Real-Time Risk Management: An AAD-PDE Approach

Luca Capriotti[⋆†], Yupeng Jiang[†], Andrea Macrina[†‡]

[⋆] Quantitative Strategies, Investment Banking Division, Credit Suisse Group,
One Cabot Square, London E14 4QJ, United Kingdom

[†] Department of Mathematics, University College London,
London WC1E 6BT, United Kingdom

[‡] Department of Actuarial Science, University of Cape Town,
Rondebosch 7701, South Africa

June 22, 2015

We apply adjoint algorithmic differentiation (AAD) to the risk management of derivative securities in the situation where the dynamics of securities prices are given in terms of partial differential equations (PDE). With simple examples, we show how AAD can be applied to both forward and backward PDEs in a straightforward manner. In particular, in the context of one-factor short-rate models for interest rates or default intensity processes, we show how one can compute price sensitivities reliably and orders of magnitude faster than with a standard finite-difference approach. This significantly increased efficiency is obtained by combining (i) the adjoint of a forward PDE for calibrating the parameters of the model, (ii) the adjoint of a backward PDE for pricing the derivative security, and (iii) the implicit function theorem to avoid iterating the calibration procedure.

1 Introduction

Risk management of derivatives securities comes with a high computational burden and technology cost. This has become especially true in the aftermath of the financial crisis because of the renewed emphasis in sound risk management practices and the introduction of a large number of valuation adjustments, collectively known as XVA, see Crépey et al. (2014). These adjustments aim at capturing counterparty risk and other funding and capital costs that, while not accounted for in classical pricing theory, significantly affect the profitability of trading operations. XVAs are computationally expensive to obtain because they typically involve the simultaneous valuation of large sets of trades, see Green (2015).

Standard approaches for the calculation of risk require repeated portfolio valuation under hundreds of market scenarios. As a result, in order to complete risk calculations in practical time spans, financial firms employ vast computational resources bearing high infrastructure

costs. Since the total cost of "through-the-life" risk management can determine whether it is profitable to execute a new trade, solving such technological challenge is of paramount importance for a securities firm to remain competitive.

In this backdrop, a computational technique known as "Adjoint Algorithmic Differentiation" (AAD) was recently introduced in Capriotti (2011), Capriotti et al. (2011), Capriotti and Giles (2010a,b). It has been proven to be effective for speeding up the calculation of sensitivities, especially for Monte Carlo applications. This powerful technique allows for fast computation of first-order sensitivities without repeating the portfolio valuation multiple times as in traditional finite-difference approaches.

"Algorithmic Differentiation" (AD), see Griewank (2000) and Naumann (2012), is a scheme for the efficient calculation of derivatives of functions, which are implemented as computer programs. What makes AD particularly attractive, when compared to standard finite-difference methods for the calculation of derivatives, is its computational efficiency. AD exploits the information on the structure of the computer code in order to optimise the calculation. In particular, when one requires the derivatives of a small number of outputs with respect to a large number of inputs, the calculation can be optimised by applying the standard chain rule of calculus through the instructions of the program in opposite order with respect to their original evaluation. This gives rise to the Adjoint (mode of) Algorithmic Differentiation (AAD).

Most of the applications considered in the financial literature so far have focused on Monte Carlo simulations. In that context, AAD can be used to implement efficiently the so-called Pathwise Derivative Method, see Broadie and Glasserman (1996) and Glasserman (2004), for the calculation of sensitivities. In this paper, we extend this line of research and demonstrate how AAD can be extremely effective for applications with partial differential equations (PDE). We show how AAD can be utilised to speed up the calculation of the sensitivities in situations where the pricing of the derivative and the calibration of the underlying stochastic model rely on solving PDEs, multiple times. We shall show how one can compute price sensitivities more reliably and orders of magnitude faster than with standard finite-difference approaches. This will be achieved with a judicious combination of the adjoint version of the numerical schemes for forward and backward PDEs, see Andersen and Piterbarg (2010), and by the so-called implicit function theorem.

This paper is organised as follows: In the next section we begin by recalling the standard formalism for the valuation of securities prices by means of PDEs. As an example, we shall focus our discussion on one-factor short-rate models, which are ubiquitous in financial practice for the pricing and risk management of interest rate and credit derivatives. Here we recall the standard numerical approach for the numerical solution of backward PDEs while in Section 3 we review the equivalent forward PDE approach and its use for the efficient implementation of calibration algorithms. Section 4 presents the general principles of AAD and their application to the numerical solution of the forward and backward PDEs. The results of numerical experiments are found in Section 5. We conclude with Section 6 where we summarise the main contributions of this work.

2 Option prices and backward PDEs

Option pricing problems can be often formulated in terms of a linear parabolic partial differential equation of second order of the form

$$\frac{\partial V}{\partial t} + \mu(x,t;\theta)\,\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2(x,t;\theta)\,\frac{\partial^2 V}{\partial x^2} - \nu(x,t;\theta)\,V = 0, \tag{1}$$

where

$$V_t(\theta) = V(x_t,t;\theta) \equiv \mathbb{E}\left[\exp\left(-\int_t^T \nu(x_u,u;\theta)\mathrm{d}u\right) P(x_T;\theta)\,\middle|\, x_t\right] \tag{2}$$

is the value of a derivative contract at time t; see, e.g., Wilmott (2007). The expectation is taken under a suitable probability measure, depending on the financial context, given the value of a state variable x_t at time $t \geq 0$. At the maturity date $T > t$, the value $P(x_T;\theta)$ of the financial derivative depends on the realisation of the underlying risk factor $\{x_t\}_{0\leq t}$ that satisfies

$$\mathrm{d}x_t = \mu(x_t,t;\theta)\mathrm{d}t + \sigma(x_t,t;\theta)\mathrm{d}W_t \tag{3}$$

where $\mu(x,t;\theta)$ and $\sigma(x,t;\theta)$ are the drift and the volatility functions, and $\{W_t\}_{0\leq t}$ is a one-dimensional Brownian motion. Here and in the following, $\theta = (\theta_1,\ldots,\theta_{N_\theta})$ represents the vector of N_θ parameters the model is dependent on. By supplying appropriate spatial boundary conditions, see Iserles (2009) and Wilmott (2007), and the terminal condition

$$V(x,T;\theta) = P(x;\theta) \tag{4}$$

at maturity T, Equation (1) can be solved backwards in time for the value $V(x,t;\theta)$ of the derivative security at any time $t \leq T$.

The Black-Scholes PDE for the price of European-style claims is of the form (1) where $\mu(x,t;\theta) = (r(t) - \delta(t))\,x$, $\sigma(x,t;\theta) = \sigma(t)x$ and $\nu(x,t;\theta) = r(t)$. Here $r(t)$ and $\delta(t)$ denote the (deterministic) risk-free interest rate and dividend yield, respectively.

One-factor short-rate models for applications to interest-rate derivatives pricing, such as the models by Hull and White (1996), Cox et al. (1985), Black et al. (1990), or by Black and Karasinski (1991) can also be described in terms of a random driver that satisfies a diffusion of the form (3). For example, the Black-Karasinski (BK) model can be expressed by setting the (stochastic) instantaneous rate of interest to $r_t \equiv r(x_t) = \exp(x_t)$ where the process $\{x_t\}_{0\leq t}$ satisfies

$$\mathrm{d}x_t = \kappa(t)\,(\mu(t) - x_t)\,\mathrm{d}t + \sigma(t)\mathrm{d}W_t, \tag{5}$$

and where $\kappa(t)$ and $\mu(t)$ are the mean-reversion speed and level, respectively. In the context of short-rate models, the value of a derivative security with expiry value $P(r_T;\theta)$ can be expressed as in (2) with $\nu(x_u,u;\theta) = r(x_u)$ where the expectation is taken under the risk-neutral measure. In this case the components of the vector θ are typically the coefficients used to parameterise the mean-reversion speed and level, and the volatility of the process.

Since the BK-model is a positive stochastic process, it can be applied for the modelling of stochastic default intensity, which is known as Cox process, see O'Kane (2011). In this

context, $x_t = \ln(h_t)$ in Equation (5) is the logarithm of the *hazard rate* $\{h_t\}_{0\leq t}$, which represents the (risk-neutral) default probability per unit of time of a reference (financial) entity between times t and $t + dt$, conditional on survival up to time t. By modelling the default event of an obligor as the first arrival time τ of a Poisson process, the conditional (risk-neutral) probability of the obligor surviving up to time T is given by

$$Q(h_t, t, T) = \mathbb{E}\left[\exp\left(-\int_t^T h_u \, \mathrm{d}u\right) \Big| \, h_t, \tau > t\right]. \tag{6}$$

Any credit derivative of which payoff at time T is a function of the hazard rate h_T, such as defaultable bonds, credit default swaps (CDS), bond options and CDS options, can be priced with the backward PDE (1).

2.1 Numerical solutions by finite-difference discretisation for backward PDEs

The solution $V_{t_0}(\theta) = V(x_{t_0}, t_0; \theta)$ of the PDE (1) can be found numerically by discretisation on the rectangular domain $(t, x) \in [t_0, T] \times [x_{\min}, x_{\max}]$ where $x_{\min}$ and $x_{\max}$ (such that $x_{\min} < x_{t_0} < x_{\max}$) are constants obtained by means of probabilistic considerations, see Wilmott (2007). In particular, by denoting (i) the points on the time axis by $t_m = t_0 + m\Delta t$ where $m = 0, \ldots, M$ and $\Delta t = (T - t_0)/M$, and (ii) the points on the spatial axis by $x_j = x_{\min} + j\Delta x$, where $j = 0, \ldots, N+1$ and $\Delta x = (x_{\max} - x_{\min})/(N+1)$, one can discretise the PDE (1) with finite-difference approximations for the first and second derivatives. A standard discretisation scheme, see Andersen and Piterbarg (2010) and Wilmott (2007), results in a matrix iteration of the form

$$L_B(t_m, \phi; \theta)V^m(\theta) = R_B(t_m, \phi; \theta)V^{m+1}(\theta) + \beta(t_{m+1}; \theta) \tag{7}$$

where $V^m(\theta) = (V(x_1, t_m; \theta), \ldots, V(x_N, t_m; \theta))^\top$ and $V(x_j, t_m; \theta)$ indicate the finite-difference approximation to the solution of the PDE (1)[1]. Here, we introduce the $N \times N$ tri-diagonal matrices

$$L_B(t_m, \phi; \theta) = \mathbb{I} - \phi\Delta t D(\tilde{t}_m(\phi); \theta), \tag{8}$$
$$R_B(t_m, \phi; \theta) = \mathbb{I} + (1 - \phi)\Delta t D(\tilde{t}_m(\phi); \theta), \tag{9}$$

where $\tilde{t}_m(\phi) = (1 - \phi)t_m + \phi\, t_m$. Both expressions are defined in terms of the tri-diagonal matrix $D(t; \theta)$ given by

$$\begin{aligned} [D(t;\theta)]_{j,j} &= c_j(t;\theta), \\ [D(t;\theta)]_{j,j+1} &= u_j(t;\theta), \\ [D(t;\theta)]_{j+1,j} &= l_{j+1}(t;\theta), \end{aligned} \tag{10}$$

[1]To keep the notation as light as possible, we denote the exact solution of the PDE (1) and its finite-difference approximation with the same symbol.

where $j = 1, \ldots, N$ in the first equation and $j = 1, \ldots, N-1$ for the second and third. The coefficients

$$\begin{aligned} c_j(t;\theta) &= -\sigma(x_j,t;\theta)^2\Delta x^{-2} - \nu(x_j,t;\theta), \\ u_j(t;\theta) &= \frac{1}{2}\mu(x_j,t;\theta)\Delta x^{-1} + \frac{1}{2}\sigma(x_j,t;\theta)^2\Delta x^{-2}, \\ l_j(t;\theta) &= -\frac{1}{2}\mu(x_j,t;\theta)\Delta x^{-1} + \frac{1}{2}\sigma(x_j,t;\theta)^2\Delta x^{-2}, \end{aligned} \tag{11}$$

are defined in terms of the functions $\mu(x,t;\theta)$, $\sigma(x,t;\theta)$ and $\nu(x_j,t;\theta)$ in the PDE (1). Moreover, $\beta(t_{m+1};(\theta))$ is an N-dimensional vector encoding suitable spatial boundary conditions which cannot be included in the matrix $D(t;\theta)$. The parameter ϕ is bounded between $\phi = 0$, corresponding to the fully explicit scheme, and $\phi = 1$, corresponding to the fully implicit scheme. Both schemes are characterised by an accuracy $O(\Delta x^2, \Delta t)$. The case $\phi = 1/2$ corresponds to the Cranck-Nicholson (CN) method, see Crank and Nicolson (1947), which is generally the method of choice in financial applications because it is characterised by an accuracy $O(\Delta x^2, \Delta t^2)$ as well as it being unconditionally stable. However, in some situations, e.g., for discontinuous payoff functions, combining the CN method with fully-implict iterations (as in the so-called "Rannacher stepping") has been shown to improve the accuracy of the numerical solution, see Pooley et al. (2003).

