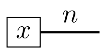
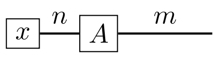
A. Graphical notation for tensor trains

In this section we provide some further material on tensor networks and their graphic notation. Let us start by noting that a vector $x \in \mathbb{R}^n$ can be interpreted as a tensor.



In the graphic representation contractions between indices are denoted by a line between the tensors. Below we contract a tensor $A \in \mathbb{R}^{n\times m}$ and $x \in \mathbb{R}^n$, which results in an element of $\mathbb{R}^m$, representing the usual matrix-vector product.



In Figure 9 an order 3 tensor $B \in \mathbb{R}^{n_1\times n_2\times n_3}$ is represented with three lines, not connected to any other tensor. As

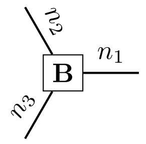


Figure 9. Graphical notation of simple tensors and tensor networks

another example, we can write the compact singular value decomposition in matrix form as $A = U\Sigma V$, with $U \in \mathbb{R}^{n,r}, \Sigma \in \mathbb{R}^{r,r}, V \in \mathbb{R}^{r,m}$, which we represent as a tensor network in Figure 10.

A.1. The local basis functions

Following the inexact description of the local basis functions we now give a precise formula. When optimizing the k-th component tensor, the local basis functions are given by setting $1 \le j_{k-1} \le r_{k-1}$, $1 \le i_k \le m$, and $1 \le j_k \le r_k$ within the following formula:

$$\begin{aligned} b_{j_{k-1},i_k,j_k}(x) = &\\ &\Bigg(\sum_{i_1,\dots,i_{k-1}}^{m,\dots,m}\ \sum_{j_1,\dots,j_{k-2}}^{r_1,\dots,r_{k-2}} u_1[i_1,j_1]\dots u_{k-1}[j_{k-2},i_{k-1},j_{k-1}]\\ &\quad \phi(x_1)[i_1]\dots\phi(x_{k-1})[i_{k-1}]\Bigg)\phi(x_k)[i_k]\\ &\Bigg(\sum_{i_{k+1},\dots,i_d}^{m,\dots,m}\ \sum_{j_k,\dots,j_{d-1}}^{r_k,\dots,r_{d-1}} u_{k+1}[j_k,i_{k+1},j_{k+1}]\dots u_d[j_{d-1},i_d]\\ &\quad \phi(x_{k+1})[i_{k+1}]\dots\phi(x_d)[i_d]\Bigg). \end{aligned} \tag{28}$$

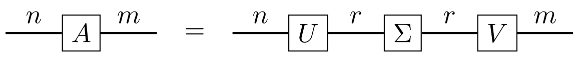


Figure 10. Graphical notation of simple tensors and tensor networks.

Note that in the above formula, every index except j_{k-1}, i_k and j_k is contracted, leaving an order three tensor. A simple reshape into one index then yields the local basis functions as used in this paper.

B. Proof of Theorem 3.1

Proof of Theorem 3.1. In this proof, we denote the underlying probability measure by $\mathbb{P}$, and the corresponding Hilbert space of random variables with finite second moments by $L^2(\mathbb{P})$. We define the linear subspace $\widetilde{\mathcal{U}} \subset L^2(\mathbb{P})$ by

$$\widetilde{\mathcal{U}} = \left\{ f(\widehat{X}_n) : f \in \mathcal{U} \right\}, \tag{29}$$

noting that $\widetilde{\mathcal{U}}$ is finite-dimensional by the assumption on $\mathcal{U}$, hence closed. The corresponding $L^2(\mathbb{P})$-orthogonal projection onto $\widetilde{\mathcal{U}}$ will be denoted by $\Pi_{\widetilde{\mathcal{U}}}$. By the nondegeneracy of σ, the law of $\widehat{X}_n$ has full support on Ω, and so $\|\cdot\|_{L^2(\mathbb{P})}$ is indeed a norm on $\widetilde{\mathcal{U}}$. Since $\widetilde{\mathcal{U}}$ is finite-dimensional, the linear operators

$$\widetilde{\mathcal{U}} \ni f(\widehat{X}_n) \mapsto \frac{\partial f}{\partial x_i}(\widehat{X}_n) \in L^2(\mathbb{P}) \tag{30}$$

