Autonomy and Machine Intelligence in Complex Systems: A Tutorial

Kyriakos G. Vamvoudakis[1], *Member IEEE*, Panos J. Antsaklis[2], *Fellow IEEE*,
Warren E. Dixon[3], *Senior Member IEEE*, João P. Hespanha[1], *Fellow IEEE*, Frank L. Lewis[4], *Fellow IEEE*,
Hamidreza Modares[4], *Student Member IEEE*, Bahare Kiumarsi[4], *Student Member IEEE*

***Abstract*— This tutorial paper will discuss the development of novel state-of-the-art control approaches and theory for complex systems based on machine intelligence in order to enable full autonomy. Given the presence of modeling uncertainties, the unavailability of the model, the possibility of cooperative/non-cooperative goals and malicious attacks compromising the security of teams of complex systems, there is a need for approaches that respond to situations not programmed or anticipated in design. Unfortunately, existing schemes for complex systems do not take into account recent advances of machine intelligence. We shall discuss on how to be inspired by the human brain and combine interdisciplinary ideas from different fields, i.e. computational intelligence, game theory, control theory, and information theory to develop new self-configuring algorithms for decision and control given the unavailability of model, the presence of enemy components and the possibility of network attacks. Due to the adaptive nature of the algorithms, the complex systems will be capable of breaking or splitting into parts that are themselves autonomous and resilient. The algorithms discussed will be characterized by strong abilities of learning and adaptivity. As a result, the complex systems will be fully autonomous, and tolerant to communication failures.**

***Index Terms*— Autonomy, cyber-physical systems, complex systems, networks, machine intelligence.**

I. Introduction

Autonomous systems have been studied for many years with the hope of achieving human-like performance in solving certain problems. There has been a recent resurgence in the field of machine intelligence and autonomy owing to the introduction of new topologies, training algorithms and VLSI implementation techniques. The potential benefits of intelligent systems such as parallel distributed processing, high computation rates, fault tolerance and adaptive capability, have lured researchers from different fields to seek solution to their complicated problems. Autonomy is a capability that enables a particular action of a system to be automatic or, within programmed boundaries, i.e. "self-governing".

[1]K. G. Vamvoudakis, and J. P. Hespanha are with the Center for Control, Dynamical-systems and Computation (CCDC), University of California, Santa Barbara, CA 93106-9560 USA, e-mail: kyriakos@ece.ucsb.edu, hespanha@ece.ucsb.edu. This material is based upon work supported by ARO MURI Grant number W911NF0910553.

[2]P. J. Antsaklis is with the Electrical Engineering Department, University of Notre Dame, Notre Dame, IN 46556, USA, e-mail:antsaklis.1@nd.edu.

[3]W. E. Dixon is with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA, e-mail: wdixon@ufl.edu.

[4]F. L. Lewis, H. Modares, B. Kiumarsi are with The University of Texas at Arlington Research Institute (UTARI), Ft. Worth, TX 76118, USA, e-mail: lewis@uta.edu, modares@uta.edu,kiumarsi@uta.edu. This material is based upon work supported by NSF grant ECCS-1405173, ONR grant N00014-13-1-0562, ARO grant W911NF-11-D-0001, China NNSF grant 61120106011, and China Education Ministry Project 111 (No.B08015).

Decentralization, uncertainty and complexity are several issues that cannot be handled with classical control methods. The power of adaptation and machine intelligence, is the underlying foundation of autonomous technology in order to enable manpower efficiencies, rapid response in harsh environments and enable capabilities beyond human limits and across operational domains. There is a need to integrate human cognitive models to advance human-agent feedback loops, optimize trust/transparency and advance data decision models. Furthermore, networks of autonomous complex systems must have secure communication protocols while their operators expand shared perception and problem solving across multiple agents and advance guidance and control. Complex systems consisting of cooperating/non-cooperating, humans and manned/unmanned airborne or ground vehicles, and their interactions and structure of group communication protocols, can yield unexpected behaviors.

Cyber-physical systems (CPS) have received much attention due to their integration and potential application in a variety of systems such as biological models (e.g. infectious diseases) [76] and social networks [10], [65], human/robot interaction systems [41], Internet [26], transportation systems [45], cyber-security (e.g. malware spreading) [2], [33], [70], [73], [82], [83], unmanned aerial/underwater vehicles [30], sensor networks [63], power networks [80] and mobile robotics [69]. Those systems are large, complex, dynamic, and highly nonlinear in their global behavior.

Synchronization of all the subsystems interacting through a communication network, to a leader behavior has been the main subject of consensus and distributed control algorithms [19], [37], [62], [68] since the work of [81]. The aforementioned results mostly focus on interacting multi-agent systems with only single or double-integrator dynamics, while most of the real applications have general dynamics that are difficult to model [11], [32].

Due to the highly uncertain and dynamic nature of conflict, enabling autonomous agents to gracefully adapt to mission and environmental changes is a very challenging task. These capabilities are necessary against insurgencies, where enemy combatants quickly adapt to new strategies and tactics. Full autonomy will enable mission tailoring, reconfigurability of the control to allow for safe recovery, improved responsiveness and agility, the ability to change missions without exchanging forces, and general adaptability to changing environmental conditions. The ability to synchronize activities between humans and machines, provides an important strategic capability. Teams of humans and

autonomous robots, have common team objectives as well as individual and adversarial member objectives.

The balance between cooperative goals and adversarial behavior forms the basis for team and individual decisions in order to integrate intelligent machines with humans to maximize mission performance in complex and contested environments.

Large complex systems [22], [26], [29], [52], [61], [77] that model the interactions in autonomous systems, are subject to exhaustive modeling, rely on specific network structure and offline computations and are fragile to intentional attacks and purposeful removals of important nodes, hence robustness to uncertainties, random attacks and random failures is an important aspect. There is a need to draw inspiration from recent neuro-physiological studies of the perception mechanism of the human brain and the processing pathways of the visual cortex to control complex systems. Machine learning [79] ideas are being used as an essential component to address problems in multi-agent systems with diverse and selfish interests, traditional algorithmic and distributed systems need to be combined with the understanding of game-theoretic and economic issues [74]. A lot of applications require cooperation of separate agents to achieve global objectives and learning is an ideal approach in the cases where classical optimization techniques are infeasible [84]. A very good book describing the state of decision algorithms in complex systems is given in [56].

In networked systems, an agent affects the agents who are close enough to her. Using a distributed machine learning approach by allowing agents to exchange what they have learned, by sparse communication, the team does better in terms of achieving its goals. However, as these agents respond by adapting their behavior, more agents will feel the consequences and eventually the choices made by a single agent will propagate throughout the entire network community. It has been shown that in order to combine the advantages of adaptation through performance improvement, one has to rely on ideas from an area of machine learning that is called reinforcement learning [79]. Recently approximate dynamic programming [16], [66], [88], [94] and game-theory [12] has been shown to be a powerful tool to solve multi-agent reinforcement learning problems in an adaptive way forward in time while simultaneously guaranteeing optimal performances, [15], [88]. The importance of learning algorithms in real applications has been shown in [1] that proposes applied apprenticeship learning algorithms for learning control policies to helicopters flying in a very wide range of highly aerobatics with a performance as close as to a human expert pilot. There is an extensive research on complex systems coordination from several scientific societies including control systems society [55] and computational intelligence society [91]. The main disadvantages of most of the existing work though is that it requires complete knowledge of the system dynamics and in most of the cases cannot provide any formal optimality guarantees. Two recent surveys are given in [19] and [15] from the control system and from the computational intelligence perspective respectively, where the authors state that distributed multi-agent optimization is a challenging task due to computational complexity issues and modeling unavailability.

A survey of existing cyber-threats in multi-agent systems and models of realistic and rational adversary models are presented in [17] and [18]. Consensus in the presence of persistent adversaries has been focused on detecting, identifying and isolating the failing nodes [64]. These algorithms are computationally expensive and most of the time they use global information and specific graph connectivity. The adversaries can easily drive the system unstable and make the system operate with an undesired behavior. *Thus it is better to fight adversaries in the networks than guard against them*. Most of the cooperative algorithms proposed in the literature either are not optimizing some performance criteria [49] or they optimize global ones and solve the problem offline with complicated Riccati matrix equations [71], [72]. The authors in [13] have evaluated the cost of every agent by considering constant states for the other agents. In [93] the authors propose a controller that suppresses the effect of constant and time varying disturbances by using information of agent's and neighbors' states. In [95] the authors propose a distributed resilient formation control algorithm that consists of a formation control block that is adapted online to minimize a local formation error function and an online learning block to collect information in real time and update the estimates of adversaries. The discussion in [34] states the attack vulnerability of the US electrical grid and how important is to focus on the development of intelligent and robust resilient algorithms.

Structure

The remainder of the tutorial paper is structured as follows. In Section 2 we provide the needed properties to make a system fully autonomous. The properties are presented with connections to CPS. Section 3 presents a unified framework to design optimal trackers and regulators for fully autonomous systems based on reinforcement learning. Section 4 builds upon the ideas in the previous two sections and uses Q-learning based techniques to design model-free optimization approaches to build autonomous complex networks. The algorithms proposed are resilient and robust while guaranteeing an optimal performance. Section 5 presents real experiments of using machine intelligence to build completely autonomous systems. Specifically it presents an experiment, of an optimal path planning algorithm with a Turtlebot. Finally Section 6 concludes and proposes new future directions.

II. The Quest for Autonomy. Are We There Yet? Are CPS a Way to Build Autonomous Systems?

Achieving autonomy has been a dream for many years. The term autonomous system has had different meanings depending on who and when it was used. Attempts to build autonomous vehicles by major corporations and grand challenges by government funding agencies have captured the public's imagination. How much closer to this dream are

we today than we were 25 years ago? The issues surrounding autonomy together with the needed properties that make a system autonomous are briefly discussed and put in context.

A. Introduction to Autonomous Systems and Autonomous Controllers

Systems with ever increasing degrees of autonomy are more prevalent and important today than ever before. Well known examples include Unmanned Aerial Vehicles (UAV), Autonomous Underwater Vehicles (AUV), office and residential buildings that regulate their energy consumption while adapting to the needs of their inhabitants (smart buildings), safety systems and environmentally friendly energy systems in automobiles (smart cars, smart highways). The trend towards increased autonomy has been around for centuries. The recent surge is fueled primarily by technological leaps in hardware and software and successes in integrating tightly the physical and computer worlds, as in the CPS. The recent successes in increasing autonomy in engineering systems are only the beginning of many more to come.

Characteristics that are necessary for high degrees or high levels of autonomy-yes there are levels or degrees of autonomy-are emphasized here. As it will be noted, adaptation and learning, failure diagnosis and identification, control reconfiguration and planning are some of these characteristics.

When one considers humans collaborating with engineered systems, then the overall system that includes humans in the loop may be considered (fully) autonomous with respect to a set of goals. Depending on the role of the humans in the loop and the level of control authority humans exert, the remaining system, the part without the human operator, will have different degrees or levels of autonomy. These ideas are discussed here. It is important to point out that in our discussion of autonomy, the system under consideration always has a set of goals to be achieved and a control mechanism to achieve them. So in our view, *every autonomous system is a control system*. It is useful to think of a system as being surrounded by a boundary separating it from its environment. The system acts upon its environment through its outputs and receives inputs in the form of additional information or disturbances. What the system includes within its boundary, expressed via the particular system model used, depends of course on the goals and the characteristics/properties used to achieve its goals. It is useful to assume as a starting point that the system may also include a human operator who acts as a highly able controller. So in an automobile, if the goal is for example to keep the vehicle inside a lane while traveling with constant speed, the system may consist of the vehicle and the driver where the system attains its goals in the presence of uncertainties/disturbances, such as gust of wind and road incline.

One can envision an autonomous system consisting of two subsystems, a (sub-)system to be controlled (the plant, as it is called in the control literature) and a controller to be designed. Note that this separation of the plant and the controller, which is common in the field of control systems theory, may be somewhat restrictive in autonomous systems, as the assumption that one can separate the plant from the controller may not necessarily be true or easy to satisfy in some cases. However it is a useful concept and it is used here. The controller may include a human in the loop in which case it may achieve full autonomy and we will call such controller *autonomous controller (meaning that such controller is in itself an autonomous system with respect to a new set of goals, that of providing the right kind or control policies to the plant; or alternatively, a controller will be called an autonomous controller if it causes the system to become autonomous)*. The controller may achieve only partial autonomy with or without the human in the loop (meaning that it may need extra help from humans or other systems to attain full autonomy), in which case we will call such controller, *controller with high or low degree of autonomy (or a sub- or a semi-autonomous controller)*. An example of an autonomous controller in an automobile is the system consisting of the human driver and all the control systems in the car with the plant being the vehicle and the goals of the autonomous controller being to provide the right steering and gas pedal commands so the vehicle maintains its course within a lane and at certain (approximately) constant speed. If one considers in this case the controller to consist of just the control systems of the car without the driver then the controller is not autonomous but semi-autonomous.

Here we present a view of autonomous systems and autonomous controllers which is based on the report [3] and earlier work [4]- [6]. One of the issues in the quoted literature was the connection to and the meaning of the term "Intelligent Control". In [7] for example several definitions of Intelligent Control were presented. The difficulty was, and still is, the fact that what constitutes intelligence is not universally agreed upon (today the IQ tests are still controversial and are not widely used around the world) and the issue is still debated, as there are many different strong views. However, throughout [3]- [7] the main point that was consistently made was that *autonomy should be the property of main interest and if for high degrees of autonomy methods that are considered intelligent are used then the name Intelligent Control may be justified*. The expression *"Quest for Autonomy"* makes this exact point that autonomy is the property of interest, while the term Intelligent describes in a eye catching way the methodologies used, not unlike today's term "Smart" which is used in many applications, such as Smart Grid, Smart Phones, Smart Buildings etc. So autonomy is the goal!

