Review of Derivatives Research (2024) 27:259–303
https://doi.org/10.1007/s11147-024-09206-z

Pricing and hedging autocallable products by Markov chain approximation

Yeda Cui[1] · **Lingfei Li**[1] · **Gongqiu Zhang**[2]

Accepted: 15 September 2024 / Published online: 4 October 2024

Abstract

We propose a unified pricing framework based on continuous-time Markov chain (CTMC) approximation for autocallable structured products. Our method is applicable to a variety of asset price models, including one-dimensional Markov jump-diffusions (the coefficients can be time dependent), regime-switching models, and stochastic local volatility (SLV) models. For SLV models, we develop a hybrid Markov chain approximation scheme that significantly improves the existing CTMC approximation method. We test our pricing method under various popular models and show that it is computationally efficient. To hedge autocallable products, we consider a dynamic hedging approach in the presence of transaction costs. To address the problem that the product's delta can become too large near the barriers, we apply payoff modification and barrier shifting techniques. We determine the optimal size of adjustments that minimize conditional value-at-risk (CVaR) of the hedging loss using stochastic gradient descent. Empirical experiments demonstrate the effectiveness of our approach in reducing CVaR of the hedging loss.

Keywords Autocallable · Markov chain approximation · Stochastic local volatility · Hedging · Payoff modification · Barrier shifting

JEL classification C63 · G13

✉ Lingfei Li
lfli@se.cuhk.edu.hk

Yeda Cui
ycui@se.cuhk.edu.hk

Gongqiu Zhang
zhanggongqiu@cuhk.edu.cn

[1] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Sha Tin District, Hong Kong SAR

[2] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China

1 Introduction

Autocallable notes are a type of structured product that link the principle and coupon payments to the evolution of one or multiple asset prices, providing investors with features of both coupon bonds and equity investment. The first autocallable notes were introduced by BNP Paribas in 2003 and their market volume has grown substantially over the past two decades (Kim & Lim, 2019; Paletta & Tunaru, 2022). Autocallable products are popular because they allow investors to earn a higher return than plain bonds by participating in the underlying asset while also providing principle protection to some extent. This feature makes them particularly appealing in a low interest rate environment. The yield enhancement in autocallable products is achieved through several typical provisions (Guillaume, 2015b).

- *Autocall*: At each monitoring date prior to maturity, an autocallable note is redeemed if the underlying asset price is above a predetermined level (autocall barrier), resulting in the investor receiving the principle and coupon. Additionally, if the asset price surpasses a higher level, the investor receives the return of this asset, which leads to a greater amount than the initial principle. The autocall mechanism provides investors with an opportunity to reach a target rate of return or higher in a shorter time period.
- *Snowball effect*: Coupon payments are ignored if the underlying asset price falls below a predetermined level (coupon barrier) at a monitoring date. However, any missed coupons are subsequently recouped and disbursed to the investor once the underlying asset price rises above the coupon barrier on a subsequent monitoring date.
- *Down-and-out American barrier*: The coupon payment on a monitoring date is made only if the price of the underlying asset remains above a predetermined level continuously for a specified duration from the previous monitoring date.

Due to their complex structure and path dependency, autocallable products pose significant challenges for pricing and risk management. Various methods have been proposed to deal with them. Guillaume (2015b) derive an explicit pricing formula for autocallable products under the Black–Scholes model and Guillaume (2015a) obtain an analytical pricing formula under Merton's jump-diffusion model for the equity with the interest rate following the Ho–Lee model. Deng et al. (2011) propose a finite difference approach for pricing autocallables, while Lee and Hong (2021); Fries and Joshi (2011), and Koster and Rehmet (2018) use Brownian bridge and payoff smoothing methods to reduce variance in Monte Carlo simulation. Paletta and Tunaru (2022) introduce a Bayesian approach to deal with parameter uncertainty in the pricing model. Kim and Lim (2019) propose a recursive approach for the static replication of autocallables with vanilla options, which simplifies the pricing and hedging of these products. Empirical studies on autocallables have been conducted by Deng et al. (2015) and Albuquerque et al. (2015).

In this paper, we propose a unified pricing framework based on continuous-time Markov chain (CTMC) approximation for autocallables with all the major

features mentioned above. The basic idea of CTMC approximation is to construct a continuous-time discrete state Markov process to approximate the original Markov model whose state space is continuous and then do computations under the approximating CTMC model, which is often tractable. This method has been applied to a wide range of financial models including general one-dimensional (1D) jump-diffusions (Mijatović & Pistorius, 2013), regime-switching Markov models (Cai et al., 2019), stochastic local volatility models (Cui et al., 2018), skew diffusions (Ding et al., 2021), and sticky diffusions (Meier et al., 2021, 2023). It has also proved to be computationally efficient for pricing a large class of derivatives, especially for path-dependent ones. Applications include European and barrier options (Mijatović & Pistorius, 2013), Asian options (Cai et al., 2015), lookback options (Zhang & Li, 2021), Parisian options (Zhang & Li, 2023a), drawdown options (Zhang et al., 2021; Li et al., 2024; Zhang & Li, 2023), equity swaps and caps/floors (Kirkby, 2023). Results about convergence rates can be found in Mijatović and Pistorius (2013); Li and Zhang (2018), and Zhang and Li (2019, 2022).

Our paper contributes to two strands of literature on autocallable products and CTMC approximation for derivatives pricing. The contributions are threefold and can be summarized as follows.

First, we develop a general pricing method for autocallables that is computationally efficient. We derive pricing recursions for autocallables over different monitoring dates, which are computed using matrix exponentiation under CTMC approximation. In contrast to previous research on autocallables, our approach is applicable to a broad range of models with diverse characteristics, including jumps with finite or infinite activity, regime-switching, local and stochastic volatility, and time dependence. The versatility of our approach is particularly useful for autocallables, as these products often require sophisticated stochastic models of the underlying asset for pricing. An important but tricky issue in applying CTMC approximation is grid design. As the autocallable payoff has many discontinuities, convergence of CTMC approximation can be slow if the grid is not designed properly. We solve this problem for autocallables based on the theoretical results of Zhang and Li (2019) to achieve stable and fast convergence. We test our pricing algorithm in various popular financial models including Black-Scholes (BS), CEV, SABR, Heston, Kou, and Variance Gamma (VG). In all the cases, we obtain highly accurate results with small computation time.

Second, we propose a hybrid Markov chain approximation method for stochastic local volatility (SLV) models that significantly improves the original two-layer CTMC approximation algorithm developed in Cui et al. (2018). In their algorithm, the underlying asset price is transformed to decorrelate the two Brownian motions in the model. This transformed process is then approximated by a double-layer continuous-time Markov chain that is embedded into a single-layer CTMC on a long grid where each point is a pair of the transformed price and volatility. This construction creates two issues. The first one is that the size of the generator matrix of the resulting CTMC can become too large for efficient computations. The other one is that it may be difficult to ensure that all the price barriers in the autocallable lie in appropriate locations on the asset price grid. To

overcome these difficulties, we approximate the variance process using a discrete-time Markov chain (DTMC) and properly adjust the grid design for the transformed asset price. Compared with the original algorithm in Cui et al. (2018), our method achieves stable convergence and substantial reduction in computation time for payoffs with discontinuities.

Third, we introduce a systematic approach to improve dynamic hedging of autocallables in the presence of transaction costs. Kim and Lim (2019) develop an interesting static hedging strategy for two types of autocallables by trading vanilla options with different maturities and strikes. However, static hedging may be difficult to implement in practice due to the illiquidity of some instruments. Furthermore, static replication may no longer be possible if the structured product has more exotic features. We study dynamic hedging in this paper and show that the delta of an autocallable product can vary rapidly near the barriers, making delta hedging costly to implement. To address this issue, we implement barrier shifting and payoff modification methods as suggested by Chan et al. (2019) to adjust delta. An important question is how to determine the size of these adjustments, which has not been studied before. To answer it, we minimize the conditional value-at-risk (CVaR) of the hedging loss at maturity with the adjusted quantities as decision variables. We solve the optimization problem using stochastic gradient descent and show that with proper adjustments we can achieve significant reduction in the CVaR.

Remark 1 Discontinuities in the autocallable payoffs can render CTMC approximation to converge slowly. To address this issue, we show how to design the CTMC grid properly in this paper. An alternative solution is applying the payoff smoothing technique, where the discontinuous payoff is first approximated by a smooth function and the smoothed payoff is then used in the CTMC approximation algorithm; see e.g., Li and Zhang (2018); Yang et al. (2019); Zhang and Li (2022), and Bayer et al. (2023) for discussions of this type of technique. Both proper grid design and payoff smoothing can help CTMC approximation achieve second-order convergence with discontinuous payoffs. However, while the former can remove convergence oscillations to make Richardson extrapolation applicable, the latter may not as shown in Li and Zhang (2018).

The rest of this paper is organized as follows. In Sect. 2, we introduce the autocallable structures considered in this study and derive recursive pricing formulas for them. In Sect. 3, we review the construction of CTMC approximation, discuss the grid design issue, and present pricing algorithms for various types of models: 1D time-homogeneous Markov models, regime-switching models, 1D time-inhomogeneous Markov models, and SLV models. In Sect. 4, we demonstrate the performance of the pricing algorithm for various popular models. In Sect. 5, we discuss the issues in delta hedging and show how to solve the dynamic hedging problem. Finally, we conclude in Sect. 6. Appendix A contains pseudocodes of the algorithms presented in this paper. All the experiments in the paper were conducted in Matlab on a workstation with Intel Xeon CPU E5-2687W and 64GB of memory.

2 Autocallable structures and pricing recursions

Autocallables are over-the-counter products, which are flexible in design and can have various structures. In this section, we introduce two common autocallable structures. We also show how to price them recursively to deal with the path dependency in their payoffs.

The autocallable we consider is written on one risky asset whose price at time t is denoted by S_t. The product has a notional value N (which is the investor's initial capital), matures at time T, and is observed on dates $t_1, t_2, \ldots, t_n = T$ after time 0 (we set $t_0 = 0$). Some autocallables allow the holder to recover lost coupons from earlier observation dates when the asset price is in a certain range, which is known as the snowball effect.

2.1 Structure I

Structure I is considered in Guillaume (2015b). At each observation date t_i ($1 \le i \le n$), there are three barriers $C_i < D_i < U_i$, and two additional barriers $B < C_n$ and $H < D_n$ at t_n. The payoff from the autocallable at t_i depends on where S_{t_i} lies. Below we describe the payoff at t_i for $1 \le i < n$.

- If $S_{t_i} < C_i$, no regular coupon is paid to the investor and the autocallable continues to the next observation date.
- If $C_i \le S_{t_i} < D_i$, a regular coupon $N \times z_i$ is paid to the investor, all previously lost coupons are recovered if the snowball effect is applied, and the autocallable continues to the next observation date.
- If $D_i \le S_{t_i} < U_i$, the autocallable terminates, a final prespecified coupon $N \times y_i$ is paid to the investor, all previously lost coupons are recovered if the snowball effect is applied, and the investor's initial capital is fully redeemed.
- If $S_{t_i} \ge U_i$, the autocallable terminates, $N \times S_{t_i}/S_0$ is paid to the investor, and all previously lost coupons are recovered if the snowball effect is applied. Note that $U_i > S_0$.

The payoff at t_n is given below.

- If $S_{t_n} < B$, only S_{t_n}/S_0 of the investor's initial capital is redeemed and no other payments is received.
- If $B \le S_{t_n} < C_n$, the investor's initial capital is fully redeemed and no other payments are received.
- If $C_n \le S_{t_n} < D_n$ or if $D_n \le S_{t_n} < U_n$ and $\inf_{t \in (t_{n-1}, t_n]} S_t \le H$, the investor's initial capital is fully redeemed and all previously lost coupons are recovered if the snowball effect is applied.

- If $D_n \le S_{t_n} < U_n$ and $\inf_{t\in(t_{n-1},t_n]} S_t > H$, a final prespecified coupon $N \times y_n$ is paid to the investor, all previously lost coupons are recovered if the snowball effect is applied, and the investor's initial capital is fully redeemed.
- If $S_{t_n} \ge U_n$, $N \times S_{t_n}/S_0$ is paid to the investor, and all previously lost coupons are recovered if the snowball effect is applied. Note that $U_n > S_0$.

Pricing an autocallable is complex because its payoff is highly path dependent. Backward induction is the standard approach to deal with path-dependent products; see e.g., Li and Linetsky (2015) for discretely monitored barrier options. We follow this approach and present the pricing recursion for autocallables. Here, we assume the asset price process $(S_t)_{t\ge 0}$ is a time-homogeneous Markov process and for notational simplicity, the observation dates are equally distanced with time step h. We also assume that the risk-free rate is a constant and denote it by r. We need to consider $m_t := \inf_{u\in[0,t]} S_u$, which is the minimum asset price from time 0 to t. The price of the autocallable at time 0 is a function of the initial asset price variable x, which we denote by $V(x)$. All pricing is done under a chosen risk-neutral measure.

Pricing recursion for structure I without snowball effect. When there is no snowball effect, any lost coupon cannot be recovered later. Let $V^i(x)$ denote the price of the autocallable at time $t_i = ih$ given $S_{t_i} = x$ for $i = 1, ..., n-1$. The value function at t_n equals the payoff, which depends on both S_T and the minimum price of the underlying asset between t_{n-1} and t_n. Thus, we write this value function as $V^n(x, \underline{x})$, where $S_T = x$ and the minimum price between t_{n-1} and t_n equals $\underline{x}$. By repeatedly conditioning and using the Markov property, we obtain the following backward recursion starting from time t_n:

$$\begin{aligned}
V^n(x,\underline{x}) = {} & \mathbf{1}_{[B,U_n)}(x)N + \mathbf{1}_{[D_n,U_n)}(x)\mathbf{1}_{(H,\infty)}(\underline{x})Ny_n \\
& + \mathbf{1}_{[U_n,\infty)}(x)Nx/S_0 + \mathbf{1}_{[0,B)}(x)Nx/S_0, \\
V^{n-1}(x) = {} & \mathbf{1}_{[0,D_{n-1})}(x)e^{-rh}\mathbf{E}_x\left[V^n\left(S_h, m_h\right)\right] + \mathbf{1}_{[C_{n-1},D_{n-1})}(x)Nz_{n-1} \\
& + \mathbf{1}_{[D_{n-1},U_{n-1})}(x)N(1+y_{n-1}) \\
& + \mathbf{1}_{[U_{n-1},\infty)}(x)Nx/S_0, \\
V^i(x) = {} & \mathbf{1}_{[0,D_i)}(x)e^{-rh}\mathbf{E}_x\left[V^{i+1}(S_h)\right] + \mathbf{1}_{[C_i,D_i)}(x)Nz_i \\
& + \mathbf{1}_{[D_i,U_i)}(x)N(1+y_i) \\
& + \mathbf{1}_{[U_i,\infty)}(x)Nx/S_0,\ i = n-2, ..., 1, \\
V(x) = {} & e^{-rh}\mathbf{E}_x\left[V^1(S_h)\right],
\end{aligned}$$

where $\mathbf{E}_x[f(S_h)] = \mathbf{E}[f(S_h)|S_0 = x]$. Calculating $V(x)$ requires evaluating

$$\mathbf{E}_x\left[V^n\left(S_h, m_h\right)\right],\ \mathbf{E}_x\left[V^{i+1}(S_h)\right] \text{ for } i = 0, ..., n-2.$$

The first expectation can be expressed as

$$\mathbf{E}_x\left[V^n\left(S_h, m_h\right)\right] = \mathbf{E}_x\left[\phi(S_h)\right] + \mathbf{E}_x\left[\upsilon(S_h)\mathbf{1}_{\{\tau_H > h\}}\right],$$

where $\tau_H := \inf\{t \ge 0 : S_t \le H\}$, and

$$\begin{aligned} \upsilon(x) =& \mathbf{1}_{[D_n,U_n)}(x)Ny_n,\ \phi(x) = \mathbf{1}_{[B,U_n)}(x)N \\ &+ \mathbf{1}_{[U_n,\infty)}(x)Nx/S_0 + \mathbf{1}_{[0,B)}(x)Nx/S_0. \end{aligned} \tag{1}$$

Pricing recursion for structure I with snowball effect. The snowball effect brings more challenge to the pricing problem. To deal with it, we introduce an additional state variable that tracks the number of missed coupons. Let Q_i be the number of missed coupons by t_i for $i = 0, 1, ..., n-1$. We have

$$Q_0 = 0,\ \ Q_i = \mathbf{1}_{\{S_{t_i} < C_i\}}(Q_{i-1} + 1), \quad i = 1, ..., n-1.$$

At t_i, if S_{t_i} is below C_i, the coupon at t_i is missed and hence increasing the number of missed coupons by one. However, if S_{t_i} is above C_i, all the previously missed coupons are recovered, thus making the number of missed coupons zero. We have Q_i taking value from $\{0, 1..., i\}$.

