# Simulation of BSDEs and Application to nonlinear PDEs for Shares Execution Programs

Yin Fu

(Mentor: Mehdi Sonthonnax)

**Abstract**

In this paper, we researched and leveraged the classical backward stochastic differential equations (BSDEs) for solving nonlinear problems in quantitative finance. By representing the nonlinear PDEs in the BSDE framework using a generalized version of the Feynman-Kac theorem, it can be solved by Monte Carlo simulations together with modern machine learning techniques. Specifically, we considered the Least Square Monte Carlo method [1] and the Deep BSDE method [2], and tested the numerical performance for various financial applications such as option pricing and stochastic optimal execution for shares liquidation. Moreover, for each problem above, we compared the numerical results to benchmark methods used in the industry to demonstrate the effectiveness of our approach.

## 1 Introduction

In quantitative finance, nonlinear PDEs naturally arise in the context of option pricing and stochastic optimal control problems in high dimensions. Traditional numerical methods for PDEs, such as the finite difference method, suffers from the curse of dimensionality as the number of points in the grid of spatial and temporal domains increases exponentially with the number of dimensions. Just like forward SDEs can be used to solve linear parabolic PDEs by Monte Carlo simulations, BSDEs have a strong connection to nonlinear PDEs through a generalization of the Feynman-Kac theorem.

Different from a forward SDE, a BSDE requires a random terminal condition to be satisfied. If the terminal condition is determined by a forward SDE, it is called a forward-backward stochastic differential equation (FBSDE). In that case, the dynamics of the BSDE can also be affected by the forward SDE. Moreover, the generalized Feynman-Kac formula shows that under suitable conditions, backward dynamics can be expressed as a deterministic function of forward dynamics, where the function is the solution to a parabolic PDE with a final condition analogous to the random terminal condition of the FBSDE. For all analysis in this paper, a BSDE is implicitly referring to the backward dynamics of an FBSDE driven by some forward dynamics, and it represents the price process in option pricing or the agent's value function in optimal execution. In this case, by simulating the forwards underlying forward SDEs using an Euler discretization, we can solve the corresponding BSDEs leveraging machine learning algorithms such that the curse of dimensionality issue is partially avoided.

We started by testing the algorithms extensively for linear and nonlinear option pricing problems. In particular, we analyzed the numerical stability and convergence of the numerical solutions and compared them to the analytical solutions. In the end, we solved nonlinear PDEs (HJB-PDEs) in stochastic optimal control problems arising from the real-world shares execution models. Examples of nonlinear PDEs will be drawn from Chapter 6 in [3]. It is essential to point out that various regularization methods is required in order to represent the nonlinear PDEs in a BSDE framework. The detailed analysis will be presented in section 3.

The algorithms we used for solving the BSDEs are the Least Square Monte Carlo method [2] and the Deep BSDE method [1], which will be discussed in section 2.

# 2 Algorithms

Given the probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{0\leq t\leq T}, \mathbb{P})$, where $\{\mathcal{F}_t\}_{0\leq t\leq T}$ is the $d$-dimensional Brownian Motion filtration, we consider the uncoupled FBSDEs in the following form

$$\begin{cases} dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \quad X_0 = x_0 \\ dY_t = -f(t, X_t, Y_t, Z_t)dt + Z_t dW_t, \quad Y_T = g(X_T). \end{cases} \tag{1}$$

Here we assume $\mu, \sigma, f, g$ all satisfy the regularity condition and $X_t$ and $Y_t$ have dimension $d$ and 1, respectively. The generalized Feynman-Kac theorem states that if $u \in C^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R}^d)$ satisfies

$$\partial_t u + \mathcal{L}u + f(t, x, u(t,x), \partial_x u(t,x)\sigma(t,x)) = 0, \quad u(T,x) = g(x) \tag{2}$$

Then the solution to (1) is the pair of $\mathcal{F}_t$-adapted processes

$$\begin{cases} Y_t := u(t, X_t) \\ Z_t := \partial_x u(t, X_t)\sigma(t, X_t). \end{cases} \tag{3}$$

Now, one is able to solve PDE (2) by solving FBSDE (1) instead. We will consider the following discretization of FBSDE (1) for the algorithms we introduced in the subsection 2.1 and 2.2.

$$\begin{cases} P = \{t_0, t_1, ..., t_N\}, \quad t_0 = 0,\ t_N = T \\ \Delta t := \frac{T}{N} \\ t_n := n\Delta t = n\frac{T}{N} \\ \Delta W_n := W_{t_n} - W_{t_{n-1}} \end{cases} \tag{4}$$

- $P$ is the partition of time interval $[0, T]$
- $N$ is the total number of time steps between $[0, T]$
- $\Delta t$ is the step size in time
- $t_n$ is the discretized time step

- $\Delta W_n$ is the d-dimensional Brownian motion increments between $t_n$ and $t_{n-1}$

In addition, we denote $\{X_n^P\}_{n\in\{0,...,N\}}$ the discretization of $\{X_t\}_t$ with respect to $P$, which is a discrete-time stochastic process

$$X_n^P := X_{t_n}^P,$$

and similarly for $\{Y_n^P\}_{n\in\{0,...,N\}}$ and $\{Z_n^P\}_{n\in\{0,...,N\}}$, we have

$$Y_n^P := Y_{t_n}^P, \quad Z_n^P := Z_{t_n}^P.$$

For simplicity, we will drop the superscript $P$ for the rest of our analysis.

