ONLINE ADAPTIVE OPTIMAL CONTROL FOR CONTINUOUS-TIME SYSTEMS

by

DRAGUNA VRABIE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2009

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation ECS-0501451 and ECCS-0801330, and the Army Research Office W91NF-05-1-0314.

November 1, 2009

ABSTRACT

ONLINE ADAPTIVE OPTIMAL CONTROL FOR CONTINUOUS-TIME SYSTEMS

DRAGUNA VRABIE

The University of Texas at Arlington, 2009

Supervising Professor: FRANK LEWIS

This work makes two major contributions.

- First, in the field of computational intelligence, it develops reinforcement learning controllers (i.e. approximate dynamic programming algorithms) for continuous-time systems, whereas in the past, reinforcement learning has been mainly developed for discrete-time systems.
- Second, in the field of control systems engineering, it develops on-line optimal adaptive controllers, whereas in the past, optimal control has been an off-line design tool, and on-line adaptive controllers have not been optimal.

The online algorithms presented herein are reinforcement learning schemes which provide online synthesis of optimal control for a class of nonlinear systems with unknown drift term. The results are direct adaptive control algorithms which converge

to the optimal control solution without using an explicit, a priori obtained, model of the drift dynamics of the system.

The online algorithms can be implemented while making use of two function approximation structures, in an Actor-Critic interconnection. In this continuous-time formulation the result is a hybrid control structure which involves a continuous-time controller and a supervisory adaptation structure which operates based on data sampled from the plant and from the continuous-time performance dynamics. Such control structure is unlike any standard form of controllers previously seen in the literature.

The research begins with the development of an adaptive controller which solves online the *linear quadratic regulation* (LQR) problem. The online procedure provides the solution of the *algebraic Riccati equation* (ARE) underlying the LQR problem while renouncing the requirement of exact knowledge on the drift term of the controlled system, while only using discrete measurements of the system's states and performance. From the perspective of computational intelligence this algorithm is a new data-based continuous-time *policy iteration* (PI) approach to the solution of the optimization problem.

It became then interesting to develop an online method which provides control solutions for a system with nonlinear dynamics. In this case the theoretical development becomes a bit more complicated since the equation underlying the optimal control problem is the Hamilton-Jacobi-Bellman (HJB) equation, a nonlinear partial differential equation which is in general impossible to be solved analytically and most often does not have smooth solution. The new online data-based approach to adaptive optimal

control is extended to provide a local approximate optimal control solution for the case of nonlinear systems. The convergence guarantee of the online algorithm is given under the realistic assumption that the two function approximators involved in the online policy iteration procedure, namely actor and critic, do not provide perfect representations for the nonlinear control and cost functions. Also in this case the algorithm reaches to the solution without using any information on the form of the drift term in the dynamics of the system.

At each step of the online iterative algorithm, a generalized HJB (GHJB) equation is solved using measured data and a reinforcement learning technique based on temporal differences. Thus it became interesting to see if these GHJB equations can be solved by iterative means. This evolved into a new formulation for the PI algorithm that allowed developing the *generalized policy iteration* (GPI) algorithm for continuous-time systems. The GPI represents a spectrum of algorithms which has at one end the exact *policy iteration* (PI) algorithm and at the other a variant of the *value iteration* (VI) algorithm. At the middle part of the spectrum lies the so called *optimistic policy iteration* (OPI) algorithm for CT systems. From this perspective the new continuous-time GPI provides a unified point of view over the *approximate dynamic programming* (ADP) algorithms that deal with continuous-time systems.

The appropriate formulation of the Value Iteration algorithm in a continuous-time framework is now straightforward. Understanding the relation between the PI and VI algorithms is now of utmost importance. The analysis is done here for linear systems with quadratic cost index. The value iteration algorithm provides computational means

for a sequence of positive definite matrices which converges to the unique positive definite solution of the ARE. While the PI algorithm is a Newton method, the VI algorithm is a quasi-Newton method. The VI algorithm does not require solution of a Lyapunov equation at each step of the iteration thus the stringent requirement of an initial stabilizing control policy is not necessary.

The last result provides an online approach to the solution of zero-sum differential games with linear dynamics and quadratic cost index. It is known that the solution of the zero-sum differential game can be obtained by means of iteration on Riccati equations. Here we exploit our first result to find the saddle point of the game in an online fashion. This work provides the equilibrium solution for the game, in an online fashion, when either the control actor or the disturbance actor is actively learning. At every stage of the game one player learns online an optimal policy to counteract the constant policy of its opponent. The learning procedure takes place based only on discrete-time measurement information of the states of the system and of the value function of the game and without requirement of exact parametric information of the drift term of the system.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

NOTATION AND TERMINOLOGY

Reinforcement Learning – the class of methods which provide solution, in an online fashion, to optimal control problems by means of a reinforcement scalar signal measured from the environment which indicates the level of control performance.

Approximate Dynamic Programming – the class of algorithms that provide online solution to optimal control problems by using approximate representations of the value function to be minimized and of the control algorithm to be performed, and employing Bellman's optimality principle, central in Dynamic Programming, to provide means for training online the two approximation structures based on measured data from the system. Being mathematically formulated, such algorithms allow development of rigorous proofs of convergence for the approximation based approaches.

Actor-Critic structure – the structural representation of approximate dynamic programming algorithms. It reflects the information interconnection between

- the Actor, which reacts in real-time to measurements from the system, and learns to adapt based on performance information from the Critic and
- the Critic which learns to approximate a value function based on performance data and state data measured from the system, and provides performance information relative to the presently used control policy to the Actor.

Adaptive Critics – all algorithms which provide means for learning optimal control policies in an online fashion while using an Actor-Critic structure.

Adaptive Optimal Control – algorithms based on reinforcement learning that provide online synthesis of optimal control policies

Positive definite matrix - Let Σ denote the linear space of all $n \times n$ symmetric matrices. For any two matrices $X, Y \in \Sigma$ one can write $X \geq Y$ if $X - Y$ is positive definite.

Hurwitz - For any matrix $X \in R^{n \times n}$ the *spectrum* of X will be denoted $\sigma(X)$. Let $\mathbb{C}_<$ denote the set of complex numbers with negative real part. A matrix X is said to be *Hurwitz* if $\sigma(X) \subset \mathbb{C}_<$.

For any matrices A, B, C the pair (A, B) is *stabilizable* if $(A - BK)$ is Hurwitz for some matrix K. The pair (C, A) is *detectable* if (A^T, C^T) is stabilizable.

CHAPTER 1

INTRODUCTION

This introductory chapter discusses motivation, background and contribution. The list of publications which resulted from this research is given in Section 1.5.

1.1 Approaches to Optimal Control

In an environment in which a number of players compete for a limited resource, optimal behavior with respect to desired long term goals leads to long term advantages. In a control engineering framework the role of the environment is played by a system to be controlled (this ranges from industrial processes such as distillation columns and power systems, to airplanes, medical equipment and mobile robots); while the controller, equipped with sensors and actuators, plays the role of the agent which is able to regulate the state of the environment such that desired performances are obtained. An intelligent controller is able to adapt its actions to confront unforeseen changes in the system dynamics. Generally, if the controller has a fixed parametric structure, the change in control behavior is reflected by changes of the values of the controller's parameters.

From a control engineering perspective, not every automatic control loop needs to be designed to exhibit intelligent behavior. In fact in industrial process control there exists a hierarchy of control loops which has at the lowest level the simplest and most robust regulation, which provides fast reaction in front of parametric and non-

parametric disturbances without controller adaptation, while at the topmost end are placed the so called money-making loops, whose operation close to optimality has the greatest impact on maximization of income. In the latter case the control performance is not explicitly defined in terms of desired trajectories for the states and/or outputs of the system, instead it is implicitly expressed through a functional that captures the nature of the desired performance in a more general sense. Such an optimality criterion characterizes the system's performance in terms of the control inputs and system states; it is in fact an implicit representation of a desired balance between the amount of effort invested in the control process and the resulting outputs.

Optimal control refers to a class of methods that can be used to synthesize a control policy which results in best possible behavior with respect to the prescribed criterion (i.e. control policy which leads to maximization of performance). The solutions of optimal control problems can be obtained either by using Pontryagin's minimum principle, which provides a necessary condition for optimality, or by solving the Hamilton-Jacobi-Bellman (HJB), which is a sufficient condition (see e.g. [29], [40]). Although mathematically elegant, both approaches present a major disadvantage posed by the requirement of complete knowledge of the system dynamics. In the case when only an approximate model of the system is available, and solution of the problem is attainable via analytical or numerical methods, the optimal controller derived with respect to the system's assumed model will not perform optimally when applied for the control of the real process. Thus, adaptation of the controller parameters such that

operation becomes optimal with respect to the behavior of the real plant is highly desired.

The class of techniques called *adaptive control* (e.g. see [30]) was developed in order to deal with the problem of designing controllers for systems with unknown or uncertain parameter models (e.g. systems for which parameters can drift slowly over time). The adaptive control techniques utilize a desired output signal and, comparing it to the actual system output, use the error difference to adapt the controller parameters in the sense of error minimization. However the controllers that will be generated will not produce trajectories that will minimize cost functions as defined in the optimal control framework, thus adaptive control is not optimal in a formal sense.

Adaptive optimal controllers have been developed either by adding optimality features to an adaptive controller (e.g. the adaptation of the controller parameters is driven by desired performance improvement reflected by an optimality criterion functional) or by adding adaptive features to an optimal controller (e.g. the optimal control policy is improved relative to the adaptation of the parameters of a model of the system).

From a different perspective, *adaptive inverse optimization* methods, extensively developed for nonlinear control (e.g. [21], [41], [34]), solve for control strategies that optimize a performance index without directly solving the underlying equation of the optimal control problem. However, this methodology restricts the choice of the performance index, which can no longer be freely specified by the designer; while at the same time requires knowledge of a stabilizing control law.

For the purpose of obtaining optimal controllers that minimize a given cost function without making use of a model of the system to be controlled, a class of *reinforcement learning (RL)* techniques, namely *adaptive critics*, was developed in the computational intelligence community [57]. These are in effect adaptive control techniques in which the controller parameters are sequentially updated based on a scalar reinforcement signal measuring the controller performance. These algorithms provide an alternative to solving the optimal control problem by approximately solving Bellman's equation for the optimal cost, and then computing the optimal control policy (*i.e.* the feedback gain for linear systems). Compared with adaptive control, the learning process does not take place at the controller tuning level alone but a new adaptive structure was introduced to learn cost functions like the ones specified in optimal control framework.

The reinforcement learning approach to *direct adaptive optimal control* [57], [56], was introduced and extensively developed in the computational intelligence and machine learning societies, generally to find optimal control policies for markovian systems with discrete state and action spaces [27]. The RL algorithms are constructed on the idea that successful control decisions should be remembered, by means of a reinforcement signal, such that they become more likely to be used a second time. Although the idea originates from experimental animal learning, where it has been observed that the dopamine neurotransmitter acts as a reinforcement informational signal which favors learning at the level of the neuronal cell (see e.g. [51], [18]), RL is

strongly connected from a theoretical point of view with direct and indirect adaptive optimal control methods.

The main advantage of using RL to solving the optimal control problems comes from the fact that a number of RL algorithms, e.g. *Q-learning* [61] (also known as *action-dependent heuristic dynamic programming* [63], [64]), do not require knowledge or identification/learning of the system dynamics. This is important since it is well known that modeling and identification procedures for the dynamics of a given nonlinear system is most often a time consuming iterative procedure which requires model design, parameter identification and model validation at each step of the iteration. This procedure is even more difficult when the system has hidden nonlinear dynamics which manifest only in certain operating regions. In the RL algorithms case the learning process is moved at a higher level having no longer as object of interest the system's dynamics but a performance index which quantifies how close to optimality is the closed loop control system operating. In other words, instead of identifying a model of the plant dynamics, to be later used for the controller design, the RL algorithms require identification of the static map which describes the system performance associated with a given control policy. One sees now that, as long as enough information is available to describe the performance associated with a given control policy at all significant operating points of the control system, the system performance map can be easily learned, conditioned by the fact that the control system maintains stability properties. This is again advantageous compared with an open loop identification procedure which, due to the excitatory inputs required for making the

system dynamics visible in the measured system states, could have as result the instability of the system.

Even in the case when complete knowledge on the system dynamics is available, a second difficulty appears from the fact that the HJB equation, underlying the optimal control problem, is generally nonlinear and most often does not possess an analytical solution; thus the optimal control solution is regularly addressed by numerical methods, [28]. Also from this point of view, RL algorithms provide a natural approach to solve the optimal control problem, as they can be implemented my means of function approximation structures, such as neural networks, that can be trained to learn the solution of the HJB equation.

RL algorithms, such as the ones developed for online implementation in this work, are conceptually based on the approach to optimal behavior learning (i.e. the technique used by a learning agent to find the behavior which results in highest amount of long term reward), which makes use of the measured rewards over short time intervals. These algorithms are mathematically built around Bellman's principle of optimality [40] which is the foundation of the mathematical dynamic programming approach to solving optimal control problems. Due to the fact that function approximation structures, such as neural networks [62], [63], are used for the implementation of these iterative learning algorithms, the approach to learning the optimal behavior has been addressed as *approximate dynamic programming* (ADP) [64] or even *neuro-dynamic programming* [10].

RL algorithms are *implemented on Actor-Critic structures* which involve two function approximators, namely the *actor*, which parameterizes the control policy, and the *critic*, a parametric representation for the cost function which describes the performance of the control system. The solution of the optimal control problem will be provided in the form of the Actor neural network for which the associated cost, i.e. the output of the Critic neural network, has an extremal value. The recent work [65] reviews four generations of general-purpose learning designs for *adaptive, approximate dynamic programming*, which provide approximate solution to optimal control problems and include reinforcement learning as a special case. Werbos argues there the relevance of such methods not only for the general goal of replicating human intelligence but also for bringing solution of efficient regulation in electrical power systems.

1.2 Motivation

Most previous research that develops *approximate dynamic programming* (ADP) methods for control engineering considers systems that operate in *discrete-time* (DT). Past successes include:

- Rigorous formulation and development for DT linear systems.
- Clear relations between these methods and known discrete-time control methodologies have been observed.
- Available model-free variants.
- Formulations for DT nonlinear systems are available.
- Implementation results relevant to industry.

This work overlooks the following practical aspects:

- The dynamics of a large class of human engineered systems unfold in continuous-time.
- Discretized models of continuous-time nonlinear systems are generally not sufficiently accurate.
- Sampling limits the control effectiveness.

Therefore, developing ADP methods in a continuous-time framework is of importance both in control engineering practice as well as from a control theory perspective.

At the same time continuous-time ADP results support and strengthen the idea that reinforcement learning is a framework-independent approach to adaptive optimal control, which has the potential of becoming the most deserving value-driven approach to controller design.

1.3 Background

Within the ADP body of work, the technique called *policy iteration*, first formulated in the framework of stochastic decision theory [27], describes the class of algorithms consisting of a two-step iteration: *policy evaluation* and p*olicy improvement*. The method starts by evaluating the cost associated with a given initial policy and then uses this information to obtain a new improved control policy. The two steps are repeated until the policy improvement step no longer changes the actual policy which converges to the optimal one; as the policy evaluation step expresses the degree of optimality of the control policy.

The policy iteration technique has been extensively studied and employed for finding the optimal control solution for Markov decision problems of all sorts. The references [66] and [10] give a comprehensive overview of the research status in this field. Although the algorithm often converges after a small number of iterations, the major drawback when it is applied to discrete state systems resides in the necessity of sweeping the entire state space before computing the cost associated with a given control policy.

Although ADP formulations have been given primarily for the case of Markovian systems with discrete state and action spaces, recently, as these algorithms have been introduced to the control engineering community, ADP has been formulated also for continuous-state systems, in both discrete-time and continuous-time frameworks. In particular discrete-time formulations of ADP algorithms, with convergence proofs, are abundant (see for example [12], [35], [4], [48], [52]).

Bradtke, Ydestie and Barto, [12], developed a policy iteration algorithm that converges to the state-feedback optimal solution of the discrete-time LQR problem using Q-functions. They gave a proof of convergence for Q-learning policy iteration for discrete-time systems, which, by virtue of using the so called Q-functions [61], [63], does not require any knowledge of the system dynamics. The recursive algorithm requires initialization with a stabilizing controller, the controller remaining stabilizing at every step of the iteration.

In the recent works by Landelius [35], and Al-Tamimi, Abu-Khalaf, and Lewis [3] iterative algorithms have been introduced with guaranteed convergence to the discrete-

time H_2 and H-infinity state-feedback control solution for linear systems without the requirement of a stabilizing controller at each iteration step. Using iterative algorithms to solve for the state feedback optimal control policy, while working with linear systems, is particularly affordable since a sweep of the entire state space is no longer necessary. In this case, the cost associated with a control policy can be easily determined using data along a single state trajectory, assuming that regular persistence of excitation conditions are satisfied.

It is beyond the purpose of this work to serve as a survey of ADP methods. Since it deals with ADP algorithms in a continuous-time framework, in the following we shall limit the referencing to the results related with the, slightly less numerous, continuous-time formulations of ADP.

A first reinforcement learning attempt to determine optimal controllers for continuous-time systems with discrete-state space was the *advantage updating algorithm* [5] which adapts discrete-time reinforcement learning techniques to the case when the sampling time goes to zero. Another RL-based solution to the continuous-time optimal control problem has been given in [17].

For continuous-time and continuous-state linear systems, [45] presented two policy iteration algorithms, mathematically equivalent to Newton's method. The convergence guarantee of the PI technique to the continuous-time LQR solution was given in [32]. These algorithms avoid the necessity of knowing the internal system dynamics either by evaluating the infinite horizon cost associated with a control policy along the entire

stable state trajectory, or by using measurements of the state derivatives to form the Lyapunov equations.

For nonlinear systems, the PI algorithm was introduced by Leake and Liu in 1967, [38]. Three decades later, PI is revisited by Beard, Saridis and Wen, in [9], and presented as a feasible adaptive optimal control solution to the CT optimal control problem. This is due to the fact that the Generalized HJB equations, a sort of Lyapunov equations for nonlinear systems, appearing at each iteration step, could be solved using successive Galerkin approximation algorithms. A neural-networks-based approach was later developed for the case of H2 and H-infinity control problems with constrained control in [1] and [2]. These are offline, model-dependent, policy iteration algorithms which solve the Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations associated with the continuous-time nonlinear optimal control problem. Neural-network-based Actor/Critic structures in a CT framework with neural network tuning laws have been given in [23].

This work introduces a new formulation of the PI algorithm for linear and nonlinear systems with continuous-time dynamics. This new formulation allows online adaptation (i.e. learning) of the continuous-time operating controller to the optimal state feedback control policy without requiring knowledge of the system's drift dynamics. Knowledge regarding the input to state dynamics is still required, but from a system identification point of view this knowledge is relatively easier to obtain.

The new formulation of the PI algorithm results in a continuous-time formulation of *generalized policy iteration*. This is a spectrum of algorithms having at one end the

policy iteration and at the other end the proper formulation of the continuous-time *heuristic dynamic programming* (HDP) algorithm.

In all previous research on continuous-time reinforcement learning algorithms which provide an online approach to the solution of optimal control problems it was assumed that the system is not affected by disturbances. There exist however situations in which it is known that the system will be affected by disturbance signals. In these cases the control problem is formulated with the purpose of finding all admissible controllers which minimize the H-infinity norm. Such controllers counteract in an optimal sense the effects of the worst case disturbance which might affect the system. Suboptimal H-infinity controllers can be determined such that the H-infinity norm is less than a given prescribed bound which is larger than the minimum H-infinity norm.

It is known that finding a solution to this problem is equivalent with finding a solution of a Riccati equation with sign indefinite quadratic term, see e.g. [68], [19], [54], [8]. It is also known that the solution of the H-infinity problem is the saddle point solution of a two player zero-sum differential game. The solution of the Algebraic Riccati Equation arising in the H-infinity optimal control problem has been approached in [16], [15], [36]. In all cases the solution is approached in an iterative manner by means of a Newton-type of algorithm. These algorithms determine sequences of matrices which are monotonically convergent to the solution of the H-infinity ARE. In all cases exact knowledge of the system dynamics is required and the solution is obtained by means of offline computation.

We were interested in developing online algorithms, which use reinforcement learning ideas, for finding the infinite horizon H-infinity state feedback optimal control for linear systems. Thus the last result presented in this thesis is a reinforcement learning approach to the saddle point solution of a two player zero-sum differential game associated with the mentioned problem. Similar to the previous algorithms, also in this case exact knowledge on the drift term of the system is not required.

1.4 Contribution

This thesis makes two major contributions.

- First, in the field of *computational intelligence*, it develops reinforcement learning controllers for continuous-time systems, whereas in the past, reinforcement learning has been mainly developed for discrete-time systems.
- Second, in the field of *control systems engineering*, it develops on-line optimal adaptive controllers, whereas in the past, optimal control has been an off-line design tool, and on-line adaptive controllers have not been optimal.

This work presents, in a continuous-time framework, new formulations of online adaptive schemes which determine state-feedback control policies that optimize infinite horizon cost indices, for systems that are affine-in-the-inputs. The online algorithms presented herein are reinforcement learning schemes which reach the optimal control solution while using only partial knowledge regarding the system dynamics. More exactly knowledge of the drift term in the dynamics of the system is never required.

The contributions of this thesis are the following

1. An online adaptive optimal controller which uses reinforcement learning principles to solve the continuous-time LQR problem; the adaptive algorithm is a data-based approach to the solution of the ARE, underlying the optimal control problem, without using knowledge of the drift term part of the system dynamics.
2. An online adaptive optimal controller for general affine in the inputs nonlinear systems; the algorithm provides local solution to the Hamilton-Jacobi-Bellman equation without using knowledge on the drift term part of the system dynamics.
3. A new continuous-time formulation for the *policy iteration* algorithm; which results in a new online adaptive data-based approach to optimal control for nonlinear systems
4. The continuous-time formulation of *generalized policy iteration*; a spectrum of algorithms which provides a bridge between continuous-time policy iteration and continuous-time *value iteration* (*heuristic dynamic programming*).
5. An online adaptive optimal controller for continuous-time systems based on *heuristic dynamic programming*.
6. An online adaptive approach to the saddle point solution of the two player linear differential game with infinite horizon quadratic cost.

The first two results are concerned with developing online versions of the policy iteration algorithm. First, the online policy iteration algorithm is formulated for the case when the optimal state feedback control is desired for linear systems, in state space form, with infinite horizon quadratic indices. Secondly, the online technique is extended to the case in which the controlled system has nonlinear dynamics. These online

techniques, based on PI, sequentially alternate between the steps of policy evaluation and policy improvement, until an update of the control policy will no longer improve the performance of the control system. Closed-loop dynamic stability is guaranteed throughout. The result is a set of direct adaptive control algorithms which converge to the optimal control solution without using an explicit, a priori obtained, model of the system internal dynamics.

The online algorithms can be implemented while making use of two function approximation structures, in an actor/critic interconnection. The actor structure serves as parametric representation for the control policy while the critic structure approximates the performance of the control system. The parameters of the two function approximators are adapted in an online fashion to become expressions of the optimal controller and optimal cost function. In this continuous-time formulation the result is a hybrid control structure which involves a continuous-time controller and a supervisory adaptation structure which operates based on data sampled from the plant and from the continuous-time performance dynamics. Such control structure is unlike any standard form of controllers previously seen in the literature.

The third result included in this thesis is a new formulation for the policy iteration algorithm. In this formulation the policy evaluation step is executed in an iterative manner by means of a contraction map. This continuous-time formulation of the policy iteration algorithm unfolds into an entire spectrum of iterative algorithms named *generalized policy iterations*. At one end of the spectrum lies the regular *policy iteration* (PI) algorithm while at the opposite side one encounters the continuous-time version of

value iteration (VI). A comparative analysis on the two PI and VI algorithms is then performed while considering the infinite horizon linear quadratic regulation problem.

The last result in this thesis illustrates the manner in which sequential approaches to the solution of the H-infinity control problem can be implemented online using the data-based approach to learning. Also in this case knowledge on the drift term, part of the model of the controlled system, is not required. For the purpose of clarity the derivation is restricted to the case of linear systems with quadratic cost indices.

1.5 List of publications which resulted from this work

1.5.1 Book Chapters

D. Vrabie, F.L. Lewis, "Direct Adaptive Optimal Control: Biologically Inspired Feedback Control", In C.-H. Won et al. (eds), *Advances in Statistical Control, Algebraic Systems Theory, and Dynamic Systems Characteristics*, Birkhauser, Boston 2008.

D. Vrabie, F.L. Lewis, "Online Adaptive Optimal Control Based on Reinforcement Learning", In *Optimization and Optimal Control: Theory and Applications* edited by Altannar Chinchuluun, Panos M. Pardalos, Rentsen Enkhbat and Ider Tseveendorj, Volume in Series in Optimization and Its Application (SOIA), Springer, 2009.

1.5.2 Invited articles and chapters

D. Vrabie, F.L. Lewis, "Approximate Dynamic Programming"*, The Control Handbook*, William S. Levine (Editor), (to appear)

F.L. Lewis, D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control"*, IEEE Circuits & Systems Magazine*, 9(3), 32-50, 2009. (Invited feature article)

1.5.3 Journal articles

D. Vrabie, F.L. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially-unknown nonlinear systems", *Neural Networks – special issue:* Goal-Directed Neural Systems, 22(3), 237-246, 2009

D. Vrabie, O. Pastravanu, F. Lewis, M. Abu-Khalaf, "Adaptive Optimal Control for Continuous-Time Linear Systems Based on Policy Iteration", *Automatica*, 45(2), 477-484, 2009.

D. Vrabie, F.L. Lewis, M. Abu-Khalaf, "Biologically Inspired Scheme for Continuous-Time approximate dynamic programming," *The Transactions of the Institute of Measurement and Control* 30: 207-223, 2008.

1.5.4 Conference papers

D. Vrabie, K. Vamvoudakis, F. Lewis, "Adaptive Optimal Controllers Based on Generalized Policy Iteration in a Continuous-Time Framework", *IEEE Mediterranean Conference on Control and Automation* (MED'09), Thessaloniki, Greece, July 2009.

D. Vrabie, F. Lewis, "Generalized Policy Iteration for Continuous-Time Systems", *IJCNN'09*, Atlanta, June 2009.

K. Vamvoudakis, D. Vrabie, F. Lewis, "Online Policy Iteration Based Algorithms to Solve the Continuous-Time Infinite Horizon Optimal Control Problem", *IEEE International Symposium on approximate dynamic programming and Reinforcement Learning* (ADPRL'09), pp. 36-41, Nashville, USA, April 2009.

D. Vrabie, F. Lewis, "Adaptive Optimal Control Algorithm for Continuous-Time Nonlinear Systems Based on Policy Iteration", Conference on Decision and Control (CDC'08), pp. 73-79, Cancun, Mexico, December 2008.

D. Vrabie, F. Lewis, D. Levine, "Neural Network-Based Adaptive Optimal Controller - A Continuous-Time Formulation -", *Proceedings of the International Conference on Intelligent Computing (ICIC'08)*, Shanghai, China, September 2008.

A. Al-Tamimi, D. Vrabie, F. L. Lewis, M. Abu-Khalaf, "Model-free approximate dynamic programming Schemes for Linear Systems", IJCNN'07, Orlando, Florida, August 2007.

F.L. Lewis, M. Abu-Khalaf, A. Al-Tamimi, and D. Vrabie, "A Perspective on the Development of Intelligent Neural Control Systems", Proceedings of the European Control Conference (ECC'07), 4447-4448, Kos, Greece, July 2007.

D. Vrabie, O. Pastravanu, F. Lewis, "Policy Iteration for Continuous-time Systems with Unknown Internal Dynamics", Proceedings of the 15th Mediterranean Conference on Control and Automation (MED'07), Athens, Greece, June 2007.

D. Vrabie, M. Abu-Khalaf, F.L. Lewis and Y. Wang, "Continuous-time ADP for linear systems with partially unknown dynamics", Proceedings of Symposium on approximate dynamic programming and Reinforcement Learning (ADPRL), Hawaii, USA, April 2007.

M. Abu-Khalaf, F.L. Lewis, A. Al-Tamimi, and D. Vrabie, "Model-free adaptive dynamic programming for unknown systems," *Proc. Int. Conf. Computer Sci. and Education*, 105-114, Xiamen, China, July 2006.

1.6 Outline

Chapter 2 presents the formulation of the *policy iteration* algorithm which provides *online* solution for the *linear quadratic regulation* (LQR) problem. From a mathematical perspective the algorithm solves online the algebraic Riccati equation associated with LQR without requiring model information for the internal dynamics of the system. The effectiveness of the algorithm is shown while finding the optimal load-frequency controller for a power system.

In Chapter 3 the online PI algorithm is formulated for the case of nonlinear systems. The convergence of the algorithm is proven under the realistic assumption that the two function approximators do not provide perfect representations for the nonlinear control and cost functions. Simulation results, obtained considering two second order nonlinear systems, are provided.

In Chapter 4 is introduced the *generalized policy iteration* (GPI) algorithm. This is derived starting from a new formulation of the continuous-time PI algorithm which involves an iterative process to solve for the value function at the policy evaluation step. It is shown that GPI represents in fact a spectrum of algorithms which has at one end the exact policy iteration algorithm and at the other the value iteration (VI) algorithm. At the middle part of the spectrum is formulated the *optimistic policy iteration* (OPI)

algorithm for continuous-time systems. From this perspective this chapter provides a unified point of view over the *approximate dynamic programming* (ADP) algorithms which have been developed for continuous-time systems.

Chapter 5 presents the *value iteration* algorithm for the LQR problem. It is also presented a discussion which uncovers a new connection between the policy iteration algorithm and the value iteration algorithm. Thus it is shown that while the PI is a Newton method, the VI algorithm is a quasi-Newton method for solving the same Riccati equation.

Chapter 6 discusses the use of the online algorithm for finding online the solution of a two player zero-sum differential game with linear dynamics and infinite-horizon quadratic cost. As the solution of the game can be obtained by means of iteration on Riccati equations, we will exploit our first result to obtain online solution for the problem. In this context the regular Actor-Critic structure becomes a double actor – single critic structure. The two actors are the control actor or the disturbance actor. It is shown how the two actors can adapt online their behavior policies to reach the saddle point equilibrium of the game. The equilibrium can be obtained while only one of the two players is *actively learning* (*leading the game*). This work provides solution for the game, in an online fashion, while either the control actor or the disturbance actor is leading the game. At every stage of the game the leading player learns online an optimal policy to counteract the constant policy of its opponent. The learning procedure takes place based only on discrete-time measurement information of the states of the system

and of the value function of the game and without requirement of exact parametric information of the drift term of the controlled system.

Chapter 7 presents conclusions and future work ideas.

CHAPTER 2

ADAPTIVE OPTIMAL CONTROL BASED ON POLICY ITERATION FOR CONTINUOUS-TIME LINEAR SYSTEMS

2.1 Introduction

In this chapter is presented a new, partially model free, algorithm based on policy iterations which provides online solution to the optimal control problem for continuous-time, linear, time-invariant systems.

It is well known that solving this problem is equivalent to finding the unique positive definite solution of the underlying algebraic Riccati equation (ARE). For this reason, considerable effort has been made to solve the ARE and the following approaches have been proposed and extended:

- backwards integration of the differential Riccati equation; or Chandrasekhar equations [31],
- eigenvector-based algorithms [43], [47] and the numerically advantageous Schur vector-based modification [37],
- matrix sign-based algorithms [6], [11], [24],
- Newton's method [32], [22], [44], [7].

All of these methods, and their numerically advantageous variants, are offline procedures which have been proved to converge to the desired solution of the ARE; however all of these techniques require exact knowledge of the state space description

of the system to be controlled, as they either operate on the Hamiltonian matrix associated with the ARE (eigenvector and matrix sign based algorithms) or require solving Lyapunov equations (Newton's method). In either case a model of the system is required and a preceding identification procedure is always necessary. Furthermore, even if a model is available the state-feedback controller obtained based on it will only be optimal for the model approximation of the real system dynamics.

In this chapter is proposed a new policy iteration technique that will solve in an online fashion, along a single state trajectory, the LQR problem for continuous-time systems using only partial knowledge about the system dynamics (*i.e.* the internal dynamics of the system need not be known) and without requiring measurements of the state derivative. This is in effect a direct (no system identification procedure is employed) adaptive control scheme for partially unknown linear systems that converges to the optimal control solution. It will be shown that the new adaptive critic based control scheme is in fact a dynamic controller with the state given by the cost or value function.

The continuous-time policy iteration formulation for linear time-invariant systems is given in Section 2.2. Equivalence with iterating on underlying Lyapunov equations is proved. It is shown that the policy iteration is in fact a Newton method for solving the Riccati equation thus convergence to the optimal control is established. In Section 2.3 is developed the online algorithm that implements the policy iteration scheme, without knowing the plant matrix, in order to find the optimal controller. To demonstrate the capabilities of the proposed policy iteration scheme in Section 2.4 are presented

simulation results of applying the algorithm to find the optimal load-frequency controller for a power plant [60].

2.2 Continuous-time adaptive critic solution for the infinite horizon optimal control problem

In this section is developed the policy iteration algorithm, with the purpose of solving online the LQR problem without using knowledge regarding the system internal dynamics.

The LQR problem

Consider the linear time-invariant dynamical system described by

$$\dot{x}(t)=Ax(t)+Bu(t) \tag{2.1}$$

where $x(t)\in\mathbb{R}^n$, $u(t)\in\mathbb{R}^m$ and (A,B) is stabilizable, and the infinite horizon quadratic cost function expressed as

$$V(x(t_0),t_0)=\int_{t_0}^{\infty}(x^T(\tau)Qx(\tau)+u^T(\tau)Ru(\tau))d\tau \tag{2.2}$$

with $Q\geq 0, R>0$ such that $(Q^{1/2},A)$ detectable. The optimal control problem requires finding the control policy

$$u^*(t)=\arg\min_{\substack{u(t)\\ t_0\leq t\leq\infty}} V(t_0,x(t_0),u(t)). \tag{2.3}$$

The solution of this optimal control problem, determined by Bellman's optimality principle, is given by $u(t)=-Kx(t)$ where

$$K=R^{-1}B^TP \tag{2.4}$$

where the matrix P is the unique positive definite solution of the algebraic Riccati equation (ARE)

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{2.5}$$

Under the detectability condition for $(Q^{1/2}, A)$ the unique positive semidefinite solution of the ARE determines a stabilizing closed loop controller given by (2.4).