are bounded, and consequently there exists a constant $C_1 > 0$ such that

$$\left\| \frac{\partial f}{\partial x_i}(\widehat{X}_n) \right\|_{L^2(\mathbb{P})} \le C_1 \left\| f(\widehat{X}_n) \right\|_{L^2(\mathbb{P})}, \tag{31}$$

for all $i = 1, \dots, d$ and $f \in \mathcal{U}$. Furthermore, there exists a constant $C_2 > 0$ such that

$$\mathbb{E}\left[f^4(\widehat{X}_n)\right]^{1/4} := \left\| f(\widehat{X}_N) \right\|_{L^4(\mathbb{P})} \le C_2 \left\| f(\widehat{X}_n) \right\|_{L^2(\mathbb{P})},$$

for all $f \in \mathcal{U}$, again by the finite-dimensionality of $\widetilde{\mathcal{U}}$ and the fact that on finite dimensional vector spaces, all norms are equivalent. By standard results on orthogonal projections, the solution to the iteration (20) is given by

$$\begin{aligned} V_n^{k+1}(\widehat{X}_n) = \Pi_{\widetilde{\mathcal{U}}}\big[-h(\widehat{X}_n, t_n, \widehat{Y}_n^k, \widehat{Z}_n^k)\Delta t +\\ \widehat{Z}_n^k \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1})\big]. \end{aligned}$$

We now consider the map $\Psi : \widetilde{\mathcal{U}} \to \widetilde{\mathcal{U}}$ defined by

$$\begin{aligned} f(\widehat{X}_n) \mapsto \Pi_{\widetilde{\mathcal{U}}}\big[-h(\widehat{X}_n, t_n, f(\widehat{X}_n), \sigma^\top \nabla f(\widehat{X}_n))\Delta t +\\ \sigma^\top \nabla f(\widehat{X}_n) \cdot \xi_{n+1}\sqrt{\Delta t} - \widehat{V}_{n+1}(\widehat{X}_{n+1})\big]. \end{aligned}$$

For $F_1, F_2 \in \widetilde{\mathcal{U}}$ with $F_i = f_i(\widehat{X}_n)$, $f_i \in \mathcal{U}$, we see that

$$
\begin{aligned}
&\|\Psi F_1 - \Psi F_2\|_{L^2(\mathbb{P})} \\
&= \big\|\Pi_{\widetilde{\mathcal{U}}}\big[- h(\widehat{X}_n, t_n, f_1(\widehat{X}_n), \sigma^\top \nabla f_1(\widehat{X}_n))\Delta t \\
&+ h(\widehat{X}_n, t_n, f_2(\widehat{X}_n), \sigma^\top \nabla f_2(\widehat{X}_n))\Delta t \\
&+ \sqrt{\Delta t}\left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n)\right) \cdot \xi_{n+1}\big]\big\|_{L^2(\mathbb{P})} \\
&\leq C_3 \left\|\Pi_{\widetilde{\mathcal{U}}}\right\|_{L^2(\mathbb{P}) \to L^2(\mathbb{P})} \Bigg(\Delta t \|F_1 - F_2\|_{L^2(\mathbb{P})} \\
&+ \sqrt{\Delta t}\left\|\left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n)\right) \cdot \xi_{n+1}\right\|_{L^2(\mathbb{P})} \Bigg)
\end{aligned}
$$

for some constant C_3 that does not depend on Δt, where we have used the triangle inequality, the Lipschitz assumption on h, the boundedness of σ, and the estimate (31). Using the Cauchy-Schwarz inequality, boundedness of σ as well as (31) and (32), the last term can be estimated as follows,

$$
\begin{aligned}
&\left\|\left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n)\right) \cdot \xi_{n+1}\right\|_{L^2(\mathbb{P})} \\
&\leq \left\|\left(\sigma^\top \nabla f_1(\widehat{X}_n) - \sigma^\top \nabla f_2(\widehat{X}_n)\right)^2\right\|^{1/2}_{L^2(\mathbb{P})} \left\|\xi^2_{n+1}\right\|^{1/2}_{L^2(\mathbb{P})} \\
&\leq C_4 \left\|F_1 - F_2\right\|_{L^2(\mathbb{P})},
\end{aligned}
$$