B. Autonomous Systems and Autonomous Controllers

It is important to stress again that in our discussion of autonomy, the system under consideration always has a set of goals to be achieved and a control mechanism, a controller, to achieve them. This implies that every autonomous system is a control system.

Autonomous means having the ability and authority for self-government. **A system is autonomous with regard to a set of goals, and with respect to a set of influences**

seen as disturbances (by humans or other systems), if the goals are attained under these disturbances without external interventions. A regular feedback control system for example is autonomous with regard to stability goals and with respect to certain level of external and internal disturbances. This is because stability is maintained even when there are internal system parameter variations and external disturbances. This robustness is due to feedback closed-loop mechanism that compensates for uncertainties; on the other hand, an open-loop system with feed-forward control has none of these robustness properties and no autonomy regarding stability with respect to parameter variations and disturbances.

Alternatively, a perhaps more useful working definition of an autonomous system is that a system has *high or low degree or level of autonomy regarding a goal*. By high degree/level of autonomy it is meant that the degree/level of human intervention (or perhaps intervention by other engineered systems) is low, while by low degree/level of autonomy, a high degree/level of human intervention is implied.

Human in the Loop and Adaptive Autonomy: Humans or other systems may insert themselves at certain levels of the functional hierarchy [3] that correspond to levels of autonomy), and take over control functions. For example, humans may insert themselves to take over planning, FDI, learning functions Or they may insert themselves to take over lower control functions e.g. a driver may want to take over from the ABS system and perform the braking pumping action herself. As mentioned above, this reference to levels connects with the hierarchical functional architecture for the autonomous controller discussed later in this paper.

Autonomous controllers have the ability and authority for self-governance in the performance of control functions. They are composed of a collection of hardware and software, which can perform the necessary control functions, without external intervention, over extended time periods. *Note that a controller will be called autonomous controller when its functions make the system of interest an autonomous system*. Alternatively, an autonomous controller can be seen itself as an autonomous system with goals to apply appropriate controls that make the system of interest autonomous.

There are several **degrees or levels of autonomy**. A fully autonomous controller should perhaps have the ability to perform even hardware repair, if one of its components fails. Note that conventional fixed controllers can be considered to have a low degree of autonomy since they can only tolerate a restricted class of plant parameter variations and disturbances. To achieve a *high degree of autonomy*, the controller must be able to perform a number of functions in addition to the conventional control functions such as tracking and regulation. These additional functions may include the ability to accommodate for drastic system failures, to plan and to learn and operate over extended periods of time.

A hierarchical functional autonomous controller architecture for a future spacecraft is described in [3] and references therein; it is designed to ensure the autonomous operation of the control system and it allows interaction with the pilot/ground station and the systems on board the autonomous vehicle. A command by the pilot or the ground station is executed by dividing it into appropriate subtasks, which are then performed by the controller. The controller can deal with unexpected situations, new control tasks, and failures within limits. To achieve this, high-level decision-making techniques for reasoning under uncertainty and taking actions must be utilized. These techniques, if used by humans, are attributed to *intelligent behavior*. Hence, one way to achieve autonomy, in some applications, is to utilize high-level decision-making techniques, "intelligent" methods, in the autonomous controller. Remember that *autonomy is the objective, and "intelligent" or "smart" controllers are one way to achieve it*.

C. Autonomous Controller Functions

Autonomous control systems must perform well under significant uncertainties in the plant and the environment for extended periods of time and they must be able to compensate for system failures without external intervention.

Such autonomous behavior is a very desirable characteristic of advanced systems. An autonomous controller provides high-level *adaptation* to changes in the plant and environment. To achieve autonomy the methods used for control system design should utilize both:

a) algorithmic-numeric methods, based on the state-of-the-art conventional control, identification, estimation, and communication theory, and

b) decision making-symbolic methods, such as the ones developed in computer science (e.g., automata theory), and specifically in the field of machine learning and Artificial Intelligence (AI).

In addition to supervising and tuning the control algorithms, the autonomous controller must also provide a high degree of tolerance to failures. To ensure system reliability, failures must first be detected, isolated, and identified (and if possible contained), and subsequently a new control law must be designed if it is deemed necessary. The autonomous controller must be capable of planning the necessary sequences of control actions to be taken to accomplish a complicated task. It must be able to interface to other systems as well as with a human operator, and it may need learning capabilities to enhance its performance while in operation. It is for these reasons that advanced planning and learning systems, among others, must work together with conventional control systems in order to achieve autonomy. The need for quantitative methods to model and analyze the dynamical behavior of such autonomous systems presents significant challenges. The development of autonomous controllers requires significant interdisciplinary research effort as it integrates concepts and methods from areas such as control, identification, estimation, and communication theory, computer science, artificial intelligence, and operations research. Autonomous controllers evolve from existing controllers in a natural way fueled by actual needs.

D. Design Methodology - History

Conventional control systems are designed using mathematical models of physical systems. A mathematical model, which captures the dynamical behavior of interest is chosen and then control design techniques are applied, aided by software packages, to design the mathematical model of an appropriate controller. The controller is then realized via hardware or software and it is used to control the physical system. The procedure may take several iterations. The mathematical model of the system must be "simple enough" so that it can be analyzed with available mathematical techniques, and "accurate enough" to describe the important aspects of the relevant dynamical behavior. It approximates the behavior of a plant in the neighborhood of an operating point or a region. The first mathematical model to describe plant behavior for control purposes is attributed to J.C. Maxwell who in 1868 used differential equations to explain instability problems encountered with James Watt's flyball governor; the governor was introduced in 1769 to regulate the speed of steam engine vehicles (the first feedback control mechanism in the historical record is the water clock of Ktesibios, 3rd century BC).

Control theory made significant strides in the past 150 years, with the use of frequency domain methods and Laplace transforms in the 1930s and 1940s and the introduction of the state space analysis in the 1960s. Optimal control in the 1950s and 1960s, stochastic, robust and adaptive control methods in the 1960s to today, have made it possible to control more accurately, significantly more complex dynamical systems than the original flyball governor. The control methods and the underlying mathematical theory were developed to meet the ever-increasing control needs of our technology. The evolution in the control area was fueled by three major needs:

a) The need to deal with increasingly complex dynamical systems.

b) The need to accomplish increasingly more demanding design requirements.

c) The need to attain these design requirements with less precise advanced knowledge of the plant and its environment, that is, the need to control under increased uncertainty.

The need to achieve the demanding control specifications for increasingly complex dynamical systems has been addressed by using more complex mathematical models such as nonlinear and stochastic ones, and by developing more sophisticated design algorithms for, say, optimal control. The use of highly complex mathematical models however, can seriously inhibit our ability to develop control algorithms. Fortunately, simpler plant models, for example linear models, can be used in the control design; this is possible because of the feedback used in control, which can tolerate significant model uncertainties. Controllers can then be designed to meet the specifications around an operating point, where the linear model is valid and then via a scheduler a controller emerges which can accomplish the control objectives over the whole operating range. This is, for example, the method typically used for aircraft flight control. *In autonomous control systems we need to significantly increase the operating range; we must be able to deal effectively with significant uncertainties in models of increasingly complex dynamical systems in addition to increasing the validity range of our control methods*. This will involve the use of intelligent decision-making processes to generate control actions so that a performance level is maintained even though there are drastic changes in the operating conditions.

Figures 1-3 illustrate the evolution of controller towards higher autonomy. There are needs today that cannot be

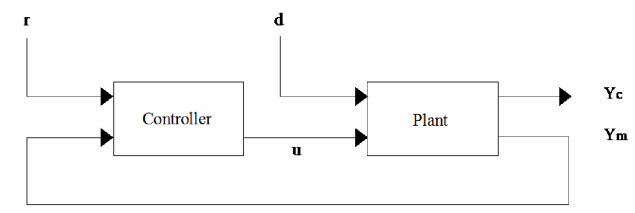

Fig. 1. Conventional Fixed Controller for Robust Control.

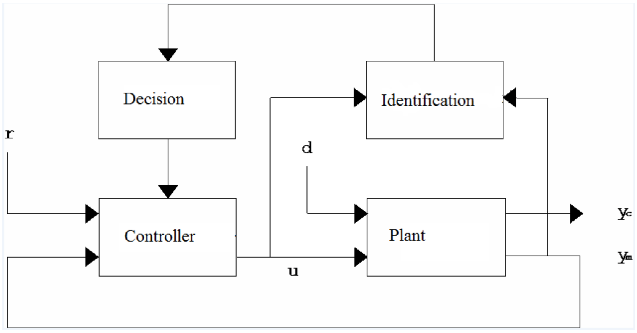

Fig. 2. Conventional Indirect Adaptive Controller.

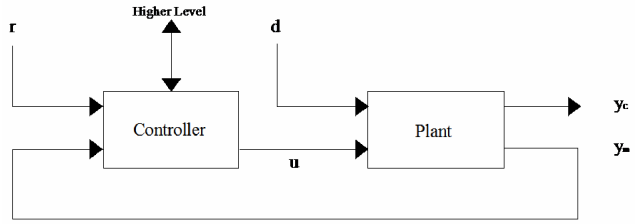

Fig. 3. Highly Adaptive Controller for Autonomous Control.

successfully addressed with the existing conventional control theory. They mainly pertain to the area of uncertainty. Heuristic methods may be needed to tune the parameters of an adaptive control law. New control laws to perform novel control functions should be designed while the system is in operation. Learning from past experience and planning control actions may be necessary. Failure detection and identification is needed. Many of these functions have been performed, in the past, by human operators. *To increase the speed of response, to relieve the pilot from mundane tasks,*

to protect operators from hazards, autonomy is desired. It should be pointed out that several functions seen as parts of an autonomous controller, have been performed in the past by separate systems; examples include fault trees in chemical process control for failure diagnosis and hazard analysis, and control reconfiguration systems in aircrafts, planning the sequence of order execution in steel mills and setting control set-points.

III. Reinforcement Learning for Optimal Tracking and Regulation: A Unified Framework for Autonomous Systems

Reinforcement learning (RL) has been widely used to design feedback controllers for both discrete-time and continuous-time dynamical systems. This technique allows for the design of a class of adaptive controllers that learn optimal control solutions forward in time, and without knowing the full system dynamics. Integral Reinforcement Learning (IRL) and off-policy RL algorithms for continuous-time (CT) systems, and Q-learning and actor-critic structure for discrete-time (DT) systems have been successfully used to learn the optimal control solutions, online in real time. The application of these methods, however, has been mostly limited to the design of optimal regulators. Nevertheless, in practice it is often required to force the states or outputs of the system to track a reference (desired) trajectory. This section proposes a unified framework for both tracking and regulation problems to show how we can develop online model-free RL algorithms to solve the tracking and regulation control problem for both CT and DT systems.

A. Optimal Regulation/Tracking Control of CT Systems

The objective in optimal regulation (tracking) problem is to make the system states go to zero (track a reference trajectory) in an optimal manner by minimizing a performance function.

Consider the nonlinear CT system,

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t),\ t \geqslant 0, \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input, $f(x(t)) \in \mathbb{R}^{n\times n}$ and $g(x(t)) \in \mathbb{R}^{n\times m}$ are the drift and the input dynamics respectively.

For the optimal regulation, a general discounted performance function for the system (1) can be defined as,

$$V(x(t)) = \int_t^\infty e^{-\gamma(\tau-t)}(x^TQx + u^TRu)d\tau,$$

where $Q \geq 0, R > 0$ are user-defined matrices of appropriate dimensions and $\gamma \geqslant 0$ is the discount factor. On the other hand, for the optimal tracking the desired trajectory $x_d(t)$ is assumed to be generated by a command generator function $h_d(x_d(t)) \in \mathbb{R}^n$ such that,

$$\dot{x}_d(t) = h_d(x_d(t)).$$

Define the tracking error as, $e_d(t) = x(t) - x_d(t)$ and a general performance index as,

$$V(e_d(t), x_d(t)) = \int_t^\infty e^{-\gamma(\tau-t)}(e_d^TQe_d + u^TRu)d\tau, \tag{2}$$

where $\gamma > 0$ for the case of tracking. It is shown in [57], [58] that by defining the augmented system state as,

$$X(t) = \begin{bmatrix} e_d(t)^T & x_d(t)^T \end{bmatrix}^T \in \mathbb{R}^{2n}$$

and the augmented system dynamics become,

$$\dot{X}(t) = F(X(t)) + G(X(t))u(t) \tag{3}$$

with some nonlinear functions $F(X(t))$ and $G(X(t))$. Moreover, the performance (2) in terms of the tracking error and the control input becomes,

$$V(X(t)) = \int_t^\infty e^{-\gamma(\tau-t)}(X^TQ_TX + u^TRu)d\tau, \tag{4}$$

with $Q_T := \begin{bmatrix} Q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ and $\mathbf{0}$ a zero matrix of appropriate dimensions.

The differential equivalent of (4) gives,

$$V_x^T(F(X) + G(X)u) - \gamma V(X) + X^TQ_TX + u^TRu = 0 \tag{5}$$

where $V_x := \frac{\partial V(x)}{\partial X}$. Therefore, the tracking problem is transformed into a regulation problem with the augmented system (3) and the performance function (4).Both tracking and regulation problems are now defined in one framework. In fact, in both of these problems, the goal is to find a control input for a system in form (3) by minimizing the general performance index (4). For the regulation problem in the system (3), and performance (4), $X(t) := x(t) \in \mathbb{R}^n$ and for the tracking problem, $X(t) := \begin{bmatrix} e_d(t)^T & x_d(t)^T \end{bmatrix}^T \in \mathbb{R}^{2n}$.

