Deep Learning-Based Numerical Methods for High-Dimensional Parabolic PDEs and BSDEs

Hao Quan

Johns Hopkins University

haoquan@jhu.edu

November 11, 2021

Semilinear Parabolic PDEs

We consider a general class of *semilinear parabolic PDEs*, which is of the form:

$$\frac{\partial u}{\partial t}(t,x)+\frac{1}{2}\mathrm{Tr}[\sigma\sigma^T(t,x)\mathbf{H}_x u(t,x)]+\nabla u(t,x)\cdot\mu_{t,x}+f(t,x,u,\sigma^T\nabla u)=0 \tag{1}$$

Note that:

- PDE is defined on $[0,T]\times\mathbb{R}^d$
- Tr is the trace
- $\mathbf{H}_x u(t,x)$ is the Hessian matrix w.r.t. x
- $\sigma(t,x):\mathbb{R}\times\mathbb{R}^d\to\mathbb{R}^{d\times d}$ is a known matrix-valued function
- $\mu(t,x):\mathbb{R}\times\mathbb{R}^d\to\mathbb{R}^d$ is a known vector-valued function
- f is a known nonlinear function

We want to find a function u satisfying the PDE given a terminal condition $u(T,x)=g(x)$ for some function g.

Examples

- Nonlinear Black-Scholes equation

$$\frac{\partial u}{\partial t} + \mu x \cdot \nabla u + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} = -[(1-\delta)Q \circ u]u - Ru \tag{2}$$

- Hamilton-Jabobi-Bellman equation (HJB) for linear-quadratic Gaussian (LQG) control

$$\frac{\partial u}{\partial t} + \Delta u - \lambda |\nabla u|^2 = 0 \tag{3}$$

- Allen-Cahn equation

$$\frac{\partial u}{\partial t} = \Delta u + u - u^3 \tag{4}$$

Curse of dimensionality

- It is well known that numerical algorithms for high-dimensional PDEs have long suffered the so called "**curse of dimensionality**", namely, the complexity of the algorithm grows exponentially as the dimension grows.
- A limited number of cases where practical high-dimensional algorithms have been developed in the literature:
  - High dimensional linear parabolic PDEs: Feynman-Kac + Monte Carlo
  - Invscid Hamilton-Jabobi Equation: Algorithms based on Hopf formula (Darbon and Osher 2016)
  - Semi-parabolic PDEs with polynomial nonlinearity: Branching Duffusion method (Henry-Labordere 2012,Henry-Labordere, Tan, and Touzi 2014)
  - General semi-parabolic PDEs: Multi-level Picard Method (Weinan, Hutzenthaler, et al. 2019)

Curse of dimensionality

- **Deep Learning** has emerged in machine learning in recent years and has proven to be very effective in dealing with large class of high-dimensional problems in computer vision, natural language processing, time series analysis, etc. This leads to the **hope** that Deep learning might hold the key to tackle the curse of dimensionality.

Hope

"Hope, it is the quintessential human delusion, simultaneously the source of your greatest strength, and your greatest weakness."

— The Architect, The Matrix Reloaded

Curse of dimensionality

- However, **Deep Learning** has emerged in machine learning in recent years and has proven to be very effective in dealing with large class of high-dimensional problems in computer vision, natural language processing, time series analysis, etc. Deep learning might hold the key to tackle the curse of dimensionality.
- The bridge between high dimensional parabolic PDEs and Deep Learning is **Backward Stochastic Differential Equation**.

Nonlinear Feynman-Kac formula

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $W : [0, T] \times \Omega \to \mathbb{R}^d$ be a d-dimensional standard Brownian motion, and $\{\mathcal{F}_t\}_{t\in[0,T]}$ be the filtration generated by $\{W_t\}_{t\in[0,T]}$. Consider $\{\mathcal{F}_t\}_{t\in[0,T]}$-adapted process $\{(X_t, Y_t, Z_t)\}_{t\in[0,T]}$ s.t.

$$X_t = \xi + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s \tag{5}$$

$$Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s)ds - \int_t^T Z_s^T dW_s \tag{6}$$

