Simultaneous upper and lower bounds of American option prices with hedging via neural networks

Ivan Guo[*1,2], Nicolas Langrené[†3], and Jiahao Wu[1]

[1]School of Mathematical Sciences, Monash University, Melbourne, Australia
[2]Centre for Quantitative Finance and Investment Strategies, Monash University, Australia
[3]Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College, Zhuhai, China

April 19, 2024

Abstract

In this paper, we introduce two novel methods to solve the American-style option pricing problem and its dual form at the same time using neural networks. Without applying nested Monte Carlo, the first method uses a series of neural networks to simultaneously compute both the lower and upper bounds of the option price, and the second one accomplishes the same goal with one global network. The avoidance of extra simulations and the use of neural networks significantly reduce the computational complexity and allow us to price Bermudan options with frequent exercise opportunities in high dimensions, as illustrated by the provided numerical experiments. As a by-product, these methods also derive a hedging strategy for the option, which can also be used as a control variate for variance reduction.

1 Introduction

Pricing American options is a type of optimal control/stopping problem for which numerical methods have been extensively explored due to the lack of analytical solutions. However, classical methods based on partial differential equations and binomial trees become expensive computationally when there are multiple factors impacting the value of the option, a

limitation known as the *curse of dimensionality*. To circumvent this difficulty, simulation-based methods have been extensively explored [47, 4, 13, 39, 48, 10, 12, 3, 36, 11, 40]. By directly solving the pricing problem, these methods typically generate a candidate optimal stopping strategy and a lower bound on the price, which is more in the interest of the buying party. On the other hand, option sellers would be more interested in an upper bound. Haugh and Kogan [30] and Rogers [45] independently explored the duality of the pricing problem, based on which a variety of methods have been proposed [2, 34, 9, 44, 46] to derive an upper bound on the option price by solving its dual problem.

Among simulation-based methods, the Least Squares Monte Carlo (LSMC) method [39, 48] has gained much popularity. In search of the optimal stopping strategy, the dynamic programming principle is used, with continuation values approximated by a set of basis functions via linear regression. However, as the dimension of the problem increases, the number of basis functions significantly increases and the method can become numerically unstable. Various studies, including Kohler et al. [35], Lapeyre and Lelong [37] and Herrera et al. [31], have proposed to replace the linear regression in the LSMC method by neural networks (NNs). Additionally, Goudenège et al. [27] and Goudenège et al. [28] have employed Gaussian process regression to estimate the continuation value. Reppen et al. [43] applied NNs to parameterise the stopping boundary. Moreover, Bayer et al. [5] have devised a forward and a backward algorithm to approximate the stopping strategy by randomising them with independent noises, while Gonon [26] has utilised neural networks to directly approximate the value function and showed that the method is free of the curse of dimensionality. Other works [29, 42, 16, 24, 41, 23] have explored the application of deep learning in option pricing by addressing the corresponding partial differential equations (PDEs) or backward stochastic differential equations (BSDEs).

Besides option pricing, hedging strategies are crucial in risk management. Most existing methods for generating hedging strategies either involve taking the first derivative of approximated option values [3, 11, 33, 41] or approximating the function representing the difference between option values at different times once the option has been priced [7, 6]. However, the efficiency of these strategies relies on the accurate differentiation of the estimated continuation value function. Since functions with similar values can have very different derivatives, even satisfying approximations of the value process can lead to ineffective hedging strategies.

The primary contribution of our work lies in incorporating the dual formulation of the option price into the modified LSMC method to design algorithms that concurrently produce both lower and upper bounds of the option price. Moreover, our method facilitates the derivation of hedging strategies as an immediate by-product, computed directly from the dual martingale used in the upper bound estimate instead of the differentiation. Unlike

traditional methods, our approach offers hedging strategies at all times before maturity, not just at exercise times, and can serve as a control variate to reduce variance, thereby yielding a more accurate lower bound. Becker et al. [7, 8] proposed a method to price Bermudan options in high-dimensions. However, in their method, they first find a stopping strategy to approximate a lower bound, based on which they then derive an upper bound using nested Monte Carlo. Similarly, the hedging strategy is also based on the stopping strategy with another independent simulation. Their other work [7] has a similar structure but approximate the stopping strategy instead. In the case of pricing Bermudan options with frequent exercise opportunities, which approximates an American option, the computational cost can be very high as the cost of nested simulation increases quadratically with the number of stopping opportunities. Similar methods designed by Lokeshwar et al. [38], Belomestny et al. [9] do not require nested simulations, but the derivation of a biased upper estimate is separate from the determination of the stopping strategy. The work by Huré et al. [32] on reflected BSDEs resolution shares some resemblance, but it only generates a point estimate, and the details in the dynamic programming are different.

In addition, we present the use of one global network instead of a series of networks in the derivation by treating time as an additional state variable. Global networks have been introduced to solve semi-linear PDEs [15] and other control problems [25, 22]. In such stopping problems, the target values are known when the training starts as they are outputs of the problem, rather than inputs. However, the training targets are unavailable at the outset of the problem. We propose to alternate the update of stopping strategies and the network training till it produces satisfactory results.

This paper is structured in the following order. Section 2 lays out the theoretical groundwork for combining the LSMC algorithm with the dual formulation. In Section 3, we introduce the numerical methods devised and then present various variants in Section 4. Section 5 is dedicated to demonstrating numerical results in both low- and high-dimensional settings, and then we conclude in Section 6.

2   The Problem Formulation

Consider an American option with maturity $T > 0$. Let $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t \in [0,T]}, \mathbb{Q})$ be a filtered probability space, where $\mathbb{F}$ is the augmented filtration of a d-dimensional Brownian motion $(W_t)_{t \in [0,T]}$, and $\mathbb{Q}$ is the equivalent martingale measure.

Define $\beta_t = e^{rt}$ as the value of the risk-free account at $t \in [0,T]$, where the constant r is the risk-free interest rate. The price of the option is based on d risky assets whose value process $(S_t)_{t \in [0,T]}$ is Markovian and is the solution to the SDE

$$dS_t = rS_t dt + \sigma(t, S_t)dW_t,$$

where $\sigma\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d \times d}$ is assumed to satisfy sufficient regularity conditions to ensure the well-posedness of the equation.

2.1 The lower bound of the option price

Let $(Z_t)_{t \in [0,T]}$ denote the $\mathbb{F}$-adapted continuous discounted payoff process of the option satisfying $\mathbb{E}[\sup_{t \in [0,T]} Z_t] < \infty$. Let $\tau\colon \Omega \to [0,T]$ be a stopping time, and $\mathcal{T}$ be the set of all stopping times with respect to the filtration $\mathbb{F}$. Then, the value of the American option at time t discounted back to time 0 is

$$V_t = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}, \tau \geqslant t} \mathbb{E}\left[Z_\tau \middle| \mathcal{F}_t \right],$$

and in particular the value at time zero is

$$V_0 = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}} \mathbb{E}[Z_\tau].$$

For any specific stopping strategy $\tau' \in \mathcal{T}$, we have $V_0' = \mathbb{E}[Z_{\tau'}] \leqslant \operatorname*{ess\,sup}_{\tau \in \mathcal{T}} \mathbb{E}[Z_\tau] = V_0$. Hence the estimate of an American option price given by one strategy is a lower bound of the real value.

2.2 The upper bound of the option price

Denote by $\mathcal{M}^{UI}$ the set of all uniformly integrable martingales with initial state set to zero. Since the discounted option value process $(V_t)_{t \in [0,T]}$ is a supermartingale of class D, it has a unique Doob-Meyer decomposition:

$$V_t = V_0 + M_t^* - A_t^*, \tag{1}$$

where $M^* \in \mathcal{M}^{UI}$, and A^* is a predictable non-decreasing process with $A_0^* = 0$.
The American option pricing problem has a dual form:

$$V_0 = \inf_{M \in \mathcal{M}^{UI}} \mathbb{E}\left[\sup_{t \in [0,T]} Z_t - M_t \right], \tag{2}$$

and the infimum is attained at $M = M^*$. We refer to Rogers [45], Haugh and Kogan [30] for proofs of this duality.

Denote $\mathcal{M} \subset \mathcal{M}^{UI}$ as the set of martingales that are both uniformly integrable and square integrable. We restrict our search for M^* within the set $\mathcal{M}$. This does not pose a problem in our numerical experiments as the optimal martingales corresponding to the options we price satisfy this condition. Since $M^* \in \mathcal{M}$ and is adapted to the Brownian

filtration $\mathbb{F}$, the Brownian martingale representation theorem states that there exists a predictable process H with values in $\mathbb{R}$ such that $\mathbb{E}\left[\int_0^T H_s^2 \, ds\right] < \infty$, and

$$M_t^* = M_0^* + \int_0^t H_s \, dW_s. \tag{3}$$

This allows us to estimate the optimal martingale M^* by approximating the process H numerically, and then generate an upper bound of the option price.

2.3 The hedging strategy

Consider a measurable adapted process $(J_t)_{t\in[0,T]}$ with values in $\mathbb{R}^{d+1}$, where J^i is the number of units of the i-th asset held in a portfolio consisting of d risky assets and one risk-free asset. The value of the portfolio at time t is

$$U_t = J_t^0 \beta_t + \sum_{i=1}^d J_t^i S_t^i.$$

The process satisfies the condition $\int_0^T |J_u|^2 \, du = \sum_{i=1}^d \int_0^T |J_u^i|^2 \, du < \infty$ a.s, and it is a self-financing hedging strategy if

$$\frac{U_t}{\beta_t} = U_0 + \int_0^t J_u \beta_u^{-1} \sigma(u, S_u) \, dW_u. \tag{4}$$

Combining the Doob-Meyer decomposition (1) and the Brownian martingale representation (3), we obtain

$$V_t = V_0 + \int_0^t H_u \, dW_u - A_t^*. \tag{5}$$

For the portfolio to super-replicate the option, we need $\frac{U_t}{\beta_t} \geqslant Z_t$ for all $t \in [0,T]$. It is well-known that the cheapest such portfolio satisfies $U_0 = V_0$ and $\frac{U_t}{\beta_t} \geqslant V_t \geqslant Z_t$ for all $t \in [0,T]$. Comparing equations (4) and (5), we see that this can be achieved by setting

$$J_t = \frac{\beta_t H_t}{\sigma(t, S_t)}.$$

Hence, the hedging strategy J can be computed directly from the process H. The process A^* can be interpreted as the losses incurred every time the optimal exercise opportunity is missed.