1) IRL Method for Optimal Regulation/Tracking of CT Systems: IRL [86]- [88] was the first RL algorithm developed to formulate online optimal adaptive control methods for continuous-time systems. These methods find the optimal control solution online in real time without knowing the system drift dynamics $f(x)$.

The idea is to write the performance function (4) in the integral reinforcement form as,

$$V(X(t-T)) = \int_{t-T}^t e^{-\gamma(\tau-t+T)}\big(X^TQ_TX + u^TRu\big)d\tau + e^{-\gamma T}V(X(t)),$$

with $T > 0$ a sampling constant. This gives a unified tracking/regulation IRL Bellman equation. Using this Bellman equation, the following IRL-based algorithm can be used to solve the optimal tracking/regulation problem using only partial knowledge about the system dynamics.

Algorithm 1: Online IRL algorithm for optimal regulation/tracking control of CT systems

1: **procedure**

2: Given a control input $u_i(X)$, where $i \in \mathbb{N}$, find $V_i(X)$ using,

$$V_i(X(t-T)) = \int_{t-T}^t e^{-\gamma(\tau-t+T)}\big(X^TQ_TX + u_i^TRu_i\big)d\tau + e^{-\gamma T}V_i(X(t))$$

3: Update the control policy using,

$$u_{i+1}(X) = -\frac{1}{2}R^{-1}G^T(X)V_x, \tag{6}$$

4: $i = i + 1$

5: **end procedure**

Synchronous policy iteration [85] can also be used to learn the optimal policy. Algorithm requires the knowledge of the input dynamics $G(X)$. The off-policy IRL algorithm [38], [39], [53] can be extended to the discounted optimal control to avoid requirement of the knowledge of $G(X)$.

2) Off-policy IRL Method for Optimal Regulation/Tracking of CT Systems: Off-policy IRL algorithm was first presented in [38], [39], [53] to develop optimal regulators for completely unknown CT systems. Inspired by [38], [39], [53] the system dynamics (3) is first written as,

$$\dot{X} = F(X) + G(X)u_i + G(X)(u - u_i). \tag{7}$$

Taking the derivative across the closed-loop trajectories (7) and using (5) and (6) one has,

$$\begin{aligned}\dot{V}_i &= V_{Xi}^T(F + Gu_i) + V_{Xi}^T G(u - u_i)\\ &= \gamma V_i - X^T Q_T X - u_i^T R u_i - 2u_{i+1}^T R(u - u_i).\end{aligned} \tag{8}$$

Multiplying both sides of (8) by $e^{-\gamma(\tau - t)}$ and integrating both sides yields the following off-policy IRL Bellman equation,

$$\begin{aligned}&e^{-\gamma T}V_i(X(t+T)) - V_i(X(t)) =\\ &-\int_t^{t+T} e^{-\gamma(\tau-t)}\big(X^T Q_T X + u_i^T R u_i\big)d\tau\\ &\quad -\int_t^{t+T} e^{-\gamma(\tau-t)}u_{i+1}R(u - u_i)d\tau.\end{aligned}$$

This off-policy regulation/tracking Bellman equation can be for V_i and u_{i+1} simultaneously without requiring any knowledge of the system dynamics. The following algorithm uses this Bellman equation to solve the optimal regulation/tracking problem without requiring any knowledge of the system dynamics,

Algorithm 2: Online Off-policy RL algorithm for solving the tracking Hamilton-Jacobi equation

1: **procedure**

2: Solve the following Bellman equation for V_i, and u_{i+1} simultaneously,

$$\begin{aligned}&e^{-\gamma T}V_i(X(t+T)) - V_i(X(t)) =\\ &-\int_t^{t+T} e^{-\gamma(\tau-t)}\big(X^T Q_T X + u_i^T R u_i\big)d\tau\\ &\quad -\int_t^{t+T} e^{-\gamma(\tau-t)}u_{i+1}R(u - u_i)d\tau\end{aligned}$$

3: Stop if a stopping criterion is met, otherwise set $i = i + 1$ and goto 2.

4: **end procedure**

B. Optimal Regulation/Tracking of DT Systems

Consider the nonlinear DT system (similarly to (1)) as,

$$x(k+1) = f(x(k)) + g(x(k))u(k), k \in \mathbb{N}.$$

The performance function for the optimal regulation DT problem can be defined as,

$$V(x(k)) = \sum_{i=k}^{\infty}\gamma^{i-k}\big(x(i)^T Q x(i) + u(i)^T R u(i)\big), \tag{9}$$

where $Q \geq 0, R > 0$ are user-defined matrices of appropriate dimensions and $0 < \gamma < 1$ is a discount factor. Similarly to the CT systems, for the tracking problem of DT systems, the desired reference trajectory is produced by the command generator mode,

$$r(k+1) = \psi(r(k)).$$

The performance function (9) for the tracking problem is written as,

$$V(x(k)) = \sum_{i=k}^{\infty}\gamma^{i-k}\big(e(i)^T Q e(i) + u(i)^T R u(i)\big)$$

where $e(k) := x(k) - r(k)$ is the tracking error. The augmented system is then defined by following the developments in [47] as,

$$X(k+1) = F(X(k)) + G(X(k))u_k, \tag{10}$$

for some nonlinear functions $F(X(k)), G(X(k))$.

By using the augmented system (10), the discounted performance function for the DT problem can be defined as,

$$V(X(k)) = \sum_{i=k}^{\infty}\gamma^{i-k}\big(X(i)^T Q_1 X(i) + u(i)^T R u(i)\big) \tag{11}$$

where $Q_1 := \begin{bmatrix} Q & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}$. Now equation (11) can be written as the following Bellman equation,

$$\begin{aligned}V(X(k)) = \gamma V(X(k+1)) + X(k)^T Q_1 X(k)\\ + u(k)^T R u(k).\end{aligned} \tag{12}$$

Now, both tracking and regulation problems are defined in one framework. For the regulation problem in the system (10) and performance (11), we need to set, $X(k) = x(k) \in \mathbb{R}^n$ and for the tracking $X(k) = \begin{bmatrix} e(k)^T & r(k)^T\end{bmatrix}^T \in \mathbb{R}^{2n}$. See [47] and [48] for further developments.

1) Q-learning for Optimal Regulation/Tracking Control of Linear DT Systems: Following the developments of [48], [92] we assume that the augmented system (10) is linear and has the form,

$$X(k+1) = AX(k) + Bu(k) \tag{13}$$

with a performance given by (11). Now we should define the Q-function as,

$$\begin{aligned}Q(X(k), u(k)) = X(k)^T Q_1 X(k) + u(k)^T R u(k)\\ + \gamma V(X(k+1)).\end{aligned} \tag{14}$$

After substituting the value function $V(X(k)) = X(k)^T P X(k)$ with P the solution to the Riccati equation, and the system (13) in (14) one has,

$$\begin{aligned} &Q(X(k), u(k)) = \\ &\begin{bmatrix} X(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 + \gamma A^T P A & \gamma A^T P B \\ \gamma B^T P A & R + \gamma B^T P B \end{bmatrix} \begin{bmatrix} X(k) \\ u(k) \end{bmatrix} \\ &:= Z(k)^T H Z(k), \end{aligned} \tag{15}$$

where $Z(k) = \begin{bmatrix} X(k) \\ u(k) \end{bmatrix}$ and $H = \begin{bmatrix} H_{XX} & H_{Xu} \\ H_{uX} & H_{uu} \end{bmatrix}$.

Using (14) and (15) for updating the Q-function and the derivative of the Q-function for finding an improved control policy, the following algorithm is developed,

Algorithm 3: Policy Iteration solution using the Linear Quadratic Tracking (LQT) Q-function

1: **procedure**

2: Policy evaluation,

$$\begin{aligned} &Z(k)^T H^{j+1} Z(k) = X(k)^T Q_1 X(k) \\ &+ (u(k)^j)^T R u(k)^j + \gamma Z(k+1)^T H^{j+1} Z(k+1) \end{aligned}$$

3: Policy improvement,

$$u(k)^{j+1} = -\left(H_{uu}^{-1}\right)^{j+1} H_{uX}^{j+1} X(k)$$

4: Stop if a stopping criterion is met, otherwise set $j = j + 1$ and goto 2.

5: **end procedure**

Note that Algorithm 3 does not require any knowledge of the system dynamics.

2) Actor-Critic Based RL Algorithm for Regulation/Tracking of Nonlinear DT Systems: The actor-critic [48] structure can be used to solve the optimal control regulation/tracking problem online for nonlinear DT systems. The critic network estimates the value function. The actor represents a control policy and is updated to minimize the value function.

The critic network: Using (12), the prediction error of the regulation/tracking Bellman equation is defined as,

$$\begin{aligned} e_c(k) = \gamma \hat{V}(X(k)) + X(k)^T Q_1 X(k) = u(k)^T R u(k) \\ - \hat{V}(X(k-1)), \end{aligned} \tag{16}$$

where $\hat{V}(X(k)) = W_1^T \phi(X(k))$ is the critic network with W_1 the weight vector and ϕ the activation function. It is desired to select the weights of the critic network to minimize the Bellman error (16). The update law for the critic weights can be performed using least squares or gradient descent methods.

The actor network: The actor network is written as a neural network of the form,

$$\hat{u}(X(k)) = W_2^T \varphi(X(k)),$$

where W_2 is the weight vector and φ is the activation function. The actor is updated to minimize the value function. This can be done by minimizing the error between a target control input (which is obtained by minimizing the value function) and the actual control input which is applied to the system. Let the current estimation of the value function be $\hat{V}$. Then, the target control input is obtained by minimizing the right-hand-side of (12) which is,

$$\tilde{u}(X(k)) = -\frac{1}{2} G(X(k))^T R^{-1} \frac{\partial \hat{V}(X(k+1))}{\partial X(k+1)}.$$

However, to obtain the value at time $k+1$, the states are required to be predicted by using a model network. But, we do not use a model network to predict the future value. Rather, we store the previous value of the system state and the state value and try to minimize the error between the target control input and the actual control input given current actor and critic estimate weights while the previous stored state $X(k-1)$ is used as the input to the actor and critic. That is to minimize,

$$e_a(k) = \tilde{u}(X(k-1)) - \hat{u}(k, k-1), \tag{17}$$

where $\hat{u}(k, k-1) := \hat{u}(X(k-1), W_2(k))$ is the output of the actor network at time $k-1$ if the current network weights are used. It is desired to select the weights of the actor network to minimize (17). The update law for the actor weights can be performed using least squares or gradient descent methods.

IV. Model-Free Plug-n-Play Optimization Techniques to Design Autonomous and Resilient Complex Systems

This section will build upon the developments in the previous sections and will show how to use machine intelligence and especially Q-learning based approaches inspired by the work of [90] to develop model-free approaches. Q-learning was the first provably convergent direct optimal adaptive control algorithm and is a model-free reinforcement learning technique developed primarily for discrete-time systems [90]. The centralized Q-function in [90] depends on both states and decision makers (controls) which means that it already includes the information about the system and the utility functions. Since it is more difficult to compute policies from value functions than Q-functions, Q-learning is preferred to value functions based algorithms (heuristic dynamic programming [92]). Specifically, Q-learning can be used to find an optimal action-selection policy based on measurements of previous state and action observations controlled using a "non-optimal policy". It learns an action-dependent value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. When such an action-dependent value function is learned, the optimal policy can be computed easily. The biggest strength of Q-learning is that it does not require a model of the environment. It has been proven in [90] that for any finite Markov Decision Process, Q-learning eventually finds an optimal policy. To guarantee the convergence of the iterative Q function in [90], the learning rate sequence of the Q-learning algorithm is constrained to a special class of positive series, where the sum of the positive series is infinite and the corresponding quadratic sum is required to be finite. Because of the strong constraints in

the learning rate sequence, the convergence properties of the Q-learning algorithms are also constrained. Q-learning at its simplest uses tables to store data. This very quickly loses viability with increasing levels of complexity and dimensionality of the system. This problem can be solved effectively by using adapted neural networks as universal approximators. Specifically, Q-learning can be improved by using the universal function approximation property of neural networks and especially in the context of approximate dynamic programming [92] or neuro-dynamic programming [14] that allows us to solve difficult optimization problems online and forward in time. It is hence possible to apply the algorithm to larger problems, even when the state space is continuous, and infinitely large.

In continuous-time systems, things are harder and most of the times one has to rely on discretization of the state and the action space to apply such techniques, and as such lose important information during discretization. Some early work on continuous-time systems learning was done in [9], [27]. The authors in [54] have established connections between Q-learning and nonlinear control of continuous-time models with general state and action space by observing that the Q-function developed in [90] is an extension of the Hamiltonian that appears in the minimum principle. A variant of Q-learning that provides a model free approach for continuous-time system has been proposed in [40] where the authors have performed a policy iteration algorithm to use repeatedly state and input information on some small fixed time intervals. An ϵ-integral Q function has been used to propose an ϵ-approximate Q-learning framework for solving the linear quadratic regulator problem of continuous-time systems in [50] but the authors can guarantee convergence and uniform-ultimate boundedness stability only when the initial policy is stabilizing. We note that the Hamilton-Jacobi equations cannot be solved for complex nonlinear systems with nonstandard, high performance measures. However, neural network approximation techniques allow one to solve these design equations approximately for complex systems with actuator constraints and high-performance maneuvering.

The below model-free technique can be easily extended to "single-player" optimization (optimal control), multi-agent systems without adversaries etc. Here we will showcase the scenario of uncertain agents in complex systems being attacked by persistent adversaries. We shall use machine-learning ideas, i.e. Q-learning based techniques, to find model-free plug-n-play algorithms.