Given the value of the derivative at maturity $V_j^M(\theta) = P(x_j;\theta)$, Equation (7) can be recursively solved, by utilising standard tri-diagonal solvers (e.g., based on the LU-decomposition found in Wilmott (2007)) for $m = M-1, \ldots, 0$, in order to find the vector $V_j^0(\theta)$. From this, the value of the derivative $V_{t_0} = V(x_{t_0}, t_0;\theta)$, corresponding to the state variable (or spot value) x_{t_0} observed at time t_0, can be computed by means of, e.g., linear interpolation,

$$V_{t_0} = V_{j^\star}^0 + \frac{V_{j^\star+1}^0 - V_{j^\star}^0}{x_{j^\star+1} - x_{j^\star}}(x_{t_0} - x_{j^\star}), \tag{12}$$

with $j^\star$ such that $x_{j^\star} \leq x_{t_0} < x_{j^\star+1}$. The associated algorithm can be described by the following steps:

(S1) Initialise the value vector on the final time slice $V_j^M(\theta) = P(x_j;\theta)$ with $j = 0, \ldots, N$:

$$V^M = \texttt{PAYOFF}(\theta). \tag{13}$$

(S2) For $m = M-1, \ldots, 0$ execute the following steps:

a) Compute the coefficient vectors $c^m(\theta) \equiv c(\tilde{t}_m(\phi);\theta)$, $u^m(\theta) \equiv u(\tilde{t}_m(\phi);\theta)$, and $l^m(\theta) \equiv l(\tilde{t}_m(\phi);\theta)$ in Equations (11):

$$(c^m, u^m, l^m) = \texttt{COMPUTECOEFFM}(\theta). \tag{14}$$

b) Compute the matrices $L_B^m(\theta) \equiv L_B(t_m, \phi;\theta)$ and $R_B^m(\theta) \equiv R_B(t_m, \phi;\theta)$ in Equations (8) and (9) from the coefficients vectors $c^m(\theta)$, $u^m(\theta)$, and $l^m(\theta)$:

$$(L_B^m, R_B^m) = \texttt{COMPUTELRB}(c^m, u^m, l^m). \tag{15}$$

c) Compute the boundary condition vector $\beta^{m+1}(\theta) \equiv \beta(t_m;\theta)$ by

$$\beta^{m+1} = \texttt{COMPUTEBC}(\theta). \tag{16}$$

d) Given V^{m+1}, solve Equation (7) for V^m by calling a suitable tri-diagonal solver such as

$$V^m = \texttt{TRIDIAGSOLVER}(L_B^m, R_B^m, \beta^{m+1}, V^{m+1}), \tag{17}$$

which we can represent mathematically as the following sequence of operations:

$$\begin{aligned} U^{m+1} &= R_B^m V^{m+1}, \\ W^{m+1} &= U^{m+1} + \beta^{m+1}, \\ V^m &= W^{m+1}/L_B^m, \end{aligned} \tag{18}$$

where we adopte the notation "B/A" to represent finding the solution X of the linear system $AX = B$.

(S3) Compute $V_{t_0} = V(x_{t_0}, t_0; \theta)$ with a suitable interpolation scheme, e.g., the scheme of (12), by calling a method of the kind

$$V_{t_0} = \texttt{COMPUTESPOTVALUE}(V^0), \tag{19}$$

Since the matrix (10) is tri-diagonal, the cost of a single iteration of Equation (7) is $O(N)$. As a result, the overall computation complexity of the algorithm above is $O(NM)$.

2.2 Intermediate cashflows

Incorporating intermediate cash flows, which for instance may arise from coupon payments, in the finite-difference scheme is immediate and results in the following modification of the second step (S2) in the previous section:

e) Initialise any additional payoff that might be necessary for the valuation of intermediate cash-flows $C(t, x_t; \theta)$, when their value is not available in closed form, by

$$\mathcal{C}_{m+1}^{m+1} = \texttt{AUXILIARYPAYOFF}(\theta). \tag{20}$$

Here $\mathcal{C}_k^m$ is the value vector at time t_m of a set of auxiliary securities with expiry t_k.

f) For $k = m+1, \ldots, M$ execute the tri-diagonal solver

$$\mathcal{C}_k^m = \texttt{TRIDIAGSOLVER}\left(L_B^m, R_B^m, \beta^{m+1}, \mathcal{C}_k^{m+1}\right). \tag{21}$$

g) Compute the intermediate cash-flow at time t_m and update the value vector

$$\begin{aligned} C^m &= \texttt{COMPUTECASHFLOW}(\{\mathcal{C}_k^m\}_{k=m+1,\ldots,M}, \theta), \\ V^m &\mathrel{+}= C^m, \end{aligned} \tag{22}$$

where $C_j^m = C(t_m, x_j; \theta)$. We make use of the notation $+=$ for the standard "addition assignment" operator.

In the most common situations, when the backward PDE (1) is used to value an interest rate (*resp.* credit derivative), the auxiliary value vectors $\mathcal{C}_k^m$ in the steps above represent the value of the conditional discount factors

$$\mathcal{C}_k^m = Z(r_{t_m}, t_m, t_k) \equiv \mathbb{E}\left[\exp\left(-\int_{t_m}^{t_k} r_u \, \mathrm{d}u\right) \,\middle|\, r_{t_m}\right], \tag{23}$$

with $r_t = r(x_t)$; respectively the value of the conditional survival probabilities $Q(h_{t_m}, t_m, t_k)$ in Equation (6).

2.3 American-style options

The numerical algorithm described in the previous section can be extended to handle the pricing of securities with early exercise features, like Bermudan-style and American-style options, see Wilmott (2007), provided that the exercise value can be expressed in terms of a deterministic function of the form $E(x_t, t; \theta)$. Indeed, on each exercise date T_e, the Bellman principle, see Bellman (1952), can be expressed as a simple jump condition

$$V(x, T_e; \theta) = \max\left(V(x, T_e^+; \theta), E(x, T_e^+; \theta)\right). \tag{24}$$

By indicating with $\mathcal{T}_e$ the set of early exercise dates (assumed for simplicity to be a subset of the discretisation dates t_m, $m = 0, \ldots, M$), early exercise can be incorporated into the finite-difference scheme of the previous section as the following modification of (S2):

e) Initialise any additional payoff that might be necessary for the valuation of the exercise function $E(x_t, t; \theta)$, or the valuation of the intermediate cash-flow $C(t, x_t; \theta)$ (if any), when their expressions are not available in closed form by

$$\mathcal{C}_{m+1}^{m+1} = \texttt{AUXILIARYPAYOFF}(\theta). \tag{25}$$

Here, as in Section 2.2, $\mathcal{C}_k^m$ is the value vector at time t_m of a set of auxiliary securities with expiry t_k.

f) As in Section 2.2.

g) As in Section 2.2.

h) If $t_m \in \mathcal{T}_e$, execute the following instructions

$$\begin{aligned} E^m &= \texttt{COMPUTEEXERCISEVALUE}(\{\mathcal{C}_k^m\}_{k=m+1,\ldots,M}, \theta), \\ H^m &= V^m, \\ V^m &= \texttt{EARLYEXERCISE}(H^m, E^m), \end{aligned} \tag{26}$$

where (i) the first function computes the early exercise function $E(x_{t_m}, t_m; \theta)$, possibly using the auxiliary information $\mathcal{C}_k^m$, $k = m{+}1, \ldots, M$, (ii) the second instruction assigns $V(x_{t_m}, t_m^+; \theta)$ to the so-called "hold value" H^m, and (iii) `EARLYEXERCISE` applies the Bellman condition (24) to determine $V(x_{t_m}, t_m; \theta)$.

If the financial option may be exercised continuously in a given time interval, e.g., as for American-style options, then the set $\mathcal{T}_e$ contains all the dates of the finite-difference grid in the time interval in which early option exercise is contractually allowed. This algorithm is generally accurate to first order in the time step, even when the CN scheme is employed, although a number of schemes are available to restore the second order convergence, see Wilmott (2007).

3 Arrow-Debreu prices and forward PDEs

An alternative approach to derivatives pricing is to solve the backward PDE (1) by the Arrow-Debreu price density, e.g., Karatzas and Shreve (1998), which is also known as Green's function. In the present setting, the Arrow-Debreu price density reads:

$$\psi(y, T \,|\, x_t, t) = \mathbb{E}\left[\delta(y - x_T) \exp\left(-\int_t^T \nu(x_u, u; \theta)\mathrm{d}u\right) \,\middle|\, x_t\right], \tag{27}$$

where $\delta(\cdot)$ denotes the standard Dirac delta distribution. In the context of interest rate derivatives and short-rate models introduced in Section 2, the price $V_{t_0}(\theta)$ at time t_0 of a European-style option with maturity date T and payoff function $P(r_T; \theta)$ is given by

$$V_{t_0}(\theta) = V(x_{t_0}, t_0; \theta) = \mathbb{E}\left[\exp\left(-\int_{t_0}^T r_u \mathrm{d}u\right) P(r_T; \theta) \,\middle|\, x_{t_0}\right], \tag{28}$$

where $r_t = r(x_t)$ is the instantaneous short rate. The option price can be computed by integrating the product of the payoff function and the Arrow-Debreu price density over all the possible values the short rate may take at time T, that is

$$V_{t_0}(\theta) = \int_{\mathbb{R}} \psi(x, T \,|\, x_{t_0}, t_0) P(x; \theta) \mathrm{d}x. \tag{29}$$

The integration is performed over the range of the function $x = r^{-1}(r_T)$. In particular, the price $Z(r_{t_0}, t_0, T)$ at time t_0 of a discount bond with maturity T, can be obtained as a special case by setting $P(x; \theta) = 1$:

$$Z(r_{t_0}, t_0, T) = \int_{\mathbb{R}} \psi(x, T \,|\, x_{t_0}, t_0) \mathrm{d}x, \tag{30}$$

where $r_{t_0} = r(x_{t_0})$. In the context of default intensity models, the conditional probability (6) can be expressed similarly to Equation (30).