It is known that the solution of the infinite horizon optimization problem can be obtained using the Dynamic Programming method and amounts to solving backwards in time a finite horizon optimization problem while extending the horizon to infinity. The following Riccati differential equation has to be solved

$$\begin{aligned} -\dot{P} &= A^T P + PA - PBR^{-1}B^T P + Q \\ P(t_f) &= P_{t_f} \end{aligned}. \tag{2.6}$$

Its solution will converge to the solution of the ARE as $t_f \to \infty$. It is important to note that, in order to solve equation (2.5), complete knowledge of the model of the system is needed, *i.e.* both the system matrix A and control input matrix B must be known. Thus a system identification procedure is required prior to solving the optimal control problem, a procedure which most often ends with finding an approximate model of the system. For this reason, developing algorithms that will converge to the solution of the optimization problem without performing prior system identification and using explicit models of the system dynamics is of particular interest from the control systems point of view.

In the following is presented a new policy iteration algorithm that will solve online for the optimal control gain, the solution of the LQR problem, without using knowledge regarding the system internal dynamics (*i.e.* the system matrix A). The result will in fact be an adaptive controller which converges to the state feedback optimal controller. The algorithm is based on an actor/critic structure and consists in a two-step iteration namely the critic update and the actor update. The update of the critic structure results in calculating the infinite horizon cost associated with the use of a given stabilizing controller. The actor parameters (*i.e.* the controller feedback gain) are then updated in the sense of reducing the cost compared to the present control policy. The derivation of the algorithm is given in section 2.2.1. An analysis is done and proof of convergence is provided in section 2.2.2.

2.2.1 Policy iteration algorithm

Let K be a stabilizing state-feedback gain for (2.1), under the assumption that (A,B) is stabilizable, such that $\dot{x}=(A-BK)x$ is a stable closed loop system. Then the corresponding infinite horizon quadratic cost is given by

$$V(x(t))=\int_t^\infty x^T(\tau)(Q+K^TRK)x(\tau)d\tau=x^T(t)Px(t) \tag{2.7}$$

where P is the real symmetric positive definite solution of the Lyapunov matrix equation

$$(A-BK)^TP+P(A-BK)=-(K^TRK+Q) \tag{2.8}$$

and $V(x(t))$ serves as a Lyapunov function for (2.1) with controller gain K. The cost function (2.7) can be written as

$$V(x(t)) = \int_{t}^{t+T} x^T(\tau)(Q + K^T R K)x(\tau)d\tau + V(x(t+T)) \ . \tag{2.9}$$

Based on (2.9), denoting $x(t)$ with x_t, with the parameterization $V(x_t) = x_t^T P x_t$, and considering an initial stabilizing control gain K_1, the following policy iteration scheme can be implemented online:

$$x_t^T P_i x_t = \int_{t}^{t+T} x_\tau^T (Q + K_i^T R K_i) x_\tau d\tau + x_{t+T}^T P_i x_{t+T} \tag{2.10}$$

$$K_{i+1} = R^{-1} B^T P_i \ . \tag{2.11}$$

Equations (2.10) and (2.11) formulate a new policy iteration algorithm motivated by the work of Murray et al. [45]. Note that implementing this algorithm does not involve the plant matrix A.

2.2.2 Proof of convergence

The next results will establish the convergence of the proposed algorithm.

Lemma 2.1 *Assuming that the system $\dot{x} = A_i x$, with $A_i = A - BK_i$, is stable, solving for P_i in equation* (2.10) *is equivalent to finding the solution of the underlying Lyapunov equation*

$$A_i^T P_i + P_i A_i = -(K_i^T R K_i + Q) \ . \tag{2.12}$$

Proof. Since A_i is a stable matrix and $K_i^T R K_i + Q > 0$ then there exists a unique solution of the Lyapunov equation (2.12), $P_i > 0$. Also, since $V_i(x_t) = x_t^T P_i x_t$, $\forall x_t$, is a Lyapunov function for the system $\dot{x} = A_i x$ and

$$\frac{d(x_t^T P_i x_t)}{dt} = x_t^T (A_i^T P_i + P_i A_i) x_t = -x_t^T (K_i^T R K_i + Q) x_t \tag{2.13}$$

then, $\forall T > 0$, the unique solution of the Lyapunov equation satisfies

$$\begin{aligned} \int_t^{t+T} x_\tau^T (Q + K_i^T R K_i) x_\tau d\tau &= -\int_t^{t+T} \frac{d(x_\tau^T P_i x_\tau)}{d\tau} d\tau \\ &= x_t^T P_i x_t - x_{t+T}^T P_i x_{t+T} \end{aligned} \tag{2.14}$$

i.e. equation (2.10). That is, provided that the system $\dot{x} = A_i x$ is asymptotically stable, the solution of (2.10) is the unique solution of (2.12). □

Remark 2.1 Although the same solution is obtained whether solving (2.12) or (2.10), equation (2.10) can be solved without using any knowledge on the system matrix *A*.

From Lemma 2.1 it follows that the iterative algorithm on (2.10) and (2.11) is equivalent to iterating between (2.12) and (2.11), without using knowledge of the system internal dynamics, if $\dot{x} = A_i x$ is stable at each iteration.

Lemma 2.2 *Assuming that the control policy* K_i *is stabilizing, and* $V_i(x_t) = x_t^T P_i x_t$ *is the cost associated with it, if* (2.11) *is used for updating the control policy then the new control policy will be stabilizing*.

Proof. Take the positive definite cost function $V_i(x_t)$ as a Lyapunov function candidate for the state trajectories generated while using the controller K_{i+1}. Taking the derivative of $V_i(x_t)$ along the trajectories generated by K_{i+1} one obtains

$$\begin{aligned} &\dot{V}_i(x_t) = x_t^T [P_i(A - BK_{i+1}) + (A - BK_{i+1})^T P_i] x_t = \\ &= x_t^T [P_i(A - BK_i) + (A - BK_i)^T P_i] x_t + x_t^T [P_i B(K_i - K_{i+1}) + (K_i - K_{i+1})^T B^T P_i] x_t \end{aligned} \tag{2.15}$$

The second term, using the update given by (2.11) and completing the squares, can be written as

$$x_t^T[K_{i+1}^T R(K_i - K_{i+1}) + (K_i - K_{i+1})^T RK_{i+1}]x_t =$$
$$x_t^T[-(K_i - K_{i+1})^T R(K_i - K_{i+1}) - K_{i+1}^T RK_{i+1} + K_i^T RK_i]x_t$$

Using (2.12) the first term in (2.15) can be written as $-x_t^T[K_i^T RK_i + Q]x_t$ and summing up the two terms one obtains

$$\begin{aligned}\dot{V}_i(x_t) = &-x_t^T[(K_i - K_{i+1})^T R(K_i - K_{i+1})]x_t - \\ &-x_t^T[Q + K_{i+1}^T RK_{i+1}]x_t\end{aligned} \tag{2.16}$$

Thus, under the initial assumptions from the problem setup $Q \geq 0, R > 0$, $V_i(x_t)$ is a Lyapunov function proving that the updated control policy $u = -K_{i+1}x$, with K_{i+1} given by equation (2.11), is stabilizing. □

Remark 2.2 Based on Lemma 2.2 one can conclude that if the initial control policy given by K_1 is stabilizing, then all policies obtained using the iteration (2.10)-(2.11) will be stabilizing policies.

Denote with $Ric(P_i)$ the matrix valued function defined as

$$Ric(P_i) = A^T P_i + P_i A + Q - P_i BR^{-1}B^T P_i \tag{2.17}$$

and let Ric'_{P_i} denote the Fréchet derivative of $Ric(P_i)$ taken with respect to P_i. The matrix function Ric'_{P_i} evaluated at a given matrix M will thus be $Ric'_{P_i}(M) = (A - BR^{-1}B^T P_i)^T M + M(A - BR^{-1}B^T P_i)$.

Lemma 2.3 *The iteration between* (2.10) *and* (2.11) *is equivalent to Newton's method*

$$P_i = P_{i-1} - (Ric'_{P_{i-1}})^{-1} Ric(P_{i-1}) \tag{2.18}$$

Proof. Equations (2.12) and (2.11) can be compactly written as

$$A_i^T P_i + P_i A_i = -(P_{i-1} B R^{-1} B^T P_{i-1} + Q) . \tag{2.19}$$

Subtracting $A_i^T P_{i-1} + P_{i-1} A_i$ on both sides gives

$$\begin{aligned} A_i^T (P_i - P_{i-1}) + (P_i - P_{i-1}) A_i = \\ -(P_{i-1} A + A^T P_{i-1} - P_{i-1} B R^{-1} B^T P_{i-1} + Q) \end{aligned} \tag{2.20}$$

which, making use of the introduced notations $Ric(P_i)$ and Ric'_{P_i}, is the Newton method formulation (2.18). □

Theorem 2.4 *(Convergence) Under the assumptions of stabilizability of* (A, B) *and detectability of* $(Q^{1/2}, A)$*, with* $Q \geq 0, R > 0$ *in the cost index* (2.3)*, the policy iteration* (2.10) *and* (2.11)*, conditioned by an initial stabilizing controller, converges to the optimal control solution given by* (2.4) *where the matrix P satisfies the ARE* (2.5)*.*

Proof. In [32] it has been shown that Newton's method, *i.e.* the iteration (2.12) and (2.11), conditioned by an initial stabilizing policy will converge to the solution of the ARE. Also, if the initial policy is stabilizing, all the subsequent control policies will be stabilizing (as by Lemma 2.2). Based on the proven equivalence between (2.12) and (2.11), and (2.10) and (2.11), we can conclude that the proposed new online policy iteration algorithm will converge to the solution of the optimal control problem (2.2) with the infinite horizon quadratic cost (2.3) – without using knowledge of the internal dynamics of the controlled system (2.1). □

Note that the only requirement for convergence to the optimal controller consists in an initial stabilizing policy that will guarantee a finite value for the cost $V_1(x_t)=x_t^T P_1 x_t$. Under the assumption that the system to be controlled is stabilizable and implementation of an optimal state feedback controller is possible and desired, it is reasonable to assume that a stabilizing (though not optimal) state feedback controller is available to begin the iteration [32], [44]. In fact in many cases the system to be controlled is itself stable such that the initial controller can be chosen as zero.

2.3 Online implementation of the adaptive optimal control algorithm without using knowledge of the system internal dynamics

For the implementation of the iteration scheme given by (2.10) and (2.11) one only needs to have knowledge of the B matrix as it explicitly appears in the policy update. The information regarding the system A matrix is embedded in the states $x(t)$ and $x(t+T)$ which are observed online, and thus the system matrix is never required for the computation of either of the two steps of the policy iteration scheme. The details regarding the online implementation of the algorithm are discussed next. Simulation results obtained while finding the optimal controller for a power system are then presented.

2.3.1 Online implementation of the adaptive algorithm based on policy iteration

To find the parameters (i.e. matrix P_i) of the cost function associated with the policy K_i in (2.10), the term $x^T(t)P_i x(t)$ is written as

$$x^T(t)P_i x(t)=\overline{p}_i^T \overline{x}(t) \tag{2.21}$$

where $\bar{x}(t)$ denotes the Kronecker product quadratic polynomial basis vector with the elements $\{x_i(t)x_j(t)\}_{i=1,n;j=i,n}$ and $\bar{p}=v(P)$ with $v(.)$ a vector valued matrix function that acts on symmetric matrices and 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 $2P_{ij}$, see [13]. Using (2.21), equation (2.10) is rewritten as

$$\bar{p}_i^T(\bar{x}(t)-\bar{x}(t+T))=\int_t^{t+T} x^T(\tau)(Q+K_i^T RK_i)x(\tau)d\tau\ . \qquad (2.22)$$

In this equation $\bar{p}_i$ is the vector of unknown parameters and $\bar{x}(t)-\bar{x}(t+T)$ acts as a regression vector. The right hand side target function, denoted $d(\bar{x}(t),K_i)$ (also known as the reinforcement on the time interval [t,t+T]),

$$d(\bar{x}(t),K_i)\equiv\int_t^{t+T} x^T(\tau)(Q+K_i^T RK_i)x(\tau)d\tau$$

is measured based on the system states over the time interval $[t,t+T]$. Considering $\dot{V}(t)=x^T(t)Qx(t)+u^T(t)Ru(t)$ as a definition for a new state $V(t)$, augmenting the system (2.1), the value of $d(\bar{x}(t),K_i)$ can be measured by taking two measurements of this newly introduced system state since $d(\bar{x}(t),K_i)=V(t+T)-V(t)$. This new state signal is simply the output of an analog integration block having as inputs the quadratic terms $x^T(t)Qx(t)$ and $u^T(t)Ru(t)$ which can also be obtained using an analog processing unit.

At each iteration step, after a sufficient number of state-trajectory points are collected using the same control policy K_i, a least-squares method can be employed to solve for the parameters $\bar{p}_i$ of the function $V_i(x_t)$ (*i.e.* the critic), which will then yield the matrix P_i. The parameter vector $\bar{p}_i$ is found by minimizing, in the least-squares sense, the error between the target function, $d(\bar{x}(t), K_i)$, and the parameterized left hand side of (2.22). Evaluating the right hand side of (2.22) at $N \geq n(n+1)/2$ (the number of independent elements in the matrix P_i) points $\bar{x}^i$ in the state space, over the same time interval *T*, the least-squares solution is obtained as

$$\bar{p}_i = (XX^T)^{-1} XY \tag{2.23}$$

where

$$\begin{aligned} X &= [\bar{x}_\Delta^1 \quad \bar{x}_\Delta^2 \quad \ldots \quad \bar{x}_\Delta^N] \\ \bar{x}_\Delta^i &= \bar{x}^i(t) - \bar{x}^i(t+T) \\ Y &= [d(\bar{x}^1, K_i) \quad d(\bar{x}^2, K_i) \quad \ldots \quad d(\bar{x}^N, K_i)]^T \end{aligned} \quad .$$

The least-squares problem can be solved in real-time after a sufficient number of data points are collected along a single state trajectory, under the regular presence of an excitation requirement. A flow chart of the algorithm is presented in Figure 1.

Alternatively, the solution given by (2.23) can be obtained also using recursive estimation algorithms (*e.g.* gradient descent algorithms or the recursive least squares algorithm) in which case a persistence of excitation condition is required. For this reason there are no real issues related to the algorithm becoming computationally expensive with the increase of the state space dimension.

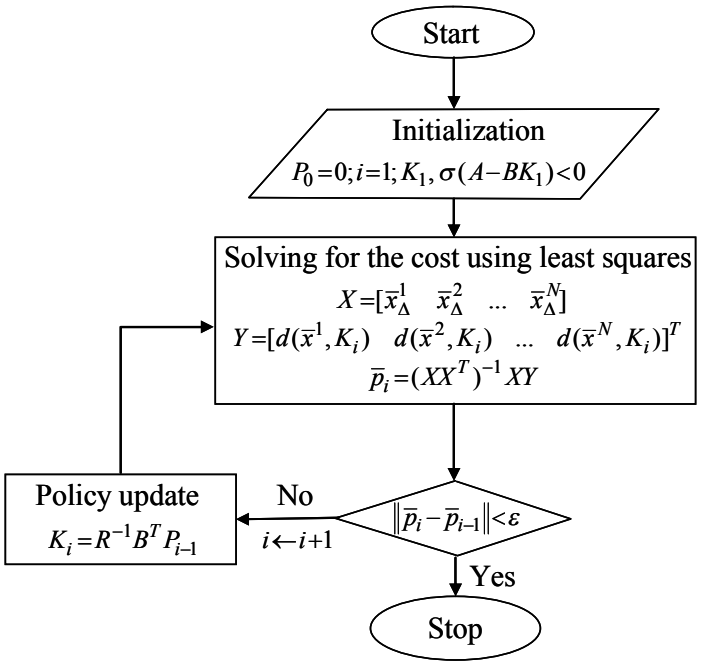

Figure 1. Flow-chart for the online policy iteration algorithm for continuous-time linear systems

Relative to the convergence speed of the algorithm, it has been proven in [32] that Newton's method has quadratic convergence; by the proven equivalence (Theorem 2.4) the online algorithm proposed in this paper has the same property in the case in which the cost function associated with a given control policy (*i.e.* equation (2.10)) is solved for in a single step (*e.g.* using a method such as using the exact least-squares described by equation (2.23)). For the case in which the solution of the equation (2.10) is obtained iteratively, the convergence speed of the online algorithm proposed in this paper will decrease. In this case at each step in the policy iteration algorithm (which involves solving equations (2.10) and (2.11)) a recursive gradient descent algorithm, which most often has exponential convergence, will be used for solving equation (2.11). From this perspective one can resolve that the convergence speed of the online algorithm will

depend on the chosen technique for solving equation (2.10); analysis along these lines are presented in details in the adaptive control literature (see e.g. [30]).

In relation with the choice of the value of the sample time T used for acquiring the data necessary in the iterations, it must be specified that this parameter does not affect in any way the convergence property of the online algorithm. It is however related to the excitation condition necessary in the setup of a numerically well posed least squares problem and obtaining the least squares solution (2.23).

At the same time, it must be observed that the data acquired in order to setup the least squares could be obtained by using different values of the sample time T for each element in the vectors X and Y, as long as the information relating the target elements in the Y vector is consistent with the state samples used for obtaining the corresponding elements in the X vector.

The proposed policy iteration procedure requires only measurements of the states at discrete moments in time, t and $t+T$, as well as knowledge of the observed cost over the time interval $[t, t+T]$, which is $d(\bar{x}(t), K_i)$. Therefore there is no required knowledge about the system A matrix for the evaluation of the cost or the update of the control policy. However the B matrix is required for the update of the control policy, using (2.11), and this makes the tuning algorithm only partially model-free.

For the algorithms presented in [45], computing the cost of a given policy required either

– several control experiments to be performed, considering different initial conditions, until the system states converged to zero (thus letting $T \to \infty$ in (2.10)) in order to have enough data to set up and solve a least-squares problem, or

– directly solving a Lyapunov equation of the sort (2.12) and avoiding the use of A matrix by measuring the system states and also their derivatives.

On the other hand, the policy iteration algorithm proposed in this paper avoids the use of A matrix knowledge and at the same time does not require measuring the state derivatives. Moreover, since the control policy evaluation requires measurements of the cost function over finite time intervals, the algorithm can converge (*i.e.* optimal control is obtained) while performing measurements along a single state trajectory, provided that there is enough initial excitation in the system. In this case, the control policy is updated at time $t+T$, after observing the state $x(t+T)$ and it is used for controlling the system during the time interval $[t+T, t+2T]$; thus the algorithm is suitable for online implementation from the control theory point of view.

The structure of the system with the adaptive controller is presented in Figure 2. Most important is that the system was augmented with an extra state $V(t)$, defined as $\dot{V} = x^T Q x + u^T R u$, in order to extract the information regarding the cost associated with the given policy. This newly introduced system dynamics is part of the adaptive critic based controller thus the control scheme is actually a dynamic controller with the state

given by the cost function V. One can observe that the adaptive optimal controller has a hybrid structure with a continuous time internal state followed by a sampler and discrete time update rule.

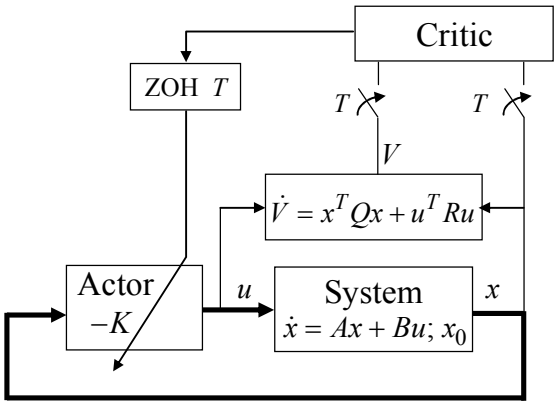

Figure 2. Structure of the system with optimal adaptive controller

It is shown that having little information about the system states, x , and the augmented system state, V (controller dynamics), extracted from the system only at specific time values (*i.e.* the algorithm uses only the data samples $x(t)$, $x(t+T)$ and $V(t+T)-V(t)$ over several time samples), the critic is able to evaluate the performance of the system associated with a given control policy. The control policy is improved after the solution given by (2.23) is obtained. In this way, over a single state trajectory in which several policy evaluations and updates have taken place the algorithm can converge to the optimal control policy. It is however necessary that sufficient excitation exists in the initial state of the system, as the algorithm iterates only on stabilizing policies which will make the states go to zero. In the case that excitation is lost prior to obtaining the convergence (the system reaches the equilibrium point) a new experiment

needs to be conducted having as a starting point the last policy from the previous experiment.

The critic will stop updating the control policy when the error between the performance of the system evaluated at two consecutive steps crosses below a designer specified threshold, *i.e.* the algorithm has converged to the optimal controller. Also, in the case that this error is bigger than the above mentioned threshold, a situation which can be caused for example by a change in the system parameters, the critic will take again the decision to start tuning the actor parameters to obtain an optimal control policy. In fact, if the dynamics described by the A matrix change suddenly, as long as the current controller is stabilizing for the new A matrix, the algorithm will converge to the solution to the corresponding new ARE.

It is observed that the updates of both the actor and the critic are performed at discrete moments in time. However, the control action is a full-fledged continuous-time control, only that its constant gain is updated only at certain points in time. Moreover, the critic update is based on the observations of the continuous-time cost over a finite sample interval. As a result, the algorithm converges to the solution of the continuous-time optimal control problem, as was proven in Section 2.2.

The next two figures provide a visual overview of the online policy iteration algorithm. Figure 3 shows that over the time intervals $[\mathrm{T}_i, \mathrm{T}_{i+1}]$ the system is controlled using a state-feedback control policy which has a constant gain K_i. During this time interval a reinforcement learning procedure, that uses data measured from the system, is employed to determine the value associated with this controller. The value is described

by the parametric structure P_i. Once the learning procedure results in convergence to the value P_i, this result is used for calculating a new gain for the state-feedback controller, namely K_i. The length of the interval $[\mathrm{T}_i, \mathrm{T}_{i+1}]$ is given by the end of the learning procedure, in the sense that T_{i+1} is the time moment when convergence of the learning procedure has been obtained and the value P_i has been determined. In view of this fact, we must emphasize that the time intervals $[\mathrm{T}_i, \mathrm{T}_{i+1}]$ need not be equal with each other and their length is not a design parameter.

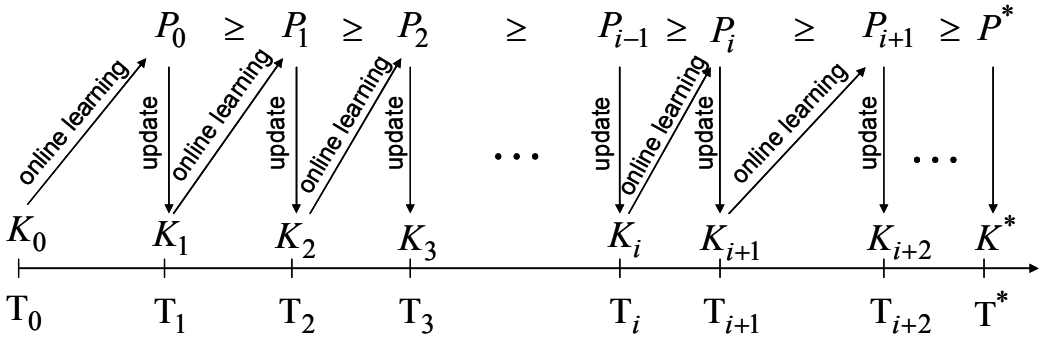

Figure 3. Representation of the online policy iteration algorithm

At every step in the iterative procedure it is guaranteed that the new controller K_{i+1} will result in a better performance, i.e. smaller associated cost, than the previous controller. This will result in a monotonically decreasing sequence of cost functions, $\{P_i\}$, that converges to the smaller possible value, i.e. optimal cost P^*, associated with the optimal control policy K^*.

Figure 4 presents sets of data that are required for online learning of the value described by P_i. We denoted with T the smallest sampling time that can be used to make measurements of the state of the system. A data point that will be used for the online learning procedure is given, in a general notation, by the quadruple $\left(x_k, x_{k+j}, V_k, V_{k+j}\right)$. Denoting with $d(x_k, x_{k+j}, K_i)=V_{k+j}-V_k$ the reinforcement over the time interval, where $j\in\mathbb{N}^*$, then $(x_k, x_{k+j}, d(x_k, x_{k+j}, K_i))$ is a data point of the sort required for setting up the solution given by (2.23). It is emphasized that the data that will be used by the learning procedure need not be collected at fixed sample time intervals.

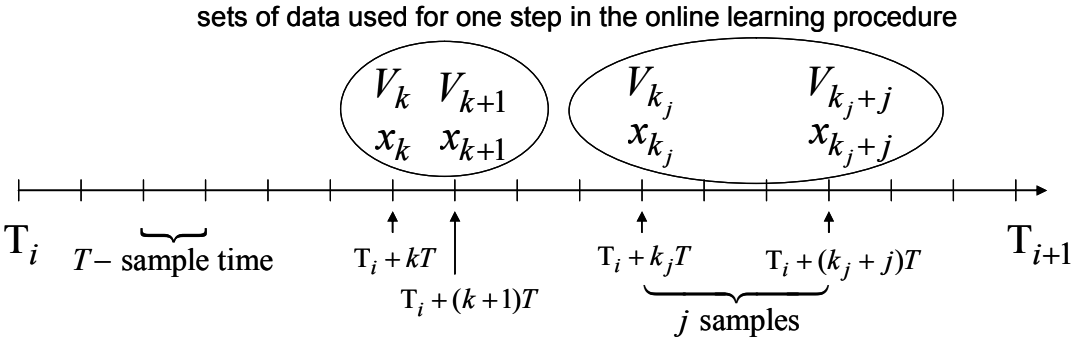

Figure 4. Data measurements used for learning of the value described by P_i over the time interval $[T_i, T_{i+1}]$, while the state feedback gain is K_i

2.4 Online load-frequency controller design for a power system

In this section are presented the results that were obtained in simulation while finding the optimal controller for a power system. The plant that was considered is the linearized model of the power system presented in [60].

Even though power systems are characterized by nonlinearities, linear state-feedback control is regularly employed for load-frequency control at a certain nominal operating points which are characterized by small variations of the system load around a constant value. Although this assumption seems to have simplified the design problem of a load-frequency controller, a new problem appears from the fact that the parameters of the actual plant are not precisely known and only the range of these parameters can be determined. For this reason it is particularly advantageous to apply model free methods to obtain the optimal LQR controller for a given operating point of the power system.

The state vector of the system is

$$x=[\Delta f \ \Delta P_g \ \Delta X_g \ \Delta E]^T \tag{2.24}$$

where the state components are the incremental frequency deviation Δf (Hz), incremental change in generator output ΔP_g (p.u. MW), incremental change in governor value position ΔX_g (p.u. MW) and the incremental change in integral control ΔE. The matrices of the *linearized nominal model* of the plant, used in [60], are

$$A_{nom}=\begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix}. \tag{2.25}$$

$$B=\begin{bmatrix} 0 & 0 & 13.736 & 0 \end{bmatrix}^T$$

Having the model of the system matrices one can easily calculate the LQR controller which is

$$K = \begin{bmatrix} 0.8267 & 1.7003 & 0.7049 & 0.4142 \end{bmatrix}. \tag{2.26}$$

The iterative algorithm can be started while using this controller, that was calculated for the nominal model of the plant. The parameters of the controller will then be adapted in an online procedure, using reinforcement learning, to converge to the parameters of the optimal controller for the real plant.

For this simulation it was considered that the linear drift dynamics of the real plant is given by

$$A = \begin{bmatrix} -0.0665 & 11.5 & 0 & 0 \\ 0 & -2.5 & 2.5 & 0 \\ -9.5 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix}. \tag{2.27}$$

Notice that the drift dynamics of the real plant, given by (2.27), differ from the nominal model used for calculation of the initial stabilizing controller, given in (2.25). In fact it is the purpose of the reinforcement learning adaptation scheme to find the optimal control policy for the real plant while starting from the "optimal" controller corresponding to the nominal model of the plant.

The simulation was conducted using data obtained from the system at every 0.05s. For the purpose of demonstrating the algorithm the closed loop system was excited with an initial condition of 0.1 MW incremental change in generator output, the initial state of the system being $x_0 = \begin{bmatrix} 0 & 0.1 & 0 & 0 \end{bmatrix}$. The cost function parameters, namely the Q and R matrices, were chosen to be identity matrices of appropriate dimensions.

In order to solve online for the values of the P matrix which parameterizes the cost function, before each iteration step a least-squares problem of the sort described in Section 2.3.1, with the solution given by (2.23), was setup. Since there are 10 independent elements in the symmetric matrix P the setup of the least-squares problem requires at least 10 measurements of the cost function associated with the given control policy and measurements of the systems states at the beginning and the end of each time interval, provided that there is enough excitation in the system.

In this case, since the system states are not continuously excited and because resetting the state at each step is not an acceptable solution for online implementation, in order to have consistent data necessary to obtain the solution given by (2.23) one has to continue reading information from the system until the solution of the least-squares problem is feasible. A least squares problem was solved after 20 sample data were acquired and thus the controller was updated every 1 sec. The trajectory of the state of the system for the duration of the online experiment is presented in Figure 5.

It is clear that the cost function (*i.e.* critic) parameters converged to the optimal ones – indicated on the figure with star shaped points – which were placed for ease of comparison at t=5s. The values of the P matrix parameters at t=0s correspond to the solution of the Riccati equation that was solved, considering the approximate model of the system, to find the initial controller (2.26).

The values of the cost function parameters, i.e. the cost for using this initial controller given by (2.26), were calculated using the least squares based on the online

measurements of the augmented system states (including $V(t)$) and are indicated by the points placed at t=1s.

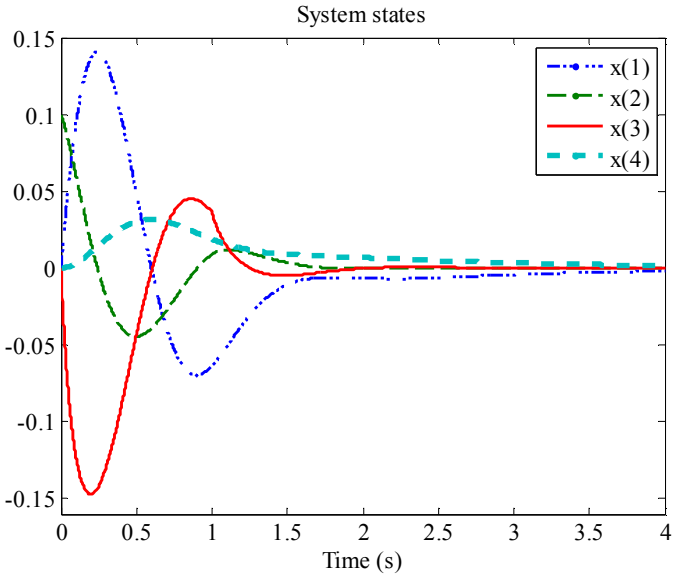

Figure 5. State trajectories of the linear closed loop power system over the duration of the experiment

The result of applying the algorithm for the power system is presented in Figure 6.

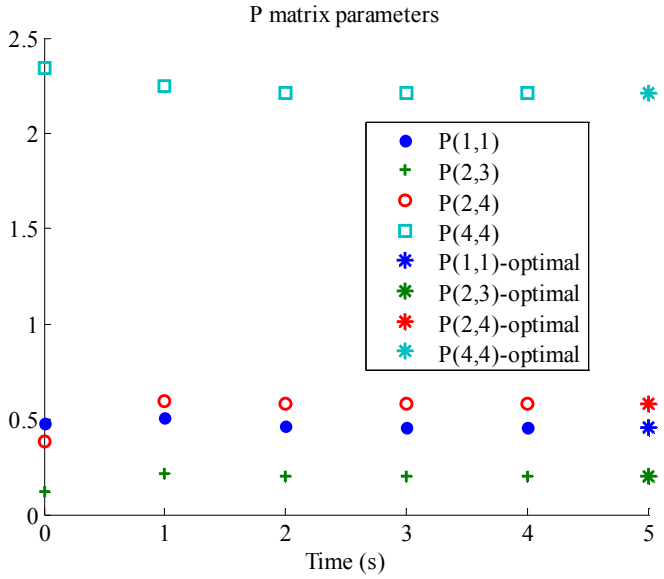

Figure 6. Evolution of the parameters of the *P* matrix for the duration of the experiment

The optimal controller, close in the range of 10^{-4} to the solution of the Riccati equation, was obtained at time t=4s, after four updates of the controller parameters. Figure 4 clearly ilustrates the fact that when the parameters of the cost function are close to the optimal ones the convergence rate of the algorithm is quadratic. This is a feature of the policy iteration algorithm, [32], which is retained by its online version.

The P matrix obtained online using the adaptive critic algorithm, without knowing the plant internal dynamics, is

$$P=\begin{bmatrix} 0.4599 & 0.6910 & 0.0518 & 0.4641 \\ 0.6910 & 1.8665 & 0.2000 & 0.5798 \\ 0.0518 & 0.2000 & 0.0532 & 0.0300 \\ 0.4641 & 0.5798 & 0.0300 & 2.2105 \end{bmatrix}. \tag{2.28}$$

The solution that was obtained by directly solving the algebraic Riccati equation considering the real plant internal dynamics (2.27) is

$$P=\begin{bmatrix} 0.4600 & 0.6911 & 0.0519 & 0.4642 \\ 0.6911 & 1.8668 & 0.2002 & 0.5800 \\ 0.0519 & 0.2002 & 0.0533 & 0.0302 \\ 0.4642 & 0.5800 & 0.0302 & 2.2106 \end{bmatrix}. \tag{2.29}$$

One can see that the error difference between the parameters of the two matrices is in the range of 10^{-4}.

