = \mu_{n-1}(\theta) + \sigma_{n-1}(\theta)\sqrt{h}Z_n$ with

$$\mu_{n-1}(\theta) = \bar{X}_{n-1}^n(\theta) + b(\theta, \bar{X}_{n-1}^n(\theta))h \;\text{ and }\; \sigma_{n-1}(\theta) = \sigma(\theta, \bar{X}_{n-1}^n(\theta))\sqrt{h}. \tag{2.6}$$

Then, for any Borel function $V : \mathbb{R}^d \to \mathbb{R}$ such that $\mathbb{E}|V(\bar{X}_n^n(\theta))| < +\infty$,

$$\mathbb{E}\left[V(\bar{X}_n^n(\theta))\right] = \mathbb{E}\left[\mathbb{E}\left[V(\bar{X}_n^n(\theta)) \mid (W_{t_k^n})_{k=0,\ldots,n-1}\right]\right] = \mathbb{E}\left[\mathbb{E}\left[V(\bar{X}_n^n(\theta)) \mid \bar{X}_{n-1}^n\right]\right]. \tag{2.7}$$

This follows from the obvious fact that the Euler scheme defines a Markov chain $\bar{X}$ with respect to the filtration $\mathcal{F}_k = \sigma(W_{t_\ell^n},\, \ell = 0, \ldots, k)$.

Furthermore, by homogeneity of the chain,

$$\mathbb{E}\left[V(\bar{X}_n^n(\theta)) \mid \bar{X}_{n-1}^n\right] = \left\{\mathbb{E}_x\left[V(\bar{X}_1^n(x,\theta))\right]\right\}_{|x=\bar{X}_{n-1}^n} = \left\{\mathbb{E}[V(\mu+\sigma\sqrt{h}Z)]\right\}_{\substack{|\mu\,=\,\mu_{n-1}(\theta)\\ |\sigma\,=\,\sigma_{n-1}(\theta)}}. \tag{2.8}$$

Where $\bar{X}_1^n(x,\theta)$ denotes the value at time t_1^n of the Euler scheme with $k=1$, starting at x and where the last expectation is with respect to Z.

2.2. First Order Vibrato. We denote $\varphi(\mu,\sigma) = \mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\right]$. From (2.7) and (2.8), for any $i \in (1,\ldots,p)$

$$\frac{\partial}{\partial\theta_i}\mathbb{E}[V(\bar{X}_n^n(\theta))] = \mathbb{E}\left[\frac{\partial}{\partial\theta_i}\left\{\mathbb{E}[V(\mu+\sigma\sqrt{h}Z)]\right\}_{\substack{|\mu\,=\,\mu_{n-1}(\theta)\\ |\sigma\,=\,\sigma_{n-1}(\theta)}}\right] = \mathbb{E}\left[\frac{\partial\varphi}{\partial\theta_i}(\mu_{n-1}(\theta),\sigma_{n-1}(\theta))\right] \tag{2.9}$$

and

$$\frac{\partial\varphi}{\partial\theta_i}(\mu_{n-1},\sigma_{n-1}) = \frac{\partial\mu_{n-1}}{\partial\theta_i}\cdot\frac{\partial\varphi}{\partial\mu}(\mu_{n-1},\sigma_{n-1}) + \frac{\partial\sigma_{n-1}}{\partial\theta_i} : \frac{\partial\varphi}{\partial\sigma}(\mu_{n-1},\sigma_{n-1}) \tag{2.10}$$

where $\cdot$ denotes the scalar product and : denotes the trace of the product of the matrices.

LEMMA 2.2.

The θ_i-tangent process to X, $Y_t = \dfrac{\partial X_t}{\partial\theta_i}$, is defined as the solution of the following SDE (see Kunita[27] for a proof)

$$dY_t = \left[b'_{\theta_i}(\theta,X_t) + b'_x(\theta,X_t)Y_t\right]dt + \left[\sigma'_{\theta_i}(\theta,X_t) + \sigma'_x(\theta,X_t)Y_t\right]dW_t, \quad Y_0 = \frac{\partial X_0}{\partial\theta_i} \tag{2.11}$$

where the primes denote standard derivatives. As for $\bar{X}_k^n$ in (2.3), we may discretize (2.11) by

$$\bar{Y}_{k+1}^n = \bar{Y}_k^n + \left[b'_{\theta_i}(\theta,\bar{X}_k^n) + b'_x(\theta,\bar{X}_k^n)\bar{Y}_k^n\right]h + \left[\sigma'_{\theta_i}(\theta,\bar{X}_k^n) + \sigma'_x(\theta,\bar{X}_k^n)\bar{Y}_k^n\right]\sqrt{h}Z_{k+1} \tag{2.12}$$

Then from (2.6),

$$\begin{aligned}\frac{\partial\mu_{n-1}}{\partial\theta_i} &= \bar{Y}_{n-1}^n(\theta) + h\left[b'_{\theta_i}(\theta,\bar{X}_{n-1}^n(\theta)) + b'_x(\theta,\bar{X}_{n-1}^n(\theta))\bar{Y}_{n-1}^n(\theta)\right]\\ \frac{\partial\sigma_{n-1}}{\partial\theta_i} &= \sqrt{h}\left[\sigma'_{\theta_i}(\theta,\bar{X}_{n-1}^n(\theta)) + \sigma'_x(\theta,\bar{X}_{n-1}^n(\theta))\bar{Y}_{n-1}^n(\theta)\right].\end{aligned} \tag{2.13}$$

So far we have shown the following lemma.

LEMMA 2.3. *When $X_n^n(\theta)$ is given by (2.3), then $\dfrac{\partial}{\partial\theta_i}\mathbb{E}[V(\bar{X}_n^n(\theta))]$ is given by (2.9) with (2.10), (2.13) and (2.12).*

In (2.3) b and σ are constant in the time interval $(kh,(k+1)h)$, therefore the conditional probability of $\bar{X}_n^n$ given $\bar{X}_{n-1}^n$ given by

$$p(x) = \frac{1}{(\sqrt{2\pi})^d\sqrt{|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{2.14}$$

where μ and $\Sigma = h\sigma\sigma^T$ are evaluated at time $(n-1)h$ and given by (2.6). As in Dwyer et al. [9],

$$\frac{\partial}{\partial\mu}\log p(x) = \Sigma^{-1}(x-\mu), \quad \frac{\partial}{\partial\Sigma}\log p(x) = -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}(x-\mu)(x-\mu)^T\Sigma^{-1} \Rightarrow$$
$$\frac{\partial}{\partial\mu}\log p(x)|_{x=X_n^n} = \sigma^{-T}\frac{Z}{\sqrt{h}}, \quad \frac{\partial}{\partial\Sigma}\log p(x)|_{x=X_n^n} = \frac{1}{2h}\sigma^{-T}(ZZ^T - I)\sigma^{-1}.$$

Finally, applying Lemma 2.3 and Lemma 2.1 yields the following proposition

THEOREM 2.4. (Vibrato, multidimensional first order case)

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\mathbb{E}[V(\bar{X}_n^n(\theta))] &= \mathbb{E}\left[\frac{\partial}{\partial\theta_i}\left\{\mathbb{E}[V(\mu+\sigma\sqrt{h}Z)]\right\}_{\substack|\mu\,=\,\mu_{n-1}(\theta)\\|\sigma\,=\,\sigma_{n-1}(\theta)}\right] \\ &= \mathbb{E}\left[\frac{1}{\sqrt{h}}\frac{\partial\mu}{\partial\theta_i}\cdot\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}Z\right]\Big|_{\substack|\mu\,=\,\mu_{n-1}(\theta)\\|\sigma\,=\,\sigma_{n-1}(\theta)}\right. \\ &\left. + \frac{1}{2h}\,\frac{\partial\Sigma}{\partial\theta_i} : \mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}(ZZ^T-I)\sigma^{-1}\right]\Big|_{\substack|\mu\,=\,\mu_{n-1}(\theta)\\|\sigma\,=\,\sigma_{n-1}(\theta)}\right]\end{aligned} \tag{2.15}$$

2.3. Antithetic Vibrato. One can expect to improve the above formula – that is, reducing its variance – by the means of antithetic transform (see section 2.6 below for a short discussion) The following holds:

$$\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}Z\right] = \frac{1}{2}\mathbb{E}\left[\left(V(\mu+\sigma\sqrt{h}Z) - V(\mu-\sigma\sqrt{h}Z)\right)\sigma^{-T}Z\right]. \tag{2.16}$$

similarly, using $E[ZZ^T - I] = 0$,

$$\begin{aligned}&\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}(ZZ^T-I)\sigma^{-1}\right] \\ &= \frac{1}{2}\mathbb{E}\left[\left(V(\mu+\sigma\sqrt{h}Z) - 2V(\mu) + V(\mu-\sigma\sqrt{h}Z)\right)\sigma^{-T}(ZZ^T-I)\sigma^{-1}\right].\end{aligned} \tag{2.17}$$

COROLLARY 2.5. (One dimensional case, d=1)

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\mathbb{E}[V(\bar{X}_n^n(\theta))] &= \frac{1}{2}\mathbb{E}\left[\frac{\partial\mu}{\partial\theta_i}\mathbb{E}\left[\left(V(\mu+\sigma\sqrt{h}Z) - V(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right]\Big|_{\substack|\mu\,=\,\mu_{n-1}(\theta)\\|\sigma\,=\,\sigma_{n-1}(\theta)}\right. \\ &\left. + \frac{\partial\sigma}{\partial\theta_i}\mathbb{E}\left[\left(V(\mu+\sigma\sqrt{h}Z) - 2V(\mu) + V(\mu-\sigma\sqrt{h}Z)\right)\frac{Z^2-1}{\sigma\sqrt{h}}\right]\Big|_{\substack|\mu\,=\,\mu_{n-1}(\theta)\\|\sigma\,=\,\sigma_{n-1}(\theta)}\right]\end{aligned} \tag{2.18}$$

Conceptual Algorithm. In figure 1 we have illustrated the Vibrato decomposition at the path level. To implement the above one must perform the following steps:

1. Choose the number of time step n, the number of Monte-Carlo path M for the $n-1$ first time steps, the number M_Z of replication variable Z for the last time step.
2. For each Monte-Carlo path $j = 1..M$
   - Compute $\{X_k^n\}_{k=1:n-1}$, μ_{n-1}, σ_{n-1} by (2.3), (2.6).

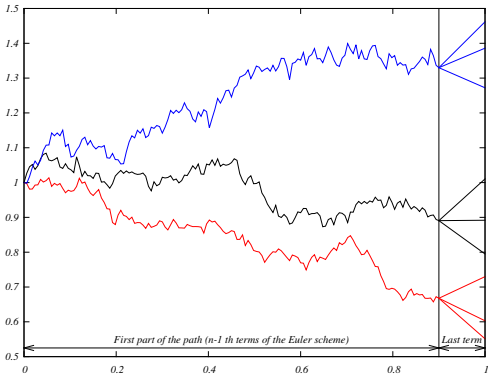

FIGURE 1. *Scheme of simulation path of the Vibrato decomposition.*

- Compute $V(\mu_{n-1})$
- Compute $\dfrac{\partial \mu_{n-1}}{\partial \theta_i}$ and $\dfrac{\partial \sigma_{n-1}}{\partial \theta_i}$ by (2.11), (2.13) and (2.12)
- Replicate M_Z times the last time step, i.e.
  For $m_Z \in (1, \dots, M_Z)$
  - Compute $V(\mu_{n-1} + \sigma_{n-1}\sqrt{h}Z^{(m_Z)})$ and $V(\mu_{n-1} - \sigma_{n-1}\sqrt{h}Z^{(m_Z)})$

3. In (2.18) compute the inner expected value by averaging over all M_Z results, then multiply by $\frac{\partial \mu}{\partial \theta_i}$ and $\frac{\partial \sigma}{\partial \theta_i}$ and then average over the M paths.

REMARK 1. *For simple cases such as of the sensibilities of European options, a small M_Z suffices; this is because there is another average with respect to M in the outer loop.*

REMARK 2. *For European options one may also use the Black-Scholes formula for the expected value in (2.15).*

2.4. Second Derivatives. Assume that X_0, b and σ depend on two parameters $(\theta_1, \theta_2) \in \Theta^2$. There are two ways to compute second order derivatives. Either by differentiating the Vibrato (2.15) while using Lemma 2.1 or by applying the Vibrato idea to the second derivative.