It is well known, see e.g., Karatzas and Shreve (1998), that the Arrow-Debreu price density (27) satisfies the following conjugate forward PDE:

$$\partial_t \psi(x, t \,|\, x_{t_0}, t_0) = \Big(-\nu(x, t; \theta) - \partial_x \mu(x, t; \theta) + \frac{1}{2}\partial_x^2 \sigma^2(x, t; \theta)\Big)\psi(x, t \,|\, x_{t_0}, t_0), \tag{31}$$

where the initial condition is given by $\psi(x, t_0 \,|\, x_{t_0}, t_0) = \delta(x_{t_0} - x)$. The Arrow-Debreu price density $\psi(x_t, t \,|\, x_{t_0}, t_0)$ can be determined for every $t > 0$ by solving Equation (31) forward in time.

3.1 Numerical solutions by finite-difference discretisation for forward PDE

The conjugate forward PDE (31) can be discretised by following similar steps to the ones illustrated in Section 2.1 for the backward PDE. For instance, by approximating the Arrow-Debreu price density $\psi(x, t_0 \,|\, x_{t_0}, t_0) = \delta(x_{t_0} - x)$ at time t_0 with its discretised counterpart

$$\psi^0 \equiv \psi(x_j, t_0 \,|\, x_{t_0}, t_0) = \Delta x^{-1} \delta_{j,j^\star}, \tag{32}$$

where $\delta_{j,j^\star}$ is Kronecker's delta and $x_{j^\star}$ is the closest spatial grid point to x_{t_0}, and by setting for the spatial boundary conditions

$$\psi(x_{\min}, t_m \,|\, x_{t_0}, t_0) = \psi(x_{\max}, t_m \,|\, x_{t_0}, t_0) = 0, \tag{33}$$

one can compute the vector $\psi^m(\theta) = (\psi(x_1, t_m \,|\, x_{t_0}, t_0), \ldots, \psi(x_N, t_m \,|\, x_{t_0}, t_0))^\top$ for $m = 0, \ldots, M$ by iterating the following matrix recursion:

$$L_F(t_m, \phi; \theta)\psi^{m+1}(\theta) = R_F(t_m, \phi; \theta)\psi^m(\theta), \tag{34}$$

for $m = 0, \ldots, M-1$, where

$$L_F(t_m, \phi; \theta) = \mathbb{I} + (1-\phi)\Delta t D^T(\tilde{t}_m(\phi); \theta), \tag{35}$$

$$R_F(t_m, \phi; \theta) = \mathbb{I} - \phi\Delta t D^T(\tilde{t}_m(\phi); \theta). \tag{36}$$

The matrix $D(t; \theta)$ is determined by Equation (10). The algorithm to solve numerically the forward PDE (31) can be described therefore by:

(S1) Initialise the value vector on the initial time slice $\psi^0(\theta) = \Delta x^{-1}\delta_{j,j^\star}$ with $j = 0, \ldots, N$ by

$$\psi^0 = \texttt{DELTA}(), \tag{37}$$

which has, in our setup, no dependence on θ [2].

(S2) For $m = 0, \ldots, M-1$, execute

$$\psi^{m+1} = \texttt{PROPAGATEADPRICE}(\psi^m; \theta), \tag{38}$$

consisting of the following steps:

a) Compute the coeffiecients $c^m(\theta) \equiv c(\tilde{t}_m(\phi); \theta)$, $u^m(\theta) \equiv u(\tilde{t}_m(\phi); \theta)$, and $l^m(\theta) \equiv l(\tilde{t}_m(\phi); \theta)$ in Equations (11) by

$$(c^m, u^m, l^m) = \texttt{COMPUTECOEFFM}(\theta). \tag{39}$$

[2] The generalization to the more general situation is straightforward.

b) Compute the matrices $L_F^m(\theta) \equiv L_F(t_m, \phi; \theta)$ and $R_F^m(\theta) \equiv R_F(t_m, \phi; \theta)$ in Equations (35) and (36) from the vectors of coefficients $c^m(\theta)$, $u^m(\theta)$ and $l^m(\theta)$ by

$$(L_F^m, R_F^m) = \texttt{COMPUTELRF}(c^m, u^m, l^m). \tag{40}$$

c) Given ψ^m, solve Equation (34) for ψ^{m+1} by calling a suitable tri-diagonal solver

$$\psi^{m+1} = \texttt{TRIDIAGSOLVER}(L_F^m, R_F^m, 0, \psi^m), \tag{41}$$

executing

$$\begin{aligned} W^{m+1} &= R_F^m \, \psi^m, \\ \psi^{m+1} &= W^{m+1} / L_F^m. \end{aligned} \tag{42}$$

Here we make use of the notation introduced at the end of Section 2.1. In order to compute the value $V_{t_0}(\theta)$ of a derivative asset, one needs to compute the integral (29) numerically, e.g., by means of Gaussian quadrature.

(S3) Given the payoff vector $P = P(x; \theta)$, execute

$$V_{t_0} = \texttt{INTEGRATE}(P, \psi^M), \tag{43}$$

where $P_j = P(x_j; \theta)$, which performs the numerical integration corresponding to Equation (29).

As for the backward PDE, the overall computational complexity of the algorithm above is of order $O(NM)$.

3.2 Forward PDEs and calibration

The valuation of a derivative security can be split in two distinct steps, a calibration and a pricing step. In the calibration step,

$$\theta = \texttt{CALIBRATION}(\mathcal{M}), \tag{44}$$

the parameters of the model $\theta = (\theta_1, \ldots, \theta_{N_\theta})$, are calibrated in order to reprice simple and liquidly-traded financial instruments. We denote the price of such instruments with the market parameter vector $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_{N_\mathcal{M}})$. For instance, for the BK-model (5), in the context of interest rate models (*resp.* credit models), the mean-reversion level function $\mu(t)$ is generally calibrated to the prices of the instruments used to build a yield curve or, equivalently, a set of discount bond rates (*resp.* a set of prices of CDS or also par spreads, see O'Kane (2011)). Similarly, the volatility function, or a combination of the volatility and the mean-reversion speed function, can be calibrated to a set of swaptions prices or implied volatilities, see Andersen and Piterbarg (2010). In the pricing step, the parameters θ are mapped to the values of the derivative security, or portfolio of N_V securities:

$$V = \texttt{PRICING}(\theta), \tag{45}$$

so that the concatenation of the calibration of the calibration and the pricing step can be seen as a map of the form $\mathcal{M} \to \theta \to V$.

The calibration step (44) typically involves an iterative routine, e.g., performing a numerical root search or least-square minimisation. Forward PDEs and combinations of forward and backward PDEs are usually used to implement efficiently the calibration step. In the following we will assume that the mean-reversion level $\mu(t)$, the mean-reversion speed $\kappa(t)$ and the volatility function $\sigma(t)$ in Equation (5) are all (left-continuous) piecewise constant on the time line $T_1 < \ldots < T_L$ (assumed uniform for simplicity in the following), which is a subset of the discretisation time axis t_m, $m = 0, \ldots, M$, with, e.g., $T_1 > t_0$ and $T_L = t_M$, and we indicate with $\eta(i)$ the map such that $T_i = t_{\eta(i)}$, for $i = 1, \ldots, L$ and $\eta(0) = 0$. The model parameter θ can therefore be expressed in terms of the levels of such functions in each piece-wise interval, namely

$$\theta = (\mu_1, \ldots, \mu_L, \kappa_1, \ldots, \kappa_L, \sigma_1, \ldots, \sigma_L),$$

where $\mu_i = \mu(T_i)$, $\kappa_i = \kappa(T_i)$ and $\sigma_i = \sigma(T_i)$, for $i = 1, \ldots, L$.

In the credit context, the first goal of the calibration is to match the survival probabilities in Equation (6), $Q(h_{t_0}, t_0, T_i)$, $i = 1, \ldots, L$, with their market-implied counterparts $Q^{\text{mkt}}(t_0, T_i)$, implied in turn via a standard bootstrap procedure (see O'Kane (2011)) from a set of CDS quotes observed in the market. Here $h_{t_0} = \exp(x_{t_0})$ is a free parameter of the model that for simplicity we assume fixed at some reasonable value.

The algorithm for the calibration of the mean-reversion level function can be described as follows. Initialise ψ^0 with (S1) of Section 3.1 and for $i = 1, \ldots, L$, proceed as follows.

(S1) Make a choice for μ_i.

(S2) Perform the instructions in Equation (38) for $m = \eta(i-1), \ldots, \eta(i) - 1$ and determine $\psi^{\eta(i)}$.

(S3) Determine the numerical approximation of the survival probability in Equation (6) given by

$$Q(h_{t_0}, t_0, T_i) = \texttt{INTEGRATE}\left(\mathbf{1}, \psi^{\eta(i)}\right), \tag{46}$$

where $\mathbf{1}$ is the N-dimensional unit vector.

(S4) If the computed value $Q(h_{t_0}, t_0, T_i)$ equals what is quoted in the market, then stop, otherwise go back to (S1) above.

The cost of the algorithm above is $O(MNN_{av})$ where N_{av} is the average number of root search iterations of (S1)-(S3) above, and it is $O(N)$ more efficient than what can be achieved with a backward PDE approach, see Andersen and Piterbarg (2010).

The calibration of the volatilities parameters to, e.g., a collection of K CDS options with expiration dates T_i^e, $i = 1, \ldots K$ (assumed for simplicity to be a subset of $\{T_i\}_{i=1}^L$) and with underlying CDS maturity at time T_L, can be implemented efficiently with a combination of

the forward and backward algorithms. Here we assume, for simplicity, a stylised payoff for a CDS (payer) option with expiry date T_i^e and underlying maturity T_L of the form

$$P^{\text{swpt}}\left(h_{T_i^e}, T_i^e, T_L\right) = \left(s\left(h_{T_i^e}\right) - c\right)^+ \mathcal{A}\left(h_{T_i^e}, T_i^e, T_L\right), \tag{47}$$

where c is the running coupon of the CDS at which the option can be exercised, and $\mathcal{A}(h_{T_i^e}, T_i^e, T_L)$ and $s(h_{T_i^e})$ are respectively, the *credit-risky* annuity and the par-spread for a T_L maturity CDS contract starting at time T_i^e. The price at time T_i^e of a credit-risky annuity is given by

$$\mathcal{A}(h_{T_i^e}, T_i^e, T_L) = \sum_{t_m \in \mathcal{C}(T_i^e, T_L)} \Delta t_c \; Z(T_i^e, t_m) Q(h_{T_i^e}, T_i^e, t_m), \tag{48}$$

where $\mathcal{C}(T_i^e, T_L)$ is the set of discretisation dates t_m corresponding to the coupon dates[3] for a CDS starting at time T_i^e and maturing at T_L. The interval Δt_c is the length of the coupon period (assumed uniform and commensurate with the spacing of the time grid $T_1, \ldots, T_L$ for simplicity), and

$$Z(t, t_m) = \exp\left(-\int_t^{t_m} r_u \mathrm{d}u\right)$$

is the deterministic discount factor. The CDS par-spread is defined by

$$s(h_{T_i^e}) = \mathcal{L}(h_{T_i^e}, T_i^e, T_L) / \mathcal{A}(h_{T_i^e}, T_i^e, T_L) \tag{49}$$

where $\mathcal{L}(h_{T_i^e}, T_i^e, T_L)$ is the (discounted expected) loss

$$\mathcal{L}(h_{T_i^e}, T_i^e, T_L) = (1 - R) \int_{T_i^e}^{T_L} Z(T_i^e, u) \left(-\frac{\mathrm{d}Q(h_{T_i^e}, T_i^e, u)}{\mathrm{d}u}\right) \mathrm{d}u, \tag{50}$$

and R is the expected recovery rate (assumed independent of the default time). In a discretised setting, the expected loss is generally approximated by

$$\begin{aligned} &\mathcal{L}(h_{T_i^e}, T_i^e, T_L) \\ &\quad = (1 - R) \sum_{t_m \in \mathcal{C}(T_i^e, T_L)} Z\left(T_i^e, t_m\right) \left(Q(h_{T_i^e}, T_i^e, t_m - \Delta t_c) - Q(h_{T_i^e}, T_i^e, t_m)\right), \end{aligned} \tag{51}$$

so that the payoff in Equation (47) can be computed, provided the conditional survival probabilities $Q(h_{T_i^e}, T_i^e, t_m)$, with $t_m \in \mathcal{C}(T_i^e, T_L)$ and $i = 1, \ldots, K$ can be calculated.

