Gradient-enhanced sparse Hermite polynomial expansions for pricing and hedging high-dimensional American options*

Jiefei Yang† and Guanglian Li‡

Abstract. We propose an efficient and easy-to-implement gradient-enhanced least squares Monte Carlo method for computing price and Greeks (i.e., derivatives of the price function) of high-dimensional American options. It employs the sparse Hermite polynomial expansion as a surrogate model for the continuation value function, and essentially exploits the fast evaluation of gradients. The expansion coefficients are computed by solving a linear least squares problem that is enhanced by gradient information of simulated paths. We analyze the convergence of the proposed method, and establish an error estimate in terms of the best approximation error in the weighted H^1 space, the statistical error of solving discrete least squares problems, and the time step size. We present comprehensive numerical experiments to illustrate the performance of the proposed method. The results show that it outperforms the state-of-the-art least squares Monte Carlo method with more accurate price, Greeks, and optimal exercise strategies in high dimensions but with nearly identical computational cost, and it can deliver comparable results with recent neural network-based methods up to dimension 100.

Key words. sparse Hermite polynomial expansion, least squares Monte Carlo, backward stochastic differential equation, high dimensions, American option

MSC codes. 60G40, 91G60, 91G20, 62J05, 65C30

1. Introduction.

The early exercise feature of American or Bermudan options gives holders the right to buy (call) or sell (put) underlying assets before the expiration date, and their accurate numerical calculation is of great practical importance. Meanwhile, the efficient estimation of Greeks (i.e., derivatives of the price function, e.g., delta and gamma) is vital for hedging and risk management, since the theory of option pricing builds on the assumption of the absence of arbitrage. For example, when the asset price is on the rise, the gain in the long position of a call writer's asset may offset the potential loss of the call option.

Nonetheless, the early exercise feature of American options poses significant challenges for computing price and Greeks, especially in high dimensions. One of the most popular methods for high-dimensional American option pricing is the least squares Monte Carlo (LSM) method [15, 22]. Computing Greeks in high dimensions is more involved, and further developments with LSM have been proposed [24, 5]. In this work, building on LSM, we shall develop a simple, fast, and accurate algorithm, termed as gradient-enhanced least squares Monte Carlo (G-LSM) method, c.f. Algorithm 4.1, for computing price and Greeks simultaneously at all time steps for dimensions up to 100. The key methodological innovations includes using sparse Hermite polynomial space with a hyperbolic cross index set as the ansatz space for approximating the continuation value functions (CVFs), and incorporating the gradient

*Submitted to the editors May XX, 2024.

Funding: JY acknowledges support from the University of Hong Kong via the HKU Presidential PhD Scholar Programme (HKU-PS). GL acknowledges the support from GRF (project number: 17317122) and Early Career Scheme (Project number: 27301921), RGC, Hong Kong.

†Department of Mathematics, University of Hong Kong, Pokfulam, Hong Kong (jiefeiy@connect.hku.hk).

‡Department of Mathematics, University of Hong Kong, Pokfulam, Hong Kong (lotusli@maths.hku.hk).

information for computing the expansion coefficients.

We elaborate on the two methodological innovations. First, the main obstacle of using gradient information lies in the computational expense: a d-variate price has d partial derivatives, which grows quickly with d, especially when the derivative evaluation is costly. With the proposed sparse Hermite polynomial ansatz space, the derivatives of polynomial bases can be obtained at almost no extra cost, cf. (3.3) below. This allows greatly reducing the computational cost. Second, although using a polynomial ansatz space for the CVFs as LSM, G-LSM constructs the expansion coefficients via solving a linear least squares problem enhanced by the gradient (and hence the name G-LSM), by minimizing the mean squared error between the approximate and exact value functions at t_{k+1}, c.f. (4.6) below. This differs markedly from LSM, which approximates the conditional expectations by projection and minimizes the mean squared error of approximating CVF at t_k. Numerical experiments show that this choice can achieve better accuracy in price, Greeks, and optimal exercise strategies than LSM.

In G-LSM, the approximation of the terminal condition at t_{k+1} is obtained by discretizing the linear backward stochastic differential equation (BSDE) for the CVF, c.f. Theorem 4.1, which was recently innovated in a deep neural network-based method for American option pricing [6]. The idea of matching the terminal condition has been widely applied in solving high-dimensional BSDEs with deep neural networks (DNNs) [7, 12]. In practice, it involves computing the gradient of the CVF, and in turn that of the basis functions in the ansatz space (in addition to function evaluation). When $d \gg 1$, for a complicated ansatz space, evaluating the derivatives at all time steps can be prohibitive. We shall show that the extra cost is nearly negligible for the sparse Hermite polynomial ansatz, and that the overall complexity of G-LSM with N time steps, M sample paths, and N_b basis functions is $\mathcal{O}(NMN_b)$, nearly identical with that for LSM. Numerical results show that the accuracy of G-LSM is competitive with DNN-based methods for dimensions up to $d = 100$.

In theory, the CVF can be formulated as a smooth, high-dimensional function in $L^2_\omega(\mathbb{R}^d)$ with a Gaussian weight function $\omega(\mathbf{y})$ [25]. This regularity enables the use of normalized and generalized Hermite polynomials, which form an orthonormal basis of $L^2_\omega(\mathbb{R}^d)$. Furthermore, drawing on the geometric convergence rate of the hyperbolic cross approximation with Hermite polynomials [17], we shall prove the global convergence of G-LSM using BSDE technique, stochastic and Malliavin calculus, and establish an error bound in terms of time step size, statistical error of the Monte Carlo approximation, and the best approximation error in weighted Sobolev space $H^1_\omega(\mathbb{R}^d)$, c.f. Theorem 5.6. In sum, the algorithmic development of G-LSM, its error analysis and extensive numerical evaluation represent the main contributions of the present work.

Now we situate the present study in existing works. Currently, there are two popular classes of methods to price American options in high dimensions: (i) least-squares Monte Carlo-based (LSM) methods and (ii) DNN-based methods. The LSM method has shown tremendous success for pricing American or Bermudan options with more than one stochastic factors. The original LSM [15] uses polynomials to approximate the CVF, and other choices have also been explored, e.g., Gaussian process [16] and DNNs [14, 4]. Recently, LSM with the hierarchical tensor train technique has been studied in [2], which demonstrates the success of polynomial approximation for CVFs in very high dimensions. The proposed G-LSM is a variant of LSM that incorporates gradient information that comes nearly for free. Due

to the excellent capability for high-dimensional approximation of DNNs, several methods based on DNNs have been proposed for pricing American or Bermudan options, based on optimal stopping problem (parameterizing the stopping time by DNNs and then maximizing the expected reward [3]), free boundary PDEs (parameterizing PDE solutions with DNNs [21]), or BSDEs (parameterizing the solution pair of the associated reflected BSDE [8] by DNNs [12]). Within the framework of BSDEs, Chen and Wan [6] suggest approximating the difference of the CVF between adjacent time steps by averaging several trained neural networks, which has a quadratic complexity in the number of time steps. Wang et al. [23] extend the deep BSDE method [7] from European option pricing to Bermudan one, with the loss function being the variance of the initial value, and Gao et al. [10] analyze its convergence. In comparison with DNN-based methods, G-LSM enjoys high efficiency and robustness, which involves only least-squares problems and is easy to implement.

The structure of this article is organized as follows. In section 2, we describe the mathematical framework of pricing and hedging high-dimensional American or Bermudan options, and in section 3, we recall several useful properties of generalized Hermite polynomials and approximation with sparse hyperbolic cross index set. Then in section 4, we derive the main algorithm, i.e., gradient-enhanced least squares Monte Carlo (G-LSM) method, and establish its local and global error estimates in section 5. In section 6, we present extensive numerical results including prices, Greeks, optimal stopping time, and computing time. We also present a comparative study with existing methods. Finally, we conclude in section 7 with further discussions.

Throughout, bold and plain letters represent multi-variable and scalars, respectively, and the capital and bold letters $\mathbf{S}$, $\tilde{\mathbf{X}}$ and $\mathbf{W}$ denote random vectors. The notation $\mathbf{a}\cdot\mathbf{b}$ denotes the dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathrm{Tr}(A)$ the trace of a square matrix A, and $^\top$ vector transpose. The notation $\nabla_{\mathbf{x}} f(t,\mathbf{x})$ and $\mathrm{Hess}_{\mathbf{x}} f(t,\mathbf{x})$ denote respectively the gradient and Hessian of f with respect to $\mathbf{x}$. For multi-index $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_d)^\top\in\mathbb{N}_0^d$, $|\boldsymbol{\alpha}|=\alpha_1+\cdots+\alpha_d$. For a positive-valued integrable function $\omega:\mathbb{R}^d\to\mathbb{R}^+$, the weighted space $L^2_\omega(\mathbb{R}^d)$ is defined by

$$L^2_\omega(\mathbb{R}^d):=\{f:\mathbb{R}^d\to\mathbb{R}:\|f\|_{L^2_\omega(\mathbb{R}^d)}<\infty\},\qquad \text{with } \|f\|^2_{L^2_\omega(\mathbb{R}^d)}:=\int_{\mathbb{R}^d} f(\mathbf{x})^2\omega(\mathbf{x})\mathrm{d}\mathbf{x}.$$

The weighted Sobolev space $H^m_\omega(\mathbb{R}^d)$, $m\in\mathbb{N}$, is defined by

$$H^m_\omega(\mathbb{R}^d):=\{f:\mathbb{R}^d\to\mathbb{R}:\|f\|_{H^m_\omega(\mathbb{R}^d)}<\infty\},\quad \text{with } \|f\|^2_{H^m_\omega(\mathbb{R}^d)}=\sum_{0\le|\boldsymbol{\alpha}|\le m}\left\|\frac{\partial^{\boldsymbol{\alpha}} f}{\partial\mathbf{x}^{\boldsymbol{\alpha}}}\right\|^2_{L^2_\omega(\mathbb{R}^d)}.$$

2. Bermudan option pricing and hedging.

Now we describe the valuation framework for American or Bermudan option pricing and hedging.

2.1. Option pricing and Greeks.

The fair price of American option $v^A(t)$ at time $t\in[0,T]$ is expressed as the solution to the optimal stopping problem in a risk neutral probability space $(\Omega,\mathcal{F},(\mathcal{F}_t)_{0\le t\le T},\mathbb{Q})$,

$$v^A(t)=\sup_{\tau_t\in[t,T]}\mathbb{E}[e^{-r(\tau_t-t)}g(\mathbf{S}_{\tau_t})|\mathcal{F}_t],$$

where τ_t is an $\mathcal{F}_t$-stopping time, $T > 0$ is the expiration date, $(\mathbf{S}_t)_{0\le t\le T}$ is a collection of d-dimensional price processes, and $g(\mathbf{S}_t) \in L^2(\Omega, \mathcal{F}_t, \mathbb{Q})$ is the payoff depending on the type of the option.