A. Problem Formulation

We consider a networked-system $\mathcal{G}$, consisting of N agents each modeled $\forall i \in \mathcal{N} := \{1, \ldots, N\}$ by the following dynamics,

$$\dot{x}_i(t) = Ax_i(t) + B_i u_i(t) + D_i v_i(t),\ t \geqslant 0, \tag{18}$$

where $x_i(t) \in \mathbb{R}^n$ is a measurable state vector, $u_i(t) \in \mathbb{R}^{m_i}$, $i \in \mathcal{N} := \{1, \ldots, N\}$ is each control input (or minimizing player as we shall see later), $v_i(t) \in \mathbb{R}^{l_i}$, $i \in \mathcal{N} := \{1, \ldots, N\}$ is each adversarial input (or maximizing player as we shall see later), and $A \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m_i}$, $D_i \in \mathbb{R}^{n\times l_i}$, $i \in \mathcal{N}$ are the plant, control input and adversarial input matrices respectively that will be considered uncertain/unknown. It is assumed that the pairs (A, B_i), $\forall i \in \mathcal{N}$ are controllable. We have a total of $2N$ players/controllers that select values for $u_i(t)$, $t \geqslant 0$, $i \in \mathcal{N}$ and $v_i(t)$, $t \geqslant 0$, $i \in \mathcal{N}$. The agents in the network seek to cooperatively asymptotically track the state of a leader node/exosystem with dynamics $\dot{x}_0 = Ax_0$, i.e. $x_i(t) \to x_0(t), \forall i \in \mathcal{N}$ while *simultaneously satisfying user-defined distributed performances*.

Figure 4 shows one such networked system $\mathcal{G}$ consisting of 10 agents with a leader node 0 pinned to node number 6.

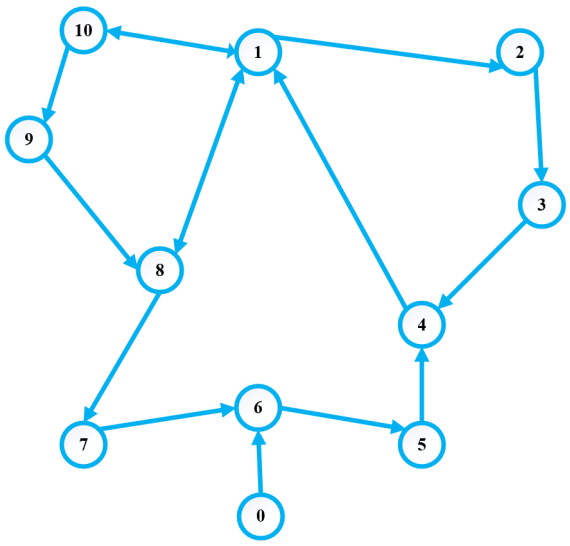

Fig. 4. A networked system $\mathcal{G}$ with 10 agents and a leader node 0 pinned to agent 6.

Now we shall proceed to the design of the user-defined *distributed performances*. For that reason, we shall define the following *neighborhood tracking error* for every agent,

$$e_i := \sum_{j\in\mathcal{N}_i} (x_i - x_j) + g_i(x_i - x_0), \forall i \in \mathcal{N}, \tag{19}$$

where $g_i \in \mathbb{R}^+$ is the pinning gain that shows if an agent is pinned to the leader node (i.e. $g_i \neq 0$) and it is nonzero for at least one node.

The dynamics of (19) are given by,

$$\begin{aligned}\dot{e}_i = Ae_i + (d_i + g_i)(B_i u_i + D_i v_i) \\ - \sum_{j\in\mathcal{N}_i} (B_j u_j + D_j v_j), \forall i \in \mathcal{N},\end{aligned} \tag{20}$$

with $e_i \in \mathbb{R}^n$.

The cost functionals associated to each agent $i \in \mathcal{N}$, that depend on the tracking error e_i, the control u_i, the controls in the neighborhood of agent i given as, $u_{\mathcal{N}_i} := \{u_j : j \in \mathcal{N}_i\}$, the adversarial input v_i and the adversarial inputs in the neighborhood of agent i given as $v_{\mathcal{N}_i} := \{v_j : j \in \mathcal{N}_i\}$, have

the following form,

$$\mathcal{J}_i(e_i(0);u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i}) = \frac{1}{2}\int_0^\infty \big(e_i^T H_i e_i + (u_i^T R_{ii} u_i - \gamma_{ii}^2 v_i^T v_i) + \sum_{j\in\mathcal{N}_i}(u_j^T R_{ij} u_j - \gamma_{ij}^2 v_j^T v_j)\big)dt,\ \forall i\in\mathcal{N}, \quad (21)$$

with user defined matrices $H_i \geq 0$, $R_{ii} > 0$, $R_{ij} \geq 0,\ \forall i,j \in \mathcal{N}$ of appropriate dimensions, and $\gamma_{ii}, \gamma_{ij} \in \mathbb{R}^+,\ \forall i \in \mathcal{N}$.

Hence, given a strongly connected graph $\mathcal{G}$, we are interested in finding a graphical Nash equilibrium [12], [84], that is translated to a saddle point u_i^*, v_i^*, for every agent $i \in \mathcal{N}$ in the sense that,

$$\mathcal{J}_i(e_i(0);u_i^*,u_{\mathcal{N}_i}^*,v_i,v_{\mathcal{N}_i}^*) \leqslant \mathcal{J}_i(e_i(0);u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*) \leqslant \mathcal{J}_i(e_i(0);u_i,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*),\ \forall u_i, v_i\ i \in \mathcal{N}. \quad (22)$$

This can be expressed by the following coupled distributed optimization problems,

$$\mathcal{J}_i(e_i(0);u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*) = \min_{u_i}\max_{v_i}\mathcal{J}_i(e_i(0);u_i,u_{\mathcal{N}_i}^*,v_i,v_{\mathcal{N}_i}^*),\ \forall i\in\mathcal{N},$$

given the dynamics in (20).

Thus, the ultimate goal is to find the distributed optimal value functions $V_i^*,\ \forall i \in \mathcal{N}$ defined by,

$$V_i^*(e_i(t)) := \min_{u_i}\max_{v_i}\int_t^\infty \frac{1}{2}\big(e_i^T H_i e_i + (u_i^T R_{ii} u_i - \gamma_{ii}^2 v_i^T v_i) + \sum_{j\in\mathcal{N}_i}(u_j^T R_{ij} u_j - \gamma_{ij}^2 v_j^T v_j)\big)dt,\ \forall t, \forall i\in\mathcal{N}, \quad (23)$$

but without any information of the system matrices A, B_i, $D_i, \forall i \in \mathcal{N}$ and pinning gains $g_i,\ \forall i \in \mathcal{N}$. First we will define the Hamiltonian associated with each agent's neighborhood tracking error (20) and each V_i^* given in (23) as follows,

$$\mathcal{H}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\frac{\partial V_i^*}{\partial e_i}) = \frac{\partial V_i^*}{\partial e_i}^T\Big(Ae_i + (d_i+g_i)(B_iu_i + D_iv_i) - \sum_{j\in\mathcal{N}_i}(B_ju_j + D_jv_j)\Big) + \frac{1}{2}e_i^T H_i e_i + \frac{1}{2}\Big(e_i^T H_i e_i + (u_i^T R_{ii} u_i - \gamma_{ii}^2 v_i^T v_i) + \sum_{j\in\mathcal{N}_i}(u_j^T R_{ij} u_j - \gamma_{ij}^2 v_j^T v_j)\Big), \forall e_i, u_i, v_i \forall i \in \mathcal{N}. \quad (24)$$

After employing the stationarity conditions, in the Hamiltonian (24) we can find the saddle-point solution, i.e. $\frac{\partial \mathcal{H}_i(\cdot)}{\partial u_i} = 0$, and $\frac{\partial \mathcal{H}_i(\cdot)}{\partial v_i} = 0$. Hence, the optimal control for each $i \in \mathcal{N}$ can be found to be,

$$u_i^*(e_i) = \arg\min_{u_i}\mathcal{H}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\frac{\partial V_i^*}{\partial e_i}) = -(d_i+g_i)R_{ii}^{-1}B_i^T\frac{\partial V_i^*}{\partial e_i},\ \forall e_i, \quad (25)$$

and the worst case adversarial input can be found to be,

$$v_i^*(e_i) = \arg\max_{v_i}\mathcal{H}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\frac{\partial V_i^*}{\partial e_i}) = \frac{(d_i+g_i)}{\gamma_{ii}^2}D_i^T\frac{\partial V_i^*}{\partial e_i},\ \forall e_i. \quad (26)$$

The saddle-point solution (25)-(26) should satisfy the appropriate coupled Hamilton-Jacobi equations,

$$\mathcal{H}_i(e_i,u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*,\frac{\partial V_i^*}{\partial e_i}) = 0, \forall i \in \mathcal{N}. \quad (27)$$

The value functions can be represented as quadratic in the neighborhood tracking error, i.e. $V_i^*(e_i) : \mathbb{R}^n \to \mathbb{R}$,

$$V_i^*(e_i) = \frac{1}{2}e_i^T P_i e_i,\ \forall e_i, \forall i \in \mathcal{N}, \quad (28)$$

where $P_i \in \mathbb{R}^{n\times n},\ \forall i \in \mathcal{N}$ are the unique symmetric positive definite matrices that solve the following complicated distributed coupled equations,

$$\begin{aligned} &e_i^T P_i\Big(Ae_i - (d_i+g_i)^2(B_iR_{ii}^{-1}B_i^T - \frac{1}{\gamma_{ii}^2}D_iD_i^T)P_ie_i \\ &+ \sum_{j\in\mathcal{N}}(d_j+g_j)(B_jR_{jj}^{-1}B_j^T - \frac{1}{\gamma_{ii}^2}D_jD_j^T)P_je_j\Big) \\ &+ \Big(Ae_i - (d_i+g_i)^2(B_iR_{ii}^{-1}B_i^T - \frac{1}{\gamma_{ii}^2}D_iD_i^T)P_ie_i \\ &+ \sum_{j\in\mathcal{N}}(d_j+g_j)(B_jR_{jj}^{-1}B_j^T - \frac{1}{\gamma_{ii}^2}D_jD_j^T)P_je_j\Big)^T P_ie_i \\ &+ \sum_{j\in\mathcal{N}_i}(d_j+g_j)^2 e_j^T P_j(B_jR_{jj}^{-T}R_{ij}R_{jj}^{-1}B_j^T \\ &\qquad - \frac{\gamma_{ij}^2}{\gamma_{jj}^4}D_jD_j^T)P_je_j \\ &+ (d_i+g_i)^2 e_i^T P_i(B_iR_{ii}B_i^T - \frac{1}{\gamma_{ii}^2}D_iD_i^T)P_ie_i \\ &\qquad + e_i^T H_i e_i = 0, \forall i \in \mathcal{N}. \end{aligned} \quad (29)$$

By using (28), the optimal control (25) for every agent $i \in \mathcal{N}$ can be written as,

$$u_i^*(e_i) = -(d_i+g_i)R_{ii}^{-1}B_i^T P_i e_i,\ \forall e_i, \quad (30)$$

and the worst case adversarial input (26) for every agent $i \in \mathcal{N}$ can be written as,

$$v_i^*(e_i) = \frac{(d_i+g_i)}{\gamma_{ii}^2}D_i^T P_i e_i,\ \forall e_i. \quad (31)$$

It is important to note that the equations (29), (30), (31), are highly coupled, difficult to solve due to the cross terms $e_i^T e_j$ and require complete knowledge of the system matrix A, the input matrices $B_i,\ D_i,\ i \in \mathcal{N}$ and leader connection information g_i. we shall show a new *cooperative Q-learning* based approach to solve the graphical Nash game problem without any information of the system dynamics while attenuating persistent adversarial inputs, by adjusting parameters in an adaptive way.

B. Q-learning Based Approach

The value functions (28) need to be parameterized as functions of the neighborhood tracking error e_i, the controls u_i and $u_{\mathcal{N}_i}$ and the adversarial inputs v_i and $v_{\mathcal{N}_i}$ to represent the distributed Q-function, i.e. cooperative learning, for each agent in the game. The optimal value given by (28) after adding the Hamiltonian from (24) can be written as the following distributed Q-function or *action-dependent value* $Q_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i}):\mathbb{R}^{n+m_i+l_i+\sum_{j\in\mathcal{N}_i}(m_j+l_j)}\to\mathbb{R}$,

$$\begin{aligned}
&Q_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i}):=V_i^*(e_i)\\
&\quad+\mathcal{H}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\frac{\partial V_i^*}{\partial e_i})\\
&=V_i^*(e_i)+\frac{\partial V_i^*}{\partial e_i}^T\Big(Ae_i+(d_i+g_i)(B_iu_i+D_iv_i)\\
&\quad-\sum_{j\in\mathcal{N}_i}(B_ju_j+D_jv_j)\Big)\\
&+\frac{1}{2}\Big(e_i^TH_ie_i+(u_i^TR_{ii}u_i-\gamma_{ii}^2v_i^Tv_i)\\
&\quad+\sum_{j\in\mathcal{N}_i}(u_j^TR_{ij}u_j-\gamma_{ij}^2v_j^Tv_j)\Big),\\
&\qquad\forall e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\ \forall i\in\mathcal{N},
\end{aligned}\tag{32}$$

where $\mathcal{H}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i},\frac{\partial V_i^*}{\partial e_i})$ is given by (24) and the optimal cost is $V_i^*(e_i)=e_i^TP_ie_i,\forall i\in\mathcal{N}$. Each agent's distributed Q-function (32) can be written in a compact quadratic in the neighborhood tracking error e_i, controls $u_i,u_{\mathcal{N}_i}$ and adversarial inputs $v_i,v_{\mathcal{N}_i}$ distributed form as in (33) (next page), where $\mathbf{0}$ are zero matrices of appropriate dimensions, $\text{diag}\left(R_{ij}\right)_{j\in\mathcal{N}_i}$, $\text{diag}\left(\gamma_{ij}\right)_{j\in\mathcal{N}_i}$ are stacked diagonal matrices, $\text{col}\big(B_j^TP_i\big)_{j\in\mathcal{N}_i}$, $\text{col}\big(D_j^TP_i\big)_{j\in\mathcal{N}_i}$ are stacked column matrices, and $\text{row}\big(P_iB_j\big)_{j\in\mathcal{N}_i}$, $\text{row}\big(P_iD_j\big)_{j\in\mathcal{N}_i}$ are stacked row matrices.