Under suitable regularity assumption on μ, σ and f, one can prove existence and uniqueness of the solution process and $\forall t \in [0, T]$, and it holds $\mathbb{P}-$a.s. (see Pardoux and Peng 1992) that

$$Y_t = u(t, X_t), \quad Z_t = \sigma^T(x, X_t)\nabla u(t, X_t) \tag{7}$$

Conversion to BSDE

Now the solution u of the PDE(1) satisfies the following BSDE

$$u(t, X_t) - u(0, X_0) = -\int_0^t f(s, X_s, u(s, X_s), \sigma^T \nabla u(s, X_s))ds \tag{8}$$

$$+ \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s) dW_s \tag{9}$$

Discretization and Euler scheme

In order to numerically solve this BSDE, we apply a temporal discretization to the equation (8)-(9). We discretize time via following partition:

$$[0, T] : 0 = t_0 < t_1 < t_2 < \cdots < t_N = T \tag{10}$$

We consider the Euler-Maruyama scheme for $n \in \{0, 1, 2, \ldots, N-1\}, \Delta t_n = t_{n+1} - t_n, \Delta W_n = W t_{n+1} - W_{t_n}$:

$$X_{t_{n+1}} - X_{t_n} \approx \mu(t_n, X_{t_n})\Delta t_n + \sigma(t_n, X_{t_n})\Delta W_n \tag{11}$$

and we approximate $u(t, X_t)$:

$$\begin{aligned} u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) = &-f(t_n, X_{t_n}, u(t_n, X_{t_n}), \sigma^T \nabla u(t_n, X_{t_n}))\Delta t_n \\ &+ [\nabla u(t_n, X_{t_n})]^T \sigma(t_n, X_{t_n})\Delta W_n \end{aligned}$$

Unknown gradient

The finite difference equation (FDE) in incremental form:

$$u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) = -f(t_n, X_{t_n}, u(t_n, X_{t_n}), (\sigma^T \nabla u)(t_n, X_{t_n}))\Delta t_n + [\nabla u(t_n, X_{t_n})]^T \sigma(t_n, X_{t_n})\Delta W_n$$

However, $\nabla u(t_n, X_{t_n})$ is unknown and hard to estimate. A deep learning approach can be used to obtain an estimate for $\sigma^T \nabla u(t_n, X_{t_n}))$ at each time step. This approach is so called "Deep BSDE method"

Deep BSDE method

Given the temporal discretization, $\{X_{t_n}\}_{0\leq n\leq N}$ can be easily sampled by Monte-Carlo using equation (11). The key step next is to approximate the function $x \to \sigma^T(t,x)\nabla u(t,x)$ at each time $t = t_n$ by feedforward neural network:

$$(\sigma^T \nabla u)(t_n, X_{t_n}) \approx (\sigma^T \nabla u)(t_n, X_{t_n}; \theta_n) \tag{12}$$

for $n = 1, \ldots, N-1$, where θ_n denotes parameters of the neural network approximating $x \to \sigma^T(t,x)\nabla u(t,x)$.

Deep BSDE method

- We stack all the subnetworks in (12) together to form a deep neural network as a whole via the summation of the incremental (FDE) over $n = 1, \ldots, N-1$.
- Input: $\{(X_{t_n}, W_{t_n})\}_{0 \le n \le N}$
  Intermediate output: $(\sigma^T \nabla u)(t_n, X_{t_n})_{0 \le n \le N-1}$
  Final output: $\hat{u}(t_{n+1}, X_{t_{n+1}})_{0 \le n \le N-1}$
- Set of all parameters:

$$\theta = \{\theta_{u_0}, \theta_{\nabla u_0}, \theta_1, \theta_2, \ldots, \theta_{N-1}\} \tag{13}$$

Deep BSDE method

- Inspired by the terminal condition:

$$\mathbb{E}[|g(X_T) - u(T, X_T)|^2] = 0 \tag{14}$$

We define the loss function:

$$\ell(\theta) := \mathbb{E}[|g(X_{t_N}) - u(t_N, X_{t_N})|^2] \tag{15}$$

- Commonly used optimization methods like Stochastic Grdient Descent (SGD) or Adam optimizer fit very well to minimize $\ell(\theta)$ over θ, as paths $\{X_t\}_{t\in[0,T]}$ can be easily simulated.
- For $X_{t_0} = \xi \in \mathbb{R}^d$, once a number of iterations have occurred, a final estimate of the initial value of the solution is reached:

