Monte Carlo pathwise sensitivities for barrier options

Thomas Gerstner, Bastian Harrach and Daniel Roth

Institut für Mathematik, Goethe-Universität Frankfurt am Main, Robert-Mayer-Straße 10, 60325 Frankfurt am Main, Germany; emails: gerstner@math.uni-frankfurt.de, harrach@math.uni-frankfurt.de, roth@math.uni-frankfurt.de

(Received September 6, 2018; revised August 5, 2019; accepted October 23, 2019)

ABSTRACT
The Monte Carlo pathwise sensitivities approach is well established for smooth payoff functions. In this work, we present a new Monte Carlo algorithm that is able to calculate the pathwise sensitivities for discontinuous payoff functions. Our main tool is to combine the one-step survival idea of Glasserman and Staum with the stable differentiation approach of Alm, Harrach, Harrach and Keller. As an application, we use the derived results for a five-dimensional calibration of a contingent convertible bond, which we model with different types of discretely monitored barrier options with time-dependent barrier levels.

Keywords: Monte Carlo; discretely monitored barrier options; pathwise sensitivities; CoCo bond.

1 INTRODUCTION
We consider Monte Carlo algorithms (see, for example, Glasserman 2003) for the pricing and the computation of sensitivities for different types of options with discontinuous payoff, particularly discretely monitored barrier options. Depending on
whether or not an underlying exceeds a predefined barrier, the payoff of a barrier option may or may not be zero. There are two substantial types of barrier options: those that pay zero when there is no barrier crossing (the so-called “knock-in” options) and those that pay zero when the barrier is crossed (the “knock-out” options). It is obvious that barrier options are cheaper than the standard option without a barrier, since they are worthless in more circumstances. For an overview of other exotic options, particularly with discontinuous payoff, we refer the reader to, for example, Zhang (1998). Many models and algorithms assume continuous monitoring for barrier options, mainly because this leads to analytical solutions. In practice, however, many traded barrier options are discretely monitored, not only because there are practical implementation issues, but also for legal and financial reasons (see, for example, Broadie et al 1997).

The price of an option is evaluated by taking the integral of its expected discounted payoff under a risk-neutral probability measure. For barrier options, however, the payoff is discontinuous over the space of all paths. If we look at simple cases, there are analytical formulas for the option price. But if we want to use a more complex stochastic process or a high-dimensional model, there will not be useful formulas. As a result, it is often useful to use Monte Carlo simulations, which are easily adapted to these models. However, for Monte Carlo algorithms, the discontinuous payoff leads to the problem that the option’s sensitivities, such as Delta and Vega, cannot be stably determined from the numerically calculated prices of the standard Monte Carlo algorithm, since even the smallest numerical errors in the price may have arbitrarily large effects on the sensitivities (see, for example, Alm et al 2013; Koster and Rehmet 2018).

Within this work, we derive a Monte Carlo algorithm that allows us to calculate the pathwise sensitivities of knock-out barrier options as well as those of digital knock-in and knock-out barrier options. The main part of this paper is based on Glasserman and Staum’s one-step survival strategy (Glasserman and Staum 2001) and the results of Alm et al (2013), from which we know that, with this approach, we can stably determine the option’s sensitivities such as Delta and Vega by using simple finite differences. The basic idea of Glasserman and Staum (2001) is to use a truncated normal distribution, which excludes the values above the barrier (eg, for knock-up-and-out options), instead of sampling from the full normal distribution. This approach avoids the discontinuity generated by any Monte Carlo path crossing the barrier, which gives a Lipschitz-continuous payoff function (Alm et al 2013). Further, the output allows stable numerical differentiation and leads to a reduction in variance.

The aim of this paper is to develop a novel extended algorithm that estimates the sensitivities for the one-step survival technique directly, without the need for simulation at multiple parameter values as in finite difference. This is an advantage, since
the choice of the step width of the finite difference varies with the input parameters to balance stability and accuracy.

The sensitivity computation of our approach involves the ideas of pathwise sensitivities, symbolic differentiation and automatic differentiation. We refer the reader to Naumann (2012) and Griewank and Walther (2008) for recent work on automatic differentiation, and to Naumann and Toit (2018), Giles and Glasserman (2006) and Capriotti (2010) for adjoint automatic differentiation (AAD) in computational finance. For several financial applications it could be very helpful to add additional information or structures to AAD tools; for example, Naumann and Toit (2018) show how an external function interface can reduce the memory requirement for common numerical patterns appearing in financial codes. However, while hand coding can guarantee maximum performance (eg, for lookback options we just need to store the maximum or minimum, or we can store the results of very expensive operations, typically $\exp$), it could be unfeasible across a large code base. The algorithm uses the one-step survival smoothing technique together with symbolic differentiation to apply an easily understandable and efficient AAD-like method, which can be easily modified by hand.

As a final example, we want to calibrate contingent convertibles as an application of the developed theory and to illustrate another benefit of pathwise sensitivities. These are debt instruments, which convert debt into equity upon a trigger event. Contingent convertibles entered the financial world in December 2009, when Lloyd’s Banking Group offered their investors the possibility to swap their bonds for this new instrument. In early February 2011, Credit Suisse managed to attract US$2 billion in new capital with this new asset class. De Spiegeleer and Schoutens (2012) give an in-depth analysis of the pricing and structuring of these contingent convertible (CoCo) bonds. They show that a CoCo bond can be priced by a corporate bond, a knock-in forward and several binary down-in options. We will compare the use of the standard Monte Carlo, the one-step survival with numerical differentiation and the one-step survival with pathwise sensitivities for the calibrating model parameters of a CoCo bond.

For a general overview of Monte Carlo Greek computation for all types of options, we refer the reader to Giles and Glasserman (2006), Glasserman (2003), Seydel (2006), Capriotti (2010), Burgos and Giles (2011) and Giles (2009). There are several ways to overcome the challenges of different exotic options with non-Lipschitz payoff functions that are investigated in Burgos and Giles (2011). In particular, for barrier options we have to handle discontinuous path-dependent payoff functions, which can be distinguished in both the continuously monitored and discretely monitored cases. For the Greeks of continuously monitored barrier options, we refer the reader to Burgos and Giles (2012), where these are handled for general stochastic
differential equations with the multilevel Monte Carlo approach first introduced by Giles (2008a,b). This approach uses pathwise sensitivities, but it is not applicable for discretely observed barrier options. Now, the challenges of discretely observed barrier options are handled in the following two ways: payoff-smoothing combined with finite differences, or the likelihood ratio method. However, Alm et al (2013) determined that the first approach is computationally more efficient. In our new approach, we will combine payoff smoothing with pathwise sensitivities to preserve efficiency and to eliminate the need for finite differences.

The structure of this work is as follows. In Section 2, we derive our Monte Carlo pricing algorithms for the above-mentioned barrier options and their pathwise sensitivities. Then, in Section 3, we study our algorithms’ properties and compare the results of the pathwise sensitivities with the finite-difference approach before presenting a case study of a CoCo bond. Section 4 contains some concluding remarks.

2 MONTE CARLO ONE-STEP SURVIVAL PATHWISE SENSITIVITIES FOR DISCRETELY OBSERVED BARRIER OPTIONS

In this section, we derive our Monte Carlo pathwise sensitivities algorithm for different types of discretely observed barrier options. For this, we will stick to the method of Alm et al (2013) and use a slightly modified construction of their first algorithm.

