Deep Learning-Based Numerical Methods for High-Dimensional Parabolic PDEs and Forward-Backward SDEs

Jiequn Han

Department of Mathematics, Princeton University
Joint work with Weinan E, Arnulf Jentzen, and Jihao Long

Financial/Actuarial Mathematics Seminar, University of Michigan, Mar 20, 2019
Table of Contents

1. Background

2. BSDE Formulation of Parabolic PDE

3. Neural Network Approximation

4. Stochastic Control in Discrete Time

5. Numerical Examples of High-Dimensional PDEs

6. Convergence of the Deep BSDE Method

7. Summary
Well-known Examples of PDEs

- The Black-Scholes equation for pricing financial derivatives,
  \[ v_t + \frac{1}{2} \text{Tr} \left( \sigma \sigma^T \text{Hess}_x v \right) + r \nabla v \cdot x - rv = 0. \]

- The Hamilton-Jacobi-Bellman equation in stochastic control (dynamic programming),
  \[ v_t + \max_m \left\{ \frac{1}{2} \text{Tr} \left( \sigma \sigma^T \text{Hess}_x v \right) + \nabla v \cdot b + f \right\} = 0. \]

- The Schrödinger equation in quantum many-body problem,
  \[ i\hbar \frac{\partial}{\partial t} \Psi(t, x) = \left( -\frac{1}{2} \Delta + V \right) \Psi(t, x). \]
Curse of Dimensionality

- The dimension can be easily large in practice.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Dimension (roughly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Scholes</td>
<td># of underlying financial assets</td>
</tr>
<tr>
<td>HJB equation</td>
<td>the same as the state space</td>
</tr>
<tr>
<td>Schrödinger equation</td>
<td># of electrons × 3</td>
</tr>
</tbody>
</table>

- A key computational challenge is the curse of dimensionality: the complexity is exponential in dimension $d$ for finite difference/element method – usually unavailable for $d \geq 4$.

- There is a huge gap between PDE modelings and computational algorithms.
Related Work in High-dimensional Case

- Linear parabolic PDEs: Monte Carlo methods based on the Feynman-Kac formula

- Semilinear parabolic PDEs:
  2. multilevel Picard approximation (E et al. 2016)

Remarkable Success of Deep Learning

- Machine learning/data analysis also face the same curse of dimensionality

- In recent years, deep learning has achieved remarkable success
Deep Learning 101

- Representation: in a compositional form rather than additive,

\[ f(x) = \mathcal{L}^{\text{out}} \circ \mathcal{L}^{N_h} \circ \mathcal{L}^{N_h-1} \circ \cdots \circ \mathcal{L}^{1}(x), \]

\[ h_p = \mathcal{L}^p(h_{p-1}) = \sigma(W_p h_{p-1} + b_p), \]

\( \sigma \): element-wise nonlinear activation function: \( \max(0, x) \), hyperbolic tangent, sigmoid, etc.

- Optimization:

\[ \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta)). \]

Algorithm: stochastic gradient descent (SGD) and its variants.
Table of Contents

1. Background

2. BSDE Formulation of Parabolic PDE

3. Neural Network Approximation

4. Stochastic Control in Discrete Time

5. Numerical Examples of High-Dimensional PDEs

6. Convergence of the Deep BSDE Method

7. Summary
Semilinear Parabolic PDE

We consider a general semilinear parabolic PDE in \([0, T] \times \mathbb{R}^d\):

\[
\frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \text{Tr}\left( \sigma \sigma^T(t, x)(\text{Hess}_x u)(t, x) \right) + \nabla u(t, x) \cdot \mu(t, x) \\
+ f(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x)) = 0.
\]

The terminal condition \(u(T, x) = g(x)\) is given. To fix ideas, we are interested in the solution at \(t = 0, x = \xi\) for some vector \(\xi \in \mathbb{R}^d\).

