IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Robust Control of Unknown Observable Nonlinear Systems Solved as a Zero-Sum Game

Mircea-Bogdan Radac[1], Member, IEEE, and Timotei Lala[1]
[1]Department of Automation and Applied Informatics, Politehnica University of Timisoara, Romania

Corresponding author: Mircea-Bogdan Radac (e-mail: mircea.radac@upt.ro).

This work was supported by a grant from the Romanian Ministry of Education and Research, CNCS-UEFISCDI, project number PN-III-P1-1.1-TE-2019-1089, within PNCDI III

ABSTRACT An optimal robust control solution for general nonlinear systems with unknown but observable dynamics is advanced here. The underlying Hamilton-Jacobi-Isaacs (HJI) equation of the corresponding zero-sum two-player game (ZS-TP-G) is learned using a Q-learning-based approach employing only input-output system measurements, assuming system observability. An equivalent virtual state-space model is built from the system's input-output samples and it is shown that controlling the former implies controlling the latter. Since the existence of a saddle-point solution to the ZS-TP-G is assumed unverifiable, the solution is derived in terms of upper-optimal and lower-optimal controllers. The learning convergence is theoretically ensured while practical implementation is performed using neural networks that provide scalability to the control problem dimension and automatic feature selection. The learning strategy is checked on an active suspension system, a good candidate for the robust control problem with respect to road profile disturbance rejection.

INDEX TERMS active suspension system, approximate dynamic programming, neural networks, optimal control, reinforcement learning, state feedback, zero-sum two-player games

I. INTRODUCTION

Feedback control systems that are robust when faced with external disturbances are a common challenge and also frequently pose a direct or indirect design specification. To this end, the robust optimal control design is a highly attractive approach that has gained renewed attention lately in the zero-sum (ZS) game framework. Although the robust optimal design is, for quite some time now, well-posed for linear systems and solved by the H-infinity framework, it was not until the works by [1], [2] that the framework was imported to nonlinear systems robust optimal control in the form of the L2-gain optimal control. The goal of this latter formulation is to solve the Hamilton-Jacobi-Isaacs (HJI) equation, accepting the fact that for general nonlinear systems, it is often impossible to find analytical solutions. A very well-studied approach to the L2 control design is formulated as a ZS differential game [3]–[7] between two competing players: the optimal controller and the worst-case optimal disturbance controller. Whenever the HJI equation is feasible, these two players are the minimax saddle-point solution for it, and the feasibility of the HJI equation is well studied for linear systems and it depends on the attenuation level as a prescribed degree of performance [8]–[10]. Whereas, for general nonlinear systems, the common approach is to not directly assume the HJI equation feasibility and rather to try to search for its solution using the concepts of upper-optimal and lower-optimal controllers who act as upper and lower bounds for the optimal controller, respectively, when the HJI solution exists, or, they act as independent optimal solutions when an infeasible HJI exists [11], [12].

There exists a large number of solution approaches to the HJI equation, stemming from the methods employed by Approximate Dynamics Programming (ADP) also known as Reinforcement Learning (RL), for which ample research is very active [13]–[26]. These works from ADP have been applied, to name just a few, on discrete-time systems [27] vs. continuous-time systems, in known-system [28] or in unknown-system approaches [29], [30]. In game theory, [27] proposed an iterative ADP algorithm for solving the HJI equation related to ZS game problems for discrete-time affine nonlinear systems with known dynamics. In [28], ADP was derived to find the optimal saddle-point in feedback strategies of a nonlinear continuous-time system ZS game

with affine input and control constraints using two-player policy iterations. In [29] ADP solves a discrete-time zero-sum game for linear systems with continuous states using Q learning for unknown system. In [31], a near optimal solution based on successive approximation of HJI equation and disturbance inputs and control update was given for a discrete time affine nonlinear system subjected to unknown internal system dynamics and disturbances. In [32], an online adaptive real-time policy learning method based on zero-sum games for nonlinear discrete-time systems was proposed for learning the HJI equation. In [33], an online adaptive robust dynamic programming algorithm using policy iteration scheme for ZS-TP-G of continuous-time unknown systems subject to uncertainties was considered. In [34], a data-based adaptive critic method using output feedback for unknown model and system states was described under disturbance measurement assumption. In [35], a data-based policy iteration Q-learning algorithm for ZS-TP-G was developed for linear systems to eliminate process dynamics knowledge. Recent game-theoretical contributions (some in nonzero-sum games) for nonlinear systems are reported in [36]–[38], in the more general framework of robust control [39]–[40].

The first ADP-based solution approach for solving the HJI equation for unknown system dynamics is the Q-learning one [29] in which learning the HJI solution relies on data collected from the system to be controlled. In Q-learning, the learning process produces the two optimal controllers as state-feedback ones that must use the entire state information available. This is contradictory to some extent to the model-free label of the method itself, since knowing the natural system state requires a significant insight into the system, such as the system order and usually the nature of the underlying phenomenological process which is highly correlated with, e.g., the time-scale of the system. On the other hand, not using the entire state for learning optimal control poses great challenges to the learning process since the system is a partially observable one. This is the reason why some ADP approaches for solving the HJI ZS-TP-G were devised for handling observable systems and they rely only on input-output (IO) samples collected from the system. These IO samples are subsequently used to build a so-called virtual state that defines a virtual state-space model transformation of the original system. Unfortunately, the approach has been tackled for linear systems only, in a number of recent works [34], [41], [42] and not for general nonlinear systems, to the authors' best knowledge. This last remark serves as an incentive to one of this work's main contributions.

On another hand, designing off-the-shelf ADP techniques is another challenge since a great deal of effort is concerned with suitable parameterization of the nonlinear cost function and of the controllers, respectively. Most often, automatic basis function selection is a difficult task. Whereas, for observable systems whose state is built from past IO samples, the order of the equivalent virtual state quickly increases, which creates a two-fold problem: the adequate exploration of the input-state-output space and the time-correlation of the successive input and output samples [43]. These issues are related to the so-called dimensionality disaster problem which is well-known to the ADP and RL methods. This is why neural networks (NNs) have been the most flexible tool employed until now for parameterizing function approximators. Their main advantage is the self-regulated scalability to the control problem dimension, approximation capacity boosted by complex architectures and overfitting avoidance mechanisms intrinsically embodied in the NNs training procedures. Therefore, the NNs are considered to be a standard tool in ADP that can automate the basis functions (or features) selection in the function approximation tasks that are mandatory with the ADP approaches.

Based on the above ideas, the goal of this paper is to integrate several concepts into a fully functional approach to designing robust control for observable general unknown nonlinear systems. The contributions of this work are as follows:

- extension of the Q-learning approach to solve the optimal robust control problem as a ZS-TP-G solution to the HJI equation of general unknown nonlinear observable systems. An equivalent virtual state-space model of the original system is built from IO samples and subjected to robust control learning. The approach does not assume that a solution to HJI is feasible, therefore it searches for one via the computation of the intermediate upper-optimal and lower-optimal controllers. Theoretical analysis provides convergence of the proposed Q-learning-based solution.
- a NN-based implementation that proves scalability to the control problem dimension and automatic feature selection, in spite of the highly-dimensional virtual state vector.
- validation on a nonlinear industrial system of practical importance: an active suspension system.

The active suspension system is a well-suited candidate for learning robust control since it inherently deals with the road profile disturbance rejection when employed on a variety of transportation vehicles (cars, trains, etc.) and it presents itself as a naturally underdamped system stemming from the two-mass-spring-damper class of systems. On another hand, the suspension system is a high order one (it has six natural states when the active hydraulic actuator dynamics is considered) and it makes it costly to measure all states. Hence it makes a good candidate for an observable system. Fortunately, it turns out to be a fully observable one where the virtual state can be constructed from present and past values of only one output measurement (the deviation of the "car body" from the rest position) and from past values of the two inputs: the control input of the actuator and the disturbance input). Since measuring the road disturbance input is not a valid option in practice, a solution is offered to this problem, which proves that better attenuation of the road

profile impact on the "car body" motion is achieved than with respect to a competitive optimal controller which is not learned with disturbance rejection ability in mind. While there exists a consistent body of scientific literature dealing with the optimal active suspension control and in particular in that of reinforcement learning applied for the suspension control [44]–[48], none of the above solutions deal with the nonlinear unknown-dynamics observable case, as done herein.

There are several advantages of learning a disturbance rejecting optimal controller for an active suspension system:

- avoidance of the system dynamics knowledge,
- the observability-based solution that requires only IO samples to reconstruct the system state,
- artificial disturbances that emulate road conditions are easily generated in fixed stands in a learning facility.

The paper structure is oriented as follows. The second Section defines the ZS robust optimal control problem formulation and proposes a Q-learning-based solution employing upper-optimal and lower-optimal controllers. Theoretical learning convergence analysis is performed. Section III describes the practical implementation of the proposed learning strategy, under neural networks used as function approximators. The case study in Section IV extensively validates the learning concept on a realistic quarter-car active suspension model and provides discussions and implementation details. Final conclusions are the subject of the final Section V.

II. THE ZS ROBUST OPTIMAL CONTROL PROBLEM DEFINITION AND SOLUTION

A. THE UNKNOWN OBSERVABLE SYSTEM

Let the nonlinear unknown system

$$S:\begin{cases}\mathbf{x}_{k+1}=\mathbf{f}(\mathbf{x}_k,\bar{\mathbf{u}}_k),\\ \mathbf{y}_k=\mathbf{h}(\mathbf{x}_k),\end{cases} \tag{1}$$

defined in discrete-time, comprise of a transition function equation and output equation respectively. The system state is $\mathbf{x}_k=[x_{k,1}...x_{k,n}]^T\in\Omega_X\subset\Re^n$, the system's input is $\bar{\mathbf{u}}_k=[\mathbf{u}_k{}^T,\mathbf{d}_k{}^T]^T\in\Omega_{\bar{U}}\subset\Re^m,\mathbf{u}_k\in R^{m_u},\mathbf{d}_k\in R^{m_d},m=m_u+m_d$, the control input is $\mathbf{u}_k=[u_{k,1},...,u_{k,m_u}]^T\in\Omega_U\subset\Re^{m_u}$, the disturbance input is $\mathbf{d}_k=[d_{k,1},...,d_{k,m_d}]^T\in\Omega_D\subset\Re^{m_d}$, the measured (and controlled) output is denoted $\mathbf{y}_k=[y_{k,1},...,y_{k,p}]^T\in\Omega_Y\subset\Re^p$. The functions $\mathbf{f},\mathbf{h}$ are assumed unknown on their definition domains and also continuously differentiable. In addition to unknown system dynamics, further system assumptions are listed:

A1. System (1) is completely state observable.

A2. System (1) is IO controllable from $\mathbf{u}_k$ to $\mathbf{y}_k$.

A3. System (1) is IO stable inside the domain defined by the input and output.

A1–A3 are common for defining control problems for systems with unknown dynamics. Since they are not verifiable due to unknown system model, they are validated from working experience with the system, or from technical datasheets. Assessment efforts of linear systems' controllability and observability was proposed e.g. in the works [49], [50].

Observation 1. The input vector lumping both the control inputs and the disturbance inputs is important for deriving the two-player formulation of the optimal robust control solution.

