# Algorithmic differentiation for callable exotics

Alexander Antonov
Numerix QR

September 15, 2016

**Abstract**

In this article, we study the algorithmic calculation of present values greeks for callable exotic instruments. The speed of greeks evaluations becomes important with recent initial margin rules, including the ISDA standard model SIMM, requiring sensitivity calculations for non-cleared deals (e.g. callable exotic/structured products) for a large set of risk factors and on a daily basis.

In general, present values of callable trades are obtained by the backward repetitive application of conditional expectation via least-squares Monte Carlo (regression) and arithmetic operations encoded in so called pricing (payoff) script. By nature, this pricing procedure is known to be computationally intensive leading to very slow greeks, if computed by a simple bump-and-reprice. The fast alternative is the adjoint differentiation (AD) method. However, a direct application of the AD is not straightforward because regressions introduce path interdependency, whereas traditional AD is applied on a path-by-path basis.

We describe how to extend the traditional AD method to include the regressions. Usually, the procedure must include the recording of information in a data structure, often referred to as the instrument tape, during the backward pricing. This is followed by a subsequent "playback" of the tape in which the recorded information is used to compute the greeks. The tape adds a significant complexity to the AD implementation and can introduce memory issues (due to the significant storage demands), coding and debugging difficulties, etc.

Next, we propose a new *tapeless* differentiation method, called backward differentiation (BD), for exotic deals. It is applied during the backward pricing procedure following the payoff script. Importantly, it completely avoids the tape and all of its related complications.

We demonstrate the efficiency of the BD on the example of a Bermudan swaption calculated with a one-factor Hull-White model. For 50–100 greeks, the BD calculation time is only 2–6 times slower than a single pricing which leads to up 20 times acceleration with respect to the bump-and-reprice. We also give a numerical convergence analysis demonstrating the high quality of the BD greeks.

## 1 Introduction

Greek calculations are complicated and time-consuming operations in financial software. A greek is defined as the response of the present value (PV) of a trade or portfolio of trades to some small change in the market parameters (spot, implied volatility etc.) and is often computed in two steps:

1. The calculation of changes in the model parameters with respect to the market change,
2. The calculation of the change in the trade's value with respect to the change in the model parameters.

We assume that we have successfully completed the first step[1] and concentrate on determining the values that measure the price's dependence on the *model* parameters, i.e., the *model greeks* described in the second step.

In this work, we address exotic instrument pricing. Calculations for these types of instruments are traditionally performed backward in time via a sequence of conditional expectations of discounted values. In general, an instrument's value is determined using a numerical method specified in terms of a payoff (or pricing) script; we will concentrate on Monte Carlo (MC) methods with regressions, keeping in mind the generalization of this method to XVA and exposure calculations.

[1]In general, the first step is easier and less time-consuming than the second.

There are three main methods for calculating model greeks (i.e., a trade or portfolio's sensitivity to the model parameters):

- Bump-and-reprice
  This is the traditional, simplest approach to calculating greeks. In this method, we change the model parameters, regenerate MC paths using the new parameters (but usually the same random numbers), and reprice the instruments. Lastly, we observe the response of the value of an instrument to the change in the model parameters by subtracting its base scenario value from the newly obtained value.
- Payoff differentiation
  This method employs the differentiation of payoffs and model driving factors over model parameters. It includes various approaches to differentiation, e.g., algorithmic or adjoint. Note that these methods require some care in handling nonsmooth payoffs, such as those related to digital or barrier instruments.
- Likelihood methods
  Here, the PV derivatives over the model parameters are calculated by differentiating the underlying probability density of the model driving factors. One example of a likelihood method is the Malliavin calculus method; see [6] for a full discussion. The main advantage of this method is that it does not assume that the payoff is smooth. The main drawback is the method's nonuniversality; it is very bespoke and highly dependent on the process and the instrument at hand.

In general, the *quality* of the bump-and-reprice derivatives (at least, for sizable bumps) is equivalent to those obtained by payoff differentiation. Thus, the decision of whether or not to use a method is based on its speed. This depends on multiple factors, including but not limited to the code architecture, the number of market bumps with respect to the model parameters,[2] and the dependence of a model's states on its parameters (analytical dependence, local parameters, etc.).

A traditional payoff differentiation is applied as a *path-by-path* differentiation to forward instruments. It can either be direct (i.e., forward in time) or adjoint (i.e., backward in time). Direct differentiation is more efficient when the number of results (outputs) is greater than the number of parameters. Adjoint differentiation (AD) is more efficient in the opposite case and is more natural in finance.[3] The AD was introduced into mathematical finance by Giles and Glasserman in 2006 [7] and has gained wide popularity since then; see Capriotti-Giles [3] and Andreasen [1], for instance.

AD coding is considered complicated and requires a substantial effort; it can sometimes lead to the rewriting of entire libraries. There are two main ways to implement the AD. The first one is a *manual* arrangement of differentiation capabilities. For example, it can be based on the overloading of certain payoff scripts components called continuation values. The second way is an *automatic* overload of the `double` type in the code to add differentiation capabilities to all of the operations containing `double` variables. To be efficient this approach requires a substantial amount of work, primarily in terms of deciding which variables to differentiate and which variables are intermediate (i.e., are not to be differentiated). It is also complicated to handle nondifferentiable operations such as comparison, which may be very common in the code. In general, the manual and automatic methods deliver the same performance if they are properly implemented.

Recall that AD is an *adjoint* action, meaning that it takes place *after* the main (pricing) calculation. Therefore, it is necessary to record information that is generated during pricing that will be utilized later on for the AD. All of the pricing operations are recorded in a data structure commonly referred to as the *tape*. During the adjoint pass, we "play back" the tape to perform the differentiation using the recorded information. The tape adds a lot of complexity to the AD implementation in the form of possible memory issues due to large storage demands, coding, debugging, and maintenance difficulties, etc. To avoid these issues, we will propose a new *tapeless* differentiation method below.

In this article, we concentrate on the manual AD method applied to exotic (structured) products. The speed of greeks calculation becomes important with recent initial margin rules, including the ISDA standard model SIMM, requiring sensitivity calculations for non-cleared deals (e.g. structured products) for a large set of risk factors and on a daily basis. It is also important for calculation of CVA sensitivities in the recent FRTB-aligned regulation SA-CVA.

To our knowledge, the public domain AD articles do not discuss the mathematical details of the regression, which is the key numerical method for exotic deals. Regression is one of the most complex and

[2]A low number of *market* bumps can result in the computation of relatively many derivatives over the *model* parameters.
[3]The widely-known abbreviation AAD is used for adjoint algorithmic (or automatic) differentiation. However, we use the term AD as defined above, keeping in mind its algorithmic property.

time-consuming numerical methods. Moreover, it introduces path interdependencies that compromise the ability to perform path-by-path valuation and differentiation. The value of a given path at a particular point in time can depend on the values of all paths (including itself) at previous time steps. This property makes a direct AD application difficult and is why people often avoid the regression and instead use different approximations; see [1], for example. Also, if the continuation value is itself a callable option, the first derivative of the exercise boundary over the model parameters is zero. This means that we do not need to differentiate the exercise boundary; we only need to differentiate the underlying swap cashflows. However, we look for a universal method that can be applied uniformly across all deal types represented by our (or any other) payoff script.

In spite of the complications, the AD of the regression is appealing because, unlike the bump-and-reprice approach, it avoids the regression repetition encountered during the instruments' repricing. Curiously, we did not find any public references including a detailed description of the AD for exotic instruments with the full regression. Nevertheless, in Davidson's report [5] we discover that a portion of the major banks applies AD to their structured products. The author cites Danske Bank, Natixis, Nomura, SecGen, etc. Moreover, various authors at conferences and workshops (e.g., Andreasen [1], Sokol-Ballabio [8], Capriotti-Giles-Nauman [4]) have stated that they can handle the regression AD; however, full details are not provided.

Here, we give a technical recipe for the traditional application of the AD approach to general instruments being priced with full regression. Moreover, we go further and propose a new differentiation technique which we call backward differentiation (BD). It is applied at the time of pricing, similar to the algorithmic exposure calculation in [2]. This completely avoids the tape and its related complications. Finally, we come up with an acceleration technique that is particularly efficient with the regression, making BD even more attractive.

To confirm our theoretical results, we come up with numerical experiments of a Bermudan swaption calculated with a one-factor Hull-White (HW) model. For 50–100 greeks, the BD calculation time is only 2–6 times slower than a single pricing calculation, comparable with a standard AD slowdown. We also give a numerical convergence analysis demonstrating the high quality of the BD greeks.

The article is organized as follows. In Section 2, we use a simple forward instrument example to provide an overview of the properties of AD and demonstrate why it is much faster than the bump-and-reprice approach. In Section 3, we recall different notions of backward pricing using the example of a Bermudan swaption. In Section 4, we describe a traditional application of the AD to general instruments followed by a description of the new, tapeless, BD technique. We complete the theoretical portion of our discussion in Section 5 by introducing an acceleration of the regression differentiation. We compare and contrast the AD and BD techniques in Section 6. Section 7 is devoted to numerical experiments in which we compare greeks calculation speed for a Bermudan swaption using the different methods described above.

## 2 Pathwise greek calculations for forward instruments

Let us introduce our notations and derive some preliminary formulas. We will study a general instrument's price dependence on the *model* parameter $\theta = \{\theta_1, \ldots, \theta_{N_\theta}\}$, an $N_\theta$-dimensional vector.

Denote by $X(t) = \{X_1(t), \ldots, X_{N_X}(t)\}$ an $N_X$-dimensional state vector of the model that is considered to be a hybrid model with a potentially high number of components. We sometimes denote the dependence of the state on the model parameters by writing $X(t;\theta)$.

We define a general Markov discretization scheme on a set of $N_t$ simulation dates $\{t_i\}_{i=1}^{N_t}$ by

$$X(t_{i+1}) = G(t_i, X(t_i), Z_i, \theta), \tag{1}$$

where $Z_i$ is a Gaussian random variable with the origin is at $t_0$. Note that in [3] the authors use a general non-Markovian scheme in which the function $G$ depends on the entire $X$ path; however, the majority of these models are Markovian.

Let us consider a general path-dependent discounted payoff $P(X) = P(X(t_1), \ldots, X(t_{N_t}))$ and the corresponding value $V(\theta) = \mathbb{E}\left[P(X(\theta))\right]$ for a given model expectation $\mathbb{E}$. When computing greeks, the goal is to calculate the price derivative over all model parameter $\theta_n$:

$$\frac{\partial V(\theta)}{\partial \theta_n} = \mathbb{E}\left[\frac{\partial P(X)}{\partial \theta_n}\right],$$

where the payoff derivative can be expressed via the chain rule:

$$\frac{\partial P(X)}{\partial \theta_n} = \sum_{k=1}^{K} \frac{\partial P(X)}{\partial X(t_k)} \frac{\partial X(t_k)}{\partial \theta_n}. \tag{2}$$

Strictly speaking, we calculate the value $V$ using MC simulations, i.e.,

$$V(\theta) \simeq \frac{1}{N_p} \sum_{p=1}^{N_p} P(X(\theta)[p]),$$

where $N_p$ is the number of MC paths and $X(t)[p]$ is the state realization of the $p$th path. Below, we calculate the underlying derivatives $\partial X(t_k)[p]/\partial \theta_n$ path-by-path. For notational brevity, we will omit the path index.