Suppose \(X_t = \xi + \int_0^t \mu(s, X_s) \, ds + \int_0^t \sigma(s, X_s) \, dW_s\), by Itô’s lemma,

\[
u(t, X_t) - u(0, X_0) = - \int_0^t f(s, X_s, u(s, X_s), \sigma^T(s, X_s) \nabla u(s, X_s)) \, ds \\
+ \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s) \, dW_s.
\]
Connection between PDE and BSDE

• The link between parabolic PDEs and backward stochastic differential equations (BSDEs) has been extensively investigated (Pardoux & Peng 1992, El Karoui et al. 1997, etc).

• In particular, Markovian BSDEs give a nonlinear Feynman-Kac representation of some nonlinear parabolic PDEs.

• Consider the following BSDE

\[
\begin{aligned}
X_t &= \xi + \int_0^t \mu(s, X_s) \, ds + \int_0^t \sigma(s, X_s) \, dW_s, \\
Y_t &= g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s) \, ds - \int_t^T (Z_s)^T \, dW_s,
\end{aligned}
\]

The solution is an adapted process \( \{(X_t, Y_t, Z_t)\}_{t \in [0,T]} \) with values in \( \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \).
Connection between PDE and BSDE

- Under suitable regularity assumptions, the BSDE is well-posed and related to the PDE in the sense that for all $t \in [0, T]$ it holds a.s. that

$$Y_t = u(t, X_t) \quad \text{and} \quad Z_t = \sigma^T(t, X_t) \nabla u(t, X_t).$$

- In other words, given the stochastic process satisfying

$$X_t = \xi + \int_0^t \mu(s, X_s) \, ds + \int_0^t \sigma(s, X_s) \, dW_s,$$

the solution of PDE satisfies the following SDE

$$u(t, X_t) - u(0, X_0) = -\int_0^t f(s, X_s, u(s, X_s), \sigma^T(s, X_s) \nabla u(s, X_s)) \, ds$$

$$+ \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s) \, dW_s.$$
Table of Contents

1. Background

2. BSDE Formulation of Parabolic PDE

3. Neural Network Approximation

4. Stochastic Control in Discrete Time

5. Numerical Examples of High-Dimensional PDEs

6. Convergence of the Deep BSDE Method

7. Summary
Neural Network Approximation

- **Key step:** approximate the function \( x \mapsto \sigma^T(t, x) \nabla u(t, x) \) at each discretized time step \( t = t_n \) by a feedforward neural network

\[
\sigma^T(t_n, X_{t_n}) \nabla u(t_n, X_{t_n}) = (\sigma^T \nabla u)(t_n, X_{t_n}) 
\approx (\sigma^T \nabla u)(t_n, X_{t_n} | \theta_n),
\]

where \( \theta_n \) denotes neural network parameters.

- **Observation:** we can stack all the subnetworks together to form a deep neural network (DNN) as a whole, based on the time discretization (see the next two slides).
Time Discretization

We consider the simple Euler scheme of the BSDE, with a partition of the time interval $[0, T]$, $0 = t_0 < t_1 < \ldots < t_N = T$:

\[ X_{t_{n+1}} - X_{t_n} \approx \mu(t_n, X_{t_n}) \Delta t_n + \sigma(t_n, X_{t_n}) \Delta W_n, \]

and

\[
\begin{align*}
&u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) \\
&\quad \approx -f(t_n, X_{t_n}, u(t_n, X_{t_n}), \sigma^T(t_n, X_{t_n}) \nabla u(t_n, X_{t_n})) \Delta t_n \\
&\quad \quad + [\nabla u(t_n, X_{t_n})]^T \sigma(t_n, X_{t_n}) \Delta W_n,
\end{align*}
\]

where

\[ \Delta t_n = t_{n+1} - t_n, \quad \Delta W_n = W_{t_{n+1}} - W_{t_n}. \]
**Figure:** Network architecture for solving parabolic PDEs. Each column corresponds to a subnetwork at time $t = t_n$. The whole network has $(H + 1)(N - 1)$ layers in total that involve free parameters to be optimized simultaneously.
Why such deep networks can be trained?