3 Valuing an American option numerically

From now on, we only approximate American options by Bermudan options whose exercise times are restricted to the discrete set $t_i = t_0 + i \cdot \Delta t$, for $i \in \{1, ..., n\}$, where

$\Delta t = \frac{T}{n}$. Note that since the pricing progress proceeds backward in time, in this paper, at t_i, the previous step refers to t_{i+1} and the next step refers to t_{i-1}.

We design two algorithms based on the combination of resolutions of both the primal and the dual problem. One uses a series of neural networks, and the other one uses only one global network. To avoid any confusion, we refer to the algorithm with multiple networks as Method I, and the global one as Method II.

3.1 Method I: Multiple Neural Networks

3.1.1 The regression rule

By taking the expectation of the discounted option value conditioned on $\mathcal{F}_{t_i}$ and applying the Doob-Meyer decomposition, we have

$$V_{t_{i+1}} = \mathbb{E}\left[V_{t_{i+1}}\Big|\mathcal{F}_{t_i}\right] + \int_{t_i}^{t_{i+1}} H_u \ \mathrm{d}W_u. \tag{6}$$

In this equation, the conditional expectation is the *continuation value*, and the integral is the *martingale increment* from t_i to t_{i+1}. Since the stock price process is Markovian, both the conditional expectation and the process $(H_{t_i})_{i \in \{0,1,\dots,n\}}$ can be estimated as functions of the state variables S_{t_i} [18, 17].

Let $\Phi^i(S_{t_i}) : \mathbb{R}^d \to \mathbb{R}$ and $\Psi^i(S_{t_i}) : \mathbb{R}^d \to \mathbb{R}^d$ be approximations of the continuation function and the process H_{t_i} at t_i, respectively. We refer to $\Psi(S_{t_i})$ as the *martingale increment function*. Due to the independence among stocks, the martingale increment can be approximated by $\sum_{j=1}^{d} \beta_{t_i}^{-1} \Psi^i(S_{t_i})^j \Delta W_{t_i}^j$, where $\Delta W_{t_i}^j = W_{t_{i+1}}^j - W_{t_i}^j$. For simplicity, we leave out the superscript that indicates dimension and the symbol for summation in the rest of this section.

Based on (6), we perform regression at each time through:

$$\min_{\Phi, \Psi} \left(\beta_{t_i} V_{t_{i+1}} - \Phi^i(S_{t_i}) - \sum_{j=1}^{d} \Psi^i(S_{t_i})^j \Delta W_{t_i}^j \right)^2 .$$

In this method, one neural network is used to regress the continuation value and the martingale increment on the current stock prices at time $t_i \in \{t_0, t_1, t_2, \dots, t_{n-1}\}$. In this work, we use fully-connected feedforward neural networks to perform these regressions. Let NN^Θ denote fully connected feedforward artificial neural networks, with Θ describing the structure of a network. $\Theta = (L, [n_1, \dots, n_L])$ represents a network with L layers, and each layer l has n_l neurons. In particular, n_1 and n_L are the number of input features and the number of outputs, respectively. Each network takes the form:

$$\sigma_{n_{L-1}} \circ A_{L-1} \circ \cdots \circ \sigma_1 \circ A_1,$$

where A_l signifies an affine transformation from layer l to layer $l + 1$:

$$A_l(x) = w_l^T x + b_l,$$

for $x \in \mathbb{R}^{n_l}$, $w_l \in \mathbb{R}^{n_l \times n_{l+1}}$, $b_l \in \mathbb{R}^{l+1}$, and σ_l is the activation function applied to A_l.

3.1.2   The stopping strategy

Let $\tau_i : \Omega \to \{t_{i+1}, \ldots, t_n = T\}$ be a stopping time, and $\mathcal{T}_i$ be the set of all stopping times that takes values greater than t_i.

The optimal stopping strategy is to exercise the option once the immediate payoff is higher than the continuation value. Let $f(S_t) : \mathbb{R}^d \to \mathbb{R}$ be the payoff of the option at $t \in [0, T]$. The stopping time can be represented as:

$$\tau_i = \min\{t_j \in \{t_{i+1}..., t_{n-1}\} : f(S_{t_j}) \geqslant \Phi^j(S_{t_j})\} \wedge t_n.$$

3.1.3   The update rule

Consider two random processes $(Y_{t_i})_{i=1}^n$ and $(X_{t_i})_{i=1}^n$, defined as the following:

- At $t_n = T$, the option holder has to either exercise the option if it is in the money or let it expire if it is out of the money. Let

$$Y_{t_n} = X_{t_n} = f(S_{t_n}).$$

- At each $t \in \{t_0, t_1, \ldots, t_{n-1}\}$, the option holder either exercises the option immediately if the payoff value is higher than the continuation value, or hold it till the next exercise point if it is lower. Let

$$Y_{t_i} = \begin{cases} f(S_{t_i}) & , \text{ if } f(S_{t_i}) \geqslant \Phi^i(S_{t_i}) \\ \beta_{\Delta t}^{-1} Y_{t_{i+1}} - \Psi^i(S_{t_i}) \cdot \Delta W_{t_i} & , \text{ if } f(S_{t_i}) < \Phi^i(S_{t_i}). \end{cases}$$

$$X_{t_i} = \begin{cases} f(S_{t_i}) & \text{if } f(S_{t_i}) \geqslant \beta_{\Delta t}^{-1} X_{t_{i+1}} - \Psi^i(S_{t_i}) \cdot \Delta W_{t_i} \\ \beta_{\Delta t}^{-1} X_{t_{i+1}} - \Psi^i(S_{t_i}) \cdot \Delta W_{t_i}, & \text{if } f(S_{t_i}) < \beta_{\Delta t}^{-1} X_{t_{i+1}} - \Psi^i(S_{t_i}) \cdot \Delta W_{t_i} \end{cases} \tag{7}$$

In the update of Y_{t_i}, neglecting the subtraction term, it simply applies the stopping strategy. By averaging Y_{t_0} over all paths, we get a lower bound of the option price. The discounted payoff at the optimal stopping time is used as the regression target, which can significantly reduce the bias but introduce a higher variance. To cancel this negative effect, we subtract the martingale increment $\Psi^i(S_{t_i}) \cdot \Delta W_{t_i}$ adjusted with the time value. If the approximation of H_{t_i} is perfect, the variance can be cancelled out completely. A proof is given in Appendix A to show that this term reduces the variance of the estimate.

In the update of X_{t_i}, the subtraction of the martingale increment serves a different purpose. Note that (7) can be written as a recursive equation:

$$X_{t_i} = \max\{f(S_{t_i}),\ \beta_{\Delta t}^{-1}X_{t_{i+1}} - \Psi^i(S_{t_i}) \cdot \Delta W_{t_i}\},$$

we can have the following:

$$X_{t_0} = \max\{f(S_{t_0}),\ \beta_{\Delta t}^{-1}\max\{f(S_{t_1}),\ \ldots \beta_{\Delta t}^{-1}\max\{f(S_{t_{n-1}}), \tag{8}$$
$$\beta_{\Delta t}^{-1}f(S_{t_n}) - \Psi^{n-1}(S_{t_{n-1}})\cdot\Delta W_{t_{n-1}}\} - \cdots - \Psi^1(S_{t_1})\cdot\Delta W_{t_1}\} - \Psi^0(S_{t_0})\cdot\Delta W_{t_0}\}.$$

From the duality (2) we have:

$$V_{t_0} \leqslant \max\{Z_{t_0}, \ldots, Z_{t_{n-1}} - \sum_{i=0}^{n-2}\Delta M_{t_i}, Z_{t_n} - \sum_{i=0}^{n-1}\Delta M_{t_i}\}$$

$$= \max\{Z_{t_0}, \ldots, \max\{Z_{t_{n-1}}, Z_{t_n} - \Delta M_{t_{n-1}}\} - \sum_{i=0}^{n-2}\Delta M_{t_i}\}$$

$$= \max\{Z_{t_0}, \max\{Z_{t_1}, \ldots, \max\{Z_{t_{n-2}}, \max\{Z_{t_{n-1}},$$
$$Z_{t_n} - \Delta M_{t_{n-1}}\} - \Delta M_{t_{n-2}}\}\ldots - \Delta M_{t_1}\} - \Delta M_{t_0}\}. \tag{9}$$

Recall that $\Delta M_{t_i} \approx \beta_{t_i}^{-1}\Psi^i(S_{t_i})\Delta W_{t_i}$ and $Z_{t_i} = \beta_{t_i}^{-1}f(S_{t_i})$, then from (8) and (9) we can see that $\mathbb{E}[X_{t_0}]$ is an upper bound of the price.

The processes X_{t_i} and Y_{t_i} can also be interpreted in the following way. The variable Y_{t_i} is a proxy of the buyer's price, as the two cases correspond to the stopping decision based on comparing the exercise payoff and the continuation value. The variable X_{t_i} is a proxy of the seller's price, as the two cases correspond to whether the seller needs to update their hedging targets based on the comparison of the exercise payoff and the hedging price.

Note that in all numerical experiments shown in this paper, t_0 is not considered an exercise date, coinciding with the fact that one does not exercise the option at the initial time. This choice is reflected in the algorithm 1 by directly letting $Y_{t_0} = \beta_{\Delta t}^{-1}Y_{t_1} - \Psi^0(S_{t_0})\cdot \Delta W_{t_0}$ without checking the comparison condition. However, the regression is still performed to obtain the martingale increment function and martingale increment from ΔM_{t_0}.

3.1.4   The whole process

We outline the entire pricing process using Method I in the algorithm below. Throughout the training process, all trained models are saved for future use. Subsequently, an independent out-of-sample simulation is conducted to derive estimates. This second simulation can be executed in two ways: following the training algorithm by determining the values backward, or starting from the initial time and making decisions forward.