To obtain the derivative (2), we make use of the *entire* dependence of the process $X(t)$ on the parameter $\theta$. This dependence can be demonstrated by rewriting $X(t_{i+1})$ using (1) iteratively:

$$\begin{aligned} X(t_{i+1}) &= G(t_i, X(t_i), Z_i, \theta) \\ &= G(t_i, G(t_{i-1}, X(t_{i-1}), Z_{i-1}, \theta), Z_i, \theta) \\ &= G(t_i, G(t_{i-1}, G(t_{i-2}, X(t_{i-2}), Z_{i-2}, \theta), Z_{i-1}, \theta), Z_i, \theta) \\ &= \cdots \\ &= X(t_{i+1}; Z_i, \cdots, Z_1; \theta). \end{aligned}$$

This can be easily presented via the "partial" derivative of $G$, i.e.,

$$\frac{\partial X(t_{i+1})}{\partial \theta} = \frac{\partial G(t_i, X(t_i), Z_i, \theta)}{\partial \theta} + \frac{\partial G(t_i, X(t_i), Z_i, \theta)}{\partial X(t_i)} \frac{\partial X(t_i)}{\partial \theta}. \tag{3}$$

This iterative *forward* scheme can be applied to calculate the desired derivatives of the states, $\partial X(t_k)/\partial \theta_n$, as well as the derivatives of the payoff, $\partial P(X)/\partial X(t_k)$, from (2).

An alternative greek calculation is based on the *backward* substitution of the (1) into the payoff:

$$\begin{aligned} P(X(t_1), \cdots, X(t_{N_t})) &= P(X(t_1), \ldots, X(t_{N_{t-1}}), X(t_{N_t}, G(t_{N_t-1}, X(t_{N_t-1}), Z_{N_t-1}, \theta)) \\ &= \cdots \\ &= P_i(X(t_1), \ldots, X(t_i); Z_i, \ldots, Z_{N_t-1}; \theta), \end{aligned}$$

where we have denoted by $P_i$ the payoff value for which we have backwardly substituted the states until its latest state dependence is on $X(t_i)$. In other words, we have calculated the payoff *conditional* on the time $t_i$, substituting all $X(t_j)$ from the very end ($j = t_{N_t}$) back to $j = i + 1$. Thus, we have obtained explicit dependence on the parameters thanks to the substitution while some information about $\theta$ can still be hidden in $X(t_i)$, i.e.,

$$P_i(X(t_1), \ldots, X(t_i); Z_i, \ldots, Z_{N_t-1}; \theta) = P(X(t_1), \ldots, X(t_{N_t}))\big|_{t_i}. \tag{4}$$

Our goal is to find the payoff dependence on the states for which we have substituted all of the theta dependencies:

$$P_0(Z_1, \ldots, Z_{N_t-1}; \theta) = P(X(t_1), \ldots, X(t_{N_t}))\big|_{t_0}. \tag{5}$$

The recursive relationship between $P_i$ and $P_{i+1}$,

$$\begin{aligned} P_i(X(t_1), \ldots, X(t_i); Z_i, \ldots, Z_{N_t-1}; \theta) &= P_i(t_1, \ldots, t_{i-1}, G(t_{i-1}, X(t_{i-1}), Z_{i-1}, \theta); Z_i, \ldots, Z_{N-1}; \theta) \\ &= P_{i-1}(X(t_1), \ldots, X(t_{i-1}); Z_{i-1}, \ldots, Z_{N-1}; \theta), \end{aligned}$$

is the key to expressing the payoff derivative over the parameters in a *backward* manner, keeping in mind that our goal is to find $\partial P_0/\partial \theta$:

$$\frac{\partial P_{i-1}}{\partial \theta} = \frac{\partial P_i}{\partial \theta} + \frac{\partial P_i}{\partial X(t_i)} \frac{\partial G(t_{i-1}, X(t_{i-1}), Z_{i-1}, \theta)}{\partial \theta}, \tag{6}$$

preparing at the same time the payoff derivative over $X(t_{i-1})$:

$$\frac{\partial P_{i-1}}{\partial X(t_{i-1})} = \frac{\partial P}{\partial X(t_{i-1})} + \frac{\partial P_i}{\partial X(t_i)} \frac{\partial G(t_{i-1}, X(t_{i-1}), Z_{i-1}, \theta)}{\partial X(t_{i-1})}. \tag{7}$$

Note that the first derivative on the right-hand side of (7) is a partial derivative of the initial payoff over its explicit dependence on $X(t_{i-1})$ while the second term is responsible for the *substituted* derivative over $X(t_{i-1})$ coming from the higher time steps.

For both types of calculations we need the following derivatives of the discretization function:

$$\frac{\partial G(t_i, X(t_i), Z_i, \theta)}{\partial \theta}, \tag{8}$$

which is an $N_\theta \times N_X$ matrix per path per time step, and

$$\frac{\partial G(t_i, X(t_i), Z_i, \theta)}{\partial X(t_i)}, \tag{9}$$

which is an $N_X \times N_X$ matrix per path per time step. Here, $N_\theta$ is the number of parameters and $N_X$ is the number of model states. In addition, the scheme defined in (6) and (7) requires the partial derivatives of the payoff,

$$\frac{\partial P}{\partial X(t_i)}, \tag{10}$$

which is a vector of length $N_X$ per path per time step.

Direct differentiation. The scheme in (3) is a direct iterative procedure where, during the forward simulations, we calculate the state derivatives over the parameters. Having the information from (8) and (9) (obtained on-the-fly during simulation, without storage), we can perform the iteration in (3), which costs $N_\theta \times {N_X}^2$ multiplications per path and per time step. (This number can be smaller if the matrix $\partial X(t_i)/\partial\theta$ is sparse enough.) Thus, the total computational cost of the direct differentiation is

$$N_\theta \times {N_X}^2 \times N_t \times N_p.$$

It is important to stress that one should repeat the whole procedure for each new parameter.

When the derivatives $\partial X(t_i)/\partial\theta$ are ready, we can calculate pathwise derivatives for any payoff at the cost of the number of explicit payoff dependencies on the states times $N_\theta$ per path.

Adjoint differentiation. For any fixed payoff $P$, the AD requires backward propagation. Having the information from (8) and (10) calculated and stored in the *tape* we go backward in time, calculating the payoff derivatives over the states and over the parameters. The derivative calculations $\partial P_i/\partial X(t_i)$ from (7) cost ${N_X}^2$ multiplications per path and per time step per payoff. The vector iteration (6) is often less expensive even for large number of parameters; this happens for time-dependent parameters when the matrix $\partial G/\partial\theta$ is quite sparse, leading to costs of $O(1)N_X$ multiplications per path and per time step per payoff. Thus, the overall cost is dominated by the calculation of derivatives over the parameter and is reduced to

$${N_X}^2 \times N_p \times N_t \tag{11}$$

multiplications.

Let us stress again that the cost of the adjoint procedure is independent of the number of parameters due to the sparse structure of $\partial G/\partial\theta$ for time-dependent parameters. If this were not the case, i.e., if all the parameters were global, the total derivatives calculation time could be as big as $\sim N_\theta \times N_X \times N_p \times N_t$, i.e., proportional to the number of parameters and only $N_X$ times faster than direct differentiation.

*Below, we assume that all of our parameters are time dependent.*

We also note that the AD procedure should be repeated for each new payoff, which is not the case for direct differentiation. This fact should absolutely be taken into account when comparing the speeds of direct and adjoint differentiation.

To finish this simple analysis on the differentiation speed, we will mention that other factors can also influence the performance of the AD with respect to the direct differentiation or the bump-and-reprice method. The most important of these factors is the speed of access to the memory, which directly influences the speed of writing and reading the tape. Another important factor is the speed of calculating the discretization function $G$ and its derivatives. If the function $G$ is "heavy",[4] its derivatives are often fast to calculate using the information about the function itself. In this case, the pathwise derivatives methods will be preferable over the bump-and-reprice method. However, heavy derivatives can sometimes still be an important slowdown factor for the pathwise methods.

[4]I.e., the function $G$ is not given in closed form and instead must be found using a numerical procedure.

## 2.1 AD structure for general forward instruments

We have seen in the simple example above that AD contains the following steps:

- *Simulation tape preparation*
  Simulate the model states and write the required information from (8) and (9). We will refer to this step as "preparing" a *simulation tape*. It can be done either path-by-path or for all paths.
- *Instrument tape preparation*
  In our example the instrument tape was simple: it contains only derivatives of the payoff over the states; i.e., (10). However, for general forward instruments one should store the derivatives of the payments over the states as well as derivatives over parameters (if present) during the forward pricing, i.e.,
  $$\frac{\partial P}{\partial X(t_i)} \tag{12}$$
  per path and per time step and
  $$\frac{\partial P}{\partial \theta_n} \tag{13}$$
  per path and per parameter.
- *Adjoint calculation of the instrument derivatives*
  In our example, the calculation was backward (i.e., in the opposite direction of the forward pricing) and path-by-path, see (6) and (7). The explicit derivatives of the payoff over the parameters results in the "initial" condition
  $$\frac{\partial P_{N_t}}{\partial \theta_n} = \frac{\partial P}{\partial \theta_n} \tag{14}$$
  for Equation (6) and $i = N_t$.

This concludes our discussion on forward instruments. However, as we discuss the application of the AD to backward deals, keep in mind that we will try to obtain the same advantages seen in the backward (in time) AD procedure for forward instruments, in spite of the regression path-mixing that arises for backward deals.

## 3 Exotic instrument pricing

In this section, we briefly explain backward instrument pricing; see [2] for more details on general scripting instruments. The main component of the backward pricing procedure is a *continuation value* (CV) at time $t$ expressed in some currency units and denoted $V(t)$. The CV is a certain function of the model state variables at time $t$. Financially, $V(t)$ is a *hold* value: it is the price of holding an option at time $t$ with all possible payments and exercises taking place thereafter. For coherence, we express all of the CVs in domestic currency units. A simple CV example is a leg with a single unit payment at time $T$. More complicated examples include a swap value or a swaption holding value.

Consider a *Bermudan swaption* giving the right to enter into a swap on exercise dates $T_i$. Suppose the underlying swap pays a *generalized payment* $\alpha_j$ in some currency at date $\tau_j$ for $j = 1, \ldots, M$. For example, it may be a fixed payment $\alpha_j =$ `notional*dcf*`$K$ with a fixed rate $K$ or a floating payment $\alpha_j =$ `notional*dcf*`$L_{j-1}$ with Libor rate $L_{j-1}$ fixed at time $\tau_{j-1}$.

Let $V(t)$ and $S(t)$ denote the CVs of the swaption and the swap, respectively. We will refer to the union of exercise and payment dates as instrument dates. The swaption's backward pricing is performed using two repeated steps. The first step is to calculate the following transformations at the specified instrument dates:

$$V(T_j^-) = \max(V(T_j), S(T_j)) \text{ at } T_j, \tag{15}$$
$$S(\tau_k^-) = S(\tau_k) + \alpha_k \, P(\tau_k, \tau_{k+1}) \text{ at } \tau_k. \tag{16}$$

where $P(t, T)$ is a zero coupon bond price with maturity $T$ as observed at $t$.

