ONLINE LEARNING ALGORITHMS FOR DIFFERENTIAL DYNAMIC GAMES AND OPTIMAL CONTROL

by

KYRIAKOS G. VAMVOUDAKIS

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

May 2011

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Professor Frank L. Lewis for his guidance, supervision and support during my doctoral research work. Also I would want to thank my parents for the moral and material support at all the duration of my study at the University of Texas at Arlington.

This work was supported by the National Science Foundation ECS-0801330, the Army Research Office W91NF-05-1-0314 and the Air Force Office of Scientific Research FA9550-09-1-0278.

March 31, 2011

ABSTRACT

ONLINE LEARNING ALGORITHMS FOR DIFFERENTIAL DYNAMIC GAMES AND OPTIMAL CONTROL

KYRIAKOS G. VAMVOUDAKIS, PhD

The University of Texas at Arlington, 2011

Supervising Professor: FRANK L. LEWIS

Optimal control deals with the problem of finding a control law for a given system that a certain optimality criterion is achieved. It can be derived using Pontryagin's maximum principle (a necessary condition), or by solving the Hamilton-Jacobi-Bellman equation (a sufficient condition). Major drawback of optimal control is that it is offline. Adaptive control involves modifying the control law used by a controller to cope with the facts that the system is unknown or uncertain. Adaptive controllers are not optimal. Adaptive optimal controllers have been proposed by adding optimality criteria to an adaptive controller, or adding adaptive characteristics to an optimal controller.

In this work, online adaptive learning algorithms are developed for optimal control and differential dynamic games by using measurements along the trajectory or input/output data. These algorithms are based on actor/critic schemes and involve simultaneous tuning of the actor/critic neural networks and provide online solutions to complex Hamilton-Jacobi equations, along with convergence and Lyapunov stability proofs.

The research begins with the development of an online algorithm based on policy iteration for learning the continuous-time (CT) optimal control solution with infinite horizon cost for nonlinear systems with known dynamics. That is, the algorithm learns online in real-time the solution to the optimal control design Hamilton-Jacobi (HJ) equation. This is called 'synchronous' policy iteration.

Then it became interesting to develop an online learning algorithm to solve the continuous-time two-player zero-sum game with infinite horizon cost for nonlinear systems. The algorithm learns online in real-time the solution to the game design Hamilton-Jacobi-Isaacs equation. This algorithm is called online gaming algorithm 'synchronous' zero-sum game policy iteration.

One of the major outcomes of this work is the online learning algorithm to solve the continuous time multi player non-zero sum games with infinite horizon for linear and nonlinear systems. The adaptive algorithm learns online the solution of coupled Riccati and coupled Hamilton-Jacobi equations for linear and nonlinear systems respectively. The optimal-adaptive algorithm is implemented as a separate actor/critic parametric network approximator structure for every player, and involves simultaneous continuous-time adaptation of the actor/critic networks.

The next result shows how to implement Approximate Dynamic Programming methods using only measured input/output data from the systems. Policy and value iteration algorithms have been developed that converge to an optimal controller that requires only output feedback.

The notion of graphical games is developed for dynamical systems, where the dynamics and performance indices for each node depend only on local neighbor information. A cooperative policy iteration algorithm, is given for graphical games, that converges to the best response when the neighbors of each agent do not update their policies and to the cooperative Nash equilibrium when all agents update their policies simultaneously.

Finally, a synchronous policy iteration algorithm based on integral reinforcement learning is given. This algorithm does not need the drift dynamics.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

LIST OF ILLUSTRATIONS viii

Chapter Page

1. INTRODUCTION 1

1.1 Adaptive Optimal Control 1

1.2 Zero-Sum Games 5

1.3 Non-Zero-Sum Games 6

1.4 Motivation 8

1.5 Contribution 8

1.6 List of publications which resulted from this work 10

1.6.1 Book 10

1.6.2 Book chapters 10

1.6.3 Journal articles 10

1.6.4 Conference papers 11

1.7 Outline 13

2. ONLINE ACTOR-CRITIC ALGORITHM TO SOLVE THE CONTINUOUS TIME INFINITE HORIZON OPTIMAL CONTROL PROBLEM 16

2.1 Introduction 16

2.2 The optimal control problem and value function approximation 17

2.2.1 Optimal Control and the continuous-time HJB equation 17

2.2.2 Policy Iteration 19

2.2.3 Value function approximation (VFA) 20

2.3 Tuning and convergence of Critic NN 24

2.4 Action Neural Network and online synchronous policy iteration 30

2.5 Simulations 33

2.5.1 Linear system example....33

2.5.2 Nonlinear system example 35

2.6 Conclusion 39

3. ONLINE GAMING: REAL TIME SOLUTION OF NONLINEAR TWO-PLAYER ZERO-SUM GAMES USING SYNCHRONOUS POLICY ITERATION....40

3.1 Introduction 40

3.2 Background: Two-Player Differential Game and Policy Iteration 41

3.2.1 Two-player zero-sum differential games and Nash equilibrium 42

3.2.2 Policy Iteration solution of the HJI equation 46

3.3 NN approximation structure for online policy iteration algorithm....48

3.3.1 Value function approximation: Critic Neural Network....48

3.3.2 Tuning and Convergence of the Critic Neural Network....51

3.3.3 Action and Disturbance Neural Network 55

3.4 Online solution of Two-Player Zero-Sum Games using Neural Networks 56

3.5 Simulations 61

3.5.1 Linear System....61

3.5.2 Single Player Linear System 63

3.5.3 Nonlinear System 66

3.6 Conclusion 70

4. MULTI-PLAYER GAMES: ONLINE ADAPTIVE LEARNING SOLUTION OF COUPLED HAMILTON-JACOBI AND COUPLED RICCATI EQUATIONS 72

4.1 Introduction 72

4.2 N-player differential game for nonlinear systems 74

4.2.1 Multi-Player Non-Zero Sum Games 74

4.2.2 Cooperation vs. Non-Cooperation in Multi-Player Games 77

4.2.3 Policy Iteration Solution for Non-Zero Sum Games 77

4.3 Value function approximation for solution
of nonlinear Lyapunov equations...78

4.4 Online Solution for 2-player games 83

4.5 Simulation Results 89

4.5.1 Nonlinear system 89

4.5.2 Linear system 93

4.5.3 Zero-Sum Game with unstable linear system 95

5. REINFORCEMENT LEARNING FOR PARTIALLY OBSERVABLE PROCESSES:
ADAPTIVE DYNAMIC PROGRAMMING USING MEASURED OUTPUT DATA.............99

5.1 Introduction 99

5.2 Background 100

5.2.1 Dynamic programming and LQR 100

5.2.2 Temporal difference, policy iteration, value iteration 102

5.3 Temporal difference, PI, and VI based on output feedback 106

5.3.1 Writing the value function in terms
of available measured data 107

5.3.2 Writing the TD error in terms of available measured data...........111

5.3.3 Writing the policy update in terms of available
measured data...111

5.4 Implementation, probing noise, bias and discount factors 115

5.5 Simulation Results 118

5.5.1 Stable linear system and policy iteration 118

5.5.2 Q-learning and output feedback ADP..120

5.5.3 Unstable linear system and value iteration 122

5.6 Conclusion 124

6. MULTI AGENT DIFFERENTIAL GRAPHICAL GAMES: POLICY ITERATION AND ONLINE ADAPTIVE LEARNING SOLUTION 126

6.1 Introduction 126

6.2 Synchronization and node error dynamics 127

6.2.1 Graphs 127

6.2.2 Synchronization and node error dynamics 128

6.3 Cooperative multi-player games on graphs 130

6.3.1 Cooperative performance index 130

6.3.2 Graphical games and cooperative Nash equilibrium 131

6.3.3 Stability and solution of graphical games 133

6.4 Policy iteration solution for cooperative multi-player games 138

6.4.1 Best response 138

6.4.2 Policy Iteration for solution of graphical games 140

6.5 Online solution of multi-agent cooperative games using neural networks 143

6.5.1 Critic neural network 144

6.5.2 Actor neural network and online solution 145

6.6 Simulations 150

6.6.1 Velocity and Position regulated to zero 151

6.6.2 All nodes synchronize to the leader node 152

7. ONLINE ADAPTIVE LEARNING OF OPTIMAL CONTROL SOLUTIONS USING INTEGRAL REINFORCEMENT LEARNING 154

7.1 Introduction 154

7.2 Optimal control problem and the policy iteration algorithm 155

7.2.1 Optimal control and the continuous time HJB equation 155

7.2.2 Policy iteration 157

7.2.3 Value function approximation (VFA) 158

7.2.4 Integral reinforcement learning 159

7.3 Online integral reinforcement learning algorithm with synchronous tuning of actor and critic neural networks 160

7.3.1 Critic NN and Bellman equation solution 161

7.3.2 Action NN and online adaptive optimal control 163

7.4 Simulation results 165

7.4.1 Linear system example 165

7.4.2 Nonlinear system example 167

7.5 Conclusion 171

8. CONCLUSIONS AND FUTURE WORK 172

APPENDIX

A. PROOFS 175

REFERENCES 198

BIOGRAPHICAL INFORMATION 206

LIST OF ILLUSTRATIONS

Figure Page

1. Actor/Critic Structure 21
2. Convergence of the critic parameters to the parameters of the optimal critic 34
3. Evolution of the system states for the duration of the experiment 35
4. Convergence of the critic parameters..37
5. Evolution of the system states for the duration of the experiment 37
6. Optimal Value function 38
7. 3D plot of the approximation error for the value function 38
8. 3D plot of the approximation error for the control 39
9. Convergence of the critic parameters to the parameters of the optimal critic 63
10. Evolution of the system states for the duration of the experiment 63
11. Convergence of the critic parameters to the parameters of the optimal critic 65
12. Evolution of the system states..65
13. Convergence of the critic parameters..68
14. Evolution of the system states..68
15. Optimal Value function 69
16. 3D plot of the approximation error for the value function 69
17. 3D plot of the approximation error for the control 70
18. 3D plot of the approximation error for the disturbance 70
19. Convergence of the critic parameters for first player....................91
20. Convergence of the critic parameters for second player....................92
21. Evolution of the system states..92
22. 3D plot of the approximation error for the value function of player 1 93

23. 3D plot of the approximation error for the control of player 1 93

24. Evolution of the system state..94

25. Convergence of the critic NN for the first player...95

26. Convergence of the critic NN for the second player...95

27. Convergence of the critic NN for the first player...97

28. Convergence of the critic NN for the second player...97

29. Evolution of the system states..98

30. Convergence of p_0, p_u, p_y 120

31. Evolution of the system states for the duration of the experiment 120

32. Convergence of H_{11}, H_{12}, H_{13} 121

33. Evolution of the system states for the duration of the experiment. 122

34. Convergence of p_0, p_u, p_y 124

35. Evolution of the system states for the duration of the experiment 124

36. Communication graph 150

37. Critic Parameters..151

38. Evolution of the system states..152

39. Critic Parameters..153

40. Synchronization of all the agents to the leader 153

41. Convergence of the critic parameters to the parameters of the optimal critic.........................167

42. Evolution of the system states for the duration of the experiment 167

43. Convergence of the critic parameters to the parameters of the optimal critic.........................169

44. Evolution of the system states for the duration of the experiment 170

45. Error between the optimal and approximated value function 170

46. Error between the optimal and the approximated control input..171

CHAPTER 1

INTRODUCTION

The introductory chapter discusses background, motivation and contribution. The list of publications which resulted from this research is given in Section 1.6.

1.1 Adaptive Optimal Control

Optimal control has emerged as one of the fundamental design philosophies of modern control systems design. Optimal control policies satisfy the specified system performance while minimizing a structured cost index which describes the balance between desired performance and available control resources.

From a mathematical point of view the solution of the optimal control problem is based on the solution of the underlying Hamilton-Jacobi-Bellman (HJB) equation. Until recently, due to the intractability of this nonlinear differential equation for continuous-time (CT) systems, which form the object of interest in this work, only particular solutions were available (e.g. for the linear time-invariant case, the HJB becomes the Riccati equation). For this reason considerable effort has been devoted to developing algorithms which approximately solve this equation [4], [13], [63]. Far more results are available for the solution of the discrete-time HJB equation. Good overviews are given in [15], [81], [98], [99], [100].

Reinforcement learning (RL) is a class of methods used in machine learning to methodically modify the actions of an agent based on observed responses from its environment [22],[23], [35], [86]. The RL methods have been developed starting from learning mechanisms observed in mammals. Every decision-making organism interacts with its environment and uses those interactions to improve its own actions in order to maximize the positive effect of its limited available resources; this in turn leads to better survival chances. RL is a means of *learning optimal behaviors by observing the response from the environment to non-optimal* *control policies*. In engineering terms, RL refers to the learning approach of an actor or agent which modifies its actions, or control policies, based on stimuli received in response to its interaction with its environment. This learning can be extended along two dimensions: i) nature of interaction (competitive or collaborative) and ii) the number of decision makers (single or multi agent).

In view of the advantages offered by the RL methods, a recent objective of control systems researchers is to introduce and develop RL techniques which result in optimal feedback controllers for dynamical systems that can be described in terms of ordinary differential or difference equations.

Some of the methods involve a computational intelligence technique known as Policy Iteration (PI) [35], [86]. PI refers to a class of algorithms built as a two-step iteration: *policy evaluation* and *policy improvement*. Instead of trying a direct approach to solving the HJB equation, the PI algorithm starts by evaluating the cost of a given initial admissible (in a sense to be defined herein) control policy. This is often accomplished by solving a nonlinear Lyapunov equation. This new cost is then used to obtain a new improved (i.e. which will have a lower associated cost) control policy. This is often accomplished by minimizing a Hamiltonian function with respect to the new cost. (This is the so-called 'greedy policy' with respect to the new cost). These two steps of policy evaluation and policy improvement are repeated until the policy improvement step no longer changes the actual policy, thus convergence to the optimal controller is achieved. One must note that the infinite horizon cost can be evaluated only in the case of admissible control policies, which requires that the policy be stabilizing. Admissibility is in fact a condition for the control policy which is used to initialize the algorithm.

Werbos defined actor-critic online learning algorithms to solve the optimal control problem based on so-called Value Iteration (VI), which does not require an initial stabilizing control policy [98], [99], [100]. He defined a family of VI algorithms which he termed Adaptive Dynamic Programming (ADP) algorithms. He used a critic neural network (NN) for value function approximation (VFA) and an actor NN for approximation of the control policy. Adaptive critics have been described in [69] for discrete-time systems and [8], [30], [91], [92] for continuous-time systems.

Generalized Policy Iteration has been discussed in [86]. This is a family of optimal learning techniques which has PI at one extreme. In generalized PI, at each step one does not completely evaluate the cost of a given control, but only updates the current cost estimate *towards* that value. Likewise, one does not fully update the control policy to the greedy policy for the new cost estimate, but only updates the policy *towards* the greedy policy. Value Iteration in fact belongs to the family of generalized PI techniques.

In the linear CT system case, when quadratic indices are considered for the optimal stabilization problem, the HJB equation becomes the well known Riccati equation and the policy iteration method is in fact Newton's method proposed by Kleinman [44], which requires iterative solutions of Lyapunov equations. In the case of nonlinear systems, successful application of the PI method was limited until [13], where Galerkin spectral approximation methods were used to solve the nonlinear Lyapunov equations describing the policy evaluation step in the PI algorithm. Such methods are known to be computationally intensive. These are all off-line methods for PI.

The key to solving practically the CT nonlinear Lyapunov equations was in the use of neural networks (NN) [4] which can be trained to become approximate solutions of these equations. In fact the PI algorithm for CT systems can be built on Werbos' actor/critic structure which involves two neural networks: the critic NN, is trained to approximate the solution of the nonlinear Lyapunov equation at the policy evaluation step, while the actor neural network is trained to approximate an improving policy at the policy improving step. The method of [4] is also an offline method.

In [91], [92], [93] was developed an online PI algorithm for continuous-time systems which converges to the optimal control solution without making explicit use of any knowledge on the internal dynamics of the system. The algorithm was based on *sequential updates* of the critic (policy evaluation) and actor (policy improvement) neural networks. That is, while one NN is tuned the other one remains constant. In this work we provide algorithms for simultaneous tuning of the two neural networks.

Most of the PI and Value Iteration (VI) algorithms require at least some knowledge of the system dynamics and measurement of the entire internal state vector which describes the dynamics of the system/environment (e.g. [7], [20], [68], [78], [86], [99]). The so called Q-learning class of algorithms [17], [97] (called action dependent HDP by Werbos [99], [100]) does not require exact or explicit description of the system dynamics, but they still use full state measurement in the feedback control loop. From control systems engineering perspective this latter requirement may be hard to fulfill as measurements of the entire state vector may not be available and/or may be difficult and expensive to obtain. Although various control algorithms (e.g. state-feedback) require full state knowledge, in practical implementations taking measurements of the entire state vector is not feasible. The state vector is generally estimated based on limited information about the system available by measuring the system's outputs. State estimation techniques have been proposed (e. g. [55], [58], [67]). These generally require a known model of the system dynamics.

In real life, it is difficult to design and implement optimal estimators because the system dynamics and the noise statistics are not exactly known. However information about the system and noise is included in a long enough set of input/output data. It would be desirable to be able to design an estimator by using input/output data without any system knowledge or noise identification. Such techniques belong to the field of data-based control techniques, where the control input depends on input/output data measured directly from the plant. These techniques are: data-based predictive control [59], unfalsified control [73], Markov data-based LQG control [82], disturbance-based control [89], simultaneous perturbation stochastic approximation [84], pulse response-based control [14], iterative feedback tuning [32], and virtual reference feedback tuning [48]. In [1] data based optimal control was achieved through identification of Markov parameters.

In this work, novel output feedback ADP algorithms are derived for affine in the control input linear time-invariant (LTI) deterministic systems. Such systems have as stochastic equivalent the partially observable Markov decision processes (POMDPs). In this work, data-based optimal control is implemented on-line using novel PI and VI ADP algorithms that require only reduced measured information available at the system outputs. These two classes of output feedback algorithms *do not require any knowledge of the system dynamics (A,B,C)* and as such are similar to Q-learning [17], [97], [99], [100] but they have an added advantage of requiring only measurements of input/output data and not the full system state. In order to ensure that the data set is sufficiently rich and linearly independent, there is a need to add (c.f. [17]) probing noise to the control input. We discuss this issue showing that probing noise leads to bias. Adding a discount factor in the cost minimizes it to an almost zero effect. This discount factor is related to adding exponential data weighting in the Kalman Filter to remove the bias effects of unmodeled dynamics [55].

1.2 Zero-Sum Games

Games provide an ideal environment in which to study computational intelligence, offering a range of challenging and engaging problems. Game theory [88] captures the behavior in which a player's success in selecting strategies depends on the choices of other players. One goal of game theory techniques is to find (saddle point) equilibria, in which each player has an outcome that cannot be improved by unilaterally changing his strategy (e.g. Nash equilibrium). The H_∞ control problem is a *minimax* optimization problem, and hence a zero-sum game where the controller is a minimizing player and the disturbance a maximizing one. Since the work of George Zames in the early 1980s, H_∞ techniques have been used in control systems, for sensitivity reduction and disturbance rejection. This work is concerned with 2-player zero-sum games that are related to the H_∞ control problem, as formulated by [11], [11], [77].

Game theory and H-infinity solutions rely on solving the Hamilton-Jacobi-Isaacs (HJI) equations, which in the zero-sum linear quadratic case reduce to the generalized game algebraic Riccati equation (GARE). In the nonlinear case the HJI equations are difficult or impossible to solve, and may not have global analytic solutions even in simple cases (e.g. scalar system, bilinear in input and state). Solution methods are generally offline and generate fixed control policies that are then implemented in online controllers in real time.

In this work we provide methods for online gaming, that is for solution of 2-player zero-sum infinite horizon games *online*, through learning the saddle point strategies in real-time. The dynamics may be nonlinear in continuous-time and are assumed known. A novel neural network adaptive control technique is given that is based on reinforcement learning techniques, whereby the control and disturbance policies are tuned online using data generated in real time along the system trajectories. Also tuned is a 'critic' approximator structure whose function is to identify the value or outcome of the current control and disturbance policies. Based on this value estimate, the policies are continuously updated. This is a sort of indirect adaptive control algorithm, yet, due to the direct form dependence of the policies on the learned value, it is affected online as direct ('optimal') adaptive control.

1.3 Non-Zero-Sum Games

Game theory [88] has been very successful in modeling strategic behavior, where the outcome for each player depends on the actions of himself and all the other players. Every player chooses a control to minimize independently from the others his *own* performance objective. None has knowledge of the others' strategy. A lot of applications of optimization theory require the solution of coupled Hamilton-Jacobi equations [11], [27]. In games with *N* players, each player decides for the Nash equilibrium depending on Hamilton-Jacobi equations coupled through their quadratic terms [27], [28]. Each dynamic game consists of three parts: i) players; ii) actions available for each player; iii) costs for every player that depend on their actions.

Multi player non-zero sum games rely on solving the coupled Hamilton-Jacobi (HJ) equations, which in the linear quadratic case reduce to the coupled algebraic Riccati equations [2], [27], [28]. Solution methods are generally offline and generate fixed control policies that are then implemented in online controllers in real time. In the nonlinear case the coupled HJ equations are difficult or impossible to solve, and may not have global analytic solutions even in simple cases (e.g. scalar system, bilinear in input and state) [83] (discussion of viscosity solutions).

For the most part, interest in the control systems community has been in the (non-cooperative) zero-sum games, which provide the solution of the H-infinity robust control problem [11], [60]. However, dynamic team games may have some cooperative objectives and some selfish objectives among the players. This cooperative/non-cooperative balance is captured in the NZS games, as detailed herein.

In this work we are interested in feedback policies with full state information, and provide methods for *online gaming*, that is for solution of *N*-player infinite horizon NZS games *online*, through learning the Nash-equilibrium in real-time. The dynamics are nonlinear in continuous-time and are assumed known. A novel adaptive control technique is given that is based on reinforcement learning techniques, whereby each player's control policies are tuned online using data generated in real time along the system trajectories. Also tuned by each player are 'critic' approximator structures whose function is to identify the values of the current control policies for each player. Based on these value estimates, the players' policies are continuously updated. This is a sort of indirect adaptive control algorithm, yet, due to the simple form dependence of the control policies on the learned value, it is affected online as direct ('optimal') adaptive control.

1.4 Motivation

Optimal control is generally an offline design technique that requires full knowledge of the systems dynamics, e.g. in the linear systems case one must solve the Riccati equation. On the other hand, adaptive control is a body of online design techniques that use measured data along system trajectories to learn to compensate for unknown system dynamics, disturbances, and modeling errors to provide guaranteed performance. Optimal adaptive controllers have been designed using indirect techniques, whereby the unknown plant is first identified and then a Riccati equation is solved [36], [87]. Inverse adaptive controllers have been provided that optimize a performance index, meaningful but not of the designer's choice [45], [57]. Direct adaptive controllers that converge to optimal solutions for nonlinear systems given a PI selected by the designer have generally not been developed.