2.4.1. Second Derivatives by Differentiation of Vibrato. Let us differentiate (2.15) with respect to a second parameter θ_j:

$$\begin{aligned}\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathbb{E}[V(X_T)] = \mathbb{E}\Bigg[\frac{1}{\sqrt{h}}\Big(&\frac{\partial^2\mu}{\partial\theta_i\partial\theta_j}\cdot\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}Z\right]\\ &+\frac{\partial\mu}{\partial\theta_i}\cdot\frac{\partial}{\partial\theta_j}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}Z\right]\Big)\Big|_{\substack{\mu=\mu_{n-1}(\theta)\\ \sigma=\sigma_{n-1}(\theta)}}\\ &+\frac{1}{2h}\Big(\frac{\partial^2\Sigma}{\partial\theta_i\partial\theta_j}:\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}(ZZ^T-I)\sigma^{-1}\right]\\ &+\frac{\partial\Sigma}{\partial\theta_i}:\frac{\partial}{\partial\theta_j}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\sigma^{-T}(ZZ^T-I)\sigma^{-1}\right]\Big)\Big|_{\substack{\mu=\mu_{n-1}(\theta)\\ \sigma=\sigma_{n-1}(\theta)}}\Bigg]\end{aligned} \tag{2.19}$$

The derivatives can be expanded further; for instance in the one dimensional case and after a tedious algebra one obtains:

THEOREM 2.6. (Second Order by Differentiation of Vibrato)

$$\frac{\partial^2}{\partial\theta^2}\mathbb{E}[V(X_T)] = \mathbb{E}\left[\frac{\partial^2\mu}{\partial\theta^2}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z}{\sigma\sqrt{h}}\right] + \left(\frac{\partial\mu}{\partial\theta}\right)^2\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^2-1}{\sigma^2 h}\right]\right.$$
$$+\left(\frac{\partial\sigma}{\partial\theta}\right)^2\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^4-5Z^2+2}{\sigma^2 h}\right]$$
$$\left.+\frac{\partial^2\sigma}{\partial\theta^2}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^2-1}{\sigma\sqrt{h}}\right] + 2\frac{\partial\mu}{\partial\theta}\frac{\partial\sigma}{\partial\theta}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^3-3Z}{\sigma^2 h}\right]\right] \tag{2.20}$$

2.4.2. Second Derivatives by Second Order Vibrato. The same Vibrato strategy can be applied also directly to second derivatives.

As before the derivatives are transfered to the PDF p of X_T:

$$\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathbb{E}[V(X_T)] = \int_{\mathbb{R}^d}\frac{V(x)}{p(x)}\frac{\partial^2 p}{\partial\theta_i\partial\theta_j}p(x)\mathrm{d}x = \int_{\mathbb{R}^d}V(x)[\frac{\partial^2\ln p}{\partial\theta_i\partial\theta_j}+\frac{\partial\ln p}{\partial\theta_i}\frac{\partial\ln p}{\partial\theta_j}]p(x)\mathrm{d}x$$
$$= \mathbb{E}\left[V(x)\left(\frac{\partial^2\ln p}{\partial\theta_i\partial\theta_j}+\frac{\partial\ln p}{\partial\theta_i}\frac{\partial\ln p}{\partial\theta_j}\right)\right] \tag{2.21}$$

Then

$$\frac{\partial^2}{\partial\theta_1\partial\theta_2}\mathbb{E}[V(\bar{X}_T^n(\theta_1,\theta_2))] = \frac{\partial^2\varphi}{\partial\theta_1\partial\theta_2}(\mu,\sigma)$$
$$= \frac{\partial\mu}{\partial\theta_1}\frac{\partial\mu}{\partial\theta_2}\frac{\partial^2\varphi}{\partial\mu^2}(\mu,\sigma)+\frac{\partial\sigma}{\partial\theta_1}\frac{\partial\sigma}{\partial\theta_2}\frac{\partial^2\varphi}{\partial\sigma^2}(\mu,\sigma)+\frac{\partial^2\mu}{\partial\theta_1\partial\theta_2}\frac{\partial\varphi}{\partial\mu}(\mu,\sigma)$$
$$+\frac{\partial^2\sigma}{\partial\theta_1\partial\theta_2}\frac{\partial\varphi}{\partial\sigma}(\mu,\sigma)+\left(\frac{\partial\mu}{\partial\theta_1}\frac{\partial\sigma}{\partial\theta_2}+\frac{\partial\sigma}{\partial\theta_1}\frac{\partial\mu}{\partial\theta_2}\right)\frac{\partial^2\varphi}{\partial\mu\partial\sigma}(\mu,\sigma).$$

We need to calculate the two new terms $\frac{\partial^2}{\partial\theta_1\partial\theta_2}\mu_{n-1}(\theta_1,\theta_2)$ and $\frac{\partial^2}{\partial\theta_1\partial\theta_2}\sigma_{n-1}(\theta_1,\theta_2)$. It requires the computation of the first derivative with respect to θ_i of the tangent process Y_t, that we denote $Y_t^{(2)}(\theta_1,\theta_2)$.

Then (2.13) is differentiated and an elementary though tedious computations yields the following proposition:

PROPOSITION 2.7.

The θ_i-tangent process $Y_t^{(i)}$ defined above in Lemma 2.11 has a θ_j-tangent process $Y_t^{(ij)}$ defined by

$$dY_t^{(ij)} = \left[b''_{\theta_i\theta_j}(\theta_1,\theta_2,X_t)+b''_{\theta_i,x}(\theta_1,\theta_2,X_t)Y_t^{(j)}+b''_{\theta_j,x}(\theta_1,\theta_2,X_t)Y_t^{(i)}\right.$$
$$\left.+b''_{x^2}(\theta_1,\theta_2,X_t)Y_t^{(i)}Y_t^{(j)}+b'_x(\theta_1,\theta_2,X_t)Y_t^{(ij)}\right]dt$$
$$+\left[\sigma''_{\theta_i\theta_j}(\theta_1,\theta_2,X_t)+\sigma''_{\theta_i,x}(\theta_1,\theta_2,X_t)Y_t^{(j)}+\sigma''_{\theta_j,x}(\theta_1,\theta_2,X_t)Y_t^{(i)}\right.$$
$$\left.+\sigma''_{x^2}(\theta_1,\theta_2,X_t)Y_t^{(i)}Y_t^{(j)}+\sigma'_x(\theta_1,\theta_2,X_t)Y_t^{(ij)}\right]dW_t.$$

Finally in the univariate case $\theta=\theta_1=\theta_2$ this gives

PROPOSITION 2.8. (Second Order Vibrato)

$$\frac{\partial^2}{\partial\theta^2}\mathbb{E}[V(X_T)] =$$

$$\mathbb{E}\left[\frac{\partial^2\mu}{\partial\theta^2}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z}{\sigma\sqrt{h}}\right]+\left(\frac{\partial\mu}{\partial\theta}\right)^2\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^2-1}{\sigma^2h}\right]+\left(\frac{\partial\sigma}{\partial\theta}\right)^2\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\right.\right.$$
$$\left.\left.\frac{Z^4-5Z^2+2}{\sigma^2h}\right]+\frac{\partial^2\sigma}{\partial\theta^2}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^2-1}{\sigma\sqrt{h}}\right]+2\frac{\partial\mu}{\partial\theta}\frac{\partial\sigma}{\partial\theta}\mathbb{E}\left[V(\mu+\sigma\sqrt{h}Z)\frac{Z^3-3Z}{\sigma^2h}\right]\right] \quad (2.22)$$

REMARK 3. *It is equivalent to Proposition 2.6 hence to the direct differentiation of Vibrato.*

2.5. Higher Order Vibrato. The Vibrato-AD method can be generalized to higher order of differentiation of Vibrato with respect to the parameter θ with the help of the Faà di Bruno formula and its generalization to a composite function with a vector argument, as given in Mishkov [32].

2.6. Antithetic Transform, Regularity and Variance. In this section, we assume $d = q = 1$ for simplicity.

Starting from Vibrato $\varphi(\mu,\sigma) = \mathbb{E}[f(\mu+\sigma\sqrt{h}Z)]$ and assuming f Lipschitz continuous with Lipschitz coefficients $[f]_{\mathrm{Lip}}$, we have

$$\frac{\partial\varphi}{\partial\mu}(\mu,\sigma) = \mathbb{E}\left[f(\mu+\sigma\sqrt{h}Z)\frac{Z}{\sigma\sqrt{h}}\right] = \mathbb{E}\left[\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma Z\sqrt{h})\Big)\frac{Z}{2\sigma\sqrt{h}}\right]. \quad (2.23)$$

Therefore the variance satisfies

$$\mathbf{Var}\left[\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\Big)\frac{Z}{2\sigma\sqrt{h}}\right] \le \mathbb{E}\left[\left|\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\Big)\frac{Z}{2\sigma\sqrt{h}}\right|^2\right]$$
$$\le [f]^2_{\mathrm{Lip}}\mathbb{E}\left[\frac{(2\sigma\sqrt{h}Z)^2}{4\sigma^2h}Z^2\right] = [f]^2_{\mathrm{Lip}}\mathbb{E}[Z^4] = 3[f]^2_{\mathrm{Lip}}.$$

As $\mathbb{E}[Z] = 0$, we also have

$$\frac{\partial\varphi}{\partial\mu}(\mu,\sigma) = \mathbb{E}\left[\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu)\Big)\frac{Z}{\sigma\sqrt{h}}\right]. \quad (2.24)$$

Then,

$$\mathbf{Var}\left[\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu)\Big)\frac{Z}{\sigma\sqrt{h}}\right] \le \mathbb{E}\left[\left|\Big(f(\mu+\sigma\sqrt{h}Z)-f(\mu)\Big)\frac{Z}{\sigma\sqrt{h}}\right|^2\right]$$
$$\le \frac{1}{\sigma^2h}[f]^2_{\mathrm{Lip}}\mathbb{E}\left[(\sigma\sqrt{h}Z)^2Z^2\right] = [f]^2_{\mathrm{Lip}}\mathbb{E}[Z^4] = 3[f]^2_{\mathrm{Lip}}$$

REMARK 4. *The variances of formulae (2.23) and (2.24) are equivalent but the latter is less expensive to compute.* If f is differentiable and f' has polynomial growth, we also have

$$\frac{\partial\varphi}{\partial\mu}(\mu,\sigma) = \mathbb{E}[f'(\mu+\sigma\sqrt{h}Z)]. \quad (2.25)$$

Thus,

$$\mathbf{Var}\left[f'(\mu+\sigma\sqrt{h}Z)\right] \le \mathbb{E}\left[\Big(f'(\mu+\sigma\sqrt{h}Z)\Big)^2\right] \le \|f'\|^2_\infty.$$

REMARK 5. *Let $f]_{Lip}$ denote the Lipschitz constant of f. If f' is bounded, we have $[f]_{Lip} = \|f'\|_\infty$ then the expression in (2.25) has a smaller variance than (2.23)*

and (2.24). Assume that f' is Lipschitz continuous with Lipschitz coefficients $[f']_{\text{Lip}}$. We can improve the efficiency of (2.25) because

$$\begin{aligned}\mathbf{Var}\left[f'(\mu+\sigma\sqrt{h}Z)\right] &= \mathbf{Var}\left[f'(\mu+\sigma\sqrt{h}Z)-f'(\mu)\right]\\ &\leq \mathbb{E}\left[\left|f'(\mu+\sigma\sqrt{h}Z)-f'(\mu)\right|^2\right] \leq [f']^2_{\text{Lip}}h\sigma^2\mathbb{E}[Z^2] \leq [f']_{\text{Lip}}h\sigma^2\end{aligned}$$

REMARK 6. *Assuming that $f(x)=\mathbf{1}_{\{x\leq K\}}$, clearly we cannot differentiate inside the expectation and the estimation of the variance seen previously can not be applied.*

2.6.1. Indicator Function. Let us assume that $f(x)=\mathbf{1}_{\{x\leq K\}}$. To simplify assume that $K\leq\mu$, we have

$$\left|f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right| = \left|\mathbf{1}_{\left\{Z\leq\frac{K-\mu}{\sigma\sqrt{h}}\right\}}-\mathbf{1}_{\left\{Z\geq\frac{\mu-K}{\sigma\sqrt{h}}\right\}}\right| = \mathbf{1}_{\left\{Z\notin\left[\frac{K-\mu}{\sigma\sqrt{h}},\frac{\mu-K}{\sigma\sqrt{h}}\right]\right\}},$$

hence

$$\left|\left(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right| = \frac{1}{\sigma\sqrt{h}}|Z|\mathbf{1}_{\left\{Z\notin\left[\frac{K-\mu}{\sigma\sqrt{h}},\frac{\mu-K}{\sigma\sqrt{h}}\right]\right\}}.$$

For the variance, we have

$$\begin{aligned}&\mathbf{Var}\left[\left(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right]\\ &\leq \mathbb{E}\left[\left|\left(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right|^2\right].\end{aligned}$$

