Gradient-augmented Supervised Learning of Optimal Feedback Laws Using State-dependent Riccati Equations

Giacomo Albi, Sara Bicego, and Dante Kalise

***Abstract*—A supervised learning approach for the solution of large-scale nonlinear stabilization problems is presented. A stabilizing feedback law is trained from a dataset generated from State-dependent Riccati Equation solvers. The training phase is enriched by the use of gradient information in the loss function, which is weighted through the use of hyperparameters. High-dimensional nonlinear stabilization tests demonstrate that real-time sequential large-scale Algebraic Riccati Equation solvers can be substituted by a suitably trained feedforward neural network.**

***Index Terms*—nonlinear feedback control, state-dependent Riccati equations, supervised learning**

I. INTRODUCTION

A large class of control problems in fluid flow control, consensus dynamics, and power networks, among many others, can be cast as optimal stabilization problems sharing two distinctive features in the dynamics: nonlinearity, and a high-dimensional state space. The natural control-theoretical framework to address these problems is through optimal stabilization using dynamic programming and Hamilton-Jacobi-Bellman (HJB) partial differential equations (PDEs). Unfortunately, the HJB PDE arising in nonlinear control is a first-order, fully nonlinear equation with no general explicit solution. Moreover, the overwhelming computational complexity associated to the solution of high-dimensional HJB PDEs poses a formidable challenge limiting the applicability of traditional grid-based computational methods to very low-dimensional control systems.

The numerical approximation of high-dimensional HJB PDEs arising in deterministic optimal control is a topic that has been the subject of extensive research. Without attempting an exhaustive literature review on the topic, effective computational approaches to this problem include the use of sparse grids [1], tree structure algorithms [2], max-plus methods [3], polynomial approximation [4], [5] and tensor decomposition methods [6], [7], [8], [9], [10]. These schemes are complemented with recent works making use of artificial neural networks [11], [12], [13], [14], [15].

In this paper, we propose a computational method for the solution of large–scale optimal stabilization problems for nonlinear dynamics avoiding the solution of the HJB PDE through a supervised learning approach. This idea dates back to [16], where the synthesis of feedback controls by interpolating finite horizon open-loop solvers was proposed. More recently, this problem has been studied in [17] using a sparse grid interpolant, in [18], [19] using deep neural networks, and in [20] through sparse polynomial regression. Similarly, the works [21], [22], [23] make use of representation formulas for HJB PDEs along with fast convex optimization solvers.

Many of the aforementioned works exploit the relation between the Hamilton-Jacobi-Bellman PDE and necessary optimality conditions through Pontryagin's Maximum Principle (PMP) in finite horizon control. Under convexity and smoothness assumptions, the PMP system represents the characteristic curves of the HJB PDE, and the value function of the problem can be computed at a given space-time point by solving a two-point boundary value problem. Unfortunately, such an interpretation is not readily available for infinite horizon optimal control, which is the case of interest for asymptotic stabilization of nonlinear dynamics.

Contributions

The methodology proposed in the present work circumvents the direct solution of the HJB PDE and the lack of PMP-like representation formula for the value function by resorting to State-dependent Riccati Equations (SDRE) [24], [25]. In the SDRE framework, after casting the nonlinear dynamics in semilinear form, a feedback control is obtained by a sequential solution of Algebraic Riccati Equations (ARE) along the trajectory. Under certain stabilizability conditions, this feedback law generates a locally asymptotically stable closed-loop and approximates the optimal feedback law from the HJB PDE. However, the main computational bottleneck of the SDRE approach is the availability of a sufficiently fast ARE solver to be called at an arbitrarily high rate. In this paper, we propose a methodology for real-time SDRE control of high-dimensional dynamics which does not require online ARE solves. Following a supervised learning approach, we train an artificial neural network that approximates the SDRE feedback law. The training is done in an offline phase. The resulting feedback law can be called in real-time scenarios at a very high rate, as its

DK was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/V04771X/1, EP/T024429/1, and EP/V025899/1. GA was supported by the PRIN Project 2017 (No. 2017KKJP4X entitled "Innovative numerical methods for evolutionary partial differential equations and applications").

G.A. and S. B. are with the Department of Computer Science, University of Verona, Strada le Grazie 15 - 37134 Verona, Italy (e-mail: giacomo.albi@univr.it, sara.bicego@studenti.univr.it)

D. K. is with the Department of Mathematics, Imperial College London, South Kensington Campus SW7 2AZ, United Kingdom (e-mail: dkaliseb@ic.ac.uk)

synthesis is reduced to the evaluation of a feedforward neural network. We provide two alternatives for training: learning the high-dimensional feedback map directly, or learning an approximation of the value function of the control problem, and recovering the feedback as a function of its gradient. We numerically assess that, through an adequate choice of network architectures, and including the use of gradient information of the model in the training, it is possible to accurately recover the SDRE feedback law for high-dimensional dynamics. To the best of our knowledge, this is the first work to propose the use of a supervised learning approach in conjunction with SDRE control for stabilization of high-dimensional dynamics.

The rest of the paper is organized as follows. In Section II we describe the nonlinear optimal stabilization problem and its solution, and in Section III we present the SDRE approach. In Section IV we discuss its numerical approximation through supervised learning, to continue in Section V with a computational assessment for two nonlinear, high-dimensional tests, and conclusions in Section VI.