In (33), the equivalences of $Q^i_{(\cdot)}$ are straightforward e.g. $Q^i_{\mathrm{e_ie_i}}=P_i+H_i+P_iA+A^TP_i$, $Q^i_{\mathrm{e_iu_i}}=(d_i+g_i)P_iB_i$, $Q^i_{\mathrm{e_iv_i}}=(d_i+g_i)P_iD_i$, $Q^i_{\mathrm{u_iu_i}}=R_{ii}$, $Q^i_{\mathrm{v_iv_i}}=\gamma_{ii}^2$ etc. and positive definite matrices $\bar{Q}^i\in\mathbb{R}^{(n+m_i+l_i+\sum_{j\in\mathcal{N}_i}(m_j+l_j))\times(n+m_i+l_i+\sum_{j\in\mathcal{N}_i}(m_j+l_j))},\forall i\in\mathcal{N}$. A model free formulation of (30) can be found by solving $\frac{\partial Q_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i})}{\partial u_i}=0$ to write,

$$\begin{aligned}
u_i^*(e_i)&=\arg\min_{u_i}Q_i(e_i,u_i,u_{\mathcal{N}_i}^*,v_i,v_{\mathcal{N}_i}^*)\\
&=-(Q^i_{\mathrm{u_iu_i}})^{-1}Q^i_{\mathrm{u_ie_i}}e_i,\ \forall i\in\mathcal{N},
\end{aligned}\tag{34}$$

and a model free formulation of (31) can be found by solving $\frac{\partial Q_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i})}{\partial v_i}=0$ to write,

$$\begin{aligned}
v_i^*(e_i)&=\arg\max_{v_i}Q_i(e_i,u_i,u_{\mathcal{N}_i}^*,v_i,v_{\mathcal{N}_i}^*)\\
&=(Q^i_{\mathrm{v_iv_i}})^{-1}Q^i_{\mathrm{v_ie_i}}e_i,\ \forall i\in\mathcal{N},
\end{aligned}\tag{35}$$

We shall use machine intelligence and especially an actor/critic framework to find (34)-(35).

The critic NN will approximate the distributed Q-function (33), the control actor NN will approximate the optimal controller (34) and the adversarial input actor NN will approximate the worst-case adversarial input (35) of each agent $i\in\mathcal{N}$. Specifically $Q_i^*(e_i,u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*)$ can be written as,

$$Q_i^*(e_i,u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*)=\frac{1}{2}z_i^T\bar{Q}^iz_i,\ \forall i\in\mathcal{N},\tag{36}$$

or

$$Q_i^*(e_i,u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*)=\frac{1}{2}\text{vech}\big(\bar{Q}^i\big)^T\big(z_i\otimes z_i\big),\ \forall i\in\mathcal{N},$$

where, $z_i:=\begin{bmatrix}e_i^T & u_i^T & u_{\mathcal{N}_i}^T & v_i^T & v_{\mathcal{N}_i}^T\end{bmatrix}^T$.

Now denote the $\text{vech}\big(\bar{Q}^i\big)\in\mathbb{R}^{\frac{1}{2}(n+m_i+l_i+\sum_{j\in\mathcal{N}_i}(m_j+l_j))(n+m_i+l_i+\sum_{j\in\mathcal{N}_i}(m_j+l_j)+1)}$ as a half-vectorization of the matrix $\bar{Q}^i$ that returns a column vector by stacking the elements of the diagonal and upper triangular part of the symmetric matrix into a vector where the off-diagonal elements are taken as $2\bar{Q}^i_{\kappa_1\kappa_2}$ and $\otimes$ denotes the Kronecker product quadratic polynomial basis vector.

By denoting as $W_{ic}:=\text{vech}\big(\bar{Q}^i\big)$ we can write (36) in a compact form as,

$$Q_i^*(e_i,u_i^*,u_{\mathcal{N}_i}^*,v_i^*,v_{\mathcal{N}_i}^*)=W_{ic}^T\big(z_i\otimes z_i\big),\ \forall i\in\mathcal{N},$$

with W_{ic} the ideal weights. Next we should estimate Q_i^*, u_i^* and v_i^* with the following actual values for the critic NN with $\hat{W}_{ic}:=\text{vech}\big(\hat{\bar{Q}}^i\big)$,

$$\hat{Q}_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i})=\hat{W}_{ic}^T\big(z_i\otimes z_i\big),\ \forall i\in\mathcal{N},\tag{37}$$

where $\hat{W}_{ic}$ are the estimated critic weights.

The control actor NN is given as

$$\hat{u}_i(e_i)=\hat{W}_{ia}^Te_i,\ \forall i\in\mathcal{N},\tag{38}$$

where $\hat{W}_{ia}\in\mathbb{R}^{n\times m_i}$ are the estimated actor weights, note also that the neighborhood tracking error, e_i, is serving as an activation function for the action NN. Note that the optimal value for each control actor NN is given by (34).

Finally for the adversarial actor NN we have

$$\hat{v}_i(e_i)=\hat{W}_{id}^Te_i,\ \forall i\in\mathcal{N},\tag{39}$$

where $\hat{W}_{id}\in\mathbb{R}^{n\times l_i}$ are the estimated actor weights, note also that the neighborhood tracking error, e_i, is serving as an activation function for the action NN. Note that the optimal value for each adversarial actor NN is given by (35).

By using integral reinforcement learning (see also previous section), we can write

$$\begin{aligned}
&Q_i^*(e_i(t),u_i^*(t),u_{\mathcal{N}_i}^*(t),v_i^*(t),v_{\mathcal{N}_i}^*(t))=\\
&Q^*(e_i(t-T),u_i^*(t-T),u_{\mathcal{N}_i}^*(t-T),v_i^*(t-T),v_{\mathcal{N}_i}^*(t-T))\\
&-\frac{1}{2}\int_{t-T}^t\big(e_i^TH_ie_i+(u_i^TR_{ii}u_i-\gamma_{ii}^2v_i^Tv_i)\\
&+\sum_{j\in\mathcal{N}_i}(u_j^TR_{ij}u_j-\gamma_{ij}^2v_j^Tv_j)\big)d\tau,\ \forall i\in\mathcal{N},
\end{aligned}\tag{40}$$

where $T\in\mathbb{R}^+$ is a small fixed time interval that defines how fast one measures the neighborhood tracking error e_i,

$$
\begin{aligned}
&Q_i(e_i,u_i,u_{\mathcal{N}_i},v_i,v_{\mathcal{N}_i}) = \frac{1}{2}z_i^T \\
&\begin{bmatrix}
P_i + H_i + P_iA + A^TP_i & (d_i+g_i)P_iB_i & -\text{row}\big(P_iB_j\big) & (d_i+g_i)P_iD_i & -\text{row}\big(P_iD_j\big)_{j\in\mathcal{N}_i} \\
(d_i+g_i)B_i^TP_i & R_{ii} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-\text{col}\big(B_j^TP_i\big)_{j\in\mathcal{N}_i} & \mathbf{0} & \text{diag}\,\big(R_{ij}\big)_{j\in\mathcal{N}_i} & \mathbf{0} & \mathbf{0} \\
(d_i+g_i)D_i^TP_i & \mathbf{0} & \mathbf{0} & \gamma_{ii}^2 & \mathbf{0} \\
-\text{col}\big(D_j^TP_i\big)_{j\in\mathcal{N}_i} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \text{diag}\,\big(\gamma_{ij}^2\big)_{j\in\mathcal{N}_i}
\end{bmatrix} z_i \\
&:= \frac{1}{2}z_i^T \begin{bmatrix}
Q^i_{\mathrm{e_i e_i}} & Q^i_{\mathrm{e_i u_i}} & Q^i_{\mathrm{e_i u_{\mathcal{N}_i}}} & Q^i_{\mathrm{e_i v_i}} & Q^i_{\mathrm{e_i v_{\mathcal{N}_i}}} \\
Q^i_{\mathrm{u_i e_i}} & Q^i_{\mathrm{u_i u_i}} & Q^i_{\mathrm{u_i u_{\mathcal{N}_i}}} & Q^i_{\mathrm{u_i v_i}} & Q^i_{\mathrm{u_i v_{\mathcal{N}_i}}} \\
Q^i_{\mathrm{u_{\mathcal{N}_i} e_i}} & Q^i_{\mathrm{u_{\mathcal{N}_i} u_i}} & Q^i_{\mathrm{u_{\mathcal{N}_i} u_{\mathcal{N}_i}}} & Q^i_{\mathrm{u_{\mathcal{N}_i} v_i}} & Q^i_{\mathrm{u_{\mathcal{N}_i} v_{\mathcal{N}_i}}} \\
Q^i_{\mathrm{v_i e_i}} & Q^i_{\mathrm{v_i u_i}} & Q^i_{\mathrm{v_i u_{\mathcal{N}_i}}} & Q^i_{\mathrm{v_i v_i}} & Q^i_{\mathrm{v_i v_{\mathcal{N}_i}}} \\
Q^i_{\mathrm{v_{\mathcal{N}_i} e_i}} & Q^i_{\mathrm{v_{\mathcal{N}_i} u_i}} & Q^i_{\mathrm{v_{\mathcal{N}_i} u_{\mathcal{N}_i}}} & Q^i_{\mathrm{v_{\mathcal{N}_i} v_i}} & Q^i_{\mathrm{v_{\mathcal{N}_i} v_{\mathcal{N}_i}}}
\end{bmatrix} z_i := \frac{1}{2}z_i^T\bar{Q}_iz_i,\ \ \forall z_i,\ \ \forall i\in\mathcal{N},
\end{aligned} \tag{33}
$$

$$
\dot{\hat{W}}_{ic} = -\alpha_{ic}\frac{\Big(z_i(t)\otimes z_i(t) - z_i(t-T)\otimes z_i(t-T)\Big)}{\Big(1+\Big(z_i(t)\otimes z_i(t) - z_i(t-T)\otimes z_i(t-T)\Big)^T\Big(z_i(t)\otimes z_i(t) - z_i(t-T)\otimes z_i(t-T)\Big)\Big)^2}E_i^T,\ \ \forall i\in\mathcal{N}, \tag{41}
$$

with

$$
\begin{aligned}
E_i &:= \hat{Q}_i(e_i(t),\hat{u}_i(t),\hat{u}_{\mathcal{N}_i}(t),\hat{v}_i(t),\hat{v}_{\mathcal{N}_i}(t)) - \hat{Q}_i(e_i(t-T),\hat{u}_i(t-T),\hat{u}_{\mathcal{N}_i}(t-T),\hat{v}_i(t-T),\hat{v}_{\mathcal{N}_i}(t-T)) \\
&\quad + \frac{1}{2}\int_{t-T}^{t}\big(e_i^TH_ie_i + (\hat{u}_i^TR_{ii}\hat{u}_i - \gamma_{ii}^2\hat{v}_i^T\hat{v}_i) + \sum_{j\in\mathcal{N}_i}(\hat{u}_j^TR_{ij}\hat{u}_j - \gamma_{ij}^2\hat{v}_j^T\hat{v}_j)\big)d\tau \\
&= \hat{W}_{ic}^T\big(z_i(t)\otimes z_i(t)\big) + \frac{1}{2}\int_{t-T}^{t}\big(e_i^TH_ie_i + (\hat{u}_i^TR_{ii}\hat{u}_i - \gamma_{ii}^2\hat{v}_i^T\hat{v}_i) + \sum_{j\in\mathcal{N}_i}(\hat{u}_j^TR_{ij}\hat{u}_j - \gamma_{ij}^2\hat{v}_j^T\hat{u}_j)\big)d\tau \\
&\quad - \hat{W}_{ic}^T\big(z_i(t-T)\otimes z_i(t-T)\big).
\end{aligned}
$$

the control u_i, the adversarial input v_i and the adversarial inputs and controls in the neighborhood $v_{\mathcal{N}_i}$, $u_{\mathcal{N}_i}$.

Now we shall find tuning updates for $\hat{W}_{ic}$, $\hat{W}_{ia}$, and $\hat{W}_{id}$. By following adaptive control techniques as in [36] we can find the gradient descent estimate of $\hat{W}_{ic}$ for the critic weights of each agent, as in (41) (next page), where $\alpha_{ic}\in\mathbb{R}^+$ is a constant gain that determines the speed of critic neural network convergence. Similarly, the gradient descent estimate of $\hat{W}_{ia}$ for the control actor weights can be constructed as,

$$
\dot{\hat{W}}_{ia} = -\alpha_{ia}e_i\big(\hat{W}_{ia}^Te_i + (\hat{Q}^i_{\mathrm{u_iu_i}})^{-1}\hat{Q}^i_{\mathrm{u_ie_i}}e_i\big)^T,\ \ \forall i\in\mathcal{N}, \tag{42}
$$

where $\alpha_{ia}\in\mathbb{R}^+$ is a constant gain that determines the speed of actor neural network convergence, and finally the gradient descent estimate of $\hat{W}_{id}$ for the adversarial actor weights can be constructed as,

$$
\dot{\hat{W}}_{id} = -\alpha_{id}e_i\big(\hat{W}_{id}^Te_i - (\hat{Q}^i_{\mathrm{v_iv_i}})^{-1}\hat{Q}^i_{\mathrm{v_ie_i}}e_i\big)^T,\ \ \forall i\in\mathcal{N}, \tag{43}
$$

where $\alpha_{id}\in\mathbb{R}^+$ is a constant gain that determines the speed of actor neural network convergence.

