Original Articles

Stratified regression-based variance reduction approach for weak approximation schemes

D. Belomestny\textsuperscript{a,b,*}, S. Häfner\textsuperscript{c}, M. Urusov\textsuperscript{a}

\textsuperscript{a}Duisburg–Essen University, Essen, Germany
\textsuperscript{b}National Research University Higher School of Economics, Moscow, Russia
\textsuperscript{c}Pricewaterhouse Coopers GmbH, Frankfurt, Germany

Received 22 January 2016; received in revised form 19 February 2017; accepted 13 May 2017
Available online 25 May 2017

Abstract

In this paper we suggest a modification of the regression-based variance reduction approach recently proposed in Belomestny et al. [1]. This modification is based on the stratification technique and allows for a further significant variance reduction. The performance of the proposed approach is illustrated by several numerical examples.

\textcopyright 2017 International Association for Mathematics and Computers in Simulation (IMACS). Published by Elsevier B.V. All rights reserved.

Keywords: Control variates; Stratification; Monte Carlo methods; Weak schemes; Regression

1. Introduction

Let $T > 0$ be a fixed time horizon. Consider a $d$-dimensional diffusion process $(X_t)_{t \in [0,T]}$ defined by the Itô stochastic differential equation

$$dX_t = \mu(X_t)\,dt + \sigma(X_t)\,dW_t, \quad X_0 = x_0 \in \mathbb{R}^d,$$

for Lipschitz continuous functions $\mu : \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \to \mathbb{R}^{d \times m}$, where $(W_t)_{t \in [0,T]}$ is a standard $m$-dimensional Brownian motion. Suppose we want to compute the expectation

$$u(t,x) := \mathbb{E}[f(X^t_{T,x})],$$

where $X^t_{T,x}$ denotes the solution to (1) started at time $t$ in point $x$. The standard Monte Carlo (SMC) approach for computing $u(0,x)$ at a fixed point $x \in \mathbb{R}^d$ basically consists of three steps. First, an approximation $\bar{X}_T$ for $X^0_{T,x}$ is constructed via a time discretisation in Eq. (1) (we refer to [4] for a nice overview of various discretisation schemes).

\textsuperscript{*}Corresponding author at: Duisburg–Essen University, Essen, Germany.

E-mail addresses: denis.belomestny@uni-due.de (D. Belomestny), stefan.haefner@de.pwc.com (S. Häfner), mikhail.urusov@uni-due.de (M. Urusov).

http://dx.doi.org/10.1016/j.matcom.2017.05.003
0378-4754/\textcopyright 2017 International Association for Mathematics and Computers in Simulation (IMACS). Published by Elsevier B.V. All rights reserved.
Next, $N_0$ independent copies of the approximation $\overline{X}_T$ are generated, and, finally, a Monte Carlo estimate $V_{N_0}$ is defined as the average of the values of $f$ at simulated points:

$$V_{N_0} := \frac{1}{N_0} \sum_{i=1}^{N_0} f\left(\overline{X}_T^{(i)}\right).$$

(3)

In the computation of $u(0, x) = \mathbb{E}[f(\overline{X}_T^{0,x})]$ by the SMC approach there are two types of error inherent: the discretisation error $\mathbb{E}[f(\overline{X}_T^{0,x})] - \mathbb{E}[f(\overline{X}_T)]$ and the Monte Carlo (statistical) error, which result from the substitution of $\mathbb{E}[f(\overline{X}_T)]$ with the sample average $V_{N_0}$. The aim of variance reduction methods is to reduce the statistical error. For example, in the so-called control variate variance reduction approach one looks for a random variable $\xi$ with $\mathbb{E}\xi = 0$, which can be simulated, such that the variation of the difference $f(\overline{X}_T) - \xi$ is minimised, that is,

$$\text{Var}[f(\overline{X}_T) - \xi] \to \min \text{ under } \mathbb{E}\xi = 0.$$

Then one uses the sample average

$$V_{N_0}^{\text{CV}} := \frac{1}{N_0} \sum_{i=1}^{N_0} \left[f\left(\overline{X}_T^{(i)}\right) - \xi^{(i)}\right]$$

(4)

instead of (3) to approximate $\mathbb{E}[f(\overline{X}_T)]$. The use of control variates for computing expectations of functionals of diffusion processes via Monte Carlo was initiated by Newton [7] and further developed in Milstein and Tretyakov [6]. Heath and Platen [3] use the integral representation to construct unbiased variance-reduced estimators. In Belomestny et al. [1] a novel regression-based approach for the construction of control variates, which reduces the variance of the approximated functional $f(\overline{X}_T)$ was proposed. As shown in [1], the “Monte Carlo approach with the Regression-based Control Variate” (abbreviated below as “RCV approach”) as well as its enhancement, called “recursive RCV (RRCV) approach”, is able to achieve a higher order convergence of the resulting variance to zero, which in turn leads to a significant complexity reduction as compared to the SMC algorithm. The RCV approaches become especially simple in the case of the so-called weak approximation schemes, i.e., the schemes, where simple random variables are used in place of Brownian increments, and which became quite popular in recent years. In this paper we further enhance the performance of the RRCV algorithm by combining it with stratification. The idea of the resulting stratified RCV (SRCV) algorithm is based on partitioning of the state space into a collection of sets $\mathcal{A}_1, \ldots, \mathcal{A}_p$ and then performing conditional regressions separately on each set. It turns out that by choosing $\mathcal{A}_1, \ldots, \mathcal{A}_p$ to be the level sets of the discrete-valued random variables used in the weak approximation scheme, we can achieve a further variance reduction effect as compared to the original approach in [1]. The paper is organised as follows. In Section 2, the SRCV algorithm is introduced and compared with the RCV and RRCV ones. The complexity analysis of the SRCV algorithm is conducted in Section 3. Section 4 is devoted to the simulation study. Necessary proofs are collected in Section 5.

2. SRCV approach and its differences with RCV and RRCV ones

In what follows $J \in \mathbb{N}$ denotes the time discretisation parameter. We set $\Delta := T/J$ and consider discretisation schemes denoted by $(X_{\Delta,j\Delta})_{j=0,\ldots,J}$, which are defined on the grid $\{j\Delta : j = 0, \ldots, J\}$. In Sections 2.1 and 2.2 we consider weak schemes of order 1. In this setting we recall the RCV and RRCV algorithms, introduce the SRCV algorithm and explain how it compares to the RCV and RRCV ones. In Section 2.3 we briefly discuss the case of weak schemes of order 2.

2.1. RCV algorithm for first order schemes

Let us consider a weak scheme of order 1, where $d$-dimensional approximations $X_{\Delta,j\Delta}$, $j = 0, \ldots, J$, satisfy $X_{\Delta,0} = x_0$ and

$$X_{\Delta,j\Delta} = \Phi_{\Delta}(X_{\Delta,(j-1)\Delta}, \xi_j), \quad j = 1, \ldots, J,$$

(5)

for some functions $\Phi_{\Delta} : \mathbb{R}^{d+m} \to \mathbb{R}^d$, with $\xi_j = (\xi_j^1, \ldots, \xi_j^m)$, $j = 1, \ldots, J$, being $m$-dimensional i.i.d. random vectors with i.i.d. coordinates satisfying

$$\mathbb{P}(\xi_j^k = \pm 1) = \frac{1}{2}, \quad k = 1, \ldots, m.$$
An important particular case is the weak Euler scheme (also called the simplified weak Euler scheme in [4, Section 14.1]), which is given by
\[
\Phi_\Delta(x, y) = x + \mu(x) \Delta + \sigma(x) y \sqrt{\Delta}.
\] (6)
The RCV approach of [1] essentially relies on the following representation, which has some resemblance to discrete the Clark–Ocone formula (see e.g. [8]).