## 2.1 Least Square Monte Carlo (LSMC)

The least square Monte Carlo algorithm proposed in Chapter 5.4 of [2] started with a discretization of $X_t$ and was followed by a backward iteration for solving $(Y_t, Z_t)$ at each time step $t_n$ using ordinary least square regression.

- Simulate $\{X_n\}_{n=1}^N$ using Euler scheme with a choice of partition $P$ for a total of $M$ paths.
- For $n = N$, compute $Y_N = g(X_N)$.
- For $n = N-1, ..., 1$ find $(\alpha^{(n)}, \beta^{(n)})$ by solving a pair of optimization problems

$$\beta^{(n)} = \underset{\beta}{\operatorname{argmin}}\, \mathbb{E}\left\|\psi_n - \sum_{i=1}^{k} \beta_i \phi_i(X_n)\right\|_2^2 \tag{5}$$

$$\alpha^{(n)} = \underset{alpha}{\arg\min}\, \mathbb{E}\left\|\xi_n - \sum_{i=1}^{k} \alpha_i \phi_i(X_n)\right\|_F^2, \tag{6}$$

  where $\psi_n := Y_{n+1} + f(t_n, X_n, Y_{n+1}, Z_n)\Delta t, \quad \xi_n := \frac{1}{\Delta t} Y_{n+1}\Delta W_{n+1}$. Then we compute

$$Y_n = \sum_{i=1}^{k} \beta_i^{(n)} \phi_i(X_n) \tag{7}$$

$$Z_n = \sum_{i=1}^{k} \alpha_i^{(n)} \phi_i(X_n). \tag{8}$$

- For $n = 0$, compute

$$Y_0 = \mathbb{E}\left[Y_1 + f(0, X_0, Y_1, Z_0)\Delta t\right]$$
$$Z_0 = \frac{1}{\Delta t}\mathbb{E}\left[Y_1 \Delta W_1\right].$$

Here, $\{\phi_i\}_{i=1}^k \subseteq C^1(\mathbb{R}^d, \mathbb{R})$ is the set of basis functions that is chosen for mapping $X_n$ into higher dimensions. One can identify the optimization (5) and (6) as the classical feature map regression problem. In that case, $(\alpha^{(n)}, \beta^{(n)})$ can be solved analytically, which results in fast computation and

make the BSDEs solvable for a choice of relatively large $N$. In this paper, we consider $\{\phi_i\}_{i=1}^k$ to be polynomial basis functions with degrees up to 3, i.e., $\{1, x, x^2, x^3\}$. In addition, the expectation operator from above will be approximated by the sample average over $M$ paths, which is similar to traditional machine learning problems.

### 2.2 The Deep BSDE Method

The Deep BSDE method presented in the founding paper [1] start with the discretized forward approximation (which is different from the backward approximation in the LSMC algorithm) for $Y_n$:

$$Y_{n+1} = Y_n - f(t_n, X_n, Y_n, Z_n)\Delta t + Z_n \Delta W_{n+1} \tag{9}$$

$$:= F(t_n, X_n, Y_n, Z_n, \Delta t, \Delta W_{n+1}). \tag{10}$$

The algorithm uses a sequence of neutral network approximations $(\mathcal{U}_n)_{n=0}^N$ for $(Y_n)_{n=0}^N$, where $\mathcal{U}_n$ is defined by

$$\begin{cases} \mathcal{U}_{n+1} = F(t_n, X_n, \mathcal{U}_n, \mathcal{Z}_n, \Delta t, \Delta W_{n+1}) \\ \mathcal{Z}_n = \mathcal{Z}(X_n \mid \theta_n). \end{cases} \tag{11}$$

At each time step, $Z_n$ is approximated by $\mathcal{Z}_n(\theta_n)$, a feedforward neural network with parameter $\theta_n$, and $Y_n$ is calculated using the forward scheme in (10). By choosing an initial guess $\mathcal{U}_0$, $\mathcal{U}_N$ forms a global neural network that depends on the set of parameters for each subnet $\mathcal{Z}_n$, $\{\theta_n\}_{n=0}^{N-1}$. The global neural network aims to minimize the mean square loss of the matching error at the final time $T$:

$$\theta^* = \operatorname*{argmin}_{\theta} \mathbb{E}\left[|g(X_N) - \mathcal{U}_N(\theta)|^2\right] \tag{12}$$

$$\theta := (\mathcal{U}_0, \theta_0, \theta_1, ..., \theta_{N-1}).$$

The optimization problem (12) is solved using stochastic gradient descent algorithms, such as Adam.

The author [4] pointed out that there are skip connections between different subnetworks (similar to residual networks) in equation (9), which is designed for numerical stability consideration. In addition, as discussed in [5], one can only approximate $Y_0 = u(0, X_0)$ in light of the algorithm above. Numerical techniques are required to approximate the entire solution path $Y_n = u(t_n, X_n)$.

## 3 Numerical Results

In the first two subsections, we leverage the LSMC method in section 2.1 to solve the option pricing problem in quantitative finance. In particular, we test the numerical performance of the algorithm and compare it to the benchmark methods which are widely used in the industry. In the third part, the Deep BSDE method was adopted to solve the HJB-PDE corresponding to the liquidation with only temporary impact model in Chapter 6 of [3].