II. Infinite Horizon Optimal Feedback Control

We study the design of feedback laws for asymptotic stabilization through infinite horizon optimal control:

$$\min_{\mathbf{u}(\cdot)\in\mathbf{U}} \mathcal{J}(\mathbf{u}(\cdot),\mathbf{x}_0) := \int_0^\infty \mathbf{x}^\top(s)\mathbf{Q}\mathbf{x}(s) \,+\, \mathbf{u}^\top(s)\mathbf{R}\mathbf{u}(s)\,ds\,, \tag{1}$$

subject to nonlinear, control-affine dynamics of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{B}(\mathbf{x}(t))\mathbf{u}(t)\,, \quad \mathbf{x}(0) = \mathbf{x}_0\,, \tag{2}$$

where $\mathbf{x}(t) = (x_1(t),\dots,x_n(t))^\top \in \mathbb{R}^n$ denotes the state of the system, $\mathbf{u}(\cdot) \in \mathbf{U} = \{\mathbf{u}(t) : \mathbb{R}^+ \to \mathbb{R}^m, \text{measurable}\}$ is an unbounded control variable, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix, and $\mathbf{R} \in \mathbb{R}^{m\times m}$ is symmetric positive definite. The system dynamics $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ and the control operator $\mathbf{B}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be $\mathcal{C}^1(\mathbb{R}^n)$ and, without loss of generality, such that $\mathbf{f}(\mathbf{0}) = \mathbf{0}$ and $\mathbf{B}(\mathbf{0}) = \mathbf{0}$. The optimal feedback law for the control problem (1) is synthesized using Dynamic Programming. For this, we define the value function of the control problem

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

which in turn satisfies a first-order, static, nonlinear Hamilton-Jacobi-Bellman PDE

$$\nabla V(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) - \frac{1}{4}\nabla V(\mathbf{x})^\top \mathbf{W}(\mathbf{x})\nabla V(\mathbf{x}) + \mathbf{x}^\top\mathbf{Q}\mathbf{x} = 0\,, \tag{4}$$

where $\mathbf{W}(\mathbf{x}) = \mathbf{B}(\mathbf{x})\mathbf{R}^{-1}\mathbf{B}(\mathbf{x})^\top$. After solving for $V(\mathbf{x})$, the optimal feedback is given by

$$\mathbf{u}(\mathbf{x}) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}(\mathbf{x})^\top \nabla V(\mathbf{x})\,. \tag{5}$$

The main difficulty when applying the dynamic programming approach to optimal feedback synthesis resides in the solution of the HJB PDE (4). This is a nonlinear PDE cast in the state-space of the system dynamics, with a dimension that can be arbitrarily high. Perhaps the most successful instance of a solution to this problem is the *linear quadratic regulator* (LQR), where, under the additional assumption that the free dynamics are linear, $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ and $\mathbf{B}(\mathbf{x}) = \mathbf{B}$, and making the ansatz $V(\mathbf{x}) = \mathbf{x}^\top\Pi\mathbf{x}$ with $\Pi \in \mathbb{R}^{n\times n}$ leads to

$$\mathbf{u}(\mathbf{x}) = -\mathbf{K}\mathbf{x} = -\mathbf{R}^{-1}\mathbf{B}^\top\Pi\mathbf{x}\,, \tag{6}$$

where Π is a positive definite solution of the Algebraic Riccati Equation (ARE)

$$\mathbf{A}^\top\Pi + \Pi\mathbf{A} - \Pi\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top\Pi + \mathbf{Q} = 0\,. \tag{7}$$

There are different methods which utilize the solution of the ARE above to generate a sub-optimal feedback control for local stabilization of nonlinear dynamics. Most notably, solving (7) with $(\mathbf{A},\mathbf{B}(\mathbf{0}))$ where $\mathbf{A}_{ij} = \frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j}|_{\mathbf{x}=0}$ leads to a linear feedback operator $\mathbf{K}_0$ which can effectively stabilize states in a vicinity of the origin. In the following, we discuss the synthesis of nonlinear feedback control laws by a sequential solution of AREs.

III. State-Dependent Riccati Equation

Having a representation of the nonlinear dynamics in semi-linear form

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})\mathbf{u}(t)\,, \tag{8}$$

we approximate the synthesis of the optimal feedback control following the State-dependent Riccati Equation (SDRE) approach. Formally, the solution of the nonlinear optimal control problem (1) is associated to an ARE where the operators are state-dependent

$$\mathbf{A}^\top(\mathbf{x})\Pi(\mathbf{x}) + \Pi(\mathbf{x})\mathbf{A}(\mathbf{x}) - \Pi(\mathbf{x})\mathbf{W}(\mathbf{x})\Pi(\mathbf{x}) + \mathbf{Q} = 0\,, \tag{9}$$

and analogously, the feedback (6) is also expressed through a state-dependent gain operator $\mathbf{K}(\mathbf{x})$

$$\mathbf{u}(\mathbf{x}) = -\mathbf{K}(\mathbf{x})\mathbf{x} = -\mathbf{R}^{-1}\mathbf{B}^\top(\mathbf{x})\Pi(\mathbf{x})\mathbf{x}\,. \tag{10}$$

