Revisiting Deep BSDE-Based PDE Solvers: A Reproducibility and Robustness Study

Anonymous authors
Paper under double-blind review

Abstract

We revisit deep BSDE-based methods for solving high-dimensional semilinear parabolic partial differential equations through a reproducibility and robustness study of the original Deep BSDE solver and its backward dynamic programming variants. We present a modular JAX-based implementation framework for stochastic differential equation simulation, BSDE discretization, and neural PDE solvers, designed for reproducible experimentation in the modern scientific machine learning ecosystem. Our study reproduces key benchmark problems including Allen-Cahn, Hamilton–Jacobi–Bellman, and nonlinear reaction–diffusion equations, while systematically analyzing optimization stability, discretization effects, gradient parameterizations, random seed sensitivity, and architectural choices across multiple solver variants. In addition to reproducing published results, we investigate failure modes such as divergence and representational limitations in high-dimensional settings. We further compare global forward-shooting approaches against backward dynamic programming formulations and discuss their tradeoffs in stability, scalability, and computational efficiency. Our work aims to provide both a reproducible experimental baseline and an extensible open-source foundation for future research on BSDE-based PDE solvers in JAX.

1 Introduction

High-dimensional partial differential equations (PDEs) arise in applications such as stochastic control, quantitative finance, uncertainty quantification, and engineering design. Classical numerical methods, including finite difference and finite element schemes, suffer from the curse of dimensionality and become impractical in high dimensions Bellman (1957). A widely adopted alternative is to reformulate PDEs as Backward Stochastic Differential Equations (BSDEs), enabling solution through stochastic simulation rather than grid-based discretization Pardoux & Peng (1992); Pardoux & Tang (1999).

The BSDE formulation has led to a class of neural PDE solvers capable of handling high-dimensional problems. The Deep BSDE method introduced the use of neural networks to approximate the value and gradient processes of the underlying stochastic system E et al. (2017); Han et al. (2018), while subsequent methods have sought to improve numerical stability and approximation accuracy.

Among these approaches, Deep Backward Dynamic Programming (DBDP) methods Huré et al. (2020) have become prominent BSDE solvers. They discretize the backward dynamics and solve a sequence of regression problems backward in time to approximate the solution process Y_t and its associated control process Z_t. DBDP1 parameterizes both Y_t and Z_t using separate neural networks, whereas DBDP2 represents only Y_t and obtains Z_t through automatic differentiation. Despite their widespread use, their reproducibility and practical behavior have received relatively little systematic study.

Most existing evaluations focus primarily on the initial solution estimate and terminal prediction error, which do not fully capture the quality of the backward recursion. Consequently, methods with similar terminal accuracy may exhibit substantially different trajectory-level behaviour. Numerous extensions have since been proposed, including second-order BSDE methods, backward dynamic programming formulations, local loss formulations, and operator-learning approaches (Beck et al. (2021); Raissi et al. (2019); Fujii et al. (2019))

In this work, we reproduce Deep BSDE, DBDP1, and DBDP2 under a unified implementation framework on four benchmark problems: the Allen-Cahn equation, Burgers' equation, a Hamilton-Jacobi-Bellman

(HJB) equation, and a nonlinear Black-Scholes equation with different borrowing and lending rates. Beyond conventional terminal metrics, we evaluate one-step and rollout errors to characterise trajectory-level accuracy and compare the structural properties of the three algorithms.

Our experiments show that trajectory-level evaluation reveals differences that are not apparent from terminal metrics alone. In particular, DBDP2 consistently exhibits greater stability and lower error accumulation across several benchmark problems, highlighting the importance of evaluating neural BSDE solvers beyond terminal accuracy.

Our main contributions are summarised as follows:

- We reproduce three representative neural BSDE solvers, the Deep BSDE method, DBDP1, and DBDP2, on four benchmark problems: Allen-Cahn, Burgers, Hamilton-Jacobi-Bellman (HJB), and a nonlinear Black-Scholes equation with different borrowing and lending rates.
- We provide a unified experimental framework for comparing BSDE and DBDP methods under identical benchmark settings, discretisations, and evaluation procedures.
- We extend the original evaluations by analysing terminal, one-step, and rollout errors, enabling a trajectory-level assessment of solution quality and error propagation.
- We provide a comparison of BSDE, DBDP1, and DBDP2, highlighting the trade-offs between terminal accuracy, trajectory consistency, and long-horizon stability.

2 Scope of Reproducibility

The objective of this work is to reproduce and evaluate three neural BSDE solvers for high-dimensional semilinear PDEs: the Deep BSDE method of E, Han, and Jentzen, together with the Deep Backwards Dynamic Programming variants DBDP1 and DBDP2 proposed by Huré et al.

We conduct our study on four benchmark equations that are commonly used in the BSDE literature: the Allen-Cahn equation, Burgers' equation, a Hamilton-Jacobi-Bellman (HJB) equation, and a nonlinear Black-Scholes equation with different borrowing and lending rates E et al. (2017); Huré et al. (2020). These problems cover a range of nonlinearities, terminal conditions, and solution characteristics, providing a diverse test bed for evaluating neural BSDE solvers.

Our reproduction study focuses on the following questions:

1. Can the Deep BSDE methods reproduce the solution accuracies reported in the original works across standard benchmark equations?
2. How do the three methods compare when evaluated under a common implementation framework and identical experimental settings?
3. Do trajectory-level metrics reveal differences in solver behaviour that are not captured by terminal prediction errors alone?

Beyond reproducing the original experiments, we provide a unified implementation and numerical evaluation of Deep BSDE, DBDP1, and DBDP2. Using this framework, we perform a systematic comparison of the three methods and extend the evaluation with trajectory-level diagnostics, including one-step and rollout errors, to study error propagation throughout the backward recursion.

3 Background

3.1 Backward Stochastic Differential Equations

We consider a class of semilinear parabolic partial differential equations (PDEs) of the form

$$\partial_t u(t,x) + \mathcal{L}u(t,x) + f\big(t, x, u(t,x), \sigma(t,x)^\top \nabla_x u(t,x)\big) = 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \tag{1}$$

with terminal condition

$$u(T,x) = g(x), \qquad x \in \mathbb{R}^d. \tag{2}$$

The differential operator $\mathcal{L}$ is defined as

$$\mathcal{L}u = \frac{1}{2}\operatorname{Tr}\left(\sigma\sigma^\top \nabla_x^2 u\right) + \mu^\top \nabla_x u, \tag{3}$$

where $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ denote the drift and diffusion coefficients, respectively.

Equation 1 admits a probabilistic representation through the nonlinear Feynman-Kac formula (Pardoux & Peng (1992); Pardoux & Tang (1999)). Consider the forward diffusion process

$$dX_t = \mu(t, X_t)\, dt + \sigma(t, X_t)\, dW_t, \qquad X_0 = x_0, \tag{4}$$

where W_t is a d-dimensional Brownian motion.

The associated backward stochastic differential equation (BSDE) is

$$Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s)\, ds - \int_t^T Z_s^\top\, dW_s, \qquad 0 \le t \le T. \tag{5}$$

Under suitable regularity assumptions, the solution of the PDE and the solution of the BSDE satisfy

$$Y_t = u(t, X_t), \qquad Z_t = \sigma(t, X_t)^\top \nabla_x u(t, X_t). \tag{6}$$

Consequently, solving the PDE can be reformulated as solving the coupled forward-backward stochastic system defined by equation 4 and equation 5. The relationship in Equation 6 plays a central role in neural BSDE solvers and will be particularly important when discussing DBDP1 and DBDP2.

3.2 Deep BSDE Solver

The Deep BSDE method reformulates the solution of high-dimensional PDEs as a learning problem. Let

$$0 = t_0 < t_1 < \cdots < t_N = T$$

be a time discretization of the interval $[0,T]$. The forward diffusion process is approximated using the Euler-Maruyama scheme Kloeden & Platen (1992)

$$X_{t_{i+1}} = X_{t_i} + \mu(t_i, X_{t_i})\Delta t_i + \sigma(t_i, X_{t_i})\Delta W_i, \tag{7}$$

where

$$\Delta t_i = t_{i+1} - t_i, \qquad \Delta W_i = W_{t_{i+1}} - W_{t_i}.$$

The BSDE is discretised as

$$Y_{t_{i+1}} = Y_{t_i} - f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i + Z_{t_i}^\top \Delta W_i. \tag{8}$$

The key idea is to parameterise the unknown gradient process using neural networks, $Z_{t_i} \approx \mathcal{Z}_i(X_{t_i}; \theta_i)$ while treating the initial value $Y_0 \approx u(0, x_0)$ as a trainable parameter.

Given sampled Brownian trajectories and forward diffusion paths, the discretized BSDE dynamics are propagated from t_0 to t_N. The network parameters are optimized by minimizing the terminal matching objective $\mathcal{L}(\theta) = \mathbb{E}\left[\left|Y_T^\theta - g(X_T)\right|^2\right]$

Rather than constructing a spatial discretisation of the PDE, the Deep BSDE method learns stochastic trajectories whose dynamics satisfy the BSDE representation while matching the prescribed terminal condition at maturity.

