Structured Noise to Reduce Deep BSDE Training Loss

Nathan Evans*

Supervisor: Dr James Foster

1 Introduction

This project focuses on improving the Deep BSDE method [1]. In particular, we investigate alternative driving processes for the forward SDE, replacing standard Brownian motion with structured increments constructed from blocks of K taken from an orthogonal matrix. The aim is to assess whether these modifications reduce the Milstein correction term and lead to improved training performance and lower loss.

2 Background

In [1] they propose the deep BSDE method to tackle the so-called curse of dimensionality when solving high dimensional PDEs. They consider semi-linear parabolic PDEs of the form

$$\frac{\partial u}{\partial t}(t,x)+\frac{1}{2}\mathrm{Tr}\Big(\sigma(t,x)\,\sigma(t,x)^\top \mathrm{Hess}_x u(t,x)\Big)+\nabla u(t,x)\cdot\mu(t,x)+f\big(t,x,u(t,x),\sigma(t,x)^\top\nabla u(t,x)\big)=0, \tag{1}$$

with terminal condition $u(T,x)=g(x)$.
Here:

- $u(t,x)$ is the unknown scalar function.
- $x\in\mathbb{R}^m$ is the spatial variable, and $t\in[0,T]$ is time.
- $\mu(t,x)\in\mathbb{R}^m$ is a given drift vector.
- $\sigma(t,x)\in\mathbb{R}^{m\times d}$ is a given diffusion matrix.
- $\mathrm{Hess}_x u(t,x)\in\mathbb{R}^{m\times m}$ is the Hessian of u with respect to x.
- $\nabla u(t,x)\in\mathbb{R}^m$ is the gradient of u with respect to x.
- $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix.
- $f(t,x,u,z)$ is a given nonlinear function.

They also focus on solving for the solution at $t=0$ and for some $x=\xi$.

Let $\{W_t\}_{t\in[0,T]}$ be a d-dimensional Brownian motion and $\{X_t\}_{t\in[0,T]}$ be a m-dimensional stochastic process which satisfies

*This work was supported jointly by the London Mathematical Society (LMS) and the University of Bath under the LMS' Undergraduate Research Bursary.

$$X_t = \xi + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s. \tag{2}$$

Then the solution of Eq. (1) satisfies the following BSDE

$$u(t, X_t) - u(0, X_0) = -\int_0^t f(s, X_s, u(x, X_s), \sigma^T(s, X_s)\nabla u(s, X_s))ds + \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s)dW_s. \tag{3}$$

An explanation of this derivation is provided in [1].

3 Time Discretisation

Given a partition of the time interval $[0, T] : 0 = t_0 < t_1 < \cdots < t_N = T$, we discretise Eq. (2) with the typical Euler scheme for $n = 1, \ldots, N-1$:

$$X_{n+1} - X_n \approx \mu(t_n, X_n)\Delta t_n + \sigma(t_n, X_n)\Delta W_n, \tag{4}$$

where $\Delta t_n = t_{n+1} - t_n$ and $\Delta W_n = W_{n+1} - W_n$.

We assume that the function u does not vary too much over the time step, so that the second-order Taylor expansion around X_n provides a good approximation. When using K-step structured increments, the expansion effectively approximates u at X_{n+K}, and the accuracy relies on the cumulative change being sufficiently small.

We now fix time, apply the PDE solution u and Taylor expand to get

$$u(X_{n+1}) \approx u(X_n) + \sum_{i=1}^{m} \frac{\partial u}{\partial x_i}(X^i_{n+1} - X^i_n) + \tfrac{1}{2}\sum_{i,j=1}^{m} \frac{\partial^2 u}{\partial x_i \partial x_j}(X^i_{n+1} - X^i_n)(X^j_{n+1} - X^j_n).$$