Here and below we will write $T^-$ to denote the time just before the time $T$. Thus, the CVs are continuous from the right in time. By avoiding special notations for the CVs on the left-hand side, this convention serves to simplify the writing of the exercise and payment conditions. Otherwise, we can write such updates in the *coding style* using the operator "+=", so that (16) is equivalent to

$$S(\tau_k) \mathrel{+}= \alpha_k \, P(\tau_k, \tau_{k+1}).$$

The second step is propagation, i.e., calculating the discounted conditional expectation in the model measure between the instrument dates of the CVs at hand:

$$S(t) = N(t)\mathbb{E}\left[\left.\frac{S(T^-)}{N(T)}\right|\mathcal{F}_t\right],$$
$$V(t) = N(t)\mathbb{E}\left[\left.\frac{V(T^-)}{N(T)}\right|\mathcal{F}_t\right],$$

where $N(t)$ is the model numeraire. The numeraire can be a deterministic function of the states at the same time (e.g., in the case of a zero bond corresponding to a forward measure, $N(t) = N(t, X(t))$) or a function of the states at multiple times (e.g., in the case of a savings account under the risk-neutral measure).

A general instrument contains a set of CVs $V_j$ that are transformed using arithmetical operations and other functions (like max, step, etc.) of the instrument dates as well as the conditional expectations between the instrument dates.

In practice, we use numerical methods, namely least-squares MC, for such procedures. Thus, each CV is presented as a vector of path values $V(t) = \{V(t)[p]\}_{p=1}^{N_p}$ and the corresponding conditional expectation is calculated using regression.

For brevity, we fix two times $t_1$ and $t_2$ and denote continuation values and discount factors with a corresponding time superscript: $V(t_i) = V^{(i)}$, $D(t_i) = D^{(i)}$. Here, the discount factor is one over the numeraire, $D(t) = 1/N(t)$. We will write $X(t_1) = X$ for the states at time $t_1$.

For the numerical calculation of the conditional expectation

$$V^{(1)}(x) = \mathbb{E}\left[\left.\frac{V^{(2)}D^{(2)}}{D^{(1)}}\right| X = x\right], \tag{17}$$

we need to fix basis functions $\phi_n(x)$ and determine coefficients $\alpha_n$ which minimize the expectation

$$\chi^2 = \mathbb{E}\left[\left(\tilde{V}^{(2)} - \sum_n \alpha_n\phi_n(X)\right)^2 D^{(1)}\right], \tag{18}$$

where we define

$$\tilde{V}^{(2)} = \frac{V^{(2)}D^{(2)}}{D^{(1)}} \tag{19}$$

and where $\{\phi_n(X)\}_{n=1}^{N_b}$ is a regression basis. In practice, the unconditional expectation $\chi^2$ is calculated over the MC paths:

$$\chi^2 \simeq \frac{1}{N_p}\sum_{p=1}^{N_p}\left(\tilde{V}^{(2)}[p] - \sum_n \alpha_n\phi_n(X[p])\right)^2 D^{(1)}[p]. \tag{20}$$

Slightly abusing notations, we denote by $V^{(1)}[p]$ the $p$th path of results for the MC regression on our basis $\phi_n(X)$. The stochastic variable defined in such a way is obviously an approximation to the exact conditional expectation in (17). The errors come from two sources: the main one traditionally is due to the incomplete basis, while the second source is the MC statistical error. In all of the regression-based methods, the choice of the basis is the key. Below, we treat all of our equations in the MC sense, i.e., by assuming that simple averages are MC sums $\mathbb{E}[X] = (1/N_p)\sum_p X[p]$ and that conditional averages are results of the corresponding MC regressions (20).

Minimizing $\chi^2$ over the $\alpha_n$ coefficients yields the set of equations

$$\sum_n \alpha_n\mathbb{E}\left[\phi_n(X)\phi_m(X)D^{(1)}\right] = \mathbb{E}\left[V^{(2)}\phi_m(X)D^{(2)}\right]. \tag{21}$$

These can be easily solved:

$$\alpha_n = \sum_m C^{-1}_{nm}\mathbb{E}\left[\phi_m(X)\tilde{V}^{(2)}D^{(1)}\right], \tag{22}$$

where we have denoted the matrix

$$C_{nm} = \mathbb{E}\left[\phi_n(X)\phi_m(X)D^{(1)}\right]. \tag{23}$$

This gives us

$$V^{(1)}(x) = \sum_n \alpha_n\phi_n(x) = \sum_{nm}\phi_n(x)C^{-1}_{nm}\mathbb{E}\left[\phi_m(X)\tilde{V}^{(2)}D^{(1)}\right]. \tag{24}$$

For a set of given paths we have

$$V^{(1)}[p] = V^{(1)}(X[p]) = \frac{1}{N_p} \sum_{nm,p'} \phi_n(x[p]) C^{-1}_{nm} \phi_m(x[p']) D^{(2)}[p'] V^{(2)}[p']. \tag{25}$$

As mentioned in the introduction, this linear operation $V^{(2)} \to V^{(1)}$ "mixes the paths".

Below, we present various derivatives of the regression that we will need later. Namely,

$$\frac{\partial V^{(1)}[p]}{\partial \tilde{V}^{(2)}[p']} = \frac{1}{N_p} \sum_{nm} \phi_n(x[p]) C^{-1}_{nm} \phi_m(x[p']) D^{(1)}[p'], \tag{26}$$

$$\begin{aligned} \frac{\partial V^{(1)}[p]}{\partial X[p']} &= \delta_{pp'} {V^{(1)}}'[p] \\ &+ \frac{1}{N_p} \sum_{nm} \phi_n(X[p]) C^{-1}_{nm} \left\{ \phi'_m(X[p'])(\tilde{V}^{(2)}[p'] - V^{(1)}[p']) - \phi_m(X[p']) {V^{(1)}}'[p'] \right\} D^{(1)}[p'], \end{aligned} \tag{27}$$

$$\frac{\partial V^{(1)}[p]}{\partial D^{(1)}[p']} = \frac{1}{N_p} \sum_{nk} \phi_n(X[p]) C^{-1}_{nk} \phi_k(X[p']) \left( \tilde{V}^{(2)}[p'] - V^{(1)}[p'] \right), \tag{28}$$

where ${V^{(1)}}'[p]$ is the derivative of the regression result over the state, i.e.,

$${V^{(1)}}'[p] = \sum_n \alpha_n \phi'_n(X[p]) = \frac{1}{N_p} \sum_{nm,p'} \phi'_n(X[p]) C^{-1}_{nm} \phi_m(X[p']) \tilde{V}^{(2)}[p'] D^{(1)}[p'],$$

and $\phi'_n(x)$ is the derivative of $\phi_n(x)$ over its argument. See Appendix A for more details. Note that for simplicity we have considered the one-state case $N_X = 1$; the formulas (27) can be easily transformed to the multistate case.

We conclude this section with a comment on the derivative matrices. Note that their size is large ($N_p \times N_p$) and they heavily mix the paths. However, their rank is equal to the number of basis functions $N_b$, which is often quite small with respect to the number of paths. We will use this fact to proceed with the AD for this seemingly difficult case of exotic options.

## 4 Adjoint differentiation for exotic instruments

We are now ready to investigate the AD of exotic instruments that are priced by backward induction. Below, we present the standard algorithm along with our comments. We will end this section by analyzing the storage requirements and speed of this method.

### 4.1 The tapes

The initial step of the AD involves the storage and preparation of the tapes:

- *Simulation tape*
  This step is done during simulation. The storage size of the values in (8) and (9) is proportional to $N_X^2 \times N_p \times N_t$ for time-dependent parameters.
- *Backwards pricing tape*
  The information to store in this tape includes:
  - All of the continuation values for all of the time steps.
  - The derivatives of the continuation values during the regressions; i.e., the values (26)–(28).

  Denoting the number of instruments dates on which we perform the regressions as $N_r$, the memory per instrument, which is dominated by the storage of the derivatives "$dV/dx$" of the CVs, is:

  $$2\, N_X \times N_r \times N_b \times N_p, \tag{29}$$

  where $N_b$ is the number of the basis functions used during the regression. The factor of two represents the fact that we store the basis functions as well as their derivatives.

Once the tapes are prepared, we begin the adjoint algorithm with the forward-in-time derivatives calculation, which is in the reverse direction of the backward pricing. This procedure is based on the playback of the instrument tape.

## 4.2 The adjoint differentiation algorithm

The AD algorithm below describes the derivatives attribution to our main objects, PV and continuation values, as well as their algorithmic manipulations during the playback of the tape.

1. Derivatives attribution

   - The PV $Q$ bears two (adjoint) derivatives: derivatives with respect to the required parameters,
   $$\frac{\partial Q}{\partial \theta_k} \quad \text{for all } k,$$
   and intermediate derivatives with respect to the model states and discount factors,
   $$\frac{\partial Q}{\partial X(t_i)[p]} \quad \text{and} \quad \frac{\partial Q}{\partial D(t_i)[p]},$$
   for all time step indexes $i$ and paths $p$.
   - Each CV bears the payoff derivatives
   $$V(t)[p] \rightarrow \frac{\partial Q}{\partial V(t)[p]},$$
   where $[p]$ represents a path number. Operations involving CVs algorithmically influence the payoff derivatives. These derivatives are updated during a differentiation in the *forward* direction, reverse one w.r.t. the *backward* pricing. The derivatives above become data members of the CV and follow all of the overloaded operations and functions.

   As an example, we consider the PV of a CV at time $t$ that is equal to
   $$Q = N_p^{-1} \sum_p V(t)[p] D(t)[p],$$
   where $N_p$ is the number of MC paths and $D(t)$ is the model discount factor. Clearly,
   $$\frac{\partial Q}{\partial V(t)[p]} = N_p^{-1} D(t)[p] \tag{30}$$
   and
   $$\frac{\partial Q}{\partial D(t)[p]} = N_p^{-1} V(t)[p]. \tag{31}$$
   This is the first operation in the reverse mode (single PV is required):
   - It initializes $\partial Q / \partial V(t)[p]$, which is a property of the CV.
   - It initializes $\partial Q / \partial D(t)[p]$, which is a property of the PV.

2. Forward calculation of the instrument derivatives

   This is the main step of the algorithmic calculation of derivatives. Essentially, we iteratively transform the derivatives $\partial Q / \partial V(t)[p]$ into derivatives over the states and parameters.

   - *Arithmetic operations.*
   A summation $V(t) = V_1(t) + V_2(t)$ will update the derivatives over CVs $V_1$ and $V_2$ given $\partial Q / \partial V(t)[p]$. The logic is based on the substitution $Q(V(t)) = Q(V_1(t) + V_2(t))$, which leads to
   $$\frac{\partial Q}{\partial V_a(t)[p]} = \frac{\partial Q}{\partial V(t)[p]}.$$
   Another example is the maximum operation $V(t) = \max(V_1(t), V_2(t))$. Substituting $Q(V(t)[p]) = Q(\max(V_1(t)[p], V_2(t)[p]))$ gives us the following derivatives over $V_1(t)$ and $V_2(t)$:
   $$\frac{\partial Q}{\partial V_1(t)[p]} \mathrel{+}= 1_{V_1(t)[p] > V_2(t)[p]} \frac{\partial Q}{\partial V(t)[p]},$$
   $$\frac{\partial Q}{\partial V_2(t)[p]} \mathrel{+}= 1_{V_1(t)[p] < V_2(t)[p]} \frac{\partial Q}{\partial V(t)[p]}.$$
   These derivatives can be found for other operations in a similar fashion. Note, however, that the STEP operation is not well defined and should be somehow smoothed.