In particular, the calculation of the price of a swaption with expiry T_i^e, $i = 1 \ldots, K$, and underlying CDS-maturity T_L,

$$V_{t_0}^{\text{swpt}}\left(T_i^e, T_L\right) = Z\left(t_0, T_i^e\right) \mathbb{E}\left[P^{\text{swpt}}\left(h_{T_i^e}, T_i^e, T_L\right) \mid x_{t_0}\right], \tag{52}$$

given a set of volatilities $\sigma_1, \ldots, \sigma_L$, can be implemented as follows:

[3]We can assume for simplicity that the coupon dates are a subset of the discretization dates t_m, $m = 1, \ldots, M$.

(S1′) By applying the forward induction above, calibrate μ_l, $l = 1, \ldots, L$. Save the Arrow-Debreu prices $\psi_j^{\eta(l)} = \psi(x_j, T_l \,|\, x_{t_0}, t_0)$ where $l = 1, \ldots, L$.

(S2′) Execute (S2) of the backward induction algorithm in Section 2.1 equipped with the steps e) and f) of Section 2.2. The auxiliary securities are chosen such that they provide a unit cash-flow at each coupon date of the CDS underlying the options, namely $\mathcal{C}_m^m = \mathbf{1}$ for $t_m \in \mathcal{C}(T_i^e, T_L)$, and zero otherwise. As a result, $[\mathcal{C}_k^m]_j$ represents the conditional survival probability $Q(x_j, t_m, t_k)$.

(S3′) Given the conditional survival probabilities $Q(x_j, T_i^e, t_m)$, for $t_m \in \mathcal{C}(T_i^e, T_L)$, computed in (S2′)

   a) create the CDS option payoff in Equation (47), and

   b) integrate the payoff against $\psi(x_j, T_i^e \,|\, x_{t_0}, t_0)$ by numerical quadrature, as in Equation (29). Then produce the value $V_{t_0}^{\text{swpt}}(T_i^e, T_L)$ of the swaption at time t_0.

For processes characterised by a weak dependence of swaption prices on instantaneous volatility after the expiry, the calibration of the volatilities $\sigma_1, \ldots, \sigma_L$ can be performed with the following bootstrap procedure (see e.g., Andersen and Piterbarg (2010)): Starting from the first option expiry date T_1^e, one can vary all the knot points of the instantaneous volatilities at times before T_1^e while keeping the others constant until the value $V_{t_0}^{\text{swpt}}(T_1^e, T_L)$ of the swaption is matched. Similarly, for the subsequent dates T_i^e, one can simultaneously vary all the knot points of the instantaneous volatilities at the times between (and including) T_{i-1}^e and T_i^e until the value $V_{t_0}^{\text{swpt}}(T_i^e, T_L)$ of the swaption is matched. On the last expiry date T_K^e, one can vary all the knot points of the instantaneous volatilities at T_{K-1}^e and after until the value $V_{t_0}^{\text{swpt}}(T_K^e, T_L)$ of the swaption is matched. Since a swaption price has a weak dependence on all the volatilities past its expiry dates, the bootstrap procedure above needs to be repeated a few times until convergence is achieved. Conversely, when swaption prices have a strong dependence on volatility after expiry, one cannot apply the bootstrap procedure above and the recourse to a multidimensional solver is necessary. This is the case for instance for the BK model in Equation (5).

4 AAD and PDEs

4.1 Adjoint Algorithmic Differentiation

The main idea underlying algorithmic differentiation (see e.g., Griewank (2000)) is that any computer implemented function – no matter how complicated – can be interpreted as a composition of basic arithmetic and intrinsic operations that are easy to differentiate. In particular, when one requires the derivatives of a small number of outputs with respect to a large number of inputs, the calculation can be optimised by applying the chain rule through the instructions of the program in opposite order with respect to their original evaluation. This gives rise to Adjoint Algorithmic Differentiation (AAD). The book by Griewank (2000)

contains a detailed discussion of the computational cost of AAD. In this section, we will only recall the main results in order to clarify how this technique can be beneficial for financial computations and implementations. The reader can find in Capriotti and Giles (2010a) several simple examples illustrating the intuition behind these results.

We now consider a function

$$Y = \texttt{FUNCTION}(X) \tag{53}$$

that maps a vector $X \in \mathbb{R}^r$ to a vector $Y \in \mathbb{R}^s$ through a sequence of steps

$$X \ \rightarrow \ \ldots \ \rightarrow \ U \ \rightarrow \ V \ \rightarrow \ \ldots \ \rightarrow \ Y. \tag{54}$$

Here, the real vectors U and V represent intermediate variables used in the calculation and each step can be a distinct high-level function or even a specific instruction.

The adjoint mode of algorithmic differentiation results from propagating the derivatives of the final output with respect to all the intermediate variables – the so called *adjoints* – until the derivatives with respect to the independent variables are formed. Using the standard AD notation, the adjoint of any intermediate variable V_k is defined by

$$\bar{V}_k = \sum_{j=1}^{s} \bar{Y}_j \frac{\partial Y_j}{\partial V_k}, \tag{55}$$

where $\bar{Y}$ is a vector in $\mathbb{R}^s$. For each of the intermediate variables U_i, by applying the chain rule, we get

$$\bar{U}_i = \sum_{j=1}^{s} \bar{Y}_j \frac{\partial Y_j}{\partial U_i} = \sum_{j=1}^{s} \bar{Y}_j \sum_k \frac{\partial Y_j}{\partial V_k} \, \frac{\partial V_k}{\partial U_i},$$

which corresponds to the adjoint mode equation for the intermediate step represented by the function $V = V(U)$. We thus have a function of the form $\bar{U} = \bar{V}(U, \bar{V})$ where

$$\bar{U}_i = \sum_k \bar{V}_k \frac{\partial V_k}{\partial U_i}.$$

Starting from the adjoint of the outputs $\bar{Y}$, we can apply this rule to each step in the calculation, working from the right to the left,

$$\bar{X} \ \leftarrow \ \ldots \ \leftarrow \ \bar{U} \ \leftarrow \ \bar{V} \ \leftarrow \ \ldots \ \leftarrow \ \bar{Y} \tag{56}$$

until we obtain $\bar{X}$. In other words, until one obtains the linear combination of the rows of the Jacobian of the function $X \rightarrow Y$, i.e.,

$$\bar{X}_i = \sum_{j=1}^{s} \bar{Y}_j \frac{\partial Y_j}{\partial X_i} \tag{57}$$

for $i = 1, \ldots, r$.

In the adjoint mode, the cost does not increase with the number of inputs, but it is linear in the number of (linear combinations of the) rows of the Jacobian that need to be evaluated independently. If the full Jacobian is required, one needs to repeat the adjoint calculation s times, setting the vector $\bar{Y}$ equal to each of the elements of the canonical basis in $\mathbb{R}^s$.

One particularly important theoretical result is that given a computer program performing some high-level function (53), the execution time of its adjoint counterpart

$$\bar{X} = \texttt{FUNCTION\_b}(X, \bar{Y}) \tag{58}$$

(with suffix `_b` for "backward" or "bar") that computes the linear combination (57), is bounded by three to four times the cost of execution of the original one. That is,

$$\frac{\text{Cost}[\texttt{FUNCTION\_b}]}{\text{Cost}[\texttt{FUNCTION}]} \leq \omega_A \tag{59}$$

where $\omega_A \in [3, 4]$, see Griewank (2000).

4.2 AAD and backward PDEs

The evaluation of the numerical solution of the PDE (1) by means of the algorithm described in Section 2.1 can be seen as a computer-implemented function mapping $\theta \to V_{t_0}(\theta)$. By following the principles of AAD, it is possible to design its adjoint counterpart $(\theta, \bar{V}_{t_0}) \to (V_{t_0}, \bar{\theta})$ which gives (for $\bar{V}_{t_0} = 1$) the sensitivities

$$\bar{\theta}_k = \frac{\partial V(\theta)}{\partial \theta_k}, \tag{60}$$

for $k = 1, \ldots, N_\theta$. The adjoint of the numerical solution of the backward PDE in Section 2.1 consists therefore of Steps 1-3 followed by their corresponding adjoint, executed in reverse order:

($\bar{\text{S3}}$) Set $\bar{V}_{t_0} = 1$, and execute

$$\bar{V}^0 = \texttt{COMPUTESPOTVALUE\_b}(V^0, \bar{V}_{t_0}) \tag{61}$$

which computes the gradient

$$\bar{V}_j^0 = \bar{V}_{t_0} \frac{\partial V_{t_0}}{\partial V_j^0} \tag{62}$$

for $j = 1, \ldots, N$, according to rule (12) .

($\bar{\text{S2}}$) For $m = 0, \ldots, M-1$, in opposite order than in (S2) of Section 2.1, execute the following steps:

$\bar{d}$) Given $\bar{V}^m$, execute the adjoint of the function in Equation (17), namely

$$(\bar{L}_B^m, \bar{R}_B^m, \bar{\beta}^{m+1}, \bar{V}^{m+1}) = \texttt{TRIDIAGSOLVER\_b}(L_B^m, R_B^m, \beta^{m+1}, V^{m+1}, \bar{V}^m), \tag{63}$$

which computes

$$
\begin{aligned}
[\bar{L}_B^m]_{j,l} &= \sum_{r=1}^{N} \bar{V}_r^m \frac{\partial V_r^m}{\partial [L_B^m]_{j,l}}, \\
[\bar{R}_B^m]_{j,l} &= \sum_{r=1}^{N} \bar{V}_r^m \frac{\partial V_r^m}{\partial [R_B^m]_{j,l}}, \\
\bar{\beta}_j^{m+1} &= \sum_{r=1}^{N} \bar{V}_r^m \frac{\partial V_r^m}{\partial \beta_j^{m+1}}, \\
\bar{V}_j^{m+1} &= \sum_{r=1}^{N} \bar{V}_r^m \frac{\partial V_r^m}{\partial V_j^{m+1}},
\end{aligned}
\tag{64}
$$

for $j = 1, \ldots, N$ and $l = 1, \ldots, N$.

$\bar{c}$) Compute the adjoint of the function in Equation (16), namely

$$
\bar{\theta} = \texttt{COMPUTEBC\_b}(\theta, \bar{\beta}^{m+1}), \tag{65}
$$

which gives

$$
\bar{\theta}_k = \sum_{r=1}^{N} \bar{\beta}_r \frac{\partial \beta_r^m}{\partial \theta_k}. \tag{66}
$$

This provides the initialisation of the vector of sensitivities $\bar{\theta}$.