In the attempt to derive state-feedback controllers, the state in (1) is not measurable. The observability theory allows to derive an alternate state-space model for (1) in terms of a virtual state. The support for this claim is given as follows.

Lemma 1. If pair $(\mathbf{f},\mathbf{h})$ in (1) is observable, then there exists a map $\mathbf{\Phi}$ and a positive integer r such that

$$\begin{aligned}&\mathbf{x}_k=\mathbf{\Phi}(\mathbf{Y}_{k,k-r},\mathbf{U}_{k-1,k-r}),\\ &\mathbf{Y}_{k,k-r}=[(\mathbf{y}_k)^T...(\mathbf{y}_{k-r})^T]^T,\mathbf{U}_{k-1,k-r}=[(\bar{\mathbf{u}}_{k-1})^T...(\bar{\mathbf{u}}_{k-r})^T]^T.\end{aligned} \tag{2}$$

Proof: See [51].

A virtual state vector is next introduced as $\mathbf{z}_k=[(\mathbf{Y}_{k,k-r})^T,(\mathbf{U}_{k-1,k-r})^T]^T\in\Omega_Z\subset R^{p(r+1)+mr}$. Then, *Theorem 1* in [51] showed that, based on (1) and (2), a new virtual state-space system with output equation is defined as

$$\begin{aligned}&\mathbf{z}_{k+1}=\mathbf{F}(\mathbf{z}_k,\bar{\mathbf{u}}_k),\\ &\mathbf{y}_k=\mathbf{z}_{k,1},\end{aligned} \tag{3}$$

which is completely observable (the components of $\mathbf{z}_k$ are sequences built from current and past successive IO samples) and controllable (since it has the same input and output as (1)).

The summarized ideas from [51] are:

a) System (3) is IO controllable since it has the same input and output as (1);

b) With unknown state dimension *n* in (1), *r* from *Lemma 1* corresponds to an observability index and it is also unknown. Increasing *r* (and, subsequently the dimension of the virtual state $\mathbf{z}_k$) is the general advice. As [51] shows, beyond some value of *r*, no information gain about the state $\mathbf{x}_k$ is obtained from $\mathbf{z}_k$;

c) Controlling (1) and (3) is the same issue, except that (3) uses a "measurable" state information. It means that learning control for (3) will render the control for (1);

d) Model (1) accommodates a wide range of processes, including time-delay ones. By properly introduced additional state variables and via variables substitutions, the time delay in the control input and in the states will result in another virtual state-space model (3) that is fully state observable and controllable (*Comment 7* from [51]).

e) When learning state feedback controllers of the form $\bar{\mathbf{u}}_k=\bar{C}(\mathbf{z}_k)$ (with some function $\bar{C}:\Omega_Z\to\Omega_U$), note that when plugging in this controller to close the control system

loop, a recurrent controller emerges, since $\mathbf{z}_k$ includes past samples of the input $\bar{\mathbf{u}}_k$. This type of recurrent controller is known as a nonlinear output error (NOE) model.

In order to develop the control solution in terms of disturbance rejection, the input $\bar{\mathbf{u}}_k$ is split to show distinctively the control input and the disturbance input (*Observation 1*) as $\bar{\mathbf{u}}_k = [\mathbf{u}_k{}^T, \mathbf{d}_k{}^T]^T, \mathbf{u}_k \in R^{m_u}, \mathbf{d}_k \in R^{m_d}$. The final virtual state space model is

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathbf{F}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k), \\ \mathbf{y}_k &= \mathbf{z}_{k,1}. \end{aligned} \tag{4}$$

Next, the optimal robust control problem of (4) is formulated as a ZS-TP-G.

B. THE ZS CONTROL PROBLEM DEFINITION AND SOLUTION

The goal is to minimize a certain cost serving as performance index, with the optimization problem of the optimal control problem defined as

$$C^*, D^* = \arg\min_C \max_D J(\mathbf{z}_k) = \sum_{j=k}^{\infty} U(\mathbf{z}_j, \mathbf{u}_j, \mathbf{d}_j), \tag{5}$$

with $\mathbf{u}_j = C(\mathbf{z}_j), C: \Omega_Z \to \Omega_u$ is the state feedback controller w.r.t. input $\mathbf{u}_k$ and $\mathbf{d}_j = D(\mathbf{z}_j), D: \Omega_z \to \Omega_d$ is the state feedback disturbance controller w.r.t. to input $\mathbf{d}_k$. Here, $\Omega_Z, \Omega_u, \Omega_d$ are domain spaces of appropriate dimension. In (5), the penalty function U is of the form $U(\mathbf{z}_j, \mathbf{u}_j, \mathbf{d}_j) = \Theta(\mathbf{z}_j) + C(\mathbf{z}_j)^T \mathbf{W}_C C(\mathbf{z}_j) - D(\mathbf{z}_j)^T \mathbf{W}_D D(\mathbf{z}_j)$ where $\Theta(\mathbf{z}_j) > 0 \in R$ is a state penalty term capturing the desired learning goal (regulation or tracking w.r.t. the state trajectory $\mathbf{z}_k$), and $\mathbf{W}_C, \mathbf{W}_D$ are square positive definite weight matrices. A controller pair $\{C, D\}$ is called *admissible* if it renders a finite cost J in (5) and it stabilizes the closed-loop control system. Minimization of the cost from (5) is interpreted as a degree of attenuation achieved by the control system faced with any disturbance $\mathbf{d}_k \in \Omega_d$, when only the optimal controller $C^*(\mathbf{z}_k)$ is used in closed-loop.

Definition 1. [12] In the existence domains spaces of the controllers C, D, the optimal controllers C^*, D^* are a saddle-point solution for the ZS-TP-G (5) if, for all C, D, it holds that

$$\begin{aligned} J(\mathbf{z}_k, C^*(\mathbf{z}_k), D(\mathbf{z}_k)) \le J^*(\mathbf{z}_k) \overset{\Delta}{=} J^*(\mathbf{z}_k, C^*(\mathbf{z}_k), D^*(\mathbf{z}_k)) \le \\ \le J(\mathbf{z}_k, C(\mathbf{z}_k), D^*(\mathbf{z}_k)). \end{aligned} \tag{6}$$

For general nonlinear systems with intractable analytical solutions for (5) and moreover, for those nonlinear systems with unknown dynamics, the existence of a saddle-point equilibrium is not guaranteed, as pointed out in [12]. In this sense, according to [2], upper-optimal and lower-optimal costs were introduced as

$$\begin{aligned} \bar{J}^*(\mathbf{z}_k) &= \min_C \max_D J(\mathbf{z}_k, C(\mathbf{z}_k), D(\mathbf{z}_k)), \\ \underline{J}^*(\mathbf{z}_k) &= \max_D \min_C J(\mathbf{z}_k, C(\mathbf{z}_k), D(\mathbf{z}_k)). \end{aligned} \tag{7}$$

These upper-optimal and lower-optimal costs ensure that $\underline{J}^*(\mathbf{z}_k) = J^*(\mathbf{z}_k) = \bar{J}^*(\mathbf{z}_k)$ when the saddle-point solution $J^*(\mathbf{z}_k)$ exists and also that $\underline{J}^*(\mathbf{z}_k) \le \bar{J}^*(\mathbf{z}_k)$ when such a solution is not feasible. Moreover, $\underline{J}^*(\mathbf{z}_k), \bar{J}^*(\mathbf{z}_k)$ satisfy the Hamilton-Jacobi-Isaacs optimality equations which suggests using iterative ADP solutions to overcome the difficulty of calculating the upper optimal and lower optimal costs for general nonlinear systems.

Notice that when the saddle-point solution (5) does not exist, the optimal controllers from (7) differ, i.e. $\bar{C}^*(\mathbf{z}_k), \bar{D}^*(\mathbf{z}_k) = \arg\min_C \max_D J(\mathbf{z}_k, C(\mathbf{z}_k), D(\mathbf{z}_k))$ differ from $\underline{C}^*(\mathbf{z}_k), \underline{D}^*(\mathbf{z}_k) = \arg\max_D \min_C J(\mathbf{z}_k, C(\mathbf{z}_k), D(\mathbf{z}_k))$.

To compute the optimal controllers from (7) for both upper and lower costs, upper and lower extended costs called Q-functions are defined as

$$\begin{aligned} \bar{Q}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) &= U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \bar{Q}(\mathbf{z}_{k+1}, \mathbf{u}_{k+1} = C(\mathbf{z}_{k+1}), \mathbf{d}_{k+1} = D(\mathbf{z}_{k+1})) \\ &= U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \bar{J}(\mathbf{z}_{k+1}), \\ \underline{Q}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) &= U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \underline{Q}(\mathbf{z}_{k+1}, \mathbf{u}_{k+1} = C(\mathbf{z}_{k+1}), \mathbf{d}_{k+1} = D(\mathbf{z}_{k+1})) \\ &= U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \underline{J}(\mathbf{z}_{k+1}), \end{aligned} \tag{8}$$

having the well-known meaning: they are the cost of taking any action ($\mathbf{u}_k$, $\mathbf{d}_k$) in state $\mathbf{z}_k$ and afterwards acting only subject to controller actions calculated by C and D in all subsequent states. They are directly connected to the original upper and lower costs J as shown in (8). The advantage of such Q-functions is that the optimal controllers are computable by directly minimizing w.r.t. $\mathbf{u}_k$ and $\mathbf{d}_k$ the upper optimal and lower-optimal Q-functions $\bar{Q}^*(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k), \underline{Q}^*(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k)$, once these are found.

Value Iteration (VI)-like algorithms are next proposed to calculate the upper-optimal and lower-optimal Q-functions. Their style is similar. For the upper optimal Q-function calculation, the VI *Algorithm 1* is as follows.

Algorithm 1. Starting from initial (not necessarily admissible) controllers $\bar{C}_0, \bar{D}_0$, and an initial upper Q-function estimate $\bar{Q}_0$, for all the possible combinations of the tuple $(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k)$, alternate the following two steps at each iteration j (starting with j=1):

S1. Update the Q-function as

$$\begin{aligned} \bar{Q}_j(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) \Leftarrow U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \\ \bar{Q}_{j-1}(\mathbf{z}_{k+1}, \mathbf{u}_{k+1} = \bar{C}_{j-1}(\mathbf{z}_{k+1}), \mathbf{d}_{k+1} = \bar{D}_{j-1}(\mathbf{z}_{k+1})) \end{aligned} \tag{9}$$

S2. Improve the controllers as in

$$\overline{D}_j(\mathbf{z}_k,\mathbf{u}_k)=\arg\max_{\mathbf{d}}\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}),$$
$$\overline{C}_j(\mathbf{z}_k)=\arg\min_{\mathbf{u}}\overline{Q}_j(\mathbf{z}_k,\mathbf{u},\overline{D}_j(\mathbf{z}_k,\mathbf{u})). \quad (10)$$

S3. If stopping criterion (no more changes from $\overline{Q}_{j-1}$ to $\overline{Q}_j$) is not met, go to S1, else stop the algorithm.

The sense of the update operator "$\Leftarrow$" in (9) is understood as the update of an infinitely dense table $\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, for all (infinitely) possible combinations $(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$.