Algorithm 1: American Option Pricing with Multiple Neural Networks
---
Result: Functions Φ^i, Ψ^i for $i \in \{0, 1, ..., n-1\}$

Simulate N stock paths

Initialise $Y_{t_n} = X_{t_n} = \max(f(S_{t_n}), 0)$

for *i=n-1:1* **do**

     Regress $\beta_{\Delta t}^{-1} Y_{t_{i+1}}$ on S_{t_i}: $\min_{\Phi^i, \Psi^i} (\beta_{\Delta t}^{-1} Y_{t_{i+1}} - \Phi^i(S_{t_i}) - \Psi^i(S_{t_i}) \Delta W_{t_i})^2$

     $Y_{t_i} = \beta_{\Delta t}^{-1} Y_{t_{i+1}} - \Psi^i(S_{t_i}) \Delta W_{t_i}$

     $X_{t_i} = \beta_{\Delta t}^{-1} X_{t_{i+1}} - \Psi^i(S_{t_i}) \Delta W_{t_i}$

     if $f(S_{t_i}) > \Phi^i(S_{t_i})$ **then**

         $Y_{t_i} = f(S_{t_i})$

     end

     if $f(S_{t_i}) > X_{t_i}$ **then**

         $X_{t_i} = f(S_{t_i})$

     end

end

Regress $\beta_{\Delta t}^{-1} Y_{t_1}$ on S_{t_0}: $\min(\beta_{\Delta t}^{-1} Y_{t_1} - \Phi^0(S_{t_0}) - \Psi^0(S_{t_0}) \Delta W_{t_0})^2$

$Y_{t_0} = \beta_{\Delta t}^{-1} Y_{t_1} - \Psi^0(S_{t_0}) \Delta W_{t_0}$

$X_{t_0} = (\beta_{\Delta t}^{-1} X_{t_1} - \Psi^0(S_{t_0}) \Delta W_{t_0}) \mathbb{1}_{f(S_{t_0}) \leqslant X_{t_0}} + f(S_{t_0}) \mathbb{1}_{f(S_{t_0}) > X_{t_0}}$

3.1.5 Discussion on the convergence

Since its introduction, numerous studies have been conducted to analyse the convergence analysis of the LSMC method. In their original work, Longstaff and Schwartz [39] showed the convergence in cases with only two early exercise opportunities. Subsequently, Clément et al. [19] established a more general almost sure convergence by modifying the method to regress all paths instead of solely in-the-money ones. Egloff [20] showed both the convergence and error estimates by using Vapnik-Chervonenkis classes with the assumption of convexity, allowing for relaxation of linearity assumption in approximation spaces. Eventually, Zanger [49] derived a general convergence result, providing new overall error estimates for the algorithm without assuming linearity or convexity of approximation spaces, and without requiring an independent data set. This result validated the application of neural networks in the method and the avoidance of an independent data set in regression. Regarding the upper bound, Huré et al. [32] established the convergence of the method for deriving upper bounds in the context of solving reflected BSDEs.

The convergence of our method, which incorporates duality into the primal problem, follows from the literature mentioned above.

Remark 3.1. *El Karoui et al. [21] showed that pricing American options is related to*

reflected BSDEs, the solution of which is an $\mathcal{F}_t$-measurable triple (V_t, H_t, K_t) for $t \in [0, T]$
with values in $(\mathbb{R}, \mathbb{R}^n, \mathbb{R}_+)$, and satisfies:

$$
\begin{cases}
V_t = Z_T + \int_t^T b(s, V_s, H_s) \, \mathrm{d}s + K_T - K_t - \int_t^T H_s \, \mathrm{d}W_s, \\
V_t \geqslant Z_t, \quad 0 \leqslant t \leqslant T, \\
K_0 = 0, \ \ and \ \int_0^T (V_t - Z_t) \, \mathrm{d}K_t = 0.
\end{cases}
$$

Our work can be easily extended to solve this type of BSDE. The processes V and H here have the same meaning as we have defined before, and our work generates numerical solutions for them. The process K can be seen as the non-decreasing process A and calculated by a second simulation where we accumulate the gap between the value process and the payoff process. Note we have $b(\cdot, \cdot, \cdot) = 0$ in our case. However, if we have a model where $b(\cdot, \cdot, \cdot) \neq 0$, we can still approximate it by adding one more term to our regression.

3.2 Method II: One Global Neural Network

After pricing a vanilla American put option under the Black-Scholes model that has 50 exercise points using Method I, we plot $\Phi^i(\bar{S}_{t_i})$ and $\Psi^i(\bar{S}_{t_i})$, for $i \in \{0, 1, ..., 49\}$, in Figure 1, to visualise the approximated functions, where $\bar{S}_{t_i}$ is the standardised stock price. We can see that continuation functions and the martingale increment functions at different times t_i have similar shapes, and they evolve continuously in time.
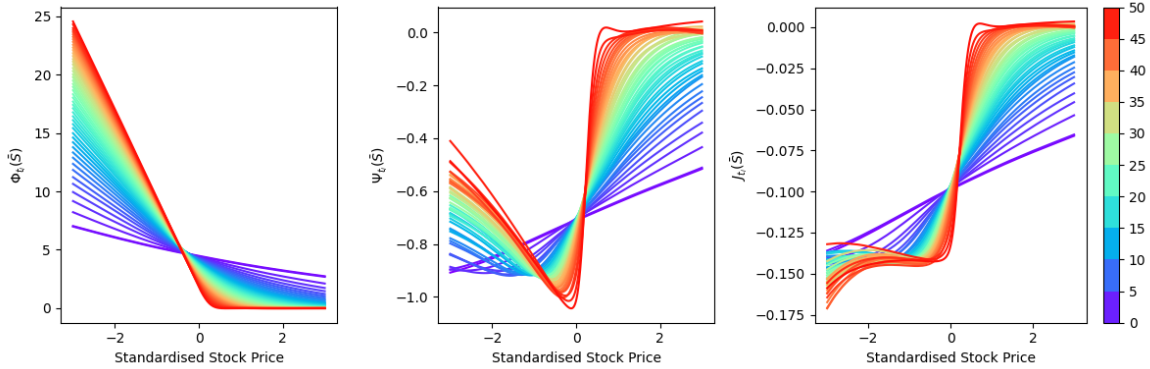

Figure 1: Estimates of continuation functions, martingale increment functions and hedging ratio of a 1D American put option with 50 exercise dates (same parameters as the one in Section 5). Each line represents a function at a step. *Left*: the continuation function; *Middle*: the martingale increment function; *Right*: the hedging ratio. The colorbar represents the step: 0 is the initial time and 50 is the maturity.

Remark 3.2. *Under the Black-Scholes model, the first derivative of a continuous function is expected to align with the hedging ratio, thereby establishing a link to the martingale increment function. While we might expect the martingale increment functions to present a*

flat trajectory near the value of -1 *along the left axis, the middle plot in Figure 1 displays a deviation from this pattern. This discrepancy arises because the plot illustrates the approximated martingale increment function of the standardised stock price, rather than the direct hedging ratio with respect to the stock price itself. Additionally, we have not imposed any constraints restriction on the shape of the function during training. The further the data points deviate from the centre, the less data is available, leading to increased extrapolation at the plot's extremities. Nonetheless, through adjustment with the diffusion term, we can represent the hedging ratio for the standardised stock price, as shown on the right plot.*

3.2.1   The whole process

Based on the similarity in the shape of functions and their continuous progression in time, we propose a second method where we only use one network for all regressions by including the time/step as an input variable.

We apply the same stopping strategy, and the regression and the updates of process X and Y at each time remain the same. However, this novel approach poses additional challenges as it requires target values at all times when we start training the model. In method I, the update of Y_{t_i} before the regression provides a relatively accurate target values for the training of the corresponding network, but this is not available in method II. To overcome this challenge, we propose a novel approach where we achieve the goal by alternating the model training and stopping strategy updates.

Initially, we set the maturity as the stopping time, so target values at $t_i \in \{t_0, \ldots, t_{n-1}\}$ are $\beta_{(n-i)\Delta t}^{-1} f(t_n)$. We train the model using these target values for a given number of epochs and then use the trained model to determine a new series of Y_{t_i} using the update rule stated before. Once all target values are updated, we do the training again. We repeat this training-updating process till some predefined criterion is met. We choose small numbers as the number of epochs among updates, especially for the earlier training, since the stopping strategies we applied are unlikely to be optimal at the start.

Denote $\Phi_{\mathrm{II}}(t_i, S_{t_i}) : \mathbb{R}_+ \times \mathbb{R}^d \longrightarrow \mathbb{R}$ and $\Psi_{\mathrm{II}}(t_i, S_{t_i}) : \mathbb{R}_+ \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ as the approximations of the continuation functions and the martingale increment functions. Method II is summarised in Algorithm 2.

Algorithm 2: American Option Pricing with One Global Network

Result: Functions Φ_{II}, Ψ_{II}

Simulate N stock price paths

Initial: $Y_{t_{i+1}} = \beta^{-1}_{(n-i+1)\Delta t} f(S_{t_n})$, for $i \in \{0, ..., n-1\}$

while stopping criterion is not met **do**

    for $i = 1:$ epoch **do**

        Regress $\beta^{-1}_{\Delta t} Y_{t_{i+1}}$ on $(t_i, S_{t_i}, \Delta W_{t_i})$ for $i \in \{0, ..., n-1\}$:

        $\min\limits_{\Phi_{\text{II}}, \Psi_{\text{II}}} \left(\beta^{-1}_{\Delta t} Y_{t_{i+1}} - \Phi_{\text{II}}(t_i, S_{t_i}) - \Psi_{\text{II}}(t_i, S_{t_i}) \Delta W_{t_i} \right)^2$

    end

    $Y_{t_n} = X_{t_n} = f(S_{t_n})$

    for $i = n-1:1$ **do**

        $Y_{t_i} = \beta^{-1}_{\Delta t} Y_{t_{i+1}} - \Psi(t_i, S_{t_i}) \Delta W_{t_i}$

        if $f(S_{t_i}) > \Phi_{II}(t_i, S_{t_i})$ **then**

            $Y_{t_i} = f(S_{t_i})$

        end

        $X_{t_i} = \beta^{-1}_{\Delta t} X_{t_{i+1}} - \Psi_{\text{II}}(t_i, S_{t_i}) \Delta W_{t_i}$

        if $f(S_{t_i}) > X_{t_i}$ **then**

            $X_{t_i} = f(S_{t_i})$

        end

    end

    $Y_{t_0} = \beta^{-1}_{\Delta t} Y_{t_1} - \Psi_{\text{II}}(t_0, S_{t_0}) \Delta W_{t_0}$

    $X_{t_0} = (\beta^{-1}_{\Delta t} X_{t_1} - \Psi_{\text{II}}(t_0, S_{t_0}) \Delta W_{t_0}) \mathbb{1}_{f(S_{t_0}) \leqslant X_{t_0}} + f(S_{t_0}) \mathbb{1}_{f(S_{t_0}) > X_{t_0}}$

end

3.2.2 Discussion on the convergence

While single global networks have been used to address a wide range of optimal stopping problems, the convergence analysis for backward methods remains lacking. Tsitsiklis and Van Roy [48], Herrera et al. [31] have proposed similar approaches to price American options with some insights into the convergence properties. A key distinction between our method to theirs is that they use estimated continuation value rather than the exact optimal payoff to make stopping decisions, and they start with a completely random initial strategy. They showed that the method converges eventually, but there are no results on the rate of convergence and the error bounds.