3.3 Deep Backward Dynamic Programming

While Deep BSDE methods optimize a global terminal loss, Deep Backward Dynamic Programming (DBDP) methods solve the BSDE through a sequence of local regression problems performed backward in time. Starting from the terminal condition $u_N(x) = g(x)$

DBDP constructs the approximations backward in time. Assume that u_{i+1} has already been computed. Then the next-time target is $\widehat{Y}_{i+1} = u_{i+1}(X_{i+1})$

At time t_i, DBDP seeks functions u_i and z_i such that the discretized BSDE relation

$$\widehat{Y}_{i+1} \approx u_i(X_i) - f(t_i, X_i, u_i(X_i), z_i(X_i))\Delta t_i + z_i(X_i)^\top \Delta W_i$$

is satisfied in the least-squares sense. The generic local regression problem is therefore

$$(u_i, z_i) = \arg\min_{u,z} \mathbb{E}\left[\left|u(X_i) - f(t_i, X_i, u(X_i), z(X_i))\Delta t_i + z(X_i)^\top \Delta W_i - u_{i+1}(X_{i+1})\right|^2\right].$$

Repeating this procedure for $i = N-1, \ldots, 0$ produces the sequence $u_{N-1}, \ldots, u_0$.

3.3.1 DBDP1

DBDP1 parameterizes the value and gradient processes using two independent neural networks,

$u_i(x) \approx U_i(x;\theta_i), z_i(x) \approx Z_i(x;\phi_i)$. Substituting these approximations into the backward Euler recursion, the parameters are obtained by minimising $\mathcal{L}_i(\theta_i, \phi_i) = \mathbb{E}\Big[|U_{i+1}(X_{i+1};\theta_{i+1}) - F_i(X_i, \Delta W_i; \theta_i, \phi_i)|^2\Big]$, where $F_i = U_i(X_i;\theta_i) - f(t_i, X_i, U_i, Z_i)\Delta t_i + Z_i(X_i;\phi_i)^\top \Delta W_i$. (Ma & Yong (1999)) Since both U_i and Z_i are independently parameterized, the value and gradient processes are learned separately at each time step.

3.3.2 DBDP2

DBDP2 removes the separate gradient network and represents only the value process, $u_i(x) \approx U_i(x;\theta_i)$ The gradient process is recovered through automatic differentiation, $z_i(x) = \sigma(t_i, x)^\top \nabla_x U_i(x;\theta_i)$ which follows from the nonlinear Feynman-Kac representation. (Gobet et al. (2005)) The resulting loss is identical to that of DBDP1, except that the gradient network is replaced by $\sigma^\top \nabla_x U_i$. Consequently, DBDP2 optimizes only the parameters θ_i, enforcing consistency between the value and gradient approximations while reducing the number of trainable networks.

4 Experimental Setup

4.1 Benchmark Problem

We evaluate Deep BSDE, DBDP1, and DBDP2 on four benchmark equations commonly used in the neural BSDE literature. These problems span a range of nonlinearities and solution structures, including reaction-diffusion equations, stochastic control problems, nonlinear pricing equations, and nonlinear conservation laws. For all experiments, we consider the semilinear PDE (Evans (2010)) as described in equation 1

The specific benchmark problems are described below.

4.1.1 Allen-Cahn Equation

The Allen-Cahn benchmark is defined by $\frac{\partial u}{\partial t}(t,x) + \Delta u(t,x) + u(t,x) - u(t,x)^3 = 0$ with terminal condition $g(x) = \left(2 + \frac{2}{5}|x|_2^2\right)^{-1}$ The corresponding BSDE driver is $f(t,x,y,z) = y - y^3$. Throughout the reported experiments, we use dimension $d = 100$ and time horizon $T = 0.3$.

4.1.2 Hamilton-Jacobi-Bellman Equation

The HJB benchmark is given by $\frac{\partial u}{\partial t}(t,x) + \Delta u(t,x) - |\nabla_x u(t,x)|_2^2 = 0$ with terminal condition $g(x) = \log\left(\frac{1+|x|_2^2}{2}\right)$. The associated BSDE driver is $f(t,x,y,z) = -|z|_2^2$ The reported HJB runs use dimension $d = 100$. The terminal time is reported as run in Section 4.2, since the BSDE and DBDP1 runs use $T = 1.0$ whereas the DBDP2 runs use $T = 0.3$.

4.1.3 Nonlinear Black-Scholes Equation

The nonlinear Black-Scholes benchmark models option pricing under different borrowing and lending rates. The terminal payoff is $g(x) = \max\left(\max_{1\le i\le d} x_i - 120, 0\right) - 2\max\left(\max_{1\le i\le d} x_i - 150, 0\right)$. The BSDE driver is $f(t,x,y,z) = -R^l y - \frac{\mu - R^l}{\sigma}\sum_{i=1}^d z_i + (R^b - R^l)\max\left(0, \frac{1}{\sigma}\sum_{i=1}^d z_i - y\right)$

where R^l and R^b denote the lending and borrowing rates, respectively. The corresponding PDE takes the form $\frac{\partial u}{\partial t} + \frac{\sigma^2}{2}\sum_{i=1}^d x_i^2 \frac{\partial^2 u}{\partial x_i^2} + \max\left\{R^b\left(\sum_{i=1}^d x_i \frac{\partial u}{\partial x_i} - u\right), R^l\left(\sum_{i=1}^d x_i \frac{\partial u}{\partial x_i} - u\right)\right\} = 0.$

Following the original benchmark, we use $d = 100$ and $T = 0.5$.

4.1.4 Burgers Equation

The multidimensional Burgers benchmark is defined by $\frac{\partial u}{\partial t}(t,x) + \frac{d^2}{2}\Delta u(t,x) + \left(u(t,x) - \frac{2+d}{2d}\right)\left(d\sum_{i=1}^d \frac{\partial u}{\partial x_i}(t,x)\right) = 0$ with terminal condition $g(x) = \frac{\exp\left(T + \frac{1}{d}\sum_{i=1}^d x_i\right)}{1+\exp\left(T+\frac{1}{d}\sum_{i=1}^d x_i\right)}$. The associated BSDE driver is $f(t,x,y,z) = \left(y - \frac{2+d}{2d}\right)\left(\sum_{i=1}^d z_i\right)$.

The reported Burgers experiments use two paper configurations: Exp. 1 with $d = 20$ and $T = 1.0$, and Exp. 2 with $d = 50$ and $T = 0.2$.

4.2 Training Configuration

All experiments were run as independent Monte Carlo training trials across five random seeds, $\{1,2,3,4,5\}$. For each solver and benchmark problem, we varied the number of time-discretization steps while keeping the architecture, optimiser (Kingma & Ba (2017)), clipping thresholds, and batch size fixed within that configuration. The results reported in the following sections are the mean and standard deviation over the five successful seed runs for each solver-problem-discretization setting.

Table 1: Training configurations used for the reported five-seed experiments. For DBDP1 and DBDP2, the update count denotes inner optimization steps per backward time index.

Problem	Solver	d	T	Time steps N	Batch size	Optimizer and updates
Allen-Cahn	BSDE	100	0.3	$\{20,50,100,250,500\}$	64	Adam 5×10^{-4}, 4000 steps
Allen-Cahn	DBDP1/DBDP2	100	0.3	$\{20,50,100,250,500\}$	64	Adam 5×10^{-4}, 400 inner steps
Black-Scholes	BSDE	100	0.5	$\{20,50,100,250,500\}$	64	Adam 5×10^{-3}, 4000 steps
Black-Scholes	DBDP1/DBDP2	100	0.5	$\{20,50,100,250,500\}$	64	Adam 5×10^{-3}, 400 inner steps
HJB	BSDE	100	1.0	$\{20,50,100,250,500\}$	64	Adam 10^{-2}, 2000 steps
HJB	DBDP1/DBDP2	100	1.0	$\{20,50,100,250,500\}$	64	Adam 5×10^{-4}, 400 inner steps
Burgers Exp. 1	BSDE/DBDP1/DBDP2	20	1.0	$\{80,160,320\}$	256	Optimizer schedule γ_m: $10^{-2}\to10^{-3}\to10^{-4}$ at $m = 30000, 50000$; 60000 updates
Burgers Exp. 2	BSDE/DBDP1/DBDP2	50	0.2	$\{30,60,120,240\}$	256	Optimizer schedule γ_m: $10^{-2}\to10^{-3}\to10^{-4}$ at $m = 15000, 25000$; 30000 updates

The neural-network widths were tied to the problem dimension. The Deep BSDE solver used a shared feed-forward network with hidden widths $(d+10, d+10)$, ReLU hidden activations, BatchNorm and input normalization enabled, and separate output heads for Y and Z. DBDP1 used two per-time-step networks, one for the value approximation and one for the Z approximation, both with hidden widths $(d+10, d+10)$ and tanh activations. DBDP2 used the same value-network architecture as DBDP1, but computed $Z = \sigma^\top \nabla_x U$ by automatic differentiation rather than fitting a separate Z network. Thus, for the non-Burgers experiments where $d = 100$, all hidden widths were $(110, 110)$.

