Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-30 (2022) 168–173

Data-driven initialization of deep learning solvers for Hamilton-Jacobi-Bellman PDEs ⋆

A. Borovykh * **D. Kalise** ** **A. Laignelet** *** **P. Parpas** ***

* *Warwick Business School, University of Warwick, Coventry CV4 7AL, UK (e-mail: anastasia.borovykh@wbs.ac.uk)*

** *Department of Mathematics, Imperial College London, London SW7 2AZ, UK (e-mail: d.kalise-balza@imperial.ac.uk)*

*** *Department of Computing, Imperial College London, London SW7 2AZ, UK (e-mail: {alexis.laignelet18, panos.parpas}@imperial.ac.uk)*

Abstract: A deep learning approach for the approximation of the Hamilton-Jacobi-Bellman partial differential equation (HJB PDE) associated to the Nonlinear Quadratic Regulator (NLQR) problem. A state-dependent Riccati equation control law is first used to generate a gradient-augmented synthetic dataset for supervised learning. The resulting model becomes a warm start for the minimization of a loss function based on the residual of the HJB PDE. The combination of supervised learning and residual minimization avoids spurious solutions and mitigate the data inefficiency of a supervised learning-only approach. Numerical tests validate the different advantages of the proposed methodology.

Keywords: Hamilton-Jacobi-Bellman PDE, NLQR, supervised learning, residual minimization.

1. INTRODUCTION

Over the last years there has been remarkable progress on the solution of high-dimensional nonlinear optimal control problems, see e.g. Dolgov et al. (2021); Onken et al. (2021); Kunisch, Karl and Walter, Daniel (2021); Meng et al. (2022), partly fuelled by the effectiveness of deep learning methods for solving nonlinear PDEs including HJB equations (see e.g. Raissi et al. (2019); Sirignano and Spiliopoulos (2018); Han et al. (2018); Güler et al. (2019)). In its general form, *unsupervised* deep learning-based PDE solvers aim at minimizing a loss function in which the interior and boundary constraints from the PDE are incorporated (see e.g. eq. (10)). Certain challenges can still arise; the works of e.g. van der Meer et al. (2020); Wang et al. (2021) have shown that the accuracy of the solution obtained by the neural network heavily relies on the weighting between the different terms (interior, boundary) in the loss function. As also noted in van der Meer et al. (2020) a long training time can be required and the number of domain points one has to sample can become large. Furthermore, if the PDE solutions are non-unique, using solely the residual can lead to an incorrect solution; we show this for the linear HJB equation in Fig. 1. As an alternative, *supervised* learning methods can be employed (see e.g. Azmi et al. (2021); Liu et al. (2019); Nakamura-Zimmerer et al. (2021)) where an optimal control solver is used to generate the solution dataset and a machine learning model is then trained on this set of data in a supervised manner. Alternatively, in Albi et al. (2021) the solutions of the state-dependent Riccati equation (SDRE) (see e.g. Banks et al. (2007)) are used as the initial dataset. A downside of such supervised settings is the propagation of errors from the initial solver into the model learned by the neural network (see Fig. 5 where the discrepancy between the residual loss and the SDRE-generated solution is highlighted).

In this work we focus on the synthesis of feedback laws for optimal nonlinear stabilization problems, that is, by solving the HJB PDE associated to the NLQR problem. We propose a neural network-based approximation for the value function where the training is done using a two-step learning approach: in the first step (i) the parameters of the neural network are initialized on the state-value dataset obtained from solving the SDREs, and in the second step (ii) the residual coming from the PDE constraints is minimized. The benefits of this approach include:

(1) by first pre-training on the data, our method ensures the convergence to the correct solution; this is particularly relevant in a setting where the solution to the HJB PDE is non-unique (see e.g. Figure 6),
(2) by using the residual-minimization in the second learning step our method allows to mitigate the errors in the SDRE solution and allows to reach higher-accuracy solutions than data-only (see e.g. Figure 4),
(3) the two-step learning approach decreases the amount of data points required to reach a satisfactory accuracy compared to training on the dataset only (see e.g. Figure 4).

2. METHODOLOGY

2.1 Infinite horizon nonlinear optimal stabilization

Let $\mathbf{y}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ be the state and control signal, respectively, of a system represented in control-

⋆ DK was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/V04771X/1, EP/T024429/1, and EP/V025899/1.