Numerically, the price of Bermudan option is used to approximate the American one. The Bermudan option can be exercised at finite discrete times $0 = t_0 < t_1 < \cdots < t_N = T$ with $\Delta t := t_{k+1} - t_k$ for all $k = 0, 1, \ldots, N-1$. Using dynamic programming principle or Snell envelope theory [20, Section 1.8.4], the Bermudan price function v_{t_k} at time t_k is given by the following backward induction:

$$
\begin{aligned}
&v_{t_N}(\mathbf{s}) = g_{t_N}(\mathbf{s}),\\
&v_{t_k}(\mathbf{s}) = \begin{cases} g_{t_k}(\mathbf{s}), & \text{if } g_{t_k}(\mathbf{s}) \ge \tilde{c}_{t_k}(\mathbf{s}),\\ \tilde{c}_{t_k}(\mathbf{s}), & \text{if } g_{t_k}(\mathbf{s}) < \tilde{c}_{t_k}(\mathbf{s}), \end{cases} \qquad \text{for } k = N-1:-1:0,
\end{aligned} \tag{2.1}
$$

where g_{t_k} is the discounted payoff (exercise value) function and $\tilde{c}_{t_k}$ is the CVF, defined by

$$
\tilde{c}_{t_k}(\mathbf{s}) = \mathbb{E}[v_{t_{k+1}}(\mathbf{S}_{t_{k+1}})|\mathbf{S}_{t_k} = \mathbf{s}]. \tag{2.2}
$$

The options delta and gamma are defined to be the first and second order derivatives of the price function v_{t_k} with respect to the price of underlying assets, i.e.,

$$
\Delta_{t_k} := \nabla v_{t_k}(\mathbf{s}) = \left(\frac{\partial v_{t_k}(\mathbf{s})}{\partial s_1}, \ldots, \frac{\partial v_{t_k}(\mathbf{s})}{\partial s_d}\right)^\top \quad \text{and} \quad \Gamma_{t_k}^{ij} := \frac{\partial^2 v_{t_k}(\mathbf{s})}{\partial s_j \partial s_i}. \tag{2.3}
$$

We consider all exercise and continuation values of Bermudan option discounted to the time $t = 0$.

2.2. Multi-asset model and transformation. One of the most classical models for high-dimensional American option pricing is the multi-asset Black-Scholes model. Under the risk-neutral probability $\mathbb{Q}$, the prices of d underlying assets, $\mathbf{S}_t = \left(S_t^1, \ldots, S_t^d\right)^\top$, follow the correlated geometric Brownian motions

$$
\mathrm{d}S_t^i = (r - \delta_i)S_t^i\,\mathrm{d}t + \sigma_i S_t^i\,\mathrm{d}\tilde{W}_t^i \quad \text{with } S_0^i = s_0^i, \quad i = 1, 2, \ldots, d, \tag{2.4}
$$

where $\tilde{W}_t^i$ are correlated Brownian motions with correlation $\mathbb{E}[\mathrm{d}\tilde{W}_t^i \mathrm{d}\tilde{W}_t^j] = \rho_{ij}\,\mathrm{d}t$, and r, δ_i and σ_i are the riskless interest rate, dividend yields, and volatility parameters, respectively. We denote the correlation matrix by $P = (\rho_{ij})_{d\times d}$, the volatility matrix by Σ (which is a diagonal matrix with volatility σ_i on the diagonal), and write the dividend yields as a vector $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_d]^\top$. Using the spectral decomposition $\Sigma P \Sigma^\top = Q \Lambda Q^\top$, the rotated log-price $\tilde{\mathbf{X}}_t := Q^\top \ln(\mathbf{S}_t./\mathbf{s}_0)$ satisfies an independent Gaussian distribution

$$
\tilde{\mathbf{X}}_t \sim \mathcal{N}\left(Q^\top\left(r - \boldsymbol{\delta} - \frac{1}{2}\Sigma^2\mathbf{1}\right)t, \Lambda t\right).
$$

Let $\boldsymbol{\mu} = Q^\top\left(r - \boldsymbol{\delta} - \frac{1}{2}\Sigma^2\mathbf{1}\right)$ and λ_i be the i-th diagonal element of Λ. This gives a transformation between underlying asset prices $\mathbf{S}_t$ and independent Brownian motions $\mathbf{W}_t = [W_t^1, \ldots, W_t^d]^\top$, i.e.,

$$
\mathbf{S}_t = \mathbf{s}_0 \odot \exp\left(Q\left(\boldsymbol{\mu}t + \sqrt{\Lambda}\mathbf{W}_t\right)\right), \tag{2.5}
$$

where $\odot$ denotes componentwise product.

From (2.5) and (2.2), the CVF c_{t_k} with respect to the independent Brownian motions is

$$c_{t_k}(\mathbf{w}) = \mathbb{E}[u_{t_{k+1}}(\mathbf{W}_{t_{k+1}})|\mathbf{W}_{t_k} = \mathbf{w}], \tag{2.6}$$

where $u_{t_{k+1}}$ represents the value function at time t_{k+1} with respect to the independent Brownian motions. Our aim is to develop a gradient-enhanced least squares Monte Carlo method that can efficiently approximate c_{t_k}, and thus provide accurate prices and their derivatives.

3. Sparse Hermite polynomial expansion and gradient. Sparse polynomial chaos expansion can serve as a surrogate model of unknown stochastic variables with finite second-order moments. The motivations of using sparse Hermite polynomial expansion for pricing and hedging American options are twofold:

1. Let ω_t, $t > 0$, be the Gaussian density function defined by $\omega_t(\mathbf{y}) := \prod_{j=1}^d \rho_t(y_j)$ with $\rho_t(y) := \frac{1}{\sqrt{2\pi t}} \exp(-\frac{y^2}{2t})$. The CVF c_{t_k} satisfies $c_{t_k} \in L^2_{\omega_{t_k}}(\mathbb{R}^d)$, since the payoff $g(\mathbf{S}_{t_k}) \in L^2(\Omega, \mathcal{F}_{t_k}, \mathbb{Q})$ has a finite second-order moment. Thus, as an orthonormal basis for $L^2_{\omega_{t_k}}(\mathbb{R}^d)$, the set of normalized and generalized Hermite polynomials emerges as a natural choice.
2. The CVF $c_{t_k} \in H^m_{\omega_{t_k}}(\mathbb{R}^d)$ for any positive integer m by [25, Lemmas 4.2 and 4.3]. The smoothness of the function implies efficient polynomial approximation.

Next, we recall one-dimensional normalized and generalized Hermite polynomials $H_n^{(t)}(y)$ and the tensorized d-dimensional polynomials $H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y})$, $\boldsymbol{\alpha} \in \mathbb{N}_0^d$. For $d = 1$, using the standard Hermite polynomials H_n, we define the nth-order normalized and generalized Hermite polynomial $H_n^{(t)}$ by

$$H_n^{(t)}(y) := \frac{H_n(\frac{y}{\sqrt{t}})}{\sqrt{n!}}, \qquad \text{for } n \in \mathbb{N}_0, t > 0, y \in \mathbb{R}.$$

Then $\{H_n^{(t)}\}_{n\in\mathbb{N}_0}$ forms a complete orthonormal basis for $L^2_{\rho_t}(\mathbb{R})$:

$$\mathbb{E}\left[H_n^{(t)}(W_t)H_m^{(t)}(W_t)\right] = \int_{\mathbb{R}} H_n^{(t)}(y)H_m^{(t)}(y)\rho_t(y)\,\mathrm{d}y = \delta_{nm},$$

with δ_{nm} being the Kronecker delta. For $d > 1$, the tensorized Hermite polynomial with the multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^\top \in \mathbb{N}_0^d$ defined by

$$H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y}) := \prod_{j=1}^d H_{\alpha_j}^{(t)}(y_j)$$

forms a complete orthonormal basis for $L^2_{\omega_t}(\mathbb{R}^d)$, satisfying

$$\mathbb{E}\left[H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{W}_t)H_{\boldsymbol{\gamma}}^{(t)}(\mathbf{W}_t)\right] = \int_{\mathbb{R}^d} H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y})H_{\boldsymbol{\gamma}}^{(t)}(\mathbf{y})\omega_t(\boldsymbol{y})\,\mathrm{d}\mathbf{y} = \delta_{\boldsymbol{\alpha},\boldsymbol{\gamma}}, \quad \boldsymbol{\alpha},\boldsymbol{\gamma} \in \mathbb{N}_0^d.$$

Here, $\delta_{\boldsymbol{\alpha},\boldsymbol{\gamma}} = \prod_{j=1}^d \delta_{\alpha_j,\gamma_j}$ and $\omega_t(\boldsymbol{y}) = \prod_{j=1}^d \rho_t(y_j)$.

For any fixed multi-index set $I \subset \mathbb{N}_0^d$, the Hermite polynomial ansatz space $P_{I,k}$ is defined by

$$P_{I,k} := \operatorname{span}\left\{ H_{\boldsymbol{\alpha}}^{(t_k)} : \boldsymbol{\alpha} \in I \right\}. \tag{3.1}$$

Then we aim to approximate the CVF c_{t_k} in $P_{I,k}$ by

$$c_{t_k}(\mathbf{W}_{t_k}) \approx \sum_{\boldsymbol{\alpha}\in I} \beta_{\boldsymbol{\alpha}} H_{\boldsymbol{\alpha}}^{(t_k)}(\mathbf{W}_{t_k}).$$

It is well-known that computing polynomial approximations in high dimensions suffers from the notorious curse of dimensionality with tensor product-type multi-index sets. Fortunately, the smoothness of c_{t_k} implies a fast decay of the coefficients in the polynomial expansion. The large coefficients usually occur in a lower multi-index set I, that is, if $\boldsymbol{\alpha} \in I$ and $\boldsymbol{\gamma} \le \boldsymbol{\alpha}$, then $\boldsymbol{\gamma} \in I$ [1, Section 1.5]. To circumvent the curse of dimensionality, the decay property of the expansion coefficients can be exploited and a hyperbolic cross sparse index set can be used to construct an approximation. The hyperbolic cross multi-index set I with maximum order $p \in \mathbb{N}$ is defined by

$$I := \left\{ \boldsymbol{\alpha} = (\alpha_j)_{j=1}^d \in \mathbb{N}_0^d : \prod_{j=1}^d \max(\alpha_j, 1) \le p \right\}, \tag{3.2}$$

which has a cardinality $\mathcal{O}(p(\ln p)^{d-1})$. The best approximation error of the sparse Hermite polynomial approximation with hyperbolic cross index set was analyzed in [17]; see section 5 for details.

Now, we introduce a property of derivatives of Hermite polynomials, which plays an important role in reducing the computational cost. The first-order derivative of the one-dimensional normalized and generalized Hermite polynomial $H_n^{(t)}(y)$ satisfies

$$\frac{\mathrm{d}}{\mathrm{d}y}\left(H_n^{(t)}(y) \right) = \sqrt{\frac{n}{t}} H_{n-1}^{(t)}(y).$$

For the d-dimensional Hermite polynomial, we have

$$\frac{\partial}{\partial y_j}\left(H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y}) \right) = \sqrt{\frac{\alpha_j}{t}} H_{\boldsymbol{\alpha}-\mathbf{e}_j}^{(t)}(\mathbf{y}), \tag{3.3}$$

where $\mathbf{e}_j$ is the j-th canonical basis vector. This implies that for a lower multi-index set $I \subset \mathbb{N}_0^d$, we have $\boldsymbol{\alpha} - \mathbf{e}_j \in I$ if $\boldsymbol{\alpha} \in I$. Thus, once the evaluations of $H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y})$ for $\boldsymbol{\alpha} \in I$ are available, the gradients of $H_{\boldsymbol{\alpha}}^{(t)}(\mathbf{y})$ for $\boldsymbol{\alpha} \in I$ can be evaluated cheaply.

4. Algorithm and complexity. Now we derive the main methodology to approximate the CVF c_{t_k} by matching values of $u_{t_{k+1}}$, and analyze its computational complexity. Below we abbreviate the notations c_{t_k}, u_{t_k} and $\mathbf{W}_{t_k}$ to c_k, u_k and $\mathbf{W}_k$, etc, for $k = 0, 1, \ldots, N$.

4.1. Gradient-enhanced Least Squares (G-LS). First we derive a linear backward stochastic differential equation (BSDE) for the CVF c_k.