Let $V^i(x, q)$ denote the value of the autocallble product at time $t_i = ih$ for $i = 1, 2, ..., n-1$ given $S_{t_i} = x$ and $Q_{i-1} = q$. For the value at t_n, we write it as $V^n(x, \underline{x}, q)$ because it also depends on the minimum price of the underlying asset between t_{n-1} and t_n. The backward recursion is given by

$$\begin{aligned} V^n(x, \underline{x}, q) =\ & \mathbf{1}_{[B,U_n)}(x)N + \mathbf{1}_{[D_n,U_n)}(x)\mathbf{1}_{(H,\infty)}(\underline{x})Ny_n + \mathbf{1}_{[U_n,\infty)}(x)Nx/S_0 \\ &+ \mathbf{1}_{[C_n,\infty)}(x)N \sum_{m=n-q}^{n-1} z_m + \mathbf{1}_{[0,B)}(x)Nx/S_0, \\ V^{n-1}(x, q) =\ & \mathbf{1}_{[0,C_{n-1})}(x)e^{-rh}\mathbf{E}_x\left[V^n\left(S_h, m_h, q+1\right)\right] \\ &+ \mathbf{1}_{[C_{n-1},D_{n-1})}(x)\left(Nz_{n-1} + e^{-rh}\mathbf{E}_x\left[V^n\left(S_h, m_h, 0\right)\right]\right) \\ &+ \mathbf{1}_{[D_{n-1},U_{n-1})}(x)N(1 + y_{n-1}) + \mathbf{1}_{[U_{n-1},\infty)}(x)Nx/S_0 \\ &+ \mathbf{1}_{[C_{n-1},\infty)}(x)N \sum_{m=n-1-q}^{n-2} z_m, \\ V^i(x, q) =\ & \mathbf{1}_{[0,C_i)}(x)e^{-rh}\mathbf{E}_x\left[V^{i+1}(S_h, q+1)\right] \\ &+ \mathbf{1}_{[C_i,D_i)}(x)\left(Nz_i + e^{-rh}\mathbf{E}_x\left[V^{i+1}(S_h, 0)\right]\right) \\ &+ \mathbf{1}_{[D_i,U_i)}(x)N(1 + y_i) + \mathbf{1}_{[U_i,\infty)}(x)Nx/S_0 \\ &+ \mathbf{1}_{[C_i,\infty)}(x)N \sum_{m=i-q}^{i-1} z_m,\ i = n-2, ..., 1, \\ V(x) =\ & e^{-rh}\mathbf{E}_x\left[V^1(S_h, 0)\right]. \end{aligned}$$

To obtain $V(x)$, we need to calculate

$$\begin{aligned} &\left\{\mathbf{E}_x\left[V^n\left(S_h, m_h, q\right)\right],\ q = 0, 1, ..., n-1\right\}, \\ &\left\{\mathbf{E}_x\left[V^{i+1}(S_h, q)\right] :\ q = 0, 1, ..., i\right\} \text{ for } i = 0, ..., n-2. \end{aligned}$$

For $q = 0, 1, ..., n-1$, set

$$\phi_q(x) = \mathbf{1}_{[B,U_n)}(x)N + \mathbf{1}_{[U_n,\infty)}(x)Nx/S_0 + \mathbf{1}_{[C_n,\infty)}(x)N \sum_{m=n-q}^{n-1} z_m + \mathbf{1}_{[0,B)}(x)Nx/S_0.$$

Then, we have

$$\mathbf{E}_x\left[V^n\left(S_h, m_h, q\right)\right] = \mathbf{E}_x\left[\phi_q(S_h)\right] + \mathbf{E}_x\left[v(S_h)\mathbf{1}_{\{\tau_H > h\}}\right].$$

2.2 Structure II

We consider another autocallable structure following Kim and Lim (2019), which applies to two popular autocallable products: autocallable barrier reverse convertible notes (ABRCN) and stepdown knock-in equity-linked securities (ELS). The former is sold and managed by securities from Europe and US and the latter stands for the autocallable note in the ELS market of South Korea.

At each t_i for $i = 1, \cdots, n-1$, there is only one barrier D_i. A regular coupon $N \times z_i$ is always paid if the product has not terminated. If $S_{t_i} \geq D_i$, the product terminates and returns the initial capital N plus a prespecified coupon $N \times y_i$ to the investor. Thus, we can write the payoff at t_i as

$$\prod_{j=1}^{i-1} \mathbf{1}_{\left\{S_{t_j} < D_j\right\}} N\left(\left(1 + y_m\right)\mathbf{1}_{\left\{S_{t_i} \geq D_i\right\}} + z_m\right)$$

where the product $\prod_{j=1}^{0} \cdot = 1$ by convention.

At the maturity time t_n, in addition to D_n, there is another barrier B, which is monitored during $[0, T]$. Typically, we have $B < D_i$ for $i = 1, \cdots, n$. Let $\tau_B = \inf\{t \geq 0 : S_t \leq B\}$. The payoff at t_n is given by

$$\prod_{j=1}^{n-1} \mathbf{1}_{\{S_j < D_j\}} N\left\{\left(\left(1 + y_n\right)\mathbf{1}_{\{S_T \geq D_n\}} + \frac{S_T}{S_0}\mathbf{1}_{\{S_T < D_n\}}\right)\mathbf{1}_{\{\tau_B \leq t_n\}} + \left(1 + y_n\right)\mathbf{1}_{\{\tau_B > t_n\}} + z_n\right\}$$

If the asset price has hit or fallen below B during $[0, T]$ and $S_T < D_n$, it is possible that the investor cannot get back the full amount of the initial capital.

Pricing recursion for structure II. We introduce a state variable to keep track of whether B has been breached. The value function at time t_i for $i = 1, \cdots, n$ is written as $V^i(x, I_B)$, where $I_B = 1$ indicates that the asset price remains above B by time t_i. We obtain the following backward recursion:

$$
\begin{aligned}
V^n(x, I_B) = &\ N\left\{(1-I_B)\left((1+y_n)\mathbf{1}_{[D_n,\infty)} + \frac{x}{S_0}\mathbf{1}_{[0,D_n)}\right) + I_B(1+y_n) + z_n\right\},\\
V^i(x, I_B) = &\ Nz_m + \mathbf{1}_{[0,D_i)}e^{-rh}\mathbf{E}_x\left[V^{i+1}(S_h, I_B\mathbf{1}_{\{\tau_B>h\}})\right]\\
&+ \mathbf{1}_{[D_i,\infty)}N(1+y_m),\ i = n-1, ..., 1,\\
V(x) = &\ e^{-rh}\mathbf{E}_x\left[V^1(S_h, \mathbf{1}_{\{\tau_B>h\}})\right].
\end{aligned}
$$

To obtain $V(x)$, we need to evaluate

$$
\mathbf{E}_x\left[V^{i+1}(S_h, I_B\mathbf{1}_{\{\tau_B>h\}})\right] \text{ for } I_B = 0 \text{ or } 1,\ i = 0, ..., n-1.
$$

Let

$$
\varphi(x) = \mathbf{1}_{[0,D_n)}(x)N(1+y_n - x/S_0),\ \psi(x) = N(1+y_n)\mathbf{1}_{[D_n,\infty)} + Nx/S_0\mathbf{1}_{[0,D_n)} + Nz_n.
$$

We have

$$
\begin{aligned}
\mathbf{E}_x\left[V^n\left(S_h, I_B\mathbf{1}_{\{\tau_B>h\}}\right)\right] &= \mathbf{E}_x\left[N(1+y_n)\mathbf{1}_{[D_n,\infty)} + N\frac{S_h}{S_0}\mathbf{1}_{[0,D_n)}(S_h) + Nz_n\right]\\
&\quad + I_B\mathbf{E}_x\left[\mathbf{1}_{\{\tau_B>h\}}N\left(1+y_n - \frac{S_h}{S_0}\right)\mathbf{1}_{[0,D_n)}(S_h)\right]\\
&= \mathbf{E}_x\left[\psi(S_h)\right] + I_B\mathbf{E}_x\left[\varphi(S_h)\mathbf{1}_{\{\tau_B>h\}}\right],
\end{aligned}
$$

and for $i = n-2, \cdots, 1$,

$$
\begin{aligned}
\mathbf{E}_x\left[V^{i+1}(S_h, I_B\mathbf{1}_{\{\tau_B>h\}})\right] =&\mathbf{E}_x\left[V^{i+1}(S_h, 0)\right]\\
&+ I_B\mathbf{E}_x\left[\left(V^{i+1}(S_h, 1) - V^{i+1}(S_h, 0)\right)\mathbf{1}_{\{\tau_B>h\}}\right].
\end{aligned}
$$

3 Markov chain approximation for pricing autocallables

3.1 Basics of CTMC approximation

We review some basic results for using CTMC approximation to calculate expectations for 1D time-homogeneous Markov processes with continuous state spaces. The asset price S_t is Markov and takes value from $[0, \infty)$ and its infinitesimal generator is given by

$$
\begin{aligned}
\mathcal{G}f(x) =&\mu(x)f'(x) + \frac{1}{2}\sigma^2(x)f''(x)\\
&+ \int_{-x}^{\infty}\left(f(x+y) - f(x) - yf'(x)1_{\{|y|\le 1\}}\right)\mathcal{J}(x, dy),
\end{aligned}
$$

for $f \in C_c^2(\mathbb{R}^+)$, where $\mu(x)$, $\sigma(x)$, and $\nu(x, dy)$ are the drift, volatility, and jump intensity measure of X, respectively. In general, $\mathcal{J}$ is state-dependent and satisfies $\int_{|y|\leq 1} y^2 \mathcal{J}(x, dy) < \infty$ for all $x \in \mathbb{R}^+$.

For the asset price, we assume that ∞ is inaccessible and 0 is either inaccessible or absorbing if accessible. We approximate S_t by a CTMC X_t with a finite state space $\mathbb{G} = \{x_1, ..., x_M\}$ and generator matrix $\mathcal{G} \in \mathbb{R}^{M\times M}$. The details of how to construct $\mathcal{G}$ can be found in Mijatović and Pistorius (2013) and Zhang and Li (2021). We impose absorbing behavior for the boundary states x_1 and x_M of the CTMC, i.e., the CTMC cannot move to other states once it arrives at x_1 or x_M, which would lead to convergence.

We need to compute two forms of expectations for S_t

$$\mathbf{E}_x\left[f(S_t)\right],\ \mathbf{E}_x\left[f(S_t)1_{\{\tau_B>h\}}\right],$$

where τ_B is the first time the process hits or falls below a generic barrier B. These expectations are approximated by the corresponding expectations for the CTMC X_t, which are

$$\mathbf{E}_x\left[f(X_t)\right],\ \mathbf{E}_x\left[f(X_t)1_{\{\tau_B^X>h\}}\right]$$

Expressions for these expectations are derived in Mijatović and Pistorius (2013) and they are given by

$$\mathbf{E}_x\left[f\left(X_t\right)\right] = (\exp(\mathcal{G}t)F)(x) \quad \text{for } x \in \mathbb{G}, \tag{2}$$

$$\mathbf{E}_x\left[f(X_t)1_{\{\tau_B^X>h\}}\right] = (\exp(\widehat{\mathcal{G}}t)\widehat{F})(x) \quad \text{for } x \in \mathbb{G} \cap (B, \infty), \tag{3}$$

where the vector F contains the values of the payoff function f evaluated at the points in $\mathbb{G}$, $\widehat{F}$ is the restriction of F to those grid points greater than B, and

$$\widehat{\mathcal{G}}(x, y) := \mathcal{G}(x, y) \quad x, y \in \mathbb{G} \cap (B, \infty)$$

and the size of matrix $\widehat{\mathcal{G}}$ is $\widehat{M} \times \widehat{M}$, where $\widehat{M}$ is the number of grid points greater than B. There exist various algorithms to calculate the matrix exponential $\exp(A)$ for a square matrix A. A popular choice is the scaling and squaring algorithm (Higham, 2005) and other computationally efficient choices for matrices with certain structures can be found in Li and Zhang (2016) and Meier et al. (2021).

Now, to price autocallables, the expectations in the pricing recursions can be approximated using formulas (2) and (3).

3.2 Grid design

The convergence behavior of CTMC approximation is strongly affected by the design of the grid for the CTMC. Financial payoffs typically lack smoothness, e.g., the payoff or its derivative is discontinuous, which can make convergence

slow (see Li and Zhang (2018)). Zhang and Li (2019) studied the grid design problem for pricing continuously monitored barrier options by CTMC approximation for diffusion models. They derived two conditions that are sufficient and necessary to ensure second-order convergence for diffusion models without noticeable oscillations so that Richardson extrapolation can be applied to obtain even faster convergence. First, there must be a grid point at each barrier. Second, the strike of the payoff must be placed exactly midway between two adjacent grid points. Although these results were developed from theoretical analysis under diffusion models, they can still be applied to achieve fast convergence in commonly used Markov models with jumps in finance as shown in Zhang and Li (2019).

The payoffs of autocallables are discontinuous at the barriers, and thus the CTMC grid must be designed with care to obtain nice convergence behavior. We apply the results in Zhang and Li (2019). In our problem, the discretely monitored barrier is the "strike" in Zhang and Li (2019) and the continuously monitored barrier is the "barrier" in Zhang and Li (2019). Take structure I as an example. There are many "strikes" $\{B, C_i, D_i, U_i : i = 1, ..., n\}$ and one "barrier" H. Let $\mathbb{K} = \{\xi_1, ..., \xi_d\} = \{B, C_i, D_i, U_i : i = 1, ..., n\}$ be the set of all non-repeated "strikes" and $\xi_1 < \cdots < \xi_d$. We assume $H \notin \mathbb{K}$ and $H \in (\xi_l, \xi_{l+1})$ for some l. To achieve second-order convergence for calculating all the expectations in the pricing recursion, we propose a grid to satisfy the two requirements in Zhang and Li (2019).

We use a piecewise uniform structure for the grid $\mathbb{G}$. Let e_1 and e_2 be the smallest and largest states, respectively. We construct $\mathbb{G}$ as

$$\mathbb{G} = \bigcup_{i=0}^{d+1} \mathbb{G}_i \tag{4}$$

where

$$\begin{aligned}
\mathbb{G}_0 &= \{e_1\} \cup \{e_1 + (0.5 + j)h_0 : 0 \le j < n_0\}, \ h_0 = (\xi_1 - e_1)/n_0, \\
\mathbb{G}_i &= \{\xi_i + 0.5h_{i-1} + (0.5 + j)h_i : 0 \le j < n_i\}, \\
h_i &= (\xi_{i+1} - \xi_i - 0.5h_{i-1})/n_i, \ 1 \le i < l, \\
\mathbb{G}_l &= \{\xi_l + 0.5h_{l-1} + jh_l : 0 \le j < n_l\}, \ h_l = (H - \xi_l - 0.5h_{l-1})/n_l, \\
\mathbb{G}_{l+1} &= \{H\} \cup \{H + (0.5 + j)h_{l+1} : 0 \le j < n_{l+1}\}, \ h_{l+1} = (\xi_{l+1} - H)/n_{l+1}, \\
\mathbb{G}_i &= \{\xi_{i-1} + 0.5h_{i-1} + (0.5 + j)h_i : 0 \le j < n_i\}, \\
h_i &= (\xi_i - \xi_{i-1} - 0.5h_{i-1})/n_i, \ l + 2 \le i < d + 1, \\
\mathbb{G}_{d+1} &= \{\xi_d + 0.5h_d + jh_{d+1} : 0 \le j \le n_{d+1}\}, \ h_{d+1} = (e_2 - \xi_d - 0.5h_d)/n_{d+1}.
\end{aligned}$$

The integer n_i specifies the number of sub-intervals in $\mathbb{G}_i$ for $i = 0, 1, \cdots, d + 1$. All the points in $\mathbb{K}$ are exactly in the middle of two adjacent grid points and H is on the grid. We observe that when the snowball effect is present, the value of q does not affect the positions of the "strikes" and "barrier". Therefore, for all the expectations in the pricing recursions for structure A, the CTMC approximation converges in second order. Furthermore, convergence is smooth, enabling us to apply Richardson extrapolation to achieve higher-order convergence.

For structure B, the grid is designed in the same way with $\{D_i, i = 1, ..., n\}$ as the "strikes" and B as the "barrier".

Remark 2 For structure A, if $H \in \mathbb{K}$, it is impossible to fulfill the two requirements simultaneously. In this case, we first design a grid $\mathbb{G}_0$ that places H on the grid so that the approximation for $\mathbf{E}_x[\upsilon(X_h)\mathbf{1}_{\{\tau_H > h\}}]$ achieves second-order smooth convergence. We then design another grid $\mathbb{G}_1$ that places H midway between two adjacent grid points, thus guaranteeing second-order smooth convergence for all other expectations. To implement the recursion, we need to obtain the values of $\mathbf{E}_x[\upsilon(X_h)\mathbf{1}_{\{\tau_H > h\}}]$ for $x \in \mathbb{G}_1$, which can be achieved by interpolating its values on $\mathbb{G}_0$. With an appropriate interpolation scheme, the error of interpolation is of a higher convergence order than CTMC approximation error (see Zhang and Li (2019)). Thus, the overall approximation error would remain second order. For structure B, if B equals some "strike", we can do the same.

Remark 3 If we have multiple autocallables with different barriers, using the piecewise uniform structure we can design a grid that places the barriers of all the products in appropriate locations to satisfy the requirements in Zhang and Li (2019). Using this grid, all the products share the same matrix exponential in the CTMC algorithm.

3.3 Pricing under 1D time-homogeneous Markov models

We present the algorithms for computing $V(x)$ in three cases when S_t is a 1D time-homogeneous Markov process. We have approximated S_t by a CTMC X_t with state space $\mathbb{G}$ and generator matrix $\mathcal{G}$.

Algorithm for structure I without snowball effect. For any $x \in \mathbb{G}$, let

$$\begin{aligned} h(i, x) &:= \mathbf{E}_x[V^i(X_h)], \quad i = 1, ..., n-1, \\ h(n, x) &:= \mathbf{E}_x[V^n(X_h, m_h^X)]. \end{aligned}$$

Our algorithm calculates $h(i, x)$ recursively in two steps.

Step 1: Let m_H denote the index such that $x_{m_H - 1} = H$. Set $\hat{\mathcal{G}} = \mathcal{G}_{m_H:M, m_H:M}$. Calculate matrix exponentials $A = \exp(\mathcal{G}h)$ and $\hat{A} = \exp(\hat{\mathcal{G}}h)$. Recall that the payoff of autocallable I at t_n is the sum of ϕ and υ defined in (1), where ϕ is the payoff without the need for monitoring H and υ gives the payoff paid to the investor only when H has not been down-crossed during $[t_{n-1}, t_n]$. Compute

$$\begin{aligned} \mathbf{E}_{x_m}[\upsilon(X_h)\mathbf{1}_{\{\tau_H^X > h\}}] &= (\hat{A}\upsilon)(x_m), \quad \text{for } m = m_H, ..., M, \\ \mathbf{E}_{x_m}[\phi(X_h)] &= (A\phi)(x_m), \quad \text{for } m = 1, ..., M. \end{aligned}$$

Then compute

$$h(n, x_m) = (\hat{A}\upsilon)(x_m)1_{\{m \geq m_H\}} + (A\phi)(x_m), \quad \text{for } m = 1, ..., M.$$

This step costs $O(M^2)$ plus the cost of computing two matrix exponentials.

Step 2: Do the following for i running backward from $n-1$ to 1: first compute

$$\begin{aligned} V^i(x_m) = Nz_i 1_{\{C_i \leq x_m < D_i\}} + N(1+y_i)1_{\{D_i \leq x_m < U_i\}} + Nx_m/S_0 1_{\{x_m \geq U_i\}} \\ + e^{-rh} h(i+1, x_m) 1_{\{x_m < D_i\}}, \quad \text{for } m = 1, ..., M, \end{aligned}$$

where $h(i+1, x_m)$ has been calculated from the previous iteration and V^i is an M-dimensional vector whose mth component is $V^i(x_m)$. Then compute

$$h(i, x_m) = (AV^i)(x_m), \quad \text{for } m = 1, ..., M.$$

Finally, if the initial asset price x is a grid point, set $V(x) = e^{-rh}h(1, x)$. Otherwise, $V(x)$ can be estimated by interpolating the values on the grid. The cost of calculating $\{h(i, x_m) : m = 1, ..., M\}$ for all i is $O(nM^2)$.

Combining the two steps, the total cost is $O(nM^2)$ plus the cost of computing two matrix exponentials.