where C_4 is a constant independent of Δt. Collecting the previous estimates, we see that $\delta > 0$ can be chosen such that for all $t \in (0, \delta)$, the mapping Ψ is a contraction on $\widetilde{\mathcal{U}}$ when equipped with the norm $\|\cdot\|_{L^2(\mathbb{P})}$, that is,

$$
\|\Psi F_1 - \Psi F_2\| \leq \lambda \|F_1 - F_2\|, \tag{35}
$$

for some $\lambda < 1$ and all $F_1, F_2 \in \widetilde{\mathcal{U}}$. Finally, the statement follows from the Banach fixed point theorem. $\square$

C. Implementation details

For the evaluation of our approximations we rely on reference values of $V(x_0, 0)$ and further define the following two loss metrics, which are zero if and only if the PDE is fulfilled along the samples generated by the discrete forward SDE (7). In the spirit of (Raissi et al., 2019), we define the *PDE loss* as

$$
\begin{aligned}
\mathcal{L}_{\text{PDE}} = \frac{1}{KN}\sum_{n=1}^{N}\sum_{k=1}^{K}\Big((\partial_t + L)V(\widehat{X}_n^{(k)}, t_n) \\
+ h(\widehat{X}_n^{(k)}, t_n, V(\widehat{X}_n^{(k)}, t_n), (\sigma^\top \nabla V)(\widehat{X}_n^{(k)}, t_n))\Big)^2,
\end{aligned}
\tag{36}
$$

where $\widehat{X}_n^{(k)}$ are realizations of (7), the time derivative is approximated with finite differences and the space derivatives are computed analytically (or with automatic differentiation tools). We leave out the first time step $n = 0$ since the regression problem within the explicit and the implicit schemes for the tensor trains are not well-defined due to the fact that $\widehat{X}_0^k = x_0$ has the same value for all k. We still obtain a good approximation since the added regularization term brings a minimum norm solution with the correct point value $V(x_0, 0)$. Still, this does not aim at the PDE being entirely fulfilled at this point in time.

Further, we define the *relative reference loss* as

$$
\mathcal{L}_{\text{ref}} = \frac{1}{K(N+1)}\sum_{n=0}^{N}\sum_{k=1}^{K}\left|\frac{V(\widehat{X}_n^{(k)}, t_n) - V_{\text{ref}}(\widehat{X}_n^{(k)}, t_n)}{V_{\text{ref}}(\widehat{X}_n^{(k)}, t_n)}\right|, \tag{37}
$$

whenever a reference solution for all x and t is available.

All computation times in the reported tables are measured in seconds.

Our experiments have been performed on a desktop computer containing an AMD Ryzen Threadripper 2990 WX 32x 3.00 GHz mainboard and an NVIDIA Titan RTX GPU, where we note that only the NN optimizations were run on this GPU, since our TT framework does not include GPU support. It is expected that running the TT approximations on a GPU will improve time performances in the future (Abdelfattah et al., 2016).

All our code is available under `https://github.com/lorenzrichter/PDE-backward-solver`.

C.1. Details on neural network approximation

For the neural network architecture we rely on the *DenseNet*, which consists of fully-connected layers with additional skip connections as for instance suggested in (E & Yu, 2018) and being rooted in (Huang et al., 2017). To be precise, we define a version of the *DenseNet* that includes the terminal condition of the PDE (1) as an additive extension by

$$
\Phi_\varrho(x) = A_L x_L + b_L + \theta g(x), \tag{38}
$$

where x_L is specified recursively as

$$
y_{l+1} = \varrho(A_l x_l + b_l), \qquad x_{l+1} = (x_l, y_{l+1})^\top \tag{39}
$$

for $1 \leq l \leq L-1$ with $A_l \in \mathbb{R}^{r_l \times \sum_{i=0}^{l-1} r_i}, b_l \in \mathbb{R}^l, \theta \in \mathbb{R}$ and $x_1 = x$. The collection of matrices A_l, vectors b_l and the coefficient θ comprises the learnable parameters, and we introduce the vector $r := (d_{\text{in}}, r_1, \dots, r_{L-1}, d_{\text{out}})$ to represent a certain choice of a DenseNet architecture, where in our setting $d_{\text{in}} = d$ and $d_{\text{out}} = 1$. If not otherwise stated we fix the parameter θ to be 1. For the activation function $\varrho : \mathbb{R} \to \mathbb{R}$, that is to be applied componentwise, we choose $\tanh$.