Aiming at directly solving (9) for a general high-dimensional operator $\Pi(\mathbf{x})$ leads to the same difficulties already present in (4). Instead, we assume the operator $\Pi(\mathbf{x})$ is a positive definite matrix in $\mathbb{R}^{n\times n}$, so that for a fixed $\mathbf{x}$, solving (9) effectively reduces the problem to an ARE. We can benefit from this SDRE framework by applying it in a receding horizon fashion. Given a current state $\bar{\mathbf{x}}$ along a trajectory, we solve (9) for $\Pi(\bar{\mathbf{x}})$ by freezing every operator accordingly, recovering the feedback $\mathbf{u}(\mathbf{x}) = -\mathbf{K}(\bar{\mathbf{x}})\mathbf{x}$, to then evolve the controlled dynamics for a reduced time frame, after which we update the state of the system and recompute the feedback law. This approach leads to two natural questions: establishing conditions under which the SDRE approach generates an asymptotically stable closed-loop, and the design of effective computational methods for the fast solution of SDREs of potentially large scale. Regarding the first question, we recall the following proposition on asymptotic stability of the closed-loop generated by the SDRE approach [24].

Proposition 1: Assume a nonlinear system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{B}(\mathbf{x}(t))\mathbf{u}(t)\,, \tag{11}$$

where $\mathbf{f}(\mathbf{x})$ is $\mathcal{C}^1$ for $\|\mathbf{x}\| \le \delta$, and $\mathbf{B}(\mathbf{x})$ is continuous. If $\mathbf{f}(x)$ is parametrized in the form $\mathbf{f}(\mathbf{x}) = \mathbf{A}(\mathbf{x})\mathbf{x}$, and the

pair $(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x}))$ is stabilizable for every $\mathbf{x}$ in a non-empty neighbourhood of the origin $\Omega \subset \mathcal{B}_\delta(\mathbf{0})$, then the closed-loop dynamics generated by the feedback law (10) are locally asymptotically stable.

Assuming the stabilizability hypothesis above, the main bottleneck in the implementation of the SDRE approach is the availability of an ARE solver sufficiently fast for real-time feedback control. Here, we assume an ARE solver is readily available, however, it is not suitable for real-time control. In order to circumvent this difficulty, we follow a supervised learning approach, as we explain in the following section.

IV. Gradient-Augmented Supervised Learning for Optimal Feedback Laws

The SDRE (9) is solved offline for a set of training states, denoted by $\mathcal{X}_t$, which is used for training a suitable artificial neural network (ANN) which is then implemented for real-time control. The use of ANNs for SDREs has been explored for learning the matrix-valued operator $\Pi(\mathbf{x})$ in (9) with an unsupervised learning approach, see e.g. [26]. A natural drawback of learning $\Pi(\mathbf{x})$ is the dimension of the output, which amounts to $\frac{n^2}{2}$ entries. This is particularly demanding in a large-scale scenario, and can lead to high training errors despite the dataset being accurately generated. We report that for the tests presented in Section V, learning $\Pi(\mathbf{x})$ led to underperfomant model training. Here, we propose:

*a) **Learning** $\mathbf{u}(\mathbf{x})$:* we train a model for the vector-valued feedback law $\mathbf{u}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ upon a set of N_s training states $\mathcal{X}_t := \{\mathbf{x}^{(i)}\}_{i=1}^{Ns}$, the solution of the corresponding $\Pi(\mathbf{x})$, and the controls $\mathbf{u}(\mathbf{x})$ via (10).

*b) **Learning** $V(\mathbf{x})$:* we train a model for the scalar function $V(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ from $V(\mathbf{x}) = \mathbf{x}^\top \Pi(\mathbf{x})\mathbf{x}$ and its gradient $\nabla V(\mathbf{x}) = 2\Pi(\mathbf{x})\mathbf{x}$, where $\Pi(\mathbf{x})$ is a positive definite solution of (9) for each $\mathbf{x} \in \mathcal{X}_t$. The feedback law is then expressed as $\mathbf{u}(\mathbf{x}) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}(\mathbf{x})^\top \nabla V(\mathbf{x})$.

Both alternatives are a direct supervised learning formulation of the SDRE approach, with the sole objective of synthesizing a feedback requiring a reduced number of operations for online implementation. However, the second approach links the solution of the SDRE with finding a function $V(\mathbf{x})$ which approximates the solution of the original HJB equation (4). As discussed in [27], there is a direct equivalence between HJB, SDRE, and ARE in the linear-quadratic case. For the general nonlinear case, the ansatz $V(\mathbf{x}) = \mathbf{x}^\top \Pi(\mathbf{x})\mathbf{x}$ with $\Pi(\mathbf{x})$ generated from the SDRE approximates the solution of the HJB PDE only in neighbourhood of the origin. However, this idea is instrumental from a computational viewpoint. The advantage of the second formulation resides in the training of a scalar function, for which both function and gradient values are available. This shall be reflected in the choice of gradient-augmented loss functions for training.

1) Network architecture: The approximation task is carried out using feedforward neural networks (FNN), with information flowing from the input nodes to the output without generating any cycles or loops. FNNs approximate a function $f(\cdot)$ by a chain of compositions

$$f(\mathbf{x}) \approx f_\theta(\mathbf{x}) = l_M \circ ... \circ l_2 \circ l_1(\mathbf{x}), \tag{12}$$

where each layer l_m is defined as $l_m(\mathbf{y}) = \sigma_m(\mathbf{A}_m\mathbf{y} + \mathbf{b}_m)$, $\mathbf{A}_m$ are the weight matrices, $\mathbf{b}_m$ are the bias vectors and $\sigma_m(\cdot)$ are nonlinear *activation functions* applied component-wise. Standard choices for $\sigma(\cdot)$ are the ReLU function $\sigma(x) = max(0, x)$ and $\sigma(x) = tanh(x)$. The activation function in the hidden layers needs to be chosen accordingly with the valuation of the model's goodness of fit, and the last layer is typically assumed to be linear, thus $\sigma_M(x) = x$.

