Second order Monte Carlo sensitivities in linear or constant time

Roberto Daluiso*

First submitted: March 24, 2019

Abstract

We consider the problem of computing efficiently the full matrix of second order sensitivities of a Monte Carlo price when the number of inputs is large. Specifically, we analyse and compare methods whose run time is at most $O(N \cdot T)$, where N is the dimension of the input and T is the time required to compute the price. Since none of the alternatives from previous literature appears satisfactory in all settings, we propose two original methods: the first one is based on differentiation in distributional sense, while the second one leverages a functional relation between first and second order derivatives. The former shows excellent generality and computational times to achieve a given target accuracy. The latter is by far the most effective in at least one relevant example, and has a theoretical interest, being the first practical estimator of the full Hessian whose complexity, as a multiple of that of the only-price implementation, does not grow with the dimension of the problem.

Keywords: Greeks; Gamma; algorithmic differentiation; derivatives pricing.

Key messages:

- The N-by-N Hessian of a price can be estimated in less than $O(N^2)$ time.
- A new $O(N)$ estimator outdoes legacy methods in accuracy and/or applicability.
- An $O(1)$ estimator exists with acceptable and sometimes outstanding accuracy.

1 Introduction

In the active management of portfolios of financial products, the sensitivity of the net position to movements of the underlying risk factors is a most precious piece of information. Indeed, one may want to monitor which market shifts could cause a significant jump in the value of the portfolio, and maybe build a hedging portfolio chosen to mitigate the corresponding sensitivities.

In mathematical terms, sensitivities are usually expressed by derivatives of the pricing functional P (also called Greeks). First order Greeks are obviously the most important indicator for a trader to see whether his overall position is approximately

*Interest Rate and Credit Models, Banca IMI. Address: Largo Mattioli 2, 20100 Milano (Italy). Email: roberto.daluiso@bancaimi.com.

market neutral, and are often kept under strict control by dynamically adjusting the hedging positions as time passes. A direct consequence of this behaviour is that the day-to-day performance of the trading desk is at leading order driven by second order Greeks (also called Gammas). By the way, this is the discrete-time counterpart of the fact that in an idealised Black-Scholes world, the continuously rebalanced delta hedging replication strategy generates an Ito component $\frac{1}{2}\frac{\partial^2 P}{\partial X_0^2}dt$ in the dynamics of total wealth.

All of the above implies that the knowledge of second order sensitivities is often crucial; which is even more true when the payoff depends on several and maybe correlated risk factors, whose co-movements impact the portfolio value through the mixed terms in the Hessian matrix (sometimes called Cross Gammas). This is the case for instance of basket products, indices, multi-asset-class hybrids, and portfolio-level non-linear valuation adjustments such as CVA and DVA (Basel Committee on Banking Supervision, 2015).

Unfortunately, these are exactly the settings in which the pricing exercise is computationally more demanding, as it often involves a multidimensional Monte Carlo simulation. This rules out as infeasible the simplest way to approximate the full $N \times N$ pricing Hessian $\mathbf{H}$, i.e.

$$H_{ij} \approx \frac{P(\boldsymbol{\psi} + \boldsymbol{h}_i + \boldsymbol{h}_j) - P(\boldsymbol{\psi} + \boldsymbol{h}_i - \boldsymbol{h}_j) - P(\boldsymbol{\psi} - \boldsymbol{h}_i + \boldsymbol{h}_j) + P(\boldsymbol{\psi} - \boldsymbol{h}_i - \boldsymbol{h}_j)}{(1 + 3\delta_{ij})h^2},$$

where $\boldsymbol{\psi}$ is the vector of pricing inputs of interest, h is a small displacement and $\boldsymbol{h}_i := h\boldsymbol{e}_i$ is h times the i-th element of the canonical basis of $\mathbb{R}^N$: this method would involve $O(N^2)$ full revaluations.

In fact, the gradient ∇P can be computed in $O(T)$ where T is the time to compute P, thanks to adjoint algorithmic differentiation (AAD, see Giles, 2007; Capriotti, 2011; Capriotti and Giles, 2012), which is more and more widespread for first order sensitivities among practitioners. This technique and its generalizations (e.g. Giles, 2009; Chan and Joshi, 2015; Daluiso and Facchinetti, 2018) coupled with the approximation

$$H_{ij} \approx \frac{\nabla_j P(\boldsymbol{\psi} + h\boldsymbol{e}_i) - \nabla_j P(\boldsymbol{\psi} - h\boldsymbol{e}_i)}{2h} \qquad \text{or} \qquad H_{ij} \approx \frac{\nabla_j P(\boldsymbol{\psi} + h\boldsymbol{e}_i) - \nabla_j P(\boldsymbol{\psi})}{h},$$

would already give an estimator in $O(N \cdot T)$, but this is in practice quite unsatisfactory, since for small h the result is very noisy, while for moderate h it is too much biased.

The latter fact stimulated research of alternative algorithms which could retain the $O(N \cdot T)$ cost while being unbiased and having less Monte Carlo variance. We point out in particular the significantly different contributions Capriotti (2015), Pagès et al. (2018) and Joshi and Zhu (2016), which seem unaware of one another and present numerical evidences which are difficult to compare.

Our first contribution is therefore a comparative analysis of the three above proposals, to see in which settings they can be applied. In fact, while not delving into the details of the estimators which are already well covered in the original papers, we sometimes go slightly beyond the authors: namely, by underlying a subtlety needed to correctly implement Capriotti (2015), and by developing a multi-fixing generalization of Pagès et al. (2018).

As we will see, in some situations the rich set of alternatives outlined above shrinks considerably: in particular, payoffs directly depending on the non random initial state or on static parameters (e.g. forwards and deterministic interest rates), and factorial models in general where the number of random drivers is smaller than the number of underlyings, are usually out of scope. This motivated the development of the second contribution of this paper, namely a fairly general-purpose method based on second order pathwise differentiation of the payoff in distributional sense.

Finally, most of the above methods run in $O(N \cdot T)$ time. This is a theoretical upper bound for general computer functions (see e.g. Griewank and Walther, 2008), but can the special structure of a pricing problem with diffusive underlyings change the verdict? To our knowledge, the only partial positive answer is provided by the already cited Joshi and Zhu (2016) in the case in which the simulation can be performed keeping at each time step only a low dimensional state variable.

In this respect, our third and last contribution is a positive answer to the above question in the complementary setting in which the Brownian driver is high dimensional (precisely, has dimensionality at least equal to the number of underlyings). Our solution is based on a relation linking a suitable first order sensitivity to $\partial^2 P/\partial \boldsymbol{X}_0^2$ via the pricing PDE, and yields the latter in $O(T)$ time by an easily implementable wrapper of any $O(T)$ first order estimator. In fact, also LRMAAD from Capriotti (2015) has constant cost and almost the same perimeter of applicability, but from the numerical analysis below it turns out to have too large Monte Carlo uncertainties to be an option in practice; while we will show at least one relevant example in which our new constant-cost algorithm significantly outperforms all its linear-cost competitors.

The rest of the paper is structured as follows. Section 2 states the problem in the diffusive setting in which all our analyses will be performed; Section 3 reviews the methods proposed in the literature; Section 4 introduces the new methods; Section 5 compares the empirical performances and Monte Carlo uncertainties on several test cases; finally, Section 6 summarizes the main findings and concludes.

2 Setting

We consider a description of the market where the risk factors $\boldsymbol{X}_t$ are modelled as an $\mathbb{R}^{N_X}$-valued diffusion driven by an N_W-dimensional Wiener process $\boldsymbol{W}_t$ under a fixed probability measure $\mathbb{P}$:

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{\theta}, \boldsymbol{X}_t, t)\mathrm{d}t + \boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{X}_t, t)\mathrm{d}\boldsymbol{W}_t, \qquad t \geq 0, \tag{2.1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$ is a vector of parameters. A price P of interest is expressed as an expected value

$$P = \mathbb{E}\left[g\left(\boldsymbol{X}_{T_1}, \ldots, \boldsymbol{X}_{T_M}, \boldsymbol{\theta}\right)\right] =: \mathbb{E}\left[g\left(\boldsymbol{X}_{\boldsymbol{T}}, \boldsymbol{\theta}\right)\right], \tag{2.2}$$

and is estimated by Monte Carlo simulation; to be concrete, we suppose that the simulation is performed by Euler stepping on a time grid $\{S_0, \ldots, S_L\} = \boldsymbol{S} \supseteq \boldsymbol{T}$:

$$\begin{gathered} \boldsymbol{X}_{S_{i+1}} = \boldsymbol{X}_{S_i} + \boldsymbol{\mu}\left(\boldsymbol{\theta}, \boldsymbol{X}_{S_i}, S_i\right)\Delta_i S + \boldsymbol{\Sigma}\left(\boldsymbol{\theta}, \boldsymbol{X}_{S_i}, S_i\right)\sqrt{\Delta_i S}\cdot \boldsymbol{Z}^{(i)}, \\ \Delta_i S := S_{i+1} - S_i, \quad \boldsymbol{Z}^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I}_d), \end{gathered} \tag{2.3}$$

although one could imagine more or less straightforward generalizations of the methods described below for more general schemes.

When the payoff does not depend on $\boldsymbol{\theta}$ directly, we will write

$$g\left(\boldsymbol{X_T}, \boldsymbol{\theta}\right) =: p\left(\boldsymbol{X_T}\right); \tag{2.4}$$

we will call "autonomous" this simpler case. Note that the price P still depends on $\boldsymbol{\theta}$ through the dynamics (2.1).

We aim at computing the full Gamma matrix

$$\boldsymbol{\Gamma} := \frac{\partial^2 P}{\partial \boldsymbol{X}_0^2},$$

and possibly the full Hessian

$$\mathbf{H} := \frac{\partial^2 P}{\partial \boldsymbol{\psi}^2}, \qquad \boldsymbol{\psi} := \left(\boldsymbol{X}_0, \boldsymbol{\theta}\right),$$

in acceptable time when N_X and N_θ are large. Note that a naive approach based on finite differences would involve respectively $O(N_X^2)$ and

$$O\left(\left(N_X + N_\theta\right)^2\right) =: O\left(N^2\right)$$

re-evaluations of the price P with displaced values of $\boldsymbol{X}_0$ and possibly $\boldsymbol{\theta}$.

Throughout the paper, the following notational conventions are adopted: scalars are in italic (e.g. a), vectors in boldface italic (e.g. $\boldsymbol{a}$ has components a_i), matrices in straight boldface (e.g. $\mathbf{A}$ has entries A_{ij}); vectors are interpreted as columns unless transposed; for a vector $\boldsymbol{a}$, the gradient $\bar{\boldsymbol{a}} = \frac{\partial P}{\partial \boldsymbol{a}}$ is a row vector; for a matrix $\mathbf{A}$, the gradient $\bar{\mathbf{A}} = \frac{\partial P}{\partial \mathbf{A}}$ is a matrix with components $\bar{A}_{ij} = \frac{\partial P}{\partial A_{ji}}$ (note the inversion of indices). Finally, for stochastic processes, we will write the time argument in the subscript (e.g. $\boldsymbol{X}_t$) if no confusion arises, or as a function argument (e.g. $X_i(t)$) otherwise.

3 Review of existing algorithms

In this section, we review from an algorithmic point of view the main methods based on the existing literature whose theoretical complexity is $O(N \cdot T)$, with a focus on the perimeter of applicability within the setting of Section 2. For quick reference, we gather in Table 1 the essential traits of all these algorithms.