$\bar{b}$) Compute the adjoint of the function in Equation (15), that is

$$
(\bar{c}^m, \bar{u}^m, \bar{l}^m) = \texttt{COMPUTELRB\_b}(c^m, u^m, l^m, \bar{L}_B^m, \bar{R}_B^m), \tag{67}
$$

which produces the adjoint of the coefficient vectors

$$
\begin{aligned}
\bar{c}_j^m &= [\bar{L}_B^m]_{j,j} \frac{\partial [L_B^m]_{j,j}}{\partial c_j^m} + [\bar{R}_B^m]_{j,j} \frac{\partial [R_B^m]_{j,j}}{\partial c_j^m}, \\
\bar{u}_j^m &= [\bar{L}_B^m]_{j,j+1} \frac{\partial [L_B^m]_{j,j+1}}{\partial u_j^m} + [\bar{R}_B^m]_{j,j+1} \frac{\partial [R_B^m]_{j,j+1}}{\partial u_j^m}, \\
\bar{l}_{j+1}^m &= [\bar{L}_B^m]_{j+1,j} \frac{\partial [L_B^m]_{j+1,j}}{\partial l_{j+1}^m} + [\bar{R}_B^m]_{j+1,j} \frac{\partial [R_B^m]_{j+1,j}}{\partial l_{j+1}^m},
\end{aligned}
\tag{68}
$$

where $j = 1, \ldots, N$ in the first equation and $j = 1, \ldots, N-1$ in the second and third. Here we have used the fact that each component of the vectors c^m, u^m and l^m appears only in one element of the tree main diagonals of the matrices L_B^m and R_B^m. By Equations (8) and (9) it is immediate to verify that

$$
\frac{\partial [L_B^m]_{j,j}}{\partial c_j^m} = \frac{\partial [L_B^m]_{j,j+1}}{\partial u_j^m} = \frac{\partial [L_B^m]_{j+1,j}}{\partial l_{j+1}^m} = -\phi, \tag{69}
$$

$$
\frac{\partial [R_B^m]_{j,j}}{\partial c_j^m} = \frac{\partial [R_B^m]_{j,j+1}}{\partial u_j^m} = \frac{\partial [R_B^m]_{j+1,j}}{\partial l_{j+1}^m} = 1 - \phi, \tag{70}
$$

for $0 < \phi < 1$. For the fully explicit, $\phi = 0$, (resp. the fully explicit case, $\phi = 1$), L_B^m (resp. R_B^m) is the identity matrix and $\bar{L}_B^m$ (resp. $\bar{R}_B^m$) is identically zero.

$\bar{a}$) Compute the adjoints of the coefficients (11),

$$\bar{\theta} \mathrel{+}= \texttt{COMPUTECOEFFM\_b}(\theta, \bar{c}^m, \bar{u}^m, \bar{l}^m). \tag{71}$$

This produces the following contribution of the adjoint of the $\bar{\theta}$ vector

$$\bar{\theta}_k \mathrel{+}= \sum_{j=1}^{N} \left[\bar{c}_j^m \frac{\partial c_j^m(\theta)}{\partial \theta_k} + \bar{u}_j^m \frac{\partial u_j^m(\theta)}{\partial \theta_k} + \bar{l}_j^m \frac{\partial l_j^m(\theta)}{\partial \theta_k} \right] \tag{72}$$

for $k = 1, \ldots, N_\theta$, with $u_N^m \equiv 0$ and $l_1^m \equiv 0$.

($\bar{\text{S1}}$) Compute the adjoint of the vector V^M given by Equation (13) by executing

$$\bar{\theta} \mathrel{+}= \texttt{PAYOFF\_b}(\theta, \bar{V}^M). \tag{73}$$

This gives the vector elements

$$\bar{\theta}_k \mathrel{+}= \sum_{j=1}^{N} \bar{V}_j^M \frac{\partial P(x_j; \theta)}{\partial \theta_k}, \tag{74}$$

for $k = 1, \ldots, N_\theta$, associated with the explicit dependence of the payoff on the model parameters θ (if any).

One can verify that the execution of the steps above produces the sensitivities (60) of the option value with respect to the parameters θ. According to the general result of AAD (59), the cost to compute all the components of the adjoint vector $\bar{\theta}$ is a small multiplier of order four times the cost of computing (S1) to (S4), therefore resulting in an overall computation complexity of $O(NM)$.

We note that obtaining the adjoint `COMPUTESPOT_b` of the linear scheme in Equation (12) is straightforward. the procedure consists of setting $\bar{V}_j^0 = 0$ for $j \notin \{j^\star, j^\star{+}1\}$, and allocating $\bar{V}_{j^\star}^0$ and $\bar{V}_{j^\star+1}^0$ with their coefficients in Equation (12), namely

$$\begin{aligned} \bar{V}_{j^\star}^0 &= \bar{V}_{t_0} \big(1 - \frac{x_{t_0} - x_{j^\star}}{x_{j^\star+1} - x_{j^\star}}\big), \\ \bar{V}_{j^\star+1}^0 &= \bar{V}_{t_0} \frac{x_{t_0} - x_{j^\star}}{x_{j^\star+1} - x_{j^\star}}. \end{aligned} \tag{75}$$

The adjoint function `TRIADIAGSOLVER_b`, which gives the adjoint of (18), can be implemented as follows:

$$\begin{aligned}
\bar{W}^{m+1} &= [L_B^m]^{-T}\,\bar{V}^m,\\
\overline{[L_B^m]^{-1}} &= \bar{V}^m\left[W^{m+1}\right]^T,\\
\bar{L}_B^m &= -\,[L_B^m]^{-T}\,\overline{[L_B^m]^{-1}}\,[L^m]_B^{-T},\\
\bar{U}^{m+1} &= \bar{W}^{m+1},\\
\bar{\beta}^{m+1} &= \bar{W}^{m+1},\\
\bar{R}_B^m &= \bar{U}^{m+1}\left[V^{m+1}\right]^T,\\
\bar{V}^{m+1} &= [R_B^m]^T\,\bar{U}^{m+1}.
\end{aligned} \tag{76}$$

Here we have used the fact that the adjoint of the linear operation $y = Bx$ is given by $\bar{x} = B^T\bar{y}$ and $\bar{B} = \bar{y}x^T$, and the identity $\bar{A} = -A^{-T}\overline{A^{-1}}A^{-T}$, which holds for any invertible matrix A, see Giles (2008). The computational cost of the instructions above is $O(N^2)$. In order to reduce the computational cost to $O(N)$, as in the original sequence (18), one needs to avoid the matrix inversion in the first instruction of (76). This is obtained by utilising the solution of a linear system and then by combining the first three instructions of Equation (76) and the third of Equation (18). We thus have:

$$\bar{L}_B^m = -\,[L_B^m]^{-T}\,\bar{V}^m\left[W^{m+1}\right]^T[L^m]_B^{-T} = -\bar{W}^{m+1}\left[[L^m]_B^{-1}\,W^{m+1}\right]^T = -\bar{W}^{m+1}\,[V^m]^T. \tag{77}$$

Then, the resulting algorithm is given by

$$\begin{aligned}
\bar{W}^{m+1} &= \bar{V}^m/\,[L_B^m]^T,\\
\bar{L}_B^m &= -\bar{W}^{m+1}\,[V^m]^T,\\
\bar{U}^{m+1} &= \bar{W}^{m+1},\\
\bar{\beta}^{m+1} &= \bar{W}^{m+1},\\
\bar{R}_B^m &= \bar{U}^{m+1}\left[V^{m+1}\right]^T,\\
\bar{V}^{m+1} &= [R_B^m]^T\,\bar{U}^{m+1}.
\end{aligned} \tag{78}$$

We emphasise that only the elements on the three main diagonals of $\bar{L}_B^m$ and $\bar{R}_B^m$ contribute to the sensitivities, so that only $3N$ multiplications are required for their computation in the second and fourth instruction of Equation (78). The overall computational cost of the adjoint tri-diagonal solver is $O(N)$, exactly as for the forward counterpart (18), and as expected from the general result (59).

The execution of the adjoint instructions (78) requires the vector V^m. This is a manifestation of the general feature of the adjoint implementation which require (i) the execution of the original code, (ii) the storage of the intermediate results and final outputs before the execution of its adjoint counterpart. In this case, `TRIADIAGSOLVER_b` needs to contain a forward sweep replicating the instructions (18) in order to compute V^m. Alternatively, if the values V_m were to be stored during the calculation in the forward sweep of (S1)-(S3),

then one could use the stored values directly as inputs in `TRIADIAGSOLVER_b`. This scheme is clearly more efficient as it avoids repeating the forward sweep. However, the first implementation comes with a reduced memory consumption as it does not require the storage of the vectors V^m for $m = 0, \ldots, M$ and represents a simple illustration of a technique known as 'checkpointing', see Capriotti and Giles (2010a).

Finally, the adjoint of the function `COMPUTECOEFFM_b` in Equation (71) can be implemented by the adjoint of Equations (11), namely

$$\begin{aligned}
\bar{\sigma}_j^m &= -\bar{c}_j^m 2\sigma(x_j, \tilde{t}_m; \theta)\Delta x^{-2} + \bar{u}_j^m \sigma(x_j, \tilde{t}_m; \theta)\Delta x^{-2} + \bar{l}_j^m \sigma(x_j, \tilde{t}_m; \theta)\Delta x^{-2}, \\
\bar{\nu}_j^m &= -\bar{c}_j^m, \\
\bar{\mu}_j^m &= \bar{u}_j^m \frac{1}{2}\Delta x^{-1} - \bar{l}_j^m \frac{1}{2}\Delta x^{-1},
\end{aligned} \tag{79}$$

for $j = 1, \ldots, N$, and

$$\begin{aligned}
\bar{\theta} &+= \bar{\sigma}(x_j, \tilde{t}_m; \theta, \bar{\sigma}_j^m), \\
\bar{\theta} &+= \bar{\mu}(x_j, \tilde{t}_m; \theta, \bar{\mu}_j^m), \\
\bar{\theta} &+= \bar{\nu}(x_j, \tilde{t}_m; \theta, \bar{\nu}_j^m),
\end{aligned} \tag{80}$$

adding the contributions to the sensitivities

$$\begin{aligned}
\bar{\theta}_k &+= \bar{\sigma}_j^m \frac{\partial \sigma(x_j, \tilde{t}_m; \theta)}{\partial \theta_k}, \\
\bar{\theta}_k &+= \bar{\mu}_j^m \frac{\partial \mu(x_j, \tilde{t}_m; \theta)}{\partial \theta_k}, \\
\bar{\theta}_k &+= \bar{\nu}_j^m \frac{\partial \nu(x_j, \tilde{t}_m; \theta)}{\partial \theta_k},
\end{aligned} \tag{81}$$

for $k = 1, \ldots, N_\theta$. The implementation of the adjoint functions in Equation (80) depends on the particular model considered.

4.2.1 Intermediate cashflows and American-style options

The adjoint algorithm presented in the previous section can be extended to include early-exercise contracts described in Sections 2.2 and 2.3 by the following modification of ($\bar{S}2$) above:

$\bar{h}$) For $t_m \in \mathcal{T}_e$, set $\{\bar{\mathcal{C}}_k^m\}_{k=m+1,\ldots,M} = 0$ and execute the following instructions:

$$(\bar{H}^m, \bar{E}^m) = \texttt{EARLYEXERCISE\_b}(H^m, E^m, \bar{V}^m), \tag{82}$$

$$\bar{V}^m = H^m, \tag{83}$$

$$(\{\bar{\mathcal{C}}_k^m\}_{k=m+1,\ldots,M}, \bar{\theta}) += \texttt{COMPUTEEXERCISEVALUE\_b}(\{\mathcal{C}_k^m\}_{k=m+1,\ldots,M}, \theta, \bar{E}^m). \tag{84}$$

It is important to note that the application of the AAD rules in Capriotti and Giles (2010a) require the adjoint $\bar{V}^m$ be overridden rather than incremented.