- *Creation of a financial object.*
  This operation deals with the creation of FX rates, zero bonds, etc. in terms of CVs which are known functions of the states or parameters:

  $$V(t) = V(t, X(t), \theta)$$

  leads to the substitution

  $$Q(V(t)) = Q(V(t, X(t), \theta)),$$

  which gives rise to the updates

  $$\frac{\partial Q}{\partial \theta} += \frac{\partial Q}{\partial V(t)[p]} \frac{\partial V(t)[p]}{\partial \theta}$$

  and

  $$\frac{\partial Q}{\partial X(t)[p]} += \frac{\partial Q}{\partial V(t)[p]} \frac{\partial V(t)[p]}{\partial X(t)[p]}.$$

- *Regression.*
  This item is the most complicated and time-consuming. Denote a general continuation value as $V(t)$. As we have seen in Section 3, the derivative of the regression $V(t_i) = \mathbb{E}\left[V(t_j)D(t_j)/D(t_i)|\mathcal{F}_{t_i}\right]$ is related to the matrix

  $$\frac{\partial V(t_i)[p]}{\partial \tilde{V}(t_j)[p']} = \frac{1}{N_p} \sum_{nm} \phi_n(X((t_i))[p]) C^{-1}_{nm} \phi_m(X((t_i))[p']) D(t_i)[p'],$$

  which has a small rank with respect to the number of path $N_p$ and which we have calculated in (26), denoting $\tilde{V}(t_j) = V(t_j)D(t_j)/D(t_i)$. This gives the derivative of the PV over the discounted CV as

  $$\frac{\partial Q}{\partial \tilde{V}(t_j)[p]} = \sum_{p'} \frac{\partial Q}{\partial V(t_i)[p']} \frac{\partial V(t_i)[p']}{\partial \tilde{V}(t_j)[p]},$$

  which has a computational cost of $N_p \times N_b$. Using the arithmetic operation logic described above (for the multiplication and division by the discount factors), we can transform this value into the final derivative $\partial Q/\partial V(t_j)[p]$, which is assigned to the CV $V(t_j)$.
  In the same way, we update the derivatives over the states:

  $$\frac{\partial Q}{\partial X(t_i)[p]} += \sum_{p'} \frac{\partial Q}{\partial V(t_i)[p']} \frac{\partial V(t_i)[p']}{\partial X(t_i)[p]}, \tag{32}$$

  where the matrix $\partial V(t_i)[p']/\partial X(t_i)[p]$ has also a reduced rank as discussed in Section 3. The cost of one regression is $2N_X \times N_p \times N_b$.
  The overall cost of the whole tape playback is dominated by the regression derivatives and is proportional to

  $$2N_r \times N_X \times N_p \times N_b$$

  multiplications.

3. Backward calculation of the PV derivatives.
   At the end of the of the previous step (instrument maturity) we have:
   - The derivatives with respect to the parameters:

     $$\frac{\partial Q}{\partial \theta_k},$$

     for all parameters $k$.
   - The derivatives with respect to the model states and discount factors:

     $$\frac{\partial Q}{\partial X(t_i)[p]} \quad \text{and} \quad \frac{\partial Q}{\partial D(t_i)[p]},$$

     for all time step indexes $i$ and paths $p$.

To get rid of the derivatives over the states and discount factors we proceed iteratively backwards starting from the last time step and ending at zero, exactly as described in our initial example in Section 3. This operation is path-by-path and, in general, is much faster than one requiring regressions.

Summing up our analysis of the standard AD we can state that the overall cost is dominated by the backward instrument Step 2 and is therefore

$$2\,N_r \times N_X \times N_p \times N_b \tag{33}$$

multiplications. We note that, strictly speaking, the multiplication cost is accompanied with a linear in $N_\theta$ term but with a tiny coefficient.

The storage cost is mainly determined by the cost of the instrument tape.

## 5 Backward differentiation of backward instruments: new approach without the instrument tape

This approach to the algorithmic differentiation works with *continuation values* instead of the final payoffs as in the usual AD. This allows us to avoid the instrument tape: the differentiation is done *during* the backwards pricing. We call the new algorithm backward differentiation (BD). Its main advantage is that it does not contain the instrument tape; this saves a lot of memory as well as the time needed to access it. Moreover, it makes the code simpler and easier to debug. Finally, we can obtain greeks for all of the CVs in a pricing script, e.g., swaptions, underlying swaps, legs, etc., in a single pass of the BD (the usual AD procedure must be rerun for every new payoff).

### 5.1 The algorithm

Below we give a detailed description of the BD algorithm. The derivative calculations are performed during the backward pricing. We overload the CV class by adding the derivatives over the states and the parameters to its data members. Each CV $V(t)$ for a time step $t$ stores the following information:

- The derivatives over the states:
$$\frac{\partial V(t)[p]}{\partial X(t)[p']}.$$
- The derivatives over the pathwise discount factors:
$$\frac{\partial V(t)[p]}{\partial D(t)[p']}.$$
- The derivatives over the parameters:
$$\frac{\partial V(t)[p]}{\partial \theta}.$$

We note that the derivatives attached to the CV are updated during its backward propagation, so there is no need to store them for all time steps. For ease of notation, we will silently include the discount factors into the states set, absorbing the $\partial V/\partial D$ derivatives into $\partial V/\partial X$, except when specifically mentioned. Also, we will implicitly assume that summations are over multiple state indexes in order to make the formulas more transparent.

The derivatives over the states are represented by a *diagonal* matrix plus a *cross-term* matrix of reduced rank $N_{V(t)}$ much smaller than the number of paths. (For example, the rank after the regression is equal to the number of basis functions $N_b$.) Explicitly, we have

$$\frac{\partial V(t)[p]}{\partial X(t)[p']} = \Delta(t)[p]\delta_{pp'} + \sum_{n,m=1}^{N_{V(t)}} A_n(t)[p]\Omega_{nm}(t)B_m(t)[p'] \tag{34}$$

or, in matrix notation,

$$N_p\left\{\underbrace{\begin{bmatrix} \\ \frac{\partial V(t)}{\partial X(t)} \\ \\ \end{bmatrix}}_{N_p}\right. = N_p\left\{\underbrace{\begin{bmatrix} \\ \Delta(t) \\ \\ \end{bmatrix}}_{N_p}\right. + N_p\left\{\underbrace{\begin{bmatrix} \\ A(t) \\ \\ \end{bmatrix}}_{N_{V(t)}}\right. \underbrace{\begin{bmatrix}\Omega(t)\end{bmatrix}}_{N_{V(t)}} \left.\underbrace{\begin{bmatrix}\quad B(t)^T \quad\end{bmatrix}}_{N_p}\right\}N_{V(t)} \tag{35}$$

where the left matrix $A(t)$ is often composed of the basis functions and the central matrix is the inverse $C^{-1}$:

$$A_n(t)[p] = \phi_n(X[p]) \quad \text{and} \quad \Omega(t) = C^{-1}(t). \tag{36}$$

There are some exceptions to this rule and so we supply the matrices $A$ and $\Omega$ with a CV index. Derivatives over the discount factor are parameterized in the same way. Their components get the superscript $D$; $A \to A^{(D)}$, etc. Note that the matrices $B$ are, in general, different for different CVs.

The backward pricing script operations modify the CV derivatives in an algorithmic way:

1. Arithmetic operations.
Arithmetic operations with CVs modify the derivatives according to the differentiation rules. For example, the maximum operation $V = \max(V_1, V_2)$ has the following derivatives:

$$\begin{aligned}\frac{\partial \max(V_1, V_2)}{\partial X(t)[p']} &= 1_{V_1(t)[p]<V_2(t)[p]}\left(\Delta_2(t)[p]\delta_{pp'} + \sum_{nm} A_{2,n}(t)[p]\Omega_{2,nm}(t)B_{2,m}(t)[p']\right) \\ &\quad + 1_{V_1(t)[p]>V_2(t)[p]}\left(\Delta_1(t)[p]\delta_{pp'} + \sum_{nm} A_{1,n}(t)[p]\,\Omega_{1,nm}(t)B_{1,m}(t)[p']\right) \\ &= \left(1_{V_1(t)[p]<V_2(t)[p]}\Delta_2(t)[p] + 1_{V_1(t)[p]>V_2(t)[p]}\Delta_1(t)[p]\right)\delta_{pp'} \\ &\qquad + \sum_{nm} 1_{V_1(t)[p]<V_2(t)[p]} A_{2,n}(t)[p]\Omega_{2,nm}(t)B_{2,m}(t)[p'] \\ &\qquad + \sum_{nm} 1_{V_1(t)[p]>V_2(t)[p]}\, A_{1,n}(t)[p]\Omega_{1,\,nm}(t)\, B_{1,m}(t)[p'].\end{aligned}$$

We see that the operation increases the rank unless $B_1(t) \sim B_2(t)$, i.e., their columns span the same space or include it. Note that the rank is also increased by the summation operation, unless $A_1 \sim A_2$ or $B_1 \sim B_2$.
Another example of an arithmetic operation is the discounting of a CV, e.g., for its backward propagation: $V(t) \to V(t)D(t)$. This will transform the diagonal term and the matrix $A$ while keeping the same rank:

$$\frac{\partial V(t)[p]D(t)[p]}{\partial X(t)[p']} = D(t)[p]\Delta(t)[p]\delta_{pp'} + \sum_{n,m=1}^{N_{V(t)}} (D(t)[p]A_n(t)[p])\Omega_{nm}(t)B_m(t)[p']$$

$$\frac{\partial V(t)[p]D(t)[p]}{\partial D(t)[p']} = \left(D(t)[p]\Delta^{(D)}(t)[p] + V[p]\right)\delta_{pp'} + \sum_{n,m=1}^{N_{V(t)}} \left(D(t)[p]A_n^{(D)}(t)[p]\right)\Omega_{nm}^{(D)}(t)B_m^{(D)}(t)[p'].$$

2. Creation of a financial object.
The creation of financial objects (FX, zero bonds, etc.) is performed similarly to the AD procedure.