In practice, the convergence of the algorithm is considered to be achieved when the difference between the measured cost and the expected cost crosses below a designer specified threshold value. It is important to note that after the convergence to the optimal controller was attained, the algorithm need not continue to be run and subsequent updates of the controller need not be performed.

In Figure 7 is presented a detail of the system state trajectories for the first two seconds of the simulation. The state values that were actually measured and subsequently used for the critic update computation are represented by the points on the state trajectories. Note that the control policy was updated at time t=1s.

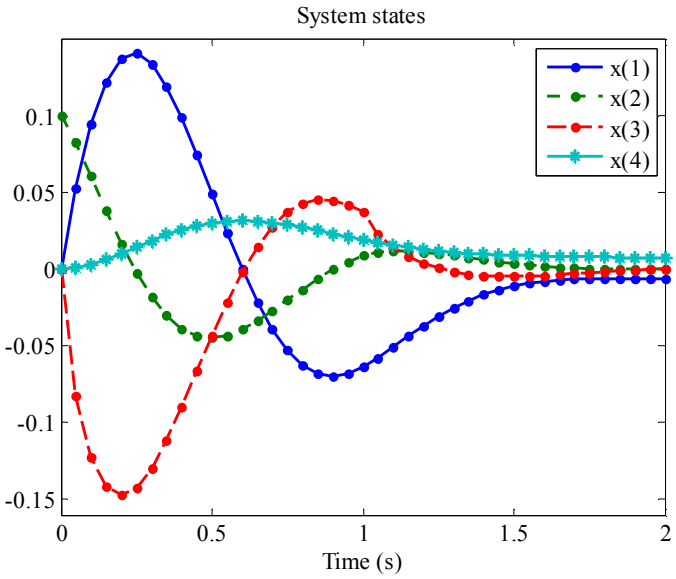

Figure 7. System state trajectories (lines), and state information that was actually used for the critic update (dots on the state trajectories)

Although in this case we had available a nominal model of the system and this allowed us to calculate a stabilizing controller to initialize the adaptive algorithm, it is important to point out that in the case when the system is itself stable this allows starting the iteration while using no controller (*i.e.* the initial controller is zero and no identification procedure needs to be performed).

In Figure 6 is presented the convergence result for the case the adaptive optimal control algorithm was initialized with no controller. The Critic parameters converged to

the optimal ones at time t=7s after seven updates of the controller parameters. The P matrix calculated with the adaptive algorithm is

$$P=\begin{bmatrix} 0.4601 & 0.6912 & 0.0519 & 0.4643 \\ 0.6912 & 1.8672 & 0.2003 & 0.5800 \\ 0.0519 & 0.2003 & 0.0533 & 0.0302 \\ 0.4643 & 0.5800 & 0.0302 & 2.2107 \end{bmatrix}. \tag{2.30}$$

The error difference between the parameters of the solution (2.30) obtained iteratively and the optimal solution (2.29) is in the range of 10^{-4}.

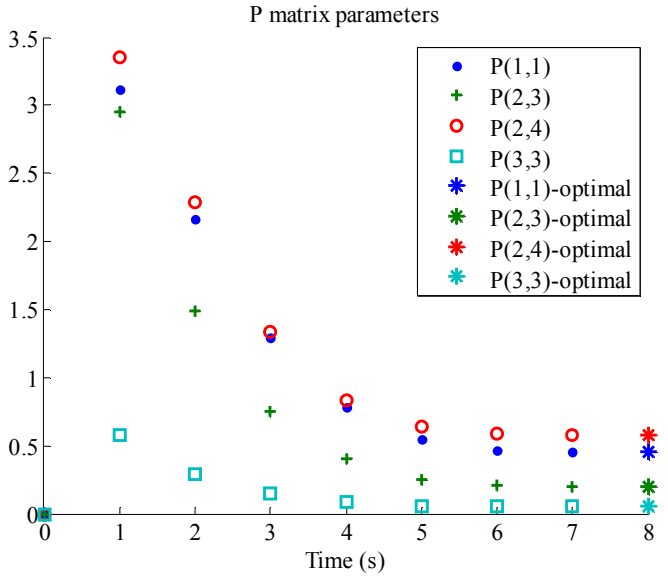

Figure 8. Evolution of the parameters of the P matrix for the duration of the experiment when the adaptive algorithm was started without controller for the power system

2.5 Conclusions

In this chapter was presented a new policy iteration technique which solves online the continuous time LQR problem without using knowledge about the system's internal dynamics (system matrix A). The algorithm is an online adaptive optimal controller based on an adaptive critic scheme in which the actor performs continuous time control

while the critic incrementally corrects the actor's behavior at discrete moments in time until best performance is obtained. The critic evaluates the actor performance over a period of time and formulates it in a parameterized form. Based on the critic's evaluation the actor behavior policy is updated for improved control performance.

The result can be summarized as an algorithm which effectively provides solution to the algebraic Riccati equation associated with the optimal control problem without using knowledge of the system matrix A. Convergence to the solution of the optimal control problem, under the condition of initial stabilizing controller, has been established by proving equivalence with the algorithm presented by Kleinman in [32]. The convergence results obtained in simulation for load-frequency optimal control of a power system generator have also been provided.

CHAPTER 3

REINFORCEMENT LEARNING APPROACH BASED ON POLICY ITERATION TO CONTINUOUS-TIME DIRECT ADAPTIVE OPTIMAL CONTROL FOR PARTIALLY UNKNOWN NONLINEAR SYSTEMS

3.1 Introduction

In this chapter is presented an adaptive method, which uses approximation structures in an actor-critic configuration, for solving online the optimal control problem for the case of nonlinear systems, in a continuous-time framework, without making use of explicit knowledge on the internal dynamics of the nonlinear system. The method is based on *policy iteration* (PI), a RL algorithm which iterates between the steps of policy evaluation and policy improvement. The PI method starts by evaluating the cost of a given admissible initial policy and then uses this information to obtain a new control policy, which is improved in the sense of having a smaller associated cost compared with the previous policy, over the domain of interest in the state space. The two steps are repeated until the policy improvement step no longer changes the present policy; this indicating that the optimal control behavior was obtained.

In the case of continuous-time systems with linear dynamics, PI was employed for finding the solution of the state feedback optimal control problem (i.e. LQR) in [45], while the convergence guarantee to the LQR solution was given in [32]. The PI algorithm, as used by Kleinman [32], requires repetitive solution of Lyapunov equations,

which involve complete knowledge of the system dynamics (i.e. both the input-to-state and internal system dynamics specified by the plant input and system matrices). Chapter 2 of this work presented the online PI algorithm which provides solution of the LQR problem using data measured along a single state trajectory, without requiring knowledge on the system's internal dynamics.

For nonlinear systems, the PI algorithm was first developed by Leake and Liu, [38], but at that time the mathematical techniques required for real implementation had not been developed. Three decades later PI was revisited and presented in [9] as a feasible adaptive solution to the CT optimal control problem. The main contribution of [9] resides in the fact that the Generalized HJB equations (a sort of nonlinear Lyapunov equations), which appear in the PI algorithm, could now be solved using successive Galerkin approximation algorithms. A neural-networks-based approach was developed and extended to the cases of H2 and H-infinity with constrained control in [2], [1]. Neural-network-based actor-critic structures, in a continuous-time framework, with neural network tuning laws have been given in [23]. All of the above mentioned methods require complete knowledge of the system dynamics.

In this Chapter is given a new formulation of the PI algorithm for continuous-time nonlinear systems. This new formulation allows online adaptation (i.e. learning) of the continuous-time operating controller to the optimal state feedback control policy, without requiring knowledge on the system internal dynamics. Knowledge regarding the

input-to-state dynamics is still required, but from a system identification point of view this knowledge is relatively easier to obtain.

In Section 3.3 the proof of convergence of the online PI algorithm is first given under the assumption the two function approximators in the actor/critic structure can provide exact representations of the control and cost functions. This shows the validity of the approach to online learning. However, the assumption of exact representation of the cost functions which have to be learned is not realistic. Thus the convergence results are then extended for the function-approximators–based algorithm, taking into account the existing approximation errors between the actor/critic structures and the control and cost functions respectively. The algorithm converges online to the optimal control solution without knowledge of the internal system dynamics. Closed-loop dynamic stability is guaranteed throughout.

The end result is a control and adaptation structure which is a hybrid combination between a continuous-time controller and a learning structure which operates based on discrete sampled data from the system and from the continuous-time dynamics reflecting the performance of the system. Such structure is unlike any of the standard forms of controllers appearing in the literature.

In the next section is given an overview of the optimal control problem for nonlinear systems. The proposed Policy Iteration algorithm which solves the HJB equation without requiring knowledge of the internal dynamics of the system is presented in Section 3.3. Convergence of the algorithm is proved by showing

equivalence with the general PI algorithm for nonlinear systems. The formulation of the introduced algorithm using approximation structures is discussed in Section 3.4. Convergence of the adaptive critic-based algorithm, while considering the error between the cost function and its approximation, is then provided. Section 3.5 gives a flowchart of the online algorithm and discusses the online implementation on an Actor/Critic structure, while commenting also on the relations between the proposed online algorithm and certain learning mechanisms in the mammal brain. Section 3.6 presents simulation results considering two nonlinear systems with quadratic and quartic cost functions.

3.2 Background in nonlinear optimal control

This section presents the formulation of the nonlinear optimal control problem.

Consider the time-invariant affine in the input dynamical system given by

$$\dot{x}(t)=f(x(t))+g(x(t))u(x(t))\,;\; x(0)=x_0 \tag{3.1}$$

with $x(t)\in R^n$, $f(x(t))\in R^n$, $g(x(t))\in R^{n\times m}$ and the input $u(t)\in U\subset\mathbb{R}^m$. Is assumed that $f(0)=0$, that $f(x)+g(x)u$ is Lipschitz continuous on a set $\Omega\subseteq\mathbb{R}^n$ which contains the origin, and that the dynamical system is stabilizable on Ω, i.e. there exists a continuous control function $u(t)\in U$ such that the system is asymptotically stable on Ω.

We note here that although global asymptotic stability is guaranteed in a linear system case, it is generally difficult to guarantee in a general continuous-time nonlinear system problem setting. This is due to the non-smooth nature of a nonlinear system

dynamics; at the points in which there exist discontinuities of $\dot{x}$, there will also exist discontinuities of the gradient of the cost function. For this reason the discussion is restricted to the case in which asymptotic stability is desired and sought for only in a region $\Omega \subseteq \mathbb{R}^n$ in which the cost function is continuously differentiable.

The infinite horizon integral cost associated with the control input $\{u(\tau);\tau \geq t\}$ is defined as

$$V^u(x(t)) = \int_t^\infty r(x(\tau), u(\tau))d\tau \tag{3.2}$$

where $x(\tau)$ denotes the solution of (3.1) for initial condition $x(t) \in \Omega$ and input $\{u(\tau);\tau \geq t\}$, $r(x,u) = Q(x) + u^T Ru$ with $Q(x)$ positive definite, i.e. $\forall x \neq 0, Q(x) > 0$ and $x = 0 \Rightarrow Q(x) = 0$, and $R \in \mathbb{R}^{m \times m}$ a positive definite matrix.

Definition 3.1 [9] (Admissible (stabilizing) policy)

A control policy $\mu(x)$ *is defined as admissible with respect to* (3.2) *on* Ω, *denoted by* $\mu \in \Psi(\Omega)$, *if* $\mu(x)$ *is continuous on* Ω, $\mu(0) = 0$, $\mu(x)$ *stabilizes* (3.1) *on* Ω *and* $V(x_0)$ *is finite* $\forall x_0 \in \Omega$.

The cost function associated with any admissible control policy $\mu \in \Psi(\Omega)$ is

$$V^\mu(x(t)) = \int_t^\infty r(x(\tau), \mu(x(\tau)))d\tau\,. \tag{3.3}$$

$V^\mu(x)$ is C^1. The infinitesimal version of (3.3) is

$$0 = r(x, \mu(x)) + (\nabla V_x^\mu)^T (f(x) + g(x)\mu(x)),\ V^\mu(0) = 0 \tag{3.4}$$

where $\nabla V_x{}^{\mu}$ (a column vector) denotes the gradient of the cost function V^{μ} with respect to x, as the cost function does not depend explicitly on time. Equation (3.4) is a Lyapunov equation for nonlinear systems which, given the controller $\mu(x)\in\Psi(\Omega)$, can be solved for the cost function $V^{\mu}(x)$ associated with it. Given that $\mu(x)$ is an admissible control policy, if $V^{\mu}(x)$ satisfies (3.4), with $r(x,\mu(x))\geq 0$, then $V^{\mu}(x)$ is a Lyapunov function for the system (3.1) with control policy $\mu(x)$.

The optimal control problem can now be formulated:

Given the continuous-time system (3.1), *the set* $u\in\Psi(\Omega)$ *of admissible control policies, and the infinite horizon cost functional* (3.2)*, find an admissible control policy such that the cost index* (3.2) *associated with the system* (3.1) *is minimized.*

Defining the Hamiltonian of the problem

$$H(x,u,V_x)=r(x(t),u(t))+(\nabla V_x)^T(f(x(t))+g(x(t))u(t)), \tag{3.5}$$

the optimal cost function $V^*(x)$ satisfies the HJB equation

$$0=\min_{u\in\Psi(\Omega)}[H(x,u,\nabla V_x^*)]. \tag{3.6}$$

Assuming that the minimum on the right hand side of the equation (3.6) exists and is unique then the optimal control function for the given problem is

$$u^*(x)=-\tfrac{1}{2}R^{-1}g^T(x)\nabla V_x{}^*. \tag{3.7}$$

Inserting this optimal control policy in the Hamiltonian we obtain the formulation of the HJB equation in terms of $\nabla V_x{}^*$

$$0 = Q(x) + (\nabla V_x^*)^T f(x) - \frac{1}{4}(\nabla V_x^*)^T g(x) R^{-1} g^T(x) \nabla V_x^*, \quad V^*(0) = 0. \qquad (3.8)$$

This is a sufficient condition for the optimal cost function (Kirk, 2004). For the linear system case, considering a quadratic cost functional, the equivalent of this HJB equation is the well known Riccati equation.

In order to find the optimal control solution for the problem one only needs to solve the HJB equation (3.8) for the cost function and then substitute the solution in (3.7) to obtain the optimal control. However, solving the HJB equation is generally difficult. It also requires complete knowledge of the system dynamics (*i.e.* the functions $f(x), g(x)$ need to be known).

3.3 Policy iteration algorithm for solving the HJB equation

In the following is presented a new online iterative algorithm which will adapt the parameters of the state feedback controller such that it will solve the infinite horizon optimal control problem without using knowledge regarding the system internal dynamics (*i.e.* the system function $f(x)$). Convergence of the algorithm to the optimal control function is then provided.

3.3.1 Policy iteration algorithm

Let $\mu^{(0)}(x(t)) \in \Psi(\Omega)$ be an admissible policy, and $T > 0$ such that as $x(t) \in \Omega$ also $x(t+T) \in \Omega$ (the existence of such $T > 0$ is guaranteed by the admissibility of $\mu^{(0)}(.)$ on Ω), then the iteration between:

1. (*policy evaluation* step) solve for $V^{\mu^{(i)}}(x(t))$ using

$$V^{\mu^{(i)}}(x(t)) = \int_{t}^{t+T} r(x(s), \mu^{(i)}(x(s)))ds + V^{\mu^{(i)}}(x(t+T)) \text{ with } V^{\mu^{(i)}}(0) = 0, \quad (3.9)$$

and

2. (*policy improvement* step) update the control policy using

$$\mu^{(i+1)} = \arg\min_{u \in \Psi(\Omega)} [H(x, u, \nabla V_x^{\mu^{(i)}})], \quad (3.10)$$

which explicitly is

$$\mu^{(i+1)}(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V_x^{\mu^{(i)}}, \quad (3.11)$$

converges to the optimal control policy $\mu^* \in \Psi(\Omega)$ with corresponding cost $V^*(x_0) = \min_{\mu}(\int_0^{\infty} r(x(\tau), \mu(x(\tau)))d\tau)$.

Equations (3.9) and (3.11) give a new formulation for the Policy Iteration algorithm which allows solving the optimal control problem without making use of any knowledge of the internal dynamics of the system, $f(x)$. This algorithm is an online version of the offline algorithms proposed in [2] and [9], motivated by the success of the online adaptive critic techniques proposed by computational intelligence researchers [45], [48], [10]. In the spirit of reinforcement learning algorithms, the integral term in (3.9) can be addressed as the *reinforcement* over the time interval $[t, t+T)$.

Equation (3.9) is a discretized version of $V^{\mu^{(i)}}(x(t)) = \int_{t}^{\infty} r(x(\tau), \mu^{(i)}(x(\tau)))d\tau$ and it can be viewed as a Lyapunov equation for nonlinear systems. In this paper we shall refer to it also as

$$LE(V^{\mu^{(i)}}(x(t))) \triangleq \int_{t}^{t+T} r(x(s), \mu^{(i)}(x(s)))ds + V^{\mu^{(i)}}(x(t+T)) - V^{\mu^{(i)}}(x(t))$$

with $V^{\mu^{(i)}}(0)=0$.

The convergence of the new PI algorithm is given in the next subsection. The implementation of the algorithm using approximation structures will be discussed in Section 3.4.

3.3.2 Convergence of the policy iteration algorithm

It has been shown that if $\mu^{(i)} \in \Psi(\Omega)$ and $V^{\mu^{(i)}}(x(t)) \in C^1(\Omega)$ satisfy equation (3.9) then one can show the new control policy $\mu^{(i+1)}$, determined based on equation (3.10), is admissible for the system (3.1). (for proof see [2] and [9])

The following result is required in order to prove the convergence of the proposed policy iteration algorithm.

Lemma 3.1 *Solving for* $V^{\mu^{(i)}}$ *in equation* (3.9) *is equivalent with finding the solution of*

$$0 = r(x, \mu^{(i)}(x)) + (\nabla V_x^{\mu^{(i)}})^T (f(x) + g(x)\mu^{(i)}(x)),\ V^{\mu^{(i)}}(0)=0. \qquad (3.12)$$

The proof is given in Appendix A.

Remark 1 Although the same solution is obtained solving either equation (3.9) or (3.12), solving equation (3.9) does not require any knowledge on the system dynamics $f(x)$, which in turn appears explicitly in (3.12).

From Lemma 3.1 it follows that the algorithm (3.9) and (3.11) is equivalent to iterating between (3.12) and (3.11), without using knowledge of the system internal dynamics $f(x)$.

Theorem 3.1 *(convergence) The policy iteration (3.9) and (3.11) converges uniformly to the optimal control solution on the trajectories originating in* Ω*, i.e.*

$$\begin{aligned} &\forall\, \varepsilon > 0\ \exists i_0 : \forall i \geq i_0 \\ &\sup_{x\in\Omega}\left|V^{\mu^{(i)}}(x) - V^*(x)\right| < \varepsilon\,, \quad \sup_{x\in\Omega}\left|\mu^{(i)}(x) - u^*(x)\right| < \varepsilon. \end{aligned} \tag{3.13}$$

Proof In [9] and [2] it was shown that iterating on equations (3.12) and (3.11), conditioned by an initial admissible policy $\mu^{(0)}(x)$, all the subsequent control policies will be admissible and the iteration (3.12) and (3.11) will converge to the solution of the HJB equation, i.e. equation (3.13) is satisfied.

Based on the proven equivalence between the equations (3.9) and (3.12) one concludes that the proposed online adaptive optimal control algorithm will converge to the solution of the optimal control problem (3.2), on Ω, without using knowledge on the internal dynamics of the controlled system (3.1). □

3.4 Adaptive critics solution of the HJB equation

For the implementation of the iteration scheme given by (3.9) and (3.11) one only needs to have knowledge of the input-to-state dynamics, i.e. the function $g(x)$, which is required for the policy update in equation (3.11). One can see that knowledge on the internal state dynamics, described by $f(x)$, is not required. The information regarding the system $f(x)$ matrix is embedded in the states $x(t)$ and $x(t+T)$ which are sampled online.

3.4.1 Approximate representation of the cost function

Equation (3.9) is solved making use of a structure which will approximate the cost function solution for any $x \in \Omega$. In here is considered that a linear combination of a finite set of basis functions can be determined such that it closely approximates the cost function $V^{\mu^{(i)}}(x)$, for $x \in \Omega$. Thus the cost function can be represented as

$$V_L{}^{\mu^{(i)}}(x) = \sum_{j=1}^{L} w_j{}^{\mu^{(i)}} \phi_j(x) = (\mathbf{w}_L{}^{\mu^{(i)}})^T \boldsymbol{\varphi}_L(x). \tag{3.14}$$

Note that given an infinite set of linearly independent activation functions $\{\phi_j(x)\}_1^{\infty}$, such that $\phi_j(x) \in C^1(\Omega),\ \phi_j(0) = 0,\ j = \overline{1,\infty}$, which satisfy the completeness property (i.e. any function $f(x) \in C^1(\Omega),\ f(0) = 0$ can be represented as a linear combination of a subset of $\{\phi_j(x)\}_1^{\infty}$), then the exact solution of equation (3.9) can be expressed as

$$V^{\mu^{(i)}}(x) = \sum_{j=1}^{\infty} c_j{}^{\mu^{(i)}} \phi_j(x) = (\mathbf{c}_{\infty}{}^{\mu^{(i)}})^T \boldsymbol{\varphi}_{\infty}(x), \tag{3.15}$$

where $\boldsymbol{\varphi}_{\infty}(x)$ is the vector of activation functions and $\mathbf{c}_{\infty}{}^{\mu^{(i)}}$ denotes the weight vector.

Using the approximate description for the cost function, equation (3.14), (3.9) can be written as

$$\mathbf{w}_L{}^{\mu^{(i)}T} \boldsymbol{\varphi}_L(x(t)) = \int_t^{t+T} r(x, \mu^{(i)}(x)) d\tau + \mathbf{w}_L{}^{\mu^{(i)}T} \boldsymbol{\varphi}_L(x(t+T)). \tag{3.16}$$

As the cost function was replaced with its approximation, (3.16) will have the residual error

$$\delta_L^{\mu^{(i)}}(x(t),T)=\int_t^{t+T} r(x,\mu^{(i)}(x))d\tau+\mathbf{w}_L{}^{\mu^{(i)}T}[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))]. \qquad (3.17)$$

From the perspective of temporal difference learning methods (e.g. [5], [17]) this error can be viewed as a temporal difference residual error for continuous-time systems.

To determine the parameters of the function that is approximating the cost function $V_L{}^{\mu^{(i)}}$, in the least-squares sense, we use the method of weighted residuals. Thus the parameters $\mathbf{w}_L{}^{\mu^{(i)}}$ of the cost function approximation $V_L{}^{\mu^{(i)}}$ are adapted such that to minimize the objective

$$S=\int_\Omega \delta_L^{\mu^{(i)}}(x,T)\delta_L^{\mu^{(i)}}(x,T)dx\,. \qquad (3.18)$$

This amounts to $\int_\Omega \frac{d\delta_L^{\mu^{(i)}}(x,T)}{d\mathbf{w}_L{}^{\mu^{(i)}}}\delta_L^{\mu^{(i)}}(x,T)dx=0$. Using the inner product notation for the Lebesgue integral one can write

$$\left\langle \frac{d\delta_L^{\mu^{(i)}}(x,T)}{d\mathbf{w}_L{}^{\mu^{(i)}}},\delta_L^i(x,T)\right\rangle_\Omega=0 \qquad (3.19)$$

which is

$$\begin{aligned}&\left\langle[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))],[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))]^T\right\rangle_\Omega \mathbf{w}_L{}^{\mu^{(i)}}+\\&+\left\langle[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))],\int_t^{t+T} r(x(s),\mu^{(i)}(x(s)))ds\right\rangle_\Omega=0\end{aligned} \qquad (3.20)$$

Conditioned by $\Phi=\left\langle[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))],[\boldsymbol{\varphi}_L(x(t+T))-\boldsymbol{\varphi}_L(x(t))]^T\right\rangle_\Omega$ being invertible, then the solution is

$$\mathbf{w}_L{}^{\mu^{(i)}} = -\Phi^{-1}\left\langle [\boldsymbol{\varphi}_L(x(t+T)) - \boldsymbol{\varphi}_L(x(t))], \int_t^{t+T} r(x(s), \mu^{(i)}(x(s)))ds \right\rangle_{\Omega} \tag{3.21}$$

To show that Φ can be inverted the following technical results are needed.

Definition 3.2 (linearly independent set of functions) [33]

A set of functions $\{\phi_j\}_1^N$ *is said to be linearly independent on a set* Ω *if* $\sum_{j=1}^{N} c_j\phi_j(x) = 0$ *a.e. on* Ω *implies that* $c_1 = \cdots = c_N = 0$.

Lemma 3.2 *If the set* $\{\phi_j\}_1^N$ *is linearly independent and* $u \in \Psi(\Omega)$ *then the set* $\{\nabla\phi_j{}^T(f+gu)\}_1^N$ *is also linearly independent.*

The proof is given in [9].

The next technical lemma shows that Φ can be inverted.

Lemma 3.3 *Let* $\mu(x) \in \Psi(\Omega)$ *such that* $f(x)+g(x)\mu(x)$ *is asymptotically stable. Given that the set* $\{\phi_j\}_1^N$ *is linearly independent then* $\exists T > 0$ *such that* $\forall x(t) \in \Omega - \{0\}$, *the set* $\{\bar{\phi}_j(x(t), T) = \phi_j(x(t+T)) - \phi_j(x(t))\}_1^N$ *is also linearly independent.*

The proof is by contradiction and is presented in Appendix A.

Based on the result of Lemma 3.3, there exist values of *T* such that Φ is invertible and the parameters $\mathbf{w}_L{}^{\mu^{(i)}}$ of the cost function $V_L{}^{\mu^{(i)}}$ can be calculated. Having solved for the cost function $V_L{}^{\mu^{(i)}}$ associated with the control policy $\mu^{(i)}$, the policy update step can be executed. The new control policy will thus be

$$\mu^{(i+1)}(x)=-\tfrac{1}{2}R^{-1}g^{T}(x)\nabla\boldsymbol{\varphi}_{L}{}^{T}(x)\mathbf{w}_{L}{}^{\mu^{(i)}}. \tag{3.22}$$

Equation (3.22) gives the output of the actor structure. Note that in this implementation the controller (actor) can be seen as an approximation structure, which has the same weight parameters as the critic, but whose basis set of functions depend on the gradients of those in the critic.

3.4.2 Convergence of $V_{L}{}^{\mu^{(i)}}(x)$ to the exact solution of the Lyapunov equation $V^{\mu^{(i)}}(x)$

The convergence of the method of least squares is now discussed for the case in which equation (3.9) is solved using a cost function approximator.

Definition 3.3 *(Convergence in the mean) A sequence of Lebesgue integrable functions on a set Ω, $\{f_n(x)\}\in L_2(\Omega)$, is said to converge in the mean to $f(x)\in L_2(\Omega)$ if $\forall\varepsilon>0,\exists N(\varepsilon)$ such that $\forall n>N(\varepsilon),\left\|f_n(x)-f(x)\right\|_{L_2(\Omega)}<\varepsilon$, where $\left\|f(x)\right\|^2_{L_2(\Omega)}=\langle f,f\rangle$.*

Equation (3.9) can be written using a linear operator *A* defined on the Hilbert space of continuous and differentiable functionals on Ω

$$\overbrace{V^{\mu}(x(t))}^{AV^{\mu}}-V^{\mu}(x(t+T))=\overbrace{\int_{t}^{t+T}r(x(s),\mu(x(s)))ds}^{d(x,\mu(x),T)}. \tag{3.23}$$

Function approximators which are defined such that the basis functions are power series of order *m* are differentiable and can uniformly approximate a continuous function with all its partial derivatives, up to order *m*, by differentiating the series term-wise. This type of series is *m*-uniformly dense as shown in Lemma 3.4.

Lemma 3.4 (Higher order Weierstrass approximation theorem) [26].

Let $f(x)\in C^m(\Omega)$*, then there exists a polynomial* $P(x)$ *such that it converges uniformly to* $f(x)$*, and all its partial derivatives up to order m converge uniformly.*

The following facts hold under the stated standard conditions in optimal control.

Fact 1 The solution of (3.9) is positive definite. This is guaranteed when the system has stabilizable dynamics and when the performance functional satisfies zero state observability (i.e. observability of the system state through the cost function). [59]

Fact 2 The system dynamics and the performance integrand $r(x(s),\mu(x(s)))$ are such that the solution of (3.9) is continuous and differentiable on Ω.

Fact 3 A complete set $\{\phi_j\}_1^\infty \in C^1(\Omega)$ can be chosen such that the solution $V\in C^1(\Omega)$ and ∇V can be uniformly approximated by the infinite series built based on $\{\phi_j\}_1^\infty$.

Fact 4 The sequence $\{\bar{\phi}_j(x(t),T)=\phi_j(x(t+T))-\phi_j(x(t))\}_1^\infty$ is linearly independent and complete.

Proof The linear independence results from Lemma 3.3, being conditioned by certain values of the sample time T. The completeness relies on the high-order Weierstrass approximation theorem.

$\forall V,\varepsilon\ \exists L,\mathbf{w}_L$ such that $|V_L-V|<\varepsilon$. This implies that as $L\to\infty$ $\sup_{x\in\Omega}|AV_L-AV|\to 0\Rightarrow\|AV_L-AV\|_{L_2(\Omega)}\to 0$ which proves completeness of $\{\bar{\phi}_j(x(t),T)=A\phi_j\}_1^\infty$.

The first three assumptions are standard in optimal control, and we have proven the fourth herein. The next result is required.

Lemma 3.5 *Given a set of N linearly independent functions* $\{f_j(x)\}_1^N$ *defined on* Ω *then*

$$\left\|\boldsymbol{\alpha}_N{}^T\mathbf{f}_N\right\|_{L_2(\Omega)}^2 \to 0 \Leftrightarrow \left\|\boldsymbol{\alpha}_N\right\|_{l_2}^2 \to 0. \tag{3.24}$$

A proof is given in [2].

The next main result shows convergence in the mean.

Theorem 3.2 *Given that the Facts 1-4 hold, then approximate solutions exist for* (3.9) *using the method of least squares and are unique for each L. In addition, as* $L \to \infty$,

R1. $\left\|LE(V_L{}^{\mu^{(i)}}(x)) - LE(V^{\mu^{(i)}}(x))\right\|_{L_2(\Omega)} \to 0$,

where $LE(V(x))$ is defined in section 3.3.1,

R2. $\left\|V_L{}^{\mu^{(i)}}(x) - V^{\mu^{(i)}}(x)\right\|_{L_2(\Omega)} \to 0$,

R3. $\left\|\nabla V_L{}^{\mu^{(i)}}(x) - \nabla V^{\mu^{(i)}}(x)\right\|_{L_2(\Omega)} \to 0$,

R4. $\left\|\mu_L{}^{(i)}(x) - \mu^{(i)}(x)\right\|_{L_2(\Omega)} \to 0$.

Proof. The least squares sense solution $V_L{}^{\mu^{(i)}}$ of (3.9) is the solution of the minimization problem

$$\left\|AV_L{}^{\mu^{(i)}} - d(x,\mu^{(i)},T)\right\|^2 = \min_{\mathbf{w}_L}\left\|\mathbf{w}_L{}^T\overline{\boldsymbol{\varphi}}(x) - d(x,\mu^{(i)},T)\right\|^2. \tag{3.25}$$

The uniqueness of the solution follows directly from the linear independence of $\{\bar{\phi}_j(x(t),T)\}_1^L$. R1 follows from the completeness of $\{\bar{\phi}_j(x(t),T)=A\phi_j\}_1^\infty$.

R2 is next proved.

$$LE(V_L{}^{\mu^{(i)}}(x))-LE(V^{\mu^{(i)}}(x))=\mathbf{w}_L{}^T\bar{\boldsymbol{\varphi}}_L(x,T)-\mathbf{c}_\infty{}^T\bar{\boldsymbol{\varphi}}_\infty(x,T)=\varepsilon_L(x,T), \quad (3.26)$$

$$(\mathbf{w}_L-\mathbf{c}_L)^T\bar{\boldsymbol{\varphi}}_L(x,T)=\varepsilon_L(x,T)+\sum_{j=L+1}^{\infty}c_j\bar{\phi}_j(x,T)=\varepsilon_L(x,T)+e_L(x,T). \quad (3.27)$$

$e_L(x,T)$ converges uniformly to zero due to the high-order Weierstrass approximation theorem (this implies convergence in the mean) and $\varepsilon_L(x,T)$ converges in the mean to zero due to R1. Then

$$\begin{aligned}\left\|(\mathbf{w}_L-\mathbf{c}_L)^T\bar{\boldsymbol{\varphi}}_L(x,T)\right\|_{L_2(\Omega)}^2&=\left\|\varepsilon_L(x,T)+e_L(x,T)\right\|_{L_2(\Omega)}^2\\&\le 2\left\|\varepsilon_L(x,T)\right\|_{L_2(\Omega)}^2+2\left\|e_L(x,T)\right\|_{L_2(\Omega)}^2\to 0\end{aligned}. \quad (3.28)$$

Since $\bar{\boldsymbol{\varphi}}_L(x,T)$ is linearly independent then, based on Lemma 3.5, one sees that $\left\|(\mathbf{w}_L-\mathbf{c}_L)\right\|_{l_2}^2\to 0$. As the set $\{\phi_j\}_1^L$ is linearly independent, it follows from Lemma 3.5 that $\left\|(\mathbf{w}_L-\mathbf{c}_L)^T\boldsymbol{\varphi}_L(x)\right\|_{L_2(\Omega)}^2\to 0$. It thus follows that, as $L\to\infty$, $\left\|V_L{}^{\mu^{(i)}}-V^{\mu^{(i)}}\right\|_{L_2(\Omega)}^2\to 0$.