For the gradient descent optimization we choose the Adam optimizer with the default parameters $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ (Kingma & Ba, 2014). In most of our experiments we chose a fixed learning rate η_{N-1} for the approximation of the first backward iteration step to approximate $\widehat{V}_{N-1}$ and another fixed learning rate η_n for all the other iteration steps to approximate $\widehat{V}_n$ for $0 \leq n \leq N-2$ (cf. Remark 3). Similarly, we denote with G_{N-1} and G_n the amount of gradient descent steps in the corresponding optimizations.

In Tables 7 and 8 we list our hyperparameter choices for the neural network experiments that we have conducted.

C.2. Details on tensor train approximation

For the implementation of the tensor networks we rely on the C++ library *xerus* (Huber & Wolf, 2014–2017) and the Python library *numpy* (Harris et al., 2020).

Within the optimization we have to specify the regularization parameter as noted in Remark 2, which we denot here by $\eta > 0$. We adapt this parameter in dependence of the current residual in the regression problem (20), i.e. $\eta = cw$, where $c > 0$ and w is the residual from the previous sweep of SALSA. In every all our experiments we set $c_\eta = 1$. Further, we have to specify the condition "*noChange* is *true*" within Algorithm 1. To this end we introduce a test set with equal size as our training set. We measure the residual within a single run of SALSA on the test set and the training set. If the change of the residual on either of this sets is below $\delta = 0.0001$ we set *noChange* = *true*. For the fixed-point iteration we have a two-fold stopping condition. We stop the iteration if either the Frobenius norm of the coefficients has a smaller relative difference than $\gamma_1 < 0.0001$ or if the values $\widehat{V}_n^{k+1}$ and $\widehat{V}_n^k$ and their gradients, evaluated at the points of the test set, have a relative difference smaller than $\gamma_2 < 0.00001$. Note that the second condition is essentially a discrete H^1 norm, which is necessary since by adding the final condition into the ansatz space the orthonormal basis property is violated.

Finally, we comment on the area $[a, b]$ where the 1-dimensional polynomials are orthonormalized w.r.t. the $H^2(a, b)$ norm, c.f. Remark 2. We obtain these polynomials by performing a Gram-Schmidt process starting with one-dimensional monomials. Thus, we have to specify the integration area $[a, b]$ for the different tests. In Section 4.1 we set $a = -6$ and $b = 6$. In Section 4.2 we set $a = -3$ and $b = 3$ for the case C diagonal and for the interacting case, where C is non-diagonal, we set $a = -8$ and $b = 2$. In Section 4.3 we choose $a = -0.2$ and $b = 6$.

HJB, $d = 10$, NN$_{\text{impl}}$ Figure 5
$K = 2000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 8000, G_{N-1} = 40000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0001$

HJB, $d = 100$, NN$_{\text{impl}}$ Table 1, Figures 6, 7
$K = 2000, \Delta t = 0.01$
$r = (100, 130, 130, 70, 70, 1)$
$G_n = 5000, G_{N-1} = 40000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0003$

HJB, $d = 100$, NN$_{\text{expl}}$ Table 1, Figures 6, 7
$K = 2000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 500, G_{N-1} = 7000$
$\eta_n = 0.00005, \eta_{N-1} = 0.0003$

HJB double well $d = 50$, NN$_{\text{impl}}$, Table 3
$K = 2000, \Delta t = 0.01$
$r = (50, 30, 30, 1)$
$G_n = 2000, G_{N-1} = 25000$
$\eta_n = 0.0002, \eta_{N-1} = 0.0005$

HJB interacting double well $d = 20$, NN$_{\text{impl}}$, Table 4
$K = 2000, \Delta t = 0.01$
$r = (50, 20, 20, 20, 20, 1)$
$G_n = 3000, G_{N-1} = 30000$
$\eta_n = 0.0007, \eta_{N-1} = 0.001$

CIR, $d = 100$, NN$_{\text{impl}}$ Table 5
$K = 1000, \Delta t = 0.01$
$r = (100, 110, 110, 50, 50, 1)$
$G_n = 2000$ for $0 \leq n \leq 15$
$G_n = 300$ for $16 \leq n \leq N-2$
$G_{N-1} = 10000$
$\eta_n = 0.00005, \eta_{N-1} = 0.0001$

Table 7. Neural network hyperparameters for the experiments in paper.