Therefore developing online learning algorithms for optimal control and games where every controller minimizes a different performance index is of great interest in the control systems society.

1.5 Contribution

The contributions of the thesis are the following:

1. An online adaptive optimal controller to solve the continuous-time infinite horizon optimal control problem which uses reinforcement learning principles.
2. An online gaming algorithm to solve the zero-sum game problem.
3. An online adaptive control algorithm based on policy iteration to solve the continuous-time multi player non zero sum game with infinite horizon for linear and nonlinear systems.
4. Reinforcement learning methods which require only output feedback and yet converge to an optimal controller.
5. Policy iteration algorithm and online adaptive learning solution for graphical games
6. An online adaptive optimal controller to solve the continuous-time infinite horizon optimal control problem based on integral reinforcement learning that does not need the drift dynamics.

The first result is concerned with developing an online algorithm where the actor/critic networks are tuned simultaneously. This algorithm converges to the solution of Hamilton-Jacobi-Bellman equation without solving it and is termed as ‘synchronous’ policy iteration. The convergence to the optimal controller is proven, and the stability of the system is also guaranteed.

The second result included in this thesis is an online gaming algorithm to solve the zero-sum game with infinite horizon for linear and nonlinear systems. The two-player zero sum problem provides the solution to the bounded L_2 gain problem and so is important for robust control. However, its solution depends on solving a design Hamilton-Jacobi-Isaacs (HJI) equation, which is generally intractable for nonlinear systems. This algorithm converges to the Hamilton-Jacobi-Isaacs and the Game Algebraic Riccati equation for nonlinear and linear systems respectively without solving them. The convergence to the optimal saddle point solution is proven and stability of the system is also guaranteed.

The third result is a new online algorithm to solve non-zero sum dynamic games. This algorithm converges to the solution of coupled Riccati and coupled Hamilton-Jacobi equations for linear and nonlinear systems respectively. The optimal-adaptive algorithm is implemented as a separate actor/critic parametric network approximator structure for every player, and involves simultaneous continuous-time adaptation of the actor/critic networks. Furthermore this method finds in real time approximations of the optimal value and the Nash equilibrium while also guaranteeing closed-loop stability.

The fourth result in this thesis develops policy iteration and value iteration algorithms that require only input/output data and not measurements of the states. This new output feedback optimal learning methods have the important advantage that knowledge of the system dynamics is not needed for their implementation.

The fifth result introduces graphical games where the dynamics and performance indices for each node depend only on local neighbor information. A policy iteration algorithm and an online learning algorithm are proposed and proofs of convergence are also provided.

The last result develops an online adaptive controller based on integral reinforcement learning that tunes the actor and critic NN simultaneously and does not need any knowledge on the drift dynamics.

1.6 List of publications which resulted from this work

1.6.1. Book

D. Vrabie, K. G. Vamvoudakis, Frank Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*, in preparation, IET 2011.

Greg R. Hudas, Dariusz Mikulski, Kyriakos G. Vamvoudakis, Frank L. Lewis, E. Gu, Decision and Control for Tactical Behaviors of Autonomous Systems, in preparation, 2011.

1.6.2. Book Chapters

K. G. Vamvoudakis, and F. L. Lewis, “Online Gaming: Real Time Solution of Nonlinear Two-Player Zero-Sum Games Using Synchronous Policy Iteration,” in Advances in Reinforcement Learning, ed. Abdelhamid Mellouk, Chapter 18, INTECH, 2011.

K. G. Vamvoudakis, and F. L. Lewis, “Online synchronous policy iteration method for optimal control,” in Recent Advances in Intelligent Control Systems, ed. Wen Yu, Chapter 14, Springer-Verlag, Berlin, 2009.

1.6.3. Journal articles

Submitted/Under review:

Kyriakos G. Vamvoudakis, F. L. Lewis, “Online Neural Network Solution of Nonlinear Two-Player Zero-Sum Games Using Synchronous Policy Iteration,” submitted to *International Journal of Robust and Nonlinear Control*, 2010.

Accepted/Published:

Kyriakos G. Vamvoudakis, and F. L. Lewis, “Multi-Player Games: Online Adaptive Learning Solution of Coupled Hamilton-Jacobi Equations,” to appear in *Automatica*, 2011.

Kyriakos G. Vamvoudakis, and F. L. Lewis, “Online Actor-Critic Algorithm to Solve the Continuous-Time Infinite Horizon Optimal Control Problem,” *Automatica*, vol. 46, no. 5, pp. 878-888, 2010.

This paper was one of the Top 25 Articles in Decision Sciences, Elsevier, April-September 2010.

F. L. Lewis, Kyriakos G. Vamvoudakis, “Reinforcement Learning for Partially Observable Dynamic Processes: Adaptive Dynamic Programming Using Measured Output Data,” *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 14-25, 2011.

Greg Hudas, K. G. Vamvoudakis, D. Mikulski and F. L. Lewis, “Online Adaptive Learning for Team Strategies in Multi-Agent Systems,” invited paper to (special issue, “Intelligent Behaviors for Tactical Unmanned Systems”) Journal of Defense Modeling & Simulation, to appear, 2010.

1.6.4. Conference papers

Submitted/Under review:

Kyriakos G. Vamvoudakis, and F. L. Lewis, “Policy Iteration Algorithms for distributed networks and graphical games,” submitted to 50th IEEE Conference on Decision and Control, Orlando, December, 2011.

Kyriakos G. Vamvoudakis, and F. L. Lewis, “Policy Iteration Algorithm for distributed networks and graphical games,” submitted to 50th IEEE Conference on Decision and Control, Orlando, December, 2011.

Kyriakos G. Vamvoudakis, F. L. Lewis, “Non-Zero Sum Games: Online Learning Solution of Coupled Hamilton-Jacobi and Coupled Riccati Equations,” submitted to *IEEE Multi-Conference on Systems and Control*, Denver, Colorado, September 28-30, 2011.

Accepted/Published in Proceedings:

Kyriakos G. Vamvoudakis, D. Vrabie, and F. L. Lewis, “Online Adaptive Learning of Optimal Control Solutions Using Integral Reinforcement Learning,” to appear in *IEEE Symp. ADPRL*, Paris, France, April 11-15, 2011.

Kyriakos G. Vamvoudakis, F. L. Lewis, “Online Solution of Nonlinear Two-Player Zero-Sum Games Using Synchronous Policy Iteration,” *Proc. 49th IEEE Conference on Decision and Control*, pp. 3040 – 3047, Atlanta, 2010.

Kyriakos G. Vamvoudakis, D. G. Mikulski, Greg R. Hudas, F. L. Lewis, E. Y. Gu, “Distributed Games for Multi-Agent Systems: Games on Communication Graphs,” *Proc. 27th Army Science Conference*, Orlando, FL, 2010.

This paper won the Autonomous/Unmanned Vehicles Best Paper Award

Greg Hudas, F.L. Lewis and K.G. Vamvoudakis, "Online Gaming for Learning Optimal Team Strategies in Real Time," *Proc. of SPIE*, vol. 7692, 76920W, doi:10.1117/12.850231, 2010.

F. L. Lewis, Kyriakos G. Vamvoudakis, “Optimal Adaptive Control for Unknown Systems Using Output Feedback by Reinforcement Learning Methods,” *Proc. 8th IEEE International Conference on Control & Automation*, pp. 2138 - 2145, Xiamen, 2010.

K. Vamvoudakis, D. Vrabie, and F. L. Lewis, “Online policy iteration based algorithms to solve the continuous-time infinite horizon optimal control problem,” *Proc. IEEE Symp. ADPRL*, pp. 36-41, Nashville, Mar. 2009.

K. G. Vamvoudakis, and F. L. Lewis, "Online Actor Critic Algorithm to solve the Continuous-Time Infinite Horizon Optimal Control Problem," *Proc. Int. Joint Conf. on Neural Networks*, pp. 3180-3187, Atlanta, June 2009.

D. Vrabie, K. Vamvoudakis, and F. L. Lewis, "Adaptive Optimal Controllers based on Generalized policy iteration in Continuous-Time Framework," *Proc. Mediterranean Conf. Control and Automation*, pp. 1402-1409, Thessaloniki, June 2009.

1.7 Outline

Chapter 2 presents an online algorithm based on policy iteration for learning the continuous time optimal control with infinite horizon for nonlinear systems with known dynamics. It is implemented as an actor/critic structure which involves simultaneous continuous-time adaptation of both actor and critic neural networks. A persistence of excitation condition is shown to guarantee convergence of the critic to the actual value function. Simulation results, obtained considering a continuous time F16 aircraft plant with quadratic function and an affine in control input nonlinear system with a quadratic cost.

Chapter 3 presents an online adaptive learning algorithm based on policy iteration to solve the continuous-time two-player zero-sum game with infinite horizon cost for nonlinear systems with known dynamics. The algorithm learns online an approximate local solution to the game HJI equation. The algorithm is implemented as an actor/critic/disturbance structure which involves simultaneous continuous-time adaptation of critic, actor and disturbance neural networks. A persistence of excitation condition is shown to guarantee convergence of the critic to the actual optimal value function. The convergence to the optimal saddle point solution is proven, and stability of the system is also guaranteed. Simulation examples show the effectiveness of the new algorithm in solving the HJI equation online for a linear systems and a complex nonlinear system.

Chapter 4 presents an adaptive algorithm based on policy iteration reinforcement learning techniques to solve the continuous-time multi player non-zero sum game with infinite horizon for linear and nonlinear systems. Non-zero sum games allow for players to have a cooperative team component and an individual selfish component of strategy. This algorithm learns online the solution of coupled Riccati and coupled Hamilton-Jacobi equations and finds in real time approximations of the optimal value and the Nash equilibrium while also guaranteeing closed-loop stability. A detailed mathematical analysis is done for two player non-zero sum games. Simulation examples for linear and nonlinear systems show the effectiveness of the new algorithm.

Chapter 5 develops policy and value iteration algorithms that converge to an optimal controller that requires only output feedback. It is shown that, similar to *Q*-learning, the new methods have the important advantage that knowledge of the system dynamics is not needed for the implementation of these learning algorithms or for the output feedback control. The learned output feedback controller is in the form of a polynomial autoregressive moving-average controller that has equivalent performance with the optimal state variable feedback gain. Simulation examples show the effectiveness of the proposed algorithms and also compare their performance to that of *Q*-learning.

Chapter 6 brings together cooperative control, reinforcement learning and game theory to solve multi-player differential games on communication graph topologies. The notion of graphical games is developed for dynamical systems, where the dynamics and performance indices for each node depend only on local neighbor information. A derivation of coupled Riccati equations for solution of graphical games is proposed. Furthermore a policy iteration algorithm is shown to converge to the best response when every agent has fixed policies for his neighbors and to the Nash equilibrium when all agents update their policies simultaneously. Finally an online adaptive learning solution is shown to solve online the graphical games.

Chapter 7 introduces an online algorithm that uses integral reinforcement knowledge for learning the continuous-time optimal control solution for nonlinear systems with infinite horizon costs and partial knowledge of the system dynamics. This algorithm is a data based approach to the solution of the Hamilton-Jacobi-Bellman equation and it does not require explicit knowledge on the system's drift dynamics.

Chapter 8 presents conclusions and future work ideas.

CHAPTER 2

ONLINE ACTOR-CRITIC ALGORITHM TO SOLVE THE CONTINUOUS-TIME INFINITE HORIZON OPTIMAL CONTROL PROBLEM

2.1 Introduction

This chapter is concerned with developing an online approximate solution, based on PI, for the infinite horizon optimal control problem for continuous-time nonlinear systems with known dynamics. We present an online adaptive algorithm which involves *simultaneous tuning* of both actor and critic neural networks (i.e. both neural networks are tuned at the same time). We term this algorithm ‘synchronous’ policy iteration. This approach is an extremal version of the generalized Policy Iteration introduced in [86].

This approach to policy iteration is motivated by work in adaptive control [36], [87]. Adaptive control is a powerful tool that uses online tuning of parameters to provide effective controllers for nonlinear or linear systems with modeling uncertainties and disturbances. Closed-loop stability while learning the parameters is guaranteed, often by using Lyapunov design techniques. Parameter convergence, however, often requires that the measured signals carry sufficient information about the unknown parameters (persistence of excitation condition).

There are two main contributions in this chapter. The first involves introduction of a nonstandard ‘normalized’ critic neural network tuning algorithm, along with guarantees for its convergence based on a persistence of excitation condition regularly required in adaptive control. The second involves adding nonstandard extra terms to the actor neural network tuning algorithm that are required to guarantee close loop stability, along with stability and convergence proofs.

The chapter is organized as follows. Section 2.2 provides the formulation of the optimal control problem, followed by the general description of policy iteration and neural network value function approximation. Section 2.3 discusses tuning of the critic NN, in effect designing an observer for the unknown value function. Section 2.4 presents the online synchronous PI method, and shows how to simultaneously tune the critic and actor NNs to guarantee convergence and closed-loop stability. Results for convergence and stability are developed using a Lyapunov technique. Section 2.5 presents simulation examples that show the effectiveness of the online synchronous CT PI algorithm in learning the optimal value and control for both linear systems and nonlinear systems.

2.2 The optimal control problem and value function approximation

2.2.1. Optimal control and the continuous-time HJB equation

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

$$\dot{x}(t) = f(x(t)) + g(x(t))\,u(x(t))\,;\;\; x(0) = x_0 \tag{2.1}$$

with state $x(t) \in \mathbb{R}^n$, $f(x(t)) \in \mathbb{R}^n$, $g(x(t)) \in \mathbb{R}^{n\times m}$ and control input $u(t) \in \mathbb{R}^m$. We assume that, $f(0)=0$, $f(x)+g(x)u$ is Lipschitz continuous on a set $\Omega \subseteq \mathbb{R}^n$ that contains the origin, and that the 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. The system dynamics $f(x), g(x)$ are assumed known.

Define the infinite horizon integral cost

$$V(x_0) = \int_0^\infty r(x(\tau), u(\tau))d\tau \tag{2.2}$$

where $r(x,u) = Q(x) + u^T R u$ 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 symmetric positive definite matrix.

Definition 2.1. [4] (Admissible policy) A control policy $\mu(x)$ is defined as admissible with respect to (2.2) on Ω, denoted by $\mu \in \Psi(\Omega)$, if $\mu(x)$ is continuous on Ω, $\mu(0)=0$, $u(x)=\mu(x)$ stabilizes (2.1) on Ω, and $V(x_0)$ is finite $\forall x_0 \in \Omega$.

For any admissible control policy $\mu \in \Psi(\Omega)$, if the associated cost function

$$V^{\mu}(x_0) = \int_0^{\infty} r(x(\tau), \mu(x(\tau)))d\tau \tag{2.3}$$

is C^1, then an infinitesimal version of (2.3) is the so-called nonlinear Lyapunov equation

$$0 = r(x, \mu(x)) + \left(V_x^{\mu}\right)^T (f(x) + g(x)\mu(x)),\ V^{\mu}(0) = 0 \tag{2.4}$$

where V_x^{μ} denotes the partial derivative of the value function V^{μ} with respect to x. (Note that the value function does not depend explicitly on time).

We define the gradient here as a column vector, and use at times the alternative operator notation $\nabla \equiv \partial/\partial x$.

Equation (2.4) is a Lyapunov equation for nonlinear systems which, given a controller $\mu(x) \in \Psi(\Omega)$, can be solved for the value function $V^{\mu}(x)$ associated with it. Given that $\mu(x)$ is an admissible control policy, if $V^{\mu}(x)$ satisfies (2.4), with , then $V^{\mu}(x)$ is a Lyapunov function for the system (2.1) with control policy $\mu(x)$.

The optimal control problem can now be formulated: Given the continuous-time system (2.1), the set $\mu \in \Psi(\Omega)$ of admissible control policies and the infinite horizon cost functional (2.2), find an admissible control policy such that the cost index (2.2) associated with the system (2.1) is minimized.

Defining the Hamiltonian of the problem

$$H(x, u, V_x) = r(x(t), u(t)) + V_x^T (f(x(t)) + g(x(t))\mu(t)), \tag{2.5}$$

the optimal cost function $V^*(x)$ defined by

$$V^*(x_0) = \min_{\mu \in \Psi(\Omega)} (\int_0^{\infty} r(x(\tau), \mu(x(\tau)))d\tau).$$

with $x_0 = x$ is known as the *value function*, and satisfies the HJB equation

$$0 = \min_{\mu \in \Psi(\Omega)} [H(x, \mu, V_x^*)] \tag{2.6}$$

Assuming that the minimum on the right hand side of (2.6) exists and is unique then the optimal control function for the given problem is

$$\mu^*(x) = -\frac{1}{2} R^{-1} g^T(x) V_x^*(x) . \tag{2.7}$$

Inserting this optimal control policy in the nonlinear Lyapunov equation we obtain the formulation of the HJB equation in terms of V_x^*

$$\begin{gathered} 0 = Q(x) + V_x^{*T}(x) f(x) - \frac{1}{4} V_x^{*T}(x) g(x) R^{-1} g^T(x) V_x^*(x) \\ V^*(0) = 0 \end{gathered} \tag{2.8}$$

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 (2.8) for the value function and then substitute the solution in (2.7) to obtain the optimal control. However, due to the nonlinear nature of the HJB equation finding its solution is generally difficult or impossible.

2.2.2. Policy Iteration

The approach of synchronous policy iteration used in this chapter is motivated by Policy iteration (PI) [86]. Therefore in this section we describe PI.

Policy iteration (PI) [86] is an iterative method of reinforcement learning for solving optimal control problems, and consists of policy evaluation based on (2.4) and policy improvement based on (2.7). Specifically, the PI algorithm consists in solving iteratively the following two equations:

Policy Iteration Algorithm:

1. given $\mu^{(i)}(x)$, solve for the value $V^{\mu^{(i)}}(x(t))$ using

$$\begin{aligned}&0 = r(x, \mu^{(i)}(x)) + (\nabla V^{\mu^{(i)}})^T (f(x) + g(x)\mu^{(i)}(x)) \\ &V^{\mu^{(i)}}(0) = 0\end{aligned} \qquad (2.9)$$

2. update the control policy using

$$\mu^{(i+1)} = \arg\min_{u \in \Psi(\Omega)} [H(x, u, \nabla V_x^{(i)})], \qquad (2.10)$$

which explicitly is

$$\mu^{(i+1)}(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V_x^{(i)}. \qquad (2.11)$$

To ensure convergence of the PI algorithm an initial admissible policy $\mu^{(0)}(x(t)) \in \Psi(\Omega)$ is required. It is in fact required by the desired completion of the first step in the policy iteration: i.e. finding a value associated with that initial policy (which needs to be admissible to have a finite value and for the nonlinear Lyapunov equation to have a solution). The algorithm then converges to the optimal control policy $\mu^* \in \Psi(\Omega)$ with corresponding cost $V^*(x)$. Proofs of convergence of the PI algorithm have been given in several references. See [4], [8], [13], [30], [35], [63], [91], [92], [93].

Policy iteration is a Newton method. In the linear time-invariant case, it reduces to the Kleinman algorithm [44] for solution of the Riccati equation, a familiar algorithm in control systems. Then, (2.9) become a Lyapunov equation.

2.2.3. Value function approximation (VFA)

The standard PI Algorithm just discussed proceeds by alternately updating the critic value and the actor policy by solving respectively the equations (2.9) and (2.11). In this chapter, the fundamental update equations in PI namely (2.9) for the value and (2.11) for the policy are used to design two neural networks. Then, by contrast to standard PI, it is shown how to tune these critic and actor neural networks *simultaneously* in real time to guarantee convergence to the control policy as well as stability during the training process.

The policy iteration algorithm, as other reinforcement learning algorithms, can be implemented on an actor/critic structure which consists of two neural network structures to approximate the solutions of the two equations (2.9) and (2.10) at each step of the iteration. The structure is presented in the next figure.

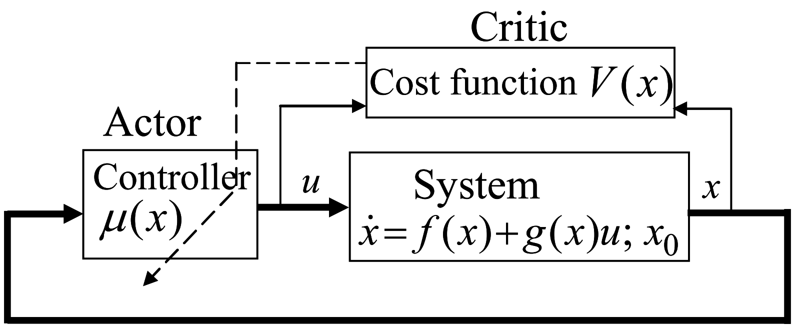

Figure 1. Actor/Critic Structure.

In the actor/critic structure [98], [99], [100] the cost $V^{\mu^{(i)}}(x(t))$ and the control $\mu^{(i+1)}(x)$ are approximated at each step of the PI algorithm by neural networks, called respectively the critic Neural Network (NN) and the actor NN. Then, the PI algorithm consists in tuning alternatively each of the two neural networks. The critic NN is tuned to solve (2.9) (in a least-squares sense [26]), and the actor NN to solve (2.11). Thus, while one NN is being tuned, the other is held constant. Note that, at each step in the iteration, the critic neural network is tuned to evaluate the performance of the current control policy.

The critic NN is based on value function approximation (VFA). In the following, it is desired to determine a rigorously justifiable form for the critic NN. Since one desires approximation in Sobolev norm, that is, approximation of the value $V(x)$ as well as its gradient, some discussion is given that relates normal NN approximation usage to the Weierstrass higher-order approximation theorem.

The solutions to the nonlinear Lyapunov equation (2.4), (2.9) may not be smooth for general nonlinear systems, except in a generalized sense [83]. However, in keeping with other work in the literature [77] we make the following assumptions.

Assumption 2.1. The solution to (2.4) is smooth, i.e. $V(x) \in C^1(\Omega)$.

Assumption 2.2. The solution to (2.4) is positive definite. This is guaranteed for stabilizable dynamics if the performance functional satisfies zero-state observability [77], which is guaranteed by the condition that $Q(x) > 0, x \in \Omega - \{0\}; Q(0) = 0$ be positive definite.

Assumption 2.1 allows us to bring in informal style of the Weierstrass higher-order approximation Theorem [4], [26] and the results of [34], which state that then there exists a complete independent basis set $\{\varphi_i(x)\}$ such that the solution $V(x)$ to (2.4) and its gradient are uniformly approximated, that is, there exist coefficients c_i such that

$$V(x) = \sum_{i=1}^{\infty} c_i \varphi_i(x) = \sum_{i=1}^{N} c_i \varphi_i(x) + \sum_{i=N+1}^{\infty} c_i \varphi_i(x)$$

$$V(x) \equiv C_1^T \phi_1(x) + \sum_{i=N+1}^{\infty} c_i \varphi_i(x) \tag{2.12}$$

$$\frac{\partial V(x)}{\partial x} = \sum_{i=1}^{\infty} c_i \frac{\partial \varphi_i(x)}{\partial x} = \sum_{i=1}^{N} c_i \frac{\partial \varphi_i(x)}{\partial x} + \sum_{i=N+1}^{\infty} c_i \frac{\partial \varphi_i(x)}{\partial x} \tag{2.13}$$

where $\phi_1(x) = [\varphi_1(x)\ \varphi_2(x) \cdots \varphi_N(x)]^T : \mathbb{R}^n \to \mathbb{R}^N$ and the last terms in these equations converge uniformly to zero as $N \to \infty$. (Specifically, the basis set is dense in the Sobolev norm $W^{1,\infty}$ [6].) Standard usage of the Weierstrass high-order approximation Theorem uses polynomial approximation. However, non-polynomial basis sets have been considered in the literature (e.g. [34], [76]).