Similarly, since $\left\{\frac{d\phi_j}{dx}\right\}_1^L$ is linearly independent, from Lemma 3.5 results that $\left\|(\mathbf{w}_L-\mathbf{c}_L)^T\nabla\boldsymbol{\varphi}_L(x)\right\|_{L_2(\Omega)}^2\to 0$, from which follows R3, $\left\|\nabla V_L{}^{\mu^{(i)}}-\nabla V^{\mu^{(i)}}\right\|_{L_2(\Omega)}^2\to 0$.

R4 follows immediately from R3 given that

$$\left\|\mu_L{}^{(i)}(x)-\mu^{(i)}(x)\right\|^2_{L_2(\Omega)}=\left\|-R^{-1}g^T(x)(\nabla V_L{}^{\mu^{(i-1)}}(x)-\nabla V^{\mu^{(i-1)}}(x))\right\|^2_{L_2(\Omega)} \quad (3.29)$$
$$\leq\left\|-R^{-1}g^T(x)\right\|^2_{L_2(\Omega)}\left\|\nabla V_L{}^{\mu^{(i-1)}}(x)-\nabla V^{\mu^{(i-1)}}(x)\right\|^2_{L_2(\Omega)}\to 0$$

□

Based on the result in Theorem 3.2 the following stronger result of uniform convergence can be shown.

Corollary 3.1 *If the results from Theorem 3.2 hold then*

$$\sup_{x\in\Omega}\left|V_L{}^{\mu^{(i)}}(x)-V^{\mu^{(i)}}(x)\right|\to 0,$$

$$\sup_{x\in\Omega}\left|\nabla V_L{}^{\mu^{(i)}}(x)-\nabla V^{\mu^{(i)}}(x)\right|\to 0 \text{ and}$$

$$\sup_{x\in\Omega}\left|\mu_L{}^{(i)}(x)-\mu^{(i)}(x)\right|\to 0.$$

For proof see [2].

The next result shows that, given an initial admissible control policy, $\mu^{(0)}(x)$, the control policy at each step *i* of the Policy Iteration algorithm, with value function approximation, $\mu_L^{(i)}(x)$ is admissible provided that the number of the basis functions in the approximation structure is sufficiently large.

Corollary 3.2 *(Admissibility of* $\mu_L{}^{(i)}(x)$*)* $\exists L_0$ *such that* $\forall L>L_0, \mu_L^{(i)}\in\Psi(\Omega)$.

The proof is given in Appendix A.

Corollary 3.3 $\sup_{x\in\Omega}\left|\mu_L{}^{(i)}(x)-\mu^{(i)}(x)\right|\to 0 \quad\Rightarrow\quad \sup_{x\in\Omega}\left|V_L{}^{(i)}(x)-V^{(i)}(x)\right|\to 0.$

3.4.3 Convergence of the method of least squares to the solution of the HJB equation

In this section we show that the successive least squares solution using neural networks converges to the solution of the HJB equation (3.8).

Theorem 3.3 *Under the assumptions of Theorem 3.2 the following is satisfied* $\forall i \geq 0$

i. $\sup_{x \in \Omega} \left| V_L^{(i)}(x) - V^{(i)}(x) \right| \to 0$

ii. $\sup_{x \in \Omega} \left| \mu_L^{(i+1)}(x) - \mu^{(i+1)}(x) \right| \to 0$

iii. $\exists L_0 : \forall L \geq L_0 \quad \mu_L^{(i)}(x) \in \Psi(\Omega)$

A proof by induction and is presented in [2].

Theorem 3.4 $\forall \varepsilon \geq 0, \exists i_0, L_0 : \forall i \geq i_0, L \geq L_0$

i. $\sup_{x \in \Omega} \left| V_L^{(i)}(x) - V^*(x) \right| < \varepsilon$,

ii. $\sup_{x \in \Omega} \left| \mu_L^{(i+1)}(x) - u^*(x) \right| < \varepsilon$,

iii. $\mu_L^{(i)}(x) \in \Psi(\Omega)$.

The proof follows directly from Theorems 3.1 and 3.3.

3.5 Online algorithm on an actor-critic structure

This section discusses the implementation of the adaptive algorithm on the actor/critic structure. The main features of the online adaptive critic structure are presented while noting similarities with learning mechanisms in the mammal brain.

3.5.1 Actor-critic structure for online implementation of the adaptive optimal control algorithm

The structure of the system with the adaptive controller is presented in Figure 9. It is important to note that the adaptation structure has dynamics consisting of the state $V(t)$, i.e. the value, which evolves based on $\dot{V}=Q(x)+u^T Ru$. This provides a dynamic memory that enables one to extract the information regarding the cost associated with the given policy. If one resets $V(t)$ to zero at the beginning of each sample interval $[t,t+T)$, then the measurement $V(t+T)$ gives the reinforcement over time interval $[t,t+T)$ required to implement the policy evaluation step in (3.9), i.e. $V(t+T)$ gives the integral reinforcement term in (3.9). From this perspective the result is a dynamic controller whose memory is exactly the value $V(t)$ of using the current policy.

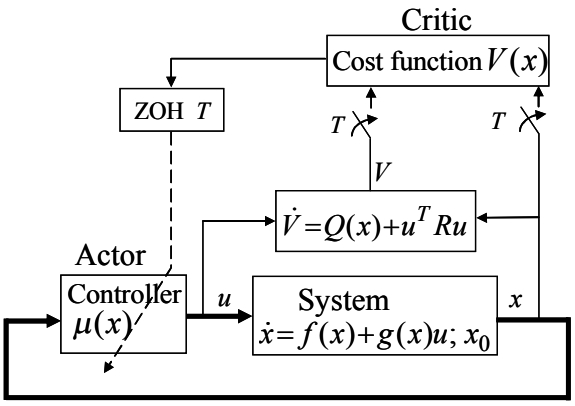

Figure 9. Structure of the system with adaptive controller

The policy iteration technique in this paper has led us to a control system structure that allows one to perform optimal control in an adaptive fashion online without knowing the internal dynamics of the system. We term this optimal adaptive control. This structure is not a standard one in the control systems literature. It is a hybrid

continuous-time/discrete-time adaptive control structure which has continuous-time dynamics, and a discrete-time sampled data portion for policy evaluation.

The algorithm is suitable for online implementation from the control theory point of view since the control policy $\mu_L^{(i+1)}(x)$, updated at time t_{i+1} after observing the state $x(t_{i+1})$, will be used for controlling the system during the time interval $(t_{i+1}, t_{i+1}+T]$.

The flowchart of the online algorithm is presented in Figure 10.

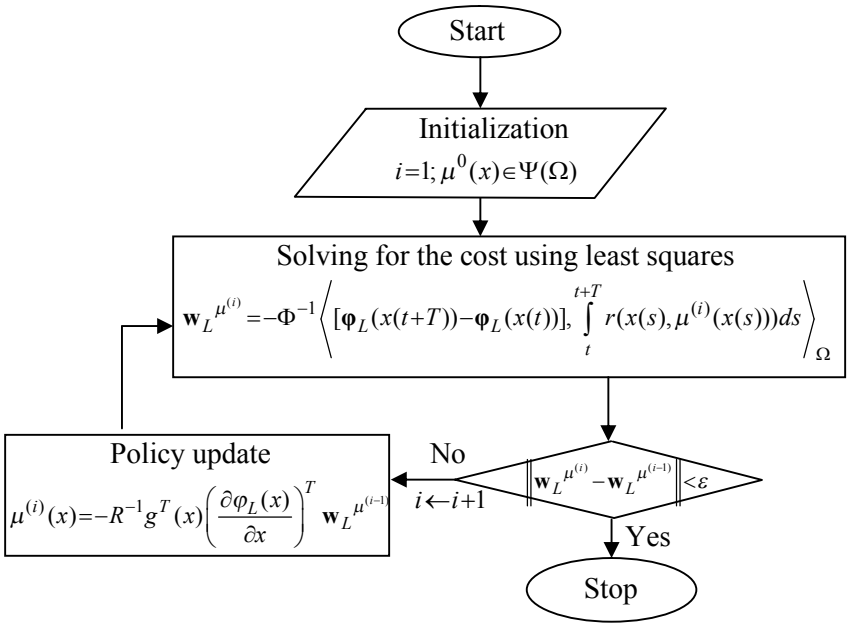

Figure 10. Flowchart of the online policy iteration algorithm

All the calculations involved are performed at a supervisory level which operates based on discrete–time data measured from the system. This high level intelligent control structure implements the policy iteration algorithm and uses the critic neural network to parameterize the performance of the continuous-time control system associated with a certain control policy. The high level supervisory structure makes the

decisions relative to the discrete-time moments at which both the actor and the critic parameters will be updated. The actor neural network is part of the control system structure and performs continuous-time control, while its constant gain is updated at discrete moments in time. The algorithm converges to the solution of the continuous-time optimal control problem, as proved in Section 3.4, since the critic's update is based on the observations of the continuous-time cost over a finite sample interval. The net result is a *continuous-time controller* incorporated in a *continuous-time/discrete-time adaptive structure*, which includes the continuous time dynamics of the cost function and operates based on sampled data, to perform the policy evaluation and policy update steps at discrete moments in time.

The cost function solution, given by (3.21), can be obtained in real-time after a sufficient number of data points are collected along state trajectories in the region of interest Ω. In practice, the matrix inversion in (3.21) is not performed, the solution of the equation being obtained using algorithms that involve techniques such as Gaussian elimination, backsubstitution, and Householder reflections. Also, the least squares method for finding the parameters of the cost function can be replaced with any other suitable, recursive or not recursive, method of parameter identification.

The iterations will be stopped (i.e. the critic will stop updating the control policy) when the error between the system performance evaluated at two consecutive steps will cross below a designer specified threshold. Also, when this error becomes bigger than the above mentioned threshold, indicating a change in the system dynamics, the critic will take again the decision to start tuning the actor parameters.

We note again that there is no required knowledge about the system dynamics $f(x)$ for the evaluation of the cost or the update of the control policy. However knowledge on the $g(x)$ function is required for the update of the control policy, using (3.22), and this makes the online tuning algorithm only partially model free.

3.5.2 Relation of the adaptive critic control structure to learning mechanisms in the mammal brain

It is interesting to note the rough similarity between the above mentioned adaptive controller structure and learning mechanisms in the mammal brain. The critic structure learns, in an episodic manner and based on samples of the reward signal from the environment, the parameters of a function which describes the actor performance. Once a performance evaluation episode was completed, the critic passes this information to the actor structure which will use it to adapt for improved performance. At all times the actor must perform continuous-time control for the system (the environment in which optimal behavior is sought). This description of the way in which the actor/critic structure works while searching for continuous-time optimal control policies points out the existence of two time scales for the mechanisms involved:

- a fast time scale which characterizes the continuous-time control process, and
- a slower time scale which characterizes the learning processes at the levels of the critic and the actor.

Thus the actor and critic structures perform tasks at different operation frequencies in relation with the nature of the task to be performed (i.e. learning or control). Evidence regarding the oscillatory behavior naturally characterizing biological neural

systems is presented in a comprehensive manner in [39]. Different oscillation frequencies are connected with the way in which different areas of the brain perform their functions of processing the information received from the sensors. Low level control structures must quickly react to new information received from the environment while higher level structures slowly evaluate the results associated with the present behavior policy.

In Section 3.4 it was shown that having little information about the system states measured from the sensors, x, and the augmented system state, i.e. V, extracted from the system only at specific time values (*i.e.* $x(t), x(t+T)$ and $V(t+T)-V(t)$), the Critic is able to evaluate the infinite horizon continuous-time performance of the system associated with a given control policy described in terms of the Actor parameters. The critic learns the cost function associated with a certain control behavior based on a computed temporal difference (TD) error signal, given by $V(t+T)-V(t)$.

It is interesting to mention here that in a number of reports, e.g. [49], [50], it is argued that the temporal difference error between the received and the expected rewards is physically encoded in the dopamine signal produced by basal ganglia structures in the mammal brain. At the same time, it is known that the dopamine signal encoding the temporal error difference favors the learning process by increasing the synaptic plasticity of certain groups of neurons.

The next section presents simulation results which were obtained considering two second order nonlinear systems.

3.6 Simulation examples

In this section the adaptive optimal control algorithm is tested in simulation considering two nonlinear systems for which the optimal cost function and optimal controller are known. The nonlinear system examples were developed using the converse HJB approach, [46], which allows construction of nonlinear systems, specified initially in a general form, starting from the known optimal cost function. In effect it solves conversely the HJB equation, given the optimal cost function, for the dynamics of the nonlinear system.

3.6.1 Example 1

The first nonlinear system is given by the equations

$$\begin{cases} \dot{x}_1 = -x_1 + x_2 \\ \dot{x}_2 = f(x) + g(x)u \end{cases} \tag{3.30}$$

with $f(x) = -\frac{1}{2}(x_1 + x_2) + \frac{1}{2}x_2 \sin^2(x_1),\ g(x) = \sin(x_1)$.

If the infinite horizon cost function to be minimized is $V^u(x(t)) = \int_t^\infty (Q(x) + u^2)d\tau$,

with $Q(x) = x_1^2 + x_2^2$, then the optimal cost function for this system is $V^*(x) = \frac{1}{2}x_1^2 + x_2^2$

and the optimal controller is $u^*(x) = -\sin(x_1)x_2$.

The simulation was conducted using data obtained from the system at every 0.1s. We note here that the value of this sample time is not relevant for the cost function identification procedure. In fact data does not have to be measured with a fixed sample time, as long as it is suitable for learning (i.e. caries new information on the cost

function to be identified). In this sense, as long as the measured signals did not reach steady state values, meaning that the measured data is not redundant, the sampling could be executed as fast the hardware permits it. At the same time, as we used a batch method for identifying the cost function parameters in the least squares sense a larger number of samples will lead to better approximation of the cost function. However this batch procedure can be replaced with a recursive one, or a recursive procedure on time windows, such that the parameters of the cost function will be adapted over time as more data is acquired.

For the purpose of demonstrating the algorithm the initial state of the system is taken to be different than zero. For each iteration we considered data measured along five trajectories defined by five different initial conditions chosen randomly in $\Omega=\{-1\le x_i\le 1; i=1,2\}$. The initial stabilizing controller was taken as $\mu^{(0)}(x)=-\frac{3}{2}\sin(x_1)(x_1+x_2)$. The cost function $V^{\mu^{(i)}}(x)$ was approximated by the following smooth function, for $x\in\Omega$, $V_L{}^{\mu^{(i)}}(x)=(\mathbf{w}_L{}^{\mu^{(i)}})^T\boldsymbol{\varphi}_L(x)$ with $L=3$, $\mathbf{w}_3{}^{\mu^{(i)}}=\begin{bmatrix} w_1^{\mu^{(i)}} & w_2^{\mu^{(i)}} & w_3^{\mu^{(i)}}\end{bmatrix}^T$ and $\boldsymbol{\varphi}_3(x)=\begin{bmatrix} x_1{}^2 & x_1x_2 & x_2{}^2\end{bmatrix}^T$.

In order to solve online for the parameters $\mathbf{w}_3{}^{\mu^{(i)}}$ of the cost function, at each iteration step we setup a least squares problem with the solution given by (3.21). At each iteration step we solved for $\mathbf{w}_3{}^{\mu^{(i)}}$ using 30 data points consisting of the measured the cost function associated with a given control policy over 30 time intervals T=0.1s, the initial state and the system state at the end of each time interval, 6 points measured

over each of the 5 trajectories in the state space. In this way, every 3s, the cost function was solved for and a policy update was performed. The result of applying the algorithm is presented in Figure 11.

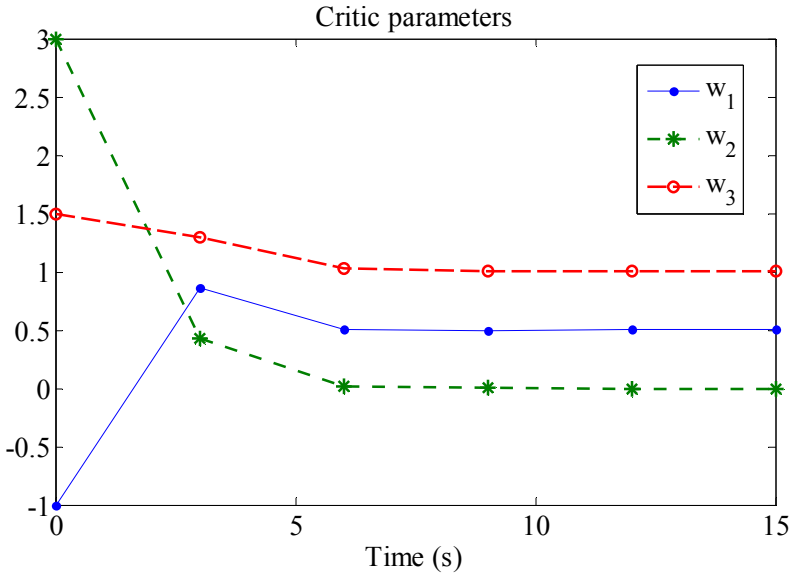

Figure 11. Convergence of the critic parameters

One can see from the figure that the parameters of the critic converged to the coefficients of the optimal cost function $V^*(x)=\frac{1}{2}x_1^2+x_2^2$, i.e. $\mathbf{w}_3{}^{u^*}=\begin{bmatrix}0.5 & 0 & 1\end{bmatrix}^T$.

3.6.2 Example 2

In this example we present the results obtained for a system which has stronger nonlinearities and quartic cost. We consider the nonlinear system given by the equations

$$\begin{cases}\dot{x}_1=-x_1+x_2+2x_2^3\\ \dot{x}_2=f(x)+g(x)u\end{cases} \tag{3.31}$$

with $f(x)=-\frac{1}{2}(x_1+x_2)+\frac{1}{2}x_2(1+2x_2^2)\sin^2(x_1),\ g(x)=\sin(x_1)$. If we define $Q(x)=x_1^2+x_2^2+2x_2^4$ the infinite horizon cost function to be minimized then the optimal cost function for this system is $V^*(x)=\frac{1}{2}x_1^2+x_2^2+x_2^4$ and the optimal controller is $u^*(x)=-\sin(x_1)(x_2+2x_2^3)$.

The simulation was conducted using data obtained from the system at every 0.1s. For each iteration we considered data measured along five trajectories defined by five different initial conditions chosen randomly in $\Omega=\{-1\le x_i\le 1; i=1,2\}$. The initial stabilizing controller was taken as $\mu^{(0)}(x)=-\frac{1}{2}\sin(x_1)(3x_2-0.2x_1^2x_2+12x_2^3)$. The cost function $V^{\mu^{(i)}}(x)$ was approximated on Ω as $V_L{}^{\mu^{(i)}}(x)=(\mathbf{w}_L{}^{\mu^{(i)}})^T\boldsymbol{\varphi}_L(x)$ with $L=8$, $\mathbf{w}_8{}^{\mu^{(i)}}=\begin{bmatrix} w_1^{\mu^{(i)}} & \dots & w_8^{\mu^{(i)}}\end{bmatrix}^T$ and

$\boldsymbol{\varphi}_8(x)=\begin{bmatrix} x_1{}^2 & x_1x_2 & x_2{}^2 & x_1{}^4 & x_1{}^3x_2 & x_1{}^2x_2{}^2 & x_1x_2{}^3 & x_2{}^4\end{bmatrix}^T$.

At each iteration step we solved for $\mathbf{w}_8{}^{\mu^{(i)}}$ using 40 data points, i.e. 8 points measured on each of the 5 trajectories in Ω. Each data point consists of the measured the cost function associated with the present control policy, over a time interval T=0.1s, and the system state at both ends of this interval. In this way, at every 4s, the cost function was solved for and a policy update was performed. One notes that each data point set measured on each trajectory is sufficient to identify the parameters of the cost function corresponding to that given trajectory. However it is often not the case that

cost function parameters associated with one trajectory are equal to the cost function parameters associated with another trajectory. The result of applying the algorithm is presented in Figure 12.

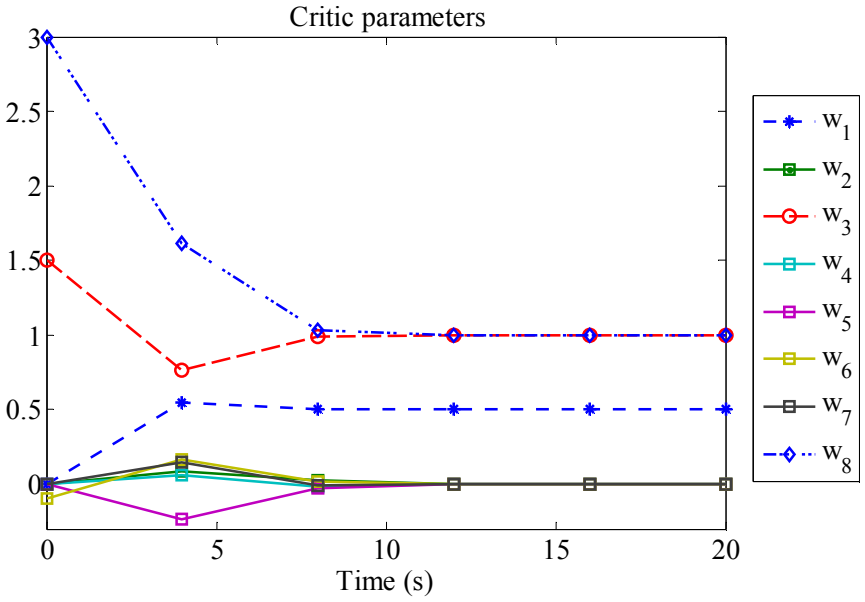

Figure 12. Convergence of the critic parameters

The figure clearly shows that the parameters of the critic neural network converged to the coefficients of the optimal cost function $V^*(x)=\frac{1}{2}x_1^2+x_2^2+x_2^4$, i.e. $\mathbf{w}_8{}^{u^*}=\begin{bmatrix}0.5 & 0 & 1 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^T$. One observes that after 3 iteration steps the parameters of the controller, obtained based on the update equation (3.22), are very close to the parameters of the optimal controller $u^*(x)=-\sin(x_1)(x_2+2x_2^3)$.

3.7 Conclusion

In this chapter was presented a continuous-time adaptive controller, based on Policy Iteration, which adapts online to learn the continuous-time optimal control policy

without using knowledge about the internal dynamics of the nonlinear system. Convergence of the proposed algorithm, under the condition of initial stabilizing controller, to the solution of the optimal control problem has been established. Proof of convergence for the online version of the algorithm, while taking into account the approximation error, was also provided. The simulation results support the effectiveness of the online adaptive optimal controller.

CHAPTER 4

GENERALIZED POLICY ITERATION FOR CONTINUOUS-TIME SYSTEMS

In this chapter is introduced, in a continuous-time framework, a class of ADP algorithms which, in the spirit of [56], will be named *generalized policy iteration* (GPI). The new class of algorithms is developed here for affine in the inputs nonlinear systems. The basis of the development is a new, partially model free (i.e. the internal dynamics of the nonlinear system need not be known), formulation for the *policy iteration* (PI) approach to optimal control. The new formulation of the PI algorithm allows formulation of the GPI and shows that it represents a spectrum of iterative algorithms which in effect includes at one end the PI algorithm and at the other the *value iteration* (VI) algorithm.

The first section of this chapter reviews the standard PI approach to the solution of the infinite horizon optimal control problem for nonlinear systems, i.e. the optimal control problem discussed in Section 3.2. Section 4.2 presents the main result: a new formulation for the PI algorithm, with convergence proof, followed by the general description of the GPI class of algorithms. Section 4.3 briefly discusses the implementation aspects of the GPI algorithms using function approximators in an actor-critic structure while Section 4.4 presents simulation results obtained first for a LQR problem and second for the case of a nonlinear system.

4.1 Policy Iteration Algorithm

4.1.1 CT PI Algorithm 1: Standard PI

The standard Policy Iteration algorithm for CT systems is described as

1. Select $u_0 \in \Psi(\Omega)$

2. (policy evaluation step) Solve for V^i

$$H(x, u_{i-1}, \nabla V_x^i) = 0 . \tag{4.1}$$

3. (policy improvement step) Find u_i which satisfies

$$u_i = \arg\min_{v \in \Psi(\Omega)} [H(x, v, \nabla V_x^i)] . \tag{4.2}$$

Conditioned by a suitable initialization, i.e. admissible initial policy, PI provides solution of the optimal control problem based on recursively solving equations (4.1) and (4.2) as the index $i \to \infty$. The solution V^i of (4.1) represents the value function associated with using the control policy u_{i-1}. In order to obtain the solution of (4.1), which can be explicitly written as

$$r(x(t), u_{i-1}(t)) + (\nabla V_x^i)^T (f(x(t)) + g(x(t)) u_{i-1}(t)) = 0 , \tag{4.3}$$

exact knowledge on the system dynamics, i.e. $f(x(t)), g(x(t))$ is required. Knowledge on the system's input-to-state dynamics $g(x)$ is always required as it is part of the closed form solution of (4.2) which is

$$u_i(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V_x^{\,i} . \tag{4.4}$$

4.1.2 CT PI Algorithm 2: Based on the integral over a time interval

In order to avoid the necessity of knowing the internal dynamics of the system, $f(x)$, and to allow online implementation, in Chapter 3 has been developed an equivalent formulation of the PI algorithm as

1. Select $u_0 \in \Psi(\Omega)$

2. (policy evaluation step) Solve for V^i

$$\int_{t}^{t+T_0} r(x,u_{i-1})d\tau + V^i(x_{t+T_0}) - V^i(x_t) = 0, \quad V^i(0) = 0. \tag{4.5}$$

3. (policy improvement step) Find u_i which satisfies

$$u_i = \arg\min_{v \in \Psi(\Omega)} [H(x, v, \nabla V_x^i)]. \tag{4.6}$$

In (4.5), x_t and x_{t+T_0} are short notations for $x(t)$ and $x(t+T_0)$, where $x(\tau)$ is the solution of (2.1) for initial condition $x(t)$ and input $\{u_{i-1}(\tau); \tau \geq t\}$.

It was proved in Chapter 3 that the two equations (4.1) and (4.5), corresponding to the policy evaluation step, have the same solution. The advantage of using (4.5) stands in the fact that this equation can be solved based on online measurements, without any requirement of knowing the system internal dynamics. Online approaches to policy iterations make use of function approximators as support for the solutions of (4.5) and (4.6) in an actor-critic type of structure. Implementation details will be discussed in Section 4.4.

4.2 Generalized Policy Iteration

In this section are formulated the *generalized policy iteration* (GPI) algorithms for continuous-time systems. In the first subsection are introduced the mathematical tools which will provide basis for the PI and GPI algorithm formulation which are given in Subsections 4.2.2 and 4.2.3.

4.2.1 Preliminaries

Let X denote the space of bounded functionals $V(.):\Omega\to\mathbb{R}$ with $V(x_t)>0,\forall x_t\in\Omega,\ \ V(0)=0$. X is a Banach space with the norm $\|V\|=\sup_{x\in\Omega}|V(x)|$.

Define the dynamic programming operator $T_\mu:\mathrm{X}\to\mathrm{X}$

$$T_\mu V(x_t)=\int_t^{t+T_0} r(x,\mu)d\tau+V(x_{t+T_0}), \tag{4.7}$$

where x_{t+T_0} is the value of $x(\tau)$ at $\tau=t+T_0$, with $x(\tau)$ the solution of (2.1) for initial condition $x(t)$ (denoted x_t) and input $\{\mu(\tau);\tau\geq t\}$.

Also, define the operator $T:\mathrm{X}\to\mathrm{X}$

$$TV(x_t)=\min_{u\in\Psi(\Omega)}\left\{\int_t^{t+T_0} r(x,u)d\tau+V(x_{t+T_0})\right\}. \tag{4.8}$$

The first operator, $T_\mu:\mathrm{X}\to\mathrm{X}$, maps the cost functional $V(.)\in\mathrm{X}$ into the cost functional denoted $T_\mu V(.)\in\mathrm{X}$, while using the control policy $\mu\in\Psi(\Omega)$ over the time interval $[t,t+T_0]$. The sample period T_0 must be chosen such that $\forall x_t\in\Omega$ the solution

of (2.1) at time $t+T_0$, using the control policy μ, satisfies $x_{t+T_0} \in \Omega_1 \subseteq \Omega$. Note that if $\mu \in \Psi(\Omega)$ there exists a lower bound T_l such that $\forall T_0 \geq T_l$ and $\forall x_t \in \Omega$ then $x_{t+T_0} \in \Omega$.

The formulation of the operator T_μ when the state feedback optimal control problem for linear systems with quadratic performance index, i.e. LQR, is considered is given now. Using the parametric description of the value function $V(x) = x^T Px, \forall x \in \mathbb{R}^n$ and $T_\mu V(x) = x^T P_1^\mu x, \forall x \in \mathbb{R}^n$, and the control policy $\mu(x) = -K^\mu x$, this operator can be written as

$$x_t^T P_1^\mu x_t = \int_t^{t+T_0} x^T(\tau)[Q + (K^\mu)^T RK^\mu]x(\tau)d\tau + \\ + x_t^T e^{(A-BK^\mu)^T T_0} Pe^{(A-BK^\mu)T_0} x_t \quad . \tag{4.9}$$

Let X^P denote the set of all positive definite matrices which can serve as parametric representations of quadratic value functions. Denoting with $A_d^{T_0} \triangleq e^{(A-BK^\mu)T_0}$ the discrete version of the continuous-time dynamics of the linear system, when a sampling period of T_0 was used, and writing the integral term as

$$x_t^T M^\mu x_t \equiv \int_t^{t+T_0} x^T(\tau)[Q + (K^\mu)^T RK^\mu]x(\tau)d\tau \ ,$$

with $M^\mu > 0$, then it can be introduced the operator $T'_\mu : \mathrm{X}^P \to \mathrm{X}^P$ defined as

$$T'_\mu \ P = P_1^\mu = M^\mu + (A_d^{T_0})^T PA_d^{T_0} \ . \tag{4.10}$$

P_k^{μ} denotes the composition of k copies of T'_{μ} applied on the parametric representation of the quadratic cost function $V(x)=x^T Px,\ x\in\mathbb{R}^n$, i.e. matrix P.

The operator defined by (4.8), $T:\mathrm{X}\to\mathrm{X}$, maps the cost functional $V(.)\in\mathrm{X}$ into the cost functional denoted $TV(.)\in\mathrm{X}$, while using over the time interval $[t,t+T_0]$ the control policy u which is solution to the finite horizon optimal control problem defined by the right hand side of (4.8).

It is important to see that the control solution of (4.8) is not the same as $u=\arg\min_{v\in\Psi(\Omega)}[H(x,v,\nabla V_x)]$; in the first case a time varying control policy is obtained while in the latter case the resulting policy is time invariant. For example, in a linear system case with quadratic cost function the control solution given by (4.8) is $u(\tau,x)=-R^{-1}B^T P(\tau)x$ where $P(\tau)$ is the solution of the differential Riccati equation over the time interval $[t,t+T_0]$ with final condition $P_{t+T_0}=P$. On the other hand, the solution obtained using $u=\arg\min_{v\in\Psi(\Omega)}[H(x,v,\nabla V_x)]$ is $u(x)=-R^{-1}B^T Px$. Thus, one can see that using (4.8) as basis for the policy improvement step would lead to a significant modification of the policy iteration algorithm.

Using the introduced operators we can also write

$$TV(x_t)=\min_{u\in\Psi(\Omega)}\left\{T_u V(x_t)\right\}. \tag{4.11}$$

Also, Bellman's optimality principle can be formulated as

$$TV^*(x_t)=\min_{u\in\Psi(\Omega)}\left\{T_u V^*(x_t)\right\}=V^*(x_t). \tag{4.12}$$

In the following T^k and T_u^k will denote the composition of k copies of T and T_u.

4.2.2 A new CT formulation of policy iteration

Two equivalent formulations of continuous-time PI were given as Algorithm 1, equations (4.1), (4.2), and Algorithm 2, equations (4.5), (4.6). In this section are presented results which allow a third formulation of the policy iteration algorithm based on the functional mapping operators which were introduced in the previous section. The following results are required.

Lemma 4.1 *Let* $\mu \in \Psi(\Omega)$. *Then* $V^\mu \in \mathrm{X}$ *is a fixed point of the mapping* $T_\mu : \mathrm{X} \to \mathrm{X}$.

Proof Let V^μ denote the cost associated with the policy μ. Then, using the definition in (4.7), we have

$$T_\mu V^\mu(x_t) = \int_t^{t+T_0} r(x,\mu)d\tau + V^\mu(x_{t+T_0}) \tag{4.13}$$

which is

$$T_\mu V^\mu(x_t) = V^\mu(x_t) \tag{4.14}$$

thus $V^\mu(x_t)$ is a fixed point of the mapping T_μ. □

Note that the equation

$$T_\mu V(x_t) = \int_t^{t+T_0} r(x,\mu)d\tau + V(x_{t+T_0}) = V(x_t) \tag{4.15}$$

has a unique solution denoted by $V^\mu(x_t)$, which is also the solution of $H(x,\mu,\nabla V_x) = 0$.

Lemma 4.2 *Let* $\mu \in \Psi(\Omega)$. *Then* $T_\mu : \mathrm{X} \to \mathrm{X}$ *is a contraction mapping on* X.

Proof Let $V, W \in \mathrm{X}$, then

$$T_\mu V(x_t) = \int_t^{t+T_0} r(x,\mu)d\tau + V(x_{t+T_0}) \tag{4.16}$$

$$T_\mu W(x_t) = \int_t^{t+T_0} r(x,\mu)d\tau + W(x_{t+T_0}). \tag{4.17}$$

Subtracting the two equations one gets

$$T_\mu V(x_t) - T_\mu W(x_t) = V(x_{t+T_0}) - W(x_{t+T_0}). \tag{4.18}$$

T_0 is chosen such that $\forall x_t \in \Omega$ and $x_{t+T_0} \in \Omega_1 \subseteq \Omega$.

