Pricing high-dimensional Bermudan options using deep learning and higher-order weak approximation

Riu Naito* and Toshihiro Yamada†

First version: January 1, 2023
This version: December 31, 2024

Abstract

The paper proposes a new deep learning-based algorithm for high-dimensional Bermudan option pricing. This is the first study for arbitrary order discretization scheme in the Bermudan option pricing or the dynamic programming problems. The price of a Bermudan option is well approximated by discretizing the interval of early-exercise dates with a higher-order weak approximation method of stochastic differential equations (SDEs). In particular, we provide the theoretical rate of convergence for the discretization of a Bermudan option price utilizing the error analysis of weak approximation of SDEs for the case of irregular payoff functions. The high-performance deep learning method using higher-order weak approximation gives a fast estimation for the conditional expectations appearing in Bermudan option pricing even if the dimension is high. The new approximation scheme will be an alternative to the least squares regression method. Numerical examples for Bermudan option pricing in high-dimensional settings including 100-dimensional SABR stochastic volatility model demonstrate the validity of the proposed scheme.

Keywords : High-dimensional Bermudan option, Deep learning, Stochastic differential equation, High-order weak approximation

1 Introduction

Pricing early-exercisable financial derivatives such as American options and Bermudan options is one of the main topics in finance. In particular, numerical computation of Bermudan option prices is so important for financial institutions in practice since the fast valuation of early-exercisable swaptions and callable swaps is required in financial business. On the other hand, it is not an easy task, especially in high-dimensional cases, because iterative computation of conditional expectations for early-exercise dates is required. To compute the price of early-exercisable options without the nest of Monte Carlo simulation, various computation methods have been developed.

We briefly review the literature for computing early-exercisable financial derivatives. After the works on American option pricing of Merton et al. (1977), Schwartz (1977) and Cox et al. (1979), many researchers studied the numerical computation methods by various approaches. The tree-based methods are studied by Nelson amd Ramaswamy (1990), Broadie and Detemple (1996) and Akyıldırım et al. (2014). The regression based least squares Monte Carlo methods are proposed by Longstaff and Schwartz (2001), Tsitsiklis and Van Roy (2001) and Glasserman and Yu (2004), which are further studied by Clément et al. (2002), Egloff (2005), Belomestny (2011), Gagliardini

*University of Toyama, Toyama, Japan (previous affiliation: Hitotsubashi University, Asset Management One Co., Ltd. and Japan Post Insurance Co., Ltd., Tokyo, Japan)

†Hitotsubashi University, Tokyo, Japan

and Ronchetti (2013). In Broadie and Glasserman (1997) and Broadie et al. (1997), they propose the Monte Carlo method based on the simulation of the random trees. The quadrature methods are developed by Andricopoulos et al. (2003) and Chen et al. (2014). In Bouchard and Touzi (2004) and Bouchard et al. (2004), they propose Malliavin calculus-based approach. The stochastic mesh method is proposed in Broadie and Glasserman (2004). The dual approaches are studied by Rogers (2002), Andersen and Broadie (2004), and Haugh and Kogan (2004). In Broadie and Yamamoto (2005), the authors introduce an algorithm based on the double-exponential integration formula and the fast Gauss transform. The quantization method are studied by Bally and Pagés (2003) and Bally et al. (2005). The eigenfunction expansion approach is developed by Li and Linetsky (2013). The applications of the COS method for early-exercisable options are studied by Fang and Oosterlee (2011), Ruijter and Oosterlee (2016) and Van der Have and Oosterlee (2018). In Feng and Linetsky (2008a), the authors propose a new method based on extrapolation approach combined with spatial discretization for the corresponding partial integro-differential equations. The PCA-ANOVA-based approach is introduced in Reisinger and Wissmann (2015) for pricing financial derivatives in high-dimensional settings. The PROJ method developed on the work of Kirkby (2015) and Kirkby (2017) is applied to early-exercisable option pricing in Kirkby (2018) and Kirkby et al. (2017). The Hilbert transformation method introduced in Feng and Linetsky (2008b) is further developed in Feng and Lin (2013), Zeng and Kwok (2014), and Li and Ye (2019) extends it using an expansion method of Li (2014). We refer to Broadie and Detemple (2004), Detemple (2005) and Detemple (2014) for further important methods.

Recently, deep learning-based algorithms for computing high-dimensional partial differential equations (PDEs) and pricing high-dimensional American-type options have been developed during the past decade, for example, see E et al. (2017), Han et al. (2018), Sirignano and Spiliopoulos (2018), Fujii et al. (2019), Becker et al. (2019), Becker et al. (2020), Huré et al. (2020), Andersson and Oosterlee (2021), Beck et al. (2021), Becker et al. (2021), Chen and Wan (2021), Lapeyre and Lelong (2021), Liang et al. (2021), Takahashi et al. (2022). In Becker et al. (2019) and Becker et al. (2021), the authors propose an algorithm for early-exercisable options using deep learning to solve optimal stopping problems directly, and is extended by Andersson and Oosterlee (2021). Another deep learning algorithm for pricing Bermudan options is introduced in Lapeyre and Lelong (2021) to compute the continuation value of the options at each exercise date. In a different stream, Fujii et al. (2019) proposes a method using the algorithm of E et al. (2017) and Han et al. (2018) with an expansion approach to price early-exercisable options.

While there are various methods to estimate early-exercisable financial derivatives including deep learning-based schemes, we believe that there is still room for improvement. Efficient higher-order time-discretization method for stochastic differential equations (SDEs) will be a fast computation scheme for pricing Bermudan options. However, most methods in the literature discretize SDEs at only each exercise date. Few papers such as Ruijter and Oosterlee (2016) and Van der Have and Oosterlee (2018) discretize the interval between the exercise dates. On the other hand, they may require enormous computational costs to obtain accurate approximation for high-dimensional Bermudan option prices, or they cannot compute high-dimensional cases due to the curse of dimensionality. Moreover, the previous studies only apply specific discretization schemes in Bermudan option prices. Also, it seems difficult to perform precise numerical error analysis for the discretization scheme for estimating Bermudan option prices. In fact, precise discretization error bounds are not obtained in the literature.

In the paper, we introduce a new deep learning-based method with an efficient discretization for high-dimensional Bermudan option pricing. To the best of our knowledge, this is the first study for arbitrary order discretization in Bermudan option pricing. More precisely, we construct a higher-order discretization scheme for pricing Bermudan options by extending the weak approximation methods

of Iguchi and Yamada (2021), Naito and Yamada (2019, 2022), Yamada (2019, 2021), Yamada and Yamamoto (2020) or the expansion-based pricing methods of Takahashi and Yamada (2012, 2015, 2016). For the error estimate of the discretization, we provide a weak approximation analysis of SDEs for irregular test functions in order to overcome the difficulties on numerical error analysis for discretizing the price of Bermudan options which comes from the nonlinearity of the exercise decisions at each exercise date. The proposed deep learning algorithm is inspired by Beck et al. (2021) and Naito and Yamada (2022) which use deep neural networks to approximate conditional expectations for solving high-dimensional nonlinear PDEs. The implementation method of this paper using deep learning and higher-order discretization makes it possible to obtain an accurate approximation for high-dimensional Bermudan option prices, which will be an extension of the algorithms in the literature such as the least squares Monte Carlo methods of Longstaff and Schwartz (2001) and/or previous deep learning-based methods. Numerical experiments show the efficiency of the proposed method for pricing Bermudan options under high-dimensional models whose dimension is up to 100.

The organization of the paper is as follows. Section 2 introduces the proposed method for high-dimensional Bermudan option prices and shows the deep learning-based implementation for the method. Numerical examples are provided in Section 3 which demonstrates the validity and the effectiveness. Section 4 concludes the method. Appendix is devoted to the mathematical proofs of Theorem and the important results after an introduction to Malliavin calculus.

2 New scheme for pricing Bermudan options

2.1 Sketch of the scheme & Contribution of the paper

Before we explain the details, we give a simple sketch of the proposed scheme and state the contribution of the paper.

The aim of the paper is to show a fast, efficient computation scheme for the price v_0 (at time 0) of a high-dimensional Bermudan option written on X starting from x.

Our scheme and the algorithm are briefly summarized as follows:

1. For each j-th date of early-exercise dates $0 < \tau_1 < \cdots < \tau_K = T$, we discretize the interval $[\tau_j, \tau_{j+1}]$ n-times and estimate the optimal parameter for a deep neural network:

$$\Theta_j^* = argmin_{\Theta\in\mathbb{R}^\nu} E\big[\big|\mathbf{DeepNN}_j^{\Theta}(\bar{X}^n_{\tau_j}) - (\mathbf{DeepNN}_{j+1}^{\Theta_{j+1}^*} \vee f(\tau_{j+1},\cdot))(\bar{X}^n_{\tau_{j+1}}) \times \mathbf{Malliavin\ weight}_{j,j+1}^{(m),n}\big|^2\big]$$

where $\mathbf{DeepNN}_j^{\Theta}$ is a deep neural network (which will be described later) with a set of parameters Θ at j-th date, $f(\tau_j,\cdot)$ is the discounted payoff function at j-th date, $\bar{X}^n_{\tau_j}$ is the Euler-Maruyama scheme at j-th date which starts a fixed $x \in \mathbb{R}^q$ at $\tau_0 = 0$, and $\mathbf{Malliavin\ weight}_{j,j+1}^{(m),n}$ is the m-order Malliavin weight on the interval. Here, $a \vee b = \max\{a, b\}$. If $j = K-1$, the algorithm is replaced with

$$\Theta_{K-1}^* = argmin_{\Theta\in\mathbb{R}^\nu} E[|\mathbf{DeepNN}_{K-1}^{\Theta}(\bar{X}^n_{\tau_{K-1}}) - f(\tau_K, \bar{X}^n_{\tau_K}) \times \mathbf{Malliavin\ weight}_{K-1,K}^{(m),n}|^2].$$

2. On the interval $[0, \tau_1]$, we perform simple (single) parameter estimation:

$$\theta^* = argmin_{\theta\in\mathbb{R}} E[|\theta - \mathbf{DeepNN}_1^{\Theta_1^*}(\bar{X}^n_{\tau_1}) \times \mathbf{Malliavin\ weight}_{0,1}^{(m),n}|^2]$$

and we finally obtain an accurate approximation $v_0^{(m),n\ *} = \theta^* \vee f(\tau_0, x)$ such that

$$|v_0 - v_0^{(m),n\ *}| = O(n^{-m}).$$

We describe a sketch of the proposed scheme in Figure 1.

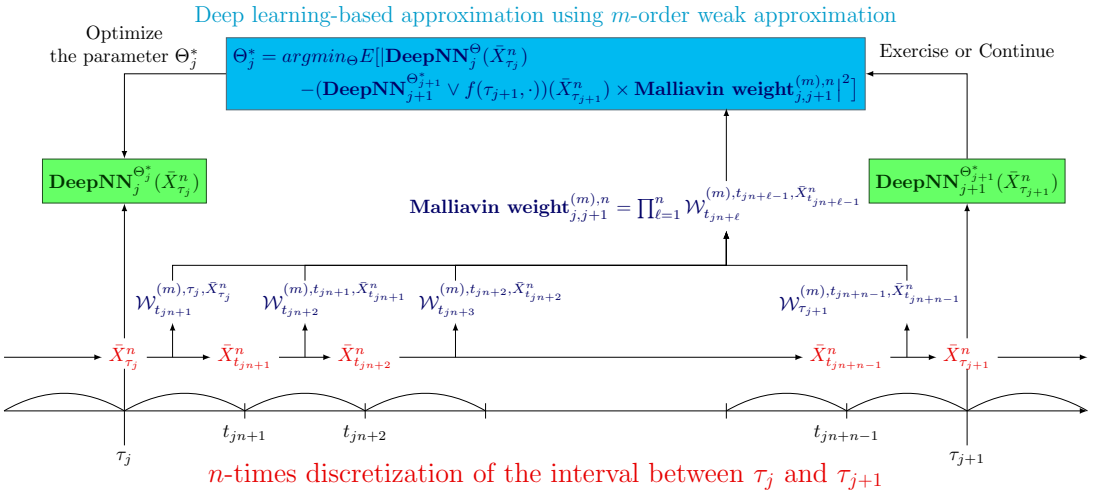

Figure 1: Sketch of the proposed scheme

The contribution of the paper is summarized as follows:

- The scheme gives a fast, efficient computation for the value functions on early-exercise dates without the curse of dimensionality due to the advantages of higher-order discretization and deep learning, which may be regarded as an extension of the method of Longstaff and Schwartz (2001);
- The discretization with the m-order Malliavin weight accelerates the convergence of the approximation as $O(n^{-m})$ which is justified by weak approximation theory, and the algorithm can be constructed for arbitrary $m \geq 1$;
- The m-order Malliavin weight is always computed with simulation of increment of multidimensional Brownian motion for all $m \geq 1$, in other words, the cost of generating random numbers is always the same as the Euler-Maruyama scheme.

2.2 Brief introduction of deep neural network

Let $s \in \{3, 4, 5, \ldots\}$ be a fixed integer. We describe a sketch of the fully-connected feedforward deep neural network:

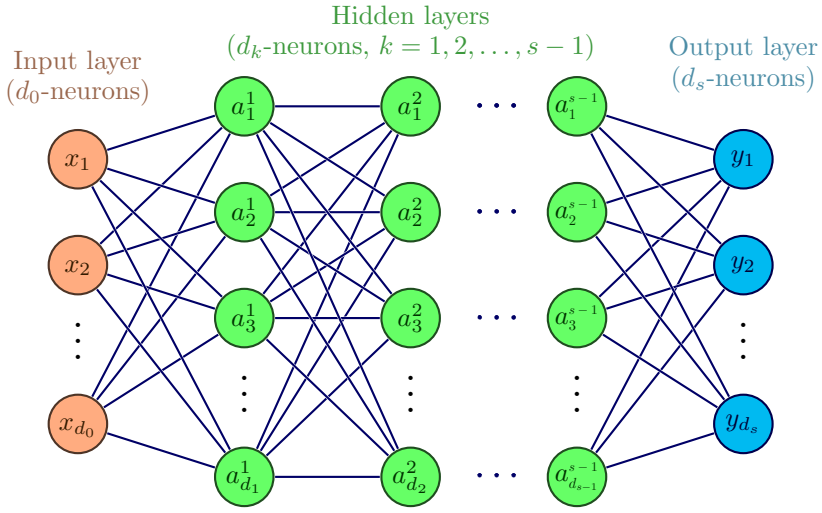

Figure 2: Sketch of the fully-connected feedforward deep neural network

The network in Figure 2 is composed of one input layer, $s-1$ hidden layers and one output layer. Roughly speaking, the deep neural network approximates the function from the input x to the output y through processing the data in hidden layers.