By Cauchy-Schwarz we can write

$$\begin{aligned}&\mathbb{E}\left[\left|\left(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right|^2\right]\\ &= \frac{1}{2\sigma^2h}\mathbb{E}\left[Z^2\left|f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right|^2\right] = \frac{1}{2\sigma^2h}\mathbb{E}\left[Z^2\mathbf{1}_{\left\{Z\notin\left[\frac{K-\mu}{\sigma\sqrt{h}},\frac{\mu-K}{\sigma\sqrt{h}}\right]\right\}}\right]\\ &\leq \frac{1}{2\sigma^2h}\left(\mathbb{E}[Z^4]\right)^{\frac{1}{2}}\left(\mathbb{P}\left(Z\notin\left[\frac{K-\mu}{\sigma\sqrt{h}},\frac{\mu-K}{\sigma\sqrt{h}}\right]\right)\right)^{\frac{1}{2}} \leq \frac{\sqrt{3}}{2\sigma^2h}\left(2\mathbb{P}\left(Z\geq\frac{\mu-K}{\sigma\sqrt{h}}\right)\right)^{\frac{1}{2}}.\end{aligned}$$

Then

$$\frac{\sqrt{3}}{2\sigma^2h}\left(2\mathbb{P}\left(Z\geq\frac{\mu-K}{\sigma\sqrt{h}}\right)\right)^{\frac{1}{2}} = \frac{\sqrt{6}}{2\sigma^2h}\left(\int_{\frac{\mu-K}{\sigma\sqrt{h}}}^{+\infty}e^{-\frac{u^2}{2}}\frac{du}{\sqrt{2\pi}}\right)^{\frac{1}{2}}.$$

Now, $\forall\ a>0,\ \mathbb{P}(Z\geq a)\leq\dfrac{e^{-\frac{a^2}{2}}}{a\sqrt{2\pi}}$, so when $a\to+\infty$,

$$\begin{aligned}&\mathbf{Var}\left[\left(f(\mu+\sigma\sqrt{h}Z)-f(\mu-\sigma\sqrt{h}Z)\right)\frac{Z}{\sigma\sqrt{h}}\right] \leq \frac{1}{\sigma^2h}\sqrt{\frac{3}{2}}\frac{e^{-\frac{(\mu-K)^2}{4\sigma^2h}}}{(2\pi)^{\frac{1}{4}}\sqrt{\frac{\mu-K}{\sigma\sqrt{h}}}}\\ &\leq \frac{1}{(2\pi)^{\frac{1}{4}}\sigma^{\frac{3}{2}}h^{\frac{3}{4}}}\sqrt{\frac{3}{2}}\frac{e^{-\frac{(\mu-K)^2}{4\sigma^2h}}}{\sqrt{\mu-K}} \underset{\sigma\to0}{\longrightarrow}\begin{cases}0 & \text{if } \mu\neq K\\ +\infty & \text{otherwise.}\end{cases}\end{aligned}$$

The fact that such estimate can be obtained with non differentiable f demonstrates the power of the Vibrato technique.

3. Second Derivatives by Vibrato plus Automatic Differentiation (VAD). The differentiation that leads to formula (2.22) can be derived automatically by AD; then one has just to write a computer program that implements the formula of proposition 2.19 and apply automatic differentiation to the computer program. We recall here the basis of AD.

3.1. Automatic Differentiation. Consider a function $z = f(u)$ implemented in `C` or `C++` by

```
double f(double u){...}
```

To find an approximation of z'_u, one could call in `C`

```
double dxdu= (f(u + du)-f(u))/du
```

because

$$z'_u = f'(u) = \frac{f(u+\mathrm{d}u) - f(u)}{\mathrm{d}u} + O(|\mathrm{d}u|).$$

A good precision ought to be reached by choosing $\mathrm{d}u$ small. However arithmetic truncation limits the accuracy and shows that it is not easy to choose $\mathrm{d}u$ appropriately because beyond a certain threshold, the accuracy of the finite difference formula degenerates due to an almost zero over almost zero ratio. As described in Squire et al.

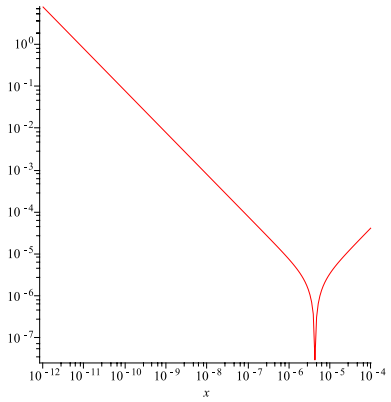

FIGURE 2. *Precision (log-log plot of $|dzdu - cos(1.)|$ computed with the forward finite difference formula to evaluate $sin'(u)$ at $u = 1$.*

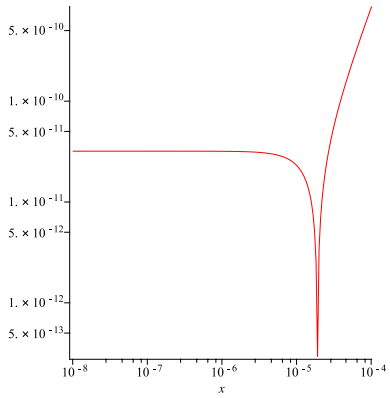

FIGURE 3. *Same as Fig. 2 but with the finite difference which uses complex increments; both test have been done with Maple-14*

[38], one simple remedy is to use complex imaginary increments because

$$Re\frac{f(u+\mathbf{i}\mathrm{d}u) - f(u)}{\mathbf{i}\mathrm{d}u} = Re\frac{f(u+\mathbf{i}\mathrm{d}u)}{\mathbf{i}\mathrm{d}u} = f'(u) - Ref'''(u+\mathbf{i}\theta\mathrm{d}u)\frac{\mathrm{d}u^2}{6}$$

leads to $f'(u) = Re[f(u+\mathbf{i}\mathrm{d}u)/(\mathbf{i}\mathrm{d}u)]$ where the numerator is no longer the result of a difference of two terms. Indeed tests show that the error does not detoriate when $\mathrm{d}u \to 0$ (figure 3). Hence one can choose $\mathrm{d}u = 10^{-8}$ to render the last term with a $O(10^{-16})$ accuracy thus obtaining an essentially exact result.

The cost for using this formula is two evaluations of $f()$, and the programming requires to redefine all `double` as `std::complex` of the Standard Template Library in C++.

3.2. AD in Direct Mode. A conceptually better idea is based on the fact that each line of a computer program is differentiable except at switching points of branching statements like `if` and at zeros of the `sqrt` functions etc.

Denoting by `dx` the differential of a variable `x`, the differential of `a*b` is `da*b+a*db`, the differential of `sin(x)` is `cos(x)dx`, etc... By operator overloading, this algebra can be built into a C++ class, called `ddouble` here:

```
class ddouble {
    public: double val[2];
    ddouble(double a=0, double b=0){ val[1]=b; val[0]=a; }
    ddouble operator=(const ddouble& a)
      { val[1] = a.val[1]; val[0]=a.val[0]; return *this; }
    ddouble operator - (const ddouble& a, const ddouble& b)
      { return ddouble(a.val[0] - b.val[0],a.val[1] - b.val[1]); }
    ddouble operator * (const ddouble& a, const ddouble& b)
      { return ddouble(a.val[0] * b.val[0], a.val[1]*b.val[0]
                              + a.val[0] * b.val[1]); }
... };
```

So all `ddouble` variables have a 2-array of data: `val[0]` contains the value of the variable and `val[1]` the value of its differential. Notice that the constructor of `ddouble` assigns zero by default to `val[1]`.

To understand how it works, consider the C++ example of figure 4 which calls a function $f(u, u_d) = (u-u_d)^2$ for $u = 2$ and $u_d = 0.1$. Figure 5 shows the same program where `double` has been changed to `ddouble` and the initialization of `u` implies that its differential is equal to 1. The printing statement displays now the differential of f which is also its derivative with respect to u if all parameters have their differential initialized to 0 except u for which has $du = 1$. Writing the class double with all

```
double f(double u, double u_d)
    { double z = u-u_d;
      return z*(u-u_d); }
int main() {
     double u=2., u_d =0.1;
     cout << f(u,u_d)<< endl;
     return 0;
}
```

FIGURE 4. *A tiny C++ program to compute* $(u - u_d)^2$ *at* $u = 2, u_d = 0.1$.

```
ddouble f(ddouble u, ddouble u_d)
    { ddouble z = u-u_d;
      return z*(u-u_d); }
int main() {
    ddouble u=ddouble(2.,1.), u_d = 0.1;
    cout << f(u,u_d).val[1] << endl;
    return 0;
}
```

FIGURE 5. *The same program now computes* $\frac{d}{du}(u - u_d)^2$ *at* $u = 2, u_d = 0.1$.

functions and common arithmetic operators is a little tedious but not difficult. An example can be downloaded from `www.ann.jussieu.fr/pironneau`.

The method can be extended to higher order derivatives easily. For second derivatives, for instance, `a.val[4]` will store a, its differentials with respected to the first and second parameter, d_1a, d_2a and the second differential $d_{12}a$ where the two parameters can be the same. The second differential of `a*b` is $a*d_{12}b+d_1a*d_2b+d_2a*d_1b+b*d_{12}a$, and so on.

Notice that $\frac{df}{du_d}$ can also be computed by the same program provided the first line in the `main()` is replaced by `ddouble u=2., u_d=ddouble(0.1,1.);`. However if both derivatives $\frac{df}{du}, \frac{df}{du_d}$ are needed, then, either the program must be run twice or the class ddouble must be modified to handle partial derivatives. In either case

the cost of computing n partial derivatives will be approximately n times that of the original program; the reverse mode does not have this numerical complexity and must be used when, say, $n > 5$ if expression templates with traits are used in the direct mode and $n > 5$ otherwise [35].

3.3. AD in Reverse Mode. Consider finding F'_θ where $(u, \theta) \to F(u, \theta) \in \mathbb{R}$ and $u \in \mathbb{R}^d$ and $\theta \in \mathbb{R}^n$. Assume that u is the solution of a well posed linear system $Au = B\theta + c$.

The direct differentiation mode applied to the C++ program which implements F will solve the linear system n times at the cost of d^2n operations at least.

The mathematical solution by calculus of variations starts with

$$F'_\theta \mathrm{d}\theta = (\partial_\theta F)\mathrm{d}\theta + (\partial_u F)\mathrm{d}u \text{ with } A\mathrm{d}u = B\mathrm{d}\theta,$$

then introduces $p \in \mathbb{R}^d$ solution of $A^T p = (\partial_u F)^T$ and writes

$$(\partial_u F)\mathrm{d}u = (A^T p)^T \mathrm{d}u = p^T B\mathrm{d}\theta \;\Rightarrow\; F'_\theta \mathrm{d}\theta = (\partial_\theta F + p^T B)\mathrm{d}\theta.$$

The linear system for p is solved only once, i.e. performing $O(d^2)$ operations at least. Thus, as the linear system is usually the costliest operation, this second method is the most advantageous when n is large.

A C program only made of assignments can be seen as a triangular linear system for the variables. Loops can be unrolled and seen as assignments and tests, etc. Then, by the above method, the ith line of the program is multiplied by p_i and p is computed from the last line up; but the biggest difficulty is the book-keeping of the values of the variables, at the time p is computed.

For instance, for the derivative of `f=u+ud` with respect to `ud` with `u` given by {`u=2*ud+4; u=3*u+ud;`},`u` in the second line is not the same as `u` in the third line and the program should be rewritten as `u1=2*ud+4; u=3*u1+ud;`. Then the system for p is `p2=1; p1=3*p2;` and the derivative is `2*p1+p2+1=8`.

In this study we have used the library `adept 1.0` by R.J. Hogan described in Hogan [25]. The nice part of this library is that the programming for the reverse mode is quite similar to the direct mode presented above; all differentiable variables have to be declared as `ddouble` and the variable with respect to which things are differentiated is indicated at initialization, as above.

3.4. Non-Differentiable Functions. In finance, non-differentiability is everywhere. For instance, the second derivative in K of $(x-K)^+$ does not exist at $x = K$ as a function, yet the second derivative of $\int_0^\infty f(x)(x-K)^+dx$ is $f(K)$. Distribution theory extends the notion of derivative: the Heavyside function $H(x) = \mathbf{1}_{\{x \geq 0\}}$ has the Dirac mass at zero $\delta(x)$ for derivative.