$\bar{g}$) Execute the adjoint of Equation (22), that is:

$$\begin{aligned} \bar{C}^m &= \bar{V}^m, \\ (\{\bar{\mathcal{C}}_k^m\}_{k=m+1,\dots,M}, \bar{\theta}) &+= \texttt{COMPUTECASHFLOW\_b}(\{\mathcal{C}_k^m\}_{k=m+1,\dots,M}, \theta, \bar{C}^m). \end{aligned} \tag{85}$$

$\bar{f}$) For $k = M, \dots, m+1$, call the adjoint tri-diagonal solver

$$(\bar{L}_B^m, \bar{R}_B^m, \bar{\beta}^{m+1}, \bar{\mathcal{C}}_k^{m+1}) += \texttt{TRIDIAGSOLVER\_b}(L_B^m, R_B^m, \beta^{m+1}, \mathcal{C}_k^{m+1}, \bar{\mathcal{A}}_k^m). \tag{86}$$

$\bar{e}$) Then execute

$$\bar{\theta} += \texttt{AUXILIARYPAYOFF\_b}(\theta, \bar{\mathcal{C}}_{m+1}^{m+1}), \tag{87}$$

which computes the contribution to the sensitivities arising form the intermediate cashflows and the early-exercise optionality.

4.3 AAD and forward PDEs

Analogous to what is discussed in Section 4.2 for the backward PDE, the adjoint of the numerical solution of the forward PDE of Section 3.1 consists of (S1)-(S3) followed by their corresponding adjoint operations executed in reverse order. That is:

($\bar{\text{S3}}$) Set $\bar{V}_{t_0} = 1$, execute

$$(\bar{P}, \bar{\psi}^M) = \texttt{INTEGRATE\_b}(P, \psi^M, \bar{V}_{t_0}) \tag{88}$$

and compute, according to the rule (12), the gradients

$$\bar{\psi}_j^M = \bar{V}_{t_0} \frac{\partial V_{t_0}}{\partial \psi_j^M}, \tag{89}$$

$$\bar{P}_j = \bar{V}_{t_0} \frac{\partial V_{t_0}}{\partial P_j}, \tag{90}$$

for $j = 1, \dots, N$. For an example in which a Gaussian quadrature scheme is applied, we refer to Capriotti and Lee (2014). The contribution to the sensitivities arising from the functional form of the payoff (if any) is then computed by

$$\bar{\theta}_k = \sum_{j=1}^{N} \bar{P}_j \frac{\partial P_j}{\partial \theta_j}, \tag{91}$$

for $k = 1, \dots, N_\theta$.

($\bar{\text{S2}}$) For $m = M-1, \dots, 0$ continue with:

$\bar{c}$) Given $\bar{\psi}^{m+1}$, execute the adjoint of the function in Equation (41), namely

$$(\bar{L}_F^m, \bar{R}_F^m, \bar{\psi}^m) = \texttt{TRIDIAGSOLVER\_b}(L_F^m, R_F^m, 0, \psi^m, \bar{\psi}^{m+1}). \tag{92}$$

$\bar{b}$) Compute the adjoint of the function in Equation (40), namely

$$(\bar{c}^m, \bar{u}^m, \bar{l}^m) = \texttt{COMPUTELRF\_b}(c^m, u^m, l^m, \bar{L}_F^m, \bar{R}_F^m). \tag{93}$$

$\bar{a}$) Compute the adjoints of the coefficients (11),

$$\bar{\theta} \mathrel{+}= \texttt{COMPUTECOEFFM\_b}(\theta, \bar{c}^m, \bar{u}^m, \bar{l}^m), \tag{94}$$

with the same adjoint functions as described in Section 4.2.

($\bar{\text{S1}}$) This step is void since the initialisation function `DELTA`() has no dependency on θ.

One can verify that the execution of the steps above produces the sensitivities of the option value with respect to the parameters θ, see Equation (60). As before, the general AAD result (59) guarantees that the cost to compute all the components of the adjoint vector $\bar{\theta}$ is a maximum of four times the cost of computing (S1)-(S4) in Section 3.2, therefore resulting in an overall computational complexity of order $O(NM)$.

4.4 Calibration algorithm: AAD and the implicit function theorem

As recalled in Section 3.2, the valuation of a derivative security can be generally separated in two distinct steps, a calibration and a pricing step. While the calculation of the sensitivities with respect to the internal model parameters $\partial V/\partial\theta$ obtained by the adjoint of the pricing step (45),

$$\bar{\theta} = \texttt{PRICING\_b}(\theta, \bar{V}), \tag{95}$$

which computes

$$\bar{\theta}_k = \sum_{i=1}^{N_V} \bar{V}_i \frac{\partial V_i}{\partial \theta_k}, \tag{96}$$

for $k = 1, \ldots, N_\theta$, is sometimes useful, what is required for the risk management of the portfolio of the derivative securities are the sensitivities $\partial V/\partial\mathcal{M}$ with respect to the liquid market prices because they define the size of the hedges, see e.g. Hull (2002). These can be obtained, according to the general principles of AAD, by reversing the order of computations so that the adjoint of the algorithm consists of the adjoint pricing step, combined with the adjoint calibration step

$$\bar{\mathcal{M}} = \texttt{CALIBRATION\_b}(\mathcal{M}, \bar{\theta}), \tag{97}$$

giving

$$\bar{\mathcal{M}}_m = \sum_{k=1}^{N_\theta} \bar{\theta}_k \frac{\partial \theta_k}{\partial \mathcal{M}_m}, \tag{98}$$

for $m = 1, \ldots, N_\mathcal{M}$. The overall adjoint algorithm can be seen therefore as a map of the form $\bar{V} \to \bar{\theta} \to \bar{\mathcal{M}}$.

The adjoint calibration step (44) can be implemented according to the general rules of AAD (Section 4.1), paying attention to its iterative nature. However, following the work by Christianson (1998) and Henrard (2011), a much better performance can be obtained by exploiting the so-called *implicit function theorem* (IFT), as described below. Here we consider the case in which the calibration algorithm in Equation (97) consists of the numerical solution of a system of equations of the form

$$G_i(\mathcal{M}, \theta) = 0 \tag{99}$$

where $\mathcal{M} \in \mathcal{R}^{N_\mathcal{M}}$, $\theta \in \mathcal{R}^{N_\theta}$, and $i = 1, \ldots, N_\theta$. The function $G_i(\mathcal{M}, \theta)$ is often of the form

$$G_i(\mathcal{M}, \theta) = T_i(\mathcal{M}) - V_i(\theta) \tag{100}$$

where $V_i(\theta)$ is the price of the i-th calibration instrument as produced by the model to be calibrated, and $T_i(\mathcal{M})$ are the prices of the target instruments, possibly generated by a simpler model utilised as a quoting mechanism, see Hull (2002).

As noted above, the adjoint calibration can be implemented in terms of the adjoint of the numerical scheme solving (99). The associated computational cost is expected to be a few times the cost of solving the numerical system (99) (but approximately less than 4 times the cost, according to the general result of AAD). A much better performance can be obtained by exploiting the so-called implicit function theorem.

Under mild regularity conditions, the implicit function theorem asserts that if there is a solution $(\mathcal{M}_0, \theta_0)$ to the root finding problem (99), such that

$$G_i(\mathcal{M}_0, \theta_0) = 0, \tag{101}$$

and the matrix of derivatives $[\partial G/\partial\theta]_{ij} = \partial G_i(\mathcal{M}_0, \theta_0)/\partial\theta_j$ is invertible, then one can define in the vicinity of $\mathcal{M}_0$ an implicit function $\theta = \theta(\mathcal{M})$ such that

$$G_i(\mathcal{M}, \theta(\mathcal{M})) = 0. \tag{102}$$

The derivatives $\partial\theta/\partial\mathcal{M}$ of such function can be expressed in terms of the derivatives of the objective function G. Indeed, by differentiating (102) with respect to $\mathcal{M}$, one obtains

$$\frac{\partial G_i}{\partial \mathcal{M}_m} + \sum_{j=1}^{N_\theta} \frac{\partial G_i}{\partial \theta_j} \frac{\partial \theta_j}{\partial \mathcal{M}_m} = 0 \tag{103}$$

for $m = 1, \ldots, N_\mathcal{M}$, or equivalently

$$\frac{\partial \theta_k}{\partial \mathcal{M}_m} = -\left[\left(\frac{\partial G}{\partial \theta} \right)^{-1} \frac{\partial G}{\partial \mathcal{M}} \right]_{km}, \tag{104}$$

with $[\partial G/\partial\mathcal{M}]_{ij} = \partial G_i/\partial\mathcal{M}_j$. This relation allows the computation of the sensitivities of the function $\theta(\mathcal{M})$, locally defined in an implicit manner by Equation (99), in terms of

the sensitivities of the function $G(\mathcal{M}, \theta)$. These can be computed by implementing the corresponding adjoint function

$$(\bar{\mathcal{M}}, \bar{\theta}) = \bar{G}(\mathcal{M}, \theta, \bar{G}) \tag{105}$$

giving, according to the general rule (Section 4.1),

$$\bar{\mathcal{M}}_m = \sum_{i=1}^{N_\theta} \bar{G}_i \frac{\partial G_i}{\partial \mathcal{M}_m}, \tag{106}$$

$$\bar{\theta}_k = \sum_{i=1}^{N_\theta} \bar{G}_i \frac{\partial G_i}{\partial \theta_k}. \tag{107}$$

This method is significantly more efficient and stable than the naïve approach of calculating the derivatives of the implicit functions $\mathcal{M} \to \theta(\mathcal{M})$ by differentiating directly the calibration step either by bumping or by applying AAD to the calibration step. This is because $G(\mathcal{M}, \theta)$ in Equation (100) are *explicit* functions of the market and model parameters, which are easy to compute and differentiate. Moreover, by avoiding the numerical noise produced by applying the finite difference approximation to the calibration procedure, the accuracy of the sensitivities is improved when compared with the bumping scheme.

5 Numerical results

In this section, we present the numerical results arising from the pricing and the calibration applications of the BK model (5) for the stochastic instantaneous hazard rate $h_t = \exp(x_t)$ that satisfies

$$\mathrm{d}\ln(\mathrm{h_t}) = \kappa(\mathrm{t})\left(\mu(\mathrm{t}) - \ln(\mathrm{h_t})\right)\mathrm{dt} + \sigma(\mathrm{t})\mathrm{dW_t}. \tag{108}$$

We fix the mean-reversion rate $\kappa = 0.01$ and assume $\mu(t)$ and $\sigma(t)$ to be left-continuous, piecewise constant functions. As shown in Section 3.2, we can calibrate $\mu(t)$ and $\sigma(t)$ to a set of survival probabilities implied from liquid CDS prices and a set of (co-terminal) CDS option prices. As it is market practice, we compute the survival probabilities in terms of a (left-continuous) piecewise constant hazard rate function $\lambda^{\mathrm{mkt}}(t)$ with L knot points $(\lambda_1^{\mathrm{mkt}}, \ldots, \lambda_L^{\mathrm{mkt}})$ at times $(T_1, \ldots, T_L)$. These are determined, for convenience, on the same time grid, with equally-spaced intervals $\Delta T = T_{i+1} - T_i = 0.5$, for $i = 1, \ldots, L-1$, as utilised for the mean-reversion level and volatility functions. The market survival probabilities, as seen at time $t_0 = 0$, are then given by

$$Q^{\mathrm{mkt}}(t_0, T_i) = \exp\left[-\int_{t_0}^{T_i} \lambda^{\mathrm{mkt}}(u)\mathrm{d}u\right] = \prod_{j=1}^{i} \exp\left[-\lambda_j^{\mathrm{mkt}} \Delta T\right]. \tag{109}$$

In these numerical examples, for simplicity, we choose the knot points of the hazard rate function to be the same and equal to $\lambda^{\mathrm{mkt}} = s/(1-R)$, where $s = 1\%$ is the the so-called par-spread and $R = 40\%$ is the recovery rate, as set by market practice.