The rationale behind our proposed method is that when the initial strategy is to wait until maturity, we have relatively accurate target values for decisions made closer to the maturity date. In particular, the target values for determining the second-to-last exercise decisions would be exact. The use of the exact optimal payoff avoids the reliance on

estimated continuation values for making stopping decisions. As training progresses, the stopping decisions at later times will improve first, which in turn, improves the accuracy of target values for earlier stopping decisions. This iterative refinement ensures that, over time, the target values for all times converge to their true values, enhancing the overall decision-making process.

4 Algorithm variants

There are two sources of errors in our methods. Firstly, there is the time discretisation error induced by approximating the continuous martingale using the Euler scheme. This error scales proportionally with the step size square root $\sqrt{\Delta t}$, potentially resulting in suboptimal upper bounds in cases where the option offers infrequent exercise opportunities. The other source is regression, which can be mitigated by using a larger data set, utilizing more suitable network architectures, and prolonging the training duration. However, these approaches come at the expense of increased computational costs and memory requirements.

To enhance the performance of our algorithms, we introduce five different variants aimed at generating more accurate results, reducing computational cost and addressing memory exhaustion issues. In this section, we present numerical results to evaluate the effectiveness of each variant in pricing 1D put options, 5D max-call options, or both. These options share parameters with those presented in Section 5. The objective of this section is to assess the impact of different variants on our methods through comparisons with the original version.

Variation 1: add a second term for martingale increment approximation

The approximation $\sum_{j=1}^{d} \Psi^i(S_{t_i})^j \Delta W_{t_i}^j$ deteriorates with the step size increasing. To improve the accuracy of the martingale increment estimates, we propose to add one more term in the regression. The choice of the term depends on the model, provided it satisfies the martingale property of having a zero mean increment. We choose $(\Delta W_{t_i})^2 - \Delta t$ in our work, which can be connected to the Milstein scheme. This variant requires more outputs from the network and results in a change of the loss function:

$$
\min_{\Phi, \Psi_1, \Psi_2} \left(\beta_{\Delta_t}^{-1} Y_{t_{i+1}} - \Phi(S_{t_i}) - \sum_{j=1}^{d} \Psi_1^i(S_{t_i})^j \Delta W_{t_i}^j - \sum_{j=1}^{d} \Psi_2^i(S_{t_i})^j ((\Delta W_{t_i}^j)^2 - \Delta t) \right)^2 .
$$

The updates of X and Y need to be changed accordingly as the martingale increment becomes the sum of two terms. This variation can be applied to both methods.

The changes in results introduced by this variation applied to method I are shown in Table 1. We can see that with similar training times in each case, this variant significantly

reduces the gap between the lower and the upper bound, mainly caused by better approximations of the upper bound. The lower bound also improves due to a more effective variance reduction. Additionally, this improvement is more pronounced when the pricing problem is more complicated.

			LB		UB		Diff	
		Time	Mean	S.D.	Mean	S.D	Mean	S.D
1D	1 Term	34	4.4748	0.0007	4.5559	0.0022	0.0811	0.0024
	2 Terms	34	4.4765	0.0002	4.4936	0.0017	0.0171	0.0018
5D	1 Term	56	26.1372	0.0090	28.2132	0.0169	2.0761	0.0177
	2 Terms	52	26.1464	0.0039	26.8974	0.0074	0.7510	0.0081

Table 1: Option Pricing with/without a second martingale increment term applied in method I. The first column indicates the option type, and the second column shows whether a second term is added in the martingale increment approximation. The networks in the 1D case have three hidden layers of 30 neurons, and the one in the 5D case has two layers of 64 neurons.

Variation 2: add sub-steps

Refining the martingale approximation can also be achieved by reducing the step size. In the context of Bermudan option pricing, this can be achieved by adding substeps between two exercise times, where we do not make stopping decisions but only accumulate martingale increments. This variant is particularly important in pricing options with less frequent exercise opportunities. The 5D max-call option we have been pricing has only 9 exercise opportunities over 3 years. As demonstrated in Figure 2, adding substeps markedly enhances the accuracy of the upper bound estimation when pricing this option. The initial introduction of substeps brings about a notably sharper improvement in the bounds, which tends to taper off as more substeps are added. However, it is important to note that the computational time increases with the addition of substeps, as also depicted in Figure 2, but the speed of increasing is slower than linear.

Figure 2 also indicates that Method II produces better results with slower training, but this observation can vary with the adjustments of training parameters. Further comparisons between these two methods will be detailed throughout this paper, forming a conclusion at the end.

Variation 3: use separate networks for the two functions

In our approaches, we initially utilized a singular network to estimate both the continuation value function and the martingale increment function. However, given the potential
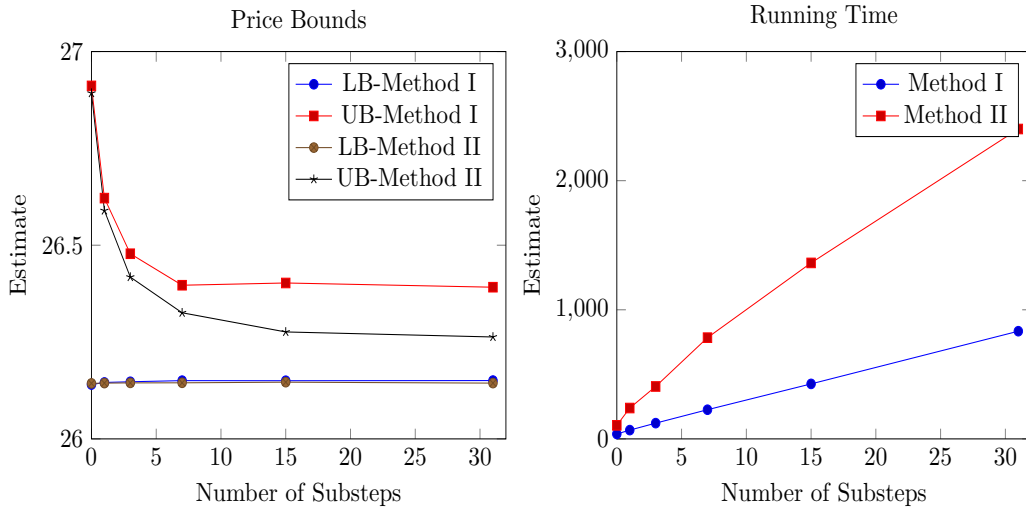
Figure 2: Price bounds (*Left*) and corresponding running times (*Right*) of a 5D max-call Bermudan option with different numbers of substeps using both method I and II.

complexity difference between these functions, especially when the model gets more complicated and the dimension gets higher, we propose to use separate networks to approximate them, where one is dedicated to generating the continuation value and the other for calculating the martingale increment functions. To evaluate the efficacy of this variant, we applied it to Method I across three different scenarios: a 1D put, a 5D max-call with no substep, and a 5D max-call with 31 substeps. For each scenario, we ensured that the networks had a comparable number of parameters. We can see from Table 2 that variant 3 can produce more accurate results with less training time in all three cases, and this effect is more notable in more complex problems (5D max-call option with 32 substeps). When implementing this variant with Method II, we observed a similar pattern, reinforcing the benefits of employing separate networks for approximating distinct functions.

Variation 4: train on data from parts of the exercise times

In Method II, the standard practice involves training the model across all simulated paths at every timestep. Anticipating that data shares similarities across different times, we suggest an alternative strategy that focuses on training with data from selectively chosen timesteps. This approach hinges on the premise that not every timestep contributes uniquely to model accuracy, allowing for strategic data reduction. Two methodologies are proposed for selecting which timesteps to include in the training process: a random selection or a systematic, evenly-spaced grid approach. For example in a scenario with 50 exercise opportunities and the aim is to train the model using data from only half of the exercise times, we could either randomly choose 25 timesteps from the set $\{0, 1, 2, ..., 49\}$ or use data

	Separate	Time		LB		UB		Diff
			Mean	S.D.	Mean	S.D	Mean	S.D
1D	False	33	4.4766	0.0001	4.4929	0.0014	0.0162	0.0014
	True	29	4.4757	0.0005	4.4899	0.0009	0.0141	0.0009
5D0S	False	65	26.1318	0.0054	26.9200	0.0070	0.7883	0.0105
	True	64	26.1138	0.0066	26.8869	0.0055	0.7731	0.0073
5D31S	False	1179	26.1528	0.0012	26.2887	0.0110	0.1359	0.0113
	True	994	26.1527	0.0011	26.2263	0.0028	0.0736	0.0026

Table 2: Options pricing with/without using separate networks in Method I. In the first column, 1D, 5D0S, and 5D31S represent the option priced: the 1D American put, the 5D max-call Bermudan with no substep and the 5D max-call Bermudan with 31 substeps. The second column indicates whether separate networks are used.

from every other timestep, i.e. $t = t_1, t_3, ..., t_{49}$.

In Figure 3, we illustrate the impact of this timestep selection strategy on training duration and the accuracy of the results when pricing a 1D American put. This modification clearly reduces the computational cost but also compromises the accuracy of the results. While this trade-off is anticipated, our goal is to strike a balance between computational efficiency and result accuracy.
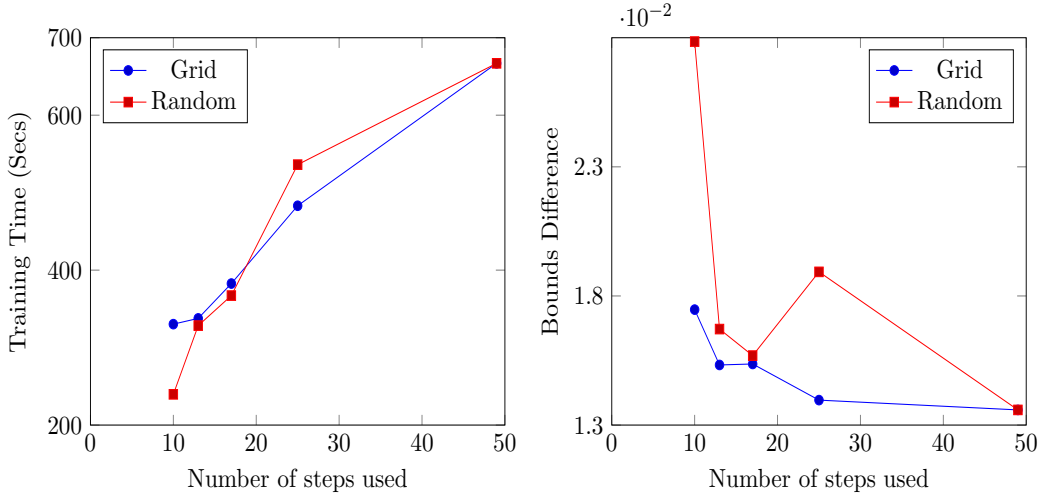

Figure 3: The changes in the estimated bounds and the training time with different numbers of timesteps used in training. The option priced is the 1D American put option.