We briefly introduce the deep neural network from mathematical perspective. For $r \in \mathbb{N}$, let $\mathcal{L}_r : \mathbb{R}^r \to \mathbb{R}^r$ be an activation function given by $\mathbb{R}^r \ni x \mapsto \mathcal{L}_r(x) = (\mathcal{L}(x_1), \ldots, \mathcal{L}(x_r))$ with a nonlinear function $\mathcal{L} : \mathbb{R} \to \mathbb{R}$. For $p, \ell \in \mathbb{N}$, $e \in \{0\} \cup \mathbb{N}$ and $\theta = (\theta^1, \ldots, \theta^\nu) \in \mathbb{R}^\nu$ such that $e + \ell(p+1) \le \nu$, let $A_{p,\ell}^{\theta,e} : \mathbb{R}^p \to \mathbb{R}^\ell$ be an affine transformation function given by

$$A_{p,\ell}^{\theta,e}(x) = \begin{pmatrix} \theta^{e+1} & \cdots & \theta^{e+p} \\ \vdots & \ddots & \vdots \\ \theta^{e+(\ell-1)p+1} & \cdots & \theta^{e+\ell p} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} + \begin{pmatrix} \theta^{e+\ell p+1} \\ \vdots \\ \theta^{e+\ell p+\ell} \end{pmatrix}, \quad x \in \mathbb{R}^p.$$

Let $d_0, d_1, \ldots, d_s \in \mathbb{N}$ and $\nu = \sum_{i=1}^{s} d_i(d_{i-1}+1)$. Then, we define a deep neural network $\Phi : \mathbb{R}^{d_0} \to \mathbb{R}^{d_s}$ as

$$\begin{aligned} \Phi(x) = (A_{d_{s-1},d_s}^{\theta,\sum_{i=1}^{s-1} d_i(d_{i-1}+1)} \circ \mathcal{L}_{d_{s-1}} \circ A_{d_{s-2},d_{s-1}}^{\theta,\sum_{i=1}^{s-2} d_i(d_{i-1}+1)} \\ \circ \cdots \circ \mathcal{L}_{d_2} \circ A_{d_1,d_2}^{\theta,d_1(d_0+1)} \circ \mathcal{L}_{d_1} \circ A_{d_0,d_1}^{\theta,0})(x), \ x \in \mathbb{R}^{d_0}. \end{aligned} \tag{2.1}$$

Here, d_i represents the number of neurons of of i-th layer for $i = 0, 1, \ldots, s$, and ν is the number of parameters of the deep neural network. The deep neural network (2.1) corresponds to the network of Figure 2, i.e. $\Phi(x) = y = (y_1, \ldots, y_{d_s})$.

It is known that various functions such as p-integrable functions and/or measurable functions are arbitrary approximated by deep neural networks under some appropriate norms. This is called the universal approximation theorem of neural networks (see Arora et al. (2018) and Calin (2020)).

A target function can be estimated by a deep neural network with an optimal parameter $\theta^* \in \mathbb{R}^\nu$ obtained by minimizing the error of the loss function through the finding minima algorithm such as stochastic gradient descent (SGD) algorithms.

Hereafter, we precisely explain the scheme, theory and algorithm in detail.

2.3 Weak approximation

Let $C_b^\infty(\mathbb{R}^\ell;\mathbb{R}^m)$ be the space of all smooth functions $f:\mathbb{R}^\ell\to\mathbb{R}^m$ such that f and its partial derivatives of any order are bounded. For notational simplicity, let $C_b^\infty(\mathbb{R}^\ell)$ denote $C_b^\infty(\mathbb{R}^\ell;\mathbb{R})$. Similarly, we define the space $C_b^\infty([0,T]\times\mathbb{R}^\ell;\mathbb{R}^m)$ of smooth functions on $[0,T]\times\mathbb{R}^\ell$ with bounded derivatives of all orders. For a bounded measurable function $f:\mathbb{R}^\ell\to\mathbb{R}$, we define $\|f\|_\infty := \sup_{x\in\mathbb{R}^\ell}|f(x)|$.

Let $(\Omega,\mathcal{F},\{\mathcal{F}_t\}_t,P)$ be a filtered probability space and let $B=\{B_t\}_t$ be a d-dimensional Brownian motion. For a sub-σ-field $\mathcal{G}(\subset\mathcal{F})$ and $p\geq 1$, let $L^p(\Omega,\mathcal{G},P)$ be the space of all p-integrable $\mathcal{G}$-measurable functions on Ω with respect to P. For a random variable X, we write $\mathfrak{S}(X)$ for a σ-field generated by X.

Let us consider the solution of the following q-dimensional SDE driven by a d-dimensional Brownian motion:

$$X_s^{t,x} = x + \int_t^s \sigma_0(r,X_r^{t,x})dr + \sum_{i=1}^d \int_t^s \sigma_i(r,X_r^{t,x})dB_r^i, \qquad X_t^{t,x}=x\in\mathbb{R}^q, \tag{2.2}$$

where $\sigma_i\in C_b^\infty([0,T]\times\mathbb{R}^q;\mathbb{R}^q)$, $i=0,1,\dots,d$. We assume the uniformly elliptic condition for the matrix $\sigma=(\sigma_1,\dots,\sigma_d)$, i.e. there is $\epsilon>0$ such that $\sigma(t,x)\otimes\sigma(t,x)>\epsilon I_q$ for all $t\in[0,T]$ and $x\in\mathbb{R}^q$, where I_q is the identity matrix.

Let $K\in\mathbb{N}$ be a fixed natural number of the early-exercise dates of a Bermudan option and let τ_j, $j=1,\dots,K$ denotes the early-exercise dates given by

$$0=:\tau_0<\tau_1<\cdots<\tau_K=T$$

and $\tau_K-\tau_{K-1}=\cdots=\tau_1-\tau_0$. For numerical computation, we choose the number of time steps $n(\in\mathbb{N})$ for each interval of the early-exercise dates, i.e.

$$0=t=t_0<t_1<\cdots<t_{n-1}<t_n<t_{n+1}<\cdots<t_{nK-1}<t_{nK}=T, \tag{2.3}$$
$$\tau_j=t_{jn},\quad t_{jn}-t_{(j-1)n}=T/K,\qquad j=1,\dots,K, \tag{2.4}$$
$$t_i-t_{i-1}=T/(nK),\qquad i=1,\dots,nK. \tag{2.5}$$

Let $f:\{\tau_0,\dots,\tau_K\}\times\mathbb{R}^q\to\mathbb{R}$ be a bounded measurable function. For $j=0,1,\dots,K$, $f(\tau_j,\cdot)$ represents the discounted payoff function at the early-exercise date τ_j. Then, the value functions of the Bermudan option written on the underlying asset X on the early-exercise dates is given by:

$$v_{\tau_K}(x)=f(\tau_K,x), \tag{2.6}$$
$$v_{\tau_j}(x)=(P_{\tau_j,\tau_{j+1}}v_{\tau_{j+1}})(x)\vee f(\tau_j,x),\quad j=K-1,\dots,1,0, \tag{2.7}$$

for $x\in\mathbb{R}^q$, where $a\vee b=\max\{a,b\}$ and

$$P_{\tau_j,\tau_{j+1}}g(x)=E[g(X_{\tau_{j+1}}^{\tau_j,x})],\quad x\in\mathbb{R}^q,\ j=0,1,\dots,K-1, \tag{2.8}$$

for a bounded measurable function $g:\mathbb{R}^q\to\mathbb{R}$.

Let us define a scheme: for $i=jn$, $j=0,1,\dots,K-1$, $x\in\mathbb{R}^q$,

$$\bar{X}_{t_i}^{t_i,x}=x, \tag{2.9}$$
$$\bar{X}_{t_\ell}^{t_i,x}=\bar{X}_{t_{\ell-1}}^{t_i,x}+\sum_{j=1}^d\sigma_j(t_{\ell-1},\bar{X}_{t_{\ell-1}}^{t_i,x})\{B_{t_\ell}^j-B_{t_{\ell-1}}^j\},\quad \ell>i. \tag{2.10}$$

Hereafter, we use notation $B^e_{t,s} = B^e_s - B^e_t$, $e = 1, \dots, d$ and $B^0_{t,s} = s - t$. We introduce polynomials (not iterated integrals) of Brownian motion as follows: for $\alpha = (\alpha_1, \dots, \alpha_e) \in \{0, 1, \dots, d\}^e$, $e \in \mathbb{N}$,

$$\mathbb{B}^{\mathrm{Poly},\alpha}_{t,s} = \mathbb{B}^{\mathrm{Poly},(\alpha_1,\dots,\alpha_{e-1})}_{t,s} B^{\alpha_e}_{t,s} - \int_t^s D^{\alpha_e}_r \mathbb{B}^{\mathrm{Poly},(\alpha_1,\dots,\alpha_{e-1})}_{t,s} dr,$$

where $D_\cdot F = (D^1_\cdot F, \dots, D^d_\cdot F)$ represents the Malliavin derivative of a smooth Wiener functional F in the sense of Malliavin (see Appendix for more details on Malliavin calculus). If $e = 1$, $\mathbb{B}^{\mathrm{Poly},\alpha}_{t,s} = B^{\alpha_1}_{t,s}$. For example,

$$\begin{aligned}
\mathbb{B}^{\mathrm{Poly},(\alpha_1,\alpha_2)}_{t,s} &= B^{\alpha_1}_{t,s} B^{\alpha_2}_{t,s} - (s-t)\mathbf{1}_{\alpha_1=\alpha_2}, \\
\mathbb{B}^{\mathrm{Poly},(\alpha_1,\alpha_2,\alpha_3)}_{t,s} &= B^{\alpha_1}_{t,s} B^{\alpha_2}_{t,s} B^{\alpha_3}_{t,s} - B^{\alpha_1}_{t,s}(s-t)\mathbf{1}_{\alpha_2=\alpha_3} - B^{\alpha_2}_{t,s}(s-t)\mathbf{1}_{\alpha_1=\alpha_3} - B^{\alpha_3}_{t,s}(s-t)\mathbf{1}_{\alpha_1=\alpha_2}.
\end{aligned}$$

Let $V_i(t,x) = \sigma_i(t,x)$ and $\hat{V}_i g(t,x) = \sum_{j=1}^q V^j_i(t,x) \frac{\partial}{\partial x_j} g(t,x)$ for $t \in [0,T]$, $x \in \mathbb{R}^q$, $i = 1, \dots, d$ and a smooth function g. We also define $V_0(t,x) = \sigma_0(t,x) - (1/2)\sum_{i=1}^d \hat{V}_i \sigma_i(t,x)$ and $\hat{V}_0 g(t,x) = \partial_t g(t,x) + \sum_{j=1}^q V^j_0(t,x) \frac{\partial}{\partial x_j} g(t,x)$ for $t \in [0,T]$, $x \in \mathbb{R}^q$ and a smooth function g.

For a multi-index $\alpha = (\alpha_1, \dots, \alpha_\ell) \in \{0, 1, \dots, d\}^\ell$, $\ell \in \mathbb{N}$, we define $|\alpha| = \ell$ and $\|\alpha\| = \#\{j; \alpha_j \neq 0\} + 2\#\{j; \alpha_j = 0\}$, where $\#$ represents the number of its elements. Moreover, for two multi-indices $\alpha = (\alpha_1, \dots, \alpha_\ell) \in \{0, 1, \dots, d\}$, $\ell \in \mathbb{N}$ and $\beta = (\beta_1, \dots, \beta_m) \in \{0, 1, \dots, d\}$, $m \in \mathbb{N}$, we define $\alpha * \beta = (\alpha_1, \dots, \alpha_\ell, \beta_1, \dots, \beta_m)$. Let us define $\alpha^* = (\alpha^*_1, \dots, \alpha^*_{M(\alpha)}) = (\alpha_{i_1}, \dots, \alpha_{i_{M(\alpha)}})$ for a multi-index $\alpha = (\alpha_1, \dots, \alpha_\ell) \in \{0, 1, \dots, d\}^\ell$ where $M(\alpha) = |\alpha^*| = \#\{j; \alpha_j \neq 0\}$ and $\alpha_{i_j} \neq 0$, $j = 1, \dots, M(\alpha)$, $1 \le i_1 < \cdots < i_{M(\alpha)} \le \ell$. We write S_ℓ as the symmetric group of permutations of the index set $\{1, \dots, \ell\}$. For $\iota \in S_\ell$ and a word $I = (i_1, \dots, i_\ell)$, we define $\iota \cdot I = (i_{\iota(1)}, \dots, i_{\iota(\ell)})$. Here we introduce the notation Shuffle$(r_1, \dots, r_\ell)$ which represents for the set of all permutations $\zeta \in S_{r_1 + \cdots + r_\ell}$ such that $\zeta(1) < \cdots < \zeta(r_1)$, $\zeta(r_1{+}1) < \cdots < \zeta(r_1{+}r_2), \dots, \zeta(r_1{+}\cdots{+}r_{\ell-1}{+}1) < \cdots < \zeta(r_1{+}\cdots{+}r_\ell)$.

Let $(J_1, \dots, J_k)$ be a partition of a multi-index J, i.e. $J = J_1 * \cdots * J_\ell$ for some ℓ such that $\|J_i\| \le 2$ for $i = 1, \dots, \ell$. Suppose there exists a multi-index I which can be partitioned into sub-indices $I_1, \dots, I_\ell$ such that $I = I_1 * \cdots * I_\ell$ and for each $i = 1, \dots, \ell$ either $J_i = I_i$ with both length 1 or $J_i = (p,p)$ and $I_i = (0)$ for some $p \in \{1, \dots, d\}$. Then we say that I is related to J through the partitions $(I_1, \dots, I_\ell)$ and $(J_1, \dots, J_\ell)$ of length ℓ and denote this relationship by $J \sim_\ell I$ where ℓ denotes the number of subsets in the related partitions. When $J \sim_\ell I$, we define

$$\eta(J, I) := \#\{i; J_i \neq I_i\}. \tag{2.11}$$

See Gyurkó and Lyons (2010) for more details.