## 3.1 European Option Pricing

The well-known Black-Scholes PDE has the following form,

$$\partial_t u + rS\partial_s u + \frac{1}{2}\sigma^2 S^2 \partial_{ss} u - ru = 0, \quad u(T, s) = g(s). \tag{13}$$

In this case, the FBSDE representation is given by

$$\begin{cases} dS_t = rS_t dt + \sigma S_t dW_t, & S_0 = s_0 \\ dY_t = -rY_t dt + Z_t dW_t, & Y_T = g(S_T). \end{cases} \tag{14}$$

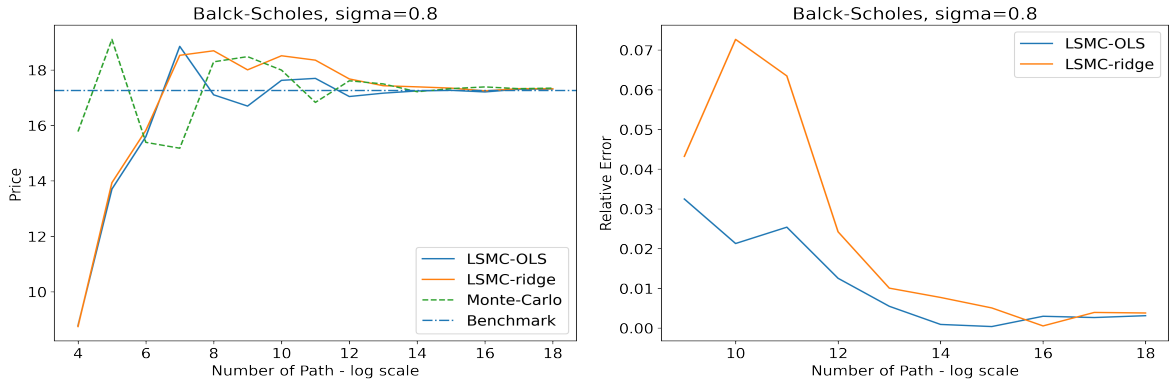
As discussed in section 2.1, polynomial basis functions will be used in the regression step (5) and (6). Although other basis functions (e.g., Fourier basis) and regression algorithms (e.g., support vector regression, neural networks) are tested, only OLS and ridge regression with polynomial feature map exhibits a stable convergence to the analytical solution with reasonably fast computation, typically less than 5 seconds for a single spot price with $M = 2^{16}$ on a MACBOOK PRO with a 2.80 Gigahertz (GHz) Intel Core i7.

If not specified, we will choose the following parameters for pricing.

$$S_0 = 40, \quad r = 0.06, \quad \sigma = 0.5, \quad K = 30, \quad T = 1, \quad N = 252.$$

### 3.1.1 Convergence

For convergence analysis, we choose $\sigma = 0.8$ to price a European vanilla option. In practice, mispricing usually happens when implied volatility is high, especially for Monte Carlo-based pricing method.




(a) (b)

Figure 1: Convergence of LSMC algorithm for European option pricing with high implied volatility. (a) Convergence for different pricing method (b) Relative error for LSMC pricing

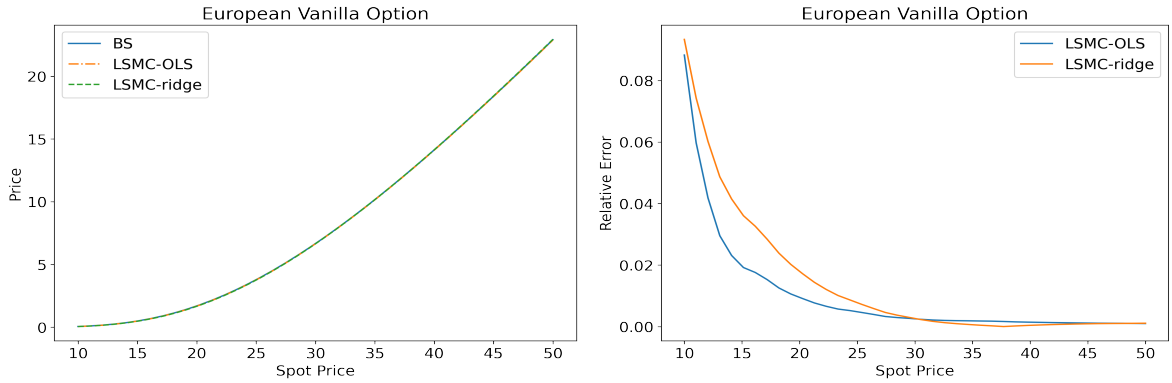
.

In Figure 1a we plot the option price for the LSMC algorithm, and compare it to the usual Monte Carlo method for a different number of paths, $M$, in the simulation. Here, the benchmark option price is given by the Black-Scholes formula. Moreover, Figure 1b shows the relative error converges to 0 at around $M = 2^{16}$. In this case, we will choose this value for the rest of the option pricing problem.

### 3.1.2 Call Option

For vanilla call option, the final condition is

$$g(s) = (s - K)^+.$$




(a) (b)

Figure 2: European vanilla call option (a) European vanilla call option price with spot price from 10 to 50 (b) Relative error

Figure 2 indicates the relative pricing error is larger when the option is deep out of the money. This numerical behavior was expected since the analytical option price will be close to 0, which results in a larger relative error. Overall, we regard the LSMC algorithm as stable in this case.