Substituting Eq. (4) we obtain

$$\begin{aligned} u(X_{n+1}) \approx u(X_n) &+ \sum_{i=1}^{m} \frac{\partial u}{\partial x_i}\,\mu^i(X_n)\,\Delta t + \sum_{i=1}^{m}\sum_{k=1}^{d} \frac{\partial u}{\partial x_i}\,\sigma^{ik}(X_n)\,\Delta W^k_n \\ &+ \tfrac{1}{2}\sum_{i,j=1}^{m}\sum_{k,\ell=1}^{d} \frac{\partial^2 u}{\partial x_i \partial x_j}\,\sigma^{ik}(X_n)\sigma^{j\ell}(X_n)\,\Delta W^k_n\,\Delta W^\ell_n. \end{aligned}$$

Adding and subtracting

$$\tfrac{1}{2}\sum_{i,j=1}^{m}\sum_{k=1}^{d} \frac{\partial^2 u}{\partial x_i \partial x_j}\,\sigma^{ik}(X_n)\sigma^{jk}(X_n)\,\Delta t,$$

we can rewrite

$$\begin{aligned} u(X_{n+1}) \approx u(X_n) &+ \left[\sum_{i=1}^{m} \frac{\partial u}{\partial x_i}\,\mu^i(X_n) + \tfrac{1}{2}\sum_{i,j=1}^{m}\sum_{k=1}^{d} \frac{\partial^2 u}{\partial x_i \partial x_j}\,\sigma^{ik}(X_n)\sigma^{jk}(X_n)\right]\Delta t \\ &+ \sum_{i=1}^{m}\sum_{k=1}^{d} \frac{\partial u}{\partial x_i}\,\sigma^{ik}(X_n)\,\Delta W^k_n \\ &+ \tfrac{1}{2}\sum_{i,j=1}^{m}\sum_{k,\ell=1}^{d} \frac{\partial^2 u}{\partial x_i \partial x_j}\,\sigma^{ik}(X_n)\sigma^{j\ell}(X_n)\,(\Delta W^k_n\,\Delta W^\ell_n - \delta_{k\ell}\Delta t), \end{aligned} \tag{5}$$

where $\delta_{k\ell}$ is the Kronecker delta.

The third term in Eq. (5) sums over $\Delta W_n^k \Delta W_n^\ell - \delta_{k\ell}\Delta t$, which is present in the Milstein correction term. In the next section we devise a scheme to try and reduce this quantity.

4 Minimising Milstein Correction

The final term in Eq. (5) can be rewritten with vector/matrix notation as

$$\frac{1}{2}\nabla^2 u(X_n)\sigma(X_n) \otimes \sigma(X_N)\left(\Delta W_n \Delta W_n^T - \Delta t I_d\right),$$

where $\otimes$ is the tensor product.

With the goal of reducing error we consider a seqeunce of K increments $\{\Delta W_n\}_{1\leq n\leq K}$ such that

$$\sum_{n=1}^{K}\left(\Delta W_n \Delta W_n^T - \Delta t I_d\right) = 0.$$

To achieve this we first define the Hadamard matrices:

$$H_0 = \begin{bmatrix}1\end{bmatrix} \quad H_{k+1} = \begin{bmatrix} H_k & H_k \\ H_k & -H_k \end{bmatrix},$$

we choose k to be the smallest integral such that $K \coloneqq 2^k \leq m$. We define the following K vectors,

$$E \coloneqq \{e_1, e_2, \ldots, e_K\},$$

where each vector $e_i \in \mathbb{R}^m$ is a truncated column of the Hadamard matrix H_k:

$$H_k = \begin{bmatrix} e_1 & e_2 & \ldots & e_K \\ r_1 & r_2 & \ldots & r_K \end{bmatrix},$$

with $r_i \in \mathbb{R}^{K-m}$. let $\pi : \{1, \ldots, K\} \to \{1, \ldots, K\}$ be a uniform random variable that shuffles $\{1, \ldots, K\}$. We then define $\{\Delta W_n\}_{1\leq n\leq K}$ as

$$\Delta W_n \coloneqq Z_n \sqrt{\Delta t} e_{\pi(n)},$$

where each $Z_n \in \{-1, 1\}$ is an independent Rademacher random variable.