10.1016/j.ifacol.2022.11.047

affine form

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(\mathbf{y}(t)) + \mathbf{g}(\mathbf{y}(t))\mathbf{u}(t), \quad \mathbf{y}(0) = \mathbf{x}\,. \tag{1}$$

We consider the following infinite horizon optimal control problem:

$$\min_{\mathbf{u}(\cdot)\in\mathcal{U}} J(\mathbf{u};\mathbf{x}) := \frac{1}{2}\int_0^\infty \mathbf{y}^\top Q\mathbf{y} + \mathbf{u}^\top R\mathbf{u}\, dt\,, \tag{2}$$

subject to (1), where the dynamics $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be continuously differentiable, and the running cost matrices are given by $Q \in \mathbb{R}^{n\times n}, Q \succeq 0$, and $R \in \mathbb{R}^{m\times m}, R \succ 0$. We set $\mathcal{U} \equiv L^2([0,\infty);\mathbb{R}^m)$.

The formulation above corresponds to the NLQR problem, which arises in the synthesis of optimal feedback laws for the stabilization of nonlinear dynamics. The solution of this problem follows standard dynamic programming argument, see e.g., Bardi et al. (1997). Defining the value function of the problem

$$V(\mathbf{x}) = \inf_{\mathbf{u}(\cdot)} J(\mathbf{u};\mathbf{x})\,, \tag{3}$$

the optimal control is expressed in feedback form as

$$\mathbf{u}^*(\mathbf{x}) = -R^{-1}\mathbf{g}(\mathbf{x})^\top \nabla V(\mathbf{x})\,, \tag{4}$$

where $V(\mathbf{x})$ satisfies the Hamilton-Jacobi-Bellman PDE

$$\begin{aligned}\mathcal{N}(\mathbf{x}, V) := -\frac{1}{2}\nabla V(\mathbf{x})^\top \mathbf{g}(\mathbf{x})R^{-1}\mathbf{g}^\top(\mathbf{x})\nabla V(\mathbf{x}) \\ + \nabla V(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} = 0\,,\end{aligned} \tag{5}$$

in addition to the boundary condition $V(\mathbf{0}) = 0$.

2.2 Neural network model

In this work, we are interested in approximating the value function using deep neural networks. For this, we consider a fully connected feedforward neural network given by

$$\hat{V}(\mathbf{x};\theta) = f_L \circ f_{L-1} \circ \cdots \circ f_1(\mathbf{x}) \tag{6}$$

where $\mathbf{x}$ is the model input, L is the number of layers and $f_i(\mathbf{x}) = \sigma(W^i\mathbf{x} + \mathbf{b}^i)$, where W^i and b^i are matrices and vectors of prescribed dimensions, and σ is a nonlinear activation function applied component-wise. The unknown parameters of the network $\{W^i, b^i\}_{i=1}^L$ are grouped in the vector θ. The neural network is trained upon a set of collocation points $\{\mathbf{x}^i\}_{i=1}^N$ sampled uniformly over a domain of interest. The training of the neural network in based on the two-step learning procedure presented in Section 2.4.

2.3 Synthetic data generation for supervised learning

In our two-step learning procedure $\hat{V}(\mathbf{x};\theta)$ is first trained following a supervised learning gradient-augmented approach, that is, upon a collection of values

$$\{\mathbf{x}^i, V(\mathbf{x}^i), \nabla V(\mathbf{x}^i)\}_{i=1}^{N_1}\,,$$

approximating pointwise valuations of the true value function and its gradient. This dataset is generated synthetically by resorting to the link between the value function of the control problem and optimality conditions. One possibility to generate the dataset in a finite horizon setting is to use Pontryagin's Maximum Principle to obtain first-order optimality conditions for (1)-(2) in the form of a two-point boundary value problem, separately for each initial condition $\mathbf{x}^i$. However, in the infinite horizon optimal stabilization problem setting studied in this work, such an interpretation is not readily available, as the two-point boundary value problem has an asymptotic terminal condition for $t \to \infty$. We therefore rely on the use of a state-dependent Riccati equation approach Banks et al. (2007) which provide a stabilizing feedback control through a sequential solution of the algebraic Riccati equations (ARE) along a trajectory.