### 3.1.3 Barrier Option

For barrier option, the final condition of the PDE (13) becomes

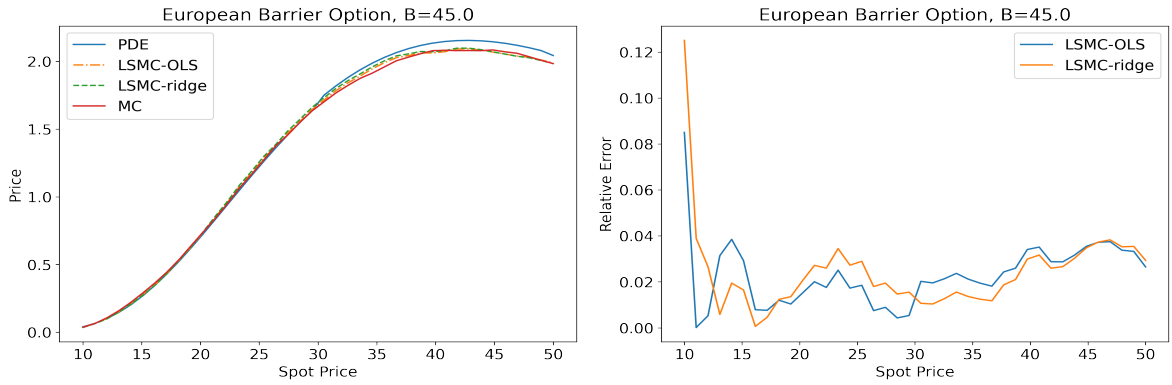
$$g(s) = (s - K)^+ \mathbb{I}_{s \leq B} \tag{15}$$

where $B$ is upper barrier of the option. In this case, we choose $B = 45$.

For barrier options, the benchmark method for pricing is the classical PDE method using implicit scheme. In this case, one can see from Figure 3b the relative error for the LSMC algorithm is larger compared to vanilla call option pricing. Moreover, the pricing error becomes significant when the spot price approaches the upper barrier $B$. The deviation from the analytical price is due to the non-monotonicity of the payoff, and it appears to be the numerical limitation for Monte Carlo-based method in general since the standard Monte-Carlo method also price the barrier call option in a similar way, as shown in Figure 3a.

### 3.1.4 Spot-Dependent Volatility

In this section, we consider spot-dependent volatility and test the numerical performance of the LSMC algorithm for European vanilla call option pricing. In particular, we will use CEV and SVI models.




(a) (b)

Figure 3: European barrier call option (a) European barrier call option price with spot price from 10 to 50 (b) Relative error

The CEV model replaced the constant volatility $\sigma$ in FBSDE (14) by

$$\sigma(S_t) = \sigma S_t^{\beta-1}.$$

Then forward dynamics has the form

$$dS_t = rS_t dt + \sigma S_t^{\beta} dW_t.$$

For SVI model, we consider the new forward dynamics in FBSDE (14) as the following

$$dS_t = r_t S_t dt + \sigma_s(S_t) S_t dW_t,$$

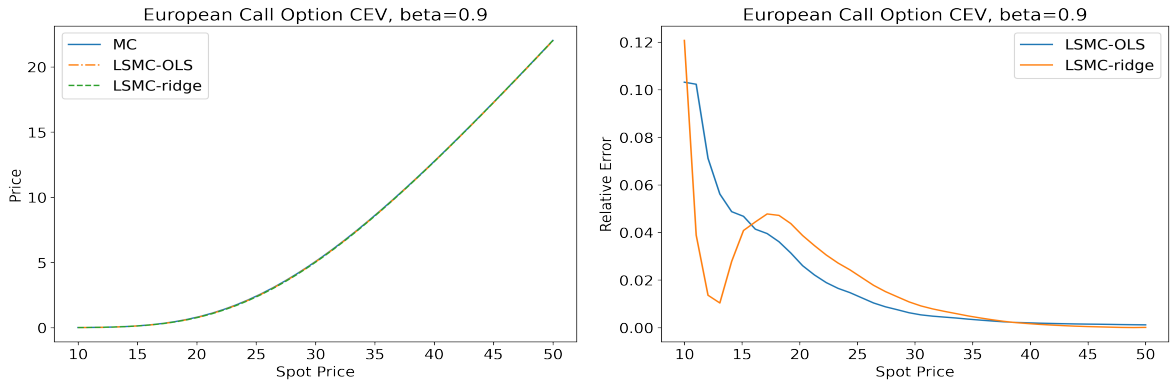
where

$$\sigma_s(S) = a + b\left[\rho(k-m) + \sqrt{(k-m)^2+\sigma^2}\right], \quad k = \ln\left(\frac{S}{F_t}\right), \quad F_t = Ke^{r(T-t)}$$