Let $m \in \{2, 3, \dots\}$. We define m-th order approximation operators $Q^{(m),n}_{\tau_j, \tau_{j+1}}$ given by $Q^{(m),n}_{\tau_j,\tau_{j+1}} = Q^{(m)}_{\tau_j, t_{jn+1}} \cdots Q^{(m)}_{t_{(j+1)n-1}, \tau_{j+1}}$ satisfying

$$\sup_{x \in \mathbb{R}^q} |E[g(X^{t_i,x}_{t_{i+1}})] - Q^{(m)}_{t_i,t_{i+1}} g(x)| = O((t_{i+1} - t_i)^{m+1}) \tag{2.12}$$

for all $g \in C^\infty_b(\mathbb{R}^q)$, and

$$\sup_{x \in \mathbb{R}^q} \left| E[g(X^{\tau_j,x}_{\tau_{j+1}})] - Q^{(m),n}_{\tau_j,\tau_{j+1}} g(x) \right| = O(n^{-m}) \tag{2.13}$$

for all bounded measurable functions $g : \mathbb{R}^q \to \mathbb{R}$ (which will be clearly stated in Lemma 1 below), more precisely, we define

$$Q^{(m)}_{t_i,t_{i+1}} g(x) = E[g(\bar{X}^{t_i,x}_{t_{i+1}}) \mathcal{W}^{(m),t_i,x}_{t_{i+1}}], \tag{2.14}$$

$$Q^{(m),n}_{\tau_j,\tau_{j+1}} g(x) = Q^{(m)}_{\tau_j,t_{jn+1}} \cdots Q^{(m)}_{t_{(j+1)n-1},\tau_{j+1}} g(x) = E\Big[g(\bar{X}^{\tau_j,x}_{\tau_{j+1}}) \prod_{\ell=1}^n \mathcal{W}^{(m),t_{jn+\ell-1},\bar{X}^{\tau_j,x}_{t_{jn+\ell-1}}}_{t_{jn+\ell}} \Big], \tag{2.15}$$

for a bounded measurable function $g:\mathbb{R}^q\to\mathbb{R}$, where $\mathcal{W}^{(m),t_i,x}_{t_{i+1}}$ is the Malliavin weight given by

$$\mathcal{W}^{(m),t_i,x}_{t_{i+1}} = 1+\sum_{\xi=1}^{2m+1}\sum_{p=1}^{\xi}\sum_{\substack{\sum_{\ell=1}^p b_\ell=\xi+p,\\ b_i\geq 2,\ell=1,\dots,p}}\sum_{I=(I_1,\dots,I_p)\in\{1,\dots,q\}^p}\frac{1}{p!}\sum_{\substack{\alpha^e=(\alpha^e_1,\dots,\alpha^e_{r_e})\in\{0,1\dots,d\}^{r_e}\\ \|\alpha^e\|=b_e, r_e\in\mathbb{N}, e=1,\dots,p}}$$
$$\prod_{e=1}^p \hat{V}_{\alpha^e_1}\cdots\hat{V}_{\alpha^e_{r_e-1}}V^{I_e}_{\alpha^e_{r_e}}(t_i,x)\sum_{\substack{\beta=\zeta^{-1}\cdot(\alpha^1*\cdots*\alpha^k)\sim_h\gamma, h\in\mathbb{N}\\ \zeta\in\mathrm{Shuffles}(r_1,\dots,r_p)\\ |\gamma|\leq m}}\frac{1}{2^{\eta(\gamma,\beta)}}\frac{1}{|\gamma|!}(t_{i+1}-t_i)^{|\gamma|-|\gamma^*|}$$
$$\times\sum_{\kappa\in\{1,\dots,d\}^{|I|}}(t_{i+1}-t_i)^{-|I|}\sum_{\ell\in\{1,\dots,q\}^{|I|}}\prod_{h=1}^{|I|}[A^{-1}]_{I_h,\ell_h}(t_i,x)V^{\ell_h}_{\kappa_h}(t_i,x)\mathbb{B}^{\mathrm{Poly},\kappa*\gamma}_{t_i,t_{i+1}},\quad(2.16)$$

where $[A^{-1}](t,\cdot)$ is the inverse matrix of $A(t,\cdot)$ given by $[A]_{i,j}(t,\cdot)=\sum_{k=1}^d V^i_k(t,\cdot)V^j_k(t,\cdot)$, $1\leq i,j\leq q$, for $t\geq 0$, and $\eta(\gamma,\beta)=\#\{i;\gamma_i\neq\beta_i\}$ (see (2.11) and the definitions on multi-indices). Note that the weight $\mathcal{W}^{(m),t_i,x}_{t_{i+1}}$ is explicitly given by a certain polynomial of Brownian motion.

Remark 1 (Malliavin weight for $m=2$)**.** *Here, we show the example for the Malliavin weight for the second order discretization. For simplicity, we introduce differential operators* $L_0g(t,x)=\partial_t g(t,x)+\sum_{j=1}^q\sigma^j_0(t,x)\frac{\partial}{\partial x_j}g(t,x)+\frac{1}{2}\sum_{j,k=1}^q\sum_{\ell=1}^d\sigma^j_\ell(t,x)\sigma^k_\ell(t,x)\frac{\partial^2}{\partial x_j\partial x_k}g(t,x)$ *and* $L_ig(t,x)=V_ig(t,x)$ *for* $t\in[0,T]$, $x\in\mathbb{R}^q$, $i=1,\dots,d$, *acting on a smooth function* g. *Then, the Malliavin weight for the second order discretization is given by*

$$\mathcal{W}^{(2),t_i,x}_{t_{i+1}} = 1+\frac{1}{t_{i+1}-t_i}\sum_{p_1,p_2=1}^q\sum_{j_1=1}^d\sigma^{p_1}_0(t_i,x)[A^{-1}]_{p_1,p_2}(t_i,x)\sigma^{p_2}_{j_1}(t_i,x)B^{j_1}_{t_i,t_{i+1}}$$
$$+\frac{1}{2(t_{i+1}-t_i)}\sum_{p_1,p_2=1}^q\sum_{j_1,j_2=0,j_3=1}^d L_{j_1}\sigma^{p_1}_{j_2}(t_i,x)[A^{-1}]_{p_1,p_2}(t_i,x)\sigma^{p_2}_{j_3}(t_i,x)\mathbb{B}^{\mathrm{Poly},j_1,j_2,j_3}_{t_i,t_{i+1}}$$
$$+\frac{1}{2}\sum_{p_1,p_2=1}^q\sum_{j_1,j_2=1}^d\sigma^{p_1}_0(t_i,x)[A^{-1}]_{p_1,p_2}(t_i,x)\sigma^{p_2}_{j_1}(t_i,x)\sigma^{p_3}_0(t_i,x)[A^{-1}]_{p_3,p_4}(t_i,x)\sigma^{p_4}_{j_2}(t_i,x)\mathbb{B}^{\mathrm{Poly},j_1,j_2}_{t_i,t_{i+1}}$$
$$+\frac{1}{4}\sum_{p_1,p_2,p_3,p_4=1}^q\sum_{j_1,j_2,j_3,j_4=1}^d L_{j_1}\sigma^{p_1}_{j_2}(t_i,x)[A^{-1}]_{p_1,p_2}(t_i,x)\sigma^{p_2}_{j_3}(t_i,x)L_{j_1}\sigma^{p_3}_{j_2}(t_i,x)[A^{-1}]_{p_3,p_4}(t_i,x)V^{p_4}_{j_4}(t_i,x)\mathbb{B}^{\mathrm{Poly},j_3,j_4}_{t_i,t_{i+1}}.$$

We have the following approximation result on the interval of the early-exercise dates.

Lemma 1 (Approximation of expectation on the interval of the early-exercise dates)**.** *There exists* $C>0$ *independent of* K *such that*

$$\sup_{x\in\mathbb{R}^q}\left|(P_{\tau_j,\tau_{j+1}}-Q^{(m),n}_{\tau_j,\tau_{j+1}})g(x)\right|\leq C\|g\|_\infty K^{-1}n^{-m},\quad(2.17)$$

for all bounded measurable functions $g:\mathbb{R}^q\to\mathbb{R}$, $j=0,1,\dots,K-1$ *and* $n\geq 1$.

Proof. See Appendix A. □

Lemma 1 tells us that the expectation $E[g(X^{t_i,x}_{t_{i+1}})]$ can be arbitrary and accurately approximated if we choose n and construct the m-order weight which is easily computed and simulated since it consists of the Brownian increments. Furthermore, the error bound of the approximation is estimated with the uniform bound of the test function g, which plays a role in the global approximation of the

price of the Bermudan option.

We now define the following functions: for $x \in \mathbb{R}^q$,

$$\bar{v}^{(m),n}_{\tau_K}(x) = f(\tau_K, x), \tag{2.18}$$
$$\bar{v}^{(m),n}_{\tau_j}(x) = (Q^{(m),n}_{\tau_j,\tau_{j+1}} \bar{v}^{(m),n}_{\tau_{j+1}})(x) \vee f(\tau_j, x), \quad j = K-1, \ldots, 1, 0. \tag{2.19}$$

The following is our main result.

Theorem 1 (m-order weak approximation for Bermudan option price v_0)**.** *There exists $C > 0$ such that*

$$\sup_{x\in\mathbb{R}^q} \left|v_0(x) - \bar{v}^{(m),n}_0(x)\right| \le C \max_{1\le j\le K} \|f(\tau_j, \cdot)\|_\infty n^{-m}, \tag{2.20}$$

for all $n \ge 1$.

Proof. See Appendix B. □

Remark 2 (On the Euler-Maruyama scheme)**.** *One can show the first order approximation result with the uniform bound of the payoff function for the Euler-Maruyama scheme (Maruyama (1955)) in Bermudan option pricing, which may be not appeared in the weak approximation literature. Let us define operators as: for $x \in \mathbb{R}^q$,*

$$Q^{\mathrm{EM}}_{t_i,t_{i+1}} g(x) = E[g(\bar{X}^{\mathrm{EM},t_i,x}_{t_{i+1}})], \tag{2.21}$$
$$Q^{\mathrm{EM},n}_{\tau_j,\tau_{j+1}} g(x) = Q^{\mathrm{EM}}_{\tau_j,t_{jn+1}} \cdots Q^{\mathrm{EM}}_{t_{(j+1)n-1},\tau_{j+1}} g(x) = E\big[g(\bar{X}^{\mathrm{EM},\tau_j,x}_{\tau_{j+1}})\big], \tag{2.22}$$

where $\bar{X}^{\mathrm{EM},t_i,x}_{t_\ell} = \bar{X}^{\mathrm{EM},t_i,x}_{t_{\ell-1}} + \sum_{e=0}^d \sigma_e(t_{\ell-1}, \bar{X}^{\mathrm{EM},t_i,x}_{t_{\ell-1}}) B^e_{t_{\ell-1},t_\ell}$, $i = jn$, $\ell > i$. We define the following functions as the approximations of the the values of a Bermudan option on the early-exercise dates: for $x \in \mathbb{R}^q$,

$$\bar{v}^{\mathrm{EM},n}_{\tau_K}(x) = f(\tau_K, x), \tag{2.23}$$
$$\bar{v}^{\mathrm{EM},n}_{\tau_j}(x) = (Q^{\mathrm{EM},n}_{\tau_j,\tau_{j+1}} \bar{v}^{\mathrm{EM},n}_{\tau_{j+1}})(x) \vee f(\tau_j, x), \quad j = K-1, \ldots, 1, 0. \tag{2.24}$$

Then, we have the following new result with the similar proof of Theorem 1, i.e. the first order approximation holds with respect to the number of time steps n based on the Euler-Maruyama scheme and its error bound is given by (Constant) $\times \max_{1\le j\le K} \|f(\tau_j, \cdot)\|_\infty n^{-1}$.

Corollary 1 (The Euler-Maruyama (first order) approximation for Bermudan option price v_0)**.** *There exists $C > 0$ such that*

$$\sup_{x\in\mathbb{R}^q} |v_0(x) - \bar{v}^{\mathrm{EM},n}_0(x)| \le C \max_{1\le j\le K} \|f(\tau_j, \cdot)\|_\infty n^{-1}, \tag{2.25}$$

for all $n \ge 1$.

Proof. See Appendix C. □

2.4 Deep learning algorithm

We implement the approximate functions $Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ and $\bar{v}^{(m),n}_{\tau_j}$ for $j = K-1,\dots,1,0$ by using deep neural networks. Let us briefly explain how the function $Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ is approximated by a deep neural network. Remark that the function $Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ satisfies

$$\begin{aligned}Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}(\bar{X}^{\tau_0,x}_{\tau_j}) = argmin_{\phi(\bar{X}^{\tau_0,x}_{\tau_j})\in L^2(\Omega,\mathfrak{S}(\bar{X}^{\tau_0,x}_{\tau_j}),P)}E\Big[\Big|\phi(\bar{X}^{\tau_0,x}_{\tau_j})\\ -(Q^{(m),n}_{\tau_{j+1},\tau_{j+2}}\bar{v}^{(m),n}_{\tau_{j+2}}\vee f(\tau_{j+1},\cdot))(\bar{X}^{\tau_0,x}_{\tau_{j+1}})\prod_{\ell=1}^{n}\mathcal{W}^{(m),t_{jn+\ell-1},\bar{X}^{\tau_0,x}_{t_{jn+\ell-1}}}_{t_{jn+\ell}}\Big|^2\Big],\end{aligned}$$

where $Q^{(m),n}_{\tau_{j+1},\tau_{j+2}}\bar{v}^{(m),n}_{\tau_{j+2}} = f(\tau_{j+1},\cdot)$ if $j = K-1$. Using this representation, the function $Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ can be estimated based on the universal approximation theorem of deep neural networks. As deep neural network approximation works well for high-dimensional cases, the high-dimensional Bermudan option prices can be computed using deep learning and a higher-order weak approximation, which means that the method of the paper gives an alternative to the classical schemes for pricing early-exercisable options.