We focus on the case where the options depend on only one underlying asset. Further, we focus on call options, but the conversion to put options is straightforward.

Let \( S_t \) describe the evolution of the underlying spot price and let \( (S_1, \ldots, S_T) \) be the vector containing the evaluations at fixed, chronologically sorted, observation dates \( (t_1, \ldots, t_T) \). We will focus on the Black–Scholes model, where \( S_t \) is assumed to follow a geometric Brownian motion,

\[
\frac{dS_t}{S_t} = \mu S_t dt + \sigma S_t dW_t.
\]

with \( \mu := r - b \), where \( r \) is the risk-free interest rate and \( b \) is the dividend yield. The volatility \( \sigma > 0 \), and \( W_t \) is the standard Brownian motion. This model yields

\[
S_{j+1} = S_j \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) \Delta t + \sigma \sqrt{\Delta t} Z_j \right),
\]

(2.1)

with \( j = 0, \ldots, T \) and \( Z_j \) independent and standard normally distributed, with \( t_0 \) and \( S_0 = s_0 \) the current time and underlying price, and with \( \Delta t \) the step width (see, for example, Hull 2008).
2.1 One-step survival pathwise sensitivities for knock-up-and-out options

In this subsection, we study the knock-up-and-out case in depth, and then discuss other cases less extensively in the following sections.

The payoff of a discretely observed knock-up-and-out barrier call option is given by

\[ V(S_1, \ldots, S_T) = \begin{cases} (S_T - K)^+ := q(S_T) & \text{if } \max_{j=1, \ldots, T} S_j \leq B, \\ 0 & \text{otherwise}, \end{cases} \tag{2.2} \]

with barrier level \( B \), strike price \( K \) and observations \( j = 1, \ldots, T \) at observation dates \( t_1, \ldots, t_T \).

**Definition 2.1** The present value of an option with payoff (2.2) is given by the discounted expected payoff

\[ \text{PV}_{t_0} = e^{-r(t_T-t_0)} \mathbb{E}(V(S_1, \ldots, S_T)) \]

at the current time \( t_0 \).

Starting with the current underlying price \( s_0 = S_0 \) and using (2.1), we can generate a path from \( s_1 \) to \( s_T \) by sampling with independent and identically standard normally distributed random variables \( Z_j \sim \mathcal{N}(0,1) \). By sampling a sequence of possible realizations \( (s_1, \ldots, s_T) \), \( n = 1, \ldots, N \), of the random variables \( (S_1, \ldots, S_T) \), we obtain an unbiased standard Monte Carlo estimator for \( \text{PV}_{t_0} \) (see, for example, Glasserman 2003).

Now we will derive an alternative unbiased Monte Carlo estimator, based on the idea of one-step survival, which allows for pathwise sensitivities. The idea of Alm et al (2013) using the one-step survival technique of Glasserman and Staum (2001) to obtain stable differentiability can be interpreted in different ways: forcing the path to stay below the barrier or considering an integral splitting (see, for example, Alm et al 2013). The latter will be the foundation of our further studies.

For the present value of a discretely observed barrier option, we have

\[ \text{PV}_{t_0}(S_0) = e^{-r\Delta t} \int_{\mathbb{R}} \phi(z) \text{PV}_{t_1}(S_1(z)) \, dz, \]

with standard normal distribution \( \phi \), time increment between the first observation and the current time \( \Delta t := (t_1 - t_0) \), and where \( S_1(z) \) is the value of \( S \) at the first observation with (2.1). In the following, we assume an equidistant time increment.
\( \Delta t = (t_1 - t_0) = \cdots = (t_T - t_{T-1}) \) between the monitoring dates, leading to \((t_T - t_0) = T\Delta t\), but a generalization would be straightforward. By splitting the integral at the first observation (see, for example, Alm et al 2013), we obtain

\[
PV_{t_0}(S_0) = e^{-r\Delta t} \left( 0 + \int_{S_1(z) < B} \phi(z) PV_{t_1}(S_1(z)) \, dz \right),
\]

(2.3)
since the payoff will be zero for \(S_1(z) \geq B\). We obtain similar formulas for the latter steps and, as explained in Alm et al (2013), the integral has to be normalized in every step to ensure a probability density. In this case, this leads to

\[
PV_{t_0}(S_0) = e^{-r\Delta t} p_0 \int_{S_1(z) < B} \frac{\phi(z)}{p_0} PV_{t_1}(S_1(z)) \, dz.
\]

(2.4)

On reaching maturity, the present value \(PV_{t_T}\) simplifies to \(q(S_T)\). Note that in practice no \(p_t\) will become zero. This method can be interpreted as a special case of importance sampling (see, for example, Glasserman 2003). Now, unlike Alm et al (2013) or Glasserman and Staum (2001), we will make an integral substitution to obtain easier access to the pathwise sensitivities. Rewriting the domain of integration (see, for example, Alm et al (2013) for a solution of \(S_1(z) < B\)) and using the substitution

\[
z = \Phi^{-1}(u \cdot p_0),
\]

(2.5)
we obtain an independent and constant domain of integration for the present value, resulting in

\[
PV_{t_0}(S_0) = e^{-r\Delta t} p_0 \int_0^1 PV_{t_1}(S_1(u)) \, du,
\]

(2.6)
and, with a slight abuse of notation,

\[
S_1(u) = S_0 \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) \Delta t + \sigma \sqrt{\Delta t} z(u) \right),
\]

(2.7)
\[
z(u) = \Phi^{-1}(p_0 u).
\]

(2.8)

By iteratively splitting and substituting until maturity, we obtain the following result.

**Theorem 2.2** The present value of a knock-up-and-out barrier option with payoff (2.2) is given by

\[
PV_{t_0}(S_0) = e^{-r(T-t_0)} \int_0^1 \cdots \int_0^1 p_0 \cdots p_{T-1} \cdot q(S_T(u^{(T)}, \ldots, u^{(1)})) \, du^{(T)} \cdots du^{(1)}
\]

(2.9)
with
\[ p_t = \Phi \left( \frac{1}{\sigma \sqrt{\Delta t}} \left( \log \left( \frac{B}{S_t(u(t))} \right) - \left( \mu - \frac{\sigma^2}{2} \Delta t \right) \right) \right), \tag{2.10} \]
\[ S_{t+1}(u^{(t+1)}) = S_t(u^{(t)}) \exp \left( \left( \mu - \frac{\sigma^2}{2} \right) \Delta t + \sigma \sqrt{\Delta t} \Phi^{-1}(p_t u^{(t+1)}) \right) \tag{2.11} \]
and the recursion base function \( S_0(u^{(0)}) = S_0 \).

**Proof** Iteratively splitting and substituting following the above considerations leads to
\[
\text{PV}_{t_0}(S_0) = e^{-r \Delta t} p_0 \int_{S_1(z^{(1)}) < B} \frac{\phi(z^{(1)})}{p_0} \ldots \times e^{-r \Delta t} p_{T-1} \int_{S_T(z^{(T)}) < B} \frac{\phi(z^{(T)})}{p_{T-1}} q(S_T(z^{(T)}, \ldots, z^{(1)})) \, dz^{(T)} \ldots dz^{(1)}
\]
\[
= e^{-r \Delta t} p_0 \int_0^1 \ldots e^{-r \Delta t} p_{T-1} \int_0^1 q(S_T(u^{(T)}, \ldots, u^{(1)})) \, du^{(T)} \ldots du^{(1)}.
\]

Using (2.10) and (2.11), a Monte Carlo estimator can create the path from \( s_1 \) to \( s_T \) by sampling with independent and identically distributed random variables \( u^{(t+1)} \sim \mathcal{U}(0, 1) \).