3.1 Finite differences of pathwise adjoint

Before describing more complex methods, let us mention that the computation of finite differences of the pathwise adjoint estimator is already a linear cost method. Indeed, first order pathwise adjoints cost at most $4T$; repeating their computation N times for small directional displacements of $\boldsymbol{\psi}$, one trivially gets an estimator for the Hessian, which will be denoted by the acronym FDPW.

However, we expect poor variance properties of this naive method, as finite differences have high Monte Carlo uncertainties when the integrand is discontinuous, as is the case for the gradient of typical payoffs. Therefore, we explore in the following sections analytical alternatives levering on the structure of the problem.

Name	1st order	2nd order	Section	Reference
FDIFF2	finite difference	finite difference	1	N.A.
FDPW	pathwise AD	finite difference	3.1	N.A.
LRMAAD	pathwise AD	likelihood ratio	3.2	Capriotti (2015)
AADLRM	likelihood ratio	pathwise AD	3.2	Capriotti (2015)
VAD	vibrato	pathwise AD	3.3	Pagès et al. (2018)
VFD	vibrato	finite difference	3.3	N.A.
HOPP	change var + AD	pathwise AD	3.4	Joshi and Zhu (2016)

Table 1: Legacy methods to compute second order sensitivities: estimators used for first and second order differentiation, references. Here and elsewhere in the paper, AD stands for algorithmic differentiation.

3.2 Likelihood ratio and adjoint differentiation

The idea of Capriotti (2015) is to combine pathwise differentiation with the likelihood ratio method. Since the latter can handle only the autonomous case, the same limitation applies to the resulting second order methods; moreover, as is clear from the matrix inversions in the equations below, one needs that $N_X \leq N_W$ with $\boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{X}_T, T)$ almost surely full rank. As we believe that a crucial detail in the correct interpretation of the final formula may be easily overlooked, we repeat the derivation briefly.

When adjoint algorithmic differentiation is applied first, one wants to compute

$$\frac{\partial}{\partial \boldsymbol{\psi}} \mathbb{E}\left[\frac{\partial p}{\partial \boldsymbol{X_T}}(\boldsymbol{X_T}) \frac{\partial \boldsymbol{X_T}}{\partial \boldsymbol{\psi}}(\boldsymbol{Z}, \boldsymbol{\psi})\right]$$

by likelihood ratio. In order to do so, one *must* rewrite the argument of $\mathbb{E}$ so that the only random variable appearing is $\boldsymbol{X_S}$, because the method relies on writing the expectation as an integral against the density of $\boldsymbol{X_S}$. Fortunately, $\boldsymbol{Z}^{(i)}$ is readily expressed as a function of $\boldsymbol{X_S}$ and $\boldsymbol{\theta}$ as

$$z^{(i)}(\boldsymbol{X_S}, \boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta}, \boldsymbol{X}_{S_i}, S_i)^{-1} (\Delta_i S)^{-\frac{1}{2}} \left(\boldsymbol{X}_{S_{i+1}} - \boldsymbol{X}_{S_i} - \boldsymbol{\mu}(\boldsymbol{\theta}, \boldsymbol{X}_{S_i}, S_i) \Delta_i S\right).$$

Now a straightforward generalization of the likelihood ratio method to the case in which the integrand has explicit dependence on the differentiation parameter leads to the following estimator of the Hessian $\mathbf{H}$:

$$\left(\frac{\partial p}{\partial \boldsymbol{X_T}}(\boldsymbol{X_T}) \frac{\partial \boldsymbol{X_T}}{\partial \boldsymbol{\psi}}(\boldsymbol{Z}, \boldsymbol{\psi})\right)^{\intercal} \boldsymbol{\Omega}_{\boldsymbol{\psi}} + \frac{\partial}{\partial \boldsymbol{\psi}}\left[\frac{\partial p}{\partial \boldsymbol{X_T}}(\boldsymbol{X_T}) \frac{\partial \boldsymbol{X_T}}{\partial \boldsymbol{\psi}}\left(z(\boldsymbol{X_S}, \boldsymbol{\theta}), \boldsymbol{\psi}\right)\right], \quad (3.1)$$

where the $\boldsymbol{\Omega}_{\boldsymbol{\psi}}$ is the customary likelihood ratio weight capturing the component of the derivative due to dependence of the distribution of $\boldsymbol{X_S}$ on $\boldsymbol{\psi}$ by differentiation of the log-density, while the future path $\boldsymbol{X}_{\boldsymbol{S}\setminus\{0\}}$ in the square bracket is a constant for the purpose of differentiation. Note that in general the second addend cannot be replaced by the simpler

$$\frac{\partial p}{\partial \boldsymbol{X_T}}(\boldsymbol{X_T}) \frac{\partial}{\partial \boldsymbol{\psi}}\left[\frac{\partial \boldsymbol{X_T}}{\partial \boldsymbol{\psi}}(\boldsymbol{Z}, \boldsymbol{\psi})\right]$$

which seems implicit in the implementation described by Capriotti (2015). The resulting algorithm is denoted by the acronym LRMAAD.

From the run time point of view, the second addend of (3.1) involves differentiation of the $\mathbb{R}^N$-valued function in square brackets, implying that the computational cost grows linearly with N. This result cannot be improved in general, since computing an N-by-N Jacobian multiplies the run time of a function by $O(N)$, regardless of whether one uses AAD as suggested in the original article, or a simpler forward differentiation (Griewank and Walther, 2008), or even finite differences.

However, if one wants to determine only $\mathbf{\Gamma}$, then this Jacobian term can be computed very efficiently. Indeed, for $\boldsymbol{\psi} = \boldsymbol{X}_0$ it reduces to

$$\frac{\partial}{\partial \boldsymbol{X}_0}\left[\frac{\partial p}{\partial \boldsymbol{X}_{\boldsymbol{T}}}(\boldsymbol{X}_{\boldsymbol{T}})\frac{\partial \boldsymbol{X}_{\boldsymbol{T}}}{\partial \boldsymbol{X}_0}\left(z(\boldsymbol{X}_{\boldsymbol{S}}), \boldsymbol{X}_0\right)\right];$$

now, one can easily note that if the quantity in square brackets is computed by AAD, then in the implementation it takes the form

$$\bar{\boldsymbol{X}}_{S_1} \cdot \frac{\partial \boldsymbol{X}_{S_1}}{\partial \boldsymbol{X}_0}\left(z(\boldsymbol{X}_{S_1}, \boldsymbol{X}_0), \boldsymbol{X}_0\right)$$

for a suitable random variable $\bar{\boldsymbol{X}}_{S_1}$ *not depending on* $\boldsymbol{X}_0$. Once this is remarked, differentiation of this term is so fast to become practically negligible, and the complexity of the algorithm is dominated by the first term in (3.1), computed in $O(T)$ time. Note that even there, the likelihood ratio weight $\mathbf{\Omega}_{\boldsymbol{X}_0}$ has non-trivial dependence on X_0 only for what concerns the first Euler step $i = 0$: this makes the overall runtime essentially comparable to that of computing the first order pathwise estimator

$$\frac{\partial p}{\partial \boldsymbol{X}_{\boldsymbol{T}}}(\boldsymbol{X}_{\boldsymbol{T}})\frac{\partial \boldsymbol{X}_{\boldsymbol{T}}}{\partial \boldsymbol{X}_0}(\boldsymbol{Z}, \boldsymbol{X}_0).$$

Instead, if one differentiates pathwise the first order likelihood ratio estimator, an algorithm called AADLRM is obtained. It involves the Jacobian of the calculation of the weight vector $\mathbf{\Omega}_{\boldsymbol{\psi}}$, inclusive of the simulation scheme, which again implies a potential $O(N)$ factor on run times. The original paper notes that this second method is empirically slower on a simple example, maybe because of the above $\mathbf{\Gamma}$-specific speedup; however, we feel that a comparison of Monte Carlo variances is needed before discarding AADLRM as less performing.

3.3 Vibrato and adjoint differentiation

Pagès et al. (2018) apply automatic differentiation to a different first order estimator called Vibrato Monte Carlo, which has much better sample variance properties especially after the introduction of a trick based on the antithetic transform.[1] (They also consider the possibility of applying Vibrato twice, but they find no relevant difference.) As in the previous section, the analysis works only for the autonomous case and under the hypothesis $N_X \leq N_W$ with $\mathbf{\Sigma}(\boldsymbol{\theta}, \boldsymbol{X}_T, T)$ almost surely full rank.

The algorithm to compute the estimator, here generalized to $M > 1$ fixing times in the spirit of what Giles (2009) mentions for first order Vibrato, is as follows:

[1] Note that the antithetic idea may also be beneficial for the likelihood ratio method itself, as first remarked in Capriotti (2008).

Algorithm 3.1 (VAD). [2]

1. For each fixing time $T_i = S_{t(i)+1}$, compute the derivatives of the (approximate) Gaussian log-density of $\boldsymbol{X}_{T_i}$ given its neighbours in the simulation grid $\boldsymbol{S}$; if $T_i = S_L$ is the maturity date of the payoff, this is given by the Euler step (2.3), otherwise it is described by the law of a Brownian bridge based interpolation.

2. For each fixing time $T_i = S_{t(i)+1}$, simulate $\boldsymbol{X}_{T_i}$ from this conditional law, by an inner sub-Montecarlo consiting of few independent draws and their corresponding antithetic variates. On these scenarios, use the conditional likelihood ratio weights computed at step 1. to estimate the first order price sensitivities to the realized drift and diffusion coefficients.

3. Propagate backwards the so obtained sensitivities through the adjoint of the drift and diffusion computation and through the adjoint of the Euler scheme.

4. Apply again automatic differentiation to all of the above steps.

Note that the last step multiplies by $O(N)$ the cost of the whole algorithm. As for the methods of the previous section, if the payoff is discontinuous then the method cannot be applied as is;[3] However, in this case we can conceive to perform the last step with a finite difference with non-small displacement: we will denote this variant as VFD.

3.4 Optimal partial proxy for Hessians

In real payoffs, direct pathwise second order differentiation is not possible only because of a few discontinuity points of the gradient or of the payoff. Exploiting this fact, Joshi and Zhu (2016) manage to change the integrand without changing the integral, so that one can differentiate algorithmically twice the modified payoff. More precisely, they rewrite the expected value as a function of a multivariate uniform $\boldsymbol{U}$,

$$P = \mathbb{E}\left[g\left(\boldsymbol{X}_{\boldsymbol{T}}, \boldsymbol{\theta}\right)\right] =: \mathbb{E}\left[g\left(\boldsymbol{x}_{\boldsymbol{T}}(\boldsymbol{Z}, \boldsymbol{\psi}), \boldsymbol{\theta}\right)\right] =: \mathbb{E}\left[g\left(\boldsymbol{x}_{\boldsymbol{T}}(\mathcal{N}^{-1}(\boldsymbol{U}), \boldsymbol{\psi}), \boldsymbol{\theta}\right)\right]$$

and then perform a change of variables $\boldsymbol{V} = \boldsymbol{v}(\boldsymbol{U}, \boldsymbol{\psi})$, designed in such a way that while moving $\boldsymbol{\psi}$ from its unperturbed value $\boldsymbol{\psi}_0$, the path generated by $\boldsymbol{v}(\boldsymbol{U}, \boldsymbol{\psi})$ does not cross the discontinuities. After the change of variables,

$$P = \mathbb{E}\left[g\left(\boldsymbol{x}_{\boldsymbol{T}}(\mathcal{N}^{-1}(\boldsymbol{v}(\boldsymbol{U}, \boldsymbol{\psi})), \boldsymbol{\psi}), \boldsymbol{\theta}\right) \cdot \omega(\boldsymbol{U}, \boldsymbol{\psi})\right]$$

for a suitable weight ω; the mapping $\boldsymbol{v}$ ensures that if the payoff is smooth except when

$$U_{N_W}^{(t(i))} = a_j^{(i)}\left(\boldsymbol{\psi}, \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(t(i)-1)}, U_1^{(t(i))}, \ldots, U_{N_W-1}^{(t(i))}\right) \qquad \text{for some } j,$$

[2] The algorithms in the previous section are called LRMAAD when LRM is applied *after* AAD, and AADLRM otherwise; while here, somewhat confusingly, VAD is the method which applies V(ibrato) *before* AD. We keep the acronyms chosen by the original authors nonetheless.