Considering a data set $\mathcal{T} = \{\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)})\}_{i=1}^{N_s}$, the NN is trained over the parameters $\theta = \{\mathbf{A}_m, \mathbf{b}_m\}_{m=1}^{M}$ to best approximate the target $f(\mathbf{x})$, i.e. minimizing the *loss* between the approximation $f_\theta(\mathbf{x}^{(i)})$ of the model and the true values $f(\mathbf{x}^{(i)})$ for every $\mathbf{x}^{(i)} \in \mathcal{T}$:

$$\min_\theta \ \mathcal{L}(f(\mathbf{x}), f_\theta(\mathbf{x})) \tag{13}$$

where the loss function $\mathcal{L}$ evaluates how well $f_\theta(\cdot)$ models the given dataset $\mathcal{T}$. The goodness of fit of f_θ within a set $\mathcal{T}' = \{\mathbf{x}^{(j)}\}_{j=1}^{N_v}$ can be measured by the *coefficient of determination*

$$r^2 = 1 - \frac{\sum_{j=1}^{N_v} \|f(\mathbf{x}^{(j)}) - f_\theta(\mathbf{x}^{(j)})\|^2}{\sum_{j=1}^{N_v} \|f(\mathbf{x}^{(j)}) - \bar{f}\|^2}, \tag{14}$$

where $\bar{f} = \frac{1}{N_v}\sum_{j=1}^{N_v} f(\mathbf{x}^{(j)})$. This coefficient typically ranges in $[0, 1]$; a value of 1 indicates that the model perfectly fits the data, while values below 0 suggest the trained model fits the data worse than a horizontal hyperplane.

We search for an approximation of the feedback control $\mathbf{u}(\mathbf{x})$, for which we consider two different approaches: to build a model $\mathbf{u}_\theta(\cdot)$ having $\mathbf{u}(\cdot)$ itself as target variable, $\mathbf{u}(\mathbf{x}) \approx \mathbf{u}_\theta(\mathbf{x})$, or to describe it through a FNN V_θ approximating $V(\cdot)$, on top of which we add a feedback layer

$$\mathbf{u}(x) \approx \mathbf{u}_V(x) = -\frac{R^{-1}B^T \nabla V_\theta(x)}{2}. \tag{15}$$

An accurate approximation of $\nabla V(\cdot)$ is essential for calculating a reasonable $\mathbf{u}_V(x)$. Here we deal with this through automatic differentiation, which allow us to compute exact gradients of V_θ in an efficient way. In this case, our training is not limited to pointwise valuations of $V(\mathbf{x})$, but also includes the discrepancy between the true gradient $\nabla V(\mathbf{x})$ and its approximation ∇V_θ. This is done choosing an ad hoc loss function V_θ.

2) Loss function: The training of the neural network for u_θ is done through a standard loss function: the mean squared error (MSE)

$$\mathcal{L}_0(\mathbf{u}, \mathbf{u}_\theta) := \frac{1}{N_s}\sum_{i=1}^{N_s} \|u(\mathbf{x}^{(i)}) - u_\theta(\mathbf{x}^{(i)})\|^2, \tag{16}$$

averaging the squared difference between approximation and actual observations.

For the training of V_θ, we consider instead

$$\mathcal{L}_1(V, V_\theta) = \mu_V \mathcal{L}_0(V, V_\theta) + \mu_{dV} \mathcal{L}_0(\nabla V, \nabla V_\theta)\,. \tag{17}$$

This loss function represents a compromise between the fitting functional $\mathcal{L}_0(V, V_\theta)$ and the gradient regulation $\mathcal{L}_0(\nabla V, \nabla V_\theta)$, suitably weighted thanks to μ_V and μ_{dV}.

V. Numerical Experiments

We assess the neural network approximation for feedback laws in two different tests. The control laws to be approximated rely on the pointwise solution of the SDRE (9), for which we resort to the `lqr` routine in MATLAB. The samples for training were generated by solving (1)-(2) for initial condition vectors $\mathcal{X}_t = \{\mathbf{x}^{(i)}\}_{i=1}^{N_s} \in \Omega \subset \mathbb{R}^n$, being populated using Halton quasi-random sequences in $[0,1]^n$.