Algorithm for structure I with snowball effect. For any $x \in \mathbb{G}$, let

$$\begin{aligned} h(i, x, q) &:= \mathbf{E}_x[V^i(X_h, q)], \quad q = 0, 1, ..., i-1, \ i = 1, ..., n-1, \\ h(n, x, q) &:= \mathbf{E}_x[V^n(X_h, m_h^X, q)], \quad q = 0, 1, ..., n-1. \end{aligned}$$

Our algorithm calculates $h(i, x, q)$ recursively in two steps.

Step 1: Let m_H denote the index such that $x_{m_H - 1} = H$. Set $\hat{\mathcal{G}} = \mathcal{G}_{m_H:M, m_H:M}$. Calculate matrix exponentials $A = \exp(\mathcal{G}h)$ and $\hat{A} = \exp(\hat{\mathcal{G}}h)$. Compute

$$\begin{aligned} \mathbf{E}_{x_m}[\upsilon(X_h)\mathbf{1}_{\{\tau_H^X > h\}}] &= (\hat{A}\upsilon)(x_m), \quad \text{for } m = m_H, ..., M, \\ \mathbf{E}_{x_m}[\phi_q(X_h)] &= (A\phi_q)(x_m), \quad \text{for } m = 1, ..., M \text{ and } q = 0, 1, ..., n-1. \end{aligned}$$

Then compute

$$\begin{aligned} h(n, x_m, q) =& (\hat{A}\upsilon)(x_m)1_{\{m \geq m_H\}} + (A\phi_q)(x_m), \\ & \text{for } m = 1, ..., M \text{ and } q = 0, 1, ..., n-1. \end{aligned}$$

This step costs $O(nM^2)$ plus the cost of computing two matrix exponentials.

Step 2: Do the following for i running backward from $n-1$ to 1: first compute

$$V_q^i(x_m) = Nz_i 1_{\{C_i \le x_m < D_i\}} + N(1+y_i)1_{\{D_i \le x_m < U_i\}} + Nx_m/S_0 1_{\{x_m \ge U_i\}} + e^{-rh} h(i+1, x_m, 0) 1_{\{C_i \le x_m < D_i\}} + e^{-rh} h(i+1, x_m, q+1) 1_{\{x_m < C_i\}} + N \sum_{k=i-q}^{i-1} z_k 1_{\{x_m \ge C_i\}}, \quad m = 1, ..., M; q = 0, 1, ..., i-1,$$

where $h(i+1, x_m, q+1)$ and $h(i+1, x_m, 0)$ have been calculated from previous iteration and V_q^i is a M-dimensional vector whose mth component is $V_q^i(x_m)$. Then compute

$$h(i, x_m, q) = (AV_q^i)(x_m), \quad \text{for } m = 1, ..., M \text{ and } q = 0, 1, ..., i-1.$$

Finally, if the initial asset price x is a grid point, $V(x) = e^{-rh} h(1, x, 0)$. Otherwise, $V(x)$ can be estimated by interpolating the values on the grid. The cost of calculating $\{h(i, x_m, q) : m = 1, ..., M, q = 0, 1, ..., i-1\}$ for all i is $O(n^2 M^2)$.

Combining the two steps, the total cost is $O(n^2 M^2)$ plus the cost of computing two matrix exponentials.

Algorithm for structure II. For any $x \in \mathbb{G}$, let

$$h(i, x, I_B) := \mathbf{E}_x[V^{i+1}(X_h, I_B \mathbf{1}_{\{\tau_B^X > h\}})], \quad i = 0, ..., n-1, \ \ I_B = 0, 1.$$

Our algorithm calculates $h(i, x, I_B)$ recursively in two steps.

Step 1: Let m_B denote the index such that $x_{m_B - 1} = B$. Set $\bar{\mathcal{G}} = \mathcal{G}_{m_B:M, m_B:M}$. Calculate matrix exponentials $A = \exp(\mathcal{G}h)$ and $\bar{A} = \exp(\bar{\mathcal{G}}h)$. Compute

$$\mathbf{E}_{x_m}[\varphi(X_h)\mathbf{1}_{\{\tau_B^X > h\}}] = (\bar{A}\varphi)(x_m), \quad \text{for } m = m_B, ..., M,$$
$$\mathbf{E}_{x_m}[\psi(X_h)] = (A\psi)(x_m), \quad \text{for } m = 1, ..., M.$$

Then compute

$$h(n-1, x_m, I_B) = (A\psi)(x_m) + I_B(\bar{A}\varphi)(x_m) 1_{\{m \ge m_B\}}, \quad \text{for } m = 1, ..., M \text{ and } I_B = 0, 1.$$

This step costs $O(M^2)$ plus the cost of computing two matrix exponentials.

Step 2: Do the following for i running backward from $n-2$ to 0: first compute

$$V^{i+1}(x_m, I_B) = Nz_i + N(1+y_i)1_{\{x_m \ge D_i\}} + e^{-rh} h(i+1, x_m, I_B) 1_{\{x_m < D_i\}}, \quad m = 1, ..., M, \ I_B = 0, 1,$$

where $\{h(i+1, x_m, I_B) : I_B = 0, 1\}$ has been calculated from previous iteration and $V^{i+1}(\cdot, I_B)$ is an M-dimensional vector whose mth component is $V^{i+1}(x_m, I_B)$. Then compute

$$h(i, x_m, I_B) = (AV^{i+1}(\cdot, 0))(x_m) + I_B\left(\bar{A}\left(V^{i+1}(\cdot, 1) - V^{i+1}(\cdot, 0)\right)\right)(x_m),$$
$$m = 1, ..., M,\ I_B = 0, 1.$$

Finally, if the initial asset price x is a grid point, set $V(x) = e^{-rh}h(0, x; 1)$. Otherwise, $V(x)$ can be estimated by interpolating the values on the grid. The cost of calculating $\{h(i, x_m; I_B) : m = 1, ..., M, I_B = 0, 1\}$ for all i is $O(nM^2)$.

Combining the two steps, the total cost is $O(nM^2)$ plus the cost of computing two matrix exponentials.

We provide the pseudocodes of these pricing algorithms in the appendix. As the algorithm is largely similar for different types of autocallables, we only present the algorithm for autocallable structure A without snowball effect for the other classes of models in the following. Changes to the algorithm can be easily made for the other two autocallables.

3.4 Pricing under regime-switching Markov models

Consider a regime-switching model where the drift, volatility, and jump intensity measure of S_t depends on the regime v_t, i.e., they can be written as $\mu(x, v)$, $\sigma(x, v)$, and $\mathcal{J}(x, v, dy)$. We assume that v_t follows a CTMC with state space $\mathbb{G}_v = \{v_1, ..., v_L\}$ and transition rate matrix $\Lambda \in \mathbb{R}^{L\times L}$, and it is independent of the random sources driving S_t. For each regime v_j ($j = 1, \cdots, L$), we construct a CTMC with state space $\mathbb{G} = \{x_1, ..., x_M\}$ and generator matrix $\mathcal{G}_{v_j} \in \mathbb{R}^{M\times M}$ to approximate S_t if $v_t = v_j$. Thus, the pair (S_t, v_t) is approximated by the regime-switching CTMC (X_t, v_t) where X moves on $\mathbb{G}$ with transition rate matrix $\mathcal{G}_v$ when $v_t = v$ for $v \in \mathbb{G}_v$ and v_t evolves according to transition rate matrix Λ.

Cai et al. (2019) showed that the bivariate regime-switching CTMC (X_t, v_t) is converted into a univariate CTMC Y_t with state space $\mathbb{G}_Y \in \mathbb{R}^{ML}$ given by

$$\mathbb{G}_Y = \{(x_1, v_1), ..., (x_M, v_1), (x_1, v_2), ..., (x_M, v_2), ..., (x_1, v_L), ..., (x_M, v_L)\},$$

and transition rate matrix

$$\mathcal{G}_Y = \begin{pmatrix} \Lambda_{11}\mathbf{I}_M + \mathcal{G}_{v_1} & \Lambda_{12}\mathbf{I}_M & \cdots & \Lambda_{1L}\mathbf{I}_M \\ \Lambda_{21}\mathbf{I}_M & \Lambda_{22}\mathbf{I}_M + \mathcal{G}_{v_2} & \cdots & \Lambda_{2L}\mathbf{I}_M \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{L1}\mathbf{I}_M & \Lambda_{L2}\mathbf{I}_M & \cdots & \Lambda_{LL}\mathbf{I}_M + \mathcal{G}_{v_L} \end{pmatrix},$$

where $\mathbf{I}_M$ is the $M \times M$ identity matrix. As the payoffs of an autocallable can also be viewed as functions of Y_t, its price can be approximated by the algorithms in Sect. 3.3.

3.5 Pricing under 1D time-inhomogeneous Markov models

In a time-inhomogenous Markov model, the drift, volatility, and jump intensity measure are also functions of time and they are written as $\mu(t, x)$, $\sigma(t, x)$, and $\mathcal{J}(t, x, dy)$.

We divide the time interval $[0, h)$ into K equal segments: $\{[k\delta, (k+1)\delta) : k = 0, ..., K-1\}$, where $\delta = h/K$. Over $[k\delta, (k+1)\delta)$, we approximate S_t by a time-homogeneous Markov process with drift $\mu(k\delta, S_t)$, volatility $\sigma(k\delta, S_t)$, and jump intensity measure $\mathcal{J}(k\delta, S_t, dy)$, which is in turn approximated by a CTMC X with state space $\mathbb{G}$ whose cardinality is M, i.e., $|\mathbb{G}| = M$, and generator matrix $\mathcal{G}_{k\delta} \in \mathbb{R}^{M\times M}$.

Consider pricing autocallable I without snowball effect. The algorithm is largely similar to the time-homogeneous case, but we must also calculate the value function at intermediate time points within any monitoring interval of length h for the autocallable, which is different. Furthermore, distinct generator matrices are used for different time segments, leading to more computations of matrix exponentials.

For any $x \in \mathbb{G}$ and $k = 0, 1, ..., K$, let

$$f(k, x) := \mathbf{E}[\phi(X_{nh})|X_{nh-k\delta} = x],$$
$$g(k, x) := \mathbf{E}[\upsilon(X_{nh})\mathbf{1}_{\{\tau_H^X > k\delta\}}|X_{nh-k\delta} = x].$$

In addition, for $i = 1, ..., n$, $k = 0, 1, ..., K$, and $x \in \mathbb{G}$, let

$$h(i, k, x) := \mathbf{E}[V^i(X_{ih})|X_{ih-k\delta} = x].$$

Our pricing algorithm calculates $h(i, k, x)$ recursively in two steps:

Step 1: Let m_H denote the index such that $x_{m_H-1} = H$. Set $\hat{\mathcal{G}}_{k\delta} = (\mathcal{G}_{k\delta})_{m_H:M,m_H:M}$ for $k = 0, 1, ..., nK-1$. Calculate matrix exponential $A_{k\delta} = \exp(\mathcal{G}_{k\delta}\delta)$ and $\hat{A}_{k\delta} = \exp(\hat{\mathcal{G}}_{k\delta}\delta)$ for $k = 0, 1, ..., nK-1$. Set $f(0, x_m) = \phi(x_m)$ and $f(0, x_m) = \upsilon(x_m)$ for $m = 1, ..., M$. Compute the following for k from 1 to K:

$$f(k, x_m) = (A_{nh-k\delta} f(k-1, \cdot))(x_m), \quad \text{for } m = 1, ..., M,$$
$$g(k, x_m) = (\hat{A}_{nh-k\delta} g(k-1, \cdot))(x_m), \quad \text{for } m = m_H, ..., M.$$

Then compute

$$h(n, K, x_m) = f(K, x_m) + g(K, x_m), \quad \text{for } m = 1, ..., M.$$

This step costs $O(M^2K)$ plus the cost of computing nK matrix exponentials.

Step 2: Do the following for i running backward from $n-1$ to 1: (1) Calculate

$$\begin{aligned} h(i, 0, x_m) = {} & Nz_i 1_{\{C_i \le x_m < D_i\}} + N(1+y_i)1_{\{D_i \le x_m < U_i\}} \\ & + Nx_m/S_0 1_{\{x_m \ge U_i\}} \\ & + e^{-rh} h(i+1, K, x_m) 1_{\{x_m < D_i\}}, \quad \text{for } m = 1, ..., M, \end{aligned}$$

where $h(i+1, K, x_m)$ has been calculated from the previous iteration.

(2) For k from 1 to K, compute:

$$h(i,k,x_m) = \big(A_{ih-k\delta}h(i,k-1,\cdot)\big)(x_m), \quad \text{for } m = 1,...,M,$$

where $h(i,k-1,\cdot)$ is an M-dimensional vector whose mth component is $h(i,k-1,x_m)$. Finally, if the initial asset price x is a grid point, $V(x) = e^{-rh}h(1,K,x)$. Otherwise, $V(x)$ can be estimated by interpolating the values on the grid. The cost of calculating $\{h(i,k,x_m) : m = 1,...,M; k = 0,...,K\}$ for all i is $O(nM^2K)$.

Combining the two steps, the total cost is $O(nM^2K)$ plus the cost of computing nK matrix exponentials. Compared with the time-homogeneous case, the computational cost scales with K as time discretization is needed.

3.6 Pricing under SLV models

We consider a general SLV model defined by

$$\begin{cases} dS_t = \omega\big(S_t, v_t\big)dt + m\big(v_t\big)\Gamma\big(S_t\big)dW_t^{(1)}, \\ dv_t = \mu\big(v_t\big)dt + \sigma\big(v_t\big)dW_t^{(2)}, \end{cases} \tag{5}$$

where $\big[W^{(1)}, W^{(2)}\big]_t = \rho t$ with $\rho \in [-1,1]$. Cui et al. (2018) show how to construct a CTMC to approximate (S_t, v_t). They define a transformed process $Z_t = \xi\big(S_t\big) - \rho\eta\big(v_t\big)$ with $\xi(x) = \int_\cdot^x \frac{1}{\Gamma(u)}du$ and $\eta(x) = \int_\cdot^x \frac{m(u)}{\sigma(u)}du$ and it follows that

$$dZ_t = \theta\big(Z_t, v_t\big)dt + \sqrt{1-\rho^2}m\big(v_t\big)dW_t^*,$$

where W^* is a standard Brownian motion independent of $W^{(2)}$ and,

$$\theta(z,v) = \left(\frac{\omega(\zeta(z,v),v)}{\Gamma(\zeta(z,v))} - \frac{\Gamma'(\zeta(z,v))}{2}m^2(v) - \rho h(v)\right),$$

with $\zeta(z,v) = \xi^{-1}(z+\rho\eta(v))$ and $h(x) = \mu(x)\frac{m(x)}{\sigma(x)} + \frac{1}{2}\big(\sigma(x)m'(x) - \sigma'(x)m(x)\big)$. The asset price S_t can be recovered as $S_t = \zeta(Z_t, v_t)$. Then, they construct a two-layer regime-switching CTMC $(X_t, \tilde{v}_t)$ with M states for X_t and L states for $\tilde{v}_t$ to approximate (Z_t, v_t). Specifically, $\tilde{v}_t$ is first constructed using the SDE of v_t. Then, Z_t is approximated by $\tilde{Z}_t$ that follows

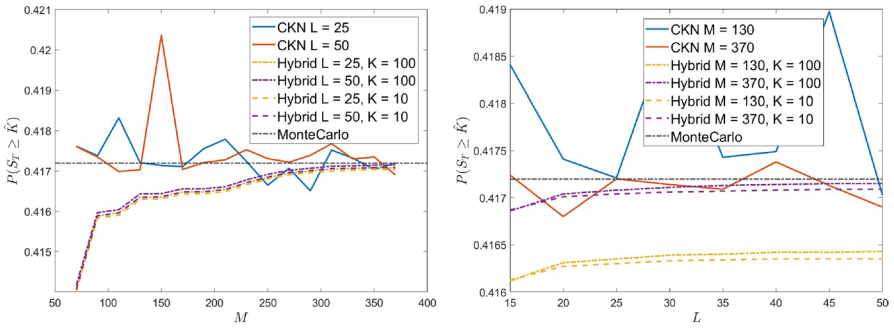
$$d\tilde{Z}_t = \theta\big(\tilde{Z}_t, \tilde{v}_t\big)dt + \sqrt{1-\rho^2}m\big(\tilde{v}_t\big)dW_t^*, \tag{6}$$

which is a regime-switching diffusion. A regime-switching CTMC $(X_t, \tilde{v}_t)$ is constructed to approximate $(\tilde{Z}_t, \tilde{v}_t)$, which also approximates (Z_t, v_t). To calculate expectations of the regime-switching CTMC, Cui et al. (2018) convert it to a univariate CTMC as in Sect. 3.4 with ML states using the result in Cai et al. (2019). They construct a nonuniform grid for $\tilde{v}_t$ denoted by $\mathbb{G}_v$ with L points. To design the grid of X_t denoted by $\mathbb{G}_X$, they start with a grid of S_t denoted by $\mathbb{G}_S$ with M points and then set $\mathbb{G}_X = \{\xi(s) - \rho\eta(v_0) : s \in \mathbb{G}_S\}$. Details of the grid design can be found in Cui et al. (2018). Hereafter, we refer to their method as CKN.

However, there are two potential issues when employing the CKN method for pricing some products. First, we need to calculate a matrix exponential with dimension ML, which is time consuming when ML is large. Second, if the product's payoff is discontinuous (like the digital payoff), convergence of the method is oscillatory. Thus, we cannot apply Richardson extrapolation to accelerate convergence.

To illustrate these two problems, we carry out a simple numerical experiment, where we consider the SABR model with its parameters given in Section 4.2.2 in Cui et al. (2018). However, instead of pricing a European call option as in Cui et al. (2018), we price a digital call option with payoff $1_{\{S_T \geq \hat{K}\}}$ as the autocallable payoff contains many such indicators. The grid design exactly follows Cui et al. (2018) and $S_0 = 0.05, T = 1$ and $\hat{K} = 0.052$. We price the digital call under two settings: (1) fixing $L = 25$ and 50, and varying M from 70 to 370 (see Fig. 1a); 2) fixing $M = 130$ and 370, and varying L from 15 to 50 (see Fig. 1b). We also price the digital call using the Monte Carlo method with 10^9 replications and 2500 time steps, and the result is stable in the first four decimal places. We use it as a benchmark for CTMC approximation. In both settings, we can see that the CKN method exhibits oscillations, making Richardson extrapolation inapplicable. We also display its running time in Table 1a, from which we observe sharp increase in the computation time as either M or L increases.