The rationale behind this idea is that we want $\sum_{n=1}^{K} \Delta W_n \Delta W_n^T$ to be diagonal so we consider sampling ΔW_n from the columns of an orthogonal matrix. The Hadamard matrices are a conveinient way to proudce such a matrix with ± 1s as the entries. Even when we truncate the column vectors $\sum_{n=1}^{K} \Delta W_n \Delta W_n^T$ is still diagonal.

You can also visualise and compare these sample paths with Brownian motion shown here [2].

5 Comparison of Increment Schemes

We compare this new sampling method with the standard Brownian increments used in [1]. Given computational constraints we look at only the Allen-Cahn and Hamilton-Jacobi-Bellman examples across dimensions, we also look at the CIR examples from [3]. For comprehensive details on the experiments refer to Appendix A

5.1 Allen-Cahn Equation

The Allen-Cahn equation is a reaction-diffusion equation introduced to model the process of phase separation in alloys. In [1] they consider a typical Allen-Cahn equation with the "double-well potential",

$$\frac{\partial u}{\partial t}(t,x) = \Delta u(t,x) + u(t,x) - [u(t,x)]^3, \tag{6}$$

with the initial condition $u(0,x) = g(x)$, where $g(x) = \frac{1}{2+0.4||x||^2}$ for $x \in \mathbb{R}^m$ and $d = m$. We look at the training loss for $m \in \{1, 5, 15, 100\}$. By applying a time reversal transformation Eq. (6) becomes of the form of Eq. (1) and can be solved using the Deep BSDE method.

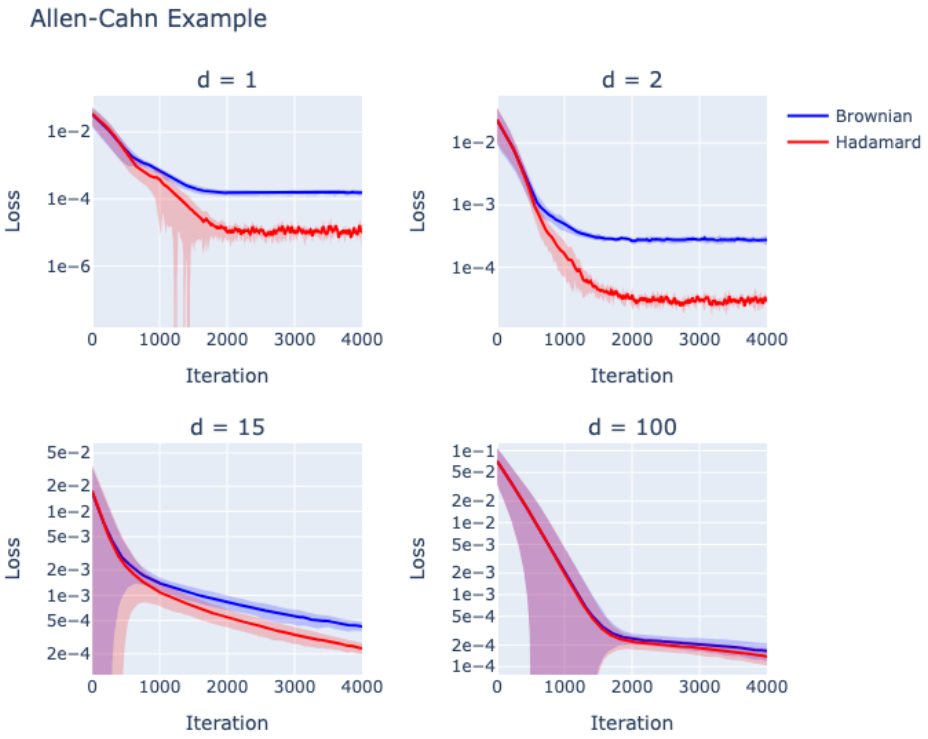

Figure 1: Mean training loss with ±1 SD for the Allen-Cahn equation across dimensions.