State-Dependent Riccati Equation For nonlinear dynamics, using the state-dependent Riccati equation (SDRE) approach is an effective method to generate a suboptimal, yet stabilizing solution which can be interpreted as an approximation of the HJB PDE (5) (see e.g. Jones and Astolfi (2020); Albi et al. (2021)). This approach is based on an extended linearization of the dynamics, expressed in semilinear form $f(\mathbf{x}) = A(\mathbf{x})\mathbf{x}$. Without loss of generality let $V(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top P(\mathbf{x})\mathbf{x}$. Note that $\nabla V(\mathbf{x}) = P(\mathbf{x})\mathbf{x} + \phi(\mathbf{x})$ where $\phi \in \mathbb{R}^n$ is a 'discrepancy' term arising from the differentiation of the non-linearity in $P(\mathbf{x})$ and its entries are given by

$$\phi(\mathbf{x})_k = \sum_{i,j=1}^n x_i x_j \nabla_{x_k} P_{ij}(\mathbf{x})\,, \quad k = 1,\dots,n\,.$$

In the SDRE approach, we approximate $\nabla V(\mathbf{x}) \approx P(\mathbf{x})\mathbf{x}$ so that the HJB PDE for the NLQR simplifies to

$$\begin{aligned}-\frac{1}{2}P(\mathbf{x})g(\mathbf{x})R^{-1}g(\mathbf{x})^\top P(\mathbf{x}) + P(\mathbf{x})A(\mathbf{x}) \\ + A(\mathbf{x})^\top P(\mathbf{x}) + Q = 0,\end{aligned} \tag{7}$$

For a fixed $\mathbf{x}$, the above equation amounts to solving an *ARE*. This suggests an implementation in a model predictive control fashion. Specifically, given a current state $\bar{\mathbf{x}}$ we solve (7) for $P(\bar{\mathbf{x}})$ by fixing every operator. Then using (4) gives $\mathbf{u}(\mathbf{x}) = -g^\top(\bar{\mathbf{x}})P(\bar{\mathbf{x}})\mathbf{x}$ and consequently using this in the dynamics of (1) gives the next value of the state. Similarly as in the ARE setting, along a trajectory we generate data for $V(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top P(\mathbf{x})\mathbf{x}$ and $\nabla V(\mathbf{x}) = P(\mathbf{x})\mathbf{x}$. However, as long as $\phi(\mathbf{x}) \neq 0$, this is a suboptimal approximation to the solution of the HJB PDE. Through supervised learning, we train a neural network which is subsequently used as a warm start of the training of a neural network which will directly approximate the solution of the HJB PDE.

2.4 The training methodology

We propose a two-step learning approach. A first neural network is trained using supervised learning with a synthetic dataset generated from the SDRE approach. In a second step, a neural network, initialized with the parameters of the first step, is trained solely for minimizing the residual of the HJB PDE (5). The idea behind such a solver is that training first on the synthetic dataset allows the residual minimization to start with a set of weights closer to the basin of attraction of the correct solution. The residual minimisation is then necessary to reduce the approximation errors induced by the data generation from the SDRE approach.

Data-driven initialization The SDRE approach generates a gradient-augmented dataset $\{\mathbf{x}^i, V(\mathbf{x}^i), \nabla V(\mathbf{x}^i)\}_{i=1}^{N_1}$.

Note that the gradient information is generated at virtually no extra computational cost. Consider the following loss function,

$$\mathcal{L}^{\text{dat}}(\theta) = \lambda_1 \frac{1}{N_1}\sum_{i=1}^{N_1} ||V(\mathbf{x}^i) - \hat{V}(\mathbf{x}^i;\theta)||_2^2 + \lambda_2 \frac{1}{N_1}\sum_{i=1}^{N_1} ||\nabla V(\mathbf{x}^i) - \nabla \hat{V}(\mathbf{x}^i;\theta)||_2^2, \tag{8}$$

where N_1 is the number of data points generated using the SDRE solver and λ_1 and λ_2 are the weights associated to the different loss function terms. Our first step then amounts to initialising the neural network parameters by finding,

$$\theta^{\text{dat}} = \arg\min_\theta \mathcal{L}^{\text{dat}}(\theta). \tag{9}$$

We remark that $\nabla \hat{V}(\mathbf{x};\theta)$ is obtained by differentiating the neural network with respect to the input $\mathbf{x}$.

2.5 Residual loss function

Based on (5), the *residual* loss function is given by,

$$\mathcal{L}^{\text{res}}(\theta) = \frac{1}{N_2}\sum_{i=1}^{N_2} ||\mathcal{N}(\mathbf{x}^i, \hat{V}(\mathbf{x}^i;\theta))||_2^2, \tag{10}$$

where N_2 is the number of collocation points $\mathbf{x}^1, ..., \mathbf{x}^{N_2}$ which we sample on the interior of the domain. As before, the gradient terms in this loss function are obtained by differentiating the neural network output $\hat{V}$ with respect to state variables. After obtaining θ^{dat} from minimizing (8), these parameters are used as the initialization of the training of a second neural network based on the residual minimization:

$$\theta^{\text{res\_valid}} = \arg\min_\theta \mathcal{L}^{\text{res}}(\theta)\,. \tag{11}$$

