Deep Learning Techniques for Solving Semilinear Parabolic Partial Differential Equations

Evan Davis [1] Guangming Yao [2] Elizabeth Javor [3] Kalani Rubasinghe [2] Luis Antonio Topete Galván [4]

[1]*University of Wisconsin, Madison* [2]*Clarkson University* [3]Rochester Institute of Technology [4]*Universidad Autónoma del Estado de Hidalgo 42074, Mexico*

Semilinear Parabolic PDE

The goal of the Deep BSDE solver is to solve semilinear parabolic partial differential equations, those of the form:

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

Note that:

- Tr is the trace
- $\mathbf{H}_x u(t,x)$ is the Hessian
- $\sigma(t,x) : \mathbb{R}\times\mathbb{R}^d \to M^{d\times d}(\mathbb{R})$ 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

The goal is to obtain $u(0,\xi)$ for some fixed $\xi \in \mathbb{R}^d$.

Examples of Semilinear Parabolic PDE Problems

Ex 1: Heat Equation:
Let $\sigma = \sqrt{2}\mathbf{I}$, $\mu = 0$ and $f = 0$. Then:

$$\frac{\partial u}{\partial t}(t,x) = \Delta u(t,x)$$

Ex 2: Allen-Cahn Equation:
Let $\sigma = \sqrt{2}\mathbf{I}$, $\mu = 0$ and $f = u(t,x) - u(t,x)^3$. Then:

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

Equivalent Stochastic PDE

This is equivalent to the BSDE:

$$\begin{aligned} u(t,X_t) - u(0,X_0) = &-\int_0^t f(s,X_s,u(s,X_s),\sigma^T(s,X_s)\nabla u(s,X_s))ds \\ &+ \int_0^t [\nabla u(s,X_s)]^T\sigma(s,X_s)dW_s \end{aligned} \tag{1}$$

Discretization of SDE

Time is discretized via a partition:

$$[0,T] : 0 = t_0 < t_1 < t_2 < \ldots < t_N = T$$

An Euler scheme is used to discretize the equations. For $n \in \{0,1,2,\ldots,N-1\}$, let $\Delta t_n = t_{n+1} - t_n$ and $\Delta W_n = W_{t_{n+1}} - W_{t_n}$. This yields the following approximation for X_t:

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

The function $u(t,X_t)$ can be approximated:

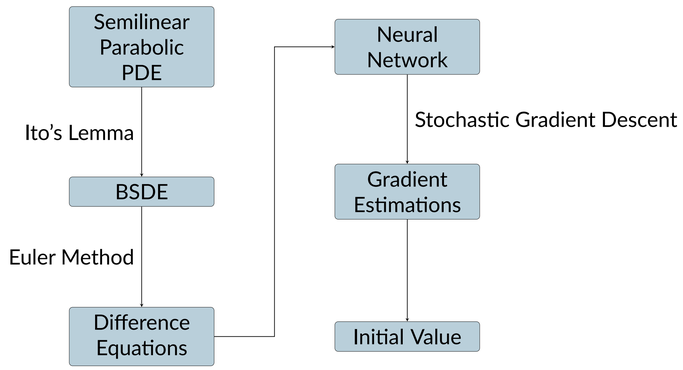
$$\begin{aligned} u(t_{n+1},X_{t_{n+1}}) - u(t_n,X_{t_n}) = &-f(t_n,X_{t_n},u(t_n,X_{t_n}),\sigma^T\nabla u(t_n,X_{t_n}))\Delta t_n \\ &+ [\nabla u(t_n,X_{t_n})]^T\sigma(t_n,X_{t_n})\Delta W_n \end{aligned}$$

The difference equations are:

$$\begin{aligned} X_{t_{n+1}} - X_{t_n} &\approx \mu(t_n,X_{t_n})\Delta t_n + \sigma(t_n,X_{t_n})\Delta W_n \\ u(t_{n+1},X_{t_{n+1}}) - u(t_n,X_{t_n}) &= -f(t_n,X_{t_n},u(t_n,X_{t_n}),\sigma^T\nabla u(t_n,X_{t_n}))\Delta t_n \\ &\quad + [\nabla u(t_n,X_{t_n})]^T\sigma(t_n,X_{t_n})\Delta W_n \end{aligned} \tag{2}$$

The critical issue is that at any given time step, an estimate of $\nabla u(t_n,X_{t_n})$ is not known. A deep learning approach can be used to obtain an estimate of $\sigma^T(t_n,X_{t_n})\nabla u(t_n,X_{t_n}))$ at each time step.

BSDE Solver Flowchart



Estimating the function value using Deep Learning

A feed-forward neural network with parameters θ is used to estimate the gradients at each time. Several Monte Carlo samples of $\{\Delta W_{t_n}\}_{0\le n\le N}$ are taken and the network estimates the end value $\hat{u}(\{X_{t_n}\},\{W_{t_n}\})$. The following loss function is used to improve the neural network.

$$\ell(\theta) = \mathbb{E}\left[\left|g(X_{t_N}) - \hat{u}(\{X_{t_n}\},\{W_{t_n}\})\right|^2\right]$$

Stochastic Gradient Descent is used to improve the network. Once a number of iterations have occurred, a final estimate of the initial function value is reached.