Dimension	Brownian	Hadamard	Improvement
1	1.5e-4	**1.2e-5**	× 12.8
2	2.7e-4	**2.8e-5**	× 9.7
15	4.3e-4	**2.3e-4**	× 1.8
100	1.7e-4	**1.4e-4**	× 1.2

Table 1: Mean final training loss for Brownian and Hadamard sampling. Improvement shows the factor by which Hadamard outperforms Brownian.

From the plots in Fig. 1, it is evident that the Hadamard method consistently performs at least as well as the standard Brownian approach. In every case, our scheme achieves a lower average training loss, as confirmed by Table 1. The improvement factor tends to decrease with increasing dimension, though substantial gains are observed in lower dimensions. This may be due to the reduced influence of the second derivative and diffusion terms in lower-dimensional settings.

5.2 Hamilton-Jacobi-Bellman Equation

The Hamilton–Jacobi–Bellman (HJB) equation is a nonlinear PDE that describes the value function in optimal control problems. A common benchmark is the Linear–Quadratic–Gaussian (LQG) problem, where the system dynamics are linear, the cost is quadratic, and the noise is Gaussian. In [1] they consider a classical linear-quadratic Gaussian (LQG) control problem,

$$dX_t = 2\sqrt{\lambda} m_t dt + \sqrt{2} dW_t, \tag{7}$$

with $t \in [0, T], X_0 = x$, and with the cost functional $J(\{m_t\}_{0 \le t \le T}) = \mathbb{E}[\int_0^T ||m_t||^2 dt + g(X_T)]$. Here $\{X_t\}_{t\in[0,T]}$ is the state process, $\{m_t\}_{t\in[0,T]}$ is the control process, λ is a positive constant representing the "strength" of the control and $\{W_T\}_{t\in[0,T]}$ is a standard Brownian motion. The goal is to minimise the cost functional though the control process.

The HJB equation for this problem is given by,

$$\frac{\partial u}{\partial t}(t, x) + \Delta u(t, x) - \lambda ||\nabla u(t, x)||^2 = 0. \tag{8}$$

The value of the solution $u(t, x)$ of Eq. (8) at $t = 0$ is the optimal cost when the state starts from x.

Again we run their experiment for $m \in \{1, 5, 15, 100\}$ where again $m = d$, the terminal condition for this experiment is $g(x) = \ln\left(\frac{1+||x||^2}{2}\right)$.