2.6 Benefits of the two-step learning approach

We illustrate, using a toy example, the benefits of warm start for residual minimization. Consider the linear quadratic control problem with $A = I$, $B = I$, $Q = I$ and $R = 0.2I$ with $n = 2$. The solution P of the corresponding ARE is a diagonal matrix with elements $\alpha_1 \approx \pm 0.69$ and/or $\alpha_2 \approx \pm -0.29$, see Figure 1 for the different solutions $V(\mathbf{x})$. By the positive-definiteness assumption the only valid solution is $P = \alpha_1 I$. However, if a neural network was trained on the residual only, four global minima exist (the solutions for P with P's elements different combinations of the roots) and convergence towards the value function will depend on a correct initialization [1]. Using Xavier initialization Glorot and Bengio (2010) the initial weights are centred around zero and hence gradient descent is most likely to converge to the minimum at $(-0.29, -0.29)$ (see the left-hand side of Fig 2). To ensure we converge to the valid solution, we can enrich the loss function with a data-driven term, similar to $\mathcal{L}^{\text{dat}}(\theta)$; this term guides convergence to the correct solution at $(0.69, 0.69)$ (see the right-hand side of Fig. 2).

This toy example motivates our combined approach of data and residual loss minimization. Using a residual-only approach inherits the multiplicity of (wrong) solutions of the HJB PDE associated to the NLQR problem. The downside of using both data and residual terms in *a single* loss function is the problem of choosing the right weights between the residual and data term (van der Meer et al. (2020)). Even more crucially, in non-linear settings the approximate value function obtained from the SDRE dataset can significantly differ from the value function that satisfies the residual (see also Figure 5); in such a setting, minimizing the data term and the residual term *at the same time* is not possible. To ensure our framework is robust and efficient in nonlinear settings, we thus propose the two-step learning approach.

[1] Note that in dimension n, the number of possible solutions in this example grows to 2^n.

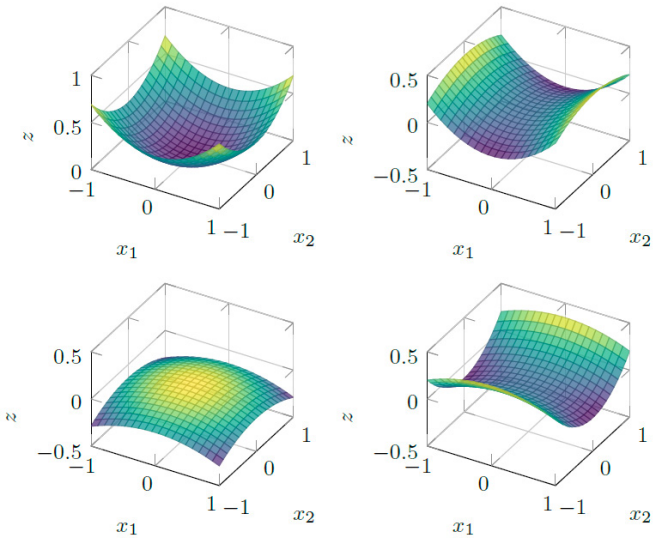

Fig. 1. Possible solutions of the Hamilton-Jacobi equation in the LQR case with $A = B = Q = I$, and $R = 0.2I$.

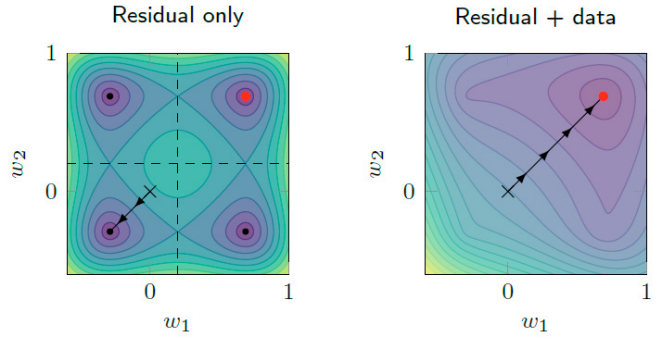

Fig. 2. Loss functions in the case of residual-only and a linear combination of the residual and the data term. The global minimum for the valid value function is represented by a red dot. A loss function combining the data and residual has a unique solution, while a residual-only approach inherits the multiple, but wrong, solutions of the original HJB PDE.