3. Backwards propagation.
The backwards propagation is the key part of the algorithm. We consider it in detail for the discounted regression
$$V(t_i) = \mathbb{E}\left[ V(t_j) \frac{D(t_j)}{D(t_i)} \middle| \mathcal{F}_{t_i} \right].$$
The induction works as follows. We suppose that we are at a time step $t_j$ with a CV $V(t_j)[p]$ and its derivatives $\partial V(t_j)[p]/\partial X(t_j)[p']$ and $\partial V(t_j)[p]/\partial \theta$. We want to proceed backwards to a time step $t_i$ to obtain $V(t_i)$ and its derivatives $\partial V(t_i)[p]/\partial X(t_i)[p']$ and $\partial V(t_i)[p]/\partial \theta$. Note the inclusion of $D(t)$ into the states $X(t)$. For formula transparency, below we denote the times as $t_i = t$ and $t_j = T$.
First, we discount the CV: $V(T) \to V(T)D(T)$, following the rules above. Then we perform propagation of the result without using regression for the moment. We do this using the iterative substitution of the states $X(t_{n+1}) = G(t_n; X(t_n), \theta)$ as in Section 2,
$$\left.\frac{\partial V(T)[p]}{\partial X(t)[p']}\right|_t = \sum_{p''} \left.\frac{\partial V(T)[p]}{\partial X(T)[p'']} \frac{\partial X(T)[p'']}{\partial X(t)[p']}\right|_t, \tag{37}$$
$$\left.\frac{\partial V(T)[p]}{\partial \theta}\right|_t = \frac{\partial V(T)[p]}{\partial \theta} + \sum_{p'} \left.\frac{\partial V(T)[p]}{\partial X(T)[p']} \frac{\partial X(T)[p']}{\partial \theta}\right|_t. \tag{38}$$
Recall that we denote the CV with already-substituted values as $V(T)|_t$. Then, we discount $V(T) \to V(T)/D(t)$, which effectively gives us the derivatives of $\tilde{V}(T) = V(T)D(T)/D(t)|_t$ over the states $X(t)$ and the parameters.
Finally, we calculate the regression derivatives $\partial V(t)[p]/\partial \tilde{V}(T)[p']$ and $\partial V(t)[p]/\partial X(t)[p']$ by (26)–(28) to obtain the desired result via the following matrix multiplication:
$$\frac{\partial V(t)[p]}{\partial X(t)[p']} = \underbrace{\frac{\partial V(t)[p]}{\partial X(t)[p']}}_{\text{from the regression}} + \sum_{p''} \left.\frac{\partial V(t)[p]}{\partial \tilde{V}(T)[p'']} \frac{\partial \tilde{V}(T)[p'']}{\partial X(t)[p']}\right|_t, \tag{39}$$
$$\frac{\partial V(t)[p]}{\partial \theta} = \sum_{p''} \left.\frac{\partial V(t)[p]}{\partial \tilde{V}(T)[p'']} \frac{\partial \tilde{V}(T)[p'']}{\partial \theta}\right|_t. \tag{40}$$
We note that the final derivative over the states (39) is composed by that obtained from the regression (the first term) and from the states substitution (the second term).

4. The results.
At the end of the backward pricing we obtain the derivatives of all the payoffs $Q_i$ associated with the continuation values $V_i$ at the origin,
$$\frac{\partial Q_i}{\partial \theta} = \frac{1}{N_p} \sum_p \frac{\partial V_i(t_0)[p]}{\partial \theta}.$$

## 5.2 Speed

We begin by calculating the number of multiplications in the regression setup before addressing other time-consuming operations.

The total cost of the propagation in (37) and (38) is equal to the cost of calculating $\partial X(t_j)/\partial X(t_i)|_{t_i}$ and $\partial X(t_j)/\partial\theta|_{t_i}$ plus the cost of their matrix multiplication. The cost of the derivatives is based on the AD cost in (11) applied to the number of time steps between $t_j$ and $t_i$, denoted $N_{rt}$, which yields
$$O(1)N_X^2 \times N_{rt} \times N_p. \tag{41}$$
The matrix multiplications will require
$$O(1)N_X \times N_{rt} \times N_p \times N_{V(t)} \tag{42}$$
`double` multiplications. This cost is dominated by the multiplications inside (38) due to diagonal structure of $\partial X(t_j)/\partial X(t_i)|_{t_i}$. We also used the fact that the derivatives $\partial X(t_j)/\partial\theta|_{t_i}$ are non-zero for $O(1)N_{rt}$ elements due to our assumption that the parameters are time dependent.

Now we calculate the number of multiplications in the second step of the regression from (39) and (40). Equation (39) contains

$$N_p \times N_b \times N_{V(t)} \times N_X \tag{43}$$

multiplications per regression date because the derivatives $\partial V/\partial V$ and $\partial V/\partial X$ have rank $N_b$ and $N_V$, respectively. Equation (40) contains

$$O(1) N_p \times N_b \times N_\theta \times N_X \tag{44}$$

multiplications per regression date. Thus, the total number of multiplication in a CV regression is dominated by those from (39) and (40) and is equal to

$$N_r \times N_p \times N_b \times N_X \times \left(O(1) N_V + O(1) N_\theta\right). \tag{45}$$

Some other operations that add a lot of time to the procedure include:

- Memory allocation, especially 1) while creating the left and right matrices ($A$ and $B$) of derivatives over the states, and 2) when arithmetic operations lead rank increases in derivative matrices.
- Memory access.
- Multiplications during the arithmetical operations and regression derivatives preparation.

According to our experiments, the leading source of slowdown is due to the relatively high rank of derivatives matrices, which can rapidly grow as the number of operations increases. In the next subsection we will present a method for reducing this slowdown.

### 5.2.1 Acceleration techniques

Consider a conditional expectation (without discounting, for simplicity) approximated by the regression on our basis:[5]

$$V(t)(x) = \mathbb{E}\left[V(T)\,|\, X(t) = x\right] = \sum_n \alpha_n \phi_n(x) = \sum_{nm} \phi_n(x) C^{-1}_{nm} \mathbb{E}\left[V(T)\phi_m(X(t))\right],$$

where $C_{nm} = \mathbb{E}\left[\phi_n(X(t))\, \phi_m(X(t))\right]$ is the covariance matrix. Or, for a given set of paths, we have

$$V(t)[p] = \frac{1}{N_p} \sum_{nm,p'} \phi_n(x[p]) C^{-1}_{nm} \phi_m(x[p']) V(T)[p']. \tag{46}$$

This expression gives us a *cross-term* $\partial V(t)[p]/\partial X(t)[p']$, which is a matrix with rank equal to the number of basis functions. Multiplications involving this matrix are slower than those performed during path-by-path differentiation, which involves *diagonal* derivatives $\partial V(t)[p]/\partial X(t)[p]$.

The cross-dependence on $X(t)[p']$ is simply due to the dependence of $\alpha_n$ on the various statistical characteristics of the states (e.g., on the average of $X(t)$ and the states' moments), which in turn depend on the model parameters:

$$\alpha_n = \alpha_n \left( \frac{1}{N_p} \sum_{p'} X(t)[p'], \frac{1}{N_p} \sum_{p'} X(t)^2[p'], \dots \right) = \alpha_n(\theta). \tag{47}$$

Therefore, if we know the derivatives $\partial \alpha_n / \partial X(t)[p']$ and the *entire* derivatives $\partial X(t)[p']/\partial\theta|_{t_0}$, we can calculate

$$\frac{\partial \alpha_n}{\partial \theta} = \sum_{p'} \frac{\partial \alpha_n}{\partial X(t)[p']} \left.\frac{\partial X(t)[p']}{\partial \theta}\right|_{t_0}. \tag{48}$$

Fortunately, the entire derivative is quite often known analytically. Even when this is not the case, it can be rapidly calculated by the pathwise AD procedure for the low cost found in (11).

This trick permits us to replace the previous CV derivatives storage of

$$\frac{\partial V(t)[p]}{\partial X(t)[p']} = \Delta(t)[p]\delta_{pp'} + \sum_{n,m=1}^{N_{V(t)}} A_n(t)[p] \Omega_{nm}(t) B_m(t)[p'] \quad \text{and} \quad \frac{\partial V(t)[p]}{\partial \theta}, \tag{49}$$

[5]According to our conventions we do not use the approximation sign here.

which contains high-rank cross derivatives, with

$$\frac{\partial V(t)[p]}{\partial X(t)[p']} = \Delta(t)[p]\delta_{pp'} \quad \text{and} \quad \frac{\partial V(t)[p]}{\partial \theta},$$

which contains modest diagonal derivatives. Essentially, the cross-path dependencies are pushed into the derivative over the parameters.

More specifically, the diagonal derivative is simply

$$\frac{\partial V(t)[p]}{\partial X(t)[p]} = V'(t)[p] \equiv \frac{1}{N_p} \sum_{nm,p'} \phi'_n(X[p])C^{-1}_{nm}\phi_m(X[p'])V(T)[p'].$$

It is shown in Appendix A that

$$\frac{\partial \alpha_n}{\partial X(t)[p']} = \frac{1}{N_p}\sum_m C^{-1}_{nm}\phi'_m(X(t)[p'])(V(T)[p'] - V(t)[p']) \\ + \frac{1}{N_p}\sum_m C^{-1}_{nm}\phi_m(X(t)[p'])\left(\frac{\partial V(T)[p']}{\partial X(T)[p']}\frac{\partial X(T)[p']}{\partial X(t)[p']} - \frac{\partial V(t)[p']}{\partial X(t)[p']}\right). \tag{50}$$

The derivative of $\alpha_n$ over $\theta$ is derived from the derivative over the states in (50) as well as the dependence of the values $V(T)$ on $\theta$ calculated at the previous time step:

$$\left.\frac{\partial \alpha_n}{\partial \theta}\right|_{t_0} = \frac{1}{N_p}\sum_{m,p'}\frac{\partial}{\partial \theta}\left(C^{-1}_{nm}\phi_m(X(t)[p''])\,V(T)[p'']\right) \tag{51}$$

$$= \frac{\partial \alpha_n}{\partial X(t)[p']}\left.\frac{\partial X(t)[p']}{\partial \theta}\right|_{t_0} + \frac{1}{N_p}\sum_{m,p''} C^{-1}_{nm}\phi_m(X(t)[p''])\,\frac{\partial V(T)[p'']}{\partial \theta}. \tag{52}$$

This gives the final CV derivative over the parameters as

$$\frac{\partial V(t)[p]}{\partial \theta} = \sum_n \phi_n(X(t)[p])\left.\frac{\partial \alpha_n}{\partial \theta}\right|_{t_0}, \tag{53}$$

which leads to

$$O(1)N_r \times N_p \times N_b \times N_\theta \tag{54}$$

multiplications in total.

One can apply the same technique directly to the CV derivatives. Namely, we take the initial storage (in matrix notation)

$$\frac{\partial V(t)}{\partial X(t)} = \Delta(t) + A(t)\Omega(t)B^T(t) \quad \text{and} \quad \frac{\partial V(t)}{\partial \theta} \tag{55}$$

and move the cross-term into the derivatives over the parameters, keeping only the diagonal; i.e., the new storage contains the diagonal $\partial V(t)/\partial X(t) = \Delta(t)$, while $\partial V/\partial\theta$ transforms as follows:

$$\frac{\partial V(t)}{\partial \theta} \to \frac{\partial V(t)}{\partial \theta} + A(t)\Omega(t)B^T(t)\left.\frac{\partial X(t)}{\partial \theta}\right|_{t_0}. \tag{56}$$

Other matrix reduction techniques can be based on various singular value decomposition compressions. Another approach is to utilize a GPU for the potentially large vector multiplications.

## 5.3 Storage

As we have mentioned, there is no instrument tape when performing the BD. Only the derivatives attached to each CV are stored, leading to a storage size requirement of

$$N_p \times (2N_X \times N_b + N_\theta)\,. \tag{57}$$

This is much smaller than the storage needed for the classic AD given in (29); in fact, it is smaller by a factor approximately equal to the number of the instrument dates $N_r$.

## 6 Adjoint differentiation vs. backward differentiation

Here we present the main differences and similarities of the two approaches.

- Speed

  Both methods are much faster than the bump-and-reprice approach. The gain comes from the fact that the bump-and-reprice method has to recalculate the regressions during each bumped run. This regression time (or "default" time) is mainly spent: 1) finding the regression basis, which is potentially dependent on the CV; 2) calculating the basis covariance matrix $C$; and 3) inverting $C$. Given a sufficiently large number of basis functions, the default time is significant and almost entirely determines the pricing time.