D. Further numerical examples

In this section we elaborate on some of the numerical examples from the paper and provide two additional problems.

PDE with unbounded solution $d = 10$, $\mathrm{NN}_{\mathrm{impl}}$, Table 9
$K = 1000, \Delta t = 0.001$
$r = (10, 30, 30, 1)$
$G_n = 100, G_{N-1} = 10000$
$\eta_n = 0.0001, \eta_{N-1} = 0.0001$

Allen-Cahn $d = 100$, $\mathrm{NN}_{\mathrm{impl}}$, Table 10
$K = 8000, \Delta t = 0.01$
$r = (10, 30, 30, 1)$
$G_n = 10000$ for $0 \leq n \leq 5$
$G_n = 6000$ for $6 \leq n \leq N-2$
$G_{N-1} = 15000$
$\eta_n = 0.0002, \eta_{N-1} = 0.001$

Table 8. Neural network hyperparameters for the additional experiments.

D.1. Hamilton-Jacobi-Bellman equation

Let us consider the HJB equation from Sections 4.1 and 4.2, which we can write as

$$(\partial_t + L)\, V(x,t) - \frac{1}{2}|(\sigma^\top \nabla V)(x,t)|^2 = 0, \tag{40a}$$

$$V(x,T) = g(x), \tag{40b}$$

in a generic form with the differential operator L being defined in (2). We can introduce the exponential transformation $\psi := e^{-V}$ and with the chain rule find that the transformed function fulfills the linear PDE

$$(\partial_t + L)\, \psi(x,t) = 0, \tag{41a}$$

$$\psi(x,T) = e^{-g(x)}. \tag{41b}$$

This is known as Hopf-Cole transformation, see also (Fleming & Soner, 2006; Hartmann et al., 2017). It is known that via the Feynman-Kac theorem (Karatzas & Shreve, 1998) the solution to this PDE has the stochastic representation

$$\psi(x,t) = \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x\right], \tag{42}$$

such that we readily get

$$V(x,t) = -\log \mathbb{E}\left[e^{-g(X_T)} \middle| X_t = x\right], \tag{43}$$

which we can use as a reference solution by approximating the expectation value via Monte Carlo simulation, however keeping in mind that in high dimensions corresponding estimators might have high variances (Hartmann & Richter, 2021).

Let us stress again that our algorithms only aim to provide a solution of the PDE along the trajectories of the forward process (4). Still, there is hope that our approximations generalize to regions "close" to where samples are available. To illustrate this, consider for instance the d-dimensional forward process

$$X_s = x_0 + \sigma W_s, \tag{44}$$

as for instance in Section 4.1, where now $\sigma > 0$ is one-dimensional for notational convenience. We know that $X_t \sim \mathcal{N}(x_0, \sigma^2 t\, \mathrm{Id}_{d\times d})$ and therefore note that for the expected distance to the origin it holds

$$\mathbb{E}\left[|X_t - x_0|\right] < \sqrt{\mathbb{E}\left[|X_t - x_0|^2\right]} = \sigma\sqrt{dt}. \tag{45}$$

This motivates evaluating the approximations along the curve

$$X_t = x_0 + \sigma\sqrt{t}\mathbf{1}, \tag{46}$$

where $\mathbf{1} = (1, \dots, 1)^\top$. Figure 11 shows that in this case we indeed have good agreement of the approximation with the reference solution when using TTs and that for NNs the deep neural network that we have specified in Table 7 generalizes worse than a shallower network with only two hidden layers consisting of 30 neurons each.

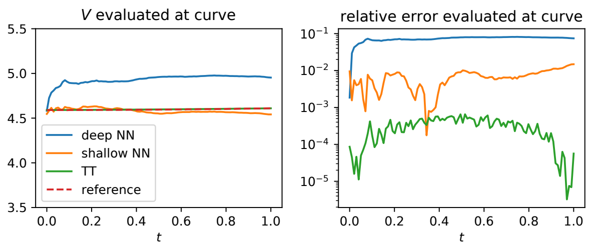


Figure 11. Approximations of the HJB equation in $d = 100$ evaluated along a representative curve.