Automatic differentiation can be extended to handle this difficulty to some degree by approximating the Dirac mass at 0 by the functions $\delta^a(x)$ defined by

$$\delta^a(x) = \frac{1}{\sqrt{a\pi}} e^{-\frac{x^2}{a}}.$$

Now, suppose f is discontinuous at $x = z$ and smooth elsewhere; then

$$f(x) = f^+(x)H(x-z) + f^-(x)(1 - H(x-z))$$

hence

$$f'_z(x) = (f^+)'_z(x)H(x-z) + (f^-)'_z(x)(1 - H(x-z)) - (f^+(z) - f^-(z))\delta(x-z)$$

Unless this last term is added, the computation of the second order sensitivities will not be right.

If in the AD library the ramp function x^+ is defined as $xH(x)$ with its derivative to be $H(x)$, if H is defined with its derivative equal to δ^a and if in the program which computes the financial asset it is written that $(x-K)^+ = \text{ramp}(x-K)$, then the second derivative in K computed by the AD library will be $\delta^a(x-K)$. Moreover, it will also compute

$$\int_0^\infty f(x)(x-K)^+ dx \approx \frac{1}{N}\sum_{i=1}^{N} f(\xi_i)\delta^a(\xi_i - K)$$

where ξ_i are the N quadrature points of the integral or the Monte-Carlo points used by the programmer to approximate the integral.

However, this trick does not solve all problems and one must be cautious; for instance writing that $(x-K)^+ = (x-K)H(x-K)$ will not yield the right result. Moreover, the precision is rather sensitive to the value of a.

REMARK 7. *Notice that finite difference (FD) is not plagued by this problem, which means that FD with complex increment is quite a decent method for first order sensitivities. For second order sensitivities the "very small over very small" problem is still persistent.*

4. VAD and the Black-Scholes Model. In this section, we implement and test VAD and give a conceptual algorithm that describes the implementation of this method (done automatically). We focus on indicators which depend on the solution of an SDE, instead of the solution of the SDE itself. Let us take the example of a standard European Call option in the Black-Scholes model.

4.1. Conceptual algorithm for VAD.

1. Generate M simulation paths with time step $h = \frac{T}{n}$ of the underlying asset X and its tangent process $Y = \dfrac{\partial X}{\partial \theta}$ with respect to a parameter θ for $k = 0, \ldots, n-2$:

$$\begin{cases} \bar{X}^n_{k+1} = \bar{X}^n_k + rh\bar{X}^n_k + \bar{X}^n_k \sigma\sqrt{h} Z_{k+1}, \quad \bar{X}^n_0 = X_0, \bar{Y}^n_0 = \dfrac{\partial X_0}{\partial \theta} \\ \bar{Y}^n_{k+1} = \bar{Y}^n_k + rh\bar{Y}^n_k + \dfrac{\partial}{\partial \theta}(rh)\,\bar{X}^n_k + \left(\bar{Y}^n_k \sigma\sqrt{h} + \dfrac{\partial}{\partial \theta}\left(\sigma\sqrt{h}\right)\bar{X}^n_k\right) Z_{k+1}, \quad . \end{cases} \tag{4.1}$$

2. For each simulation path
   (a) Generate M_Z last time steps ($\bar{X}_T = \bar{X}^n_n$)

$$\bar{X}^n_n = \bar{X}^n_{n-1}(1 + rh + \sigma\sqrt{h} Z_n). \tag{4.2}$$

   (b) Compute the first derivative with respect to θ by Vibrato using the antithetic technique (formula (2.19) with $\sigma(X_t)$ equal $X_t\sigma$)

$$\begin{aligned} \frac{\partial V_T}{\partial \theta} = {} & \frac{\partial \mu_{n-1}}{\partial \theta}\frac{1}{2}(V_{T_+} - V_{T_-})\frac{Z_n}{\bar{X}^n_{n-1}\sigma\sqrt{h}} \\ & + \frac{\partial \sigma_{n-1}}{\partial \theta}\frac{1}{2}(V_{T_+} - 2V_{T_\bullet} + V_{T_-})\frac{Z_n^2 - 1}{\bar{X}^n_{n-1}\sigma\sqrt{h}}. \end{aligned} \tag{4.3}$$

With $V_{T_{\pm,\bullet}} = (\bar{X}_{T_{\pm,\bullet}} - K)^+$,

$$\begin{cases} \bar{X}_{T_\pm} = \bar{X}^n_{n-1} + rh\bar{X}^n_{n-1} \pm \sigma\bar{X}^n_{n-1}\sqrt{h}Z_n \\ \bar{X}_{T_\bullet} = \bar{X}^n_{n-1} + rh\bar{X}^n_{n-1}. \end{cases} \tag{4.4}$$

and

$$\begin{aligned} \frac{\partial \mu_{n-1}}{\partial\theta} &= \bar{Y}^n_{n-1}(1+rh) + \bar{X}^n_{n-1}\frac{\partial}{\partial\theta}(rh) \\ \frac{\partial \sigma_{n-1}}{\partial\theta} &= \bar{Y}^n_{n-1}\sigma\sqrt{h} + X^n_{n-1}\frac{\partial}{\partial\theta}(\sigma\sqrt{h}) \end{aligned} \tag{4.5}$$

If $\theta = T$ or $\theta = r$, we have to add $\dfrac{\partial}{\partial\theta}(e^{-rT})V_T$ to the result above.

(c) Apply an Automatic Differentiation method on the computer program that implements step 4.3 to compute the second derivative with respect to θ at some θ^*.

(d) Compute the mean per path i.e. over M_Z.

3. Compute the mean of the resulting vector (over the M simulation paths) and discount it.

4.2. Greeks. The Delta measures the rate of changes in the premium $\mathbb{E}[V(X_T)]$ with respect to changes in the spot price X_0.

The Gamma measures the rate of changes of the Delta with respect to changes in the spot price. Gamma can be important for a Delta-hedging of a portfolio.

The Vanna is the second derivative of the premium with respect to σ and X_0. The Vanna measures the rate of changes of the Delta with respect to changes in the volatility.

4.3. Numerical Test. For the generation of the random numbers, we chose the standard Mersenne-Twister generator available in the version 11 of the C++ STL. We take $M_Z = 1$ i.e. we simulate only one last time step per path; for all the test cases except for the European Call contract in the Black-Scholes model. However, for the European Call in a Black-Scholes model, we used a multiple time steps with the Euler scheme with or without a Brownian bridge.

The parameters considered in the following numerical experiments are $K = 100$, $\sigma = 20\%$ and $r = 5\%$, $T = 1$ year. The initial price of the risky asset price is varying from 1 to 200. The Monte Carlo parameters are set to $100,000$ simulation paths, 25 time steps.

4.3.1. Preliminary Numerical Test. Here, we focus on the numerical precision of VAD on the Gamma of a standard European Call contract with constant volatility and drift for which there is an analytical Black Scholes formula. Since Vibrato of Vibrato is similar to Vibrato+AD (VAD) it is pointless to compare the two.

Recall (Proposition 2.6 & 2.8) that it is equivalent to apply Vibrato to Vibrato or to apply automatic differentiation to Vibrato. However, the computation times are different and naturally double Vibrato is faster.

We compare the analytical solution to those obtained with VAD but now for each new set of parameters, we reuse the same sample of the random variables.

On figure 6, the Gammas are compared at $X_0 = 120$; true value of the Gamma is $\Gamma_0 = 0.0075003$. The convergence with respect to the number of paths is also displayed for two values of M_Z. The method shows a good precision and fast convergence when the number of paths for the final time step is increased.

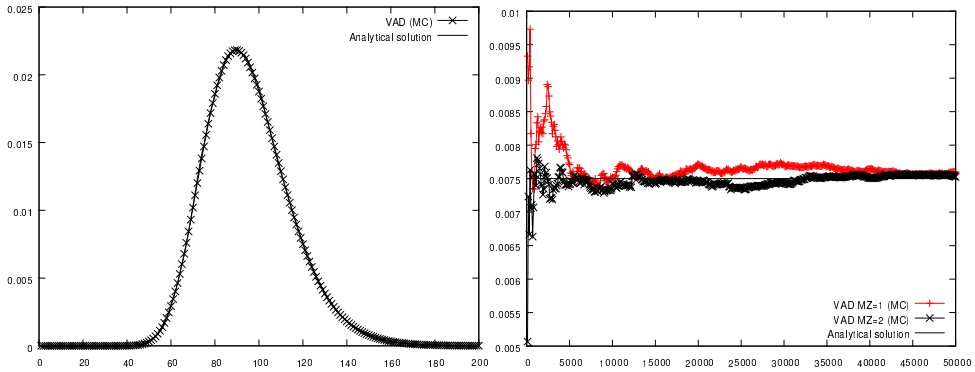

FIGURE 6. *On the left the Gamma versus Price is displayed when computed by VAD; the analytical exact Gamma is also displayed; both curves overlap. On the right, the convergence history at one point $X_0 = 120$ is displayed with respect to the number of Monte Carlo samples M_W. This is done for two values of M_Z (the number of the final time step), $M_Z = 1$ (low curve) and $M_Z = 2$ (upper curve).*

The L^2-error denoted by ε_{L^2} is defined by

$$\varepsilon_{L^2} = \frac{1}{P}\sum_{i=1}^{P}(\bar{\Gamma}^i - \Gamma_0)^2. \tag{4.6}$$

On figure 7, we compare the results with and without variance reduction on Vibrato at the final time step i.e. antithetic variables. The convergence history against the number of simulation paths is displayed. Results show that variance reduction is efficient on that test case. The standard error against the number of simulation paths is also displayed. It is clear that a reduction variance is needed. It requires almost ten times the number of simulation paths without the reduction variance technique to obtain the same precision. The Gamma is computed for the same set of parameters as given above.

On figures 8 we display the Vanna of an European Call option, computed with VAD. And again, the convergence with respect to the number of simulation paths is accelerated by more sampling of the final time step. Note that the Vanna requires double the number of time steps

4.3.2. Third Order Derivatives. For third order derivatives, we compute second derivatives by Vibrato of Vibrato 2.6 and differentiate by AD (VVAD). The sensitivity of the Gamma with respect to changes in X_0 is $\partial^3 V/\partial X_0^3$. The sensitivity of the Vanna with respect to changes in the interest rate is $\partial^3 V/\partial X_0 \partial\sigma\partial r$. The parameters of the European Call are the same but the Monte Carlo path number is $1,000,000$ and 50 time steps for the discretization. The results are displayed on figure 9. The convergence is slow; we could not eliminate the small difference between the analytical solution and the approximation by increasing the number of paths.

4.3.3. Ramp Function and High Order Derivatives. As mentioned in Section 3.4, it is possible to handle the non-differentiability of the function $(x - K)^+$

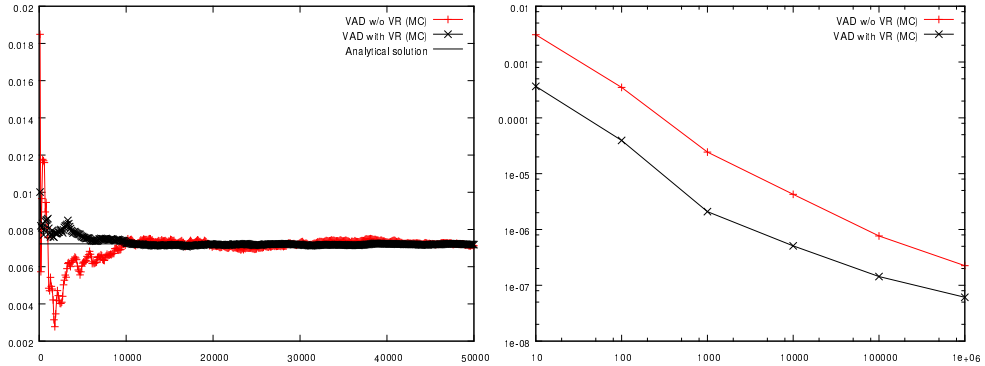

FIGURE 7. *On the left the Gamma versus the number of simulation paths is displayed when computed by VAD with and without the variance reduction method on Z, the straight line is the analytical solution at one point $X_0 = 120$; On the right, the standard error of the two methods versus the number of simulation paths with and without variance reduction.*