All methods used gradient clipping with threshold 1.0 and loss clipping with threshold 50.0. DBDP1 and DBDP2 used warm starts across adjacent backward time indices, so the network at one time index was initialized from the next later time index when available. Training progress was recorded every 100 optimization steps. Held-out diagnostics were computed using batches of size $\max(\text{batch size}, 4\times\text{batch size})$, giving 256 validation samples for the non-Burgers experiments and 1024 validation samples for Burgers. For the Allen-Cahn Deep BSDE runs, we additionally clipped simulated states at magnitude 10 to improve numerical stability; this clipping was not used for the DBDP runs.

4.3 Evaluation Metrics

We evaluate the performance of Deep BSDE, DBDP1, and DBDP2 using a collection of metrics designed to assess both terminal accuracy and trajectory-level behavior.

Initial Value Estimate. For each method, we report the estimated value of the solution at the initial state, $\hat{u}_0 = \hat{u}(0, x_0)$. which corresponds to the quantity typically reported in the original Deep BSDE and DBDP studies. When an analytical reference value is available, the estimate can be directly compared against the true solution.

Terminal Mean Squared Error. The primary training objective for BSDE-based solvers is the terminal matching condition. We therefore evaluate the terminal mean squared error (MSE), $\mathrm{MSE}_{\mathrm{terminal}} = \mathbb{E}\left[|Y_T - g(X_T)|^2\right]$ where $g(X_T)$ denotes the prescribed terminal condition and Y_T is the terminal value produced by the learned solver.

One-Step Error. To evaluate how accurately the learned solution satisfies the discretized BSDE recursion, we compute the average one-step prediction error $\mathcal{E}_{\mathrm{step}} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbb{E}\left[\left|Y_{i+1} - \left(Y_i - f_i\Delta t + Z_i^\top \Delta W_i\right)\right|^2\right]$, where $f_i = f(t_i, X_i, Y_i, Z_i)$. This metric measures the local consistency of the learned approximation with the discretized BSDE dynamics. Since the Bellman residual used in DBDP methods is equivalent to the one-step prediction error in our implementation, we report only the latter.

Rollout Error. While one-step errors quantify local consistency, small local errors may accumulate over long horizons. To measure this effect, we compute the rollout error by recursively propagating the learned solution through the entire time horizon and evaluating the discrepancy at the terminal time: $\mathcal{E}_{\mathrm{rollout}} = \mathbb{E}\left[\left|\widetilde{Y}_T - g(X_T)\right|^2\right]$, where $\widetilde{Y}_T$ is obtained by repeatedly applying the learned BSDE dynamics from t_0 to t_N without using intermediate supervision. The rollout error therefore captures the cumulative effect of local approximation errors and provides a measure of long-horizon stability.

For each experiment, all reported metrics are averaged across multiple independent training runs, and we report the mean together with the corresponding standard deviation.

5 Reproduction Results

This section reports the direct reproduction runs used to check whether the implemented solvers recover the behavior of the original neural BSDE benchmarks. We first reproduce the Deep BSDE baseline on Allen-Cahn, HJB, and Black-Scholes using the $N = 20$ time-discretization setting reported in the original Deep BSDE paper (tables2). For each benchmark, we train five independent runs with seeds $\{1, 2, 3, 4, 5\}$ and summarize the final checkpoint by the across-seed mean and standard deviation. The optimizer step count follows the final reported paper row for each problem: 4000 optimization steps for Allen-Cahn and Black-Scholes, and 2000 optimization steps for HJB. We compare the reproduced initial value estimate and terminal MSE against the original paper's reported initial value, terminal loss, and relative L^1 error.

We then run the corresponding DBDP1 and DBDP2 reproduction experiments on the same three benchmark PDEs, again using $N = 20$ time steps and five independent seeds. Unlike the Deep BSDE method, which trains a forward trajectory parameterization against the terminal condition, the DBDP methods train a sequence of neural networks backward in time. At each time index i, the method fits a neural approximation for the value at that time step using the already-trained approximation at the next time step as the target in the one-step BSDE recursion. Each per-time-step network is optimized for 400 inner gradient steps before moving to the next earlier time index. DBDP1 learns separate neural approximations for the value and gradient/control components at each time step, while DBDP2 learns the value approximation and obtains the gradient/control component by differentiating the value network. The per-time-step plots below show the final inner prediction loss after this 400-step optimization procedure, with individual seed curves and the across-seed mean and standard-deviation band.

6 Comparative Analysis

While the reproduction experiments verify that the implementations recover the behavior reported in the original benchmarks, they do not by themselves explain how the three solvers differ in practice. We therefore compare Deep BSDE, DBDP1, and DBDP2 using the common evaluation metrics reported in

Problem	Time steps N	Optimizer steps m	Paper U^{Θ_m}	Our u_0 estimate	Paper loss	Our terminal MSE
Allen-Cahn	20	4000	0.0528 ± 0.0002	0.050191 ± 0.003852	$2.0 \times 10^{-4} \pm 1.0 \times 10^{-5}$	0.027317 ± 0.002853
HJB	20	2000	4.5977 ± 0.0019	4.49218 ± 0.089186	0.0231 ± 0.0026	0.094155 ± 0.037525
Black-Scholes	20	4000	21.217 ± 0.043	21.193 ± 0.20919	29.029 ± 3.236	39.093 ± 22.511

Table 2: Deep BSDE reproduction comparison against the original paper tables for the $N = 20$ time-discretization setting. Our results are five-seed means and standard deviations from the final optimization checkpoint; the original paper values are taken from the final row of each corresponding benchmark table.

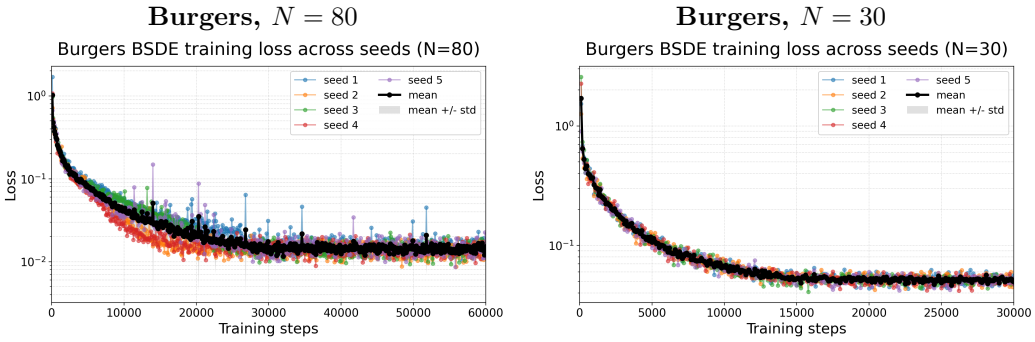

Figure 1: Training loss curves for the five independent Burgers Deep BSDE reproduction runs. The $N = 80$ configuration uses the first Burgers paper setup with $d = 20$ and $T = 1.0$ and is trained for 60000 optimizer steps, while the $N = 30$ configuration uses the second Burgers paper setup with $d = 50$ and $T = 0.2$ and is trained for 30000 optimizer steps. Each plot shows the per-seed loss recorded every 100 optimizer steps together with the across-seed mean and standard-deviation band.

Tables 3,4,5,6. Beyond the initial value estimate, we focus on the one-step error and rollout terminal MSE, which measure local BSDE consistency and accumulated trajectory error, respectively.

6.1 Terminal Accuracy and Rollout Stability

Across the benchmark problems, the initial value estimates are often broadly reasonable for all methods, but the rollout metrics reveal substantial differences in trajectory stability. For Allen-Cahn, the Deep BSDE solver produces rollout terminal MSE values between approximately 87 and 99, whereas DBDP2 remains near 10^{-3} across all discretizations. A similar pattern appears in Black-Scholes, where Deep BSDE rollout errors increase from roughly 441 to 3896, while DBDP2 remains between approximately 30 and 32. These results indicate that accurate terminal or initial-value estimates alone can obscure significant accumulated trajectory errors.

6.2 Effect of Time Discretisation

Increasing the number of time steps generally improves the local consistency of the DBDP methods. This trend is particularly evident in the one-step errors, which decrease as the temporal grid is refined (Kloeden & Platen (1992)). For example, in the Allen-Cahn problem, DBDP1 improves from approximately 5.5×10^{-3} at $N = 20$ to 3.3×10^{-5} at $N = 500$, while DBDP2 improves from approximately 1.9×10^{-4} to 1.1×10^{-5}. In contrast, Deep BSDE does not exhibit the same consistent improvement in rollout behavior under finer discretizations, suggesting that reducing the discretization error alone is insufficient when the learned trajectory is constrained only through a terminal objective.