To address the computational issue, we propose a two-layer hybrid Markov chain approximation scheme as a remedy, and sometimes we simply call it Hybrid. We approximate (Z_t, v_t) by $(X_t, \tilde{v}_t)$, which are constructed as follows. First, we divide the time interval $[0, h)$ into K small equal segments $\{[k\delta, (k+1)\delta) : k = 0, 1, ..., K-1\}$ with $\delta = h/K$, and set $\tilde{v}_t$ constant over each segment. Given $\mathbb{G}_v = \{v_1, ..., v_L\}$ as the grid of $\tilde{v}_t$, the transition from $\tilde{v}_{k\delta}$ to $\tilde{v}_{(k+1)\delta}$ follows a discrete-time Markov chain (DTMC) with state space $\mathbb{G}_v$ and transition probability matrix $A^v = \exp(\Lambda\delta) \in \mathbb{R}^{L\times L}$ where $\Lambda \in \mathbb{R}^{L\times L}$ is the generator matrix of the CTMC approximation for v_t with state space $\mathbb{G}_v$. Given $\mathbb{G}_X = \{x_1, ..., x_M\}$ as the state space of X_t, on each time segment $[k\delta, (k+1)\delta)$, if $\tilde{v}_{k\delta} = v_l \in \mathbb{G}_v$, X_t is a CTMC with generator matrix $\mathcal{G}_{v_l} \in \mathbb{R}^{M\times M}$, which is constructed by approximating the generator of the SDE (6).



(a) $P(S_T \geq \hat{K})$ vs. M. (b) $P(S_T \geq \hat{K})$ vs. L.

Fig. 1 Approximations of $P(S_T \geq \hat{K})$ under SABR

Table 1 Running times (seconds) of two Markov chain approximation schemes for calculating $P(S_T \geq \hat{K})$. In (b), the first and second times are for $K = 100$ and $K = 10$, respectively

		M			
		90	190	270	370
(a) CKN					
L	25	1	52	113	257
	40	18	200	582	1327
	50	29	403	1100	2584
(b) Hybrid, $K = 100, 10$.					
L	25	2, 1	8, 6	19, 15	35, 34
	40	6, 2	22, 14	44, 34	82, 74
	50	11, 3	37, 21	73, 53	144, 115

Compared with CKN, in our scheme the volatility state v_t is approximated by a DTMC $\tilde{v}_t$ instead of a CTMC. As $\tilde{v}_t$ stays constant on $t \in [k\delta, (k+1)\delta)$, we only need to consider the transition of X_t on this time interval. In contrast, if $\tilde{v}_t$ is a CTMC as in CKN, its state can change on $[k\delta, (k+1)\delta)$ and thus we must consider the joint transition of $(X_t, \tilde{v}_t)$. As a result, CKN requires computing the matrix exponential of an $ML \times ML$ matrix, but our method only needs to compute the matrix exponentials of an $L \times L$ matrix and L^2 matrices of size $M \times M$, which can be substantially faster when ML is large (see Table 1).

Convergence. We show that by increasing the number of time steps, K, to infinity, the Markov chain in our Hybrid scheme converges weakly to the two-layer CTMC constructed in Cui et al. (2018). Let $(X_t^{(M,L)}, \tilde{v}_t^{(L)})$ be the CTMC living on the grid $\mathbb{G}^{(M,L)} := \mathbb{G}_X \times \mathbb{G}_v$ in the CKN method with fixed M, L. The time grid is given by $\{0, \delta_K, 2\delta_K, \cdots\}$ where $\delta_K \to 0$ as $K \to \infty$. Consider the Markov chain in the Hybrid scheme, $\{(X_t^{(M,L,K)}, \tilde{v}_t^{(L,K)}),\ t = 0, \delta_K, \cdots\}$. Let $J := \{f : \mathbb{G}^{(M,L)} \to \mathbb{R}\}$ be the space of functions defined on the grid $\mathbb{G}^{(M,L)}$, which are all bounded because they are defined on a finite number of points.

Proposition 1 *For any $f \in J$ and for all $t \geq 0$, we have for any $(x, v) \in \mathbb{G}^{(M,L)}$,*

$$\mathbf{E}_{x,v}\left[f\left(X^{(M,L,K)}_{\lfloor t/\delta_K \rfloor \delta_K}, \tilde{v}^{(L,K)}_{\lfloor t/\delta_K \rfloor \delta_K}\right)\right] \to \mathbf{E}_{x,v}\left[f\left(X_t^{(M,L)}, \tilde{v}_t^{(L)}\right)\right], \quad \text{as } K \to \infty. \tag{7}$$

The proof is provided in Appendix B. The proposition implies that with sufficiently small time steps the results from the Hybrid scheme are very close to those from CKN. It is shown in Proposition 4 of Cui et al. (2018) that $(X_t^{(M,L)}, \tilde{v}_t^{(L)})$ weakly converges to the original process (Z_t, v_t) as $M, L \to \infty$ under suitable conditions. Thus, combining these results, we see that our hybrid scheme can achieve accurate approximations by choosing large values for M, L, K.

To address the issue of convergence oscillations, we allow the grid of X_t to depend on the volatility state in our implementation. This creates another difference from CKN, where the grid of X_t is independent from the volatility state. We consider this flexibility for grid design because we want to utilize the results in Zhang and Li (2019) to remove oscillations. Below we describe how the grids are constructed.

As $S_t = \zeta(Z_t, v_t)$, we can view $(\tilde{S}_t := \zeta(X_t, \tilde{v}_t), \tilde{v}_t)$ as a Markov chain that approximates (S_t, v_t). Note that the payoff is only a function of the stock price S_t. Thus, to ensure second-order convergence and remove oscillations, we only need the grid of $\tilde{S}_t$ to satisfy the design principles laid out in Zhang and Li (2019). We construct a piecewise uniform grid $\mathbb{G}_S$ for $\tilde{S}_t$ by (4), which satisfies the grid design principles. We also construct a piecewise uniform grid for $\tilde{v}_t$ as

$$\mathbb{G}_v = \{v_1 + jh_1 : 0 \le j \le n_1\} \cup \{v_0 + jh_2 : 1 \le j \le n_2\},$$

where $n_1 = \lfloor L(v_0 - v_1)/(v_L - v_1) \rfloor$, $n_2 = L - n_1 - 1$, $h_1 = (v_0 - v_1)/n_1$ and $h_2 = (v_L - v_0)/n_2$. This design makes v_0 on the grid. For both $\mathbb{G}_S$ and $\mathbb{G}_v$, the smallest and largest grid points are chosen sufficiently small and large. As $\tilde{S}_t = \zeta(X_t, \tilde{v}_t)$ implies $X_t = \xi(\tilde{S}_t) - \rho\eta(\tilde{v}_t)$, we can obtain the grid for X_t given $\tilde{v}_t = v_l$ from $\mathbb{G}_S$ as

$$\mathbb{G}_X^{(l)} = \{\xi(S) - \rho\eta(v_l) : S \in \mathbb{G}_S\}, .$$

Thus, the grid of X_t depends on the value $\tilde{v}_t$ takes.

We apply the hybrid Markov chain method to price the product with digital payoff $\mathbf{1}_{S_T \ge \tilde{K}}$ using our grid design under the SABR parameters specified in Section 4.2.2 of Cui et al. (2018). The results demonstrate that our method achieves stable convergence, as illustrated in Fig. 1. Moreover, Table 1 show that our method enhances computational efficiency considerably. We have tried two values of K, and clearly the choice of K brings a tradeoff between accuracy and computational efficiency. Reducing K from 100 to 10, the running time decreases in Table 1b, but the accuracy becomes slightly worse in Fig. 1.

Remark 4 One advantage of the CKN method is that it does not discretize time. By using a DTMC instead of a CTMC to approximate v_t, we introduce time discretization error. However, our numerical experiment shows that this error is insignificant compared with other sources of error if K is chosen properly. As can be seen from Table 1, even if K is quite large, such as 100, our scheme still takes much less time than CKN.

Autocallable pricing. We present our pricing algorithm for autocallable I without the snowball effect under the SLV model (the algorithms for the other cases can be developed in a similar way). For any $x \in \mathbb{G}_X^{(l)}$ and $v_l \in \mathbb{G}_v$, let

$$\begin{aligned} h(i,k,x,v_l) &:= \mathbf{E}_{x,v_l}[V^{i+1}(\zeta(X_{h-k\delta}, \tilde{v}_{h-k\delta}))], \quad i = 0, ..., n-2; k = 0, 1, ..., K, \\ h(n-1,k,x,v_l) &:= \mathbf{E}_{x,v_l}[V^{n}(\zeta(X_{h-k\delta}, \tilde{v}_{h-k\delta}), m^X_{h-k\delta})], \quad k = 0, 1, ..., K, \end{aligned}$$

where $h(i,k,x,v_l)$ represents the value of the autocallable at time $ih + k\delta$ with state (x, v_l). Then by the law of iterated expectations, it satisfies that for any $i = 0, ..., n-2$ and $k = 0, 1, ..., K-1$,

$$h(i,k,x,v_l) = \mathbf{E}_{x,v_l}\big[h(i,k+1,X_\delta,\tilde{v}_\delta)\big] = \mathbf{E}_{x,v_l}\big[\mathbf{E}\big[h(i,k+1,X_\delta,\tilde{v}_\delta)|\tilde{v}_\delta,X_0=x,\tilde{v}_0=v_l\big]\big],$$

Let

$$\tilde{h}(i,k+1,x,v_l,v_s) := \mathbf{E}\big[h(i,k+1,X_\delta,\tilde{v}_\delta)|\tilde{v}_\delta=v_s,X_0=x,\tilde{v}_0=v_l\big].$$

For $x \in \mathbb{G}_X^{(s)}$, we have

$$\tilde{h}(i,k+1,x,v_l,v_s) = (\exp(\mathcal{G}_{v_l}^{(s)}\delta)h(i,k+1,\cdot,v_s))(x).$$

For any $v_s \in \mathbb{G}_v$, we apply interpolation to the values of $\tilde{h}(i,k+1,x,v_l,v_s)$ on the grid $\mathbb{G}_X^{(s)}$ to obtain the value of $\tilde{h}(i,k+1,x,v_l,v_s)$ for $x \in \mathbb{G}_X^{(l)}$. Then, for any $x \in \mathbb{G}_X^{(l)}$, we obtain

$$h(i,k,x,v_l) = (A^v\tilde{h}(i,k+1,x,v_l,\cdot))(v_l).$$

Thus, $h(i,0,x,v)$ can be computed recursively. Recall that the payoff of autocallable I at t_n is the summation of ϕ and υ defined in (1), where ϕ is the payoff without the need for monitoring H and υ gives the payoff paid to the investor only when H has not been down-crossed during $[t_{n-1},t_n]$. For any $x \in \mathbb{G}_X^{(l)}$ and $v_l \in \mathbb{G}_v$, let

$$f(k,x,v_l) := \mathbf{E}_{x,v_l}[\phi(\zeta(X_{h-k\delta},\tilde{v}_{h-k\delta}))], \quad k=0,1,...,K,$$

where $f(k,x,v_l)$ represents the value of the autocallable payoff without the need for monitoring H at time $(n-1)h+k\delta$ with state (x,v_l). Similarly, by the law of iterated expectations, it satisfies that for any $k=0,1,...,K-1$,

$$f(k,x,v_l) = \mathbf{E}_{x,v_l}\big[\mathbf{E}\big[f(k+1,X_\delta,\tilde{v}_\delta)|\tilde{v}_\delta,X_0=x,\tilde{v}_0=v_l\big]\big].$$

Let

$$\tilde{f}(k+1,x,v_l,v_s) := \mathbf{E}\big[f(k+1,X_\delta,\tilde{v}_\delta)|\tilde{v}_\delta=v_s,X_0=x,\tilde{v}_0=v_l\big].$$

For $x \in \mathbb{G}_X^{(s)}$, we have

$$\tilde{f}(k+1,x,v_l,v_s) = (\exp(\mathcal{G}_{v_l}^{(s)}\delta)f(k+1,\cdot,v_s))(x).$$

For any $v_s \in \mathbb{G}_v$, we apply interpolation to the values of $\tilde{f}(k+1,x,v_l,v_s)$ on the grid $\mathbb{G}_X^{(s)}$ to obtain the value of $\tilde{f}(k+1,x,v_l,v_s)$ for $x \in \mathbb{G}_X^{(l)}$. Then, for any $x \in \mathbb{G}_X^{(l)}$, we obtain

$$f(k,x,v_l) = (A^v\tilde{f}(k+1,x,v_l,\cdot))(v_l).$$

Thus, $f(k,x,v_l)$ can also be computed recursively. Let $m_H^{(l)} = \{1 \le m \le M : \zeta(x_m^{(l)},v_l) > H\}$ denote the index of $\mathbb{G}_X^{(l)}$ such that the associated approximated asset price is larger than continuous barrier H. For any $x \in \mathbb{G}_X^{(l)}$ and $v_l \in \mathbb{G}_v$, let

$$g(k,x,v_l) := \mathbf{E}_{x,v_l}\left[v(\zeta(X_{h-k\delta},\tilde{v}_{h-k\delta}))\mathbf{1}_{\{\tau_H^X > h-k\delta\}}\right], \quad k=0,1,...,K,$$

where $g(k,x,v_l)$ represents the value of autocallable payoff with the need for monitoring H at time $(n-1)h+k\delta$ with state (x,v_l). By the law of iterated expectations, we have for any $k=0,1,...,K-1$,

$$g\big(k,x,v_l\big) = \mathbf{E}_{x,v_l}\Big[\mathbf{E}\Big[g(k+1,X_\delta,\tilde{v}_\delta)\mathbf{1}_{\{\tau_H^X>\delta\}}|\tilde{v}_\delta = v_s, X_0 = x, \tilde{v}_0 = v_l\Big]\Big].$$

Let

$$\tilde{g}(k+1,x,v_l,v_s) := \mathbf{E}\big[g(k+1,X_\delta,\tilde{v}_\delta)|\tilde{v}_\delta = v_s, X_0 = x, \tilde{v}_0 = v_l\big].$$

For $m \in m_H^{(s)}$, we have

$$\tilde{g}(k+1,x_m^{(s)},v_l,v_s) = \begin{cases} (\exp(\hat{\mathcal{G}}_{v_l}^{(s)}\delta)g(k+1,\cdot,v_s))(x), & m \in m_H^{(s)} \\ 0, & m \notin m_H^{(s)} \end{cases},$$

with $\hat{\mathcal{G}}_{v_l}^{(s)} = (\mathcal{G}_{v_l}^{(s)})_{m_H^{(s)},m_H^{(s)}}$. For any $v_s \in \mathbb{G}_v$, we apply interpolation to the values of $\tilde{g}(k+1,x,v_l,v_s)$ on the grid $\mathbb{G}_X^{(s)}$ to obtain the value of $\tilde{f}(k+1,x,v_l,v_s)$ for $x \in \mathbb{G}_X^{(l)}$. Then, we obtain

$$g(k,x_m^{(l)},v_l) = \begin{cases} (A^v\tilde{g}(k+1,x_m^{(l)},v_l,\cdot))(v_l), & m \in m_H^{(l)} \\ 0, & m \notin m_H^{(l)} \end{cases}.$$

Thus, $g(k,x,v_l)$ can also be computed recursively. Then our algorithm calculates $h(i,k,x,v)$ recursively in two steps:

Step 1: For every $l=1,...,L$, let $m_H^{(l)}$ denote the set $\{m : \zeta(x_m^{(l)},v_l) > H, x_m \in \mathbb{G}_X^{(l)}\}$ and set $\hat{\mathcal{G}}_{v_l}^{(s)} = (\mathcal{G}_{v_l}^{(s)})_{m_H^{(s)},m_H^{(s)}}$. Calculate matrix exponential $A^v = \exp(\Lambda\delta)$, $A_l^{(s)} = \exp(\mathcal{G}_{v_l}^{(s)}\delta)$ and $\hat{A}_l^{(s)} = \exp(\hat{\mathcal{G}}_{v_l}^{(s)}\delta)$ for $l,s=1,...,L$. $f(K,x,v) = \phi(\zeta(x,v))$ and $g(K,x,v) = v(\zeta(x,v))$. Compute the following for k from $K-1$ to 0:

$$\tilde{f}(k+1,x_m^{(s)},v_l,v_s) = (A_l^{(s)}f(k+1,\cdot,v_s))(x_m^{(s)}), \quad \text{for } m=1,...,M \text{ and } l,s=1,...,L,$$
$$\tilde{g}(k+1,x_m^{(s)},v_l,v_s) = (\hat{A}_l^{(s)}g(k+1,\cdot,v_s))(x_m^{(s)}), \quad \text{for } m \in m_H^{(s)} \text{ and } l,s=1,...,L,$$

Set $\tilde{g}(k+1,x_m^{(s)},v_l,v_s) = 0$ for $m \notin m_H^{(s)}$ and $l,s=1,...,L$. For every $l,s=1,...,L$, $\tilde{f}(k+1,x,v_l,v_s)$ and $\tilde{g}(k+1,x,v_l,v_s)$ are available for $x \in \mathbb{G}_X^{(s)}$ and we use interpolation to obtain $\tilde{f}(k+1,x,v_l,v_s)$ and $\tilde{g}(k+1,x,v_l,v_s)$ for $x \in \mathbb{G}_X^{(l)}$. Compute

$$f(k,x_m^{(l)},v_l) = (A^v\tilde{f}(k+1,x_m^{(l)},v_l,\cdot))(v_l), \quad \text{for } m=1,...,M \text{ and } l=1,...,L,$$
$$g(k,x_m^{(l)},v_l) = (A^v\tilde{g}(k+1,x_m^{(l)},v_l,\cdot))(v_l), \quad \text{for } m \in m_H^{(l)} \text{ and } l=1,...,L.$$

Set $g(k,x_m^{(l)},v_l) = 0$ for $m \notin m_H^{(l)}$ and $l=1,...,L$. Then compute

$$h(n-1,0,x_m^{(l)},v_l)=f(0,x_m^{(l)},v_l)+g(0,x_m^{(l)},v_l),\quad \text{for } m=1,...,M \text{ and } l=1,...,L.$$

This step costs $O(M^2L^2K)+O(ML^2K)=O(M^2L^2K)$ plus the cost of computing the $2L^2+1$ matrix exponentials.

Step 2: Do the following for i running backward from $n-2$ to 0: (1) Calculate

$$\begin{aligned}h(i,K,x_m^{(l)},v_l)=&Nz_i1_{\{C_i\le\zeta(x_m^{(l)},v_l)<D_i\}}+N(1+y_i)1_{\{D_i\le\zeta(x_m^{(l)},v_l)<U_i\}}\\&+N\zeta(x_m^{(l)},v_l)/S_01_{\{\zeta(x_m^{(l)},v_l)\ge U_i\}}\\&+e^{-rh}h(i+1,0,x_m^{(l)},v_l)1_{\{\zeta(x_m^{(l)},v_l)<D_i\}},\\&m=1,...,M \text{ and } l=1,...,L,\end{aligned}$$

where $h(i+1,0,\cdot,\cdot)$ has been calculated from previous iteration.