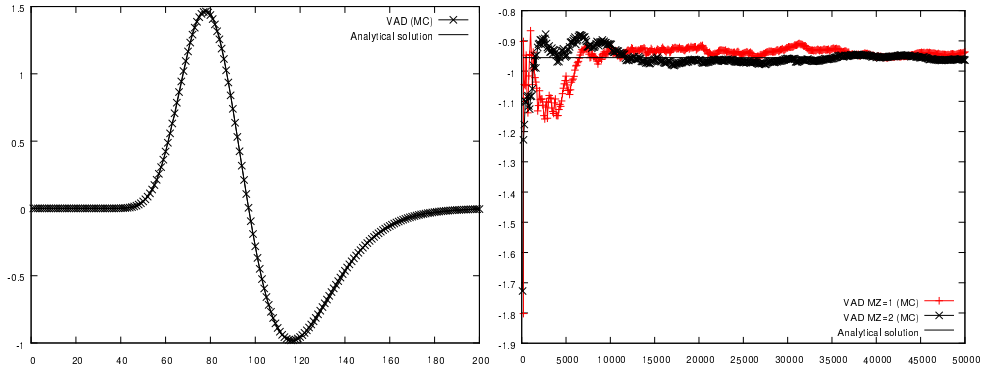

FIGURE 8. *On the left the Vanna versus Price is displayed when computed by VAD; the analytical exact Vanna is also displayed; both curves overlap. On the right, the convergence history at one point $X_0 = 120$ is displayed with respect to the number of Monte Carlo samples M_W. This is done for two values of M_Z, $M_Z = 1$ (lower curve) and $M_Z = 2$ (upper curve).*

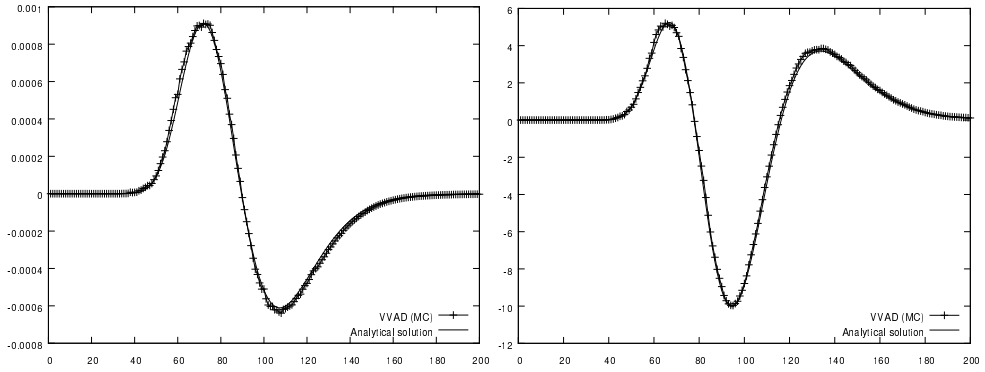

FIGURE 9. *On the left $\partial^3 V/\partial X_0^3$ versus Price is displayed when computed by VVAD; the analytical exact curve is also displayed; both curves practically overlap. On the right, the same for the Vanna with respect to changes in interest rate ($\partial^3 V/\partial X_0 \partial\sigma\partial r$).*

at $x = K$ by using distribution theory and program the ramp function explicitly with a second derivative equal to an approximate Dirac function at K. We illustrate this technique with a standard European Call option in the Black-Scholes model. We computed the Gamma and the sixth derivative with respect to X_0. For the first derivative, the parameter a does not play an important role but, as we evaluate higher derivatives, the choice of the parameter a becomes crucial for the quality of a good approximation and it requires more points to catch the Dirac approximation with small a. Currently the choice of a is experimental.

We took the same parameters as previously for the standard European Call option but the maturity for the Gamma now set at $T = 5$ years and $T = 0.2$ year for the sixth derivative with respect to X_0. The initial asset price varies from 1 to 200. The Monte Carlo parameters are also set to $100,000$ simulation paths and 25 time steps. The results are displayed on figure 10.

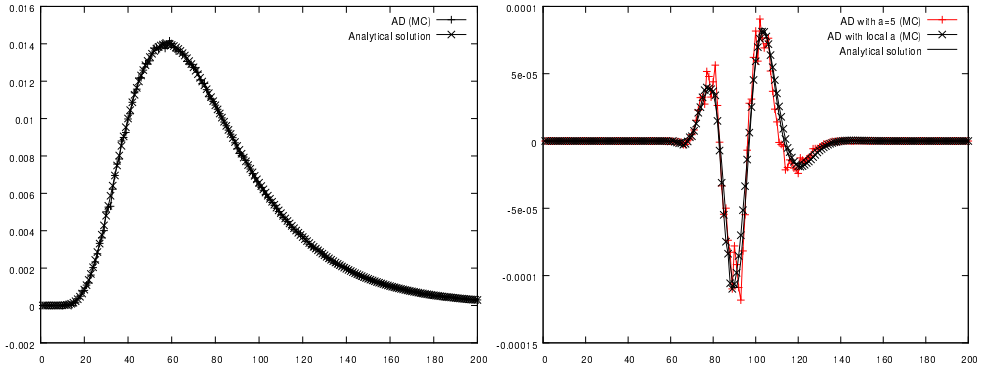

FIGURE 10. *On the left the Gamma versus Price is displayed when computed by AD with the ramp function (with $a = 1$); the analytical exact Gamma is also displayed; both curves overlap. On the right, the sixth derivative with respect to the parameter X_0 is displayed when computed via the same method; the analytical solution is also displayed. We computed the approximation with local parameter a and with $a = 5$.*

For the Gamma, the curves are overlapping but for the sixth derivative with respect to the parameter X_0, we cannot take a constant parameter a anymore. When we choose locally adapted parameter a, the curves are practically overlapping.

4.4. Baskets. A Basket option is a multidimensional derivative security whose payoff depends on the value of a weighted sum of several risky underlying assets.

As before, X_t is given by (2.3). But now $(W_t)_{t\in[0,T]}$ is a d-dimensional *correlated* Brownian motion with $\mathbb{E}[dW_t^i dW_t^j] = \rho_{i,j}dt$.

To simplify the presentation, we assume that r and σ_i are real constants and the payoff is given by

$$\mathbf{V}_T = e^{-rT}\mathbf{E}[(\sum_{i=1}^{d} \omega_i X_{iT} - K)^+] \tag{4.7}$$

where $(\omega_i)_{i=1,\dots,d}$ are positive weights with $\sum_{i=1}^{d} \omega_i = 1$. Here, we choose to compare three different methods. The reference values coming from an approximated moment-matching dynamics (Levy [30] and in Brigo et al. [4]), VAD and second order finite difference (FD).

4.4.1. Algorithm to compute the Gamma of a Basket option. We make use of the fact that r and σ are constant.

1. Generate M simulation paths using a one time step for the Euler scheme.

$$\bar{X^i}_{T\pm} = {X^i}_{T_\bullet} \exp\left(-\frac{1}{2}\sum_{j=1}^{d}|\Sigma^{ij}|^2 T \pm \sum_{j=1}^{d}\Sigma^{ij}\sqrt{T}Z_j\right), \quad i = 1, \ldots, d,$$

   with $X_{T_\bullet} = X_0 \exp(rT)$, where Z denotes an $\mathcal{N}(0; I_d)$ random vector.
2. For each simulation path, with $C = \Sigma\Sigma^{\mathbf{T}}$, compute (Vibrato)

$$\begin{aligned}\Delta = &\left(\frac{\partial\mu}{\partial X_{i0}}\right)^T \frac{1}{2\sqrt{h}}(V_{T_+} - V_{T_-})C^{-T}Z \\ &+ \frac{1}{4h}(V_{T_+} - 2V_{T_\bullet} + V_{T_-})\frac{\partial\Sigma}{\partial X_{i0}} : C^{-T}(ZZ^T - I_d)C^{-1}\end{aligned} \tag{4.8}$$

   with $V_{T_\cdot} = (\omega \cdot \bar{X}_{T_\cdot} - K)^+$
3. Compute the mean of the resulting vector and discount the result.
4. Apply Automatic Differentiation to what precedes.

4.4.2. Numerical Test. In this numerical test $d = 7$ and the underlying asset prices are:

$${X_0}^T = (1840, 1160, 3120, 4330.71, 9659.78, 14843.24, 10045.40). \tag{4.9}$$

The volatility vector is:

$$\sigma^T = (0.146, 0.1925, 0.1712, 0.1679, 0.1688, 0.2192, 0.2068). \tag{4.10}$$

The correlation matrix is

$$\begin{pmatrix} 1.0 & 0.9477 & 0.8494 & 0.8548 & 0.8719 & 0.6169 & 0.7886 \\ 0.9477 & 1.0 & 0.7558 & 0.7919 & 0.8209 & 0.6277 & 0.7354 \\ 0.8494 & 0.7558 & 1.0 & 0.9820 & 0.9505 & 0.6131 & 0.9303 \\ 0.8548 & 0.7919 & 0.9820 & 1.0 & 0.9378 & 0.6400 & 0.8902 \\ 0.8719 & 0.8209 & 0.9505 & 0.9378 & 1.0 & 0.6417 & 0.8424 \\ 0.6169 & 0.6277 & 0.6131 & 0.6400 & 0.6417 & 1.0 & 0.5927 \\ 0.7886 & 0.7354 & 0.9303 & 0.8902 & 0.8424 & 0.5927 & 1.0 \end{pmatrix}. \tag{4.11}$$

The number of Monte Carlo paths varies from 1 to 10^6 with only one time step for the time integration. Errors are calculated with reference to a solution computed by approximate moment matching.

On figures 11 and 12, the plot of convergence for the computation of the Gamma of a Basket made of the first 4 and 7 assets are displayed versus the number of simulation paths Vibrato plus AD (direct mode) and for Finite differences applied to a brute force Monte Carlo algorithm. The convergence speed of these methods is almost the same (with a slight advantage for the Finite difference).

Table 3 displays results for a Basket with the 7 assets, in addition the table 4 displays the CPU time for Vibrato plus AD (direct mode); the finite difference method is one third more expensive. Again, the method is very accurate.

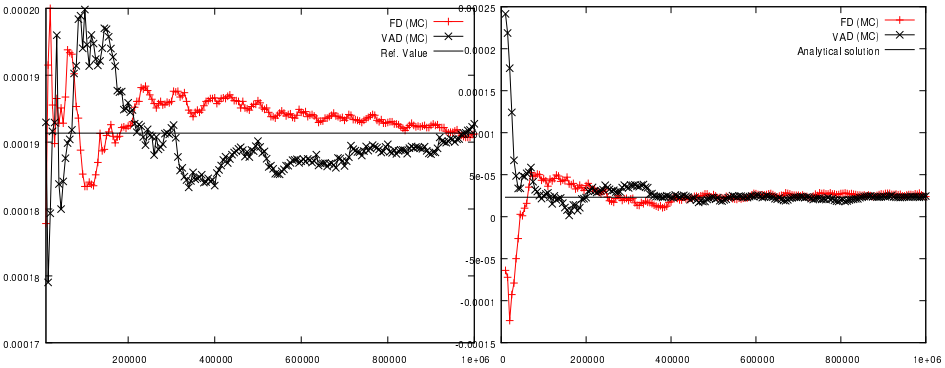

FIGURE 11. *d=4.* FIGURE 12. *d=7.*

Convergence of the computation of the Gamma of a Basket option when $d = 4$ and 7 via Vibrato plus Automatic Differentiation on Monte Carlo and via Finite differences, versus the number of simulation paths. The parameters are for $T = 0.1$.

5. American Option. Recall that an American option is like a European option which can be exercised at any time before maturity. The value V_t of an American option requires the best exercise strategy. Let φ be the payoff, then

$$V_t := \operatorname*{ess\,sup}_{\tau \in \mathcal{T}_t} \mathbf{E}[e^{-r(\tau - t)} \varphi(X_\tau) \mid X_t] \tag{5.1}$$

where $\mathcal{T}_t$ denotes the set of $[t, T]$-valued stopping times (with respect to the (augmented) filtration of the process $(X_s)_{s \in [0,T]}$).