Then

$$\sup_{\Omega_1}(|V-W|)(x) \le \sup_{\Omega}(|V-W|)(x) \tag{4.19}$$

which together with (26) gives

$$\sup_{\Omega}(|T_\mu V - T_\mu W|)(x) \le \sup_{\Omega}(|V-W|)(x).$$

This is,

$$\|T_\mu V - T_\mu W\| \le \|V-W\|. \tag{4.20}$$

This proves the lemma. □

Subtracting (4.13) from (4.16), and making use of (4.14), one obtains

$$T_\mu V(x_t) - V^\mu(x_t) = V(x_{t+T_0}) - V^\mu(x_{t+T_0})$$

which has as result

$$\left\| T_\mu V - V^\mu \right\| \le \left\| V - V^\mu \right\|, \ \forall V \in \mathrm{X}. \tag{4.21}$$

The next corollary of Lemma 4.2 considers the formulation of the infinite horizon optimal control problem for linear systems with quadratic performance index, i.e. the LQR problem. In this case it is known that the value function associated with a given admissible state feedback policy can be exactly represented by the parametric description $V(x) = x^T P x, \forall x \in \mathbb{R}^n$. The operator T'_μ defined by equation (4.10) is now used. X^P, equipped with the spectral radius matrix norm defined as $\rho(A) \equiv \|A\|_\rho \triangleq \max_i(|\lambda_i|)$, where λ_i the eigenvalues of A, is a Banach space.

Corollary 4.1 *$T'_\mu : \mathrm{X}^P \to \mathrm{X}^P$ is a contraction map on X^P.*

The proof is given in Appendix A.

Lemma 4.3 *The mapping $T_\mu : \mathrm{X} \to \mathrm{X}$ has a unique fixed point on X which is can be obtained using*

$$V^\mu(x_t) = \lim_{k \to \infty} T_\mu^k V(x_t) \ \ \forall V(x_t) \in \mathrm{X}. \tag{4.22}$$

Proof Using the Banach fixed point theorem, since $T_\mu : \mathrm{X} \to \mathrm{X}$ is a contraction on X (Lemma 4.2) then its fixed point $V^\mu \in \mathrm{X}$ (Lemma 4.1) is the unique fixed point. Moreover, the unique fixed point can be determined as the limit of the iterative sequence defined by $V_k^\mu(.) = T_\mu V_{k-1}^\mu(.) = T_\mu^k V_0^\mu(.), \ k \ge 1, \ V_0^\mu(.) = V(.) \in \mathrm{X}$, i.e. $V^\mu(x_t) = \lim_{k \to \infty} T_\mu^k V(x_t) \ \ \forall V(x_t) \in \mathrm{X}$. □

4.2.3 Continuous-time PI Algorithm 3: Iterative solution of the policy evaluation step

Using the result in Lemma 4.3 now is given a third formulation for the policy iteration algorithm which makes use of the operator defined by (4.7). Thus

1. Select $u_0 \in \Psi(\Omega)$

2. (policy evaluation step) Solve for $V^{u_{i-1}}(x_t)$ (denoted with $V^i(x_t)$) using the iteration

$$V^i(x_t) = \lim_{k\to\infty} T_{u_{i-1}}^k V(x_t) \tag{4.23}$$

starting with any $V(x_t) \in \mathrm{X}$, and T_μ defined by (4.7).

3. (policy improvement step) Find u_i which satisfies

$$u_i = \arg\min_{v\in\Psi(\Omega)} [H(x, v, \nabla V_x^i)]. \tag{4.24}$$

A variant of the above policy iteration algorithm is obtained when one starts the policy evaluation step for $V^{u_{i-1}}(x_t) = V^i(x_t)$ with the cost functional obtained at the previous step $V^{u_{i-2}}(x_t) = V^{i-1}(x_t)$, i.e. (4.23) becomes

$$V^i(x_t) = \lim_{k\to\infty} T_{u_{i-1}}^k V^{i-1}(x_t). \tag{4.25}$$

4.2.4 Generalized policy iteration – a continuous-time formulation

The formulation of the *generalized policy iteration* (GPI) algorithm for continuous-time systems with continuous state and action space is now given.

<u>GPI for CT systems</u>

1. Select $u_0 \in \Psi(\Omega)$

2. (approximate policy evaluation step) Approximately solve for $V^i(x_t)$ using the iteration

$$V^i(x_t) \triangleq V_k^i(x_t) = T_{u_{i-1}}^k V(x_t) \tag{4.26}$$

starting with any $V(x_t) \in \mathrm{X}$, for some $k \geq 1$.

Note that this step can also be replaced with

$$V^i(x_t) \triangleq V_k^i(x_t) = T_{u_{i-1}}^k V^{i-1}(x_t). \tag{4.27}$$

3. (policy improvement step) Find u_i which satisfies

$$u_i = \arg\min_{v \in \Psi(\Omega)} [H(x, v, \nabla V_x^i)]. \tag{4.28}$$

One clearly sees that when $k \to \infty$ in (4.22) we encounter the regular PI algorithm with the policy evaluation step given by (4.23).

For the case of $\infty > k \geq 1$, one obtains the so called *optimistic policy iteration* [56], with the policy evaluation step given by

$$V_o^i(x_t) = T_{u_{i-1}}^k V(x_t). \tag{4.29}$$

In this "optimistic" case the policy update step is executed prior to the convergence to the true value associated with the current control policy. In (4.29) the notation $V_o^i(.)$ was used to make the point that the value resulting from (4.26) is not the true value associated with the current control policy $u_{i-1}(x)$, i.e. $V^i(.)$.

<u>CT value iteration variant with initial stabilizing policy</u>

When $k = 1$, the GPI becomes a variant of the value iteration algorithm given next.

1. Select $u_0 \in \Psi(\Omega)$

2. (Value function update step) Solve for $V^i(x_t)$ using only one value update step

$$V^i(x_t) = T_{u_{i-1}} V^{i-1}(x_t). \tag{4.30}$$

3. (policy update step) Find u_i which satisfies

$$u_i = \arg\min_{v \in \Psi(\Omega)} [H(x, v, \nabla V_x^i)]. \tag{4.31}$$

The key difference between this GPI with $k=1$ and the value iteration algorithm is that in the latter case the requirement of $u_0 \in \Psi(\Omega)$ is removed, i.e. the initial policy in VI need not be stabilizing.

The flowchart of the GPI algorithm is presented in Figure 13.

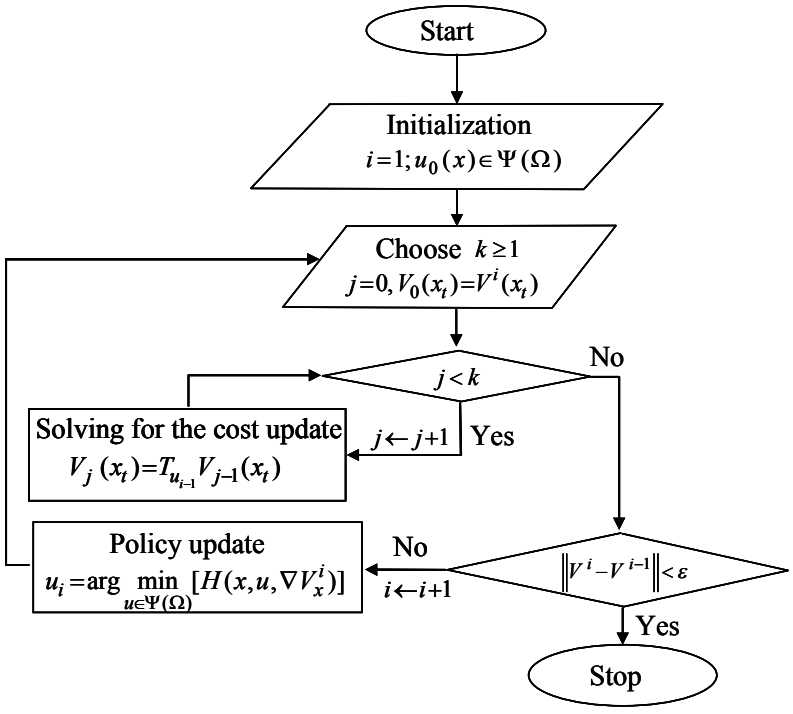

Figure 13. Flow chart of the generalized policy iteration (GPI) algorithm

4.3 Online implementation of generalized policy iteration

The online learning GPI algorithm is implemented on an actor-critic structure. In a general case the two structures can be neural networks (NN) which are universal approximators, [26].

For value function approximation the cost $V^i(x) \in \mathrm{X}$ will be represented as

$$V^i(x) = \sum_{j=1}^{L} w_j{}^i \phi_j(x) = (\mathbf{w}_L{}^i)^T \boldsymbol{\varphi}_L(x). \tag{4.32}$$

This could be seen as a neural network with L neurons on the hidden layer and activation functions $\phi_j(x) \in C^1(\Omega), \phi_j(0) = 0$. $\boldsymbol{\varphi}_L(x)$ denotes the vector of activation functions and $\mathbf{w}_L{}^i$ the vector of the parameters of the neural network, with $w_j{}^i$ the weights of the neural network. The activation functions should be selected to provide a complete basis for the space of value functions over Ω.

In order to solve for the cost function $V^i(x)$ in equation (4.27), the j-th step of the value function update, which is

$$V_j^i(x_t) = T_{u_{i-1}} V_{j-1}^{i-1}(x_t), V_0^i = V^{i-1}, 1 \le j \le k, \tag{4.33}$$

can be written as

$$(\mathbf{w}_L{}^j)^T \boldsymbol{\varphi}_L(x_t) = \int_t^{t+T_0} r(x, u_i(x)) d\tau + (\mathbf{w}_L{}^{j-1})^T \boldsymbol{\varphi}_L(x_{t+T}) \tag{4.34}$$

with $V^i(x_t)=(\mathbf{w}_L{}^k)^T\boldsymbol{\varphi}_L(x_t)$. The parameters of the value function approximation will be tuned, at each iterative step (4.33) of (4.27), to minimize, in the least-squares sense, the objective

$$S=\int_{\Omega^{u_i}_{\{x_0\}_n}}\delta_L^j(x)\delta_L^j(x)dx\ . \tag{4.35}$$

In (4.35) $\Omega^{u_i}_{\{x_0\}_n}$ denotes a set of trajectories generated by the policy u_i, from the initial conditions $\{x_0\}_n\subset\Omega$, and

$$\delta_L^j(x_t)=\int_t^{t+T_0} r(x,u_i(x))d\tau+(\mathbf{w}_L^{j-1})^T\boldsymbol{\varphi}_L(x_{t+T})-(\mathbf{w}_L^j)^T\boldsymbol{\varphi}_L(x_t).$$

The quantity $\delta_L^j(x_t)$ can be viewed as the *temporal difference* residual error.

Using the inner product notation for the Lebesgue integral, the least squares solution of (4.34) is

$$\mathbf{w}_L{}^j=-\Phi^{-1}\left\langle\boldsymbol{\varphi}_L(x_t),\int_t^{t+T_0} r(x,u_i(x)d\tau+(\mathbf{w}_L^{j-1})^T\boldsymbol{\varphi}_L(x_{t+T})\right\rangle_{\Omega^{u_i}_{\{x_0\}_n}} \tag{4.36}$$

where $\Phi=\left\langle\boldsymbol{\varphi}_L(x_t),\boldsymbol{\varphi}_L(x_t)^T\right\rangle_{\Omega^{u_i}_{\{x_0\}_n}}$.

After updating the value function to solve equation (4.34) *k* times, i.e. once the approximate solution of (4.27) had been obtained, the policy update step, given by (4.28), is

$$u_{i+1}(x)=-R^{-1}g^T(x)\left(\frac{\partial\varphi_L(x)}{\partial x}\right)^T\mathbf{w}_L{}^i\ . \tag{4.37}$$

Note that in this implementation, after the policy update step is executed, the parameters of the two approximation structures, namely actor and critic, are the same.

Finally, it is noted that all ADP algorithms are developed on the assumption that correct estimation of the value function is possible. This means that, in order to successfully apply the online learning algorithm, enough excitation must be present in the system to guarantee correct estimation of the value function at each value update step. In relation with this, one must also note that the greedy policies obtained at the policy update steps are admissible policies and thus not excitatory. This is the well known exploration/exploitation dilemma, [56], which characterizes adaptive controllers that have simultaneous conflicting goals such as optimal control and fast and effective adaptation/learning.

4.4 Simulation Examples

4.4.1 Example 1 - a linear system

In this section are presented comparative simulation results using the GPI approach to solving the LQR problem considering the linear model of the F16 short period dynamics given in [53].

The description of the linear system is given by matrices

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.1755 \\ 0 & 0 & -20.2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 20.2 \end{bmatrix}.$$

The infinite horizon quadratic cost function to be minimized is characterized by the identity matrices Q and R of appropriate dimensions. The optimal value function obtained by solving the ARE is

$$P = \begin{bmatrix} 1.4116 & 1.1539 & -0.0072 \\ 1.1539 & 1.4191 & -0.0087 \\ -0.0072 & -0.0087 & 0.0206 \end{bmatrix}.$$

Figure 14 presents a comparative view of the results obtained with the various GPI algorithms (for different values of the parameter k) in terms of the norm of the cost function matrix P, considering the F-16 system.

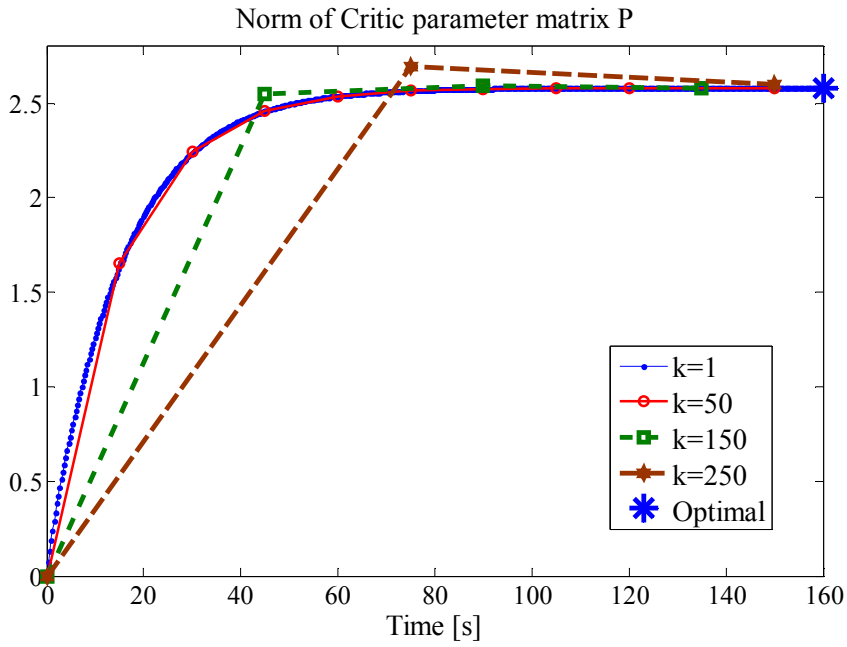

Figure 14. Comparative view of the results obtained while using the GPI algorithm for different values of the parameter k in terms of the norm of the critic parameters given by matrix P; the relevant values are indicated by the marker points while the connecting lines are only intended to provide ease of visualization

As the system is stable, the GPI algorithms (applied for different values of k) were initialized using the state-feedback controller $K_0 = 0$. The simulation was conducted using data obtained from the system at every 0.05s. In this way, at each 0.3s, enough data is collected from the system to solve for the value of the matrix P and perform a

value function update, as there are six independent elements in the symmetric matrix P which parameterizes the value function associated with any admissible state feedback controller. After a number of k updates the solution given by (4.27) is used to perform a policy update step.

One can see from Figure 14 that the number of iterative steps (i.e. value function and policy update steps) required for the GPI algorithm to converge is inversely proportional to the number of iterative updates used to solve the value function update step (i.e. parameter k of the GPI algorithm).

4.4.2 Example 2 - a nonlinear system

We now consider the nonlinear system described by the equation

$$\begin{cases} \dot{x}_1 = -x_1 + x_2 \\ \dot{x}_2 = -\frac{1}{2}x_1 - \frac{1}{2}x_2((1-(\cos(2x_1)+2)^2)) + (\cos(2x_1)+2))u \end{cases} \tag{4.38}$$

This system was designed, using the converse HJB approach, such that, when the cost function to be minimized is described by $r(x,u)=x^TQx+u^TRu$, with Q, R identity matrices of appropriate dimensions, the optimal cost function is $V^*(x) = \frac{1}{2}x_1{}^2 + x_2^2$.

The critic is given by the equation

$$V_{W_1}(x) = [w_1 \quad w_2 \quad w_3][x_1^2 \quad x_1x_2 \quad x_2^2]^T + \varepsilon(x) \tag{4.39}$$

The next three figures show the convergence of the parameters of the critic when the sequential GPI algorithm was used. The number of iterative steps to solve for the value function using iteration (4.27) were $K=1$ (i.e. HDP algorithm), $K=5$ and $K=50$.

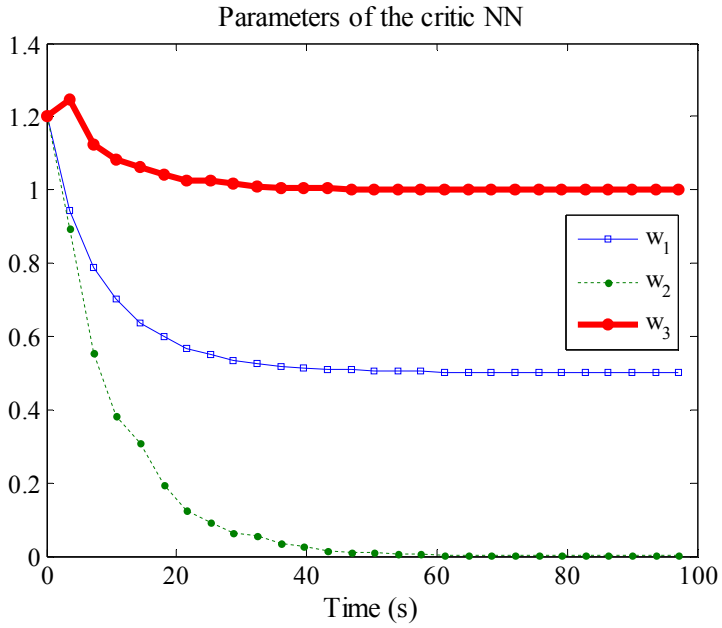

Figure 15. Convergence of the critic parameters to the optimal values using sequential GPI with K=1

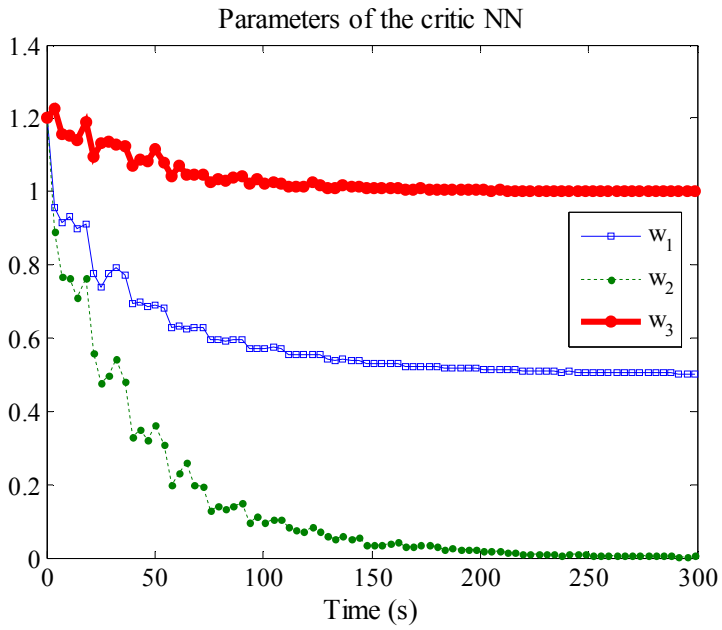

Figure 16. Convergence of the critic parameters to the optimal values using sequential GPI with K=5

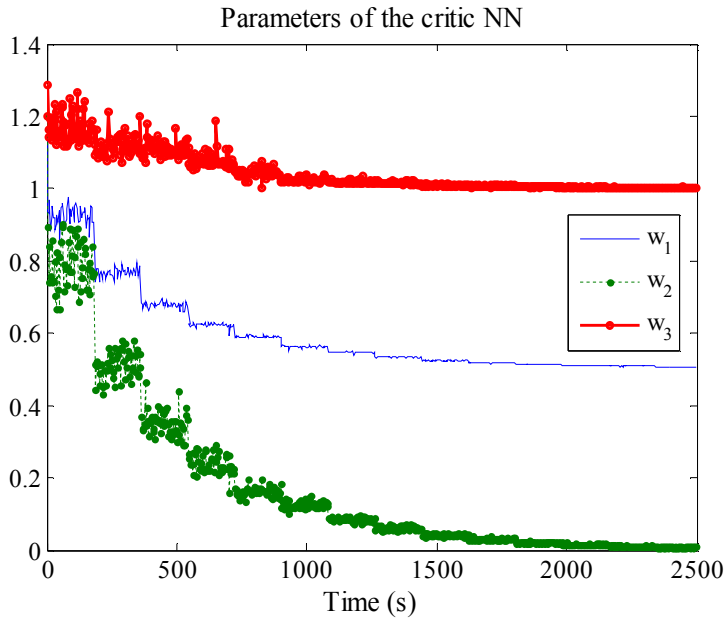

Figure 17. Convergence of the critic parameters to the optimal values using sequential GPI with *K*=50

All the results were obtained while measuring data from the system using a sampling time interval $T_0 = 0.1\text{sec}$. At each iterative step, (4.34) was solved in the least squares sense using 36 discrete-time measurements from the system along 6 different , randomly chosen, trajectories in

$\Omega = \{(x_1, x_2); -1 \le x_1 \le 1, -1 \le x_2 \le 1\}$.

The number of points and of trajectories used to solve (4.34) are a matter of fine tuning the algorithm and depends on the experience of the engineer relative to the system dynamics, in a similar manner with the choice of the sample time T_0.

Comparing the results in Figures 15-17, one observes that increasing the number of steps in the iterative algorithm used at the critic training phase leads to an increase in the time until the critic converges to the optimal critic. Nonetheless, in all instances the

sequential GPI algorithm leads to convergence to the optimal cost and optimal control policy.

4.5 Conclusions

In this chapter was given the formulation of generalized policy iteration algorithms in a continuous-time framework. It was argued that GPI is in fact a spectrum of iterative algorithms which has at one end the policy iteration algorithm and at the other a variant of the value iteration algorithm. The algorithms can be implemented in a partially model free setup using function approximators on an actor-critic structure.

The new PI algorithm (GPI with $k \to \infty$), which is known to converge to the optimal control solution, could prove useful in relation to showing the convergence of the continuous-time value iteration algorithm (i.e. GPI with $k = 1$). At the same time the GPI formulation could lead to a continuous-time formulation of the Q-function. This is an important step which would have as result a set of model-free direct adaptive optimal control algorithms for continuous-time systems.

CHAPTER 5

ADAPTIVE OPTIMAL CONTROL BASED ON HDP FOR CONTINUOUS-TIME LINEAR SYSTEMS

The investigation of the ADP method for the case of linear systems is relevant for real world control applications. Even though generally the controlled system has nonlinear dynamics, it is often required to be controlled for best performance around a specific operating point where a linear model regularly offers a good approximate description of the system. It is also the case that at such point the exact values of the model parameters (i.e. matrix A) are not easy to find, requiring a tedious identification procedure, and also these values can drift over time. Systems that fit this description could be chemical plants, airplanes and power systems.

In this chapter is formulated the *heuristic dynamic programming* (HDP) algorithm which provides solution to the continuous-time LQR problem with infinite horizon cost index. The formulation for the particular LQR problem allows easy comparison between the policy iteration algorithm (i.e. Newton's method) and the HDP (value iteration) algorithm, while providing some insight relative to the convergence of the algorithm.

5.1 Introduction

Approximate dynamic programming is a combination between *reinforcement learning* designs and *dynamic programming* that determines an approximate solution

for the optimal control problem using a forward-in-time computation based on real-time data. Reinforcement learning structures, namely *adaptive critics*, were first proposed by Werbos [62]. The ADP methods are iterative procedures of updating the control policy and value function estimate in order to bring them closer to the optimal control policy and the corresponding optimal value function. Each iteration step consists of an update of the value function estimate based on the current control policy, followed by a greedy update of the control policy based on the new value function estimation.

Initially developed for systems with finite state and action spaces the ADP methods were based on Werbos' *heuristic dynamic programming* (HDP) [62], Sutton's temporal difference method [55], Watkins's Q-learning [61]. For the case of discrete-time systems with continuous state and action spaces different adaptive critic architectures were reported, with successful implementations and rigorous proofs; an incomplete list being [35], [42], [20], [45].

As the ADP techniques have been introduced and developed in the computational intelligence community, most results have been focused on the control of discrete-time systems. The equivalent ADP formulation, for systems with continuous state spaces and continuous-time dynamics, can not be obtained in a straight forward manner as an extension of the existent discrete-time techniques. For continuous-time systems there are difficult issues related to sampling times, and requirements for knowing the system dynamical equations. It is known, for instance, that as the sampling time becomes small, discrete-time ADP does not provide the optimal control solution for continuous-time systems [5]. Also, unlike the discrete-time Hamiltonian, central in the discrete-time

ADP techniques, which does not involve the dynamics of the discrete-time system, the Hamiltonian for continuous-time systems immediately involves the system dynamics, [40], which must therefore be known. This is making much more difficult the formulation of a mathematical approach which will not require the system model knowledge for continuous-time systems.

For continuous-time systems a dynamic programming-based reinforcement learning scheme, formulated using a so-called *advantage function*, was introduced in [5]. Reinforcement learning techniques based on the temporal difference method were proposed in [17].

In this chapter, bringing together concepts from ADP and control systems theory, and based on the formulation of the GPI algorithm introduced in Chapter 4, we formulate and analyze an ADP technique which offers solution, obtained online in a forward-in-time fashion, to the continuous-time infinite horizon optimal control problem for linear systems. The online learning method makes use only of partial knowledge on the system dynamics (i.e. the drift dynamics, specified by the system matrix A need not be known).

This technique is a continuous-time approach to HDP (CT-HDP) for linear systems. The adaptive critic is solving online for the continuous-time version of the optimal value function - an approach also known as *V-learning*. It will be shown that the proposed iterative ADP algorithm is in fact a quasi-Newton method to solve the algebraic Riccati equation (ARE) underlying the optimal control problem. Unlike in the case of the policy iteration algorithm, this time an initial gain that determines a

stabilizing control policy is no longer required. It is thus obtained a online direct adaptive control algorithm which determines the *optimal* control solution *without knowing the system A matrix.*

The CT-HDP algorithm is presented next. A mathematical formulation of the algorithm is given proving the equivalence of the ADP iteration with a Quasi-Newton method. The algorithm is then tested in simulation and the optimal controller for a linear power system model is obtained without making use of any knowledge regarding the system matrix A.

5.2 Continuous-time Heuristic Dynamic Programming for the LQR problem

We consider the linear quadratic regulation problem described in Section 2.2. As discussed in Chapter 2, the LQR problem can be solved using online policy iteration. It is an online version of the underlying Newton method, which requires that the iteration is initialized by a stable state-feedback control policy [32]. The resulting iterative algorithm is:

Given K_0 such that $A_0 = A - BK_0$ is Hurwitz, a sequence $\{P_i\}_{i\geq 1}$ can be determined by solving successively the Lyapunov equation

$$0 = A_i^T P_{i+1} + P_{i+1} A_i + Q + P_i B R^{-1} B^T P_i \tag{5.1}$$

where $A_i = A - BR^{-1}B^T P_i$. Kleinman showed that the sequence $\{P_i\}_{i\geq 1}$ is monotonically decreasing and lower bounded by the unique positive definite solution of the ARE.

Equation (5.1) can be expressed in a Newton's method-like setting as

$$P_{i+1} = P_i - (Ric'_{P_i})^{-1} Ric(P_i) \tag{5.2}$$

where

$$Ric(P_i) = A^T P_i + P_i A + Q - P_i B R^{-1} B^T P_i \tag{5.3}$$

and Ric'_{P_i} denotes the Frechet derivative of $Ric(P_i)$ taken with respect to P_i. The matrix function Ric'_{P_i} evaluated at a given matrix M will thus be $Ric'_{P_i}(M) = A_i^T M + M A_i$.

5.2.1 Continuous-time HDP formulation

Based on the results obtained in Chapter 4 we can formulate the *heuristic dynamic programming* approach to ADP for continuous-time systems.

Let $T_\mu : \mathrm{X} \to \mathrm{X}$ be the dynamic programming operator defined as

$$T_\mu V(x_t) = \int_t^{t+T_0} r(x, \mu) d\tau + V(x_{t+T_0}), \tag{5.4}$$

where x_{t+T_0} is the value of $x(\tau)$ at $\tau = t + T_0$, with $x(\tau)$ the solution of (2.1) for initial condition $x(t)$ (denoted x_t) and input $\{\mu(\tau); \tau \geq t\}$.

CT value iteration (heuristic dynamic programming)

1. Select $V^0(x_t) : \Omega \to \mathbb{R}; V^0(.) \subset \mathrm{X}$, $i = 0$

2. (policy update step) Find u_i which satisfies

$$u_i = \arg\min_{v \in \Psi(\Omega)} [H(x, v, \nabla V_x^i)]. \tag{5.5}$$

3. (Value function update step) Solve for $V^i(x_t)$ using

$$V^{i+1}(x_t) = T_{u_i} V^i(x_t). \tag{5.6}$$

The key difference between the GPI with $k=1$ and the Value Iteration algorithm is that in the latter case the requirement of $u_0 \in \Psi(\Omega)$ is removed, i.e. the initial policy need not be admissible.

For the case of the LQR problem the value iteration scheme is

1. Select $P_0 \geq 0$ such that $V_0(x_t):\mathbb{R}^n \to \mathbb{R}; V_0(x_t)=x_t{}^T P_0 x_t \geq 0$, $i=0$

2. (policy update step) Find u_i which satisfies

$$u_i = -R^{-1}B^T P_i x = K_i x \tag{5.7}$$

3. (Value function update step) Solve for $V_{i+1}(x_t)$ using

$$V_{i+1}(x(t)) = \int_t^{t+T_0} (x^T Q x + u_i{}^T R u_i) d\tau + V_i(x(t+T_0)) \tag{5.8}$$

with the *V*-function parameterized as $V_i(x) = x^T P_i x$, then make $i \to i+1$.

Equation (5.8) can be explicitly written in parametric form as

$$x^T(t) P_{i+1} x(t) = \int_t^{t+T_0} (x^T Q x + u_i{}^T R u_i) d\tau + x^T(t+T_0) P_i x(t+T_0) \tag{5.9}$$

and the ADP value function update amounts to the update of the kernel matrix P_i.

A restriction on the initial matrix P_0 such that the corresponding K_0 be a stabilizing controller is not required. In fact the algorithm can simply be initialized with $P_0 = 0$.

5.2.2 Online tuning based on V-learning algorithm for partially unknown systems

The algorithm can be implemented online without having any knowledge about the plant internal dynamics, i.e. the *A* matrix need not be known (matrix *B* is required), and

without starting with an initial stabilizing policy. The information on the A matrix of the system is embedded in the states $x(t)$ and $x(t+T_0)$ which are observed online.

To find the parameters of V_{i+1} in (5.8), the left-hand side of (5.9) is written as

$$V_{i+1}(\overline{x}(t), \overline{p}_{i+1}) = x^T(t) P_{i+1} x(t) = \overline{p}_{i+1}{}^T \overline{x}(t) \tag{5.10}$$

where $\overline{x}(t)$ denotes the Kronecker product quadratic polynomial basis vector with the elements $\{x_i(t)x_j(t)\}_{i=1,n;j=i,n}$ and $\overline{p} = v(P)$, where $v(.)$ is a vector valued matrix function that acts on $n \times n$ matrices and gives a column vector by stacking the elements of the symmetric matrix into a vector with the off-diagonal elements summed as $P_{ij} + P_{ji}$, (Brewer, 1978).The right-hand side of (5.8), using (5.7), is

$$d(x(t), P_i) = \int_t^{t+T_0} x(\tau)^T (Q + P_i B R^{-1} B^T P_i) x(\tau) d\tau + \overline{p}_i{}^T \overline{x}(t+T_0) . \tag{5.11}$$

Denoting with

$$r(x(t), P_i, T_0) = \int_t^{t+T_0} x^T(\tau)(Q + P_i B R^{-1} B^T P_i) x(\tau) d\tau . \tag{5.12}$$

the observed reward over the sample time interval $[t, t+T_0]$ and equating (5.10) and (5.11), then the iteration (5.8) and (5.7) can be written as

$$\overline{p}_{i+1}{}^T \overline{x}(t) = r(x(t), P_i, T_0) + \overline{p}_i{}^T \overline{x}(t+T_0) . \tag{5.13}$$

At each iteration step, after a sufficient number of state-trajectory points are collected using the same control policy K_i, a least-squares method is employed to solve for the V-function parameters, $\overline{p}_{i+1}$, which will then yield P_{i+1}. The parameter vector $\overline{p}_{i+1}$ is found by minimizing, in the least-squares sense, the error between the target function

given by (5.11) and the parameterized relation (5.10) over a compact set $\Omega \subset R^n$. Evaluating (5.13) at $N \geq n(n+1)/2$ points $\bar{x}^i$ in the data space, the least-squares solution is obtained as

$$\bar{p}_{i+1} = (XX^T)^{-1} XY \tag{5.14}$$

where $X = [\bar{x}^1 \quad \bar{x}^2 \quad ... \quad \bar{x}^N]$ and $Y = [d(\bar{x}^1, P_i) \quad d(\bar{x}^2, P_i) \quad ... \quad d(\bar{x}^N, P_i)]^T$.

To obtain a solution for the least-squares problem (5.14) one requires at least $N = n(n+1)/2$ points, which is the number of independent elements in the matrix P. The least-squares problem can be solved in online after a sufficient number of data points are collected along a single state trajectory. The solution of equation (5.13) can also be obtained using the recursive least squares algorithm (RLS) in which case a persistence of excitation condition is required.