6.3 Comparison Between DBDP1 and DBDP2

DBDP2 is consistently the most stable method according to the trajectory-level metrics. Its one-step errors are typically smaller than those of DBDP1, and its rollout terminal MSE is frequently lower by one or more orders of magnitude. This difference is particularly pronounced in Burgers Exp. 1, where DBDP1 exhibits unstable u_0 estimates and large variance at coarser discretizations, whereas DBDP2 remains stable and achieves rollout errors on the order of 10^{-3}. These observations support the interpretation that recovering Z_t through automatic differentiation of the value network provides a more consistent coupling between the value and gradient processes.

Table 3: Allen Cahn Common Metrics

PDE	Solver	Steps	u_0 Estimate	One-step Avg.	Rollout Terminal MSE
Allen Cahn	DBDP_1	20	0.058174 ± 0.01075	$0.0055323 \pm 2.901 \times 10^{-4}$	0.099436 ± 0.0053677
Allen Cahn	DBDP_2	20	0.055871 ± 0.0076294	$1.867 \times 10^{-4} \pm 4.071 \times 10^{-5}$	$0.0030945 \pm 8.834 \times 10^{-4}$
Allen Cahn	BSDE	20	0.050191 ± 0.0038521	1.4364 ± 1.1572	87.432 ± 19.196
Allen Cahn	DBDP_1	50	0.060976 ± 0.010478	$0.0012966 \pm 9.02 \times 10^{-5}$	0.056598 ± 0.0043898
Allen Cahn	DBDP_2	50	0.05238 ± 0.006687	$8.741 \times 10^{-5} \pm 1.965 \times 10^{-5}$	$0.0031466 \pm 9.48 \times 10^{-4}$
Allen Cahn	BSDE	50	0.043757 ± 0.0043002	0.95269 ± 0.46487	95.341 ± 7.8571
Allen Cahn	DBDP_1	100	0.054948 ± 0.023405	$3.849 \times 10^{-4} \pm 2.181 \times 10^{-5}$	0.031089 ± 0.0027487
Allen Cahn	DBDP_2	100	0.049856 ± 0.0062359	$4.626 \times 10^{-5} \pm 8.385 \times 10^{-6}$	$0.0026345 \pm 4.989 \times 10^{-4}$
Allen Cahn	BSDE	100	0.038295 ± 0.0048068	1.1582 ± 1.5925	95.871 ± 4.844
Allen Cahn	DBDP_1	250	0.048903 ± 0.015798	$8.977 \times 10^{-5} \pm 9.513 \times 10^{-6}$	0.017061 ± 0.0021653
Allen Cahn	DBDP_2	250	0.051221 ± 0.0052555	$2.025 \times 10^{-5} \pm 3.684 \times 10^{-6}$	$0.0023542 \pm 3.648 \times 10^{-4}$
Allen Cahn	BSDE	250	0.035644 ± 0.0060154	1.4543 ± 2.7081	97.425 ± 2.4705
Allen Cahn	DBDP_1	500	0.053107 ± 0.0069564	$3.293 \times 10^{-5} \pm 2.888 \times 10^{-6}$	0.011483 ± 0.0018879
Allen Cahn	DBDP_2	500	0.049527 ± 0.0057013	$1.118 \times 10^{-5} \pm 1.548 \times 10^{-6}$	0.0033329 ± 0.0015583
Allen Cahn	BSDE	500	0.03248 ± 0.010245	7.6494 ± 13.677	99.159 ± 0.27564

Table 4: Black Scholes Common Metrics

PDE	Solver	Steps	u_0 Estimate	One-step Avg.	Rollout Terminal MSE
Black Scholes	DBDP_1	20	21.495 ± 0.51702	1.9903 ± 0.16681	40.674 ± 4.3321
Black Scholes	DBDP_2	20	21.699 ± 0.47394	1.6075 ± 0.1362	32.09 ± 2.822
Black Scholes	BSDE	20	21.193 ± 0.20919	10.684 ± 3.6217	441.32 ± 252.07
Black Scholes	DBDP_1	50	21.438 ± 0.79736	0.76029 ± 0.09772	40.355 ± 6.988
Black Scholes	DBDP_2	50	21.489 ± 0.6181	0.62824 ± 0.071394	31.375 ± 3.5524
Black Scholes	BSDE	50	21.039 ± 0.40954	3.0854 ± 0.93881	795.09 ± 544.19
Black Scholes	DBDP_1	100	21.581 ± 0.8334	0.38422 ± 0.02473	38.434 ± 2.4942
Black Scholes	DBDP_2	100	21.308 ± 0.65574	0.30136 ± 0.017312	30.239 ± 1.647
Black Scholes	BSDE	100	20.881 ± 0.29161	0.8713 ± 0.51404	924.57 ± 1051.8
Black Scholes	DBDP_1	250	22.102 ± 0.46958	0.14698 ± 0.019015	35.822 ± 5.8788
Black Scholes	DBDP_2	250	21.045 ± 0.36827	0.12081 ± 0.0182	30.103 ± 4.4949
Black Scholes	BSDE	250	21.255 ± 0.28296	0.37021 ± 0.22071	3327 ± 3494.3
Black Scholes	DBDP_1	500	20.833 ± 0.82911	0.078075 ± 0.0078185	38.13 ± 4.8275
Black Scholes	DBDP_2	500	21.139 ± 0.64225	0.065434 ± 0.0074353	31.954 ± 4.0142
Black Scholes	BSDE	500	21.318 ± 0.16132	0.40684 ± 0.12648	3895.6 ± 4096.3

Table 5: HJB Common Metrics

PDE	Solver	Steps	u_0 Estimate	One-step Avg.	Rollout Terminal MSE
HJB	DBDP_1	20	1.2276 ± 0.064547	0.18326 ± 0.014112	3.4356 ± 0.29172
HJB	DBDP_2	20	3.7663 ± 0.059949	0.045545 ± 0.003347	0.74287 ± 0.027241
HJB	BSDE	20	4.4922 ± 0.089186	61.347 ± 46.657	9681.8 ± 8083.5
HJB	DBDP_1	50	2.4722 ± 0.078954	0.053035 ± 0.0054841	2.2909 ± 0.30361
HJB	DBDP_2	50	3.790 ± 0.033413	0.017267 ± 0.001950	0.74554 ± 0.056602
HJB	BSDE	50	4.5926 ± 0.048113	2.6051 ± 1.1242	1929.5 ± 1136.4
HJB	DBDP_1	100	3.2427 ± 0.11295	$0.019004 \pm 4.73 \times 10^{-4}$	1.1746 ± 0.16758
HJB	DBDP_2	100	3.8922 ± 0.105164	$0.0090685 \pm 6.028 \times 10^{-4}$	0.65410 ± 0.063662
HJB	BSDE	100	4.5563 ± 0.050345	7.0768 ± 9.4367	$2.039 \times 10^{4} \pm 2.843 \times 10^{4}$
HJB	DBDP_1	250	3.7709 ± 0.091392	$0.0056896 \pm 6.054 \times 10^{-4}$	0.69079 ± 0.095008
HJB	DBDP_2	250	3.8695 ± 0.059737	$3.637 \times 10^{-3} \pm 3.513 \times 10^{-4}$	0.52041 ± 0.071060
HJB	BSDE	250	4.6045 ± 0.010951	0.65952 ± 0.52647	8428.9 ± 8787.3
HJB	DBDP_1	500	3.9018 ± 0.12247	$0.0026923 \pm 1.083 \times 10^{-4}$	0.55717 ± 0.085542
HJB	DBDP_2	500	3.9021 ± 0.053362	$1.979 \times 10^{-3} \pm 1.863 \times 10^{-4}$	0.53005 ± 0.084848
HJB	BSDE	500	4.5467 ± 0.070437	0.42283 ± 0.32019	$1.971 \times 10^{4} \pm 1.743 \times 10^{4}$