Thus, it is justified to assume there exist weights W_1 such that the value function $V(x)$ is approximated as

$$V(x) = W_1^T \phi_1(x) + \varepsilon(x) \tag{2.14}$$

Then $\phi_1(x):\mathbb{R}^n \to \mathbb{R}^N$ is called the NN activation function vector, *N* the number of neurons in the hidden layer, and $\varepsilon(x)$ the NN approximation error. As per the above, the NN activation functions $\{\varphi_i(x): i=1,N\}$ are selected so that $\{\varphi_i(x): i=1,\infty\}$ provides a complete independent basis set such that $V(x)$ and its derivative

$$\frac{\partial V}{\partial x} = \left(\frac{\partial \phi_1(x)}{\partial x}\right)^T W_1 + \frac{\partial \varepsilon}{\partial x} = \nabla \phi_1^T W_1 + \nabla \varepsilon \qquad (2.15)$$

are uniformly approximated. Then, as the number of hidden-layer neurons $N \to \infty$, the approximation errors $\varepsilon \to 0,\ \nabla\varepsilon \to 0$ uniformly [26]. In addition, for fixed *N*, the NN approximation errors $\varepsilon(x)$, and $\nabla\varepsilon$ are bounded by constants on a compact set [34].

Using the NN value function approximation, considering a fixed control policy $u(t)$, the nonlinear Lyapunov equation (2.4) becomes

$$H(x,u,W_1) = W_1^T \nabla \phi_1 (f+gu) + Q(x) + u^T R u = \varepsilon_H \qquad (2.16)$$

where the residual error due to the function approximation error is

$$\varepsilon_H = -\left(\nabla\varepsilon\right)^T (f+gu) = -(C_1 - W_1)^T \nabla\phi_1 (f+gu) - \sum_{i=N+1}^{\infty} c_i \nabla\varphi_i(x)(f+gu) \qquad (2.17)$$

Under the Lipschitz assumption on the dynamics, this residual error is bounded on a compact set.

Define $|v|$ as the magnitude of a scalar *v*, $\|x\|$ as the vector norm of a vector *x*, and $\|\ \|_2$ as the induced matrix 2-norm.

Definition 2.2. (uniform convergence). A sequence of functions $\{p_n\}$ converges uniformly to *p* on a set Ω if $\forall \varepsilon > 0, \exists N(\varepsilon): \sup_{x\in\Omega} \|p_n(x) - p(x)\| < \varepsilon, n > N(\varepsilon)$.

The following Lemma has been shown in [4].

Lemma 2.1. For any admissible policy $u(t)$, the least-squares solution to (2.16) exists and is unique for each *N*. Denote this solution as W_1 and define

$$V_1(x) = W_1^T \phi_1(x) \tag{2.18}$$

Then, as $N \to \infty$:

$a.\ \ \sup_{x\in\Omega} |\varepsilon_H| \to 0$

$b.\ \ \|W_1 - C_1\|_2 \to 0$

$c.\ \ \sup_{x\in\Omega} |V_1 - V| \to 0$

$d.\ \ \sup_{x\in\Omega} \|\nabla V_1 - \nabla V\| \to 0$

■

This result shows that $V_1(x)$ converges uniformly in Sobolev norm $W^{1,\infty}$ [6] to the exact solution $V(x)$ to (2.4) as $N \to \infty$, and the weights W_1 converge to the first *N* of the weights, *C₁*, which exactly solve (2.4).

Since the object of interest in this chapter is finding the solution of the HJB using the above introduced function approximator, it is interesting now to look at the effect of the approximation error on the HJB equation (2.8)

$$W_1^T \nabla\varphi_1 f - \tfrac{1}{4} W_1^T \nabla\varphi_1 g R^{-1} g^T \nabla\varphi_1^T W_1 + Q(x) = \varepsilon_{HJB} \tag{2.19}$$

where the residual error due to the function approximation error is

$$\varepsilon_{HJB} = -\nabla\varepsilon^T f + \tfrac{1}{2} W_1^T \nabla\varphi_1 g R^{-1} g^T \nabla\varepsilon + \tfrac{1}{4} \nabla\varepsilon^T g R^{-1} g^T \nabla\varepsilon \tag{2.20}$$

It was also shown in [4] that this error converges uniformly to zero as the number of hidden layer units *N* increases. That is, $\forall \varepsilon > 0, \exists N(\varepsilon): \sup_{x\in\Omega} \|\varepsilon_{HJB}\| < \varepsilon$.

2.3 Tuning and Convergence of Critic NN

In this section we address the issue of tuning and convergence of the critic NN weights when a fixed admissible control policy is prescribed. Therefore, the focus is on the nonlinear Lyapunov equation (2.4) for a fixed policy u.

In fact, this amounts to the *design of an observer for the value function* which is known as ‘cost function’ in the optimal control literature. Therefore, this algorithm is consistent with adaptive control approaches which first design an observer for the system state and unknown dynamics, and then use this observer in the design of a feedback control.

The weights of the critic NN, W_1 which provide the best approximate solution for (2.16) are unknown. Therefore, the output of the critic neural network is

$$\hat{V}(x) = \hat{W}_1^T \phi_1(x) \tag{2.21}$$

where $\hat{W}_1$ are the current estimated values of the ideal critic NN weights W_1. Recall that $\phi_1(x): \mathbb{R}^n \to \mathbb{R}^N$ is the vector of activation functions, with N the number of neurons in the hidden layer. The approximate nonlinear Lyapunov equation is then

$$H(x, \hat{W}_1, u) = \hat{W}_1^T \nabla\phi_1(f + gu) + Q(x) + u^T R u = e_1 \tag{2.22}$$

In view of Lemma 2.1, define the critic weight estimation error

$$\tilde{W}_1 = W_1 - \hat{W}_1.$$

Then

$$e_1 = -\tilde{W}_1^T \nabla\phi_1(f + gu) + \varepsilon_H.$$

Given any admissible control policy u, it is desired to select $\hat{W}_1$ to minimize the squared residual error

$$E_1 = \tfrac{1}{2} e_1^T e_1.$$

Then $\hat{W}_1(t) \to W_1$ and $e_1 \to \varepsilon_H$. We select the tuning law for the critic weights as the normalized gradient descent algorithm

$$\dot{\hat{W}}_1 = -a_1 \frac{\partial E_1}{\partial \hat{W}_1} = -a_1 \frac{\sigma_1}{(\sigma_1^T \sigma_1 + 1)^2} [\sigma_1^T \hat{W}_1 + Q(x) + u^T R u] \tag{2.23}$$

where $\sigma_1 = \nabla\phi_1(f + gu)$. This is a modified gradient descent algorithm where $(\sigma_1^T \sigma_1 + 1)^2$ is used for normalization instead of $(\sigma_1^T\sigma_1+1)$. This is required in the proofs, where one needs both appearances of $\sigma_1/(1+\sigma_1^T\sigma_1)$ in (2.23) to be bounded [36], [87].

Note that, from (2.16),

$$Q(x)+u^T Ru=-W_1^T\nabla\varphi_1(f+gu)+\varepsilon_H. \tag{2.24}$$

Substituting (2.24) in (2.23) and, with the notation

$$\bar{\sigma}_1=\sigma_1/(\sigma_1^T\sigma_1+1)\ \ ,\ m_s=1+\sigma_1^T\sigma_1 \tag{2.25}$$

we obtain the dynamics of the critic weight estimation error as

$$\dot{\tilde{W}}_1=-a_1\bar{\sigma}_1\bar{\sigma}_1^{\mathrm{T}}\tilde{W}_1+a_1\bar{\sigma}_1\frac{\varepsilon_H}{m_s}. \tag{2.26}$$

Though it is traditional to use critic tuning algorithms of the form (2.23), it is not generally understood when convergence of the critic weights can be guaranteed. In this chapter, we address this issue in a formal manner. This development is motivated by adaptive control techniques that appear in [36], [87].

To guarantee convergence of $\hat{W}_1$ to W_1, the next Persistence of Excitation (PE) assumption and associated technical lemmas are required.

Persistence of Excitation (PE) Assumption. Let the signal $\bar{\sigma}_1$ be persistently exciting over the interval $[t,t+T]$, *i.e.* there exist constants $\beta_1>0$, $\beta_2>0$, $\mathrm{T}>0$ such that, for all *t*,

$$\beta_1\mathrm{I}\le S_0\equiv\int_t^{t+T}\bar{\sigma}_1(\tau)\bar{\sigma}_1^{\mathrm{T}}(\tau)d\tau\le\beta_2\mathrm{I}. \tag{2.27}$$

The PE assumption is needed in adaptive control if one desires to perform system identification using e.g. RLS [36]. It is needed here because one effectively desires to identify the critic parameters to approximate $V(x)$.

Technical Lemma 2.1. Consider the error dynamics system with output defined as

$$\dot{\tilde{W}}_1 = -a_1\bar{\sigma}_1\bar{\sigma}_1^{\mathrm{T}}\tilde{W}_1 + a_1\bar{\sigma}_1\frac{\varepsilon_H}{m_s}$$

$$y = \bar{\sigma}_1^{T}\tilde{W}_1 . \tag{2.28}$$

The PE condition (2.27) is equivalent to the uniform complete observability (UCO) [50] of this system, that is there exist constants $\beta_3 > 0$, $\beta_4 > 0$, $\mathrm{T} > 0$ such that, for all *t*,

$$\beta_3 \mathrm{I} \le S_1 \equiv \int_t^{t+T} \Phi^T(\tau,t)\bar{\sigma}_1(\tau)\bar{\sigma}_1^{\mathrm{T}}(\tau)\Phi(\tau,t)d\tau \le \beta_4 \mathrm{I}. \tag{2.29}$$

with $\Phi(t_1,t_0), t_0 \le t_1$ the state transition matrix of (2.28).

Proof: System (2.28) and the system defined by $\dot{\tilde{W}}_1 = a_1\bar{\sigma}_1 u,\ y = \bar{\sigma}_1^{T}\tilde{W}_1$ are equivalent under the output feedback $u = -y + \varepsilon_H / m_s$. Note that (2.27) is the observability gramian of this last system.

■

The importance of UCO is that bounded input and bounded output implies that the state $\tilde{W}_1(t)$ is bounded. In Theorem 2.1 we shall see that the critic tuning law (2.23) indeed guarantees boundedness of the output in (2.28).

Technical Lemma 2.2. Consider the error dynamics system (2.28). Let the signal $\bar{\sigma}_1$ be persistently exciting. Then:

a) The system (2.28) is exponentially stable. In fact if $\varepsilon_H = 0$ then $\|\tilde{W}(k\mathrm{T})\| \le e^{-\alpha kT}\|\tilde{W}(0)\|$ with

$$\alpha = -\frac{1}{\mathrm{T}}\ln(\sqrt{1-2a_1\beta_3}). \tag{2.30}$$

b) Let $\|\varepsilon_H\| \le \varepsilon_{\max}$ and $\|y\| \le y_{\max}$ then $\|\tilde{W}_1\|$ converges exponentially to the residual set

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1}\left\{\left[y_{\max} + \delta\beta_2 a_1\left(\varepsilon_{\max} + y_{\max}\right)\right]\right\}. \tag{2.31}$$

where δ is a positive constant of the order of 1.

Proof: See Appendix A.

■

The next result shows that the tuning algorithm (2.23) is effective under the PE condition, in that the weights $\hat{W}_1$ converge to the actual unknown weights W_1 which solve the nonlinear Lyapunov equation (2.16) for the given control policy $u(t)$. That is, (2.21) converges close to the actual value function of the current control policy.

Theorem 2.1. Let $u(t)$ be any admissible bounded control policy. Let tuning for the critic NN be provided by (2.23) and assume that $\bar{\sigma}_1$ is persistently exciting. Let the residual error in (2.16) be bounded $\|\varepsilon_H\| < \varepsilon_{\max}$. Then the critic parameter error converges exponentially with decay factor given by (2.30) to the residual set

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1}\left\{\left[1 + 2\delta\beta_2 a_1\right]\varepsilon_{\max}\right\}. \tag{2.32}$$

Proof:

Consider the following Lyapunov function candidate

$$L(t) = \frac{1}{2} tr\{\tilde{W}_1^T a_1^{-1} \tilde{W}_1\}. \tag{2.33}$$

The derivative of L is given by

$$\dot{L} = -tr\{\tilde{W}_1^T \frac{\sigma_1}{m_s^2}[\sigma_1^T \tilde{W}_1 - \varepsilon_H]\}$$

$$\dot{L} = -tr\{\tilde{W}_1^T \frac{\sigma_1 \sigma_1^T}{m_s^2} \tilde{W}_1\} + tr\{\tilde{W}_1^T \frac{\sigma_1}{m_s}\frac{\varepsilon_H}{m_s}\}$$

$$\dot{L} \le -\| \frac{\sigma_1^T}{m_s} \tilde{W}_1 \|^2 + \| \frac{\sigma_1^T}{m_s} \tilde{W}_1 \| \left\| \frac{\varepsilon_H}{m_s} \right\|$$

$$\dot{L} \le -\| \frac{\sigma_1^T}{m_s} \tilde{W}_1 \| \left[\| \frac{\sigma_1^T}{m_s} \tilde{W}_1 \| - \left\| \frac{\varepsilon_H}{m_s} \right\| \right]. \tag{2.34}$$

Therefore $\dot{L} \le 0$ if

$$\| \frac{\sigma_1^T}{m_s} \tilde{W}_1 \| > \varepsilon_{\max} > \left\| \frac{\varepsilon_H}{m_s} \right\|, \tag{2.35}$$

since $\|m_s\| \ge 1$.

This provides an effective practical bound for $\left\| \bar{\sigma}_1^T \tilde{W}_1 \right\|$, since $L(t)$ decreases if (2.35) holds.

Consider the estimation error dynamics (2.28) with the output bounded effectively by $\| y \| < \varepsilon_{\max}$, as just shown. Now Technical Lemma 2.2 shows exponential convergence to the residual set

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1} \left\{ \left[1 + 2a_1 \delta \beta_2 \right] \varepsilon_{\max} \right\}. \tag{2.36}$$

This completes the proof.

■

Remark 2.1. Note that, as $N \to \infty$, $\varepsilon_H \to 0$ uniformly [4]. This means that $\varepsilon_{\max}$ decreases as the number of hidden layer neurons in (2.21) increases.

Remark 2.2. This theorem requires the assumption that the control policy $u(t)$ is bounded, since $u(t)$ appears in ε_H. In the upcoming Theorem 2.2 this restriction is removed.

2.4 Action Neural Network and online synchronous policy iteration

We will now present an online adaptive PI algorithm which involves simultaneous, or synchronous, tuning of both the actor and critic neural networks. That is, the weights of both neural networks are tuned at the same time. This approach is a version of Generalized Policy Iteration (GPI), as introduced in [86]. In standard policy iteration, the critic and actor NN are tuned sequentially, with the weights of the other NN being held constant. By contrast, we tune both NN simultaneously in real-time.

It is desired to determine a rigorously justified form for the actor NN. To this end, let us consider one step of the Policy Iteration algorithm (2.9)-(2.11). Suppose that the solution $V(x) \in C^1(\Omega)$ to the nonlinear Lyapunov equation (2.9) for a given admissible policy $u(t)$ is given by (2.12). Then, according to (2.13) and (2.11) one has for the policy update

$$u = -\frac{1}{2} R^{-1} g^T(x) \sum_{i=1}^{\infty} c_i \nabla \varphi_i(x) \tag{2.37}$$

for some unknown coefficients c_i. Then one has the following result.

Lemma 2.2. Let the least-squares solution to (2.16) be W_1 and define

$$u_1(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla V_1(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla \phi_1^T(x) W_1 \tag{2.38}$$

with V_1 defined in (2.18).

Then, as $N \to \infty$:

a. $\sup_{x \in \Omega} \| u_1 - u \| \to 0$

b. There exists an N_0 such that $u_1(x)$ is admissible for $N > N_0$.

Proof: See [4].

In light of this result, the ideal control policy update is taken as (2.38), with W_1 unknown. Therefore, define the control policy in the form of an action neural network which computes the control input in the structured form

$$u_2(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \phi_1^T \hat{W}_2, \tag{2.39}$$

where $\hat{W}_2$ denotes the current estimated values of the ideal NN weights *W₁*. Define the actor NN estimation error as

$$\tilde{W}_2 = W_1 - \hat{W}_2 \tag{2.40}$$

The next definition and facts complete the machinery required for our main result.

Definition 2.3. [50] (UUB) The equilibrium point $x_e = 0$ of (2.1) is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset R^n$ so that for all $x_0 \in S$ there exists a bound *B* and a time $T(B, x_0)$ such that $\|x(t) - x_e\| \le B$ for all $t \ge t_0 + T$.

Facts 2.1.

a. $f(.)$*,* is Lipschitz, and $g(.)$ is bounded by a constant

$\|f(x)\| < b_f \|x\|, \quad \|g(x)\| < b_g$

b. The NN approx error and its gradient are bounded on a compact set containing Ω so that

$\|\varepsilon\| < b_\varepsilon$

$\|\nabla \varepsilon\| < b_{\varepsilon_x}$

c. The NN activation functions and their gradients are bounded so that

$\|\phi_1(x)\| < b_\phi$

$\|\nabla \phi_1(x)\| < b_{\phi_x}$

■

Fact 2.*1c* is satisfied, e.g. by sigmoids, *tanh*, and other standard NN activation functions.

We now present the main Theorem, which provides the tuning laws for the actor and critic neural networks that guarantee convergence of the synchronous online PI algorithm to the optimal controller, while guaranteeing closed-loop stability.

Theorem 2.2. Let the dynamics be given by (2.1), the critic NN be given by (2.21) and the control input be given by actor NN (2.39). Let tuning for the critic NN be provided by

$$\dot{\hat{W}}_1 = -a_1 \frac{\sigma_2}{(\sigma_2^T \sigma_2 + 1)^2} [\sigma_2{}^T \hat{W}_1 + Q(x) + u_2{}^T R u_2] \tag{2.41}$$

where $\sigma_2 = \nabla\phi_1(f + gu_2)$, and assume that $\bar{\sigma}_2 = \sigma_2 / (\sigma_2^T \sigma_2 + 1)$ is persistently exciting. Let the actor NN be tuned as

$$\dot{\hat{W}}_2 = -\alpha_2 \{(F_2 \hat{W}_2 - F_1 \bar{\sigma}_2^T \hat{W}_1) - \frac{1}{4} \overline{D}_1(x) \hat{W}_2 m^T(x) \hat{W}_1\} \tag{2.42}$$

where $\overline{D}_1(x) \equiv \nabla\phi_1(x) g(x) R^{-1} g^T(x) \nabla\phi_1{}^T(x)$, $m \equiv \dfrac{\sigma_2}{(\sigma_2{}^T \sigma_2 + 1)^2}$,

and $F_1 > 0$ and $F_2 > 0$ are tuning parameters. Let Assumptions 2.1-2.2 and facts 2.1 hold, and the tuning parameters be selected as detailed in the proof. Then there exists an N_0 such that, for the number of hidden layer units $N > N_0$ the closed-loop system state, the critic NN error $\tilde{W}_1$, and the actor NN error $\tilde{W}_2$ are UUB. Moreover, Theorem 2.1 holds with $\varepsilon_{\max}$ defined in the proof, so that exponential convergence of $\hat{W}_1$ to the approximate optimal critic value W_1 is obtained.

Proof: See appendix A.

■

Remark 2.3. Let $\varepsilon > 0$ and let N_0 be the number of hidden layer units above which $\sup_{x\in\Omega} \|\varepsilon_{HJB}\| < \varepsilon$. In the proof it is seen that the theorem holds for $N > N_0$. Additionally, ε provides an effective bound on the critic weight residual set in Theorem 2.1. That is, $\varepsilon_{\max}$ in (2.32) is effectively replaced by ε.

Remark 2.4. The theorem shows that PE is needed for proper identification of the value function by the critic NN, and that a nonstandard tuning algorithm is required for the actor NN to guarantee stability. The second term in (2.42) is a cross-product term that involves both the critic weights and the actor weights. It is needed to guarantee good behavior of the Lyapunov function, i.e. that the energy decreases to a bounded compact region.

Remark 2.5. The tuning parameters F_1 and F_2 in (2.42) must be selected to make the matrix *M* in (A.22) positive definite.

Note that the dynamics is required to implement this algorithm in that $\sigma_2 = \nabla\phi_1(f + gu_2)$, $\bar{D}_1(x)$, and (2.39) depend on $f(x), g(x)$.

2.5 Simulations

2.5.1. Linear System Example

Consider the continuous-time F16 aircraft plant with quadratic cost function used in [85]

$$\dot{x} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

where Q and R in the cost function are identity matrices of appropriate dimensions. In this linear case the solution of the HJB equation is given by the solution of the algebraic Riccati equation (ARE). Since the value is quadratic in the LQR case, the critic NN basis set $\phi_1(x)$ was selected as the quadratic vector in the state components. Solving the ARE gives the parameters of the optimal critic as

$$W_1^* = [1.4245 \quad 1.1682 \quad \text{-}0.1352 \quad 1.4349 \quad \text{-}0.1501 \; 0.4329]^T .$$

which are the components of the Riccati solution matrix *P*.

The synchronous PI algorithm is implemented as in Theorem 2.2. PE was ensured by adding a small probing noise to the control input. Figure 2 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \; W_{c4} \quad W_{c5} \quad W_{c6}]^T$$

converging to the optimal values. In fact after 800s the critic parameters converged to

$$\hat{W}_1(t_f) = [1.4279 \quad 1.1612 \quad -0.1366 \;\; 1.4462 \quad -0.1480 \; 0.4317]^T.$$

The actor parameters after 800s converge to the values of $\hat{W}_2(t_f) = \hat{W}_1(t_f)$.

Then, the actor NN is given by (2.39) as

$$\hat{u}_2(x) = -\tfrac{1}{2} R^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & 0 & x_1 \\ 0 & 2x_2 & 0 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}^T \hat{W}_2(t_f)$$

i.e. approximately the correct optimal control solution $u = -R^{-1} B^T P x$.

The evolution of the system states is presented in Figure 3. One can see that after 750s convergence of the NN weights in both critic and actor has occurred. This shows that the probing noise effectively guaranteed the PE condition. On convergence, the PE condition of the control signal is no longer needed, and the probing signal was turned off. After that, the states remain very close to zero, as required.