**Corollary 2.3** The one-step survival Monte Carlo estimator for the present value of a knock-up-and-out barrier option given by the average discounted one-step survival payoff
\[
\hat{\text{PV}}_N = e^{-r(t - t_0)} \frac{1}{N} \sum_{n=1}^{N} p_{0,n} \ldots p_{T-1,n} q(s_{T,n})
\]
is unbiased.

We derived a recursion formula, illustrated in Figure 1, for the modified asset price process and the barrier hitting probabilities.

We see that the integral domains are compact now, as stated above, and the integrand is composed of Lipschitz-continuous functions. Therefore, the differentiation can be pulled into the integral. In exchange for this, the new asset price \( S_{t+1} \) has an extra term depending on \( p_t \) and its dependencies.

At this point, we want to study the new pathwise sensitivities, for which we will use the formulas derived in Theorem 2.2. In the following, we will use \( \Theta \) as the
variable of differentiation. Further, we will rewrite the recursion and base functions in a more general notation in order to more easily study how the derivatives of the paths can be calculated recursively. Letting $\Theta := (\Theta_1, \ldots, \Theta_5) = (S_0, B, \mu, \sigma, \Delta t)$, (2.10) and (2.11) can be written as

\begin{align*}
p_t(\Theta, u) &= f(s, \theta)_{|s=S_t(\Theta, u), \theta=\Theta}, \\
S_{t+1}(\Theta, u) &= g(\pi, s, \theta, \omega)_{|\pi=p_t(\Theta, u), s=S_t(\Theta, u), \omega=u(t), \theta=\Theta},
\end{align*}

with $u = (u^{(1)}, \ldots, u^{(T)})$ and recursion base function $S_0(\Theta, u) = \Theta_1$.

For (2.10) and (2.11), this leads to

\begin{align*}
f(s, \theta) &= \Phi\left(\frac{1}{4\sqrt{\theta_5}}\left(\log\left(\frac{\theta_2}{s}\right) - \left(\theta_3 - \frac{\theta_4^2}{2\theta_5}\right)\right)\right), \\
g(\pi, s, \theta, \omega) &= s \exp\left(\left(\theta_3 - \frac{\theta_4^2}{2}\right)\theta_5 + \theta_4\sqrt{\theta_5} \Phi^{-1}(\pi \omega)\right).
\end{align*}

Using this notation, the present value of the option in (2.9) can be written as

\begin{equation}
P_{V_0}(\Theta) = \int_0^1 \cdots \int_0^1 e^{-rT\Delta t} q^*(\Theta, u) \, du^{(T)} \cdots du^{(1)},
\end{equation}

where $q^*(\Theta, u)$ is the one-step survival payoff defined by

\begin{equation}
q^*(\Theta, u) := p_0(\Theta, u) \cdots \cdots p_{T-1}(\Theta, u) \cdot q(S_T(\Theta, u)).
\end{equation}

Using this notation, we can formulate the following result.

**Theorem 2.4** The partial derivatives of the present value of a knock-up-and-out barrier option with payoff (2.2) with respect to $\Theta_1, \ldots, \Theta_4$ are given by

\begin{equation}
\frac{\partial P_{V_0}}{\partial \Theta_i}(\Theta) = e^{-rT\Delta t} \int_0^1 \cdots \int_0^1 \frac{\partial (q^*(\Theta, u))}{\partial \Theta_i} \, du^{(T)} \cdots du^{(1)},
\end{equation}
where \( \partial q^*(\theta, u)/\partial \theta_i \) are the derivatives of the one-step survival payoff (2.17) given by

\[
\frac{\partial q^*}{\partial \theta_i}(\theta, u) = 1_{S_T(\theta, u) > K} \frac{\partial S_T}{\partial \theta_i}(\theta, u) \cdot \prod_{j=0}^{T-1} p_j(\theta, u)
\]

\[+ q(S_T(\theta, u)) \cdot \sum_{j=0}^{T-1} \frac{\partial p_j}{\partial \theta_i}(\theta, u) \cdot \prod_{k \neq j} p_k(\theta, u). \tag{2.19} \]

The derivatives of \( p_T(\theta, u) \) and \( S_T(\theta, u) \) are recursively given by

\[
\frac{\partial p_T}{\partial \theta_i}(\theta, u) = \frac{\partial f}{\partial s}(s, \vartheta) \bigg|_{s = S_T(\theta, u), \vartheta = \theta} + \frac{\partial f}{\partial \theta_i}(s, \vartheta) \bigg|_{s = S_T(\theta, u), \vartheta = \theta},
\]

\[
\frac{\partial S_{T+1}}{\partial \theta_i}(\theta, u) = \frac{\partial g}{\partial s}(\pi, s, \vartheta, \omega) \bigg|_{\pi = p_T(\theta, u), s = S_T(\theta, u), \omega = \omega^{(t)}, \vartheta = \theta} + \frac{\partial g}{\partial \pi}(\pi, s, \vartheta, \omega) \bigg|_{\pi = p_T(\theta, u), s = S_T(\theta, u), \omega = \omega^{(t)}, \vartheta = \theta} \frac{\partial p_T}{\partial \theta_i}(\theta, u)
\]

\[+ \frac{\partial g}{\partial \theta_i}(\pi, s, \vartheta, \omega) \bigg|_{\pi = p_T(\theta, u), s = S_T(\theta, u), \omega = \omega^{(t)}, \vartheta = \theta} \frac{\partial S_T}{\partial \theta_i}(\theta, u). \tag{2.20} \]

\[
\frac{\partial S_0}{\partial \theta_1}(\theta, u) = 1, \quad \frac{\partial S_0}{\partial \theta_i}(\theta, u) = 0 \quad \forall i > 1. \tag{2.22} \]

**Proof** Knowing that we have compact domains, a Lipschitz-continuous integrand and we have not taken the derivatives with respect to \( r, T \) or \( \Delta t \), (2.18) follows through the below calculation:

\[
\frac{\partial PV_{T_0}}{\partial \theta_i}(\theta) = \frac{\partial (\int_0^1 \cdots \int_0^1 e^{-rT\Delta t} q^*(\theta, u) \, du^{(T)} \cdots du^{(1)})}{\partial \theta_i}
\]

\[= \int_0^1 \cdots \int_0^1 \frac{\partial (e^{-rT\Delta t} q^*(\theta, u))}{\partial \theta_i} \, du^{(T)} \cdots du^{(1)}. \]

Equations (2.19)–(2.21) are calculated from (2.17), (2.12) and (2.13) using the product rule. \( \square \)

We obtain expressions for \( D^{(\theta_1, \ldots, \theta_4, s)}(f(s, \vartheta)) \) and \( D^{(\theta_1, \ldots, \theta_4, s, \pi)}(g(\pi, u, s, \vartheta)) \) using our previously introduced model. These can be viewed as (A.1) and (A.2) in the online appendix.