Hereafter, we use the Rectified Linear Unit (ReLU) activation function given by

$$\mathcal{L}_r(x) = (\max\{x_1,0\},\dots,\max\{x_r,0\}),\quad x\in\mathbb{R}^r.$$

We construct a deep learning-based approximation $Q^{(m),n,\mathcal{NN},\theta^*_{\tau_j}}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ parametrized by $\theta^*_{\tau_j}\in\mathbb{R}^\nu$ for the function $Q^{(m),n}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ with an enough large $\nu\in\mathbb{N}$ and then obtain an approximation:

$$\bar{v}^{(m),n}_{\tau_j}\approx Q^{(m),n,\mathcal{NN},\theta^*_{\tau_j}}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}\vee f(\tau_j,\cdot),$$

where $Q^{(m),n,\mathcal{NN},\theta^*_{\tau_j}}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}$ is a deep neural network:

$$\begin{aligned}Q^{(m),n,\mathcal{NN},\theta}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}(x) = (A^{\theta,\sum_{i=1}^{s-1}d_i(d_{i-1}+1)}_{d_{s-1},1}\circ\mathcal{L}_{d_{s-1}}\circ A^{\theta,\sum_{i=1}^{s-2}d_i(d_{i-1}+1)}_{d_{s-2},d_{s-1}}\\ \circ\cdots\circ\mathcal{L}_{d_2}\circ A^{\theta,d_1(d_0+1)}_{d_1,d_2}\circ\mathcal{L}_{d_1}\circ A^{\theta,0}_{d_0,d_1})(x),\ x\in\mathbb{R}^q\end{aligned}$$

with $\theta^*_{\tau_j}$ given by $\theta^*_{\tau_j} = argmin_\theta L^{\tau_j}(\theta)$ with the loss function $L^{\tau_j}:\mathbb{R}^\nu\to\mathbb{R}$:

$$\begin{aligned}L^{\tau_j}(\theta) = E\Big[\Big|Q^{(m),n,\mathcal{NN},\theta}_{\tau_j,\tau_{j+1}}\bar{v}^{(m),n}_{\tau_{j+1}}(\bar{X}^{\tau_0,x}_{\tau_j})\\ -(Q^{(m),n,\mathcal{NN},\theta^*_{\tau_{j+1}}}_{\tau_{j+1},\tau_{j+2}}\bar{v}^{(m),n}_{\tau_{j+2}}\vee f(\tau_{j+1},\cdot))(\bar{X}^{\tau_0,x}_{\tau_{j+1}})\prod_{\ell=1}^{n}\mathcal{W}^{(m),t_{jn+\ell-1},\bar{X}^{\tau_0,x}_{t_{jn+\ell-1}}}_{t_{jn+\ell}}\Big|^2\Big].\end{aligned}$$

The algorithm of the proposed scheme is as follows:

Algorithm 1 Higher-order discretization for pricing Bermudan option with deep learning

Require: $M \in \mathbb{N}$ (batch size), $J \in \mathbb{N}$ (number of train-steps), $\gamma \in (0,1)$ (learning rate), $K \in \mathbb{N}$ (number of exercise dates), $n \in \mathbb{N}$ (number of time steps for each interval of the early-exercise dates)

for $i = K-1$ to 0 **do**

for $j = 1$ to J **do**

for $r = 1$ to M **do**

$\bar{X}_{\tau_0}^{\tau_0,x,r,j} = x$,

for $\ell = 1$ to $(i+1)n$ **do**

Generate i.i.d. Gaussian RVs $\Delta B_{t_{\ell-1},t_\ell}^{1,r,j}, \dots, \Delta B_{t_{\ell-1},t_\ell}^{d,r,j}$

$$\bar{X}_{t_\ell}^{\tau_0,x,r,j} = \bar{X}_{t_{\ell-1}}^{\tau_0,x,r,j} + \sum_{e=1}^{d} \sigma_e(t_{\ell-1}, \bar{X}_{t_{\ell-1}}^{\tau_0,x,r,j}) \Delta B_{t_{\ell-1},t_\ell}^{e,r,j}$$

end for

Compute $\mathcal{W}_{t_{in+\ell}}^{(m),t_{in+\ell-1},\bar{X}_{t_{in+\ell-1}}^{\tau_0,x,r,j}}$ for $\ell = 1, \dots, n$

end for

if $i = K-1$ **then**

$Q_{\tau_{i+1},\tau_{i+2}}^{(m),n,\mathcal{NN},\Theta_{\tau_{i+1}}} \bar{v}_{\tau_{i+2}}^{(m),n}(y) = f(\tau_{i+1}, y)$

end if

$$L^{\tau_i,j}(\theta_{\tau_i}^j) = \frac{1}{M} \sum_{r=1}^{M} \Big[Q_{\tau_i,\tau_{i+1}}^{(m),n,\mathcal{NN},\theta_{\tau_i}^j} \bar{v}_{\tau_{i+1}}^{(m),n}(\bar{X}_{\tau_i}^{\tau_0,x,r,j}) - (Q_{\tau_{i+1},\tau_{i+2}}^{(m),n,\mathcal{NN},\Theta_{\tau_{i+1}}} \bar{v}_{\tau_{i+2}}^{(m),n} \vee f(\tau_{i+1},\cdot))(\bar{X}_{\tau_{i+1}}^{\tau_0,x,r,j}) \prod_{\ell=1}^{n} \mathcal{W}_{t_{in+\ell}}^{(m),n,t_{in+\ell-1},\bar{X}_{t_{in+\ell-1}}^{\tau_0,x,r,j}} \Big]^2,$$

where $Q_{\tau_i,\tau_{i+1}}^{(m),n,\mathcal{NN},\theta} \bar{v}_{\tau_{i+1}}^{(m),n}$ is a function given by neural network

Minimize the loss function $L^{\tau_i,j}(\theta_{\tau_i}^j)$ and obtain the parameter $\theta_{\tau_i}^{*j}$ using ADAM optimizer (see Remark 4 below)

Update the parameter $\theta_{\tau_i}^{j+1} = \theta_{\tau_i}^{*j}$

end for

$\Theta_{\tau_i} = \theta_{\tau_i}^J$

end for

Return $(Q_{\tau_0,\tau_1}^{(m),n,\mathcal{NN},\Theta_{\tau_0}} \bar{v}_{\tau_1}^{(m),n} \vee f(\tau_0,\cdot))(x)$

Remark 3. *Due to the universal approximation theorem, the target functions appearing in the paper are arbitrary approximated by deep neural networks. This means that the target functions can be estimated within the error $O(n^{-m})$, and there exists θ such that*

$$\|v_0 - (Q_{\tau_0,\tau_1}^{(m),n,\mathcal{NN},\theta} \bar{v}_{\tau_1}^{(m),n} \vee f(\tau_0,\cdot))\|_\infty = O(n^{-m}),$$

which is also checked by numerical examples in the next section.

Remark 4. *In the above algorithm, the optimized parameter $\theta_{\tau_j}^*$ is obtained by using SGD J-times repeatedly with M-paths of the SDE, for each $j = K-1, \dots, 1, 0$. In the numerical experiments below, we employ Adam (see Kingma and Ba (2014)) and batch normalization (see Ioffe and Szegedy (2015)) as SGD to accelerate the learning processes.*

Remark 5. *In the above algorithm, at $\tau_0 = 0$, we can replace the function approximation $x \mapsto (Q_{\tau_0,\tau_1}^{(m),n,\mathcal{NN},\theta_{\tau_0}^*} \bar{v}_{\tau_1}^{(m),n} \vee f(\tau_0,\cdot))(x)$ with a scalar approximation $(Q_{\tau_0,\tau_1}^{(m),n,\mathcal{NN},\theta_{\tau_0}^*} \bar{v}_{\tau_1}^{(m),n} \vee f(\tau_0,\cdot))(x)$ for a fixed $x \in \mathbb{R}^q$ by optimizing the following loss function:*

$$L^{\tau_0}(\theta) = E\Big[\Big|\theta - (Q_{\tau_1,\tau_2}^{(m),n,\mathcal{NN},\theta_{\tau_1}^*} \bar{v}_{\tau_2}^{(m),n} \vee f(\tau_1,\cdot))(\bar{X}_{\tau_1}^{\tau_0,x}) \prod_{\ell=1}^{n} \mathcal{W}_{t_\ell}^{(m),t_{\ell-1},\bar{X}_{t_{\ell-1}}^{\tau_0,x}}\Big|^2\Big],$$

with the $K-1$-step deep neural network $Q^{(m),n,\mathcal{NN},\theta^*_{\tau_1}}_{\tau_1,\tau_2}\bar{v}^{(m),n}_{\tau_2} \vee f(\tau_1,\cdot)$. The optimized parameter $\theta^*_0 = argmin_{\theta\in\mathbb{R}} L^{\tau_0}(\theta)$ gives

$$|v_0(x) - \theta^*_0 \vee f(\tau_0, x)| = O(n^{-m}). \tag{2.26}$$

We employ this algorithm in the numerical examples later because it is enough to obtain the approximation value of $v_0(x)$ for a fixed $x \in \mathbb{R}^q$ in practice and also the scalar approximation is faster than the function approximation.

3 Numerical examples

We provide numerical examples in order to check the results (Theorem 1 and Corollary 1) with the algorithms, i.e. Malliavin based deep learning algorithm for the order of weak approximation $m \geq 2$, and Euler-Maruyama based deep learning algorithm for $m = 1$. The all experiments are performed in Python with Tensorflow 1.15 on two GPUs (NVIDIA GA102GL [RTX A6000] 48GB with NVLink) where the underlying system consists of an AMD EPYC 7402P 2.80GHz 24 cores CPU with 128 GB DDR4-3200 memory running on Ubuntu 18.04.

3.1 5-dimensional Black-Scholes model

First, we check that whether the proposed scheme works well for pricing Bermudan options by demonstrating a numerical experiment for the standard example with the given reference value. We refer to the result of Becker et al. (2019) since it reports the details of the computation result and provides the confidence interval of the computation. We consider the Bermudan maximum call option under the following 5-dimensional Black-Scholes model:

$$dX^i_t = (r-q)X^i_t dt + \sigma X^i_t dB^i_t, \tag{3.1}$$

for $i = 1,\dots,5$ and $X_0 = x \in \mathbb{R}^5$. We set the parameters as $K = 9$, $T = 3.0$, $x = (90.0,\dots,90.0)$, $r = 0.05$, $q = 0.1$, $\sigma = 0.2$ and the payoff function as $f : [0,T]\times\mathbb{R}^5 \to \mathbb{R}$ as $f(t,x) = e^{-rt}(0.0 \vee (x_1 - 100.0) \vee \cdots \vee (x_5 - 100.0))$.

We use the deep neural network composed of 1-input layer with 5-neurons, 2-hidden layers with 15-neurons each and 1-output layer with 1-neuron to implement the proposed scheme. We set the batch size M, train steps J and the learning rate $\gamma(j)$, $j \leq J$ as $M = 32768$, $J = 2000$ and $\gamma(j) = 2.5\times10^{-1}\mathbf{1}_{[0,0.3J]}(j) + 2.5\times10^{-2}\mathbf{1}_{(0.3J,0.6J]}(j) + 2.5\times10^{-3}\mathbf{1}_{(0.6J,J]}(j)$.

To check the efficiency of the proposed scheme, we compare the computing result of the deep optimal stopping method of Becker et al. (2019). For a consistent comparison, we discretize the interval of early-exercise dates n-times by the Euler-Maruyama scheme for SDEs and compute the expectation with the same number of paths as the batch size in our implementation of the method of Becker et al. (2019). To implement the deep optimal stopping method, we use the deep neural network with the same number of layers and neurons as the proposed scheme except for the input layer. We employ the input layer with $(5+1)$-neurons by adding the payoff to the input for each neural network according to Becker et al. (2019), i.e., we use $(5+1)$-dimensional input $(\bar{X}^{\tau_0,x}_{\tau_j}, f(\tau_j, \bar{X}^{\tau_0,x}_{\tau_j}))_{j=0,\dots,N}$ for each neural network instead of 5-dimensional input of $(\bar{X}^{\tau_0,x}_{\tau_j})_{j=0,\dots,N}$ in the implementation for the deep optimal stopping method.

We compute the approximation $v_0^{1\mathrm{st},n}(x)$, $v_0^{2\mathrm{nd},n}(x)$ and $v_0^{\mathrm{DOS},n}(x)$ obtained by the average of 25 independent trials of the proposed method of order 1, 2 and the deep optimal stopping method of Becker et al. (2019) and compare the relative error by $|v_0^{\mathrm{Ref}}(x) - v_0^{1\mathrm{st},n}(x)|/v_0^{\mathrm{Ref}}(x)$, $|v_0^{\mathrm{Ref}}(x) - v_0^{2\mathrm{nd},n}(x)|/v_0^{\mathrm{Ref}}(x)$ and $|v_0^{\mathrm{Ref}}(x) - v_0^{\mathrm{DOS},n}(x)|/v_0^{\mathrm{Ref}}(x)$ for each number of time steps n, where the reference value $v_0^{\mathrm{Ref}}(x) = 16.644$ is obtained from Table 1 of Becker et al. (2019).

The numerical result is summarized in Table 1.