Variation 5: generate fresh data while training

In general, larger training sets often yield more accurate and robust results, albeit at the expense of increased computational demands. However, due to the nature of the problem, we have to simulate the whole path before the training. The memory requirement can become extremely high, particularly in high-dimensional problems. To address this challenge, Chan et al. [14], Aïd et al. [1] recommended storing the random seed used during simulation. This enables us to only preserve data points at one step in a path and discard the remainder. Once the network is trained, the state values for subsequent networks are reconstructed using the saved seed and the current states. However, this introduces more calculation during training and can only be applied to Method I. To overcome these limitations, we introduce an alternative solution designed to circumvent the memory constraints in scenarios where Method II is employed, where data from all times are needed when training starts.

In the original Method II, all N paths are generated at the start. Among updates, the data set is split into the training and the validation set randomly and the training set is then grouped into batches of size N_{batch}. The network is then trained for a given number of epochs by looping over all training batches in each epoch. The validation set is then used to check the stopping criteria after each update. With this variant, we only generate the validation set before the start of the training, serving the same purpose as in the original version. Among updates, we generate N_{batch} paths, and train the network using this batch for a given number of epochs, and then discard them. We repeat this generating-training-discarding process multiple times before the stopping criteria are evaluated. By utilizing smaller batch sizes and continually generating additional paths as needed, we can effectively train on a larger number of paths without encountering memory exhaustion issues.

Figure 4 shows the difference between the lower and the upper bound of the option price throughout the training process for both the 1D put option (upper two plots) and the 5D max-call option (bottom two plots). The left two plots show these differences when employing various numbers of batches among updates. Initially, a higher number of batches leads to more favourable results. The difference diminishes when we train the model for a longer time. However, there is no definite conclusion on the optimal number of batches. The right two plots illuminate the difference using Method I, Method II, and Method II with variation 5 and 25 batches among updates. In all three cases, the second martingale term and separate networks are applied. All three schemes produce satisfactory results in pricing the 1D put option, but the base of Method II performs worse when pricing the 5D max-call option. We can see that Method I is more stable and converges faster among all three schemes, variant 5 exhibits superior performance to the base version.

Discussion

We summarise the contributions of each variation to our methods in Table 3 based on numerous experiments that were conducted apart from the ones demonstrated in this section. The conclusion is generic. The second column indicates to which method one variation can be applied. There are three aspects one variation can affect: the accuracy of estimates, the training time and the computational memory required. We use ✓ and ✗ to indicate an improvement and a deterioration respectively. If a variation does not significantly impact one of these aspects, the corresponding cell remains blank.

From Table 3, we can see that variant 1 and variant 3 can improve the accuracy without prolonging the training time. Variant 2 improves the accuracy at the expense of the computational speed, while the opposite is true for variant 4. Variation 5 is the only one that helps us overcome the memory exhaustion problem, and compared to the base of Method II, it also produces narrower bounds differences with shorter running time. All variations that can be applied to one method can be used at the same time to combine their effects.

In addition to these five variants, we also tested a warm-start approach applied to Method I, where previously trained network is used as the initial network at the next step. Since it is a standard method in the field, its impact is detailed in Appendix B.

Variations	Method	Accuracy	Time	Memory
V1: Add a second martingale term	I, II	✓		
V2: Add sub-steps	I, II	✓	✗	
V3: Use two separate networks	I, II	✓		
V4: Train on partial data	II	✗	✓	
V5: Train on fresh data	II	✓	✓	✓

Table 3: Algorithm Variants and their impacts on three different aspects. ✓ and ✗ represent an improvement and deterioration, respectively.

5   Numerical Results

This section presents the numerical results obtained through both methods we proposed, incorporating variations 1 and 3 in Method I, and variations 1, 3, and 5 in Method II, The warm-start training has also been applied to Method I. In addition, substeps are used across all cases except the 1D put option. Our experimental setup includes the use of the ADAM optimizer, and mean squared error for the loss function. Softplus is chosen as the activation function due to its smoothness property. We standardise all input variables, except for the time variable in Method II. To mitigate overfitting, cross-validation is rigorously applied

throughout the training phase. The out-of-sample test set has 10^6 paths across all scenarios. Networks with different structures were used in different cases, as detailed in each subsection. The selection of a specific network is based on extensive experimentation. We choose the ones that require the least hyperparameter tuning. Computations were executed on an NVIDIA P100 GPU under the system Intel Xeon-E5-2680-v4. The program is written in Python 3.8.5 using PyTorch 1.8.

Subsequent subsections demonstrate statistics of the pricing results for each option priced by repeating the process 10 times, including means and standard deviations of the lower bound, the upper bound and their difference. The total running time (in seconds) for each repetition is also recorded.

Additionally, we plot histograms to depict both the total hedging errors ε_1 and the worst hedging errors ε_2 using an independent dataset containing 10^4 paths. These metrics are computed as follows. Let τ^i be the stopping time for path i. The error for that path at τ^i is defined as

$$\varepsilon_1^i = V_0 + \sum_{t=t_0}^{\tau^i - \Delta t} \beta_t^{-1} H(S_t^i) \Delta W_t^i - \beta_{\tau^i}^{-1} Z_{\tau^i}^i,$$

and the worst error is defined as

$$\varepsilon_2^i = V_0 + \min_{t^i \in \{t_1, \ldots, t_n\}} \left(\sum_{t=t_0}^{t^i - \Delta t} \beta_t^{-1} H(S_t^i) \Delta W_t^i - \beta_{t^i}^{-1} Z_{t^i}^i \right).$$

5.1 Options under Black–Scholes Models

Consider American options with d underlying assets, whose prices follow the dynamics

$$\mathrm{d}S_t^i = (r - \delta^i) S_t^i \, \mathrm{d}t + \sigma^i S_t^i \, \mathrm{d}W_t^i,$$

for $i \in \{1, 2, \ldots, d\}$, where the risk-free interest rate $r \in \mathbb{R}$, the dividend rate $\delta^i \in \mathbb{R}$ and the volatility $\sigma^i \in \mathbb{R}^+$. Each Brownian Motion is independent of the others.

5.1.1 1D American Put Option

We first test our method on a 1D vanilla American put option with the following parameters:

$$T = 1, K = 40, n = 50, S_0 = 36, r = 0.06, \delta = 0, \sigma = 0.2$$

where T is the maturity, K is the strike price, and n is the number of exercise opportunities. We use the same notation for all cases in this section. The payoff function at t is

$$f(S_t, K) = (K - S_t)^+.$$

We use 10^5 paths to train the model.

The benchmark computed by the finite difference method is 4.478. The results generated by our schemes are shown in Table 4. In method I, at each time the training ceases once the loss of the validation set stagnates for 20 epochs. In method II, the training stops when the validation set loss stagnates for more than 5 updates, and we train 20 batches for 20 epochs among updates. Method I uses two networks with the structures ([1, 20, 20, 1],[1, 20, 20, 2]) at each time, while Method II employs a total of two networks with structures ([2, 20, 20, 20, 1], [2, 20, 20, 20, 2]). The total numbers of parameters trained are 49150 and 1863 for Methods I and II, respectively. From the table, we can see that both methods generate tight bounds. Though Method I exhibits shorter training times, it involves significantly more free variables in the training process. Figure 5 shows the hedging errors, from which we can see that hedging errors are tightly centred and have a bell shape.

		LB		UB		Diff	
	Time	Mean	S.D.	Mean	S.D	Mean	S.D
I	70	4.4770	0.0003	4.4899	0.0006	0.0129	0.0008
II	93	4.4749	0.0008	4.4880	0.0008	0.0131	0.0014

Table 4: 1D American put option pricing using both schemes. Benchmark estimate: 4.478. The first column indicates the method used. Method I uses two networks of the structure ([1, 20, 20, 1],[1, 20, 20, 2]) at each time, while Method II uses in total two networks of the structure ([2, 20, 20, 20, 1], [2, 20, 20, 20, 2]).

5.1.2 High Dimensional Bermudan max-call option

Consider an option with d underlying assets. We assume there is no correlation between Brownian Motions W^i and W^j, $i, j \in \{1, 2, ..., d\}$, on which each stock price is based. The model has the following parameters:

$$T = 3, K = 100, n = 9, r = 0.05, \delta^i = 0.1, \sigma = 0.2.$$

The payoff of this option is

$$\left(\max_{i \in \{1,2,...,d\}} S_t^i - K \right)^+ .$$

Given the sizable step interval of $\Delta t = \frac{1}{3}$, we engage variation 2, employing 32 substeps in our experiments. In Table 6, we present the pricing results of max-call with three different initial stock prices in both 5D and 10D settings. The benchmark given is extracted from Becker et al. [8], including the approximated bounds where the left (right) value is the lower (upper) bound (the number on the top), alongside the aggregate computation time (the summation at the bottom). The numbers in the summation are calculation

time in seconds for lower bounds, upper bounds and hedging strategies, respectively. The benchmark duration for hedging reflects the time to formulate a complete hedging strategy from 0 to T with 96 substeps, chosen due to the resemblance of its hedging error plots to ours, as illustrated in Figure 6.

In Method I, training concludes when the validation set's loss ceases to decrease after 5 epochs. In method II, the training stops when the validation set loss stagnates for more than 5 updates, and we train 20 batches for 20 epochs among updates. Method I consistently employs 10^6 training paths, while Method II's path count varies due to the nature of the scheme, as detailed in Table 5.

d	$S_0 = 90$	$S_0 = 100$	$S_0 = 110$
5	1.189×10^7	1.083×10^7	1.239×10^7
10	7.98×10^6	1.056×10^7	9.4×10^6

Table 5: The number of training paths used in pricing the Bermudan max-call option using method II. The first column shows the number of underlying assets.