A compacted update form of the *Algorithm* 1 is possible, by repeating the Q-function update as in

$$\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)\Leftarrow U(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)+\min_{\mathbf{u}}\max_{\mathbf{d}}\overline{Q}_{j-1}(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d}) \quad (11)$$

and to compute the upper-optimal controllers $\overline{C}^*,\overline{D}^*$ by directly minimizing $\overline{Q}^*(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, once $\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$ has converged.

It is important to notice the order of the *min max* operations in the VI updates for the upper Q-function, namely, *max*(.) is performed before *min*(.). This is the important difference w.r.t. the VI *Algorithm 2* update for finding the lower-optimal Q-function, described as follows:

Algorithm 2. Starting from initial (not necessarily admissible) controllers $\underline{C}_0,\underline{D}_0$, and an initial lower Q-function estimate $\underline{Q}_0$, for all the possible combinations of the tuple $(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, alternate the following two steps at each iteration *j* (starting with *j*=1):

S1. Update the Q-function as

$$\underline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)\Leftarrow U(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)+\underline{Q}_{j-1}(\mathbf{z}_{k+1},\mathbf{u}_{k+1}=\underline{C}_{j-1}(\mathbf{z}_{k+1}),\mathbf{d}_{k+1}=\underline{D}_{j-1}(\mathbf{z}_{k+1})) \quad (12)$$

S2. Improve the controllers as in

$$\underline{C}_j(\mathbf{z}_k,\mathbf{d}_k)=\arg\min_{\mathbf{u}}\underline{Q}_j(\mathbf{z}_k,\mathbf{u},\mathbf{d}_k),$$
$$\underline{D}_j(\mathbf{z}_k)=\arg\max_{\mathbf{d}}\underline{Q}_j(\mathbf{z}_k,\underline{C}_j(\mathbf{z}_k,\mathbf{d}),\mathbf{d}). \quad (13)$$

S3. If stopping criterion (no more changes from $\underline{Q}_{j-1}$ to $\underline{Q}_j$) is not met, go to S1, else stop the algorithm.

A compacted update form of *Algorithm* 2 is again possible by repeating the Q-function update as in

$$\underline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)\Leftarrow U(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)+\max_{\mathbf{d}}\min_{\mathbf{u}}\underline{Q}_{j-1}(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d}) \quad (14)$$

and to compute the upper-optimal controllers $\overline{C}^*,\overline{D}^*$ by directly minimizing $\overline{Q}^*(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, once $\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$ has converged.

Convergence of the VI update for the upper Q-function to the upper-optimal controllers and to the upper-optimal original cost is next analysed.

Theorem 1. The updates (9)–(10) (in compacted form as in (11)) starting from $\overline{C}_0,\overline{D}_0$ and from an initial upper Q-function estimate $\overline{Q}_0(\mathbf{z}_k,\overline{C}_0(\mathbf{z}_k),\overline{D}_0(\mathbf{z}_k))\geq 0$, generating the sequences $\{\overline{Q}_j\},\{\overline{C}_j\},\{\overline{D}_j\}$ according to *Algorithm 1*, will converge to the upper-optimal Q-function $\overline{Q}^*(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, to the upper-optimal original cost $\overline{J}^*(\mathbf{z}_k)$ and to the upper-optimal controllers $\overline{C}^*(\mathbf{z}_k),\overline{D}^*(\mathbf{z}_k)$.

Proof. From the compact update (11), notice that on the right-hand side we have, based on (8), that

$$\min_{\mathbf{u}}\max_{\mathbf{d}}\overline{Q}_{j-1}(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d})\overset{(8)}{=}\min_{\mathbf{u}}\max_{\mathbf{d}}\{U(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d})+\overline{J}_{j-1}(F(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d}))\} \quad (15)$$

where $\overline{J}_{j-1}(\mathbf{z}_k)$ is the upper original cost associated with the upper extended cost $\overline{Q}_{j-1}(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$. Notice that the right-hand side of (15) is in fact the VI update performed in the space of the upper original cost:

$$\overline{J}_j(\mathbf{z}_{k+1})\Leftarrow\min_{\mathbf{u}}\max_{\mathbf{d}}\{U(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d})+\overline{J}_{j-1}(F(\mathbf{z}_{k+1},\mathbf{u},\mathbf{d}))\} \quad (16)$$

which holds for all $\mathbf{z}_{k+1}$. In addition, notice that $\overline{Q}_0(\mathbf{z}_k,\overline{C}_0(\mathbf{z}_k),\overline{D}_0(\mathbf{z}_k))=\overline{J}_0(\mathbf{z}_k)\geq 0$ is a positive definite initialization of the upper original cost sequence $\{\overline{J}_j(\mathbf{z}_k)\}$. Altogether, update (11), based on (15) and (16) define a uniquely associated paired sequence $\{(\overline{Q}_j(\mathbf{z}_k),\overline{J}_j(\mathbf{z}_k))\}$.

It was shown in *Lemma 1* from [12] that, with a proper positive definite initialization $\overline{J}_0(\mathbf{z}_k)\geq 0$, the VI update performed in the space of the original cost preserves $\overline{J}_j(\mathbf{z}_k)\geq 0$ for all iterations *j*. *Theorem 2* in [12] shows that $\lim_{j\to\infty}\overline{J}_j(\mathbf{z}_k)=\overline{J}^*(\mathbf{z}_k)$, for all $\mathbf{z}_k$.

Following that the update (11) in the upper Q-function's space embeds the update (16) in the space of the original upper cost and the latter converges to $\overline{J}^*(\mathbf{z}_k)$, it implies by definition (8), that $\lim_{j\to\infty}\overline{Q}_j(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)=\overline{Q}^*(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)\overset{\Delta}{=}$ $\overset{\Delta}{=}U(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)+\overline{J}^*(\mathbf{z}_{k+1})$. It also implies that the controller sequences $\{\overline{C}_j(\mathbf{z}_k)\},\{\overline{D}_j(\mathbf{z}_k)\}$ converge to their upper-optimal values $\overline{C}^*(\mathbf{z}_k),\overline{D}^*(\mathbf{z}_k)$. ■

By similar reasoning, convergence of the VI update for the lower Q-function to the lower-optimal controllers and to the lower-optimal original cost is captured by the next *Theorem 2*.

Theorem 2. The updates (12)–(13) (in compacted form as in (14)) starting from $\underline{C}_0,\underline{D}_0$ and from an initial lower Q-function estimate $\underline{Q}_0(\mathbf{z}_k,\underline{C}_0(\mathbf{z}_k),\underline{D}_0(\mathbf{z}_k))\geq 0$, generating the sequences $\{\underline{Q}_j\},\{\underline{C}_j\},\{\underline{D}_j\}$ according to *Algorithm 2*, will converge to the lower-optimal Q-function $\underline{Q}^*(\mathbf{z}_k,\mathbf{u}_k,\mathbf{d}_k)$, to

the lower-optimal original cost $\underline{J}^*(\mathbf{z}_k)$ and to the lower-optimal controllers $\underline{C}^*(\mathbf{z}_k), \underline{D}^*(\mathbf{z}_k)$.

Proof. The proof uses a similar reasoning with the proof of *Theorem 1*, but relies instead on the convergence of the lower original cost sequence updates, shown in *Lemma 2* and in *Corollary 2* from [12]. It is therefore not detailed here. ■

Observation 2. The proposed algorithms for computing the upper-optimal and lower-optimal Q-functions corresponding to the ZS-TP-G game do not use the system dynamics knowledge. Practical implementations of the proposed algorithms are detailed in the following Section.

Observation 3. Following *Theorem* 5 and *Corollary* 7 from [12], important practical implications of the convergence of the upper-optimal and lower-optimal Q-function updates exist. Convergence of the upper-optimal and lower-optimal Q-functions to the same value is a necessary and sufficient condition for the existence of the saddle-point solution to the ZS-TP-G game. Meaning that $\underline{Q}^* = Q^* = \overline{Q}^*$, where Q^* is the saddle-point solution in the space of Q-functions. This is a consequence of $\underline{J}^* = J^* = \overline{J}^*$ in the space of the original costs. On the other hand, convergence of the upper and lower Q-functions to different values $\underline{Q}^* \neq \overline{Q}^*$ means that a saddle-point solution to the ZS-TP-G game is infeasible.

III. PRACTICAL ALGORITHMS IMPLEMENTATION

A. ZS-TP-G NN IMPLEMENTATION

Algorithms 1 and *2* described in the previous Section are practically implemented using function approximators, to deal with large continuous state and action spaces affected by high dimensionality. Neural networks (NNs) are the most common structures employed to this purpose, owing to their high approximation capability and well-established tuning rules.

Let the NN function approximators for the Q-function, and for the controllers *C*, *D*, be denoted $\hat{Q}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \boldsymbol{\pi}_Q), \hat{C}(\mathbf{z}_k, \boldsymbol{\pi}_C)$ and $\hat{D}(\mathbf{z}_k, \boldsymbol{\pi}_D)$, respectively, where $\boldsymbol{\pi}_i, i \in \{Q, C, D\}$ represents the tuneable NN weights of each individual approximator. Most VI-like algorithms such as the batch-fitted Q-learning variant that is going to be implemented in this work, operate batch-wise and rely on a dataset of transition samples collected from the process by interaction. These samples form a collection (set) of tuples $M = \{(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \mathbf{z}_{k+1})\}$ which allows the calculation of the penalty function. Especially for the VI for unknown dynamics case, these tuples must efficiently explore the state-action space and to cover as uniformly as possible the entire space $\Omega_z \times \Omega_u \times \Omega_d$, i.e. to try all possible actions ($\mathbf{u}_k$, $\mathbf{d}_k$) in every state $\mathbf{z}_k$. The advantage of the VI algorithms is that they are off-policy in nature and they learn the optimal controllers from transition samples collected under any other controllers that can be used for efficient state-action space exploration.

In terms of updating the approximated Q-function iteratively, based on the transition samples dataset *M*, the step S1 from *Algorithms* 1 and 2 ((9) and (12) respectively) is captured by the optimization problem

$$\boldsymbol{\pi}_Q^{j+1} = \arg\min_{\boldsymbol{\pi}} \frac{1}{|M|} \sum_{k=1}^{|M|} \left(\begin{array}{l} \hat{Q}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \boldsymbol{\pi}) - U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) - \\ \hat{Q}(\mathbf{z}_{k+1}, \hat{C}(\mathbf{z}_{k+1}, \boldsymbol{\pi}_C^j), \hat{D}(\mathbf{z}_{k+1}, \boldsymbol{\pi}_D^j), \boldsymbol{\pi}_Q^j) \end{array} \right)^2, \quad (17)$$

where $\hat{Q}, \hat{C}, \hat{D}$ can be any of $\overline{Q}, \overline{C}, \overline{D}$ (*Algorithm* 1) or $\underline{Q}, \underline{C}, \underline{D}$ (*Algorithm* 2). The iteration number *j* has been moved from the subscript of $\overline{Q}, \overline{C}, \overline{D}$ ($\underline{Q}, \underline{C}, \underline{D}$) to the superscript of their corresponding parameterizations. Equation (17) improves the Q-function estimate by bootstrapping on its most recent estimate: (17) is the mean sum of squared errors (MSE) training cost of the neural network $\hat{Q}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \boldsymbol{\pi})$, having targets $\left\{ U(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k) + \hat{Q}(\mathbf{z}_{k+1}, \hat{C}(\mathbf{z}_{k+1}, \boldsymbol{\pi}_C^j), \hat{D}(\mathbf{z}_{k+1}, \boldsymbol{\pi}_D^j), \boldsymbol{\pi}_Q^j) \right\}$. This makes the Q-function estimate improvement directly amenable to standard NN training procedures (e.g. gradient-based backpropagation). The squared error term under the sum in (17) is the well-known one-step temporal difference.