[3] Of course one can always smooth out the payoff, as is suggested in the original paper; see also the discussion at the beginning of Section 5.

where $\{T_{t(i)+1}\}_{i=1,\dots,h}$ are the maturities of the payoff's discontinuities, then[4]

$$U_{N_W}^{(t(i))} \in \left[a_j^{(i)}(\boldsymbol{\psi}_0, \boldsymbol{U}), a_{j+1}^{(i)}(\boldsymbol{\psi}_0, \boldsymbol{U})\right) \iff \forall\, \boldsymbol{\psi}, V_{N_W}^{(t(i))} \in \left[a_j^{(i)}(\boldsymbol{\psi}, \boldsymbol{V}), a_{j+1}^{(i)}(\boldsymbol{\psi}, \boldsymbol{V})\right).$$

Joshi and Zhu (2016) proceed suggesting an implementation based on a smart backpropagation of second order derivatives (Joshi and Yang, 2011), which should improve the computational burden when there are significantly fewer state variables in the stochastic process than parameters in the model. We do not pursue this level of optimization in our implementation, which aims at being generic; the analysis of the applicability and efficacy of such model-dependent and payoff-dependent trick on the different competitors described in this paper could be the subject of future research.[5] For the moment being, we sketch a less *ad hoc* algorithm to compute the estimator:

Algorithm 3.2 (HOPP)**.**

1. During simulation, compute the critical values $a_j^{(i)}$, the transformed drivers $\boldsymbol{V}$ and the weight ω. (In fact we know *a priori* that $\boldsymbol{V} = \boldsymbol{U}$ and $\omega = 1$ at $\boldsymbol{\psi} = \boldsymbol{\psi}_0$, but the instructions computing $\boldsymbol{V}$ and ω should be taken into account when in the sequel this step will be differentiated.)

2. Multiply the payoff g by the weight ω (with the same remark about $\omega = 1$).

3. Differentiate algorithmically twice the previous two steps, inclusive of simulation, with respect to $\boldsymbol{\psi}$ for fixed $\boldsymbol{\psi}_0$.

Step 3 is an $O(N)$ multiplier on the execution time of the first two steps, which in turn is typically little more than that of a plain Monte Carlo run, the overhead of computing ω being slightly significant only when the discontinuities are very frequent as in barrier options.

4 New methods

In this section we move to new proposals which try to overcome some limitations of the above methods. Table 2 completes with the contents of this section the summary which was provided in Table 1 for the algorithms of Section 3.

4.1 Second order differentiation by conditioning

We isolate in this small section a simple mathematical property, which will be used to derive the new second order method of Section 4.2, but which is meaningful *per se*. The general question is whether the ability to compute efficiently an estimator of the Hessian of a *conditional* expectation can enable the fast computation of the Hessian of the unconditional expectation.

[4] At least to second order, as the original article points out.

[5] Note that the optimization would not apply anyway to the numerical tests of this paper, whose validity is therefore unaffected by the simplification.

Name	1st order	2nd order	Section
FDDAAD	distributional AD	finite differences	4.2
DAAD2	(distributional) AD	distributional AD	4.2
FΓ	any	none	4.3

Table 2: New methods to compute second order sensitivities: estimators used for first and second order differentiation, references. See also Table 1 for legacy methods.

More specifically, we take a set of times $\widetilde{\boldsymbol{T}} \subseteq \boldsymbol{S}$, and let

$$\varepsilon\left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right) = \mathbb{E}\left[g\left(\boldsymbol{X}_{\boldsymbol{T}}, \boldsymbol{\theta}\right) | \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}\right].$$

Then one can compute the desired Hessian as $\mathbb{E}\left[\frac{\partial^2}{\partial \boldsymbol{\psi}^2}\varepsilon\left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)\right]$, i.e.

$$\mathbb{E}\left[\left(\frac{\partial(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})}{\partial \boldsymbol{\psi}}\right)^{\intercal} \frac{\partial^2 \varepsilon}{\partial\left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)^2}\left(\frac{\partial(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})}{\partial \boldsymbol{\psi}}\right) + \frac{\partial \varepsilon}{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}\left(\frac{\partial^2 \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}{\partial \boldsymbol{\psi}^2}\right)\right], \tag{4.1}$$

where by the tower rule, one can substitute the terms

$$\frac{\partial \varepsilon}{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}, \frac{\partial^2 \varepsilon}{\partial\left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)^2}$$

with $\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}$-conditional sample estimators. We note that the second addend of (4.1) is readily computed in linear time by differentiation of the function

$$\boldsymbol{\psi} \mapsto \bar{\boldsymbol{X}}_{\widetilde{\boldsymbol{T}}} \cdot \frac{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}{\partial \boldsymbol{\psi}}, \qquad \text{with} \qquad \bar{\boldsymbol{X}}_{\widetilde{\boldsymbol{T}}} \text{ any estimator of } \frac{\partial \varepsilon}{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}},$$

provided that $\bar{\boldsymbol{X}}_{\widetilde{\boldsymbol{T}}}$ is interpreted as an exogenous constant vector: which function is easily computed by pathwise AAD. The Jacobians in the first addend of (4.1) are similarly not a problem, and so we just need an estimator of the Hessian of ε, as desired.

4.2 Distributional algorithmic differentiation

The methods of Section 3 are devised to avoid at least in part the need of second order differentiation of the payoff, because its first derivative is typically discontinuous: a vanilla call already has

$$\frac{\mathrm{d}}{\mathrm{d}X}(X-K)^+ = I_{X>K}.$$

However, Daluiso and Facchinetti (2018) present a generalization of the first order pathwise method which works for discontinuous payoffs, based on careful handling of the Dirac deltas formally arising from differentiation of Heaviside functions. Then on the one hand, by finite differences one may immediately form a simple biased second order estimator for discontinuous payoffs (FDDAAD); on the other hand, a

new unbiased method for both continuous and discontinuous payoffs working also in the non-autonomous case can be introduced as follows.

As in most applications, suppose that there exist smooth scalar functions

$$\tilde{f}_i(\boldsymbol{X_T}, \boldsymbol{\theta}),\ \tilde{g}_{\boldsymbol{a}}(\boldsymbol{X_T}, \boldsymbol{\theta}) \qquad \text{for} \qquad i \in 1, \dots, h,\ \boldsymbol{a} \in \{-1, +1\}^h,$$

such that g is naturally represented as

$$g(\boldsymbol{X_T}, \boldsymbol{\theta}) = \tilde{g}_{\boldsymbol{a}}(\boldsymbol{X_T}, \boldsymbol{\theta}) \text{ on } \mathcal{X}_{\boldsymbol{a}} := \{(\boldsymbol{X_T}, \boldsymbol{\theta}) : a_i \cdot \tilde{f}_i(\boldsymbol{X_T}, \boldsymbol{\theta}) \geq 0\ \forall i \leq h\}. \tag{4.2}$$

We call $M_i = S_{t(i)+1}$ the last time in $\boldsymbol{T}$ such that $\tilde{f}_i$ depends on $\boldsymbol{X}_{M_i}$.

Now we exploit the conditioning idea of (4.1) using as $\widetilde{\boldsymbol{T}}$ the (often quite small) set obtained substituting in $\boldsymbol{T}$ each M_i with its first neighbours in the simulation grid $\boldsymbol{S}$. Recall that $\boldsymbol{X}_{M_i}$ can be simulated conditionally on its neighbours by Brownian bridge interpolation, because diffusions are locally Gaussian and the simulation grid is fine. With this choice of $\widetilde{\boldsymbol{T}}$, we can estimate the Hessian of ε by applying firstly pathwise differentiation to its definition (with distributional corrections if the payoff itself is discontinuous), and then distributional algorithmic differentiation to the result.

In more detail, call f_i, $g_{\boldsymbol{a}}$ the functions computing $\tilde{f}_i$, $\tilde{g}_{\boldsymbol{a}}$ from pre-simulated $\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}$, which functions include conditional simulation of $\boldsymbol{X}_{M_i}$. In view of (4.1), we only need an estimator of the Hessian of the conditional expectation ε, which thanks to our hypotheses has the following form:

$$\varepsilon = \mathbb{E}\left[\phi\left(I_{f_1(\boldsymbol{\Theta},\boldsymbol{Z})>0}, \dots, I_{f_h(\boldsymbol{\Theta},\boldsymbol{Z})>0}, \boldsymbol{\Theta}, \boldsymbol{Z}\right)\right] = \mathbb{E}\left[\phi\left(\boldsymbol{I}_{\boldsymbol{f}(\boldsymbol{\Theta},\boldsymbol{Z})>\boldsymbol{0}}, \boldsymbol{\Theta}, \boldsymbol{Z}\right)\right],$$

where $\boldsymbol{Z}$ is a standard Gaussian random vector, $\boldsymbol{\Theta} = (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})$ is a constant in the context of this expected value, and ϕ and the f_i are suitable functions, smooth in $\boldsymbol{\Theta}$ and $\boldsymbol{Z}$.

To proceed, we take from Daluiso and Facchinetti (2018) the following conventions: given a vector $\boldsymbol{v} \in \mathbb{R}^d$ and $k \in \{1, \dots, d\}$, we denote by $\boldsymbol{v}_{-k} \in \mathbb{R}^{d-1}$ the vector obtained from $\boldsymbol{v}$ removing its k-th component, and with $\boldsymbol{v}(v_k = x)$ the vector obtained from $\boldsymbol{v}$ substituting the k-th component with the value x; $\boldsymbol{I}_{\boldsymbol{v}>\boldsymbol{0}}$ is the vector $(I_{v_1>0}, \dots, I_{v_d>0})$, not to be confounded with the d-by-d identity matrix $\mathbf{I}_d$; finally, for a function ψ defined on $\{0,1\}^h$ and for $i = 1, \dots, h$, we define

$$\Delta^{(i)}\psi(\boldsymbol{a}) = \psi(\boldsymbol{a}(a_i = 1)) - \psi(\boldsymbol{a}(a_i = 0)).$$

We can now express the following weak assumption:

Hypothesis 4.1. f_i is twice differentiable with respect to $\boldsymbol{\Theta}$ and $z_{k(i)}$. Moreover, for almost every $\boldsymbol{Z}_{-k(i)}$, the function $z_{k(i)} \mapsto f_i(\boldsymbol{\Theta}, \boldsymbol{Z}(Z_{k(i)} = z_{k(i)}))$ is zero only for $z_{k(i)}$ in a finite set $A^i_{k(i)}(\boldsymbol{\Theta}, \boldsymbol{Z}_{-k(i)})$, and its derivative in such points is non-null.