**Intuition:** there are skip connections between different subnetworks

\[ u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) \approx -f(t_n, X_{t_n}, u(t_n, X_{t_n}), (\sigma^T \nabla u)(t_n, X_{t_n} | \theta_n)) \Delta t_n \]

\[ + (\sigma^T \nabla u)(t_n, X_{t_n} | \theta_n) \Delta W_n \]
Optimization

- This network takes the paths \( \{X_{tn}\}_{0 \leq n \leq N} \) and \( \{W_{tn}\}_{0 \leq n \leq N} \) as the input data and gives the final output, denoted by \( \hat{u}(\{X_{tn}\}_{0 \leq n \leq N}, \{W_{tn}\}_{0 \leq n \leq N}) \), as an approximation to \( u(t_N, X_{t_N}) \).

- The error in the matching of given terminal condition defines the expected loss function

\[
l(\theta) = \mathbb{E}\left[ |g(X_{t_N}) - \hat{u}(\{X_{tn}\}_{0 \leq n \leq N}, \{W_{tn}\}_{0 \leq n \leq N})|^2 \right].
\]

- The paths can be simulated easily. Therefore the commonly used SGD algorithm fits this problem well.

- We call the introduced methodology deep BSDE method since we use the BSDE and DNN as essential tools.
Table of Contents

1. Background
2. BSDE Formulation of Parabolic PDE
3. Neural Network Approximation
4. Stochastic Control in Discrete Time
5. Numerical Examples of High-Dimensional PDEs
6. Convergence of the Deep BSDE Method
7. Summary
Consider a classical linear-quadratic-Gaussian (LQG) control problem in $\mathbb{R}^d$:

$$dX_t = 2\sqrt{\lambda} m_t \, dt + \sqrt{2} \, dW_t,$$

with cost functional $J(\{m_t\}_{0 \leq t \leq T}) = \mathbb{E} \left[ \int_0^T \|m_t\|_2^2 \, dt + g(X_T) \right]$. The HJB equation for this problem is

$$\frac{\partial u}{\partial t}(t, x) + \Delta u(t, x) - \lambda \|\nabla u(t, x)\|_2^2 = 0.$$

The optimal control is given by

$$m^*_t = -\sqrt{\lambda} \nabla u(t, x), \quad \text{(recall } Z_t = \sigma^T(t, X_t) \nabla u(t, X_t)).$$

In the context of BSDE for control, $Y_t$ denotes the optimal value and $Z_t$ denotes the optimal control (up to a constant scaling).
Formulation of Stochastic Control

Model dynamics:

\[ s_{t+1} = s_t + b_t(s_t, a_t) + \xi_{t+1}, \]

\( s_t \) is state, \( a_t \) is control, \( \xi_t \) is randomness. Consider objective:

\[
\min_{\{a_t\}_{t=0}^{T-1}} \mathbb{E}\left\{ \sum_{t=0}^{T-1} c_t(s_t, a_t(s_t)) + c_T(s_T) \mid s_0 \right\},
\]

Define cumulative cost for later use:

\[
C_t = \sum_{\tau=0}^{t} c_{\tau}(s_{\tau}, a_{\tau}), \quad t = 0, 1, \cdots, T - 1,
\]

\[
C_T = C_{T-1} + c_T(s_T).
\]

Real examples of stochastic control: portfolio optimization, robotics, resource allocation, mean field games, etc.
We look for a feedback control:

\[ a_t = a_t(s_t). \]

- Traditional methods in operation research: discretize state and/or control into finite spaces + approximate dynamic programming.

- Neural network approximation:

\[ a_t(s_t) \approx a_t(s_t|\theta_t), \]

Solve directly the approximate optimization problem

\[
\min_{\{\theta_t\}_{t=0}^{T-1}} \mathbb{E}\left\{ \sum_{t=0}^{T-1} c_t(s_t, a_t(s_t|\theta_t)) + c_T(s_T) \right\},
\]

rather than dynamic programming principle.
Network Architecture\(^1\)

Figure: Network architecture for solving stochastic control in discrete time. The whole network has \((N + 1)T\) layers in total that involve free parameters to be optimized simultaneously. Each column (except \(\xi_t\)) corresponds to a sub-network at \(t\).