For the controller improvement steps in *Algorithms* 1 and 2 (equations (10) and (13) respectively), the controller parameters $\boldsymbol{\pi}_C^{j+1}, \boldsymbol{\pi}_D^{j+1}$ are obtained from the cascaded NN $\hat{Q}(\mathbf{z}_k, \hat{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \hat{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^j), \boldsymbol{\pi}_Q^{j+1})$ again by gradient descent and ascent steps (per the *min*(.) and *max*(.) operations required by *Algorithms* 1 and 2). Since the succession of the *min*(.) and *max*(.) operations is different for the upper-optimal Q-function calculation *Algorithm 1* and for the lower-optimal Q-function calculation *Algorithm 2*, the details are next given for the former.

In *Algorithm* 1, the *max*(.) operation is performed first, aiming at maximizing $\overline{Q}(\mathbf{z}_k, \overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D), \boldsymbol{\pi}_Q^{j+1})$ w.r.t. $\boldsymbol{\pi}_D$. This is equivalent to setting the targets of $\overline{Q}(\mathbf{z}_k, \overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D), \boldsymbol{\pi}_Q^{j+1})$ equal to zero and take a number of gradient ascent steps

$$\boldsymbol{\pi}_D^{[i+1]} = \boldsymbol{\pi}_D^{[i]} + \frac{\alpha_1}{B_1} \sum_{k=1}^{B_1} \left(\left. \frac{\partial \overline{Q}(\mathbf{z}_k, \overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^{[i]}), \boldsymbol{\pi}_Q^{j+1})}{\partial \overline{D}} \right|_{\boldsymbol{\pi}_D^{[i]}} \left. \frac{\partial \overline{D}(\mathbf{z}_k, \boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \right|_{\boldsymbol{\pi}_D^{[i]}} \right) \quad (18)$$

for a specified number T_1 of gradient ascent steps, starting from an initial inner-loop iteration value $\boldsymbol{\pi}_D^{[i]} = \boldsymbol{\pi}_D^j$, over a number of B_1 selected states $\mathbf{z}_k$ (either randomly picked from the dataset *M* or randomly generated in the domain Ω_z), and using a step-size α_1. At each iteration of (18), the number of B_1 states $\mathbf{z}_k$ are first forward propagated through $\overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^{[i]})$ and afterwards through $\overline{Q}(\mathbf{z}_k, \overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^{[i]}), \boldsymbol{\pi}_Q^{j+1})$. Then, the gradient of $\overline{Q}(\mathbf{z}_k, \overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j), \overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^{[i]}), \boldsymbol{\pi}_Q^{j+1})$ w.r.t. input $\overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D)$ is

calculated with backpropagation and multiplied by the gradient of $\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D)$ w.r.t. $\boldsymbol{\pi}_D$, again calculated by backpropagation. After T_1 iterations of (18), $\boldsymbol{\pi}_D^{j+1}=\boldsymbol{\pi}_D^{[T_1]}$ is rendered.

The *min*(.) operation in *Algorithm* 1 follows, to minimize $\overline{Q}(\mathbf{z}_k,\overline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C),\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{j+1}),\boldsymbol{\pi}_Q^{j+1})$ w.r.t. $\boldsymbol{\pi}_C$. Notice that the $\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{j+1})$ NN already employs the most recent updated parameter obtained after (18). Similarly, this is equivalent to setting zero targets for $\overline{Q}(\mathbf{z}_k,\overline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C),\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{j+1}),\boldsymbol{\pi}_Q^{j+1})$ and minimize w.r.t. $\boldsymbol{\pi}_C$, accomplished by a specified number T_2 of gradient descent steps of the form

$$\boldsymbol{\pi}_C^{[i+1]}=\boldsymbol{\pi}_C^{[i]}-\frac{\alpha_2}{B_2}\sum_{k=1}^{B_2}\left(\left.\frac{\partial\overline{Q}(\mathbf{z}_k,\overline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C^{[i]}),\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{j+1}),\boldsymbol{\pi}_Q^{j+1})}{\partial\overline{C}}\right|_{\boldsymbol{\pi}_C^{[i]}}\left.\frac{\partial\overline{C}(\mathbf{z}_k,\boldsymbol{\pi})}{\partial\boldsymbol{\pi}}\right|_{\boldsymbol{\pi}_C^{[i]}}\right) \tag{19}$$

performed by starting from an initial inner-loop iteration value $\boldsymbol{\pi}_C^{[i]}=\boldsymbol{\pi}_C^{j}$, over a number of B_2 selected states $\mathbf{z}_k$ (either randomly picked from the dataset M or randomly generated in the domain Ω_z), and using a step-size α_2. The same computation mechanism applies, as in the case of the *max*(.) operation (18). After T_2 iterations of (19), $\boldsymbol{\pi}_C^{j+1}=\boldsymbol{\pi}_C^{[T_2]}$ is rendered.

Algorithm 2 dedicated to the optimal lower Q-function and optimal lower controllers' calculations differs in the order of the *min*(.) and afterwards *max*(.) operations. Meaning that $\boldsymbol{\pi}_C^{j+1}$ is first updated, then used to update $\boldsymbol{\pi}_D^{j+1}$. They are given as

$$\boldsymbol{\pi}_C^{[i+1]}=\boldsymbol{\pi}_C^{[i]}-\frac{\alpha_1}{B_1}\sum_{k=1}^{B_1}\left(\left.\frac{\partial\underline{Q}(\mathbf{z}_k,\underline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C^{[i]}),\underline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{j}),\boldsymbol{\pi}_Q^{j+1})}{\partial\underline{C}}\right|_{\boldsymbol{\pi}_C^{[i]}}\left.\frac{\partial\underline{C}(\mathbf{z}_k,\boldsymbol{\pi})}{\partial\boldsymbol{\pi}}\right|_{\boldsymbol{\pi}_C^{[i]}}\right) \tag{20}$$

$$\boldsymbol{\pi}_D^{[i+1]}=\boldsymbol{\pi}_D^{[i]}+\frac{\alpha_2}{B_2}\sum_{k=1}^{B_2}\left(\left.\frac{\partial\underline{Q}(\mathbf{z}_k,\underline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C^{j+1}),\underline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^{[i]}),\boldsymbol{\pi}_Q^{j+1})}{\partial\underline{D}}\right|_{\boldsymbol{\pi}_D^{[i]}}\left.\frac{\partial\underline{D}(\mathbf{z}_k,\boldsymbol{\pi})}{\partial\boldsymbol{\pi}}\right|_{\boldsymbol{\pi}_D^{[i]}}\right) \tag{21}$$

We summarize the NN-based solutions to the ZS-TP-G aiming at computing the upper-optimal and lower-optimal Q-functions and upper-optimal and lower-optimal controllers, respectively, using the batch-fitted Q-learning style. For the upper-optimal Q-function and upper-optimal controller, *Algorithm 3* is described first.

Algorithm 3. NN-based solution for the upper-optimal Q-function and upper-optimal controller for the ZS-TP-G.

1. Take the dataset M of collected transition samples as input.
2. Initialize $\bar{j},B_1,B_2,T_1,T_2,\alpha_1,\alpha_2,\Delta_\pi$, all NNs' architecture and training settings and the values $j=0,\boldsymbol{\pi}_Q^0,\boldsymbol{\pi}_C^0,\boldsymbol{\pi}_D^0$.
3. At a certain iteration step j, obtain an improved NN estimate of the upper Q-function as the solution $\boldsymbol{\pi}_Q^{j+1}$ of (17), using the entire dataset M of transition samples.
4. Initialize $\boldsymbol{\pi}_D^{[i]}=\boldsymbol{\pi}_D^{j}$ and iterate for T_1 times on (18) to find $\boldsymbol{\pi}_D^{j+1}$. A set of B_1 states $\mathbf{z}_k$ is used.
5. Initialize $\boldsymbol{\pi}_C^{[i]}=\boldsymbol{\pi}_C^{j}$ and iterate for T_2 times on (19) to find $\boldsymbol{\pi}_C^{j+1}$. A set of B_2 states $\mathbf{z}_k$ is used.
6. If the stopping criteria is not met in terms of maximum number of iterations ($j<\bar{j}$) and in terms of significant changes in the Q-function NN parameters between iterations ($\left\|\boldsymbol{\pi}_Q^{j+1}-\boldsymbol{\pi}_Q^{j}\right\|>\Delta_\pi$), update j=j+1 and go to 3, otherwise stop.

For the lower-optimal Q-function and lower-optimal controller calculations, the following *Algorithm 4* is described.

Algorithm 4. NN-based solution for the lower-optimal Q-function and lower-optimal controller for the ZS-TP-G.

1. Take the dataset M of collected transition samples as input.
2. Initialize $\underline{j},B_1,B_2,T_1,T_2,\alpha_1,\alpha_2,\Delta_\pi$, all NNs' architecture and training settings and the values $j=0,\boldsymbol{\pi}_Q^0,\boldsymbol{\pi}_C^0,\boldsymbol{\pi}_D^0$.
3. At a certain iteration step j, obtain an improved NN estimate of the upper Q-function as the solution $\boldsymbol{\pi}_Q^{j+1}$ of (17), using the entire dataset M of transition samples.
4. Initialize $\boldsymbol{\pi}_C^{[i]}=\boldsymbol{\pi}_C^{j}$ and iterate for T_1 times on (20) to find $\boldsymbol{\pi}_C^{j+1}$. A set of B_1 states $\mathbf{z}_k$ is used.
5. Initialize $\boldsymbol{\pi}_D^{[i]}=\boldsymbol{\pi}_D^{j}$ and iterate for T_2 times on (21) to find $\boldsymbol{\pi}_D^{j+1}$. A set of B_2 states $\mathbf{z}_k$ is used.
6. If the stopping criteria is not met in terms of maximum number of iterations ($j<\underline{j}$) and in terms of significant changes in the Q-function NN parameters between iterations ($\left\|\boldsymbol{\pi}_Q^{j+1}-\boldsymbol{\pi}_Q^{j}\right\|>\Delta_\pi$), update j=j+1 and go to 3, otherwise stop.