Theorem 4.1. *The CVF $c_k(\mathbf{W}_k)$ satisfies the linear BSDE*

$$c_k(\mathbf{W}_k) = u_{k+1}(\mathbf{W}_{k+1}) - \int_{t_k}^{t_{k+1}} \nabla_{\mathbf{w}} f(t, \mathbf{W}_t) \cdot \mathrm{d}\mathbf{W}_t,$$

where $f(t,\mathbf{w}), t \in [t_k, t_{k+1}]$, is defined by $f(t,\mathbf{w}) := \mathbb{E}[u_{k+1}(\mathbf{W}_{k+1})|\mathbf{W}_t = \mathbf{w}]$.

Proof. This is a direct consequence of martingale representation theorem [26, Theorem 2.5.2]. For the sake of completeness, we provide a brief proof. By the Feynman-Kac formula, $f(t,\mathbf{w})$ satisfies a d-dimensional parabolic PDE subject to a terminal condition

$$\begin{cases} \dfrac{\partial f}{\partial t}(t,\mathbf{w}) + \dfrac{1}{2}\operatorname{Tr}\left(\operatorname{Hess}_{\mathbf{w}} f(t,\mathbf{w})\right) = 0, \quad t \in [t_k, t_{k+1}), \\ f(t_{k+1},\mathbf{w}) = u_{t_{k+1}}(\mathbf{w}). \end{cases} \tag{4.1}$$

Let $Y_t := f(t,\mathbf{W}_t)$. By Itô's formula, we have

$$\mathrm{d}Y_t = \left(\frac{\partial f}{\partial t}(t,\mathbf{W}_t) + \frac{1}{2}\operatorname{Tr}\left(\operatorname{Hess}_{\mathbf{w}} f(t,\mathbf{W}_t)\right)\right)\mathrm{d}t + \nabla_{\mathbf{w}} f(t,\mathbf{W}_t)\cdot \mathrm{d}\mathbf{W}_t.$$

In view of (4.1), the drift term vanishes. After taking the stochastic integral and using the terminal condition in (4.1), we obtain the desired assertion. ∎

Next, we construct an approximation of the CVF c_k by matching the terminal condition over the interval $[t_k, t_{k+1}]$. By Theorem 4.1, the terminal value over $[t_k, t_{k+1}]$ can be approximated by the Euler discretization, i.e.,

$$\bar{u}_{k+1}(\mathbf{W}_{k+1}) = c_k^{\mathrm{CLS}}(\mathbf{W}_k) + \nabla c_k^{\mathrm{CLS}}(\mathbf{W}_k)\cdot \Delta\mathbf{W}_k, \tag{4.2}$$

where $\Delta\mathbf{W}_k := \mathbf{W}_{k+1} - \mathbf{W}_k$ is the Brownian increment and c_k^{CLS} denotes an approximation to the CVF c_k. Let $\hat{u}_{k+1}$ be the value function computed in the last time step. Then using $P_{I,k}$ defined in (3.1) and (3.2) as the ansatz space for c_k^{CLS}, we solve for c_k^{CLS} using the least squares regression:

$$c_k^{\mathrm{CLS}}(\mathbf{W}_k) = \operatorname*{argmin}_{\psi \in P_{I,k}} E_k(\psi), \tag{4.3}$$

with $E_k(\cdot) : P_{I,k} \to \mathbb{R}^+$ being the quadratic loss (i.e., mean squared error) defined by

$$E_k(\psi) := \mathbb{E}\left[\left(\hat{u}_{k+1}(\mathbf{W}_{k+1}) - \psi(\mathbf{W}_k) - \nabla\psi(\mathbf{W}_k)\cdot\Delta\mathbf{W}_k\right)^2\right].$$

Finally, the numerical value $\hat{u}_k$ at time t_k is updated to be the discounted exercise or continuation value at t_k:

$$\hat{u}_k(\mathbf{W}_k) = \begin{cases} g_k(\mathbf{S}_k), & \text{if exercise,} \\ c_k^{\mathrm{CLS}}(\mathbf{W}_k), & \text{if continue.} \end{cases} \tag{4.4}$$

Since the option is only profitable when exercised in the in-the-money region $\Omega_{\mathrm{ITM}} = \{\mathbf{s} \in \mathbb{R}^d : g_k(\mathbf{s}) > 0\}$, we make the decision of exercising the option when $c_k^{\mathrm{CLS}}(\mathbf{W}_k) < g_k(\mathbf{S}_k)$ and $\mathbf{S}_k \in \Omega_{\mathrm{ITM}}$; otherwise, the option will be continued.

4.2. Gradient-enhanced Least Squares Monte Carlo (G-LSM). Now we derive the methodology based on the Monte Carlo method to solve (4.3) numerically and analyze its computational complexity. Let $N_b = |I|$ and let $\{\phi_n^k(\mathbf{W}_k)\}_{n=1}^{N_b}$ be the set of Hermite polynomials in a scalar-indexed form. Then any function $\psi \in P_{I,k}$ can be expressed as

$$\psi(\mathbf{W}_k) = \sum_{n=1}^{N_b} \beta_n \phi_n^k(\mathbf{W}_k),$$

with its gradient given by

$$\nabla\psi(\mathbf{W}_k) = \sum_{n=1}^{N_b} \beta_n \nabla\phi_n^k(\mathbf{W}_k). \tag{4.5}$$

In practice, the continuous least squares problem (4.3) is solved by minimizing its Monte Carlo approximation:

$$\hat{c}_k := \underset{\psi\in P_{I,k}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \left(\hat{u}_{k+1}(\mathbf{W}_{k+1}^m) - \psi(\mathbf{W}_k^m) - \nabla\psi(\mathbf{W}_k^m)\cdot\Delta\mathbf{W}_k^m\right)^2, \tag{4.6}$$

where $\{\mathbf{W}_k^m\}_{m=1}^M$ are M independent paths of the Gaussian random process $\mathbf{W}_k$ and $\Delta\mathbf{W}_k^m := \mathbf{W}_{k+1}^m - \mathbf{W}_k^m$ is the mth path of increment $\Delta\mathbf{W}_k$. Let $A_k \in \mathbb{R}^{M\times N_b}$, $\boldsymbol{\beta}_k \in \mathbb{R}^{N_b}$ and $\hat{\mathbf{u}}_{k+1} \in \mathbb{R}^M$ with their components defined by

$$(A_k)_{mn} = \phi_n^k(\mathbf{W}_k^m) + \nabla\phi_n^k(\mathbf{W}_k^m)\cdot\Delta\mathbf{W}_k^m,$$
$$(\boldsymbol{\beta}_k)_n = \beta_n, \quad (\hat{\mathbf{u}}_{k+1})_m = \hat{u}_{k+1}(\mathbf{W}_{k+1}^m)$$

for $m = 1,\dots,M$, $n = 1,\dots,N_b$.

Then finding the optimal polynomial in (4.6) amounts to solving the classical least squares problem

$$\boldsymbol{\beta}_k = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|A_k\boldsymbol{\beta} - \hat{\mathbf{u}}_{k+1}\|_2^2 = (A_k^\top A_k)^{-1} A_k^\top \hat{\mathbf{u}}_{k+1}.$$

The proposed algorithm is summarized in Algorithm 4.1.

Remark 4.2. Note that the well-known least squares Monte Carlo (LSM) method [15, 22] also approximates the CVF with a finite number of basis functions. The orthonormal Hermite polynomials is one possible choice. However, LSM computes the coefficients by projecting the value, $u_{t_{k+1}}(\mathbf{W}_{t_{k+1}})$ in (2.6), onto a finite-dimensional space spanned by the basis functions due to the projection nature of conditional expectation:

$$\hat{c}_k^{LSM} := \underset{\psi\in P_{I,k}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \left(\hat{u}_{k+1}(\mathbf{W}_{k+1}^m) - \psi(\mathbf{W}_k^m)\right)^2.$$

In contrast, G-LSM computes the coefficients by matching the approximate and exact value of $u_{t_{k+1}}(\mathbf{W}_{t_{k+1}})$, see (4.6). Their numerical performance will be compared in section 6.

Algorithm 4.1 G-LSM

Input: Market parameters: $\mathbf{S}_0, r, \delta_i, \sigma_i, P$
Option parameters: payoff function $g(\mathbf{s})$
Algorithm parameters: N, M, p

Output: Option price v_0

1: Compute the hyperbolic cross multi-index set I with maximum order p
2: Generate M sample paths
3: Initialize values $\hat{\mathbf{u}}_N = (e^{-rT} g(\mathbf{S}_N^m))_{m=1}^M$
4: Initialize stopping times $\tau_\star = T$
5: **for** $k = N-1 : -1 : 1$ **do**
6: $\quad \Phi_k =$ basis matrix$(\mathbf{W}_k, I)$ with $(\Phi_k)_{m,n} = \phi_n^k(\mathbf{W}_k^m)$
7: $\quad$ Compute matrix A_k with Algorithm 4.2
8: $\quad$ Solve system of linear equations: $A_k \boldsymbol{\beta}_k = \mathbf{u}_{k+1}$
9: $\quad$ Update $(\hat{\mathbf{u}}_k)_m = \begin{cases} e^{-rk\Delta t} g(\mathbf{S}_k^m), & \text{if exercise} \\ (\Phi_k \boldsymbol{\beta}_k)_m, & \text{if continue} \end{cases}$ and $\tau_\star = \begin{cases} k\Delta t, & \text{if exercise} \\ \tau_\star, & \text{if continue} \end{cases}$
10: **end for**
11: $v_0 = \max\left\{ \frac{1}{M} \sum_{m=1}^M e^{-r\tau_\star^m} g(\mathbf{S}_{\tau_\star^m}^m), g(\mathbf{S}_0) \right\}$

For the efficient computation of the matrix A_k, using (3.3), i.e.,

$$\frac{\partial \phi_n^k}{\partial w_j}(\mathbf{W}_k^m) = \sqrt{\frac{\alpha_j}{t_k}} \phi_{n'}^k(\mathbf{W}_k^m) \quad \text{for } \phi_n^k = H_{\boldsymbol{\alpha}}^{(t_k)}, \phi_{n'}^k = H_{\boldsymbol{\alpha}-\mathbf{e}_j}^{(t_k)},$$

we can compute the matrix A_k from the basis matrix Φ_k and the increment $\Delta \mathbf{W}_k$. This routine is summarized in Algorithm 4.2, where $(A_k)_n$ represents the n-th column of the matrix A_k.