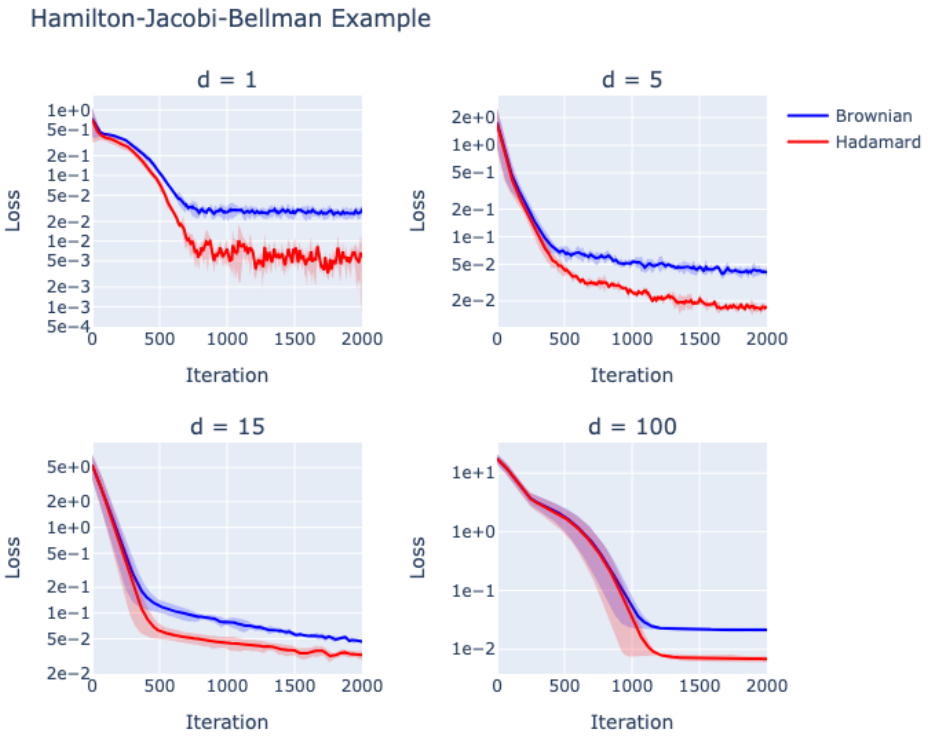

Figure 2: Mean training loss with ±1 SD for the HJB equation across dimensions.

Dimension	Brownian	Hadamard	Improvement
1	3.0e-2	**6.0e-3**	× 4.9
2	4.4e-2	**9.6e-3**	× 4.5
15	4.7e-2	**3.2e-2**	× 1.4
100	2.1e-2	**6.8e-3**	× 3.1

Table 2: Mean final training loss for Brownian and Hadamard sampling. Improvement shows the factor by which Hadamard outperforms Brownian.

The plots in Fig. 2 support the view that the Hadamard method never underperforms compared to the Brownian approach. As shown in Table 2, however, the improvement factor is not monotonic as a function of the dimension.

We also investigated whether the dimension influences training loss. In particular, we compared experiments run in dimensions close to powers of two with those run exactly at powers of two, in order to assess whether this had any significant effect on performance.

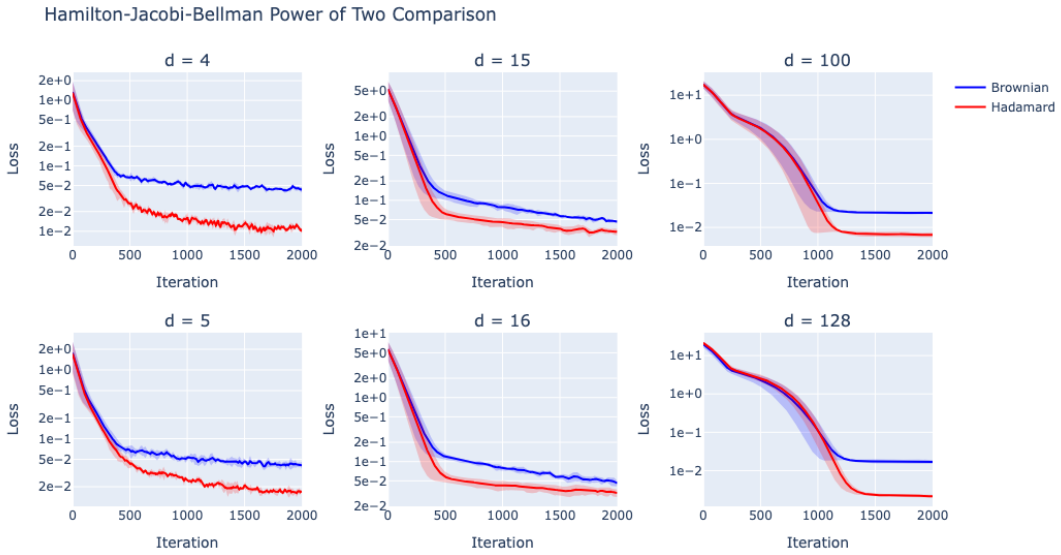

Figure 3: Mean training loss with ±1 SD for the HJB equation for dimension near and of powers of two.

Dimension	Brownian	Hadamard	Improvement
4	4.3e-2	1.0e-2	× **4.3**
5	4.1e-2	1.7e-2	× 2.4
15	4.7e-2	3.2e-2	× **1.4**
16	4.6e-2	3.2e-2	× **1.4**
100	2.1e-2	6.8e-3	× 3.1
128	1.7e-2	2.2e-3	×**7.9**

Table 3: Mean final training loss for Brownian and Hadamard sampling. Improvement shows the factor by which Hadamard outperforms Brownian.