Observation 4. After *Algorithms 3* and *4* converge, it is established whether the saddle-point solution to the ZS-TP-G exists (the upper-optimal and lower-optimal Q-functions converge to the same value) or, on the contrary, the saddle-point solution does not exist. In practice, it is more convenient to measure and analyse the upper and lower original costs values, evaluated with the current iteration controllers on a test scenario, that is, $\overline{J}_j(\mathbf{z}_k)=\overline{Q}(\mathbf{z}_k,\overline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C^j),\overline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^j),\boldsymbol{\pi}_Q^j)$ and $\underline{J}_j(\mathbf{z}_k)=\underline{Q}(\mathbf{z}_k,\underline{C}(\mathbf{z}_k,\boldsymbol{\pi}_C^j),\underline{D}(\mathbf{z}_k,\boldsymbol{\pi}_D^j),\boldsymbol{\pi}_Q^j)$, respectively. $\overline{J}_j(\mathbf{z}_k)$ and $\underline{J}_j(\mathbf{z}_k)$ will be measured in the next case study.

In the following, a state feedback optimal controller is introduced for comparing the performance of the upper and lower optimal controllers in terms of disturbance rejection capability.

B. A STATE FEEDBACK OPTIMAL CONTROLLER NN IMPLEMENTATION

To assess the performance of the learned optimal upper and lower controllers, a state feedback optimal controller (SFOC) is learned, that is set to solve the next optimization problem:

$$C^* = \arg\min_C J(\mathbf{x}_k) = \sum_{j=k}^{\infty} U(\mathbf{x}_j, \mathbf{u}_j), \quad \text{with } U(\mathbf{x}_j, \mathbf{u}_j) = \Theta(\mathbf{x}_j) + C(\mathbf{x}_j, \boldsymbol{\pi}_C)^T \mathbf{W}_C C(\mathbf{x}_j, \boldsymbol{\pi}_C)^T. \tag{22}$$

The above cost preserves a part of the penalty term in the original cost from (5), without penalizing the disturbance term ($\mathbf{W}_D$=0). The controller that learns to solve (22) uses the straightforward system state and not a virtual state but it is not designed to aim for disturbance attenuation. (22) is solvable by any variant of *Algorithm* 3 or 4, without a dedicated disturbance controller NN approximator $D(\mathbf{x}_k, \boldsymbol{\pi}_D^j)$, but only with a NN controller $C(\mathbf{x}_k, \boldsymbol{\pi}_C^j)$ that is improved at each step by minimizing the current iteration Q-function NN $Q(\mathbf{x}_k, u_k, \boldsymbol{\pi}_Q^j)$. Let the *Algorithm* 5 used for SFOC learning be

Algorithm 5. SFOC learning

1. Collect another dataset M_1 of transition samples from the system, to be used as input to the algorithm. The collection task takes place under d_k=0.

2. Initialize $\bar{j}, B, T, \alpha, \Delta_\pi$ and initialize architectures and training settings for the NNs $Q(\mathbf{x}_k, u_k, \boldsymbol{\pi}_Q^j)$ and $C(\mathbf{x}_k, \boldsymbol{\pi}_C^j)$. Initialize the values $j = 0, \boldsymbol{\pi}_Q^0, \boldsymbol{\pi}_C^0$.

3. At a certain iteration step *j*, using the entire dataset M_1 of transition samples, obtain an improved NN estimate of the Q-function as the solution $\boldsymbol{\pi}_Q^{j+1}$ of

$$\boldsymbol{\pi}_Q^{j+1} = \arg\min_{\boldsymbol{\pi}} \frac{1}{|M_1|} \sum_{k=1}^{|M_1|} \left(\begin{matrix} \hat{Q}(\mathbf{z}_k, \mathbf{u}_k, \boldsymbol{\pi}) - U(\mathbf{z}_k, \mathbf{u}_k) \\ -\hat{Q}(\mathbf{z}_{k+1}, \hat{C}(\mathbf{z}_{k+1}, \boldsymbol{\pi}_C^j), \boldsymbol{\pi}_Q^j) \end{matrix} \right)^2, \tag{23}$$

4. Initialize $\boldsymbol{\pi}_C^{[i]} = \boldsymbol{\pi}_C^j$ and iterate for *T* times on

$$\boldsymbol{\pi}_C^{[i+1]} = \boldsymbol{\pi}_C^{[i]} - \frac{\alpha}{B} \sum_{k=1}^{B} \left(\left. \frac{\partial \hat{Q}(\mathbf{x}_k, \hat{C}(\mathbf{x}_k, \boldsymbol{\pi}_C^{[i]}), \boldsymbol{\pi}_Q^{j+1})}{\partial \hat{C}} \right|_{\boldsymbol{\pi}_C^{[i]}} \left. \frac{\partial \hat{C}(\mathbf{x}_k, \boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \right|_{\boldsymbol{\pi}_C^{[i]}} \right), \tag{24}$$

to find $\boldsymbol{\pi}_C^{j+1}$. A number of $B \le |M_1|$ states $\mathbf{x}_k$ from the dataset M_1 can be used.

5. If the stopping criteria is not met in terms of maximum number of iterations ($j < \bar{j}$) and in terms of significant changes in the Q-function NN parameters between iterations ($\left\| \boldsymbol{\pi}_Q^{j+1} - \boldsymbol{\pi}_Q^j \right\| > \Delta_\pi$), update *j*=*j*+1 and go to 3, otherwise stop.

IV. VALIDATION CASE STUDY

A. THE ACTIVE SUSPENSION SYSTEM

The continuous-time state-space model of the active suspension system for a quarter-car is [52]

$$\begin{cases} \dot{\bar{x}}_1 = \bar{x}_2 \\ \dot{\bar{x}}_2 = \frac{1}{m_s}(-b_s(\bar{x}_2 - \bar{x}_4) - k_s(\bar{x}_1 - \bar{x}_3) - k_{sn}(\bar{x}_1 - \bar{x}_3)^3 + \frac{A}{\varphi}\bar{x}_5) \\ \dot{\bar{x}}_3 = \bar{x}_4 \\ \dot{\bar{x}}_4 = \frac{1}{m_u}(b_s(\bar{x}_2 - \bar{x}_4) + k_s(\bar{x}_1 - \bar{x}_3) + k_{sn}(\bar{x}_1 - \bar{x}_3)^3 - k_t(\bar{x}_3 - \bar{x}_6) \\ \qquad - b_t(\bar{x}_4 - \Xi_1 d) - \frac{A}{\varphi}\bar{x}_5) \\ \dot{\bar{x}}_5 = -\beta\bar{x}_5 - \varphi A\alpha(\bar{x}_2 - \bar{x}_4) + \varphi\phi\kappa\Xi_2 u \\ \dot{\bar{x}}_6 = \Xi_1 d \end{cases}$$
$$y = \bar{x}_1$$
$$\text{with} \quad \kappa = \text{sgn}[P_s - \text{sgn}(\Xi_2 u)\frac{\bar{x}_5}{\varphi}]\sqrt{\left| P_s - \text{sgn}(\Xi_2 u)\frac{\bar{x}_5}{\varphi} \right|}, \tag{25}$$

where the model parameters are given as [52]: $m_s = 600kg$, $m_u = 60kg$, $k_t = 200000\,N/m$, $b_t = 1000\,Ns/m$, $k_{sn} = 1000\,N/m$, $k_s = 18000\,N/m$, $b_s = 2500\,Ns/m$, $\varphi = 1\times10^{-7}$, $\beta = 1s^{-1}$, $A = 3.35\times10^{-4}m^2$, $P_s = 10342500\,Pa$, $\alpha = 4.151\times10^{13}\,N/m^{5/2}$, $\phi = 1.545\times10^{9}\,N/m^{5/2}$. The displacements $\bar{x}_1$ and $\bar{x}_3$ of the sprung (car body) and unsprung mass (wheel), respectively, are defined in (25) w.r.t. to their resting position. A four-way valve-piston that is actuated hydraulically, generates the force denoted as $\bar{x}_5$ in (25) as a consequence of applying a voltage on the actuator input – this is the control input *u*. The road profile derivative w.r.t. time models the input disturbance *d*. To normalize the disturbance in $d \in [-1;1]$ (corresponding to +/– 3 cm/s maximum amplitude of the road profile derivative), a scaling constant $\Xi_1 = 0.03$ multiplies the input *d* in the model (25). Similarly, the input *u* is brought to $u \in [-1;1]$ by using the scaling constant $\Xi_2 = 0.001$. In (25), the sgn(.) denotes the sign function. Clearly from (25), the output equation extracts $\bar{x}_1$ as a measurable. The active suspension is schematically depicted in Fig. 1.

Since the IO data from model (25) will be collected at the fixed sample period of $T_s = 0.01$ sec, the model is regarded as an equivalent discrete-time one (with a zero-order hold on the inputs that preserves their value constant for one sample period) and used for IO measurement. Importantly, the active suspension model is not used in the learning process. Let the states of the discrete-time equivalent model (25) be grouped by $\mathbf{x}_k = [x_{1,k}, ..., x_{6,k}]^T$ (i.e. $x_{i,k}$ corresponds to $\bar{x}_i$). It then follows that (25) can be expressed as $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \bar{\mathbf{u}}_k = [u_k, d_k]^T)$.

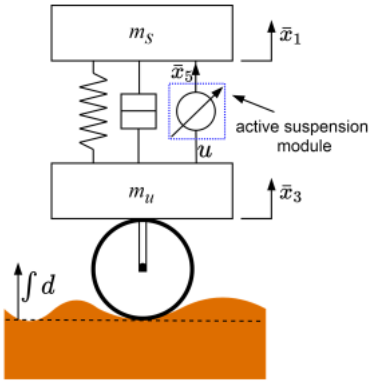

FIGURE 1. Diagram of the active suspension system.

For the active suspension, the artificial disturbance $d \in [-1;1]$ is easily generated in fixed indoor stands and is therefore measurable for learning purposes.

B. ACTIVE SUSPENSION SYSTEM OBSERVABILITY DISCUSSION

The observability of (25) is next discussed, according to the analysis of [53]. The system observability does not depend on the inputs (u_k, d_k) who are set to zero in (25). The analysis is carried out on the continuous-time system (25) and if the continuous-time system is observable, the observability of its discrete-time counterpart is implied when a sufficiently small sampling period is employed for discretization. Kou et al. showed in [53] that for a nonlinear dynamic autonomous continuous-time system with state equation $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), x \in R^n$ and with output equation $\mathbf{y}(t) = h(\mathbf{x}(t)) \in R^m$, under smoothness assumptions for $\mathbf{y}(t)$ implying the existence of the k^{th} order derivative of $h(.)$ such that $km \geq n$, under Taylor series expansion of $\mathbf{y}(t)$ on a time interval $t \in [t_0, t_1]$ in the vicinity of any initial time t_0, a nonlinear map $H(.)$ is built as in

$$
\begin{aligned}
&\mathbf{z} = H(\mathbf{x}(t_0)), \quad \text{where} \\
&\mathbf{z} = [\mathbf{y}(t_0)^T, \mathbf{y}^{(1)}(t_0)^T, ..., \mathbf{y}^{(k-1)}(t_0)^T]^T, \\
&H(\mathbf{x}(t_0)) = [h_0(\mathbf{x}(t_0))^T, h_0(\mathbf{x}(t_0))^T, ..., h_{k-1}(\mathbf{x}(t_0))^T]^T, \\
&\text{where} \\
&\mathbf{y}(t_0) = h(\mathbf{x}(t_0)) \overset{\Delta}{=} h_0(\mathbf{x}(t_0)), \\
&\dot{\mathbf{y}}(t_0) = \frac{\partial h_0}{\partial t}(\mathbf{x}(t_0)) = \frac{\partial h_0}{\partial \mathbf{x}(t_0)}(\mathbf{x}(t_0)) \cdot f(\mathbf{x}(t_0)) \overset{\Delta}{=} h_1(\mathbf{x}(t_0)), \\
&... \\
&\mathbf{y}^{(k-1)}(t_0) = \frac{\partial h_{k-2}}{\partial \mathbf{x}(t_0)}(\mathbf{x}(t_0)) \cdot f(\mathbf{x}(t_0)) \overset{\Delta}{=} h_{k-1}(\mathbf{x}(t_0)).
\end{aligned}
\tag{26}
$$

The dynamic system described above is completely observable on $t \in [t_0, t_1]$, if $H(.)$ is injective (univalent, or one-to-one) from an initial state $\mathbf{x}(t_0)$ to $\mathbf{z}$. The univalence of $H(.)$ is a sufficient observability condition, since $\mathbf{z}$ contains only the output and its derivatives at initial time t_0 [53] and not on the entire $t \in [t_0, t_1]$. If one can show the map $H(\mathbf{x})$ is (locally) invertible (i.e., a bijection) then its injectivity follows. Local map invertibility is ensured by the non-singularity of its Jacobian matrix determinant at a certain given point, which for a square map $H(\mathbf{x})$ is equivalent to the maximum matrix rank of the Jacobian at the given point.