Table 6: Burgers Common Metrics

PDE	Solver	Steps	Experiment	u_0 Estimate	One-step Avg.	Rollout Terminal MSE
Burgers	BSDE	30	Exp. 2	$0.5285 \pm 4.676 \times 10^{-4}$	0.56862 ± 0.46718	205.22 ± 431.84
Burgers	DBDP_1	30	Exp. 2	0.51151 ± 0.020858	0.0054896 ± 0.001072	0.048276 ± 0.0027059
Burgers	DBDP_2	30	Exp. 2	0.51408 ± 0.0090346	$6.196 \times 10^{-4} \pm 2.984 \times 10^{-4}$	0.015613 ± 0.0096248
Burgers	BSDE	60	Exp. 2	$0.52844 \pm 6.94 \times 10^{-4}$	0.19775 ± 0.10217	69.737 ± 143.16
Burgers	DBDP_1	60	Exp. 2	0.50999 ± 0.015486	$0.0027216 \pm 5.454 \times 10^{-4}$	0.055667 ± 0.0036854
Burgers	DBDP_2	60	Exp. 2	0.51581 ± 0.0092212	$3.039 \times 10^{-4} \pm 1.432 \times 10^{-4}$	0.015572 ± 0.009169
Burgers	BSDE	80	Exp. 1	0.62572 ± 0.003605	271.83 ± 337.96	$6.978 \times 10^{10} \pm 1.56 \times 10^{11}$
Burgers	DBDP_1	80	Exp. 1	54.351 ± 78.603	57.471 ± 124.04	5.048 ± 6.7943
Burgers	DBDP_2	80	Exp. 1	0.46114 ± 0.0074689	$9.77 \times 10^{-5} \pm 1.082 \times 10^{-6}$	$0.0026209 \pm 1.02 \times 10^{-4}$
Burgers	BSDE	120	Exp. 2	$0.52857 \pm 9.997 \times 10^{-4}$	0.090765 ± 0.035663	32.749 ± 65.884
Burgers	DBDP_1	120	Exp. 2	0.51002 ± 0.011274	$0.0012855 \pm 2.065 \times 10^{-4}$	0.059641 ± 0.0034574
Burgers	DBDP_2	120	Exp. 2	0.51612 ± 0.0087602	$1.319 \times 10^{-4} \pm 5.74 \times 10^{-5}$	0.016073 ± 0.009232
Burgers	BSDE	160	Exp. 1	0.62594 ± 0.0024873	58.746 ± 86.633	$3.438 \times 10^{12} \pm 7.685 \times 10^{12}$
Burgers	DBDP_1	160	Exp. 1	-2.22 ± 0.91316	0.033739 ± 0.04102	0.15886 ± 0.22643
Burgers	DBDP_2	160	Exp. 1	0.47254 ± 0.0059389	$2.781 \times 10^{-5} \pm 1.763 \times 10^{-6}$	$0.0015401 \pm 2.122 \times 10^{-4}$
Burgers	BSDE	240	Exp. 2	$0.5277 \pm 9.245 \times 10^{-4}$	0.056343 ± 0.020442	14.905 ± 26.767
Burgers	DBDP_1	240	Exp. 2	0.51151 ± 0.0086772	$6.316 \times 10^{-4} \pm 7.736 \times 10^{-5}$	0.062689 ± 0.0028585
Burgers	DBDP_2	240	Exp. 2	0.51668 ± 0.0079646	$6.162 \times 10^{-5} \pm 2.403 \times 10^{-5}$	0.01775 ± 0.0097117
Burgers	BSDE	320	Exp. 1	0.62609 ± 0.0025706	31.317 ± 46.391	$6.798 \times 10^{25} \pm 1.52 \times 10^{26}$
Burgers	DBDP_1	320	Exp. 1	-2.0375 ± 0.71321	0.0020913 ± 0.0018197	0.041141 ± 0.0086853
Burgers	DBDP_2	320	Exp. 1	0.48691 ± 0.015133	$9.514 \times 10^{-6} \pm 4.606 \times 10^{-6}$	0.0012475 ± 0.0010951

6.4 Benchmark-Specific Observations

The Burgers experiments highlight the importance of trajectory-level evaluation most clearly. In Exp. 2, Deep BSDE obtains stable-looking u_0 estimates near 0.528, but its rollout terminal MSE remains substantially larger than that of either DBDP variant. In Exp. 1, the Deep BSDE rollout error becomes extremely large as the number of time steps increases, whereas DBDP2 remains stable. The HJB results also show a clear separation between Deep BSDE and the DBDP methods in terms of rollout behavior. However, these comparisons should be interpreted with some caution because the reported DBDP2 HJB experiments were conducted using a shorter time horizon than the corresponding Deep BSDE and DBDP1 experiments.

Overall, the comparative results show that the DBDP methods, and particularly DBDP2, provide stronger trajectory-level consistency than Deep BSDE. The primary empirical distinction lies not only in the final solution estimate but also in the quality of the learned dynamics throughout the entire time horizon. These findings further motivate the use of one-step and rollout diagnostics as essential evaluation metrics for neural BSDE solvers.

7 Theoretical Analysis

7.1 Constraint Structure of BSDE and DBDP Training

Although Deep BSDE and DBDP are derived from the same discretized BSDE, they optimize fundamentally different objectives.

Consider the Euler discretization $Y_{n+1} = Y_n - f(t_n, X_n, Y_n, Z_n)\Delta t + Z_n^\top \Delta W_n$, with terminal condition $Y_N = g(X_N)$. Define the local residual $R_n = Y_{n+1} - Y_n + f(t_n, X_n, Y_n, Z_n)\Delta t - Z_n^\top \Delta W_n$. The Deep BSDE objective supervises only the terminal condition, $\mathcal{L}_{\text{BSDE}} = \mathbb{E}\left[|Y_N - g(X_N)|^2\right]$, whereas DBDP supervises every local transition, $\mathcal{L}_{\text{DBDP}} = \sum_{n=0}^{N-1} \mathbb{E}\left[|R_n|^2\right]$.

To formalise the distinction, define $\mathcal{C}_{\text{BSDE}} = (Y, Z) : Y_N = g(X_N)$, and $\mathcal{C}_{\text{DBDP}} = (Y, Z) : R_n = 0\ \forall n,\ Y_N = g(X_N)$.

Proposition 1. *The feasible set of DBDP is a strict subset of the feasible set of Deep BSDE:*

$$\mathcal{C}_{\text{DBDP}} \subsetneq \mathcal{C}_{\text{BSDE}}. \tag{9}$$

Proof. If $(Y, Z) \in \mathcal{C}_{\text{DBDP}}$, then all local BSDE recursions are satisfied exactly and the terminal condition holds. Hence $(Y, Z) \in \mathcal{C}_{\text{BSDE}}$.

Conversely, consider any trajectory satisfying $Y_N = g(X_N)$ but violating at least one local recursion, i.e., $R_k \neq 0$ for some k.

Then $(Y, Z) \in \mathcal{C}_{\text{BSDE}}$ but $(Y, Z) \notin \mathcal{C}_{\text{DBDP}}$. Therefore, $\mathcal{C}_{\text{DBDP}} \subsetneq \mathcal{C}_{\text{BSDE}}$, and the inclusion is strict. □

The proposition shows that DBDP imposes a strictly stronger set of constraints than Deep BSDE. While Deep BSDE supervises only the endpoint of the trajectory, DBDP supervises every local transition.

Proposition 2. *Assume that the BSDE driver is Lipschitz in (y, z), i.e.,*

$$|f(t, x, y_1, z_1) - f(t, x, y_2, z_2)| \leq L_y|y_1 - y_2| + L_z|z_1 - z_2|.$$

Let

$$e_n = Y_n - \widehat{Y}_n, \qquad \delta Z_n = Z_n - \widehat{Z}_n.$$

Then there exists a constant $C > 0$ such that

$$\mathbb{E}|e_{n+1}|^2 \leq (1 + C\Delta t)\mathbb{E}|e_n|^2 + C\Delta t\,\mathbb{E}|\delta Z_n|^2.$$

Consequently,

$$\mathbb{E}|e_N|^2 \leq e^{CT}\mathbb{E}|e_0|^2 + Ce^{CT}\sum_{k=0}^{N-1}\Delta t\,\mathbb{E}|\delta Z_k|^2.$$

Therefore, the terminal error depends only on an accumulated trajectory-level quantity and does not uniquely determine the individual local errors.

Proof. Consider the true and learned discrete BSDE recursions

$$Y_{n+1} = Y_n - f(t_n, X_n, Y_n, Z_n)\Delta t + Z_n^\top \Delta W_n,$$

and

$$\widehat{Y}_{n+1} = \widehat{Y}_n - f(t_n, X_n, \widehat{Y}_n, \widehat{Z}_n)\Delta t + \widehat{Z}_n^\top \Delta W_n.$$

Subtracting the two recursions gives

$$e_{n+1} = e_n - \Delta f_n \Delta t + \delta Z_n^\top \Delta W_n,$$

where

$$\Delta f_n = f(t_n, X_n, Y_n, Z_n) - f(t_n, X_n, \widehat{Y}_n, \widehat{Z}_n).$$

Taking squared norms and using the inequality

$$|a + b + c|^2 \leq (1 + \alpha)|a|^2 + C_\alpha|b + c|^2$$

for some $\alpha > 0$, we obtain

$$|e_{n+1}|^2 \leq (1 + \alpha)|e_n|^2 + C_\alpha\left|-\Delta f_n \Delta t + \delta Z_n^\top \Delta W_n\right|^2.$$

Using

$$|a + b|^2 \leq 2|a|^2 + 2|b|^2,$$

this gives

$$|e_{n+1}|^2 \leq (1 + \alpha)|e_n|^2 + C_\alpha \Delta t^2|\Delta f_n|^2 + C_\alpha|\delta Z_n^\top \Delta W_n|^2.$$

Taking expectations and using

$$\mathbb{E}\left[|\delta Z_n^\top \Delta W_n|^2\right] = \Delta t\,\mathbb{E}|\delta Z_n|^2,$$

we get

$$\mathbb{E}|e_{n+1}|^2 \leq (1 + \alpha)\mathbb{E}|e_n|^2 + C_\alpha \Delta t^2\mathbb{E}|\Delta f_n|^2 + C_\alpha \Delta t\,\mathbb{E}|\delta Z_n|^2.$$