D.2. PDE with unbounded solution

As an additional problem, we choose an example from (Huré et al., 2020) which offers an analytical reference solution. For the PDE as defined in (1) we consider the coefficients

$$b(x,t) = \mathbf{0}, \;\; \sigma(x,t) = \frac{\mathrm{Id}_{d\times d}}{\sqrt{d}}, \;\; g(x) = \cos\left(\sum_{i=1}^d i x_i\right), \tag{47}$$

$$h(x,t,y,z) = k(x) + \frac{y}{2\sqrt{d}}\sum_{i=1}^d z_i + \frac{y^2}{2}, \tag{48}$$

where, with an appropriately chosen k, a solution can shown to be

$$V(x,t) = \frac{T-t}{d}\sum_{i=1}^{d}\left(\sin(x_i)\mathbb{1}_{x_i<0} + x_i\mathbb{1}_{x_i\geq 0}\right) + \cos\left(\sum_{i=1}^{d} i x_i\right). \quad (49)$$

In Table 9 we compare the results for $d = 10, K = 1000, T = 1, \Delta t = 0.001, x_0 = (0.5,\dots,0.5)^\top$. For the TT case it was sufficient to set the ranks to 1 and the polynomial degree to 6. We see that the results are improved significantly if we increase the sample size K from 1000 to 20000. Note that even when increasing the sample size by a factor 20, the computational time is still lower than the NN implementation. It should be highlighted that adding the function g to the neural network (as explained in Appendix C) is essential for its convergence in higher dimensions and thereby mitigates the observed difficulties in (Huré et al., 2020)).

	TT_{impl}	$\text{TT}^*_{\text{impl}}$	NN_{impl}
$\widehat{V}_0(x_0)$	-0.1887	-0.2136	-0.2137
relative error	1.22e^{-1}	6.11e^{-3}	5.50e^{-3}
ref loss	2.47e^{-1}	7.57e^{-2}	3.05e^{-1}
abs. ref loss	2.52e^{-2}	9.29e^{-3}	1.69e^{-2}
PDE loss	2.42	0.60	1.38
computation time	360	1778	4520

Table 9. Approximation results for the PDE with an unbounded analytic solution. For $\text{TT}^*_{\text{impl}}$ we choose $K = 20000$, for the others we choose $K = 1000$.

D.3. Allen-Cahn like equation

Finally, let us consider the following Allen-Cahn like PDE with a cubic nonlinearity in $d = 100$:

$$(\partial_t + \Delta)V(x,t) + V(x,t) - V^3(x,t) = 0, \quad (50a)$$
$$V(x,T) = g(x), \quad (50b)$$

where we choose $g(x) = \left(2 + \frac{2}{5}|x|^2\right)^{-1}$, $T = \frac{3}{10}$ and are interested in an evaluation at $x_0 = (0,\dots,0)^\top$. This problem has been considered in (E et al., 2017), where a reference solution of $V(x_0, 0) = 0.052802$ calculated by means of the branching diffusion method is provided. We consider a sample size of $K = 1000$ and a stepsize $\Delta t = 0.01$ and provide our approximation results in Table 10. Note that for this example it is again sufficient to use a TT-rank of 1 and a polynomial degree of 0.

	TT_{impl}	TT_{expl}	NN_{impl}	$\text{NN}^*_{\text{impl}}$
$\widehat{V}_0(x_0)$	0.052800	0.05256	0.04678	0.05176
relative error	4.75e^{-5}	4.65e^{-3}	1.14e^{-1}	1.97e^{-2}
PDE loss	2.40e^{-4}	2.57e^{-4}	9.08e^{-1}	6.92e^{-1}
comp. time	24	10	23010	95278

Table 10. Approximations for Allen-Cahn PDE, where $\text{NN}^*_{\text{impl}}$ uses $K = 8000$ and the others $K = 1000$ samples.