$$a = 0.4, \quad b = 0.04, \quad \rho = 0.1, \quad m = 0.01, \quad \sigma = 30$$

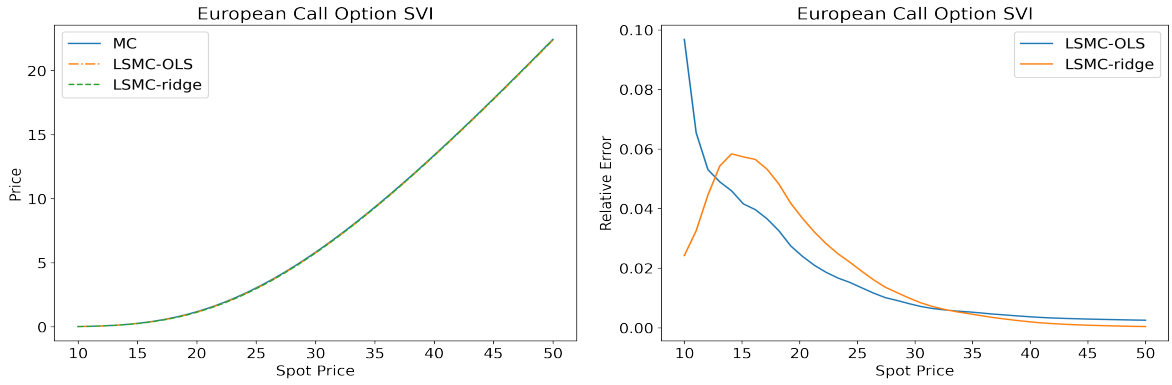
In this case, SVI parametrization is adopted to capture the volatility smile in the market. Although the typical usage of the SVI model in the quantitative finance industry is different, one should be convinced that it is sufficient for numerical testing purposes.

In this case, the benchmark method was chosen to be the standard Monte Carlo pricing, Figure 4 and 5 exhibit a similar relative error behavior of LSMC performance on both the CEV and SVI model. In this case, the relative error can fluctuate, especially for deep out-of-the-money options, since the benchmark Monte Carlo pricing is not analytical. Overall, the LSMC algorithm is stable in a spot-dependent volatility setting.




(a) (b)

Figure 4: European vanilla call option in CEV model (a) European call option price (CEV) with spot price from 10 to 50 (b) Relative error




(a) (b)

Figure 5: European vanilla call option in SVI model (a) European call option price (SVI) with spot price from 10 to 50 (b) Relative error

## 3.2 American Option Pricing

The pricing PDE [6] for American option is given by

$$\partial_t u + \mathcal{L}u + (-ru + (\mathcal{L}g - rg)^{-}\mathbb{I}_{\{u(t,s)\leq g(s)\}}) = 0, \quad u(T,s) = g(s) \tag{16}$$

which has the corresponding FBSDE representation:

$$\begin{cases} dS_t = rS_t dt + \sigma S_t dW_t, \quad S_0 = s_0 \\ dY_t = -rY_t + (\mathcal{L}g(S_t) - rg(S_t))^{-}\mathbb{I}_{\{Y_t \leq g(S_t)\}} dt + Z_t dW_t, \quad Y_T = g(S_T). \end{cases} \tag{17}$$

Different from the usual Black-Scholes PDE (13), PDE (16) is derived from the well-known variational inequality and hence nonlinear. In fact, it is semilinear and can be considered as a harder problem compared to European option pricing in the last section. In this case, by taking advantage of nonlinearity in the drift term of the BSDE in (17), PDE (16) can be solved in a BSDE framework. In

this section, we leverage the LSMC algorithm to price an American put option

$$g(s) = (K - s)^+. \tag{18}$$

with the following parameters

$$r = 0.05, \quad \sigma = 0.5, \quad K = 50, \quad T = 1, \quad N = 252.$$

### 3.2.1 Regularization

In PDE (16), $g''(s)$ is required to exist due to the presence of Ito generator $\mathcal{L}$. However, the put option payoff (18) is not differentiable at $s = K$. In order to improve the numerical stability, one has to regularize or smooth the payoff using numerical interpolation around $s = K$. In this case, the put option payoff (18) is replaced by the following polynomial interpolation scheme in the LSMC algorithm,

$$g(s) \sim g_\epsilon(s) = \begin{cases} K - s & \text{for } s < K - \epsilon \\ \frac{-(s-(K+\epsilon))^2}{-4\epsilon} & \text{for } K - \epsilon \leq s \leq K + \epsilon \\ 0 & \text{for } s \geq K + \epsilon \end{cases} \tag{19}$$

where $\epsilon$ is the hyperparameter that controls the width of the interpolation. The approximation scheme (19) guarantees is differentiability everywhere and the existence of $g''_\epsilon(s)$ at $s = K$, although the second order differentiability is not satisfied at $K - \epsilon$ and $K + \epsilon$. We consider $\epsilon = 0.001$ in this paper, while the payoff approximation with $\epsilon = 0.1$ is shown below for better visualization.