Example: Simple Reaction-Diffusion

Consider the following 2D problem:

$$\frac{\partial u}{\partial t}(t,x) = 0.2\Delta u(t,x) + 0.1u(t,x)$$

subject to the initial condition:

$$u(0,x) = g(x) = \cos(x_1) + \cos(x_2)$$

This problem has exact analytical solution:

$$u(t,x) = \exp(-0.1t)\left[\cos(x_1) + \cos(x_2)\right]$$

Consider the time reversal $t \to T - t$. Then the problem becomes:

$$\frac{\partial u}{\partial t}(t,x) + 0.2\Delta u(t,x) + 0.1u(t,x) = 0$$

with have terminal condition

$$u(T,x) = g(x) = \cos(x_1) + \cos(x_2)$$

2D surfaces of the solution and error

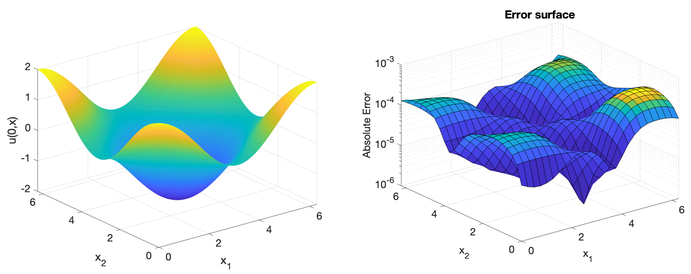


Figure 1. The surface of the solution in two dimensions and the surface of the absolute error in two dimensions.

Effect of Time Step Size

The following results were obtained for the deep BSDE solver. A 6×6 grid of points were tested over the domain $[0,2\pi]\times[0,2\pi]$. A total of 5000 iterations were used with a learning rate of 0.01 for the first half and 0.0002 for the second half. In two separate rounds of testing, 10 and 100 step sizes were used.

Δt	L_∞	L_{rms}	Average elapsed time
$\Delta t=10^{-2}$	3.002×10^{-4}	1.598×10^{-4}	38
$\Delta t=10^{-3}$	3.456×10^{-4}	1.340×10^{-4}	408.5

Table 1. Comparison of BSDE solution at different step sizes. Decreasing the step size by a factor of 10 had minimal impact on the error while increasing time greatly.

Effect of the Number of Iterations

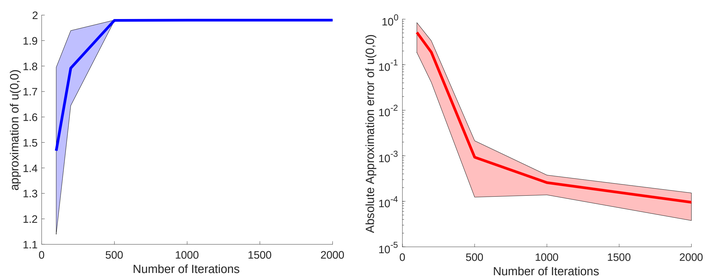


Figure 2. Left: The approximate value of $u(0,0)$ at different numbers of iterations of the Deep BSDE solver. As the number of iterations increases, the solver increases to an asymptotic value equal to the analytical solution. Right: The approximation error of the Deep BSDE solver at various numbers of iterations. Error decreases with increasing number of iterations.

Effect of the Number of Dimensions

To evaluate the performance of the Deep BSDE solver, the previous equation was considered in d dimensions with d being 5, 10, 20, 50, 100, and 200:

$$\frac{\partial u}{\partial t}(t,x) = 0.2\Delta u(t,x) + 0.1u(t,x)$$

Subject to $u(0,x) = g(x) = \sum_{i=1}^d \cos(x_i)$. this has solution:

$$u(t,x) = \exp(-0.1t)\sum_{i=1}^d \cos(x_i)$$

d	5	10	20	50	100	200
Y_0 (BSDE)	4.951	9.900	19.80	49.50	99.00	198.0
Y_0 (exact)	4.951	9.901	19.80	49.50	99.01	198.0
Absolute error	2.63×10^{-4}	4.82×10^{-4}	2.70×10^{-3}	2.30×10^{-3}	6.74×10^{-4}	5.47×10^{-4}
Relative error	0.0053%	0.0049%	0.0136%	0.0046%	0.0007%	0.0003%
Time Elapsed	50.8	51.4	53.4	61.4	75.8	109.1

References

[1] Jiequn Han and Jihao Long. Convergence of the deep bsde method for coupled fbsdes. *Probability, Uncertainty and Quantitative Risk*, 5(1):1–33, 2020.

[2] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

Acknowledgment

This work was supported by grant NSA REU H98230-21-1-0336.