Table 6 presents the empirical results from both methods, aligning closely with the benchmarks yet achieving quicker total computation times. Notably, our lower bounds invariably remain beneath our upper bounds, while there is a contradicting case in their results. We can also see that our methods become more competitive in the $10D$ case, and this is particularly true for Method II which can generate tighter bounds with less running time. This advantage is expected to become increasingly significant in more complicated cases, given its lower requirement for computing resources. Figure 5 depicts the hedging errors, which are tightly centred and display a bell shape.

5.2 American Put Option under Heston Model

Finally, we test our methods under the Heston model, where the volatility itself is also stochastic:
$$\begin{cases} \mathrm{d}S_t & = & rS_t \, \mathrm{d}t + \sqrt{V_t}S_t \, \mathrm{d}W_S, \\ \mathrm{d}V_t & = & \lambda(\sigma^2 - V_t) \, \mathrm{d}t + \xi\sqrt{V_t} \, \mathrm{d}W_V. \end{cases}$$
The option we price is the same as the one in Lapeyre and Lelong [37], characterized by the parameters:

$$T = 1, K = 100, n = 10, S_0 = 100, V_0 = 0.01, r = 0.1, \sigma = 0.1, \lambda = 2, \xi = 0.2, \rho = -0.3,$$

where ρ is the correlation between Brownian Motions W_S and W_V.

Since there are two Brownian motions involved in this scenario and we apply variation 1 for enhanced precision, we have

$$\Psi_1^S(S_{t_i})\Delta W_{t_i}^S + \Psi_1^V(S_{t_i})\Delta W_{t_i}^V + \Psi_2^S(S_{t_i})\Delta((W_{t_i}^S)^2 - \Delta t) + \Psi_2^V(S_{t_i})\Delta((W_{t_i}^V)^2 - \Delta t)$$

				LB		UB		Diff		
d	S0		Time	Mean	S.D.	Mean	S.D	Mean	S.D	Benchmark
5	90	I	1002	16.6377	0.0009	16.6862	0.0023	0.0484	0.0026	(16.644, 16.648)
		II	1127	16.6314	0.0030	16.6856	0.0025	0.0542	0.0040	132+8+1546
	100	I	1022	26.1523	0.0012	26.2195	0.0021	0.0672	0.0029	(26.156, 26.152)
		II	1034	26.1411	0.0050	26.2259	0.0029	0.0848	0.0067	134+8+1668
	110	I	1177	36.7551	0.0252	36.8724	0.0068	0.1173	0.0311	(36.780, 36.796)
		II	1038	36.7767	0.0013	36.8646	0.0024	0.0879	0.0024	133+8+1673
10	90	I	989	26.2613	0.0057	26.4823	0.0177	0.2210	0.0226	(26.277, 26.283)
		II	788	26.2446	0.0200	26.3822	0.0162	0.1376	0.0353	136+8+1792
	100	I	1035	38.3503	0.0067	38.5974	0.0286	0.2471	0.0345	(38.355, 38.378)
		II	1045	38.3159	0.0235	38.4894	0.0144	0.1735	0.0375	136+7+1803
	110	I	1023	50.8961	0.0047	51.1810	0.0184	0.2849	0.0228	(50.869, 50.932)
		II	928	50.8764	0.0151	51.0601	0.0083	0.1837	0.0197	135+8+1777

Table 6: Bermudan max-call option pricing using both methods. The first and the second columns show the number of underlying assets and the initial stock price, respectively. The third column indicates the method used. Benchmarks in the last column are extracted from [8]. The top value is the estimated price bounds, the three values shown below are the times for deriving a lower bound, an upper bound and a hedging strategy. Method I uses two networks with structure ([d, 50, 25, 1],[d, 50, 50, 2*d]) at each time, while Method II uses in total two networks with structure ([d+1, 50, 50, 50, 1], [d+1, 50, 50, 50, 2*d])

as our martingale increment. Similar to the max-call option in section 5.1.2, the step size $\Delta t = 0.1$ is big, so we use substeps for the implementation.

Figure 7 shows the change in the estimates with an increasing number of substeps using both methods. We can see that both lower and upper bounds decrease. This is not only because the martingale approximation improves with decreasing step size, but also because the Heston model simulation becomes more accurate. As the number of steps increases, the estimated bounds decrease, whereas the computational time rises. Given the insights from Figure 7, we choose to add 9 substeps since further improvement becomes trivial while significantly increasing computational expenses.

Table 7 shows the results from both methods with 9 substeps using different numbers of training paths. The estimated bounds we generated are tight, and the computation time can be very small with the adjustment of the training path. These results exhibit slight deviations from the benchmark, potentially attributable to model simulation variations, as indicated by the diminishing estimates with increased substeps.

			LB		UB		Diff	
	Path	Time	Mean	S.D.	Mean	S.D	Mean	S.D
I	1×10^5	36	1.6416	0.0001	1.6659	0.0033	0.0244	0.0033
	1×10^6	294	1.6419	0.0001	1.6514	0.0005	0.0094	0.0005
II	9.45×10^5	28	1.6364	0.0030	1.6979	0.0271	0.0615	0.0291
	1.72×10^7	348	1.6406	0.0007	1.6521	0.0007	0.0115	0.0008

Table 7: 1D American put option under Heston model with 9 substeps added. The benchmark from [37] is 1.7 ± 0.0016. The second column indicates the number of training paths employed. Method I uses two networks with structure ([2, 50, 50, 1],[2, 50, 50, 4]) at each time, while Method II uses in total two networks with structure ([3, 50, 50, 50, 1], [3, 50, 50, 50, 4])

6 Conclusion

In this paper, we introduce two innovative approaches aimed at simultaneously addressing the American option pricing problem and its dual form, providing both lower and upper bounds on the option price using deep learning using neural networks. Both methods are based on the least squares Monte Carlo method with the incorporation of duality. The first method employs a series of networks to approximate the continuation values and martingale increments at each exercise time. The second method applies one global network by adding time as a state variable to perform the regression and alternates the network training and the update of the stopping strategy till a stopping criterion is met. We propose several variants to enhance the methods from different perspectives. One notable advantage of our methods is that nested simulations are avoided, significantly reducing the computation complexity when pricing American/Bermudan options that have frequent exercise opportunities. Moreover, the methods naturally yield hedging strategies, serving as effective control variates for variance reduction.

Although the numerical results predominantly rely on the geometric Brownian Motion, it is important to emphasize that the applicability of our methods extends beyond this model. Our methods can take any model that can be simulated and satisfy conditions of the martingale representation theorem such that martingale increments can be approximated. The demonstrated effectiveness in pricing options within the Heston model underscores the versatility of our approaches. This property of our methods provides a ground for exploration, encouraging their application to problems in more complicated models.

From the results shown, we can see that both methods yield tight bounds for the approximated option price in both low and high-dimensional cases. Though the training process can be time-intensive for high-dimensional problems, the resulting models can be directly used to derive a hedging strategy without additional effort. In both methods, the inclusion

of a second martingale increment term and the introduction of substeps for options with less frequent exercise points play important roles in improving the accuracy. In conclusion, Method I demonstrates greater stability and yields narrower bounds differences. However, its performance diminishes as the complexity of the problem increases and the required number of training paths grows too large. On the other hand, Method II with the application of variation 5, effectively overcomes these challenges. This is evidenced by its successful pricing of the 10D max-call option. Further exploration of this variant could be conducted to fully assess its capabilities and potential enhancements.

Code availability

Our code is openly available at:
https://github.com/JiahaoWu27/American-Option-Pricing.git

References

[1] R. Aïd, L. Campi, N. Langrené, and H. Pham. A probabilistic numerical method for optimal multiple switching problems in high dimension. *SIAM Journal on Financial Mathematics*, 5(1):191–231, 2014.

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

[3] V. Bally, G. Pagès, and J. Printems. A quantization tree method for pricing and hedging multidimensional American options. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 15(1):119–168, 2005.

[4] J. Barraquand and D. Martineau. Numerical valuation of high dimensional multivariate American securities. *Journal of Financial and Quantitative Analysis*, 30(3):383–405, 1995.

[5] C. Bayer, D. Belomestny, P. Hager, P. Pigato, and J. Schoenmakers. Randomized optimal stopping algorithms and their convergence analysis. *SIAM Journal on Financial Mathematics*, 12(3):1201–1225, 2021.

[6] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *Discrete and Continuous Dynamical Systems - Series B*, 2022.

[7] S. Becker, P. Cheridito, and A. Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 20(74):1–25, 2019.

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

[9] D. Belomestny, C. Bender, and J. Schoenmakers. True upper bounds for Bermudan products via non-nested Monte Carlo. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 19(1):53–71, 2009.

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

[11] B. Bouchard and X. Warin. Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods. In *Numerical methods in finance*, pages 215–255. Springer, 2012.

[12] M. Broadie, P. Glasserman, et al. A stochastic mesh method for pricing high-dimensional American options. *Journal of Computational Finance*, 7:35–72, 2004.

[13] J. F. Carriere. Valuation of the early-exercise price for options using simulations and nonparametric regression. *Insurance: Mathematics and Economics*, 19(1):19–30, 1996.

[14] R. H. Chan, C.-Y. Wong, and K.-M. Yeung. Pricing multi-asset American-style options by memory reduction Monte Carlo methods. *Applied Mathematics and Computation*, 179(2):535–544, 2006.

[15] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. Machine learning for semi linear PDEs. *Journal of scientific computing*, 79(3):1667–1712, 2019.

[16] 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(1):45–67, 2021.

[17] R. Chitashvili and M. Mania. On functions transforming a Wiener process into a semimartingale. *Probability Theory and Related Fields*, 109(1):57–76, 1997.

[18] E. Çinlar, J. Jacod, P. Protter, and M. J. Sharpe. Semimartingales and Markov processes. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 54(2):161–219, 1980.

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

[20] D. Egloff. Monte Carlo algorithms for optimal stopping and statistical learning. *Ann. Appl. Probab*, 15(2):1396–1432, 2005.

[21] N. El Karoui, E. Pardoux, and M. Quenez. American options. *Numerical methods in finance*, 13:215, 1997.