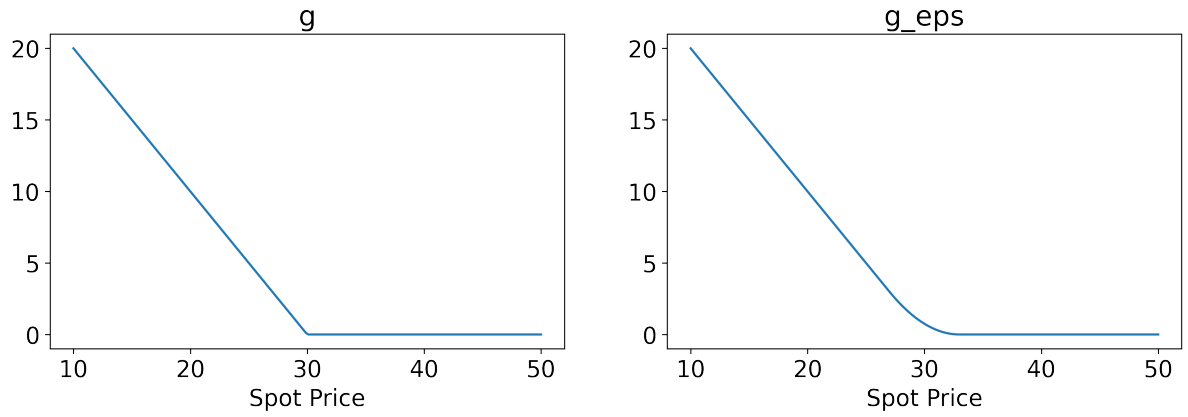
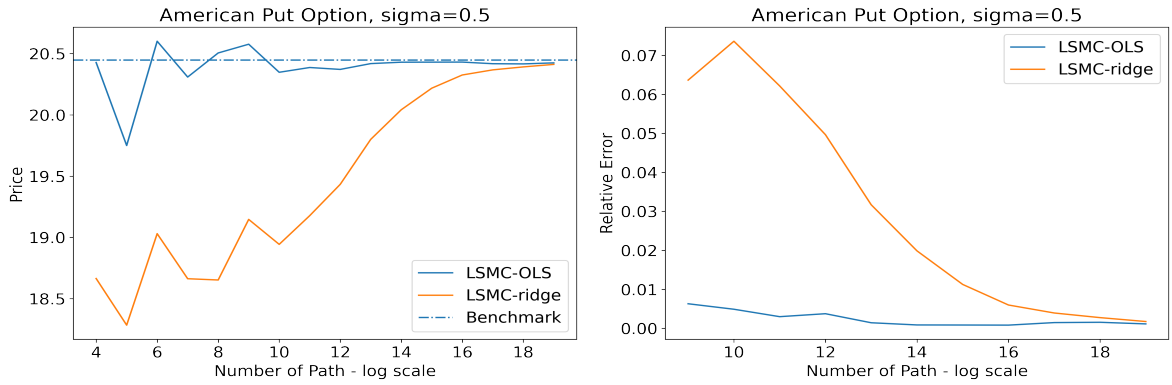



Figure 6: Put Option Payoff Approximation

### 3.2.2 Convergence

Similar to previous convergence analysis, we choose $\sigma = 0.8$. In this case, the benchmark price is obtained by solving the variational inequality using an implicit scheme. Moreover, as shown in Figure 7, with the same choice of parameters, LSMC with OLS outperformed ridge regression in terms of the rate of convergence, which implies the ridge penalty is not suitable in this case.

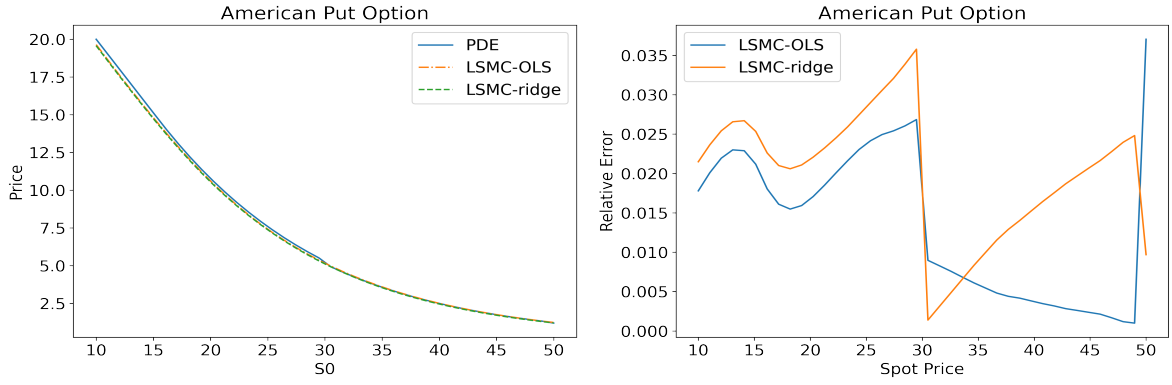



(a) (b)

Figure 7: Convergence of LSMC algorithm for American option pricing with high implied volatility (a) Convergence for different pricing method (b) Relative error for LSMC pricing

### 3.2.3 Put Option

Compared to the European option pricing in Figure 2, the LSMC algorithm shows more pricing error, in particular, noticeably underpriced deep-in-the-money options, as displayed in Figure 8. This shows the limitation of capturing the nonlinearity of backward dynamics of price process $Y_t$ using a linear combination of polynomials. Hence, it is expected that ridge regression is not well-performed since the ridge penalty makes the above polynomial approximation even harder, especially when the number of the training sample is small, i.e., $M$ is small. Numerical experiments showed that the pricing gap will be larger if the regularization in section 3.2.1 is not performed. One should notice the difference between the LSMC algorithm for BSDE and the Longstuff-Schwartz algorithm [7]. The former uses linear regression to approximate the option price directly, and the latter uses it for approximating the exercise boundary.




(a) (b)

Figure 8: American put option (a) American put option price with spot price from 10 to 50 (b) Relative error

## 3.3 Liquidation with Only Temporary Impact

In this section, we consider the stochastic optimal control problem of liquidating $q$ shares over a time interval $[0, T]$ subject to only temporary price impact. In this case, the agent aims to find the optimal trading speed at the current time $t = 0$. The model is formally stated in chapter 6.3 [3].

Consider the stock price process $S_t$ and inventory process $Q_t$ follows the following dynamics,

$$\begin{cases} dS_t &= \sigma_s dW_t, \quad S_0 = s \\ dQ_t &= -\nu_t dt, \quad Q_0 = q \end{cases} \tag{20}$$

with the execution price

$$\hat{S}_t = S_t - f(\nu_t) \tag{21}$$

$$f(\nu_t) = kv \tag{22}$$

where $\nu_t$ is the trading speed, and $f$ is the temporary price impact function with parameter $k$, which determines the magnitude of the impact in trading.