  As we will see in Section 7, the BD time is comparable with the default time for standard settings. In general, the BD is around 15–20 times faster than the bump-and-reprice method.

  Formally, the AD has $N_\theta$ times fewer multiplications than the BD; this can be seen by comparing (33) and (45). However, the CPU is also spent on writing and reading the tape during the AD, which results in its performance being comparable with the BD. We do not perform numerical experiments with the standard AD since our goal is to study tapeless methods of algorithmic differentiation, but we can estimate the AD time as being 4–10 times slower than the pricing using the common "rule of thumb".

  For all of the methods (the AD, the BD and the bump-and-reprice), the calculation time can be approximated by a linear function of the number of parameters,

  $$\alpha + \beta N_\theta, \tag{58}$$

  where $\alpha$ is a fixed time and $\beta$ is a proportional time. The bump-and-reprice has zero fixed time with $\beta$ almost equal to the pricing time. The AD fixed time is more than the pricing time by a factor of 4–10 (it mainly includes the tape preparation and writing) while its proportional time $\beta$ is tiny. The BD fixed time is fairly close to the pricing time, but its $\beta$ is bigger than the $\beta$ of the AD while remaining much smaller than the pricing time.

- Tape and storage

  The BD does not use the instrument tape, so its implementation is definitely easier. Moreover, the BD storage requirement is much smaller than that of the AD (by approximately the number of instrument dates).

- Multiple PVs

  The BD is designed to deliver model greeks for all of the PVs in the pricing script. The AD, on the other hand, should be rerun for each new PV.

- Coding/debug

  As we mentioned above, the AD implementation is complicated and can sometimes require the rewriting of whole libraries. Its tape structure adds a lot of complexity to the coding, debugging, and maintenance work. The BD implementation is much simpler and more compatible with the coding structure, especially given the underlying pricing script approach.

## 7 Numerical experiments

In this section, we numerically demonstrate the efficiency of the backward algorithmic differentiation using the HW1F model with constant parameters: the volatility is 2%, the reversion is 5%, and the rate is 2%.

The exotic instrument in hand is a 10-year Bermudan ATM swaption with coinciding quarterly fixing and exercise dates from 0.25 years to 10.25 years. The payment dates are shifted forward by 0.25 years, with the first occuring at 0.5 years and the last at 10.5 years. The swaption price is 0.107 for a unit notional. The pricing mechanism was described in the Section 3 and contains about 80 regressions.

We use four-step-per-year discretization, which results in 42 total simulation time steps. For our experiments, we calculate sensitivities to different intervals of the volatility and the discount factor[6]. Namely, we bump the discount factor on all simulation dates except the origin (42 times) while the volatility is bumped either on the annually spaced intervals (10 times) or on the simulation intervals (43 times). Thus, the first experiment has 52 parameters and the second one has 85 parameters.

[6]By the discount factor is this context we mean the df constructed from the yield curve, not $D(t)$ as being *path* discount factor.

We utilize (53) to apply the BD acceleration during each regression.

We record the calculation time for 1250, 2500, 5000, and 10000 high-quality, low-discrepancy MC paths. The number of basis functions depends linearly on the number of paths, with 10 basis functions being used for 1250 paths and 20 basis functions being used for 10000 paths.

In the tables below, we present the timing results (in seconds) for the calculations with 52 and 85 parameters.

| **Method** | **Number of Paths** | | | |
|---|---|---|---|---|
| | **1250** | **2500** | **5000** | **10000** |
| Pricing | 0.1 | 0.3 | 0.6 | 1.4 |
| Bump-and-reprice | 5.9 | 13.4 | 31.3 | 71.9 |
| BD | 0.3 | 0.9 | 2.1 | 5.6 |

Table 1: Timing results for 52 parameters (in seconds)

| **Method** | **Number of Paths** | | | |
|---|---|---|---|---|
| | **1250** | **2500** | **5000** | **10000** |
| Pricing | 0.1 | 0.3 | 0.6 | 1.4 |
| Bump-and-reprice | 9.7 | 21.9 | 51.1 | 117.5 |
| BD | 0.5 | 1.2 | 3.1 | 8.4 |

Table 2: Timing results for 85 parameters (in seconds).

The next table shows the ratio of the algorithmic differentiation time over the pricing time.

| **Method** | **Number of Paths** | | | |
|---|---|---|---|---|
| | **1250** | **2500** | **5000** | **10000** |
| BD for 52 parameters | 2.8 | 3.3 | 3.6 | 4.1 |
| BD for 85 parameters | 4.3 | 4.6 | 5.2 | 6.1 |

Table 3: Ratio of the BD time to the pricing time.

According to the adjoint differentiation rule of thumb (i.e., the AD is 4–10 times slower than the pricing) the BD performs very decently. In the table below we give the acceleration coefficient of the BD with respect to the bump-and-reprice method. Note that the BD is up to 20 times faster than the bump-and-reprice procedure.

| **Method** | **Number of paths** | | | |
|---|---|---|---|---|
| | **1250** | **2500** | **5000** | **10000** |
| BD for 52 parameters | 18.3 | 15.5 | 14.6 | 12.8 |
| BD for 85 parameters | 20.0 | 18.5 | 16.4 | 14.0 |

Table 4: Acceleration with respect to the bump-and-reprice method.

Analyzing the results for 2500 paths, we can roughly extrapolate the fixed and proportional times from the formula (58), obtaining a calculation time approximation of

$$0.4 + 0.01 N_\theta .$$

We note that the proportional time is tiny, approximately 30 times smaller then the pricing time.

As we mentioned in the introduction, the quality of the greeks calculated using algorithmic differentiation and the bump-and-reprice is often similar given a sufficiently large bump size. For smaller bumps, the finite difference derivatives calculation can lead to a bigger statistical error (i.e., a higher standard deviation over multiple independent Monte Carlo runs), but, potentially, a smaller bias. In Table 5 and Figure 1 we illustrate this comparison of quality using the example of the first model vega, which is the derivative over the model volatility on the first simulation interval [0, 0.25].

| **Paths** | **Mean over 25 runs** | | | **Standard deviation over 25 runs** | | |
|---|---|---|---|---|---|---|
| | **BD** | **Bump 1 bp** | **Bump 10 bp** | **BD** | **Bump 1 bp** | **Bump 10 bp** |
| 1250 | 0.370 | 0.383 | 0.383 | 0.023 | 0.032 | 0.023 |
| 2500 | 0.371 | 0.374 | 0.380 | 0.016 | 0.022 | 0.016 |
| 5000 | 0.373 | 0.378 | 0.384 | 0.009 | 0.013 | 0.009 |
| 10000 | 0.375 | 0.377 | 0.385 | 0.006 | 0.009 | 0.007 |

Table 5: Convergence analysis for the first model vega.

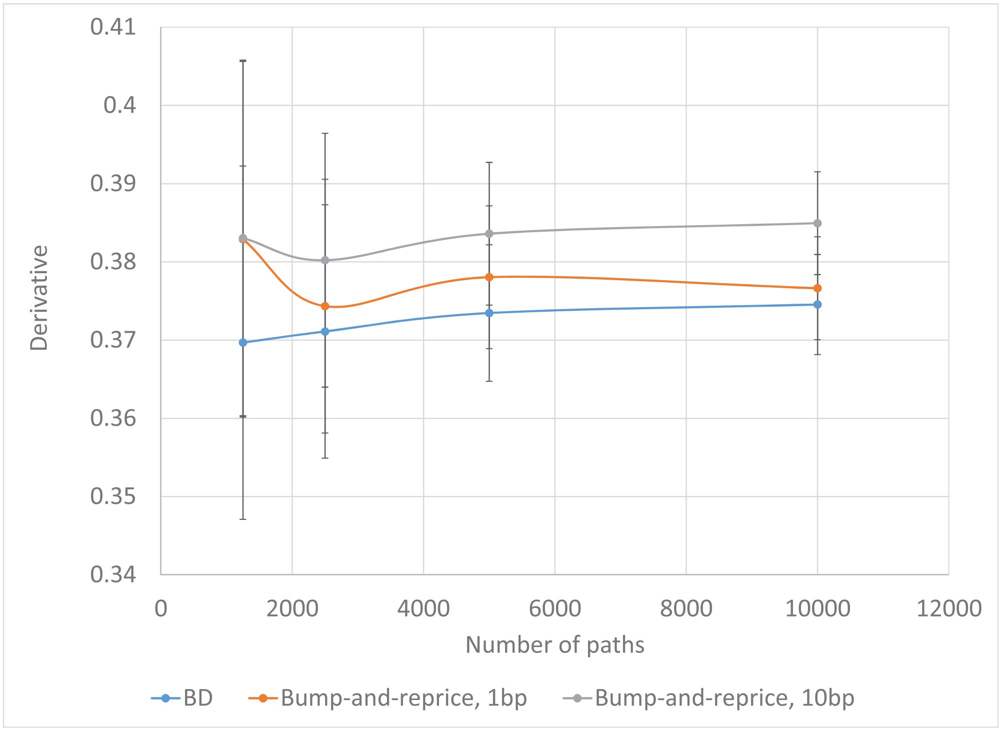

Figure 1: Convergence plot for the first model vega.

We observe that the standard deviations for the BD are similar to the bump-and-reprice using 10 basis points (bp) and smaller than the bump and reprice using 1 bp. On the other hand, there is a (small) bias for the 10 bp bump and almost an exact match between the 1bp bump and the BD.

Finally, we present the greeks profile (i.e., their mean and standard deviation over 25 runs) calculated for 2500 paths using the BD method in Figures 2 and 3. The corresponding tables, Table 6 and Table 7, also include the bump-and-reprice results and can be found in Appendix B. We use 0.001 bump for the discount factors in their bump-and-reprice.

We observe a small stochastic error for the vega profile and a fairly negligible error for the discount factor derivatives.

Interestingly, the swaption price is highly sensitive to the last discount factor. Indeed, if we enter to a swap at some date we will receive a set of fix-floating cashflows equivalent to 1 at this exercise date, `-strike*dcf` for all other fixing dates and, finally, -1 at the last day. Thus, the equivalent payment of -1 will be present whenever we exercise. Its derivative over the last discount factor is approximately the exercise probability. The first equivalent payment of 1 will be spread over the exercise dates, leading to small derivative over the intermediate discount factors.