Once the solution of the SDRE is computed for each sample $\mathbf{x}^{(i)} \in \mathcal{X}_t$, the training set $\{\mathbf{x}^{(i)}, \mathbf{u}(\mathbf{x}^{(i)})\}_{i=1}^{N_s}$ for u_θ can be computed as in (10), while the ANN V_θ is trained upon an enriched dataset, containing both the value function $V(\mathbf{x})$ and its gradient $\nabla V(\mathbf{x})$. Both these quantities can be obtained as a by-product of solving the SDRE at no additional computational cost since $V(\mathbf{x}) = \mathbf{x}^\top \Pi(\mathbf{x})\mathbf{x}$ and $\nabla V(\mathbf{x}) = 2\Pi(\mathbf{x})\mathbf{x}$. The sampling datasets are split into *training sets* and *valuation sets*, with a ratio of $80/20$. The goodness of fit in the valuation set, measured by the coefficient of determination r^2, guided the choice of the NN's architecture within the FNN family. The minimization of the loss function (13) was performed using the quasi-Newton method `lbfgs`. The parameters to be optimized are the number of hidden layers, the number of neurons per layer, the activation function, and the number of epochs taken into account during the training (we fixed the batches' size to 100). For $V_\theta(\mathbf{x})$, we also optimize the hyper-parameters μ_V and μ_{dV} weighting the terms in the loss function (17) by doing a grid search in the parametric space (μ_V, μ_{dV}). The goodness of fit of the trained models is finally evaluated in the *test set*, a uniform grid of $N_v = 10^4$ points within the state space, where the approximated control is compared with the pointwise computation through the SDRE solution. Goodness of fit of trained models in both tests are presented in Table I. Closed-loop simulations are performed with a RK45 solver with a timestep of 10^{-4}, determining the rate at which the ANN-based feedbacks are called.

	Test 1		Test 2	
Predicted variable	r^2	*MSE*	r^2	*MSE*
V_θ	0.67236	0.39829	0.81681	0.00025
∇V_θ	0.94921	0.07906	0.87114	0.00026
$\mathbf{u}_V$	0.92415	56.2208	0.91976	0.01218
$\mathbf{u}_\theta$	0.96039	29.3591	0.85443	0.02210

TABLE I
Goodness of fit for Tests 1 and 2.

A. Test 1: Stabilization for the Cucker-Smale model

We test our approach over a high-dimensional, nonlinear and nonlocal control problem related to consensus dynamics of agent-based Cucker-Smale model [28]. We consider $N_a = 20$ agents having states $\mathbf{x}_i = (y_i, v_i) \in \mathbb{R}^2$, denoting position and velocity respectively, in $\Omega = [-3,3]^{40} \subset \mathbb{R}^{20} \times \mathbb{R}^{20}$ and governed by the dynamics

$$\dot{y}_i = v_i\,, \quad \dot{v}_i = \frac{1}{N_a}\sum_{j=1}^{N_a} \frac{v_j - v_i}{1 + ||y_i - y_j||^2} + u_i\,, \tag{18}$$

where $i = 1, \dots, N_a$. Here, the control vector $\mathbf{u}(t)$ belongs to $\mathcal{L}^2([0,T]; \mathbb{R}^{N_a})$ and is optimized according to

$$\min_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{x}(\cdot)) = \frac{1}{N_a}\int_0^T \sum_{i=1}^{N_a} ||y_i||^2 + ||v_i||^2 + ||u_i||^2 dt \tag{19}$$

and it can be written in semilinear form as

$$\begin{bmatrix}\dot{\mathbf{y}}\\ \dot{\mathbf{v}}\end{bmatrix} = \begin{bmatrix}\mathbb{O}_{N_a} & \mathbb{I}_{N_a}\\ \mathbb{O}_{N_a} & \mathcal{A}_{N_a}(\mathbf{y})\end{bmatrix}\begin{bmatrix}\mathbf{y}\\ \mathbf{v}\end{bmatrix} + \begin{bmatrix}\mathbb{O}_{N_a}\\ \mathbb{I}_{N_a}\end{bmatrix}\mathbf{u}\,,$$

$$\big[\mathcal{A}(\mathbf{y})\big]_{i,j} = \begin{cases} -\frac{1}{N_a}\sum_{k=1}^{N_a} P(y_i, y_k) & \text{if } i = j\,,\\ \frac{1}{N_a} P(y_i, y_j) & \text{otherwise}\end{cases}$$

$$P(y_i, y_j) = \frac{1}{1 + ||y_i - y_j||^2}\,,\ \mathbf{Q} = \frac{1}{N_a}\mathbb{I}_{2N_a},\ \mathbf{R} = \frac{1}{N_a}\mathbb{I}_{N_a},$$

where $\mathbb{O}_n$ denotes a matrix of zeros in $\mathbb{R}^{n\times n}$. We train a model for V_θ consisting of a FNN with 3 hidden layers with 400 neurons per layer and activation function $\sigma(x) = \max(0, x)$. The best configuration resulting from hyper-parameter tuning was $(\mu_V, \mu_{dV}) = (0.1, 2)$, where the NN reaches the maximum r^2 being trained for 41 epochs, just before overfitting. Finally, applying the trained model to a grid of points in the hypercube $[-3,3]^{40}$, we compute the gradient of the model w.r.t. its input via automatic differentiation, computing the approximate control as in (15). The direct feedback model $\mathbf{u}_\theta(\mathbf{x}) \in \mathbb{R}^{N_a}$ consists of 2 hidden layers, with 400 neurons per layer, and activation function $\sigma(x) = tanh(x)$, while being trained for 20 epochs.