Note that with our definition of f_i, this means that we should be able to find $k(i)$ such that the $k(i)$-th component of

$$\frac{\partial \tilde{f}_i}{\partial \boldsymbol{X}_{M_i}} \boldsymbol{\Sigma}^{(t(i))}$$

is almost-surely different from zero.

We also suppose for notational convenience and with no loss of generality that

$$\Delta^{(i)}\phi\left(\boldsymbol{I}_{\boldsymbol{f}(\boldsymbol{\Theta},\boldsymbol{Z})>\boldsymbol{0}},\boldsymbol{\Theta},\boldsymbol{Z}\right)=0 \qquad \forall i<c,$$

so that the indicator functions before c represent the discontinuities in the gradient of the payoff, while the indicator functions from c onwards represent the discontinuities of the payoff itself, if any.

The cited paper tells us that the first order sensitivity is

$$\frac{\partial\varepsilon}{\partial\boldsymbol{\Theta}}=\boldsymbol{d}_0+\sum_{i=c}^{h}\boldsymbol{d}_i,$$

where

$$\begin{aligned}
\boldsymbol{d}_0&=\mathbb{E}\left[\frac{\partial\phi}{\partial\boldsymbol{\Theta}}\left(\boldsymbol{I}_{\boldsymbol{f}(\theta,\boldsymbol{Z})>\boldsymbol{0}},\boldsymbol{\Theta},\boldsymbol{Z}\right)\right],\\
\boldsymbol{d}_i&=\mathbb{E}\left[\sum_{\zeta\in A^i_{k(i)}(\theta,\boldsymbol{Z}_{-k(i)})}\left\{\frac{e^{-\zeta^2/2}}{\sqrt{2\pi}}\left[\left|\frac{\partial f_i}{\partial z_{k(i)}}\right|^{-1}\frac{\partial f_i}{\partial\boldsymbol{\Theta}}\right](\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta))\right.\right.\\
&\qquad\left.\left.\times\,\Delta^{(i)}\phi\left(\boldsymbol{I}_{\boldsymbol{f}(\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta))>\boldsymbol{0}},\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta)\right)\right\}\right].
\end{aligned}$$

Now we have to differentiate $\boldsymbol{d}_0$ and $\boldsymbol{d}_i$. The first task is a direct application of the first order result to $\nabla\phi$:

$$\frac{\partial\boldsymbol{d}_0}{\partial\boldsymbol{\Theta}}=\mathbf{H}_{00}+\sum_{i=1}^{h}\mathbf{H}_{0i},$$

where

$$\begin{aligned}
\mathbf{H}_{00}&=\mathbb{E}\left[\frac{\partial^2\phi}{\partial\boldsymbol{\Theta}^2}\left(\boldsymbol{I}_{\boldsymbol{f}(\theta,\boldsymbol{Z}(Z_{k(i)}=\zeta))>\boldsymbol{0}},\boldsymbol{\Theta},\boldsymbol{Z}\right)\right],\\
\mathbf{H}_{0i}&=\mathbb{E}\left[\sum_{\zeta\in A^i_{k(i)}(\theta,\boldsymbol{Z}_{-k(i)})}\left\{\Delta^{(i)}\left(\frac{\partial\phi}{\partial\boldsymbol{\Theta}}\right)^{\intercal}\left(\boldsymbol{I}_{\boldsymbol{f}(\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta))>\boldsymbol{0}},\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta)\right)\right.\right.\\
&\qquad\left.\left.\times\,\frac{e^{-\zeta^2/2}}{\sqrt{2\pi}}\left[\left|\frac{\partial f_i}{\partial z_{k(i)}}\right|^{-1}\frac{\partial f_i}{\partial\boldsymbol{\Theta}}\right](\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta))\right\}\right].
\end{aligned}$$

If the payoff is continuous ($c>h$), we are done; otherwise, we need the derivative of $\boldsymbol{d}_i$, which is slightly trickier. In order to compute it, we note that by the implicit function theorem, in a neighbourhood of $\boldsymbol{\Theta}$, the zeros ζ in the definition of each $\boldsymbol{d}_i$ can be expressed as functions $\zeta_i^{(l)}$ of $\boldsymbol{Z}_{-k(i)}$ and $\boldsymbol{\Theta}$ with gradient[6]

$$-\left[\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\left(\frac{\partial f_i}{\partial\left(\boldsymbol{\Theta},\boldsymbol{z}_{-k(i)}\right)}\right)\right](\boldsymbol{\Theta},\boldsymbol{Z}(Z_{k(i)}=\zeta)).$$

[6]In fact the neighbourhood may depend on $\boldsymbol{Z}$; we neglect this kind of technicalities, which should be addressable along the lines of the Appendix of Daluiso and Facchinetti (2018).

Now each summand in the definition of $\boldsymbol{d}_i$ can be differentiated with the DAAD method, provided that the composition of f_j with $\zeta_i^{(l)}$ satisfies Hypothesis 4.1 for some coordinate $h(i,j) \neq k(i)$. We define

$$A^{i,j}(\boldsymbol{\Theta}, \boldsymbol{Z}) := \left\{(\zeta,\eta) : (f_i, f_j)\left(\boldsymbol{\Theta}, \boldsymbol{Z}(Z_{k(i)} = \zeta, Z_{h(i,j)} = \eta)\right) = (0,0)\right\}.$$

We are ready to state the result:

$$\frac{\partial \boldsymbol{d}_i}{\partial \boldsymbol{\Theta}} = \mathbf{H}_{i0} + \sum_{j=c}^{h} \mathbf{H}_{ij},$$

where

$$\mathbf{H}_{i0} = \mathbb{E}\left\{ \sum_{\zeta \in A^i_{k(i)}(\theta, \boldsymbol{Z}_{-k(i)})} \frac{\partial}{\partial(\boldsymbol{\Theta}, z_{k(i)})}\bigg|_{z_{k(i)}=\zeta} \left[\frac{e^{-\zeta^2/2}}{\sqrt{2\pi}} \left|\frac{\partial f_i}{\partial z_{k(i)}}\right|^{-1} \left(\frac{\partial f_i}{\partial \boldsymbol{\Theta}}\right)^{\intercal} \Delta^{(i)}\phi \right] \right.$$
$$\left. \times \begin{pmatrix} \mathbf{I}_{N_\theta} \\ -\left[\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\left(\frac{\partial f_i}{\partial \boldsymbol{\Theta}}\right)\right](\boldsymbol{\Theta}, \boldsymbol{Z}(Z_{k(i)} = \zeta)) \end{pmatrix} \right\},$$

$$\mathbf{H}_{ij} = \mathbb{E}\left\{ \sum_{(\zeta,\eta)\in A^{i,j}(\boldsymbol{\Theta},\boldsymbol{Z})} \left[\frac{e^{-\frac{\zeta^2+\eta^2}{2}}}{2\pi} \left|\frac{\partial f_i}{\partial z_{k(i)}}\right|^{-1} \left(\frac{\partial f_i}{\partial \boldsymbol{\Theta}}\right)^{\intercal} \left| \frac{\partial f_j}{\partial z_{h(i,j)}} - \frac{\partial f_j}{\partial z_{k(i)}}\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\frac{\partial f_i}{\partial z_{h(i,j)}}\right|^{-1} \right.\right.$$
$$\left.\left. \left(\frac{\partial f_j}{\partial \boldsymbol{\Theta}} - \frac{\partial f_j}{\partial z_{k(i)}}\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\frac{\partial f_i}{\partial \boldsymbol{\Theta}}\right) \Delta^{(j)}\Delta^{(i)}\phi\left(\boldsymbol{I}_{\boldsymbol{f}>\mathbf{0}}, \boldsymbol{\Theta}, \boldsymbol{Z}\right) \right]_{Z_{k(i)}=\zeta, Z_{h(i,j)}=\eta} \right\}.$$

Note that the expression in the second absolute value should be different from zero, or to say it differently, $\partial(f_i, f_j)/\partial(z_{k(i)}, z_{h(i,j)})$ should be full rank.

All the gradients appearing in these formulas are readily computed by algorithmic differentiation; the only practical concern is the fact that there are (up to) $(h-c+1)^2$ terms $\mathbf{H}_{ij}$. In fact in most applications, f_i and f_j depend on non-overlapping sets of fixing times, so that

$$\left| \frac{\partial f_j}{\partial z_{h(i,j)}} - \frac{\partial f_j}{\partial z_{k(i)}}\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\frac{\partial f_i}{\partial z_{h(i,j)}}\right|^{-1} \left(\frac{\partial f_j}{\partial \boldsymbol{\Theta}} - \frac{\partial f_j}{\partial z_{k(i)}}\left(\frac{\partial f_i}{\partial z_{k(i)}}\right)^{-1}\frac{\partial f_i}{\partial \boldsymbol{\Theta}}\right)$$
$$= \left|\frac{\partial f_j}{\partial z_{k(j)}}\right|^{-1}\left(\frac{\partial f_j}{\partial \boldsymbol{\Theta}}\right),$$

which does not depend on i any more. As a consequence, the only portion of the algorithm which truly grows quadratically with the number of payoff discontinuities is $\Delta^{(j)}\Delta^{(i)}\phi$. This should not be dramatic in the economy of a Monte Carlo pricing exercise inclusive of a costly Euler simulation, unless there is a really large number of digital features as in frequently monitored barrier options, which anyway are often already handled by regularizing the payoff accounting for touch probabilities (see e.g. Glasserman, 2004).

For ease of reference, we collect as a conclusion the full set of addends contributing to the estimator of the Hessian:

$$\frac{\partial^2 \varepsilon}{\partial \boldsymbol{\Theta}^2} = \mathbf{H}_{00} + \sum_{i=1}^{h} \mathbf{H}_{0i} + \sum_{i=c}^{h} \mathbf{H}_{i0} + \sum_{i,j=c}^{h} \mathbf{H}_{ij}.$$

To conclude this section, we detail here below for the sake of clarity the algorithm under the hypothesis that only the gradient is discontinuous. Recall that f_i, $g_{\boldsymbol{a}}$ are the composition of $\tilde{f}_i$, $\tilde{g}_{\boldsymbol{a}}$ defined in (4.2) with the conditional simulation of the $\boldsymbol{X}_{M_i}$ given $\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}$, well approximated by Brownian bridges.

Algorithm 4.2. [DAAD2] On each Monte Carlo scenario:

1. Define $\boldsymbol{a}_0 = \boldsymbol{I}_{\tilde{\boldsymbol{f}}(\boldsymbol{X}_{\boldsymbol{T}}, \boldsymbol{\theta}) > \boldsymbol{0}}$, and apply pathwise adjoint differentiation to $g_{\boldsymbol{a}_0}$, getting estimators $\bar{\boldsymbol{\theta}}$, $\bar{\boldsymbol{X}}_{\widetilde{\boldsymbol{T}}}$.

2. Differentiate again the previous step ignoring discontinuities, getting a matrix $\mathbf{D}_0$ representing the smooth part of the $\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}$-conditional sample estimator for the Hessian of ε.

3. For each $i = 1, \ldots, h$ compute the values ω which substituted into $W^{(i)}_{k(i)}$ make a $\tilde{\boldsymbol{W}}^{(i)}$ such that f_i equals zero, and the corresponding perturbed version of $\boldsymbol{X}_{\boldsymbol{T}}$, which we denote by $\tilde{\boldsymbol{X}}_{\boldsymbol{T}}$. Then for each such ω:

   (a) There should be exactly two $\boldsymbol{a} \in \{-1, +1\}^h$ such that $\tilde{\boldsymbol{X}}_{\boldsymbol{T}} \in \mathcal{X}_{\boldsymbol{a}}$, corresponding to $a_i = \pm 1$: call them $\boldsymbol{a}_\pm$.