\(^1\)J. Han and W. E, arXiv:1611.07422
Deep Reinforcement Learning (DRL) has achieved great success in game domains and sophisticated control tasks. A common strategy is to represent policy function (control) through neural networks.

Recall that in the example of LQG control problem, $Z_t$ denotes the optimal control, which is approximated by neural networks.

**Table:** Informal analogy

<table>
<thead>
<tr>
<th>Deep BSDE method</th>
<th>DRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSDE</td>
<td>Markov decision model</td>
</tr>
<tr>
<td>gradient of the solution</td>
<td>optimal policy function</td>
</tr>
</tbody>
</table>
Table of Contents

1. Background

2. BSDE Formulation of Parabolic PDE

3. Neural Network Approximation

4. Stochastic Control in Discrete Time

5. Numerical Examples of High-Dimensional PDEs

6. Convergence of the Deep BSDE Method

7. Summary
Implementation

- Each subnetwork has 4 layers, with 1 input layer \((d\text{-dimensional})\), 2 hidden layers (both \(d + 10\text{-dimensional}\)), and 1 output layer \((d\text{-dimensional})\).

- Choose the rectifier function (ReLU) as the activation function and optimize with Adam method.

- The means and the standard deviations of the relative errors are approximated by 5 independent runs of the algorithm with different random seeds.

- Implement in Tensorflow and reported examples are all run on a Macbook Pro.

- Github: https://github.com/frankhan91/DeepBSDE
LQG Example Revisited

We solve the introduced HJB equation in $[0, 1] \times \mathbb{R}^{100}$. It admits an explicit formula, which allows accuracy test:

$$u(t, x) = -\frac{1}{\lambda} \ln \left( \mathbb{E} \left[ \exp \left( -\lambda g(x + \sqrt{2W_{T-t}}) \right) \right] \right).$$

**Figure:** Left: Relative error of the deep BSDE method for $u(t=0, x=(0, \ldots, 0))$ when $\lambda = 1$, which achieves $0.17\%$ in a runtime of 330 seconds. Right: Optimal cost $u(t=0, x=(0, \ldots, 0))$ against different $\lambda$. 

Black-Scholes Equation with Default Risk

- The classical Black-Scholes model can and should be augmented by some important factors in real markets, including defaultable securities, transactions costs, uncertainties in the model parameters, etc.

- Ideally the pricing models should take into account the whole basket of financial derivative underlyings, resulting in high-dimensional nonlinear PDEs.

- To test the deep BSDE method, we study a special case of the recursive valuation model with default risk (Duffie et al. 1996, Bender et al. 2015).
• Consider the fair price of a European claim based on 100 underlying assets conditional on no default having occurred yet.

• The underlying asset price moves as a geometric Brownian motion and the possible default is modeled by the first jump time of a Poisson process.

• The claim value is modeled by a parabolic PDE with the nonlinear function

\[
f(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x)) = -(1 - \delta) Q(u(t, x)) u(t, x) - R u(t, x).
\]
Black-Scholes Equation with Default Risk

The not explicitly known “exact” solution at $t = 0$
$x = (100, \ldots, 100)$ is computed by the multilevel Picard method.

Figure: Approximation of $u(t=0, x=(100, \ldots, 100))$ against number of iteration steps. The deep BSDE method achieves a relative error of size 0.46% in a runtime of 617 seconds.
The Allen-Cahn equation is a reaction-diffusion equation for the modeling of phase separation and transition in physics. Here we consider a typical Allen-Cahn equation with the “double-well potential” in 100-dimensional space:

\[
\frac{\partial u}{\partial t}(t, x) = \Delta u(t, x) + u(t, x) - [u(t, x)]^3,
\]

with initial condition \( u(0, x) = g(x) \).
Allen-Cahn Equation

The not explicitly known “exact” solution at $t = 0.3$, $x = (0, \ldots, 0)$ is computed by the branching diffusion method.