As one can see in order to enable full autonomy and resiliency without any offline computations or exhaustive modeling one can simply plug equations (37), (38), (39), (41), (42) and (43) in every agent.

V. Experiments using Approximate Optimal Path Following with Concurrent Learning

Advances in sensing and computational capabilities have enabled autonomous mobile robots to become vital assets across multiple disciplines. This surge of interest over the last few decades has drawn considerable attention to motion control of autonomous vehicular systems. As the technology matures, there is a desire to improve the performance (e.g., minimum control effort, time, distance) of such systems to better achieve their objectives.

Guidance laws for autonomous vehicles are typically divided into three categories: point regulation, trajectory tracking, and path-following. Path-following refers to a class of problems where the control objective is to converge to and remain on a desired geometric path without the requirement of temporal constraints (cf. [51], [59], [60]). Path-following is ideal for applications intolerant of spatial error (e.g., nav-

igating cluttered environments, executing search patterns). Path-following heuristically yields smoother convergence to a desired path and reduces the risk of control saturation. A path-following control structure can also alleviate difficulties in the control of nonholonomic vehicles (cf. [59] and [23]).

Optimal control techniques have been applied to path-following to improve path-following performance (cf. [31], [44], [75], [78]). From a survey of such results, motivation exists to provide emerging autonomous systems with an online optimal feedback control approach for path-following that can incorporates the system's nonlinear dynamics within the design to provide stability and performance guarantees.

In a similar manner as in the previous sections, the optimal path-following problem can be formulated in terms of the HJB equation using Bellman's principle of optimality. Motivated by the desire for optimal path-following, an ADP-based controller can be developed for a unicycle-type mobile robot where the optimal policy is parametrized by a neural network (NN) [89]. Path-following is achieved by tracking a virtual target placed on the desired path. The motion of the virtual target is described by a predefined state-dependent ordinary differential equation (cf. [24], [28], [51]). The state associated with the virtual target's location along the path is unbounded due to the infinite time horizon of the guidance law, which presents several challenges related to the use of a NN.[1] In addition, the vehicle requires a constant control effort to remain on the path; therefore, any policy that results in path-following also results in infinite cost, rendering the associated control problem ill-defined (as in the tracking problem discussion in the previous sections).

In this section and the work in [89], the motion of the virtual target is redefined to facilitate the use of the NN, and a modified control input is developed to render feasible optimal policies. The cost function is formulated in terms of the modified control and redefined virtual target motion, a unique challenge not addressed in previous ADP literature. The controller yields uniformly ultimately bounded (UUB) convergence of the approximate policy to the optimal policy and the vehicle state to the path while maintaining a desired speed profile. Simulation results compare the policy obtained using the developed technique to an offline numerical optimal solution. The proposed method is also experimentally validated on a differential drive mobile robot.

A. Problem Description

Path-following refers to a class of problems where the control objective is to converge to and remain on a desired geometric path. The desired path is not necessarily parametrized by time, but by some convenient parameter (e.g., path length). The path-following method in this section utilizes a virtual target that moves along the desired path. The location of the virtual target is determined by the path parameter $s_p \in \mathbb{R}$ (e.g., arc length). It is convenient to select the arc length as the path parameter for a vehicle, since the desired speed can be defined as unit length per unit time.

$$\dot{\theta} = -\kappa \dot{s}_p + w,$$

For a nonholonomic vehicle moving in a plane, the kinematic error dynamics for the path tracking problem can be expressed as [51], [89]

$$\begin{aligned} \dot{x} &= \dot{s}_p (\kappa y - 1) + v \cos\theta \\ \dot{y} &= -x\kappa \dot{s}_p + v \sin\theta \\ \dot{\theta} &= \omega_v - \kappa \dot{s}_p \end{aligned} \tag{44}$$

where $x, y \in \mathbb{R}$ denote the position of the vehicle in the plane, $\theta \in \mathbb{R}$ denotes the orientation of the vehicle with respect to a fixed coordinate system, $v, w_v \in \mathbb{R}$ denote the linear and angular velocity of the vehicle, respectively, and $\kappa \in \mathbb{R}$ is the path curvature. The location of the virtual target can be determined as described in [51], [89] as

$$\dot{s}_p \triangleq v_{des} \cos\theta + k_1 x, \tag{45}$$

where $v_{des} \in \mathbb{R}$ is a desired positive, bounded and time-invariant speed profile, and $k_1 \in \mathbb{R}$ is an adjustable positive gain.

Assumption 1. *The desired path is regular and C^2 continuous; hence, the path curvature κ is bounded and continuous.* □

To facilitate the subsequent control development, an auxiliary function $\phi : \mathbb{R} \to (-1, 1)$ is defined as

$$\phi \triangleq \tanh(k_2 s_p), \tag{46}$$

where $k_2 \in \mathbb{R}$ is a positive gain. From (45) and (46), the time derivative of ϕ is

$$\dot{\phi} = k_2 \left(1 - \phi^2\right) (v_{des} \cos\theta + k_1 x). \tag{47}$$

Note that the path curvature and desired speed profile can be written as functions of ϕ.

Based on (44) and (45), auxiliary control inputs $v_e, w_e \in \mathbb{R}$ are designed as

$$\begin{aligned} v_e &\triangleq v - v_{ss}, \\ w_e &\triangleq w_v - w_{ss}, \end{aligned} \tag{48}$$

where $w_{ss} \triangleq \kappa v_{des}$ and $v_{ss} \triangleq v_{des}$ are computed based on the control input required to remain on the path.

Substituting (45) and (48) into (44), and augmenting the system state with (47), the closed-loop system is

$$\begin{aligned} \dot{x} &= \kappa y v_{des} \cos\theta + k_1 \kappa x y - k_1 x + v_e \cos\theta \\ \dot{y} &= v_{des} \sin\theta - \kappa x v_{des} \cos\theta - k_1 \kappa x^2 + v_e \sin\theta \\ \dot{\theta} &= \kappa v_{des} - \kappa (v_{des} \cos\theta + k_1 x) + w_e \\ \dot{\phi} &= k_2 \left(1 - \phi^2\right) (v_{des} \cos\theta + k_1 x). \end{aligned} \tag{49}$$

The closed-loop system in (49) can be rewritten in a control affine form as in the previous sections as

$$\dot{X} = F(X) + G(X) u, \tag{50}$$

[1] For an infinite horizon problem, time and hence the virtual target's location along the path do not lie on a compact set, and thus can not be used as an input to a NN.

where $X = \begin{bmatrix} x & y & \theta & \phi \end{bmatrix}^T \in \mathbb{R}^4$ is the state vector, $u = \begin{bmatrix} v_e & w_e \end{bmatrix}^T \in \mathbb{R}^2$ is the control vector, and the locally Lipschitz functions $F : \mathbb{R}^4 \to \mathbb{R}^4$ and $G : \mathbb{R}^4 \to \mathbb{R}^{4\times 2}$ are defined as

$$F(X) \triangleq \begin{bmatrix} \kappa y v_{des} \cos\theta + k_1 \kappa x y - k_1 x \\ v_{des} \sin\theta - \kappa x v_{des} \cos\theta - k_1 \kappa x^2 \\ \kappa v_{des} - \kappa \left(v_{des} \cos\theta + k_1 x\right) \\ k_2 \left(1 - \phi^2\right) \left(v_{des} \cos\theta + k_1 x\right) \end{bmatrix}, \quad (51)$$

$$G(X) \triangleq \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

B. Formulation of the Optimal Control Problem

The cost functional for the optimal path following control problem considered in this section is defined as

$$J(X, u) \triangleq \int_t^\infty r\left(X(\tau), u(\tau)\right) d\tau, \quad (52)$$

where $r : \mathbb{R}^4 \to [0, \infty)$ is the local cost defined as

$$r(X, u) \triangleq X^T Q_T X + u^T R u$$

where Q_T and R are introduced in Section III-A. The infinite-time scalar value function $V : \mathbb{R}^4 \to [0, \infty)$ for this problem can be written as

$$V(X) = \min_{u \in \mathcal{U}} \int_t^\infty r\left(X(\tau), u(\tau)\right) d\tau, \quad (53)$$

where $\mathcal{U}$ is the set of admissible control policies.

The objective of the optimal control problem is to determine the optimal policy u^* that minimizes the cost functional in (52) subject to the constraints in (50). The Hamiltonian is defined as

$$\mathcal{H} \triangleq r(X, u^*) + \frac{\partial V}{\partial X}\left(F + G u^*\right). \quad (54)$$

Assuming a minimizing policy exists and the value function is continuously differentiable, the value function satisfies the HJB equation given as [46]

$$0 = \frac{\partial V}{\partial t} + \mathcal{H}, \quad (55)$$

where $\frac{\partial V}{\partial t} = 0$ since there exists no explicit dependence on time. The optimal policy is derived from (55) as

$$u^* = -\frac{1}{2} R^{-1} G^T \left(\frac{\partial V}{\partial X}\right)^T. \quad (56)$$

As described in the previous sections, the analytical expression for the optimal controller in (56) requires knowledge of the value function which is the solution to the HJB. Given the kinematics in (51), it is unclear how to determine an analytical solution to (55); hence, the subsequent development focuses on the development of an approximate solution. Specifically, over any compact domain $\chi \subset \mathbb{R}^4$, the value function $V : \mathbb{R}^4 \to [0, \infty)$ can be represented by a single-layer NN with L neurons as

$$V(X) = W^T \sigma(X) + \epsilon(X), \quad (57)$$

where $W \in \mathbb{R}^L$ is the ideal weight vector bounded above by a known positive constant, $\sigma : \mathbb{R}^4 \to \mathbb{R}^L$ is a bounded, continuously differentiable activation function, and $\epsilon : \mathbb{R}^4 \to \mathbb{R}$ is the bounded, continuously differentiable function reconstruction error. From (56) and (57), the optimal policy can be represented as

$$u^* = -\frac{1}{2} R^{-1} G^T \left(\sigma'^T W + \epsilon'^T\right), \quad (58)$$

where $\sigma' \in \mathbb{R}^{L\times 4}$ and $\epsilon' \in \mathbb{R}^{1\times 4}$ are partial derivatives with respect to the state. Based on (57) and (58), the value function and optimal policy NN approximations are defined as [42]

$$\hat{V} = \hat{W}_c^T \sigma, \quad (59)$$

$$\hat{u} = -\frac{1}{2} R^{-1} G^T \sigma'^T \hat{W}_a, \quad (60)$$

where $\hat{W}_c, \hat{W}_a \in \mathbb{R}^L$ are estimates of the ideal weight vector W. The weight estimation errors are defined as $\tilde{W}_c \triangleq W - W_c$ and $\tilde{W}_a \triangleq W - W_a$. The NN approximation of the Hamiltonian is given as

$$\hat{\mathcal{H}} = r(X, \hat{u}) + \frac{\partial \hat{V}}{\partial X}\left(F + G\hat{u}\right) \quad (61)$$

by substituting (59) and (60) into (54). The Bellman error $\delta \in \mathbb{R}$ is defined as the error between the optimal and approximate Hamiltonian and is given as

$$\delta \triangleq \hat{\mathcal{H}} - \mathcal{H}, \quad (62)$$

where $\mathcal{H} = 0$. Therefore, the Bellman error can be written in a measurable form as

$$\delta = r(X, \hat{u}) + \hat{W}_c^T \omega, \quad (63)$$

where $\omega \triangleq \sigma'\left(F + G\hat{u}\right) \in \mathbb{R}^L$.

Assumption 2. *There exists a set of sampled data points $\{\zeta_j \in \chi | j = 1, 2, \ldots, N\}$ such that $\forall t \in [0, \infty)$,*

$$\operatorname{rank}\left(\sum_{j=1}^N \frac{\omega_j \omega_j^T}{p_j}\right) = L, \quad (64)$$

where $p_j \triangleq \sqrt{1 + \omega_j^T \omega_j}$ denotes the normalization constant, and ω_j refers to the expression defined after (63) evaluated at the specified data point ζ_j. □

As discussed in [42], [43], the rank condition in (64) cannot be guaranteed to hold a priori. However, heuristically, the condition can be met by sampling redundant data, i.e., $N \gg L$. Based on Assumption 2, it can be shown that $\sum_{j=1}^N \frac{\omega_j \omega_j^T}{p_j} > 0$ such that

$$\underline{c} \left\|\xi_c\right\|^2 \leqslant \xi_c^T \left(\sum_{j=1}^n \frac{\omega_j \omega_j^T}{p_j}\right) \xi_c \leqslant \overline{c} \left\|\xi_c\right\|^2, \ \forall \xi_c \in \mathbb{R}^4$$

even in the absence of persistent excitation (cf. [20] and [21]).