   (b) Use AAD to compute in the perturbed scenario the gradients

   $$\frac{\partial f_i}{\partial \left(\boldsymbol{W}^{(i)}, \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)}, \frac{\partial g_{\boldsymbol{a}_\pm}}{\partial \left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)}.$$

   (c) Increment the discontinuity's contribution to $\dfrac{\partial^2 \varepsilon}{\partial \left(\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta}\right)^2}$ as follows:

   $$\mathbf{D}_i \mathrel{+}= \frac{e^{-\frac{\omega^2}{2}}}{\sqrt{2\pi}} \left| \frac{\partial f_i}{\partial W^{(i)}_{k(i)}} \right|^{-1} \left(\frac{\partial f_i}{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})} \right)^{\intercal} \left[\frac{\partial g_{\boldsymbol{a}_+}}{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})} - \frac{\partial g_{\boldsymbol{a}_-}}{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})} \right]. \quad (4.3)$$

4. Compute the Jacobian $\dfrac{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}{\partial \boldsymbol{\psi}}$, which is the only unknown part of $\dfrac{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})}{\partial \boldsymbol{\psi}}$.

5. Compute by adjoint differentiation $\bar{\boldsymbol{X}}_{\widetilde{\boldsymbol{T}}} \cdot \dfrac{\partial \boldsymbol{X}_{\widetilde{\boldsymbol{T}}}}{\partial \boldsymbol{\psi}}$, then differentiate it again with respect to $\boldsymbol{\psi}$ getting a matrix $\bar{\bar{\boldsymbol{\psi}}}$.

6. The final estimator is $\bar{\bar{\psi}} + \left(\dfrac{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})}{\partial \boldsymbol{\psi}} \right)^{\intercal} \left(\sum_{i=1}^{h} \mathbf{D}_i \right) \left(\dfrac{\partial (\boldsymbol{X}_{\widetilde{\boldsymbol{T}}}, \boldsymbol{\theta})}{\partial \boldsymbol{\psi}} \right)$.

As for complexity, the multiplier on the cost of simulation comes from step 5, with a factor of order $O(N)$, plus step 4, with a factor either of order $O(h \cdot N_X)$ if computed by adjoint differentiation, or of order $O(N)$ if computed by forward differentiation. The cost of steps 1 through 3 has to do with the number of discontinuities h, but is also proportional to the computational time of f_i and g, which is usually much less relevant than the time required to perform the Euler simulation.

4.3 Functional Gamma

The idea of this section is that if $N_X \leq N_W$, the Feynman-Kac theorem can supply enough equations to determine the Gamma matrix $\mathbf{\Gamma} = \partial^2 P / \partial \boldsymbol{X}_0^2$ given the time decay of suitable first order sensitivities, whose computation is fast ($O(T)$) thanks to adjoint differentiation. This fact is essentially a consequence of the multidimensional version of a known relationship between Gamma and "functional Vega", which is exploited for instance in Reghai et al. (2015) to compute pricing valuation adjustments.

Indeed, it is well known that the price P, as a function of $\boldsymbol{X}_0$ and of the evaluation time, satisfies the PDE

$$\frac{\partial P}{\partial t} + \frac{\partial P}{\partial x} \cdot \boldsymbol{\mu} + \frac{1}{2} \mathrm{tr} \left[\frac{\partial^2 P}{\partial x^2} \mathbf{\Sigma} \mathbf{\Sigma}^\intercal \right] - Pr + \sum C_i \delta(t - t_i) = 0,$$

where r is the risk-free rate and C_i is the cashflow paid at time t_i. Now one can note that if P' is the price for a different diffusion coefficient function $\mathbf{\Sigma}'$, then $P' - P$ satisfies the same equation except that the discrete last addend is substituted by a continuous source term

$$\frac{1}{2} \mathrm{tr} \left[\frac{\partial^2 P}{\partial x^2} \left(\mathbf{\Sigma}' \mathbf{\Sigma}'^\intercal - \mathbf{\Sigma} \mathbf{\Sigma}^\intercal \right) \right],$$

so that by Feynman-Kac we can conclude that

$$P'(x,t) = P(x,t) + \frac{1}{2} \mathbb{E} \left\{ \int_t^T e^{-\int_t^s r(\boldsymbol{X}_u, u) \mathrm{d}u} \, \mathrm{tr} \left[\frac{\partial^2 P}{\partial x^2} \left(\mathbf{\Sigma}' \mathbf{\Sigma}'^\intercal - \mathbf{\Sigma} \mathbf{\Sigma}^\intercal \right) \right] (\boldsymbol{X}_s, s) \, \mathrm{d}s \right\}.$$

Now we fix a perturbation matrix $\mathbf{K}$ and time $\tau > 0$ and choose

$$\mathbf{\Sigma}'(t,x) = \mathbf{\Sigma}(t,x) + \mathbf{K} I_{t<\tau};$$

differentiation with respect to $\mathbf{K}$ gives

$$\frac{\partial P'(x,0)}{\partial \mathbf{K}} = \int_0^\tau \mathbb{E} \left\{ e^{-\int_0^s r(\boldsymbol{X}_u, u) \mathrm{d}u} \left[\mathbf{\Sigma}^\intercal \frac{\partial^2 P}{\partial x^2} \right] (\boldsymbol{X}_s, s) \right\} \mathrm{d}s;$$

finally, differentiation in $\tau = 0$ gives

$$\frac{\partial^2 P'(x,0)}{\partial \tau \partial \mathbf{K}} = \mathbf{\Sigma}^\intercal (\boldsymbol{X}_0, 0) \, \mathbf{\Gamma}, \tag{4.4}$$

which for full rank $\mathbf{\Sigma}(\boldsymbol{X}_0, 0)$ allows for the determination of $\mathbf{\Gamma}$ given $\partial^2 P'(x,0)/(\partial \tau \partial \mathbf{K})$; we expect the computation of the latter to be faster because the second order differentiation is with respect to a scalar τ.

To compute $\partial^2 P'(x,0)/(\partial\tau\partial\mathbf{K})$, we note that for small τ, the payoff depends on $\mathbf{K}$ only through the first simulated point $\boldsymbol{X}_{S_1}$. This makes the following decomposition possible:

$$\frac{\partial P'}{\partial \mathbf{K}} = \mathbb{E}\left[\frac{\partial P(\boldsymbol{X}_{S_1}, S_1)}{\partial \boldsymbol{X}_{S_1}} \cdot \frac{\partial \boldsymbol{X}_{S_1}}{\partial \mathbf{K}}\right], \tag{4.5}$$

where we suppose that the price P at time S_1 is a smooth function of $\boldsymbol{X}_{S_1}$: this is true unless the payoff has a discontinuity maturing at time S_1, in which case one can add a point to the simulation grid $\boldsymbol{S}$.

If we substitute the original SDE with its Euler discretization, in which drift and diffusion coefficients are constant on time intervals $[S_i, S_{i+1})$, we get:[7]

$$\begin{aligned}\boldsymbol{X}_{S_1} &= (\boldsymbol{\Sigma}(\boldsymbol{X}_0, 0) + \mathbf{K})\,\boldsymbol{W}_\tau + \boldsymbol{\Sigma}(\boldsymbol{X}_0, 0)\,(\boldsymbol{W}_{S_1} - \boldsymbol{W}_\tau) = \boldsymbol{\Sigma}(\boldsymbol{X}_0, 0)\boldsymbol{W}_{S_1} + \mathbf{K}\boldsymbol{W}_\tau \\ &= \boldsymbol{\Sigma}(\boldsymbol{X}_0, 0)\boldsymbol{W}_{S_1} + \mathbf{K}\left(S_1^{-1}\tau \boldsymbol{W}_{S_1} + \sqrt{S_1^{-1}(S_1 - \tau)\tau}\,\tilde{\boldsymbol{Z}}\right)\end{aligned} \tag{4.6}$$

for $\tilde{\boldsymbol{Z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{N_W})$ independent from $\boldsymbol{W}_{\boldsymbol{S}}$, as prescribed by the Brownian bridge distribution of $\boldsymbol{W}_\tau$ given $\boldsymbol{W}_{\boldsymbol{S}}$ (see e.g. Karatzas and Shreve, 1991). This means that we can rewrite (4.5) as

$$\frac{\partial P'}{\partial \mathbf{K}} = \mathbb{E}\left[\boldsymbol{W}_\tau \cdot \nabla P(\boldsymbol{X}_{S_1}, S_1)^\intercal\right] = \mathbb{E}\left[\left(\frac{\tau}{S_1}\boldsymbol{W}_{S_1} + \sqrt{\frac{(S_1 - \tau)\tau}{S_1}}\,\tilde{\boldsymbol{Z}}\right) \cdot \nabla P(\boldsymbol{X}_{S_1}, S_1)^\intercal\right].$$

This expression is not yet ready for ordinary differentiation with respect to τ because of the $\sqrt{\tau}$ dependence in the second addend. However, since we are only concerned with the value of the right hand side in $\mathbf{K} = \mathbf{0}$, in which point $\boldsymbol{X}_{S_1}$ is a function of $\boldsymbol{W}_{S_1}$ but not of $\tilde{\boldsymbol{Z}}$, we can use independence to deduce

$$\mathbb{E}\left[\sqrt{\frac{(S_1 - \tau)\tau}{S_1}}\,\tilde{\boldsymbol{Z}} \cdot \nabla P(\boldsymbol{X}_{S_1}, S_1)^\intercal\right] = \mathbb{E}\left[\sqrt{\frac{(S_1 - \tau)\tau}{S_1}}\,\tilde{\boldsymbol{Z}}\right] \cdot \mathbb{E}\left[\nabla P(\boldsymbol{X}_{S_1}, S_1)^\intercal\right] = \mathbf{0}.$$

To sum up, to apply (4.4), we have to estimate simply

$$\frac{\partial}{\partial \tau}\mathbb{E}\left[\frac{\tau}{S_1}\boldsymbol{W}_{S_1} \cdot \nabla P(\boldsymbol{X}_{S_1}, S_1)^\intercal\right] = \mathbb{E}\left[\frac{\boldsymbol{W}_{S_1}}{S_1} \cdot \bar{\boldsymbol{X}}_{S_1}\right] \tag{4.7}$$

where in the second equality we have substituted the Delta sensitivity ∇P with any estimator thereof.

The final equation (4.7) closely resembles that of likelihood ratio based estimators like LRMAAD, while being both simpler to implement since it does not involve second order differentiation of the drift and diffusion coefficients, and more flexible since it does not prescribe which first order estimator $\bar{\boldsymbol{X}}_{S_1}$ to use. This also means that it shares the main drawback of likelihood ratio weights, i.e. the reciprocal dependence on the small discretization step S_1, which may impact negatively the

[7] This way we differentiate the price *as approximated by the pricing engine*; this should make little difference in practice, and is anyway arguably even more useful than the differentiation of the abstract price which would arise from the theoretical exact solution of the SDE. Note that the argument in fact needs this approximation only in the first time interval $[0, S_1)$.