From the plots in Fig. 3, there is no clear evidence that having the dimension of the Brownian motion equal to a power of two leads to greater improvement. Similarly, the data in Table 3 is

too limited to draw any definitive conclusions. However, the observed differences in performance between dimensions four and five, and between 100 and 128, suggest that further experimentation could be worthwhile.

5.3 Cox-Ingersoll-Ross Model

The Cox–Ingersoll–Ross (CIR) model is a widely used stochastic model for interest rates, where the short rate is mean-reverting and remains non-negative. It is commonly applied in finance to model the evolution of interest rates or other quantities that must stay positive.

We now turn to [3] for another example, in their numerical experiments they consider the CIR model. Let X_t be a short rate following an m-dimensional CIR process, i.e., each component of X_t follows a CIR process,

$$dX_t^i = a^i(b^i - X_t^i)dt + \sigma^i\sqrt{X_t^i}dW_t.$$

Assume Y_t is the zero-coupon bond paying 1 at maturity T. Under nonarbitrage condition, it is natural that

$$Y_t = \mathbb{E}\left[\exp\left(-\int_t^T \max_{1\le i\le n} X_s^i ds\right)|\mathcal{F}_t\right],$$

where $\{\mathcal{F}_t\}_{0\le t\le T}$ is the natural filtration generated by W and augmented by the P-null sets.

By some manipulation and assuming $Y_t = u(t,x)$ they end up with

$$\begin{cases} u_t + \sum_{i=1}^n a^i(b^i - x^i)\dfrac{\partial u}{\partial x^i} + \dfrac{1}{2} + \sum_{1\le i,j\le n} \sigma^i\sigma^j\sqrt{x^i x^j}\dfrac{\partial^2 u}{\partial x^i \partial x^j} - \left(\max_{1\le i\le n} x^i\right)u = 0, \\ u(T,x) = 1. \end{cases}$$

They set $d = T = 1$, and $\xi = (1,\dots,1) \in \mathbb{R}^m$. a^i, b^i, and σ^i are selected in $[0,1]$.

Notice that the Brownian motion used in this experiment is always one dimensional even if the spatial dimension is higher. We ran their experiment for $m \in \{1, 10\}$.

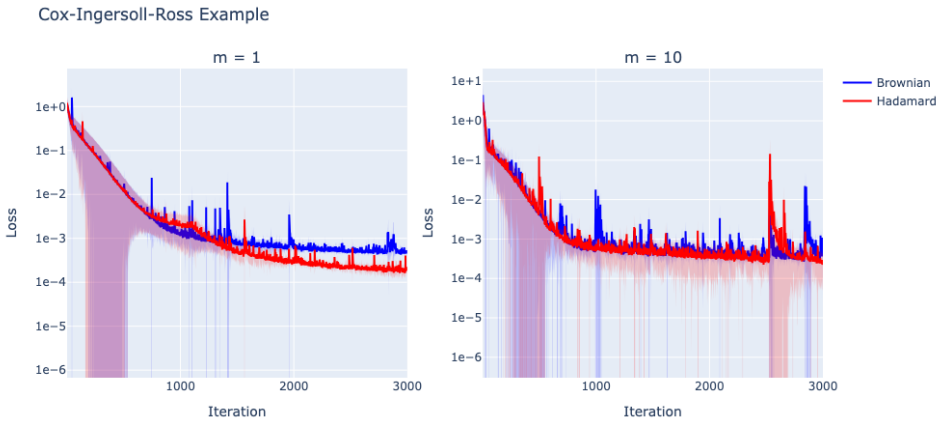

Figure 4: Mean training loss with ±1 SD for the CIR model across spatial dimensions.

Dimension	Brownian	Hadamard	Improvement
1	5.1e-4	**2.0e-4**	$\times$ 2.6
10	7.7e-4	**2.5e-4**	$\times$ 3.1

Table 4: Mean loss after the last training iteration for Brownian and Hadamard sampling. Improvement shows how many times lower the Hadamard mean loss is compared to Brownian.