Consider a time grid $0 < t_1 < \cdots < t_n = T$ with time step h, i.e. $t_k = kh$. To discretize the problem we begin by assuming that the option can be exercised only at $t_k,\ k = 0, .., n$; its value is defined recursively by

$$\begin{cases} \bar{V}_{t_n} = e^{-rT} \varphi(\bar{X}_T) \\ \bar{V}_{t_k} = \max\limits_{0 \le k \le n-1} \left(e^{-rt_k} \varphi(\bar{X}_{t_k}), \mathbb{E}[\bar{V}_{t_{k+1}} \mid \bar{X}_{t_k}] \right), \end{cases} \tag{5.2}$$

5.1. Longstaff-Schwartz Algorithm . Following Longstaff et al. [31] let the continuation value $C_{t_k} = \mathbb{E}[e^{-rh} \bar{V}_{t_{k+1}} \mid \bar{X}_{t_k}]$ as X is a Markov process. The holder of the contract exercises only if the payoff at t_k is higher than the continuation value C_{t_k}. The continuation value is approximated by a linear combination of a finite set of R real basis functions:

$$C_k \simeq \sum_{i=1}^{R} \alpha_{k,i} \psi_{k,i}(\bar{X}_{t_k}). \tag{5.3}$$

Typically, the $(\alpha_{k,i})_{i=1,\ldots,R}$ are computed by least squares,

$$\min_{\alpha} \left\{ \mathbb{E}\left[\left(\mathbb{E}[e^{-rh} \bar{V}_{t_{k+1}} \mid \bar{X}_{t_k}] - \sum_{i=1}^{R} \alpha_{k,i} \psi_{k,i}(\bar{X}_{t_k}) \right)^2 \right] \right\}. \tag{5.4}$$

This leads to a Gram linear system

$$\sum_{j=1}^{R} \alpha_{k,i} \mathbf{Gram}\left\{\psi_{k,i}(\bar{X}_{t_k}), \psi_{k,j}(\bar{X}_{t_k})\right\} = \mathbb{E}[\mathbb{E}[e^{-rh} V_{k+1} \mid X_{t_k}]\psi_{k,i}(\bar{X}_{t_k})], \;\; i = 1, \ldots, R. \tag{5.5}$$

REMARK 8. *Once the optimal stopping time is known, the differentiation with respect to θ of (5.2) can be done as for a European contract. The dependency of the τ^* on θ is neglected; arguably this dependency is second order but this point needs to be validated.* Hence, the following algorithm is proposed.

5.2. Algorithm to compute the Gamma of an American option.

1. Generate M simulation paths of an Euler scheme with n time steps of size $h = \frac{T}{n}$.
2. Compute the terminal value of each simulation path

$$V_T = (K - \bar{X}_T)^+ \tag{5.6}$$

3. Compute the Gamma of the terminal condition using (4.3) in section (4.1) for each simulation path.
4. Iterate from $n-1$ to 1 and perform the following at the k-th time step.
   (a) Solve the Gram linear system (5.5).
   (b) Calculate the continuation value of each path.

$$C_{k+1}(\bar{X}_{t_k}) = \sum_{i=1}^{R} \alpha_{k,i} \psi_i(\bar{X}_k^n). \tag{5.7}$$

   (c) Compute the Gamma by differentiating the Vibrato formula from the time step $k-1$ with respect to X_0

$$\tilde{\Gamma}_k = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial X_0} \left(\bar{Y}_{k-1}^n \, (1 + rh) \, \frac{1}{2} (\tilde{V}_{k_+}^i - \tilde{V}_{k_-}^i) \frac{Z_k^i}{X_0 \sigma \sqrt{h}} \right. \tag{5.8}$$

$$\left. + \bar{Y}_{k-1}^n \sigma \sqrt{h} \frac{1}{2} (\tilde{V}_{k_+}^i - 2\tilde{V}_{k_\bullet}^i + \tilde{V}_{k_-}^i) \frac{(Z_k^i)^2 - 1}{\bar{X}_0 \sigma \sqrt{h}} \right). \tag{5.9}$$

   (d) For $i = 1, \ldots, M$

$$\begin{cases} V_k^i = \tilde{V}_k^i, \;\; \Gamma_k^i = \tilde{\Gamma}_k^i & \text{if } \tilde{V}_k^i \geq C_{k+1}(\bar{X}_k^{n,i}), \\ V_k^i = e^{-rh} V_{k+1}^i, \;\; \Gamma_k^i = e^{-rh} \Gamma_{k+1}^i & \text{otherwise} \end{cases} \tag{5.10}$$

   with $\tilde{V}_{k+1} = (K - \bar{X}_{k+1}^n)^+$ and

$$\begin{cases} \bar{X}_{k_\pm} = \bar{X}_{k-1} + rh\bar{X}_{k-1} \pm \sigma \bar{X}_{k-1} \sqrt{h} Z_k \\ \bar{X}_{k_\bullet} = \bar{X}_{k-1} + rh\bar{X}_{k-1}. \end{cases} \tag{5.11}$$

5. Compute the mean of the vector V and Γ.

REMARK 9. *The differentiation with respect to X_0 is implemented by automatic differentiation of the computer program.*

5.2.1. Numerical Test. We consider the following value : $\sigma = 20\%$ or $\sigma = 40\%$, X_0 varying from 36 to 44, $T = 1$ or $T = 2$ year, $K = 40$ and $r = 6\%$. The Monte Carlo parameters are: $50,000$ simulation paths and 50 time steps for the time grid. The basis in the Longstaff-Scharwtz algorithm is $(x^n)_{n=0,1,2}$.

We compare with the solution of the Black-Scholes partial differential equation discretized by an implicit Euler scheme in time, finite element in space and semi-smooth Newton for the inequalities [1]. A second order finite Difference approximation is used to compute the Gamma. A large number of grid points are used to make it a reference solution. The parameters of the method are $10,000$ and 50 time steps per year. Convergence history for Longstaff Schwartz plus Vibrato plus AD is shown on figure 13 with respect to the number of Monte Carlo paths (Finite Difference on Monte Carlo is also displayed).

On figure 13, we display the history of convergence for the approximation of the Gamma of an American Put option versus the number of simulation paths for Vibrato plus Automatic differentiation and for Finite Difference applied to the American Monte Carlo, the straight line is the reference value computed by PDE+ semi-smooth Newton. The convergence is faster for VAD than with second order Finite Difference (the perturbation parameter is taken as 1% of the underlying asset price).

On table 5, the results are shown for different set of parameters taken from Longstaff et al. [31]. The method provides a good precision when variance reduction (**??**) is used, for the different parameters, except when the underlying asset price is low with a small volatility. As for the computation time, the method is faster than Finite Difference applied to the American Monte Carlo which requires three evaluations of the pricing function whereas VAD is equivalent to two evaluations (in direct mode).

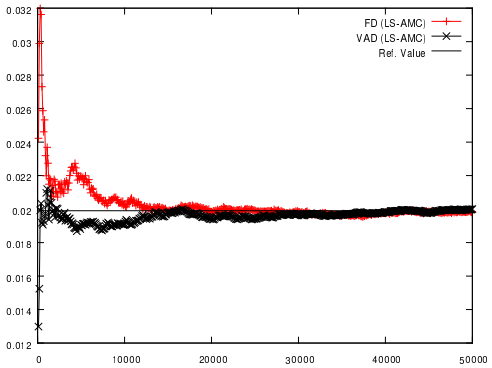

FIGURE 13. *Convergence of the Gamma of an American option via Vibrato plus Automatic Differentiation on the Longstaff-Schwartz algorithm and via Finite Difference, versus the number of simulation paths. The parameters are* $\sigma = 40\%$ *and* $X_0 = 40$.

6. Second Derivatives of a Stochastic Volatility Model. The Heston model [23] describes the evolution of an underlying asset $(X_t)_{t\in[0,T]}$ with a stochastic volatility $(\mathcal{V}_t)_{t\in[0,T]}$:

$$\begin{aligned} dX_t &= rX_t dt + \sqrt{\mathcal{V}_t} X_t dW_t^1, \\ d\mathcal{V}_t &= \kappa(\eta - \mathcal{V}_t)dt + \xi\sqrt{\mathcal{V}_t} dW_t^2, \quad t \in [0,T]; \quad \mathcal{V}_0, X_0 \text{ given.} \end{aligned} \tag{6.1}$$

Here ξ is the volatility of the volatility, η denotes the long-run mean of $\mathcal{V}_t$ and κ the mean reversion velocity. The standard Brownian process $(W_t^1)_{t\in[0,T]}$ and $(W_t^2)_{t\in[0,T]}$

are correlated: $\mathbb{E}[dW_t^1 W_t^2] = \rho dt$, $\rho \in (-1,1)$. If $2\kappa\eta > \xi^2$, it can be shown that $\mathcal{V}_t > 0$ for every $t \in [0,T]$. We consider the evaluation of a standard European Call with payoff

$$V_T = \mathbb{E}[(X_T - K)^+]. \tag{6.2}$$

6.1. Algorithm to Compute second derivatives in the Heston Model. To compute the Gamma by Vibrato method for the first derivative coupled to automatic differentiation for the second derivative one must do the following:

1. Generate M simulation paths for the underlying asset price $(\bar{X}, \bar{\mathcal{V}})$ and its tangent process $(\bar{Y}, \bar{\mathcal{U}}) = \frac{\partial(\bar{X},\bar{\mathcal{V}})}{\partial X_0}$ using an Euler scheme with n time steps of size $h = \frac{T}{n}$,

$$\begin{cases} \bar{X}_{k+1}^n = \bar{X}_k^n + rh\bar{X}_k^n + \sqrt{\bar{\mathcal{V}}_k^n}\bar{X}_k^n\sqrt{h}\tilde{Z}_{k+1}^1, & \bar{X}_0^n = X_0, \\ \bar{Y}_{k+1}^n = \bar{Y}_k^n + rh\bar{Y}_k^n + \sqrt{\bar{\mathcal{V}}_k^n}\bar{Y}_k^n\sqrt{h}\tilde{Z}_{k+1}^1, & \bar{Y}_0^n = 1, \\ \bar{\mathcal{V}}_{k+1}^n = \bar{\mathcal{V}}_k^n + \kappa(\eta - \bar{\mathcal{V}}_k^n)h + \xi\sqrt{\bar{\mathcal{V}}_k^n}\sqrt{h}\tilde{Z}_{k+1}^2, & \bar{\mathcal{V}}_0^n = \mathcal{V}_0 \end{cases} \tag{6.3}$$

   with

$$\begin{pmatrix} \tilde{Z}^1 \\ \tilde{Z}^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix} \begin{pmatrix} Z^1 \\ Z^2 \end{pmatrix} \tag{6.4}$$

   where $(Z_k^1, Z_k^2)_{1\le k\le n}$ denotes a sequence of $\mathcal{N}(0; I_2)$-distributed random variables.
2. For each simulation path
   (a) Compute the payoff

$$V_T = (\bar{X}_n^n - K)^+. \tag{6.5}$$

   (b) Compute the Delta using Vibrato at maturity with the $n-1$ time steps and the following formula

$$\bar{\Delta}^n = \bar{Y}_{n-1}^n\,(1+rh)\,\frac{1}{2}(V_{T_+} - V_{T_-})\frac{Z_n^1}{\bar{X}_{n-1}^n\sqrt{\bar{\mathcal{V}}_{n-1}^n}\sqrt{h}} \tag{6.6}$$

$$+\bar{Y}_{n-1}^n\sqrt{\bar{\mathcal{V}}_{n-1}^n}\sqrt{h}\frac{1}{2}(V_{T_+} - 2V_{T_\bullet} + V_{T_-})\frac{{Z_n^1}^2 - 1}{\bar{X}_{n-1}^n\sqrt{\bar{\mathcal{V}}_{n-1}^n}\sqrt{h}} \tag{6.7}$$

   with

$$\begin{cases} \bar{X}_{T_\pm} = \bar{X}_{n-1}^n + rh\bar{X}_{n-1}^n \pm \sqrt{\bar{\mathcal{V}}_{n-1}^n}\bar{X}_{n-1}^n\sqrt{h}\tilde{Z}_n^1, \\ \bar{X}_{T_\bullet} = \bar{X}_{n-1}^n + rh\bar{X}_{n-1}^n. \end{cases} \tag{6.8}$$

   (c) Apply an Automatic Differentiation method on step (2b) to compute the Gamma.
3. Compute the mean of the result and discount it.

6.1.1. Numerical Test. We have taken the following values: the underlying asset price $X_0 \in [60, 130]$, the strike is $K = 90$, the risk-free rate $r = 0.135\%$ and the maturity o is $T = 1$.

The initial volatility is $\mathcal{V}_0 = 2.8087\%$, the volatility of volatility is $\xi = 1\%$, the mean reversion is $\kappa = 2.931465$ and the long-run mean is $\nu = 0.101$. The correlation between the two standard Brownian motions is $\rho = 50\%$.

The number of Monte Carlo path is $500,000$ with 100 time steps each.

The results are displayed on figures 14, 15.