In this setting, we ignore the bid-ask spread and assume a linear temporary price impact (22). In addition, $S_t$ has no drift term in (20), indicating the absence of the permanent price impact ($S_t$ is not controlled by $\nu_t$). Suppose the agent has the following value function by maximizing the expected revenues over all adapted trading speed $\nu_t$,

$$\begin{aligned} H(t, s, q) &= \sup_{\nu_t \in \mathcal{A}(t,T)} E_t \left( \int_t^T \hat{S}_u \nu_u du \right) \\ &= \sup_{\nu_t \in \mathcal{A}(t,T)} E_t \left( \int_t^T (S_u - k\nu_u)\nu_u du \right). \end{aligned}$$

It can be shown [3] that $H(t, s, q)$ satisfies the PDE,

$$\partial_t H + \frac{1}{2}\sigma_s^2 \partial_{ss} H + \frac{1}{4k}(s - \partial_q H)^2 = 0 \tag{23}$$

with final condition

$$H(T, s, q) = -\infty, \quad q > 0 \tag{24}$$

$$H(T, s, q) = 0, \quad q = 0 \tag{25}$$

Therefore, the agent is forced to liquidate all the holdings due to (24). In this case, the optimal trading speed is obtained by

$$v^* = \frac{1}{2k}(s - \partial_q H) \tag{26}$$

### 3.3.1 Regularization

PDE (23) cannot be directly represented in an FBSDE framework due to the non-stochastic nature of the inventory process $Q_t$. Moreover, negative infinity in the final condition (24) is infeasible computationally. Here, we propose a regularization scheme that addresses both problems above.

Firstly, the final condition (24) is replaced by

$$H(T, s, q) = qs - \lambda q^2, \tag{27}$$

which can be considered as imposing a quadratic penalty on the book value at the terminal time $T$. One can solve the PDE (23) with (27) to obtain the analytical solution

$$H(t, s, q) = qs - q^2 \left( \frac{1}{\lambda} + \frac{1}{k}(T - t) \right)^{-1} \tag{28}$$

Moreover, to deal with the singularity of PDE (23), we regularized the process $Q_t$ by introducing a volatility term controlled by the hyperparameter $\epsilon$. Then the corresponding FBSDE for a regularized PDE (23) becomes

$$\begin{cases} dS_t &= \sigma_s dW_t^S, \quad S_0 = s \\ dQ_t &= \epsilon Q_t dW_t^Q, \quad Q_0 = q \\ dY_t &= -\frac{1}{4k}(a \cdot X_t + A \cdot Z_t)^2 dt + Z_t dW_t, \quad g(X_T) = qs - \lambda q^2 \end{cases} \tag{29}$$

where

$$X_t = (S_t, Q_t), \quad a = (1, 0)^T, \quad A = \left( 0, -\frac{1}{\epsilon q} \right)^T$$

Hence, the regularized PDE becomes

$$\partial_t H + \frac{1}{2}\sigma_s^2 \partial_{ss} H + \frac{1}{2}\epsilon^2 q^2 \partial_{qq} H + \frac{1}{4k}(s - \partial_q H)^2 = 0 \tag{30}$$

with final condition (27). Specifically, one who wants to solve PDE (30) instead of (23) should consider $\epsilon \to 0$.

### 3.3.2 Convergence

In this section, we use the Deep BSDE method [1] to solve the regularized PDE (30) proposed in the previous section. In particular, the Deep BSDE method has the advantage of capturing complex non-linearity structures of the solution compared to the LSMC method.

One should point out that this liquidation problem is nonlinear, with an FBSDE of dimension two since the forward dynamics consists of both $S_t$ and $Q_t$. The following parameters are chosen for testing:

$$T = 0.25, \quad \lambda = 1, \quad \epsilon = 0.01, \quad k = 0.01, \quad \sigma_s = 0.5, \quad q = 100, \quad N = 50.$$

In the Deep BSDE model, each subnet, $\mathcal{Z}_n(\theta_n)$, consists of two hidden layers with 22 neurons in each layer. We use the Adam optimizer for model training with a piece-wise constant decay learning rate of 0.04, 0.03, and 0.02 after 500 and 1000 iterations. Moreover, we use a batch size of 64 for each iteration and a validation set of 32 for every 200 iterations.

For convergence analysis, we choose the spot price to be $s = 35$. After several rounds of testing, the convergence of the Deep BSDE problem for this problem appears to be non-stable. In particular,