This procedure requires only measurements of the states at discrete moments in time, t and $t+T_0$, as well as knowledge of the observed reward over the sample time interval $[t, t+T_0]$. Therefore there is no required knowledge about the system A matrix for the update of the critic or the action. However the B matrix is required for the update of the control policy (actor), using (5.7), and this makes the tuning algorithm only partially model free.

It has to be observed that the update of both the actor and the critic is performed at discrete moments in time. However, the control action (5.7) is a continuous-time control, with gain updated at the sample points. Moreover, the critic update is based on

the observations of the continuous-time cost over a finite sample interval. As a result, the algorithm converges to the solution of the continuous-time optimal control problem.

5.3 Mathematical formulation of the ADP algorithm

In this section the HDP algorithm is analyzed and placed in relation with known results form optimal control theory.

Lemma 5.1 *The ADP iteration between (5.7) and (5.8) is equivalent to the quasi-Newton method*

$$P_{i+1} = P_i - (Ric'_{P_i})^{-1}\left(Ric(P_i) - (e^{A_i T_0})^T Ric(P_i) e^{A_i T_0}\right). \tag{5.15}$$

Proof:

Differentiating (5.8) with respect to time one obtains

$$\begin{aligned}\dot{V}_{i+1}(x(t)) = &-x^T(t)Qx(t) - u_i^T(t)Ru_i(t) + \\ &+ x^T(t+T_0)Qx(t+T_0) + u_i^T(t+T_0)Ru_i(t+T_0) \\ &+ \dot{V}_i(x(t+T_0))\end{aligned} \tag{5.16}$$

which can be written as

$$\begin{aligned}&(A+BK_i)^T P_{i+1} + P_{i+1}(A+BK_i) + K_i^T R K_i + Q = \\ &\quad = (e^{(A+BK_i)T_0})^T (A^T P_i + P_i A - P_i B R^{-1} B^T P_i + Q) e^{(A+BK_i)T_0}\end{aligned} \tag{5.17}$$

Adding and subtracting $A_i^T P_i + P_i A_i$ and making use of (5.3), (5.17) becomes

$$A_i^T (P_{i+1} - P_i) + (P_{i+1} - P_i) A_i = -Ric(P_i) + (e^{A_i T_0})^T Ric(P_i) e^{A_i T_0} \tag{5.18}$$

and can be written in a Quasi-Newton formulation as

$$P_{i+1} = P_i - (Ric'_{P_i})^{-1}\left(Ric(P_i) - (e^{A_i T_0})^T Ric(P_i) e^{A_i T_0}\right). \qquad \square$$

Remark 5.1 If $A_i = A + BK_i$ is stable and $T_0 \to \infty$ one may recover from (5.15) the standard Newton method, (5.2), to solve the ARE, for which Kleinman, [32], proved convergence conditioned by an initial stabilizing control gain K_0. It seems that the last term, appearing in the formulation of the new ADP algorithm, compensates for the need of an initial stabilizing gain.

Equations (5.7) and (5.8) can be written as

$$P_{i+1} = \int_0^{T_0} (e^{A_i t})^T (Q + P_i B R^{-1} B^T P_i) e^{A_i t} dt + (e^{A_i T_0})^T P_i e^{A_i T_0} \tag{5.19}$$

so that we obtain the next result.

Lemma 5.2 *Iteration (5.19) is equivalent to*

$$P_{i+1} = P_i + \int_0^{T_0} (e^{A_i t})^T Ric(P_i) e^{A_i t} dt \,. \tag{5.20}$$

Proof:

From matrix calculus one may write

$$P_i - (e^{A_i T_0})^T P_i e^{A_i T_0} = -\int_0^{T_0} (e^{A_i t})^T (A_i^T P_i + P_i A_i) e^{A_i t} dt \tag{5.21}$$

From (5.19) and (5.21) follows (5.20). □

Note that (5.20) is just a different way of writing (5.19).

Remark 5.2 As $T_0 \to 0$, (5.20) becomes

$$\begin{aligned} \dot{P} &= A^T P + PA - PBR^{-1} B^T P + Q \\ P(0) &= P_0 \end{aligned} \tag{5.22}$$

which is a *forward-in-time* computation of the ARE solution, with the terminal boundary condition considered at the starting time, $P_{t_f} = P_0$.

Remark 5.3 The term $e^{A_i T_0}$ is the discrete-time version, obtained for the sample time T_0, of the closed-loop system matrix A_i. Therefore (5.19) is the expression of a hybrid discrete-time/continuous-time Riccati equation recursion.

Lemma 5.3 *Let the ADP algorithm converge so that* $P_i \to P^*$. *Then* P^* *satisfies* $Ric(P^*) = 0$, *i.e.* P^* *is the solution the continuous-time ARE.*

Proof:

If $\{P_i\}$ converges, then taking the limit in (5.20),

$$\lim_{P_i \to P^*} (P_{i+1} - P_i) = \lim_{P_i \to P^*} \int_0^{T_0} e^{A_i t^T} Ric(P_i) e^{A_i t} dt . \tag{5.23}$$

This implies

$$\int_0^{T_0} (e^{A^* t})^T Ric(P^*) e^{A^* t} dt = 0 \tag{5.24}$$

with $A^* = A - BR^{-1}B^T P^*$, and thus $Ric(P^*) = 0$. □

5.4 Simulation result illustrating the online CT HDP design for a power system

5.4.1 System model and motivation

In this section the continuous-time V-learning ADP algorithm is used to determine an optimal controller for the power system introduced in Section 2.4. As previously discussed the nonlinearity in the dynamics of such systems is determined by the load value, which under normal operation is constant. At the same time the values of the

parameters defining the linear model of the actual plant are not precisely known. In view of these facts the presented HDP design technique presented is a good candidate for the design of the desired LQR controller, for a given operating point of the system. The model of the system [60] is

$$\dot{x} = Ax(t) + Bu(t) \tag{5.25}$$

where

$$x(t) = [\Delta f(t) \quad \Delta P_g(t) \quad \Delta X_g(t) \quad \Delta E(t)]^T$$

$$A = \begin{bmatrix} -1/T_p & K_p/T_p & 0 & 0 \\ 0 & -1/T_T & 1/T_T & 0 \\ -1/RT_G & 0 & -1/T_G & -1/T_G \\ K_E & 0 & 0 & 0 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & 0 & 1/T_G & 0 \end{bmatrix}$$

The system states are: $\Delta f(t)$ - incremental frequency deviation (Hz), $\Delta P_g(t)$ - incremental change in generator output (p.u. MW), $\Delta X_g(t)$ - incremental change in governor position (p.u. MW), $\Delta E(t)$ - incremental change in integral control; and the system parameters are: T_G - the governor time constant, T_T - turbine time constant, T_P - plant model time constant, K_P - plant model gain, R - speed regulation due to governor action, K_E - integral control gain.

5.4.2 Simulation setup and results

In the simulation, only the time constant T_G of the governor, which appears in the B matrix, is considered to be known, while the values for all the other parameters appearing in the system A matrix are not known.

The system parameters, necessary for simulating the system behavior are picked randomly before the simulation is started in some realistic ranges, as specified in [60], such that:

$1/T_p \in [0.033, 0.1]$
$K_p/T_p \in [4,12]$
$1/T_T \in [2.564, 4.762]$
$1/T_G \in [9.615, 17.857]$
$1/RT_G \in [3.081, 10.639]$

Note that, even if the values of the parameters are known to be in the above mentioned ranges, the algorithm does not make use of any of this knowledge, only the exact value of T_G being necessary. Also, although the values of the controller parameters K_E and R are known, as they are specified by the engineer, this information is not used by the CT HDP algorithm to determine the optimal controller.

The simulation results that are presented next were obtained considering a randomly picked set of values (in the above mentioned ranges) for the systems unknown parameters, i.e. matrix A. In all the simulations the B matrix is $B = [0 \quad 0 \quad 13.7355 \quad 0]$ and it is considered to be known. For the purpose of demonstrating the CT-HDP algorithm the initial state of the system is taken different than zero, $x_0 = [0 \quad 0.1 \quad 0 \quad 0]$, and the matrix $P_0 = 0$.

The online implementation requires the setup of a least-squares problem of the kind presented in Section 5.2 to solve for the values of the critic parameters, the matrix P_i, at each iteration step i. In the simulations the matrix P_i is determined after collecting 12

points for each least-squares problem. Each such point is calculated after observing the value of the reward over a time interval of $T = 0.1\text{s}$. Therefore a least-squares problem is solved and the critic is updated at each 1.2 s. The simulations were performed over a time interval of 50 s. As such, a number of 41 iterations were performed during each simulation experiment.

For the simulation the unknown values of the system parameters were randomly picked in the specified ranges and the system matrix was

$$A = \begin{bmatrix} -0.0596 & 5.0811 & 0 & 0 \\ 0 & -3.0938 & 3.0938 & 0 \\ -10.0912 & 0 & -13.7355 & -13.7355 \\ 0.6 & 0 & 0 & 0 \end{bmatrix}.$$

The algorithm was run and at each iteration step a solution of (18), explicitly given by (19), was obtained. The convergence of few of the critic parameters $P(1,1)$, $P(1,3)$, $P(2,4)$ and $P(4,4)$ is presented in Figure 18.

The solution of the ARE for this given matrix A and $Q = I_4, R = 1$ is

$$P = \begin{bmatrix} 0.6920 & 0.5388 & 0.0551 & 0.6398 \\ 0.5388 & 0.7361 & 0.1009 & 0.4173 \\ 0.0551 & 0.1009 & 0.0451 & 0.0302 \\ 0.6398 & 0.4173 & 0.0302 & 2.3550 \end{bmatrix}.$$

After 41 iteration steps the critic is characterized by

$$P_{41} = \begin{bmatrix} 0.6914 & 0.5381 & 0.0551 & 0.6371 \\ 0.5381 & 0.8922 & 0.1008 & 0.4144 \\ 0.0551 & 0.1008 & 0.0451 & 0.0299 \\ 0.6371 & 0.4144 & 0.0299 & 2.3442 \end{bmatrix}.$$

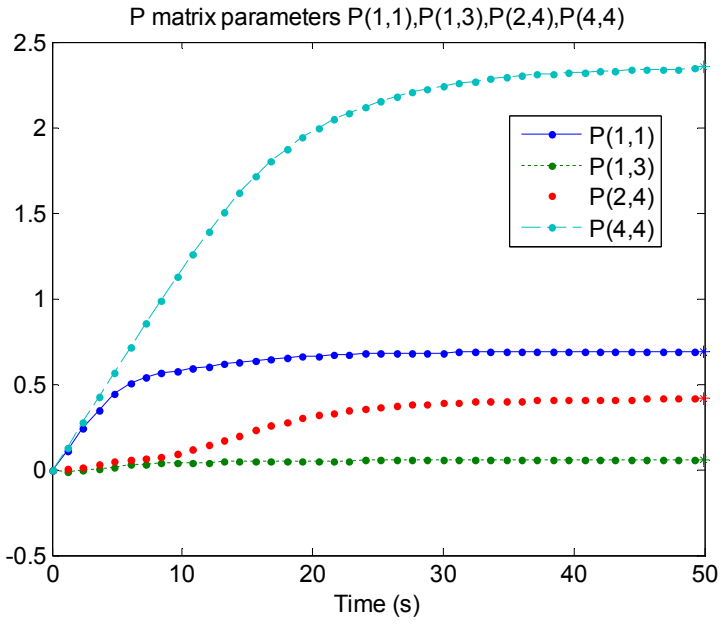

Figure 18. Convergence of P matrix parameters in online CT-HDP

Comparing the values of the two matrices it can be noted that after 41 iteration steps the error between their parameters is of order 10^{-3}, i.e. the algorithm converged to the solution of the ARE.

In Figure 19 is presented the evolution of the states of the system during the simulation. In Figure 20 the system states are showed in detail during the first 6 seconds, i.e. the first 5 iteration steps of the simulation. The control signal that was applied to the system during the CT HDP tuning is presented in Figure 21.

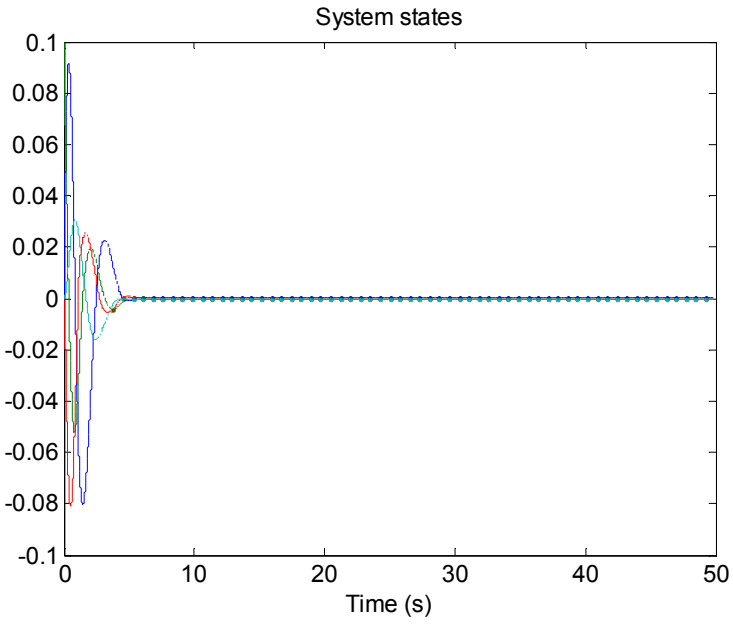

Figure 19. System states during the simulation

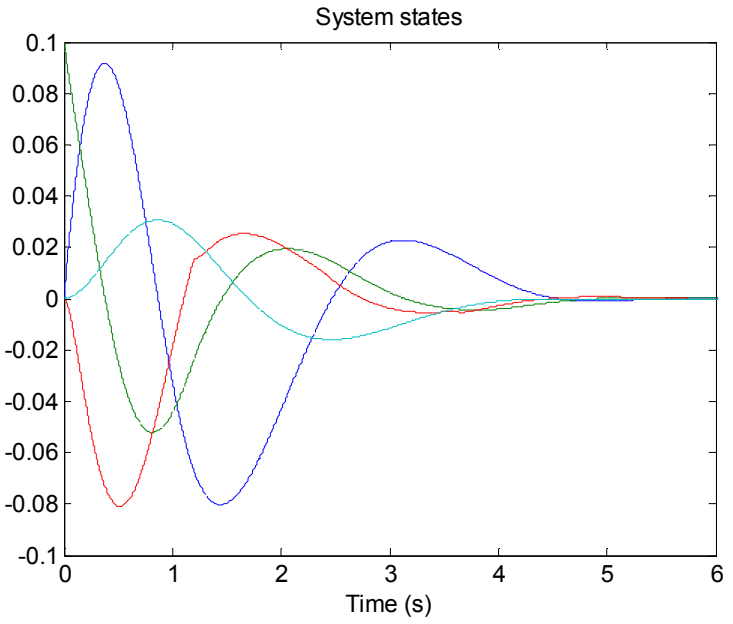

Figure 20. System states during the first 5 iteration steps

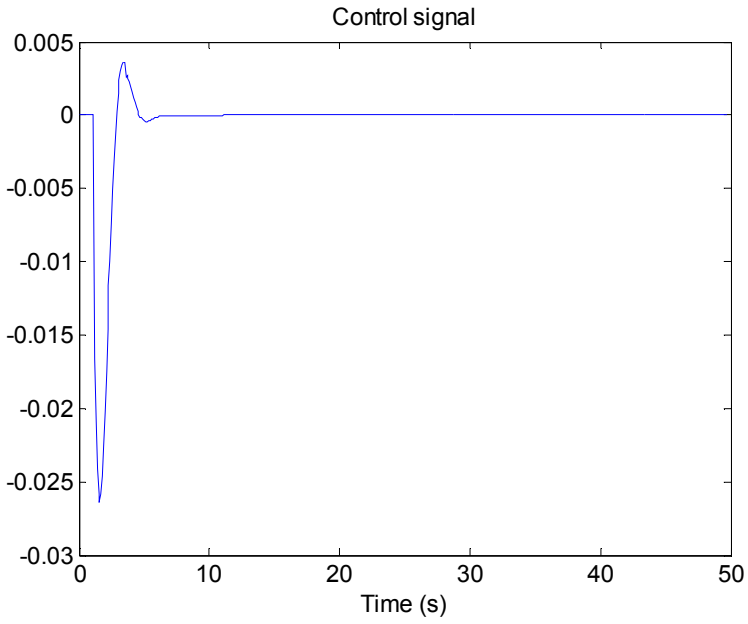

Figure 21. Control signal for simulation of online CT HDP

5.4.3 Comments on the convergence of CT HDP algorithm

The relation between the time period T over which the value function is observed at each step and the algorithm convergence is investigated in the following.

Figure 22 shows the convergence of the critic parameters, in the case of the second simulation setup, when the time period is taken $T = 0.2$ s. Over the 50 s duration of the simulation only 20 iterations are performed, the necessary data (12 points) for solving each least-squares problem being collected over an interval of 2.4 s.

By comparing the results plotted in Figure 18 with the ones presented in Figure 22 it becomes clear that the amount of time necessary for convergence is not dependent on the sample period that is used for observation. However, the number of iteration steps that are required for convergence is reduced when a large sample period is considered.

The reason is that, in case a larger observation sample is used, an increased amount of information regarding the system is carried in the data points collected for the critic update. As such, at each step of the iteration the critic improvement is larger when the time period is increased.

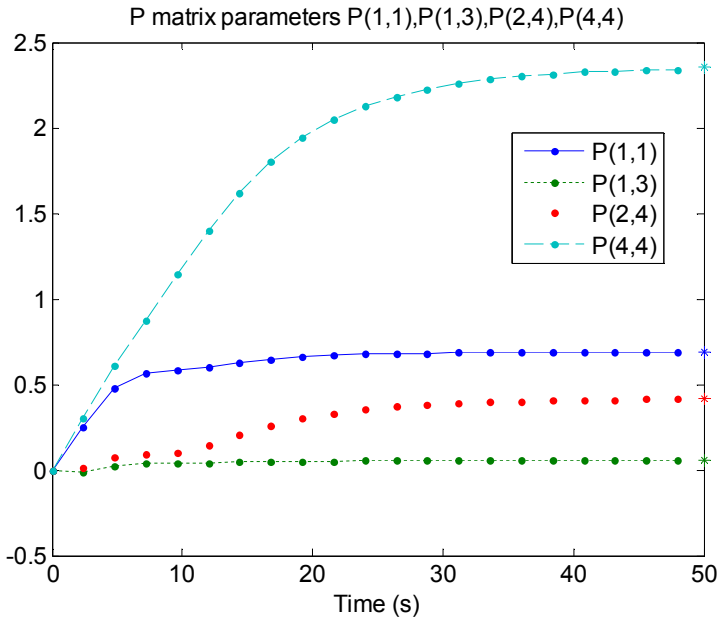

Figure 22. Convergence of P matrix parameters in online CT HDP for T=0.2s

5.5 Conclusion

In this chapter was presented a continuous-time ADP scheme which solves the continuous-time infinite horizon optimal control problem.

The control signal is applied to the system in a continuous time fashion. The actor's continuous time performance is measured over given time intervals and, based on this acquired information data, the critic reevaluates the infinite horizon cost and updates the actor's parameters (i.e. the continuous time system controller) in the sense of improving the over all system performance (i.e. to minimize the infinite horizon continuous-time

cost). As such, the system performance informational loop, which involves the critic entity, handles discrete information regarding the continuous time performance while the system control loop, which involves the actor, operates entirely in continuous time. The algorithm, equivalent to a quasi-Newton method, solves the continuous-time ARE and obtains the optimal controller in an online, forward in time iteration without using knowledge of the internal dynamics of the plant and without starting with an initial stabilizing policy.

CHAPTER 6

ONLINE ADAPTIVE APPROACH BASED ON REINFORCEMENT LEARNING TO CONTINUOUS-TIME LINEAR DIFFERENTIAL ZERO-SUM GAMES

6.1 Introduction

This chapter will describe the manner in which the reinforcement learning approach to optimal control presented in Chapter 2 can be used to determine in an online fashion the saddle point solution of linear differential zero-sum games. Here will be considered the infinite horizon, state-feedback, linear-quadratic case of the problem.

The solution of the problem is connected with the unique positive definite solution of a Riccati equation that has a sign indefinite quadratic term. One approach to solving this equation is the Newton procedure. In this approach the solution of the problem of our interest is obtained as the limit of a sequence of matrices. Every matrix in this sequence is determined while solving a Riccati equation with sign definite quadratic term of the sort associated with the optimal control problem presented in Chapter 2.

The online algorithms which will be described here are built on two known results, namely the one in [36], and the one introduced in [59] and further developed in [2], which involve solving a sequence of Riccati equations with sign definite quadratic term. It will also be shown using a common formulation that the two iterative approaches to the solution of the problem are in fact two faces of the same coin. The complementarity of the two algorithms becomes obvious from a game theoretical perspective.

We begin our investigation by looking at the formulation of the problem.

6.1.1 Formulation of the problem

Consider the linear system

$$\begin{cases} \dot{x} = Ax + Dw + B_2 u \\ z = \begin{bmatrix} Cx \\ u \end{bmatrix} \end{cases} \tag{6.1}$$

where w denoted the disturbance signal affecting the dynamics of the linear system.

Here we will give the formulation of the zero-sum game problem in connection to the H-infinity control problem as the solution of both problems satisfies the same underlying Riccati equation. We first give the definition of the H-infinity norm which will be used to measure the performance of the control system.

We note that the H-infinity norm of a system has been first defined from a frequency domain perspective and consequently, based on the original definition, the concept can not be directly extended for the case of nonlinear systems. For this reason in the following we prefer to use the time domain equivalent of the H-infinity norm (the L_2-gain) which can be used also for the case of nonlinear systems.

Definition 6.1 Let $\gamma \geq 0$ and $u = Kx$. The system (6.1) is said to have L_2 gain less than or equal to γ if

$$\int_0^\infty \|z(t)\|^2 \, dt \leq \gamma^2 \int_0^\infty \|w(t)\|^2 \, dt \tag{6.2}$$

for all $w \in L_2(0,\infty)$, where $\|w\|^2 = w^T w$. The system has an L_2 gain less than γ if there exists $\tilde{\gamma}$, $0 \leq \tilde{\gamma} < \gamma$ such that (6.2) holds for $\tilde{\gamma}$.

Let γ^* denote the smallest L_2 gain of the system (6.1). Then the linear infinite horizon state-feedback H-infinity optimal control problem is formulated as follows:

Definition 6.2 *Linear state-feedback H-infinity optimal control problem.* Find the smallest value $\gamma^* \geq 0$ such that for any $\gamma > \gamma^*$ there exists a state-feedback control law $u = Kx$ such that the L_2 gain from *w* to *z* is less than or equal to γ and the closed loop system is asymptotically stable.

It has been shown in the literature (see e.g. [67], [8]) that the H-infinity control solution is connected with the solution of a matrix equation of Riccati type which has a sign indefinite quadratic term. To obtain the H-infinity optimal controller, given that the smallest L_2 gain γ^* is known, one needs to find the solution of the Algebraic Riccati Equation

$$0 = A^T P + PA + C^T C - P(B_2 B_2{}^T - \frac{1}{\gamma^{*2}} DD^T)P \tag{6.3}$$

To simplify the mathematical notation in the Riccati equation (6.3) in the following we will denote $B_1 = \frac{1}{\gamma^*} D$. Thus (6.3) will be written as

$$0 = A^T P + PA + C^T C - P(B_2 B_2{}^T - B_1 B_1{}^T)P. \tag{6.4}$$

For any $\gamma > \gamma^*$, one can find a suboptimal H-infinity state-feedback controller, which admits a performance level of at least γ, by solving

$$0 = A^T P + PA + C^T C - P(B_2 B_2{}^T - \frac{1}{\gamma^2} DD^T)P. \tag{6.5}$$

In [8] is given the following result which states that the Riccati equation (6.5) has a unique positive definite solution for every $\gamma > \gamma^*$.

Theorem 6.1 For the infinite horizon H-infinity control problem with closed-loop perfect state information, let (*A*, *B*) be stabilizable and (*A*, *C*) detectable. Let λ^* be the smallest positive scalar with the property that for all $\lambda > \lambda^*$ the algebraic Riccati equation

$$0 = A^T P + PA + C^T C - P(B_2 B_2{}^T - \frac{1}{\lambda^2} DD^T)P \tag{6.6}$$

admits a minimal positive definite solution P_λ. Then $\lambda^* = \gamma^*$ and for all $\lambda > \lambda^*$ the feedback control policy

$$u = -B_2{}^T P_\lambda x \tag{6.7}$$

delivers a performance level of at least λ, and under it the linear system

$$\dot{x} = (A - B_2 B_2{}^T P_\lambda)x + Dw \tag{6.8}$$

is *bounded input bounded state* (BIBS) stable.

This theorem relates the solvability of a Riccati equation (6.6), that has a sign indefinite quadratic term and $\lambda > 0$, with the L_2 gain characterization of the linear system given in definition 6.1.

The H-infinity optimal control problem can also be formulated as the two-player zero-sum differential game.

Consider the system

$$\begin{cases} \dot{x} = Ax + B_1 w + B_2 u \\ y = Cx \end{cases} \tag{6.9}$$

with the performance index

$$V(x_0,u,w)=\int_0^{\infty}(u^Tu+x^TC^TCx-w^Tw)dt\,. \tag{6.10}$$

The control policy player desires to minimize the performance index while the disturbance policy player desires to maximize it. The goal is to determine the saddle point solution.

We will refer to (6.4) as *game algebraic Riccati equation* (GARE).

Denoting with Π the unique positive definite solution of (6.4) the saddle point of the Nash game is

$$\begin{aligned} u&=-B_2{}^T\Pi x\\ w&=B_1{}^T\Pi x\\ V(x_0,u,w)&=x_0{}^T\Pi x_0 \end{aligned}\quad . \tag{6.11}$$

We shall use the notations $u=Kx$ and $w=Lx$ for the state feedback control and respectively disturbance policies. We say that K is the gain of the control policy and L is the gain of the disturbance policy. The meaning of the saddle point solution of the Nash differential game is that for any state feedback control policy $\tilde{u}=\tilde{K}x$ and any state-feedback disturbance policy $\tilde{w}=\tilde{L}x$, different than the ones in (6.11), the value of the game will satisfy

$$V(x_0,\tilde{u},w)\ge V(x_0,u,w)\ge V(x_0,u,\tilde{w})\,. \tag{6.12}$$

6.1.2 Online approach to the solution of the differential game and secondary contributions

Based on the results presented in the previous subsection it is clear that finding the saddle point equilibrium solution of the two player zero-sum differential game, or that

of the H-infinity optimal control problem, is equivalent with finding the unique positive definite solution of the Riccati equation (6.4). The goal of this chapter is to provide an online, data-based, approach to the solution of the problem.

In the next section we use the same framework to formulate the two iterative algorithms given in [36] and [59] which find the solution of the GARE (6.4). Both algorithms are iterative procedures that, by means of finding solution for a sequence of sign definite Riccati equations, build a sequence of symmetric positive definite matrices which converges to the solution of the sign indefinite Riccati equation associated with the zero-sum differential game. In Section 6.3 it will be shown how to use reinforcement learning techniques to implement each of the two algorithms online, based on measured data from the system.

This chapter has two secondary achievements:

- It shows, using a unified framework, that the two algorithms are complementary to each other. One approaches the solution of the equation (6.4) by building a monotonically increasing and upper bounded sequence while the other one build a monotonically decreasing and lower bounded one. In both cases the bound is given by the unique positive definite solution of (6.4).
- From the perspective of game theory it is shown that each of the two algorithms leads to the equilibrium solution of the Nash game while only one of the two players learns to optimize its actions and the other player is passive. Specifically, in the case of the algorithm presented in [36] the

optimizing player is the controller, while in the case of the algorithm presented in [59] and [2] the optimizing player is the disturbance.

The next section presents the two iterative algorithms using the framework developed in [36] and thus includes the first secondary achievement of this chapter.

6.2 Iterative approaches to the H-infinity control solution

In this section are discussed, using the same formulation framework, namely the one used by Lanzon et al. in their 2008 paper, two iterative approaches to finding the saddle point solution of the two payer zero-sum differential game. From this point forward we will look at the problem only from the perspective of game theory. This perspective provides an interesting interpretation regarding the active or passive behavior with respect to learning of the two payers. A discussion from this perspective will be included in the third section of this chapter.

6.2.1 Iterations on the control policy

In [36] has been introduced the following iterative method for solving the game algebraic Riccati equation (GARE) (6.4).

Algorithm 6.1 – iterations on the control policy.

0. Start with

$$P_u^0 = 0 . \tag{6.13}$$

1. Solve

$$\begin{aligned} 0 = {} & Z_u^i (A + B_1 B_1^T P_u^{i-1} - B_2 B_2^T P_u^{i-1})^T + (A + B_1 B_1^T P_u^{i-1} - B_2 B_2^T P_u^{i-1}) Z_u^i \\ & - Z_u^i B_2 B_2^T Z_u^i + F(P_u^{i-1}) \end{aligned} . \tag{6.14}$$

2. Update

$$P_u^i = P_u^{i-1} + Z_u^i . \tag{6.15}$$

The convergence of the algorithm to the unique positive definite solution of the GARE (6.4) is proven based on the following three results given and proved in [36].

Lemma 6.1 Given real matrices A, B_1, B_2, C with compatible dimensions, define

$$\begin{aligned} &F : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n} \\ &F(P) = A^T P + PA + C^T C - P(B_2 B_2^{\ T} - B_1 B_1^{\ T})P \end{aligned} \tag{6.16}$$

Given $P \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times n}$ symmetric matrices then

$$\begin{aligned} F(P+Z) = (A - B_2 B_2^{\ T} P + B_1 B_1^{\ T} P)^T Z + Z(A - B_2 B_2^{\ T} P + B_1 B_1^{\ T} P) + \\ + F(P) - Z(B_2 B_2^{\ T} - B_1 B_1^{\ T})Z \end{aligned} \tag{6.17}$$

It directly follows that if

$$(A - B_2 B_2^{\ T} P + B_1 B_1^{\ T} P)^T Z + Z(A - B_2 B_2^{\ T} P + B_1 B_1^{\ T} P) + F(P) - ZB_2 B_2^{\ T} Z = 0 \tag{6.18}$$

then

$$F(P+Z) = ZB_1 B_1^{\ T} Z . \tag{6.19}$$

Lemma 6.2 Given real matrices A, B_1, B_2, C with compatible dimensions, $P \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times n}$ satisfying (6.17), and $\Pi > 0$ a stabilizing solution of (6.4), such that

$$0 = A^T \Pi + \Pi A + C^T C - \Pi(B_2 B_2^{\ T} - B_1 B_1^{\ T})\Pi . \tag{6.20}$$

a. If $A - B_2 B_2^{\ T} \Pi + B_1 B_1^{\ T} P$ is Hurwitz then $\Pi \geq P + Z$.

b. If $\Pi \geq P + Z$ then $A - B_2 B_2^{\ T} \Pi + B_1 B_1^{\ T}(P+Z)$ is Hurwitz.

Proof: The proof, given in [36], uses the sum of (6.18) with (6.19) and (6.20) and a Lyapunov argument.

Theorem 6.2 Given real matrices A, B_1, B_2, C with compatible dimensions, such that all unobservable modes of (C, A) are strictly stable and (A, B_2) stabilizable, define the map F as in (6.16). Suppose that there exists a stabilizing solution $\Pi > 0$ of (6.4).

Then

(I) there exist two square matrix series $P_u^i \in \mathbb{R}^{n\times n}$ and $Z_u^i \in \mathbb{R}^{n\times n}$ for all $i \in \mathbb{N}$ satisfying Algorithm 6.1.

(II) the elements of the two series, defined recursively, have the following properties:

a. $(A + B_1 B_1^T P_u^i, B_2)$ is stabilizable for all $i \in \mathbb{N}$.

b. $Z_u^i \geq 0 \; \forall i \in \mathbb{N}$

c. $F(P_u^{i+1}) = Z_u^i B_1 B_1^T Z_u^i \;\; \forall i \in \mathbb{N}$

d. $A + B_1 B_1^T P_u^i - B_2 B_2^T P_u^{i+1}$ is Hurwitz $\forall i \in \mathbb{N}$

e. $\Pi \geq P_u^{i+1} \geq P_u^i \geq 0 \;\; \forall i \in \mathbb{N}$

(III) let $P_u^\infty = \lim_{i\to\infty} P_u^i \geq 0$

then $P_u^\infty = \Pi$.

The proof, given in [36], uses the results of the two lemmas with an inductive argument.

We now introduce two propositions which provide equivalent formulations for Algorithm 6.1. We are introducing them here in order to bring meaning to every of the iterative algorithm.

Proposition 6.1 The iteration between (6.14) and (6.15) in Algorithm 6.1 can be written as

$$P_u^i(A+B_1B_1^TP_u^{i-1})+(A+B_1B_1^TP_u^{i-1})^TP_u^i-P_u^iB_2B_2^TP_u^i-P_u^{i-1}B_1B_1^TP_u^{i-1}+C^TC=0 \quad (6.21)$$

This result was obtained by writing compactly the two equations and making use of the definition of the map F.

Proposition 6.2 The iteration between (6.14) and (6.15) in Algorithm 6.1 can be written as

$$\begin{aligned}0=&(P_u^i-P_u^{i-1})(A+B_1B_1^TP_u^{i-1}-B_2B_2^TP_u^{i-1})+(A+B_1B_1^TP_u^{i-1}-B_2B_2^TP_u^{i-1})^T(P_u^i-P_u^{i-1})\\&-(P_u^i-P_u^{i-1})B_2B_2^T(P_u^i-P_u^{i-1})+(P_u^{i-1}-P_u^{i-2})B_1B_1^T(P_u^{i-1}-P_u^{i-2})\end{aligned} \quad (6.22)$$

This results directly from (6.14), (6.15) and (6.19).