**Theorem 1.** It holds
\[
f(X_{\Delta,T}) = \mathbb{E}f(X_{\Delta,T}) + \sum_{j=1}^{J} \sum_{k \in \{0,1\}^{m} \setminus \{0\}} a_{j,k}(X_{\Delta,(j-1)\Delta}) \prod_{i=1}^{m} (\xi_{j}^{i})^{k_i},
\] (7)
where \( k = (k_1, \ldots, k_m) \) and \( 0 = (0, \ldots, 0) \) (in the second summation). Moreover, the coefficients \( a_{j,k} : \mathbb{R}^d \to \mathbb{R} \) can be computed by the formula
\[
a_{j,k}(x) = \mathbb{E} \left[ f(X_{\Delta,T}) \prod_{i=1}^{m} (\xi_{j}^{i})^{k_i} \middle| X_{\Delta,(j-1)\Delta} = x \right]
\] (8)
for all \( j \) and \( k \) as in (7).

**Theorem 1** is an equivalent reformulation of Theorem 3.1 in [1].

**Discussion.** Under appropriate conditions on the functions \( f, \mu \) and \( \sigma \) (see e.g. Theorem 2.1 in [5]) the discretisation error \( \mathbb{E}[f(X_T)] - \mathbb{E}[f(X_{\Delta,T})] \) for the scheme (6) is of order \( \Delta \) (first order scheme). Furthermore, by **Theorem 1**, with
\[
M_{\Delta,T}^{(1)} := \sum_{j=1}^{J} \sum_{k \in \{0,1\}^{m} \setminus \{0\}} a_{j,k}(X_{\Delta,(j-1)\Delta}) \prod_{i=1}^{m} (\xi_{j}^{i})^{k_i},
\] (9)
we have \( \mathbb{E}[M_{\Delta,T}^{(1)}] = 0 \) and \( \text{Var}[f(X_{\Delta,T}) - M_{\Delta,T}^{(1)}] = 0 \), that is, \( M_{\Delta,T}^{(1)} \) is a perfect control variate reducing the statistical error down to zero. However, in practice we cannot simulate \( M_{\Delta,T}^{(1)} \) because the coefficients \( a_{j,k} \) are generally unknown. In the RCV algorithm, we construct a practically implementable control variate \( \tilde{M}_{\Delta,T}^{(1)} \) of the form (9) with regression-based estimates \( \tilde{a}_{j,k} : \mathbb{R}^d \to \mathbb{R} \) of the functions \( a_{j,k} \). It is worth noting that the sample average of \( f(X_{\Delta,T}) - \tilde{M}_{\Delta,T}^{(1)} \) (cf. (4)) should be computed on the paths independent of those used to construct \( \tilde{a}_{j,k} \). This ensures that \( \mathbb{E}[\tilde{M}_{\Delta,T}^{(1)}] = 0 \), and, thus, that \( \tilde{M}_{\Delta,T}^{(1)} \) is a valid control variate (because of the martingale transform structure in (9)).

**2.2. RRCV and SRCV algorithms for first order schemes**

The coefficients given by (8) can be approximated using various regression algorithms. From a computational point of view it is however advantageous to look for other representations, which only involve regressions over one time step (notice that in (8) the regression is performed over \( J - j + 1 \) time steps). To this end, for \( j \in \{1, \ldots, J\} \), we introduce the functions
\[
q_j(x) := \mathbb{E}[f(X_{\Delta,T})|X_{\Delta,j\Delta} = x].
\] (10)
The next result is Proposition 3.3 of [1].

**Proposition 1.** We have \( q_j \equiv f \) and, for each \( j \in \{2, \ldots, J\} \),
\[
q_{j-1}(x) = \mathbb{E}[q_j(X_{\Delta,j\Delta})|X_{\Delta,(j-1)\Delta} = x] = \frac{1}{2^m} \sum_{y \in \{-1,1\}^m} q_j(\Phi_\Delta(x, y)).
\] (11)
Moreover, for all \( j \in \{1, \ldots, J\} \) and \( k = (k_i) \in \{0,1\}^{m} \setminus \{0\} \) (with \( 0 \equiv (0, \ldots, 0) \)), the functions \( a_{j,k}(x) \) in (8) can be expressed in terms of the functions \( q_j(x) \) as follows:
\[
a_{j,k}(x) = \frac{1}{2^m} \sum_{y=(y_1,\ldots,y_m) \in \{-1,1\}^m} \prod_{i=1}^{m} y_{i}^{k_i} q_j(\Phi_\Delta(x, y)).
\] (12)
The first equality in (11) shows that we can recursively approximate the functions \( q_j(x) \) via regressions over one time step only. This gives the RRCV algorithm of [1]: first compute regression-based approximations \( \tilde{q}_j(x) \) of the functions \( q_j(x) \) (via regressions over one time step based on the first equality in (11)), then obtain approximations \( \tilde{a}_{j,k}(x) \) of the functions \( a_{j,k}(x) \) via (12) with \( q_j \) being replaced by \( \tilde{q}_j \), and, finally, construct the control variate \( \tilde{M}^{(1)}_{\Delta,T} \) using (9) with \( a_{j,k}(x) \) being replaced by \( \tilde{a}_{j,k}(x) \).

To introduce the SRCV algorithm, we first define functions \( h_{j,y} \), for all \( j \in \{1, \ldots, J\} \) and \( y \in \{-1, 1\}^m \), by the formula

\[
\begin{align*}
\tilde{h}_{j,y}(x) := q_j(\Phi_\Delta(x, y)) & = \mathbb{E}[q_j(X_{\Delta,j}\Delta)|X_{\Delta,(j-1)\Delta} = x, \xi_j = y] \\
\text{for all } j \in \{1, \ldots, J\} \text{ and } y \in \{-1, 1\}^m,
\end{align*}
\]  

(13)

(the second equality is straightforward) and observe that the knowledge of these functions for some \( j \) and \( y \) provides us with the functions \( q_{j-1} \) and \( a_{j,k} \), \( k \in \{0, 1\}^m \setminus \{0\} \), via the second equality in (11) and via (12). Inspired by this observation together with the second equality in (13), we arrive at the idea of the stratified regression: approximate each function \( \tilde{h}_{j,y}(x) \) via its projection on a given set of basis functions \( \psi_1(x), \ldots, \psi_K(x) \). In detail, the SRCV algorithm consists of two phases: “training phase” and “testing phase”.

Training phase of the SRCV algorithm: First, simulate a sufficient number \( N \) of (independent) “training paths” of the discretised diffusion. Let us denote the set of these \( N \) paths by \( D_N^r \) (cf. (4)), where

\[
\begin{align*}
\tilde{h}_{j,y}(x) & = \mathbb{E}[q_j(X_{\Delta,j}\Delta)|X_{\Delta,(j-1)\Delta} = x, \xi_j = y] \\
\text{for all } j \in \{1, \ldots, J\} \text{ and } y \in \{-1, 1\}^m,
\end{align*}
\]  

(14)

(the superscript “tr” comes from “training”). Next, proceed as follows.