Table 1: Numerical result for 5-dimensional Black-Scholes model

Scheme	n	Rel. Err.	Std. Dev.	Avg. Comp. Time	Total Comp. Time
Deep 1st order	2^6	0.00064	0.0639	277.16 s	6928.94 s
Deep 1st order	2^7	0.00026	0.0629	511.07 s	12776.81 s
Deep 2nd order	2^2	0.00025	0.0735	82.68 s	2067.07 s
DOS	2^7	0.00046	0.1293	125.49 s	3137.25 s
DOS	2^8	0.00027	0.1035	214.82 s	5370.52 s

Scheme	n	Value	95% CI of Becker et al. (2019)	Ref. Val.
Deep 1st order	2^6	16.655	[16.633, 16.648]	16.644
Deep 1st order	2^7	16.648	[16.633, 16.648]	16.644
Deep 2nd order	2^2	16.640	[16.633, 16.648]	16.644
DOS	2^7	16.636	[16.633, 16.648]	16.644
DOS	2^8	16.640	[16.633, 16.648]	16.644

Notes: "Deep 1st order", "Deep 2nd order" and "DOS" refer to the proposed scheme with $m = 1, 2$ and the deep optimal stopping method of Becker et al. (2019). "Rel. Err.", "Std. Dev.", "Avg. Comp. Time" and "Total Comp. Time" represent the relative error, the standard deviation, the average computing time (seconds) estimated by 25 independent trials of each scheme and the total computing time for 25 trials of each scheme. "95% CI of Becker et al. (2019)" refers to the 95% confidence interval given by Table 1 of Becker et al. (2019). The parameters are $K = 9$, $T = 3.0$, $x = (90.0, \dots, 90.0)$, $r = 0.05$, $q = 0.1$, $\sigma = 0.2$ and $f(t, x) = e^{-rt}(0.0 \vee (x_1 - 100.0) \vee \cdots \vee (x_5 - 100.0))$. The proposed algorithm is implemented by the deep neural network composed of 1-input layer with 5-neurons, 2-hidden layers with 15-neurons each and 1-output layer with 1-neuron. The parameters of the algorithm are $M = 32768$, $J = 2000$ and $\gamma(j) = 2.5 \times 10^{-1}\mathbf{1}_{[0,0.3J]}(j) + 2.5 \times 10^{-2}\mathbf{1}_{(0.3J,0.6J]}(j) + 2.5 \times 10^{-3}\mathbf{1}_{(0.6J,J]}(j)$. The deep optimal stopping method is implemented under the same neural network settings as the proposed algorithm with $(5 + 1)$-dimensional input.

The table shows that both the proposed schemes and the deep optimal stopping method provide accurate results which take values within the 95% confidence interval of Becker et al. (2019). In particular, the proposed second order scheme achieves a lower standard deviation with a shorter computing time than the deep optimal stopping method does. We also note that the computing time of the deep optimal stopping method is shorter than that of the proposed first order scheme for the reason that the deep optimal stopping method simultaneously learns all the neural networks while the proposed schemes recursively do that as many times as the number of early-exercise dates.

Since the validity of the proposed scheme has been confirmed through the numerical experiment above, we employ the proposed scheme to obtain the reference value of each numerical experiment in the following sections.

3.2 10-dimensional Black-Scholes model

Next, we apply the proposed method to price the Bermudan maximum put option under the 10-dimensional Black-Scholes model:

$$dX_t^i = (r - q)X_t^i dt + \sigma X_t^i dB_t^i, \tag{3.2}$$

for $i = 1, \dots, 10$, and $X_0 = x \in \mathbb{R}^{10}$. Here, we set the parameters as $K = 4$, $T = 1.0$, $x = (100.0, \dots, 100.0)$, $r = 0.0$, $q = 0.0$ and $\sigma = 0.2$. We set the payoff function $f : \mathbb{R}^{10} \to \mathbb{R}$ as $f(x) = 0.0 \vee (100.0 - x_1) \vee \cdots \vee (100.0 - x_{10})$. In the implementation of the algorithm, we employ the deep neural network composed of 1-input layer with 10-neurons, 2-hidden layers with 20-neurons each and 1-output layer with 1-neuron. Also we set the batch size M, train steps J and the learning rate $\gamma(j)$, $j \leq J$ as $M = 4096$, $J = 5000$ and $\gamma(j) = 10^{-1}\mathbf{1}_{[0,0.3J]}(j) + 10^{-2}\mathbf{1}_{(0.3J,0.6J]}(j) + 10^{-3}\mathbf{1}_{(0.6J,J]}(j)$. We compute

the approximation $v_0^{1\mathrm{st},n}(x)$, $v_0^{2\mathrm{nd},n}(x)$ and $v_0^{3\mathrm{rd},n}(x)$ obtained by the average of 25 independent trials of the proposed method of order 1, 2 and 3 and compare the relative error by $|v_0^{\mathrm{Ref}}(x)-v_0^{1\mathrm{st},n}(x)|/v_0^{\mathrm{Ref}}(x)$, $|v_0^{\mathrm{Ref}}(x)-v_0^{2\mathrm{nd},n}(x)|/v_0^{\mathrm{Ref}}(x)$ and $|v_0^{\mathrm{Ref}}(x)-v_0^{3\mathrm{rd},n}(x)|/v_0^{\mathrm{Ref}}(x)$ for each number of time steps n, where the reference value $v_0^{\mathrm{Ref}}(x)$ is computed by the average of 50 independent trials of the proposed first order method with the numerical solution of the forward SDE obtained by Itô formula.

We also compute the Bermudan option price by the least squares Monte Carlo method of Longstaff and Schwartz (2001) to compare the accuracy of the proposed method. The least squares Monte Carlo method is implemented with 10^6-paths and the polynomial basis functions of order up to 5. According to Longstaff and Schwartz (2001), we use the "in the money" paths only instead of all paths, which reduces the memory consumption. For a consistent comparison, we discretize the interval of early-exercise dates n-times by the Euler-Maruyama scheme for SDEs in our implementation of the method of Longstaff and Schwartz (2001)

We compute the approximation $v_0^{\mathrm{LSMC},n}(x)$ obtained by the average of 25 independent trials of the least squares Monte Carlo method of Longstaff and Schwartz (2001) and estimate the relative error by $|v_0^{\mathrm{Ref}}(x)-v_0^{\mathrm{LSMC},n}(x)|/v_0^{\mathrm{Ref}}(x)$ for each number of time steps n.

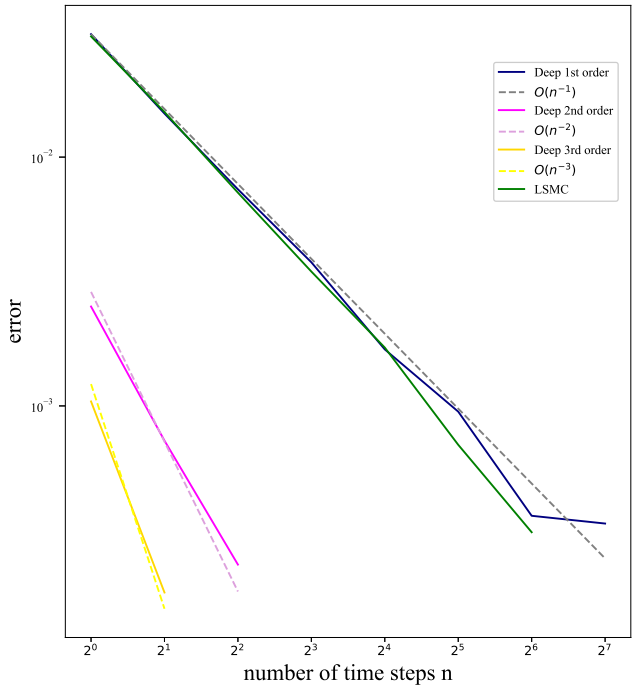

Figure 3: Relative error of the proposed scheme (10-dimensional Black-Scholes model)

Notes: The solid lines of "Deep 1st order" ,"Deep 2nd order", "Deep 3rd order" and "LSMC" represent the relative error estimated by 25 independent trials of the proposed scheme with $m = 1, 2, 3$ and the least squares Monte Carlo method of Longstaff and Schwartz (2001) for each n. The dashed lines of $O(n^{-1}), O(n^{-2})$ and $O(n^{-3})$ stand for the benchmark values decreasing with order $1, 2$ and 3 in n. The parameters are $K = 4$, $T = 1.0$, $x = (100.0, \ldots, 100.0)$ and $\sigma = 0.2$. The payoff function is $f(x) = 0.0 \vee (100.0 - x_1) \vee \cdots \vee (100.0 - x_{10})$. The algorithm is implemented by the deep neural network composed of 1-input layer with 10-neurons, 2-hidden layers with 20-neurons each and 1-output layer with 1-neuron. The parameters of the algorithm are $M = 4096$, $J = 5000$ and $\gamma(j) = 10^{-1}\mathbf{1}_{[0,0.3J]}(j) + 10^{-2}\mathbf{1}_{(0.3J,0.6J]}(j) + 10^{-3}\mathbf{1}_{(0.6J,J]}(j)$. The least squares Monte Carlo method is implemented with 10^5-paths and polynomial basis functions of order up to 5.

The numerical error for each n is plotted in Figure 3. This figure shows that the proposed higher-order methods require only a few number of time steps n to obtain an accurate approximation while the first order method based on the plain Euler-Maruyama scheme and the learst squares Monte Carlo method of Longstaff and Schwartz (2001) needs a quite large n to achieve the same level of the error. Furthermore, the slopes of the higher-order methods are steeper than that of the first order one, which is consistent with the theoretical rate of convergence of each scheme.

Table 2 summarizes the result of the numerical experiment. By the table, we can check that the relative error of the proposed second order method when $n = 2^2$ and the third order method when $n = 2^1$ are much small compared with that of the first order method when $n = 2^7$ and the least squares Monte Carlo method of Longstaff and Schwartz (2001) when $n = 2^6$ with the same level of the standard deviation. Moreover, in terms of the computing time, the proposed second and the third order method achieve more efficient performance than the first order one and the least squares Monte Carlo method.

Table 2: Numerical result for 10-dimensional Black-Scholes model

Scheme	n	Rel. Err.	Std. Dev.	Avg. Comp. Time	Total Comp. Time
Deep 1st order	2^7	0.00034	0.0341	247.90 s	6197.61 s
Deep 2nd order	2^2	0.00023	0.0323	63.64 s	1590.92 s
Deep 3rd order	2^1	0.00018	0.0347	62.53 s	1563.28 s
LSMC	2^6	0.00031	0.0323	150.32 s	3758.06 s

Notes: "Deep 1st order" ,"Deep 2nd order", "Deep 3rd order" and "LSMC" represent the relative error estimated by 25 independent trials of the proposed scheme with $m = 1, 2, 3$ and the least squares Monte Carlo method of Longstaff and Schwartz (2001) with polynomial basis functions of order up to 5 for each n. "Rel. Err.", "Std. Dev.", "Avg. Comp. Time" and "Total Comp. Time" represent the relative error, the standard deviation, the average computing time (seconds) estimated by 25 independent trials of each scheme and the total computing time for 25 trials of each scheme. The parameters and computational settings are the same as those listed in the notes of Figure 3.

3.3 100-dimensional SABR model (50-assets and 50-volatility processes)

Furthermore, we compute the price of the Bermudan maximum put option under the 100-dimensional SABR model which describes the dynamics of 50-assets with 50-stochastic volatility processes:

$$\begin{cases} dX_t^{2i-1} = X_t^{2i-1} X_t^{2i} dB_t^{2i-1}, \\ dX_t^{2i} = \nu X_t^{2i} (\rho dB_t^{2i-1} + \sqrt{1-\rho^2} dB_t^{2i}), \end{cases} \tag{3.3}$$

for $i = 1, \ldots, 50$, and $X_0 = x \in \mathbb{R}^{100}$. This model is developed in Hagan et al. (2002) and is now widely used by practitioners in financial institutions since it describes well the dynamics of the volatility smile. Another advantage of this model is that it has an easily implementable semi-analytic formula for the implied volatility unlike other models. However, since the SDE (3.3) has no explicit solution, it requires a heavy computational cost to obtain an accurate approximation of the option price. Therefore, it is practically important to construct an efficient computational scheme for the SDE (3.3).

We set the parameters as $K = 4, T = 1.0$, $(x_1, x_3, \ldots, x_{99}) = (100.0, \ldots, 100.0)$, $(x_2, x_4, \ldots, x_{100}) = (0.1, \ldots, 0.1)$, $\nu = 0.1$ and $\rho = -0.5$. Assume that the risk-free rate is zero again. The payoff function f is given by $f(x) = 0.0 \vee (100.0 - x_1) \vee (100.0 - x_3) \vee \cdots \vee (100.0 - x_{99})$. The proposed algorithm is implemented by the deep neural network composed of 1-input layer with 100-neurons, 2-hidden layers with 110-neurons each and 1-output layer with 1-neuron. We set the batch size $M = 4096$ and the train steps $J = 5000$ and use the learning rate $\gamma(j) = 5.0 \times 10^{-1} \mathbf{1}_{[0,0.3J]}(j) + 5.0 \times 10^{-2} \mathbf{1}_{(0.3J,0.6J]}(j) +$

$5.0 \times 10^{-3}\mathbf{1}_{(0.6J,J]}(j)$, $j \leq J$. We compare the relative error obtained in the same manner as the previous example by using the average of 25 independent trials of the proposed method of order 1 and 2, where the reference value is computed by the average of 50-independent trials of the first order method with $n = 2^8$.

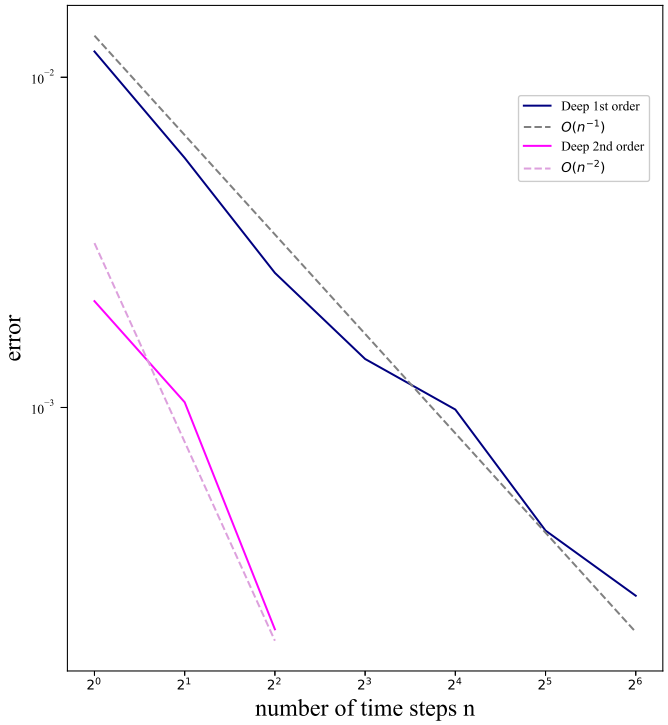

Figure 4: Relative error of the proposed scheme (100-dimensional SABR model)

Notes: The solid lines of "Deep 1st order" and "Deep 2nd order" represent the relative error estimated by 25 independent trials of the proposed scheme with $m = 1, 2$ for each n. The dashed lines of $O(n^{-1})$ and $O(n^{-2})$ stand for the benchmark values decreasing with order 1 and 2 in n. The parameters are $K = 4$, $T = 1.0$, $(x_1, x_3, \ldots, x_{99}) = (100.0, \ldots, 100.0)$, $(x_2, x_4, \ldots, x_{100}) = (0.1, \ldots, 0.1)$, $\nu = 0.1$ and $\rho = -0.5$. The payoff function is $f(x) = 0.0 \vee (100.0 - x_1) \vee (100.0 - x_3) \vee \cdots \vee (100.0 - x_{99})$. The algorithm is implemented by the deep neural network composed of 1-input layer with 100-neurons, 2-hidden layers with 110-neurons each and 1-output layer with 1-neuron. The parameters of the algorithm are $M = 4096$, $J = 5000$ and $\gamma(j) = 5.0 \times 10^{-1}\mathbf{1}_{[0,0.3J]}(j) + 5.0 \times 10^{-2}\mathbf{1}_{(0.3J,0.6J]}(j) + 5.0 \times 10^{-3}\mathbf{1}_{(0.6J,J]}(j)$.