It is important to notice at this point that the result given in Proposition 6.2 includes three instances of the index of the sequence $\{P_u^i\}_{i\geq 0}$, namely $P_u^{i-2}, P_u^{i-1}, P_u^i$. For this reason it can only be used for calculating the values $\{P_u^i\}_{i\geq 2}$ provided that the first two elements in the sequence are available.

The next two propositions formulate optimal control problems which are associated with the Riccati equations (6.21) and (6.22). This is important since they attach meaning to the recursive algorithm enhancing both the reinforcement learning perspective and the game theoretical reasoning.

Proposition 6.3 Solving the Riccati equation (6.21) is equivalent to finding solution for the following optimal control problem:

"For the system $\dot{x} = A + B_1 w_{i-1} + B_2 u_i$ let the state-feedback disturbance policy gain be $L_u^{i-1} = B_1^T P_u^{i-1}$ such that $w_{i-1} = B_1^T P_u^{i-1} x$. Determine the state-feedback control policy u_i such that the infinite horizon quadratic cost index

$\int_0^\infty [x^T C^T C x - w_{i-1}^T w_{i-1} + u_i^T u_i] dt$ is minimized."

Let $x_0^T P_u^i x_0 = \min_{u_i} \int_0^\infty [x^T (C^T C - P_u^{i-1} B_1 B_1^T P_u^{i-1}) x + u_i^T u_i] dt$ then the optimal control policy is given by $K_u^i = -B_2^T P_u^i$ such that the optimal state-feedback control is $u_i = -B_2^T P_u^i x$.

Proposition 6.4 Solving the Riccati equation (6.22) is equivalent to finding solution for the following optimal control problem:

"For the system $\dot{x} = A + B_1 w_{i-1} + B_2 (u_{i-1} + \hat{u}_i)$ let the state-feedback disturbance policy be $w_{i-1} = B_1^T P_u^{i-1} x$ and the base state-feedback control policy be $u_{i-1} = -B_2^T P_u^{i-1} x$. Determine the correction for the state-feedback control policy, $\hat{u}_i$, which minimizes the infinite horizon quadratic cost index

$\int_0^\infty [x^T (P_u^{i-1} - P_u^{i-2}) B_1 B_1^T (P_u^{i-1} - P_u^{i-2}) x + \hat{u}_i^T \hat{u}_i] dt$."

Let $x_0^T Z_u^i x_0 = \min_{\hat{u}_i} \int_0^\infty [x^T (P_u^{i-1} - P_u^{i-2}) B_1 B_1^T (P_u^{i-1} - P_u^{i-2}) x + \hat{u}_i^T \hat{u}_i] dt$ then the optimal control policy $u_i = u_{i-1} + \hat{u}_i$ is $u_i = -B_2^T (P_u^{i-1} + Z_u^i) x$.

6.2.2 Iterations on the disturbance policy

The next algorithm provides a new formulation, using the framework used in [36], for the algorithm which was introduced in [59] and used in [2] to find solution for the Hamilton-Jacobi-Isaacs equation underlying the H-infinity optimal control problem for nonlinear systems.

We note that a formulation for the linear case has not been previously given and is introduced in this work.

Algorithm 6.2 – iterations on the disturbance policy

0. Start with

$$P_w^0 \text{ such that } A - B_2 B_2^T P_w^0 \text{ and } A - B_2 B_2^T P_w^0 + B_1 B_1^T \Pi \text{ are Hurwitz and } P_w^0 \geq \Pi \,. \quad (6.23)$$

1. Solve

$$\begin{aligned} 0 = {} & Z_w^i (A + B_1 B_1^T P_w^{i-1} - B_2 B_2^T P_w^{i-1}) + (A + B_1 B_1^T P_w^{i-1} - B_2 B_2^T P_w^{i-1})^T Z_w^i \\ & - Z_w^i B_1 B_1^T Z_w^i - F(P_w^{i-1}) \end{aligned} \quad (6.24)$$

2. Update

$$P_w^i = P_w^{i-1} - Z_w^i \,. \quad (6.25)$$

The next three results, which we developed, provide assurance for the convergence to the solution of the ARE. It is important to mention that we structured our results such that they mirror the structure of the results given in [36]. This will allow the reader to clearly notice the similarity between the two algorithms.

As these results make use of the definition of the operator *F*, introduced in Lemma 6.1, we view our next result as a corollary to Lemma 6.1.

Corollary 6.1 Using the definition in Lemma 6.1, it directly follows that if

$$(A - B_2 B_2^T P + B_1 B_1^T P)^T Z + Z(A - B_2 B_2^T P + B_1 B_1^T P) - F(P) - Z B_1 B_1^T Z = 0 \quad (6.26)$$

then

$$F(P-Z) = -ZB_2B_2^{\,T}Z\,. \tag{6.27}$$

The next Lemma will prove valuable for showing lower boundedness for the matrix sequence which is constructed by the algorithm.

Lemma 6.3 Given real matrices A, B_1, B_2, C with compatible dimensions, $P \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times n}$ satisfying

$$(A - B_2B_2^{\,T}P + B_1B_1^{\,T}P)^T Z + Z(A - B_2B_2^{\,T}P + B_1B_1^{\,T}P) - F(P) - ZB_1B_1^{\,T}Z = 0 \tag{6.28}$$

and $\Pi > 0$ a stabilizing solution of the GARE (6.4).

a. If $A - B_2B_2^{\,T}P + B_1B_1^{\,T}\Pi$ is Hurwitz then $\Pi \le P - Z$.

b. If $\Pi \le P - Z$ then $A - B_2B_2^{\,T}(P-Z) + B_1B_1^{\,T}\Pi$ is Hurwitz.

The proof, given in the Appendix, uses the sum of (6.28) with (6.20) and (6.27) and a Lyapunov argument.

The next proposition provides an equivalent formulation for Algorithm 6.2 which will be used as the base for proving its convergence.

Proposition 6.5 The iteration between (6.24) and (6.25) can be written as

$$P_w^i(A - B_2B_2^{\,T}P_w^{i-1}) + (A - B_2B_2^{\,T}P_w^{i-1})^T P_w^i + P_w^iB_1B_1^{\,T}P_w^i + P_w^{i-1}B_2B_2^{\,T}P_w^{i-1} + C^TC = 0\,. \tag{6.29}$$

This result follows directly when writing compactly the two equations and making use of the definition of the map *F*.

The next lemma provides an iterative approach to the solution of (6.29) that will be used as a base for the online implementation of the algorithm. Moreover the results will prove useful in showing convergence of the resulting sequence.

Lemma 6.4 Let $Q^i \triangleq C^TC + P_w^{i-1}B_2B_2^TP_w^{i-1}$, $A - B_2B_2^TP_w^{i-1}$ - Hurwitz, $(A - B_2B_2^TP_w^{i-1}, B_1)$ controllable and $(\sqrt{Q^i}, A - B_2B_2^TP_w^{i-1})$ detectable (i.e. all its unobservable modes are in the left half of the complex plane).

Then the unique positive definite solution of

$$P_w^i(A - B_2B_2^TP_w^{i-1})^T + (A - B_2B_2^TP_w^{i-1})P_w^i + C^TC + P_w^iB_1B_1^TP_w^i + P_w^{i-1}B_2B_2^TP_w^{i-1} = 0 \quad (6.30)$$

such that $(A - B_2B_2^TP_w^{i-1} + B_1B_1^TP_w^i)$ is Hurwitz, can be determined using the policy iteration algorithm

a) $$P_w^{i(k)}(A - B_2B_2^TP_w^{i-1} + B_1L^{(k-1)}) + (A - B_2B_2^TP_w^{i-1} + B_1L^{(k-1)})^TP_w^{i(k)} + Q^i - (L^{(k-1)})^TL^{(k-1)} = 0 \quad (6.31)$$

b) $$L^{(k)} = B_1^TP_w^{i(k)}, \quad (6.32)$$

where $L^{(0)} = 0$.

Moreover, $A - B_2B_2^TP_w^i$ is Hurwitz and the available storage function of the system $\dot{x} = (A - B_2B_2^TP_w^i)x + B_1d$, i.e. the solution of (6.30) for $i \to i+1$, is such that $P_w^{i+1} \le P_w^i$.

The proof is given in the appendix section.

The next lemma summarizes the properties of Algorithm 6.2.

Lemma 6.5 Given real matrices A, B_1, B_2, C with compatible dimensions, such that all unobservable modes of (A, C) are stable and (A, B_1) stabilizable, define the map F as in (6.16). Suppose that there exists a stabilizing solution $\Pi > 0$ of (6.4).

Then

(I) there exist two square matrix series $P_w^i \in \mathbb{R}^{n\times n}$ and $Z_w^i \in \mathbb{R}^{n\times n}$ for all $i \in \mathbb{N}$ satisfying Algorithm 6.2.

(II) the elements of the two series, defined recursively, have the following properties:

a. $A - B_2 B_2{}^T P_w^i$ is Hurwitz for all $i \in \mathbb{N}$.

b. $(A - B_2 B_2{}^T P_w^i, B_1)$ is stabilizable for all $i \in \mathbb{N}$.

c. $Z_w^i \geq 0 \; \forall i \in \mathbb{N}^*$

d. $F(P_w^{i+1}) = -Z_w^i B_2 B_2{}^T Z_w^i \;\; \forall i \in \mathbb{N}$

e. $A - B_2 B_2{}^T P_w^i + B_1 B_1{}^T P_w^{i+1}$ is Hurwitz $\forall i \in \mathbb{N}$

f. $0 \leq \Pi \leq P_w{}^{i+1} \leq P_w{}^i \;\; \forall i \in \mathbb{N}$

(III) let $P_w^\infty = \lim_{i\to\infty} P_w^i \geq 0$

then $P_w^\infty = \Pi$.

A proof by induction along the lines of the proof of the Theorem 6.2, which uses the results in Corollary 6.1, Lemma 6.3, Lemma 6.4 and Proposition 6.5, is straightforward and is omitted here. The reader is referred also to the [2] for a proof concerning the nonlinear version of the GARE, the HJI equation.

Proposition 6.6 The iteration in Algorithm 6.2 can be written as

$$0 = (P_w^i - P_w^{i-1})(A + B_1 B_1^T P_w^{i-1} - B_2 B_2^T P_w^{i-1})^T + (A + B_1 B_1^T P_w^{i-1} - B_2 B_2^T P_w^{i-1})(P_w^i - P_w^{i-1}) + (P_w^{i-1} - P_w^{i-2}) B_2 B_2^T (P_w^{i-1} - P_w^{i-2}) - (P_w^i - P_w^{i-1}) B_1 B_1^T (P_w^i - P_w^{i-1}) \quad (6.33)$$

This results directly from (6.24), (6.25) and (6.27).

Proposition 6.7 Solving the equation (6.29) is equivalent to finding solution for the following optimization problem:

"For the system $\dot{x} = A + B_1 w_i + B_2 u_{i-1}$ let the state-feedback control policy gain be $K_w^{i-1} = -B_2^T P_w^{i-1}$ such that $u_{i-1} = -B_2^T P_w^{i-1} x$. Determine the state feedback disturbance policy w_i such that the infinite horizon quadratic cost index $\int_0^\infty [x^T (C^T C + P_w^{i-1} B_2 B_2^T P_w^{i-1}) x - w_i^T w_i] dt$ is maximized."

Let $x_0^T P_w^i x_0 = \max_{w_i} \int_0^\infty [x^T (C^T C + P_w^{i-1} B_2 B_2^T P_w^{i-1}) x - w_i^T w_i] dt$ then the disturbance state-feedback policy which maximizes the cost index is $w_i = L_w^i x = B_1^T P_w^i x$.

Proposition 6.8 Solving equation (6.33) is equivalent to finding solution for the optimization problem:

"For the system $\dot{x} = A + B_1 (w_{i-1} + \hat{w}_i) + B_2 u_{i-1}$ let the state-feedback control policy be given by $u_{i-1} = K_w^{i-1} x = -B_2^T P_w^{i-1} x$ and the state feedback disturbance policy be given by the gain $w_{i-1} = L_w^{i-1} x = -B_2^T P_w^{i-1} x$. Determine the state-feedback

corrective disturbance policy such that the infinite horizon quadratic cost index $\int_0^\infty [x^T (P_w^{i-1} - P_w^{i-2}) B_2 B_2^T (P_w^{i-1} - P_w^{i-2}) x + \hat{w}_i^T \hat{w}_i] dt$ is maximized."

Let $x_0^T Z_w^i x_0 = \max_{\hat{w}_i} \int_0^\infty [x^T (P_w^{i-1} - P_w^{i-2}) B_2 B_2^T (P_w^{i-1} - P_w^{i-2}) x + \hat{w}_i^T \hat{w}_i] dt$ then the optimal disturbance policy $w_i = w_{i-1} - \hat{w}_i$ is $w_i = -B_1^T (P_w^{i-1} - Z_w^i) x$.

We now summarize the features of the two algorithms.

- The first algorithm has the advantage of simple initialization however it does not guarantee the stability property of the system $\dot{x} = (A + B_1 B_1^T P_u^i) x, \forall i \in \mathbb{N}$. This will introduce difficulties in finding online the solution of the Riccati equation (6.18) while using the online reinforcement learning - based Policy Iteration algorithm which requires knowledge of an initial stabilizing control gain.
- The initialization of the second algorithm is slightly more involving, however , when properly done, the stability of $\dot{x} = (A + B_1 B_1^T P_u^i) x$ is guaranteed $\forall i \in \mathbb{N}$. This makes the algorithm more feasible for online implementation using reinforcement learning algorithms.

6.3 Online adaptive optimal approach to the solution of the two-player zeros sum game

The two iterative algorithms discussed in the previous section can be used as the backbone for the online approach to the saddle point solution of the zero-sum differential game. In this chapter we will describe the online algorithms. It is appropriate here to give a brief review of the online approaches for solving Riccati equations with sign definite quadratic term introduced in Chapters 2 and 5.

6.3.1 Online approaches to the solution of algebraic Riccati equations

The goal of this section is to present the two online algorithms which use reinforcement learning ideas to solve the ARE

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 .$$

A. *Online Policy Iteration algorithm*

Let K_1 be an initial stabilizing control gain. Denote the state measured at time t with x_t. The following reinforcement learning based policy iteration algorithm

a) $$x_t{}^T P_i x_t = \int_t^{t+T_0} x_\tau{}^T (Q + K_i{}^T R K_i) x_\tau d\tau + x_{t+T_0}{}^T P_i x_{t+T_0} \tag{6.34}$$

b) $$K_{i+1} = R^{-1}B^T P_i . \tag{6.35}$$

can be implemented online as follows.

In the step a) is desired to find the parameters (i.e. matrix P_i) of the cost function associated with the policy K_i. The term $x_t{}^T P_i x_t$ is written as

$$x_t{}^T P_i x_t = \overline{p}_i{}^T \overline{x}_t \tag{6.36}$$

where $\overline{x}_t$ denotes the Kronecker product quadratic polynomial basis vector with the elements $\{x_i(t)x_j(t)\}_{i=1,n;j=i,n}$ and $\overline{p} = v(P)$ with $v(.)$ a vector valued matrix function that acts on symmetric matrices and 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 $2P_{ij}$ [13]. Denote the reinforcement over the time interval $[t, t+T_0]$ by

$$d(\bar{x}_t, K_i) \equiv \int_t^{t+T_0} x^T(\tau)(Q + K_i^T R K_i)x(\tau)d\tau \, .$$

Then the first equation is rewritten as

$$\bar{p}_i^T(\bar{x}_t - \bar{x}_{t+T_0}) = d(\bar{x}_t, K_i) \, . \tag{6.37}$$

The vector of unknown parameters is $\bar{p}_i$, and $\bar{x}(t) - \bar{x}(t+T)$ acts as a regression vector. The right hand side target reinforcement function is measured based on the state trajectories over the time interval $[t, t+T_0]$. Considering $\dot{V}(t) = x^T(t)Qx(t) + u^T(t)Ru(t)$ as a definition for a new state $V(t)$, the value of $d(\bar{x}(t), K_i)$ can be obtained by taking two measurements of this newly introduced system state since $d(\bar{x}(t), K_i) = V(t+T) - V(t)$. This new state signal is simply the output of an analog integration block having as inputs the quadratic terms $x^T(t)Qx(t)$ and $u^T(t)Ru(t)$ which can also be obtained using an analog processing unit.

The parameter vector $\bar{p}_i$ is found by minimizing, in the least-squares sense, the error between the target reinforcement function, $d(\bar{x}(t), K_i)$, and the parameterized left hand side of (6.37). Evaluating the right hand side of (6.37) at $N \geq n(n+1)/2$ (the number of independent elements in the matrix P_i) points $\bar{x}^i$ in the state space, over the same time interval *T*, the batch least-squares solution is

$$\bar{p}_i = (XX^T)^{-1} XY \tag{6.38}$$

where

$$X=[\bar{x}_\Delta^1 \quad \bar{x}_\Delta^2 \quad ... \quad \bar{x}_\Delta^N]$$
$$\bar{x}_\Delta^i=\bar{x}^i(t)-\bar{x}^i(t+T) \qquad .$$
$$Y=[d(\bar{x}^1,K_i) \quad d(\bar{x}^2,K_i) \quad ... \quad d(\bar{x}^N,K_i)]^T$$

The least-squares problem can be solved in real-time after a sufficient number of data points are collected along a single state trajectory, under the regular presence of an excitation requirement.

Alternatively, the solution given by (6.38) can also be obtained by means of recursive estimation algorithms (e.g. gradient descent algorithms or the recursive least squares algorithm) in which case a persistence of excitation condition is required. For this reason there are no real issues related to the algorithm becoming computationally expensive with the increase of the state space dimension.

B. Online value iteration algorithm

The *reinforcement learning-based value iteration* algorithm is the following:

Let $P_0=0$ and K_0 a state-feedback control policy (not necessarily stabilizing). Iterate between

a) $$x_t{}^T P_{i+1} x_t = \int_t^{t+T_0} x_\tau{}^T (Q+K_i{}^T R K_i) x_\tau d\tau + x_{t+T_0}{}^T P_i x_{t+T_0} \tag{6.39}$$

b) $$K_{i+1}=R^{-1}B^T P_{i+1} \tag{6.40}$$

until convergence.

The online implementation of the algorithm is given next.

In the step a) is desired to find the parameters of the matrix P_{i+1} of the cost function. The two quadratic cost functions will be written as in (6.36) and we will use

the same notation $d(\bar{x}_t, K_i)$ for the reinforcement signal over the interval $[t, t+T_0]$. Based on these notations and structures the first equation is rewritten as

$$\bar{p}_{i+1}{}^T \bar{x}_t = d(\bar{x}_t, K_i) + \bar{p}_i{}^T \bar{x}_{t+T_0}. \tag{6.41}$$

The vector of unknown parameters is $\bar{p}_{i+1}$, and $\bar{x}_t$ acts as a regression vector. The right hand side target reinforcement function is measured based on the state trajectories over the time interval $[t, t+T_0]$ and the state value at $t+T_0$, $\bar{x}_{t+T_0}$.

The parameter vector $\bar{p}_{i+1}$ is found by minimizing, in the least-squares sense, the error between the target expected cost over the infinite horizon, which is the sum between the measured reinforcement over the time interval and the expected cost based on the present cost model, $d(\bar{x}_t, K_i) + \bar{p}_i{}^T \bar{x}_{t+T_0}$, and the parameterized left hand side of (6.41). The solution can be obtained using batch least squares or the recursive least squares algorithms.

Both the online policy iteration and online value iteration algorithms are data-based approaches, which use reinforcement learning ideas to find the solution of the algebraic Riccati equation with sign definite quadratic term, that do not require explicit knowledge on the model of the drift dynamics of the controlled system.

In the following we formulate the iterative algorithms which provide the saddle point solution of the two-player zero-sum differential game in terms of iterations on Riccati equations with sign definite quadratic term. At every step these Riccati equations can be solved by means of one of the online reinforcement learning algorithms, namely online policy iteration or value iteration. The end results is an online

algorithm which leads to the saddle point solution of the differential game while neither of the two players uses any knowledge on the drift dynamics of the environment.

We will name the two players *controller player* and *disturbance player*. The solution of the game is found online while the game is played. We shall see that in the case of both algorithms only one of the payers is learning and optimizing his behavior strategy while the other is playing based on fixed policies. We shall say that *the learning player is "leading the game"* while his opponent is a passive player. The passive player will change his behavior policy only based on information regarding his opponent's optimal strategy. In this case the passive player will simply adopt his opponent's strategy as his own. From this perspective, depending on which one of the two players is leading the game, one obtains one or the other online algorithm.

6.3.2 Online policy iteration algorithm on the control policy

In this section we present the online approach to the solution of the zero-sum differential game by means of reinforcement learning. We shall see that in this case the reinforcement learning technique is employed only by the Controller. First we formulate algorithm 6.1. as an iteration on Riccati equations.

Algorithm 6.1 – A

1. Let $P_u^0 = 0$

2. Solve online the Riccati equation

$P_u^1 A + A^T P_u^1 - P_u^1 B_2 B_2{}^T P_u^1 + C^T C = 0$.

Let $Z_u^1 = P_u^1$.

3. For $i \geq 2$ solve online the Riccati equation

$$0 = Z_u^i (A + B_1 B_1^T P_u^{i-1} - B_2 B_2^T P_u^{i-1}) + (A + B_1 B_1^T P_u^{i-1} - B_2 B_2^T P_u^{i-1})^T Z_u^i$$
$$- Z_u^i B_2 B_2^T Z_u^i + Z_u^{i-1} B_1 B_1^T Z_u^{i-1}$$

$$P_u^i = P_u^{i-1} + Z_u^i .$$

At every step the Riccati equations can be solved using the online data-based approaches reviewed in section 6.3.1 without using exact knowledge on the drift term in the system dynamics.

Explicitly one can write

Algorithm 6.1 – B

1. Let $P_u^0 = 0$
2. Let $K_u^{0(0)}$ be such that $A + B_2 K_u^{0(0)}$ is Hurwitz, let *k*=0
    a. solve
    $$P_u^{0(k+1)} (A + B_2 K_u^{0(k)}) + (A + B_2 K_u^{0(k)})^T P_u^{0(k+1)} + (K_u^{0(k)})^T K_u^{0(k)} + C^T C = 0$$
    b. update $K_u^{0(k+1)} = -B_2^T P_u^{0(k+1)}$, $k = k + 1$
    c. until $\left\| P_u^{0(k)} - P_u^{0(k-1)} \right\| < \varepsilon$.
3. $P_u^1 = P_u^{0(k)}$, $Z_u^1 = P_u^1$
4. For $i \geq 2$
    a. let $K_u^{i-1(0)}$ be such that $A + B_1 B_1^T P_u^{i-1} + B_2 K_u^{i-1(0)}$ is Hurwitz, let *k*=0
    b. solve
    $$Z_u^{i(k+1)} (A + B_1 B_1^T P_u^{i-1} + B_2 K_u^{i-1(k)}) + (A + B_1 B_1^T P_u^{i-1} + B_2 K_u^{i-1(k)})^T Z_u^{i(k+1)}$$
    $$+ (K_u^{i-1(k)})^T K_u^{i-1(k)} + Z_u^{i-1} B_1 B_1^T Z_u^{i-1} = 0$$

c. update $K_u^{i-1(k+1)} = K_u^{i-1(k)} - B_2{}^T Z_u^{i(k+1)}$, $k = k+1$

d. until $\left\| Z_u^{i(k)} - Z_u^{i(k-1)} \right\| < \varepsilon$.

5. $Z_u^i = Z_u^{i(k)}$, $P_u^i = P_u^{i-1} + Z_u^i$

6. until $\left\| P_u^i - P_u^{i-1} \right\| < \varepsilon_P$.

From the perspective of two-player zero-sum games, the algorithm translates as follows:

1. Let the initial disturbance policy be zero, $w = 0$.
2. Let $K_u^{0(0)}$ be a stabilizing control policy for the system (6.1) with zero disturbance $w = 0$, and let *k*=0.
    a. Find the value associated with the stabilizing controller $K_u^{0(k)}$;
    b. update the control policy, $k = k+1$; (Note that the new controller will have a higher value.)
    c. until the controller with the highest value (minimum cost) has been obtained.
3. Update the disturbance policy using the gain of the control policy.
4. For $i \geq 2$
    a. Let $K_u^{i-1(0)}$ be a stabilizing policy for the system (6.1) with disturbance policy $w = B_1{}^T P_u^{i-1} x$, let *k*=0
        i. find the added value associated with the change in the control policy

ii. update the control policy, $k = k + 1$

iii. until the controller with the highest value has been obtained.

5. Go to step 3 until the control policy and disturbance policy have the same gain.

Concisely the game is played as follows.

1. The game starts while the disturbance player does not play.
2. The controller player plays the game without opponent and uses reinforcement learning to find the optimal behavior which minimizes his costs; then informs his opponent on his new behavior policy.
3. The disturbance player starts playing using the behavior policy of his opponent.
4. The controller player corrects iteratively his behavior using reinforcement knowledge such that his costs are again minimized; then informs his opponent on his new behavior policy.
5. The two players execute successively steps 3 and 4 until the controller player can no longer lower his costs by changing his behavior policy. The saddle point equilibrium has been obtained.

The online setting of the algorithm is given next.

Algorithm 6.1 – C

1. Let $P_u^0 = 0$
2. Let $K_u^{0(0)}$ be such that $A + B_2 K_u^{0(0)}$ is Hurwitz, let *k*=0

    a. solve online

$$x_t^T P_u^{0(k+1)} x_t = \int_t^{t+T_0} x_\tau^T (C^T C + K_u^{0(k)T} R K_u^{0(k)}) x_\tau d\tau + x_{t+T_0}^T P_u^{0(k+1)} x_{t+T_0}$$

b. update $K_u^{0(k+1)} = -B_2{}^T P_u^{0(k+1)}$, $k = k+1$

c. until $\left\| P_u^{0(k)} - P_u^{0(k-1)} \right\| < \varepsilon$.

3. $P_u^1 = P_u^{0(k)}$, $Z_u^1 = P_u^1$

4. For $i \geq 2$

a. let $K_u^{i-1(0)}$ be such that $A_u^{i-1(0)} = A + B_1 B_1{}^T P_u^{i-1} + B_2 K_u^{i-1(0)}$ is Hurwitz, let k=0. Denote $A_u^{i-1(k)} = A + B_1 B_1{}^T P_u^{i-1} + B_2 K_u^{i-1(k)}$ and let x_τ be the solution of $\dot{x} = A_u^{i-1(k)} x$ with initial condition x_t over the interval $[t, t+T_0]$.

b. solve online for the value of $Z_u^{i(k+1)}$ using either

i. policy iteration

$$x_t{}^T Z_u^{i(k+1)} x_t = \int_t^{t+T_0} x_\tau{}^T (Z_u^{i-1} B_1 B_1{}^T Z_u^{i-1} + K_u^{i-1(k)T} K_u^{i-1(k)}) x_\tau d\tau + x_{t+T_0}{}^T Z_u^{i(k+1)} x_{t+T_0}$$

ii. or value iteration

$$x_t{}^T Z_u^{i(k+1)} x_t = \int_t^{t+T_0} x_\tau{}^T (Z_u^{i-1} B_1 B_1{}^T Z_u^{i-1} + K_u^{i-1(k)T} K_u^{i-1(k)}) x_\tau d\tau + x_{t+T_0}{}^T Z_u^{i(k)} x_{t+T_0}$$

c. update $K_u^{i-1(k+1)} = K_u^{i-1(k)} - B_2{}^T Z_u^{i(k+1)}$, $k = k+1$

d. until $\left\| Z_u^{i(k)} - Z_u^{i(k-1)} \right\| < \varepsilon$.

5. $Z_u^i = Z_u^{i(k)}$, $P_u^i = P_u^{i-1} + Z_u^i$

6. until $\left\| P_u^i - P_u^{i-1} \right\| < \varepsilon_P$.

6.3.3 Online policy iteration algorithm on the disturbance policy

This section introduces a second online approach to the solution of the zero-sum differential game by means of reinforcement learning. From a game theoretic perspective, in this case the reinforcement learning technique which leads to the saddle point solution of the differential game is employed only by the Disturbance player. First we give formulation of algorithm 6.2. as an iteration on Riccati equations.

Algorithm 6.2 – A

1. Let P_w^0 be such that $A - B_2 B_2^T P_w^0$ and $A - B_2 B_2^T P_w^0 + B_1 B_1^T \Pi$ are Hurwitz and $P_w^0 \geq \Pi$
2. For every $i \geq 1$ solve online the Riccati equation

   $$P_w^i (A - B_2 B_2{}^T P_w^{i-1}) + (A - B_2 B_2{}^T P_w^{i-1})^T P_w^i + P_w^i B_1 B_1{}^T P_w^i + P_w^{i-1} B_2 B_2{}^T P_w^{i-1} + C^T C = 0$$

   using the reinforcement learning technique which solves introduced in Chapter 2 making use of the Policy Iteration algorithm given in Lemma 6.4.

At every step the Riccati equations can be solved using the online data-based approaches reviewed in section 6.3.1 without using exact knowledge on the drift term in the system dynamics.

Explicitly one can write

Algorithm 6.2 – B

1. Let P_w^0 be such that $A - B_2 B_2^T P_w^0$ and $A - B_2 B_2^T P_w^0 + B_1 B_1^T \Pi$ are Hurwitz and $P_w^0 \geq \Pi$
2. For $i \geq 2$

3. Let $L_w^{i(0)} = 0$, as $A - B_2 B_2^T P_w^{i-1}$ is Hurwitz, k=0

    a. solve the Lyapunov equation

$$P_w^{i(k)}(A - B_2 B_2^T P_w^{i-1} + B_1 L^{i(k-1)}) + (A - B_2 B_2^T P_w^{i-1} + B_1 L^{i(k-1)})^T P_w^{i(k)} + Q^i - (L^{i(k-1)})^T L^{i(k-1)} = 0$$

    b. update $L^{i(k)} = B_1^T P_w^{i(k)}$, $k = k + 1$

    c. until $\left\| P_w^{i(k)} - P_w^{i(k-1)} \right\| < \varepsilon$.

4. until $\left\| P_w^i - P_w^{i-1} \right\| < \varepsilon_P$.

From the perspective of two-player zero-sum games, the game is played as follows.

1. The controller player starts by selecting a stabilizing control policy such that initial requirements are satisfied.
2. The disturbance player plays the game to find the optimal behavior which maximizes his long term goal; then informs his opponent on his new behavior policy.
3. The controller player starts playing using the behavior policy of his opponent having guarantees that this new policy is a stabilizing one.
4. The two players execute successively steps 2 and 3 until the disturbance player can no longer increase his long term benefits by changing his behavior policy. This means that the saddle point equilibrium has been obtained.

The online version of the algorithm is given next.

Algorithm 6.2 – C

1. Let P_w^0 be such that $A - B_2 B_2^T P_w^0$ and $A - B_2 B_2^T P_w^0 + B_1 B_1^T \Pi$ are Hurwitz and $P_w^0 \geq \Pi$
2. For $i \geq 2$
3. Let $L_w^{i(0)} = 0$, as $A - B_2 B_2{}^T P_w^{i-1}$ is Hurwitz, *k*=0. Denote with x_τ the solution to $\dot{x} = (A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{i(k-1)})x$ over the time interval $[t, t+T_0]$ given the initial condition x_t.
    a. solve online for the value of $P_w^{i(k)}$ using either
        i. policy iteration

$$x_t{}^T P_w^{i(k)} x_t = \int_t^{t+T_0} x_\tau{}^T (Q^i - (L^{i(k-1)})^T L^{i(k-1)}) x_\tau d\tau + x_{t+T_0}{}^T P_w^{i(k)} x_{t+T_0}$$

        ii. or value iteration

$$x_t{}^T P_w^{i(k)} x_t = \int_t^{t+T_0} x_\tau{}^T (Q^i - (L^{i(k-1)})^T L^{i(k-1)}) x_\tau d\tau + x_{t+T_0}{}^T P_w^{i(k-1)} x_{t+T_0}$$

    b. update $L^{i(k)} = B_1{}^T P_w^{i(k)}$, $k = k+1$
    c. until $\left\| P_w^{i(k)} - P_w^{i(k-1)} \right\| < \varepsilon$.
4. until $\left\| P_w^i - P_w^{i-1} \right\| < \varepsilon_P$.

6.4 Conclusion

This chapter introduces an online data-based approach that makes use of reinforcement learning techniques to provide online solution to the two-player zero-sum differential game with linear dynamics. The result is based on two existing algorithms which involve iterations on Riccati equations to build a sequence of controllers (and respectively disturbance policies) which converges monotonically to the state-feedback saddle point solution of the two-player zero-sum differential game.

The Riccati equation appearing at each step of the iteration can be solved using online measured data using either the online policy iteration algorithm presented in Chapter 2 or the value iteration algorithm discussed in Chapter 5. In this way the H-infinity state-feedback optimal controller, or the solution of the differential game, can be obtained online without using exact knowledge on the drift dynamics of the system.

The chapter provides two secondary achievements. First it gives formulation of the two iterative algorithms such that the duality of the two approaches became obvious. Second it discusses the two algorithms from the perspective of game theory.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis have been developed algorithms which use reinforcement learning ideas to solve in an online manner continuous-time state-feedback optimal control problems. The algorithms make use of discrete-time state information and only partial knowledge regarding the system dynamics (i.e. exact knowledge on the drift term in the system dynamics is not required).

1. First a reinforcement learning based procedure has been developed to solve online the continuous-time LQR problem with infinite horizon cost.

2. A second algorithm generalizes the first results to obtain a suboptimal controller for affine in the inputs nonlinear systems. In this case the algorithm provides local solution to the continuous-time Hamilton-Jacobi-Bellman equation without using knowledge on the drift term part of the system dynamics.

3. The third result is a new continuous-time formulation for the policy iteration algorithm; which results in a new online data-based approach to optimal control for nonlinear systems.

4. The generalized policy iteration algorithm for continuous-time systems is then given. It is in fact a spectrum of algorithms which provides a bridge between

continuous-time policy iteration and continuous-time value iteration (heuristic dynamic programming).