The adaptive update law for $\hat{W}_c$ in (59) is given by

$$\dot{\hat{W}}_c = -\Gamma \left(\eta_{c1} \frac{\partial \delta}{\partial \hat{W}_c} \frac{\delta}{p} + \frac{\eta_{c2}}{N} \sum_{j=1}^{N} \frac{\partial \delta_j}{\partial \hat{W}_c} \frac{\delta_j}{p_j} \right), \tag{65}$$

where $\eta_{c1}, \eta_{c2} \in \mathbb{R}$ are positive adaptation gains, $\Gamma \in \mathbb{R}^{L \times L}$ is a positive and diagonal weighting matrix, $\frac{\partial \delta}{\partial \hat{W}_c}$ is the regressor matrix, and $p \triangleq \sqrt{1 + \omega^T \omega}$ is a normalization constant. The structure of the concurrent learning-based adaptive update law in (65) is motivated by the fact that it yields a $-\tilde{W}_c^T \tilde{W}_c$ in the derivative of the Lyapunov-function candidate without requiring persistence of excitation (cf. [42], [43]). The update law for $\hat{W}_a$ in (60) is given by

$$\dot{\hat{W}}_a = \text{proj} \left\{ -\eta_a \left(\hat{W}_a - \hat{W}_c \right) \right\}, \tag{66}$$

where $\eta_a \in \mathbb{R}$ is a positive gain, and $\text{proj}\{\cdot\}$ is a smooth projection operator [25]. Using the properties of the projection operator, the policy NN weight estimation errors are bounded above by positive constants.

C. Simulation and Experimental Results

To demonstrate the performance of the developed ADP-based guidance law, simulation and experimental results are presented. Simulations allow the developed method to be compared to other optimal solutions, whereas the experimental results demonstrate the real-time optimal performance. For both, the vehicle is commanded to follow a figure eight path with a desired speed of $v_{des} = 0.25\,\text{m/s}$. The virtual target is initially placed at the position corresponding to an initial path parameter of $s_p(0) = 0\,\text{m}$, and the initial error state is selected as $e(0) = \begin{bmatrix} -0.5\,\text{m} & -0.5\,\text{m} & \pi/2\,\text{rad} \end{bmatrix}^T$. Therefore, the initial augmented state is $X(0) = \begin{bmatrix} -0.5\,\text{m} & -0.5\,\text{m} & \pi/2\,\text{rad} & 0\,\text{m} \end{bmatrix}^T$. The sampled data points are selected on a $5 \times 5 \times 3 \times 3$ grid about the origin.

The simulation result uses the kinematic model in (44) as the simulated mobile robot. Since an analytical solution is not feasible for this problem, the simulation results are directly compared to results obtained by an offline optimal solver GPOPS [67]. Figures 5 and 6 illustrate that the state and control trajectories (denoted by lines) approach the solution found using the offline optimal solver (denoted by markers), and Figure 7 shows the NN critic and actor weight estimates converge to steady state values[2]. The true values of the ideal NN network weights are unknown. However, after the NN converges to a steady state value, the system trajectories and control values obtained using the developed method correlate with the system trajectories and control value of the offline optimal solver. The overall performance of the controller is demonstrated in the plot of the vehicle's planar trajectory in Figure 8.

[2]It takes ~125 seconds for the mobile robot to traverse the desired path. However, all figures with the exception of the vehicle trajectory are plotted only for 60 seconds to provide clarity on the transient response. The steady-state response remains the same after the initial transient (~20 seconds).

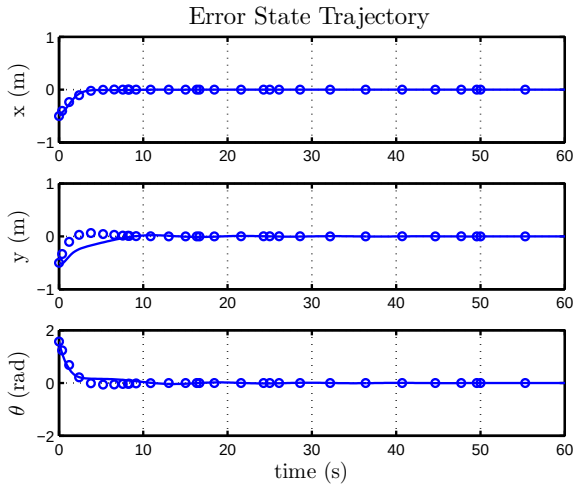

Fig. 5. The error state trajectory generated by the developed method is shown as solid lines, and the collocation points from GPOPS are shown as markers.

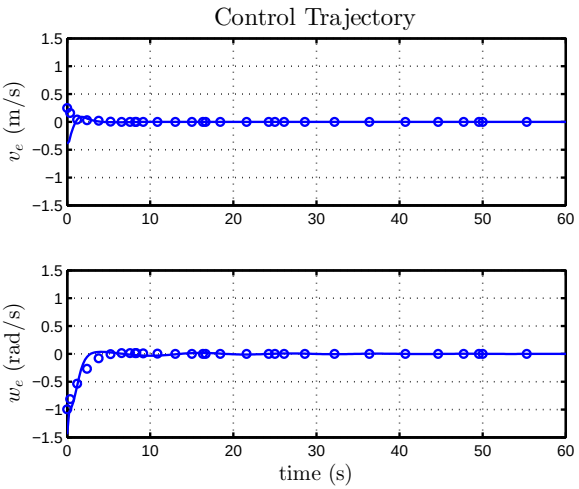

Fig. 6. The control trajectory generated by the developed method is shown as solid lines, and the collocation points from GPOPS are shown as markers.

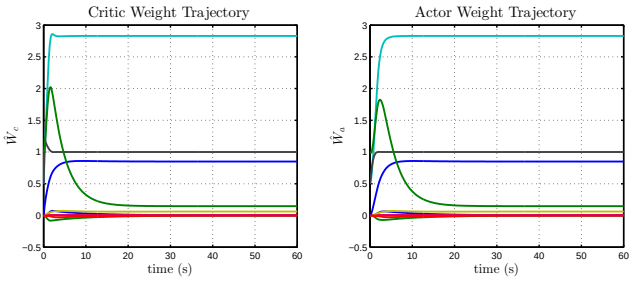

Fig. 7. The estimated NN weight trajectories generated by the developed method in simulation.

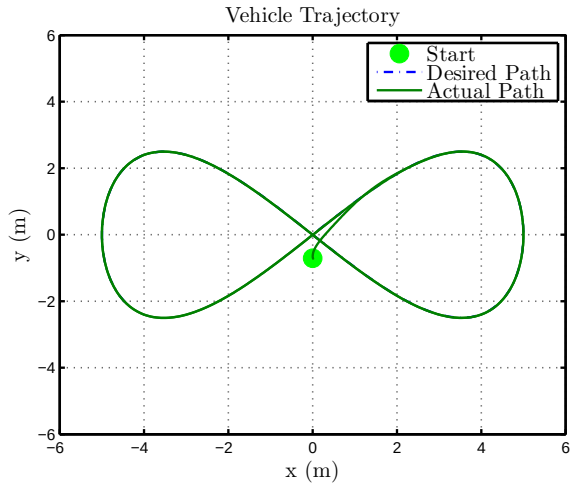

Fig. 8. The planar trajectory achieved by the developed method in simulation.

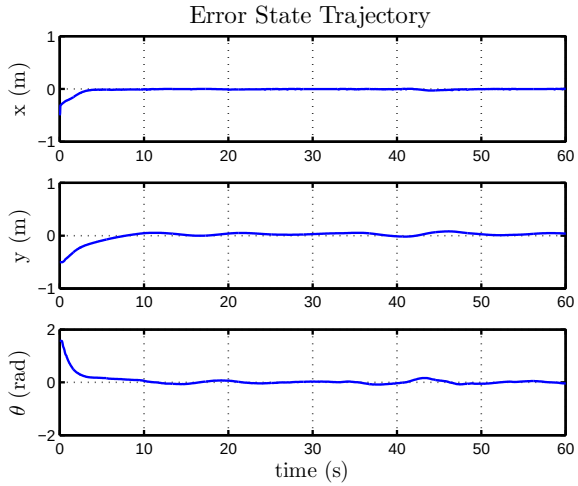

Fig. 9. The error state trajectory generated by the developed method implemented on the Turtlebot.

Experimental results also demonstrate the ability of the developed controller to perform on real-world hardware. The ADP-based guidance law is implemented on a Turtlebot wheeled mobile robot. Computation of the optimal guidance law takes place on the Turtlebot's on-board ASUS Eee PC netbook with 1.8 GHz Intel Atom processor. The Turtlebot is provided velocity commands from the guidance law where the Turtlebot's existing low-level controller minimizes the velocity tracking error. Figure 9 shows convergences of the error state to a ball about the origin. Figure 10 shows the NN critic and actor weight estimates converge to steady state values that are similar to the simulation result. The ability of the mobile robot to track the desired path is demonstrated in Figure 11.

VI. Conclusion and Future Research Directions

This tutorial paper presented different state-of-the-art control approaches and theory for complex systems based on machine intelligence in order to enable full autonomy. Given

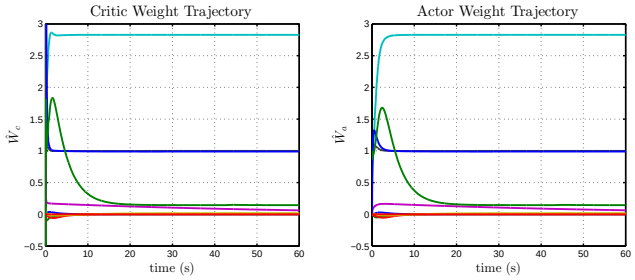

Fig. 10. The estimated NN weight trajectories generated by the developed method implemented on the Turtlebot.

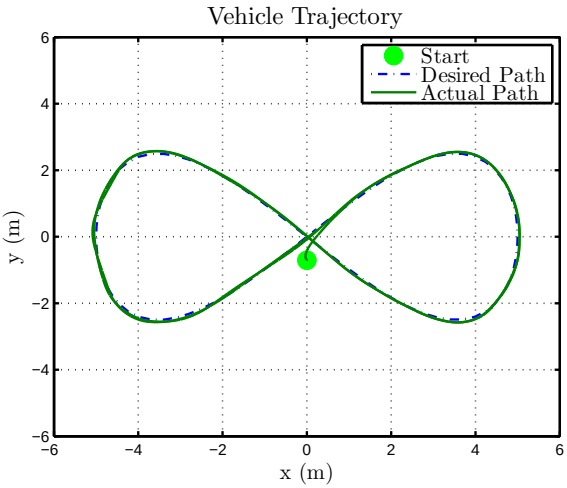

Fig. 11. The planar trajectory achieved by the developed method implemented on the Turtlebot.

the presence of modeling uncertainties, the unavailability of the model, the possibility of cooperative/non-cooperative goals and malicious attacks compromising the security of networked teams, there is a need for approaches that respond to situations not programmed or anticipated in design. The integration of machine intelligence and human cognitive models could advance the human-agent feedback loops while optimizing performance and advancing data decision models.

There are several open questions that need to be addressed in order to provide assurance for machine intelligence and decision-making in complex, uncertain and dynamic environments.

i) How do we go about realizing the properties that are essential to autonomy in a safe, secure manner, to obtain a resilient system that keeps performing well over the lifetime of the control system?

ii) Could CPS provide an approach towards building autonomous systems?

iii) How would autonomous control architectures look like?

These are important open research problems. Approaches that use CPS and energy like concepts such as passivity/dissipativity to preserve properties when subsystems are interconnected offer some promise [8].

REFERENCES

[1] P. Abbeel, Ad. Coates and A. Y. Ng, "Autonomous Helicopter Aerobatics through Apprenticeship Learning," *In the International Journal of Robotics Research (IJRR)*, vol 29, no 13, 2010

[2] T. Alpcan and T. Basar, "An intrusion detection game with limited observations," *Proc. 12th Int. Symp. on Dynamic Games and Applications*, 2006

[3] P. J. Antsaklis, "The Quest for Autonomy Revisited," ISIS Technical Report ISIS-2011-004, September 2011 (http://www3.nd.edu/ isis/techreports/isis-2011-005.pdf)

[4] P. J. Antsaklis, K. M. Passino and S. J. Wang, "An Introduction to Autonomous Control Systems," *IEEE Control Systems Magazine*, vol.11, no.4, pp. 5-13, 1991, Reprinted in *Neuro-Control Systems: Theory and Applications*, M.M. Gupta and D.H. Rao Eds., Chapter 4, Part 1, pp. 81-89, IEEE Press 1994

[5] P. J. Antsaklis, "Intelligent Control," *Encyclopedia of Electrical and Electronics Engineering*, Vol.10, pp. 493-503, John Wiley & Sons, Inc., 1999

[6] P. J. Antsaklis, "Intelligent Learning Control," Guest Editor's Introduction, *IEEE Control Systems Magazine*, vol. 15, no. 3, pp. 5-7, 1995; Special Issue on 'Intelligence and Learning' of the *IEEE Control Systems Magazine*, vol.15, no.3, pp. 5-80, 1995

[7] P. J. Antsaklis, "Defining Intelligent Control," Report of the Task Force on Intelligent Control, P.J Antsaklis, Chair, *IEEE Control Systems Magazine*, pp. 4-5 & 58-66, 1994, Also in "Proceedings of the 1994 International Symposium on Intelligent Control," pp. (i)-(xvii), Columbus, OH, 1994

[8] P. J. Antsaklis, B. Goodwine, V. Gupta, M. J. McCourt, Y. Wang, P. Wu, M. Xia, H. Yu, and F. Zhu, "Control of Cyber-Physical Systems using Passivity and Dissipativity Based Methods," *European Journal of Control*, vol.19, no. 5, pp. 379-388, 2013

[9] L. C. III Baird, "Reinforcement learning in continuous-time: advantage updating," *In Proc. of ICNN.* vol. 4, pp. 2448-2453, 1994