On figure 14 we compare the results obtained by Vibrato plus Automatic Differentiation (direct mode), with second order Finite Difference method applied to a standard Monte Carlo simulation. On figures 15 we display the Vanna of an European Call option in the Heston model, and again, the convergence with respect to the number of simulation paths. As for the Gamma, the method is quite precise. provides a good precision for the approximation of the Vomma and the Vanna. Both are computed at one point $(X_0, \mathcal{V}_0) = (85, 2.8087)$ with the same set of parameters as

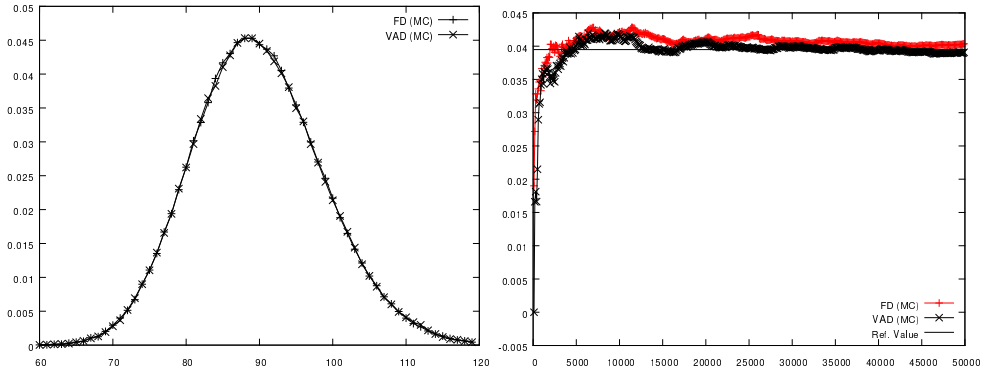

FIGURE 14. *On the left the Gamma versus Price is displayed when computed by VAD; the approximated Gamma via Finite Difference is also displayed; both curves overlap. On the right, the convergence history at one point* $(X_0, \mathcal{V}_0) = (85, 2.8087)$ *is displayed with respect to the number of Monte Carlo samples.*

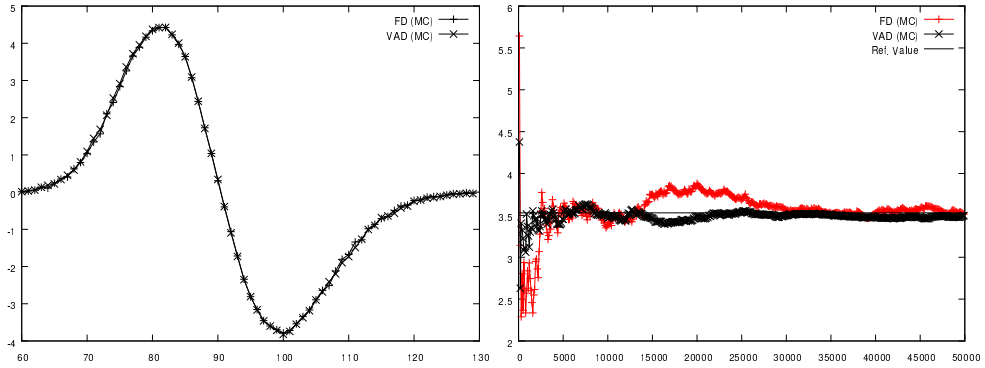

FIGURE 15. *On the left the Vanna versus Price is displayed when computed by VAD; the approximated Vanna via Finite Difference is also displayed; both curves overlap. On the right, the convergence history at one point* $(X_0, \mathcal{V}_0) = (85, 2.8087)$ *is displayed with respect to the number of Monte Carlo samples.*

given above. The computation by VAD is 30% faster for the Gamma compared with

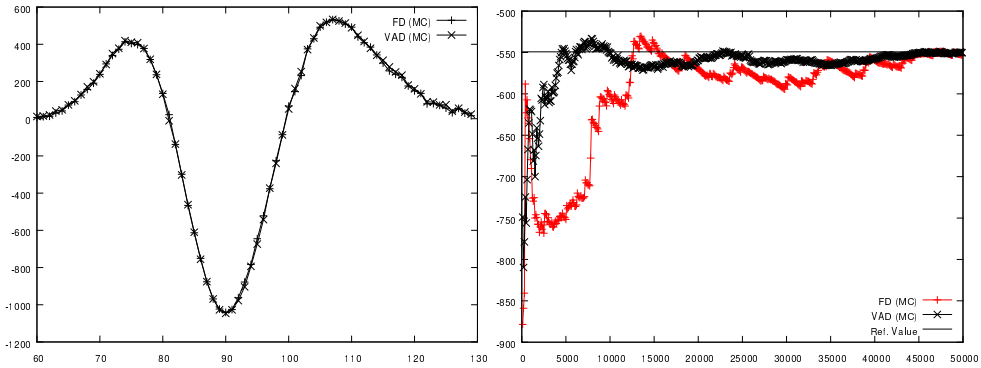

FIGURE 16. *On the left the Vomma versus Price is displayed when computed by VAD; the approximated Vomma via Finite Difference is also displayed; both curves overlap. On the right, the convergence history at one point* $(X_0, \mathcal{V}_0) = (85, 2.8087)$ *is displayed with respect to the number of Monte Carlo samples.*

the Vanna. In the case of the Vomma and the Gamma, VAD is 30% faster. For the Vanna Finite difference requires four times the evaluation of the pricing function so VAD is twice times faster.

7. Vibrato plus Reverse AD (VRAD). If several greeks are requested at once then it is better to use AD in reverse mode. To illustrate this point, we proceed to compute all second and cross derivatives i.e. the following Hessian matrix for a standard European Call option:

$$\begin{pmatrix} \dfrac{\partial^2 V}{\partial X_0^2} & \dfrac{\partial^2 V}{\partial v \partial X_0} & \dfrac{\partial^2 V}{\partial r \partial X_0} & \dfrac{\partial^2 V}{\partial T \partial X_0} \\ \dfrac{\partial^2 V}{\partial X_0 \partial \sigma} & \dfrac{\partial^2 V}{\partial \sigma^2} & \dfrac{\partial^2 V}{\partial v \partial r} & \dfrac{\partial^2 V}{\partial T \partial v} \\ \dfrac{\partial^2 V}{\partial X_0 \partial r} & \dfrac{\partial^2 V}{\partial v \partial r} & \dfrac{\partial^2 V}{\partial r^2} & \dfrac{\partial^2 V}{\partial T \partial r} \\ \dfrac{\partial^2 V}{\partial X_0 \partial T} & \dfrac{\partial^2 V}{\partial v \partial T} & \dfrac{\partial^2 V}{\partial r \partial T} & \dfrac{\partial^2 V}{\partial T^2} \end{pmatrix}. \tag{7.1}$$

It is easily seen that a Finite Difference procedure will require 36 (at least 33) evaluations of the original pricing function whereas we only call this function once if AD is used in reverse mode. Furthermore, we have to handle 4 different perturbation parameters.

The parameters are $X_0 = 90,\ K = 100,\ \sigma = 0.2,\ r = 0.05$ and $T = 1$ year. The parameters of Monte Carlo are set to $200,000$ simulation paths and 50 time steps. We used the library `adept 1.0` for the reverse mode. One great aspect here is that we only have one formula in the computer program to compute all the greeks, consequently one has just to specify which parameters are taken as variable for differentiation.

The results are shown in the table 1, clearly the reverse automatic differentiation combined with Vibrato is almost 4 times faster than the finite difference procedures.

8. Malliavin Calculus and Likelihood Ratio Method . Here, we want to point out that Malliavin calculus and LRM are excellent methods but they have their own numerical issues especially with short maturities which may make VAD more attractive for a general purpose software.

Mode	FD (MC)	VRAD (MC)
Time (sec)	2.01	0.47

TABLE 1

CPU time (in seconds) to compute the Hessian matrix of a standard European Call option (considering X_0, σ, r, T as variables) in the Black-Scholes model.

Let us start by recalling briefly the foundations of Malliavin calculus (further details are available in Nualart [34], Fournié et al.[10] and in Gobet et al. [18], for instance). We recall the Bismut-Elworthy-Li formula (see [3], for example):

PROPOSITION 8.1. (Bismut-Elworthy-Li formula) *Let X be a diffusion process given by (2.3) with $d = 1$, b and σ in $\mathcal{C}^1$. Let $f : \mathbb{R} \to \mathbb{R}$ be $\mathcal{C}^1$ with $\mathbb{E}[f(X_T)^2]$ and $\mathbb{E}[f'(X_T)^2]$ bounded. Let $(H_t)_{t\in[0,T]}$ an $\mathcal{F}$-progressively measurable process in $L^2([0,T]\times\Omega, dt\otimes d\mathbb{P})$ such that $\mathbb{E}\left[\int_0^T H_s^2 ds\right]$ is finite. Then*

$$\mathbb{E}\left[f(X_T)\int_0^T H_s dW_s\right] = \mathbb{E}\left[f'(X_T)Y_T\int_0^T \frac{\sigma(X_s)H_s}{Y_s}ds\right] \tag{8.1}$$

where $Y_t = \dfrac{dX_t}{dx}$ is the tangent process defined in (2.11). By choosing $H_t = Y_t/\sigma(X_t)$ the above yields

$$\frac{\partial}{\partial x}\mathbb{E}\left[f(X_T^x)\right] = \mathbb{E}\left[f(X_T^x)\underbrace{\frac{1}{T}\int_0^T \frac{Y_s}{\sigma(X_s^x)}dW_s}_{\text{Malliavin weight}}\right] \tag{8.2}$$

provided f has polynomial growth and $\mathbb{E}\left[\int_0^T\left(\dfrac{Y_t}{\sigma(X_t^x)}\right)^2\right]$ is finite.

Second Derivative.. In the context of the Black-Scholes model, the Malliavin weights, π_Γ, for the Gamma is (see [2]):

$$\pi_\Gamma = \frac{1}{X_0^2\sigma T}\left(\frac{W_T^2}{\sigma T} - \frac{1}{\sigma} - W_T\right). \tag{8.3}$$

Hence

$$\Gamma_{\text{Mal}} = e^{-rT}\mathbb{E}\left[(X_T-K)^+\frac{1}{X_0^2\sigma T}\left(\frac{W_T^2}{\sigma T} - \frac{1}{\sigma} - W_T\right)\right]. \tag{8.4}$$

The pure likelihood ratio method gives a similar formula (see Lemma 2.1)

$$\Gamma_{\text{LR}} = e^{-rT}\mathbb{E}\left[(X_T-K)^+\left(\frac{Z^2-1}{X_0^2\sigma^2 T} - \frac{Z}{X_0^2\sigma\sqrt{T}}\right)\right]. \tag{8.5}$$

LRPW is an improvement of LRM obtained by combining it with a pathwise method [15].

$$\Gamma_{\text{LRPW}} = \frac{\partial}{\partial X_0}\left(e^{-rT}\mathbb{E}\left[(X_T-K)^+\frac{Z}{X_0\sigma\sqrt{T}}\right]\right) = e^{-rT}\frac{K}{X_0^2\sigma\sqrt{T}}\mathbb{E}[Z\mathbf{1}_{\{X_T>K\}}]. \tag{8.6}$$

LRPW is much cheaper than VAD, Malliavin or LRM and it is also less singular at $T = 0$. However all these methods require new analytically derivations for each new problem.

8.1. Numerical Tests. We compared VAD with LRPW and Malliavin calculus. The results are shown on Table 2

T	VAD (MC)	FD (MC)	LRPW (MC)	Malliavin (MC)
1.00e+0	3.63e-5	1.76e-4	3.40e-4	9.19e-3
5.00e-1	8.55e-5	3.11e-4	7.79e-4	1.62e-2
1.00e-1	6.64e-4	1.50e-3	4.00e-3	6.54e-2
5.00e-2	1.49e-3	2.80e-3	7.51e-3	1.21e-1
1.00e-2	8.78e-3	1.84e-2	3.76e-2	5.44e-1
5.00e-3	1.86e-2	3.95e-2	7.55e-2	1.10e+0
1.00e-3	9.62e-2	1.77e-1	3.76e-1	5.74e+0
5.00e-4	1.85e-1	3.34e-1	7.56e-1	1.07e+1
1.00e-4	1.01e+0	1.63e+0	3.77e+0	5.26e+1
5.00e-5	1.98e+0	3.46e+0	7.54e+0	1.09e+2
1.00e-5	1.03e+1	1.78e+1	3.79e+1	5.40e+2

TABLE 2

Variance of the Gamma of a standard European Call with short maturities in the Black-Scholes model. Gamma is computed with VAD, FD, LRPW and Malliavin. The computation are done on the same samples.

The Gamma is computed with the same parameters as in the section 4.3. The maturity is varying from $T = 1$ to 10^{-5} year. The Monte Carlo parameters are also set to $100,000$ simulation paths and 25 time steps.

Notice the inefficiency of LRPW, Malliavin Calculus and to a lesser degree of VAD and Finite Difference when T is small.

Note on CPU. Tests have been done on an Intel(R) Core(TM) i5-3210M Processor @ 2,50 GHz. The processor has turbo speed of 3.1 GHz and two cores. We did not use parallelization in the code.

9. Conclusion. This article extends the work of Mike Giles and investigates the Vibrato method for higher order derivatives in quantitative finance.