3. EXPERIMENTS

3.1 Implementation details

Neural network configuration and loss function The architecture we consider is a fully connected neural network

with the sigmoid activation function $\sigma(x) = 1/(1+e^{-x})$ with 3 hidden layers. For the two-dimensional problem we set 20 neurons per layer while for the 10-dimensional case we use 50 neurons per layer. The neural network is trained using the Adam optimizer Kingma and Ba (2014).

For the 2D examples, 20 SDRE data points are sampled for supervised learning and 50 collocation points are used to minimize the HJB PDE residual. The training procedure on the dataset is the following: we use 10^3 iterations with a learning rate of 0.01 followed by 10^3 iterations with a learning rate of 10^{-3}. When training simultaneously on the data and gradient we set $\lambda_1, \lambda_2 = 1$. For the residual minimization, the training is first done for 2×10^3 iterations with a learning rate of 0.01, followed by 4×10^3 iterations with a learning rate of 10^{-3}.

For the Cucker-Smale model, 500 data points are sampled, and 5×10^3 points are used for training on the residual. The pre-training uses 2×10^3 iterations with a learning rate of 0.01 followed by 4×10^3 iterations with 10^{-3}. The residual minimization is done using 10^4 iterations with a learning rate of 10^{-3} followed by 2×10^4 iterations with 10^4.

Data generation For the 2D examples, the ARE/SDRE is solved analytically to obtain a closed-form solution for $P(\mathbf{x})$ which is then used in the expressions for $V(\mathbf{x})$ and $\nabla V(\mathbf{x})$. In the case of the Cucker Smale model, the SDRE is solved numerically using the SciPy Linear Algebra package and $\nabla V(\mathbf{x}) = P(\mathbf{x})\mathbf{x}$ is used for the derivative. The collocation points are uniformly sampled in the space.

Evaluation The evaluation of the accuracy of the final solution is done by computing the mean squared error (MSE) between the neural network on newly generated interior domain points and the exact solution. For the 2D tests, the exact solution is obtained either through a closed-form solution, whenever available, or using the numerical solution of a semi-Lagrangian scheme over a fine grid Alla et al. (2015). The final MSE's are computed as the median over 10 runs of the optimization.

3.2 A 2D linear example

We revisit the same linear quadratic toy example discussed in Section 2.6. The left-hand side of Fig. 3 shows the MSE for three different training methods: i) training on value function data only, ii) training on data on the value function and the gradient, and iii) training on residual only. By minimizing over the residual only, the neural network converges to the incorrect solution (as was also shown in the toy example in Fig. 2); by incorporating the gradient of the value function into the loss, a more accurate solution can be obtained. In the left-hand side of Fig. 4 it is shown that two-step learning requires just 10 data points to obtain a solution that is of accuracy 10^{-6}, while when learning only over the data, one would require 100 data points to achieve the same error. Comparing the left-hand sides of Fig 2 and 4 we also note the two-step learning can converge to a solution with a smaller MSE than the data-only training. Finally, the left-hand side of Table 1 shows the MSE for the data-only (using value function and gradient) and two-step learning. It is clear that when

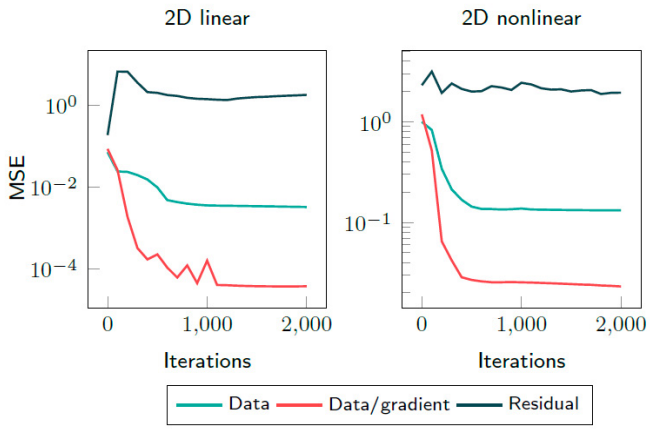

Fig. 3. Mean squared error of different training methods for 2D linear and 2D non linear case (computed over 100×100 points in the domain).