E. Some background on BSDEs and their numerical discretizations

BSDEs have been studied extensively in the last three decades and we refer to (Pardoux, 1998; Pham, 2009; Gobet, 2016; Zhang, 2017) for good introductions to the topic. Let us note that given some assumptions on the coefficients b, σ, h and g one can prove existence and uniqueness of a solution to the BSDE system as defined in (4) and (6), see for instance Theorem 4.3.1 in (Zhang, 2017).

We note that the standard BSDE system can be generalized to

$$\mathrm{d}X_s = \left(b(X_s,s) + v(X_s,s)\right)\mathrm{d}s + \sigma(X_s,s)\mathrm{d}W_s, \quad (51a)$$
$$X_0 = x, \quad (51b)$$
$$\mathrm{d}Y_s = (-h(X_s,s,Y_s,Z_s) + v(X_s,s)\cdot Z_s)\mathrm{d}s + Z_s\cdot \mathrm{d}W_s, \quad (51c)$$
$$Y_T = g(X_T), \quad (51d)$$

where $v : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ is any suitable control vector field that can be understood as pushing the forward trajectories into desired regions of the state space, noting that the relations

$$Y_s = V(X_s,s), \qquad Z_s = (\sigma^\top \nabla V)(X_s,s), \quad (52)$$

with $V : \mathbb{R}^d \times [0,T] \to \mathbb{R}$ being the solution to the parabolic PDE (1), hold true independent of the choice of v (Hartmann et al., 2019). Our algorithms readily transfer to this change in sampling the forward process by adapting the backward process and the corresponding loss functionals (10) and (11) accordingly.

In order to understand the different numerical discretization schemes in Section 2.1, let us note that we can write the backward process (5) in its integrated form for the times $t_n < t_{n+1}$ as

$$Y_{t_{n+1}} = Y_{t_n} - \int_{t_n}^{t_{n+1}} h(X_s,s,Y_s,Z_s)\mathrm{d}s + \int_{t_n}^{t_{n+1}} Z_s\cdot\mathrm{d}W_s. \quad (53)$$

In a discrete version we have to replace the integrals with suitable discretizations, where for the deterministic integral we can decide which endpoint to consider, leading to either of the following two discretization schemes

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_n \Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}, \tag{54a}$$
$$\widehat{Y}_{n+1} = \widehat{Y}_n - h_{n+1} \Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}, \tag{54b}$$

as defined in (8), where we recall the shorthands

$$h_n = h(\widehat{X}_n, t_n, \widehat{Y}_n, \widehat{Z}_n), \tag{55a}$$
$$h_{n+1} = h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1}). \tag{55b}$$

The L^2-projection scheme (10) can be motivated as follows. Consider the explicit discrete backward scheme as in (54b)

$$\widehat{Y}_{n+1} = \widehat{Y}_n - h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1})\Delta t + \widehat{Z}_n \cdot \xi_{n+1} \sqrt{\Delta t}. \tag{56}$$

Taking conditional expectations w.r.t. to the σ-algebra generated by the discrete Brownian motion at time step n, denoted by $\mathcal{F}_n$, yields

$$\widehat{Y}_n = \mathbb{E}\left[\widehat{Y}_{n+1} + h(\widehat{X}_{n+1}, t_{n+1}, \widehat{Y}_{n+1}, \widehat{Z}_{n+1})\Delta t \middle| \mathcal{F}_n\right]. \tag{57}$$

We can now recall that a conditional expectation can be characterized as a best approximation in L^2, namely

$$\mathbb{E}[B|\mathcal{F}_n] = \underset{\substack{Y \in L^2 \\ \mathcal{F}_n-\text{measurable}}}{\arg\min} \ \mathbb{E}\left[|Y - B|^2\right], \tag{58}$$

for any random variable $B \in L^2$, which brings

$$\widehat{Y}_n = \underset{\substack{Y \in L^2 \\ \mathcal{F}_n-\text{measurable}}}{\arg\min} \ \mathbb{E}\left[\left(Y - h_{n+1}\Delta t - \widehat{Y}_{n+1}\right)^2\right]. \tag{59}$$

This then yields the explicit scheme depicted in (10). We refer once more to (Gobet et al., 2005) for extensive numerical analysis, essentially showing that the proposed scheme is of order $\frac{1}{2}$ in the time step Δt.