Monte Carlo variance. However, as is the case in vibrato, a simple antithetic trick can save the day, since the resulting estimator becomes

$$\boldsymbol{\Sigma}^{\intercal}\left(\boldsymbol{X}_0,0\right)^{-1}\frac{\boldsymbol{W}_{S_1}}{2S_1}\cdot\left[\bar{\boldsymbol{X}}_{S_1}(\boldsymbol{W}_{S_1})-\bar{\boldsymbol{X}}_{S_1}(-\boldsymbol{W}_{S_1})\right], \tag{4.8}$$

where, roughly speaking, we expect

$$\boldsymbol{W}_{S_1}=O(\sqrt{S_1})\quad\text{and}\quad\bar{\boldsymbol{X}}_{S_1}(\boldsymbol{W}_{S_1})-\bar{\boldsymbol{X}}_{S_1}(-\boldsymbol{W}_{S_1})=O(\boldsymbol{W}_{S_1})=O(\sqrt{S_1}),$$

jointly counterbalancing the S_1^{-1} factor.

We stress that for maximum effectiveness of this variant, one should change the sign only of $\boldsymbol{W}_{S_1}$, unlike in standard antithetic Monte Carlo which inverts the full Brownian path. This does not prevent the usage of ordinary antithetic trajectories on top of the final estimator (4.8), as one would do for any other method.

5 Empirical results

In this section, we will compare empirically the above algorithms on several test cases. We will always suppose that the payoff is written on one or more underlying assets A_i with the simplest Black-Scholes dynamics

$$A_i=\exp(X_i),\qquad \mathrm{d}X_i=-\frac{\sigma_i^2}{2}\mathrm{d}t+\sigma_i\mathrm{d}B_i,\qquad \mathrm{d}\langle B_i,B_j\rangle=\rho_{ij}\mathrm{d}t, \tag{5.1}$$

where B_i are correlated Brownian motions; of course, in this stylised case Euler simulation is not strictly necessary since finite-step transition probabilities are available in closed form, but we neglect this simplification which would not apply to general diffusive models. Note that (5.1) is readily expressed in the form (2.1), which formally prescribes independent Brownian motions, by a Choleski factorization of the instantaneous correlation matrix $\mathbf{R}=(\rho_{ij})_{i,j\le N_W}$:

$$\begin{gathered}\mathrm{d}\boldsymbol{X}=\boldsymbol{\mu}\mathrm{d}t+\boldsymbol{\Sigma}\mathrm{d}\boldsymbol{W},\\ \boldsymbol{\mu}:=\left(-\sigma_i^2/2\right)_{i\le N_W}\quad\text{and}\qquad\boldsymbol{\Sigma}:=\operatorname{diag}\left(\sigma_i\right)_{i\le N_W}\cdot\operatorname{Choleski}(\mathbf{R}).\end{gathered}$$

The following names will be used to denote subsets of the full Hessian $\mathbf{H}$:

$$\text{Gamma}=\left(\frac{\partial^2P}{\partial X_i(0)\partial X_j(0)}\right)_{i,j},\ \text{Vanna}=\left(\frac{\partial^2P}{\partial X_i(0)\partial\sigma_j}\right)_{i,j},\text{Volga}=\left(\frac{\partial^2P}{\partial\sigma_i\partial\sigma_j}\right)_{i,j};$$

this is slight abuse of terminology, since to adhere to common jargon one should substitute in the definitions $\boldsymbol{X}$ with $\boldsymbol{A}$, but in our numerical analysis we prefer to display the sensitivity with respect to the simulated driver $\boldsymbol{X}$, because for most estimators this is more natural. One can always obtain the sensitivity to $\boldsymbol{A}$ plugging the result into the elementary relations

$$\begin{aligned}\frac{\partial^2P}{\partial A_i(0)\partial A_j(0)}&=\frac{1}{A_i(0)A_j(0)}\frac{\partial^2P}{\partial X_i(0)\partial X_j(0)}-\delta_{ij}\frac{1}{A_i(0)^2}\frac{\partial P}{\partial X_i(0)},\\ \frac{\partial^2P}{\partial A_i(0)\partial\sigma_j}&=\frac{1}{A_i(0)}\frac{\partial^2P}{\partial X_i(0)\partial\sigma_j}.\end{aligned}$$

Vanna will sometimes be referred to by the phrases "Delta of Vega" of "Vega of Delta" to stress the order of differentiation: in the former case we mean that derivatives are first taken with respect to volatility, while in the latter case that they are first taken with respect to $\boldsymbol{X}_0$.

In the sequel, all algorithms will be denoted by acronyms: we point to Tables 1 and 2 for easy reference. Note that FΓ can be based on any first order estimator, hence we will use the notations FΓ-PW, FΓ-VB, FΓ-OPP, FΓ-DAAD to indicate the functional Gamma method of Section 4.3 where the first order estimator used to compute $\bar{\boldsymbol{X}}_{S_1}$ is respectively given by pathwise differentiation, vibrato Monte Carlo, Optimal Partial Proxy and DAAD.

All the test payouts will be autonomous in the sense of (2.4), so that all methods described above apply; only when the payoff is discontinuous, some of them will be unavoidably excluded. In fact in those cases one might smooth the payoff, not only to enable otherwise unusable estimators, but also to try and improve the performance of other ones. We neglect this possibility since smoothing techniques would involve a careful choice of their parameters to correctly balance the variance gain with the bias introduced, which is problematic already for first order estimators: see the analysis in Daluiso and Facchinetti (2018).

In fact, also the choice of the displacement for finite difference based algorithms (FDIFF2, FDPW, VFD, FDDAAD) may be a serious concern. We will use for $\Delta X_i(0)$ and $\Delta\sigma_i$ the largest power of 10 such that the confidence interval of the finite difference estimator contains the analytical result when available, or does not appear to diverge significantly from the confidence intervals of unbiased methods otherwise. This is a bit of cheating, and is only justified because we are just looking for benchmarks to the analytical methods which are the main subject of our analysis; however, if one would actually choose a finite difference method as his only sensitivity calculation engine, such comparison terms would not be available, and much attention should be paid to accurate heuristics for the choice of a good finite shift.

5.1 Single-asset payoffs: stability of the estimators

Before analysing settings in which the high number of sensitivities to compute is a major driver in the choice of the algorithm, we deem it useful to explore the behaviour of the several estimators in the simpler task of computing the Gamma sensitivity with respect to a single underlying asset. This way, we can for a moment forget about the computational burden, which is comparable and affordable for all methods, and concentrate on more intrinsic and implementation-independent properties, among which we put the focus on the following:

- Variance: how large is the confidence interval for a fixed number of Monte Carlo paths? We will analyse the standard deviation of the result for 10^5 simulated paths, which by the central limit theorem should be proportional to the width of a centred confidence interval for any confidence level desired.

- Stability: does the result change smoothly while moving the initial condition S_0? This is a highly desirable property in practice, since one does not want the Greeks to jump wildly as the market evolves.

Method	$T = 1d$	$T = 1y$
FDIFF2	48.562839	3.604733
FDPW	41.621283	3.097918
HOPP	47.695944	3.719352
LRMAAD	104.264491	12.588643
AADLRM	104.264491	12.588643
VAD	23.330228	1.881182
DAAD2	19.015445	1.552514
FΓ-PW	53.021142	6.652639
FΓ-VB	65.882049	8.471453

Table 3: Gamma sensitivity of a vanilla call: standard deviation for 10^5 Monte Carlo paths, averaged over 40 values of A_0.

In all experiments, the volatility parameter is set to $\sigma = 0.2$ in yearly time units.

5.1.1 Call option

As a prototypical example of a continuous payoff we take a vanilla call option $(A_T - K)^+$ with strike $K = 100$, and vary the initial condition A_0 and the maturity time T. In particular, we take both a moderate maturity T of one year, divided for the purpose of Euler simulation into 100 equal time intervals; and a short maturity $T = 1$ day, with hourly steps in the Euler simulation. The analytical price is well known and will serve as a check.

First of all, we compare the Monte Carlo uncertainties in Table 3. We make two main remarks:

- Likelihood ratio based methods have by far the largest confidence intervals, followed by functional Gamma.
- For this simple payoff, only DAAD2 and VAD offer a material improvement over naive finite differences.

As far as smooth dependence on A_0 is concerned, we first consider the case in which T is one year. Figure 1 plots the complete results: all methods appear to work reasonably well, but those based on likelihood ratio or functional Gamma display larger errors, because of the larger confidence intervals. The smoothest analytical methods are HOPP and DAAD2; also VAD performs well, while FDPW is quite more erratic. However, one should again note that for this simple payoff, elementary second order finite differences FDIFF2, here with displacement $\Delta X_0 = 0.01$, are not that bad.

We tried to stress the exercise considering an option expiring in $T = 1$ day: the results are in Figure 2. One notable difference is that FDIFF2 ($\Delta X_0 = 0.001$) and HOPP lose some smoothness, while DAAD2 still follows the shape of the exact solution remarkably well. Moreover, FΓ-VB becomes the most stable alternative, even though it is still significantly more noisy than most of its competitors, as we saw in Table 3. This smoothness is not observed in FΓ-PW, showing that the flexibility

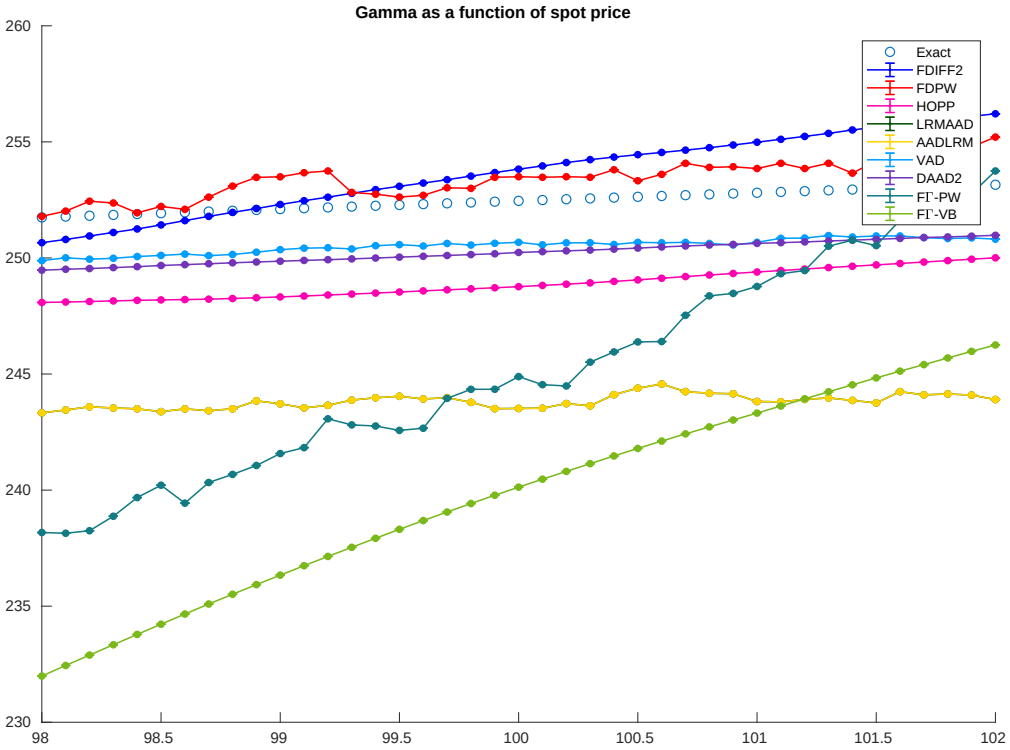

Figure 1: Gamma sensitivity of a vanilla call with maturity $T = 1y$ and strike $K = 100$ as a function of the spot price A_0: all estimators, 10^5 paths.

in the choice of the underlying first order estimator which functional Gamma offers over traditional likelihood ratio methods is a valuable asset.