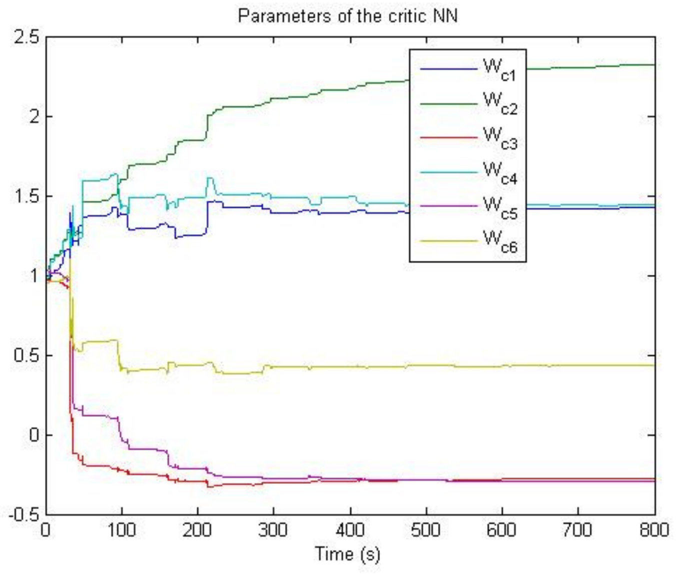

Figure 2. Convergence of the critic parameters to the parameters of the optimal critic.

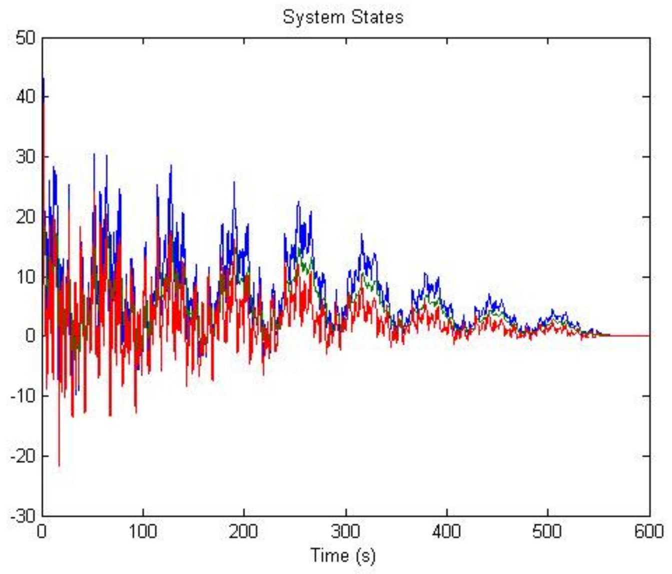

Figure 3. Evolution of the system states for the duration of the experiment.

2.5.2. Nonlinear System Example

Consider the following affine in control input nonlinear system, with a quadratic cost derived as in [64], [93]

$$\dot{x} = f(x) + g(x)u,\ x \in R^2$$

where

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2(1-(\cos(2x_1)+2)^2) \end{bmatrix}$$
$$g(x) = \begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}.$$

One selects $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ R = 1.$

Using the procedure in [64] the optimal value function is

$$V^*(x) = \frac{1}{2}x_1{}^2 + x_2{}^2$$

and the optimal control signal is

$$u^*(x) = -(\cos(2x_1)+2)x_2 .$$

One selects the critic NN vector activation function as

$$\phi_1(x) = [\; x_1^2 \quad x_1x_2 \quad x_2^2 \,]^T ,$$

Figure 4 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3}]^T .$$

These converge after about 80s to the correct values of

$$\hat{W}_1(t_f) = [0.5017 \quad -0.0020 \quad 1.0008]^T .$$

The actor parameters after 80s converge to the values of

$$\hat{W}_2(t_f) = [0.5017 \quad -0.0020 \quad 1.0008]^T .$$

So that the actor NN (2.39)

$$\hat{u}_2(x) = -\tfrac{1}{2}R^{-1}\begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.5017 \\ -0.0020 \\ 1.0008 \end{bmatrix}$$

also converged to the optimal control.

The evolution of the system states is presented in Figure 5. One can see that after 80s convergence of the NN weights in both critic and actor has occurred. This shows that the probing noise effectively guaranteed the PE condition. On convergence, the PE condition of the control signal is no longer needed, and the probing signal was turned off. After that, the states remain very close to zero, as required.

Figure 6 show the optimal value function. The identified value function given by $\hat{V}_1(x) = \hat{W}_1^T \phi_1(x)$ is virtually indistinguishable. In fact, Figure 7 shows the 3-D plot of the difference between the approximated value function, by using the online algorithm, and the optimal one. This error is close to zero. Good approximation of the actual value function is being evolved. Finally Figure 8 shows the 3-D plot of the difference between the approximated control, by using the online algorithm, and the optimal one. This error is close to zero.

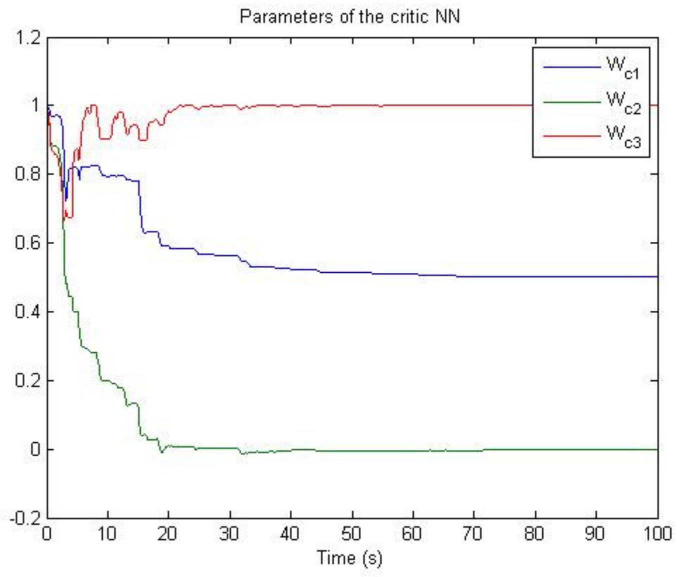

Figure 4. Convergence of the critic parameters.

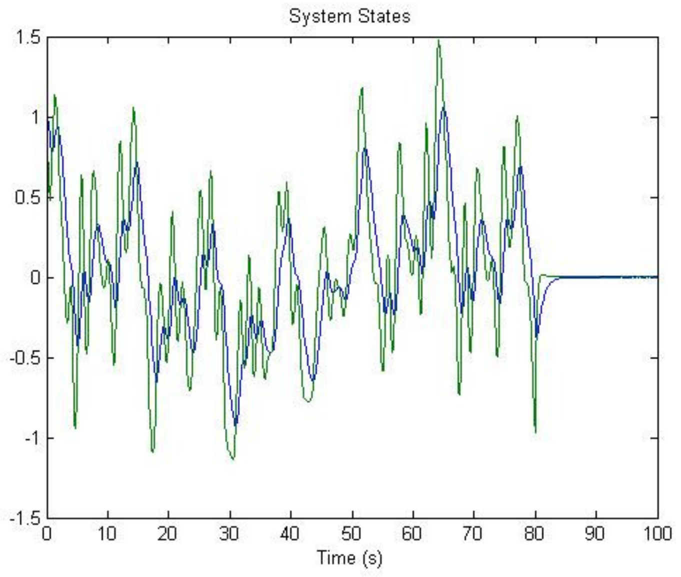

Figure 5. Evolution of the system states for the duration of the experiment.

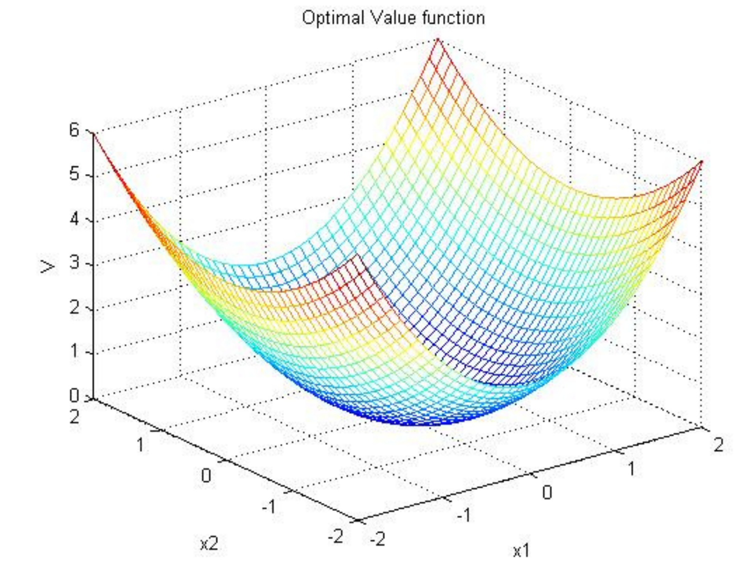

Figure 6. Optimal Value function.

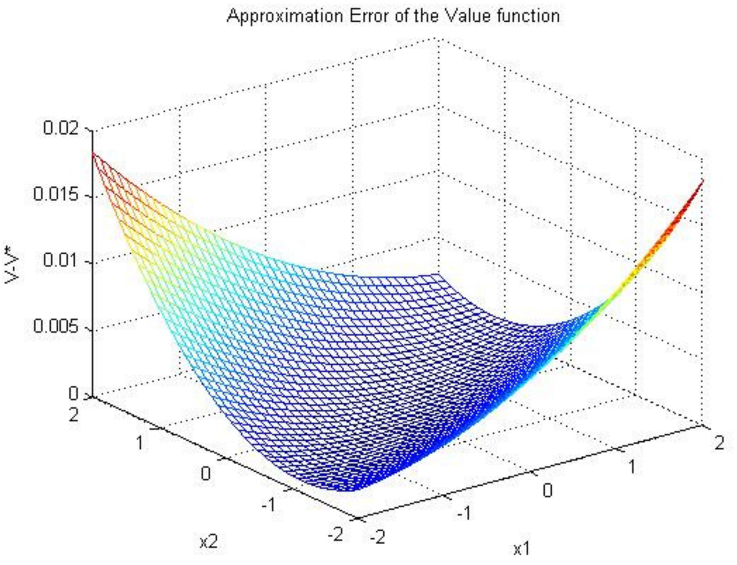

Figure 7. 3D plot of the approximation error for the value function.

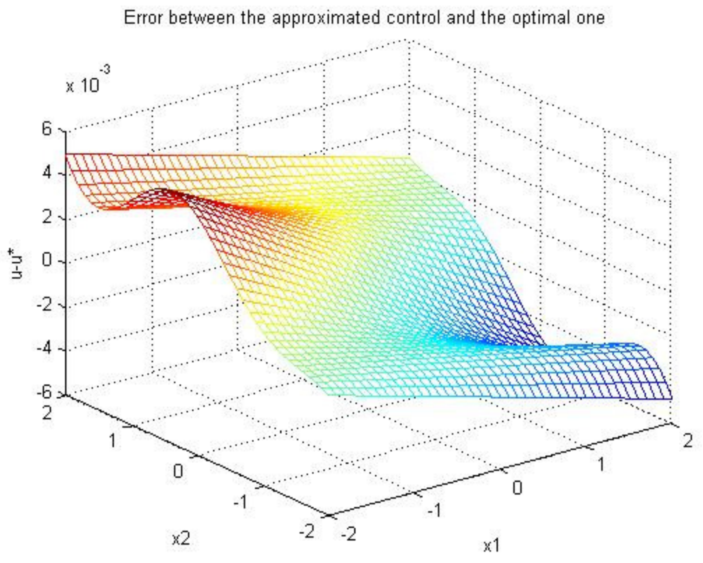

Figure 8. 3D plot of the approximation error for the control.

2.6 Conclusion

In this chapter we have proposed a new adaptive algorithm which solves the continuous-time optimal control problem for affine in the inputs nonlinear systems. We call this algorithm synchronous online PI for CT systems. The algorithm requires complete knowledge of the system model.

CHAPTER 3

ONLINE GAMING: REAL TIME SOLUTION OF NONLINEAR TWO-PLAYER ZERO-SUM GAMES USING SYNCHRONOUS POLICY ITERATION

3.1 Introduction

In this chapter we provide methods for online gaming, that is for solution of 2-player zero-sum infinite horizon games *online*, through learning the saddle point strategies in real-time. The dynamics may be nonlinear in continuous-time and are assumed known. A novel neural network adaptive control technique is given that is based on reinforcement learning techniques, whereby the control and disturbance policies are tuned online using data generated in real time along the system trajectories. Also tuned is a 'critic' approximator structure whose function is to identify the value or outcome of the current control and disturbance policies. Based on this value estimate, the policies are continuously updated. This is a sort of indirect adaptive control algorithm, yet, due to the direct form dependence of the policies on the learned value, it is affected online as direct ('optimal') adaptive control.

This chapter presents an optimal adaptive control method that converges online to the solution to the zero sum 2-player differential game (and hence the solution of the bounded L_2 gain problem). Three approximator structures are used. Parameter update laws are given to tune critic, actor, and disturbance neural networks simultaneously online to converge to the solution to the HJI equation and the saddle point policies, while also guaranteeing closed-loop stability. Rigorous proofs of performance and convergence are given.

The chapter is organized as follows. Section 3.2 reviews the formulation of the two-player zero-sum differential game. A policy iteration algorithm is given to solve the HJI equation by successive solutions on nonlinear Lyapunov-like equations. This essentially extends

Kleinman's algorithm to nonlinear zero-sum differential games. Section 3.3 describes the neural network value function approximation. First a suitable 'critic' approximator structure is developed for the value function and its tuning method is pinned down. A persistence of excitation is needed to guarantee proper convergence. Next, suitable 'actor' approximator structures are developed for the control and disturbance policies. Finally in section 3.4 the main results of the chapter are presented. Theorem 3.2, shows how to tune all three approximators simultaneously by using measurements along the system trajectories in real time and Theorem 3.3, proves exponential convergence to the critic neural network and convergence to the approximate Nash solution. Proofs using Lyapunov techniques guarantee convergence and closed-loop stability. Section 3.5 presents simulation examples that show the effectiveness of the online synchronous zero-sum game CT PI algorithm in learning the optimal value, control and disturbance for both linear and nonlinear systems.

3.2 Background: Two Player Differential Game and Policy Iteration

This section presents a background review of 2-player zero-sum differential games. The objective is to lay a foundation for the structure needed in subsequent sections for online solution of these problems in real-time. In this regard, the Policy Iteration Algorithm for 2-player games presented at the end of this section is key.

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

$$\dot{x} = f(x) + g(x)u(x) + k(x)d(x) \tag{3.1}$$

where state $x(t) \in \mathbb{R}^n$, $f(x(t)) \in \mathbb{R}^n$, $g(x(t)) \in \mathbb{R}^{nxm}$,control $u(x(t)) \in \mathbb{R}^m$, $k(x(t)) \in \mathbb{R}^{nxq}$ and disturbance $d(x(t)) \in \mathbb{R}^q$, Assume that $f(x)$ is locally Lipschitz, $f(0)=0$ so that $x=0$ is an equilibrium point of the system.

Define the performance index [53]

$$J(x(0),u,d) = \int_0^\infty \left(Q(x) + u^T Ru - \gamma^2 \|d\|^2 \right) dt \equiv \int_0^\infty r(x,u,d)\, dt \tag{3.2}$$

for $Q(x) \geq 0$, $R = R^T > 0$, $r(x,u,d) = Q(x) + u^T Ru - \gamma^2 \|d\|^2$ and $\gamma \geq \gamma^* \geq 0$, where γ^* is the smallest γ for which the system is stabilized [77]. For feedback policies $u(x)$ and disturbance policies $d(x)$, define the value or cost of the policies as

$$V(x(t),u,d) = \int_t^\infty \left(Q(x) + u^T Ru - \gamma^2 \|d\|^2 \right) dt \tag{3.3}$$

When the value is finite, a differential equivalent to this is the nonlinear Lyapunov-like equation

$$0 = r(x,u,d) + \left(\nabla V\right)^T (f(x) + g(x)u(x) + k(x)d(x)), \; V(0) = 0 \tag{3.4}$$

where $\nabla V = \partial V / \partial x \in R^n$ is the (transposed) gradient and the Hamiltonian is

$$H(x,\nabla V,u,d) = r(x,u,d) + \left(\nabla V\right)^T (f(x) + g(x)u(x) + k(x)d) \tag{3.5}$$

For feedback policies [1], a solution $V(x) \geq 0$ to (3.4) is the value (3.3) for given feedback policy $u(x)$ and disturbance policy $d(x)$.

3.2.1. Two-player zero-sum differential games and Nash equilibrium

Define the 2-player zero-sum differential game [11], [1]

$$V^*(x(0)) = \min_u \max_d J(x(0),u,d) = \min_u \max_d \int_0^\infty \left(Q(x) + u^T Ru - \gamma^2 \|d\|^2 \right) dt \tag{3.6}$$

subject to the dynamical constraints (3.1). Thus, u is the minimizing player and d is the maximizing one. This 2-player optimal control problem has a unique solution if a game theoretic saddle point exists, i.e., if the Nash condition holds

$$\min_u \max_d J(x(0),u,d) = \max_d \min_u J(x(0),u,d) \tag{3.7}$$

A necessary condition for this is Isaacs' condition

$$\min_u \max_d H(x,\nabla V,u,d) = \max_d \min_u H(x,\nabla V,u,d) \tag{3.8}$$

or, equivalently

$$H(x,\nabla V,u^*,d) \le H(x,\nabla V,u^*,d^*) \le H(x,\nabla V,u,d^*) \tag{3.9}$$

for some saddle point solution (u^*,d^*).

To this game is associated the Hamilton-Jacobi-Isaacs (HJI) equation

$$0 = Q(x) + \nabla V^T(x) f(x) - \frac{1}{4}\nabla V^T(x) g(x) R^{-1} g^T(x) \nabla V(x) + \frac{1}{4\gamma^2}\nabla V^T(x) k k^T \nabla V(x), \qquad V(0)=0 \tag{3.10}$$

Given a solution $V^*(x) \ge 0 : \mathbb{R}^n \to \mathbb{R}$ to the HJI (3.10), denote the associated control and disturbance as

$$u^* = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V^* \tag{3.11}$$

$$d^* = \frac{1}{2\gamma^2} k^T(x) \nabla V^* \tag{3.12}$$

and write

$$0 = H(x,\nabla V,u^*,d^*) = Q(x) + \nabla V^T(x) f(x) - \frac{1}{4}\nabla V^T(x) g(x) R^{-1} g^T(x) \nabla V(x) + \frac{1}{4\gamma^2}\nabla V^T(x) k k^T \nabla V(x) \tag{3.13}$$

Lemma 3.1. Isaacs condition satisfied. Select $\gamma > 0$. Suppose $V(x): R^n \to R$ is smooth, $V(x) \ge 0$, and is a solution to the HJI equation (3.10). Then the Isaacs condition (3.8) is satisfied for control $u^*(t)$ given by (3.11) and $d^*(t)$ given by (3.12) in terms of $V(x)$.

Proof : If $V(x) \ge 0$ satisfies the HJI, then complete the squares to obtain

$$H(x,\nabla V,u,d) = H(x,\nabla V,u^*,d^*) - \gamma^2 \left\| d-d^* \right\|^2 + (u-u^*)^T R (u-u^*) = -\gamma^2 \left\| d-d^* \right\|^2 + (u-u^*)^T R (u-u^*) \tag{3.14}$$

with $u(t) = u^*(t), d(t) = d^*(t)$ given respectively by (3.11), (3.12). This implies (3.9).

■

The next result shows that this implies (3.7) under certain conditions. System (3.1) is said to be *zero state observable* if $u(t) \equiv 0, y(t) \equiv 0 \Rightarrow x(t) \equiv 0$. The HJI equation (3.13) may have more than one nonnegative definite solution $V(x) \ge 0$. A minimal non negative definite solution $V_a(x) \ge 0$ is one such that there exists no other non negative definite solution $V(x) \ge 0$ such that $V_a(x) \ge V(x) \ge 0$.

Lemma 3.2. Solution of 2-player zero-sum game. [1] Select $\gamma > 0$. Suppose $V_a(x): R^n \to R$ is smooth, $V_a(x) \ge 0$, and is a minimal non negative definite solution to the HJI equation (3.13). Assume, (3.1) is zero state observable. Then condition (3.7) is satisfied for control $u^*(t)$ given by (3.11) and $d^*(t)$ given by (3.12) in terms of $V_a(x)$. Then the system is in Nash equilibrium, the game has a value given by $V_a(x(0))$, and (u^*, d^*) is a saddle point equilibrium solution among strategies in $L_2[0,\infty)$. Moreover, the closed-loop systems $(f(x)+g(x)u^*)$ and $(f(x)+g(x)u^*+k(x)d^*)$ are locally asymptotically stable.

Proof Outline: One has for any smooth $V(x): \mathbb{R}^n \to \mathbb{R}$

$$J_T(x(0),u,d) \equiv \int_0^T \left(h^T(x)h(x) + u^T Ru - \gamma^2 \|d\|^2 \right) dt = \int_0^T \left(h^T(x)h(x) + u^T Ru - \gamma^2 \|d\|^2 \right) dt$$
$$+ \int_0^T \dot{V} dt \quad - V(x(T)) + V(x(0))$$

$$= \int_0^T H(x, \nabla V, u, d)\, dt \quad - V(x(T)) + V(x(0))$$

Now, suppose $V(x)$ satisfies the HJI equation (3.13) and use (3.14) to obtain

$$J_T(x(0),u,d) = \int_0^T \left((u-u^*)^T R(u-u^*) - \gamma^2 \left\| d - d^* \right\|^2 \right) dt \quad - V^*(x(T)) + V^*(x(0))$$

Therefore, for finite *T*, $J_T(x(0),u,d)$ has a saddle point induced by that of the Hamiltonian. The remainder of the proof involves showing that, for the specific HJI solution selected, the closed-loop system $(f(x)+g(x)u^*+k(x)d^*)$ is locally asymptotically stable. Then taking the limit as $T \to \infty$ yields the result.

■

The issue in this proof is that existence of a Nash equilibrium on an infinite horizon requires that the closed-loop system $(f(x)+g(x)u^*+k(x)d^*)$ be locally asymptotically stable. It is proven in [1] that the minimum non negative definite solution to the HJI is the unique non negative definite solution for which this is true. Linearize the system (3.1) about the origin to obtain the Generalized ARE. Of the non negative definite solutions to the GARE, select the one corresponding to the stable invariant manifold of the Hamiltonian matrix. Then, the minimum non negative definite solution of the HJI is the one having this stabilizing GARE solution as its Hessian matrix evaluated at the origin [77].

The next result provides a solution to the bounded L_2 gain problem.