$$\theta_{u_0} \approx u_0(\xi) = u(t_0, \xi) \tag{16}$$

Network Architecture

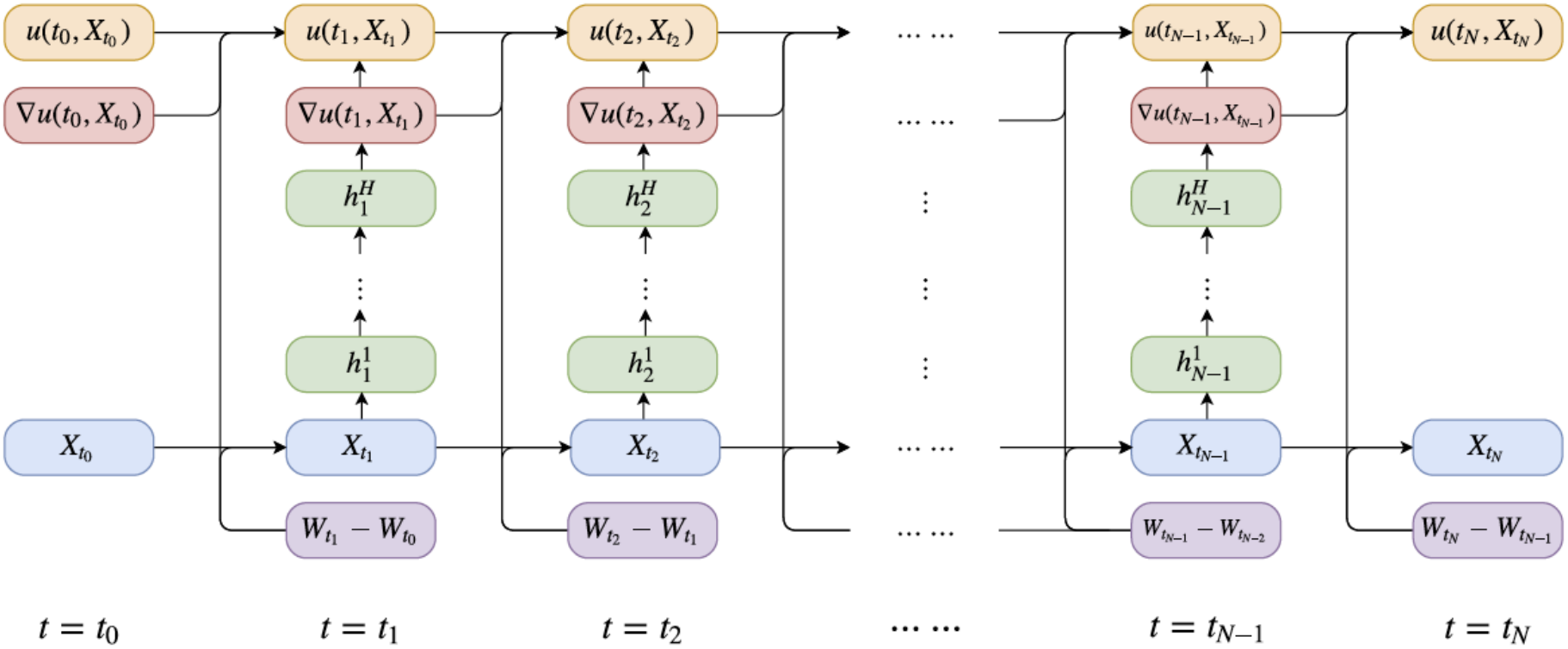


Figure: (Weinan, Han, and Jentzen 2017) Illustration of the network architecture for solving semilinear parabolic PDEs. Each column corresponds to a subnetwork at time $t = t_n$ with H hidden layers and intermediate neurons $h_n^1, \ldots, h_n^H$. The whole network has (H+1)(N-1) layers in total that involve free parameters to be optimized simultaneously

Ex.1 Hamilton-Jacobi-Bellman equation

We consider a classical linear-quadratic Gaussian control problem in dimension $d = 100$:

$$dX_t = 2\sqrt{\lambda}c(t)dt + \sqrt{2}dW_t \tag{17}$$

with $t \in [0, T], X_0 = x$, constant $\lambda > 0$, the control $\{c(t)\}_{t\in[0,T]}$ and the state process $\{X_t\}_{t\in[0,T]}$.
The goal is to minimize the cost functional:

$$J(c) = \mathbb{E}\left[\int_0^T |c(t)|^2 dt + g(X_T)\right] \tag{18}$$

with

$$g(x) = \log((1 + |x|^2)/2) \tag{19}$$

Ex.1 Hamilton-Jacobi-Bellman equation

Then the HJB equation for this problem is given by:

$$\frac{\partial u}{\partial t} + \Delta u - \lambda |\nabla u|^2 = 0, \quad u(T, x) = g(x) \tag{20}$$