**Figure:** Left: relative error of the deep BSDE method for $u(t=0.3, x=(0, \ldots, 0))$, which achieves 0.30% in a runtime of 647 seconds. Right: time evolution of $u(t, x=(0, \ldots, 0))$ for $t \in [0, 0.3]$, computed by means of the deep BSDE method.
An Example with Oscillating Explicit Solution

We consider an example studied for the numerical methods of PDE in literature (Gobet & Turkedjiev 2017). We set $d = 100$ instead of $d = 2$.

The PDE is constructed artificially in a form

$$\frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \Delta u(t, x) + \min\left\{1, (u(t, x) - u^*(t, x))^2\right\} = 0,$$

in which $u^*(t, x)$ is the explicit oscillating solution

$$u^*(t, x) = \kappa + \sin(\lambda \sum_{i=1}^{d} x_i) \exp\left(\frac{\lambda^2 d(t-T)}{2}\right).$$
Effect of Number of Hidden Layers

Table: The mean and standard deviation (std.) of the relative error for the above PDE, obtained by the deep BSDE method with different number of hidden layers.

<table>
<thead>
<tr>
<th>Number of layers†</th>
<th>29</th>
<th>58</th>
<th>87</th>
<th>116</th>
<th>145</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of relative error</td>
<td>2.29%</td>
<td>0.90%</td>
<td>0.60%</td>
<td>0.56%</td>
<td>0.53%</td>
</tr>
<tr>
<td>Std. of relative error</td>
<td>0.0026</td>
<td>0.0016</td>
<td>0.0017</td>
<td>0.0017</td>
<td>0.0014</td>
</tr>
</tbody>
</table>

† We only count the layers that have free parameters to be optimized.
## Effect of Activation Function

<table>
<thead>
<tr>
<th></th>
<th>Nonlinear BS</th>
<th>LQG</th>
<th>Allen-Cahn</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU</td>
<td>0.46% (0.0008)</td>
<td>0.17% (0.0004)</td>
<td>0.30% (0.0021)</td>
</tr>
<tr>
<td>Tanh</td>
<td>0.44% (0.0006)</td>
<td>0.17% (0.0005)</td>
<td>0.28% (0.0024)</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>0.46% (0.0004)</td>
<td>0.19% (0.0008)</td>
<td>0.38% (0.0026)</td>
</tr>
<tr>
<td>Softplus</td>
<td>0.45% (0.0007)</td>
<td>0.17% (0.0004)</td>
<td>0.18% (0.0017)</td>
</tr>
</tbody>
</table>

**Table:** The mean and standard deviation (in parenthesis) of relative error obtained by the deep BSDE method with different activation functions, for the nonlinear Black-Scholes equation, the Hamilton-Jacobi-Bellman equation, and the Allen-Cahn equation.
References and Follow-up Works

• References:
  ▶ Han, Jentzen, and E, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences (2018)
  ▶ E, Han, and Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Communications in Mathematics and Statistics (2017)

• Follow-up works:
Table of Contents

1. Background
2. BSDE Formulation of Parabolic PDE
3. Neural Network Approximation
4. Stochastic Control in Discrete Time
5. Numerical Examples of High-Dimensional PDEs
6. Convergence of the Deep BSDE Method
7. Summary
Deep BSDE Method for Coupled Case

Coupled FBSDE

\[
\begin{cases}
X_t = \xi + \int_0^t b(s, X_s, Y_s) \, ds + \int_0^t \sigma(s, X_s, Y_s) \, dW_s, \\
Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s) \, ds - \int_t^T (Z_s)^T \, dW_s,
\end{cases}
\]