Lemma 3.3. Solution to the Bounded L_2 gain problem. [77] Select $\gamma > 0$. Suppose $V(x): \mathbb{R}^n \to \mathbb{R}$ is smooth, $V(x) \geq 0$, and is a solution to the HJI equation (3.13). Assume, (3.1) is zero state observable. Then system (3.1) has L_2 gain $\leq \gamma$. Moreover the control $u^*(t)$ selected as (3.11) in terms of the HJI solution solves the L_2 gain problem and renders the equilibrium point locally asymptotically stable (when $d(t)=0$), e.g. $(f(x)+g(x)u^*)$ is locally asymptotically stable. Finally, $u^*(t) \in L_2[0,\infty)$.

■

Note that under the hypotheses of the Lemma, specifically zero state observability, if $V(x) \geq 0$ is a solution to the HJI equation (3.13), then one has in fact $V(x) > 0$ [77].

Lemma 3.4. [77] Select $\gamma > 0$. Assume (3.1) is locally detectable from $y = h(x)$, and there exists a control policy $u(x)$ so that locally the system has L_2 gain$\leq \gamma$ and the system is asymptotically stable. Then there exists a local smooth solution $V(x) \geq 0$ to the HJI equation (3.13). Furthermore, the control given in terms of this solution by (3.11) renders the L_2 gain $\leq \gamma$ for all trajectories originating at the origin locally.

■

Note that global solutions to the HJI (3.13) may not exist. Moreover, if they do, they may not be smooth. For a discussion on viscosity solutions to the HJI, see [9], [10], [1]. The HJI equation (3.13) may have more than one nonnegative local smooth solution $V(x) \geq 0$. A minimal nonnegative solution $V_a(x) \geq 0$ is one such that there exists no other nonnegative solution $V(x) \geq 0$ such that $V_a(x) \geq V(x) \geq 0$. Linearize the system (3.1) about the origin to obtain the Generalized ARE (See Section 3.5.1). Of the nonnegative solutions to the GARE, select the one corresponding to the stable invariant manifold of the Hamiltonian matrix. Then, the minimum nonnegative solution of the HJI is the one having this stabilizing GARE solution as its Hessian matrix evaluated at the origin [77].

It is shown in [1] that if $V^*(x)$ is the minimum non-negative solution to the HJI (3.13) and (3.1) is locally detectable, then (3.11), (3.12) given in terms of $V^*(x)$ are in Nash equilibrium solution to the zero-sum game and $V^*(x)$ is its value.

3.2.2. Policy Iteration solution of the HJI equation

The HJI equation (3.13) is usually intractable to solve directly. One can solve the HJI iteratively using one of several algorithms that are built on iterative solutions of the Lyapunov equation (3.4). Included are [25] which uses an inner loop with iterations on the control, and [3], [5], [77] which uses an inner loop with iterations on the disturbance. These are in effect extensions of Kleinman's algorithm [44] to nonlinear 2-player games. The complementarity of these algorithms is shown in [91], [92]. Here, we shall use the latter algorithm (e.g.[3], [5], [77]).

The structure given in the algorithm below will be used as the basis for approximate online solution techniques in the next section.

Policy Iteration (PI) Algorithm for 2-Player Zero-Sum Differential Games

Initialization: Start with a stabilizing feedback control policy u_0

1. For $j = 0,1,...$ given u_j

2. For $i = 0,1,...$ set $d^0 = 0$, solve for $V_j^i(x(t))$, d^{i+1} using

$$0 = Q(x) + \nabla V_j^{iT}(x)(f + gu_j + kd^i) + u_j^T R u_j - \gamma^2 \left\| d^i \right\|^2 \tag{3.15}$$

$$d^{i+1} = \arg\max_d [H(x, \nabla V_j^i, u_j, d)] = \frac{1}{2\gamma^2} k^T(x) \nabla V_j^i \tag{3.16}$$

On convergence, set $V_{j+1}(x) = V_j^i(x)$

3. Update the control policy using

$$u_{j+1} = \arg\min_u [H(x, \nabla V_{j+1}), u, d] = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V_{j+1} \tag{3.17}$$

Go to 1.

■

Nota Benne: In practice, the iterations in i and j are continued until some convergence criterion is met, e.g. $\left\| V_j^{i+1} - V_j^i \right\|$ or, respectively $\left\| V_{j+1} - V_j \right\|$ is small enough in some suitable norm.

Given a feedback policy $u(x)$, write the Hamilton-Jacobi (HJ) equation

$$0 = Q(x) + \nabla V^T(x)\big(f(x) + g(x)u(x)\big) + u^T(x) R u(x) + \frac{1}{4\gamma^2} \nabla V^T(x) k k^T \nabla V(x), \qquad V(0) = 0 \tag{3.18}$$

for fixed $u(x)$. The minimal nonnegative solution $V(x)$ to this equation is the so-called available storage for the given $u(x)$ [77]. Note that the inner loop of this algorithm finds the available storage for u_j, where it exists.

Assuming that the available storage at each index j is smooth on a local domain of validity, the convergence of this algorithm to the minimal nonnegative solution to the HJI equation is shown in [5], [77]. Under these assumptions, the existence of smooth solutions at each step to the Lyapunov-like equation (3.15) was further shown in [5]. Also shown was the asymptotic stability of $(f + gu_j + kd^i)$ at each step. In fact, the inner loop yields $V_j^{i+1}(x) \ge V_j^i(x), \forall x$ while the outer loop yields $V_{j+1}(x) \le V_j(x), \forall x$ until convergence to V^*. Note that this algorithm relies on successive solutions of nonlinear Lyapunov-like equations (3.15).

As such, the discussion surrounding (3.4) shows that the algorithm finds the value $V_j^i(x(t))$ of successive control policy/disturbance policy pairs.

3.3 Neural Network approximation structure for online policy iteration algorithm

The PI Algorithm is a *sequential* algorithm that solves the HJI equation (3.13) and finds the Nash solution (u^*,d^*) based on sequential solutions of the nonlinear Lyapunov equation(3.15). That is, while the disturbance policy is being updated, the feedback policy is held constant. In this section, we use PI to lay a rigorous foundation for the NN approximator structure required *on-line solution of the 2-player zero-sum differential game in real time.* In the next section, this structure will be used to develop an adaptive control algorithm of novel form that converges to the zero-sum (ZS) game solution. It is important to define the neural network structures and the NN estimation errors properly or such an adaptive algorithm cannot be developed.

The PI algorithm itself is not implemented in this work. Instead, here one implements both loops, the outer feedback control update loop and the inner disturbance update loop, *simultaneously* using neural network learning implemented as differential equations for tuning the weights, while simultaneously keeping track of and learning the value $V(x(t),u,d)$ (3.3) of the current control and disturbance by solution of the Lyapunov equation (3.4)/(3.15). We call this *synchronous PI* for zero-sum games.

3.3.1. Value function approximation: Critic Neural Network

This work uses nonlinear approximator structures (e.g. neural networks) for Value Function Approximation (VFA) [15], [98], [99], therefore sacrificing some representational accuracy in order to make the representation manageable in practice. Sacrificing accuracy in the representation of the value function is not so critical, since the ultimate goal is to find an optimal policy and not necessarily an accurate value function. Based on the structure of the PI algorithm in Section 3.2.2, VFA for online 2-player games requires three approximators, which are taken as neural networks (NN), one for the value function, one for the feedback control policy, and one for the disturbance policy. These are motivated respectively by the need to solve equations (3.15), (3.17), and (3.16).

To solve equation (3.15), we use VFA, which here requires approximation in Sobolev norm [6], that is, approximation of the value $V(x)$ as well as its gradient $\nabla V(x)$. The following definition describes uniform convergence that is needed later.

Definition 3.1. (uniform convergence). A sequence of functions $\{f_n\}$ converges uniformly to f if $\forall \varepsilon > 0, \exists N(\varepsilon):\ \sup\|f_n(x) - f(x)\| < \varepsilon, n > N(\varepsilon)$.

According to the Weierstrass higher-order approximation Theorem [4], [6], [26], [34], there exists a complete independent basis set $\{\varphi_i(x)\}$ such that the solution $V(x)$ to (3.4) and its gradient are uniformly approximated, that is, there exist coefficients c_i such that

$$V(x) = \sum_{i=1}^{\infty} c_i \varphi_i(x) = \sum_{i=1}^{N} c_i \varphi_i(x) + \sum_{i=N+1}^{\infty} c_i \varphi_i(x)$$

$$V(x) \equiv C_1^T \phi_1(x) + \sum_{i=N+1}^{\infty} c_i \varphi_i(x) \tag{3.19}$$

$$\frac{\partial V(x)}{\partial x} = \sum_{i=1}^{\infty} c_i \frac{\partial \varphi_i(x)}{\partial x} = \sum_{i=1}^{N} c_i \frac{\partial \varphi_i(x)}{\partial x} + \sum_{i=N+1}^{\infty} c_i \frac{\partial \varphi_i(x)}{\partial x} \tag{3.20}$$

where $\phi_1(x) = [\varphi_1(x)\, \varphi_2(x) \cdots \varphi_N(x)]^T : \mathbb{R}^n \to \mathbb{R}^N$, and the second terms in these equations converge uniformly to zero as $N \to \infty$. Specifically, the linear subspace generated by the basis set is dense in the Sobolev norm $W^{1,\infty}$ [6].

Therefore, assume there exist NN weights W_1 such that the value function $V(x)$ is approximated as

$$V(x) = W_1^T \phi_1(x) + \varepsilon(x) \tag{3.21}$$

with $\phi_1(x) : \mathbb{R}^n \to \mathbb{R}^N$ the NN activation function vector, *N* the number of neurons in the hidden layer, and $\varepsilon(x)$ the NN approximation error. For approximation in Sobolev space, the NN activation functions $\{\varphi_i(x) : i = 1, N\}$ should be selected so that $\{\varphi_i(x) : i = 1, \infty\}$ provides a complete independent basis set such that $V(x)$ and its derivative are uniformly approximated, e.g., additionally

$$\frac{\partial V}{\partial x} = \left(\frac{\partial \phi_1(x)}{\partial x} \right)^T W_1 + \frac{\partial \varepsilon}{\partial x} = \nabla \phi_1{}^T W_1 + \nabla \varepsilon \tag{3.22}$$

Then, as the number of hidden-layer neurons $N \to \infty$, the approximation errors $\varepsilon \to 0,\ \nabla\varepsilon \to 0$ uniformly [6], [26]. In addition, for fixed *N*, the NN approximation errors $\varepsilon(x)$, and $\nabla\varepsilon$ are bounded by constants locally [34].

We refer to the NN with weights W_1 that performs VFA as the *critic* NN.

Standard usage of the Weierstrass high-order approximation Theorem uses polynomial approximation. However, non-polynomial basis sets have been considered in the literature (e.g.[34], [76]). The NN approximation literature has considered a variety of activation functions including sigmoids, tanh, radial basis functions, etc.

Using the NN VFA, considering fixed feedback and disturbance policies $u(x(t)), d(x(t))$, equation (3.4) becomes

$$H(x, W_1, u, d) = Q(x) + u^T R u - \gamma^2 \|d\|^2 + W_1^T \nabla \phi_1 (f(x) + g(x)u(x) + k(x)d(x)) = \varepsilon_H \tag{3.23}$$

where the residual error is

$$\varepsilon_H = -(\nabla \varepsilon)^T (f + gu + kd) = -(C_1 - W_1)^T \nabla \phi_1 (f + gu + kd) - \sum_{i=N+1}^{\infty} c_i \nabla \varphi_i(x)(f + gu + kd) \tag{3.24}$$

Under the Lipschitz assumption on the dynamics, this residual error is bounded locally.

The following Proposition has been shown in [3], [6].

Proposition 3.1. For any policies $u(x(t)), d(x(t))$ the least-squares solution to (3.23) exists and is unique for each *N*. Denote this solution as W_1 and define

$$V_1(x) = W_1^T \phi_1(x) \tag{3.25}$$

Then, as $N \to \infty$:

a. $\sup|\varepsilon_H| \to 0$

b. $\|W_1 - C_1\|_2 \to 0$

c. $\sup|V_1 - V| \to 0$

d. $\sup\|\nabla V_1 - \nabla V\| \to 0$

■

This result shows that $V_1(x)$ converges uniformly in Sobolev norm $W^{1,\infty}$ [6] to the exact solution $V(x)$ to (3.4) as $N \to \infty$, and the weights W_1 converge to the first *N* of the weights, C_1, which exactly solve (3.4).

The effect of the approximation error on the HJI equation (3.10) is

$$Q(x) + W_1^T \nabla\varphi_1(x) f(x) - \frac{1}{4} W_1^T \nabla\varphi_1(x) g(x) R^{-1} g^T(x) \nabla\varphi_1^T W_1 + \frac{1}{4\gamma^2} W_1^T \nabla\varphi_1(x) kk^T \nabla\varphi_1^T W_1 = \varepsilon_{HJI} \tag{3.26}$$

where the residual error due to the function approximation error is

$$\varepsilon_{HJI} \equiv -\nabla\varepsilon^T f + \tfrac{1}{2} W_1^T \nabla\varphi_1 g R^{-1} g^T \nabla\varepsilon + \tfrac{1}{4} \nabla\varepsilon^T g R^{-1} g^T \nabla\varepsilon - \tfrac{1}{2\gamma^2} W_1^T \nabla\varphi_1 kk^T \nabla\varepsilon - \tfrac{1}{4\gamma^2} \nabla\varepsilon^T kk^T \nabla\varepsilon \tag{3.27}$$

It was also shown in [3], [6] that this error converges uniformly to zero as the number of hidden layer units *N* increases. That is, $\forall\varepsilon > 0, \exists N(\varepsilon): \sup\|\varepsilon_{HJI}\| < \varepsilon, N > N(\varepsilon)$.

3.3.2. Tuning and Convergence of the Critic Neural Network

In this section are addressed the tuning and convergence of the critic NN weights when fixed feedback control and disturbance policies are prescribed. Therefore, the focus is on solving the nonlinear Lyapunov-like equation (3.4) (e.g. (3.15)) for a fixed feedback policy u and fixed disturbance policy d.

In fact, this amounts to the *design of an observer for the value function*. Therefore, this algorithm is consistent with adaptive control approaches which first design an observer for the system state and unknown dynamics, and then use this observer in the design of a feedback control.

The ideal weights of the critic NN, W_1 which provide the best approximate solution for (3.23) are unknown. Therefore, the output of the critic neural network is

$$\hat{V}(x) = \hat{W}_1^T \phi_1(x) \tag{3.28}$$

where $\hat{W}_1$ are the current estimated values of W_1. The approximate nonlinear Lyapunov-like equation is then

$$H(x, \hat{W}_1, u, d) = \hat{W}_1^T \nabla\phi_1(f + gu + kd) + Q(x) + u^T Ru - \gamma^2 \|d\|^2 = e_1 \tag{3.29}$$

with e_1 a residual equation error. In view of Proposition 3.1, define the critic weight estimation error

$$\tilde{W}_1 = W_1 - \hat{W}_1 .$$

Then,

$$e_1 = -\tilde{W}_1^T \nabla\phi_1(f + gu) + \varepsilon_H .$$

Given any feedback control policy u, it is desired to select $\hat{W}_1$ to minimize the squared residual error

$$E_1 = \tfrac{1}{2} e_1^T e_1 .$$

Then $\hat{W}_1(t) \to W_1$ and $e_1 \to \varepsilon_H$. Select the tuning law for the critic weights as the normalized gradient descent algorithm

$$\dot{\hat{W}}_1 = -a_1 \frac{\partial E_1}{\partial \hat{W}_1} = -a_1 \frac{\sigma_1}{(1 + \sigma_1^{\mathrm{T}} \sigma_1)^2} [\sigma_1^{\mathrm{T}} \hat{W}_1 + Q(x) + u^T Ru - \gamma^2 \|d\|^2] \tag{3.30}$$

where $\sigma_1 = \nabla\phi_1(f + gu + kd)$. This is a nonstandard modified gradient descent algorithm where $(\sigma_1^T \sigma_1 + 1)^2$ is used for normalization instead of $(\sigma_1^T \sigma_1 + 1)$. This is required in the theorem proofs, where one needs both appearances of $\sigma_1 / (1 + \sigma_1^T \sigma_1)$ in (3.30) to be bounded [36], [87].

Note that, from (3.23),

$$Q(x)+u^T Ru-\gamma^2\|d\|^2=-W_1^T\nabla\varphi_1(f+gu+kd)+\varepsilon_H. \tag{3.31}$$

Substituting (3.31) in (3.30) and, with the notation

$$\bar{\sigma}_1=\sigma_1/(\sigma_1^T\sigma_1+1)\ ,\ m_s=1+\sigma_1^T\sigma_1 \tag{3.32}$$

we obtain the dynamics of the critic weight estimation error as

$$\dot{\tilde{W}}_1=-a_1\bar{\sigma}_1\bar{\sigma}_1^{\mathrm{T}}\tilde{W}_1+a_1\bar{\sigma}_1\frac{\varepsilon_H}{m_s}. \tag{3.33}$$

To guarantee convergence of $\hat{W}_1$ to W_1, the next Persistence of Excitation (PE) assumption and associated technical lemmas are required.

Persistence of Excitation (PE) Assumption. Let the signal $\bar{\sigma}_1$ be persistently exciting over the interval $[t,t+T]$, *i.e.* there exist constants $\beta_1>0$, $\beta_2>0$, $\mathrm{T}>0$ such that, for all *t*,

$$\beta_1\mathrm{I}\le S_0\equiv\int_t^{t+T}\bar{\sigma}_1(\tau)\bar{\sigma}_1^{\mathrm{T}}(\tau)d\tau\le\beta_2\mathrm{I}. \tag{3.34}$$

with I the identity matrix of appropriate dimensions.

The PE assumption is needed in adaptive control as described thoroughly in Chapter 2.

The properties of tuning algorithm (3.30) are given in the subsequent results. They are proven in Chapter 2.

Technical Lemma 3.1. Consider the error dynamics system with output defined as

$$\dot{\tilde{W}}_1=-a_1\bar{\sigma}_1\bar{\sigma}_1^{\mathrm{T}}\tilde{W}_1+a_1\bar{\sigma}_1\frac{\varepsilon_H}{m_s}$$

$$y=\bar{\sigma}_1^T\tilde{W}_1. \tag{3.35}$$

The PE condition (3.34) is equivalent to the uniform complete observability (UCO) [50]of this system, that is there exist constants $\beta_3>0$, $\beta_4>0$, $\mathrm{T}>0$ such that, for all *t*,

$$\beta_3\mathrm{I}\le S_1\equiv\int_t^{t+T}\Phi^T(\tau,t)\bar{\sigma}_1(\tau)\bar{\sigma}_1^{\mathrm{T}}(\tau)\Phi(\tau,t)d\tau\le\beta_4\mathrm{I}. \tag{3.36}$$

with $\Phi(t_1,t_0), t_0 \le t_1$ the state transition matrix of (3.35) and I the identity matrix of appropriate dimensions.

■

Technical Lemma 3.2. Consider the error dynamics system (3.35). Let the signal $\bar{\sigma}_1$ be persistently exciting. Then:

a) The system (3.35) is exponentially stable. In fact if $\varepsilon_H = 0$ then $\|\tilde{W}(k\mathrm{T})\| \le e^{-\alpha kT} \|\tilde{W}(0)\|$ with

$$\alpha = -\frac{1}{\mathrm{T}} \ln(\sqrt{1-2a_1\beta_3}) . \tag{3.37}$$

b) Let $\|\varepsilon_H\| \le \varepsilon_{\max}$ and $\|y\| \le y_{\max}$. Then $\|\tilde{W}_1\|$ converges exponentially to the residual set

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1} \left\{ \left[y_{\max} + \delta\beta_2 a_1 \left(\varepsilon_{\max} + y_{\max} \right) \right] \right\} . \tag{3.38}$$

where δ is a positive constant of the order of 1. ■

The next result shows that the tuning algorithm (3.30) is effective under the PE condition, in that the weights $\hat{W}_1$ converge to the actual unknown weights W_1 which solve the nonlinear Lyapunov-like equation (3.23) in a least-squares sense for the given feedback and disturbance policies $u(x(t)), d(x(t))$. That is, (3.28) converges close to the actual value function of the current policies. The proof is in Chapter 2 in Theorem 2.1.

Theorem 3.1. Let $u(x(t)), d(x(t))$ be any bounded policies. Let tuning for the critic NN be provided by (3.30) and assume that $\bar{\sigma}_1$ is persistently exciting. Let the residual error in (3.23) be bounded $\|\varepsilon_H\| < \varepsilon_{\max}$. Then the critic parameter error converges exponentially with decay factor given by (3.37) to the residual set

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1} \left\{ \left[1 + 2\delta\beta_2 a_1 \right] \varepsilon_{\max} \right\} . \tag{3.39}$$

■

Remark 3.1. Note that, as $N \to \infty$, $\varepsilon_H \to 0$ uniformly [3], [6]. This means that $\varepsilon_{\max}$ decreases as the number of hidden layer neurons in (3.28) increases.

Remark 3.2. This theorem requires the assumption that the feedback policy $u(x(t))$ and the disturbance policy $d(x(t))$ are bounded, since the policies appear in (3.24). In the upcoming Theorems 3.2 and 3.3 this restriction is removed.

3.3.3. Action and Disturbance Neural Network

It is important to define the neural network structure and the NN estimation errors properly for the control and disturbance or an adaptive algorithm cannot be developed. To determine a rigorously justified form for the actor and the disturbance NN, consider one step of the Policy Iteration algorithm (3.15)-(3.17). Suppose that the solution $V(x)$ to the nonlinear Lyapunov equation (3.15) for given control and disturbance policies is smooth and given by (3.19). Then, according to (3.20) and (3.16), (3.17) one has for the policy and the disturbance updates:

$$u = -\frac{1}{2} R^{-1} g^T(x) \sum_{i=1}^{\infty} c_i \nabla \varphi_i(x) \tag{3.40}$$

$$d = \frac{1}{2\gamma^2} k^T(x) \sum_{i=1}^{\infty} c_i \nabla \varphi_i(x) \tag{3.41}$$

for some unknown coefficients c_i. Then one has the following result.

The following proposition is proved in [3] for constrained inputs. Non-constrained inputs are easier to prove.