[22] S. Fécamp, J. Mikael, and X. Warin. Risk management with machine-learning-based algorithms. *arXiv preprint arXiv:1902.05287*, 2019.

[23] 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(4):1290–1303, 2023.

[24] M. Germain, H. Pham, and X. Warin. Neural networks-based algorithms for stochastic control and PDEs in finance. In *Machine Learning and Data Sciences for Financial Markets: a Guide To Contemporary Practices*, pages 426–452. Cambridge University Press, 2023.

[25] E. Gobet and R. Munos. Sensitivity analysis using Itô–Malliavin calculus and martingales, and application to stochastic optimal control. *SIAM Journal on control and optimization*, 43(5):1676–1713, 2005.

[26] L. Gonon. Deep neural network expressivity for optimal stopping problems. *arXiv preprint arXiv:2210.10443*, 2022.

[27] L. Goudenège, A. Molent, and A. Zanette. Variance reduction applied to machine learning for pricing Bermudan/American options in high dimension. *arXiv preprint arXiv:1903.11275*, 2019.

[28] L. Goudenège, A. Molent, and A. Zanette. Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models. *Quantitative Finance*, 20(4):573–591, 2020.

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

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

[31] C. Herrera, F. Krach, P. Ruyssen, and J. Teichmann. Optimal stopping via randomized neural networks. *Frontiers of Mathematical Finance*, 3(1):31–77, 2024.

[32] C. Huré, H. Pham, A. Bachouch, and N. Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557, 2021.

[33] S. Jain and C. W. Oosterlee. The stochastic grid bundling method: efficient pricing of Bermudan options and their Greeks. *Applied Mathematics and Computation*, 269: 412–431, 2015.

[34] F. Jamshidian. Numeraire-invariant option pricing and American, Bermudan, trigger stream rollover. In *4th Winter School on Financial Mathematics*, 2004.

[35] M. Kohler, A. Krzyżak, and N. Todorovic. Pricing of high-dimensional American options by neural networks. *Mathematical Finance*, 20(3):383–410, 2010.

[36] A. Kolodko and J. Schoenmakers. Iterative construction of the optimal Bermudan stopping time. *Finance and Stochastics*, 10(1):27–49, 2006.

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

[38] V. Lokeshwar, V. Bharadwaj, and S. Jain. Explainable neural network for pricing and universal static hedging of contingent claims. *Applied Mathematics and Computation*, 417:126775, 2022.

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

[40] M. Ludkovski. Kriging metamodels and experimental design for Bermudan option pricing. *Journal of Computational Finance*, 22(1):37–77, 2018.

[41] A. S. Na and J. W. Wan. Efficient pricing and hedging of high-dimensional American options using deep recurrent networks. *Quantitative Finance*, 23(4):631–651, 2023.

[42] M. Raissi. Forward–backward stochastic neural networks: deep learning of high-dimensional partial differential equations. In *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, pages 637–655. World Scientific, 2024.

[43] A. M. Reppen, H. M. Soner, and V. Tissot-Daguette. Neural optimal stopping boundary. *arXiv preprint arXiv:2205.04595*, 2022.

[44] L. Rogers. Dual valuation and hedging of Bermudan options. *SIAM Journal on Financial Mathematics*, 1(1):604–608, 2010.

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

[46] J. Schoenmakers, J. Zhang, and J. Huang. Optimal dual martingales, their analysis, and application to new algorithms for Bermudan products. *SIAM Journal on Financial Mathematics*, 4(1):86–116, 2013.

[47] J. Tilley. Valuing American options in a path simulation model. *Transactions of the Society of Actuaries*, 45:499–519, 1993.

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

[49] D. Z. Zanger. Convergence of a least-squares Monte Carlo algorithm for American option pricing with dependent sample data. *Mathematical Finance*, 28(1):447–479, 2018.

A   Variance Reduction

We have mentioned the process H can be approximated by a function of the stock price due to the Markovian property of the stock processes. From now on, let $H_t = H(S_t)$.

Proposition A.1. *Given the option has not been exercised at $t \in [0, T)$. Let $\tau^* \in \mathcal{T}_t$ be the optimal stopping time. The martingale increment $\int_t^{\tau^*} H(S_u) \, \mathrm{d}W_u$ can be used as control variate to reduce variance.*

Proof. Since V_{τ^*} is $\mathcal{F}_T$ measurable, we can apply martingale representation:

$$V_{\tau^*} = \mathbb{E}[V_{\tau^*}] + \int_0^{\tau^*} H(S_u) \, \mathrm{d}W_u + \int_{\tau^*}^T H(S_u) \, \mathrm{d}W_u \tag{10}$$

By taking expectations on both sides of (10) conditioned on $\mathcal{F}_{\tau^*}$, we can get

$$\mathbb{E}\left[V_{\tau^*} \Big| \mathcal{F}_{\tau^*}\right] = \mathbb{E}[V_{\tau^*}] + \int_0^{\tau^*} H(S_u) \, \mathrm{d}W_u + \mathbb{E}\left[\int_{\tau^*}^T H(S_u) \, \mathrm{d}W_u | \mathcal{F}_{\tau^*}\right]$$

Note that $\int_{\tau^*}^T H(S_u) \, \mathrm{d}W_u = 0$. This can be interpreted through the theory of hedging. We have $H_t = 0$ for $t \in (\tau^*, T]$ because we stop hedging once the option is exercised. Hence,

$$V_{\tau^*} = \mathbb{E}[V_{\tau^*}] + \int_0^t H(S_u) \, \mathrm{d}W_u + \int_t^{\tau^*} H(S_u) \, \mathrm{d}W_u. \tag{11}$$

We then take expectations on both sides of (11) conditioned on $\mathcal{F}_t$:

$$\mathbb{E}[V_{\tau^*}|\mathcal{F}_t] = \mathbb{E}[\mathbb{E}[V_{\tau^*}]\,|\mathcal{F}_t] + \mathbb{E}\left[\int_0^t H(S_u) \, \mathrm{d}W_u|\mathcal{F}_t\right] + \mathbb{E}\left[\int_t^{\tau^*} H(S_u) \, \mathrm{d}W_u|\mathcal{F}_t\right]$$

$$= \mathbb{E}[V_{\tau^*}] + \int_0^t H(S_u) \, \mathrm{d}W_u$$

$$= V_{\tau^*} - \int_t^{\tau^*} H(S_u) \, \mathrm{d}W_u. \tag{12}$$

We can have

$$\mathbb{E}\left[\mathbb{E}[V_{\tau^*}|\mathcal{F}_t]\cdot\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\bigg|\mathcal{F}_t\right]=\mathbb{E}[V_{\tau^*}|\mathcal{F}_t]\,\mathbb{E}\left[\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\bigg|\mathcal{F}_t\right]=0,$$

and

$$\mathbb{E}[\mathbb{E}[V_{\tau^*}|\mathcal{F}_t]]\,\mathbb{E}\left[\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\right]=0,$$

implying $\mathbb{E}[V_{\tau^*}|\mathcal{F}_t]$ and $\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u$ are uncorrelated given $\mathcal{F}_t$, so

$$\mathrm{Var}\left(V_{\tau^*}\right)=\mathrm{Var}\left(\mathbb{E}[V_{\tau^*}|\mathcal{F}_t]\right)+\mathrm{Var}\left(\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\right). \tag{13}$$

Combine (12) and (13), we have

$$\mathrm{Var}\left(V_{\tau^*}-\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\right)=\mathrm{Var}\left(V_{\tau^*}\right)-\mathrm{Var}\left(\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u\right)\leqslant\mathrm{Var}\left(V_{\tau^*}\right). \tag{14}$$

Hence, by subtracting the term $\int_t^{\tau^*}H(S_u)\ \mathrm{d}W_u$ from V_{τ^*}, the variance is reduced.

$\square$

Therefore, adding the control variate $\int_{t_i}^{\tau_i}H_s dW_s$ in the derivation of Y_{t_i} reduces the variance.

Note that we also show that $\int_{\tau_i}^T H_s dW_s=0$. This is in line with the hedging theory as we stop hedging once the stopping time is reached (the option is exercised).

B   Warm-start training with the network trained one step before

In the original version of method I, no technique for parameter initialisation is employed, resulting in the random initialisation of weights and biases at the onset of the training. While this randomness typically does not pose a problem in practice, there is a possibility for training stagnation from the start due to subpar parameter choices, and it can lead to time-consuming processes. To enhance efficiency, we adopt a strategy where the parameters of a previously trained network serve as the initial values for the model under the current training. The rationale behind this technique is the observed similarities in the shapes of both continuation functions and martingale increment functions at different times, suggesting that parameters across different networks should exhibit similarities. Table 8 and Figure 8 illustrate the impact of random and non-random initialisation on results.

The table highlights that with this variant more accurate results are achievable in one-third of the time required for the base scheme. The enhancement in accuracy can be

attributed to a better initial guess, facilitating more effective training in the right direction. The figure affirms the effectiveness of this modification. At the step before maturity, both versions commence with random initialisation, resulting in similar numbers of epochs. However, this number significantly decreases for all other steps. In most steps, less than half the number of epochs is needed. This effect is particularly pronounced at the initial time where the same S_0 is used for all paths. Although ΔW_{t_0} values differ, the training becomes highly versatile. In the present example, the number of epochs is 10 times greater without the variant. This ratio can vary due to the randomness, with observed instances ranging from a worst-case scenario of 80 times more to a best-case scenario of 3 times more in experiments.