The left plot in Fig. 4 shows that our scheme performs better compared to the Brownian increment when $m = 1$, looking at Table 4 it is clear that we also perform better in the $m = 10$ case by a similar improvement factor.

6 Conclusion & Future Work

Across the Allen–Cahn example, we demonstrated consistent improvements in average training loss with our method. Similar gains were observed for the HJB equation across all cases where the Brownian motion dimension matched the spatial variable. Finally, in the CIR model, we showed that even when a single-dimensional Brownian motion drives processes of dimension 1 and 10, our approach again achieves lower average training loss.

This study opens up several opportunities for further research,

- Extend the experiments to a broader range of examples, with a particular focus on how the dimension of the Brownian motion influences performance gains. Further investigate whether meaningful statements can be made for problems subject to specific structural constraints on u and σ.
- Our experiments focused on training loss, but future work could instead evaluate the PDE error, i.e. the discrepancy between the network's solution and the true PDE solution. This could be assessed in low-dimensional cases with known solutions, or via residuals of the PDE operator, giving a clearer view of approximation quality.
- In [4], a related approach is considered, where the BSDE is instead converted to Stratonovich form. This transformation also addresses the term identified in Eq. (5) , but it requires computing the trace of a Hessian matrix, which significantly increases the average training time. It would be interesting to see if our approach can be applied to this work to avoid expensive computation.
- Contribute a Deep BSDE solver to open source libraries such as Diffrax [5].

Appendix

A Experiment Methodology

For each experiment, we trained models across a range of dimensions using two different sampling methods for the noise in the forward SDE. We refer to the standard Brownian simulation as *Brownian* and to our new scheme as *Hadamard*. To ensure a fair comparison, the random number generators were seeded identically so that each pair of models started from the same initialisation, differing only in the sampling scheme. For every dimension, we conducted four independent runs and recorded the training loss. The reported performance corresponds to the average across these runs, with shaded regions in the plots indicating the ± 1 standard deviation interval. All code used and developed for this project is available here [6].

Table 5 summarises the key parameters used for all experiments. These parameters are adopted from the original papers referenced for each equation. The "number of runs" indicates how many independent trainings were performed to compute the mean training loss and the ± 1 standard deviation shown in the results.

Parameter	**Allen-Cahn**	**HJB**	**CIR** ($m = 1$)	**CIR** ($m = 10$)
Time horizon T	0.3	1	1	1
Number of time steps N	20	20	100	100
Monte Carlo paths / batch	64	64	1000	1000
Training iterations	4000	2000	3000	3000
Hidden layers	110, 110	110, 110	11	20
Activation	ReLU	ReLU	ReLU	ReLU
Weight initialization	Xavier	Xavier	He	He
Optimizer	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.01	0.001	0.001
Number of runs	4	4	4	4
Equation-specific parameter	-	$\lambda = 1$	-	-

Table 5: Experimental parameters for Allen-Cahn, HJB, and CIR equations.

References

[1] Jiequn Han, Arnulf Jentzen, and Weinan E. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510. DOI: 10.1073/pnas.1718942115. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1718942115.

[2] Nathan Evans. *Path Visualiser*. 2025. URL: https://nathanevaans.github.io/path_visualiser/.

[3] Yifan Jiang and Jinfeng Li. "Convergence of the Deep BSDE method for FBSDEs with non-Lipschitz coefficients". In: *Probability, Uncertainty and Quantitative Risk* 6.4 (2021), p. 391. ISSN: 2367-0126. DOI: 10.3934/puqr.2021019. URL: http://dx.doi.org/10.3934/puqr.2021019.

[4] Sungje Park and Stephen Tu. *Integration Matters for Learning PDEs with Backwards SDEs*. 2025. arXiv: 2505.01078 [cs.LG]. URL: https://arxiv.org/abs/2505.01078.

[5] Patrick Kidger. "On Neural Differential Equations". Github repository can be found at, https://github.com/patrick-kidger/diffrax. PhD thesis. University of Oxford, 2021.

[6] Nathan Evans. *Project Github Rrepository*. 2025. URL: https://github.com/nathanevaans/LMS-URB-Project.