Theorem 2.4 leads to the following unbiased one-step survival pathwise sensitivities Monte Carlo estimator.
COROLLARY 2.5 The one-step survival pathwise sensitivities Monte Carlo estimator with respect to \((\Theta_1, \ldots, \Theta_4)\) of the present value of a knock-up-and-out barrier option given by the average

\[
\overline{D_{\Theta_i} \text{PV}_N} = e^{-r(T-t_0)}
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} \left( 1_{s_{T,n}>K} \frac{\partial S_{T,n}}{\partial \Theta_i}(\Theta, u) \cdot \prod_{j=0}^{T-1} p_{j,n}(\Theta, u) 
+ q(s_{T,n}(\Theta, u)) \cdot \prod_{j=0}^{T-1} \left[ \frac{\partial p_{j,n}}{\partial \Theta_i}(\Theta, u) \prod_{k=0, k \neq j}^{T-1} p_{k,n}(\Theta, u) \right] \right)
\]

is unbiased.

Note that if we are interested in second-order Greeks, the indicator function in the final step can be smoothed out by using a combination of the methods in this section and the next, forcing the path to stay between \(B\) and \(K\).

COROLLARY 2.6 The second-order one-step survival pathwise sensitivities Monte Carlo estimator with respect to \((\Theta_1, \ldots, \Theta_4)\) of the present value of a knock-up-and-out barrier option is given by

\[
\overline{D_{\Theta_i} \text{PV}_N}
\]

\[
= e^{-r(T-t_0)}
\]

\[
= \frac{1}{N} \sum_{n=1}^{N} \left( \frac{\partial^2 S_{T,n}}{\partial \Theta_i^2}(\Theta, u) \cdot \prod_{j=0}^{T-1} p_{j,n}(\Theta, u) 
+ 2 \frac{\partial S_{T,n}}{\partial \Theta_i}(\Theta, u) \cdot \sum_{j=0}^{T-1} \left[ \frac{\partial p_{j,n}}{\partial \Theta_i}(\Theta, u) \prod_{k=0, k \neq j}^{T-1} p_{k,n}(\Theta, u) \right] 
+ q(s_{T,n}(\Theta, u)) \cdot \prod_{j=0}^{T-1} \left[ \frac{\partial^2 p_{j,n}}{\partial \Theta_i^2}(\Theta, u) \prod_{k=0, k \neq j}^{T-1} p_{k,n}(\Theta, u) \right] \right)
\]

\[
\times \frac{1}{N} \sum_{n=1}^{N} \left( \frac{\partial^2 S_{T,n}}{\partial \Theta_i^2}(\Theta, u) \cdot \prod_{j=0}^{T-1} p_{j,n}(\Theta, u) 
+ 2 \frac{\partial S_{T,n}}{\partial \Theta_i}(\Theta, u) \cdot \sum_{j=0}^{T-1} \left[ \frac{\partial p_{j,n}}{\partial \Theta_i}(\Theta, u) \prod_{k=0, k \neq j}^{T-1} p_{k,n}(\Theta, u) \right] 
+ q(s_{T,n}(\Theta, u)) \cdot \prod_{j=0}^{T-1} \left[ \frac{\partial^2 p_{j,n}}{\partial \Theta_i^2}(\Theta, u) \prod_{k=0, k \neq j}^{T-1} p_{k,n}(\Theta, u) \right] \right)
\]

\[
+ \frac{\partial p_{j,n}}{\partial \Theta_i}(\Theta, u) \prod_{k=j+1}^{T-1} p_{k,n}(\Theta, u) \sum_{k=0}^{j-1} \left[ \frac{\partial p_{k,n}}{\partial \Theta_i}(\Theta, u) \prod_{m=j+1, m \neq k}^{T-1} p_{m,n}(\Theta, u) \right] 
+ \frac{\partial^2 p_{j,n}}{\partial \Theta_i^2}(\Theta, u) \prod_{k=0}^{T-1} p_{k,n}(\Theta, u) \sum_{k=0}^{j-1} \left[ \frac{\partial p_{k,n}}{\partial \Theta_i}(\Theta, u) \prod_{m=0, m \neq k}^{T-1} p_{m,n}(\Theta, u) \right]
\]
While (A.1) and (A.2) give the results for the pathwise sensitivities estimator of Theorem 2.5, the necessary derivatives can be calculated via an automatic-differentiation-like method using, for example, an easy MATLAB script (MATLAB 2015). To explain this idea in more detail, we present Algorithm 1, which uses the Symbolic Math Toolbox™ of MATLAB for the differentiation.

The algorithm calculates the necessary derivatives for (2.20) and (2.21) and inserts these into the Monte Carlo simulation, while the derivatives of the payoff are calculated manually from Theorem 2.5.

A straightforward procedure holds for second-order Greeks while using similar MATLAB commands for second-order differentiation and taking the derivatives of the payoff out of Theorem 2.6. For a better understanding, since Algorithm 1 uses the syntax and some functions of MATLAB, we will explain the code row by row in the following paragraph.

In rows 3 and 4, we define the functions (2.14) and (2.15), where all variables are handled in a symbolic way. To generate the symbolic expressions, we use the MATLAB function `syms`. From row 7 to row 13, the algorithm calculates the symbolic derivatives of $f$ with respect to $\partial_1, \ldots, \partial_4$ and $s$, and of $g$ with respect to $\partial_1, \ldots, \partial_4$, $s$ and $\pi$, with the symbolic MATLAB function `jacobian`. Recall that in this script these are equal to matrixes (A.1) and (A.2) of the online appendix. After calculating the symbolic derivatives, the algorithm converts the symbolic expressions to function handles with `matlabFunction()`.

After defining the functions, at row 17 the algorithm starts the Monte Carlo simulation. In addition to the asset price and survival value simulation in rows 23 to 25, the algorithm calculates the derivatives of these in rows 27 and 28, as in the formulas derived in (2.20) and (2.21).

After the simulation of the paths, the algorithm calculates first, in row 35, the one-step survival payoff as in Corollary 2.3 and then, from rows 37 to 48, the pathwise sensitivities of the option (Delta, Beta, Rho, Vega) as in Theorem 2.5, with a slight
Algorithm 1 One-step survival pathwise sensitivities estimator with respect to $S_0, B, \mu, \sigma$ (Delta, Beta, Rho, Vega) of a knock-up-and-out barrier option.