		Lower Bound		Upper Bound		Difference	
	Time(sec)	Mean	S.D.	Mean	S.D	Mean	S.D
Random start	360	4.4738	0.0007	4.4889	0.0005	0.0151	0.0010
Warm start	135	4.4769	0.0002	4.4877	0.0004	0.0108	0.0005

Table 8: Pricing 1D vanilla American put option (with the same parameters as the one in Section 4.1.1). The first row displays results where weights are randomly initialised at each time. The second row shows the estimate when warm-start is applied.
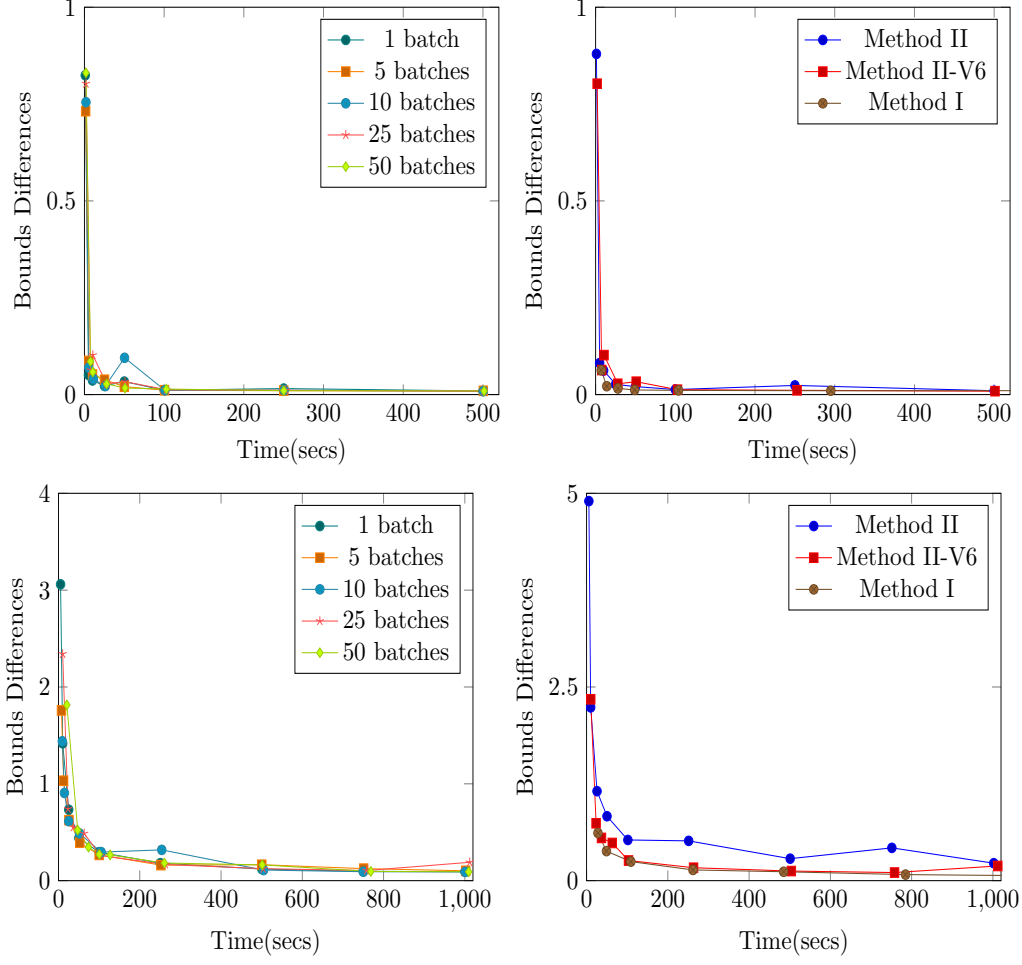
Figure 4: Left: Changes in results from different numbers of batches used among updates of the stopping strategy when we generate fresh data for training. Right: Changes of the results using different methods: blue line: original method II; red line: method II with variation 5 and 25 batches were used among updates; brown line: method I with variation 1. The top and bottom two plots correspond to the 1D put option and the 5D max-call option, respectively.
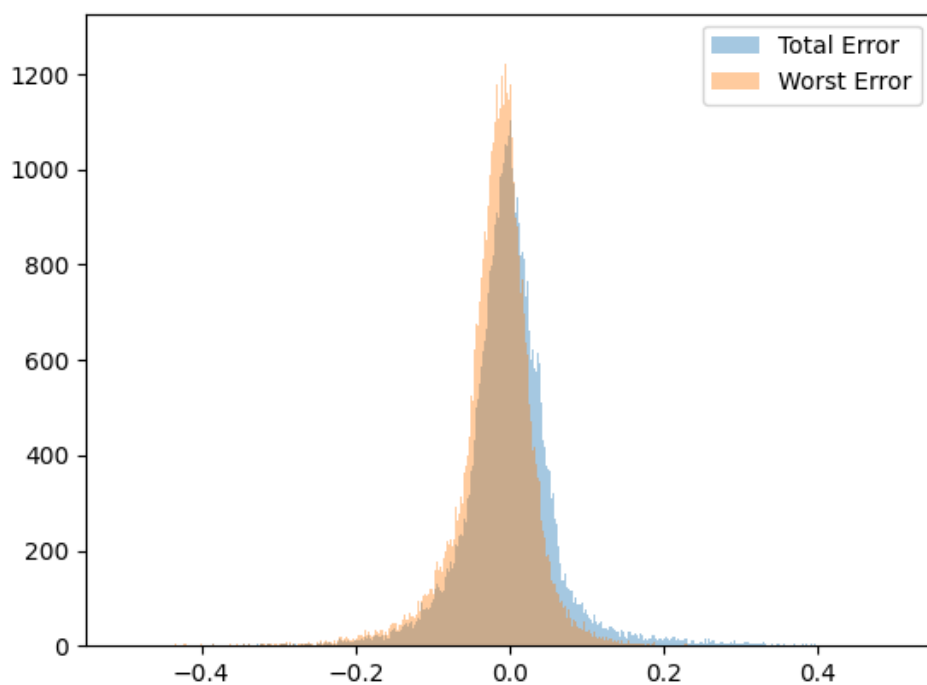
Figure 5: Hedging errors for the 1D American put option along 10,000 paths by directly using the model we trained via the base method II with variation 3
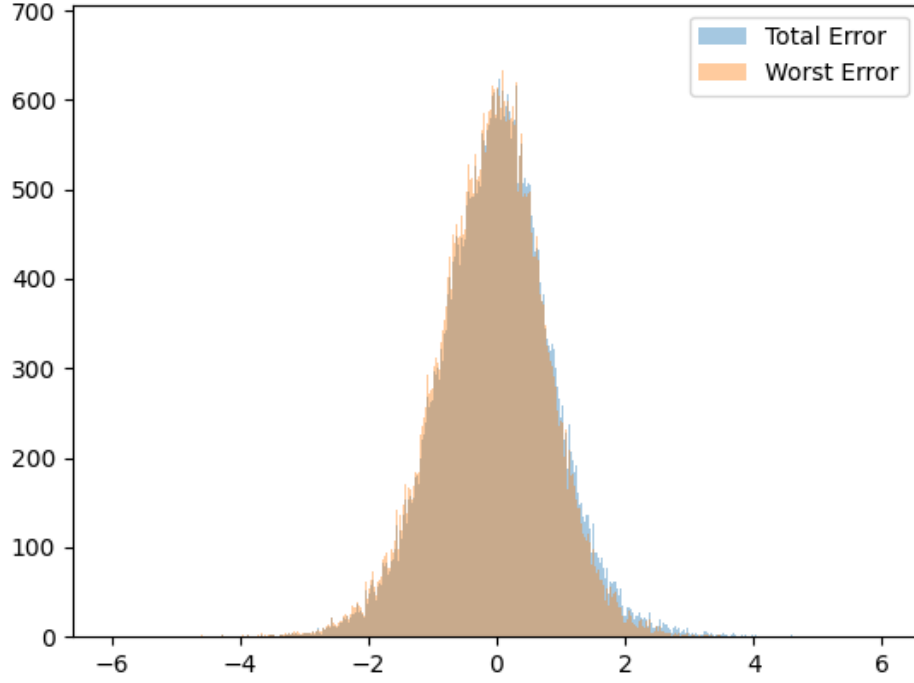
Figure 6: Hedging errors of 5D Bermudan max-call option generated by method II with variation 5 and 32 sub-steps
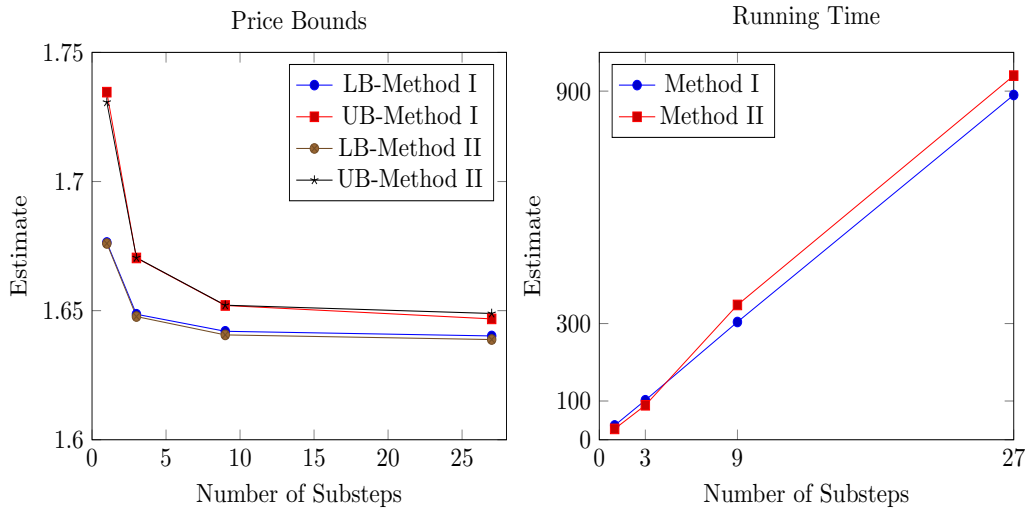

Figure 7: Price bounds (*Left*) and corresponding running times (*Right*) of the Heston put option with different numbers of substeps using both method I and II (variant 5).
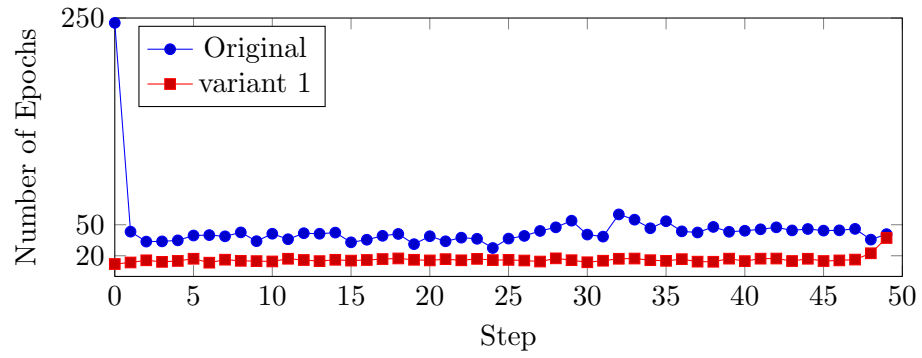
Figure 8: The number of epochs needed till the training stagnates at different steps with/without variant 1 when pricing a 1D vanilla American put option