The value of the solution u at time $t = 0$ represents the optimal cost when the state starts from x. By Itô's formula, one can show that (20) admits the explicit formula:

$$u(t, x) = -\frac{1}{\lambda} \log(\mathbb{E}[\exp(-\lambda g(x + \sqrt{2} W_{T-t}))]) \tag{21}$$

Ex.1 Hamilton-Jacobi-Bellman equation

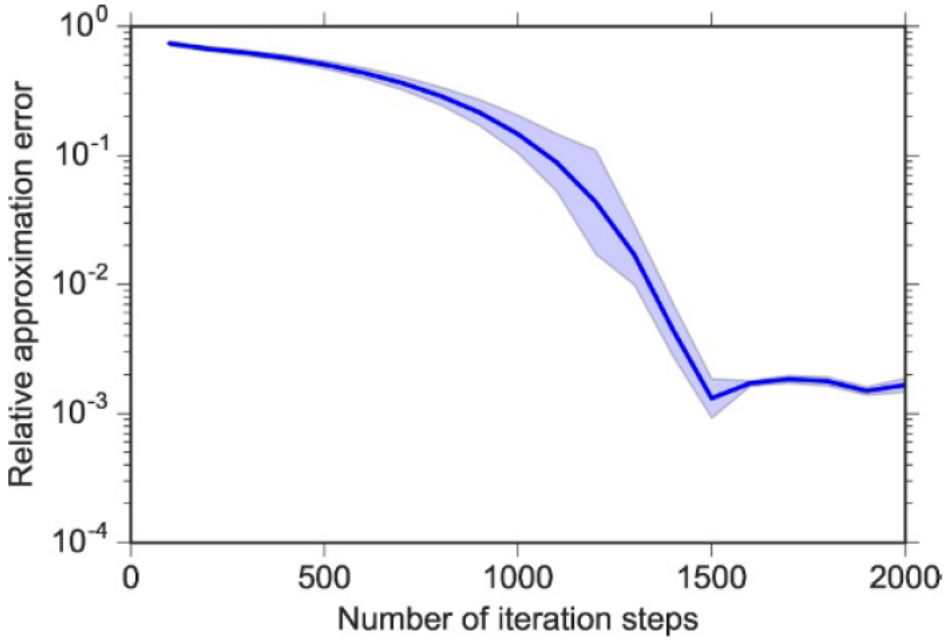


Figure: (Han, Jentzen, and Weinan 2018) Relative error of the deep BSDE method for $u(t = 0, x = (0, ..., 0))$ when $\lambda = 1$ against the number of iteration steps in the case of the 100-dimensional HJB Eq. (20) with $N = 20$ equidistant time steps and learning rate 0.01. The shaded area depicts the mean $\pm$ the standard deviation of the relative error for five different runs. The deep BSDE method achieves a relative error of size 0.17% in a runtime of 330 s