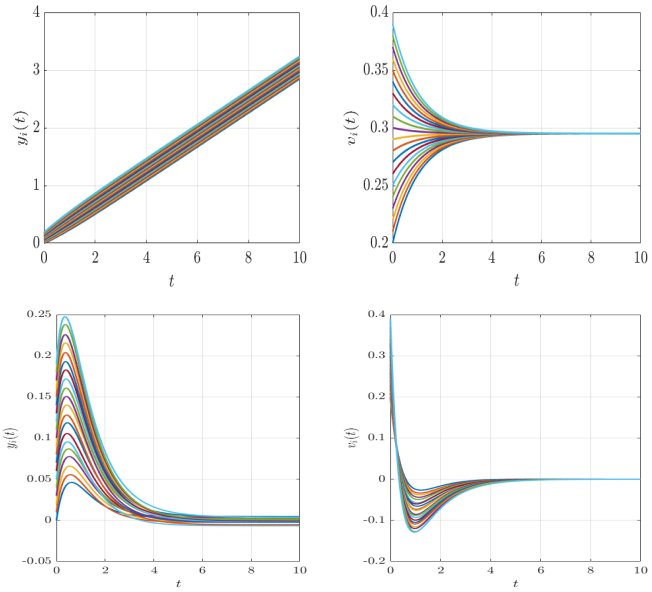

Fig. 1. Test 1. Top: uncontrolled positions (left) and velocities (right). Bottom: the controller $\mathbf{u}_\theta(\mathbf{x})$ stabilizes the dynamics (18) to the origin.

Figure 1 depicts trajectories for an initial condition $(\mathbf{y}(0), \mathbf{v}(0))$ that is a vector of equally spaced entries in $[0, 0.4]$. For this choice of dimension of the physical space and number of agents, the model $\mathbf{u}_\theta$ performs better than the gradient-augmented $\mathbf{u}_V$. The differences between both control signals can be observed in Figure 2.

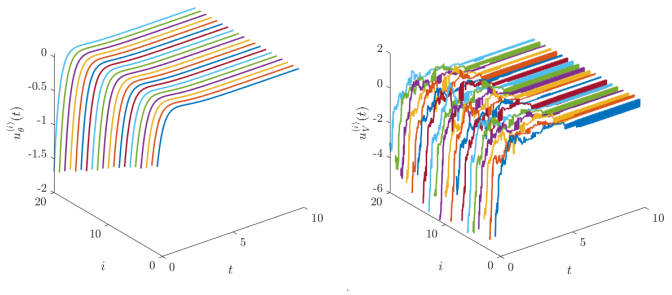

Fig. 2. Test 1. Control signals $\mathbf{u}_\theta(t)$ (left) and $\mathbf{u}_V(t)$ (right). In this nonlinear test, learning directly a model for $\mathbf{u}$ leads to better results. We omit the plot for $\mathbf{u}(t)$ since it is well approximated by $\mathbf{u}_\theta(t)$.

B. Test 2: Feedback control of the Allen-Cahn PDE

Following a test presented in [9], we consider the control of the nonlinear Allen-Cahn PDE

$$\partial_t x(\xi,t) = 0.1\partial^2_{\xi\xi} x + x(1-x^2) + \chi_\omega(\xi)u(t) \tag{20}$$

in $[0,1]\times\mathbb{R}_+$ with Neumann boundary conditions, where the scalar control signal $u:[0,+\infty]\to\mathbb{R}$ acts through the indicator function of the interval $\omega=[0.6,0.9]$. Without control action, these dynamics are bistable with $x\equiv\pm1$ being the stable equilibria. We are interested in minimizing

$$\mathcal{J}(u,x) = \int_0^{+\infty} ||x(\xi,t)||^2 + 0.1u^2(t)dt\,, \tag{21}$$

thus stabilizing the dynamics towards the equilibrium $x=0$. The PDE (20) is discretized in space via finite differences with $N=51$ nodes, leading to the nonlinear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{x}\odot(\mathbf{1}-\mathbf{x}\odot\mathbf{x}) + \mathbf{B}u(t), \tag{22}$$

where $\mathbf{x}(t) = (x(\xi_1,t),...,x(\xi_n,t))$ is the discrete state, $\odot$ denotes the Hadamard product and $\mathbf{A},\mathbf{B}$ correspond to a discretization of the Laplace operator and the indicator function $\chi_\omega(\xi)$ over a uniform grid $\{\xi_i\}_{i=1}^n$.

We consider a dataset $\{\mathbf{x}^{(i)}, V(\mathbf{x}^{(i)}), \nabla V(\mathbf{x}^{(i)})\}_{i=1}^{N_s}$ with $N_s = 1000$, where the states have been sampled from $[-2,2]^{51}$. We train a model for V_θ with 3 hidden layers, 500 neurons per layer, and activation function $\sigma(x)=\max(0,x)$. The best configuration of hyper-parameters is found to be $(\mu_V,\mu_{dV}) = (0.9,7)$, with the NN being trained for 71 epochs. Finally, we test the trained model V_θ in a test grid of points in $[-2,2]^{51}$. For the model $\mathbf{u}_\theta$, the architecture is built with 4 hidden layers, with 500 neurons per layer, and activation function $\sigma(x)=\max(0,x)$. The output layer for $\mathbf{u}_\theta$ is made only of a single neuron, since the feedback law is scalar. The model was trained for 50 epochs.

In Figs 3 and 4 we compare the trajectories resulting from the integration of the discretized dynamics (22) with $t\in[0,10]$, for an initial condition $x(\xi,0) = 1+(1-\xi)\xi$, and different feedback laws: the constant zero function, the feedback obtained considering the linear control operator $\mathbf{K}_0$, the control resulting from the gradient-augmented approximation V_θ, and the one given by $\mathbf{u}_\theta$. In this high-dimensional local problem, the approximation done through the gradient-augmented model V_θ happens to outperform $\mathbf{u}_\theta$ in terms of goodness of fit. On the other hand, observing the different closed-loop evolutions and control signals, we can see how both approximated feedback laws succeed in stabilizing the trajectories near $x=0$, while the uncontrolled system is stable in $x=1$ and the $\mathbf{u}_0$ results in a system which, for $t=10$ has not yet approached the equilibrium.