1: % symbolic definition of (2.14) and (2.15)
2: syms $\theta_1, \ldots, \theta_5, u, \pi, s$
3: $f = \Phi(\ln(\theta_2/\theta_3 - \theta_4^2/2)\theta_5)/(\theta_4\sqrt{\theta_5})/\Phi^{-1}(u \cdot \pi))$ % $g(\pi, s, \theta, \omega)$
4: $g = s \cdot \exp((\theta_3 - \theta_4^2/2)\theta_5 + \theta_4\sqrt{\theta_5} \cdot \Phi^{-1}(u \cdot \pi))$ % $g(\pi, s, \theta, \omega)$
5: % symbolic partial derivatives of $f$ and $g$, resulting in (A.1) to (A.2) and the conversion of the symbolic expressions to function handles
6: $f = \text{matlabFunction}(f)$
7: $g = \text{matlabFunction}(g)$
8: $D_{\theta} f = \text{matlabFunction}($jacobian$(f, [\theta_1, \ldots, \theta_4]))$
9: $D_{s} f = \text{matlabFunction}($jacobian$(f, s))$
10: $D_{\theta} g = \text{matlabFunction}($jacobian$(g, [\theta_1, \ldots, \theta_4]))$
11: $D_{s} g = \text{matlabFunction}($jacobian$(g, s))$
12: $D_{\pi} g = \text{matlabFunction}($jacobian$(g, \pi))$
13: % Monte Carlo simulation
14: % Initialize random seed
15: Initialize model parameters $\Theta = (\Theta_1, \ldots, \Theta_8) = (S_0, B, \mu, \sigma, \Delta t, r, K, T)$
16: for $n = 1, \ldots, N$ do
17: % base derivative recursion vector as in (2.22)
18: $D_{\theta} S_0 = [1, 0, 0, 0]$
19: for $j = 0 : T - 1$ do
20: % simulate paths as in (2.12) and (2.13)
21: $p_j := f(S_j, \Theta)$
22: Sample $u \sim U(0, 1)$
23: $S_{j+1} := g(p_j, S_j, \Theta, u)$
24: % simulate derivatives of paths as in (2.20) and (2.21)
25: $D_{\theta} p_j := D_s f(S_j, \Theta) \cdot D_{\theta} S_j + D_{\theta} g(p_j, S_j, \Theta, u)$
26: $D_{\theta} S_{j+1} := D_s g(p_j, S_j, \Theta, u) \cdot D_{\theta} S_j$
27: $+ D_{\pi} g(p_j, S_j, \Theta, u) \cdot D_{\theta} p_j + D_{\theta} g(p_j, S_j, \Theta, u)$
28: end for
29: % calculate price as in Corollary 2.3
30: $P_n := \text{prod}(p) \cdot \max(S_T - K, 0)$
31: % calculate the pathwise sensitivities as in Theorem 2.5
32: $D_{\theta} P_n := [1]_{S_T > K} \cdot D_{\theta} S_T \cdot \text{prod}(p)$
33: for $i = 1, \ldots, T$ do
34: $D_{\theta} P_n := D_{\theta} P_n + \max(S_T - K, 0) \cdot D_{\theta} p_i \cdot \text{prod}(p)/p_i$
35: end for
36: end for
37: return $PV_{t_0} := e^{-r \cdot T \Delta t} \sum_{n=1}^{N} P_n, D_{\theta} PV_{t_0} = e^{-r \cdot T \Delta t} \sum_{n=1}^{N} D_{\theta} P_n$
modification emphasizing the relevance of hand coding that we now explain in more
detail: in terms of complexity, Theorem 2.5 tends to be $O(NT^2)$, with $N$ Monte
Carlo simulations and $T$ observations. As seen in row 36, we can reduce the com-
plexity to $O(NT)$ by replacing $\prod_{k=0}^{T-1} p_k$ with $(\prod_{k=0}^{T-1} p_k)/p_j$, since the product
can be precalculated.

Note that we did not have any problems with $p_j$ close to zero. On the one hand,
there are no cancellation or absorption problems with division; on the other hand,
if $p_j = 0$ (very unlikely for Black–Scholes, but we could imagine it at the last
step if using a numerical approximation, such as Euler–Maruyama, for more general
models), the division would not be necessary, since the path would not “survive”.
Nonetheless, $\prod_{k=0,k\neq j}^{T-1} p_k$ could alternatively be computed through the multiplica-
tion of two precalculated cumulative products (increasing and decreasing, respec-
tively) without increasing the order. Nevertheless, the preallocation increases the
memory usage. Further, we could use similar ideas to reduce the complexity of
the second-order Greeks of Theorem 2.6, ie, almost $O(NT^3)$, to an optimal com-
plexity of $O(NT)$. We next explain some necessary modifications: $\prod_{k=0}^{j-1} p_k$ and
$\prod_{k=j+1}^{T-1} p_k$ are exactly the cumulative products described for Delta. The remaining
expressions can be modified through recursion formulas of the form

$$\sum_{k=0}^{j} \left( \frac{\partial p_k}{\partial \Theta_i} \prod_{m=0,m\neq k}^{j} p_m \right) = p_j \sum_{k=0}^{j-1} \left( \frac{\partial p_k}{\partial \Theta_i} \prod_{m=0,m\neq k}^{j-1} p_m \right) + \frac{\partial p_j}{\partial \Theta_i} \prod_{i=0}^{j-1} p_i,$$

ie, starting with $j = 1$, we can recursively calculate the expressions, while saving
them in a vector. Finally, the vectors can be summed in order of $T$.

We now give some remarks on our particular implementation in MATLAB. In
the MATLAB version we used, the function $\text{matlabFunction()}$ sets the input
parameters in alphabetical order, which we therefore considered in the algorithm.

To implement the algorithm, the normal distribution and the inverse normal
distribution should be replaced by the following formulas:

$$\Phi(x) = 0.5 \left( \text{erf} \left( \frac{x}{\sqrt{2}} \right) + 1 \right),$$

$$\Phi^{-1}(x) = \text{inverf}(2x - 1)\sqrt{2},$$

since MATLAB is unable to differentiate symbolically the normal inverse cumulative
distribution function ($\text{norminv}$) at this time.

### 2.2 One-step survival for other types of barrier options

In this section, we will briefly explain the changes that need to be made for other
types of barrier options. For knock-up-and-out barrier options, we used (2.5) for the
substitution. However, for knock-down-out options, defined by

\[ V(S_1, \ldots, S_T) = \begin{cases} (S_T - K)^+ = q(S_T) & \text{if } \min_{j=1,\ldots,T} S_j \geq B, \\ 0 & \text{otherwise,} \end{cases} \]  

(2.23)

the path survives while staying above the barrier. Thus, after splitting the integral and with a modified normalization, we obtain

\[ PV_{t_0}(S_0) = e^{-r\Delta t} (1 - p_0) \int_{S_1(z) \geq B} \frac{\phi(z)}{(1 - p_0)} PV_{t_1}(S_1(z)) \, dz \]  

(2.24)

at the first observation date. Now, demanding an independent and compact domain of integration, we use

\[ z = \Phi^{-1}((1 - p_0)u + p_0) \]  

(2.25)

for the substitution. Now, similarly to Theorem 2.2 and Corollary 2.3, we could formulate similar results for knock-down-out options. At this point, we just present the essential consequences for Theorem 2.4.

First, we determine that, instead of (2.17), we obtain the modified one-step survival payoff

\[ q^*(\Theta, u) := (1 - p_0(\Theta, u)) \cdot \cdots \cdot (1 - p_{T-1}(\Theta, u)) \cdot q(S_T(\Theta, u)), \]  

(2.26)

and, instead of (2.15), we have to use

\[ g(\pi, u, s, \vartheta) = s \cdot \exp \left( \left( \vartheta_3 - \frac{\vartheta_4^2}{2} \right) \vartheta_5 + \vartheta_4 \sqrt{\vartheta_5} \Phi^{-1}((1 - \pi)u + \pi) \right). \]  

(2.27)

All in all, to obtain the one-step survival Monte Carlo estimator for the pathwise sensitivities and the payoff of a knock-down-out barrier option, Algorithm 1 only has to be modified at row 4 with the new \( g(\pi, s, \vartheta, \omega) \) from (2.27) and at rows 31 to 37 with the new payoff (2.26) and its derivatives, calculated straightforwardly by hand, as seen in (2.19).

The techniques introduced cannot easily be applied to knock-in options, since none of the split integrals becomes zero.

For the digital knock-in barrier options defined by

\[ V(S_1, \ldots, S_T) = \begin{cases} c =: q(S_T) & \text{if } \max_{j=1,\ldots,T} S_j \geq B, \\ 0 & \text{otherwise,} \end{cases} \]  

(2.28)

we can similarly apply one-step survival, since one part of the split integrals will be constant and does not need to be simulated any further. By splitting, normalizing and
substituting the integral at the first observation date, with (2.5) for the first summand and (2.25) for the second, we obtain

$$PV_{t_0}(S_0) = e^{-r\Delta t} \left( (1 - p_0) \cdot c + p_0 \int_0^1 PV_{t_1}(S_1^{(1)}(u)) \, du \right),$$

with (2.10) for $p_0$ and (2.11) for $S_1^{(1)}(u)$, since the payoff will be $c \in \mathbb{R}$ if the underlying hits the barrier.