Ex.1 Hamilton-Jacobi-Bellman equation

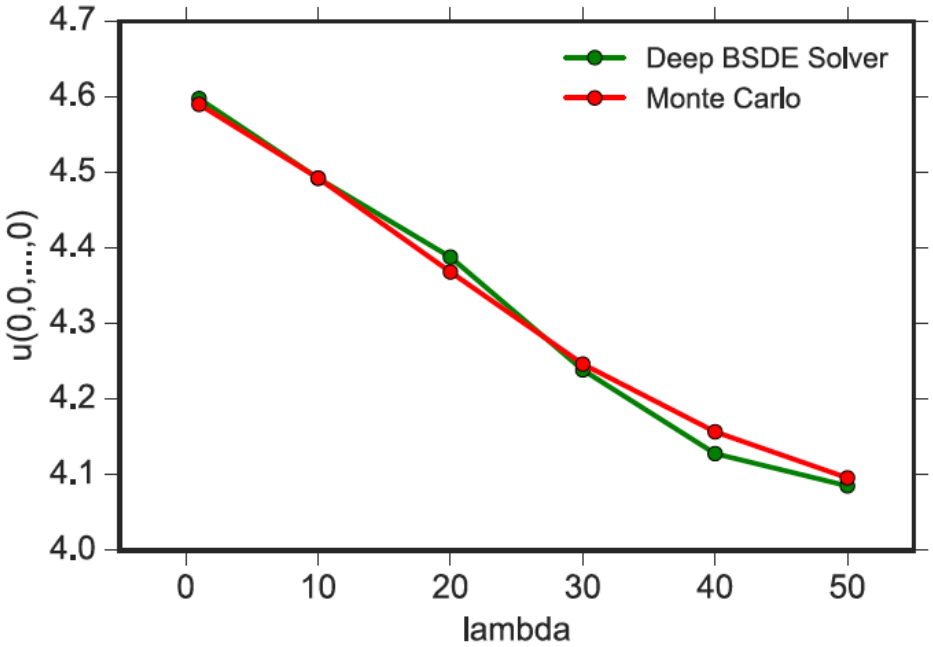


Figure: (Han, Jentzen, and Weinan 2018) Optimal cost $u(t = 0, x = (0, ..., 0))$ against different values of λ in the case of the 100-dimensional HJB Eq. (20), obtained by the deep BSDE method and classical Monte Carlo simulations of Eq. (21)

Ex.2 Allen Cahn equation

We consider the following problem of dimension $d = 100$:

$$\frac{\partial u}{\partial t} = \Delta u + u - u^3 \tag{22}$$

subject to the initial condition:

$$u(0, x) = g(x) = 1/(2 + 0.4|x|^2) \tag{23}$$

Consider the time reversal $\tilde{t} = T - t$. Then the problem becomes:

$$\frac{\partial \tilde{u}}{\partial \tilde{t}} + \Delta \tilde{u} + \tilde{u} - \tilde{u}^3 = 0 \tag{24}$$

subject to the terminal condition:

$$\tilde{u}(T, x) == u(0, x) = g(x) = 1/(2 + 0.4|x|^2) \tag{25}$$

Ex.2 Allen Cahn equation

Suppose we want to estimate the solution u at $t = 0.3, x = \mathbf{0} = (0, \ldots, 0) \in \mathbb{R}^d$. Since the equation (22) lacks analytical solution, we replace the exact solution $u(0.3, \mathbf{0})$ by 0.0528, which is computed by the branching diffusion method (see Henry-Labordere 2012 and Henry-Labordere, Tan, and Touzi 2014)