Discrete scheme with approximation

\[
\begin{cases}
X_0^{\pi} = \xi, \quad Y_0^{\pi} = \mu_0^{\pi}(\xi), \\
X_{t_i+1}^{\pi} = X_{t_i}^{\pi} + b(t_i, X_{t_i}^{\pi}, Y_{t_i}^{\pi}) h + \sigma(t_i, X_{t_i}^{\pi}, Y_{t_i}^{\pi}) \Delta W_i, \\
Z_{t_i}^{\pi} = \phi_i^{\pi}(X_{t_i}^{\pi}, Y_{t_i}^{\pi}), \\
Y_{t_i+1}^{\pi} = Y_{t_i}^{\pi} - f(t_i, X_{t_i}^{\pi}, Y_{t_i}^{\pi}, Z_{t_i}^{\pi}) h + (Z_{t_i}^{\pi})^T \Delta W_i.
\end{cases}
\]

Objective function

\[
\inf_{\mu_0^{\pi} \in \mathcal{N}_0', \phi_i^{\pi} \in \mathcal{N}_i} F(\mu_0^{\pi}, \phi_0^{\pi}, \ldots, \phi_{N-1}^{\pi}) := \mathbb{E}|g(X_T^{\pi}) - Y_T^{\pi}|^2.
\]
Main Theorem 1

**Theorem (A Posteriori Estimation)**

*Under some assumptions, there exists a constant $C$, independent of $h$, $d$, and $m$, such that for sufficiently small $h$,*

\[
\sup_{t \in [0, T]} \left( \mathbb{E}|X_t - \hat{X}_t^\pi|^2 + \mathbb{E}|Y_t - \hat{Y}_t^\pi|^2 \right) + \int_0^T \mathbb{E}|Z_t - \hat{Z}_t^\pi|^2 \, dt \\
\leq C \left[ h + \mathbb{E}|g(X_T^\pi) - Y_T^\pi|^2 \right],
\]

*where $\hat{X}_t^\pi = X_{t_i}^\pi$, $\hat{Y}_t^\pi = Y_{t_i}^\pi$, $\hat{Z}_t^\pi = Z_{t_i}^\pi$ for $t \in [t_i, t_{i+1})$.***
Main Theorem 2

**Theorem (Upper Bound of Optimal Loss)**

*Under some assumptions, there exists a constant $C$, independent of $h$, $d$ and $m$, such that for sufficiently small $h$,*

$$
\mathbb{E}|g(X_T^\pi) - Y_T^\pi|^2 \\
\leq C \left\{ h + \mathbb{E}|Y_0 - \mu^\pi_0(\xi)|^2 + \sum_{i=0}^{N-1} \mathbb{E}\left[ \mathbb{E}[\tilde{Z}_{t_i} | X_{t_i}^\pi, Y_{t_i}^\pi] - \phi^\pi_i(X_{t_i}^\pi, Y_{t_i}^\pi)|^2 h \right] \right\},
$$

where $\tilde{Z}_{t_i} = h^{-1}\mathbb{E}[\int_{t_i}^{t_{i+1}} Z_t \, dt | \mathcal{F}_{t_i}]$. If $b$ and $\sigma$ are independent of $Y$, the term $\mathbb{E}[\tilde{Z}_{t_i} | X_{t_i}^\pi, Y_{t_i}^\pi]$ can be replaced with $\mathbb{E}[\tilde{Z}_{t_i} | X_{t_i}^\pi]$. 
Assumptions

1. $b, \sigma, f$ are $\frac{1}{2}$-Hölder continuous with respect to $t$ and bounded at $(t, 0, 0)$ with respect to $t$. $b, \sigma, f, g$ are uniformly Lipschitz continuous with respect to $(x, y, z)$.