Proposition 3.2. Let the least-squares solution to (3.23) be W_1 and define

$$u_1 = -\frac{1}{2} R^{-1} g^T(x) \nabla V_1(x) = -\frac{1}{2} R^{-1} g^T(x) \nabla \phi_1^T(x) W_1 \tag{3.42}$$

$$d_1 = \frac{1}{2\gamma^2} k^T(x) \nabla V_1(x) = \frac{1}{2\gamma^2} k^T(x) \nabla \phi_1^T(x) W_1 \tag{3.43}$$

with V_1 defined in (3.25). Then, as $N \to \infty$:

a. $\sup\|u_1 - u\| \to 0$

b. $\sup\|d_1 - d\| \to 0$

c. There exists a number of NN hidden layer neurons *N_0* such that u_1 and d_1 stabilize the system (3.1) for *$N>N_0$.*

■

In light of this result, the ideal feedback and disturbance policy updates are taken as (3.42), (3.43) with W_1 unknown. Therefore, define the feedback policy in the form of an action neural network which computes the control input in the structured form

$$\hat{u}(x) = -\tfrac{1}{2}R^{-1}g^T(x)\nabla\phi_1^T\hat{W}_2, \tag{3.44}$$

where $\hat{W}_2$ denotes the current estimated values of the ideal NN weights W_1. Define the actor NN estimation error as

$$\tilde{W}_2 = W_1 - \hat{W}_2 \tag{3.45}$$

Likewise, define the disturbance in the form of a disturbance neural network which computes the disturbance input in the structured form

$$\hat{d}(x) = \tfrac{1}{2\gamma^2}k^T(x)\nabla\phi_1^T\hat{W}_3, \tag{3.46}$$

where $\hat{W}_3$ denotes the current estimated values of the ideal NN weights W_1. Define the disturbance NN estimation error as

$$\tilde{W}_3 = W_1 - \hat{W}_3 \tag{3.47}$$

3.4 Online Solution of Two-Player Zero-Sum Games Using Neural Networks

This section presents our main results of Chapter 3. An online adaptive PI algorithm is given for online solution of the zero-sum game problem which involves simultaneous, or synchronous, tuning of critic, actor, and disturbance neural networks. That is, the weights of all three neural networks are tuned at the same time. This approach is a version of Generalized

Policy Iteration (GPI), as introduced in [86]. In the standard Policy Iteration algorithm (3.15)-(3.17), the critic and actor NNs are tuned sequentially, e.g. one at a time, with the weights of the other NNs being held constant. By contrast, we *tune all NN simultaneously in real-time*.

The next Facts complete the machinery required for the main results.

Facts 3.1.

a. For each feedback control and disturbance policy the nonlinear Lyapunov equation (3.15) has a smooth local solution $V(x) \geq 0$.

b. $f(.)$, is Lipschitz, and $g(.), k(.)$ are bounded by constants:

$\|f(x)\| < b_f \|x\|, \quad \|g(x)\| < b_g, \quad \|k(x)\| < b_k$

c. The NN approximation error and its gradient are bounded locally so that

$\|\varepsilon\| < b_\varepsilon$

$\|\nabla\varepsilon\| < b_{\varepsilon_x}$

d. The NN activation functions and their gradients are bounded locally so that

$\|\phi_1(x)\| < b_\phi$

$\|\nabla\phi_1(x)\| < b_{\phi_x}$

e. The critic NN weights are bounded by known constants

$\|W_1\| < W_{\max}$

■

The discussion in Section 3.2 justifies Fact 3.1a. Fact 3.1d is satisfied, e.g. by sigmoids, *tanh*, and other standard NN activation functions.

The main Theorems are now given, which provide the tuning laws for the actor, critic and disturbance neural networks that guarantee convergence of the synchronous online zero-sum game PI algorithm in real-time to the game saddle point solution, while guaranteeing closed-loop stability.

Theorem 3.2. System stability and convergence of NN weights.

Let the dynamics be given by (3.1), the critic NN be given by (3.28), the control input be given by actor NN (3.44) and the disturbance input be given by disturbance NN (3.46). Let tuning for the critic NN be provided by

$$\dot{\hat{W}}_1 = -a_1 \frac{\sigma_2}{(\sigma_2^T \sigma_2 + 1)^2} [\sigma_2{}^T \hat{W}_1 + Q(x) - \gamma^2 \left\|\hat{d}\right\|^2 + \hat{u}^T R \hat{u}] \tag{3.48}$$

where $\sigma_2 = \nabla\phi_1(f + g\hat{u} + k\hat{d})$. Let the actor NN be tuned as

$$\dot{\hat{W}}_2 = -a_2 \left\{ (F_2 \hat{W}_2 - F_1 \bar{\sigma}_2^T \hat{W}_1) - \frac{1}{4} \overline{D}_1(x) \hat{W}_2 m^T(x) \hat{W}_1 \right\} \tag{3.49}$$

and the disturbance NN be tuned as

$$\dot{\hat{W}}_3 = -a_3 \left\{ \left(F_4 \hat{W}_3 - F_3 \bar{\sigma}_2^T \hat{W}_1 \right) + \frac{1}{4\gamma^2} \overline{\mathrm{E}}_1(x) \hat{W}_3 m^T \hat{W}_1 \right\} \tag{3.50}$$

where $\overline{D}_1(x) \equiv \nabla\phi_1(x) g(x) R^{-1} g^T(x) \nabla\phi_1{}^T(x), \overline{E}_1(x) \equiv \nabla\phi_1(x) k k^T \nabla\phi_1{}^T(x)$, $m \equiv \dfrac{\sigma_2}{(\sigma_2{}^T \sigma_2 + 1)^2}$,

and $F_1 > 0, F_2 > 0, F_3 > 0, F_4 > 0$ are tuning parameters. Let Facts 1 hold and let $Q(x) > 0$. Suppose that $\bar{\sigma}_2 = \sigma_2 / (\sigma_2^T \sigma_2 + 1)$ is persistently exciting. Let the tuning parameters F_1, F_2, F_3, F_4 in (3.49), and (3.50) be selected to make the matrix D_{22} in (A.35) positive definite. Then there exists an N_0 such that, for the number of hidden layer units $N > N_0$ the closed-loop system state, the critic NN error $\tilde{W}_1$, the actor NN error $\tilde{W}_2$ and the disturbance NN error $\tilde{W}_3$ are UUB.

Proof: See appendix A.

■

Remark 3.3. See the comments following equation (3.27). Let $\varepsilon > 0$ and let N_0 be the number of hidden layer units above which $\sup\left\|\varepsilon_{HJI}\right\| < \varepsilon$. In the proof it is seen that the theorem holds for $N > N_0$.

Remark 3.4. The theorem shows that PE is needed for proper identification of the value function by the critic NN, and that nonstandard tuning algorithms are required for the actor and the disturbance NN to guarantee stability.

Remark 3.5. The assumption $Q(x) > 0$ is sufficient but not necessary for this result. If this condition is replaced by zero state observability, the proof still goes through, however it is tedious and does not add insight. The method used would be the technique used in the proof of technical Lemma 2.2 Part a in Chapter 2, or the standard methods of [36], [87].

Theorem 3.3. Exponential Convergence and Nash equilibrium.

Suppose the hypotheses of Theorem 3.2, hold. Then Theorem 3.1 holds so that exponential convergence of $\hat{W}_1$ to the approximate optimal critic value W_1 is obtained. Then:

a. $H(x,\hat{W}_1,\hat{u}_1,\hat{d}_1)$ is UUB. That is, $\hat{W}_1$ converges to the approximate HJI solution, the value of the ZS game. Where

$$\hat{u}_1 = -\frac{1}{2}R^{-1}g^T(x)\nabla\phi_1{}^T(x)\hat{W}_1 \tag{3.51}$$

$$\hat{d}_1 = \frac{1}{2\gamma^2}k^T(x)\nabla\phi_1{}^T(x)\hat{W}_1 \tag{3.52}$$

b. $\hat{u}(x),\hat{d}(x)$ (see (3.44) and (3.46)) converges to the approximate Nash equilibrium solution of the ZS game.

Proof. Consider the UUB weights $\tilde{W}_1$, $\tilde{W}_2$ and $\tilde{W}_3$ as proved in Theorem 3.2. Also $\varepsilon_{\max}$ in Theorem 3.1 is practically bounded by the bound in (A.40).

a. The approximate HJI equation is

$$H(x,\hat{W}_1) = Q(x) + \hat{W}_1{}^T\nabla\varphi_1(x)f(x) - \frac{1}{4}\hat{W}_1{}^T D_1\hat{W}_1 + \frac{1}{4\gamma^2}\hat{W}_1{}^T E_1\hat{W}_1 - \hat{\varepsilon}_{HJI} \tag{3.53}$$

After adding zero we have

$$H(x,\hat{W}_1) = \tilde{W}_1^T \nabla\varphi_1(x) f(x) - \frac{1}{4}\tilde{W}_1^T D_1 \tilde{W}_1 - \frac{1}{2}\tilde{W}_1^T D_1 \hat{W}_1 + \frac{1}{4\gamma^2}\tilde{W}_1^T E_1 \tilde{W}_1 + \frac{1}{2\gamma^2}\tilde{W}_1^T E_1 \hat{W}_1 - \varepsilon_{HJI} \quad (3.54)$$

But

$$\hat{W}_1 = -\tilde{W}_1 + W_1 \quad (3.55)$$

After taking norms in (3.55) and letting $\|W_1\| < W_{\max}$ one has

$$\|\hat{W}_1\| = \|-\tilde{W}_1 + W_1\| \le \|\tilde{W}_1\| + \|W_1\| \le \|\tilde{W}_1\| + W_{\max} \quad (3.56)$$

Now (3.54) becomes by taking into account (3.56),

$$\begin{aligned}\|H(x,\hat{W}_1)\| \le{}& \|\tilde{W}_1\| \|\nabla\varphi_1(x)\| \|f(x)\| + \frac{1}{4}\|\tilde{W}_1\|^2 \|\bar{D}_1\| + \frac{1}{2}\|\tilde{W}_1\| \|\bar{D}_1\| \left(\|\tilde{W}_1\| + W_{\max}\right) \\ &+ \frac{1}{4\gamma^2}\|\tilde{W}_1\|^2 \|\bar{E}_1\| + \frac{1}{2\gamma^2}\|\tilde{W}_1\| \|\bar{E}_1\| \left(\|\tilde{W}_1\| + W_{\max}\right) + \|\varepsilon_{HJI}\|\end{aligned} \quad (3.57)$$

Let Facts 3.1 hold and also $\sup\|\varepsilon_{HJI}\| < \varepsilon$ then (3.57) becomes

$$\begin{aligned}\|H(x,\hat{W}_1)\| \le{}& b_{\phi_x} b_f \|\tilde{W}_1\| \|x\| + \frac{1}{4}\|\tilde{W}_1\|^2 \|\bar{D}_1\| + \frac{1}{2}\|\tilde{W}_1\| \|\bar{D}_1\| \left(\|\tilde{W}_1\| + W_{\max}\right) \\ &+ \frac{1}{4\gamma^2}\|\tilde{W}_1\|^2 \|\bar{E}_1\| + \frac{1}{2\gamma^2}\|\tilde{W}_1\| \|\bar{E}_1\| \left(\|\tilde{W}_1\| + W_{\max}\right) + \varepsilon\end{aligned} \quad (3.58)$$

All the signals on the right hand side of (3.58) are UUB. So $\|H(x,\hat{W}_1)\|$ is UUB and convergence to the approximate HJI solution is obtained.

b. According to Theorem 3.1 and equations (3.42), (3.43) and (3.44), (3.46), $\|\hat{u} - u_1\|$ and $\|\hat{d} - d_1\|$ are UUB because $\|\hat{W}_2 - W_1\|$ and $\|\hat{W}_3 - W_1\|$ are UUB

So the pair $\hat{u}(x), \hat{d}(x)$ gives the Nash equilibrium solution of the zero-sum game. This completes the proof.

■

Remark 3.6. The theorems make no mention of finding the minimum nonnegative solution to the HJI. However they do guarantee convergence to a solution $(u(x), d(x))$ such that $(f(x) + g(x)u(x) + k(x)d(x))$ is stable. This is only accomplished by the minimal nonnegative HJI solution. Practical implementation, in view of the Policy Iteration Algorithm, would start with initial weights of zero in the disturbance NN (3.46). NN usage suggests starting with the initial control NN weights in (3.44) randomly selected and nonzero.

Note that the dynamics is required to be known to implement this algorithm in that $\sigma_2 = \nabla\phi_1(f + g\hat{u} + k\hat{d})$, $\bar{D}_1(x)$, $\bar{E}_1(x)$ and (3.44), (3.46) depend on $f(x), g(x), k(x)$.

3.5 Simulations

Here we present simulations of a linear and a nonlinear system to show that the game can be solved ONLINE by learning in real time, using the method of this chapter.

3.5.1. Linear System

Consider the continuous-time F16 aircraft plant with quadratic cost function used in [85]. The system state vector is $x = [\alpha \quad q \quad \delta_e]$, where α denotes the angle of attack, q is the pitch rate and δ_e is the elevator deflection angle. The control input is the elevator actuator voltage and the disturbance is wind gusts on angle of attack. One has the dynamics $\dot{x} = Ax + Bu + Kd$,

$$\dot{x} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} d$$

where Q and R in the cost function are identity matrices of appropriate dimensions and $\gamma = 5$. Also $a_1 = a_2 = a_3 = 1$, $F_1 = I, F_2 = 10I, F_3 = I, F_4 = 10I$ where I is an identity matrix of appropriate dimensions. In this linear case the solution of the HJI equation is given by the solution of the game algebraic Riccati equation (GARE)

$$A^T P + PA + Q - PBR^{-1}B^T P + \frac{1}{\gamma^2} PKK^T P = 0$$

Since the value is quadratic in the LQR case, the critic NN basis set $\phi_1(x)$ was selected as the quadratic vector in the state components $x \otimes x$ with $\otimes$ the Kronecker product. Redundant terms were removed to leave $n(n+1)/2 = 6$ components. Solving the GARE gives the parameters of the optimal critic as

$$W_1^* = [1.6573 \quad 1.3954 \quad -0.1661 \quad 1.6573 \quad -0.1804 \quad 0.4371]^T .$$

which are the components of the Riccati solution matrix *P*.

The synchronous zero-sum game PI algorithm is implemented as in Theorem 2. PE was ensured by adding a small probing noise to the control and the disturbance input. Figure 9 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \quad W_{c4} \quad W_{c5} \quad W_{c6}]^T$$

converging to the optimal values. In fact after 300s the critic parameters converged to

$$\hat{W}_1(t_f) = [1.7090 \quad 1.3303 \quad -0.1629 \quad 1.7354 \quad -0.1730 \quad 0.4468]^T.$$

The actor parameters after 300s converge to the values of

$$\hat{W}_2(t_f) = [1.7090 \quad 1.3303 \quad -0.1629 \quad 1.7354 \quad -0.1730 \quad 0.4468]^T.$$

The disturbance parameters after 300s converge to the values of

$$\hat{W}_3(t_f) = [1.7090 \quad 1.3303 \quad -0.1629 \quad 1.7354 \quad -0.1730 \quad 0.4468]^T.$$

Then, the actor NN is given as

$$\hat{u}_2(x) = -\tfrac{1}{2} R^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & 0 & x_1 \\ 0 & 2x_2 & 0 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}^T \hat{W}_2(t_f) .$$

Then, the disturbance NN is given as

$$\hat{d}(x) = \tfrac{1}{2\gamma^2} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & 0 & x_1 \\ 0 & 2x_2 & 0 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}^T \hat{W}_3(t_f)$$

The evolution of the system states is presented in Figure 10. One can see that after 300s convergence of the NN weights in critic, actor and disturbance has occurred.

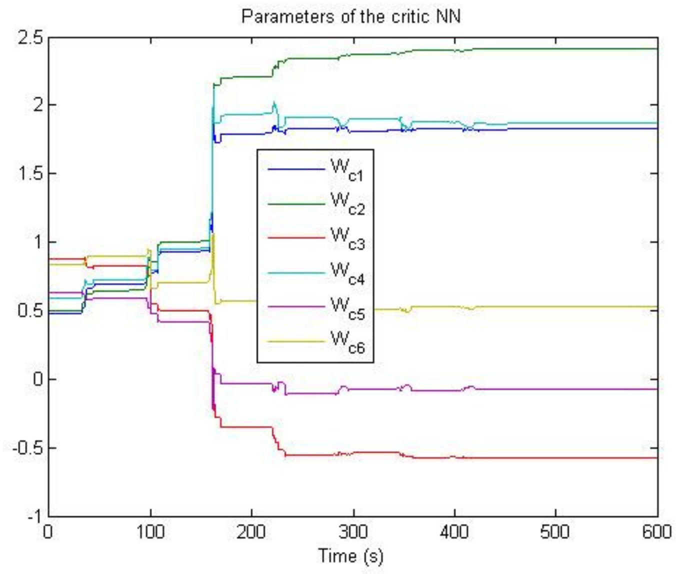

Figure 9. Convergence of the critic parameters to the parameters of the optimal critic.

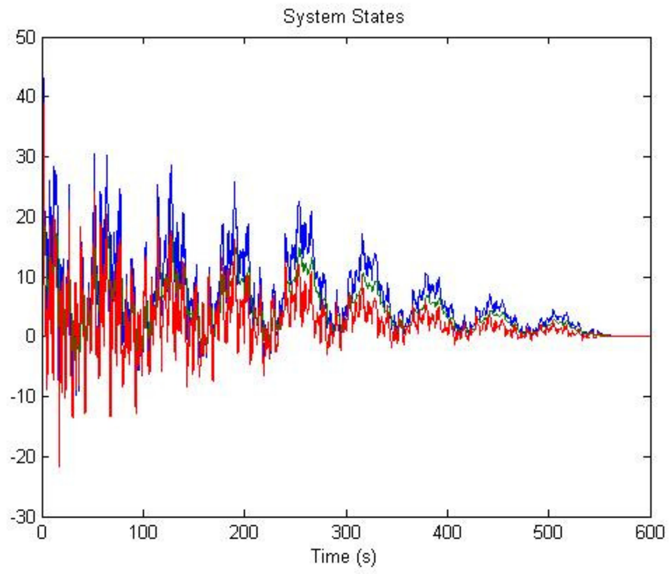

Figure 10. Evolution of the system states for the duration of the experiment.

3.5.2. Single Player Linear System

The purpose of this example is to show that one learns FASTER if one has an opponent. That is, the online two-player game converges faster than an equivalent online 1-

player (optimal control) problem. In this example, we use the method for online solution of the optimal control problem presented in Chapter 2.

Consider the continuous-time F16 aircraft plant described before but with $d = 0$. Solving the ARE with Q and R identity matrices of appropriate dimensions, gives the parameters of the optimal critic as

$$W_1^* = [1.4245 \quad 1.1682 \quad -0.1352 \quad 1.4361 \quad -0.1516 \quad 0.4329]^T .$$

Figure 11 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \quad W_{c4} \quad W_{c5} \quad W_{c6}]^T$$

converging to the optimal values. In fact after 800s the critic parameters converged to

$$\hat{W}_1(t_f) = [1.4270 \quad 1.1654 \quad -0.1367 \quad 1.4387 \quad -0.1496 \quad 0.4323]^T .$$

The actor parameters after 800s converge to the values of

$$\hat{W}_2(t_f) = [1.4270 \quad 1.1654 \quad -0.1367 \quad 1.4387 \quad -0.1496 \quad 0.4323]^T .$$

The actor NN is given as

$$\hat{u}_2(x) = -\tfrac{1}{2} R^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & 0 & x_1 \\ 0 & 2x_2 & 0 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}^T \begin{bmatrix} 1.4270 \\ 1.1654 \\ -0.1367 \\ 1.4387 \\ -0.1496 \\ 0.4323 \end{bmatrix} .$$

The evolution of the system states is presented in Figure 12. One can see that after 800s convergence of the NN weights in critic and actor has occurred. This shows that the probing noise effectively guaranteed the PE condition.

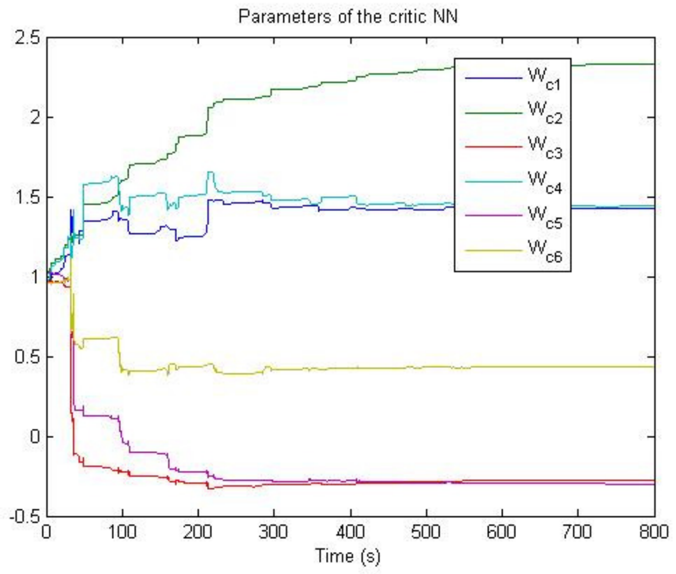

Figure 11. Convergence of the critic parameters to the parameters of the optimal critic.

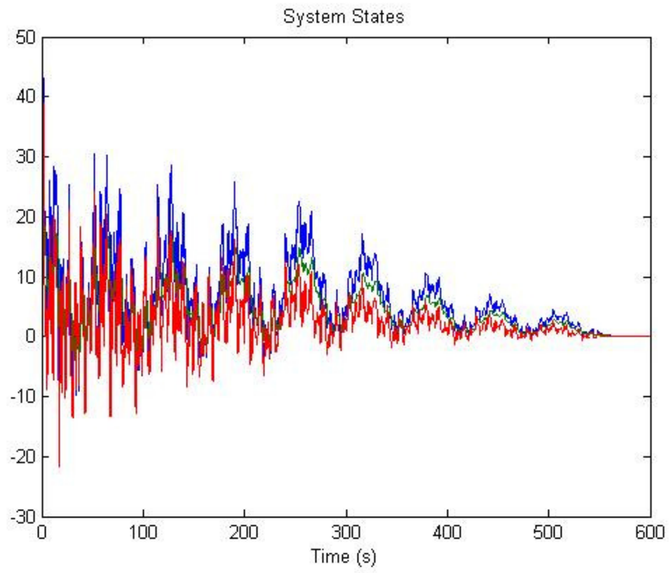

Figure 12. Evolution of the system states.

In comparison with part 3.5.1, it is very clear that the two-player zero-sum game algorithm has faster convergence skills than the single-player. As a conclusion the critic NN learns faster when there is an oponent for the control input, named disturbance.

3.5.3. Nonlinear System

Consider the following affine in control input nonlinear system, with a quadratic cost constructed as in [64]

$$\dot{x} = f(x) + g(x)u + k(x)d, x \in \mathbb{R}^2$$

where

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -x_1^3 - x_2^3 + 0.25x_2(\cos(2x_1)+2)^2 - 0.25x_2\frac{1}{\gamma^2}(\sin(4x_1)+2)^2 \end{bmatrix}$$
$$g(x) = \begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}, \quad k(x) = \begin{bmatrix} 0 \\ (\sin(4x_1)+2) \end{bmatrix}.$$

One selects $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = 1, \gamma = 8.$

Also $a_1 = a_2 = a_3 = 1$, $F_1 = I, F_2 = 10I, F_3 = I, F_4 = 10I$ where I is an identity matrix of appropriate dimensions.