For system (25), by repeated substitutions ($y = \bar{x}_1, \dot{y} = \dot{\bar{x}}_1 = \bar{x}_2$, …) using the model equations (25), it is verified that the Jacobian of $H(\mathbf{x})$ is of full rank six, irrespective of the point at which it is calculated. Meaning that (25) is observable in continuous-time (and subsequently in discrete-time, for a small enough sampling period). This implies that a virtual state can be constructed from past inputs-outputs samples.

In practice, the model (25) is assumed unknown and the observability must be assumed if not verifiable from literature or from working experience with the process. Since the number of true states as well as the observability index are unknown, the virtual state should be built from more inputs-outputs past samples. It was reported in [51] that beyond a certain number (the presumed observability index) of past IO samples, there is no gain in information about the state value.

C. COLLECTIONG TRANSITION SAMPLES FOR THE LEARNING PROCESS

The first goal is to collect a transition samples dataset $M = \{(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \mathbf{z}_{k+1})\}$. Since the system (25) is observable, a controllability index equal to six builds a virtual state from past samples of the inputs u_k, d_k and from present and past samples of the output y_k. The virtual state has the form $\mathbf{z}_k = [y_k, ..., y_{k-6}, u_{k-1}, ..., u_{k-6}, d_{k-1}, ..., d_{k-6}]^T \in R^{19}$ and the system (25) is transformed to a virtual state-space model of the form $\mathbf{z}_{k+1} = \mathbf{F}(\mathbf{z}_k, u_k, d_k)$ with output equation $y_k = z_{1,k}$. In addition, the system is IO stable due to existing friction and therefore it can be open-loop excited.

Then, the transition samples are gathered using the next parameters for u_k and d_k: the input $u_k \in [-1;1]$ is modelled as a sequence of piece-wise constant steps, while the amplitude follows a random uniform distribution. Each step last for 0.5 sec, and it is perturbed with a random noise extracted from another uniform distribution of amplitudes inside $[-0.2;0.2]$. This noise is added to u_k every T_s seconds. The disturbance input $d_k \in [-1;1]$ is modelled similar to u_k but each constant portion lasts 0.6 sec. And it is additively perturbed by a similar uniform random noise with the same random uniform noise of amplitude $[-0.2;0.2]$ every T_s seconds. The additive noise on the two input channels u_k, d_k are uncorrelated. The database M of

$|M| = 20'000$ transition samples is built from 200 sec experiment time with the system (25) in open-loop, excited by the above u_k, d_k. Since the inputs were already normalized in $[-1;1]$ by introducing the input normalizing coefficients in the model (25), the output is normalized to $y_k \in [-1;1], k = \overline{1,|M|}$ by dividing each sample with the maximal absolute value $\max_k |y_k|$ from the recorded history. Usually, all the inputs and outputs should be normalized, leading to all the components of the virtual state being normalized.

The virtual state's components normalization is extremely important since NNs approximators are going to be used. The normalization coefficients of all states are memorized and used to de-normalize the states, when running the learned controller in the loop.

D. CONTROLLER LEARNING SETTINGS AND RESULTS

Before the learning process, the penalty in the original cost (5) is constructed as $20y_k^2 + u_k^2 - 4d_k^2$ (for $\mathbf{W}_C = 1, \mathbf{W}_D = 4$). It is computed for each transition sample $(\mathbf{z}_k, \mathbf{u}_k, \mathbf{d}_k, \mathbf{z}_{k+1})$.

In order to find the upper-optimal and lower-optimal controllers and Q-functions according to *Algorithms* 3 and 4 respectively, some approximators are selected as follows. For each algorithm the same architectures are being used. The Q-function is a 21–50–1 feedforward NN with *tanh*(.) activation in the hidden layer and linear output activation. The input of this approximator $\hat{Q}(\mathbf{z}_k, u_k, d_k, \boldsymbol{\pi}_Q)$ is formed from 19 components of the virtual state and the two control inputs u_k, d_k, while $\boldsymbol{\pi}_Q$ formally captures the NN weights. These weights are initialized with uniform random numbers inside $[-0.005;0.005]$. A random 80% of the training samples are used for effective training and the rest are used as validation data, for forcing early stopping in order to prevent training overfitting. The training algorithm is a fast, scaled conjugate gradient, for maximum 500 episodes. The training uses the MSE criterion over the entire batch of transition samples. The Q-function NN training (in both optimal upper and optimal lower Q-function search process) solves in fact (17).

For the controllers $\hat{C}(\mathbf{z}_k, \boldsymbol{\pi}_C)$ and $\hat{D}(\mathbf{z}_k, \boldsymbol{\pi}_D)$, the NN approximators are also feedforward NNs of the form 19–10–1, with *tanh*(.) activation in the hidden layer and linear output activation. Their weights captured by $\boldsymbol{\pi}_C, \boldsymbol{\pi}_D$ are initialized as for the Q-function NN, but the two NNs' training must comply with the gradient ascent/descent steps imposed by the upper-optimal controllers' search (equations (18) and (19) performed inside *Algorithm 3*) and by the lower-optimal controllers' search (equations (20) and (21) performed inside *Algorithm 4*).

Other parameters are selected as follows. For *Algorithm 3*, $\bar{j} = 500, T_1 = T_2 = 50, \alpha_1 = \alpha_2 = 10^{-3}, \Delta_\pi = 10^{-5}$, while the gradient ascent/descent steps from (18) and (19) are performed on a number of B_1=B_2=256 values $\mathbf{z}_k$ randomly picked from the dataset M at each ascent or descent step. For the *Algorithm 4*, the same parameter settings are used.

The results obtained after the learning process takes place is shown in Fig. 2, in terms of the normed difference between successive Q-function weights vectors and in terms of the measured attenuation cost [27], [31]

$$J_{test} = \left(\sum_{k=0}^{10000} 20y_k^2 + u_k^2\right) / \left(4\sum_{k=0}^{10000} d_k^2\right), \quad (27)$$

defined and measured on test scenario lasting 100 seconds, where a disturbance $d_k \in [-1;1]$ (modelled as successive piece-wise constant steps of uniform random amplitudes and lasting for 0.5 sec) is used. The sequence d_k has not been presented to the system in the transition samples collection phase used for learning the upper-optimal and lower-optimal controllers.

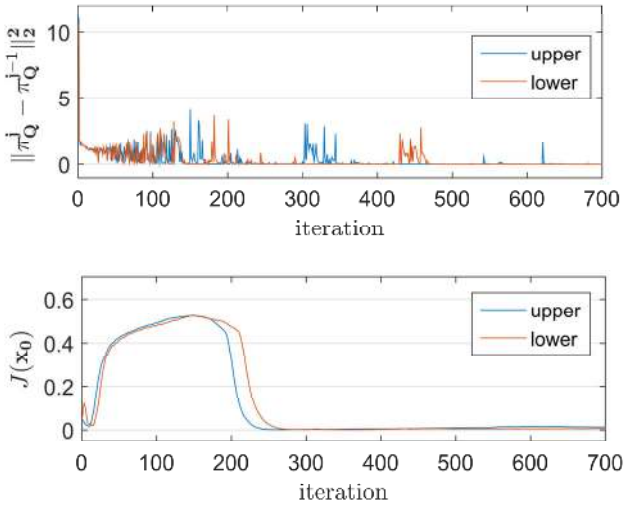

FIGURE 2. The learning process for the upper-optimal upper and lower-optimal controllers.

Observation 5. Importantly, at every iteration, J_{test} is measured with the upper and lower controllers ($\overline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j)$ and $\underline{C}(\mathbf{z}_k, \boldsymbol{\pi}_C^j)$ respectively) in closed-loop, without using the disturbance controllers $\overline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^j)$ and $\underline{D}(\mathbf{z}_k, \boldsymbol{\pi}_D^j)$, their outputs being replaced by the test input signal d_k. These latter disturbance controllers are necessary only throughout the learning process of *Algorithms* 3 and 4.

Inspecting the bottom subplot in Fig. 2, the original costs $\bar{J}_j(\mathbf{z}_k)$ and $\underline{J}_j(\mathbf{z}_k)$ converge to the same value, meaning that the saddle-point solution to the game exists. The upper subplot in Fig. 2 also indicates that after many iterations, no more changes tend to occur in the upper and lower Q-function estimates, a sign of learning process' convergence.

E. COMPARISONS AND DISCUSSIONS OF THE RESULTS

For learning the SFOC via *Algorithm* 5 for comparison purposes, the following optimization problem is solved

$$C^* = \arg\min_C J(\mathbf{x}_k) = \sum_{j=k}^{\infty} 20x_{1,j}^2 + u_j^2, \quad \text{with } u_j = C(\mathbf{x}_j), \tag{28}$$

which is computed for the transition samples collected under the same u_k settings used in the previous subsection, letting d_k=0. As a consequence of null disturbance, a fifth order state model version of (25) results. Two feed-forward NNs of 6–30–1 and 5–5–1 are employed for the Q-function estimate and for the controller estimate, respectively. $T = 50$ gradient descent steps (24) are repeated with each major iteration of the *Algorithm* 5. Each state component $x_{i,k}$ from $\mathbf{x}_k$ is normalized $x_{i,k} \in [-1;1], i = \overline{1,5}, k = \overline{1,|M_1|}$ by dividing each sample with its greatest modulus $\max_k |x_{i,k}|$ over the recorded history.

The rest of the parameters in *Algorithm* 5 are $\bar{j} = 500, \alpha = 0.005, B = 128, \Delta_\pi = 10^{-5}$. After the maximum number of 500 elapsed iterations, the optimal controller and the optimal Q-function estimates result.

For comparison, on the same test scenario, the control obtained with the upper-optimal controller and with the SFOC are shown in Fig. 3.