2. **Weak coupling or monotonicity conditions:** one of the following five cases holds:
   - Small time duration, that is, $T$ is small.
   - Weak coupling of $Y$ into the forward SDE, that is, $b_y$ and $\sigma_y$ are small.
   - Weak coupling of $X$ into the backward SDE, that is, $f_x$ and $g_x$ are small.
   - $f$ is strongly decreasing in $y$, that is,
     \[ [f(t, x, y_1, z) - f(t, x, y_2, z)] \Delta y \leq k_f |\Delta y|^2, \text{ with } k_f \text{ being very negative.} \]
   - $b$ is strongly decreasing in $x$, that is,
     \[ [b(t, x_1, y) - b(t, x_2, y)]^T \Delta x \leq k_b |\Delta x|^2, \text{ with } k_b \text{ being very negative.} \]
Two Useful Theorems for Proof (1)

Theorem (Path Regularity)

There is constant $C$ independent of $h$, $d$ and $m$, such that

$$\sup_{t \in [0,T]} (\mathbb{E}|X_t - \tilde{X}_t|^2 + \mathbb{E}|Y_t - \tilde{Y}_t|^2) + \int_0^T \mathbb{E}|Z_t - \tilde{Z}_t|^2 \, dt \leq C(1 + \mathbb{E}|\xi|^2)h,$$

in which $\tilde{X}_t = X_{t_i}$, $\tilde{Y}_t = Y_{t_i}$, $\tilde{Z}_t = h^{-1} \mathbb{E}[\int_{t_i}^{t_i+1} Z_t \, dt \mid \mathcal{F}_{t_i}]$ for $t \in [t_i, t_{i+1})$. If $Z_t$ is càdlàg, we can replace $h^{-1} \mathbb{E}[\int_{t_i}^{t_i+1} Z_t \, dt \mid \mathcal{F}_{t_i}]$ by $Z_{t_i}$. 
Theorem (Convergence of Implicit Scheme)

For sufficiently small $h$, the following discrete time equation ($0 \leq i \leq N - 1$)

\[
\begin{align*}
X_0^\pi &= \xi, \\
X^\pi_{t_{i+1}} &= X^\pi_{t_i} + b(t_i, X^\pi_{t_i}, Y^\pi_{t_i})h + \sigma(t_i, X^\pi_{t_i}, Y^\pi_{t_i})\Delta W_i, \\
Y^\pi_T &= g(X^\pi_T), \\
Z^\pi_{t_i} &= \frac{1}{h} \mathbb{E}[Y^\pi_{t_{i+1}} \Delta W_i | \mathcal{F}_{t_i}], \\
Y^\pi_{t_i} &= \mathbb{E}[Y^\pi_{t_{i+1}} + f(t_i, X^\pi_{t_i}, Y^\pi_{t_i}, Z^\pi_{t_i})h | \mathcal{F}_{t_i}],
\end{align*}
\]

has a solution $(X^\pi_{t_i}, Y^\pi_{t_i}, Z^\pi_{t_i})$ such that $X^\pi_{t_i} \in L^2(\Omega, \mathcal{F}_{t_i}, \mathbb{P})$ and

\[
\sup_{t \in [0,T]} (\mathbb{E}|X_t - \overline{X}_t^\pi|^2 + \mathbb{E}|Y_t - \overline{Y}_t^\pi|^2) + \int_0^T \mathbb{E}|Z_t - \overline{Z}_t^\pi|^2 \, dt \leq C(1 + \mathbb{E}|\xi|^2)h,
\]

where $\overline{X}_t^\pi = X^\pi_{t_i}$, $\overline{Y}_t^\pi = Y^\pi_{t_i}$, $\overline{Z}_t^\pi = Z^\pi_{t_i}$ for $t \in [t_i, t_{i+1})$. $C$ is a constant depending on coefficient functions and $T$. 

Table of Contents

1. Background
2. BSDE Formulation of Parabolic PDE
3. Neural Network Approximation
4. Stochastic Control in Discrete Time
5. Numerical Examples of High-Dimensional PDEs
6. Convergence of the Deep BSDE Method
7. Summary
We propose the so-called **deep BSDE method**, which can solve general nonlinear high-dimensional parabolic PDEs.

1. We reformulate the parabolic PDEs as BSDEs and approximate the unknown gradient by deep neural networks.

2. Numerical results validate the proposed algorithm in high dimensions, in terms of both accuracy and speed.

3. A posteriori error estimation is given and there indeed exist good approximate solutions given a universal approximator.

4. This opens up new possibilities in various disciplines involving dynamical system modelings.