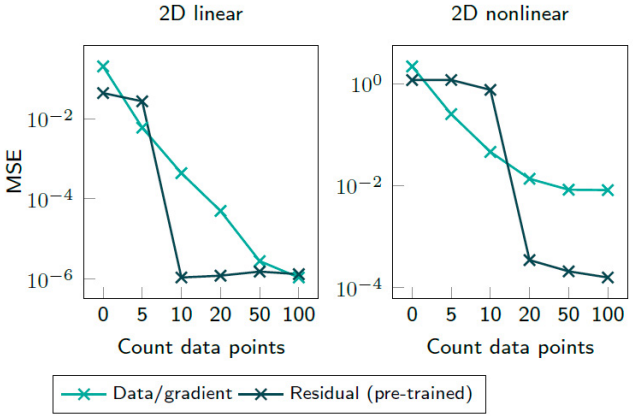

Fig. 4. Mean squared error between training on dataset and two step learning (computed over 100×100 points in the domain).

using more than 5 data points a more accurate solution is obtained using two-step learning.

Table 1. Mean squared error for 2D cases comparing the supervised learning on the dataset and the two steps learning.

	2D linear		2D non linear	
Point	Data/grad	2 steps	Data/grad	2 steps
0	$2.0 \cdot 10^{-1}$	$4.4 \cdot 10^{-2}$	2.2	1.2
5	$6.0 \cdot 10^{-3}$	$2.7 \cdot 10^{-2}$	$2.5 \cdot 10^{-1}$	1.2
10	$4.4 \cdot 10^{-4}$	$1.1 \cdot 10^{-6}$	$4.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-1}$
20	$5.0 \cdot 10^{-5}$	$1.2 \cdot 10^{-6}$	$1.4 \cdot 10^{-2}$	$3.4 \cdot 10^{-4}$
50	$2.8 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$8.3 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$
100	$1.1 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$	$8.1 \cdot 10^{-3}$	$1.6 \cdot 10^{-4}$

3.3 A 2D nonlinear example

We next consider the dynamics given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ \epsilon x_1^2 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)\,,$$

with $R = Q = I$. Note that the coefficient ϵ controls the intensity of the nonlinear term. The solution of the SDRE in this case is given by:

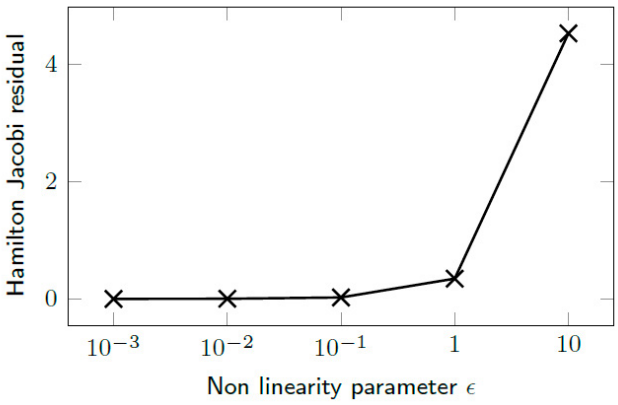

Fig. 5. Influence of the intensity of the non linearity induced by ϵ.

$$P = \begin{bmatrix} \sqrt{\epsilon^2 x_1^4 + 1}\sqrt{1 + 2P_{12}} & P_{12} \\ P_{12} & \sqrt{1 + 2P_{12}} \end{bmatrix}$$

with $P_{12} = \epsilon x_1^2 + \sqrt{\epsilon^2 x_1^4 + 1}$. Remember that the exact SDRE solution is an approximation of the original HJB PDE. In Fig. 5 we show the residual generated by the exact SDRE solution when inserted into the HJB PDE as a function of ϵ - this shows that the more non-linear the problem is, the more relevant the consequent minimization of the residual will be. This also shows why it is not possible to combine both (10) and (8) into a single objective: the SDRE loss term and the residual loss term lead to different solutions and minimizing both at the same time leads to a decrease in accuracy compared to the two-step learning approach. The right-hand side of Fig. 3 shows that using no data points and training on the residual only leads to inaccurate solutions, while training on the dataset only does not result in an error lower than 10^{-2} due to the SDRE solution being only an approximate solution. Combining data-driven initialisation with the residual minimization results in the most accurate solution as shown in the right-hand side of Fig. 4: training on 20 points first, and then minimizing the residual leads to a solution with a 10^{-4} accuracy. This level of accuracy is not obtained by increasing the amount of data points used in the supervised learning step. This highlights the necessity of the residual minimization to improve the accuracy of the final solution. The right-hand side of Table 1 validates these results. Finally, in Fig. 6 we present the results from training on the residual only (the left-hand side) and the two-step learning (the right-hand side). The residual-only training results in an invalid solution. Based on the value of the residual loss in (10) (the bottom plot) one could conclude that correct convergence has been obtained (the residual loss is sufficiently small); however as shown by the MSE computed by comparing the exact solution with the neural network output the network has converged to the incorrect solution. The value of two-step learning is clear from the right-hand side: the residual and MSE are both sufficiently low so that the solution obtained is close to the true solution.