We plot the numerical error of this experiments in Figure 4. By the figure, we can check that the error of the second order method is much smaller than that of the first order one even when $n = 1$. Also, this figure shows that the both method achieves the theoretical rate of convergence.

Furthermore, in order to compare the computational efficiency, we implement the least squares Monte Carlo method of Longstaff and Schwartz (2001) and the deep optimal stopping method of Becker et al. (2019) as in the previous sections. However, we could not implement the least squares Monte Carlo method even with 10^4-paths and the polynomial basis functions of order up to 3 due to the significant memory consumption. The deep optimal stopping method can be implemented under the same neural network settings of the proposed scheme with $(100 + 1)$-dimensional input as in the previous example.

The numerical result is summarized in Table 3.

Table 3: Numerical result for 100-dimensional SABR model

Scheme	n	Rel. Err.	Std. Dev.	Avg. Comp. Time	Total Comp. Time
Deep 1st order	2^6	0.00027	0.0292	333.91 s	8347.70 s
Deep 2nd order	2^2	0.00021	0.0276	111.69 s	2792.17 s
LSMC	–	–	–	– s	– s
DOS	2^6	0.00055	0.0608	162.47 s	4061.66 s

Notes: "Deep 1st order" and "Deep 2nd order" refer to the proposed scheme with $m = 1, 2$ and "LSMC" and "DOS" refer to the least squares Monte Carlo method of Longstaff and Schwartz (2001) and the deep optimal stopping method of Becker et al. (2019). "Rel. Err.", "Std. Dev.", "Avg. Comp. Time" and "Total Comp. Time" represent the relative error, the standard deviation, the average computing time (seconds) estimated by 25 independent trials of each scheme and the total computing time for 25 trials of each scheme. The proposed schemes are implemented with the same parameters and computational settings as those listed in the notes of Figure 4. We could not implement the least squares Monte Carlo method of Longstaff and Schwartz (2001) due to the significant memory consumption. The computational settings of the deep optimal stopping method of Becker et al. (2019) is the same as the proposed scheme with $(100 + 1)$-dimensional input.

The table shows that the proposed scheme and the deep optimal stopping method work even in the 100-dimensional case while we could not implement the least squares Monte Carlo method due to the significant memory consumption. Moreover, the proposed second order method when $n = 2^2$ achieves relatively lower error than the first order method and the deep optimal stopping method when $n = 2^6$. In terms of the standard deviation and the computing time, the above results show the efficiency of the second order method. We also note that the computing time of the deep optimal stopping method is shorter than the first order one with the same number of time steps n due to the same reason in Section 3.1.

Throughout the numerical experiments, we confirm that the proposed higher-order method outperforms the first order method based on the plain Euler-Maruyama scheme, the least-squares Monte Carlo method and the deep optimal stopping method.

4 Concluding remarks

The paper shows a new deep learning-based arbitrary order discretization scheme for Bermudan option pricing. The price of a Bermudan option is well approximated by discretizing the interval of early-exercise dates using a higher-order weak approximation method. The theoretical rate of convergence is provided for general measurable payoff functions. A deep learning-based algorithm combining with the higher-order discretization is proposed to solve high-dimensional problems. Numerical experiments confirmed that the proposed scheme efficiently approximates the price of a Bermudan option under high-dimensional models whose dimension is up to 100.

Backtesting the proposed method for more practical models and/or studying pricing and hedging strategy with the approach of Yamada and Yamamoto (2019) will be future work. It is important to consider whether the semi-closed form Kusuoka approximation of Yamada (2022) with deep learning will work well in Bermudan option pricing. It may also be interesting to study the convergence rate of the price of a Bermudan option to that of American option with respect to the number of early-exercise dates K, while we showed the convergence rate of the proposed algorithm with respect to the number of time steps n for a fixed K in Theorem 1 and Corollary 1.

Acknowledgements

We thank Prof. Shigeo Kusuoka (University of Tokyo) for grateful advice for the method of the paper. This work is supported by JST PRESTO (Grant Number JPMJPR2029), Japan.

Appendix

We prepare Malliavin calculus (see Ikeda and Watanabe (1989); Malilavin and Thalmaier (2006); Nourdin and Peccati (2012) and Nualart (2006), for example). Let $H = \{h \in \Omega;\ h \text{ is absolutely continuous, } \frac{d}{dt}h \in L^2([0,T];\mathbb{R}^d)\}$ be the Cameron-Martin space with the inner product $\langle \cdot, \cdot \rangle_H$. For $p \in [1,\infty)$, let $\| \cdot \|_p$ be the norm of the space of $L^p(\Omega)$. For $k \in \mathbb{N}$ and $p \in [1,\infty)$, let $\mathbb{D}^{k,p}$ be the Sobolev space with the norm $\|F\|_{k,p} = \|F\|_p + \sum_{j=1}^k \|D^j F\|_{L^p(\Omega;H^{\otimes j})}$ where D is the Malliavin derivative operator. Let $\mathbb{D}^\infty = \cap_{k,p}\mathbb{D}^{k,p}$ be the space of $\mathbb{R}$-valued smooth Wiener functionals in the sense of Malliavin. For $F \in (\mathbb{D}^\infty)^m$, let $\sigma^F = (\sigma^F_{ij})_{1\le i,j\le m}$ be the Malliavin covariance matrix given by

$$\sigma^F_{ij} = \langle DF_i, DF_j \rangle_H, \quad i,j = 1,\dots,m.$$

We say that $F \in (\mathbb{D}^\infty)^m$ is nondegenerate if σ^F is invertible a.s. and $\| \det(\sigma^F)^{-1} \|_p < \infty$ for all $p \in (1,\infty)$. Let $F \in (\mathbb{D}^\infty)^m$ be a nondegenerate Wiener functional and let $G \in \mathbb{D}^\infty$. For $f \in C_b^\infty(\mathbb{R}^m)$ and a multi-index $\alpha = (\alpha_1,\dots,\alpha_k)$, we have

$$E[\partial^\alpha f(F) G] = E[f(F) H_\alpha(F,G)],$$

where $\partial^\alpha f(x) = \frac{\partial^k}{\partial x_{\alpha_1}\cdots\partial x_{\alpha_k}} f(x)$ and $H_\alpha(F,G) = H_{(\alpha_k)}(F, H_{(\alpha_1,\dots,\alpha_{k-1})}(F,G))$ with
$H_{(i)}(F,G) = \sum_{j=1}^m D^*([\sigma^F]^{-1}_{ij} DF^j G)$, $i = 1,\dots,m$, where D^* is the adjoint operator of D.

A Proof of Lemma 1

We use a strategy based on Yamada (2021). For $\varphi \in C_b^\infty(\mathbb{R}^q)$, $j = 0,1,\dots,K-1$ and $\ell = 1,\dots,n$, we first expand $E[\varphi(X^{t_{jn+\ell-1},x}_{t_{jn+\ell}})]$ around $E[\varphi(\bar{X}^{t_{jn+\ell-1},x}_{t_{jn+\ell}})]$ using Itô, Stratonovich and Malliavin calculus (Gyurkó and Lyons (2010); Kloeden and Platen (1999); Kusuoka and Stroock (1984); Watanabe

(1987)) as follows:

$$
\begin{aligned}
&E[\varphi(X_{t_{jn+\ell}}^{t_{jn+\ell-1},x})] = E[\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})] \\
&+ \sum_{\xi=1}^{2m+1} \sum_{p=1}^{\xi} \sum_{\substack{\sum_{\ell=1}^{p} b_\ell = \xi+p,\\ b_i \geq 2, \ell=1,\dots,p}} \sum_{I=(I_1,\dots,I_p)\in\{1,\dots,q\}^p} \frac{1}{p!} \sum_{\substack{\alpha^e=(\alpha_1^e,\dots,\alpha_{r_e}^e)\in\{0,1\dots,d\}^{r_e}\\ \|\alpha^e\|=b_e, r_e\in\mathbb{N}, e=1,\dots,p}} \\
&\qquad \prod_{e=1}^{p} \hat{V}_{\alpha_1^e} \cdots \hat{V}_{\alpha_{r_e-1}^e} V_{\alpha_{r_e}^e}^{I_e}(t_{jn+\ell-1}, x) \\
&\qquad\qquad \times \sum_{\substack{\beta=\zeta^{-1}\cdot(\alpha^1 * \cdots * \alpha^k)\sim_h \gamma, h\in\mathbb{N}\\ \zeta\in \mathrm{Shuffles}(r_1,\dots,r_p)}} \frac{1}{2^{\eta(\gamma,\beta)}} E[\partial^I \varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x}) \mathbb{B}_{t_{jn+\ell-1},t_{jn+\ell}}^{\mathrm{Itô},\gamma}] \\
&\qquad\qquad\qquad + R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x) \\
&= E[\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})] + \sum_{\xi=1}^{2m+1} \sum_{p=1}^{\xi} \sum_{\substack{\sum_{\ell=1}^{p} b_\ell = \xi+p,\\ b_i \geq 2, \ell=1,\dots,p}} \sum_{I=(I_1,\dots,I_p)\in\{1,\dots,q\}^p} \frac{1}{p!} \sum_{\substack{\alpha^e=(\alpha_1^e,\dots,\alpha_{r_e}^e)\in\{0,1\dots,d\}^{r_e}\\ \|\alpha^e\|=b_e, r_e\in\mathbb{N}, e=1,\dots,p}} \\
&\qquad \prod_{e=1}^{p} \hat{V}_{\alpha_1^e} \cdots \hat{V}_{\alpha_{r_e-1}^e} V_{\alpha_{r_e}^e}^{I_e}(t_{jn+\ell-1}, x) \\
&\qquad\qquad \times \sum_{\substack{\beta=\zeta^{-1}\cdot(\alpha^1 * \cdots * \alpha^k)\sim_h \gamma, h\in\mathbb{N}\\ \zeta\in \mathrm{Shuffles}(r_1,\dots,r_p)}} \frac{1}{2^{\eta(\gamma,\beta)}} \frac{1}{|\gamma|!} (T/(nK))^{|\gamma|-|\gamma^*|} \\
&\times \sum_{\kappa\in\{1,\dots,d\}^{|I|}} (T/(nK))^{-|I|} \sum_{\ell\in\{1,\dots,q\}^{|I|}} \prod_{h=1}^{|I|} [A^{-1}]_{I_h,\ell_h}(t_{jn+\ell-1}, x) V_{\kappa_h}^{\ell_h}(t_{jn+\ell-1}, x) \\
&\qquad\qquad\qquad \times E[\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x}) \mathbb{B}_{t_{jn+\ell-1},t_{jn+\ell}}^{\mathrm{Poly},\kappa*\gamma}] \\
&\qquad\qquad\qquad\qquad + R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x), \qquad \text{(A.1)}
\end{aligned}
$$

where $R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}$ is a function satisfying there exists $C > 0$ independent of n, K and φ such that

$$
\|R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}\|_\infty \leq C\|\varphi\|_\infty (nK)^{-(m+1)}. \qquad \text{(A.2)}
$$

Here, $\mathbb{B}_{t,s}^{\mathrm{Itô},\gamma}$ is the iterated Itô integral with a multi-index $\gamma \in \{0, 1, \dots, d\}^e$ of length $e \in \mathbb{N}$, i.e.

$$
\mathbb{B}_{t,s}^{\mathrm{Itô},\gamma} = \int_{t<t_1<\cdots<t_e<s} dB_{t_1}^{\gamma_1} \cdots dB_{t_1}^{\gamma_e}.
$$

We are able to see that some terms in the expansion in (A.1) will also be $O(n^{-(m+1)})$ using the duality formula of Malliavin calculus (or Stein's method on Wiener space):

$$
E[\psi(F)D^*(u)] = E[(\nabla\psi)(F)\langle DF, u\rangle_H]
$$

for $\psi \in C_b^\infty(\mathbb{R}^q)$, $F \in \mathbb{D}^{1,2}$ and an adapted process to the Brownian filtration $u \in L^2(\Omega; H)$ satisfying that there exists $\eta \in L^2(\Omega)$ such that $E[\zeta\eta] = E[\langle D\zeta, u\rangle_H]$ for all $\zeta \in \mathbb{D}^{1,2}$ (see Nourdin and Peccati