By the Lipschitz condition,

$$|\Delta f_n|^2 \leq C\left(|e_n|^2 + |\delta Z_n|^2\right).$$

Hence

$$\mathbb{E}|e_{n+1}|^2 \leq (1 + \alpha + C\Delta t^2)\mathbb{E}|e_n|^2 + C(\Delta t + \Delta t^2)\mathbb{E}|\delta Z_n|^2.$$

Choosing $\alpha = C\Delta t$ and absorbing higher-order terms into the constant gives

$$\mathbb{E}|e_{n+1}|^2 \leq (1 + C\Delta t)\mathbb{E}|e_n|^2 + C\Delta t\,\mathbb{E}|\delta Z_n|^2.$$

Iterating this inequality (Kloeden & Platen (1992)) yields

$$\mathbb{E}|e_N|^2 \leq (1 + C\Delta t)^N \mathbb{E}|e_0|^2 + C\sum_{k=0}^{N-1}(1 + C\Delta t)^{N-1-k}\Delta t\,\mathbb{E}|\delta Z_k|^2.$$

Since

$$(1 + C\Delta t)^N \leq e^{CT},$$

we obtain

$$\mathbb{E}|e_N|^2 \leq e^{CT}\mathbb{E}|e_0|^2 + Ce^{CT}\sum_{k=0}^{N-1}\Delta t\,\mathbb{E}|\delta Z_k|^2.$$

The right-hand side depends on the accumulated sum of local trajectory-level errors. Therefore, the terminal error is only an aggregate summary: different distributions of the local errors $\{\mathbb{E}|\delta Z_k|^2\}_{k=0}^{N-1}$ can lead to the same, or similar, terminal error bound. This proves the claim. □

Proposition 2 shows that terminal error is a compressed summary of local trajectory errors accumulated over all time steps. Consequently, many distinct collections of local errors can produce similar terminal losses. Therefore, DBDP is expected to produce trajectories with better local consistency and reduced error accumulation. Taken together, Proposition 1 and 2 explain the empirical differences observed in our experiments. DBDP imposes stronger supervision by enforcing the local BSDE dynamics throughout the trajectory, whereas Deep BSDE optimises only the terminal objective. Since terminal error is an aggregate measure of trajectory errors, comparable terminal losses do not imply comparable trajectory quality. This theoretical distinction is reflected in our experiments, where DBDP1 and DBDP2 consistently achieve improved one-step and rollout accuracy relative to Deep BSDE.

7.2 A Structural Error Decomposition for DBDP1 and DBDP2

The empirical results show that DBDP2 often achieves rollout errors roughly one order of magnitude smaller than DBDP1. To explain this observation, we isolate a structural error term that is present in DBDP1 but absent in DBDP2.

Let U_n^θ denote the value network at time t_n. In DBDP1, the gradient process is represented by an independent network V_n^ϕ, whereas in DBDP2 it is obtained by differentiating the value network. Define the structural inconsistency of DBDP1 by

$$\delta_n(X_n) = V_n^\phi(X_n) - \sigma(t_n, X_n)^\top \nabla_x U_n^\theta(X_n). \tag{10}$$

Thus δ_n measures the deviation of the independently learned Z-network from the gradient field induced by the learned value network.

Proposition 3. *Consider the one-step discretized BSDE update*

$$Y_{n+1} = Y_n - f(t_n, X_n, Y_n, Z_n)\Delta t + Z_n^\top \Delta W_n. \tag{11}$$

Let

$$\widetilde{Z}_n = \sigma(t_n, X_n)^\top \nabla_x U_n^\theta(X_n), \qquad \widehat{Z}_n = \widetilde{Z}_n + \delta_n. \tag{12}$$

Assume that the driver f is Lipschitz in z with constant L_z, and that δ_n is $\mathcal{F}_{t_n}$-measurable. Let $\widetilde{Y}_{n+1}$ and $\widehat{Y}_{n+1}$ denote the one-step updates obtained using $\widetilde{Z}_n$ and $\widehat{Z}_n$, respectively, while keeping the same value $Y_n = U_n^\theta(X_n)$. Then

$$\mathbb{E}\left[|\widehat{Y}_{n+1} - \widetilde{Y}_{n+1}|^2 \,\middle|\, \mathcal{F}_{t_n}\right] \leq 2L_z^2\Delta t^2\|\delta_n\|^2 + 2\Delta t\|\delta_n\|^2. \tag{13}$$

Consequently,

$$\mathbb{E}\left[|\widehat{Y}_{n+1} - \widetilde{Y}_{n+1}|^2\right] \leq 2\left(L_z^2\Delta t^2 + \Delta t\right)\mathbb{E}\|\delta_n\|^2. \tag{14}$$

Proof. The difference between the two updates is

$$\widehat{Y}_{n+1} - \widetilde{Y}_{n+1} = -\Big[f(t_n, X_n, Y_n, \widetilde{Z}_n + \delta_n) - f(t_n, X_n, Y_n, \widetilde{Z}_n)\Big]\Delta t + \delta_n^\top \Delta W_n. \tag{15}$$

Using the inequality $|a+b|^2 \le 2|a|^2 + 2|b|^2$ gives

$$|\widehat{Y}_{n+1} - \widetilde{Y}_{n+1}|^2 \le 2\Big|f(t_n, X_n, Y_n, \widetilde{Z}_n + \delta_n) - f(t_n, X_n, Y_n, \widetilde{Z}_n)\Big|^2 \Delta t^2 + 2|\delta_n^\top \Delta W_n|^2. \tag{16}$$

By the Lipschitz assumption in z,

$$\Big|f(t_n, X_n, Y_n, \widetilde{Z}_n + \delta_n) - f(t_n, X_n, Y_n, \widetilde{Z}_n)\Big| \le L_z \|\delta_n\|. \tag{17}$$

Furthermore, since δ_n is $\mathcal{F}_{t_n}$-measurable and ΔW_n is independent of $\mathcal{F}_{t_n}$ with covariance $\Delta t I_d$,

$$\mathbb{E}\left[|\delta_n^\top \Delta W_n|^2 \,\middle|\, \mathcal{F}_{t_n}\right] = \Delta t \|\delta_n\|^2. \tag{18}$$

Combining the above inequalities proves equation 13. □

The theorem shows that DBDP1 contains an additional local error channel generated by the mismatch between the independently learned gradient network and the gradient field induced by the value network. In contrast, DBDP2 defines $Z_n = \sigma(t_n, X_n)^\top \nabla_x U_n^\theta(X_n)$, so that $\delta_n = 0$ by construction. Therefore, the structural inconsistency contribution in Equation 13 vanishes for DBDP2.

Under the assumptions of the proposition 3, the cumulative inconsistency contribution over a rollout satisfies the scaling

$$\sum_{n=0}^{N-1} \mathbb{E}\left[|\widehat{Y}_{n+1} - \widetilde{Y}_{n+1}|^2\right] \le 2 \sum_{n=0}^{N-1} \left(L_z^2 \Delta t^2 + \Delta t\right) \mathbb{E}\|\delta_n\|^2. \tag{19}$$

Proposition 3 identifies an error component that depends solely on the inconsistency between the independently learned value and gradient networks. This component is identically zero for DBDP2 because the gradient is obtained directly from the value network. Therefore, under identical approximation quality of the value function, DBDP2 eliminates one source of local discretisation error that is necessarily present in DBDP1.

8 Code Availability

All experiments were implemented in JAX Bradbury et al. (2018) within a modular framework comprising reusable BSDE problem definitions, forward BSDE rollout simulators, solver implementations for Deep BSDE, DBDP1, and DBDP2, training and evaluation pipelines, and experiment configuration utilities. The repository includes scripts for reproducing all reported experiments, tables, and figures, and is designed to facilitate extension to additional BSDE solvers, benchmark PDEs, and evaluation protocols. Code is available at: https://anonymous.4open.science/r/jax-bsde-repro/

9 Conclusion and Limitation

We presented a reproducibility study of Deep BSDE, DBDP1, and DBDP2 under a unified implementation framework. Our results demonstrate that trajectory-level metrics provide substantially richer insight than terminal error alone, with DBDP2 consistently exhibiting improved stability across several benchmark problems. Finally, we release our modular JAX implementation as an open-source baseline to support reproducible research and future development of neural BSDE solvers. Our study is limited to semilinear PDEs and three representative BSDE solvers. Extending the evaluation to second-order BSDE methods, reflected BSDEs, or more recent transformer-based solvers remains future work. Future work will focus on extending the framework with additional BSDE-based algorithms, benchmark PDEs, and evaluation protocols.

References

Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic pdes. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154, January 2021. ISSN 1095-7197. doi: 10.1137/19m1297919. URL http://dx.doi.org/10.1137/19M1297919.

Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Weinan E, 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. doi: 10.1007/s40304-017-0117-6. URL https://doi.org/10.1007/s40304-017-0117-6.

Lawrence C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2 edition, 2010. ISBN 978-0-8218-4974-3. doi: 10.1090/gsm/019.

Masaaki Fujii, Akihiko Takahashi, and Masayuki Takahashi. Asymptotic expansion as prior knowledge in deep learning method for high dimensional bsdes, 2019. URL https://arxiv.org/abs/1710.07030.

Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. A regression-based monte carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability*, 15(3), August 2005. ISSN 1050-5164. doi: 10.1214/105051605000000412. URL http://dx.doi.org/10.1214/105051605000000412.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, August 2018. ISSN 1091-6490. doi: 10.1073/pnas.1718942115. URL http://dx.doi.org/10.1073/pnas.1718942115.

Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear pdes, 2020. URL https://arxiv.org/abs/1902.01599.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*, volume 23 of *Applications of Mathematics*. Springer, Berlin, Heidelberg, 1992. ISBN 978-3-540-54062-5. doi: 10.1007/978-3-662-12616-5. URL https://doi.org/10.1007/978-3-662-12616-5.

Jin Ma and Jiongmin Yong. *Forward-Backward Stochastic Differential Equations and Their Applications*, volume 1702 of *Lecture Notes in Mathematics*. Springer, Berlin, Heidelberg, 1999. ISBN 978-3-540-48831-6. doi: 10.1007/978-3-540-48831-6.

E. Pardoux and S. Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In Boris L. Rozovskii and Richard B. Sowers (eds.), *Stochastic Partial Differential Equations and Their Applications*, pp. 200–217, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-47015-1.

Etienne Pardoux and Shanjian Tang. Forward-backward stochastic differential equations and quasilinear parabolic pdes. *Probability Theory and Related Fields*, 114(2):123–150, 1999. doi: 10.1007/s004409970001. URL https://doi.org/10.1007/s004409970001.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

A Validation Results from Final Checkpoints

We additionally evaluated the final checkpoints from the five-seed DBDP1 and DBDP2 runs using held-out validation batches. For each benchmark problem and time-discretization level, the plots below report the mean validation terminal RMSE together with one standard deviation across seeds. The x-axis corresponds to the number of time steps $N \in \{20, 50, 100, 250, 500\}$ used in the time discretization.

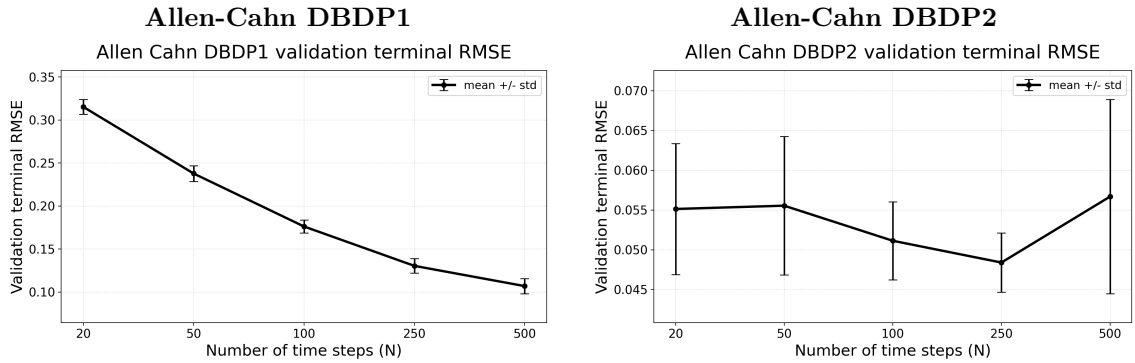

Figure 2: Final-checkpoint validation terminal RMSE for the Allen-Cahn benchmark. Each point reports the five-seed mean at the corresponding time-discretization level, and the error bar reports one standard deviation across seeds.

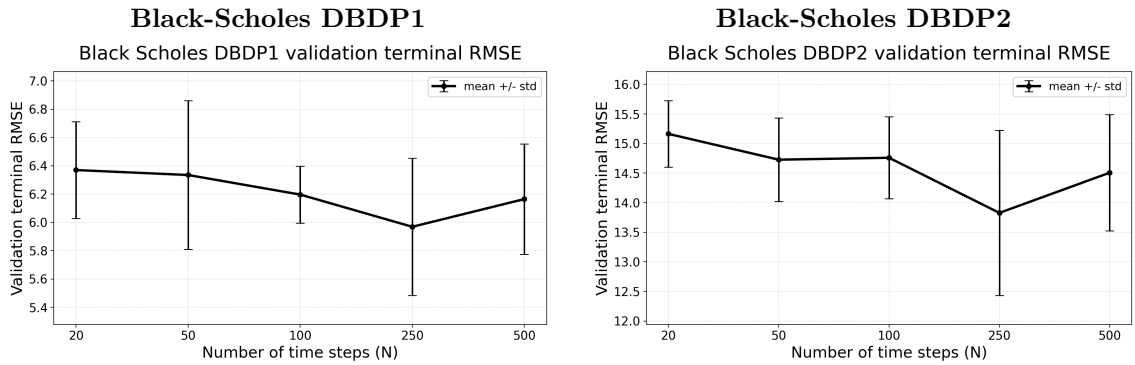

Figure 3: Final-checkpoint validation terminal RMSE for the nonlinear Black-Scholes benchmark. The plots compare DBDP1 and DBDP2 over the same discretization grid $N \in \{20, 50, 100, 250, 500\}$, with error bars showing one standard deviation over five independent seeds.

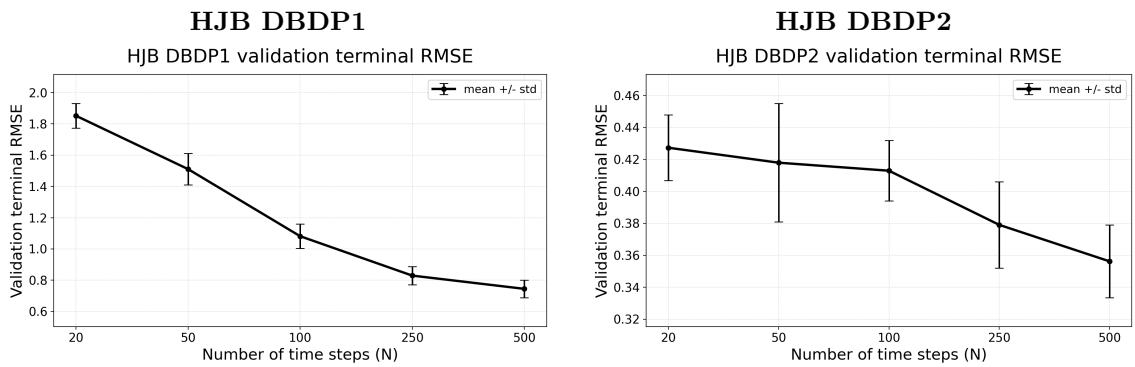

Figure 4: Final-checkpoint validation terminal RMSE for the HJB benchmark. Each subplot shows the validation terminal RMSE mean and standard deviation across five seeds for one DBDP solver variant.

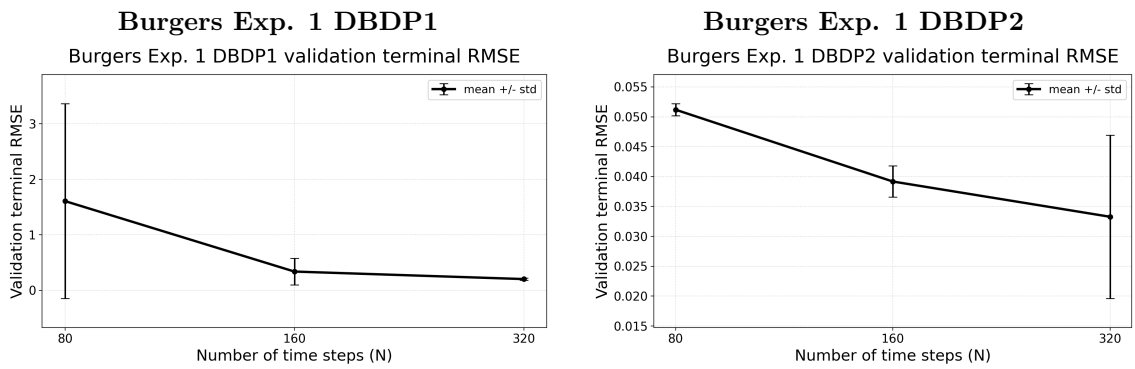

Figure 5: Final-checkpoint validation terminal RMSE for the first Burgers paper setup. The plots compare DBDP1 and DBDP2 over $N \in \{80, 160, 320\}$, with error bars showing one standard deviation over five independent seeds.