(2) For k from $K-1$ to 0, recursively compute the following:

$$\begin{aligned}\tilde{h}(i,k+1,x_m^{(s)},v_l,v_s)=&(A_l^{(s)}h(i,k+1,\cdot,v_s))(x_m^{(s)}),\\&\text{for } m=1,...,M \text{ and } l,s=1,...,L.\end{aligned}$$

For every $l,s=1,...,L$, $\tilde{h}(i,k+1,x,v_l,v_s)$ are available for $x\in\mathbb{G}_X^{(s)}$ and we use interpolation to obtain $\tilde{h}(i,k+1,x,v_l,v_s)$ for $x\in\mathbb{G}_X^{(l)}$. Compute

$$\begin{aligned}h(i,k,x_m^{(l)},v_l)=&\left(A^v\tilde{h}(i,k+1,x_m^{(l)},v_l,\cdot)\right)(v_l),\\&\text{for } m=1,...,M \text{ and } l=1,...,L.\end{aligned}$$

Finally, if the initial asset price and volatility $(x,\ v)$ is a grid point, $V(x)=e^{-rh}h(0,0,x,v)$. Otherwise, $V(x)$ can be estimated by interpolating the values on the grid. The cost of calculating $\{h(i,k,x_m,v_l):m=1,...,M;k=0,...,K;l=1,...,L\}$ for all i is $O(nM^2L^2K)+O(nML^2K)=O(nM^2L^2K)$.

Combining the two steps, the total cost is $O(nM^2L^2K)$ plus the cost of computing $2L^2+1$ matrix exponentials.

4 Numerical results

We evaluate the convergence behavior, efficiency, and accuracy of our method under various representative models. In all the examples, we set the risk-free interest rate $r=0.025$, the dividend yield $d=0$, and the initial asset price $S_0=100$. We set the model parameters by referencing other papers in the literature.

- The BS model: $dS_t=(r-d)S_tdt+\sigma S_tdW_t, t\ge 0$, where W is a standard 1D Brownian motion. We set $\sigma=0.3$.
- A regime-switching BS (RSBS) model with three regimes: $dS_t=(r-d)S_tdt+\sigma_{v_t}S_tdW_t, t\ge 0$, where v_t is a CTMC with state space $\{1,2,3\}$, and its transition rate matrix is given by

$$\Lambda = \begin{pmatrix} -0.75 & 0.75 & 0 \\ 0.25 & -0.5 & 0.25 \\ 0 & 0.75 & -0.75 \end{pmatrix}.$$

We set $\sigma = 0.2, 0.3, 0.4$ in the first, second, and third regime, respectively.

- A time-inhomogeneous BS (TIBS) model: $dS_t = (r-d)S_t dt + \sigma(t)S_t dW_t, t \geq 0$, where $\sigma(t) = 0.6|\sin(\pi/6 + t)|$.
- The CEV model (Davydov & Linetsky, 2001): $dS_t = (r-d)S_t dt + \sigma S_t^{1+\beta} dW_t, t \geq 0$. We set $\sigma = 300$ and $\beta = -1.5$.
- Kou's double-exponential jump-diffusion model (Kou, 2002):

$$\frac{dS_t}{S_{t-}} = (r - d - \mu\zeta)dt + \sigma dW_t + d\left(\sum_{i=1}^{N_t}\left(V_i - 1\right)\right),$$

where N_t is a Poisson process with arrival rate μ, $\{V_i, i \geq 1\}$ is a sequence of i.i.d. random variables with the density of $\ln V_i$ given by $f_{\ln V_i}(y) = p^+\eta^+ e^{-\eta^+ y}1_{\{y\geq 0\}} + p^-\eta^- e^{\eta^- y}1_{\{y<0\}}$, and $\zeta = \mathbf{E}[V_i] - 1 = \frac{p^+\eta^+}{\eta^+ - 1} + \frac{p^-\eta^-}{\eta^- + 1} - 1$. We set $\sigma = 0.3, p^+ = p^- = 0.5, \eta^+ = \eta^- = 10$, and $\mu = 3.0$.

- The VG model (Madan et al., 1998):

$$S_t = S_0 \exp\left((r-d)t + Z_t + \omega t\right),$$

where $Z_t = \theta\gamma_t(1;\nu) + \sigma W_{\gamma_t(1;\nu)}, \omega = \ln\left(1 - \theta\nu - \sigma^2\nu/2\right)/\nu$, and $\gamma_t(1;\nu)$ is an independent Gamma process with mean rate 1 and variance rate ν. We set $\theta = 0, \sigma = 0.3$, and $\nu = 0.006$.

- The SABR model (Hagan et al., 2002):

$$\begin{cases} dS_t = (r-d)S_t dt + v_t S_t^{\beta} dW_t^{(1)}, \\ dv_t = \alpha v_t dW_t^{(2)}, \end{cases}$$

where $W^{(1)}$ and $W^{(2)}$ are two 1D standard Brownian motions with correlation ρ. We set $\rho = -0.25, \alpha = 0.2, \beta = 0.5$, and $v_0 = 3$. In this case, $\xi(x) = x^{1-\beta}/(1-\beta)$, $\eta(x) = x/\alpha$, and the auxiliary state Z_t follows

$$dZ_t = \left[(r-d)(1-\beta)\left(Z_t + \rho\frac{v_t}{\alpha}\right) - \frac{\beta}{2(1-\beta)}\frac{v_t^2}{Z_t + \rho\frac{v_t}{\alpha}}\right]dt + \sqrt{1-\rho^2}v_t dW_t^*,$$

where W^* is a 1D standard Brownian motion independent of $W^{(2)}$.

- The Heston model (Heston, 1993):

$$\begin{cases} dS_t = (r-d)S_t dt + \sqrt{v_t}S_t dW_t^{(1)}, \\ dv_t = \kappa(\theta - v_t)dt + \sigma_v\sqrt{v_t}dW_t^{(2)}, \end{cases} \tag{8}$$

where $W^{(1)}$ and $W^{(2)}$ are two 1D standard Brownian motions with correlation ρ. We set $\rho = -0.75$, $\kappa = 4$, $\theta = 0.09$, $\sigma_v = 0.15$, and $v_0 = 0.09$. Under this model, $\xi(x) = \log(x)$, $\eta(x) = x/\sigma_v$, and the auxiliary state Z_t follows

$$dZ_t = \left(r - d - \frac{v_t}{2} - \frac{\rho\kappa(\theta - v_t)}{\sigma_v}\right)dt + \sqrt{1-\rho^2}\sqrt{v_t}dW_t^*,$$

where W^* is a 1D standard Brownian motion independent of $W^{(2)}$.

The BS and CEV models serve as two examples of diffusions. The Kou model represents jump-diffusions with finite jump activity while the VG model is a popular pure-jump model with infinite jump activity. For these 1D Markov models, the grid design follows Sect. 3.2 with $n_1 = n_2 = \cdots = n_d$ and $n_0 = n_{d+1} = 4n_1$. For SLV models, we consider SABR and Heston, which are two popular choices. We construct the grids for the volatility state and auxillary state following Sect. 3.6.

4.1 Autocallable I

We consider a 4-year contract with

$$S_0 = 100, n = 4, h = 1, D_1 = D_2 = D_3 = D_4 = 100, U_1 = U_2 = U_3 = U_4 = 115,$$
$$C_1 = C_2 = C_3 = C_4 = 90, B = 75, H = 80, y_1 = y_2 = y_3 = y_4 = 8\%,$$
$$z_1 = z_2 = z_3 = z_4 = 5\%.$$

We use our method to price it with and without the snowball effect.

Figure 2 shows the convergence behavior of our method under six models: BS, Kou, CEV, VG, RSBS, and TIBS. In each model, the benchmark for calculating the error is obtained by running our method with a large number of grid points until no change is observed in the fourth decimal place. In each plot, we draw triangles with slopes -1 and -2 in the lower left corner as a reference for the convergence rate. As a comparison, we also try the grid design in Mijatović and Pistorius (2013), which will be simply called the MP grid, under the BS model. MP makes the grid around several selected critical points dense. In our problem, they are given by the "strikes", "barrier", and the initial asset price S_0. The denseness around the critical points is controlled by a parameter α, whose value can have a significant influence on the error. We have tried various values of α and plot the result of $\alpha = 10^{-5}$ in Fig. 2a, which is the most accurate.

Figure 2a illustrates the importance of grid design for the convergence rate. While convergence is only first order using the MP grid, it becomes second order with our grid design. Second-order convergence is also observed in all the other models using our grid design except VG.

Figure 3 shows the convergence behavior of our method for pricing the autocallable under the SABR model. In each plot of 3c to 3e, we fix two of the three parameters M, L, and K, and check convergence in the remaining one. The benchmark for calculating the error is obtained by increasing the parameter under analysis until no change is observed in the fifth decimal place. Convergence with respect to M and L

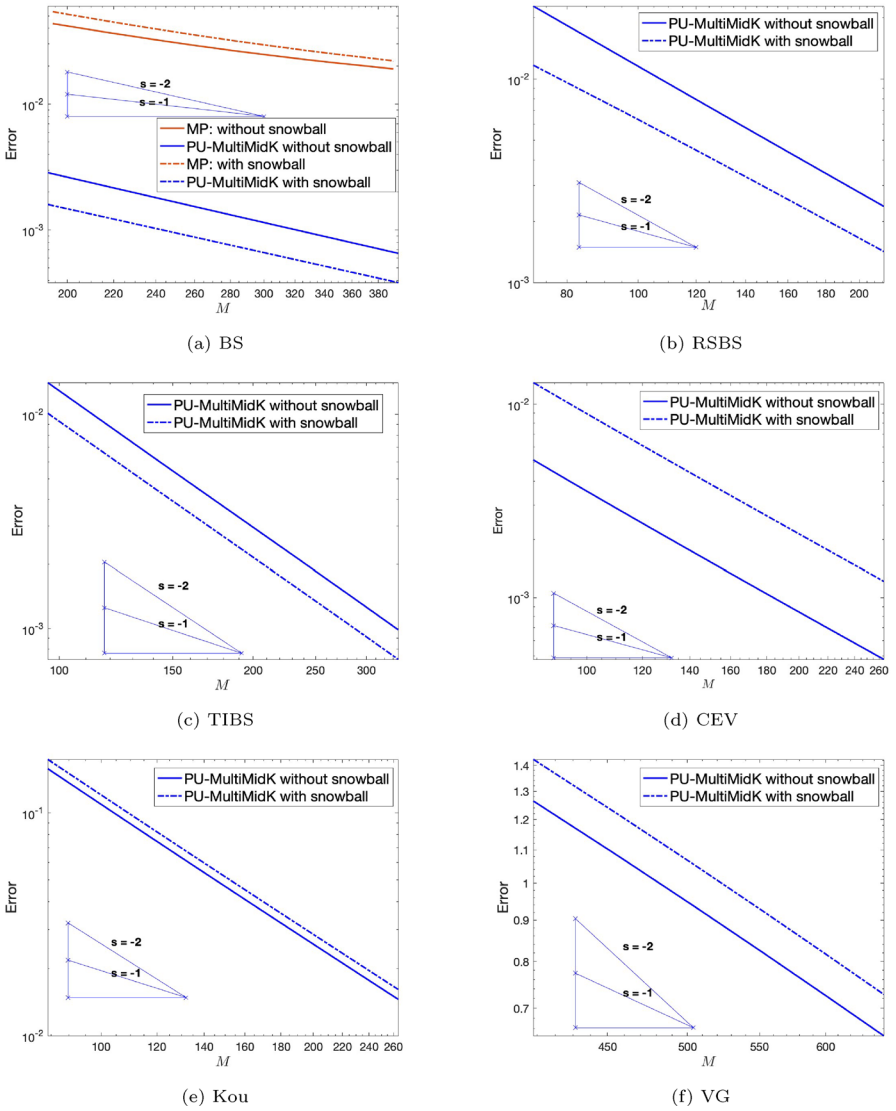
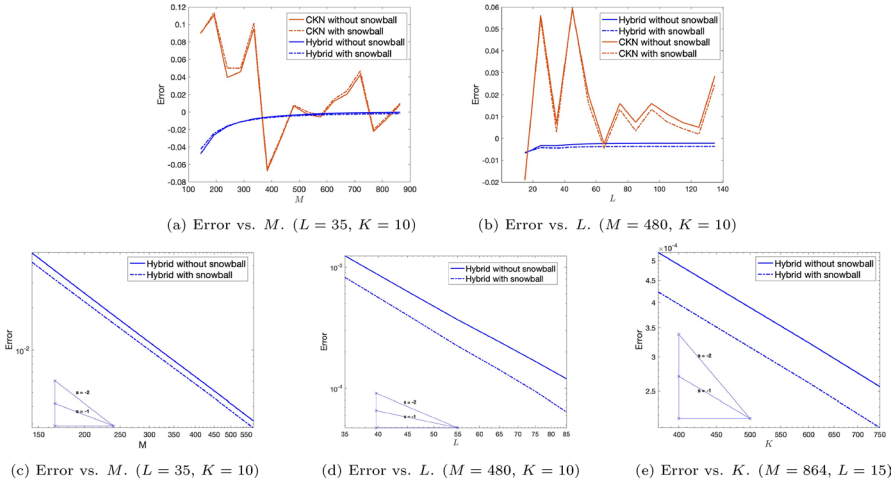


Fig. 2 Error versus M for pricing autocallable I (on log-log scale). "MP" represents the grid proposed by Mijatović and Pistorius (2013) and "PU-MultiMidK" represents the grid in Sect. 3.2. The estimated slope is – 1 for MP under the BS model. For the PU-MultiMidK grid, the estimated slopes are around – 2.00 under all the models except VG. Under VG model, the estimated slopes are around – 1.5

is at least second order while convergence of time discretization (i.e., with respect to K) is first order.

For comparison, we also implement the CKN method. In Fig. 3a and b, we fix L, K or M, K and show the convergence of CKN and our method with respect to M or L, and the benchmark price is obtained by Monte Carlo simulation accurate



(a) Error vs. M. ($L = 35$, $K = 10$) (b) Error vs. L. ($M = 480$, $K = 10$)

(c) Error vs. M. ($L = 35$, $K = 10$) (d) Error vs. L. ($M = 480$, $K = 10$) (e) Error vs. K. ($M = 864$, $L = 15$)

Fig. 3 Error versus *M*/*L*/*K* for the price of autocallable I (plots on the second row are on log-log scale). In 3c, the estimated slopes are close to − 2.00. In 3d, the estimated slope is − 2.61 for the case without snowball and − 2.85 for the case with snowball. In 3e, the estimated slopes are close to − 1.00

to the third decimal place. We also display their running times in Table 2. It is clear that our hybrid Markov chain approximation method can achieve more accurate results with substantially less time than CKN. Moreover, the convergence of our method is smooth so that one can further apply Richardson extrapolation to obtain more accurate results.

We further demonstrate the accuracy and speed of our method in Table 3. We run Monte Carlo simulation with 10^7 replications to generate an accurate benchmark and the 99% confidence interval. We observe that the result produced by our method lies within the confidence interval in every model. For the diffusion models, the number of grid points required to reach a high accuracy level is small (less than 100 in the experiment), and computation is completed in a small fraction of a second. For the jump and SLV models, we can apply extrapolation and still obtain highly accurate results quickly.

Table 2 Running time (in seconds) for pricing autocallable I under the SABR model with various M. We set $K = 10$ in Hybrid and $L = 35$ for both methods

		M			
		144	384	480	576
CKN	Without snowball	175	1571	3546	5767
	With snowball	175	1572	3547	5768
Hybrid	Without snowball	9	76	139	220
	With snowball	11	80	144	227

Table 3 Price of autocallable I under various models. Column "M" shows the number of grid points in the CTMC

Model	M	CTMC	Monte Carlo	99% CI	Abs. err.	Rel. err. (%)	Seconds
Price of autocallable I without the snowball effect							
BS	60	102.05	102.09	(102.04, 102.13)	0.03	0.03	0.004
RSBS	72	102.10	102.13	(102.10, 102.15)	0.03	0.03	0.017
TIBS	96	101.31	101.32	(101.29, 101.36)	0.02	0.02	0.104
CEV	60	101.40	101.38	(101.35, 101.40)	0.02	0.02	0.004
Kou	60/72	101.57	101.60	(101.55, 101.64)	0.03	0.03	0.011
VG	288/576	102.11	102.12	(102.09, 102.15)	0.01	0.01	1.476
SABR	144/192	101.88	101.88	(101.87, 101.89)	0.01	0.00	5.752
Price of autocallable I with the snowball effect							
BS	60	103.50	103.51	(103.47, 103.56)	0.02	0.02	0.004
RSBS	72	103.53	103.54	(103.51, 103.57)	0.01	0.01	0.019
TIBS	96	102.55	102.58	(102.54, 102.62)	0.03	0.03	0.106
CEV	60	102.78	102.77	(102.75, 102.80)	0.00	0.00	0.004
Kou	60/72	102.98	103.01	(102.97, 103.06)	0.03	0.03	0.013
VG	288/576	103.54	103.55	(103.52, 103.57)	0.01	0.01	1.706
SABR	144/192	103.27	103.27	(103.25, 103.28)	0.02	0.00	6.177

If two numbers are shown, it means that we apply Richardson extrapolation to the prices from these two numbers of grid points. For the TIBS model, we set $K = 10$. For the SABR model, we set $L = 15$ and $K = 10$

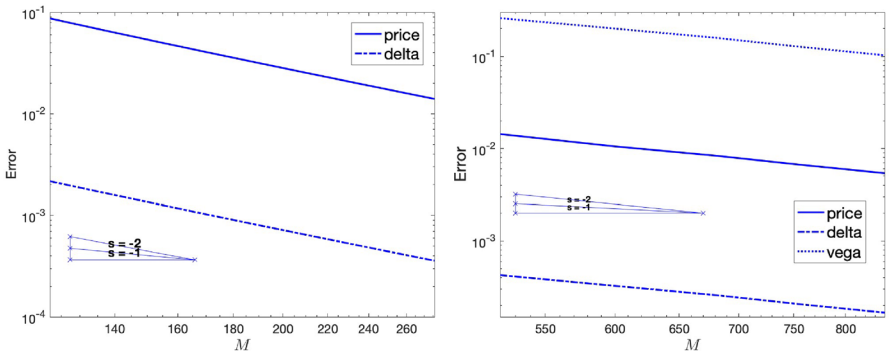
4.2 Autocallable II

We assess the performance of our method for autocallable structure II. We consider a 3-year maturity ELS product following Kim and Lim (2019):

$$\begin{aligned} S_0 = 100, n = 6, h = 0.5, D = (95, 95, 95, 90, 90, 90), B = 55, \\ y = (2\%, 4\%, 6\%, 8\%, 10\%, 12\%), z = (0, 0, 0, 0, 0, 0). \end{aligned} \tag{9}$$

We consider the BS and Heston models as examples. In addition to the price, we are concerned about the performance of approximating delta and vega, which are the derivative of the price with respect to S_t and v_t, respectively. These are important Greek letters for hedging, and we approximate them by applying finite difference to the prices given by our method. We obtain benchmarks by refining the grid until the fifth decimal places of the price and Greeks no longer change.