The Fig. 3 is interpreted as follows. The black line in all subplots correspond to open-loop (u_k=0) and the profile of $x_{1,k}$ (the "car body") is the same as the profile of $x_{3,k}$ ("the wheel" as the unsprung mass). It means that the wheel follows the road profile obtained as the discrete-time integral of d_k from Fig. 3. Therefore, $x_{1,k} - x_{3,k} = 0$ in Fig. 3A in black line. On the other hand, the blue line $x_{1,k}$ in Fig. 3B means that the car body is insensitive to the road disturbance and the active suspension control manages to absorb the road profile via the unsprung mass $x_{3,k}$ using the control input in blue line from Fig. 3D. The SFOC control (red line $x_{1,k}$ in Fig. 3B) does not reject the disturbance as the upper-optimal controller, since it was not learned having in mind the disturbance rejection goal. After measuring the attenuation obtained with both the upper-optimal controller and with the SFOC controller, it results that $J_{test}^{OptUppContr} = 6.6 \cdot 10^{-3}, J_{test}^{SFOC} = 34.5 \cdot 10^{-3}$, clearly indicating the effective attenuation attained by the former. This is despite the SFOC using the full state information directly, which may be considered an advantage.

On another hand, the virtual state used for learning the upper-optimal and lower-optimal controllers incorporates the measured disturbance which is the road profile derivative. This may not be acceptable in practice since it is difficult to measure the road profile disturbance. The learned upper-optimal controller is tested next, by setting $\mathbf{z}_k = [y_k, ..., y_{k-6}, u_{k-1}, ..., u_{k-6}, d_{k-1} = 0, ..., d_{k-6} = 0]^T \in R^{19}$ with null disturbance in the virtual state. The actual disturbance d_k affects the controlled system, and the closed-loop is tested under the same scenario as before, under both the upper-optimal controller and under the SFOC. The results are shown again in Fig. 4.

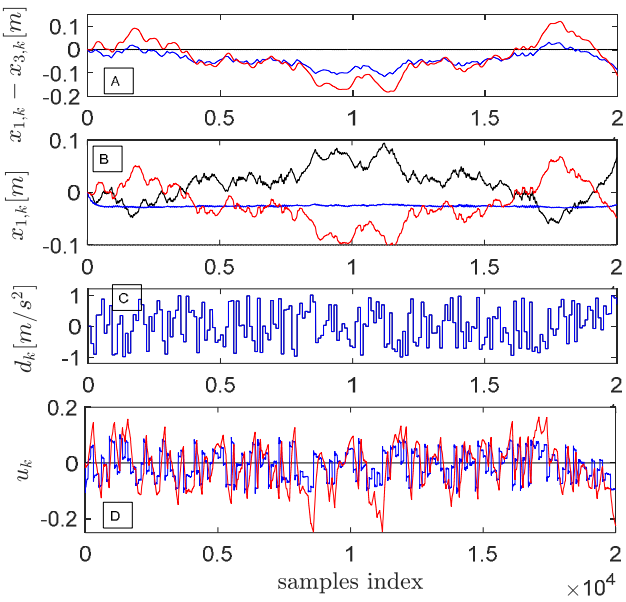

FIGURE 3. The response obtained with the optimal upper controller (blue), with the SFOC (red) and in open-loop with u_k=0 (black).

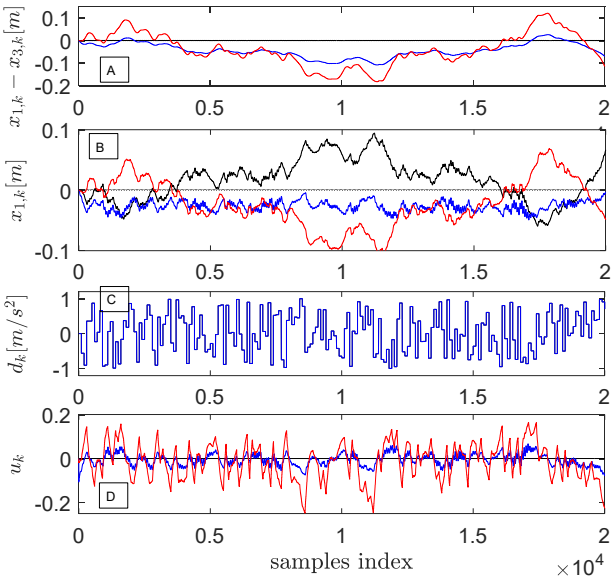

FIGURE 4. The response obtained with the upper-optimal controller (blue), with the SFOC (red) and in open-loop with u_k=0 (black). This time, the virtual state is fed with d_k=0.

The conclusion from Fig. 4 is obvious. Even in the case when a null disturbance is fed to the virtual state, the disturbance rejection is better with the upper-optimal

controller than with the SFOC, in terms of $x_{1,k}$ in Fig. 4B being closer to zero than the same $x_{1,k}$ obtained with the SFOC. Meaning that effective disturbance attenuation is still obtained, without measuring the road profile derivative.

The transmissibility from the disturbance input d_k to the output $y_k = x_{1,k}$ is also measured in the frequency domain, assuming an approximate linear model both for the open-loop suspension system and for the closed-loop suspension control system. The frequency response function estimator is identified in three cases: a) in open-loop setting (u_k=0); b) the loop closed with the upper-optimal controller, with the virtual state $\mathbf{z}_k$ fed by the disturbance input d_k and c) the loop closed with the upper-optimal controller with $\mathbf{z}_k$ fed by d_k=0. The results captured by Fig. 5 were obtained after exciting either of the open-loop system or the closed-loop control system with a zero-mean sine-stream signal d_k of amplitude 0.5, for 100 logarithmically-spaced frequencies in the range of 0.01–1000 Hz. Then the magnitudes of the ratio between the Fast Fourier Transform (FFT) of the output $y_k = x_{1,k}$ and the FFT of the input d_k obtained at each particular frequency, are calculated. Even in the active suspension provides natural attenuation in open-loop, it is observed that in the case b) (corresponding to measured disturbance in the virtual state), the low-frequency attenuation is significantly stronger than that obtained with the SFOC (Fig. 5, left). Still, better low-frequency attenuation obtained with the upper-optimal controller is measured in the case c) (Fig. 5, right), when the disturbance is not measured and it enters as a null value in the virtual state.

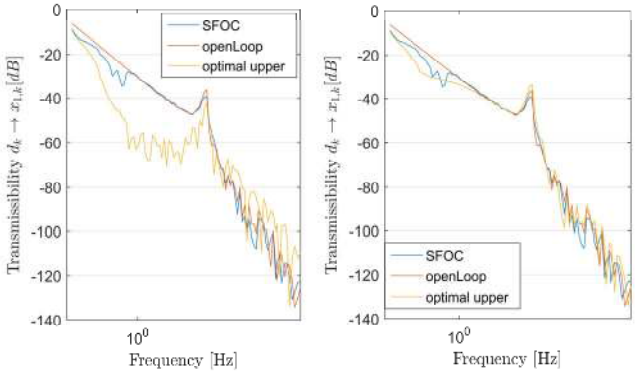

FIGURE 5. Transmissibility in open-loop (u_k=0), with the SFOC and with the upper-optimal controller. In the left, the virtual state z_k is fed by the actual disturbance input values d_k; on the right, d_k=0 in z_k.

The learned upper-optimal and lower-optimal controllers for the active suspension observable system was shown feasible. For the active suspension system, the proposed ZS-TP-G robust control learning approach is highly attractive since it takes place in a fixed test rig where artificial disturbances that emulate the road conditions are easily generated. Afterwards, the disturbance controller is discarded and the control loop is closed by either the upper-optimal controller or the lower-optimal controller. Subsequently, the active suspension can then be used in real-world road conditions. The learned attenuation was shown efficient even in the case when the virtual state is constructed from a null measured disturbance. This aspect expands the applicability range of the approach. All features above may stimulate industrial implementation owing to the reduced number of sensors and to the on-site learning ability.

VII. CONCLUSION

The approach presented in this paper proposes several features, enumerated next. It learns an optimal robust controller using ADP formulated as a ZS-TP-G for systems with unknown dynamics. The learned controller is the saddle-point of the ZS-TP-G when the solution is feasible, otherwise it can be any of the upper-optimal or lower-optimal controllers that solve the game. The learning process consisting of the operations that are specific to the upper-optimal and lower-optimal controllers' calculations, was shown to converge by theoretical analysis.

NNs approximators were used for the practical learning implementation. This is advantageous for general nonlinear systems since it enables automatic feature selection in the Q-function and controller parameterization. The proposed framework deals with observable systems perceived from IO data, therefore solving the partial observability problem that can prevent successful learning. It relies on the virtual state built from present and past values of the input and output samples. Learning a robust control for the virtual state space system is shown equivalent to learning a robust control for the underlying system. Since the virtual state construction leads to a higher-order virtual state-space system, NNs ensure the scalability of the learning problem in all aspects, except for the efficient exploration problem which is one of the major issues with ADP and reinforcement learning.

The approach presented here is believed to handle many practical systems (such as Markov jump systems and nonlinear multiagent systems [54]–[57]), therefore it is a further goal to validate it on observable systems of even higher order who, similarly to the active suspension, show significant practical interest.

REFERENCES

[1] A. J. van der Schaft, "L2-gain analysis of nonlinear systems and nonlinear state-feedback H∞ control," *IEEE Trans. Autom. Control*, vol. 37, no. 6, pp. 770–784, 1992.
[2] T. Basar and P. Bernhard, H∞-Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach, 2nd ed. Boston, MA, USA: Birkhäuser, 1995.
[3] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for

synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, Aug. 2012.

[4] H. Zhang, L. Cui, and Y. Luo, "Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP," *IEEE Trans. Cybern*., vol 43, no. 1, pp. 206–216, Feb. 2013.

[5] D. Liu, H. Li, and D. Wang, "Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics," *IEEE Trans. Syst., Man, Cybern. Syst*., vol. 44, no. 8, pp. 1015–1027, Aug. 2014.

[6] D. Zhao, Q. Zhang, D. Wang, and Y. Zhu, "Experience replay for optimal control of nonzero-sum game systems with unknown dynamics," *IEEE Trans. Cybern*., vol. 46, no. 3, pp. 854–865, Mar. 2016.

[7] R. Song, F. L. Lewis, and Q. Wei, "Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 28, no. 3, pp. 704–713, Mar. 2017.

[8] H. Xu and K. Mizukami, "Linear-quadratic zero-sum differential games for generalized state space systems," *IEEE Trans. Autom. Control*, vol. 39, no. 1, pp. 143–147, Jan. 1994.

[9] J. Engwerda, "Uniqueness conditions for the affine open-loop linear quadratic differential game," *Automatica*, vol. 44, no. 2, pp. 504–511, Feb. 2008.

[10] X. Yang and J. Gao, "Linear–Quadratic uncertain differential game with application to resource extraction problem," *IEEE Trans. Fuzzy Syst*., vol. 24, no. 4, pp. 819–826, Aug. 2016.

[11] H. Zhang, Q. Wei, and D. Liu, "An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games," *Automatica*, vol. 47, no. 1, pp. 207–214, Jan. 2011.