Time	0.25	0.5	0.75	1	1.5	2
SP	0.996	0.992	0.988	0.983	0.975	0.967
Time	2.5	3	3.5	4	4.5	5
SP	0.959	0.951	0.943	0.935	0.928	0.920

Table 1: Survival probabilities utilised for the calibration of the parameters θ.

Expiry	0.25	0.5	1	1.5	2	2.5	3	4
Price (10^{-2})	0.570	0.719	0.894	0.947	0.926	0.853	0.739	0.417

Table 2: Prices of option written on a five-year CDS utilised for the calibration of the parameters θ.

Similarly, the CDS option prices are derived using the standard Black formula from a set of market-implied volatilities $\sigma_j^{\rm mkt}$, $j = 1, \ldots, K$, corresponding to option maturities $(T_1^e, \ldots, T_K^e)$. We have:

$$V^{\rm mkt}(t_0, T_j^e; T_L) = \Big(c\,\Phi(-d_1) - s\,\Phi(-d_2)\Big)\mathcal{A}^{\rm mkt}(T_j^e, T_L), \tag{110}$$

$$\mathcal{A}^{\rm mkt}(T_j^e, T_L) = \left(\sum_{m \in \mathcal{C}(T_j^e, T_L)} Z(t_0, t_m) Q^{\rm mkt}(t_0, t_m) \Delta t_c\right), \tag{111}$$

$$d_1 = \frac{\ln s/c + (\sigma_j^{\rm mkt})^2 (T_j^e - t_0)/2}{\sigma_j^{\rm mkt}\sqrt{T_j^e - t_0}}, \tag{112}$$

$$d_2 = d_1 - \sigma_j^{\rm mkt}\sqrt{T_j^e - t_0}, \tag{113}$$

where Φ is the standard normal distribution function, and c is the payment rate by the protection buyer for the CDS against which the option can be exercised. Here we set c equal to the par spread s, so as to represent at-the-money option quotes. We assume, for simplicity, zero interest rates and we choose the implied volatilities to have all the same value, $\sigma^{\rm mkt} = 60\%$. The market data we obtained are given in Table 1 and 2 and the results of the calibration are shown in Figure 1.

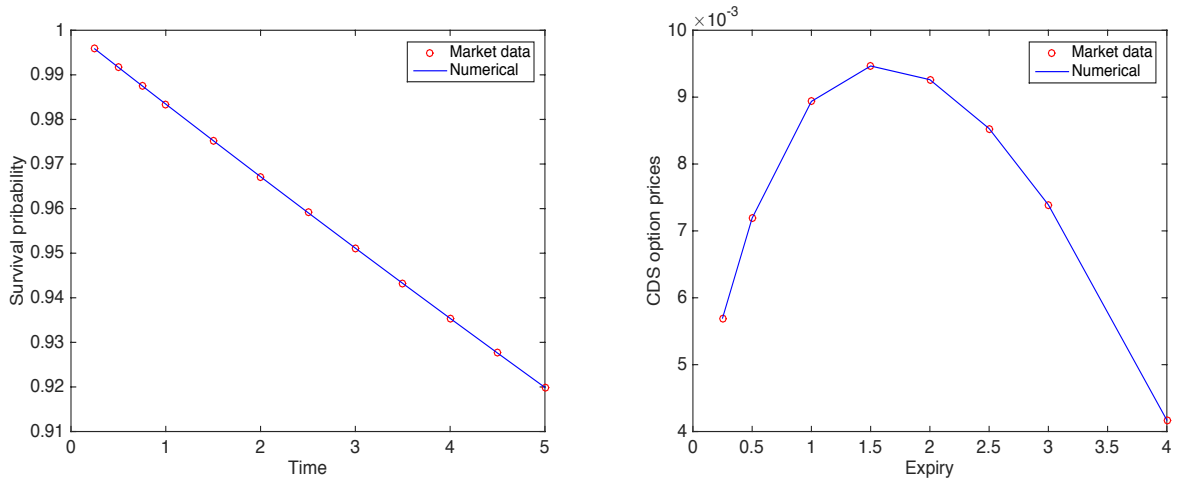

Figure 1: Calibration of the Black-Karasinski hazard rate model to a set of market implied survival probabilities and option prices.

5.1 AAD versus bumping for computation of sensitivities

We consider the pricing of a defaultable discount bond with a five-year maturity and a unit redemption value. The value at time $t_0 = 0$ of such instrument is given by Equation (6), and it can be determined either by a backward or forward PDE by setting $V(T, h_T; \theta) = 1$. In order to illustrate the reliability of the AAD calculation of sensitivities in the PDE framework, Table 3 displays the comparison of the sensitivities obtained by the AAD algorithms, described in Section 4.2 and 4.3, and by means of one-sided finite-difference approximation (bumping) with a perturbation (bump) size of 10^{-5}. As expected, the results obtained with both the AAD version of the backward and forward PDE are consistent with the ones obtained by bumping with minor differences due to discretisation errors and the finite precision of the finite-difference approach. Similarly, in Table 4 we compare the sensitivities results for a CDS option and a bond option using a combination of the forward and backward PDE approach described in Section 3.2. Here we consider a two-year at-the-money swaption written on a five-year CDS, and a two-year European-style call option issued on the five-year defaultable bond with a strike of 0.75. As in the previous example, these results confirm that the AAD approach provides accurate estimates of the sensitivities when benchmarked with the standard finite-difference approach.

	Bwd PDE (AAD)	Fwd PDE (AAD)	Bwd PDE (FD)	Fwd PDE (FD)
μ_1	-1.8e-4	-1.8e-4	-1.8e-4	-1.8e-4
μ_2	-1.7e-4	-1.6e-4	-1.7e-4	-1.6e-4
μ_3	-1.6e-4	-1.6e-4	-1.6e-4	-1.6e-4
μ_4	-1.5e-4	-1.5e-4	-1.5e-4	-1.5e-4
μ_5	-2.7e-4	-2.7e-4	-2.7e-4	-2.7e-4
μ_6	-2.3e-4	-2.3e-4	-2.3e-4	-2.3e-4
μ_7	-2.0e-4	-2.0e-4	-2.0e-4	-2.0e-4
μ_8	-1.6e-4	-1.6e-4	-1.6e-4	-1.6e-4
μ_9	-1.3e-4	-1.3e-4	-1.3e-4	-1.3e-4
μ_{10}	-9.1e-5	-9.1e-5	-9.2e-5	-9.1e-5
μ_{11}	-5.6e-5	-5.5e-5	-5.6e-5	-5.5e-5
μ_{12}	-1.8e-5	-1.8e-5	-1.9e-5	-1.8e-5
σ_1	-7.8e-3	-7.8e-3	-7.5e-3	-7.8e-3
σ_2	-0.011	-0.01	-0.011	-0.01
σ_3	-0.016	-0.016	-0.017	-0.016
σ_4	-0.015	-0.015	-0.015	-0.015
σ_5	-0.013	-0.013	-0.013	-0.013
σ_6	-0.012	-0.011	-0.012	-0.011
σ_7	-9.9e-3	-9.9e-3	-0.01	-9.9e-3
σ_8	-0.019	-0.019	-0.02	-0.019

Table 3: Parameters sensitivities of a five-year defaultable discount bond computed by the AAD version of the forward and backward PDE schemes and by finite-difference (FD) approximations with a bump size of 10^{-5}.

	CDS option		Bond option	
	AAD	FD	AAD	FD
μ_1	-2.3e-5	-2.3e-5	-1.2e-4	-1.1e-4
μ_2	-2.3e-5	-2.3e-5	-1.1e-4	-1.1e-4
μ_3	-2.3e-5	-2.3e-5	-1.1e-4	-1.1e-4
μ_4	-2.3e-5	-2.3e-5	-1.1e-4	-1.1e-4
μ_5	-4.5e-5	-4.5e-5	-2.1e-4	-2.1e-4
μ_6	-4.5e-5	-4.4e-5	-2.1e-4	-2.1e-4
μ_7	-4.1e-5	-4.0e-5	-1.9e-4	-1.9e-4
μ_8	-3.4e-5	-3.3e-5	-1.6e-4	-1.6e-4
μ_9	-2.6e-5	-2.6e-5	-1.2e-4	-1.2e-4
μ_{10}	-1.9e-5	-1.9e-5	-8.8e-5	-8.8e-5
μ_{11}	-1.1e-5	-1.2e-5	-5.3e-5	-5.4e-5
μ_{12}	-3.7e-6	-3.9e-6	-1.7e-5	-1.8e-5
σ_1	3.5e-4	3.5e-4	-4.3e-3	-4.1e-3
σ_2	5.0e-4	5.0e-4	-6.0e-3	-6.0e-3
σ_3	8.3e-4	8.3e-4	-0.01	-0.01
σ_4	8.3e-4	8.3e-4	-0.01	-0.01
σ_5	7.6e-4	8.2e-4	-0.01	-0.01
σ_6	-2.6e-3	-2.6e-3	-0.011	-0.011
σ_7	-2.3e-3	-2.3e-3	-9.7e-3	-9.7e-3
σ_8	-4.3e-3	-4.3e-3	-0.019	-0.019

Table 4: Parameters sensitivities of a CDS option and defaultable discount bond option computed by means of AAD and by finite-difference (FD) approximations with a bump size of 10^{-5}.

The computational efficiency of AAD is shown in Figure 2. Here we plot the cost of computing the sensitivities of a defaultable discount bond with respect to the knot points of the mean-reversion level μ_i, $i = 1, \ldots, L$ and volatility σ_i, $i = 1, \ldots, K$, relative to the cost of performing a *single* valuation. As illustrated in Figure 2, for both, the AAD version of the backward and the forward PDE scheme, the calculation of the sensitivities can be performed for about 3.3 times the cost of computing the value of the bond, i.e., well within the theoretical bound (59). In contrast, the cost of bumping is in general $(1 + N_\theta)$ times the cost of as single valuation, i.e., in this case over 20 times the cost of computing the value of the bond. Similarly, the cost of computing the sensitivities of a bond and CDS option by AAD is also bounded, but the cost of the bumping scheme is proportional to the number of parameters. Furthermore, as shown in Figure 3, the overall cost of running the AAD scheme to obtain all the sensitivities relative to the cost of computing the option value through a single valuation of the PDE scheme is independent of the number of sensitivities so that the computational gains, when compared to the bumping scheme, *increase* with the number of

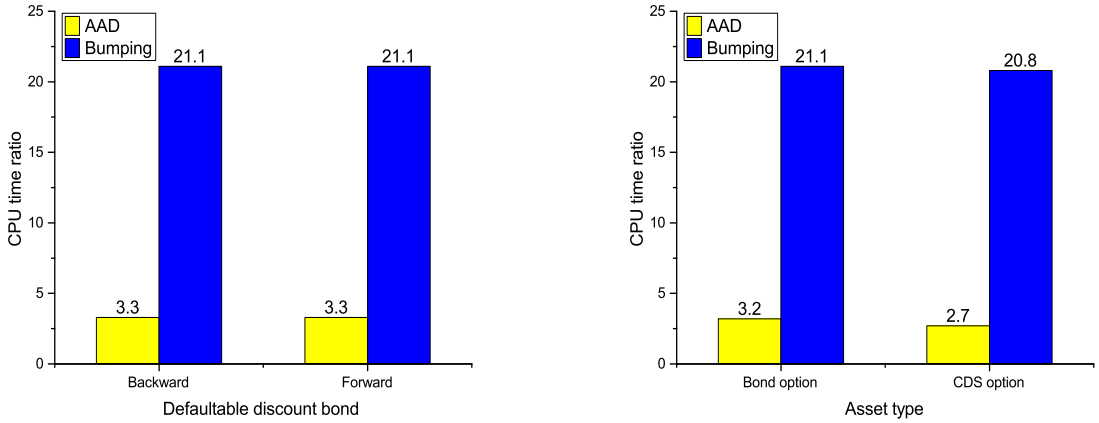

Figure 2: Cost of computing the sensitivities for a defaultable discount bond, CDS option and defaultable discount bond option, relative to the cost of a single valuation.

sensitivities[4].