In the Heston model, we fix L and K and show the convergence pattern with respect to M. Figure 4 shows that convergence is about second order without oscillations for all the quantities in both models. The error of delta is much smaller than that of the price in both models. However, the error of vega can be significantly larger in the Heston model. Thus, a larger M is required for the vega than for the price to obtain similar accuracy.



(a) BS (b) Heston ($L = 55$, $K = 10$)

Fig. 4 Error versus M for price/delta/vega of autocallable II (on log-log scale). The estimated slopes are all around -2

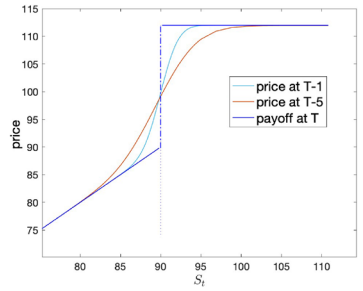
5 Hedging autocallables

The complex payoff structure of an autocallable can present significant challenges for risk management. If the hedger wants to apply delta hedging, which involves adjusting the portfolio's exposure to the underlying asset based on its delta, they may encounter problems when the delta becomes extremely large when the price of the underlying is near the barriers. This issue is especially acute for autocallable II, which involves a series of barriers monitored on payout dates as well as a barrier monitored continuously over the lifetime of the product. In this section, we consider hedging autocallable II to illustrate the issue and show how to address it. We calculate the autocallable's delta by using the CTMC approximation algorithm to get the price and apply finite difference.

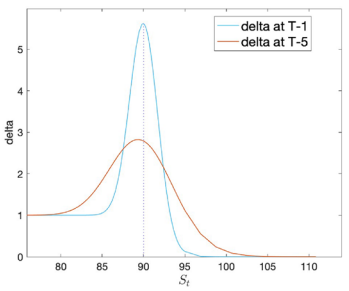
5.1 Behavior of delta near barriers

We demonstrate the behavior of delta near barriers by considering the 3-year ELS product studied in Kim and Lim (2019) with parameters given by (9). When the price of the underlying asset approaches the barriers close to the observation dates, the delta can become extremely large. Figure 5 provides two such examples, where we show the behavior of the price and delta of the autocallable 1 day and 5 days before the maturity time T (these two times are denoted by $T-1$ and $T-5$).

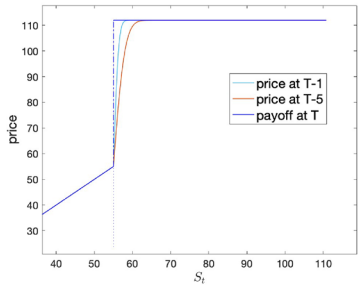
- Figure 5a and b show the price and delta around the discretely monitored barrier $D_6 = 90$ in the case where B has been down-crossed. The price increases rapidly as the asset price approaches D_6 from below. This is because whether the underlying price is greater than D_6 determines whether the initial capital and the final coupon y_n can be received, thereby creating a payoff gap at T. As a consequence, the delta can become large around D_6. This phenomenon is particularly pronounced when time is close to the payout date, which is T in the example.



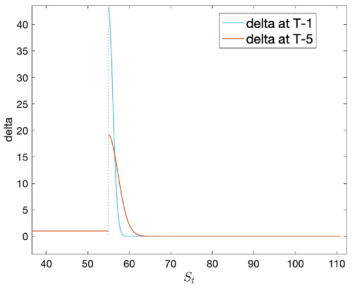
(a) price around $D_6 = 90$



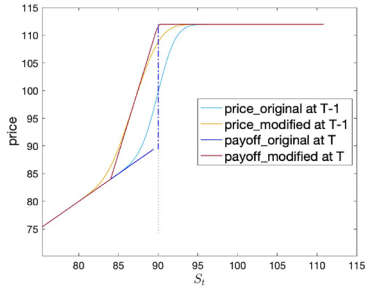
(b) delta around $D_6 = 90$



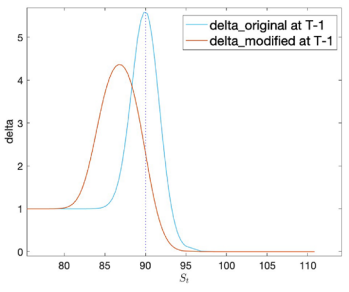
(c) price around $B = 55$



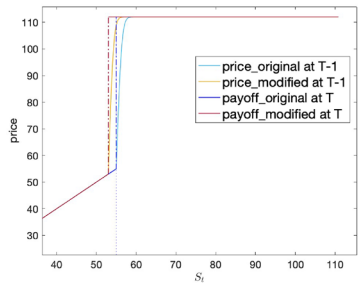
(d) delta around $B = 55$



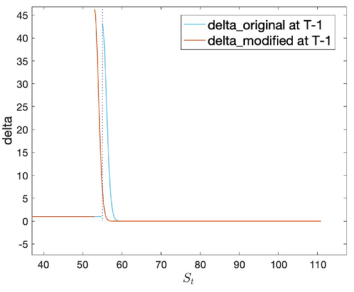
(e) prices of the original and modified products around $D_6 = 90$



(f) deltas of the original and modified products around $D_6 = 90$



(g) prices of the original and modified products around $B = 55$



(h) deltas of the original and modified products around $B = 55$

Fig. 5 Prices and deltas of the original and modified products. We make the plots around D_6 by assuming that B has been down-crossed and the plots around B by assuming that B has not been down-crossed

- Figure 5c and d show the price and delta around the continuously monitored barrier $B = 55$ in the case where B has not been down-crossed. Before down-crossing B, the price of the autocallable decreases sharply as the asset price approaches B from above. This is because the closer the asset price moves to B, the more likely B is to be down-crossed, which leads to a loss in the initial capital and hence a payoff gap at T. It follows that the delta increases sharply and the effect is even more extreme than the one for D_6.

Due to the substantial change in the delta near certain barriers, delta hedging can be costly or even impractical to implement. For instance, in Fig. 5d, as the asset price approaches B from above at $T-1$, the hedger has to buy a large amount of the underlying asset with the delta topping 40. But once the barrier B is breached, she has to liquidate a lot of the underlying as the delta jumps down to 1. Such dramatic changes in the position of the underlying can generate hefty transaction costs and may even be impossible to achieve in practice. Overall, the larger the payoff gap around the barriers, the more sensitive the delta is to changes in the asset price around these barriers, which makes the implementation of delta hedging very expensive.

To address this issue, Chan et al. (2019) apply two techniques, payoff modification and barrier shifting, to discretely and continuously monitored barriers, respectively. We use these techniques for our problem.

For the barrier D_6, the original payoff at T is given by

$$\mathbf{1}_{\{S_T<D_6\}}NS_T/S_0 + \mathbf{1}_{\{S_T\geq D_6\}}N(1+y_6).$$

We modify it as

$$\mathbf{1}_{\{S_T<D_6'\}}NS_T/S_0 + \mathbf{1}_{\{D_6'\leq S_T<D_6\}}\left(\Delta(S_T-D_6') + NS_T/S_0\right) + \mathbf{1}_{\{S_T\geq D_6\}}N(1+y_6), \tag{10}$$

where

$$D_6' < D_6 \quad \text{and} \quad \Delta = \frac{N(1+y_6) - NS_T/S_0}{D_6 - D_6'}.$$

Figure 5e shows that the modified payoff is smoother than the original payoffs. As a result, the delta of the product with the modified payoff changes much less in Fig. 5f. This makes it easier to hedge and results in lower transaction costs.

For the barrier B, we shift it to some B' less than B. Figure 5g and h show the payoff, price, and delta of the modified product with barrier $B' = 53$. Clearly, the delta does not exhibit substantial changes now when the asset price approaches B from above. The hedger implements delta hedging with the delta given by the modified product before B is down-crossed. But after B is breached, the hedger returns to using the delta of the original product for hedging.

The above discussion highlights the effectiveness of payoff modification and barrier shifting in improving delta hedging. However, the important question of how much to modify the payoff and how much to shift the barrier remains. Too much

modification and shifting can lead to substantial deviation of the modified product from the original one, thus increasing the hedging error. In general, there is a trade-off between the hedging error and transaction cost that must be balanced. Below we provide a systematic approach to address it.

5.2 Dynamic hedging by minimizing CVaR

We consider constructing a dynamic hedging portfolio to hedge a short position in autocallable II. Besides the bank account, there are d hedging instruments with their prices collected in the vector $Z_t \in \mathbb{R}^d$. The portfolio is self-financing with initial capital p_0, and the savings (or loans) in the bank account are expected to grow at the risk-free interest rate r. Let $\mathcal{T}$ denote the terminal date of the autocallable where $\mathcal{T} \in \{t_1, ..., t_n\}$. Our hedging strategy rebalances the hedging portfolio at a discrete set of time points $t_0 = \ell_0 < \ell_1 < \cdots < \ell_K = \mathcal{T}$. The number of shares held in the d instruments at ℓ_k is denoted by $\delta_{\ell_k} \in \mathbb{R}^d$ for $k = 0, 1, ..., K$. We define $\delta_{\ell_{-1}} = \delta_{\ell_K} = 0$ for convenience. For each trade, the transaction cost associated with the jth hedging instrument at time ℓ_k is determined by the function $c_k^j(\delta^j_{\ell_k} - \delta^j_{\ell_{k-1}})$ where $c_k^j(\cdot)$ represents the cost function.

We implement a dynamic hedging strategy with initial capital p_0. Let Φ_{ℓ_k} and B_{ℓ_k} denote the portfolio value and bank account value at ℓ_k after rebalancing for $k = 0, ..., K$. We have

$$\Phi_k = B_{\ell_k} + \sum_{j=1}^{d} Z^j_{\ell_k} \delta^j_{\ell_k}, \tag{11}$$

with $\Phi_{-1} = 0$. Then by self-financing, we obtain

$$\Phi_{\ell_{k+1}} - \Phi_{\ell_k} = \left(e^{r(\ell_{k+1} - \ell_k)} - 1\right) B_{\ell_k} + \sum_{j=1}^{d} \left(Z^j_{\ell_{k+1}} - Z^j_{\ell_k} \right) \delta^j_{\ell_k} - c^j_{k+1}\left(\delta^j_{\ell_{k+1}} - \delta^j_{\ell_k} \right) \Big). \tag{12}$$

It follows that

$$
\begin{aligned}
e^{-r\ell_K}\Phi_{\ell_K} &= \sum_{k=0}^{K}\left(e^{-r\ell_k}\Phi_{\ell_k} - e^{-r\ell_{k-1}}\Phi_{\ell_{k-1}}\right)\\
&= \sum_{k=0}^{K} e^{-r\ell_k}\left(\Phi_{\ell_k} - e^{r(\ell_k-\ell_{k-1})}\Phi_{\ell_{k-1}}\right)\\
&= \sum_{k=0}^{K} e^{-r\ell_k}\left(\Phi_{\ell_k} - \Phi_{\ell_{k-1}} - \left(e^{r(\ell_k-\ell_{k-1})}-1\right)\Phi_{\ell_{k-1}}\right)\\
&= \sum_{k=0}^{K} e^{-r\ell_k}\left(\sum_{j=1}^{d}\left((Z^j_{\ell_k} - Z^j_{\ell_{k-1}})\delta^j_{\ell_{k-1}} - c^j_k\left(\delta^j_{\ell_k} - \delta^j_{\ell_{k-1}}\right)\right)\right.\\
&\quad \left. -\left(e^{r(\ell_k-\ell_{k-1})}-1\right)\sum_{j=1}^{d} Z^j_{\ell_{k-1}}\delta^j_{\ell_{k-1}}\right)\\
&= (\delta\cdot Z)_{\mathcal{T}} - C_{\mathcal{T}}(\delta),
\end{aligned}
$$

where we use (11) and (12) to obtain the second to the last equality,

$$
(\delta\cdot Z)_{\mathcal{T}} := \sum_{k=1}^{K} e^{-r\ell_k}\sum_{j=1}^{d}\left[\left(Z^j_{\ell_k} - Z^j_{\ell_{k-1}}\right)\delta^j_{\ell_{k-1}} - \left(e^{r(\ell_k-\ell_{k-1})}-1\right)Z^j_{\ell_{k-1}}\delta^j_{\ell_{k-1}}\right]
$$

represents the total discounted net profit from holding the hedging instruments, and

$$
C_{\mathcal{T}}(\delta) := \sum_{k=0}^{K} e^{-r\ell_k}\sum_{j=1}^{d} c^j_k\left(\delta^j_{\ell_k} - \delta^j_{\ell_{k-1}}\right)
$$

gives the total discounted transaction cost.

At time t_0, the discounted hedging error is given by

$$
\mathcal{E}(p_0,\delta) = p_0 + (\delta\cdot Z)_{\mathcal{T}} - C_{\mathcal{T}}(\delta) - V,
$$

where

$$
\begin{aligned}
V = &\sum_{i=1}^{n} e^{-rt_i} N z_i \mathbf{1}_{\{t_i\le T\}}\\
&+ e^{-rT}N\left(\left((1+y_n)\mathbf{1}_{S_T\ge D_n} + \frac{S_T}{S_0}\mathbf{1}_{S_T<D_n}\right)(1-I_B(T)) + (1+y_n)I_B(T)\right)\mathbf{1}_{\{\mathcal{T}=T\}}
\end{aligned}
$$

represents the total discounted payoff to the holder of the autocallable product.

Controlling the tail risk of the hedging error is often considered in the literature; see Buehler et al. (2019) and the references therein. Here, we minimize the conditional value at risk (CVaR) of the hedging error $\mathcal{E}(p_0,\delta)$. CVaR is proposed in Artzner et al. (1999) and has been studied extensively as a coherent risk measure (McNeil et al., 2015). In our problem, given the confidence level $1-\alpha$, CVaR is defined by

$$\mathrm{CVaR}_\alpha(\mathcal{E}) = \mathbf{E}\big[-\mathcal{E} \mid -\mathcal{E} > \mathrm{VaR}_\alpha(\mathcal{E})\big], \quad \text{where} \quad \mathrm{VaR}_\alpha(\mathcal{E}) = \inf\{m \in \mathbb{R} : \mathbb{P}(\mathcal{E} < -m) \le \alpha\}.$$

The probability of the hedging shortfall exceeding $\mathrm{VaR}_\alpha(\mathcal{E})$ is bounded by α, and $\mathrm{CVaR}_\alpha(\mathcal{E})$ shows the expected hedging shortfall given that the shortfall is greater than $\mathrm{VaR}_\alpha(\mathcal{E})$.

In our hedging problem, p_0 is exogenously given (e.g., it is given by the risk-neutral price), and we look for the hedging strategy δ that minimizes $\mathrm{CVaR}_\alpha(\mathcal{E}(p_0, \delta))$. Using the cash invariance property of CVaR, we have

$$\mathrm{CVaR}_\alpha(\mathcal{E}(p_0, \delta)) = \mathrm{CVaR}_\alpha\big((\delta \cdot Z)_\mathcal{T} - C_\mathcal{T}(\delta) - V\big) - p_0. \tag{13}$$

Thus, in the following we just need to minimize the first term on the right hand side of (13).

In general, there is no closed-form formula to calculate the CVaR in our problem. We employ the following representation (see Buehler et al. (2019), Lemma 3.5 and Example 3):

$$\mathrm{CVaR}_\alpha(X) = \inf_{w \in \mathbb{R}} \left\{ w + \mathbf{E}\left[\frac{\max(-X - w, 0)}{1 - \alpha}\right] \right\}$$

Thus, the hedging problem becomes

$$\min_{w \in \mathbb{R}, \delta} \left\{ w + \frac{\mathbf{E}\big[\max(V - (\delta \cdot Z)_\mathcal{T} + C_\mathcal{T}(\delta) - w, 0)\big]}{1 - \alpha} \right\}.$$

In this paper, we do not consider general hedging strategies for δ, but instead look for the optimal strategy in a specific class. Below, we consider two types of hedging strategies depending on the asset price model.

Delta hedging strategy. We use the underlying asset as the only hedging instrument and perform delta hedging. Hence, we have $d = 1$, $Z_t = S_t$, and $\delta \in \mathbb{R}$. To deal with the delta explosion problem, we monitor $\Gamma(t, S_t)$, the absolute value of gamma of the original autocallable product. When the absolute gamma gets larger than a threshold Γ_0 around certain barriers, signaling that delta is changing fast, we begin to use the delta of the modified product for hedging.

Payoff modification is parametrized by a parameter D_i' at time t_i for $i = 1, \cdots, n$ as in (10) and we have $D_i' < D_i$. Barrier shifting is parametrized by a new barrier B' that satisfies $B' < B$. Let $I_B(t)$ be the indicator of the event that B has not been down-crossed by time t. We denote by $\Delta(t, S_t; (D_1, ..., D_i', ..., D_n), B')$ the delta of the modified product with continuously monitored barrier shifted to B' and payoff modification applied to D_i at time t. If $B' = B$ or $D_i' = D_i$, there is no barrier shifting or payoff modification applied to B or D_i. To calculate the delta of the modified product, we apply finite difference to the prices from our pricing algorithm. We denote by $\Delta_I(t, S_t)$ the delta at time t when implementing the hedging strategy. Let $I_\Gamma(t)$ be the indicator of the event that the absolute value of gamma has been larger than Γ_0 around B by time t, i.e., $I_\Gamma(t) = 1$ if and only if there is some $t' \le t$ such that

$$\Gamma(t', S_{t'}) \geq \Gamma_0 \text{ and } |S_{t'} - B| \leq |S_{t'} - D_{i'}|, \text{ where } i' = \min\{i = 1, ..., n : t_i \geq t'\}.$$

In addition, let $I^i_\Gamma(t)$ be the indicator of the event that the absolute value of gamma exceeds Γ_0 around D_i from t_{i-1} to t for $i = 1, ..., n$ and $t \in (t_{i-1}, t_i]$, i.e., $I^i_\Gamma(t) = 1$ if and only if there is $t' \in (t_{i-1}, t]$ such that

$$\Gamma(t', S_{t'}) \geq \Gamma_0 \text{ and } |S_{t'} - D_i| < |S_{t'} - B|.$$

Then for $t \in (t_{i-1}, t_i]$, $\Delta_I(t, S_t)$ is given by

$$\Delta_I(t, S_t) = \begin{cases} \Delta(t, S_t; (D_1, ..., D'_i, ..., D_n), B), & \text{if } I_\Gamma(t) I_B(t) = 0 \text{ and } I^i_\Gamma(t) = 1, \\ \Delta(t, S_t; (D_1, ..., D_n), B'), & \text{if } I_\Gamma(t) I_B(t) = 1 \text{ and } I^i_\Gamma(t) = 0, \\ \Delta(t, S_t; (D_1, ..., D'_i, ..., D_n), B'), & \text{if } I_\Gamma(t) I_B(t) = 1 \text{ and } I^i_\Gamma(t) = 1, \\ \Delta(t, S_t; (D_1, ..., D_n), B), & \text{otherwise.} \end{cases} \quad (14)$$

This means that for $t \in (t_{i-1}, t_i]$,

- If B has been down-crossed or the absolute gamma has not been larger than Γ_0 around B by t and the absolute gamma exceeds Γ_0 around D_i from t_{i-1} to t (case 1), we use the delta of the modified product with payoff modification applied to D_i;
- If B has not been down-crossed and the absolute gamma has been larger than Γ_0 around B by t and the absolute gamma has not exceeded Γ_0 around D_i from t_{i-1} to t (case 2), we use the delta of modified product with barrier shift applied to B;
- If B has not been down-crossed and the absolute gamma has been larger than Γ_0 around B by t and the absolute gamma has exceeded Γ_0 around D_i from t_{i-1} to t (case 3), we use the delta of modified product with both payoff modification applied to D_i and barrier shift applied to B;
- Otherwise, we use the delta of the original product.