The optimal value function is

$$V^*(x) = \frac{1}{4}x_1^4 + \frac{1}{2}x_2^2$$

the optimal control signal is

$$u^*(x) = -\tfrac{1}{2}(\cos(2x_1)+2)x_2$$

and

$$d^*(x) = \frac{1}{2\gamma^2}(\sin(4x_1)+2)x_2$$

One selects the critic NN vector activation function as

$$\varphi_1(x) = [x_1^2 \quad x_2^2 \quad x_1^4 \quad x_2^4]$$

Figure 13 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \quad W_{c4}]^T$$

by using the synchronous zero-sum game algorithm. After convergence at about 50s have

$$\hat{W}_1(t_f) = [\text{-}0.0006 \quad 0.4981 \quad 0.2532 \quad 0.0000]^T$$

The actor parameters after 80s converge to the values of

$$\hat{W}_2(t_f) = [\text{-}0.0006 \quad 0.4981 \quad 0.2532 \quad 0.0000]^T,$$

and the disturbance parameters after 50s converge to the values of

$$\hat{W}_3(t_f) = [\text{-}0.0006 \quad 0.4981 \quad 0.2532 \quad 0.0000]^T.$$

So that the actor NN

$$\hat{u}_2(x) = -\tfrac{1}{2} R^{-1} \begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \\ 4x_1^3 & 0 \\ 0 & 4x_2^3 \end{bmatrix}^T \hat{W}_2(t_f)$$

also converged to the optimal control, and the disturbance NN

$$\hat{d}(x) = \tfrac{1}{2\gamma^2} \begin{bmatrix} 0 \\ \sin(4x_1)+2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \\ 4x_1^3 & 0 \\ 0 & 4x_2^3 \end{bmatrix}^T \hat{W}_3(t_f)$$

also converged to the optimal disturbance.

The evolution of the system states is presented in Figure 14. Figure 15 shows the optimal value function. The identified value function given by $\hat{V}_1(x) = \hat{W}_1^T \phi_1(x)$ is virtually indistinguishable from the exact solution and so is not plotted. In fact, Figure 16 shows the 3-D plot of the difference between the approximated value function and the optimal one. This error is close to zero. Good approximation of the actual value function is being evolved. Figure 17 shows the 3-D plot of the difference between the approximated control, by using the online algorithm, and the optimal one. This error is close to zero.

Finally Figure 18 shows the 3-D plot of the difference between the approximated disturbance, by using the online algorithm, and the optimal one. This error is close to zero.

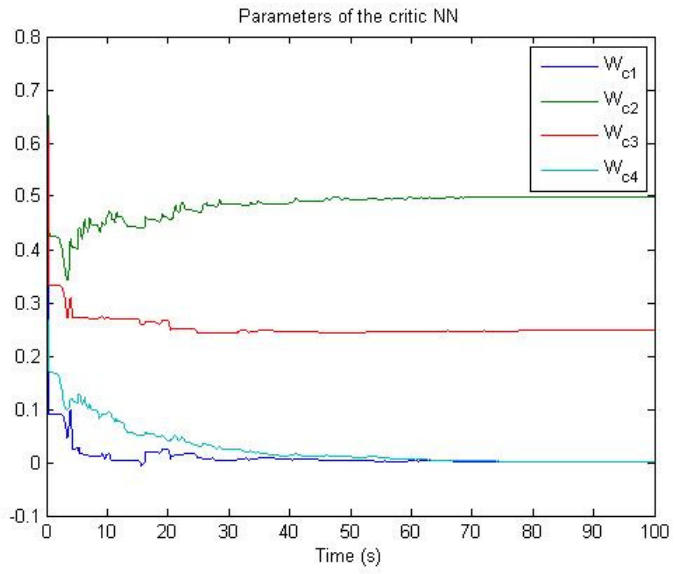

Figure 13. Convergence of the critic parameters.

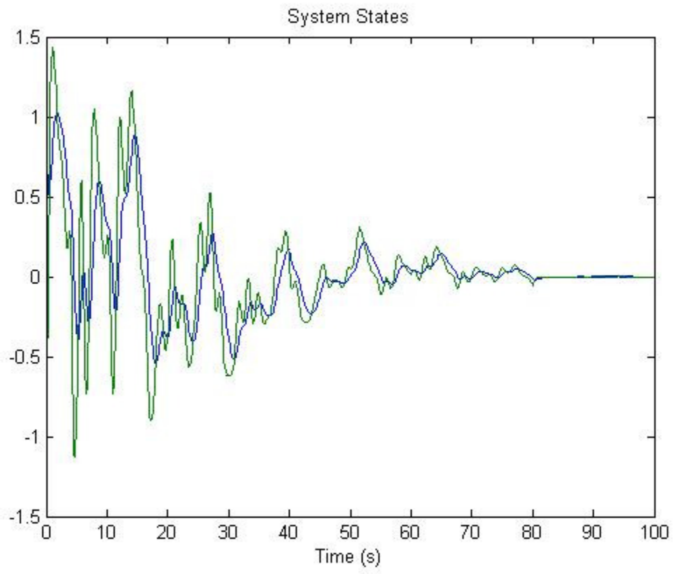

Figure 14. Evolution of the system states.

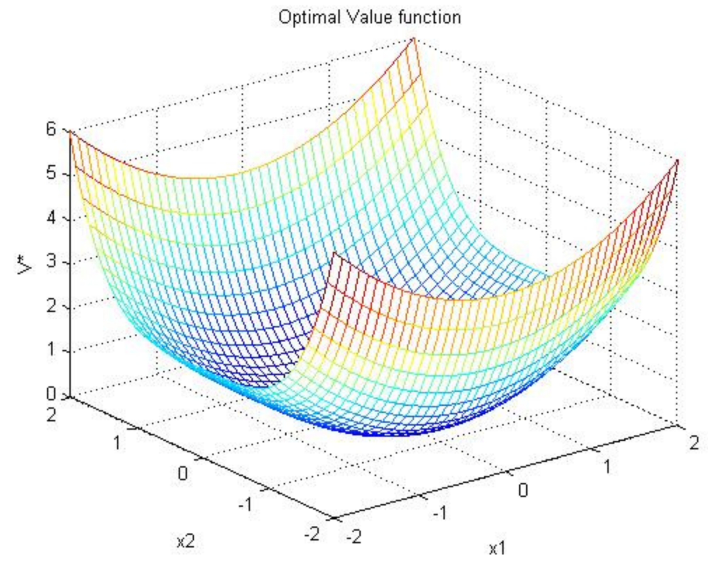

Figure 15. Optimal Value function.

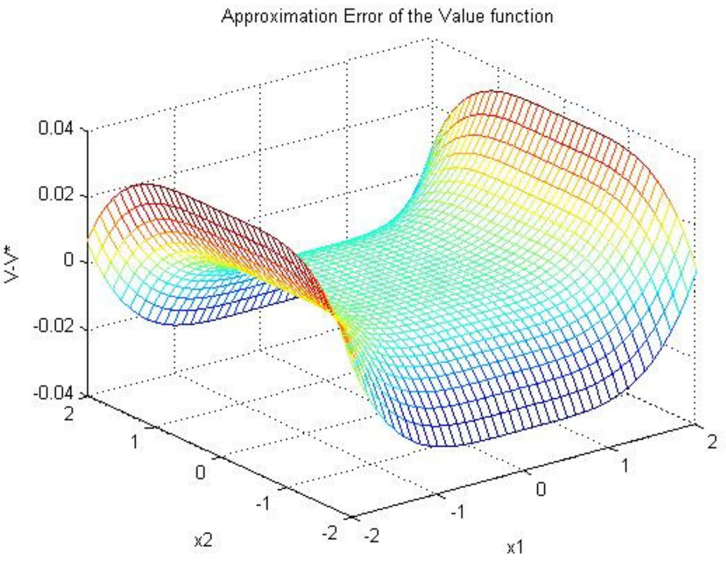

Figure 16. 3D plot of the approximation error for the value function.

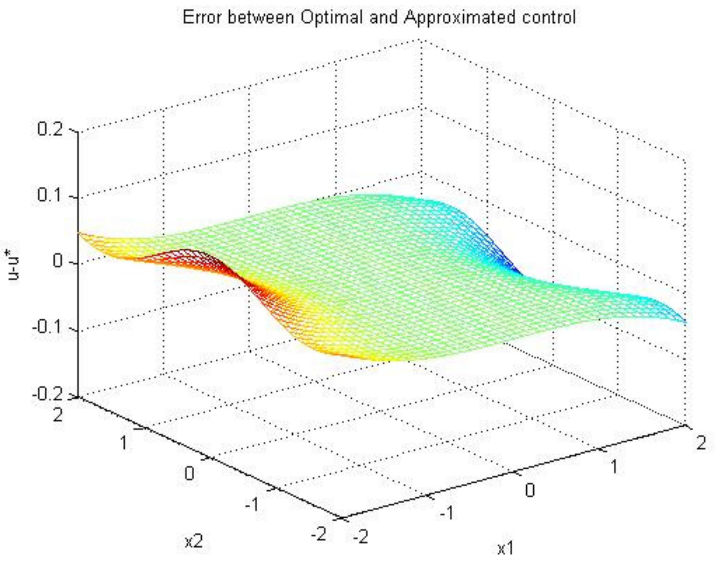

Figure 17. 3D plot of the approximation error for the control.

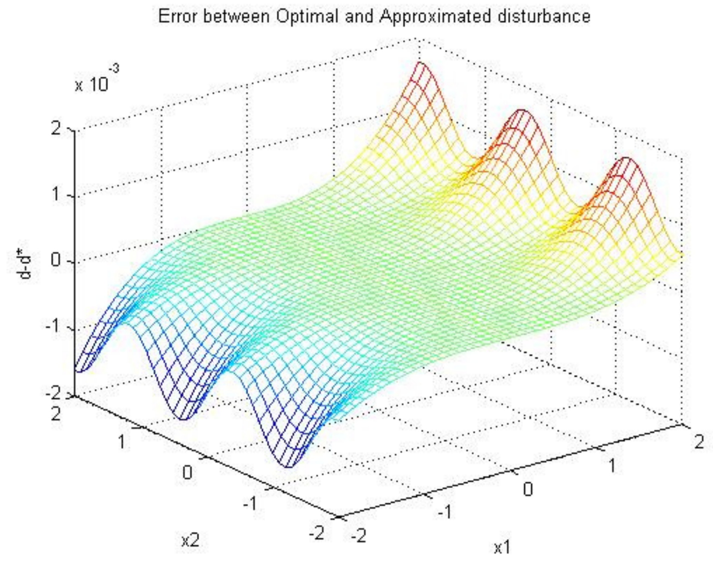

Figure 18. 3D plot of the approximation error for the disturbance.

3.6 Conclusion

In this chapter we have proposed a new optimal adaptive algorithm which solves online the continuous-time zero-sum game problem for affine in the inputs nonlinear systems. The importance of this algorithm relies the simultaneous tuning of the actor, critic and disturbance neural networks and the convergence to HJI without solving this equation. Proofs using Lyapunov techniques guarantee convergence and closed-loop stability. Simulation examples show the effectiveness of the proposed algorithm.

CHAPTER 4

MULTI-PLAYER GAMES: ONLINE ADAPTIVE LEARNING SOLUTION OF COUPLED HAMILTON-JACOBI AND COUPLED RICCATI EQUATIONS

4.1 Introduction

For the most part, interest in the control systems community has been in the (non-cooperative) zero-sum games, which provide the solution of the H-infinity robust control problem [11], [60]. However, dynamic team games may have some cooperative objectives and some selfish objectives among the players. This cooperative/non-cooperative balance is captured in the NZS games, as detailed herein.

In this chapter we are interested in feedback policies with full state information, and provide methods for *online gaming*, that is for solution of *N*-player infinite horizon NZS games *online*, through learning the Nash-equilibrium in real-time. The dynamics are nonlinear in continuous-time and are assumed known. A novel adaptive control technique is given that is based on reinforcement learning techniques, whereby each player's control policies are tuned online using data generated in real time along the system trajectories. Also tuned by each player are 'critic' approximator structures whose function is to identify the values of the current control policies for each player. Based on these value estimates, the players' policies are continuously updated. This is a sort of indirect adaptive control algorithm, yet, due to the simple form dependence of the control policies on the learned value, it is affected online as direct ('optimal') adaptive control.

This chapter proposes an algorithm for nonlinear continuous-time systems with known dynamics to solve the *N*-player non-zero sum (NZS) game problem where each player wants to optimize his *own performance index* [11]. The number of parametric approximator structures that are used is $2N$. Each player maintains a critic approximator neural network (NN) to learn his optimal value and a control actor NN to learn his optimal control policy. Parameter update laws are given to tune the *N*-critic and *N*-actor neural networks simultaneously online to converge to the solution to the coupled HJ equations, while also guaranteeing closed-loop stability. Rigorous proofs of performance and convergence are given. For the sake of clarity, we restrict ourselves to two player differential games in the actual proof. The proof technique can be directly extended using further careful bookkeeping to multiple players.

The chapter is organized as follows. It is necessary to develop policy iteration (PI) techniques for solving multi-player games, for these PI algorithms give the controller structure needed for the online adaptive learning techniques presented in this work. Therefore, Section 4.2 presents the formulation of multi-player NZS differential games for nonlinear systems [11] and presents a policy iteration algorithm to solve the required coupled nonlinear HJ equations by successive solutions of nonlinear Lyapunov-like equations. Based on this structure, Section 4.4 develops an adaptive control method for on-line learning of the solution to the two-player NZS game problem. The method generalizes to the multi-player games, and particularizes to the special case of zero-sum (ZS) games. Based on the PI algorithm in Section 4.3, suitable approximator structures (based on neural networks) are developed for the value function and the control inputs of the two players. A rigorous mathematical analysis is carried out. It is found that the actor and critic neural networks require novel nonstandard tuning algorithms to guarantee stability and convergence to the Nash equilibrium. A persistence of excitation condition [36], [87] is needed to guarantee proper convergence to the optimal value functions. A Lyapunov analysis technique is used. Section 4.5 presents simulation examples that show the effectiveness of the synchronous online game algorithm in learning the optimal values for both linear and nonlinear systems.

4.2 *N*-Player differential game for nonlinear systems

In this section is presented the formulation of *N*-player games for nonlinear systems. A Policy Iteration solution algorithm is given. The objective is to lay a foundation for the control structure needed in Section 4.4 for online solution of the game problems in real-time.

4.2.1 Multi-Player Non Zero Sum Games

Consider the *N*-player nonlinear time-invariant differential game on an infinite time horizon

$$\dot{x} = f(x) + \sum_{j=1}^{N} g_j(x) u_j \tag{4.1}$$

where state $x(t) \in \mathbb{R}^n$, players or controls $u_j(t) \in \mathbb{R}^{m_j}$. Assume that $f(0) = 0$ and $f(x)$, $g_j(x)$ are locally Lipschitz.

The cost functionals associated with each player are

$$J_i(x(0), u_1, u_2, \ldots u_N) = \int_0^\infty (Q_i(x) + \sum_{j=1}^{N} u_j^T R_{ij} u_j)\, dt \equiv \int_0^\infty r_i(x(t), u_1, u_2, \ldots u_N)\, dt; \quad i \in N \tag{4.2}$$

where function $Q_i(x) \geq 0$ is generally nonlinear, and $R_{ii} > 0, R_{ij} \geq 0$ are symmetric matrices.

We seek optimal controls among the set of feedback control policies with complete state information.

Definition 4.1. (Admissible policies) Feedback control policies $u_i = \mu_i(x)$ are defined as admissible with respect to (4.2) on a set $\Omega \subset \mathbb{R}^n$, denoted by $\mu_i \in \Psi(\Omega)$ if $\mu_i(x)$ is continuous on Ω, $\mu_i(0) = 0$, $\mu_i(x)$ stabilizes (4.1) on Ω, and (4.2) is finite $\forall x_0 \in \Omega$.

Given admissible feedback policies/strategies $u_i(t) = \mu_i(x)$, the value is

$$V_i(x(0), \mu_1, \mu_2, \ldots \mu_N) = \int_t^\infty (Q_i(x) + \sum_{j=1}^{N} \mu_j^T R_{ij} \mu_j)\, d\tau \equiv \int_t^\infty r_i(x(t), \mu_1, \mu_2, \ldots \mu_N)\, d\tau; \quad i \in N \tag{4.3}$$

Define the *N*-player game

$$V_i^*(x(t),\mu_1,\mu_2,\ldots\mu_N)=\min_{\mu_i}\int_t^\infty (Q_i(x)+\sum_{j=1}^N \mu_j^T R_{ij}\mu_j)\,d\tau;\quad i\in N \tag{4.4}$$

By assuming that all the players have the same hierarchical level, we focus on the so-called Nash equilibrium that is given by the following definition.

Definition 4.2. [11] **(**Nash equilibrium strategies) An *N-tuple* of strategies $\left\{\mu_1^*,\mu_2^*,\ldots,\mu_N^*\right\}$ with $\mu_i^*\in\Omega_i, i\in N$ is said to constitute a Nash equilibrium solution for an *N-player* finite game in extensive form, if the following *N* inequalities are satisfied for all $\mu_i^*\in\Omega_i, i\in N$:

$$J_i^*\triangleq J_i\,(\mu_1^*,\mu_2^*,\mu_i^*,\ldots,\mu_N^*)\le J_i\,(\mu_1^*,\mu_2^*,\mu_i\,,\ldots,\mu_N^*), i\in N \tag{4.5}$$

The *N-tuple* of quantities $\left\{J_1^*,J_2^*,\ldots,J_N^*\right\}$ is known as a Nash equilibrium outcome of the *N*-player game.

Differential equivalents to each value function are given by the following nonlinear Lyapunov equations (different form from those given in [11])

$$0=r(x,u_1,\ldots,u_N)+(\nabla V_i\;)^T(f(x)+\sum_{j=1}^N g_j(x)u_j),\quad V_i(0)=0,\; i\in N \tag{4.6}$$

where $\nabla V_i=\partial V_i/\partial x\in\mathbb{R}^{n_i}$ is the gradient vector (e.g. transposed gradient). Then, suitable nonnegative definite solutions to (4.6) are the values evaluated using the infinite integral (4.4) along the system trajectories. Define the Hamiltonian functions

$$H_i(x,\nabla V_i\;,u_1,\ldots,u_N)=r(x,u_1,\ldots,u_N)+(\nabla V_i\;)^T(f(x)+\sum_{j=1}^N g_j(x)u_j),\qquad i\in N \tag{4.7}$$

According to the stationarity conditions, associated feedback control policies are given by

$$\frac{\partial H_i}{\partial u_i}=0\Rightarrow \mu_i(x)=-\tfrac{1}{2}R_{ii}^{-1}g_i^T(x)\nabla V_i,\qquad i\in N \tag{4.8}$$

Substituting (4.8) into (4.6) one obtains the *N* coupled Hamilton-Jacobi (HJ) equations

$$0=(\nabla V_i)^T\left(f(x)-\tfrac{1}{2}\sum_{j=1}^{N}g_j(x)R_{jj}^{-1}g_j^T(x)\nabla V_j\right)+Q_i(x)+\tfrac{1}{4}\sum_{j=1}^{N}\nabla V_j^{\,T}g_j(x)R_{jj}^{-T}R_{ij}R_{jj}^{-1}g_j^T(x)\nabla V_j,\, V_i(0)=0 \quad (4.9)$$

These coupled HJ equations are in 'closed-loop' form. The equivalent 'open-loop' form is

$$0=\nabla V_i^{\,T}f(x)+Q_i(x)-\tfrac{1}{2}\nabla V_i^{\,T}\sum_{j=1}^{N}g_j(x)R_{jj}^{-1}g_j^T(x)\nabla V_j+\tfrac{1}{4}\sum_{j=1}^{N}\nabla V_j^{\,T}g_j(x)R_{jj}^{-T}R_{ij}R_{jj}^{-1}g_j^T(x)\nabla V_j,\, V_i(0)=0 \quad (4.10)$$

In linear systems of the form $\dot{x}=Ax+\sum_{j=1}^{N}B_ju_j$, (4.9) becomes the *N* coupled generalized algebraic Riccati equations

$$0=P_iA_c+A_c^{\,T}P_i+Q_i+\tfrac{1}{4}\sum_{j=1}^{N}P_jB_jR_{jj}^{-T}R_{ij}R_{jj}^{-1}B_j^TP_j,\quad i\in N \quad (4.11)$$

where $A_c=A-\tfrac{1}{2}\sum_{i=1}^{N}B_iR_{ii}^{-1}B_i^TP_i$. It is shown in [11] that if there exist solutions to (4.11) and further satisfying the conditions that for each $i\in N$ the pair $\left(A-\tfrac{1}{2}\sum_{\substack{j\in N\\ j\neq i}}B_jR_{jj}^{-1}B_j^TP_j,\quad B_i\right)$ is stabilizable and the pair $\left(A-\tfrac{1}{2}\sum_{\substack{j\in N\\ j\neq i}}B_jR_{jj}^{-1}B_j^TP_j,\quad Q_i+\tfrac{1}{4}\sum_{\substack{j\in N\\ j\neq i}}P_jB_jR_{jj}^{-T}R_{ij}R_{jj}^{-1}B_j^TP_j\right)$ is detectable, then the *N*-tuple of the stationary feedback policies $\mu_i^*(x)=-K_ix=-\tfrac{1}{2}R_{ii}^{-1}B_i^TP_ix,\quad i\in N$ provides a Nash equilibrium solution for the linear quadratic *N*-player differential game among feedback policies with full state information. Furthermore the resulting system dynamics, described by $\dot{x}=A_cx,\quad x(0)=x_0$ are asymptotically stable.

4.2.2 Cooperation vs. Non-Cooperation in Multi-Player Games

One may write the costs as

$$J_i = \frac{1}{N}\sum_{j=1}^{N} J_j + \frac{1}{N}\sum_{j=1}^{N}(J_i - J_j) \equiv \bar{J} + \tilde{J}_i, \quad i \in N \tag{4.12}$$

where $\bar{J}$ is an overall cooperative 'team' cost and $\tilde{J}_i$ a 'conflict' cost for player *i*. If $\bar{J} = 0$ one has a zero-sum game. The most studied case is the 2-player zero-sum game when $J_1 = -J_2$. Such games are known as convex-concave games, result in saddle-point Nash equilibria, and have been extensively studied in control systems due to their relation to the H_∞ control problem [11], [1], [60]. However, general dynamic team games may have some cooperative objectives and some selfish objectives among the players. This interplay between cooperative and non-cooperative objectives is captured in NZS games, as detailed in (4.12). Therefore, this work is interested in general *N*-player games which may or may not be zero-sum.

Though NZS games may contain non-cooperative components, note that the solution to each player's coupled HJI equations (4.9) requires knowledge of all the other players' strategies (4.8). This is in the spirit of rational opponents [11] whereby the players share information, yet from the definition of Nash equilibrium each decides to remain at the equilibrium policy.

4.2.3 Policy Iteration Solution for Non-Zero Sum Games

Equations (4.9) are difficult or impossible to solve. An iterative offline solution technique is given by the following policy iteration algorithm. It solves the coupled HJ equations by iterative solution of uncoupled nonlinear Lyapunov equations.