5.2 Calibration and the implicit function theorem

As described in Section 4.4, the sensitivities with respect to the internal model parameter θ can be converted into the more practically relevant sensitivities with respect to the market parameters M by combining sensitivities obtained with the AAD version of the forward and backward PDE executed during the calibration of the model parameters and the so-called implicit function theorem (IFT).

As an illustration, we can again consider the two-year option on a five-year defaultable bond with strike price 0.6. By making use of the scheme described in Section 4.4, the sensitivities with respect to the model parameters $\bar{\theta} = \partial V/\partial\theta$ can be transformed into the sensitivities with respect to the market observables $\partial V/\partial\mathcal{M}$. In this case, they are the sensitivities with respect to the implied hazard rates and the CDS options implied volatilities, which are used for the calibration in Figure 1. Table 5 displays the bond option market sensitivities obtained by converting the model sensitivities in Table 4 by means of the AAD-IFT approach, and shows the good agreement with those obtained with the standard finite-difference approach.

[4]In this examples we have included the sensitivities with respect to the knot points of the mean-reversion speed function.

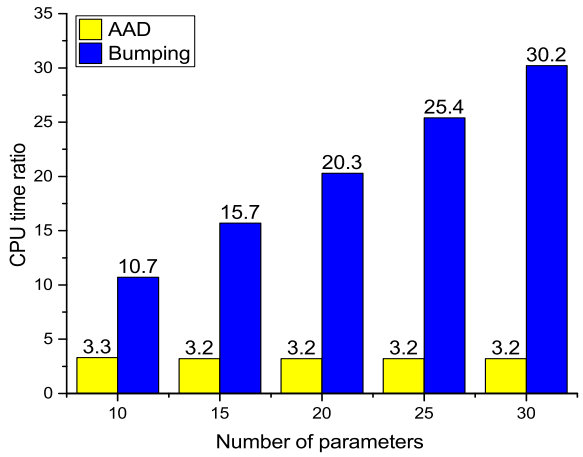

Figure 3: Cost of computing the sensitivities for a defaultable discount bond option, relative to the cost of a single valuation, as a function of the number of sensitivities.

Market observables	IFT	FD
λ_1^{mkt}	-0.088	-0.078
λ_2^{mkt}	-0.083	-0.089
λ_3^{mkt}	-0.087	-0.081
λ_4^{mkt}	-0.084	-0.086
λ_5^{mkt}	-0.172	-0.168
λ_6^{mkt}	-0.170	-0.174
λ_7^{mkt}	-0.454	-0.453
λ_8^{mkt}	-0.454	-0.452
λ_9^{mkt}	-0.455	-0.455
$\lambda_{10}^{\text{mkt}}$	-0.454	-0.453
$\lambda_{11}^{\text{mkt}}$	-0.455	-0.455
$\lambda_{12}^{\text{mkt}}$	-0.454	-0.454
σ_1^{mkt}	7.2e-07	7.9e-07
σ_2^{mkt}	2.5e-06	2.8e-06
σ_3^{mkt}	4.2e-06	-5.1e-06
σ_4^{mkt}	-6.4e-06	-5.7e-06
σ_5^{mkt}	0.001	0.001
σ_6^{mkt}	-2.5e-05	-2.1e-5
σ_7^{mkt}	-2.2e-05	-1.9e-05
σ_8^{mkt}	-1.5e-05	-1.2e-05

Table 5: Sensitivities of a defaultable discount bond option with respect to the market observables as obtained with AAD and by finite-difference (FD) approximations with a bump size of 10^{-5}.

The remarkable computational gains that can be achieved with the AAD-IFT scheme are shown in Figure 4 (left, yellow column). Here we plot the ratio of time necessary to convert the model sensitivities into market sensitivities by means of both the ADD-IFT approach and standard finite differences, relative to the cost of performing a *single* calibration and valuation. For this application, the time necessary to compute the Jacobian $\partial\theta/\partial\mathcal{M}$ and model parameter sensitivities $\partial V/\partial\theta$ by the AAD-IFT approach is just 0.8% the amount of time necessary to perform a single calibration and valuation, thus resulting in 3 orders of magnitude speed-up with respect to standard bumping. This staggering difference in efficiency is due in part to the computationally intensive calibration procedure of the BK model involving, as described in Section 3.2, a multidimensional root-search over the instantaneous volatilities. However, even for models for which the more efficient bootstrap procedure can be used, the computational gains of the AAD-IFT scheme are still very substantial. This is illustrated in the right panel of Figure 4 displaying analogous results for the model by Hull and White (1996).

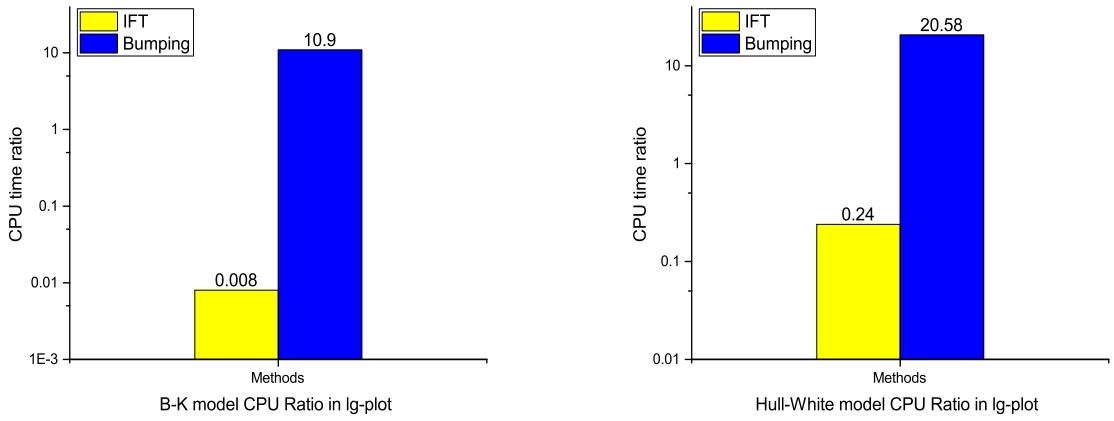

Figure 4: Cost of computing the market parameter sensitivities for a defaultable discount bond option relative to the cost of a single calibration and valuation for the BK model (left panel) and the model by Hull and White (1996) (right panel).

6 Conclusions

We have shown how Adjoint Algorithmic Differentiation (AAD) can be used to implement the calculation of sensitivities of option prices computed as numerical solutions of PDEs efficiently and for generic financial securities. In particular, AAD can be of great benefit for computing the sensitivities associated with PDE-based calibration algorithms. With an example of practical relevance, we have demonstrated how by combining the adjoint versions of the algorithms for the numerical solution of backward and forward PDEs together with the implicit function theorem, one can avoid repeating multiple times the calibration algorithm

or implementing the AAD version of the calibration routine. This allows the calculation of all the price sensitivities for an additional computational cost that is *a fraction* of the cost of computing the P&L of the portfolio, thus typically resulting in procedures orders of magnitude faster than standard finite-difference approaches. AAD, when necessary combined with the implicit function theorem, paves the way to risk management in real-time with PDEs at a very limited computational cost. We expect the insights presented in this work to be of significant importance for the efficient implementation of pricing and hedging approaches in a real-world set-up, and thus be appealing in particular to financial engineering and the industry by and large.

Acknowledgments

The opinions and views expressed in this paper are uniquely those of the authors, and do not necessarily represent those of Credit Suisse Group. The authors thank C. A. Garcia Trillos for useful comments and Shinghoi (Jacky) Lee for a careful reading of the manuscript.

References

Andersen, L. and Piterbarg, V. (2010). *Interest Rate Modeling, Volume I: Foundations and Vanilla Models*. Atlantic Financial Press.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716.

Black, F., Derman, E., and Toy, W. (1990). A one-factor model of interest rates and its application to treasury bond options. *Financial Analysts Journal*, 46(1):33–39.

Black, F. and Karasinski, P. (1991). Bond and option pricing when short rates are lognormal. *Financial Analysts Journal*, 47(4):52–59.

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

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

Capriotti, L. and Giles, M. (2010a). Algorithmic differentiation: Adjoint greeks made easy. *Risk*, 25:92–98.

Capriotti, L. and Giles, M. (2010b). Fast correlation risk by adjoint algorithmic differentiation. *Risk*, 23:79–85.

Capriotti, L. and Lee, J. (2014). Adjoint credit risk management. *Risk*, 27:90–96.

Capriotti, L., Peacock, M., and Lee, J. (2011). Real time counterparty credit risk management in monte carlo. *Risk*, 24:86–90.

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

Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). An equilibrium characterization theory of the term structure. *Econometrica*, 53:385–407.

Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 43, pages 50–67. Cambridge University Press.

Crépey, S., Bielecki, T. R., and Brigo, D. (2014). *Counterparty Risk and Funding–A Tale of Two Puzzles*. Chapman and Hall/CRC Financial Mathematics Series.

Giles, M. B. (2008). Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In *Advances in Automatic Differentiation*, pages 35–44. Springer.

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

Green, A. (2015). *XVA: Credit, Funding and Capital Valuation Adjustments*. Wiley.

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics, Philadelphia.

Henrard, M. (2011). Adjoint algorithmic differentiation: Calibration and implicit function theorem. *OpenGamma Quantitative Research*, 1.

Hull, J. C. (2002). *Options, Futures and Other Derivatives*. Prentice Hall, New Jersey.

Hull, J. C. and White, A. D. (1996). Using Hull-White interest rate trees. *Journal of Derivatives*, 3(3):26–36.

Iserles, A. (2009). *A First Course in the Numerical Analysis of Differential Equations*, volume 44. Cambridge University Press.

Karatzas, I. and Shreve, S. E. (1998). *Methods of Mathematical Finance*, volume 39. Springer Science & Business Media.

Naumann, U. (2012). *The Art of Differentiating Computer Programs*. SIAM.

O'Kane, D. (2011). *Modelling Single-name and Multi-name Credit Derivatives*, volume 573. John Wiley & Sons.

Pooley, D. M., Vetzal, K. R., and Forsyth, P. A. (2003). Convergence remedies for non-smooth payoffs in option pricing. *Journal of Computational Finance*, 6(4):25–40.

Wilmott, P. (2007). *Paul Wilmott Introduces Quantitative Finance*. John Wiley & Sons.