(2012) or Nualart (2006), for example). We have that for all multi-index $\gamma \in \{0,1,\dots,d\}^L$, $L \geq m+1$,

$$\begin{aligned}\left|E[\partial^I \varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})\mathbb{B}^{\mathrm{It\hat{o}},\gamma}_{jn+\ell-1,t_{jn+\ell}}]\right| &= \left|\sum_{|\eta|\leq|\gamma^*|} E[\partial^{I*\eta}\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})] \prod_{e\leq|\gamma|} V^{\eta_e}_{\gamma^*_e}(x)\frac{1}{|\gamma|!}(T/(nK))^{-|\gamma|}\right| \\ &\leq C\|\nabla^p\varphi\|_\infty (nK)^{-(m+1)}.\end{aligned}$$

for some $C>0$ and $p\in\mathbb{N}$, where $\|\nabla^p\varphi\|_\infty = \max_{e\leq p}\max_{\alpha\in\{1,\dots,q\}^e}\|\partial^\alpha\varphi\|_\infty$. Therefore, we have

$$\begin{aligned}
&P_{t_{jn+\ell-1},t_{jn+\ell}}\varphi(x) \\
&= E[\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})] + \sum_{\xi=1}^{2m+1}\sum_{p=1}^{\xi}\sum_{\substack{\sum_{\ell=1}^p b_\ell=\xi+p,\\ b_i\geq 2,\ell=1,\dots,p}}\sum_{I=(I_1,\dots,I_p)\in\{1,\dots,q\}^p}\frac{1}{p!}\sum_{\substack{\alpha^e=(\alpha^e_1,\dots,\alpha^e_{r_e})\in\{0,1\dots,d\}^{r_e}\\ \|\alpha^e\|=b_e, r_e\in\mathbb{N}, e=1,\dots,p}} \\
&\qquad \prod_{e=1}^p \hat{V}_{\alpha^e_1}\cdots\hat{V}_{\alpha^e_{r_e-1}}V^{I_e}_{\alpha^e_{r_e}}(t_{jn+\ell-1},x) \\
&\qquad\qquad \times \sum_{\substack{\beta=\zeta^{-1}\cdot(\alpha^1*\cdots*\alpha^k)\sim_h\gamma, h\in\mathbb{N}\\ \zeta\in\mathrm{Shuffles}(r_1,\dots,r_p)\\ |\gamma|\leq m}} \frac{1}{2^{\eta(\gamma,\beta)}}\frac{1}{|\gamma|!}(T/(nK))^{|\gamma|-|\gamma^*|} \\
&\times \sum_{\kappa\in\{1,\dots,d\}^{|I|}}(T/(nK))^{-|I|}\sum_{\ell\in\{1,\dots,q\}^{|I|}}\prod_{h=1}^{|I|}[A^{-1}]_{I_h,\ell_h}(t_{jn+\ell-1},x)V^{\ell_h}_{\kappa_h}(t_{jn+\ell-1},x) \\
&\qquad\qquad \times E[\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})\mathbb{B}^{\mathrm{Poly},\kappa*\gamma}_{t_{jn+\ell-1},t_{jn+\ell}}] \\
&\qquad\qquad\qquad + R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x) + R^{2,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x) \\
&= Q^{(m)}_{t_{jn+\ell-1},t_{jn+\ell}}\varphi(x) + R^{1,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x) + R^{2,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x),
\end{aligned}$$

where $R^{2,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}$ is a function satisfying

$$R^{2,\varphi}_{t_{jn+\ell-1},t_{jn+\ell}}(x) = \sum_{|\gamma|\leq\nu_m} E[\partial^\gamma\varphi(\bar{X}_{t_{jn+\ell}}^{t_{jn+\ell-1},x})\Phi_\gamma^{t_{jn+\ell-1},t_{jn+\ell}}(x)], \tag{A.3}$$

for some integer $\nu_m\in\mathbb{N}$ such that $\nu_m\leq 2m$, and some functions $\Phi_\gamma^{t_{jn+\ell-1},t_{jn+\ell}}\in C_b^\infty(\mathbb{R}^q)$, $\gamma\in\{1,\dots,q\}^\eta$, $\eta\leq\nu_m$ such that for any $e\in\mathbb{N}\cup\{0\}$ and $\gamma\in\{1,\dots,q\}^\eta$, $\eta\leq\nu_m$,

$$\|\nabla^e\Phi_\gamma^{t_{jn+\ell-1},t_{jn+\ell}}\|_\infty = O((nK)^{-(m+1)}). \tag{A.4}$$

Hereafter, we assume that $g:\mathbb{R}^q\to\mathbb{R}$ is a bounded measurable function. We note that the following decomposition holds:

$$\begin{aligned}
&(P_{\tau_j,\tau_{j+1}} - Q^{(m),n}_{\tau_j,\tau_{j+1}})g(x) \\
&= \sum_{\ell=1}^n Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}(P_{t_{jn+\ell-1},t_{jn+\ell}} - Q^{(m)}_{t_{jn+\ell-1},t_{jn+\ell}})P_{t_{jn+\ell},\tau_{j+1}}g(x) \\
&= \sum_{\ell=1}^n Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}(R^{1,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}} + R^{2,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}})(x).
\end{aligned}\tag{A.5}$$

By the construction of $Q^{(m)}_{t_e,t_{e+1}}$, there exists $C>0$ independent of n and K such that

$$\|Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}\varphi\|_\infty \le C\|\varphi\|_\infty \tag{A.6}$$

for any bounded measurable function $\varphi:\mathbb{R}^q\to\mathbb{R}$ and $\ell\le n$. Thus, there exist $C_1,C_2>0$ independent of n such that

$$\|Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}R^{1,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}}\|_\infty \tag{A.7}$$

$$\le C_1\|R^{1,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}}\|_\infty \le C_2\|P_{t_{jn+\ell},\tau_{j+1}}g\|_\infty(nK)^{-(m+1)} \le C_2\|g\|_\infty(nK)^{-(m+1)}. \tag{A.8}$$

We next estimate the bound of $\|Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}R^{2,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}}\|_\infty$. Note that the function $Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}R^{2,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}}$ has the form:

$$Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}R^{2,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}} = \sum_{|\gamma|\le\nu_m} E[\partial^\gamma P_{t_{jn+\ell},\tau_{j+1}}g(\bar X^{\tau_j,x}_{t_{jn+\ell}})\zeta^{t_{jn},t_{jn+\ell},x}_\gamma] \tag{A.9}$$

where $\zeta^{t_{jn},t_{jn+\ell},x}_\gamma = \Phi^{t_{jn+\ell-1},t_{jn+\ell}}_\gamma(\bar X^{\tau_j,x}_{t_{jn+\ell-1}})\prod_{r=1}^{\ell}\mathcal{W}^{(m),t_{jn+r-1},\bar X^{\tau_j,x}_{t_{jn+r-1}}}_{t_{jn+r}}\in\mathbb{D}^\infty$. It holds that for $r\in\{0\}\cup\mathbb{N}$ and $p\ge1$, there exists $C>0$ independent of n and K such that $\|\zeta^{t_{jn},t_{jn+\ell},x}_\gamma\|_{r,p}\le C(nK)^{-(m+1)}$ using (A.4) with the fact $\|\bar X^{\tau_j,x}_{t_{jn+\ell}}\|_{r,p}\le C$ and the property of polynomials of Brownian motion: for $i=1,\dots,d$ and $p\in\mathbb{N}$, $E[(B^i_t)^p]=\frac{p!}{2^{p/2}(p/2)!}t^{p/2}$ if p is even, $E[(B^i_t)^p]=0$ if p is odd. To give the upper bound of $Q^{(m)}_{\tau_j,t_{jn+1}}\cdots Q^{(m)}_{t_{jn+\ell-2},t_{jn+\ell-1}}R^{2,P_{t_{jn+\ell},\tau_{j+1}}g}_{t_{jn+\ell-1},t_{jn+\ell}}$ with the uniform estimate of g, we take the following strategy. For $t_{jn+\ell}\le\tau_j+(\tau_{j+1}-\tau_j)/2$, we have

$$\|\partial^\gamma P_{t_{jn+\ell},\tau_{j+1}}g\|_\infty \le C\|g\|_\infty\frac{1}{(\tau_{j+1}-\tau_j)^{|\gamma|/2}} = C\|g\|_\infty\frac{1}{(T/K)^{|\gamma|/2}}, \tag{A.10}$$

by Theorem 3.5 and Corollary 3.7 of Kusuoka and Stroock (1984), and thus

$$\sup_{x\in\mathbb{R}^q}\Big|\sum_{|\gamma|\le\nu_m}E[\partial^\gamma P_{t_{jn+\ell},\tau_{j+1}}g(\bar X^{\tau_j,x}_{t_{jn+\ell}})\zeta^{t_{jn},t_{jn+\ell},x}_\gamma]\Big| \le C\|g\|_\infty\frac{K^{2m/2}}{(nK)^{m+1}} = C\|g\|_\infty K^{-1}n^{-(m+1)}, \tag{A.11}$$

by taking $\nu_m\le 2m$ into account. For $\tau_j+(\tau_{j+1}-\tau_j)/2<t_{jn+\ell}\le\tau_{j+1}$, we have

$$\begin{aligned}
&\sup_{x\in\mathbb{R}^q}\Big|\sum_{|\gamma|\le2m}E[\partial^\gamma P_{t_{jn+\ell},\tau_{j+1}}g(\bar X^{\tau_j,x}_{t_{jn+\ell}})\zeta^{t_{jn},t_{jn+\ell},x}_\gamma]\Big|\\
\le&\sup_{x\in\mathbb{R}^q}\Big|\sum_{|\gamma|\le2m}E[P_{t_{jn+\ell},\tau_{j+1}}g(\bar X^{\tau_j,x}_{t_{jn+\ell}})H_\gamma(\bar X^{\tau_j,x}_{t_{jn+\ell}},\zeta^{t_{jn},t_{jn+\ell},x}_\gamma)]\Big|\\
\le&C\|g\|_\infty\frac{K^{2m/2}}{(nK)^{m+1}} = C\|g\|_\infty K^{-1}n^{-(m+1)},
\end{aligned} \tag{A.12}$$

again by applying Theorem 3.5 and Corollary 3.7 of Kusuoka and Stroock (1984). Therefore, by (A.5), (A.8), (A.11) and (A.12), we have

$$\|(P_{\tau_j,\tau_{j+1}}-Q^{(m),n}_{\tau_j,\tau_{j+1}})g\|_\infty \le C\sum_{\ell=1}^{n}\|g\|_\infty K^{-1}n^{-(m+1)} = C\|g\|_\infty K^{-1}n^{-m} \tag{A.13}$$

for some $C>0$ independent of n and K. □

B Proof of Theorem 1

We first note that by using the inequality $|x \vee z - y \vee z| \leq |x - y|$ for all $x, y, z \in \mathbb{R}$, we have

$$\begin{aligned} |v_0(x) - \bar{v}_0^{(m),n}(x)| &= |P_{\tau_0,\tau_1} v_{\tau_1}(x) \vee f(\tau_0, x) - Q^{(m),n}_{\tau_0,\tau_1} \bar{v}^{(m),n}_{\tau_1}(x) \vee f(\tau_0, x)| \\ &\leq |P_{\tau_0,\tau_1} v_{\tau_1}(x) - Q^{(m),n}_{\tau_0,\tau_1} \bar{v}^{(m),n}_{\tau_1}(x)| \\ &\leq |P_{\tau_0,\tau_1}(v_{\tau_1}(x) - \bar{v}^{(m),n}_{\tau_1}(x))| + |(P_{\tau_0,\tau_1} - Q^{(m),n}_{\tau_0,\tau_1})\bar{v}^{(m),n}_{\tau_1}(x)|. \end{aligned} \tag{B.1}$$

By Lemma 1 and an inequality $\|P_{\tau_0,\tau_1} g\|_\infty \leq \|g\|_\infty$ for a bounded measurable function g, we have

$$\|v_0 - \bar{v}_0^{(m),n}\|_\infty \leq \|v_{\tau_1} - \bar{v}^{(m),n}_{\tau_1}\|_\infty + CK^{-1} n^{-m} \|\bar{v}^{(m),n}_{\tau_1}\|_\infty, \tag{B.2}$$

and also

$$\|v_{\tau_j} - \bar{v}^{(m),n}_{\tau_j}\|_\infty \leq \|v_{\tau_{j+1}} - \bar{v}^{(m),n}_{\tau_{j+1}}\|_\infty + CK^{-1} n^{-m} \|\bar{v}^{(m),n}_{\tau_{j+1}}\|_\infty, \tag{B.3}$$

for $j = 1, \ldots K-1$. Using $v_{\tau_K} = \bar{v}^{(m)}_{\tau_K}$, we get

$$\|v_0 - \bar{v}_0^{(m),n}\|_\infty \leq C \sum_{j=1}^{K} K^{-1} n^{-m} \|\bar{v}^{(m),n}_{\tau_j}\|_\infty. \tag{B.4}$$

Using the estimate

$$\|Q^{(m),n}_{\tau_j,\tau_{j+1}} g\|_\infty \leq \|g\|_\infty (1 + c/(nK))^n \leq \|g\|_\infty e^{c/K} \tag{B.5}$$

for a bounded measurable function g and the fact $\bar{v}^{(m),n}_{\tau_K} = f(\tau_K, \cdot)$, we have that there exists $C > 0$ which does not depend on n and K such that

$$\|\bar{v}^{(m),n}_{\tau_j}\|_\infty = \|Q^{(m),n}_{\tau_j,\tau_{j+1}} \bar{v}^{(m),n}_{\tau_{j+1}} \vee f(\tau_j, \cdot)\|_\infty \tag{B.6}$$

$$\leq e^{c/K} \|\bar{v}^{(m),n}_{\tau_{j+1}}\|_\infty \vee \|f(\tau_j, \cdot)\|_\infty \leq \cdots \leq C \max_{j \leq i \leq K} \|f(\tau_i, \cdot)\|_\infty, \tag{B.7}$$

for all $j = 1, \ldots, K$. Thus, we finally obtain

$$\|v_0 - \bar{v}_0^{(m),n}\|_\infty \leq C \sum_{j=1}^{K} K^{-1} n^{-m} \max_{j \leq i \leq K} \|f(\tau_i, \cdot)\|_\infty \leq C \max_{1 \leq j \leq K} \|f(\tau_j, \cdot)\|_\infty n^{-m}. \ \square \tag{B.8}$$

C Proof of Corollary 1

First, we follow the proof of Lemma 1. For $\varphi \in C_b^\infty(\mathbb{R}^q)$, $j = 0, 1, \ldots, K-1$ and $\ell = 1, \ldots, n$, we have

$$(P_{t_{jn+\ell-1}, t_{jn+\ell}} - Q^{\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}})\varphi(x) = R^{1,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}(x) + R^{2,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}(x), \tag{C.1}$$

where $R^{1,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}$ is a function satisfying there exists $C > 0$ independent of n, K and φ such that

$$\|R^{1,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}\|_\infty \leq C \|\varphi\|_\infty (nK)^{-2}, \tag{C.2}$$

and $R^{2,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}$ is a function which have the form:

$$R^{2,\varphi,\mathrm{EM}}_{t_{jn+\ell-1}, t_{jn+\ell}}(x) = \sum_{|\gamma| \leq 3} E[\partial^\gamma \varphi(\bar{X}^{t_{jn+\ell-1}, x}_{t_{jn+\ell}}) \Phi^{t_{jn+\ell-1}, t_{jn+\ell}, \mathrm{EM}}_\gamma(x)] \tag{C.3}$$

for some functions $\Phi_\gamma^{t_{jn+\ell-1},t_{jn+\ell},\mathrm{EM}} \in C_b^\infty(\mathbb{R}^q)$, $\gamma \in \{1,\dots,q\}^\eta$, $\eta \le 3$ such that for all $e \in \mathbb{N}\cup\{0\}$ and $\gamma \in \{1,\dots,q\}^\eta$, $\eta \le 3$,

$$\|\nabla^e \Phi_\gamma^{t_{jn+\ell-1},t_{jn+\ell},\mathrm{EM}}\|_\infty = O((nK)^{-2}). \tag{C.4}$$

Here, the following inequality can be proved similarly as in the proof of Lemma 1:

$$\sup_{x\in\mathbb{R}^q} |(P_{\tau_j,\tau_{j+1}} - Q^{\mathrm{EM},n}_{\tau_j,\tau_{j+1}})f(\tau_j,x)| \le C\|f(\tau_j,\cdot)\|_\infty K^{-1/2}n^{-1}. \tag{C.5}$$

We follow the proof of Theorem 1 with (C.5) instead of Lemma 1. Then, there exists $C>0$ dependent on K and independent of n such that

$$\|v_0 - \bar{v}_0^{\mathrm{EM},n}\|_\infty \le C \max_{1\le j\le K} \|f(\tau_j,\cdot)\|_\infty n^{-1}. \qquad \square \tag{C.6}$$

References

Akyıldırım, E., Dolinsky, Y., and Soner, H. M. (2014). Approximating stochastic volatility by recombinant trees. *Annals of Applied Probability* 24(5), 2176–2205.