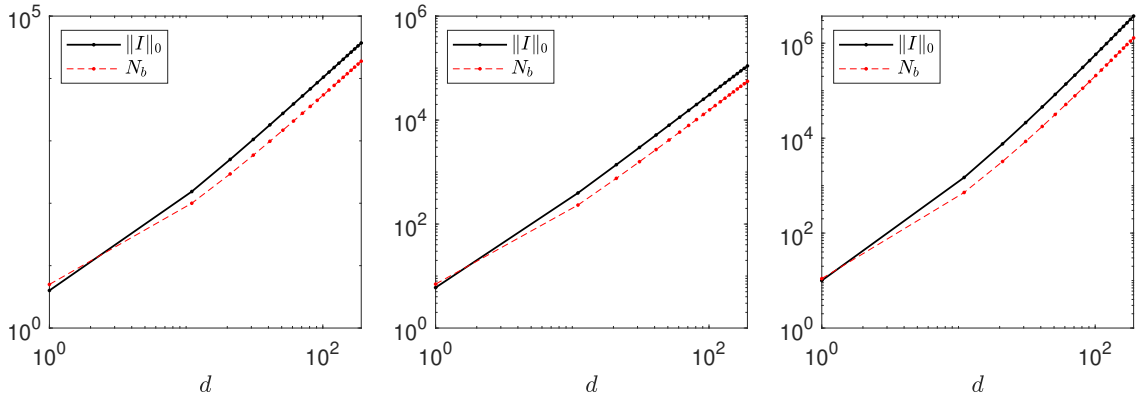
Finally, we analyze the computational complexity of Algorithm 4.1. We employ backward induction (2.1) with N time steps to price American options, which has a complexity $\mathcal{O}(N)$. For each fixed t_k, the computation consists of steps 6-9 in Algorithm 4.1. Step 6 has a complexity $\mathcal{O}(MN_b)$, when using M samples and N_b basis functions. Step 7 is detailed in Algorithm 4.2, which has a linear complexity in the number of nonzero α_j for all $\boldsymbol{\alpha} \in I$ and $j = 1, \ldots, d$,

$$\|I\|_0 := \sum_{\boldsymbol{\alpha} \in I} \#\{j = 1, \cdots, d : \alpha_j \neq 0\}.$$

Note that $\|I\|_0$ can be calculated by $(N_b - N_{b,p-1})d$ for maximum order p, where $N_{b,p-1}$ represents the number of basis functions with maximum order $p-1$. Figure 1 shows that $\|I\|_0$ scales nearly linearly with respect to N_b. Thus, the complexity of Algorithm 4.2 is linear in N_b. Step 8 involves solving a linear system of size $M \times N_b$. Using an iterative solver, the cost is $\mathcal{O}(MN_b)$ if the matrix A is well-conditioned. Step 9 involves matrix-vector multiplication, which has a complexity $\mathcal{O}(MN_b)$. Hence, with a fixed sample size M and number N of time steps, the total cost of the proposed G-LSM is $\mathcal{O}(NMN_b)$. We will numerically verify the complexity analysis in subsection 6.3.

Algorithm 4.2 Compute the matrix A_k using I, Φ_k and $\Delta\mathbf{W}_k$

Initialize $A_k = \Phi_k$
for $j = 1 : d$ **do**
  for $\boldsymbol{\alpha} \in I$ **do**
    if $\alpha_j \geq 1$ **then**
      $(A_k)_n = (A_k)_n + (\Delta W_k)_j \odot (\Phi_k)_{n'} \sqrt{\frac{\alpha_j}{t_k}}$.
    end if
  end for
end for



(a) $p = 4$ (b) $p = 6$ (c) $p = 10$

Figure 1. *$\|I\|_0$ scales linearly with respect to N_b.*

4.3. Computing deltas. Now we present the approximation of deltas for constructing hedging strategies based on Algorithm 4.1. In the backward induction loop, (2.1) and (2.3) imply the deltas at time t_k is given by

$$\frac{\partial v_k(\mathbf{s})}{\partial s_j} = \begin{cases} \dfrac{\partial g_k(\mathbf{s})}{\partial s_j}, & \text{if exercise,} \\ \dfrac{\partial \tilde{c}_k(\mathbf{s})}{\partial s_j}, & \text{if continue,} \end{cases} \qquad j = 1, \ldots, d.$$

In the continuation region, $\tilde{c}_k(\mathbf{s}) = c_k(\mathbf{w})$ with $\mathbf{s} = \mathbf{s}_0 \odot \exp(Q(\mu t_k + \sqrt{\Lambda}\mathbf{w}))$. Once we have obtained the coefficients $\boldsymbol{\beta}_k$ of the expansion, the gradient $\nabla c_k(\mathbf{w})$ of the continuation value $c_k(\mathbf{w})$ is approximated by $\nabla \hat{c}_k(\mathbf{w})$ in (4.5). Hence, we calculate the deltas in the continuation region by

$$\frac{\partial \tilde{c}_k(\mathbf{s})}{\partial s_j} \approx \sum_{i=1}^{d} \frac{\partial \hat{c}_k(\mathbf{w})}{\partial w_i} \frac{\partial w_i}{\partial s_j}, \quad k = 1, \ldots, N-1.$$

The gammas $\Gamma_{t_k}^{ij}$ can be approximated in a similar way.

Particularly, the Greeks at time $t = 0$ involves solving an additional linear system, which arises from minimizing the $L^2_{\omega_{t_1}}(\mathbb{R}^d)$ error of approximating $u_1(\mathbf{W}_1)$. We employ a slightly different ansatz from the case of $t_k > 0$, since the expansions in $H^{(0)}_{\boldsymbol{\alpha}}$ is not applicable. Instead, we take

$$\hat{c}_0(\mathbf{w}) = \sum_{\boldsymbol{\alpha}\in I} \beta_{\boldsymbol{\alpha}} H^{(\Delta t)}_{\boldsymbol{\alpha}}(\mathbf{w}).$$

Since $c_0(\mathbf{W}_0)$ is a deterministic value given $\mathbf{W}_0 = \mathbf{0}$, we consider the Euler approximation

$$u_1(\mathbf{W}_1) \approx \hat{c}_0(\mathbf{0}) + \nabla \hat{c}_0(\mathbf{0}) \cdot \mathbf{W}_1.$$

Similar to (4.6), the coefficients $\beta_{\boldsymbol{\alpha}}$ can be approximated by solving an $M \times N_b$ linear system. After that, the delta at $t = 0$ is approximated by

$$\frac{\partial \tilde{c}_0(\mathbf{s}_0)}{\partial s_j} \approx \sum_{i=1}^{d} \frac{\partial \hat{c}_0(\mathbf{0})}{\partial w_i} \frac{\partial w_i}{\partial s_j}(\mathbf{s}_0), \quad \text{with } \mathbf{s} = \mathbf{s}_0 \odot \exp(Q\sqrt{\Lambda}\mathbf{w}).$$

5. Convergence analysis. Now we analyze the convergence of Algorithm 4.1. We assume the Lipschitz continuity of the discounted payoff function $g_k(\cdot)$.

Assumption 5.1. *The discounted payoff function $g_k(\cdot)$ is L-Lipschitz continuous: for $k = 0, 1, \ldots, N$*

$$|g_k(\mathbf{s}) - g_k(\mathbf{s}')| \le L\|\mathbf{s} - \mathbf{s}'\|, \quad \forall \mathbf{s}, \mathbf{s}' \in \mathbb{R}^d. \tag{5.1}$$

Recall that $\hat{u}_k(\mathbf{w})$ is the numerical value function at time t_k computed by Algorithm 4.1, which approximates the exact value $u_k(\mathbf{w})$ for $k = 0, 1, \ldots, N-1$. We aim to establish an upper bound for $\max_{0\le k\le N-1} \|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}$.

5.1. One-step error estimation. First, we analyze the one step error over the interval $[t_k, t_{k+1}]$ for $k = 0, 1, \ldots, N-1$, i.e., the numerical value function $\hat{u}_k(\mathbf{w})$ as an approximation to the exact one $u_k(\mathbf{w})$. Given the previous value function $\hat{u}_{k+1}$, we define the current CVF $\check{c}_k(\mathbf{W}_k)$ by

$$\check{c}_k(\mathbf{W}_k) = \mathbb{E}[\hat{u}_{k+1}(\mathbf{W}_{k+1})|\mathbf{W}_k]. \tag{5.2}$$

The classic backward Euler scheme for BSDE approximate $\nabla \check{c}_k$ by

$$\check{Z}_k := \frac{1}{\Delta t}\mathbb{E}[\hat{u}_{k+1}(\mathbf{W}_{k+1})\Delta \mathbf{W}_k|\mathbf{W}_k]. \tag{5.3}$$

With the sparse Hermite polynomial ansatz space $P_{I,k}$, let $c^*_k \in P_{I,k}$ be the best approximation to $\check{c}_k$ in $H^1_{\omega_k}(\mathbb{R}^d)$ defined by

$$c^*_k := \operatorname*{argmin}_{\psi\in P_{I,k}} \|\check{c}_k - \psi\|_{H^1_{\omega_k}(\mathbb{R}^d)}. \tag{5.4}$$

Also we define the best approximation error $\mathcal{E}_k^{\text{best}}$ by

$$\mathcal{E}_k^{\text{best}} := \|\check{c}_k - c_k^*\|^2_{H^1_{\omega_k}(\mathbb{R}^d)}. \tag{5.5}$$

Next, we denote the statistical error of solving the discrete least squares problem (4.6) by

$$\mathcal{E}_k^{\text{stat}} := \|\hat{c}_k - c_k^{\text{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}, \tag{5.6}$$

where c_k^{CLS} solves the continuous least squares problem (4.3) for $k = 0, 1, \ldots, N-1$. In Theorem 5.4 below, we derive the convergence of c_k^{CLS} to the exact c_k provided that the previous approximation error $\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)}$, $\mathcal{E}_k^{\text{best}}$ in (5.5), and time step size Δt are small. We first provide in Lemma 5.2 an estimate of $\mathcal{E}_k^{\text{best}}$ in (5.5), which is a direct application of [17, Theorem 4.2].

Lemma 5.2. *For the hyperbolic cross index set I with maximum order p defined in (3.2), given $\check{c}_k \in \mathcal{K}^m(\mathbb{R}^d, \omega_k)$, we have*

$$\mathcal{E}_k^{\text{best}} \le C(m,d)\frac{1}{p^{m-1}}|\check{c}_k|^2_{\mathcal{K}^m_{\omega_k}(\mathbb{R}^d)},$$

where $\mathcal{K}^m_{\omega_k}(\mathbb{R}^d)$ is the weighted Koborov-type space defined by

$$\mathcal{K}^m_{\omega_k}(\mathbb{R}^d) = \{u : \partial^{\boldsymbol{\alpha}} u \in L^2_{\omega_k}(\mathbb{R}^d), 0 \le |\boldsymbol{\alpha}|_\infty \le m\},$$

and $|\cdot|_{\mathcal{K}^m_{\omega_k}(\mathbb{R}^d)}$ is the seminorm defined by

$$|u|_{\mathcal{K}^m_{\omega_k}(\mathbb{R}^d)} = \left(\sum_{|\boldsymbol{\alpha}|_\infty = m} \|\partial^{\boldsymbol{\alpha}} u\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}\right)^{1/2}.$$

The next result provides a one-step error estimate of the Z component in the BSDE by the backward Euler scheme (5.3). The proof is inspired by the idea of [11, pages 816-817]. By martingale representation theorem, the definition of $\check{c}_k$ (5.2) implies the existence of a square integrable process $\tilde{Z}_s$, $t_k \le s \le t_{k+1}$, such that

$$\hat{u}_{k+1}(\mathbf{W}_{k+1}) = \check{c}_k(\mathbf{W}_k) + \int_{t_k}^{t_{k+1}} \tilde{Z}_t \cdot \mathrm{d}\mathbf{W}_t. \tag{5.7}$$

Lemma 5.3. *Let $\check{Z}_k$ and $\tilde{Z}_t$ be defined in (5.3) and (5.7). Then for some constant $C_1 > 0$,*

$$\mathbb{E}\left[(\check{Z}_k - \tilde{Z}_{t_k})^2\right] \le C_1 \Delta t^2.$$

Proof. By the integration by parts formula of Malliavin calculus, we obtain

$$\check{Z}_k = \frac{1}{\Delta t}\mathbb{E}\left[\hat{u}_{k+1}(\mathbf{W}_{k+1})\Delta\mathbf{W}_k|\mathbf{W}_k\right] = \frac{1}{\Delta t}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} D_t\mathbf{W}_{k+1}\nabla\hat{u}_{k+1}(\mathbf{W}_{k+1})\,\mathrm{d}t\middle|\mathbf{W}_k\right],$$

where $D_t \mathbf{W}_{k+1}$ is an identity matrix of size $d \times d$. Together with the identity

$$\tilde{Z}_{t_k} = \nabla \check{c}_k(\mathbf{W}_k) = \frac{1}{\Delta t} \int_{t_k}^{t_{k+1}} \nabla \check{c}_k(\mathbf{W}_k)\, \mathrm{d}t,$$

it further leads to

$$\check{Z}_k - \tilde{Z}_{t_k} = \frac{1}{\Delta t} \mathbb{E}\left[\int_{t_k}^{t_{k+1}} \nabla \hat{u}_{k+1}(\mathbf{W}_{k+1}) - \nabla \check{c}_k(\mathbf{W}_k)\, \mathrm{d}s \middle| \mathbf{W}_k \right].$$

Then, by [11, Equation (26)], we obtain

$$\check{Z}_k - \tilde{Z}_{t_k} = \int_{t_k}^{t_{k+1}} \mathbb{E}\left[G(t, \mathbf{W}_t) | \mathbf{W}_k \right] \mathrm{d}t,$$

for a bounded function G. Hence, there holds $|\check{Z}_k - \tilde{Z}_{t_k}| = \mathcal{O}(\Delta t)$, and there exists a constant $C_1 > 0$ such that $\mathbb{E}\left[(\check{Z}_k - \tilde{Z}_{t_k})^2 \right] \leq C_1 \Delta t^2$. ∎

Next, we provide an error estimate of solving the continuous least squares problem (4.3) in terms of the error of the previous value function, the best approximation error in the sparse Hermite polynomial ansatz space (5.5) and the time step size Δt. The proof is inspired by the foundational work [12].