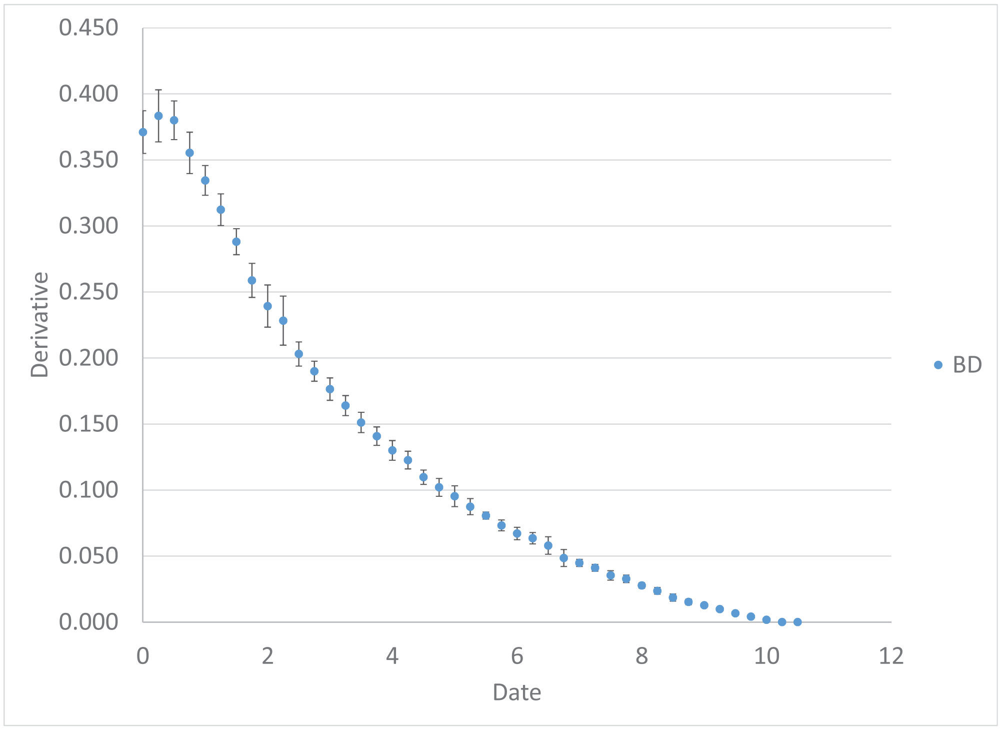

Figure 2: Derivatives profile over the model volatilities located between the indicated date and the following date. (The other parts of the model volatility are unchanged.)

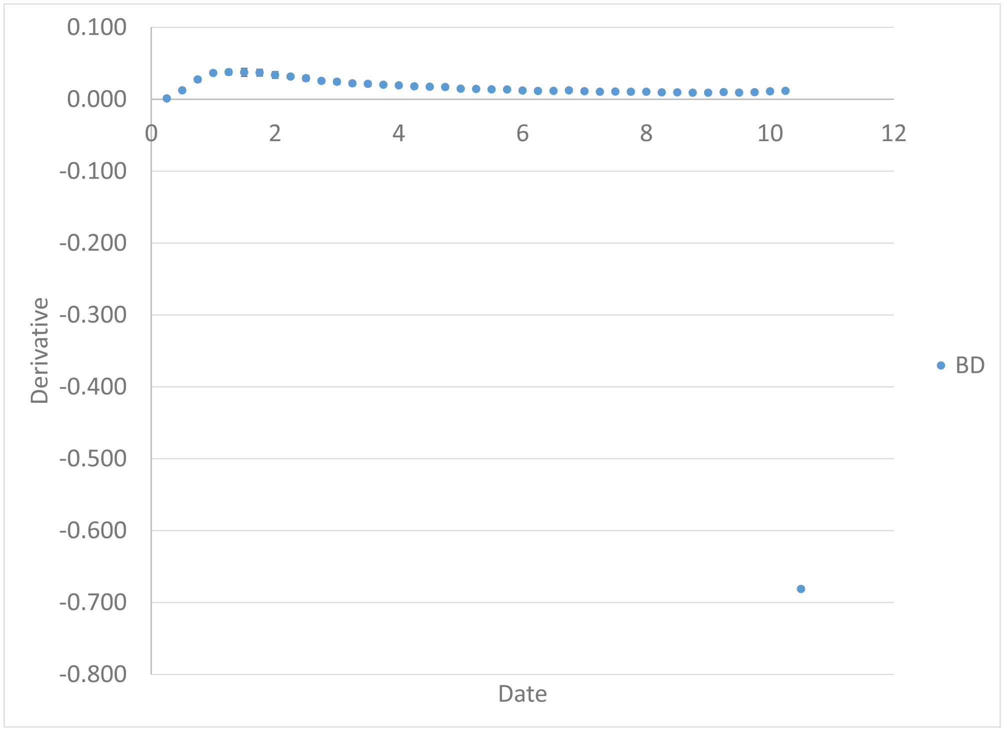

Figure 3: Derivatives profile over the model discount factor at the corresponding date. (The other dates' discount factors are unchanged).

## 8 Conclusion

We have studied the algorithmic calculation of model greeks for callable exotic instruments. Their price is obtained via the backward repetitive application of regression (using least-squares Monte Carlo) and arithmetic operations encoded in a pricing script. A direct application of the AD, the usual method for fast greeks calculation, is not straightforward because the regression mixes the paths. (The AD is traditionally applied for path-by-path simulations).

We described the application of the AD to general instruments being priced with a full regression. The procedure follows the usual steps: 1) preparation of a certain storage, called the instrument tape, during the backward pricing; 2) the subsequent playback of that tape that delivers the greeks. The tape adds a lot of complexity to the AD implementation, including possible memory issues (since the storage demands of the tape can be huge), debugging difficulties, etc.

In this article, we proposed a new *tapeless* differentiation method, called backward differentiation, for callable exotic deals. It is applied at the time of the pricing and, thus, completely avoids the tape and its related complications.

We have demonstrated the efficiency of the BD on the example of a Bermudan swaption calculated with the one-factor HW model. For 50–100 greeks, the BD calculation time is only 2–6 times slower than the pricing calculation.

The author is indebted to his colleagues at Numerix, especially to Gregory Whitten, Serguei Issakov, and Michael Konikov for supporting this work and Luke Mohr for the excellent editing. The author is thankful to Paul Glasserman for stimulating discussions.

## References

[1] Andreasen, J. CVA on an iPad Mini. Global Derivatives, ICBI, Amsterdam, 2014.

[2] Antonov, A., Issakov, S., and Mechkov, S. Backward induction for future values. *Risk Magazine* (January 2015), 92–97.

[3] Capriotti, L., and Giles, M. Adjoint greeks made easy. *Risk Magazine* (September 2012), 96–102.

[4] Capriotti, L., Giles, M., and Nauman, U. Algorithmic differentiation methodologies, 2016. Workshop at the Global Derivatives Conference, Budapest.

[5] Davidson, C. Structured products desks join the AAD revolution. *Risk Magazine* (July 2015).

[6] Fournié, E., Lasry, J., Lebuchoux, J., and Lions, P. Applications of Malliavin calculus to Monte-Carlo methods in finance. *Finance and Stochastics 3* (1999), 391–412.

[7] Giles, M., and Glasserman, P. Smoking adjoints: fast Monte Carlo Greeks. *Risk Magazine* (January 2006), 88–92.

[8] Sokol, A., and Ballabio, L. Retrofitting AAD to your existing C++ library, 2015. Workshop at the 11th Fixed Income Conference, Paris.

## A Regression derivatives

In this appendix we address details on the regression derivatives.

The derivative $\partial V^{(1)}/\partial \tilde{V}^{(2)}$ is a direct consequence of (25):

$$\frac{\partial V^{(1)}[p]}{\partial \tilde{V}^{(2)}[p']} = \frac{1}{N_p} \sum_{nm} \phi_n(x[p]) C^{-1}_{nm} \phi_m(x[p']) D^{(1)}[p']. \tag{59}$$

Now let us concentrate of the derivatives over the state and the discount factor. Noticing that

$$\frac{\partial C_{km}}{\partial x[p]} = \frac{1}{N_p} \phi'_k(x[p]) \phi_m(x[p]) D^{(1)}[p] + \frac{1}{N_p} \phi_k(x[p]) \phi'_m(x[p]) D^{(1)}[p], \tag{60}$$

$$\frac{\partial C_{km}}{\partial D^{(1)}[p]} = \frac{1}{N_p} \phi_k(x[p]) \phi_m(x[p]), \tag{61}$$

and

$$\partial C^{-1} = -C^{-1} \partial C \, C^{-1}, \tag{62}$$

we obtain

$$\begin{aligned}
\frac{\partial V^{(1)}[p]}{\partial x[p']} &= \delta_{pp'}\frac{1}{N_p}\sum_{nm,p''}\phi'_n(x[p])C^{-1}_{nm}\phi_m(x[p''])\tilde{V}^{(2)}[p'']D^{(1)}[p''] + \frac{1}{N_p}\sum_{nm}\phi_n(x[p])C^{-1}_{nm}\phi'_m(x[p'])\tilde{V}^{(2)}[p']D^{(1)}[p'] \\
&\qquad - \frac{1}{N_p}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\frac{\partial C_{km}}{\partial x[p']}C^{-1}_{ml}\phi_l(x[p''])\tilde{V}^{(2)}[p'']D^{(1)}[p''] \\
&= \delta_{pp'}\,{V^{(1)}}'[p] + \frac{1}{N_p}\sum_{nm}\phi_n(x[p])C^{-1}_{nm}\phi'_m(x[p'])V^{(2)}[p'] \\
&\qquad - \frac{1}{N_p^2}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\left(\phi'_k(x[p'])\phi_m(x[p']) + \phi_k(x[p'])\phi'_m(x[p'])\right)C^{-1}_{ml}\phi_l(x[p''])V^{(2)}[p''] \\
&= \delta_{pp'}\,{V^{(1)}}'[p] + \frac{1}{N_p}\sum_{nm}\phi_n(x[p])C^{-1}_{nm}\phi'_m(x[p'])V^{(2)}[p'] \\
&\qquad - \frac{1}{N_p^2}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\phi'_k(x[p'])\phi_m(x[p'])C^{-1}_{ml}\phi_l(x[p''])V^{(2)}[p''] \\
&\qquad - \frac{1}{N_p^2}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\phi_k(x[p'])\phi'_m(x[p'])C^{-1}_{ml}\phi_l(x[p''])V^{(2)}[p''] \\
&= \delta_{pp'}{V^{(1)}}'[p] + \frac{1}{N_p}\sum_{nm}\phi_n(x[p])C^{-1}_{nm}\phi'_m(x[p'])V^{(2)}[p'] - \frac{1}{N_p}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\phi'_k(x[p'])V^{(1)}[p'] \\
&\qquad - \frac{1}{N_p}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\phi_k(x[p']){V^{(1)}}'[p'] \\
&= \delta_{pp'}{V^{(1)}}'[p] + \frac{1}{N_p}\sum_{nm}\phi_n(x[p])C^{-1}_{nm}\left\{\phi'_m(x[p'])(\tilde{V}^{(2)}[p'] - V^{(1)}[p']) - \phi_m(x[p']){V^{(1)}}'[p']\right\}D^{(1)}[p'],
\end{aligned}$$

where

$${V^{(1)}}'[p] = \frac{1}{N_p}\sum_{nm,p'}\phi'_n(x[p])C^{-1}_{nm}\phi_m(x[p'])\tilde{V}^{(2)}[p']D^{(1)}[p'].$$