[10] N. Bailey, *The mathematical theory of infectious diseases and its applications*, Charles Griffin and Company Ltd., 1975

[11] A. Barrat, M. Barthelemy, A. Vespignani, *Dynamical processes on complex networks*, Cambridge University Press, 2008

[12] T. Basar, G. J. Olsder, *Dynamic noncooperative game theory* (2nd ed.), Philadelphia, PA: SIAM, 1999

[13] D. Bauso, L. Giarre, and R. Pesenti, "Mechanism design for optimal consensus problems," *In Proc. Conference on Desision and Control*, pp. 3381-3386, 2006

[14] D. P. Bertsekas, J. N. Tsitsiklis, *Neuro-dynamic programming* MA: Athena Scientific, 1996

[15] L. Busoniu, R. Babuska, B. De Schutter, "A Comprehensive Survey of Multi-Agent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics -Part C: Applications and Reviews*, vol. 38, no. 2, pp. 15-172, 2008

[16] L. Busoniu, R. Babuska, B. deSchutter, D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, CRC Press, 2010

[17] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," *In Workshop on Future Directions in Cyberphysical Systems Security*, 2009

[18] A. Cardenas, S. Amin, and S. S. Sastry, "Secure control: Towards survivable cyber-physical systems," *In First International Workshop on Cyber- Physical Systems (WCPS2008)*, pp. 495-500, Beijing, China, 2008

[19] Y. Cao, W. Yu, W. Ren and G. Chen, "An overview of recent progress in the study of distributed multiagent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427-438, 2013

[20] G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *J. Guid. Control Dynam.*, vol. 34, no. 2, pp. 592–607, March 2011

[21] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adapt. Control Signal Process.*, vol. 27, no. 4, pp. 280–301, 2013

[22] R. Cohen and S. Havlin, *Complex Networks: Structure, Robustness and Function*, Cambridge University Press, 2010

[23] D. Dacic, D. Nesic, and P. Kokotovic, "Path-following for nonlinear systems with unstable zero dynamics," *IEEE Trans. Autom. Control*, vol. 52, no. 3, pp. 481–487, 2007

[24] W. E. Dixon, D. M. Dawson, E. Zergeroglu, and A. Behal, *Nonlinear Control of Wheeled Mobile Robots*, ser. Lecture Notes in Control and Information Sciences. Springer-Verlag London Ltd, 2000, vol. 262

[25] W. E. Dixon, A. Behal, D. M. Dawson, and S. Nagarkatti, *Nonlinear Control of Engineering Systems: A Lyapunov-Based Approach*. Birkhauser: Boston, 2003

[26] S. N. Dorogovtsev, J. F. Mendes, "Evolution of networks: From biological nets to the Internet and WWW," *Oxford University Press*, 2013

[27] K. Doya, "Reinforcement learning in continuous-time and space," *Neural Computation*, vol. 12, pp. 219-245, 2000

[28] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *IEEE Trans. Autom. Control*, vol. 46, no. 11, pp. 1777–1782, Nov 2001

[29] E. Estrada, *The Structure of Complex Networks: Theory and Applications*, Oxford University Press, 2011

[30] J. Fax, R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465-1476, 2004

[31] T. Faulwasser and R. Findeisen, "Nonlinear model predictive path-following control," in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer, 2009, vol. 384, pp. 335–343

[32] N. Ganguly, A. Deutsch, and A. Mukherjee, *Dynamics On and Of Complex Networks Applications to Biology, Computer Science, and the Social Sciences*, Springer, 2009

[33] M. Garetto, W. Gong, and D. Towsley, "Modeling malware spreading dynamics," *Proc. IEEE INFOCOM*, vol. 3, pp. 1869-1879, 2003

[34] S. Gorman, "Electricity grid in U.S. penetrated by spies," *The Wall Street Journal*, page A1, April 8th 2009

[35] M. Harmon, E., L. C. Baird, A. H. Klopf, "Reinforcement learning applied to a differential game," *Adaptive behavior*, vol. 4, no. 1, pp. 3-28, 1995

[36] P. A. Ioannou, B. Fidan, *Adaptive Control Tutorial*, Advances in design and control, SIAM (PA), 2006

[37] A. Jadbabaie, J. Lin, A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988-1001, 2003

[38] Y. Jiang, and Z.P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, pp. 2699-2704, 2012

[39] Y. Jiang, and Z.P. Jiang, "Robust adaptive dynamic programming with an application to power systems," *IEEE Trans. Neural Networks Learning Syst.*, vol. 24, no. 7, pp. 1150-1156, 2013

[40] C. Jiang, Y. Chen, K.J.R. Liu, "Distributed Adaptive Networks: A Graphical Evolutionary Game-Theoretic View," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5675-5688, 2013

[41] D. Jin, S. Cho, Y. Sung, K. Cho, K. Um, "Reactive virtual agent learning for NUI-based HRI applications," *Multimedia Tools and Applications*, pp. 1-14, 2014

[42] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Concurrent learning-based approximate optimal regulation," in *Proc. IEEE Conf. Decis. Control*, Florence, IT, Dec. 2013, pp. 6256–6261

[43] R. Kamalapurkar, J. Klotz, and W. E. Dixon, "Model-based reinforcement learning for on-line feedback-nash equilibrium solution of n-player nonzero-sum differential games," in *Proc. Am. Control Conf.*, 2014, pp. 3000–3005

[44] K. Kanjanawanishkul and A. Zell, "Path following for an omnidirectional mobile robot based on model predictive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 3341–3346

[45] M. A. Khamis, W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 134-151, 2014

[46] D. Kirk, *Optimal Control Theory: An Introduction*. Dover, 2004

[47] B. Kiumarsi, F. L. Lewis, "Actor-critic based optimal tracking for partially unknown nonlinear discrete-time systems," in press, *IEEE Trans. Neural Networks Learning Syst*, 2014

[48] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, M.-B. Naghibi-Sistani, "Reinforcement -learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol 50, no. 4, pp. 1167-1175, 2014

[49] H. J. LeBlanc and X. D. Koutsoukos, "Low complexity resilient consensus in networked multi-agent systems with adversaries," *In Pro-*

ceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control (HSCC '12), pp. 5-14, 2012

[50] J. Young Lee, J. Bae Park, Y. Ho Choi, "Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems," *Automatica*, vol. 48, no. 11, pp. 2850-2859, 2012

[51] L. Lapierre, D. Soetanto, and A. Pascoal, "Non-singular path-following control of a unicycle in the presence of parametric modeling uncertainties," *Int. J. Robust Nonlinear Control*, vol. 16, pp. 485–503, 2003

[52] Y. Y. Liu, J. J. Slotine, A. L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167-173, 2011

[53] B. Luo, H. N. Wu, and T. Huang, "Off-policy reinforcement learning for H∞ control design," in press, *IEEE Trans. Cybern.*, 2014

[54] P. Mehta, S. Meyn, "Q-learning and Pontryagin's Minimum Principle," *Proc. IEEE Decision and Control*, pp. 3598-3605, 2009

[55] M. Mesbahi, M. Egerstedt, *Graph theoretic methods in multiagent networks*, Princeton University Press, 2010

[56] S. P. Meyn, *Control Techniques for Complex Networks*, Cambridge University Press, 2007

[57] H. Modares, and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," in press, *IEEE Trans. Autom. Control*, 2014

[58] H. Modares, and F. L. Lewis, "Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning," *Automatica*, vol. 50, pp. 1780-1792, 2014

[59] P. Morin and C. Samson, "Motion control of wheeled mobile robots," in *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008, pp. 799–826

[60] A. Morro, A. Sgorbissa, and R. Zaccaria, "Path following for unicycle robots with an arbitrary path curvature," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 1016–1023, 2011

[61] M. Newman, A.-L. Barabasi, and D.J. Watts, *The Structure and Dynamics of Networks*, Princeton University Press, 2006

[62] R. Olfati-Saber, R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520-1533, 2004

[63] R. Olfati-Saber, J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," *Proc. of 44th IEEE Conference on Decision and Control, and the European Conference*, pp. 6698-6703, 2005

[64] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus Computation in Unreliable Networks: A System Theoretic Approach," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 90-104, 2012

[65] S. Phelps, "Emergence of social networks via direct and indirect reciprocity," *Autonomous agents and multi-agent systems*, vol. 27, no. 3, pp. 355-374, 2014

[66] W.B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley Series in Probability and Statistics, 2007

[67] A. V. Rao, D. A. Benson, C. L. Darby, M. A. Patterson, C. Francolin, and G. T. Huntington, "Algorithm 902: GPOPS, A MATLAB software for solving multiple-phase optimal control problems using the Gauss pseudospectral method," *ACM Trans. Math. Softw.*, vol. 37, no. 2, pp. 1–39, 2010

[68] W. Ren, R. W. Beard, E. M. Atkins, "A survey of consensus problems in multi-agent coordination," *Proc. American Control Conference*, pp. 1859-1864, 2005

[69] W. Ren, "Distributed cooperative attitude synchronization and tracking for multiple rigid bodies," *IEEE Transactions on Control System Technology*, vol. 18, no. 2, pp. 383-392, 2010

[70] S. Roy, M. Xue, and S. Das, "Security and discoverability of spread dynamics in cyber-physical networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1694-1707, 2012

[71] E. Semsar and K. Khorasani, "Optimal control and game theoretic approaches to cooperative control of a team of multi-vehicle unmanned systems," *in Proc. IEEE International Conference on Networking, Sensing and Control*, pp. 628-633, 2007

[72] E. Semsar-Kazerooni, K. Khorasani, "An LMI Approach to Optimal Consensus Seeking in Multi-Agent Systems," *In Proc. Amer. Control Conf.*, pp. 4519-4524, 2009

[73] S. Shahaboddin, A. Patel, N. B. Anuar, M. L. M. Kiah, and A. Abraham, "Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228-241, 2014

[74] Y.Shoham, K. Leyton-Brown, *Multiagent systems: algorithmic, game-metheoretic, and logical foundations*, Cambridge University Press, 2009

[75] J. da Silva and J. Borges de Sousa, "A dynamic programming approach for the control of autonomous vehicles on planar motion," in *Int. Conf. Auton. Intell. Syst.*, June 2010, pp. 1–6

[76] D. F. Sittig, H. Singh, "A new sociotechnical model for studying health information technology in complex adaptive healthcare systems," *Quality and Safety in Health Care*, vol. 19, no. 3, pp. i68-i74, 2010

[77] S.H. Strogatz, "Exploring Complex Networks," *Nature*, vol. 410, pp. 268-276, 2001

[78] P. Sujit, S. Saripalli, and J. Borges Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicles," *IEEE Control Syst. Mag.*, vol. 34, no. 1, pp. 42–59, Feb 2014

[79] R. S. Sutton, A. G. Barto, *Reinforcement Learning- An Introduction*, MIT Press, Cambridge, Massachusetts, 1998

[80] A. Teixeira, H. Sandberg, and K.H. Johansson, "Networked control systems under cyber attacks with applications to power networks," *In Proc. American Control Conf.*, pp. 3690-3696, 2010

[81] J. Tsitsiklis, "Problems in Decentralized Decision Making and Computation," Ph.D. dissertation, Dept. Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 1984

[82] K. G. Vamvoudakis, J. P. Hespanha, B. Sinopoli, Y. Mo, "Detection in Adversarial Environments," *IEEE Transactions on Automatic Control (Special Issue on Control of Cyber-Physical Systems)*, vol. 59, no. 12, pp. 3209-3223, 2014

[83] K. G. Vamvoudakis, J. P. Hespanha, R. A. Kemmerer, G. Vigna, "Formulating Cyber-Security as Convex Optimization Problems," in *Control of Cyber-Physical Systems, Lecture Notes in Control and Information Sciences*, ed. Danielle Tarraf, Volume 449, pp. 85-100, Springer-Verlag, Berlin, 2013

[84] K. G. Vamvoudakis, F. L. Lewis, G. R. Hudas, "Multi-Agent Differential Graphical Games: Online Adaptive Learning Solution for Synchronization with Optimality," *Automatica*, vol. 48, no. 8, pp. 1598-1611, 2012

[85] K. G. Vamvoudakis, F. L. Lewis, "Online Actor-Critic Algorithm to Solve the Continuous-Time Infinite Horizon Optimal Control Problem," *Automatica*, vol. 46, no. 5, pp. 878-888, 2010

[86] D. Vrabie, and F. L. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural Networks*, vol. 22, pp. 237-246, 2009

[87] D. Vrabie, O. Pastravanu, M. Abou-Khalaf, and F.L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, pp. 477-484, 2009

[88] D. Vrabie, K. G. Vamvoudakis, F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*, Control Engineering Series, IET Press, 2012

[89] P. Walters, R. Kamalapurkar, L. Andrews, and W. E. Dixon, "Online approximate optimal path-following for a mobile robot," in *Proc. IEEE Conf. Decis. Control*, 2014

[90] C. J. C. H. Watkins, P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992

[91] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*, MIT press, 1999

[92] P. J. Werbos, *Approximate dynamic programming for real-time control and neural modeling* In D. A. White, D. A. Sofge (Eds.), Handbook of intelligent control. New York: Van Nostrand Reinhold, 1992

[93] T. Yucelen, M. Egerstedt, "Control of Multiagent Systems under Persistent Disturbances," *In Proc. Amer. Control. Conf.*, pp. 5263-5269, 2012

[94] H. Zhang, D. Liu, Y. Luo, D. Wang, *Adaptive Dynamic Programming for Control: Algorithms and Stability*, Communications and Control Engineering, Springer, 2012

[95] M. Zhu, S. Martínez, "Attack-resilient distributed formation control via online adaptation," *In Proc. CDC-ECE Conf.*, pp. 6624-6629, 2011