Andersen, L., and Broadie, M. (2004). Primal-dual simulation algorithm for pricing multidimensional American options. *Management Science* 50(9), 1222–1234.

Andersson, K., and Oosterlee, C.W. (2021). Deep learning for CVA computations of large portfolios of financial derivatives. *Applied Mathematics and Computation* 409, 126399.

Andricopoulos, A. D., Widdicks, M., Duck, P. W., and Newton, D. P. (2003). Universal option valuation using quadrature methods. *Journal of Financial Economics* 67(3), 447–471.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. *Proceedings of the International Conference on Learning Representations*.

Bally, V., and Pagés, G. (2003). Error analysis of the optimal quantization algorithm for obstacle problems. *Stochastic Processes and their Applications* 106(1), 1–40.

Bally, V., Pagés, and G., Printems, J. (2005). A quantization tree method for pricing and hedging multidimensional American options. *Mathematical Finance* 15(1), 119–168.

Beck, C., Becker, S., Cheridito, P., Jentzen, A., and Neufeld, A. (2021). Deep splitting method for parabolic PDEs. *SIAM Journal on Scientific Computing* 43(5), 3135–3154.

Becker, S., Cheridito, P., and Jentzen, A. (2019). Deep optimal stopping. *Journal of Machine Learning Research*, 20(1), 2712-2736.

Becker, S., Cheridito, P., and Jentzen, A. (2020). Pricing and hedging American-style options with deep Learning. *Journal of Risk and Financial Management* 13(7), 158.

Becker, S., Cheridito, P., Jentzen, A., and Welti, T. (2021). Solving high-dimensional optimal stopping problems using deep learning. *European Journal of Applied Mathematics* 32(3), 470–514.

Belomestny, D. (2011). Pricing Bermudan options by nonparametric regression: optimal rates of convergence for lower estimates. *Finance and Stochastics* 15(4), 655–683.

Bouchard, B., Ekeland, I., and Touzi, N. (2004). On the Malliavin approach to Monte Carlo approximation of conditional expectations. *Finance and Stochastics* 8(1), 45–71.

Bouchard, B., and Touzi, N. (2004). Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their Applications* 111(2), 175–206.

Broadie, M., and Detemple, J. (1996). American option valuation: new bounds, approximations, and a comparison of existing methods. *The Review of Financial Studies* 9(4), 1211–1250.

Broadie, M., and Detemple, J. (2004). Anniversary article: Option pricing: Valuation models and applications. *Management science* 50(9), 1145–1177.

Broadie, M., and Glasserman, P. (1997). Pricing American-style securities using simulation. *Journal of Economic Dynamics and Control* 21(8–9), 1323–1352.

Broadie, M., and Glasserman, P. (2004). A stochastic mesh method for pricing high-dimensional American options. *Journal of Computational Finance* 7, 35–72.

Broadie, M., Glasserman, P., and Jain, G. (1997). Enhanced Monte Carlo estimates for American option prices. *Journal of Derivatives* 5(1), 25–44.

Broadie, M., and Yamamoto, Y. (2005). A double-exponential fast Gauss transform algorithm for pricing discrete path-dependent options. *Operations Research* 53(5), 764–779.

Calin, O. (2020). *Deep Learning Architectures*. Springer.

Chen, D., Härkönen, H. J., and Newton, D. P. (2014). Advancing the universality of quadrature methods to any underlying process for option pricing. *Journal of Financial Economics* 114(3), 600–612.

Chen, Y., and Wan, J. W. (2021). Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions. *Quantitative Finance* 21(1), 45–67.

Clément, E., Lamberton, D., and Protter, P. (2002). An analysis of a least squares regression method for American option pricing. *Finance and Stochastics* 6(4), 449–471.

Cox, J. C., Ross, S. A., and Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics* 7(3), 229–263.

Detemple, J. (2005). *American-style derivatives: Valuation and computation*. CRC Press.

Detemple, J. (2014). Optimal exercise for derivative securities. *Annual Review of Financical Economics* 6(1), 459–487.

E, W., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics* 5(4), 349–380.

Egloff, D. (2005). Monte Carlo algorithms for optimal stopping and statistical learning. *Annals of Applied Probability* 15(2), 1396–1432.

Fujii, M., Takahashi, A., and Takahashi, M. (2019). Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs. *Asia-Pacific Financial Markets* 26(3), 391–408.

Fang, F., and Oosterlee, C. W. (2011). A Fourier-based valuation method for Bermudan and barrier options under Heston's model. *SIAM Journal on Financial Mathematics* 2(1), 439–463.

Feng, L., and Lin, X. (2013). Pricing Bermudan options in Lévy process models. *SIAM Journal on Financial Mathematics* 4(1), 474–493.

Feng, L., and Linetsky, V. (2008a). Pricing options in jump-diffusion models: an extrapolation approach. *Operations Research* 56(2), 304–325.

Feng, L., and Linetsky, V. (2008b). Pricing discretely monitored barrier options and defaultable bonds in Lévy process models: a fast Hilbert transform approach. *Mathematical Finance* 18(3), 337–384.

Gagliardini, P., and Ronchetti, D. (2013). Semi-parametric estimation of American option prices. *Journal of Econometrics* 173(1), 57–82.

Glasserman, P., and Yu, B. (2004). Number of paths versus number of basis functions in American option pricing. *Annals of Applied Probability* 14(4), 2090–2119.

Gyurkó, L. G., and Lyons, T. (2010). Rough paths based numerical algorithms in computational finance. *Contemporary Mathematics*, AMS, 515, 17–46.

Hagan, P. S., Kumar, D., Lesniewski, A. S., and Woodward, D. E. (2002). Managing smile risk. *The Best of Wilmott* 1, 249–296.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences* 115(34), 8505–8510.

Haugh, M. B., and Kogan, L. (2004). Pricing American options: a duality approach, *Operations Research* 52(2), 258–270.

Huré, C., Pham, H., and Warin, X. (2020). Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation* 89(324), 1547–1579.

Ikeda, N., and Watanabe, S. (1989). *Stochastic Differential Equations and Diffusion Processes*. 2nd ed., North-Holland, Amsterdam, Kodansha, Tokyo.

Iguchi, Y., and Yamada, T. (2021). Operator splitting around Euler-Maruyama scheme and high order discretization of heat kernels. *ESAIM: Mathematical Modelling and Numerical Analysis* 55, 323–367.

Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning* 37, 448–456.

Kingma, D., and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv*, https://doi.org/10.48550/arXiv.1412.6980 .

Kirkby, J. L. (2015). Efficient option pricing by frame duality with the fast Fourier transform. *SIAM Journal on Financial Mathematics* 6(1), 713–747.

Kirkby, J. L. (2017). Robust barrier option pricing by frame projection under exponential Lévy dynamics. *Applied Mathematical Finance* 24(4), 337–386.

Kirkby, J. L. (2018). American and exotic option pricing with jump diffusions and other Lévy processes. *Journal of Computational Finance* 22(3), 89–148.

Kirkby, J. L., Nguyen, D., and Cui, Z. (2017). A unified approach to Bermudan and barrier options under stochastic volatility models with jumps. *Journal of Economic Dynamics and Control* 80, 75–100.

Kloeden, P. E., and Platen, E. (1999). *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin.

Kusuoka, S., and Stroock, D. (1984). Applications of Malliavin Calculus, Part I. *Proceedings of the Taniguchi International Symposium on Stochastic Analysis*, ed. by K. Itô, Kyoto and Katata, 1982, Kinokuniya, 271–360.

Lapeyre, B., Lelong, J. (2021). Neural network regression for Bermudan option pricing. *Monte Carlo Methods and Applications* 27(3), 227–247.

Li, C. (2014). Closed-form expansion, conditional expectation, and option valuation. *Mathematics of Operations Research* 39(2), 487–516.

Li, C., and Ye, Y. (2019). Pricing and exercising American options: an asymptotic expansion approach. *Journal of Economic Dynamics and Control* 107, 103729.

Li, L., and Linetsky, V. (2013). Optimal stopping and early exercise: an eigenfunction expansion approach. *Operations Research* 61(3), 625–643.

Liang, J., Xu, Z., and Li, P. (2021). Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing. *Quantitative Finance* 21(8), 1309–1323.

Longstaff, F. A., and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. *The Review of Financial Studies* 14(1), 113–147.

Malliavin, P., and Thalmaier, A. (2006). *Stochastic Calculus of Variations in Mathematical Finance*. Springer.

Maruyama, G. (1955). Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo* 4(1), 48–90.

Merton, R. C., Brennan, M. J., and Schwartz, E. S. (1977). The valuation of American put options. *Journal of Finance* 32(2), 449–462.

Naito, R., and Yamada, T. (2019). A third-order weak approximation of multidimensional Itô stochastic differential equations. *Monte Carlo Methods and Applications* 25(2), 97–120.

Naito, R., and Yamada, T. (2022). A higher order weak approximation of McKean-Vlasov type SDEs. *BIT Numerical Mathematics* 62(2), 521–559.

Naito, R., and Yamada, T. (2022). A deep learning-based high-order operator splitting method for high-dimensional nonlinear parabolic PDEs via Malliavin calculus: application to CVA computation. *2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)* 1–8, IEEE.

Nelson, D. B., and Ramaswamy, K. (1990). Simple binomial processes as diffusion approximations in financial models. *The Review of Financial Studies* 3(3), 393–430.

Nourdin, I., and Peccati, G. (2012). *Normal Approximations with Malliavin Calculus*. Cambridge University Press.

Nualart, D. (2006). *The Malliavin Calculus and Related Topics*. Springer.

Reisinger, C., and Wissmann, R. (2015). Numerical valuation of derivatives in high-dimensional settings via partial differential equation expansions. *Journal of Computational Finance* 18(4), 95–127.

Ruijter, M. J., and Oosterlee, C. W. (2016). Numerical Fourier method and second-order Taylor scheme for backward SDEs in finance. *Applied Numerical Mathematics* 103, 1–26.

Rogers, L. C. (2002). Monte Carlo valuation of American options. *Mathematical Finance* 12(3), 271–286.

Schwartz, E. S. (1977). The valuation of warrants: Implementing a new approach. *Journal of Financial Economics* 4(1), 79–93.

Sirignano, J., and Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics* 375, 1339–1364.

Takahashi, A., Tsuchida, Y., and Yamada, T. (2022). A new efficient approximation scheme for solving high-dimensional semilinear PDEs: control variate method for Deep BSDE solver. *Journal of Computational Physics* 454, 110956.

Takahashi, A., and Yamada, T. (2012). An asymptotic expansion with push-down of Malliavin weights. *SIAM Journal on Financial Mathematics* 3(1), 95–136.

Takahashi, A., and Yamada, T. (2015). On error estimates for asymptotic expansions with Malliavin weights: Application to stochastic volatility model. *Mathematics of Operations Research* 40(3), 513–541.

Takahashi, A., and Yamada, T. (2016). A weak approximation with asymptotic expansion and multidimensional Malliavin weights. *Annals of Applied Probability* 26(2), 818–856.

Tsitsiklis, J. N., and Van Roy, B. (2001). Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks* 12(4), 694–703.

Van der Have, and Z., Oosterlee, C. W. (2018). The COS method for option valuation under the SABR dynamics. *International Journal of Computer Mathematics* 95(2), 444–464.

Watanabe, S. (1987). Analysis of Wiener functionals (Malliavin calculus) and its application to heat kernels. *Annals of Probability* 15(1), 1–39.

Yamada, T. (2019). An arbitrary high order weak approximation of SDE and Malliavin Monte Carlo: analysis of probability distribution functions. *SIAM Journal on Numerical Analysis* 57(2), 563–591.

Yamada, T. (2021). High order weak approximation for irregular functionals of time-inhomogeneous SDEs. *Monte Carlo Methods and Applications* 27(2), 117–136.

Yamada, T. (2022). A Gaussian Kusuoka-approximation without solving random ODEs. *SIAM Journal on Financial Mathematics* 13 (1), SC1-11.

Yamada, T., and Yamamoto, K. (2019). Second order discretization of Bismut-Elworthy-Li formula: application to sensitivity analysis. *SIAM/ASA Journal on Uncertainty Quantification* 7(1), 143–173.

Yamada, T., and Yamamoto, K. (2020). A second order discretization with Malliavin weight and Quasi Monte Carlo method for option pricing. *Quantitative Finance* 20(11), 1825–1837.

Zeng, P., and Kwok, Y. K. (2014). Pricing barrier and Bermudan style options under time-changed Lévy processes: fast Hilbert transform approach. *SIAM Journal on Scientific Computing* 36(3), B450–B485.