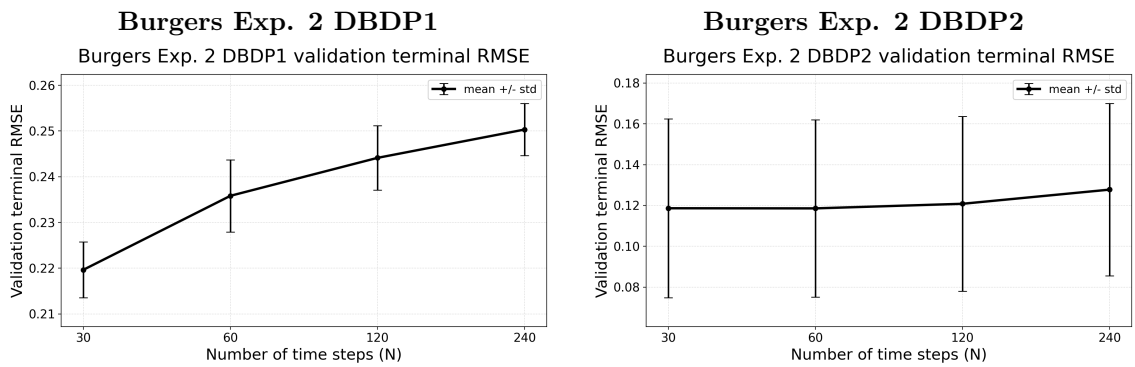

Figure 6: Final-checkpoint validation terminal RMSE for the second Burgers paper setup. The plots compare DBDP1 and DBDP2 over $N \in \{30, 60, 120, 240\}$, with error bars showing one standard deviation over five independent seeds.

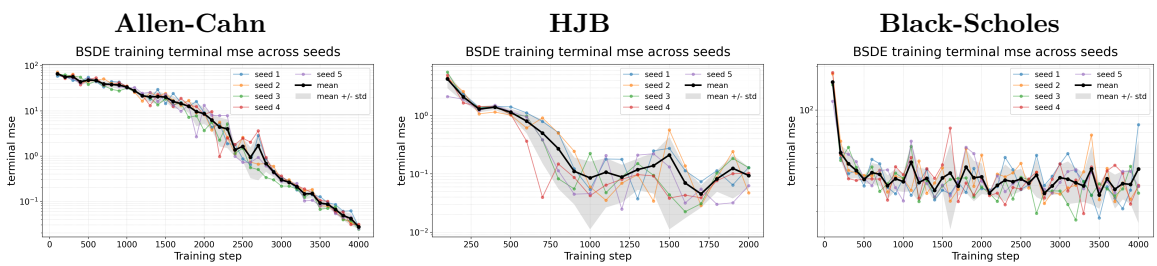

Figure 7: Training loss curves for the five independent Deep BSDE reproduction runs used in Table 2. Each plot shows the per-seed terminal MSE recorded every 100 optimization steps, together with the across-seed mean and standard-deviation band. Allen-Cahn and Black-Scholes are shown through 4000 optimizer steps, while HJB is shown through 2000 optimizer steps, matching the final rows compared against the original paper.

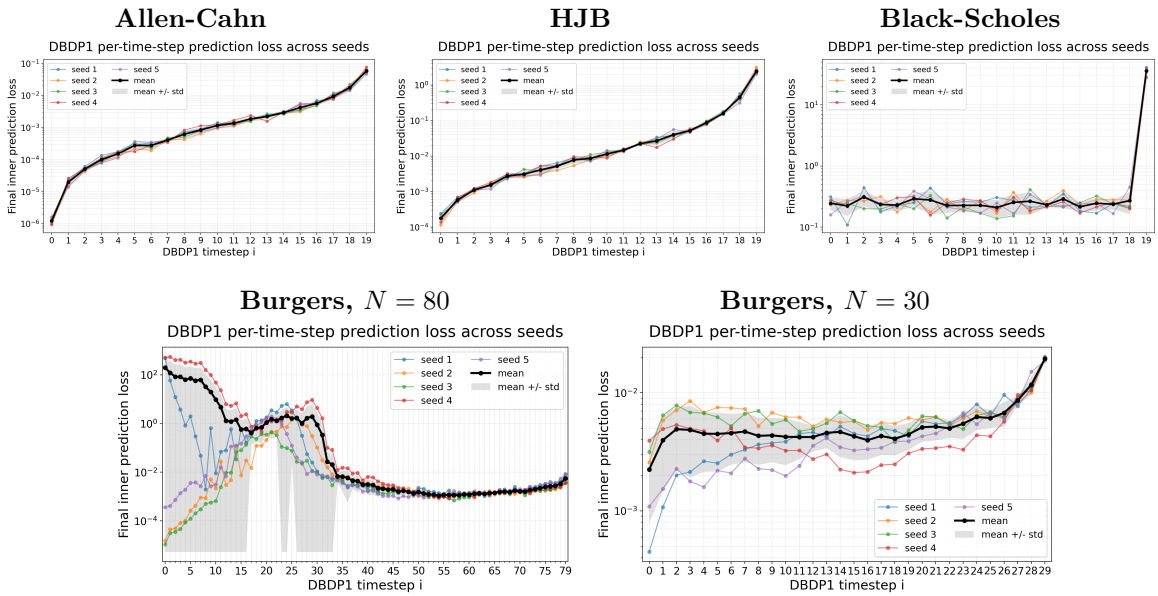

Figure 8: Per-time-step prediction losses for the five independent DBDP1 reproduction runs. Allen-Cahn, HJB, and Black-Scholes are shown for the $N = 20$ reproduction setting, while Burgers is shown for the $N = 80$ first paper setup and the $N = 30$ second paper setup. Each point reports the final prediction loss at the corresponding backward time index, with individual seed curves and the across-seed mean and standard-deviation band.

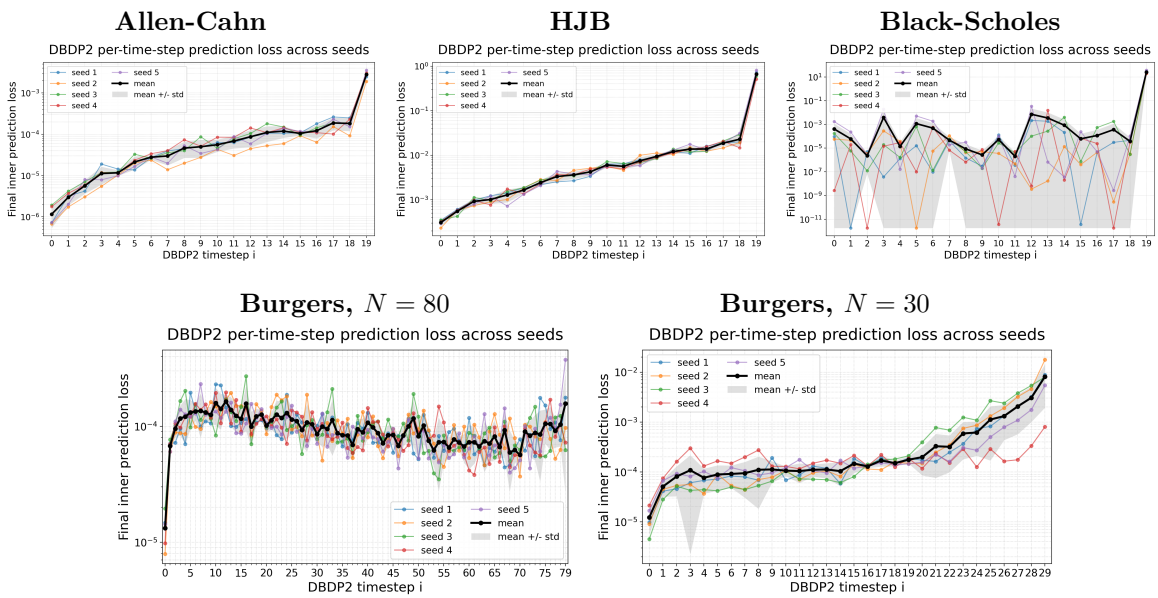

Figure 9: Per-time-step prediction losses for the five independent DBDP2 reproduction runs. Allen-Cahn, HJB, and Black-Scholes are shown for the $N = 20$ reproduction setting, while Burgers is shown for the $N = 80$ first paper setup and the $N = 30$ second paper setup. As in Figure 8, the curves show the final loss at each backward time index, separating individual seed behavior from the across-seed aggregate trend.

B Auxiliary Inequality

We make repeated use of the following elementary inequality.

Lemma 1. *For any vectors $a, b, c \in \mathbb{R}^d$ and any $\alpha > 0$,*

$$|a + b + c|^2 \leq (1 + \alpha)|a|^2 + C_\alpha |b + c|^2,$$

where

$$C_\alpha = 1 + \frac{1}{\alpha}.$$

Proof. Expanding the squared norm gives

$$|a + b + c|^2 = |a|^2 + 2\langle a, b + c\rangle + |b + c|^2.$$

Applying Young's inequality,

$$2\langle a, b + c\rangle \leq \alpha|a|^2 + \frac{1}{\alpha}|b + c|^2,$$

for any $\alpha > 0$.

Substituting this bound into the previous expression yields

$$\begin{aligned} |a + b + c|^2 &\leq |a|^2 + \alpha|a|^2 + \frac{1}{\alpha}|b + c|^2 + |b + c|^2 \\ &= (1 + \alpha)|a|^2 + \left(1 + \frac{1}{\alpha}\right)|b + c|^2. \end{aligned}$$

Therefore,

$$|a + b + c|^2 \leq (1 + \alpha)|a|^2 + C_\alpha |b + c|^2,$$

where $C_\alpha = 1 + \alpha^{-1}$. □