Finally, in the delta hedging strategy, the holding position $\delta_t = \Delta_I(t, S_t)$.

Delta and vega hedging strategy. We use both the underlying S_t and variance swap as instruments to hedge against both delta and vega exposures. In this case, $d = 2$ and $Z_t = (S_t, V^{vs}_t)$, where V^{vs}_t denotes the price of the variance swap. Consider the general SLV model (5). The payoff of the variance swap with strike K_{var} is given by

$$V^{var}_T = \frac{1}{T}[\log(S_t)]_T - K_{var}$$

at terminal date T. The expectation of realized variance $\frac{1}{T}[\log(S_t)]_T$ can be expressed as rT minus the expectation of a log payoff (see Demeterfi et al. (1999)):

$$\mathbf{E}\left[\frac{1}{T}[\log(S_t)]_T\right] = \frac{2}{T}\left(rT - \mathbf{E}\left[\log\left(\frac{S_T}{S_0}\right)\right]\right)$$

which is also the fair strike K_{var} at inception. At any time $t_0 < T$, the price of variance swap $V^{vs}_t(S_t, v_t)$ should be give by

$$V_t^{var}(S_t, v_t) = e^{-r(T-t)}\left(\frac{1}{T}[\log(S_s)]_t + \mathbf{E}\left[\frac{1}{T}[\log(S_s)]_{t\le s\le T}|\mathcal{F}_t\right] - K_{var}\right)$$
$$= e^{-r(T-t)}\left(\frac{1}{T}[\log(S_s)]_t + \frac{2}{T}\left(r(T-t) - \mathbf{E}\left[\log\left(\frac{S_T}{S_t}\right)|\mathcal{F}_t\right]\right) - K_{var}\right)$$

where $\mathcal{F}_t = \sigma((S_s, v_s), 0 \le s \le t)$. The expectation of the log payoff in the above formula can be approximated by the hybrid Markov chain method in Sect. 3.6.

Given the price of the variance swap at any time, we show the dynamic hedging strategy against delta and vega exposures. Due to the high delta issue discussed in Sect. 5.1, we can use the delta of the modified product, denoted by $\Delta(t, S_t, v_t; (D_1, ..., D_i', ..., D_n), B')$, once the absolute gamma exceeds the threshold Γ_0. As there exists no such issue for vega, we always use the vega of the original product, denoted by $\mathcal{V}(t, S_t, v_t)$.

Let $\Delta_I(t, S_t, v_t)$ be the delta at time t when implementing the hedging strategy, which is determined by (14) but with $\Delta(t, S_t; \cdots)$ replaced by $\Delta(t, S_t, v_t; \cdots)$. In our hedging strategy, the holding positions of the underlying asset and variance swap are calculated by solving the following linear system:

$$\begin{cases} \Delta_I(t, S_t, v_t) = \delta_t^1 + \frac{\partial V_t^{var}(S_t, v_t)}{\partial s}\delta_t^2, \\ \mathcal{V}(t, S_t, v_t) = \frac{\partial V_t^{var}(S_t, v_t)}{\partial v}\delta_t^2. \end{cases}$$

In both strategies, the holding position of the underlying asset is a function of B' and D_i'. Thus, we can reformulate the optimization problem as

$$\min_{w, B'\le B, D_i'\le D_i, i=1,\cdots,n}\left\{w + \frac{\mathbf{E}\left[\max(V - (\delta\cdot Z)_T + C_{\mathcal{I}}(\delta) - w, 0)\right]}{1-\alpha}\right\} \tag{15}$$

where δ is a function of B' and D'. Problem (15) determines the optimal barrier shift and payoff modification for minimizing CVaR. The threshold Γ_0 for monitoring gamma can be viewed as a hyperparameter of the problem and we can try different values for it. The optimization problem only needs to be solved at the inception of the product. To solve it, we approximate the expectation by taking sample average (Kleywegt et al., 2002) and then apply stochastic gradient descent (SGD; see e.g., Bottou et al. (2018) for an introduction). On each rebalancing date along each sample path, we need to price the product and calculate its delta by the CTMC approximation algorithm. In our test, the sample paths are generated from the stochastic model used for pricing. However, it should be noted that the expectation in (15) is taken under the physical measure while we use the risk-neutral measure for pricing.

5.3 Empirical hedging results

We consider the 3-year ELS product discussed in Sect. 5.1 and test in two environments. One is given by the BS model with drift $\mu = 0.1$ and volatility $\sigma = 0.3$. The other is given by the Heston model (8) with $r - d$ replaced by $\mu = 0.1$, $\rho = -0.75$, $\kappa = 4$, $\theta = 0.09$, $\sigma_v = 0.15$, and $v_0 = 0.09$. Note that all these parameters are under

the physical measure. For the risk-neutral parameters, we replace μ by $r - d = 0.025$ (we set $r = 0.025$ and $d = 0$) and keep the others unchanged. In the first case, we hedge using only the underlying. In the second case, we hedge using the underlying and variance swap.

For the transaction cost of trading the underlying, we use the popular quadratic cost formulation proposed in Almgren and Chriss (2001), which imposes a heavier penalty on larger trades. Specifically,

$$c(q) = \frac{\eta(\bar{N}q)^2}{\bar{N}} = \eta\bar{N}q^2,$$

where q is the trading quantity. Following Almgren and Chriss (2001), we set $\eta = 2.5 \times 10^{-6}$, and $\bar{N} = 1.25 \times 10^5, 2.5 \times 10^5$ to consider different levels of liquidity. Almgren and Chriss (2001) assume that the daily trading volume is 5×10^6, so these two levels of $\bar{N}$ correspond to 2.5% and 5% of daily trading volume, respectively. For the variance swap, we assume a proportional transaction cost of 0.5% for simplicity. We have no reference to determine η and $\bar{N}$ for the variance swap if the quadratic cost formulation is used.

As p_0 is given, we consider the CVaR on the right hand side of (13) at 5% and 1% significance levels and try $\Gamma_0 \in \{0, 0.125, 0.25, 0.5, 1\}$. We generate 4×10^5 simulated paths for training. We optimize (15) using SGD with mini-batches of size 1000 based on these paths and set the learning rate to 0.5. With this setting, we observe convergence in both cases. We generate 10^5 paths (independent of the training data) to assess the CVaR of the optimized strategy under different values of Γ_0 (the same paths are used for every Γ_0). As a benchmark, we also estimate the CVaR of the hedging strategy without modifying the contract for delta calculation with the same 10^5 paths.

Tables 4 and 5 show the resulting CVaR of our optimized strategy for different values of Γ_0 and the hedging strategy without modifying the contract for delta calculation under the BS and Heston model, respectively. It is evident from these tables that by modifying the contract appropriately for delta calculation we can achieve substantial reductions in CVaR, particularly at the 1% significance level.

Interestingly, the optimal Γ_0 is 0, implying that we should use the modified product for hedging from the beginning. This result is surprising, as any decrease in Γ_0 would typically result in decreased transaction costs and increased hedging errors, leading to a balance at some Γ_0. However, the hedging error from using of the delta of the modified product can be either negative (loss) or positive (gain). For example, shifting the barrier B causes the delta to be smaller for $S_t > B$ compared with the delta of the original product. As a result, when S_t rises, the modified product increases less in value than the original one, leading to a hedging loss. Conversely, when S_t falls, the modified product decreases less in value than the original one, leading to a hedging gain. Considering that in the worst-case scenarios, S_t often falls from $S_0 = 100$ to approximately $B = 55$, where substantial transaction costs are incurred, hedging with the modified product can result in less hedging loss. This explains why hedging with the delta of the modified product from the outset can yield the lowest CVaR.

Table 4 Optimal adjustment under various values of Γ_0. Column "original" shows the original barriers in the autocallable note. Row "*B*" and row "*D*" show the new barriers after adjustment

		$\Gamma_0 = 0$		$\Gamma_0 = 0.125$		$\Gamma_0 = 0.25$		$\Gamma_0 = 0.5$		$\Gamma_0 = 1$	
	Original	$CVaR_{0.05}$	$CVaR_{0.01}$	$CVaR_{0.05}$	$CVaR_{0.01}$	$CVaR_{0.05}$	$CVaR_{0.01}$	$CVaR_{0.05}$	$CVaR_{0.01}$	$CVaR_{0.05}$	$CVaR_{0.01}$
(a) BS, $\bar{N} = 1.25 \times 10^5$											
D	95	94.22	94.50	94.76	95.00	94.80	94.99	94.99	94.98	95.00	95.00
	95	94.79	94.96	94.98	95.00	94.94	94.92	95.00	95.00	95.00	95.00
	95	94.99	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00
	90	89.79	89.94	90.00	90.00	89.99	89.59	90.00	89.97	89.99	89.98
	90	89.88	89.85	89.98	89.63	89.92	89.70	89.90	89.53	90.00	90.00
	90	89.63	89.80	89.93	88.52	89.97	89.83	89.93	89.68	89.97	89.84
B	55	52.31	51.43	52.20	51.32	52.88	51.66	52.49	51.05	50.57	50.31
$CVaR^0$	–	103.62	124.89	103.62	124.89	103.62	124.89	103.62	124.89	103.62	124.89
$CVaR^*$	–	98.96	102.59	99.84	104.88	100.15	105.92	100.16	107.08	100.39	109.00
CVaR↓	–	5%	18%	4%	16%	3%	15%	3%	14%	3%	13%
(b) BS, $\bar{N} = 2.5 \times 10^5$											
D	95	94.51	95.00	94.89	95.00	94.96	95.00	94.98	95.00	95.00	95.00
	95	94.86	94.98	94.99	94.99	95.00	95.00	95.00	95.00	95.00	95.00
	95	94.98	94.94	95.00	94.99	95.00	94.98	95.00	95.00	95.00	95.00
	90	89.98	89.90	89.87	89.92	89.91	89.75	89.90	89.82	89.74	89.56
	90	89.89	89.89	86.68	82.48	89.69	88.24	89.83	88.89	89.83	89.47
	90	89.90	88.47	87.85	80.07	90.00	85.43	89.32	86.80	89.81	89.38
B	55	50.98	44.94	51.00	51.22	50.39	51.18	49.69	49.50	50.55	50.78
$CVaR^0$	–	110.78	152.78	110.78	152.78	110.78	152.78	110.78	152.78	110.78	152.78
$CVaR^*$	–	101.19	107.34	102.42	108.40	102.67	110.94	103.01	114.16	103.78	118.05
CVaR↓	–	9%	30%	8%	29%	7%	27%	7%	25%	6%	23%

$CVaR^0$ and $CVaR^*$ show the hedging results without modifying and after modifying the contract, respectively, and CVaR↓ gives the relative decrease of $CVaR^*$ compared with $CVaR^0$

Table 5 Optimal adjustment under various values of Γ_0. Column "original" shows the original barriers in the autocallable note. Row "*B*" and row "*D*" show the new barriers after adjustment

		$\Gamma_0 = 0$		$\Gamma_0 = 0.125$		$\Gamma_0 = 0.25$		$\Gamma_0 = 0.5$		$\Gamma_0 = 1$	
	Original	$\text{CVaR}_{0.05}$	$\text{CVaR}_{0.01}$	$\text{CVaR}_{0.05}$	$\text{CVaR}_{0.01}$	$\text{CVaR}_{0.05}$	$\text{CVaR}_{0.01}$	$\text{CVaR}_{0.05}$	$\text{CVaR}_{0.01}$	$\text{CVaR}_{0.05}$	$\text{CVaR}_{0.01}$
(a) Heston, $\bar{N} = 1.25 \times 10^5$											
D	95	92.19	94.11	91.84	94.58	94.27	94.42	92.30	94.51	95.00	95.00
	95	94.05	94.94	93.48	94.99	94.90	94.88	93.99	95.00	95.00	95.00
	95	94.85	94.90	94.61	94.79	94.98	95.00	94.70	95.00	95.00	95.00
	90	89.69	89.90	87.93	88.01	90.00	88.48	89.98	89.47	89.99	89.98
	90	89.89	89.67	86.72	87.50	90.00	86.42	88.68	87.80	90.00	90.00
	90	89.19	86.66	84.84	83.68	88.97	84.74	87.58	86.20	89.97	89.84
B	55	52.41	52.57	52.77	52.15	51.66	52.41	50.90	51.73	50.57	50.31
CVaR^0	–	102.68	120.43	102.68	120.43	102.68	120.43	102.68	120.43	102.68	120.43
CVaR^*	–	98.79	103.55	98.95	103.23	99.59	104.03	99.54	105.05	99.73	106.40
CVaR↓	–	4%	14%	4%	14%	3%	14%	3%	13%	3%	12%
(b) Heston, $\bar{N} = 2.5 \times 10^5$											
D	95	90.67	94.62	93.68	94.86	94.15	93.99	94.73	94.96	94.29	94.11
	95	93.99	95.00	94.77	94.82	94.77	93.81	95.00	95.00	93.87	93.81
	95	94.85	94.91	95.00	94.72	94.98	94.82	94.53	94.53	94.91	94.90
	90	89.90	87.63	90.00	87.71	88.35	88.87	87.26	87.29	87.72	87.79
	90	89.99	87.76	89.55	89.32	86.13	87.41	88.51	88.73	86.07	86.56
	90	89.61	83.16	87.66	86.17	84.21	85.78	83.37	84.15	84.38	84.63
B	55	51.34	50.56	51.22	49.65	50.80	51.23	50.06	50.14	49.68	50.08
CVaR^0	–	109.55	145.37	109.55	145.37	109.55	145.37	109.55	145.37	109.55	145.37
CVaR^*	–	101.42	107.47	102.07	109.05	102.22	109.62	102.34	110.80	102.59	112.10
CVaR↓	–	7%	26%	7%	25%	7%	25%	7%	24%	6%	23%

CVaR^0 and CVaR^* show the hedging results without modifying and after modifying the contract, respectively, and CVaR↓ gives the relative decrease of CVaR^* compared with CVaR^0

6 Conclusion

This paper proposes a general approach based on CTMC approximation for pricing autocallable structured products, which are popular in financial markets. Autocallables combine the features of both fixed-income and equity investments and allow investors to enhance their yields. The complex payoff of autocallables with many discontinuities presents a difficulty for many numerical methods to perform well. By carefully designing the grid of the Markov chain, we are able to achieve stable and fast convergence. Our method is applicable to a variety of commonly used stochastic models for derivatives pricing, including 1D Markov jump-diffusions (the coefficients can be time dependent), regime-switching models, and SLV models. The versatility of our method together with its computational efficiency makes it a competitive pricing tool in practice.

Hedging autocallables is also a difficult problem with transaction costs. We present a dynamic hedging approach in which we apply payoff modification and barrier shifting to deal with the high delta issue near the barriers. We determine the optimal size of adjustments that minimize the CVaR of the hedging loss and show that our approach can reduce CVaR significantly.

It is possible to apply our approach to the dividend model of Buehler (2010). We leave the detailed treatment of this model for future research. We also plan to extend our approach to deal with autocallables involving more than one asset and study pricing and hedging in models with stochastic interest rates.

A. Pseudocodes of pricing algorithms

We provide pseudocodes for pricing three types of autocallables when the underlying asset price follows a CTMC.