For the derivative over the DF we have

$$\begin{aligned}
\frac{\partial V^{(1)}[p]}{\partial D^{(1)}[p']} &= \frac{1}{N_p}\sum_{nm,p'}\phi_n(x[p])C^{-1}_{nm}\phi_m(x[p'])\tilde{V}^{(2)}[p'] \\
&\qquad - \frac{1}{N_p}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\frac{\partial C_{km}}{\partial D^{(1)}[p']}C^{-1}_{ml}\phi_l(x[p''])V^{(2)}[p'']D_2[p''] \\
&= \frac{1}{N_p}\sum_{nm,p'}\phi_n(x[p])C^{-1}_{nm}\phi_m(x[p'])\tilde{V}^{(2)}[p'] \\
&\quad - \frac{1}{N_p}\sum_{nmkl,p''}\phi_n(x[p])C^{-1}_{nk}\frac{1}{N_p}\phi_k(x[p'])\phi_m(x[p'])C^{-1}_{ml}\phi_l(x[p''])V^{(2)}[p'']D_2[p''] \\
&= \frac{1}{N_p}\sum_{nk}\phi_n(x[p])C^{-1}_{nk}\phi_k(x[p'])(\tilde{V}^{(2)}[p'] - V^{(1)}[p']).
\end{aligned}$$

Then, we pass to derivatives of the regression coefficients

$$\alpha_n = \frac{1}{N_p}\sum_{m,p''}C^{-1}_{nm}\phi_m(x[p''])V(T)[p''] \tag{63}$$

over $x[p']$ and $\theta$. Indeed,

$$\begin{aligned}
\frac{\partial \alpha_n}{\partial X(t)[p']} &= \frac{1}{N_p}\sum_{m,p'} \frac{\partial}{\partial X(t)[p']}\left(C^{-1}_{nm}\phi_m(X(t)[p''])V(T)[p'']\right) \\
&= \frac{1}{N_p}\sum_{m,p''} \frac{\partial}{\partial X(t)[p']}\left(C^{-1}_{nm}\phi_m(X(t)[p''])\right)V(T)[p''] + \frac{1}{N_p}\sum_{m,p''}\frac{\partial V(T)[p'']}{\partial X(t)[p']}C^{-1}_{nm}\phi_m(X(t)[p'']) \\
&= \frac{1}{N_p}\sum_{m} C^{-1}_{nm}\left\{\phi'_m(X(t)[p'])(V(T)[p'] - V(t)[p']) - \phi_m(X(t)[p'])\frac{\partial V(t)[p']}{\partial X(t)[p']}\right\} \\
&\qquad\qquad\qquad\qquad + \frac{1}{N_p}\sum_{m}\frac{\partial V(T)[p']}{\partial x(T)[p']}\frac{\partial x(T)[p']}{\partial X(t)[p']}C^{-1}_{nm}\phi_m(X(t)[p']) \\
&= \frac{1}{N_p}\sum_{m} C^{-1}_{nm}\phi'_m(X(t)[p'])(V(T)[p'] - V(t)[p']) \\
&\qquad\qquad + \frac{1}{N_p}\sum_{m} C^{-1}_{nm}\phi_m(X(t)[p'])\left(\frac{\partial V(T)[p']}{\partial x(T)[p']}\frac{\partial x(T)[p']}{\partial X(t)[p']} - \frac{\partial V(t)[p']}{\partial X(t)[p']}\right).
\end{aligned}$$

## B Derivatives profiles

| Date | Mean over 25 runs | | | Standard deviation over 25 runs | | |
|---|---|---|---|---|---|---|
| | **BD** | **Bump 1 bp** | **Bump 10 bp** | **BD** | **Bump 1 bp** | **Bump 10 bp** |
| 0.00 | 0.371 | 0.374 | 0.380 | 0.016 | 0.022 | 0.016 |
| 0.25 | 0.383 | 0.383 | 0.395 | 0.020 | 0.025 | 0.016 |
| 0.50 | 0.380 | 0.381 | 0.390 | 0.015 | 0.020 | 0.017 |
| 0.75 | 0.355 | 0.357 | 0.366 | 0.016 | 0.026 | 0.018 |
| 1.00 | 0.335 | 0.335 | 0.342 | 0.011 | 0.020 | 0.012 |
| 1.25 | 0.312 | 0.317 | 0.322 | 0.012 | 0.021 | 0.014 |
| 1.50 | 0.288 | 0.285 | 0.296 | 0.010 | 0.017 | 0.011 |
| 1.75 | 0.259 | 0.260 | 0.265 | 0.013 | 0.021 | 0.014 |
| 2.00 | 0.239 | 0.236 | 0.244 | 0.016 | 0.019 | 0.015 |
| 2.25 | 0.228 | 0.225 | 0.233 | 0.019 | 0.020 | 0.019 |
| 2.50 | 0.203 | 0.206 | 0.207 | 0.009 | 0.016 | 0.009 |
| 2.75 | 0.190 | 0.191 | 0.194 | 0.008 | 0.016 | 0.009 |
| 3.00 | 0.176 | 0.177 | 0.179 | 0.008 | 0.013 | 0.009 |
| 3.25 | 0.164 | 0.161 | 0.168 | 0.008 | 0.015 | 0.008 |
| 3.50 | 0.151 | 0.152 | 0.154 | 0.008 | 0.015 | 0.007 |
| 3.75 | 0.141 | 0.140 | 0.144 | 0.007 | 0.011 | 0.008 |
| 4.00 | 0.130 | 0.130 | 0.133 | 0.007 | 0.014 | 0.008 |
| 4.25 | 0.123 | 0.123 | 0.125 | 0.007 | 0.015 | 0.007 |
| 4.50 | 0.110 | 0.113 | 0.112 | 0.005 | 0.011 | 0.006 |
| 4.75 | 0.102 | 0.102 | 0.104 | 0.007 | 0.013 | 0.008 |
| 5.00 | 0.095 | 0.094 | 0.097 | 0.008 | 0.011 | 0.008 |
| 5.25 | 0.087 | 0.089 | 0.088 | 0.006 | 0.009 | 0.006 |
| 5.50 | 0.081 | 0.082 | 0.082 | 0.003 | 0.009 | 0.004 |
| 5.75 | 0.073 | 0.073 | 0.075 | 0.004 | 0.006 | 0.005 |
| 6.00 | 0.067 | 0.068 | 0.069 | 0.005 | 0.010 | 0.005 |
| 6.25 | 0.064 | 0.063 | 0.065 | 0.004 | 0.007 | 0.005 |
| 6.50 | 0.058 | 0.058 | 0.059 | 0.007 | 0.010 | 0.006 |
| 6.75 | 0.049 | 0.048 | 0.050 | 0.006 | 0.009 | 0.007 |
| 7.00 | 0.045 | 0.044 | 0.046 | 0.003 | 0.006 | 0.003 |
| 7.25 | 0.041 | 0.041 | 0.042 | 0.003 | 0.005 | 0.004 |
| 7.50 | 0.035 | 0.035 | 0.036 | 0.004 | 0.006 | 0.004 |
| 7.75 | 0.033 | 0.031 | 0.034 | 0.003 | 0.005 | 0.003 |
| 8.00 | 0.028 | 0.028 | 0.029 | 0.002 | 0.003 | 0.002 |
| 8.25 | 0.024 | 0.023 | 0.024 | 0.003 | 0.004 | 0.002 |
| 8.50 | 0.019 | 0.019 | 0.019 | 0.003 | 0.004 | 0.003 |
| 8.75 | 0.015 | 0.015 | 0.016 | 0.002 | 0.003 | 0.002 |
| 9.00 | 0.013 | 0.012 | 0.013 | 0.001 | 0.001 | 0.001 |
| 9.25 | 0.010 | 0.010 | 0.010 | 0.001 | 0.001 | 0.001 |
| 9.50 | 0.007 | 0.007 | 0.007 | 0.000 | 0.001 | 0.001 |
| 9.75 | 0.004 | 0.004 | 0.004 | 0.000 | 0.000 | 0.000 |
| 10.00 | 0.002 | 0.002 | 0.002 | 0.000 | 0.000 | 0.000 |
| 10.25 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10.50 | 0.000 | 0.000 | | 0.000 | 0.000 | |

Table 6: Derivatives over the model volatilities located between the indicated date and the following date. (The other parts of the model volatility are unchanged.)

| Date | Mean over 25 runs | | Standard deviation over 25 runs | |
|---|---|---|---|---|
| | **BD** | **Bump** | **BD** | **Bump** |
| 0.25 | 0.001 | 0.001 | 0.001 | 0.001 |
| 0.50 | 0.012 | 0.014 | 0.002 | 0.003 |
| 0.75 | 0.027 | 0.029 | 0.003 | 0.003 |
| 1.00 | 0.036 | 0.039 | 0.003 | 0.003 |
| 1.25 | 0.038 | 0.042 | 0.004 | 0.006 |
| 1.50 | 0.037 | 0.040 | 0.006 | 0.006 |
| 1.75 | 0.037 | 0.040 | 0.005 | 0.004 |
| 2.00 | 0.034 | 0.036 | 0.005 | 0.004 |
| 2.25 | 0.032 | 0.034 | 0.003 | 0.003 |
| 2.50 | 0.029 | 0.031 | 0.004 | 0.004 |
| 2.75 | 0.026 | 0.028 | 0.003 | 0.004 |
| 3.00 | 0.024 | 0.027 | 0.003 | 0.003 |
| 3.25 | 0.022 | 0.025 | 0.003 | 0.003 |
| 3.50 | 0.021 | 0.024 | 0.003 | 0.003 |
| 3.75 | 0.020 | 0.022 | 0.002 | 0.003 |
| 4.00 | 0.019 | 0.021 | 0.002 | 0.002 |
| 4.25 | 0.018 | 0.019 | 0.002 | 0.002 |
| 4.50 | 0.017 | 0.019 | 0.003 | 0.003 |
| 4.75 | 0.017 | 0.019 | 0.003 | 0.002 |
| 5.00 | 0.015 | 0.017 | 0.002 | 0.002 |
| 5.25 | 0.014 | 0.016 | 0.002 | 0.002 |
| 5.50 | 0.014 | 0.015 | 0.002 | 0.002 |
| 5.75 | 0.014 | 0.015 | 0.002 | 0.002 |
| 6.00 | 0.012 | 0.014 | 0.002 | 0.002 |
| 6.25 | 0.011 | 0.013 | 0.002 | 0.002 |
| 6.50 | 0.012 | 0.014 | 0.002 | 0.002 |
| 6.75 | 0.012 | 0.015 | 0.002 | 0.002 |
| 7.00 | 0.011 | 0.013 | 0.001 | 0.002 |
| 7.25 | 0.010 | 0.012 | 0.002 | 0.002 |
| 7.50 | 0.011 | 0.012 | 0.001 | 0.002 |
| 7.75 | 0.010 | 0.012 | 0.002 | 0.002 |
| 8.00 | 0.010 | 0.012 | 0.002 | 0.002 |
| 8.25 | 0.010 | 0.012 | 0.001 | 0.001 |
| 8.50 | 0.010 | 0.012 | 0.002 | 0.001 |
| 8.75 | 0.009 | 0.012 | 0.002 | 0.002 |
| 9.00 | 0.009 | 0.012 | 0.001 | 0.002 |
| 9.25 | 0.010 | 0.013 | 0.002 | 0.002 |
| 9.50 | 0.009 | 0.012 | 0.001 | 0.001 |
| 9.75 | 0.010 | 0.014 | 0.001 | 0.001 |
| 10.00 | 0.011 | 0.015 | 0.001 | 0.001 |
| 10.25 | 0.012 | 0.019 | 0.001 | 0.001 |
| 10.50 | -0.681 | -0.676 | 0.001 | 0.001 |

Table 7: Derivatives over the model discount factor at the corresponding date. (The other dates' discount factors are unchanged.)