Step 1. Set \( j = J \), \( \tilde{a}_j = f \). Compute the values \( \tilde{h}_{j,y}(X_{\Delta,j}\Delta) \) on all training paths \( i = 1, \ldots, N \). 

Step 2. For all \( y \in \{-1, 1\}^m \), construct regression-based approximations \( \tilde{h}_{j,y} \) of the functions \( \tilde{h}_{j,y} \) (via regressions over one time step based on the second equality in (13) with \( q_j \) being replaced by \( \tilde{q}_j \)). In fact, only training paths with \( \xi_j = y \) are used to construct \( \tilde{h}_{j,y} \).

Step 3. Using the approximations \( \tilde{h}_{j,y} \) for all \( y \in \{-1, 1\}^m \), via (12) compute the coefficients \( \alpha_1, \ldots, \alpha_K \) in the representations \( \alpha_1 \psi_1 + \cdots + \alpha_K \psi_K \) of the approximations \( \tilde{a}_{j,k} \). Furthermore, again using \( \tilde{h}_{j,y} \) for all \( y \in \{-1, 1\}^m \), compute the values \( \tilde{a}_{j-1}(X_{\Delta,(j-1)\Delta}) \) on all training paths \( i = 1, \ldots, N \) via the second equality in (11).

Step 4. If \( j > 1 \), set \( j = j - 1 \) and go to step 2.

Thus, after the training phase is completed, we have the approximations \( \tilde{a}_{j,k}(x) \) of \( a_{j,k}(x) \) for all \( j \in \{1, \ldots, J\} \) and \( k \in \{0, 1\}^m \setminus \{0\} \). Let us emphasise that, in fact,

\[
\begin{align*}
\tilde{a}_{j,k}(x) & = \tilde{a}_{j,k}(x, D_N^r),
\end{align*}
\]  

(15)

that is, our approximations are random and depend on the simulated training paths.

Testing phase of the SRCV algorithm: Simulate \( N_0 \) “testing paths” \( (X_{\Delta,j\Delta})_{i=0,\ldots,J}, i = 1, \ldots, N_0, \) that are independent from each other and from the training paths and construct the Monte Carlo estimate

\[
\begin{align*}
\frac{1}{N_0} \sum_{i=1}^{N_0} f(X_{\Delta,T}) - \tilde{M}^{(1),(i)}_{\Delta,T}
\end{align*}
\]  

(16)

(cf. (4)), where \( \tilde{M}^{(1),(i)}_{\Delta,T} \) is given by

\[
\begin{align*}
\tilde{M}^{(1),(i)}_{\Delta,T} := \sum_{j=1}^{J} \sum_{k \in \{0,1\}^m \setminus \{0\} \text{ that } a_{j,k}(x) \text{ is approximated}} \tilde{a}_{j,k}(x) (X_{\Delta,(j-1)\Delta}) \cdot D_N^r \prod_{l=1}^{m} (\xi_j)^{k_l}
\end{align*}
\]  

(17)

(cf. (9)).

Discussion. Let us briefly discuss the main differences between the RRCV and SRCV algorithms. In the training phase of the RRCV algorithm the functions \( q_j, j \in \{1, \ldots, J\} \), are approximated recursively via regressions using the first equality in (11) (the second equality in (11) is not used at all), and the approximations are linear combinations of \( K \) basis functions \( \psi_1, \ldots, \psi_K \). This allows to get the control variate in the testing phase via the formula like (17) with the coefficients \( \tilde{a}_{j,k} \) constructed on the testing paths via (12) with approximated in the training phase functions \( q_j \). On the contrary, in the training phase of the SRCV algorithm regressions are based on the second equality in (13), and we get approximations for all functions \( h_{j,y} \) (\( \equiv q_j(\Phi_\Delta(., y)) \), \( j \in \{1, \ldots, J\} \), \( y \in \{-1, 1\}^m \), where the approximations \( \tilde{h}_{j,y} \)
are again linear combinations of $K$ basis functions $\psi_1, \ldots, \psi_K$ (notice that what we now need from (11) is the second equality but not the first one). Having the approximations $\hat{h}_{j,y}$, we get the approximations of the functions $\tilde{a}_{j,k}$ via (12) as linear combinations of $\psi_1, \ldots, \psi_K$ already in the training phase, while the testing phase is completely described by (16)–(17). Let us compare the computational costs of the RRCV and SRCV algorithms. For the sake of simplicity we restrict our attention to the case of “large” parameters $J$, $K$, $N$ and $N_0$ as well as at the “big” constant $c_m := 2^m$ ignoring other constants such as e.g. $d$ or $m$. As for the RRCV algorithm, $J$ regressions with $N$ training paths and $K$ basis functions result in the cost of order $JK^2N$, while the cost of the testing phase is of order $JKc_mN_0$, which results in the overall cost of order $JK \max \{KN, c_mN_0\}$. As for the SRCV algorithm, we perform $Jc_m$ regressions with $K$ basis functions in the training phase, but have in average $N\mathbb{P}(\xi_j = y) (\equiv N/c_m)$, $y \in \{-1, 1\}^m$, training paths in each regression, which again results in the cost of order $JK^2N$, while in the testing phase we now have the cost of order $JK(c_m - 1)N_0$. This gives us the overall cost of order $JK \max \{KN, (c_m - 1)N_0\}$, which is the same order as for the RRCV algorithm. Finally, regarding the quality of the regressions in the RRCV and SRCV approaches, it is to expect that the regressions in the SRCV algorithm, which are based on the second equality in (13), achieve better approximations than the regressions in the RCV algorithm, provided there are enough training paths and the basis functions are chosen properly, because we have

$$\text{Var}[q_j(X_{\Delta,j}\Delta)|X_{\Delta,j-1}\Delta = x, \xi_j = y] = \text{Var}[q_j(\Phi_{\Delta}(x, y))] = 0.$$  

(18)

The latter property implies the absence of the statistical error while approximating $h_{j,y}$. This is well illustrated by the first three plots in Fig. 1 (the plots are performed for the example of Section 4.1).

### 2.3. RCV, RRCV and SRCV algorithms for second order schemes

Let us define the index set

$$\mathcal{I} = \{(k, l) \in \{1, \ldots, m\}^2 : k < l\}$$

and consider a weak scheme of order 2, where $d$-dimensional approximations $X_{\Delta,j}\Delta$, $j = 0, \ldots, J$, satisfy $X_{\Delta,0} = x_0$ and

$$X_{\Delta,j}\Delta = \Phi_{\Delta}(X_{\Delta,(j-1)\Delta}, \xi_j, V_j), \quad j = 1, \ldots, J,$$

for some functions $\Phi_{\Delta} : \mathbb{R}^{d+m+m(m-1)/2} \rightarrow \mathbb{R}^d$. Here,

- $\xi_j = (\xi_j^k)_{k=1}^m$, $j = 1, \ldots, J$, are $m$-dimensional random vectors with i.i.d. coordinates satisfying
  $$\mathbb{P}(\xi_j^k = \pm \sqrt{3}) = \frac{1}{6}, \quad \mathbb{P}(\xi_j^k = 0) = \frac{2}{3};$$

- $V_j = (V_j^l)_{(k,l) \in \mathcal{I}}$, $j = 1, \ldots, J$, are $m(m-1)/2$-dimensional random vectors with i.i.d. coordinates satisfying
  $$\mathbb{P}(V_j^l = \pm 1) = \frac{1}{2},$$

- the pairs $(\xi_j, V_j)$, $j = 1, \ldots, J$, are independent,
- for each $j$, the random vectors $\xi_j$ and $V_j$ are independent.