5. The continuous-time value iteration algorithm is given. An analysis for the case of LQR is provided. This new online approach to learning the optimal control policy does not need initialization with a stabilizing controller thus reducing even further the amount of knowledge required for solving the optimal control problem.

6. Online reinforcement learning based approaches to the saddle point solution of linear differential zero-sum games with infinite horizon quadratic indices have also been given.

The results presented in this thesis:

- bring reinforcement learning ideas into control systems theory allowing formulation of new adaptive optimal control strategies for systems with continuous-time dynamics,
- provide connection between the reinforcement learning methods, which solve the continuous-time optimal control problems in an online manner based on measured data, and known iterative techniques which provide offline solution for the same problem based on complete and exact knowledge on the system dynamics,
- provide new arguments for the idea that approximate dynamic programming, previously developed mainly for systems with discrete-time dynamics, is a framework independent approach to optimization.

The following are some of the directions for continuation of this work.

1. Providing proof of convergence for the value iteration (VI) algorithm - a generalized policy iteration variant which does not require initial stabilizing control policy.
2. Giving a continuous-time formulation for the Q-function and making use of it to develop model-free online adaptive optimal controllers.
3. Developing online adaptive optimal controllers which use reinforcement learning principles to find the H-infinity solution for nonlinear systems
4. Online adaptive optimal controllers for systems with periodic dynamics.
5. Online adaptive optimal controllers using output feedback.

APPENDIX A

PROOFS FOR SELECTED RESULTS

Proofs for selected results from Chapter 3

Lemma 3.1 *Solving for* $V^{\mu^{(i)}}$ *in equation* (3.9) *is equivalent with finding the solution of*

$$0=r(x,\mu^{(i)}(x))+(\nabla V_x^{\mu^{(i)}})^T(f(x)+g(x)\mu^{(i)}(x)),\ V^{\mu^{(i)}}(0)=0. \qquad (3.12)$$

Proof Since $\mu^{(i)}\in\Psi(\Omega)$, then $V^{\mu^{(i)}}\in C^1(\Omega)$, defined as $V^{\mu^{(i)}}(x(t))=\int_t^\infty r(x(s),\mu^{(i)}(x(s)))ds$, is a Lyapunov function for the system $\dot{x}(t)=f(x(t))+g(x(t))\mu^{(i)}(x(t))$. $V^{\mu^{(i)}}\in C^1(\Omega)$ satisfies

$$(\nabla V_x^{\mu^{(i)}})^T(f(x)+g(x)\mu^{(i)}(x))=-r(x(t),\mu^{(i)}(x(t))) \qquad (A.1)$$

with $r(x(t),\mu^{(i)}(x(t)))>0;\ x(t)\neq 0$. Integrating (A.1) over the time interval $[t,t+T]$ one obtains

$$V^{\mu^{(i)}}(x(t))=\int_t^{t+T} r(x(s),\mu^{(i)}(x(s)))ds+V^{\mu^{(i)}}(x(t+T)). \qquad (A.2)$$

This means that the unique solution of (3.12), $V^{\mu^{(i)}}$, satisfies also (A.2).

To complete de proof uniqueness of solution of equation (A.2) must be established. The proof is by contradiction.

Thus, assume that there exists another cost function $V\in C^1(\Omega)$ which satisfies (14) with the end condition $V(0)=0$. This cost function also satisfies $\dot{V}(x(t))=-r(x(t),\mu^{(i)}(x(t)))$. Subtracting this from (A.2) we obtain

$$\left(\frac{d[V(x(t))-V^{\mu^{(i)}}(x(t))]}{dx}\right)^T\dot{x}=\left(\frac{d[V(x(t))-V^{\mu^{(i)}}(x(t))]}{dx}\right)^T(f(x(t))+g(x(t))\mu^{(i)}(x(t)))=0 \tag{A.3}$$

which must hold for any x on the system trajectories generated by the stabilizing policy $\mu^{(i)}$. Thus $V(x(t))=V^{\mu^{(i)}}(x(t))+c$. As this relation must hold also for $x(t)=0$ then $V(0)=V^{\mu^{(i)}}(0)+c\Rightarrow 0=c$ and thus $V(x(t))=V^{\mu^{(i)}}(x(t))$, *i.e.* equation (3.9) has a unique solution which is equal with the unique solution of (3.12). □

Lemma 3.3 *Let* $\mu(x)\in\Psi(\Omega)$ *such that* $f(x)+g(x)\mu(x)$ *is asymptotically stable. Given that the set* $\{\phi_j\}_1^N$ *is linearly independent then* $\exists T>0$ *such that* $\forall x(t)\in\Omega-\{0\}$, *the set* $\{\bar{\phi}_j(x(t),T)=\phi_j(x(t+T))-\phi_j(x(t))\}_1^N$ *is also linearly independent.*

Proof The proof is by contradiction.

The vector field $\dot{x}=f(x)+g(x)\mu(x)$ is asymptotically stable. Denote with $\eta(\tau;x(t),\mu), x(t)\in\Omega$ the system trajectories obtained using the policy $\mu(x)$ for any $x(t)\in\Omega$. Then, along the system trajectories, we have that

$$\phi(x(t+T))-\phi(x(t))=\int_t^{t+T}\nabla\phi_x{}^T(f+g\mu)(\eta(\tau;x(t),\mu))d\tau\ . \tag{A.4}$$

Suppose that the result is not true, then $\forall T>0$ there exists a nonzero constant vector $c\in\mathbb{R}^N$ such that $\forall x(t)\in\Omega$ $c^T[\phi(x(t+T))-\phi(x(t))]\equiv 0$. This implies that $\forall T>0$, $c^T\int_t^{t+T}\nabla\phi_x{}^T(f+g\mu)(\varphi(\tau;x(t),\mu))d\tau\equiv 0$ and thus, $\forall x(t)\in\Omega$,

$c^T \nabla \phi_x{}^T (f+g\mu)(\varphi(\tau; x(t), \mu)) \equiv 0$. This means that $\left\{\nabla \phi_j{}^T (f+gu)\right\}_1^N$ is not linearly independent contradicting Lemma 3.2. Thus, $\exists T>0$ such that $\forall x(t_0) \in \Omega$ the set $\left\{\bar{\phi}_j(x(t_0), T)\right\}_1^N$ is also linearly independent. □

Corollary 3.2 (Admissibility of $\mu_L{}^{(i)}(x)$) $\exists L_0$ *such that* $\forall L > L_0, \mu_L^{(i)} \in \Psi(\Omega)$.

Proof Consider the function $V^{\mu^{(i)}}$, which is a Lyapunov function for the system (3.1) with control policy $\mu^{(i)}$. Taking derivative of $V^{\mu^{(i)}}$ along the trajectories generated by the controller $\mu_L{}^{(i+1)}(x)$ one obtains

$$\dot{V}^{\mu^{(i)}} = (\nabla V_x^{\mu^{(i)}})^T (f(x) + g(x)\mu_L{}^{(i+1)}(x)). \tag{A.5}$$

We also have that $\dot{V}^{\mu^{(i)}} = (\nabla V_x^{\mu^{(i)}})^T (f(x) + g(x)\mu^{(i)}(x)) = -Q(x) - (\mu^{(i)}(x))^T R\mu^{(i)}(x)$ and thus $(\nabla V_x^{\mu^{(i)}})^T f(x) = -(\nabla V_x^{\mu^{(i)}})^T g(x)\mu^{(i)}(x) - Q(x) - (\mu^{(i)}(x))^T R\mu^{(i)}(x)$. With this, (A.5) becomes

$$\dot{V}^{\mu^{(i)}} = -Q(x) - (\mu^{(i)}(x))^T R\mu^{(i)}(x) - (\nabla V_x^{\mu^{(i)}})^T g(x)(\mu^{(i)}(x) - \mu_L{}^{(i+1)}(x)). \tag{A.6}$$

Using the controller update $\mu^{(i+1)}(x) = -\frac{1}{2} R^{-1} g(x)^T \nabla V_x^{\mu^{(i)}}$ we can write $(\nabla V_x^{\mu^{(i)}})^T g(x) = 2R\mu^{(i+1)}(x)$. Thus (A.6) becomes

$$\begin{aligned} \dot{V}^{\mu^{(i)}} &= -Q - (\mu^{(i)})^T R\mu^{(i)} - 2\mu^{(i+1)} R(\mu^{(i)} - \mu_L{}^{(i+1)}) \\ &= -Q - (\mu^{(i)} - \mu^{(i+1)})^T R(\mu^{(i)} - \mu^{(i+1)}) - (\mu_L{}^{(i+1)})^T R\mu_L{}^{(i+1)} + \\ &\quad + (\mu^{(i+1)} - \mu_L{}^{(i+1)})^T R(\mu^{(i+1)} - \mu_L{}^{(i+1)}) \end{aligned} \tag{A.7}$$

Since $\sup_{x\in\Omega}\left|\mu_L{}^{(i)}(x)-\mu^{(i)}(x)\right|\to 0$ as $L\to\infty$ then there exists a L_0 such that $\forall L>L_0$, $\dot{V}^{\mu^{(i)}}<0$, which means that $V^{\mu^{(i)}}$ is a Lyapunov function for the system with control policy $\mu_L{}^{(i+1)}(x)$, which proves the corollary. □

Proofs for selected results from Chapter 4

Corollary 4.1 *$T'_\mu :\mathrm{X}^P\to\mathrm{X}^P$ is a contraction map on X^P.*

Proof The fixed point of $T'_\mu\ P=P_1^\mu\triangleq M^\mu+(A_d^{T_0})^T PA_d^{T_0}$ is the unique positive definite solution of the discrete-time Lyapunov equation

$$P^\mu=M^\mu+(A_d^{T_0})^T P^\mu A_d^{T_0}. \tag{A.8}$$

Using the recursion k times, with $P_0^\mu=P$ we have

$$P_k^\mu=M^\mu+(A_d^{T_0})^T P_{k-1}^\mu A_d^{T_0} \tag{A.9}$$

Subtracting (A.8) from (A.9)

$$P^\mu-P_k^\mu=(A_d^{T_0})^T(P^\mu-P_{k-1}^\mu)A_d^{T_0} \tag{A.10}$$

Using the norm operator (A.10) becomes

$$\left\|P^\mu-P_k^\mu\right\|_\rho\le\left\|A_d^{T_0}\right\|_\rho^2\left\|P^\mu-P_{k-1}^\mu\right\|_\rho \tag{A.11}$$

Since $A_d^{T_0}$ is a discrete version of the closed loop continuous-time system stabilized by the state feedback control policy $\mu(x)=-K^\mu x$, then $0<\rho(A_d^{T_0})<1$. Thus T'_μ is a contraction map on $(\mathrm{X}^P,\|\ \|_\rho)$ □

Proofs for selected results from Chapter 6

Lemma 6.3 *Given real matrices* A, B_1, B_2, C *with compatible dimensions,* $P \in \mathbb{R}^{n\times n}$ *and* $Z \in \mathbb{R}^{n\times n}$ *satisfying*

$$(A - B_2B_2^T P + B_1B_1^T P)^T Z + Z(A - B_2B_2^T P + B_1B_1^T P) - F(P) - ZB_1B_1^T Z = 0 \quad (6.29)$$

and $\Pi > 0$ *a stabilizing solution of the GARE (6.4)*

a. If $A - B_2B_2^T P + B_1B_1^T \Pi$ *is Hurwitz then* $\Pi \le P - Z$.

b. If $\Pi \le P - Z$ *then* $A - B_2B_2^T (P - Z) + B_1B_1^T \Pi$.

Proof Adding (6.29) with (6.20) and rearranging one writes

$$\begin{aligned}&(A - B_2B_2^T P + B_1B_1^T \Pi)^T (P - Z - \Pi) + (P - Z - \Pi)(A - B_2B_2^T P + B_1B_1^T \Pi)\\ &= -(P - Z - \Pi)B_1B_1^T (P - Z - \Pi) - (P - \Pi)B_2B_2^T (P - \Pi)\end{aligned} \quad (A.12)$$

As, by assumption, $(A - B_2B_2^T P + B_1B_1^T \Pi)$ - Hurwitz then $P - Z \ge \Pi$.

Using the notation $W = \begin{bmatrix} B_2^T (P - Z - \Pi) \\ B_1^T (P - Z - \Pi) \\ B_2^T Z \end{bmatrix}$, (A.12) can be brought to the form

$$\begin{aligned}&(A - B_2B_2^T (P - Z) + B_1B_1^T \Pi)^T (P - Z - \Pi) + (P - Z - \Pi)(A - B_2B_2^T (P - Z) + B_1B_1^T \Pi)\\ &= -W^T W\end{aligned} \quad (A.13)$$

One can see that $(W, (A - B_2B_2^T (P - Z) + B_1B_1^T \Pi))$ is observable since
$A - B_2B_2^T (P - Z) + B_1B_1^T \Pi + \begin{bmatrix} B_2 & 0 & 0 \end{bmatrix} W = A - B_2B_2^T \Pi + B_1B_1^T \Pi$ is Hurwitz.
As $P - Z - \Pi \ge 0$, $W^T W \ge 0$ and $(W, (A - B_2B_2^T (P - Z) + B_1B_1^T \Pi))$ - observable then
$A - B_2B_2^T (P - Z) + B_1B_1^T \Pi$ is Hurwitz. □

Lemma 6.4 *Let* $Q^i \triangleq C^T C + P_w^i B_2 B_2^T P_w^i$, $A - B_2 B_2^T P_w^i$ *- Hurwitz,* $(A - B_2 B_2^T P_w^i, B_1)$*-controllable and* $(\sqrt{Q^i}, A - B_2 B_2^T P_w^i)$ *- detectable (i.e. all its unobservable modes are in the left half of the complex plane).*

Then the unique positive definite solution of

$$P_w^i (A - B_2 B_2^T P_w^{i-1})^T + (A - B_2 B_2^T P_w^{i-1}) P_w^i + C^T C + P_w^i B_1 B_1^T P_w^i + P_w^{i-1} B_2 B_2^T P_w^{i-1} = 0$$

such that $(A - B_2 B_2^T P_w^{i-1} + B_1 B_1^T P_w^i)$ *is Hurwitz, can be determined using the policy iteration algorithm*

a) $$P_w^{i(k)} (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-1)}) + (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-1)})^T P_w^{i(k)} + Q^i - (L^{(k-1)})^T L^{(k-1)} = 0 \quad (6.31)$$

b) $$L^{(k)} = B_1^T P_w^{i(k)} \quad (6.32)$$

where $L^{(0)} = 0$.

Proof For $k = 1$ the first step in the algorithm can be executed and it is equivalent to solving the Lyapunov equation $P_w^{i(1)} (A - B_2 B_2^T P_w^{i-1}) + (A - B_2 B_2^T P_w^{i-1})^T P_w^{i(1)} + Q^i = 0$. Thus $P_w^{i(1)} \geq 0$.

Assume that (a) is satisfied at step *k*-1,

$$P_w^{i(k-1)} (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-2)}) + (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-2)})^T P_w^{i(k-1)} + \\ + Q^i - (L^{(k-2)})^T L^{(k-2)} = 0$$

then it can be written as

$$P_w^{i(k-1)} (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-2)} + B_1 L^{(k-1)} - B_1 L^{(k-2)}) + \\ + (A - B_2 B_2^T P_w^{i-1} + B_1 L^{(k-2)} + B_1 L^{(k-1)} - B_1 L^{(k-2)})^T P_w^{i(k-1)} + Q^i - (L^{(k-2)})^T L^{(k-2)} = 0$$

This is

$$P_w^{i(k-1)}(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^TP_w^{i(k-1)}$$
$$+P_w^{i(k-1)}B_1(L^{(k-2)}-L^{(k-1)})+(L^{(k-2)}-L^{(k-1)})^TB_1{}^TP_w^{i(k-1)}+Q^i-(L^{(k-2)})^TL^{(k-2)}=0$$

which, making use $L^{(k)}=B_1{}^TP_w^{i(k)}$, becomes

$$P_w^{i(k-1)}(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^TP_w^{i(k-1)}$$
$$-(L^{(k-2)}-L^{(k-1)})^T(L^{(k-2)}-L^{(k-1)})+Q^i-(L^{(k-1)})^TL^{(k-1)}=0$$

.

Subtracting the equation at step *k*

$$P_w^{i(k)}(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^TP_w^{i(k)}+Q^i-(L^{(k-1)})^TL^{(k-1)}=0$$

from this last result, one obtains

$$(P_w^{i(k-1)}-P_w^{i(k)})(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^T(P_w^{i(k-1)}-P_w^{i(k)})$$
$$-(L^{(k-2)}-L^{(k-1)})^T(L^{(k-2)}-L^{(k-1)})=0$$

As $A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)}$ is Hurwitz then $P_w^{i(k-1)}-P_w^{i(k)}\le 0$.

Thus the sequence $\{P_w^{i(k-1)}\}$ is monotonically increasing and positive definite since its first element $P_w^{i(1)}\ge 0$.

Next we will show that the sequence is also upper bounded. We rewrite

$$P_w^i(A-B_2B_2{}^TP_w^{i-1})^T+(A-B_2B_2{}^TP_w^{i-1})P_w^i+Q+P_w^iB_1B_1{}^TP_w^i+P_w^{i-1}B_2B_2{}^TP_w^{i-1}=0$$

as

$$P_w^i(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^TP_w^i$$
$$+(P_w^i-P_w^{i(k-1)})^TB_1B_1{}^T(P_w^i-P_w^{i(k-1)})+Q^i-P_w^{i(k-1)}B_1B_1{}^TP_w^{i(k-1)}=0$$

.

Subtracting

$$P_w^{i(k)}(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})+(A-B_2B_2{}^TP_w^{i-1}+B_1L^{(k-1)})^TP_w^{i(k)}+Q^i-(L^{(k-1)})^TL^{(k-1)}=0$$

one obtains

$$(P_w^i - P_w^{i(k)})(A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k-1)}) + (A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k-1)})^T (P_w^i - P_w^{i(k)})$$
$$= -(P_w^i - P_w^{i(k-1)})^T B_1 B_1{}^T (P_w^i - P_w^{i(k-1)})$$

As $A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k-1)}$ is Hurwitz then $P_w^i - P_w^{i(k)} \geq 0$.

It is now left to prove that $A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)}$ is Hurwitz.

One can write

$$(P_w^i - P_w^{i(k)})(A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)}) + (A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)})^T (P_w^i - P_w^{i(k)})$$
$$= -(P_w^i - P_w^{i(k-1)})^T B_1 B_1{}^T (P_w^i - P_w^{i(k-1)}) - (P_w^i - P_w^{i(k)}) B_1 (L^{(k-1)} - L^{(k)})$$
$$-(L^{(k-1)} - L^{(k)})^T B_1{}^T (P_w^i - P_w^{i(k)})$$

which is

$$(P_w^i - P_w^{i(k)})(A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)}) + (A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)})^T (P_w^i - P_w^{i(k)})$$
$$= -(P_w^i - P_w^{i(k)})^T B_1 B_1{}^T (P_w^i - P_w^{i(k)}) - (L^{(k-1)} - L^{(k)})^T (L^{(k-1)} - L^{(k)})$$

.

Using the notation $W_w^{i(k)} = \begin{bmatrix} B_1{}^T (P_w^i - P_w^{i(k)}) \\ B_1{}^T (P_w^{i(k-1)} - P_w^{i(k)}) \end{bmatrix}$ then

$(P_w^i - P_w^{i(k)})(A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)}) + (A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)})^T (P_w^i - P_w^{i(k)}) = -(W_w^{i(k)})^T W_w^{i(k)}$

.

$(W_w^{i(k)}, A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)})$ is detectable since

$A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)} + F W_w^{i(k)} = A - B_2 B_2{}^T P_w^{i-1} + B_1 B_1{}^T P_w^i$ for $F = [B_1 \quad 0]$ and

$A - B_2 B_2{}^T P_w^{i-1} + B_1 B_1{}^T P_w^i$ is Hurwitz since P_w^i is the stabilizing solution for the equation

$P_w^i (A - B_2 B_2{}^T P_w^{i-1})^T + (A - B_2 B_2{}^T P_w^{i-1}) P_w^i + Q + P_w^i B_1 B_1{}^T P_w^i + P_w^{i-1} B_2 B_2{}^T P_w^{i-1} = 0$.

Since $P_w^i - P_w^{i(k)} \geq 0$ and $(W_w^{i(k)}, A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)})$ is detectable then $A - B_2 B_2{}^T P_w^{i-1} + B_1 L^{(k)}$ is Hurwitz.

Consequently the iterations can be continued, and $P_w^{i(k+1)} \geq 0$ can be determined.

The iteration of the two equations generates the monotonically increasing and upper bounded sequence of positive definite matrices $\{P_w^{i(k-1)}\}$. Let $P_w^i = \lim_{k\to\infty} P_w^{i(k)}$ and thus the controller is $L_w^i = B_1^T P_w^i$. Using this controller in equation $P_w^i(A - B_2 B_2^T P_w^{i-1} + B_1 L_w^i) + (A - B_2 B_2^T P_w^{i-1} + B_1 L_w^i)^T P_w^i + Q^i - (L_w^i)^T L_w^i = 0$ shows that P_w^i is a positive definite solution (unique and stabilizing such that $A - B_2 B_2^T P_w^{i-1} + B_1 B_1^T P_w^i$ is Hurwitz) of

$$P_w^i(A - B_2 B_2^T P_w^{i-1})^T + (A - B_2 B_2^T P_w^{i-1})P_w^i + C^T C + P_w^i B_1 B_1^T P_w^i + P_w^{i-1} B_2 B_2^T P_w^{i-1} = 0$$

Next we show that $A - B_2 B_2^T P_w^i$ is Hurwitz. We will show that $V(x) = x^T P_w^i x$ is a Lyapunov function for the system $\dot{x} = (A - B_2 B_2^T P_w^i)x$. Thus one writes

$$\begin{aligned} &P_w^i(A - B_2 B_2^T P_w^i) + (A - B_2 B_2^T P_w^i)^T P_w^i \\ &\quad = -C^T C - P_w^i B_2 B_2^T P_w^i - P_w^i B_1 B_1^T P_w^i - (P_w^{i-1} - P_w^i) B_2 B_2^T (P_w^{i-1} - P_w^i)^{'} \end{aligned}$$

which proves that $A - B_2 B_2^T P_w^i$ is Hurwitz.

We finally note that since P_w^i is a storage function for the system $\dot{x} = (A - B_2 B_2^T P_w^i)x + B_1 d$ then the available storage function corresponding to this system, which solves the Riccati equation

$$P_w^{i+1}(A - B_2 B_2^T P_w^i)^T + (A - B_2 B_2^T P_w^i)P_w^{i+1} + C^T C + P_w^{i+1} B_1 B_1^T P_w^{i+1} + P_w^i B_2 B_2^T P_w^i = 0$$

must satisfy $P_w^{i+1} \leq P_w^i$. □

REFERENCES

[1] Abu-Khalaf M., Lewis F. L. and Huang J. (2006). Policy Iterations and the Hamilton-Jacobi-Isaacs Equation for H-infinity State Feedback Control with Input Saturation, *IEEE Trans. on Automatic Control*, 51(12), 1989- 1995.

[2] Abu-Khalaf M. and F. L. Lewis, (2005). Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach, *Automatica*, 41(5), 779-791.

[3] Al-Tamimi A., Abu-Khalaf M. and Lewis F. L. (2007). Model-Free Q-Learning Designs for Discrete-Time Zero-Sum Games with Application to H-Infinity Control, *Automatica*, 43(3), 473 – 482.

[4] A. Al-Tamimi, M. Abu-Khalaf, F. L. Lewis, (2007). Adaptive Critic Designs for Discrete-Time Zero-Sum Games With Application to H-infinity Control, *IEEE Trans. on Sys., Man. and Cyb –B*, 37(1).

[5] Baird L. C. III, (1994). Reinforcement Learning in Continuous Time: Advantage Updating, *Proc. of ICNN*, Orlando FL.

[6] Balzer L. A. (1980). Accelerated Convergence of the Matrix Sign Function Method of Solving Lyapunov, Riccati and Other Equations, *Int. J. Control*, 32(6), 1076-1078.

[7] Banks H. T. and Ito K. (1991). A Numerical Algorithm for Optimal Feedback Gains in High Dimensional Linear Quadratic Regulator Problems, *SIAM J. Control Optimal*, 29(3), 499-515.

[8] Basar, T., Olsder, G. J. (1999). *Dynamic Noncooperative Game Theory*, 2nd ed., (Classics in Applied Mathematics; 23), SIAM.

[9] Beard, R., Saridis, G., and Wen, J. (1997). Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation. *Automatica*, *33*(12), 2159–2177.

[10] Bertsekas D. P. and Tsitsiklis J. N. (1996). *Neuro-Dynamic Programming*, Athena Scientific, MA.

[11] Byers R. (1987). Solving the Algebraic Riccati Equation with the Matrix Sign, *Linear Algebra and Its Applications*, 85, 267-279.

[12] Bradtke S. J., Ydestie B. E. and Barto A. G. (1994), Adaptive Linear Quadratic Control Using Policy Iteration, *Proc. of ACC*, 3475–3476.

[13] Brewer J. W. (1978). Kronecker Products and Matrix Calculus in System Theory, *IEEE Trans. on Circuit and System*, 25(9), 772–781.

[14] Callier, F. M., Desoer C. A. (1991). *Linear Systems Theory*, Springer-Verlag, New York.

[15] Cherfi, L., Abou-Kandil H., Bourles H. (2005). Iterative method for general algebraic Riccati equation, *Proc. ACSE'05*.

[16] Damm, T. (2004). *Rational Matrix Equations in Stochastic Control*, Springer-Verlag, Berlin, Germany.

[17] Doya, K. (2000). Reinforcement Learning In Continuous Time and Space. *Neural Computation*, 12(1), 219-245.

[18] Doya, K., Kimura, H., and Kawato, M. (2001). Neural Mechanisms of Learning and Control, *IEEE Control Systems Magazine*, 21(4), 42-54.

[19] Doyle J., Glover K., Khargonekar P. and Francis B. (1989). State-space solutions to standard H2 and H-infinity control problems, *IEEE Trans. Aut. Control*, 34, 831-847.

[20] Ferrari, S., Stengel R. (2002). An Adaptive Critic Global Controller, *Proceedings of the American Control Conference*, 2665-2670.

[21] Freeman R. A. and Kokotovic P. (1996). *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*, Birkhauser, Boston, MA.

[22] Guo C. H. and Lancaster P. (1998). Analysis and Modification of Newton's Method for Algebraic Riccati Equations, *Mathematics of Computation*, 67(223), 1089-1105.

[23] Hanselmann, T., Noakes, L., and Zaknich, A. (2007). Continuous-Time Adaptive Critics. *IEEE Transactions on Neural Networks*, 18(3), 631-647.

[24] Hasan M. A., Yang J. S. and Hasan A. A. (1999). A Method for solving the Algebraic Riccati and Lyapunov Equations using Higher Order Matrix Sign Function Algorithms, *Proc. of ACC*, 2345–2349.

[25] Hewer G. (1971). An Iterative Technique for the Computation of the Steady State Gains for the Discrete Optimal Regulator, *IEEE Trans. on Automatic Control*, 16(4), 382–384.

[26] Hornik, K., M. Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3, 551–560.

[27] Howard R. A. (1960). *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA.

[28] Huang, J., and Lin, C. F. (1995). Numerical approach to computing nonlinear H_∞ control laws. *Journal of Guidance, Control and Dynamics*, 18(5), 989–994.

[29] Kirk, D. E. (2004). *Optimal Control Theory – an introduction*, New York: Dover Pub. Inc., Mineola.

[30] Ioannou P. and Fidan B. (2006). *Adaptive Control Tutorial*, SIAM, Advances in Design and Control, PA.

[31] Kailath T. (1973). Some New Algorithms for Recursive Estimation in Constant Linear Systems, *IEEE Trans. on Information Theory*, 19(6), 750-760.

[32] Kleinman D. (1968). On an Iterative Technique for Riccati Equation Computations, *IEEE Trans. on Automatic Control*, 13(1), 114–115.

[33] Kolmogorov, A. N., and Fomin, S. V. (1999). *Elements of the Theory of Functions and Functional Analysis*, New York: Dover Pub. Inc., Mineola.

[34] Krstic M. and Deng H. (1998). *Stabilization of Nonlinear Uncertain Systems*, Springer.

[35] Landelius T. (1997). *Reinforcement Learning and Distributed Local Model Synthesis*, PhD Dissertation, Linkoping University, Sweden.

[36] Lanzon A., Feng Y., Anderson B. D. O, Rotkowitz M. (2008). Computing the Positive Stabilizing Solution to Algebraic Riccati Equations With an Indefinite Quadratic Term via a Recursive Method, *IEEE Trans. Aut. Control*, 53(10), 2280-2291.

[37] Laub A. J. (1979). A Schur Method for Solving Algebraic Riccati Equations, IEEE *Trans. on Automatic Control*, 24(6), 913–921.

[38] Leake, R. J., Liu, Ruey-Wen, (1967). Construction of Suboptimal Control Sequences, *J. SIAM Control*, 5(1), 54-63.

[39] Levine, D. S., Brown, V. R., Shirey V. T. eds. (2000). *Oscillations in Neural Systems*, Mahwah, NJ: Lawrence Erlbaum Associates Publ.

[40] Lewis F. L., Syrmos V. L. (1995). *Optimal Control*, John Wiley.

[41] Li Z. H. and Krstic M. (1997), Optimal design of adaptive tracking controllers for nonlinear systems, *Proc. of ACC*, 1191-1197.

[42] Liu, X., Balakrishnan S. N. (2000). Convergence Analysis of Adaptive Critic Based Optimal Control, *Proceedings of the American Control Conference*, 1929-1933.

[43] MacFarlane A. G. J. (1963), An Eigenvector Solution of the Optimal Linear Regulator Problem, *J. Electron. Contr*., 14, 643–654.

[44] Moris K. and Navasca C. (2006). Iterative Solution of Algebraic Riccati Equations for Damped Systems, *Proc. of CDC'06*, 2436–2440.

[45] Murray J. J., Cox C. J., Lendaris G. G. and Saeks R. (2002). Adaptive Dynamic Programming, *IEEE Trans. on Systems, Man and Cybernetics,* 32(2), 140–153.

[46] Nevistic, V., and Primbs, J. (1996). Constrained nonlinear optimal control: a converse HJB approach. Technical Report 96-021, California Institute of Technology.

[47] Potter J. E. (1966). Matrix quadratic solutions, *SIAM J. App. Math*., 14, 496–501.

[48] Prokhorov, D., and Wunsch, D. (1997). Adaptive critic designs. *IEEE Trans. on Neural Networks*, 8(5), 997-1007.

[49] Schultz, W. (2004). Neural coding of basic reward terms of animal learning theory, game theory, microeconomics and behavioral ecology, *Current Opinion in Neurobiology*, 14, 139-147.

[50] Schultz, W., Dayan, P., Read Montague, P. (1997). A Neural Substrate of Prediction and Reward, *Science*, 275, 1593-1599.

[51] Schultz, W., Tremblay, L., and Hollerman, J. R. (2000). Reward Processing in Primate Orbitofrontal Cortex and Basal Ganglia. *Cerebral Cortex*, 10, 272-283.

[52] Si J., Barto, A., Powell, W., Wunsch, D. (2004). *Handbook of Learning and approximate dynamic programming*, John Wiley, New Jersey.

[53] Stevens, B. L., Lewis, F. L. (2003), *Aircraft Control and Simulation*, Willey, 2nd Edition.

[54] Stoorvogel A. A., (1992). *The H-infinity Control Problem: A State Space Approach*, Prentice-Hall, New York.

[55] Sutton, R. (1988), Learning to predict by the method of temporal differences, *Machine Learning*, 3, 9-44.

[56] Sutton, R. S., Barto, A. G. (1998). *Reinforcement Learning – An Introduction*. Cambridge, MT: MIT Press.

[57] Sutton, R. S., Barto, A. G., Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 19-22.

[58] Tsitsiklis, J. N., (2002). On the convergence of optimistic policy iteration, *Journal of Machine Learning Research*, 3, 59-72.

[59] Van Der Schaft, A. J. (1992). L2-gain analysis of nonlinear systems and nonlinear state feedback H∞ control. *IEEE Transactions on Automatic Control*, 37(6), 770–784.

[60] Wang Y., Zhou R. and Wen C. (1993), Robust load-frequency controller design for power systems, *IEE Proc. C*, 140(1), 11–16.

[61] Watkins C.J.C.H. (1989). *Learning from delayed rewards*, PhD Thesis, University of Cambridge, England.

[62] Werbos, P. J., (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*, Ph.D. Thesis.

[63] Werbos P. (1989), Neural networks for control and system identification, *Proc. of CDC'89*, 260–265.

[64] Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. White and D. A. Sofge (Eds.), *Handbook of Intelligent Control* (493-525.). New York: Van Nostrand Reinhold.

[65] Werbos, P. (2009). Intelligence in the Brain: A Theory of How it Works and How to Build It. *Neural Networks – special issue: Goal-Directed Neural Systems*, 22(3).

[66] White D.A. and Sofge D.A. (Editors) (1992). *Handbook of Intelligent Control*, New York: Van Nostrand Reinhold.

[67] Zhou K., Doyle, J. C. and Glover, K. (1996). *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ.

[68] Zhou K. and Khargonekar, P. P. (1988). An algebraic Riccati equation approach to H-infinity optimization, *Syst. Contr. Lett.*, 11, 85-91.

BIOGRAPHICAL INFORMATION

Draguna Vrabie received her B.Sc. and M.Sc. from the Automatic Control and Computer Engineering Dept. of the "Gh. Asachi" Technical University of Iasi, Romania. Since 2005 she has been working as a research assistant at the Automation and Robotics Research Institute, University of Texas at Arlington. Her work on direct adaptive optimal controllers based on reinforcement learning resulted in invited papers both in the computational intelligence and control engineering societies and also a new chapter on approximate dynamic programming (ADP) in the Control Handbook. She has coauthored a book on classical control techniques, two book chapters, and several journal and conference papers. Her research interests include reinforcement learning, approximate dynamic programming, optimal control, adaptive control, model predictive control, and general theory of nonlinear systems.