Theorem 5.4. *For $\Delta t \leq 1/4$, with the constant $C_1 > 0$ in Lemma 5.3, there holds*

$$\|c_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \leq (1 + 4\Delta t) \|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)} + \tfrac{1}{2\Delta t} \mathcal{E}_k^{\mathrm{best}} + C_1 \Delta t^2.$$

Proof. By the inequality $(a+b)^2 \leq (1+4\Delta t) a^2 + (1 + \frac{1}{4\Delta t}) b^2$, we have

$$\begin{aligned} \|c_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} &\leq (1+4\Delta t) \|c_k - \check{c}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} + (1 + \tfrac{1}{4\Delta t}) \|\check{c}_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \\ &\leq (1+4\Delta t) \|c_k - \check{c}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} + \tfrac{1}{2\Delta t} \|\check{c}_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}, \end{aligned}$$

since $\Delta t \leq 1/4$. For the first term, the definitions (2.6) and (5.2), the tower property, and Jensen's inequality lead to

$$\begin{aligned} \mathbb{E}\left[(c_k(\mathbf{W}_k) - \check{c}_k(\mathbf{W}_k))^2 \right] &= \mathbb{E}\left[(\mathbb{E}\left[u_{k+1}(\mathbf{W}_{k+1}) - \hat{u}_{k+1}(\mathbf{W}_{k+1}) | \mathbf{W}_k \right])^2 \right] \\ &\leq \mathbb{E}\left[(u_{k+1}(\mathbf{W}_{k+1}) - \hat{u}_{k+1}(\mathbf{W}_{k+1}))^2 \right] = \|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)}. \end{aligned}$$

Hence, it remains to show

$$\|\check{c}_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \leq \mathcal{E}_k^{\mathrm{best}} + 2C_1 \Delta t^3. \tag{5.8}$$

By plugging (5.7) into the quadratic loss function $E_k(\cdot)$ in (4.3), we obtain

$$\begin{aligned} E_k(\psi) &= \mathbb{E}\left[\left(\check{c}_k(\mathbf{W}_k) - \psi(\mathbf{W}_k) + \int_{t_k}^{t_{k+1}} \tilde{Z}_s \cdot \mathrm{d}\mathbf{W}(s) - \nabla \psi(\mathbf{W}_k) \cdot \Delta \mathbf{W}_k \right)^2 \right] \\ &= \mathbb{E}\left[(\check{c}_k(\mathbf{W}_k) - \psi(\mathbf{W}_k))^2 \right] + \mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} \tilde{Z}_s \cdot \mathrm{d}\mathbf{W}(s) - \nabla \psi(\mathbf{W}_k) \cdot \Delta \mathbf{W}_k \right)^2 \right]. \end{aligned}$$

Now Itô isometry implies the identity

$$\begin{aligned}
&\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} \tilde{Z}_s \cdot \mathrm{d}\mathbf{W}(s) - \nabla\psi(\mathbf{W}_k)\cdot\Delta\mathbf{W}_k\right)^2\right]\\
=&\mathbb{E}\left[\left(\int_{t_k}^{t_{k+1}} \tilde{Z}_s \cdot \mathrm{d}\mathbf{W}(s) - \int_{t_k}^{t_{k+1}} \check{Z}_k \cdot \mathrm{d}\mathbf{W}(s) + \check{Z}_k\cdot\Delta\mathbf{W}_k - \nabla\psi(\mathbf{W}_k)\cdot\Delta\mathbf{W}_k\right)^2\right]\\
=&\mathbb{E}\left[\int_{t_k}^{t_{k+1}} |\tilde{Z}_s - \check{Z}_k|^2\,\mathrm{d}s\right] + \Delta t\mathbb{E}\left[\left|\check{Z}_k - \nabla\psi(\mathbf{W}_k)\right|^2\right]\\
&+ 2\mathbb{E}\left[\int_{t_k}^{t_{k+1}} (\tilde{Z}_s - \check{Z}_k)\,\mathrm{d}s\right]\cdot\mathbb{E}[\check{Z}_k - \nabla\psi(\mathbf{W}_k)].
\end{aligned}$$

The definitions of $\check{Z}_k$ and $\tilde{Z}_s$ in (5.3) and (5.7), together with Itô isometry, yield

$$\check{Z}_k = \frac{1}{\Delta t}\mathbb{E}\left[\int_{t_k}^{t_{k+1}} \tilde{Z}_s\,\mathrm{d}s\middle|\mathbf{W}_k\right],$$

which implies

$$\mathbb{E}\left[\int_{t_k}^{t_{k+1}} (\tilde{Z}_s - \check{Z}_k)\,\mathrm{d}s\right] = 0.$$

Hence, we can express the loss function $E_k(\psi)$ as

$$E_k(\psi) = \mathbb{E}\left[(\check{c}_k(\mathbf{W}_k) - \psi(\mathbf{W}_k))^2\right] + \mathbb{E}\left[\int_{t_k}^{t_{k+1}} |\tilde{Z}_s - \check{Z}_k|^2\,\mathrm{d}s\right] + \Delta t\mathbb{E}\left[\left|\check{Z}_k - \nabla\psi(\mathbf{W}_k)\right|^2\right].$$

The equality (4.3) implies that $E_k(c_k^{\mathrm{CLS}}) \le E_k(\psi)$ for all $\psi \in P_{I,k}$. Taking $\psi := c_k^*$ as defined in (5.4), we obtain

$$\begin{aligned}
&\mathbb{E}\left[\left(\check{c}_k(\mathbf{W}_k) - c_k^{\mathrm{CLS}}(\mathbf{W}_k)\right)^2\right] + \Delta t\mathbb{E}\left[\left(\check{Z}_k - \nabla c_k^{\mathrm{CLS}}(\mathbf{W}_k)\right)^2\right]\\
\le\ &\mathbb{E}\left[(\check{c}_k(\mathbf{W}_k) - c_k^*(\mathbf{W}_k))^2\right] + \Delta t\mathbb{E}\left[\left(\check{Z}_k - \nabla c_k^*(\mathbf{W}_k)\right)^2\right].
\end{aligned}$$

Note that $\tilde{Z}_{t_k} = \nabla\check{c}_k(\mathbf{W}_k)$. By subtracting and adding $\tilde{Z}_{t_k}$, we have

$$\begin{aligned}
&\|\check{c}_k - c_k^{\mathrm{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}\\
\le\ &\|\check{c}_k - c_k^*\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} + 2\Delta t\mathbb{E}\left[(\check{Z}_k - \tilde{Z}_{t_k})^2\right] + 2\Delta t\mathbb{E}\left[|\nabla\check{c}_k(\mathbf{W}_k) - \nabla c_k^*(\mathbf{W}_k)|^2\right]\\
\le\ &\|\check{c}_k - c_k^*\|^2_{H^1_{\omega_k}(\mathbb{R}^d)} + 2C_1\Delta t^3,
\end{aligned}$$

where the last inequality follows from the condition $2\Delta t \le 1$ and Lemma 5.3. This shows the estimate (5.8), and completes the proof. ∎

Now, we can give the one-step error propagation of $\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}$ given $\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)}$.

Corollary 5.5. *For $\Delta t \le 1/6$, there holds*

$$\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \le (1+6\Delta t)\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)} + \left(1 + \frac{1}{2\Delta t}\right)\mathcal{E}_k^{\text{best}} + \frac{6}{5}C_1\Delta t^2 + \frac{6}{5\Delta t}\mathcal{E}_k^{\text{stat}},$$

where $\mathcal{E}_k^{\text{best}}$ and $\mathcal{E}_k^{\text{stat}}$ are defined in (5.5) and (5.6), and $C_1 > 0$ is the constant in Lemma 5.3.

Proof. First, using inequality $(a+b)^2 \ge (1-\Delta t)a^2 - \frac{1}{\Delta t}b^2$ for any $a, b \in \mathbb{R}$, leads to

$$\|c_k - c_k^{\text{CLS}}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \ge (1-\Delta t)\|c_k - \hat{c}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} - \frac{1}{\Delta t}\|\hat{c}_k - c_k^{CLS}\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}.$$

This and Theorem 5.4 yield

$$\begin{aligned}
&\|c_k - \hat{c}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \\
&\le \frac{1+4\Delta t}{1-\Delta t}\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)} + \frac{1}{2\Delta t(1-\Delta t)}\mathcal{E}_k^{\text{best}} + \frac{C_1\Delta t^2}{1-\Delta t} + \frac{1}{\Delta t(1-\Delta t)}\mathcal{E}_k^{\text{stat}} \\
&\le (1+6\Delta t)\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)} + \left(1 + \frac{1}{2\Delta t}\right)\mathcal{E}_k^{\text{best}} + \frac{6}{5}C_1\Delta t^2 + \frac{6}{5\Delta t}\mathcal{E}_k^{\text{stat}},
\end{aligned}$$

provided that $\Delta t \le 1/6$. Finally, using inequality $|\max(a,b) - \max(a,c)| \le |b-c|$, we derive

$$\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} = \|\max(G_k, c_k) - \max(G_k, \hat{c}_k)\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \le \|c_k - \hat{c}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)}.$$

This completes the proof. ∎

5.2. Global error estimation. Finally, we prove a global error estimate.

Theorem 5.6. *For $\Delta t \le 1/6$, there holds*

$$\max_{0\le k\le N-1}\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \le C\left(\Delta t + N\sum_{k=0}^{N-1}\left(\mathcal{E}_k^{\text{best}} + \mathcal{E}_k^{\text{stat}}\right)\right),$$

where the constant $C = \max\left(\frac{6}{5T}e^{6T}, \frac{6T}{5}C_1e^{6T}\right)$ only depends on the finite time horizon T and the constant $C_1 > 0$ defined in Lemma 5.3.