---

1. We need to have $J \rightarrow \infty$, $K \rightarrow \infty$, $N \rightarrow \infty$, $N_0 \rightarrow \infty$ in order to make both the discretisation and the statistical error tend to zero (see Section 3 for more details).

2. In contrast to $J$, $K$, $N$ and $N_0$, the value $c_m := 2^m$ is fixed, but can be relatively big (compared to other involved constants such as e.g. $d$ or $m$). Notice that $c_m$ is the number of scenarios that the random variables $\xi_j$ can take, and it comes into play via formulae like (17) ($J(c_m - 1)$ summands) or (12) ($c_m$ summands).

3. Naive implementation of the testing phase in the RRCV algorithm via (12) and (9) gives the cost order $JKc_m(c_m - 1)N_0$. To get $JKc_mN_0$, one should implement (12) on the testing paths in two steps: first, for all $i \in \{1, \ldots, N_0\}$, $j \in \{1, \ldots, J\}$ and $y \in \{-1, 1\}^m$, compute the values $\tilde{a}_{ij}(X_{\Delta,(j-1)\Delta}^i, y)$ (the cost is $N_0Jc_KK$); then, using these values, for all $i \in \{1, \ldots, N_0\}$, $j \in \{1, \ldots, J\}$ and $k \in \{0, 1\}^m \setminus \{0\}$, compute $\tilde{a}_{i,k}(X_{\Delta,(j-1)\Delta}^i)$ via (12) (the cost is $N_0J(c_m - 1)c_m$). In this way, the maximal cost order is $JKc_mN_0$. 
An important example of such a scheme is the simplified order 2 weak Taylor scheme in Section 14.2 of [4], which has the discretisation error $\mathbb{E}[f(X_T)] - \mathbb{E}[f(X_{\Delta,T})]$ of order $\Delta^2$ under appropriate conditions on $f$, $\mu$ and $\sigma$ (also see Theorem 2.1 in [5]). Let us introduce the notation

$$\mathcal{U} = \{ (o,r) \in \{0,1,2\}^m \times \{0,1\}^J : o_i \neq 0 \text{ for some } i \text{ or } r_{kl} \neq 0 \text{ for some } k, l \},$$

where $o_i, i = 1, \ldots, m$ (resp. $r_{kl}, (k,l) \in \mathcal{F}$), denote the coordinates of $o$ (resp. $r$). The following result is an equivalent reformulation of Theorem 3.5 in [1].

**Theorem 2.** The following representation holds

$$f(X_{\Delta,T}) = \mathbb{E}[f(X_{\Delta,T})] + \sum_{j=1}^{J} \sum_{(o,r) \in \mathcal{U}} a_{j,o,r}(X_{\Delta,(j-1)\Delta}) \prod_{i=1}^{m} H_{o_i}(\xi^i_j) \prod_{(k,l) \in \mathcal{F}} (V_{kl}^j)^{r_{kl}},$$

where $H_0(x) := 1$, $H_1(x) := x$, $H_2(x) := \frac{x^2 - 1}{\sqrt{2}}$, and the coefficients $a_{j,o,r} : \mathbb{R}^d \rightarrow \mathbb{R}$ are given by the formula

$$a_{j,o,r}(x) = \mathbb{E} \left[ f(X_{\Delta,T}) \prod_{i=1}^{m} H_{o_i}(\xi^i_j) \prod_{(k,l) \in \mathcal{F}} (V_{kl}^j)^{r_{kl}} \left| X_{\Delta,(j-1)\Delta} = x \right. \right]$$

for all $j \in \{1, \ldots, J\}$ and $(o,r) \in \mathcal{U}$. 

---

**Fig. 1.** 1. Top left: first regression task for RCV (Section 4.1), 2. top right: first regression task for RRCV (Section 4.1), 3. centre left: first regression task for SRCV (Section 4.1), 4. centre right: ECDF of the log-scaled sample for SRCV and SMC (Section 4.1), 5. bottom left: ECDF of the log-scaled sample for SRCV and SMC (Section 4.2), 6. bottom right: ECDF of the log-scaled sample for SRCV and SMC (Section 4.3).
Thus, with
\[
M^{(2)}_{\Delta, T} := \sum_{j=1}^{J} \sum_{(o, r) \in \mathcal{U}} a_{j, o, r}(X_{\Delta, (j-1)\Delta}) \prod_{i=1}^{m} H_{o_i}(\xi_j) \prod_{(k, l) \in \mathcal{S}} (V_{kl}^{(j)})^{r_{kl}},
\]  
(22)

we have \( \mathbb{E} [M^{(2)}_{\Delta, T}] = 0 \) and \( \text{Var} [f(X_{\Delta, T}) - M^{(2)}_{\Delta, T}] = 0 \) in the case of second order schemes. The RCV approach for second order schemes relies on Theorem 2 in the same way as the one for first order schemes relies on Theorem 1.

We now introduce the functions \( q_j(x) \), \( j \in \{ 1, \ldots, J \} \), by formula (10) also in the case of second order schemes and, for all \( x \in (-\sqrt{3}, 0, \sqrt{3})^m \), set
\[
p_m(y) := \frac{4\sum_{i=1}^{m} I(y_i = 0)}{6^m 2^{m(m-1)/2}}.
\]  
(23)

Notice that \( p_m(y) = \mathbb{P}(\xi_j = y, V_j = z) \) for all \( z \in \{-1, 1\}^\mathcal{S} \). The next result is Proposition 3.7 of [1].

**Proposition 2.** We have \( q_j \equiv f \) and, for each \( j \in \{ 2, \ldots, J \} \),
\[
q_{j-1}(x) = \mathbb{E}[q_j(X_{\Delta, j\Delta})|X_{\Delta, (j-1)\Delta} = x] = \sum_{y \in (-\sqrt{3}, 0, \sqrt{3})^m} \sum_{z \in \{-1, 1\}^\mathcal{S}} p_m(y) q_j(\Phi_{\Delta}(x, y, z)).
\]  
(24)

Moreover, for all \( j \in \{ 1, \ldots, J \} \) and \( (o, r) \in \mathcal{U} \), the functions \( a_{j, o, r}(x) \) of (21) can be expressed in terms of the functions \( q_j(x) \) as
\[
a_{j, o, r}(x) = \sum_{y \in (-\sqrt{3}, 0, \sqrt{3})^m} \sum_{z \in \{-1, 1\}^\mathcal{S}} \left[ p_m(y) \prod_{i=1}^{m} H_{o_i}(y_i) \prod_{(k, l) \in \mathcal{S}} z_{kl}^{(j)} \right] q_j(\Phi_{\Delta}(x, y, z)),
\]  
(25)

where \( o_i \) and \( y_i \), \( i = 1, \ldots, m \), denote the coordinates of \( o \) and \( y \), while \( r_{kl} \) and \( z_{kl} \), \( (k, l) \in \mathcal{S} \), are the coordinates of \( r \) and \( z \).