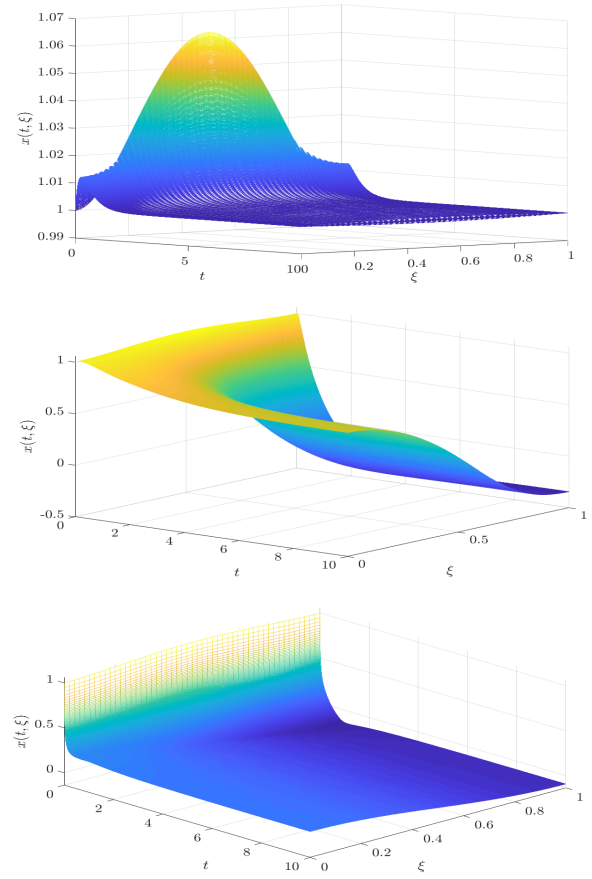

Fig. 3. Test 2. Top: uncontrolled state, converges to $x=1$. Middle: controlled state with linearized feedback $\mathbf{u}_0(\mathbf{x}) = -\mathbf{K}_0\mathbf{x}$ around the origin, fails to stabilize. Bottom: the nonlinear feedback $\mathbf{u}_V$ stabilizes the system to $x=0$.

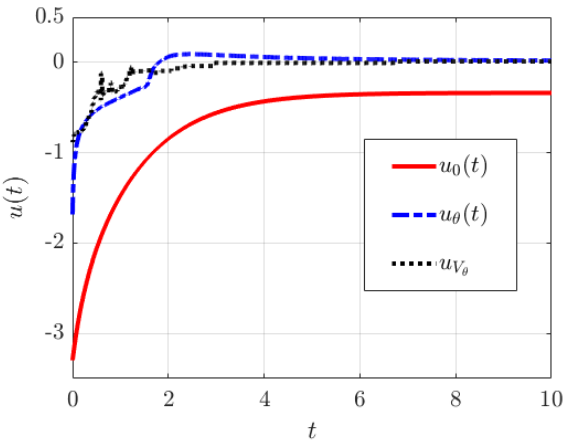

Fig. 4. Test 2. Different control signals: $\mathbf{u}_0$, $\mathbf{u}_\theta$ and $\mathbf{u}_V$. The feedback law trained with gradient-augmented data $\mathbf{u}_V$ outperforms $\mathbf{u}_\theta$.

VI. Conclusions

We have presented a novel computational method for the approximation of stabilizing feedback laws in nonlinear dynamics based on a supervised learning approach. The training data originates from the pointwise solution of the State-Dependent Riccati Equation. We have studied the approximation of the feedback control through feedforward neural networks, and analysed different choices of architectures and loss functions for training. We have provided computational evidence that for high-dimensional nonlinear problems, the SDRE feedback law can be effectively approximated through FNNs, thus removing the stringent requirement of a fast ARE solver for real-time closed-loop control. This is illustrated in Table II, where the runtime of an ARE solve is compared the evaluation time of the ANN-based controllers. As the number of states increases, the real-time computation of ARE solves becomes unfeasible, while the ANN-based feedbacks can be evaluated at reduced computational cost. We observe that for genuinely nonlinear control problems, such as agent-based dynamics, better results are achieved by learning directly the feedback $\mathbf{u}(\mathbf{x})$ from the SDRE solves. However, for problems where a linear structure is more prominent, such as in the control of semilinear parabolic PDEs, learning a model for a local approximation of the value function $V(\mathbf{x})$ and computing the control from its gradient is more accurate and efficient.

	Test 1			Test 2		
DoF	SDRE	u_V	u_θ	SDRE	u_V	u_θ
20	0.3715	0.0443	0.0375	0.3223	0.1004	0.0923
40	0.5157	0.0797	0.0637	0.3555	0.1143	0.0783
80	1.1382	0.0848	0.0585	0.4577	0.1016	0.0775
160	6.7549	0.1019	0.0927	1.0780	0.1000	0.0703
200	12.3	0.1393	0.0770	1.7496	0.12111	0.0901

TABLE II
AVERAGED ELAPSED TIME (SECONDS) WHEN COMPUTING A SINGLE ARE SOLUTION VERSUS EVALUATING ANN-BASED CONTROLLERS.