3.4 Cucker-Smale model

We conclude with a high-dimensional nonlinear test consisting of agent-based consensus dynamics, known as the

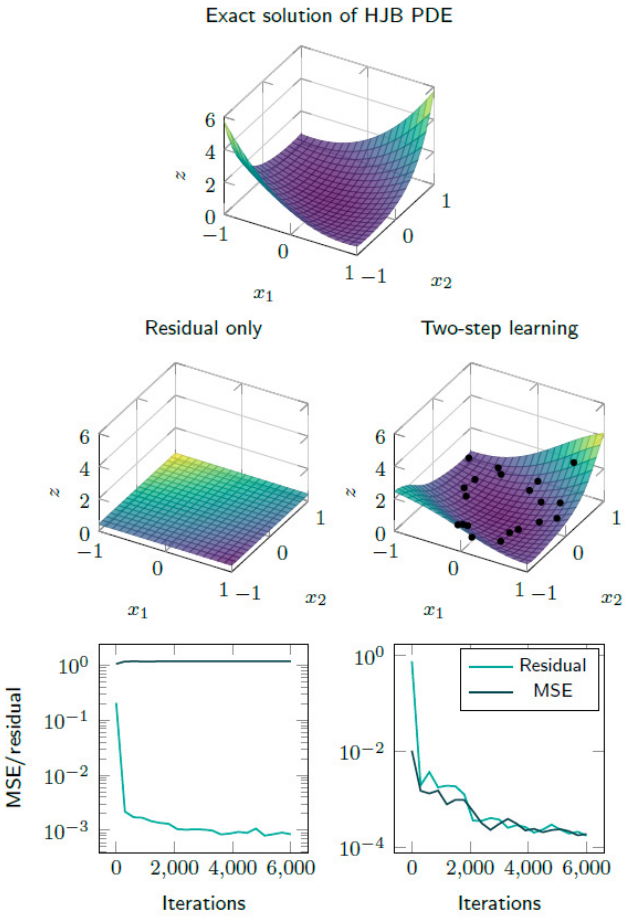

Fig. 6. Training the residual from a random initialisation (left) and starting from a pre-training on a dataset (right). From top to bottom: the initialisation, the solution after training, the true solution and the value of the residual loss in (10) and the MSE.

Cucker-Smale model. We consider 5 agents having states $\mathbf{x}_i = (y_i, v_i) \in \mathbb{R}^2$ in $[-3, 3]^{10} \in \mathbb{R}^5 \times \mathbb{R}^5$, with dynamics given in semilinear form by

$$\dot{\mathbf{x}} = \begin{bmatrix} O_5 & I_5 \\ O_5 & A(\mathbf{y}) \end{bmatrix} \mathbf{x} + \begin{bmatrix} O_5 \\ I_5 \end{bmatrix} \mathbf{u}\,, \tag{12}$$

where O_5 is an 5×5 matrix of zeros and I_5 is the 5×5 identity matrix and where

$$A(\mathbf{y})_{ij} = \begin{cases} -\frac{1}{5}\sum_{k=1}^{5} \frac{1}{1+||y_i - y_k||^2} & \text{for } i = j\,, \\ \frac{1}{5}\frac{1}{1+||y_i - y_j||^2} & \text{otherwise}\,. \end{cases} \tag{13}$$

We set $Q = \frac{1}{5} I_{10}$ and $R = I_5$. Note that the resulting HJB PDE is 10-dimensional. In Fig. 7 we present the results for the position, velocity and control for i) residual-only training, ii) data and gradient-only training and iii) two-step learning. As before, it can be seen that training on just the residual does not lead to the correct control. However, by first training on a dataset of 500 points, and then minimizing the residual, we are able to obtain a smooth stabilizing control. We remark that for this particular problem, 500 points is not enough for the supervised learning task, as in this data-only training the obtained control is not sufficiently smooth and many more points would be required to get a satisfying control