5.1.2 Digital option

The archetypal example of a discontinuous payoff is the digital option $I_{A_T>K}$, for which we repeat the analyses of the previous subsection. Note that several methods cannot be used any more.

The detail of Monte Carlo uncertainties is in Table 4. The winners are FΓ-DAAD, DAAD2 and FΓ-VB, followed by the biased VFD; all methods are much better than FDIFF2, so we exclude it from the graphs of this subsection.

Figure 3 shows that for $T = 1$ year, DAAD based methods (DAAD2, FDDAAD, FΓ-DAAD) display the smoothest dependence on A_0. The same remarks apply to $T = 1$ day, which was plotted in Figure 4, where the analytical exact Gamma has been subtracted from the estimated value for a better display cleaned from the trend.

5.2 Multi-asset payoffs: efficiency of the algorithms

We now move to multi-asset payoffs, where the full Hessian $\mathbf{H}$ has a high number of entries. This imposes a rethinking of how the various estimators are compared, because a more noisy one may still be preferable if it is significantly faster. We believe that the most objective metric is therefore the estimated run time which

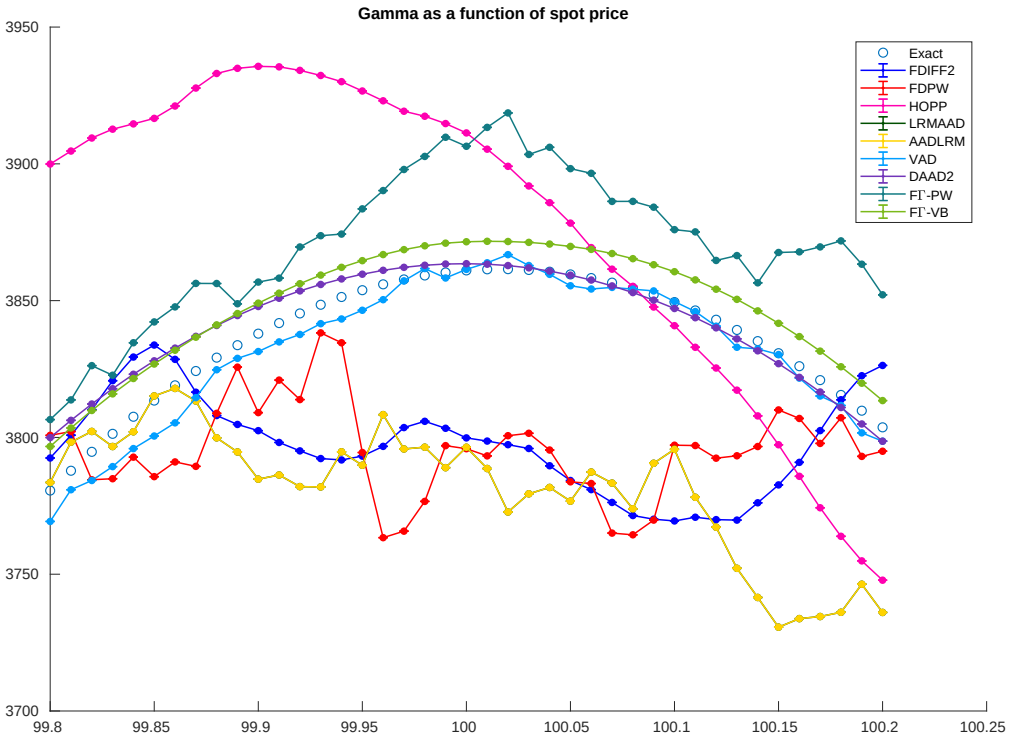

Figure 2: Gamma sensitivity of a vanilla call with maturity $T = 1d$ and strike $K = 100$ as a function of the spot price A_0: all estimators, 10^5 paths.

Method	$T = 1d$	$T = 1y$
FDIFF2	865.468515	6.303787
HOPP	198.053629	1.672700
VFD	88.642863	0.699658
FDDAAD	134.068248	1.095060
DAAD2	73.659168	0.587246
FΓ-VB	75.564853	0.652947
FΓ-OPP	123.806197	1.030655
FΓ-DAAD	61.683419	0.547219

Table 4: Gamma sensitivity of a digital call: standard deviation for 10^5 Monte Carlo paths, averaged over 40 values of A_0.

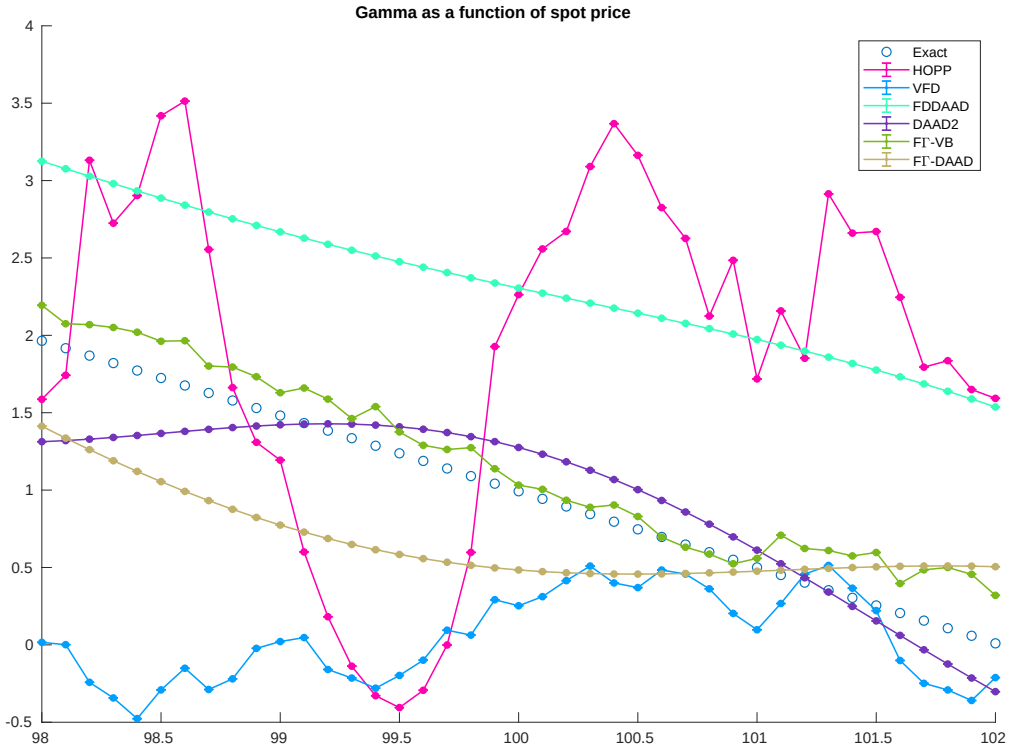

Figure 3: Gamma sensitivity of a digital call with maturity $T = 1y$ and strike $K = 100$ as a function of the spot price A_0: low-variance estimators, 10^5 paths.

would be needed to achieve a unit Monte Carlo variance. This is easily computed as

$$(\text{run time}) \times (\text{estimated standard deviation})^2, \tag{5.2}$$

since it is well known that to reduce the uncertainty by a factor of two one should increase the number of simulated paths by a factor of four.

In all examples, all assets have spot value $A_i(0) = 100$ and volatility parameter $\sigma_i = 0.2$, while the instantaneous correlation between couples of Brownian drivers is set to a flat value $\rho_{ij} = 0.5$. Each method was tested on a Monte Carlo simulation with 10^4 paths.

We computed the chosen metric (5.2) separately for each second order sensitivity in the Hessian matrix, because while on the one hand the run time is a common number, on the other hand the standard deviation is different entry by entry. Then, to get synthetic tables, for each Greek we produced two averages of the resulting metrics: one comprising those sensitivities in which both differentiation variables refer to the same asset (i.e. the diagonal entries), and another one comprising the remaining "cross" sensitivities.

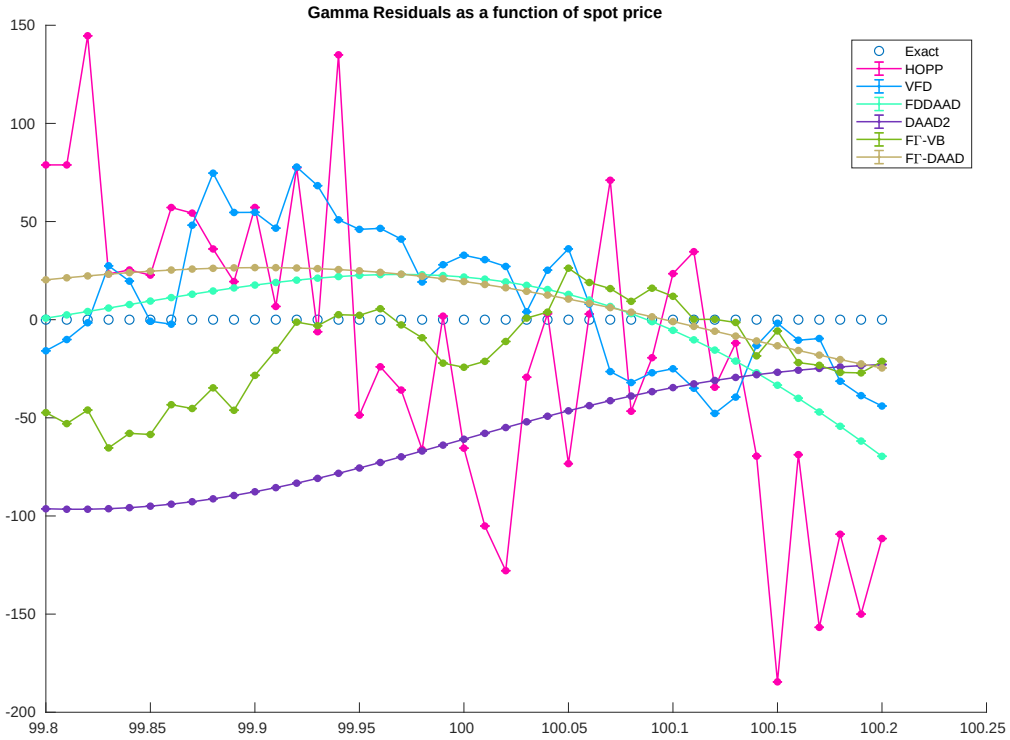

Figure 4: Gamma sensitivity of a digital call with maturity $T = 1d$ and strike $K = 100$ as a function of the spot price A_0: residual of low-variance estimators with respect to the analytical value, 10^5 paths.

Method	Gamma		Volga	
	diagonal	cross	diagonal	cross
FDIFF2	120.91	57.70	57.49	10.24
FDPW	31.80	30.23	16.49	5.66
HOPP	25.87	23.19	20.52	5.00
LRMAAD	1904.96	1909.96	3467.84	3392.80
AADLRM	5365.08	5379.16	210343.67	9555.40
VAD	16.11	15.59	173.42	10.64
DAAD2	6.56	5.80	4.37	1.01

Table 5: Gamma and Volga sensitivities of a call with maturity $T = 1y$ and strike $K = 100$ on a basket of 8 underlyings: average time per unit Monte Carlo variance, all methods.