Ex.2 Allen Cahn equation

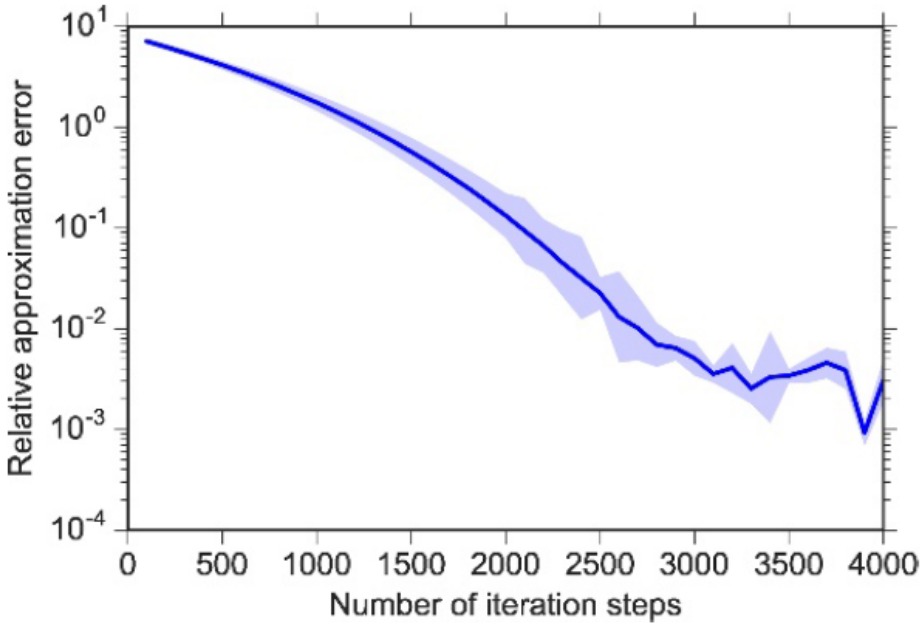


Figure: (Han, Jentzen, and Weinan 2018) Relative error of the deep BSDE method for $u(t = 0.3, x = \mathbf{0})$ against the number of iteration steps in the case of the 100- dimensional Allen–Cahn Eq.(22) with $N = 20$ equidistant time steps and learning rate 0.0005. The shaded area depicts the mean $\pm$ the standard deviation of the relative error for five different runs. The deep BSDE method achieves a relative error of size 0.30% in a runtime of 647 seconds

Ex.2 Allen Cahn equation

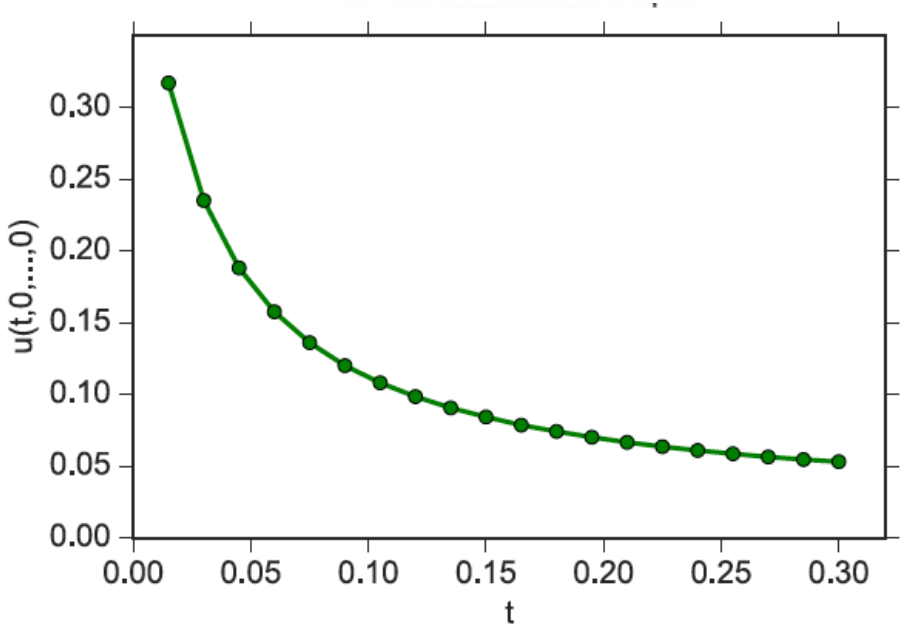
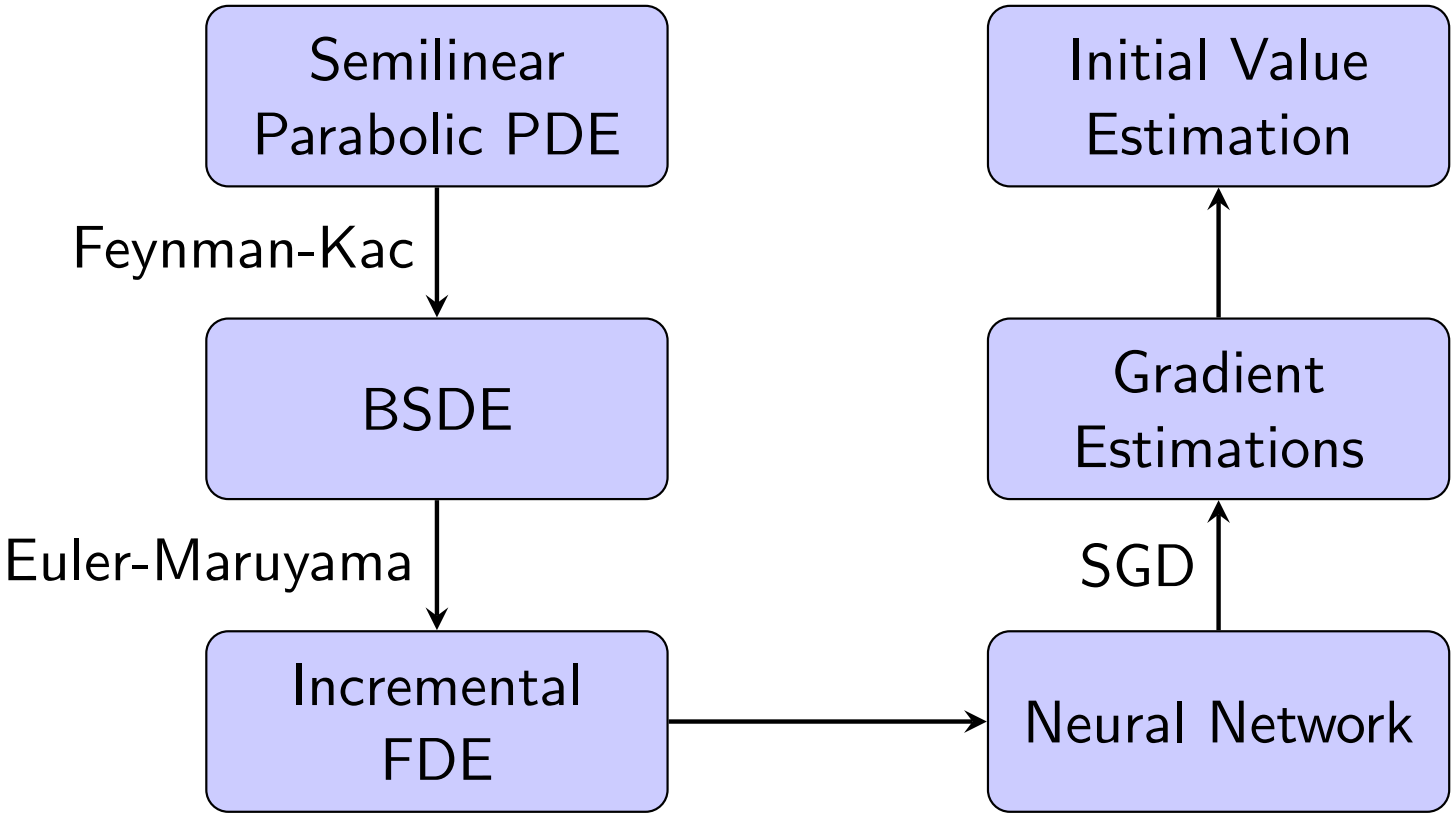