For a general purpose software Vibrato of Vibrato is too complex but we showed that it is essentially similar to the analytical differentiation of Vibrato. Thus AD of Vibrato is both general, simple and essentially similar to Vibrato of Vibrato of second derivatives. We have also shown that Automatic differentiation can be enhanced to handle the singularities of the payoff functions of finance. While AD for second derivatives is certainly the easiest solution, it is not the safest and it requires an appropriate choice for the approximation of the Dirac mass.

Finally we compared with Malliavin calculus and LRPW.

The framework proposed is easy to implement, efficient, faster and more stable than its competitors and does not require analytical derivations if local volatilities or payoffs are changed.

Further developments are in progress around nested Monte Carlo and Multilevel-Multistep Richardson-Romberg extrapolation [29] (hence an extension to [6]).

Acknowledgment. This work has been done with the support of ANRT and Global Market Solution inc. with special encouragements from Youssef Allaoui and Laurent Marcoux.

REFERENCES

[1] Y. Achdou and O. Pironneau. *Computation methods for option pricing*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2005. xviii+297 pp., ISBN 0-89871-573-3.

[2] E. Benhamou. Optimal Malliavin weighting function for the computation of the greeks. *Mathematical Finance*, 13:37–53, 2003.

[3] J. M. Bismut, K. D. Elworthy, and X. M. Li. Bismut type formulae for differential forms. *Probability Theory*, 327:87–92, 1998.

[4] D. Brigo, F. Mercurio, F. Rapisarda, and R. Scotti. *Approximated moment-matching dynamics for basket options simulation*. Product and Business Development Group, Banca IMI, 2002. Working paper.

[5] M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management Science*, 42(2):269–285, 1996.

[6] S. Burgos and M. B. Giles. The computation of greeks with multilevel monte carlo. 2011. arXiv:1102.1348.

[7] L. Capriotti. Fast greeks by algorithmic differentiation. *Journal of Computational Finance*, 14(3):3–35, 2011.

[8] L. Capriotti. Likelihood ratio method and algorithmic differentiation: fast second order greeks. Preprint SSRN:1828503, 2014.

[9] P. S. Dywer and M. S. Macphail. Symbolic matrix derivatives. *The Annals of Mathematical Statistics*, 19(4):517–534, 1948.

[10] E. Fournié, J. M. Lasry, J. Lebuchoux, and P. L. Lions. Application of Malliavin calculus to Monte Carlo methods in finance. *Finance and Stochastics*, 2(5):201–236, 2001.

[11] M. Giles and P. Glasserman. Smoking adjoints: fast evaluation of greeks in Monte Carlo calculations. NA-05/15, Numerical Analysis Group, Oxford University, July 2005.

[12] M. B. Giles. Vibrato Monte Carlo sensitivities. In P. L'Ecuyer and A. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 369–382. New York, Springer edition, 2009.

[13] M. B. Giles. Monte carlo evalutation of sensitivities in computational finance. September 20–22, 2007.

[14] P. Glasserman. *Gradient estimation via pertubation analysis*. Kluwer Academic Publishers, Norwell, Mass, 1991.

[15] P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53 of *Application of Mathematics*. Springer, New York, 2003. xiii+598 pp., ISBN 0-387-00451-3.

[16] P. Glasserman and X. Zhao. Fast greeks by simulation in forward LIBOR models. *Journal of Computational Finance*, 3(1):5–39, 1999.

[17] P. W. Glynn. Likelihood ratio gradient estimation: an overview. In *Proceedings of the Winter Simulation Conference*, pages 366–374, New York, 1987. IEEE Press.

[18] E. Gobet and R. Munos. Sensitivity analysing using Itô-Malliavin calculus and martingales: applications to stochastic optimal control. *SIAM Journal on Control and Optimization*, 43(5):1676–1713, 2005.

[19] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. 1989. Kluwer Akedemic Publishern Dordrecht.

[20] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2008. xxi+426 pp., ISBN 978-0-89871-659-7.

[21] A. Griewank and A. Walther. *ADOL-C: A Package for the Automatic differentiation of algorithm written in C/C++*. University of Paderborn, Germany, 2010.

[22] L. Hascoët and V. Pascual. The Tapenade automatic differentiation tool: principles, model, and specification. *ACM Transactions On Mathematical Software*, 39(3), 2013.

[23] S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–343, 1993.

[24] Y. C. Ho and X. R. Cao. Optimization and perturbation analysis of queuing networks. *Journal of Optimization Theory and Applications*, 40:559–582, 1983.

[25] R. J. Hogan. Fast reverse-mode automatic differentiation using expression templates in C++. *Transactions on Mathematical Software*, 40(26):1–26, 2014.

[26] C. Homescu. Adjoints and automatic (algorihmtic) differentiation in computational finance. arXiv:1107.1831, 2011.

[27] H. Kunita. *Stochastic Flows and Stochastic Differential Equations*. Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, 1990. xiv+361 pp., ISBN 0-521-35050-6.

[28] P. L'Ecuyer. A unified view of the ipa, sf and lr gradient estimation techniques. *Management Science*, 36(11):1364–1383, 1990.

[29] V. Lemaire and G. Pagès. Multistep Richardson-Romberg extrapolation. Preprint arXiv:1401.1177, 2014.

[30] E. Levy. Pricing European average rate and currency options. *Journal of International Money and Finance*, 11:474–491, 1992.

[31] F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: a simple least squares approach. *Review of Financial Studies*, 14:113–148, 2001.

[32] R. L. Mishkov. Generalization of the formula of fa'a di Bruno for a composite function with a vector argument. *Internation Journal of Mathematics and Mathematical Sciences*, 24(7):481–491, 2000.

[33] U. Naumann. *The art of differentiating computer programs: an introduction to algorithmic differentiation*. Software, Enivronments and Tools. SIAM, RWTH Aachen University, Aachen, Germany, 2012. xviii+333 pp., ISBN 978-1-61197-206-1.

[34] D. Nualart. *The Malliavin calculus and related topics*. Probability and its Applications. Springer-Verlag, Berlin, 2006. x+390 pp., ISBN 978-3-540-28328-7.

[35] O. Pironneau. *Automatic differentiation for financial engineering*. Université Pierre et Marie Curie, Paris VI, 2008.

[36] M. Reiman and A. Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37:830–844, 1989.

[37] R. Rubinstein. Sensitivity analysis and performance extrapolation for computer simulation models. *Operations Research*, 37:72–81, 1989.

[38] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998.

[39] R. Suri and M. Zazanis. Pertubation analysis gives strongly consistent sensitivity estimates for the m/g/1 queue. *Management Science*, 34:39–64, 1988.

TABLE 3

Results for the price, the Delta and the Gamma of a Basket Option priced with the moment-matching approximation (reference values), Finite Difference on Monte Carlo and Vibrato plus Automatic Differentiation on Monte Carlo. The settings of Monte Carlo simulation are 1 *time step and* $1,000,000$ *simulation paths.*

d	T	Price AMM	Price (MC)	Delta AMM	Delta Vibrato (MC)	Delta FD (MC)	Gamma AMM	Gamma VAD (MC)	Gamma FD (MC)
1	0.1	38.4285	37.3823	0.55226	0.55146	0.55423	4.65557e-3	4.66167e-3	4.64998e-3
2	0.1	34.4401	34.1232	0.27452	0.27275	0.28467	1.28903e-3	1.34918e-3	1.28193e-3
3	0.1	46.0780	45.9829	0.18319	0.18220	0.18608	4.29144e-4	4.28572e-4	4.21012e-4
4	0.1	59.6741	58.7849	0.13750	0.13639	0.14147	1.86107e-4	1.93238e-4	1.79094e-4
5	0.1	92.8481	90.9001	0.10974	0.10889	0.10956	7.64516e-5	7.79678e-5	7.59901e-5
6	0.1	139.235	141.766	0.09128	0.09017	0.09048	3.54213e-5	3.71834e-5	3.41114e-5
7	0.1	155.492	153.392	0.07820	0.07744	0.07766	2.31624e-5	2.09012e-5	2.18123e-5
1	1	155.389	154.797	0.66111	0.66039	0.67277	1.30807e-3	1.30033e-3	1.32812e-3
2	1	135.441	133.101	0.32583	0.32186	0.32547	3.80685e-4	3.86998e-4	3.83823e-4
3	1	181.935	182.642	0.21775	0.21497	0.21619	1.26546e-4	1.34423e-4	1.24927e-4
4	1	234.985	232.018	0.16304	0.16055	0.01610	5.49161e-5	5.62931e-5	5.50990e-5
5	1	364.651	363.363	0.13023	0.12780	0.12804	2.25892e-5	2.38273e-5	2.19203e-5
6	1	543.629	540.870	0.10794	0.10477	0.10489	1.04115e-5	8.99834e-6	1.13878e-5
7	1	603.818	607.231	0.92420	0.08995	0.89945	6.87063e-6	7.70388e-6	7.22849e-6

TABLE 4

Time computing (in seconds) for the Gamma with Finite Difference on Monte Carlo and with Vibrato plus Automatic Differentiation on Monte Carlo simulation, dimension of the problem are varying. The settings of Monte Carlo algorithm are the same as above.

Method (Computing Gamma)	$d=1$	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$
FD (MC)	0.49	0.95	1.33	1.82	2.26	2.91	3.36
VAD (MC)	0.54	0.77	0.92	1.21	1.50	1.86	2.31

TABLE 5

Results of the price, the Delta and the Gamma of an American option. The reference values are obtained via the Semi-Newton method plus Finite Difference, they are compared to Vibrato plus Automatic Differentiation on the Longstaff-Schwartz algorithm. We compute the standard error for each American Monte Carlo results. The settings of the American Monte Carlo are 50 *time steps and* 50,000 *simulation paths.*

S	σ	T	Price Ref. Value	Price (AMC)	Standard Error	Delta Ref. Value	Delta Vibrato (AMC)	Standard Error	Gamma Ref. Value	Gamma VAD (AMC)	Standard Error
36	0.2	1	4.47919	4.46289	0.013	0.68559	0.68123	1.820e-3	0.08732	0.06745	6.947e-5
36	0.2	2	4.83852	4.81523	0.016	0.61860	0.59934	1.813e-3	0.07381	0.06398	6.846e-5
36	0.4	1	7.07132	7.07985	0.016	0.51019	0.51187	1.674e-3	0.03305	0.03546	4.852e-5
36	0.4	2	8.44139	8.45612	0.024	0.44528	0.44102	1.488e-3	0.02510	0.02591	5.023e-5
38	0.2	1	3.24164	3.23324	0.013	0.53781	0.53063	1.821e-3	0.07349	0.07219	1.198e-4
38	0.2	2	3.74004	3.72705	0.015	0.48612	0.46732	1.669e-3	0.05907	0.05789	1.111e-4
38	0.4	1	6.11553	6.11209	0.016	0.44726	0.45079	1.453e-3	0.02989	0.03081	5.465e-5
38	0.4	2	7.59964	7.61031	0.025	0.39786	0.39503	1.922e-3	0.02233	0.02342	4.827e-5
40	0.2	1	2.31021	2.30565	0.012	0.41106	0.40780	1.880e-3	0.06014	0.05954	1.213e-4
40	0.2	2	2.87877	2.86072	0.014	0.38017	0.39266	1.747e-3	0.04717	0.04567	5.175e-4
40	0.4	1	5.27933	5.28741	0.015	0.39051	0.39485	1.629e-3	0.02689	0.02798	1.249e-5
40	0.4	2	6.84733	6.85873	0.026	0.35568	0.35446	1.416e-3	0.01987	0.02050	3.989e-5
42	0.2	1	1.61364	1.60788	0.011	0.30614	0.29712	1.734e-3	0.04764	0.04563	4.797e-5
42	0.2	2	2.20694	2.19079	0.014	0.29575	0.28175	1.601e-3	0.03749	0.03601	5.560e-5
42	0.4	1	4.55055	4.57191	0.015	0.33973	0.34385	1.517e-3	0.02391	0.02426	3.194e-5
42	0.4	2	6.17459	6.18424	0.023	0.31815	0.29943	1.347e-3	0.01768	0.01748	2.961e-5
44	0.2	1	1.10813	1.09648	0.009	0.21302	0.20571	1.503e-3	0.03653	0.03438	1.486e-4
44	0.2	2	1.68566	1.66903	0.012	0.22883	0.21972	1.487e-3	0.02960	0.02765	2.363e-4
44	0.4	1	3.91751	3.90838	0.015	0.29466	0.29764	1.403e-3	0.02116	0.02086	1.274e-4
44	0.4	2	5.57268	5.58252	0.028	0.28474	0.28447	1.325e-3	0.01574	0.01520	2.162e-4