Method	Delta of Vega		Vega of Delta	
	diagonal	cross	diagonal	cross
FDIFF2	30.40	24.38	30.40	24.38
FDPW	14.06	12.27	16.77	14.44
HOPP	13.69	10.37	13.69	10.37
LRMAAD	2124.18	2165.37	3003.85	2997.39
AADLRM	8459.99	8441.78	133381.34	6098.50
VAD	8.14	7.44	147.59	13.74
DAAD2	3.36	2.41	3.36	2.41

Table 6: Vanna sensitivities of a call with maturity $T = 1y$ and strike $K = 100$ on a basket of 8 underlyings: average time per unit Monte Carlo variance, all methods.

5.2.1 Basket call

The multidimensional generalization of the first example of Section 5.1 is a call on the average of several assets:

$$\left(\frac{1}{N_X} \sum_{i=1}^{N_X} A_i(T) - K \right)^+ ,$$

which we will price for $N_X = 8$, $T = 1$ year and $K = 100$.

The results are in Tables 5 and 6. First of all, we note that likelihood ratio methods are more than one order of magnitude slower than all other algorithms. In particular, finite differences should be preferred to likelihood ratio regardless of their higher cost per path; while all other estimators outperform FDIFF2. DAAD2 is consistently much more effective than its competitors, while the second-best depends on the sensitivity of interest: it is VAD for Gamma, but becomes FDPW for Volga, VAD for Delta of Vega, HOPP for Vega of Delta (although Delta of Vega and Vega of Delta estimate theoretically the same number by the Schwarz theorem, see the last paragraph of this subsection).

If one wants to compute only Gamma, then functional methods are also available; moreover, LRMAAD might in principle become competitive thanks to its

Method	Gamma		
	diagonal	cross	cross (symm)
FDIFF2	29.64	14.14	13.96
FDPW	15.16	14.41	14.01
HOPP	12.87	11.54	11.80
LRMAAD	164.90	165.33	72.24
VAD	7.97	7.71	3.73
DAAD2	3.40	3.01	3.23
FΓ-PW	71.14	70.83	33.66

Table 7: Gamma sensitivities of a call with maturity $T = 1y$ and strike $K = 100$ on a basket of 8 underlyings: average time per unit Monte Carlo variance on and off the diagonal, with and without symmetrization, all methods.

efficient implementation described in Section 3.2. For a fair comparison we updated also the run times of the other methods, which are smaller if sensitivities to σ_i are not required. Our empirical findings in this setting are in the first two columns of Table 7, and are not encouraging: LRMAAD is still far too noisy even considering that it is much faster than linear-cost methods (not to speak of FDIFF2), and FΓ, while significantly better, does not improve the variance sufficiently. We remark that now that the number of required sensitivities is "only" 64, beating the dumb FDIFF2 benchmark is harder then one might expect, and off the diagonal only DAAD2 is a neat improvement.

Before resigning, we note that most of the analytical estimators we test produce asymmetric matrices, although we know that Hessians are symmetric: so we have two distinct estimators for each off-diagonal entry. Therefore, we can look for a linear combination to reduce the Monte Carlo variance: equivalently, we can correct the estimator Γ_{ij} using the mean-zero $(\Gamma_{ji} - \Gamma_{ij})$ as a control variate. The coefficients will be chosen by a small-sample preliminary Monte Carlo run as is customary for control variates (Glasserman, 2004). Some methods benefit from this trick more than others, as one can see from the last column of Table 7: in particular, functional Gamma comes closer to finite differences (but is still takes more than twice the time to get unit variance), and VAD becomes as good as DAAD2 on Cross Gammas.

5.2.2 Basket digital

A good representative example of a discontinuous multi-asset product is a digital option on a basket:

$$I_{(K,+\infty)}\left(\frac{1}{N_X}\sum_{i=1}^{N_X} A_i(T)\right).$$

As in the previous subsection, we take $N_X = 8$, $T = 1$ year and $K = 100$.

The results for the full Hessian are in Table 8: they show that FDIFF2 is inadequate for discontinuous multi-asset payoffs, while FDDAAD and HOPP are not optimal either. DAAD2 and VFD are comparable, with a slight preference on the latter especially on the diagonals.

Method	Gamma		Volga		Vanna	
	diagonal	cross	diagonal	cross	diagonal	cross
FDIFF2	24494.36	2521.50	11644.63	870.75	1282.64	1277.06
HOPP	619.69	565.07	308.62	102.61	277.85	229.02
VFD	85.31	86.34	45.81	24.15	46.36	45.30
FDDAAD	377.14	380.68	165.97	164.79	265.57	268.29
DAAD2	131.27	120.78	74.61	26.69	62.35	55.48

Table 8: Second order sensitivities of a digital call with maturity $T = 1y$ and strike $K = 100$ on a basket of 8 underlyings: average time per unit Monte Carlo variance on and off the diagonal, all methods.

Method	Gamma	
	diagonal	cross
FDIFF2	6311.20	649.68
HOPP	344.73	314.35
VFD	45.28	45.83
FDDAAD	206.68	205.29
DAAD2	71.28	65.58
FΓ-VB	26.50	26.25
FΓ-OPP	27.02	27.73
FΓ-DAAD	11.66	11.50

Table 9: Gamma sensitivities of a digital call with maturity $T = 1y$ and strike $K = 100$ on a basket of 8 underlyings: average time per unit Monte Carlo variance on and off the diagonal, all methods.

The biggest surprise, however, comes from the estimators restricted to the Gamma matrix in Table 9. Indeed, we see that the best linear-cost methods DAAD2 and VFD are clearly overcome by constant-cost functional Gamma methods, with FΓ-DAAD being 4-5 times faster than the fastest linear-complexity algorithm (always in uncertainty-adjusted terms). This is at odds with the results obtained in the previous subsection on the "easy" call payoff, probably because on continuous payouts traditional methods already work very well.

6 Conclusions

In this paper, we have compared a wide range of old and new methods for the computation of second order sensitivities of Monte Carlo prices of financial products, with special emphasis on algorithmic efficiency when a large number of Greeks must be computed.

From the literature review viewpoint, we have examined, with extensions (LR-MAAD, VAD) and variations (VFD), the main algorithms to compute the full Hessian matrix of an expected value when the underlying is driven by a multidimensional Brownian diffusion. Several limitations of applicability are found:

1. Many methods have too high Monte Carlo uncertainties: in particular, those based on likelihood ratio (LRMAAD, AADLRM), and to a lesser extent those based on finite differences of the pathwise estimator (FDPW).

2. Most methods only apply when there is at least one random driver per underlying ($N_X \leq N_W$) and the payoff does not depend directly on the initial state $\boldsymbol{X}_0$, nor on the model parameters $\boldsymbol{\theta}$ (the exceptions being HOPP and the dumb FDPW).

3. Most methods cannot handle discontinuous payoffs (the exceptions being HOPP and VFD).

4. For all methods except LRMAAD restricted to sensitivities to $\boldsymbol{X}_0$ only, the multiplicative overhead on the run time for only-price grows linearly in the number of rows of the Hessian.

The original estimators of this paper should be considered in view of the above limitations of their many competitors (see Table 10 for a visual summary):

- DAAD2 is the only method besides HOPP which solves simultaneously issues 1, 2 and 3, and in our tests was always significantly more effective then the latter from the statistical uncertainty point of view, while showing remarkable smoothness properties with respect to small changes in the input parameters.

- FΓ is the first estimator solving simultaneously issues 1 and 4 in at least one relevant case; it is also the first constant-complexity algorithm applicable to discontinuous payoffs. Its best-of-class efficacy in the basket digital test case, coupled with the easy implementation, qualify it as a must-try when tackling a new tough high-dimensional problem. Since it is in fact a meta-algorithm based on any first order estimator, it offers significant flexibility for either seamless introduction in any financial library already capable of computing first order sensitivities, or for fine tuning to specific payoffs.

Besides systematic testing on a large range of payoff types and model dynamics, a couple of issues remain open for further research. In the field of linear-cost methods, the main challenge is related to payouts with many discontinuities, for which the computational complexity of the adjustments becomes relevant, particularly when it scales quadratically in the number of digital features as for the current implementation of DAAD2. Moreover, this paper opens the new field of constant-cost methods, where a general-purpose algorithm is still to be found, since FΓ in one example showed poor performance, and is anyway limited to derivatives with respect to $\boldsymbol{X}_0$. Finally, one might consider the possibly varied effectiveness of variance reduction techniques to different estimators.

Declaration of interest

The author reports no conflicts of interest. The author alone is responsible for the content and writing of the paper.

Table 10: Properties and domain of applicability of the methods.

	FDPW	LRMAAD	AADLRM	VAD	HOPP	DAAD2	FΓ
Low variance	✗	✗	✗	✓	✓	✓	✗/✓[1]
Non-autonomous	✓	✗	✗	✗	✓	✓	✓
Discontinuous	✗	✗	✗	✓[2]	✓	✓	✓[3]
Constant-cost Γ	✗	✓	✗	✗	✗	✗	✓
Full Hessian	✓	✓	✓	✓	✓	✓	✗

[1] Depending on the payoff.
[2] In fact one must substitute AD with a finite difference (VFD).
[3] If handled by the underlying algorithm.

Acknowledgements

The author is grateful to Giorgio Facchinetti and Massimo Morini for their comments on the drafts of this paper, and to Andrea Pallavicini, Giulio Sartorelli, Giuseppe Di Poto and Riccardo Longoni for helpful discussions on its topic.

References

Basel Committee on Banking Supervision (2015). Review of the credit valuation adjustment risk framework.

Capriotti, L. (2008). Reducing the variance of likelihood ratio Greeks in Monte Carlo. In Mason, S. J., Hill, R. R., Mönch, L., Rose, O., Jefferson, T., and Fowler, J. W., editors, *Proceedings of the 2008 Winter Simulation Conference*, pages 587–593.

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

Capriotti, L. (2015). Likelihood ratio method and algorithmic differentiation: Fast second order Greeks. *Algorithmic Finance*, 4:81–87.

Capriotti, L. and Giles, M. (2012). Adjoint Greeks made easy. *Risk*, (September):92–99.

Chan, J. H. and Joshi, M. (2015). Optimal limit methods for computing sensitivities of discontinuous integrals including triggerable derivative securities. *IIE Transactions*, 47:978–997.

Daluiso, R. and Facchinetti, G. (2018). Algorithmic differentiation for discontinuous payoffs. *International Journal of Theoretical and Applied Finance*, 21.

Giles, M. (2007). Monte Carlo evaluation of sensitivities in computational finance. Technical Report NA07/12, Oxford University Computing Lab.

Giles, M. (2009). Vibrato Monte Carlo sensitivities. In L'Ecuyer, P. and Owen, A. B., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 369–382, Berlin. Springer.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer, Berlin.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, second edition.

Joshi, M. and Yang, C. (2011). Algorithmic Hessians and the fast computation of cross-gamma risk. *IIE Transactions*, 43:878–892.

Joshi, M. and Zhu, D. (2016). Optimal partial proxy method for computing gammas of financial products with discontinuous and angular payoffs. *Applied Mathematical Finance*, 23(1):22–56.

Karatzas, I. and Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*. Springer, New York.

Pagès, G., Pironneau, O., and Sall, G. (2018). Vibrato and automatic differentiation for high order-derivatives and sensitivities of financial options. *Journal of Computational Finance*, 22(2):1–34.

Reghai, A., Kettani, O., and Messaoud, M. (2015). CVA with Greeks and AAD. *Risk*, (December).