All in all, we see that the recursion formulas from Section 2.1 hold here, and a theorem for the present value and the sensitivities can be formulated analogously to Theorems 2.2 and 2.4 using the modified one-step survival payoff

$$q^*(\Theta, u) = c((1 - p_0) + p_0(1 - p_1) + \cdots + p_0 \cdots p_{T-2}(1 - p_{T-1}))$$

and its straightforward calculated derivatives.

Finally, we remark that the theory and Algorithm 1 can be adjusted straightforwardly for knock-down-in digital options with these results. As mentioned before, it is not straightforward to use the techniques presented for general (nondigital) knock-in barrier options. However, in practice we can use the in–out parity

$$\frac{\partial (V^{\text{knock-in}}(S, T))}{\partial \Theta_i} = \frac{\partial (V^{\text{opt}}(S, T))}{\partial \Theta_i} - \frac{\partial (V^{\text{knock-out}}(S, T))}{\partial \Theta_i}, \quad (2.29)$$

since for the differentiation of the nonbarrier option we can use, for example, the standard pathwise sensitivities approach from Glasserman (2003). We note briefly that if the variable $\Theta_i$ is the barrier $B$, the derivative of the plain option $\partial (V^{\text{opt}}(S, T)) / \partial \Theta_i$ will become zero.

3 NUMERICAL RESULTS

In this section, we provide some numerical results for the new algorithm. Therefore, we consider a simple discretely observed up-and-out barrier option in two different settings, one with 50 observations and one with 360 observations before maturity. We will use the parameter values of Table 1, where the example is fictitious.

In the first column of Figure 3, we see the estimated value of the option as a function of the initial asset price at $t_0$. The results of the standard Monte Carlo estimator and the one-step survival Monte Carlo estimator are plotted with a blue (dashed) line and a red (solid) line, respectively. From top to bottom, the rows use $N = 10^2, 10^3, 10^4, 10^5$ Monte Carlo samples, respectively.\(^1\) For each of these

\(^1\) Color figures are available in the online version of this paper.
TABLE 1 Parameters for both up-and-out barrier options, which differ only in the number of observation dates.

| Parameter | Value |
|-----------|-------|
| $t_0$     | 0     |
| $t_T$     | 1     |
| $S_0$     | 50    |
| $B$       | 60    |
| $T$       | (50,360) |
| $r$ (%)   | 10    |
| $b$ (%)   | 0     |
| $\sigma$ (%) | 20 |
| $K$       | 50    |

$T$ is the number of observation dates. The observations are distributed equidistantly until maturity. For this reason, the first observation will be at $(t_0 - t_T)/T = 1/50$ for the first example and at $1/360$ for the second.

calculations and the following ones, the same random seed was used. The second column of Figure 3 shows the first derivative of the option value with respect to the underlying price $S_0$ (the Delta) calculated by applying central finite differences,

\[
\frac{\hat{PV}(S_0 + \delta_s) - \hat{PV}(S_0 - \delta_s)}{2\delta_s},
\]

with $\delta_s$ chosen as 0.5% of $S_0$. The third column of Figure 3 compares the one-step survival finite-difference derivatives and the new one-step survival pathwise derivatives. Figure 4 shows similar results for an option with 360 observation dates.

The plots clearly demonstrate the instability of the standard Monte Carlo estimator with respect to numerical differentiation and the stability of the one-step survival Monte Carlo estimator, as already mentioned in Alm et al (2013). Further, we see that the pathwise sensitivities are quite close to the finite-difference results. Hence, we want to take a deeper look at the comparison of these two methods for sensitivity computation, as, being more complex, the pathwise sensitivity approach, as developed in (2.20) and (2.21), does not need any additional path for the sensitivity computation. We present in Table 2 the processing times (computed on a single core of an Intel i7-4790 central processing unit (CPU)) of the first- and second-order pathwise sensitivities and finite differences (calculated through first- and second-order difference quotients), respectively, for both parameter settings. Further, we compare the absolute errors for Delta and Gamma in Figures 5 and 6.

First, we determine that the considerations and modifications made regarding the complexity with respect to the observations hold.

Next, we see that the new pathwise estimator does not perform much better with regard to the absolute error (except without discretization error), but, as can be seen
from Table 2, it takes substantially less time to calculate finite differences, with a ratio of approximately 0.71 for Delta and 0.8 for Gamma, respectively. Changing the number of observations or having a barrier level near the spot price can increase or decrease the variance of the payoff or change the payoff structure. Therefore, an improperly chosen $\delta_s$ could lead to unfeasible results (discretization error) as we can see, for example, by comparing the Gamma simulations using $\delta_s = 10(0)$ (red dashed line) in Figures 5 and 6.

3.1 Calibration of a CoCo bond

In this section, we take a closer look at the time-saving factor for multiple Greeks, and we determine another benefit of pathwise sensitivities. Since it is independent of discretization errors, the new approach provides a larger radius of convergence for calibrations.
CoCo bonds are debt instruments converting upon a certain trigger event. There are several common ways of defining this trigger, such as an accounting trigger, a multivariate trigger or a regulatory trigger (see, for example, De Spiegeleer and Schoutens (2012) for explanations).

Here, to model CoCo bonds, we will use the equity derivative approach of De Spiegeleer and Schoutens (2012). As they explain, this approach does not model the true trigger event, but rather uses an approximate model where the bond is triggered whenever the underlying falls below a certain level $B$, where $B$ is to be calibrated from market data.

Let us stress that this makes calibration problems for CoCo bonds fundamentally different than those for standard barrier options. For the pricing of standard barrier options, we can completely avoid calibration problems with discontinuous
TABLE 2  Computation (CPU) times in seconds.

| $T$  | Present value | Delta: PW (FD)   | Delta + Gamma: PW (FD) |
|------|---------------|------------------|------------------------|
| 50   | 0.2285        | 0.3289 (0.4571)  | 0.5565 (0.6857)        |
| 360  | 1.56283       | 2.2407 (3.1256)  | 3.7560 (4.6885)        |

The table shows the computation times of the one-step survival algorithm for the present value, Delta and Gamma (including Delta) with pathwise sensitivities (PW) and central finite differences (FD, given in parentheses) for both options computed with fixed Monte Carlo sample size ($10^5$). $T$ denotes the number of observations.

FIGURE 5  Absolute error for (a) the Delta and (b) the Gamma of the pathwise sensitivity approach (green solid line) versus finite-differences approaches, with $\delta_s = 10^{-0}$, $\delta_s = 10^{-1}$, $\delta_s = 10^{-2}$, $\delta_s = 10^{-3}$ (blue, red, black and yellow dashed lines, respectively) for $T = 50$.

The figures show the absolute error using fixed $S_0 = 50$, $B = 60$ and $T = 50$ depending on the number of Monte Carlo simulations.

payoffs, since the barrier is a known pay-off feature, and other model parameters such as $\mu$ and $\sigma$ are model parameters of the underlying and can thus be calibrated through, eg, European options. However, for CoCo bonds, the barrier $B$ is a model parameter that can only be determined by calibrating the CoCo bond itself, assuming the Spiegeleer–Schoutens model (De Spiegeleer and Schoutens 2012). Thus, CoCo bonds require stable calibration methods for discontinuous payoffs.