References

[1] J. Garcke and A. Kröner, "Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids," *J. Sci. Comput.*, vol. 70, no. 1, pp. 1–28, 2017.

[2] A. Alla, M. Falcone, and L. Saluzzi, "An efficient DP algorithm on a tree-structure for finite horizon optimal control problems," *SIAM J. Sci. Comput.*, vol. 41, no. 4, pp. A2384–A2406, 2019.

[3] M. Akian, S. Gaubert, and A. Lakhoua, "The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis," *SIAM J. Control Optim.*, vol. 47, no. 2, pp. 817–848, 2008.

[4] D. Kalise, S. Kundu, and K. Kunisch, "Robust feedback control of nonlinear PDEs by numerical approximation of high-dimensional Hamilton-Jacobi-Isaacs equations," *SIAM J. Appl. Dyn. Syst.*, vol. 19, no. 2, pp. 1496–1524, 2020.

[5] D. Kalise and K. Kunisch, "Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs," *SIAM J. Sci. Comput.*, vol. 40, no. 2, pp. A629–A652, 2018.

[6] M. B. Horowitz, A. Damle, and J. W. Burdick, "Linear Hamilton Jacobi Bellman equations in high dimensions," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 5880–5887.

[7] E. Stefansson and Y. P. Leong, "Sequential alternating least squares for solving high dimensional linear Hamilton-Jacobi-Bellman equation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3757–3764.

[8] A. Gorodetsky, S. Karaman, and Y. Marzouk, "High-dimensional stochastic optimal control using continuous tensor decompositions," *Int. J. Robot. Res.*, vol. 37, no. 2-3, pp. 340–377, 2018.

[9] S. Dolgov, D. Kalise, and K. Kunisch, "Tensor Decomposition Methods for High-dimensional Hamilton–Jacobi–Bellman Equations," *SIAM J. Sci. Comput.*, vol. 43, no. 3, pp. A1625–A1650, 2021.

[10] M. Oster, L. Sallandt, and R. Schneider, "Approximating the stationary Hamilton-Jacobi-Bellman equation by hierarchical tensor products," 2019, arXiv preprint:1911.00279.

[11] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *Proc. Natl. Acad. Sci. USA*, vol. 115, no. 34, pp. 8505–8510, 2018.

[12] J. Darbon, G. P. Langlois, and T. Meng, "Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures," *Res. Math. Sci.*, vol. 7, no. 20, 2020.

[13] N. Nüsken and L. Richter, "Solving high-dimensional Hamilton-Jacobi-Bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space," 2020, arXiv preprint:2005.05409.

[14] K. Ito, C. Reisinger, and Y. Zhang, "A Neural Network-Based Policy Iteration Algorithm with Global H^2-Superlinear Convergence for Stochastic Games on Domains," *Found. Comput. Math.*, vol. 21, pp. 331–374, 2021.

[15] K. Kunisch and D. Walter, "Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation," *ESAIM: COCV*, vol. 27, p. 16, 2021.

[16] S. C. Beeler, H. T. Tran, and H. T. Banks, "Feedback control methodologies for nonlinear systems," *J. Optim. Theory Appl.*, vol. 107, no. 1, pp. 1–33, 2000.

[17] W. Kang and L. C. Wilcox, "Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and HJB equations," *Comput. Optim. Appl.*, vol. 68, no. 2, pp. 289–315, 2017.

[18] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, "Adaptive Deep Learning for High-Dimensional Hamilton–Jacobi–Bellman Equations," *SIAM J. Sci. Comput.*, vol. 43, no. 2, pp. A1221–A1247, 2021.

[19] W. Kang, Q. Gong, and T. Nakamura-Zimmerer, "Algorithms of Data Development For Deep Learning and Feedback Design," 2019, arXiv preprint:1912.00492.

[20] B. Azmi, D. Kalise, and K. Kunisch, "Optimal feedback law recovery by gradient-augmented sparse polynomial regression," *J. Machin. Learn. Res.*, vol. 22, no. 48, pp. 1–32, 2021.

[21] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, "Algorithm for overcoming the curse of dimensionality for state-dependent Hamilton-Jacobi equations," *J. Comput. Phys.*, vol. 387, pp. 376–409, 2019.

[22] ——, "Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton-Jacobi equations arising from optimal control and differential games problems," *J. Sci. Comput.*, vol. 73, no. 2-3, pp. 617–643, 2017.

[23] J. Darbon and S. Osher, "Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere," *Res. Math. Sci.*, vol. 3, pp. Paper No. 19, 26, 2016.

[24] H. T. Banks, B. M. Lewis, and H. T. Tran, "Nonlinear feedback controllers and compensators: a state-dependent Riccati equation approach," *Comput. Optim. Appl.*, vol. 37, no. 2, pp. 177–218, Jun 2007.

[25] J. R. Cloutier, "State-dependent Riccati equation techniques: an overview," in *Proceedings of the 1997 American Control Conference*, vol. 2, 1997, pp. 932–936 vol.2.

[26] J. Wang and G. Wu, "A multilayer recurrent neural network for solving continuous-time algebraic Riccati equations," *Neural Networks*, vol. 11, no. 5, pp. 939–950, 1998.

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

[28] G. Albi, M. Herty, D. Kalise, and C. Segala, "Moment-driven predictive control of mean-field collective dynamics," *arXiv preprint arXiv:2101.01970*, 2021.