Algorithm 1 Calculate $V(x)$ for autocallable I without snowball effect

Require: $M, \mathbf{x}, \mathcal{G}, n, h, \mathbf{y}, \mathbf{z}, C, D, U, H, B, N, x_0$
Ensure: $\mathbf{A} \leftarrow \exp(\mathcal{G}h)$, $\hat{\mathbf{A}} \leftarrow \exp(\hat{\mathcal{G}}h)$,
$m_B \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m > B\}$, $m_D \leftarrow \min\{m : m_B \le m \le M, \mathbf{x}_m > D_n\}$, $m_U \leftarrow \min\{m : m_D \le m \le M, \mathbf{x}_m > U_n\}$, $m_H \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m \ge H\}$, $\upsilon \leftarrow (1_{m_D \le m < m_U})_{m_H \le m \le M} N \mathbf{y}_n$,
$\mathbf{v} \leftarrow (N 1_{m_B \le m < m_U} + N \mathbf{x}_m / x_0 1_{m \ge m_U} + N \mathbf{x}_m / x_0 1_{1 \le m < m_B})_{1 \le m \le M}$,
$\mathbf{v} \leftarrow \mathbf{A}\mathbf{v}$, $\mathbf{v}_{m_H \le m \le M} \leftarrow \mathbf{v}_{m_H \le m \le M} + \hat{\mathbf{A}}\upsilon$, $i \leftarrow n-1$
while $i > 0$ **do**
  $m_C \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m > C_i\}$
  $m_D \leftarrow \min\{m : m_C \le m \le M, \mathbf{x}_m > D_i\}$
  $m_U \leftarrow \min\{m : m_D \le m \le M, \mathbf{x}_m > U_i\}$
  $\mathbf{v} \leftarrow (N \mathbf{z}_i 1_{m_C \le m < m_D} + N(1 + \mathbf{y}_i) 1_{m_D \le m < m_U} + N \mathbf{x}_m / x_0 1_{m \ge m_U} + e^{-rh} \mathbf{v}_m 1_{1 \le m < m_D})_{1 \le m \le M}$
  $\mathbf{v} \leftarrow \mathbf{A}\mathbf{v}$
  $i \leftarrow i - 1$
end while
$m_0 \leftarrow \max\{m : 1 \le m \le M, \mathbf{x}_m < x_0\}$
$V \leftarrow e^{-rh}((\mathbf{x}_{m_0+1} - x_0)\mathbf{v}_{m_0} + (x_0 - \mathbf{x}_{m_0})\mathbf{v}_{m_0+1})/(\mathbf{x}_{m_0+1} - \mathbf{x}_{m_0})$

Algorithm 2 Calculate $V(x)$ for autocallable I with snowball effect

Require: $M, \mathbf{x}, \mathcal{G}, n, h, \mathbf{y}, \mathbf{z}, C, D, U, H, B, N, x_0$
Ensure: $\mathbf{A} \leftarrow \exp(\mathcal{G}h)$, $\hat{\mathbf{A}} \leftarrow \exp(\hat{\mathcal{G}}h)$,
$m_B \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m > B\}$, $m_C \leftarrow \min\{m : m_B \le m \le M, \mathbf{x}_m > C_n\}$, $m_D \leftarrow \min\{m : m_C \le m \le M, \mathbf{x}_m > D_n\}$, $m_U \leftarrow \min\{m : m_D \le m \le M, \mathbf{x}_m > U_n\}$,$m_H \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m \ge H\}$, $\upsilon \leftarrow (1_{m_D \le m < m_U})_{m_H \le m \le M} N \mathbf{y}_n$,
$\mathbf{v}^0 \leftarrow (N 1_{m_B \le m < m_U} + N\mathbf{x}_m/x_0 1_{m \ge m_U} + N\mathbf{x}_m/x_0 1_{1 \le m < m_B})_{1 \le m \le M}$,
$\mathbf{v}^0 \leftarrow \mathbf{A}\mathbf{v}^0$, $\mathbf{v}^0_{m_H \le m \le M} \leftarrow \mathbf{v}^0_{m_H \le m \le M} + \hat{\mathbf{A}}\upsilon$,
$\hat{\mathbf{v}} \leftarrow (N 1_{m \ge m_C})_{1 \le m \le M}$, $\hat{\mathbf{v}} \leftarrow \hat{\mathbf{A}}\hat{\mathbf{v}}$, $\tilde{z} \leftarrow 0$
for $q \in \{1, ..., n-1\}$ **do**
  $\tilde{z} \leftarrow \tilde{z} + z_{n-q}$
  $\mathbf{v}^q \leftarrow \mathbf{v}^0 + \tilde{z}\hat{\mathbf{v}}$
end for
$i \leftarrow n - 1$
while $i > 0$ **do**
  $m_C \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m > C_i\}$
  $m_D \leftarrow \min\{m : m_C \le m \le M, \mathbf{x}_m > D_i\}$
  $m_U \leftarrow \min\{m : m_D \le m \le M, \mathbf{x}_m > U_i\}$
  $\tilde{\mathbf{v}} \leftarrow (N\mathbf{z}_i 1_{m_C \le m < m_D} + N(1 + \mathbf{y}_i) 1_{m_D \le m < m_U} + N\mathbf{x}_m/x_0 1_{m \ge m_U} + e^{-rh}\mathbf{v}^0_m 1_{m_C \le m < m_D})_{1 \le m \le M}$
  $\mathbf{v}^0 \leftarrow \tilde{\mathbf{v}} + (e^{-rh}\mathbf{v}^{q+1}_m 1_{1 \le m < m_C})_{1 \le m \le M}$
  $\mathbf{v}^0 \leftarrow \mathbf{A}\mathbf{v}^0$
  $\tilde{z} \leftarrow 0$
  for $q \in \{1, ..., i-1\}$ **do**
    $\tilde{z} \leftarrow \tilde{z} + z_{i-q}$
    $\mathbf{v}^q \leftarrow \tilde{\mathbf{v}} + (e^{-rh}\mathbf{v}^{q+1}_m 1_{1 \le m < m_C} + N\tilde{z} 1_{m \ge m_C})_{1 \le m \le M}$
    $\mathbf{v}^q \leftarrow \mathbf{A}\mathbf{v}^q$
  end for
  $m \leftarrow m - 1$
end while
$m_0 \leftarrow \max\{m : 1 \le m \le M, \mathbf{x}_m < x_0\}$
$V \leftarrow e^{-rh}((\mathbf{x}_{m_0+1} - x_0)\mathbf{v}^0_{m_0} + (x_0 - \mathbf{x}_{m_0})\mathbf{v}^0_{m_0+1})/(\mathbf{x}_{m_0+1} - \mathbf{x}_{m_0})$

Algorithm 3 Calculate $V(x)$ for autocallable II

Require: $M, \mathbf{x}, \mathcal{G}, n, h, \mathbf{y}, \mathbf{z}, D, B, N, x_0$
Ensure: $\mathbf{A} \leftarrow \exp(\mathcal{G}h)$, $\bar{\mathbf{A}} \leftarrow \exp(\bar{\mathcal{G}}h)$,
$m_B \leftarrow \min\{m : 1 \le m \le M, \mathbf{x}_m \ge B\}$, $m_D \leftarrow \min\{m : m_B \le m \le M, \mathbf{x}_m > D_n\}$,
$\mathbf{v} \leftarrow (N(1 + \mathbf{y}_n) 1_{m \ge m_D} + N\mathbf{x}_m/x_0 1_{m < m_D} + N\mathbf{z}_n)_{1 \le m \le M}$,
$\mathbf{v}^1 \leftarrow (1_{1 \le m < m_D} N(1 + \mathbf{y}_n - \mathbf{x}_m))_{m_B \le m \le M}$,
$\mathbf{v} \leftarrow \mathbf{A}\mathbf{v}$, $\mathbf{v}^1 \leftarrow \bar{\mathbf{A}}\mathbf{v}^1$, $i \leftarrow n - 1$
while $i > 0$ **do**
  $m_D \leftarrow \min\{m : m_C \le m \le M, \mathbf{x}_m > D_i\}$
  $\mathbf{v} \leftarrow (N\mathbf{z}_i + N(1 + \mathbf{y}_i) 1_{m \ge m_D} + e^{-rh}\mathbf{v}_m 1_{1 \le m < m_D})_{1 \le m \le M}$
  $\mathbf{v}^1 \leftarrow (e^{-rh}\mathbf{v}^1_m 1_{1 \le m < m_D})_{m_B \le m \le M}$
  $\mathbf{v} \leftarrow \mathbf{A}\mathbf{v}$
  $\mathbf{v}^1 \leftarrow \bar{\mathbf{A}}\mathbf{v}^1$
  $i \leftarrow i - 1$
end while
$\mathbf{v}_{m_B \le m \le M} \leftarrow \mathbf{v}_{m_B \le m \le M} + \mathbf{v}^1$
$m_0 \leftarrow \max\{m : 1 \le m \le M, \mathbf{x}_m < x_0\}$
$V \leftarrow e^{-rh}((\mathbf{x}_{m_0+1} - x_0)\mathbf{v}_{m_0} + (x_0 - \mathbf{x}_{m_0})\mathbf{v}_{m_0+1})/(\mathbf{x}_{m_0+1} - \mathbf{x}_{m_0})$

B. Proof of Proposition 1

Let

$$(T(t)f)(x,v) := \mathbf{E}_{x,v}\left[f\left(X_t^{(M,L)}, \tilde{v}_t^{(L)}\right)\right], \quad f \in J,\ (x,v) \in \mathbb{G}^{(M,L)}.$$

$\{T(t),\ t \geq 0\}$ is a strongly continuous contraction semigroup on J with generator $\Lambda^{(M,L)}$ such that for any $m_0 = 1, ..., M$, $l_0 = 1, ..., L$,

$$(\Lambda^{(M,L)}f)(x_{m_0}, v_{l_0}) = \sum_{m=1}^{M}\left(\mathcal{G}_{v_{l_0}}\right)_{m_0,m} f(x_m, v_{l_0}) + \sum_{l=1}^{L}\Lambda_{l_0,l} f(x_{m_0}, v_l).$$

Clearly, J is a core of $\Lambda^{(M,L)}$. Consider

$$(T_K f)(x,v) := \mathbf{E}_{x,v}\left[f\left(X_{\delta_K}^{(M,L,K)}, \tilde{v}_{\delta_K}^{(L,K)}\right)\right], \quad f \in J,\ (x,v) \in \mathbb{G}^{(M,L)},$$

where T_K is a linear contraction on J and

$$\mathbf{E}_{x,v}\left[f\left(X_{\lfloor t/\delta_K\rfloor\delta_K}^{(M,L,K)}, \tilde{v}_{\lfloor t/\delta_K\rfloor\delta_K}^{(L,K)}\right)\right] = \left(T_K\right)^{\lfloor t/\delta_K\rfloor} f(x,v).$$

Let $A_K := \delta_K^{-1}(T_K - I)$. By the Theorem 6.5 in Chapter 1 of Ethier and Kurtz (2009), to obtain (7), it is sufficient to prove that for any $f \in J$,

$$\| A_K f - \Lambda^{(M,L)} f \| \to 0, \quad \text{as } K \to \infty, \tag{16}$$

where $\| \cdot \|$ represents the maximum norm on the function space J. To show this, observe that for any $(x,v) \in \mathbb{G}^{(M,L)}$, we have

$$(T_K f)(x,v) = \sum_{v' \in \mathbb{G}_v} P(\tilde{v}_{\delta_K}^{(L,K)} = v') g_{x,v}(v') = \left(\exp(\delta_K \Lambda) g_{x,v}\right)(v), \tag{17}$$

with $g_{x,v} := (g_{x,v}(v'))_{v' \in \mathbb{G}_v}$ is a vector, and

$$g_{x,v}(v') := \mathbf{E}_{x,v}\left[f\left(X_{\delta_K}^{(M,L,K)}, v'\right) | \tilde{v}_{\delta_K}^{(L,K)} = v'\right] = \left(\exp(\delta_K \mathcal{G}_v) f_{v'}\right)(x), \tag{18}$$

where vector $f_{v'} = (f(x,v'))_{x \in \mathbb{G}}$. Applying Taylor expansion to (18) yields

$$g_{x,v}(v') = f(x,v') + \delta_K(\mathcal{G}_v f_{v'})(x) + \frac{1}{2}\delta_K^2(\mathcal{G}_v^2 \exp(\tau \mathcal{G}_v) f_{v'})(x), \tag{19}$$

where $\tau \in (0, \delta_K)$. For (17), we also expand it and obtain

$$(T_K f)(x,v) = g_{x,v}(v) + \delta_K(\Lambda g_{x,v})(v) + \frac{1}{2}\delta_K^2(\Lambda^2 \exp(\tau' \Lambda) g_{x,v})(v), \tag{20}$$

where $\tau' \in (0, \delta_K)$. Substituting (19) into (20), we obtain

$$(T_K f)(x,v) = f(x,v) + \delta_K\left[(\mathcal{G}_v f_v))(x) + (\Lambda f_x)(v)\right] + o(\delta_K),$$

where vector $f_v = (f(x,v))_{x\in\mathbb{G}}$ and $f_x = (f(x,v))_{v\in\mathbb{G}_v}$. Thus, for any $f \in J$ and $(x,v) \in \mathbb{G}^{(M,L)}$, we have

$$(A_K f)(x,v) = \frac{(T_K f)(x,v) - f(x,v)}{\delta_K} = (\mathcal{G}_v f_v)(x) + (\Lambda f_x)(v) + o(1) \to \Lambda^{(M,L)} f(x,v) \text{ as } K \to \infty.$$

As J is the space of functions defined on a finite number of points, pointwise convergence is equivalent to convergence in the maximum norm. Therefore, we have shown (16) holds, hence the claim of the proposition. □

Acknowledgement We would like to add an acknowledgement section in the paper for funding support as follows: Lingfei Li was partially supported by the Hong Kong Research Grant Council General Research Fund Grant No. 14206020. Gongqiu Zhang was supported by National Natural Science Foundation of China Grant No. 12171408 and Shenzhen Fundamental Research Program Project JCYJ20190813165407555.

References

Albuquerque, R., Gaspar, R. M., & Michel, A. (2015). Investment analysis of autocallable contingent income securities. *Financial Analysts Journal, 71*(3), 61–83.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk, 3*, 5–40.

Artzner, P., Delbaen, F., Eber, J.-M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance, 9*(3), 203–228.

Bayer, C., Ben Hammouda, C., & Tempone, R. (2023). Numerical smoothing with hierarchical adaptive sparse grids and quasi-Monte Carlo methods for efficient option pricing. *Quantitative Finance, 23*(2), 209–227.

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review, 60*(2), 223–311.

Buehler, H. (2010). Volatility and dividends—Volatility modelling with cash dividends and simple credit risk. Available at SSRN 1141877.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance, 19*(8), 1271–1291.

Cai, N., Kou, S., & Song, Y. (2019). A unified framework for regime-switching models. Available at SSRN 3310365.

Cai, N., Song, Y., & Kou, S. (2015). A general framework for pricing Asian options under Markov processes. *Operations Research, 63*(3), 540–554.

Chan, R. H., Guo, Y. Z., Lee, S. T., & Li, X. (2019). *Financial mathematics*. Derivatives and Structured Products: Springer.

Cui, Z., Kirkby, J. L., & Nguyen, D. (2018). A general valuation framework for SABR and stochastic local volatility models. *SIAM Journal on Financial Mathematics, 9*(2), 520–563.

Davydov, D., & Linetsky, V. (2001). Pricing and hedging path-dependent options under the CEV process. *Management Science, 47*(7), 949–965.

Demeterfi, K., Derman, E., Kamal, M., & Zou, J. (1999). A guide to volatility and variance swaps. *The Journal of Derivatives, 6*(4), 9–32.
Deng, G., Dulaney, T., Husson, T., McCann, C., & Yan, M. (2015). Ex post structured-product returns: Index methodology and analysis. *The Journal of Investing, 24*(2), 45–58.
Deng, G., Mallett, J., & McCann, C. (2011). Modeling autocallable structured products. *Journal of Derivatives & Hedge Funds, 17*, 326–340.
Ding, K., Cui, Z., & Wang, Y. (2021). A Markov chain approximation scheme for option pricing under skew diffusions. *Quantitative Finance, 21*(3), 461–480.
Ethier, S. N., & Kurtz, T. G. (2009). *Markov processes: Characterization and convergence*. John Wiley & Sons.
Fries, C. P., & Joshi, M. S. (2011). Perturbation stable conditional analytic Monte Carlo pricing scheme for auto-callable products. *International Journal of Theoretical and Applied Finance, 14*(2), 197–219.
Guillaume, T. (2015). Analytical valuation of autocallable notes. *International Journal of Financial Engineering, 2*(2), 1550016.
Guillaume, T. (2015). Autocallable structured products. *The Journal of Derivatives, 22*(3), 73–94.
Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing smile risk. *The Best of Wilmott, 1*, 249–296.
Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies, 6*(2), 327–343.
Higham, N. J. (2005). The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications, 26*(4), 1179–1193.
Kim, K.-K., & Lim, D.-Y. (2019). A recursive method for static replication of autocallable structured products. *Quantitative Finance, 19*(4), 647–661.
Kirkby, J. L. (2023). Hybrid equity swap, cap, and floor pricing under stochastic interest by Markov chain approximation. *European Journal of Operational Research, 305*(2), 961–978.
Kleywegt, A. J., Shapiro, A., & Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization, 12*(2), 479–502.
Koster, F., & Rehmet, A. (2018). Monte Carlo payoff smoothing for pricing autocallable instruments. *Journal of Computational Finance 21*(4), 59–77.
Kou, S. G. (2002). A jump-diffusion model for option pricing. *Management Science, 48*(8), 1086–1101.
Lee, M., & Hong, J. (2021). Semi closed-form pricing autocallable ELS using Brownian bridge. *Communications for Statistical Applications and Methods, 28*(3), 251–265.
Li, L., & Linetsky, V. (2015). Discretely monitored first passage problems and barrier options: An eigenfunction expansion approach. *Finance and Stochastics, 19*(4), 941–977.
Li, L., Zeng, P., & Zhang, G. (2024). Speed and duration of drawdown under general Markov models. *Quantitative, 3*, 367–386.
Li, L., & Zhang, G. (2016). Option pricing in some non-Lévy jump models. *SIAM Journal on Scientific Computing, 38*(4), B539–B569.
Li, L., & Zhang, G. (2018). Error analysis of finite difference and Markov chain approximations for option pricing. *Mathematical Finance, 28*(3), 877–919.
Madan, D. B., Carr, P. P., & Chang, E. C. (1998). The variance gamma process and option pricing. *Review of Finance, 2*(1), 79–105.
McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: Concepts, techniques, and tools*. Princeton University Press.
Meier, C., Li, L., & Zhang, G. (2021). Markov chain approximation of one-dimensional sticky diffusions. *Advances in Applied Probability, 53*(2), 335–369.
Meier, C., Li, L., & Zhang, G. (2023). Simulation of multidimensional diffusions with sticky boundaries via Markov chain approximation. *European Journal of Operational Research, 305*(3), 1292–1308.
Mijatović, A., & Pistorius, M. (2013). Continuously monitored barrier options under Markov processes. *Mathematical Finance, 23*(1), 1–38.
Paletta, T., & Tunaru, R. (2022). A Bayesian view on autocallable pricing and risk management. *The Journal of Derivatives, 29*(5), 40–59.
Yang, N., Chen, N., & Wan, X. (2019). A new delta expansion for multivariate diffusions via the Itô-Taylor expansion. *Journal of Econometrics, 209*(2), 256–288.
Zhang, G., & Li, L. (2019). Analysis of Markov chain approximation for option pricing and hedging: Grid design and convergence behavior. *Operations Research, 67*(2), 407–427.
Zhang, G., & Li, L. (2021). A general approach for lookback option pricing under Markov models. arXiv Preprint retrieved from arXiv:2112.00439.

Zhang, G., & Li, L. (2022). Analysis of Markov chain approximation for diffusion models with nonsmooth coefficients. *SIAM Journal on Financial Mathematics, 13*(3), 1144–1190.

Zhang, G., & Li, L. (2023a). A general approach for Parisian stopping times under Markov processes. *Finance and Stochastics, 27*(3), 769–829.

Zhang, G., & Li, L. (2023b). A general method for analysis and valuation of drawdown risk under Markov models. *Journal of Economic Dynamics and Control, 152*, 104669.

Zhang, X., Li, L., & Zhang, G. (2021). Pricing American drawdown options under Markov models. *European Journal of Operational Research, 293*(3), 1188–1205.