Note also that, while a one-dimensional calibration of $B$ is unavoidable, the calibration of $\mu$ and $\sigma$ can be done using either standard European option market prices (for simpler calibration) or CoCo bond market prices (for greater consistency).
The figures show the absolute error using fixed parameters $S_0 = B = 50$, $T = 360$ and $K = 40$ depending on the number of Monte Carlo simulations.

The latter leads to multidimensional calibration problems for discontinuous barrier options. Also, multidimensional calibration seems unavoidable for multivariate underlyings or when the barrier $B$ is modeled to be time dependent. To demonstrate the feasibility of our approach for multidimensional calibration problems while remaining comparable with the example in De Spiegeleer and Schoutens (2012), we will employ the example of a univariate CoCo bond and model five variable time-dependent barrier levels.

Since we use the equity derivative approach of De Spiegeleer and Schoutens (2012), the holder obtains coupons as long as the trigger event has not occurred. The trigger event is observed discretely at the dates of coupon payments. In contrast to coupons for plain corporate bonds, coupons for these CoCo bonds can stop when the trigger event occurs. This is valued as a short position in a binary down-and-in barrier option. We notice that, for every coupon, there is indeed a corresponding short position in a binary option that is knocked in at the same barrier.

The parameters used are given in Table 3 (see De Spiegeleer and Schoutens (2012) for further information on payoff structures). For the individual barrier levels, we modeled a convex "smile" pattern.
TABLE 3 Parameters of the model and the CoCo bond price computations in ascending semiannual sequence (0, 0.5, 1, 1.5, 2).

| Parameter | Value |
|-----------|-------|
| $t$       | 4.5   |
| $r$ (%)   | 3.42  |
| $\sigma$ (%) | 10 |
| $(s_0, s_{0.5}, s_1, s_{1.5}, s_2)$ | (0.6075, 0.61, 0.6025, 0.6125, 0.605) |
| $c_p$     | 0.5900|
| $(b_{0.5}, ..., b_2)$ | 0.5 |
| $(b_{2.5}, b_3, b_{3.5}, b_4, b_{4.5})$ | (0.51, 0.48, 0.45, 0.49, 0.52) |
| Coupon (%) | 30    |
| Frequency | Semiannually |
| Face value | 100  |
| Conversation ratio | 100 |

The first five barrier levels are assumed to be constant.

TABLE 4 CPU time of a single iteration.

| # simulations | OSS       | PW        |
|---------------|-----------|-----------|
| $10^7$        | 10 669.20 | 2 196.17  |
| $10^6$        | 1 067.34  | 219.7     |
| $10^5$        | 106.02    | 22.00     |

CPU time (in seconds) of one iteration step (including all derivatives) of the one-step survival estimator (OSS) using finite differences and the one-step survival pathwise sensitivity estimator (PW).

We calculated the benchmark prices of the CoCo bond with a large number of Monte Carlo samples. The results are given in the following order. First, we compare the computation processing time (Table 4) of one-step survival using finite differences and the one-step survival pathwise sensitivities for a single calibration iteration step (ie, present value and all derivatives). Next, we compare the calibrations using a gradient-based algorithm (Table 5), starting with the initial vector $[0.4, 0.4, 0.4, 0.4, 0.4]$. Finally, using a slightly modified initial vector, namely $[0.3, 0.3, 0.3, 0.3, 0.3]$, we determine the increased radius of convergence (Table 6) using the new pathwise sensitivity approach.

For illustrative purposes, all MATLAB optimization algorithms are used as “black boxes”, without any modification and without any additional data.

Studying the results of lsqnonlin (see Table 5), which is based on a trust-region-reflective method, we see the expected weakness of the standard Monte Carlo.
TABLE 5  Calibration results for \texttt{lsqnonlin} with initial values (0.4, 0.4, 0.4, 0.4, 0.4).

(a) Standard

| # MC | Iterations | Result                  | Resnorm |
|------|------------|-------------------------|---------|
| $10^7$ | 0          | (0.4000, 0.4000, 0.4000, 0.4000, 0.4000) | 152.3039 |
| $10^6$ | 0          | (0.4000, 0.4000, 0.4000, 0.4000, 0.4000) | 152.1053 |
| $10^5$ | 0          | (0.4000, 0.4000, 0.4000, 0.4000, 0.4000) | 147.0367 |

(b) One-step survival

| # MC | Iterations | Result                  | Resnorm |
|------|------------|-------------------------|---------|
| $10^7$ | 34         | (0.5100, 0.4766, 0.4677, 0.4887, 0.5194) | 3.1451e−07 |
| $10^6$ | 43         | (0.5071, 0.4952, 0.4761, 0.4710, 0.5165) | 4.3222e−05 |
| $10^5$ | 84         | (0.4998, 0.5097, 0.4984, 0.4319, 0.4713) | 9.5003e−06 |

(c) One-step survival pathwise sensitivities

| # MC | Iterations | Result                  | Resnorm |
|------|------------|-------------------------|---------|
| $10^7$ | 34         | (0.5102, 0.4753, 0.4661, 0.4883, 0.5202) | 4.3762e−07 |
| $10^6$ | 43         | (0.5071, 0.4952, 0.4761, 0.4710, 0.5165) | 4.3222e−05 |
| $10^5$ | 84         | (0.4998, 0.5096, 0.4984, 0.4305, 0.4713) | 9.4110e−06 |

Results of calibrations using \texttt{lsqnonlin} with initial values (0.4, 0.4, 0.4, 0.4, 0.4) for the standard and one-step survival (both with finite differences) and one-step survival pathwise sensitivities Monte Carlo estimator. Depending on the number of Monte Carlo simulations (# MC), the table shows the number of iterations taken (iterations), the solution returned (result) and the squared norm of the residual (resnorm). The true value was (0.51, 0.48, 0.45, 0.49, 0.52).

Even with a huge number of samples, the standard Monte Carlo estimator is unable to provide viable solutions, since it aborts instantly. Hence, for calibrations with standard Monte Carlo estimators, we would typically resort to regularized differentiation schemes. The calibrations using the pathwise sensitivity approach deliver similar residuals with a similar number of iterations in approximately one-fifth of the time (as one-step survival with finite differences; see Table 4). Hence, we clearly see the expected strength of the pathwise sensitivities for multidimensional cases. Further, we determine (see Table 6) that the new approach provides a greater radius of convergence, while finite differences deliver unfeasible results.

We note that, while experimenting with these multidimensional calibrations, we also detected several advantages of one-step survival over standard Monte Carlo, using a gradient-free algorithm (eg, \texttt{fminsearch}). This effect is based on the idea that unstable differentiation also influences gradient-free algorithms.
TABLE 6  Calibration results for lsqnonlin with initial values (0.3, 0.3, 0.3, 0.3, 0.3).