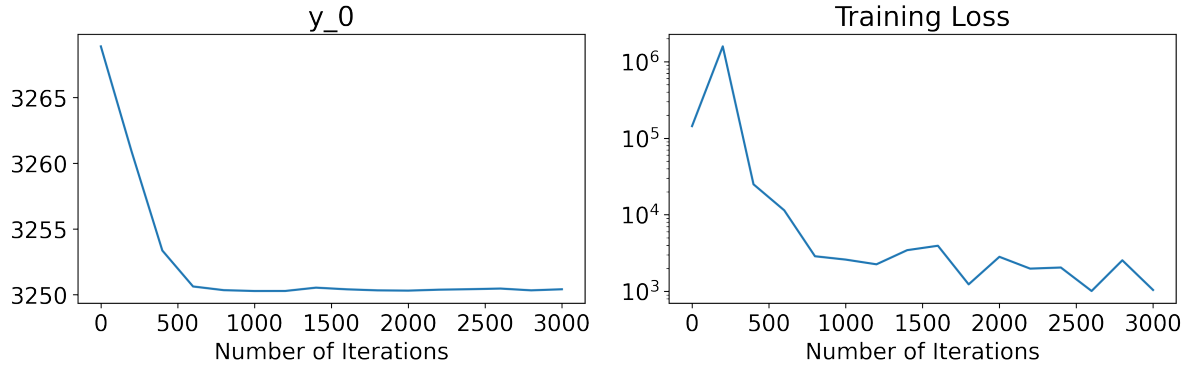



Figure 9: Convergence of Deep BSDE solution for $y_0 = H(0, s, q)$

we have to choose a reasonably close initial guess to make the algorithm converge, especially when the hyperparameter $\epsilon$ is very close to 0 (which is a desired property in the regularization scheme (29)). However, we choose a relatively large $\epsilon$ for numerical stability consideration in this case. The drawback of this compensation will be discussed in the next section.

### 3.3.3 Solution

With the same choice of parameters and training procedure, we solve $H(0, s, q)$ for different spot prices from 36 to 46. The Deep BSDE solution is compared with the analytical solution (28) as shown in Figure 10.

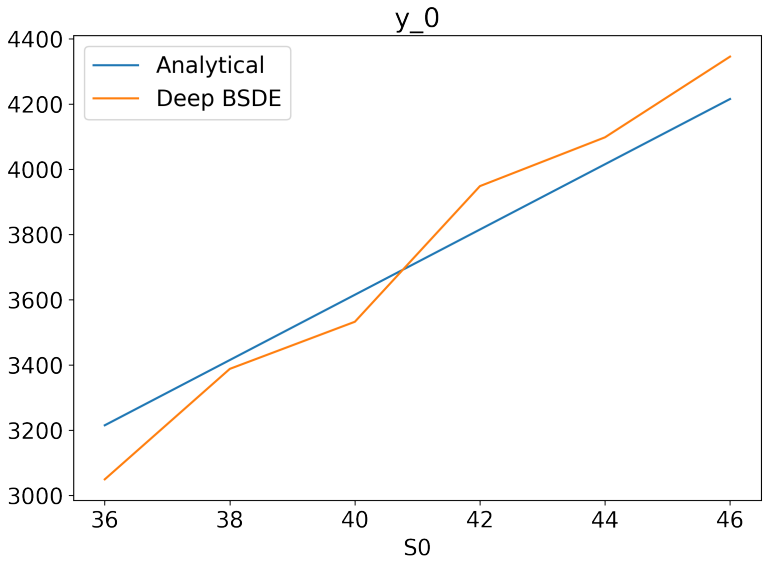



Figure 10: $y_0 = H(0, s, q)$ for different spot price.

The deviation of the Deep BSDE solution is noticeably significant due to the choice of $\epsilon$. In this case, the Deep BSDE solver approximates the solution of PDE (30), where the coefficient $\epsilon^2 q^2 = 1$ is significantly larger than 0. Hence, the deviation in Figure 10 is because two solutions are obtained from

very different PDEs. Thus, the regularization scheme needs to be improved, and we will reserve it for future research.

# 4 Conclusion

In this paper, we explored the possibility of using BSDE to solve nonlinear problems in quantitative finance. In the end, we aim to solve the stochastic optimal execution problem by first testing the algorithm with simpler numerical examples. We begin with solving a linear European option pricing problem where the analytical solution is well-known, and the dynamics of the system is easy to implement. We used the LSMC algorithm to solve the BSDE and gradually extended our analysis into more complicated dynamics (spot-dependent volatility) and payoff types (barrier options). Then, we solved a one-dimensional American option pricing problem to test the numerical performance of how the LSMC algorithm captures the nonlinear dynamics. Finally, we proposed a regularization scheme for the optimal control problem on shares liquidation, which is considered a two-dimensional nonlinear problem, and solved the proposed regularized PDE leveraging the Deep BSDE method. We pointed out that the proposed regularization scheme needs to be improved, observing the convergence of the Deep BSDE solution.

Besides the numerical results above, we conclude that regularization is critical to converting a nonlinear PDE to a BSDE for a broader range of applications, as the generalized Feynman-Kac theorem does not always satisfied for every nonlinear PDE. Depending on the problem, a well-developed regularization scheme can significantly improve the numerical performance of the BSDE solvers, e.g., regularization for American option pricing in section 3.2.1.

For future research, several directions can be taken. Firstly, the regularization scheme presented in section 3.3 can be improved to address the numerical stability issues. In addition, one can develop new BSDE solvers capable of solving the PDE on the entire sample path rather than only at its initial point $X_0$. We can also move to different shares execution models, not limited to liquidation, in a higher dimension, together with other user-defined nonlinear terminal penalties.

# References


[1] W. E, J. Han, and A. Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations," 2017.

[2] J. Zhang, *Backward Stochastic Differential Equations*. Springer, 2017.

[3] Á. Cartea, J. Penalva, and S. Jaimungal, *Algorithmic and High-Frequency Trading*. Cambridge University Press, 2015.

[4] W. E, "Deep learning-based algorithms for high-dimensional pdes and control problems," 2019.

[5] M. Raissi, "Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations," 2018.

[6] J. Guyon and P. Henry-Labordere, *Nonlinear Option Pricing*. Chapman and Hall/CRC, 2013.

[7] F. Longstaff and E. Schwartz, "Valuing american options by simulation: A simple least-squares approach," 2001.