Figure: (Han, Jentzen, and Weinan 2018) Time evolution of $u(t, x = \mathbf{0})$ for $t \in [0, 0.3]$ in the case of the 100-dimensional Allen–Cahn Eq. (22) computed by means of the deep BSDE method

BSDE Method Flowchart



Conlcusion

- The deep BSDE method was used to obtain results for Hamilton-Jacobi-Bellman equation and Allen-Cahn equation
- The deep BSDE method was able to obtain results close to analytical solutions as well as solutions by other numerical methods
- The deep BSDE method works well for high-dimensional problems. However, for low dimensional problems, it might take much longer time than traditional method, considering that it only tracks one point at a time (as opposed to tracking over the whole domain)
- Exploration of boundary value problem; reduction of redundancy of the network architecture

References I

- Darbon, Jérôme and Stanley Osher (2016). "Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere". In: *Research in the Mathematical Sciences* 3.1, pp. 1–26.
- Han, Jiequn, Arnulf Jentzen, and E Weinan (2018). "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34, pp. 8505–8510.
- Henry-Labordere, Pierre (2012). "Counterparty risk valuation: A marked branching diffusion approach". In: *Available at SSRN 1995503*.
- Henry-Labordere, Pierre, Xiaolu Tan, and Nizar Touzi (2014). "A numerical algorithm for a class of BSDEs via the branching process". In: *Stochastic Processes and their Applications* 124.2, pp. 1112–1140.

References II

- Pardoux, Etienne and Shige Peng (1992). "Backward stochastic differential equations and quasilinear parabolic partial differential equations". In: *Stochastic partial differential equations and their applications*. Springer, pp. 200–217.
- Weinan, E, Jiequn Han, and Arnulf Jentzen (2017). "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations". In: *Communications in Mathematics and Statistics* 5.4, pp. 349–380.
- Weinan, E, Martin Hutzenthaler, et al. (2019). "On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations". In: *Journal of Scientific Computing* 79.3, pp. 1534–1571.