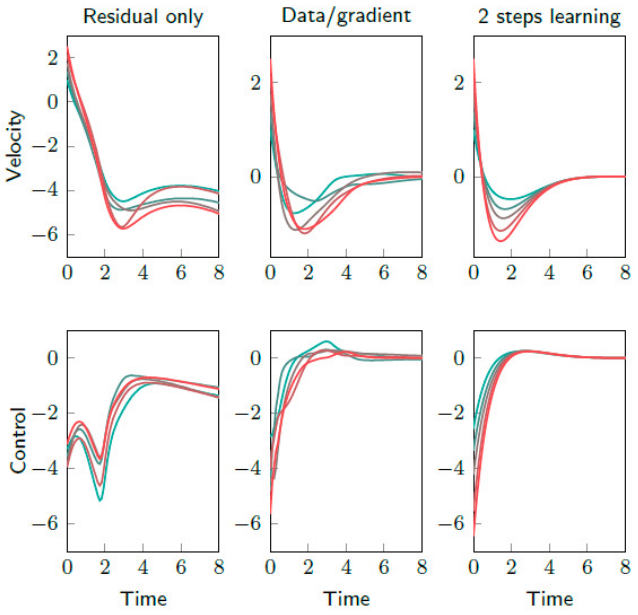

Fig. 7. Trajectories and velocities of a group of particles under the control found by the neural network after (left) dataset-only training on 500 points and (right) pre-training on 500 points & minimising the residual.

that generalises well on unseen data. This highlights the advantage of our combined approach.

Conclusion and discussion We have shown how a two-step learning approach can result in efficient and accurate solutions to the HJB PDE. In our setting, the output of the neural network was set to directly be the value function $V(\mathbf{x})$. However, as in the SDRE derivation, it is possible to assume that $V(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top P(\mathbf{x})\mathbf{x}$ and adapt the neural network architecture accordingly. Depending on the problem this can result in an efficient solver; we will address the choice of architecture for HJB PDE solvers in future work. Our results show that in a setting where the residual is sufficiently low for multiple solutions, the neural network converges to the one closest to initialization, which can be analyzed in the context of implicit bias Chizat and Bach (2020).

REFERENCES

Albi, G., Bicego, S., and Kalise, D. (2021). Gradient-augmented supervised learning of optimal feedback laws using state-dependent riccati equations. *IEEE Contr. Syst. Lett*, 6, 836–841.

Alla, A., Falcone, M., and Kalise, D. (2015). An efficient policy iteration algorithm for dynamic programming equations. *SIAM J. Sci. Comput.*, 37(1), A181–A200.

Azmi, B., Kalise, D., and Kunisch, K. (2021). Optimal feedback law recovery by gradient-augmented sparse polynomial regression. *J. Mach. Learn. Res.*, 22(48), 1–32.

Banks, H., Lewis, B., and Tran, H.T. (2007). Nonlinear feedback controllers and compensators: a state-dependent riccati equation approach. *Comput. Optim. Appl.*, 37(2), 177–218.

Bardi, M., Dolcetta, I.C., et al. (1997). *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer.

Chizat, L. and Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. ArXiv:2002.04486.

Dolgov, S., Kalise, D., and Kunisch, K.K. (2021). Tensor decomposition methods for high-dimensional hamilton–jacobi–bellman equations. *SIAM J. Sci. Comput.*, 43(3), A1625–A1650.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Güler, B., Laignelet, A., and Parpas, P. (2019). Towards robust and stable deep learning algorithms for forward backward stochastic differential equations. *arXiv:1910.11623*.

Han, J., Jentzen, A., and Weinan, E. (2018). Solving high-dimensional partial differential equations using deep learning. *Proc. Nat. Acad. Sci.*, 115(34), 8505–8510.

Jones, A. and Astolfi, A. (2020). On the solution of optimal control problems using parameterized state-dependent riccati equations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 1098–1103.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kunisch, Karl and Walter, Daniel (2021). Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation*. *ESAIM: COCV*, 27, 16.

Liu, S., Oosterlee, C.W., and Bohte, S.M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1), 16.

Meng, T., Zhang, Z., Darbon, J., and Karniadakis, G.E. (2022). Sympocnet: Solving optimal control problems with applications to high-dimensional multi-agent path planning problems. ArXiv:2201.05475.

Nakamura-Zimmerer, T., Gong, Q., and Kang, W. (2021). Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM J. Sci. Comput.*, 43(2), A1221–A1247.

Onken, D., Nurbekyan, L., Li, X., Fung, S.W., Osher, S., and Ruthotto, L. (2021). A neural network approach for real-time high-dimensional optimal control. ArXiv:2104.03270.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378, 686–707.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375, 1339–1364.

van der Meer, R., Oosterlee, C., and Borovykh, A. (2020). Optimally weighted loss functions for solving pdes with neural networks. *arXiv:2002.06269*.

Wang, S., Teng, Y., and Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5), A3055–A3081.