Proof. For $\Delta t \le 1/6$, Corollary 5.5 implies

$$\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \le (1+6\Delta t)\|u_{k+1} - \hat{u}_{k+1}\|^2_{L^2_{\omega_{k+1}}(\mathbb{R}^d)} + \frac{6}{5\Delta t}\left(\mathcal{E}_k^{\text{best}} + \mathcal{E}_k^{\text{stat}}\right) + \frac{6}{5}C_1\Delta t^2.$$

Using the discrete Gronwall's inequality and $\Delta t = T/N$, we obtain

$$\max_{0\le k\le N-1}\|u_k - \hat{u}_k\|^2_{L^2_{\omega_k}(\mathbb{R}^d)} \le e^{6T}\|u_N - \hat{u}_N\|^2_{L^2_{\omega_N}(\mathbb{R}^d)} + e^{6T}\frac{6N}{5T}\sum_{k=0}^{N-1}\left(\mathcal{E}_k^{\text{best}} + \mathcal{E}_k^{\text{stat}}\right) + e^{6T}\frac{6T}{5}C_1\Delta t.$$

Since the terminal condition u_N is known, the result follows by $\|u_N - \hat{u}_N\|_{L^2_{\omega_N}(\mathbb{R}^d)} = 0$. ∎

Theorem 5.6 implies that the numerical value function $\hat{u}_k$ computed by Algorithm 4.1 at each time t_k approximates the exact one u_k well if Δt, $\mathcal{E}_k^{\text{best}}$, and $\mathcal{E}_k^{\text{stat}}$ are small. Using Lemma 5.2, the best approximation error $\mathcal{E}_k^{\text{best}}$ decays geometrically with the increase of the order p of Hermite polynomials. By the law of large numbers, the statistical error $\mathcal{E}_k^{\text{stat}}$ between the discrete and the continuous least squares will be small when using a large number of sample paths.

6. Numerical examples. In this section, we present several examples of high-dimensional Bermudan option pricing to show the efficiency and accuracy of Algorithm 4.1. The codes for the numerical experiments can be founded in the GitHub repository https://github.com/jiefeiy/glsm-american. The accuracy of the computed prices $\hat{v}_0(\mathbf{s}_0)$ and deltas $\nabla\hat{v}_0(\mathbf{s}_0)$ are measured by the relative errors defined by

$$\frac{|\hat{v}_0(\mathbf{s}_0) - v_0^\dagger|}{|v_0^\dagger|} \times 100\% \text{ and } \frac{\|\nabla\hat{v}_0(\mathbf{s}_0) - \Delta_0^\dagger\|}{\|\Delta_0^\dagger\|} \times 100\%,$$

respectively, where $v_0^\dagger$ and $\Delta_0^\dagger$ are exact price and delta at time $t = 0$. Unless otherwise stated, the results are the average of 10 independent runs. The parameter settings of the examples are summarized in Table 1, which has been considered previously [25, 21, 6, 3, 9].

Table 1

The parameters used in Examples 1-5. Examples 3 and 4 share the parameters except the volatility σ_i.

Example	Parameters
1. Geometric basket put	$K = 100, T = 0.25, r = 0.03, \delta_i = 0, \sigma_i = 0.2, \rho_{ij} = 0.5, N = 50$
2. Geometric basket call	$K = 100, T = 2, r = 0, \delta_i = 0.02, \sigma_i = 0.25, \rho_{ij} = 0.75, N = 50$
3. Max-call with d symmetric assets	$K = 100, T = 3, r = 0.05, \delta_i = 0.1, \sigma_i = 0.2, \rho_{ij} = 0, N = 9$
4. Max-call with d asymmetric assets	$\sigma_i = \begin{cases} 0.08 + 0.32 \times (i-1)/(d-1), & \text{if } d \le 5 \\ 0.1 + i/(2d), & \text{if } d > 5 \end{cases}$
5. Put option under Heston model	$K = 10, T = 0.25, r = 0.1, v_0 = 0.0625, \rho = 0.1,$ $\kappa = 5, \theta = 0.16, \nu = 0.9, N = 50$

6.1. Example 1: Bermudan geometric basket put. Geometric basket options are benchmark tests for high-dimensional option pricing problems, since they can be reduced to one-dimensional problems, and thus highly accurate prices are available. Indeed, the price of the d-dimensional problem equals that of the one-dimensional American option with initial price $\hat{s}_0$, volatility $\hat{\sigma}$, and dividend yield $\hat{\delta}$ given respectively by

$$\hat{s}_0 = \Big(\prod_{i=1}^d s_0^i\Big)^{1/d}, \quad \hat{\sigma} = \frac{1}{d}\sqrt{\sum_{i,j}\sigma_i\sigma_j\rho_{ij}}, \quad \hat{\delta} = \frac{1}{d}\sum_{i=1}^d\Big(\delta_i + \frac{\sigma_i^2}{2}\Big) - \frac{\hat{\sigma}^2}{2}.$$

We consider the example of Bermudan geometric basket put from [13, 25]. The exact prices are computed by solving the reduced one-dimensional problem via a quadrature and interpolation-based method [25] for Bermudan options. We present in Table 2 the computed option prices and their relative errors using G-LSM and LSM, with the same ansatz space for the CVF. The results show that G-LSM achieves higher accuracy than LSM in high

dimensions: G-LSM has a relative error 0.55% for $d = 15$, which is almost ten times smaller than that by LSM. That is, by incorporating the gradient information, the accuracy of LSM can be substantially improved.

Table 2

The price and relative errors of Bermudan geometric basket put computed by LSM and G-LSM using $M = 100,000$ *samples, with* $s_0^i = 100$ *and* $p = 10$.

d	N_b	LSM	error	G-LSM	error	$v_0^\dagger$
1	11	3.6715	0.16%	3.6532	0.34%	3.6658
2	29	3.1886	0.18%	3.1791	0.12%	3.1831
3	56	3.0113	0.28%	2.9896	0.44%	3.0029
5	141	2.8700	0.70%	2.8381	0.41%	2.8499
10	581	2.8030	2.71%	2.7160	0.48%	2.7290
15	1446	2.8258	5.15%	2.6726	0.55%	2.6874

Table 3 gives the computed option prices and their relative errors by G-LSM with different maximum polynomial orders p for the 20-dimensional geometric basket put. The relative error decays steadily as the order p of Hermite polynomials increases, which agrees with Theorem 5.6.

Table 3

The computed prices and their relative errors by G-LSM for $d = 20$ *with polynomial order* p *and* $M = 100,000$ *samples.* $v_0^\dagger = 2.6664$.

p	2	5	10	12
$\hat{v}_0(\mathbf{s}_0)$	2.6389	2.6414	2.6529	2.6634
error	1.03%	0.94%	0.50%	0.11%

6.2. Example 2: American geometric basket call. Now we consider the example of American geometric basket call option from [21, 6] to demonstrate that the proposed G-LSM can achieve the same level of accuracy as the DNN-based method [6], and take $M = 720,000$ samples as in [6]. The prices and deltas are given in Table 4 and Table 5, respectively, where the results of [6] are the average of 9 independent runs. From Table 4, both methods have similar accuracy for the price. From Table 5, the relative error of delta using G-LSM and DNN varies slightly with the dimension d. This is probably because the DNN-based method computes the delta via a sample average, while G-LSM uses the derivatives of the value function directly. The relative error of the delta computed by G-LSM increases slightly for larger dimensions, possibly due to the small magnitude of the exact delta values.

Table 4

Prices of American geometric basket call at $t = 0$ *using* $M = 720,000$ *samples.*

				G-LSM		DNN [6]	
d	s_0^i	p	exact	price	error	price	error
7	100	10	10.2591	10.2475	0.11%	10.2468	0.12%
13	100	10	10.0984	10.0781	0.20%	10.0822	0.16%
20	100	10	10.0326	10.0141	0.18%	10.0116	0.21%
100	100	6	9.9345	9.8980	0.37%	9.9163	0.18%

Table 5

Deltas of American geometric basket call at $t = 0$ using $M = 720,000$ samples. $\mathbf{1} = (1, 1, \ldots, 1)^\top$.

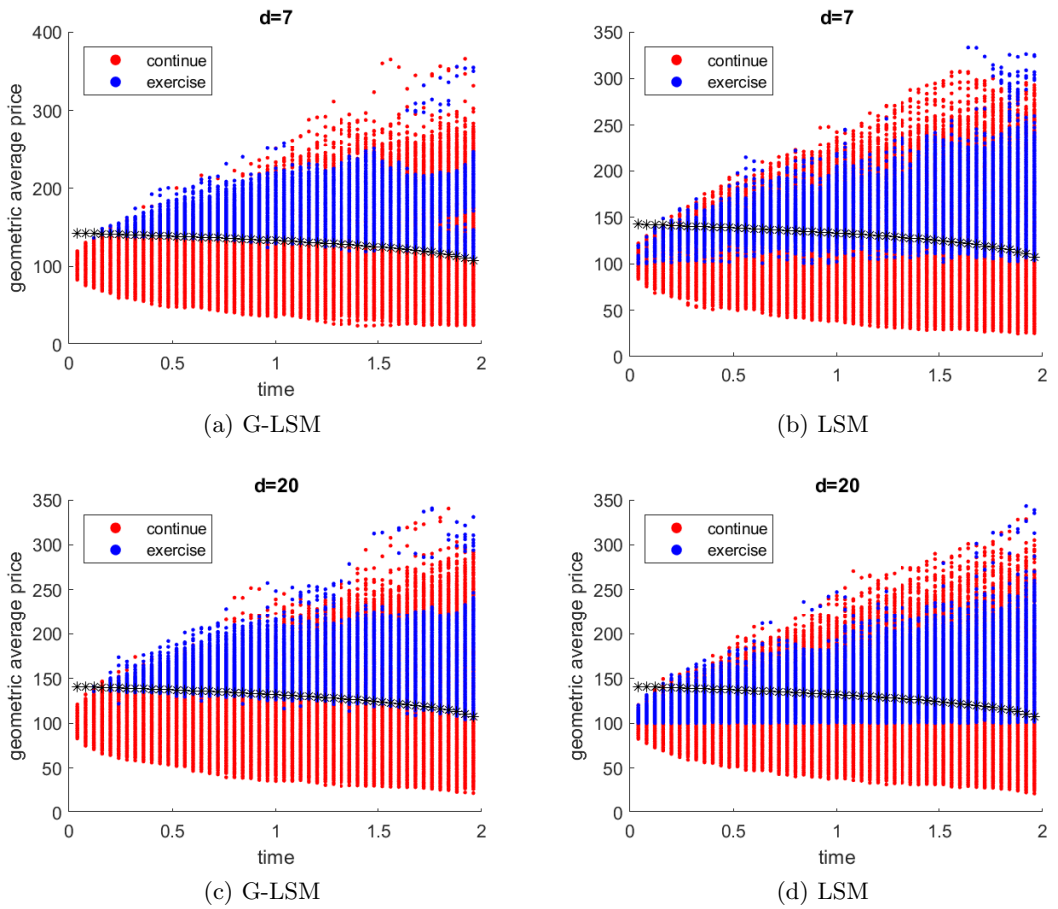
				G-LSM		DNN [6]	
d	s_0^i	p	exact	delta	error	delta	error
7	100	10	$0.0722 * \mathbf{1}$	$(0.0724, 0.0725, \ldots, 0.0724)^\top$	0.32%	$0.0717 * \mathbf{1}$	0.69%
13	100	10	$0.0387 * \mathbf{1}$	$(0.0389, 0.0387, \ldots, 0.0388)^\top$	0.39%	$0.0384 * \mathbf{1}$	0.78%
20	100	10	$0.0251 * \mathbf{1}$	$(0.0253, 0.0253, \ldots, 0.0254)^\top$	0.59%	$0.0249 * \mathbf{1}$	0.80%
100	100	6	$0.00502 * \mathbf{1}$	$(0.00501, 0.00508, \ldots, 0.00498)^\top$	1.45%	$0.00498 * \mathbf{1}$	0.80%

With a priori knowledge of the regularity of the CVF, we can approximate functions with polynomials in high dimensions. We briefly compare the complexity of the two approaches. Compared with the DNN approximator, G-LSM can involve fewer unknown parameters, and involves a simpler numerical task (solving least-squares problems versus minimizing nonconvex losses). Indeed, at each fixed time step, [6] suggests training the neural network with $L = 7$ hidden layers and width $d + 5$ in each layer, leading to more than $L(d + 5)^2$ parameters. In contrast, the number of undetermined parameters in G-LSM is the number N_b of basis functions, which has a cardinality $\mathcal{O}(p(\ln p)^{d-1})$. For example, for $d = 100$, the neural network approach involves more than 77175 parameters, whereas G-LSM with $p = 6$ involves only $N_b = 15451$ basis functions.