Similar to (13), we define functions \( h_{j, y, z} \), for all \( j \in \{ 1, \ldots, J \} \), \( y \in (-\sqrt{3}, 0, \sqrt{3})^m \) and \( z \in \{-1, 1\}^\mathcal{S} \), by the formula
\[
h_{j, y, z}(x) := q_j(\Phi_{\Delta}(x, y, z)) = \mathbb{E}[q_j(X_{\Delta, j\Delta})|X_{\Delta, (j-1)\Delta} = x, \xi_j = y, V_j = z].
\]  
(26)

The RRCV and SRCV algorithms for second order schemes now rely on Proposition 2 and on (26) in the same way as the ones for first order schemes rely on Proposition 1 and on (13). The whole discussion in the end of Section 2.2, and, in particular, the formula \( J K \max\{KN, (c_m-1)N_0\} \) for the overall cost order of the SRCV algorithm, apply also in the case of second order schemes, where we only need to change the value of \( c_m \); here \( c_m := 3^m 2^{m(m-1)/2} \).

### 3. Complexity analysis

In this section we extend the complexity analysis presented in [1] to the case of the stratified regression algorithm. Below we only sketch the main results for the second order schemes. We make the following assumptions.

(A1) All functions \( h_{j, y, z}(x) \) of (26) are uniformly bounded, i.e. there is a constant \( A > 0 \) such that \( \sup_{x \in \mathbb{R}^d} |h_{j, y, z}(x)| \leq A < \infty \).

(A2) The functions \( h_{j, y, z}(x) \) can be well approximated by the functions from \( \mathcal{V}_K := \text{span}\{\psi_1, \ldots, \psi_K\} \), in the sense that there are constants \( \kappa > 0 \) and \( C_\kappa > 0 \) such that
\[
\inf_{g \in \mathcal{V}_K} \int_{\mathbb{R}^d} (h_{j, y, z}(x) - g(x))^2 \mathbb{P}_{\Delta, j-1}(dx) \leq \frac{C_\kappa}{K^\kappa},
\]
where \( \mathbb{P}_{\Delta, j-1} \) denotes the distribution of \( X_{\Delta, (j-1)\Delta} \).

**Remark 1.** A sufficient condition for (A1) is boundedness of \( f \). As for (A2), this is a natural condition to be satisfied for good choices of \( \mathcal{V}_K \). For instance, under appropriate assumptions, in the case of **piecewise polynomial regression** as described in [1], (A2) is satisfied with \( \kappa = \frac{2(p+1)}{2d(p+1)+dv} \), where the parameters \( p \) and \( v \) are explained in [1].
In Lemma 1 we present an $L^2$-upper bound for the estimation error on step 2 of the training phase of the SRCV algorithm (see page 6). To this end, we need to describe more precisely, how exactly the regression-based approximations $\hat{h}_{j,y,z}$ are constructed:

(A3) Let functions $\hat{h}_{j,y,z}(x)$ be obtained by linear regression (based on the second equality in (26)) onto the set of basis functions $\{\psi_1, \ldots, \psi_K\}$, while the approximations $\hat{h}_{j,y,z}(x)$ on step 2 of the training phase of the SRCV algorithm be the truncated estimates, which are defined as follows:

$$\hat{h}_{j,y,z}(x) := T_A \hat{h}_{j,y,z}(x) := \begin{cases} \hat{h}_{j,y,z}(x) & \text{if } |\hat{h}_{j,y,z}(x)| \leq A, \\ A \text{ sgn} \hat{h}_{j,y,z}(x) & \text{otherwise} \end{cases}$$

(A is the constant from (A1)).

**Lemma 1.** Under (A1)–(A3), we have

$$\mathbb{E}[|\hat{h}_{j,y,z} - h_{j,y,z}|^2_{L^2(\mathbb{P}_{\Delta,j-1})}] \leq \tilde{c} A^2 (\log N + 1) \frac{K}{N p_m(y)} + \frac{8 C}{K^*},$$

(27)

where $\tilde{c}$ is a universal constant and $p_m(y)$ is given in (23).

It is necessary to explain once in detail how to understand the left-hand side of (27). The functions $\hat{h}_{j,y,z}(x)$ (see (A3)) are linear combinations $\alpha_1 \psi_1(x) + \cdots + \alpha_K \psi_K(x)$ of the basis functions, where the coefficients $\alpha_i$ are random in that they depend on the simulated training paths. That is, we have, in fact, $\hat{h}_{j,y,z}(x) = \hat{h}_{j,y,z}(x, D_N^T)$ and, consequently, $h_{j,y,z}(x) = \hat{h}_{j,y,z}(x, D_N^T)$ (cf. (15)). Thus, the expectation on the left-hand side of (27) means averaging over the randomness in $D_N^T$.

The next step is to provide an upper bound for the regression-based estimates of the coefficients $a_{j,o,r}$, which are constructed on step 3 of the training phase of the SRCV algorithm.

**Lemma 2.** Under (A1)–(A3), we have

$$\mathbb{E}[|\hat{a}_{j,o,r} - a_{j,o,r}|^2_{L^2(\mathbb{P}_{\Delta,j-1})}] \leq c_m \tilde{c} A^2 (\log N + 1) \frac{K}{N} + \frac{8 C}{K^*}$$

(28)

where $C_{m,o} := \sum_{y \in \{-3, 0, 3\}} c_m 2^{m+1} \left[ p_m(y) \sum_{i=1}^m H_{o,i}(y_i) \right]^2$.

Let $(X_{\Delta,j})_{j=0,\ldots,T}$ be a testing path, which is independent of the training paths $D_N^T$. We now define

$$\tilde{M}^{(2),T}_{\Delta,T} := \sum_{j=1}^T \sum_{(o,r) \in \mathcal{G}} \tilde{a}_{j,o,r}(X_{\Delta,(j-1)\Delta}, D_N^T) \sum_{i=1}^m H_{o,i}(\xi_j^i) \prod_{(k,l) \in \mathcal{G}} (V_j^{kl})^{y_j^k}$$

(29)

(cf. (22)) and bound the variance $\text{Var}[f(X_{\Delta,T}) - \tilde{M}^{(2),T}_{\Delta,T}]$ from above. With the help of Lemmas 1 and 2 we now derive the main result of this section:

**Theorem 3.** Under (A1)–(A3), it holds

$$\text{Var}[f(X_{\Delta,T}) - \tilde{M}^{(2),T}_{\Delta,T}] \leq J \left( c_m - (\tilde{c}^m) \left( c_m - \left( \frac{1}{2} \right)^m \right) \right),$$

where $\tilde{c}_m = c_m - (\tilde{c}^m)$.