[12] Q. Wei, D. Liu, Q. Lin, and R. Song, "Adaptive dynamic programming for discrete-time zero-sum games," *IEEE Trans. on Neural Netw. Learn. Syst*., vol. 29, no. 4, pp. 957–969, Apr. 2018.

[13] R. E. Bellman, *Dynamic Programming.* Princeton, NJ, USA: Princeton Univ. Press, 1957.

[14] K. V. Berkel, B. D. Jager, T. Hofman, and M. Steinbuch, "Implementation of dynamic programming for optimal control problems with continuous states," *IEEE Trans. Control Syst. Technol*., vol 23, no. 3, pp. 1172–1179, 2015.

[15] P. J. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," *Gen. Syst. Yearbook*, vol. 22, pp. 25–38, 1977.

[16] P. J. Werbos, "A menu of designs for reinforcement learning over time, in Neural Networks for Control," W. T. Miller, R. S. Sutton, and P. J. Werbos Eds., Cambridge, MA: MIT Press, 1990, pp. 67–95.

[17] Z. Ni, H. He, D. Zhao, X. Xu, and D. Prokhorov, "GrDHP: A general utility function representation for dual heuristic dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 26, no. 3, pp. 614–627, Mar. 2015.

[18] H. Modares, F. L. Lewis, and Z. P. Jiang, "H∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 26, no. 10, pp. 2550–2562, Oct. 2015.

[19] C. Mu, Z. Ni, C. Sun, and H. He, "Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems," *IEEE Trans. Cybern*., vol. 47, no. 6, pp. 1460–1470, Jun. 2017.

[20] P. Deptula, J. A. Rosenfeld, R. Kamalapurkar, and W. E. Dixon, "Approximate dynamic programming: combining regional and local state following approximations," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 29, no. 6, pp. 2154–2166, Jun. 2018.

[21] T. Sardarmehni and A. Heydari, "Suboptimal scheduling in switched systems with continuous-time dynamics: a least squares approach," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 29, no. 6, pp. 2167–2178, Jun. 2018.

[22] W. Guo, J. Si, F. Liu, and S. Mei, "Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 29, no. 7, pp. 2794–2807, Jul. 2018.

[23] L. Buşoniu, T. de Bruin, D. Tolic, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Ann. Rev. Control*, vol. 46, pp. 8–28, 2018.

[24] S. Al-Dabooni and D. Wunsch, "The boundedness conditions for model-free HDP(λ)," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 30, no. 7, pp. 1928–1942, Jul 2019.

[25] B. Luo, Y. Yang, D. Liu, and H.-N. Wu, "Event-triggered optimal control with performance guarantees using adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 31, no. 1, Jan. 2020.

[26] B. Zhao, D. Liu, and C. Luo, "Reinforcement learning-based optimal stabilization for unknown nonlinear systems subject to inputs with uncertain constraints," *IEEE Trans. Neural Netw. Learn. Syst*., to be published, DOI: 10.1109/TNNLS.2019.2954983.

[27] D. Liu, H. Li, and D. Wang, "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm," *Neurocomputing*, vol. 110, no. 13, pp. 92–100, Jun. 2013.

[28] M. Abu-Khalaf, F. L. Lewis, and J. Huang, "Policy iterations and the Hamilton-Jacobi-Isaacs equation for the H∞ state feedback control with input saturation," *IEEE Trans. Autom. Control*, vol. 51, pp. 1989–1995, Dec. 2006.

[29] A. Al-Tamimi, M. Abu-Khalaf, and F. L. Lewis, "Adaptive critic designs for discrete-time zero-sum games with application to H∞ control," *IEEE Trans. Syst. Man Cybern. B Cybern*., vol. 37, no. 1, pp. 240–247, Feb. 2007.

[30] K. H. Kim, and F. L. Lewis, "Model-free H∞ control design for unknown linear discrete-time systems via Q-learning with LMI," *Automatica*, vol. 46, pp. 1320–1326, Aug. 2010.

[31] S. Mehraeen, T. Dierks, S. Jagannathan, and M. L. Crow, "Zero-sum two-player game theoretic formulation of affine nonlinear discrete-time systems using neural networks," *IEEE Trans. Cybern*., vol. 43, no. 6, pp. 1641–1655, Dec. 013.

[32] H. Zhang, C. Qin, B. Jiang, and Y. Luo, "Online adaptive policy learning algorithm for H∞ state feedback control of unknown affine nonlinear discrete-time systems," *IEEE Trans. Cybern*., vol. 44, no. 12, pp. 2706–2718, Dec. 2014.

[33] Y. Fu. J. Fu, and T. Chai, "Robust adaptive dynamic programming of two-player zero-sum games for continuous-time linear systems," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 26, no. 12, pp. 3314–3319, Dec. 2015.

[34] L. Cui, H. Zhang, X. Zhang, and Y. Luo, "Data-based adaptive critic design for discrete-time zero-sum games using output feedback," in *Proc. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Paris, France, 2011, pp. 190–195.

[35] B. Luo, Y. Yang, and D. Liu, "Policy iteration Q-learning for data-based two-player zero-sum game of linear discrete-time systems," *IEEE Trans. Cybern*., to be published, DOI:10.1109/TCYB.2020.2970969.

[36] J. Li and Z. Xiao, "H∞ control for discrete-time multi-player systems via off-policy Q-Learning," *IEEE Access*, vol. 8, pp. 28831–28846, Jan. 2020.

[37] Y. Ji and H. Zhou, and B. Bai, "Event-driven-modular adaptive backstepping optimal control for strict-feedback systems through zero-sum differential games," *IEEE Access*, vol. 8, pp. 126511–126522, Jul. 2020.

[38] B. Dong, T. An, F. Zhou, K. Liu, W. Yu, and Y. Li, "Actor-critic-identifier structure-based decentralized neuro-optimal control of modular robot manipulators with environmental collisions," *IEEE Access*, vol. 7, pp. 96148–96165, Jul. 2019.

[39] X.-K. Du, H. Zhao, X.-H. Chang, "Unknown input observer design for fuzzy systems with uncertainties," *Applied Mathematics and Computation*, vol. 266, pp. 108-118, Sep. 2015.

[40] Z.-M. Li, X.-H. Chang, "Robust H∞ control for networked control systems with randomly occurring uncertainties: Observer-based case," ISA Transactions, vol. 83, pp. 13-24, Dec. 2018.

[41] W. Wang, X. Chen, H. Fu, and M. Wu, "Data-driven adaptive dynamic programming for partially observable nonzero-sum games via Q-learning method," *Int. J. Syst. Sci*., vol. 50, no. 7, pp. 1338–1352, 2019.

[42] Y. Liu, H. Zhang, R. yu, Z. Xing, "H∞ tracking control of discrete-time system with delays via data-based adaptive dynamic

programming," *IEEE Trans. Syst. Man Cybern. Syst.,* to be published, DOI: 10.1109/TSMC.2019.2946397.
[43] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, "Integrating state representation learning into deep reinforcement learning," *IEEE Robot. Autom. Lett*., vol. 3, no. 3, 1394–1401, Jul. 2018.
[44] M. N. Howell, G. P. Frost, T. J. Gordon, and Q. H. Wu, "Continuous action reinforcement learning applied to vehicle suspension control," *Mechatronics*, vol. 7, no. 3, pp. 263–276, Apr. 1997.
[45] S. Tognetti, S. M. Savaresi, C. Spelta, and M Restelli, "Batch reinforcement learning for semi-active suspension control," in *Proc. 18th IEEE Intl. Conf. Control Appl*., Saint Petersburg, Russia, 2009, pp. 582–587.
[46] I. O. Bucak, and H. R. Oz, "Vibration control of a nonlinear quarter-car active suspension system by reinforcement learning," *Intl. J. Syst. Sci*., vol. 43, no. 6, pp. 1177–1190, 2012.
[47] M. Akraminia, M. Tatari, M. Fard., M., and R. N. Jazar, "Designing active vehicle suspension system using critic-based control strategy," *Nonl. Eng*., vol. 4, no. 3, pp. 141–154, Jun. 2015.
[48] X. Wang, "Semi-active adaptive optimal control of vehicle suspension with a magnetorheological damper based on policy iteration," *J. Intell. Material Syst. Struct*., vol. 29, no. 2, pp. 255–264, Jan. 2018.
[49] Z. Wang and D. Liu, "Data-based controllability and observability analysis of linear discrete-time systems," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2388–2392, Dec. 2011.
[50] D. Liu, P. yan, and Q. Wei, "Data-based analysis of discrete-time linear systems in noisy environment: Controllability and observability," *Inf. Sci.*, vol. 288, pp. 314–329, Dec. 2014.
[51] M.-B. Radac, R.-E. Precup, E.-L. Hedrea, and I.-C. Mituletu, "Data-driven model-free model-reference nonlinear virtual state feedback control from input-output data," in *Proc. of 2018 26th Mediterranean Conference on Control and Automation*, Zadar, Croatia, 2018, pp. 332–338.
[52] Y. Huang, J. Na, X. Wu, and G. Gao, "Approximation-free control for vehicle active suspension with hydraulic actuator," *IEEE Trans. Ind. Electron*., vol. 65, no. 9, pp. 7258–7267, Sep. 2018.
[53] S. R. Kou, D. L. Elliott, and T. J. Tarn, "Observability of nonlinear systems," *Information and Control*, vol. 22, pp. 89–99, 1973.
[54] C. Ren, S. He, X. Luan, F. Liu, and H. R. Karimi, ''Finite-time L2-gain asynchronous control for continuous-time positive hidden Markov jump systems via T-S fuzzy model approach,'' *IEEE Trans. Cybern.*, 2020, doi: 10.1109/TCYB.2020.2996743.
[55] C. Ren, S. He, "Finite-time stabilization for positive Markovian jumping neural networks," *Applied Mathematics and Computation,* vol. 365, pp. 124631, Jan. 2020.
[56] C. Ren, R. Nie, S. He, "Finite-time positiveness and distributed control of Lipschitz nonlinear multi-agent systems," *Journal of the Franklin Institute*, vol. 356, issue 15, pp. 8080-8092, Oct. 2019.
[57] C. Liu, G. Zhao, J. Wang, H. Wu, H. Li, C. Fietkiewicz, K. A. Loparo "Neural Network-Based Closed-Loop Deep Brain Stimulation for Modulation of Pathological Oscillation in Parkinson's Disease," *IEEE Access,* vol. 8, pp. 161067-161079, Aug. 2020.

MIRCEA-BOGDAN RADAC (Member IEEE) received the Dipl.Ing. degree in systems and computer engineering and the Ph.D. degree in systems engineering from the Politehnica University of Timisoara, Romania, in 2008 and 2011, respectively.

He is the co-author of more than 150 papers published in scientific journals, refereed conference proceedings, and contributions to books. His current research interests cover learning control systems, with a focus on machine learning methods applied in data-driven control.

Dr. Radac is a member of the Romanian Society of Control Engineering and Technical Informatics.

TIMOTEI LALA received the B. Sc. degree from the Politehnica University of Timisoara, in 2019, and he is currently pursuing the M.Sc. degree with the same university.
His current research interests include adaptive dynamic programming, neural networks, and their control applications.