Next, Figure 2 shows the classification results of continued and exercised data using G-LSM and LSM with the number of simulated paths $M = 100,000$ in $d = 7$ or 20. Compared with the exact exercise boundary, G-LSM achieves better accuracy in determining the exercise boundary than LSM despite of using the same ansatz and number of paths. Thus, even with the right ansatz space, LSM might fail the task of finding exercise boundary in high dimensions using only a limited number of samples. Compared with [6, Figure 5] and [18, Figure 6], Figure 2 domonstrates that G-LSM can detect accurate exercise boundary with fewer number of paths than the DNN-based method.

6.3. Example 3: Bermudan max-call with symmetric assets. To benchmark G-LSM on high-dimensional problems without exact solutions and to validate the complexity analysis in section 4, we test Bermudan max-call option and report the computing time. The computing time is calculated as follows. For a fixed time step, T_{bas} is the time for generating basis matrix Φ, T_{mat} is the time for assembling matrix A, T_{lin} is the time for solving linear system, and T_{up} is the time for updating values. The overall computing time is $T_{\text{tot}} \approx (N - 1)(T_{\text{bas}} + T_{\text{mat}} + T_{\text{lin}} + T_{\text{up}})$.

Table 6 presents the prices and computing time (in seconds) for Bermudan max-call options with d symmetric assets. The reference 95% confidence interval (CI) is taken from [3]. The reference CI is computed with more than 3000 training steps and a batch of 8192 paths in each step, which in total utilizes more than 10^7 paths. The last five columns of the table report the computing time for the step 6, 7, 8, 9 in Algorithm 4.1, and the total time, respectively. All the computation for this example was performed on an Intel Core i9-10900 CPU 2.8 GHz desktop with 64GB DDR4 memory using MATLAB R2023b. It is observed that the prices computed by G-LSM fall into or stay very close to the reference 95% CI, confirming the high accuracy of G-LSM. Furthermore, the time for generating basis matrix, T_{bas}, dominates the



(a) G-LSM (b) LSM

(c) G-LSM (d) LSM

Figure 2. *Classification of the simulated continued and exercised data using* $M = 100,000$ *samples, with* $p = 10$. *Black star dots represent the exact exercise boundary.*

overall computing time. Hence, the cost mainly arises from evaluating Hermite polynomials on sampling paths, which is also required by LSM. In comparison with LSM , T_{mat} is the extra cost to incorporate the gradient information and takes only a small fraction of the total time. Therefore, G-LSM has nearly identical cost with LSM.

We present in Table 7 the computing time for each basis function to verify that the complexity is almost linear in N_b as analyzed in section 4. The ratio does not vary much with the dimension d. Since $N_b = \mathcal{O}(p(\ln p)^{d-1})$ with polynomials up to order p, the total computing cost of G-LSM exhibits a polynomial growth, which overcomes the so-called curse of dimensionality. Figure 3 shows the growth of N_b with respect to the dimension for $1 \leq d \leq 200$ and $p = 6$, indicating that N_b exhibits a nearly quadratic growth in dimensions for $d \leq 200$.

Table 6 reports the time T_{lin} of solving linear system using the built-in MATLAB function `cgs`. The main memory requirement is storing the matrix $A \in \mathbb{R}^{M \times N_b}$, which takes $M \times N_b \times 8$ bytes in double-precision floating-point format. For $d = 100$, the memory for storing A is about 4GB. This limits the application of G-LSM with direct solvers in higher dimensions. To

Table 6

The results for Bermudan max-call options with d symmetric assets using $M = 100,000$ samples. $s_0^i = 100$ for $i = 1, \dots, d$.

d	p	N_b	reference 95% CI	G-LSM	T_{bas}	T_{mat}	T_{lin}	T_{up}	T_{tot}
2	10	29	[13.880, 13.910]	13.8970	0.1678	0.0128	0.0024	0.0039	1.5725
3	10	56	[18.673, 18.699]	18.6715	0.2890	0.0265	0.0048	0.0060	2.6347
5	10	141	[26.138, 26.174]	26.0553	0.5718	0.0758	0.0149	0.0132	5.4662
10	10	581	[38.300, 38.367]	38.1738	1.9932	0.3164	0.1397	0.0497	19.9308
20	10	2861	[51.549, 51.803]	51.6508	11.3278	1.7153	3.2000	0.2434	131.9362
30	5	1456	[59.476, 59.872]	59.5475	7.8154	0.7750	0.7044	0.1247	75.3066
50	5	3926	[69.560, 69.945]	69.7216	29.3088	2.0516	5.6871	0.3281	299.4528
100	4	5351	[83.357, 83.862]	83.6777	71.7678	2.7185	10.8018	0.4306	684.9927

Table 7

Ratio of the computing time and number N_b of basis functions.

d	T_{bas}/N_b	T_{mat}/N_b	T_{lin}/N_b	T_{up}/N_b	T_{tot}/N_b
2	0.0058	0.0004	0.0001	0.0001	0.0542
3	0.0052	0.0005	0.0001	0.0001	0.0470
5	0.0041	0.0005	0.0001	0.0001	0.0388
10	0.0034	0.0005	0.0002	0.0001	0.0343
20	0.0040	0.0006	0.0011	0.0001	0.0461
30	0.0054	0.0005	0.0005	0.0001	0.0517
50	0.0075	0.0005	0.0014	0.0001	0.0763
100	0.0134	0.0005	0.0020	0.0001	0.1280

remedy the issue, one can solve the linear system on a large RAM server, or use single-precision floating-point, or with stochastic gradient descent.

Figure 4 shows the classification of continued and exercised sample points computed by G-LSM and LSM in the example of two-dimensional max-call. G-LSM yields a smoother exercise boundary than LSM. Compared with the exercise boundary computed in literature [19, Figure 3], G-LSM exhibits higher accuracy, albeit that the same ansatz space for the CVF is employed.

6.4. Example 4: Bermudan max-call with asymmetric assets. Example 3 assumes that all underlying assets follow the same dynamic. To further show the robustness of G-LSM, we consider the Bermudan max-call option but each asset has different volatility. The reference 95% confidence interval (CI) is taken from [3]. The pricing results for different initial price s_0^i and dimension d are listed in Table 8. The standard error (s.e.) is calculated by $\sqrt{\frac{1}{10(10-1)}\sum_{i=1}^{10}(v_0^{(i)} - \bar{v})^2}$ with $\bar{v}$ being the average of 10 independent runs. Similar to Example 3, the prices computed by G-LSM always fall into or stay very close to the reference 95% CI.

6.5. Example 5: Bermudan put under Heston model. The previous experiments and theoretical analysis are concerned with the most frequently used multi-asset Black-Scholes model, cf. subsection 2.2. We now generalize G-LSM to price Bermudan option under the Heston model. The Heston model defines the dynamic of the log-price process, $X_t^1 := \ln(S_t/S_0)$,

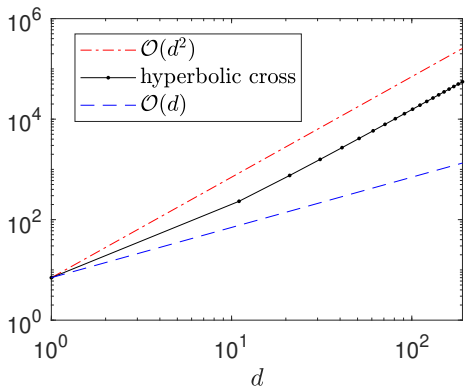
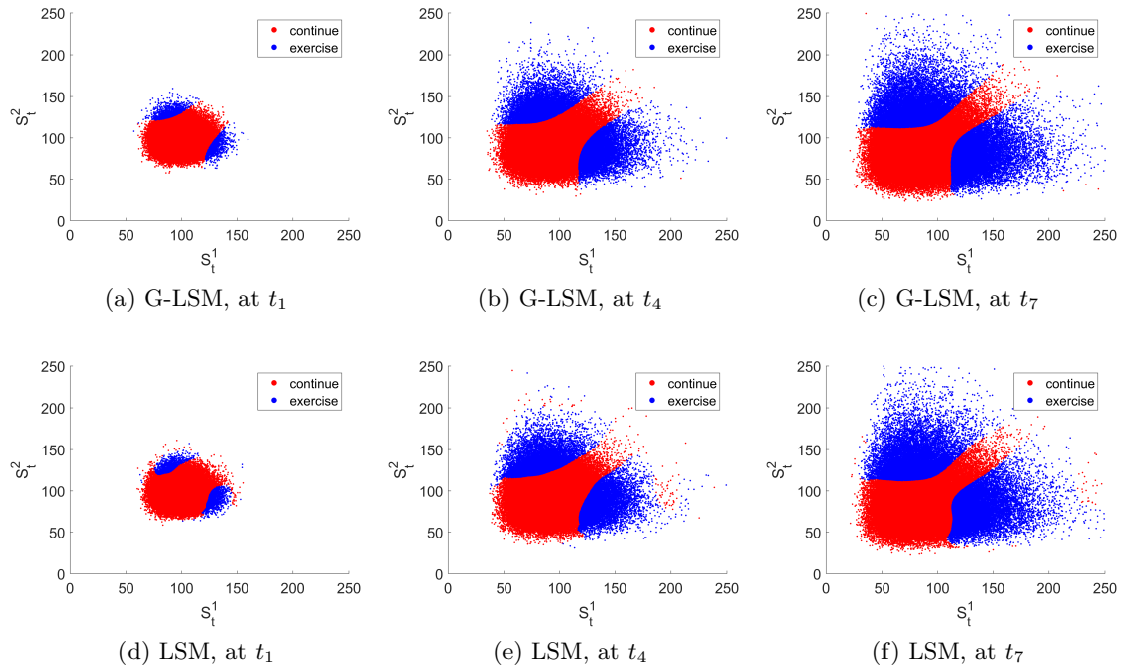


Figure 3. *The number N_b of basis functions in dimension $1 \leq d \leq 200$ with hyperbolic cross index set versus linear and quadratic scale in d, with $p = 6$.*



(a) G-LSM, at t_1 (b) G-LSM, at t_4 (c) G-LSM, at t_7

(d) LSM, at t_1 (e) LSM, at t_4 (f) LSM, at t_7

Figure 4. *Classification of simulated continued and exercised data for 2-d Bermudan max-call option at different times t_k, $k = 1, 4, 7$. Here, we take $s_0^i = 100$ for $i = 1, 2$, $M = 100,000$ and $p = 20$.*

and the volatility process, v_t, by two-dimensional SDEs:

$$\begin{aligned} \mathrm{d}X_t^1 &= (r - \tfrac{1}{2}v_t)\,\mathrm{d}t + \sqrt{v_t}\left(\rho\,\mathrm{d}W_t^1 + \sqrt{1-\rho^2}\,\mathrm{d}W_t^2\right), \\ \mathrm{d}v_t &= \kappa(\theta - v_t)\,\mathrm{d}t + \nu\sqrt{v_t}\,\mathrm{d}W_t^1, \end{aligned}$$