**Complexity of the SRCV approach**

Let us study the complexity of the SRCV approach. The overall cost is of order $JK \max \{NK, (c_m - 1)N_0\}$. We have the following constraints

$$\max \left\{ \frac{1}{N^2}, \frac{JK \log(N) c_m (c_m - 1)}{N N_0}, \frac{JC \tilde{c}_m}{N^* N_0} \right\} \leq \epsilon^2.$$
where the first term comes from the squared bias of the estimator and the remaining two ones come from the variance of the estimator (see Theorem 3 as well as footnote 4 on page 13). It is natural to expect that the optimal solution is given by all constraints being active as well as $NK \approx (c_m - 1)N_0$, that is, both terms in the overall cost are of the same order. Provided that $\kappa > 1$, we obtain the following parameter values:

$$J \approx \varepsilon^{-\frac{1}{2}}, \quad K \approx \left[ \frac{c_m^2 C_m^2}{\varepsilon^2} \right]^{\frac{1}{2}}, \quad N \approx \frac{(c_m - 1)\sqrt{c_m}}{\varepsilon^\frac{5}{4}} \sqrt{\log \left( \frac{\varepsilon^{-\frac{5}{4}}}{\varepsilon} \right)},$$

$$N_0 \approx \frac{NK}{c_m - 1} \approx \left[ \frac{c_m^2 C_m^2}{\varepsilon^{\frac{7}{2}}} \right]^{\frac{1}{2}} \sqrt{\log \left( \frac{\varepsilon^{-\frac{7}{4}}}{\varepsilon} \right)}.$$  

Thus, we have for the complexity

$$G \approx JNK^2 \approx JN_0K(c_m - 1) \approx \left[ \frac{(c_m - 1)2^{x+2}c_m^{x-1}C_m^4}{\varepsilon^{\frac{7}{2}+17}} \right]^{\frac{1}{2}} \sqrt{\log \left( \frac{\varepsilon^{-\frac{7}{4}}}{\varepsilon} \right)}.$$  

Note that the log-term in the solution of $N$ and $N_0$ has been added afterwards to satisfy all constraints. Complexity estimate (30) shows that one can go beyond the complexity order $\varepsilon^{-2}$, provided that $\kappa > 9$, and that we can achieve the complexity order $\varepsilon^{-1.75-\delta}$, for arbitrarily small $\delta > 0$, provided $\kappa$ is large enough.

4. Numerical results

In this section, we present several numerical examples showing the efficiency of the SRCV approach. It turns that even the weak Euler scheme (6) already shows the advantage of the new methodology over the standard Monte Carlo (SMC) as well as over the original RCV and RRCV approaches in terms of variance reduction effect. Regarding the choice of basis functions, we use for the RCV, RRCV and SRCV approaches polynomials of degree $d \leq p$, that is, $\psi(t) = \prod_{j=1}^d x_j^{l_j}$, where $l = (l_1, \ldots, l_d) \in \{0, 1, \ldots, p\}^d$ and $\sum_{i=1}^d l_i \leq p$. In addition to the polynomials, we consider the function $f$ as a basis function. We choose $J = 100$, $N = 10^3$, $N_0 = 10^5$, $p = 1$ in all examples. Hence, we have overall $K = (p_d + d + 1 = d + 2$ basis functions in each regression. Then we compute the estimated variances for the SMC, RCV, RRCV and SRCV approaches. More precisely, when speaking about “variance” below (e.g., in Tables 1–3) we mean sample variance of one summand $f(X_{\Delta, T}^{(i)}) - \bar{M}^{(1, (i))}$ (see (16)) in the case of RCV, RRCV and SRCV, while, in the case of SMC, the sample variance of $f(X_{\Delta, T})$ is meant. Thus, we analyse the variance reduction effect only, since the bias is the same for all these methods. To measure the numerical performance of a variance reduction method, we look at the ratio of variance vs. computational time, i.e., for the SRCV, we look at

$$\theta_{SRCV} := \frac{\text{Var}_{SRCV}}{\text{Var}_{SMC}} \cdot \frac{\text{Time}_{SRCV}}{\text{Time}_{SMC}},$$

where $\text{Var}_{SRCV}$ and $\text{Time}_{SRCV}$ denote the variance and the overall computational time of the SRCV approach ($\text{Var}_{SMC}$ and $\text{Time}_{SMC}$ have the similar meaning). The smaller $\theta_{SRCV}$ is, the more profitable is the SRCV algorithm compared to the SMC one. We similarly define $\theta_{RCV}$ and $\theta_{RRCV}$ (each of the regression-based algorithms is compared with the SMC approach).

4.1. Geometric brownian motion (GBM) with high volatility

Here $d = m = 1$ ($K = 3$). We consider the following SDE

$$dX_t = rX_t dt + \sigma X_t dW_t, \quad X_0 = 1,$$

for $t \in [0, 1]$, where $r = -1$ and $\sigma = 4$. Furthermore, we consider the functional $f(x) = x^2$. In the following, we plot the empirical cumulative distribution function (ECDF) of the “log-scaled sample”, which is

$$\log(1 + f_i - f_{\text{min}}) - \log(1 + \bar{f} - f_{\text{min}})$$
Table 1
Results of the algorithms for a quadratic function $f$ under a GBM model.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Min</th>
<th>Max</th>
<th>Variance</th>
<th>Time (s)</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCV</td>
<td>−0.5</td>
<td>0.2</td>
<td>$6.3 \cdot 10^{-8}$</td>
<td>30.5</td>
<td>$1.32 \cdot 10^{-23}$</td>
</tr>
<tr>
<td>RRCV</td>
<td>−25.4</td>
<td>1.7</td>
<td>$2.7 \cdot 10^{16}$</td>
<td>65.3</td>
<td>12.38</td>
</tr>
<tr>
<td>RCV</td>
<td>−27.8</td>
<td>0.1</td>
<td>$1.4 \cdot 10^{17}$</td>
<td>30.0</td>
<td>28.57</td>
</tr>
<tr>
<td>SMC</td>
<td>−10.6</td>
<td>15.9</td>
<td>$9.6 \cdot 10^{15}$</td>
<td>15.1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2
Results of the algorithms for a Call-on-max-option under a high-dimensional GBM.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Min</th>
<th>Max</th>
<th>Variance</th>
<th>Time (s)</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCV</td>
<td>−5.8</td>
<td>2.0</td>
<td>14.6</td>
<td>573.9</td>
<td>0.13</td>
</tr>
<tr>
<td>RCV</td>
<td>−10.4</td>
<td>0.7</td>
<td>11271.0</td>
<td>288.2</td>
<td>51.50</td>
</tr>
<tr>
<td>SMC</td>
<td>−1.9</td>
<td>7.2</td>
<td>448.9</td>
<td>140.5</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3
Results of the algorithms for a Call-on-max-option in a high-dimensional Heston model.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Min</th>
<th>Max</th>
<th>Variance</th>
<th>Time (s)</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCV</td>
<td>−6.4</td>
<td>2.6</td>
<td>50.1</td>
<td>444.7</td>
<td>0.09</td>
</tr>
<tr>
<td>RCV</td>
<td>−10.2</td>
<td>1.0</td>
<td>3208.8</td>
<td>328.6</td>
<td>4.33</td>
</tr>
<tr>
<td>SMC</td>
<td>−1.7</td>
<td>9.8</td>
<td>1478.8</td>
<td>164.5</td>
<td>1</td>
</tr>
</tbody>
</table>

for the SMC, and

$$\log(1 + u_i - u_{\min}) - \log(1 + \bar{u} - u_{\min})$$

for the RCV, and RRCV and SRCV, where

$$f_i := f(X_{\Delta T}^{(i)}), \quad u_i := f_i - \bar{M}_{\Delta T}^{(1)}, \quad i \in \{1, \ldots, N_0\},$$

$$f_{\min} := \min_{i=1,\ldots,N_0} f_i, \quad u_{\min} := \min_{i=1,\ldots,N_0} u_i,$$

$$\tilde{f} := \frac{1}{N_0} \sum_{i=1}^{N_0} f_i, \quad \bar{u} := \frac{1}{N_0} \sum_{i=1}^{N_0} u_i.$$  