(a) One-step survival

| # MC | Iterations | Result | Resnorm |
|------|------------|--------|---------|
| $10^7$ | 56 | (0.3000, 0.3054, 0.5270, 0.3720, 0.3240) | 4.5501 |
| $10^6$ | 56 | (0.3000, 0.3072, 0.5269, 0.3711, 0.3194) | 4.4726 |
| $10^5$ | 59 | (0.3000, 0.3001, 0.5265, 0.3880, 0.3981) | 4.5162 |

(b) One-step survival pathwise sensitivities

| # MC | Iterations | Result | Resnorm |
|------|------------|--------|---------|
| $10^7$ | 47 | (0.5101, 0.4782, 0.4618, 0.4881, 0.5204) | 5.4531e−07 |
| $10^6$ | 35 | (0.5071, 0.4963, 0.4669, 0.4756, 0.5167) | 4.3761e−05 |
| $10^5$ | 54 | (0.4994, 0.5109, 0.4934, 0.4564, 0.4770) | 2.7085e−05 |

Results of calibrations using lsqnonlin with initial values (0.3, 0.3, 0.3, 0.3, 0.3) for the one-step survival (with finite differences) and one-step survival pathwise sensitivities Monte Carlo estimator. Depending on the number of Monte Carlo samples (# MC), the table shows the number of iterations taken (iterations), the solution returned (result) and the squared norm of the residual (resnorm). The true value was (0.51, 0.48, 0.45, 0.49, 0.52).

4 CONCLUSIONS

We adjusted the idea of pathwise sensitivities to the idea of the one-step survival Monte Carlo method suggested by Glasserman and Staum (2001). It followed that we were able to calculate the pathwise sensitivities of options with discontinuous payoff, namely barrier options. From the numerical results, we saw that these derivatives behave stably and can be calculated efficiently in relation to finite differences, without evaluating a second or a third path. Thus, there appears to be no problem in choosing a discretization shift size for balancing accuracy and stability, which would depend on the input parameters and the underlying Greek.

In the case study, we calibrated a CoCo bond, which we modeled with time-dependent barrier levels and saw that the new pathwise sensitivity estimator outperformed one-step survival (with finite differences) in terms of computation time. Further, we determined another advantage, since, being without discretization error, the pathwise sensitivities improved the radius of convergence of the calibration.

This work can be extended to nonconstant parameters by using smaller time steps, as we can assume the parameters are constant at these steps. Then, at the observation dates, our algorithm can be applied with constant parameters and a smaller step width.

For a simplified presentation, we derived the algorithm within the classical Black–Scholes model. But, as we used a more general notation for the recursions of the
paths and their derivatives, this may be extended to other models, providing the idea of one-step survival is feasible. The basic concepts of generalizing one-step survival to the correlated multivariate case are given in Alm et al (2013), and we believe the method of pathwise sensitivities can also be applied. We divide the multivariate case into payoffs depending on the maximum or the minimum. We believe that pathwise sensitivities can easily be extended for the first case (maximum). However, we must be careful with the second case (minimum), since the resulting estimator is Lipschitz continuous but not necessarily differentiable. We also believe that the method can be adapted to more general models, eg, those whose underlying depends upon the solution of stochastic differential equations, by modifying the approximation method, eg, the Euler–Maruyama or Milstein scheme, while sampling the approximation conditional on survival at specified approximation steps. By considering discretely observed barrier options, this method leads to several of the usual approximation steps and one modified step at the observation date, ensuring that the path survives (see, for example, Giles (2009) for a similar combination idea). For continuously monitored barrier options, we could try to combine the one-step survival idea with the Brownian bridge interpolation in every step (see, for example, Glasserman 2003) to smooth out the discontinuities of the barrier-crossing probabilities. However, this would end in a loss of linearity, since many dependencies would be added. We also believe the method of pathwise sensitivities can be adapted to other kinds of options with path-dependent discontinuous payoff.

Finally, note that other variance reduction methods, such as control variates or antithetic sampling (Glasserman 2003), can be combined with the new algorithm.

DECLARATION OF INTEREST

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of the paper.

REFERENCES

Alm, T., Harrach, B., Harrach, D., and Keller, M. (2013). A Monte Carlo pricing algorithm for autocallables that allows for stable differentiation. The Journal of Computational Finance 17(1), 43–70 (https://doi.org/10.21314/JCF.2013.265).

Broadie, M., Glasserman, P., and Kou, S. (1997). A continuity correction for discrete barrier options. Mathematical Finance 7(4), 325–349 (https://doi.org/10.1111/1467-9965.00035).

Burgos, S., and Giles, M. B. (2011). The computation of Greeks with multilevel Monte Carlo. Preprint (arXiv:1102.1348).
Burgos, S., and Giles, M. B. (2012). Computing Greeks using multilevel path simulation. In Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 281–296. Springer Proceedings in Mathematics and Statistics, Volume 23. Springer (https://doi.org/10.1007/978-3-642-27440-4_13).

Capriotti, L. (2010). Fast Greeks by algorithmic differentiation. The Journal of Computational Finance 14(3), 3–35 (https://doi.org/10.21314/JCF.2011.234).

De Spiegeleer, J., and Schoutens, W. (2012). Pricing contingent convertibles: a derivatives approach. Journal of Derivatives 20(2), 27–36 (https://doi.org/10.3905/jod.2012.20.2.027).

Giles, M. B. (2008a). Improved multilevel Monte Carlo convergence using the Milstein scheme. In Monte Carlo and Quasi-Monte Carlo Methods 2006, Keller, A., Heinrich, S., and Niederreiter, H. (eds), pp. 343–358. Springer (https://doi.org/10.1007/978-3-540-74496-2_20).

Giles, M. B. (2008b). Multilevel Monte Carlo path simulation. Operations Research 56(3), 607–617 (https://doi.org/10.1287/opre.1070.0496).

Giles, M. B. (2009). Vibrato Monte Carlo sensitivities. In Monte Carlo and Quasi-Monte Carlo Methods 2008, L’Ecuyer, P., and Owen, A. B. (eds), pp. 369–392. Springer (https://doi.org/10.1007/978-3-642-04107-5_23).

Giles, M. B., and Glasserman, P. (2006). Smoking adjoints: fast Monte Carlo Greeks. Risk 19(1), 88–92.

Glasserman, P. (2003). Monte Carlo Methods in Financial Engineering. Stochastic Modelling and Applied Probability, Volume 53. Springer (https://doi.org/10.1007/978-0-387-21617-1).

Glasserman, P., and Staum, J. (2001). Conditioning on one-step survival for barrier option simulations. Operations Research 49(6), 923–937 (https://doi.org/10.1287/opre.49.6.923.10018).

Griewank, A., and Walther, A. (2008). Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd edn. SIAM, Philadelphia, PA (https://doi.org/10.1137/1.9780898717761).

Hull, J. (2008). Options, Futures and Other Derivatives, 7th edn. Prentice Hall.

Koster, F., and Rehmet, A. (2018). Monte Carlo payoff smoothing for pricing autocallable instruments. The Journal of Computational Finance 21(4), 59–77 (https://doi.org/10.21314/JCF.2018.340).

MATLAB (2015). Version 8.6.0.267246. Release R2015b. The MathWorks Inc., Natick, MA.

Naumann, U. (2012). The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation. Software, Environments and Tools, Volume 24. SIAM, Philadelphia, PA (https://doi.org/10.1137/1.9781611972078).

Naumann, U., and Toit, J. (2018). Adjoint algorithmic differentiation tool support for typical numerical patterns in computational finance. The Journal of Computational Finance 21(4), 23–57 (https://doi.org/10.21314/JCF.2018.339).

Seydel, R. (2006). Tools for Computational Finance, 3rd edn. Springer.

Zhang, P. G. (1998). Exotic Options: A Guide to Second Generation Options, 2nd edn. World Scientific (https://doi.org/10.1142/3800).