Table 8

Results for Bermudan max-call options with d asymmetric assets, with $M = 100,000$.

d	p	s_0^i	reference 95% CI	G-LSM	s.e.
2	10	90	[14.299, 14.367]	14.3472	0.0230
2	10	100	[19.772, 19.829]	19.8019	0.0371
2	10	110	[27.138, 27.163]	27.1041	0.0213
3	10	90	[19.065, 19.104]	19.0266	0.0241
3	10	100	[26.648, 26.701]	26.6931	0.0411
3	10	110	[35.806, 35.835]	35.8363	0.0472
5	10	90	[27.630, 27.680]	27.6032	0.0259
5	10	100	[37.940, 38.014]	37.9309	0.0405
5	10	110	[49.445, 49.533]	49.3711	0.0473
10	10	90	[85.857, 86.087]	85.8221	0.0376
10	10	100	[104.603, 104.864]	104.7052	0.1029
10	10	110	[123.570, 123.904]	123.4777	0.0686
20	10	90	[125.819, 126.383]	126.4276	0.0980
20	10	100	[149.480, 150.053]	150.4028	0.1194
20	10	110	[173.144, 173.937]	173.8584	0.1349
30	5	90	[154.378, 155.039]	154.6913	0.1128
30	5	100	[181.155, 182.033]	181.6733	0.1385
30	5	110	[208.091, 209.086]	208.1267	0.1160
50	5	90	[195.793, 196.963]	196.6921	0.0890
50	5	100	[227.247, 228.605]	227.7831	0.1385
50	5	110	[258.661, 260.092]	259.7261	0.1413
100	4	90	[263.043, 264.425]	263.0543	0.1515
100	4	100	[301.924, 303.843]	302.0623	0.2130
100	4	110	[340.580, 342.781]	340.8581	0.2233

where W_t^1 and W_t^2 are two independent Wiener processes, and the model parameters r, κ, θ, ν and ρ represent the interest rate, the speed of mean reversion, the mean level of variance, the variance of volatility process, and the correlation coefficient, respectively. Since the transition density of log-variance has better regularity [9], we take the log-variance process $X_t^2 := \ln(v_t)$ as the regression variable. Let $\mathbf{X}_t = [X_t^1, X_t^2]^\top$ be a two-dimensional process. The discounted continuation value at time t_k is given by

$$c_k(\mathbf{X}_{t_k}) = \mathbb{E}[u_{k+1}(\mathbf{X}_{t_{k+1}})|\mathbf{X}_{t_k}],$$

with u_{k+1} being the discounted value function. In view of the martingale representation theorem and backward Euler approximation, we obtain

$$u_{k+1}(\mathbf{X}_{t_{k+1}}) \approx c_k(\mathbf{X}_{t_k}) + \big(\sigma(\mathbf{X}_{t_k})^\top \nabla c_k(\mathbf{X}_{t_k})\big) \cdot \Delta\mathbf{W}_k,$$

where $\sigma(\cdot)$ is the covariance matrix of $\mathbf{X}_t$ given by

$$\sigma(\mathbf{X}_t) = \begin{bmatrix} \rho \exp(X_t^2/2) & \sqrt{1-\rho^2}\exp(X_t^2/2) \\ \nu \exp(-X_t^2/2) & 0 \end{bmatrix}.$$

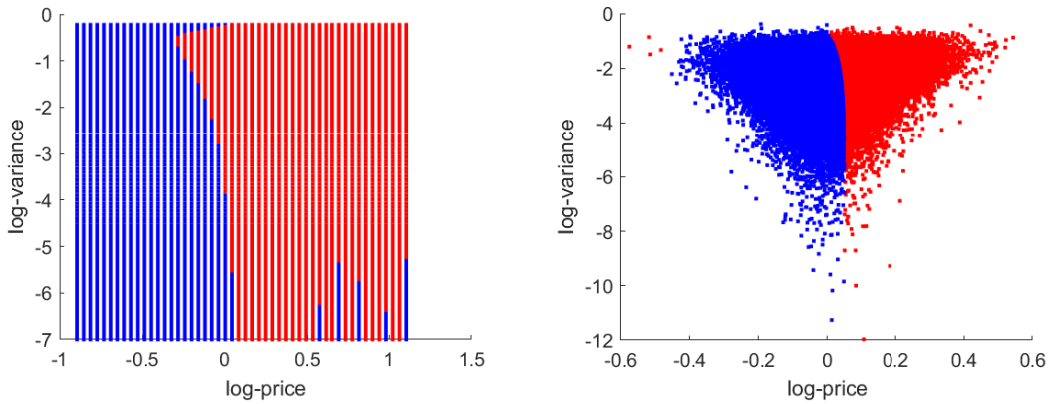
The reference prices are computed by the COS method [9], with 2^7 cosine basis functions to approximate the transition density and 2^7 points for the quadrature rule in log-variance

dimension. The results of G-LSM are calculated using 70 basis functions (polynomials up to order 20) with Hermite polynomials in log-price dimension and Chebyshev polynomials in log-variance dimension. The results in Table 9 show that the computed prices by G-LSM coincide with the reference prices up to the two places after the decimal separator for most cases. In Figure 5, we plot the classification results of continued and exercised points using two approaches. Given that the G-LSM method is simulation-based, it has great potential for stochastic volatility models in higher dimensions.

Table 9

Prices of Bermudan put option under the Heston model. $M = 100,000$. $p = 20$.

s_0	8	9	10	11	12
COS	1.9958	1.1061	0.5186	0.2131	0.0818
G-LSM	1.9949	1.0972	0.5146	0.2118	0.0807



(a) COS (b) G-LSM

Figure 5. *Classification of continued and exercised grid/simulated points using COS and G-LSM under the Heston model at time* t_{25} *with initial price* $s_0 = 8$.

7. Conclusions and outlook. In this work, we have proposed a novel gradient-enhanced least squares Monte Carlo (G-LSM) method that employs sparse Hermite polynomials as the ansatz space to price and hedge American options. The method enjoys low complexity for the gradient evaluation, ease of implementation and high accuracy for high-dimensional problems. We analyzed rigorously the convergence of G-LSM based on the BSDE technique, stochastic and Malliavin calculus. Extensive benchmark tests clearly show that it outperforms least squares Monte Carlo (LSM) in high dimensions with almost the same cost and it can also achieve competitive accuracy relative to the deep neural networks-based methods.

There are several avenues for further research. The superiority of G-LSM over LSM in high dimensions indicates that matching option values at t_{k+1} might be a better choice than at t_k for approximating the continuation value function. There are other variants of LSM with different ansatz spaces, and it is natural to ask whether incorporating the gradient information will also result in improved performance for these variants. Moreover, to solve

higher dimensional problems, e.g., $d = 1000$, the hierarchical tensor train technique can be applied, which has been combined with LSM in [2], and it is natural to combine the technique with G-LSM. Numerical results also show the potential of G-LSM for stochastic volatility models. It would be interesting to investigate G-LSM for more challenging financial models, e.g., rough volatility models.

Acknowledgments. The authors acknowledge the support of research computing facilities offered by Information Technology Services, the University of Hong Kong.

REFERENCES

[1] B. Adcock, S. Brugiapaglia, and C. G. Webster, *Sparse Polynomial Approximation of High-Dimensional Functions*, SIAM, Philadelphia, PA, 2022.
[2] C. Bayer, M. Eigel, L. Sallandt, and P. Trunschke, *Pricing high-dimensional Bermudan options with hierarchical tensor formats*, SIAM Journal on Financial Mathematics, 14 (2023), pp. 383–406.
[3] S. Becker, P. Cheridito, and A. Jentzen, *Deep optimal stopping*, The Journal of Machine Learning Research, 20 (2019), pp. 2712–2736.
[4] S. Becker, P. Cheridito, and A. Jentzen, *Pricing and hedging American-style options with deep learning*, Journal of Risk and Financial Management, 13 (2020), p. 158.
[5] B. Bouchard and X. Warin, *Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods*, in Numerical Methods in Finance: Bordeaux, June 2010, Springer, Berlin, 2012, pp. 215–255.
[6] Y. Chen and J. W. Wan, *Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions*, Quantitative Finance, 21 (2021), pp. 45–67.
[7] W. E, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Communications in Mathematics and Statistics, 5 (2017), pp. 349–380.
[8] N. El Karoui, C. Kapoudjian, E. Pardoux, S. Peng, and M.-C. Quenez, *Reflected solutions of backward SDE's, and related obstacle problems for PDE's*, The Annals of Probability, 25 (1997), pp. 702–737.
[9] F. Fang and C. W. Oosterlee, *A Fourier-based valuation method for Bermudan and barrier options under Heston's model*, SIAM Journal on Financial Mathematics, 2 (2011), pp. 439–463.
[10] C. Gao, S. Gao, R. Hu, and Z. Zhu, *Convergence of the backward deep bsde method with applications to optimal stopping problems*, SIAM Journal on Financial Mathematics, 14 (2023), pp. 1290–1303.
[11] E. Gobet and C. Labart, *Error expansion for the discretization of backward stochastic differential equations*, Stochastic Processes and their Applications, 117 (2007), pp. 803–829.
[12] C. Huré, H. Pham, and X. Warin, *Deep backward schemes for high-dimensional nonlinear PDEs*, Mathematics of Computation, 89 (2020), pp. 1547–1579.
[13] P. Kovalov, V. Linetsky, and M. Marcozzi, *Pricing multi-asset American options: A finite element method-of-lines with smooth penalty*, Journal of Scientific Computing, 33 (2007), pp. 209–237.
[14] B. Lapeyre and J. Lelong, *Neural network regression for Bermudan option pricing*, Monte Carlo Methods and Applications, 27 (2021), pp. 227–247.
[15] F. Longstaff and E. Schwartz, *Valuing American options by simulation: a simple least-squares approach*, The Review of Financial Studies, 14 (2001), pp. 113–147.
[16] M. Ludkovski, *Kriging metamodels and experimental design for Bermudan option pricing*, Journal of Computational Finance, 22 (2018), pp. 37–77.
[17] X. Luo, *Error analysis of the Wiener–Askey polynomial chaos with hyperbolic cross approximation and its application to differential equations with random input*, Journal of Computational and Applied Mathematics, 335 (2018), pp. 242–269.
[18] A. S. Na and J. W. L. Wan, *Efficient pricing and hedging of high-dimensional American options using deep recurrent networks*, Quantitative Finance, 23 (2023), pp. 631–651.

[19] A. M. Reppen, H. M. Soner, and V. Tissot-Daguette, *Deep stochastic optimization in finance*, Digital Finance, 5 (2023), pp. 91–111.
[20] R. Seydel and R. Seydel, *Tools for computational finance*, vol. 3, Springer, 2006.
[21] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
[22] J. Tsitsiklis and B. Van Roy, *Regression methods for pricing complex American-style options*, IEEE Transactions on Neural Networks, 12 (2001), pp. 694–703.
[23] H. Wang, H. Chen, A. Sudjianto, R. Liu, and Q. Shen, *Deep learning-based BSDE solver for LIBOR market model with application to Bermudan swaption pricing and hedging*, arXiv preprint arXiv:1807.06622, (2018).
[24] Y. Wang and R. Caflisch, *Pricing and hedging American-style options: a simple simulation-based approach*, The Journal of Computational Finance, 13 (2009), pp. 95–125.
[25] J. Yang and G. Li, *On sparse grid interpolation for American option pricing with multiple underlying assets*, arXiv preprint arXiv:2309.08287, (2023).
[26] J. Zhang, *Backward stochastic differential equations*, Springer, 2017.