The results for such a log-scaled sample are illustrated in Table 1. As can be also seen from the fourth plot in Fig. 1 (ECDFs of the SRCV and SMC), the variance reduction works absolutely fine for SRCV. Most of the sample values produced by SMC are much smaller than the corresponding mean value, whereas the deviation w.r.t. the mean $\bar{u}$ is very small for the SRCV approach. The main problem of the SMC approach in this case is that almost all paths tend to zero so that the small number of outliers is not sufficient to reach the (large) expectation $\mathbb{E}[f(X_{\Delta T})]$, i.e. $N_0$ has to be increased a lot to approach the expectation. In contrast, for the SRCV approach all paths (paths close to zero as well as outliers) are “shifted” close to the expectation and thus we obtain a very small variance. We only plot the ECDFs of the SRCV and SMC in Fig. 1, since the ECDFs of the RCV and RRCV look visually very similar to that for SRCV. The difference is, however, revealed in the “Min” and “Max” columns of Table 1. That is, the RCV and RRCV algorithms produce several outliers which result in that the RCV and RRCV do not give us any variance reduction effect! One reason for this significant difference between the algorithms is given in the first three plots in Fig. 1, where we illustrate the regression results for the RCV, RRCV and SRCV algorithms at the last time point, which means the first regression task. Here, we have accurate estimates only for the SRCV (cf. the discussion around (18)).

4.2. High-dimensional geometric Brownian motion

We consider the following SDE for $d = m = 10$ ($K = 12$):

$$dX^i_t = rX^i_t dt + \sigma^i X^i_t A^i dW_t, \quad t \in [0, 1], \quad i = 1, \ldots, 10,$$
where $X^i_0 = 1$, $\sigma^i = 2 \forall i$, $r = 0.05$ and $A^i := (A^{i,1}, \ldots, A^{i,10})$, $AA^T = (\rho_{ik})_{i,k=1,\ldots,10}$ with $\rho_{ik} = \rho_{ki} \in [-1, 1]$ and $\rho_{ik} = 1$ for $i = k$ (that is, $A^i W_i, i = 1, \ldots, 10$, are correlated Brownian motions). For $i < k$ we choose

$$\rho_{ik} = \begin{cases} 0.9 & \text{if } i = 1, k = 2, \\ -0.95 & \text{if } i = 3, k = 4, \\ 0.5 & \text{if } i = 5, k = 6, \\ -0.9 & \text{if } i = 7, k = 8, \\ 0.8 & \text{if } i = 9, k = 10, \\ 0 & \text{otherwise.} \end{cases}$$

In this example, we illustrate the performances of the algorithms by means of the functional $f(x) = \max \{\max_{i \in \{1,\ldots,10\}} x^i - 1, 0\}$. For saving a lot of computing time, we use the “simplified control variate”

$$\tilde{z}^{(1)}_{\Delta,T} := \sum_{j=1}^{J} \sum_{r=1}^{m} \tilde{a}_{j,r}(X_{\Delta,(j-1)\Delta}, D^{(r)}_N) \xi^i_j$$

rather than $\tilde{M}^{(1)}_{\Delta,T}$ for RCV and SRCV, where $\tilde{a}_{j,r}$ is a shorthand notation for $\tilde{a}_{j,k(r)}$ with $k(r) = (k_1, \ldots, k_m)$, $k_r = 1$, $k_l = 0$ for $i \neq r$. This simplification already takes much of the variance reduction power into account, while significantly reduces the number of summands needed to construct the control variate ($m = 10$ vs. $c_m - 1 = 2^m - 1 = 1023$ summands in the second sum above). For the SRCV algorithm, this results in the cost order $N_0 J m K$ instead of $N_0 J (c_m - 1) K$ in the testing phase ($10^{11}$ vs. $10^{13}$ in this example). Such a reduction in computational time due to using $\tilde{M}^{(1)}_{\Delta,T}$ applies also to the RCV algorithm, but does not apply to the RRCV algorithm. Namely, with $\tilde{M}^{(1)}_{\Delta,T}$ the testing phase of the RRCV algorithm would now cost $N_0 J c_m K + N_0 J m c_m$ (in the second summand we now have the factor $m$ instead of $c_m - 1$, cf. footnote 3 on page 8), which is still of order $10^{13}$ in the present example. Therefore, we do not consider the RRCV approach in this example. The results for the log-scaled sample are illustrated in Table 2. Again, the SRCV approach achieves a much smaller variance compared to the SMC and RCV (see the fifth plot in Fig. 1).

### 4.3. High-dimensional Heston model

We consider the following SDE for $d = m = 9$ ($K = 11$):

$$dX^i_t = r X^i_t dt + \sigma^i X^i_t \sqrt{X^9_t} dW^i_t, \quad i = 1, \ldots, 8,$$

$$dX^9_t = \lambda (\bar{v} - X^9_t) dt + \eta \sqrt{X^9_t} dW^9_t,$$

where $t \in [0, 1]$, $X^i_0 = 1$, $\sigma^i = 1$ for $i = 1, \ldots, 8$ as well as $X^9_0 = 4$, $r = 0.05$, $\lambda = 0.1$, $\bar{v} = 4$, $\eta = 1$ and $A^i := (A^{i,1}, \ldots, A^{i,10})$, $AA^T = (\rho_{ik})_{i,k=1,\ldots,9}$. Here, for $i < k$ we choose

$$\rho_{ik} = \begin{cases} 0.9 & \text{if } i = 1, k = 2, \\ -0.95 & \text{if } i = 3, k = 4, \\ 0.5 & \text{if } i = 5, k = 6, \\ -0.9 & \text{if } i = 7, k = 8, \\ -0.2 & \text{if } i \in \{1, 2, 3, 5, 6, 7\}, k = 9, \\ 0.2 & \text{if } i \in \{4, 8\}, k = 9, \\ 0 & \text{otherwise.} \end{cases}$$

One might think about $X^1, \ldots, X^8$ as price process of 8 stocks, while the CIR process $X^9$ is their common stochastic volatility. Notice that Feller’s condition for $X^9$ is not satisfied ($\frac{2\lambda}{\eta^2} = 0.8 < 1$), that is, 0 is accessible boundary point for $X^9$ (with reflecting boundary behaviour). The discretised process $(X^9_{\Delta,j\Delta})_{j=0,\ldots,J}$ can become negative. We, therefore, use the following discretisation scheme

$$X^i_{\Delta,j\Delta} = X^i_{\Delta,(j-1)\Delta} \left( 1 + r \Delta + \sigma^i \sqrt{X^9_{\Delta,(j-1)\Delta}} + A^i \sqrt{\Delta \xi^i_j} \right),$$

$$X^9_{\Delta,j\Delta} = X^9_{\Delta,(j-1)\Delta} + \lambda \left( \bar{v} - (X^9_{\Delta,(j-1)\Delta})^+ \right) \Delta + \eta \sqrt{(X^9_{\Delta,(j-1)\Delta})^+} A^9 \sqrt{\Delta \xi^9_j},$$

where $i \in \{1, \ldots, 8\}$ and $x^+ := \max \{x, 0\}$. Here, we consider the functional $f(x) = \max \{\max_{i \in \{1,\ldots,8\}} x^i - 1, 0\}$ and, as in Section 4.2, use the simplified control variate $\tilde{M}^{(1)}_{\Delta,T}$ (we again exclude the RRCV approach). The results for the log-scaled sample are illustrated in Table 3. We get that the ECDF for the SRCV approach has a similar form as the one from Section 4.2 (see the sixth plot in Fig. 1). Notice that the values of the estimators lie in all cases around
4.6 (SMC: 4.62, RCV: 4.59, SRCV: 4.60). Nevertheless, in the case of the SRCV approach 75.5% of the paths are located within the interval (3, 6), whereas in case of the SMC approach this holds for only 13.0% of the paths and in case of the RCV approach for only 9.9%. This is a further indication of a better numerical performance of the SRCV approach.

5. Proofs

5.1. Proof of Lemma 1

Applying Theorem 11.1 in [2] and using Assumption (A1) leads to
\[
\mathbb{E}\|\tilde{h}_{j,y,z} - h_{j,y,z}\|_{L^2(P_{\Delta,j-1})}^2 \leq \tilde{c} A^2 (\log (N p_m(y)) + 1) \frac{K}{N p_m(y)} + 8 \inf_{g \in \Psi_K} \int_{\mathbb{R}^d} (h_{j,y,z} (x) - g (x))^2 \, \mathbb{P}_{\Delta,j-1}(dx),
\]

since the expected number of (training) paths given \(\xi_j = y, V_j = z\) is \(N p_m(y)\) and
\[
\text{Var}[q_j(X_{\Delta,j} \mid \xi_j = y, V_j = z)] = \text{Var}[q_j(\Phi_{\Delta}(x, y, z))] = 0.
\]

By means of Assumption (A2) and \(\log p_m(y) < 0\) we finally obtain (27).

5.2. Proof of Lemma 2

Let us first recall that \(2^{m(m-1)/2} p_m(y) = \mathbb{P}(\xi_j = y) = \prod_{i=1}^{m} \mathbb{P}(\xi^i_j = y_i)\) (cf. (23)). Lemma 1 together with formulae (25) and (27) as well as \((\sum_{i=1}^{m} b_i)^2 \leq c_m \sum_{i=1}^{m} b_i^2\) yields
\[
\mathbb{E}\|\tilde{a}_{j,o,r} - a_{j,o,r}\|_{L^2(P_{\Delta,j-1})}^2
\leq c_m \sum_{y \in \{-\sqrt{3}, 0, \sqrt{3}\}^m} \sum_{z \in \{-1, 1\}^m} \left[ \prod_{i=1}^{m} H_{\xi_j^i} (y_i) \prod_{(k,l) \in \mathcal{J}} z_{kl} \right] \left( \frac{K p_m(y)}{N} + \frac{8 C_K p_m(y)^2}{K^*} \right)
\]
\[
= c_m \tilde{c} A^2 (\log N + 1) \frac{K}{N} \left[ \prod_{i=1}^{m} H_{\xi_j^i} (y_i) \right] + \frac{8 C_K}{K^*} \sum_{y \in \{-\sqrt{3}, 0, \sqrt{3}\}^m} c_m 2^{m(m-1)/2} p_m(y) \prod_{i=1}^{m} H_{\xi_j^i} (y_i)
\]
\[
= c_m \tilde{c} A^2 (\log N + 1) \frac{K}{N} + \frac{8 C_K}{K^*} c_m o,
\]

where in the last equality we used that \(\xi_j^1, \ldots, \xi_j^m\) are independent and all \(H_{\xi_j^i} (\xi_j^i)\) have unit \(L^2\)-norm.

5.3. Proof of Theorem 3

It holds
\[
\text{Var}[f(X_{\Delta,T} - \tilde{M}_{\Delta,T}^{(2)}]) = \text{Var}[M_{\Delta,T}^{(2)} - \tilde{M}_{\Delta,T}^{(2)}]
\]
\[
= \mathbb{E} \text{Var}[M_{\Delta,T}^{(2)} - \tilde{M}_{\Delta,T}^{(2)} | D_N^{(r)}] + \text{Var} \mathbb{E}[M_{\Delta,T}^{(2)} - \tilde{M}_{\Delta,T}^{(2)} | D_N^{(r)}].
\]

Due to the martingale transform structure in (22) and (29), we have
\[
\mathbb{E}[M_{\Delta,T}^{(2)} - \tilde{M}_{\Delta,T}^{(2)} | D_N^{(r)}] = 0.
\]

Together with the fact that the system \(\left\{ \prod_{i=1}^{m} H_{\xi_j^i} (\xi_j^i) \prod_{(k,l) \in \mathcal{J}} (V^{kl})^{y_{ki}} : (o, r) \in \mathcal{O} \right\}\) is orthonormal in \(L^2\), we get
\[
\text{Var}[f(X_{\Delta,T} - \tilde{M}_{\Delta,T}^{(2)}]] = \sum_{j=1}^{J} \sum_{(o, r) \in \mathcal{O}} \mathbb{E}\|\tilde{a}_{j,o,r} - a_{j,o,r}\|_{L^2(P_{\Delta,j-1})}^2.
\]
With the expression $C_{m,o}$ of Lemma 2 we compute
\[
\sum_{(\alpha, \tau) \in \mathcal{W}} C_{m,o} = \sum_{\alpha \in \{0,1,2\}^m \tau \in \{0,1\}^m} C_{m,o} - \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} c_m 2^{m(m-1)/2} p_m(y)^2
\]
\[
= \sum_{\alpha \in \{0,1,2\}^m} 2^{m(m-1)/2} C_{m,o} - \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} c_m 2^{m(m-1)/2} p_m(y)^2
\]
\[
:= \alpha - \beta,
\]
where $\alpha$ (resp. $\beta$) denotes the first (resp. second) big sum in the above expression. Let us compute $\alpha$ and $\beta$. Recalling that
\[
2^{m(m-1)/2} p_m(y) = \mathbb{P}(\xi_j = y) = \prod_{i=1}^m \mathbb{P}(\xi_j = y_i),
\]
we get
\[
\alpha = c_m \sum_{\alpha \in \{0,1,2\}^m} \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} \prod_{i=1}^m \left[ \mathbb{P}(\xi_j = y_i) H_{o_i}(y_i) \right]^2
\]
\[
= c_m \left( \sum_{\alpha \in \{0,1,2\}^m} \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} \left[ \mathbb{P}(\xi_j = y_1) H_{o_1}(y_1) \right]^2 \right)^m = c_m,
\]
where the last equality follows by a direct calculation. Recalling that $c_m = 3^m 2^{m(m-1)/2}$ (we consider second order schemes), we obtain
\[
\beta = 3^m \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} \prod_{i=1}^m \mathbb{P}(\xi_j = y_i)^2 = 3^m \left( \sum_{\gamma \in \{-\sqrt{3},0,\sqrt{3}\}^m} \mathbb{P}(\xi_j = y_1)^2 \right)^m = \left( \frac{3}{2} \right)^m.
\]
Thus,
\[
\sum_{(\alpha, \tau) \in \mathcal{W}} C_{m,o} = c_m - \left( \frac{3}{2} \right)^m = \tilde{c}_m.
\]
The last expression together with Lemma 2 and (32) yields the result.

Acknowledgement

This study has been funded by the Russian Academic Excellence Project “5-100”.

References