Algorithm 1. Policy Iteration for N-player games

a) Start with stabilizing initial policies $\mu^0{}_1(x),\ldots,\mu^0{}_N(x)$.

b) Given the *N*-tuple of policies $\mu^k{}_1(x),\ldots,\mu^k{}_N(x)$, solve for the *N*-tuple of costs $V^k{}_1(x(t)),V^k{}_2(x(t))\ldots V^k{}_N(x(t))$ using

$$0 = r(x, \mu^k{}_1, \dots, \mu^k{}_N) + (\nabla V_i^k)^T \left(f(x) + \sum_{j=1}^{N} g_j(x)\mu_j^i \right), \quad V^k{}_i(0) = 0 \qquad i \in N \quad (4.13)$$

c) Update the *N*-tuple of control policies using:

$$\mu_i^{k+1} = \underset{u_i \in \Psi(\Omega)}{\arg\min}[H_i(x, \nabla V_i \ , u_1, \dots, u_N)] \qquad i \in N \qquad (4.14)$$

which explicitly is

$$\mu_i^{k+1}(x) = -\tfrac{1}{2} R_{ii}^{-1} g_i^T(x) \nabla V_i^k \qquad i \in N \qquad (4.15)$$

■

A linear 2-player version of Algorithm 1 is given in [28] and can be considered as an extension of Kleiman's Algorithm [44] to 2-player games.

In the next section we use PI Algorithm 1 to motivate the control structure for an online adaptive *N*-player game solution algorithm. Then it is proven that 'optimal adaptive' control algorithm converges online to the solution of coupled HJs (4.10), while guaranteeing closed-loop stability.

4.3 Value function approximation for solution of nonlinear Lyapunov equations

This work uses nonlinear approximator structures for Value Function Approximation (VFA) [15], [98], [99] to solve (4.13). We show how to solve the 2-player non-zero sum game presented in Section 2, the approach can easily be extended to more than 2 players.

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

$$\dot{x} = f(x) + g(x)u(x) + k(x)d(x) \qquad (4.16)$$

where state $x(t) \in \mathbb{R}^n$, first control input $u(x) \in \mathbb{R}^m$, and second control input $d(x) \in \mathbb{R}^q$. Assume that $f(0) = 0$ and $f(x)$, $g(x), k(x)$ are locally Lipschitz, $\|f(x)\| < b_f \|x\|$.

Assumption 1. For admissible feedback control policies the nonlinear Lyapunov equations (4.13) have locally smooth solutions $V_1'(x) \geq 0, \ V_2'(x) \geq 0, \quad \forall x \in \Omega$.

According to the Weierstrass higher-order approximation Theorem [4], [26], [34], there exist complete independent basis sets such that the solution $V_1'(x(t)), V_2'(x(t))$ to (4.6) and their gradients are uniformly approximated. Specifically, the basis sets are dense in the Sobolev norm $W^{1,\infty}$ [6].

Therefore, assume there exist constant neural network (NN) weights W_1 and W_2 such that the value functions $V_1'(x)$ and $V_2'(x)$ are approximated on a compact set Ω as

$$V_1'(x) = W_1^T \phi_1(x) + \varepsilon_1(x) \tag{4.17}$$

$$V_2'(x) = W_2^T \phi_2(x) + \varepsilon_2(x) \tag{4.18}$$

with $\phi_1(x): \mathbb{R}^n \to \mathbb{R}^K$ and $\phi_2(x): \mathbb{R}^n \to \mathbb{R}^K$ the NN activation function basis set vectors, *K* the number of neurons in the hidden layer, and $\varepsilon_1(x)$ and $\varepsilon_2(x)$ the NN approximation errors. From the approximation literature, the basis functions can be selected as sigmoids, tanh, polynomials, etc.

The value function derivatives are also uniformly approximated, e.g., additionally, $\forall x \in \Omega$,

$$\frac{\partial V_i'}{\partial x} = \left(\frac{\partial \phi_i(x)}{\partial x}\right)^T W_i + \frac{\partial \varepsilon_i}{\partial x} = \nabla \phi_i^T W_i + \nabla \varepsilon_i, \quad i = 1,2 \tag{4.19}$$

Then, as the number of hidden-layer neurons $K \to \infty$, the approximation errors $\varepsilon_i \to 0,\ \nabla \varepsilon_i \to 0,\ i = 1,2$ uniformly [4], [26]. In addition, for fixed *K*, the NN approximation errors $\varepsilon_1(x), \varepsilon_2(x)$ and $\nabla \varepsilon_1, \nabla \varepsilon_2$ are bounded on a set Ω by constants if Ω is compact [34]. We refer to the NN with weights W_1 and W_2 that perform VFA as the *critic* NNs for each player.

Using the NN VFA (4.17), (4.18) considering fixed feedback policies *u* and *d* the Hamiltonians (4.7) become:

$$H_1(x, W_1, u, d) = Q_1(x) + u^T R_{11} u + d^T R_{12} d + W_1^T \nabla \phi_1 (f(x) + g(x)u + k(x)d) = \varepsilon_{H_1} \tag{4.20}$$

$$H_2(x,W_2,u,d)=Q_2(x)+u^TR_{21}u+d^TR_{22}d+W_2^T\nabla\phi_2(f(x)+g(x)u+k(x)d)=\varepsilon_{H_2} \tag{4.21}$$

The residual errors for the two players are

$$\varepsilon_{H_i}\equiv-(\nabla\varepsilon_i)^{\mathrm{T}}(f+gu+kd),\quad i=1,2 \tag{4.22}$$

Substituting W_1, W_2 into the HJ equations (4.10), one obtains

$$\begin{aligned}&Q_1(x)-\tfrac{1}{4}W_1{}^T\nabla\phi_1(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1{}^T(x)W_1+W_1{}^T\nabla\phi_1(x)f(x)\\&+\tfrac{1}{4}W_2{}^T\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2{}^T(x)W_2\\&-\tfrac{1}{2}W_1{}^T\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2{}^T(x)W_2=\varepsilon_{HJ_1},\quad V_1(0)=0\end{aligned} \tag{4.23}$$

$$\begin{aligned}&Q_2(x)-\tfrac{1}{4}W_2{}^T\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2{}^T(x)W_2+W_2{}^T\nabla\phi_2(x)f(x)\\&+\tfrac{1}{4}W_1{}^T\nabla\phi_1(x)g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T(x)W_1\\&-\tfrac{1}{2}W_2{}^T\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1{}^T(x)W_1=\varepsilon_{HJ_2},\quad V_2(0)=0\end{aligned} \tag{4.24}$$

where the residual errors due to function approximation error for the first player is

$$\begin{aligned}\varepsilon_{HJ_1}\equiv{}&\tfrac{1}{2}W_1\nabla\phi_1{}^Tg(x)R_{11}^{-1}g^T(x)\nabla\varepsilon_1+\tfrac{1}{4}\nabla\varepsilon_1{}^Tg(x)R_{11}^{-1}g^T(x)\nabla\varepsilon_1-\tfrac{1}{4}\nabla\varepsilon_2{}^Tk(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varepsilon_2\\&-\tfrac{1}{4}W_2\nabla\phi_2{}^Tk(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varepsilon_2-\nabla\varepsilon_1^Tf(x)+\tfrac{1}{2}W_1\nabla\phi_1{}^Tk(x)R_{22}^{-1}k^T(x)\nabla\varepsilon_2\\&+\tfrac{1}{2}\nabla\varepsilon_1{}^Tk(x)R_{22}^{-1}k^T(x)\nabla\varepsilon_2+\tfrac{1}{2}W_2\nabla\phi_2{}^Tk(x)R_{22}^{-1}k^T(x)\nabla\varepsilon_1\end{aligned}$$

and ε_{HJ_2} for the second player is similarly defined.

The following proposition follows as in [4], [5].

Proposition 4.2. For any admissible feedback policies $u(x(t)), d(x(t))$ the least-squares pair solution to (4.20) and (4.21) exists and is unique for each *K*. Denote these solutions as W_1 and W_2 and define

$$V_1(x)=W_1{}^T\phi_1(x) \tag{4.25}$$

$$V_2(x)=W_2{}^T\phi_2(x) \tag{4.26}$$

Then, as $K\to\infty$:

a) $\sup_{x\in\Omega}\left|\varepsilon_{H_i}\right|\to 0,\quad i=1,2$

b) $\sup_{x\in\Omega}\left|\varepsilon_{HJ_i}\right| \to 0, \quad i=1,2$

c) $\sup_{x\in\Omega}\left|V_i - V_i'\right| \to 0, \quad i=1,2$

d) $\sup_{x\in\Omega}\left\|\nabla V_i - \nabla V_i'\right\| \to 0, \quad i=1,2$

■

This result shows that $V_1(x), V_2(x)$ converge uniformly in Sobolev norm $W^{1,\infty}$ [6] to the exact solution $V_1'(x), V_2'(x)$ to (4.6). Therefore, $\forall \overline{\varepsilon}_1 > 0\ \exists K(\overline{\varepsilon}_1) : \sup_{x\in\Omega}\left\|\varepsilon_{HJ_1}\right\| < \overline{\varepsilon}_1, K > K(\overline{\varepsilon}_1)$ and $\forall \overline{\varepsilon}_2 > 0\ \exists K(\overline{\varepsilon}_2) : \sup_{x\in\Omega}\left\|\varepsilon_{HJ_2}\right\| < \overline{\varepsilon}_2, K > K(\overline{\varepsilon}_2)$.

Assuming current NN weight estimates $\hat{W}_1$ and $\hat{W}_2$, the outputs of the two critic NN are given by

$$\hat{V}_1(x) = \hat{W}_1^T \phi_1(x) \tag{4.27}$$

$$\hat{V}_2(x) = \hat{W}_2^T \phi_2(x) \tag{4.28}$$

The approximate Lyapunov-like equations are then

$$H_1(x, \hat{W}_1, u_1, d_2) = Q_1(x) + u_1^T R_{11} u_1 + d_2^T R_{12} d_2 + \hat{W}_1^T \nabla\phi_1 (f(x) + g(x)u_1 + k(x)d_2) = e_1 \tag{4.29}$$

$$H_2(x, \hat{W}_2, u_1, d_2) = Q_2(x) + u_1^T R_{21} u_1 + d_2^T R_{22} d_2 + \hat{W}_2^T \nabla\phi_2 (f(x) + g(x)u_1 + k(x)d_2) = e_2 \tag{4.30}$$

It is desired to select $\hat{W}_1$ and $\hat{W}_2$ to minimize the square residual error

$$E_1 = \tfrac{1}{2} e_1^T e_1 + \tfrac{1}{2} e_2^T e_2$$

Then $\hat{W}_1(t) \to W_1, \hat{W}_2(t) \to W_2$ and $e_1 \to \varepsilon_{H_1}, e_2 \to \varepsilon_{H_2}$.

Select the tuning laws for the critic weights as the normalized gradient descent algorithm

$$\dot{\hat{W}}_1 = -a_1 \frac{\partial E_1}{\partial \hat{W}_1} = -a_1 \frac{\sigma_1}{(1+\sigma_1{}^{\mathrm{T}}\sigma_1)^2}[\sigma_1{}^{\mathrm{T}}\hat{W}_1 + Q_1(x) + u_1{}^T R_{11} u_1 + d_2{}^T R_{12} d_2] \tag{4.31}$$

$$\dot{\hat{W}}_2 = -a_2 \frac{\partial E_1}{\partial \hat{W}_2} = -a_2 \frac{\sigma_2}{(1+\sigma_2{}^{\mathrm{T}}\sigma_2)^2}[\sigma_2{}^{\mathrm{T}}\hat{W}_2 + Q_2(x) + u_1{}^T R_{21} u_1 + d_2{}^T R_{22} d_2] \tag{4.32}$$

where $\sigma_i = \nabla\phi_i(f(x) + g(x)u_1 + k(x)d_2), \quad i = 1,2$. Note that these are modified gradient descent algorithms with the normalizing terms in the denominators raised to the power 2.

Persistence of Excitation (PE) Assumption. Let the signals $\bar{\sigma}_1, \bar{\sigma}_2$ be persistently exciting over the interval $[t, t+T]$, *i.e.* there exist constants $\beta_1 > 0$, $\beta_2 > 0$, $\beta_3 > 0$, $\beta_4 > 0 \quad \mathrm{T} > 0$ such that, for all *t*,

$$\beta_1 \mathrm{I} \le S_0 \equiv \int_t^{t+T} \bar{\sigma}_1(\tau)\bar{\sigma}_1{}^{\mathrm{T}}(\tau) d\tau \le \beta_2 \mathrm{I} \tag{4.33}$$

$$\beta_3 \mathrm{I} \le S_1 \equiv \int_t^{t+T} \bar{\sigma}_2(\tau)\bar{\sigma}_2{}^{\mathrm{T}}(\tau) d\tau \le \beta_4 \mathrm{I} \tag{4.34}$$

where $\bar{\sigma}_i = \sigma_i / (\sigma_i^T \sigma_i + 1), \quad i = 1,2$, and I the identity matrix of appropriate dimensions.

The PE assumption is needed in adaptive control if one desires to perform system identification using e.g. RLS [36], [87]. It is needed in the upcoming Theorem 4.1 because one effectively desires to identify the critic parameters to approximate solutions $V_1(x)$ and $V_2(x)$ to the nonlinear Lyapunov equations (4.6).

The properties of tuning algorithms (4.31) and (4.32) and their exponential convergence to residual sets are given in the following theorem that was proven for a single player in Chapter 2.

Theorem 4.1. Let $u(x(t)), d(x(t))$ be any admissible bounded feedback policies. Let tuning for the critic NNs be provided by (4.31), (4.32) and assume that $\bar{\sigma}_1$ and $\bar{\sigma}_2$ are persistently exciting. Let the residual errors in (4.22) be bounded by $\left\|\varepsilon_{H_1}\right\| < \varepsilon_{\max_1}$ and $\left\|\varepsilon_{H_2}\right\| < \varepsilon_{\max_2}$. Then the critic parameter errors converge exponentially to the residual sets

$$\tilde{W}_1(t) \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1}\left\{\left[1 + 2\delta_1\beta_2 a_1\right]\varepsilon_{\max_1}\right\} \tag{4.35}$$

$$\tilde{W}_2(t) \le \frac{\sqrt{\beta_4 \mathrm{T}}}{\beta_3}\left\{\left[1 + 2\delta_2\beta_4 a_2\right]\varepsilon_{\max_2}\right\} \tag{4.36}$$

where δ_1, δ_2 are positive constants of the order of 1.

Proof: Follows as Chapter 2 (Theorem 2.1).

■

4.4 Online Solution for 2-player games

In this section we develop an optimal adaptive control algorithm that solves the 2-player game problem *online* using data measured along the system trajectories. A special case is the zero-sum 2-player game that was derived in Chapter 3. The technique given here generalizes directly to the *N*-player game. A Lyapunov technique is used to derive novel parameter tuning algorithms for the values and control policies that guarantee closed-loop stability as well as convergence to the approximate game solution of (4.10).

A suitable 'actor-critic' control structure is developed based on the PI Algorithm in the previous section. The adaptive control structure relies on each player maintaining two approximator structures, one for its current value estimate (c.f. (4.13)) and one for its current policy (c.f. (4.14), (4.15)). This structure is based on reinforcement learning precepts.

Define

$$u_1(x) = -\tfrac{1}{2}R_{11}^{-1}g^T(x)\nabla V_1(x) = -\tfrac{1}{2}R_{11}^{-1}g^T(x)\nabla\phi_1^{\,T}(x)W_1 \tag{4.37}$$

$$d_2(x) = -\tfrac{1}{2}R_{22}^{-1}k^T(x)\nabla V_2(x) = -\tfrac{1}{2}R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)W_2 \tag{4.38}$$

with V_1 and V_2 defined in terms of the least squares solutions to (4.20), (4.21) e.g. (3.25) and

(4.26) respectively. According to Proposition 4.2, as $K \to \infty$ (3.42) and (3.43) converge uniformly to (4.8). The next result follows as in [4], [5].

Proposition 4.3. There exists a number of hidden layer units *K₀* such that $u_1(x)$ and $d_2(x)$ are admissible for *K>K₀*.

■

In light of this result, the ideal control policy updates are taken as (4.37) and (4.38) with W_1 and W_2 unknown. Therefore, define the control policies in the form of action neural networks which compute the control inputs in the structured form

$$u_3(x) = -\tfrac{1}{2} R_{11}{}^{-1} g^T(x) \nabla \phi_1{}^T \hat{W}_3 \tag{4.39}$$

$$d_4(x) = -\tfrac{1}{2} R_{22}{}^{-1} k^T(x) \nabla \phi_2{}^T \hat{W}_4 \tag{4.40}$$

where $\hat{W}_3$ and $\hat{W}_4$ denote the current estimated values of the ideal NN weights W_1 and W_2 respectively. Define the critic and the actor NN estimation errors respectively as

$$\tilde{W}_1 = W_1 - \hat{W}_1,\ \tilde{W}_2 = W_2 - \hat{W}_2,\ \tilde{W}_3 = W_1 - \hat{W}_3\ \ \tilde{W}_4 = W_2 - \hat{W}_4. \tag{4.41}$$

Facts 4.1. For a given compact set $\Omega \subset \mathbb{R}^n$:

a. $g(.), k(.)$ are bounded by constants:

$\|g(x)\| < b_g,\ \ \|k(x)\| < b_k$

b. The NN approx errors and their gradients are bounded so that

$\|\varepsilon_1\| < b_{\varepsilon 1}$, $\|\nabla \varepsilon_1\| < b_{\varepsilon_{1x}}$, $\|\varepsilon_2\| < b_{\varepsilon 2}$, $\|\nabla \varepsilon_2\| < b_{\varepsilon_{2x}}$

c. The NN activation functions and their gradients are bounded so that

$\|\phi_1(x)\| < b_{\phi_1}, \|\nabla \phi_1(x)\| < b_{\phi_{1x}}, \|\phi_2(x)\| < b_{\phi_2}, \|\nabla \phi_2(x)\| < b_{\phi_{2x}}$

d. The critic NN weights are bounded by known constants

$\|W_1\| < W_{1\max}$, $\|W_2\| < W_{2\max}$

■

The main Theorems of Chapter 4 are now given, which provide the tuning laws for the actor and critic neural networks that guarantee convergence of the 2-player nonzero-sum game algorithm in real-time to the Nash equilibrium solution, while also guaranteeing closed-loop stability.

Theorem 4.2. Stability and bounded Neural Network weight errors. Let the dynamics be given by (4.16), and consider the 2-player game formulation in Section 4.2. Let the critic NNs be given by (4.27) and (4.28), the control inputs be given (4.39) and (4.40). Let tuning for the critic NNs be provided by

$$\dot{\hat{W}}_1 = -a_1 \frac{\sigma_3}{(\sigma_3^T\sigma_3+1)^2}[\sigma_3{}^T\hat{W}_1 + Q_1(x) + u_3{}^T R_{11}u_3 + d_4{}^T R_{12}d_4] \tag{4.42}$$

$$\dot{\hat{W}}_2 = -a_2 \frac{\sigma_4}{(\sigma_4^T\sigma_4+1)^2}[\sigma_4{}^T\hat{W}_2 + Q_2(x) + u_3{}^T R_{21}u_3 + d_4{}^T R_{22}d_4] \tag{4.43}$$

where $\sigma_3 = \nabla\phi_1(f + gu_3 + kd_4)$ and $\sigma_4 = \nabla\phi_2(f + gu_3 + kd_4)$. Let the first actor NN (first player) be tuned as

$$\dot{\hat{W}}_3 = -a_3\{(F_2\hat{W}_3 - F_1\bar{\sigma}_3^T\hat{W}_1) - \frac{1}{4}(\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\hat{W}_3 m_2{}^T\hat{W}_2 + \bar{D}_1(x)\hat{W}_3 m_1{}^T\hat{W}_1)\} \tag{4.44}$$

and the second actor (second player) NN be tuned as

$$\dot{\hat{W}}_4 = -a_4\{(F_4\hat{W}_4 - F_3\bar{\sigma}_4^T\hat{W}_2) - \frac{1}{4}(\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\hat{W}_4 m_1{}^T\hat{W}_1 + \bar{D}_2(x)\hat{W}_4 m_2{}^T\hat{W}_2)\} \tag{4.45}$$

where

$$\bar{D}_1(x) \equiv \nabla\phi_1(x)g(x)R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T(x),\quad \bar{D}_2(x) \equiv \nabla\phi_2(x)kR_{22}{}^{-1}k^T\nabla\phi_2{}^T(x),\ m_1 \equiv \frac{\sigma_3}{(\sigma_3{}^T\sigma_3+1)^2},$$

$m_2 \equiv \dfrac{\sigma_4}{(\sigma_4{}^T\sigma_4+1)^2}$ and $F_1>0, F_2>0, F_3>0, F_4>0$ are tuning parameters. Let Assumption 4.1 hold, and also assume $Q_1(x)>0$ and $Q_2(x)>0$. Suppose that $\bar{\sigma}_3 = \sigma_3/(\sigma_3^T\sigma_3+1)$ and $\bar{\sigma}_4 = \sigma_4/(\sigma_4^T\sigma_4+1)$ are persistently exciting. Let the tuning parameters be selected as detailed in the proof. Then there exists a *K₀* such that, for the number of hidden layer units $K > K_0$ the closed-loop system state, the critic NN errors $\tilde{W}_1$ and $\tilde{W}_2$, the first actor NN error $\tilde{W}_3$ and the second actor NN error $\tilde{W}_4$ are UUB.

Proof: See appendix A.

■

Remark 4.1. The tuning parameters F_1, F_2, F_3, F_4 in (4.44), and (4.45) must be selected to make the matrix *M* in (A.50) positive definite (more discussion is presented in the appendix).

Remark 4.2. NN usage suggests starting with the initial control NN weights in (4.39) and (4.40) randomly selected and nonzero.

Remark 4.3. The assumption $Q_1(x) > 0$ and $Q_2(x) > 0$ is sufficient but not necessary.

Remark 4.4. Standard neuroadaptive controllers for single players require only 1 NN, namely the control action NN [50]. However, *optimal* neuroadaptive controllers based on RL (e.g. through PI Algorithm) require also a critic NN that identifies the value function for each player. In the optimal control (single-player) case this requires 2 NN, as described in Chapter 2. That is, the price paid for an adaptive controller that converges to an *optimal control* solution is a doubling in the number of NN, which approximately doubles the computational complexity.

Theorem 4.3. Nash solution of the game. Suppose the hypotheses of Theorem 4.2 hold. Then:

a. $H_1(x,\hat{W}_1, \hat{u}_1, \hat{d}_2)$ and $H_2(x,\hat{W}_2, \hat{u}_1, \hat{d}_2)$ are UUB, where

$$\hat{u}_1 = -\tfrac{1}{2} R_{11}^{-1} g^T(x) \nabla \phi_1^T(x) \hat{W}_1 \tag{4.46}$$

$$\hat{d}_2 = -\tfrac{1}{2} R_{22}^{-1} k^T(x) \nabla \phi_2^T(x) \hat{W}_2 \tag{4.47}$$

That is, $\hat{W}_1$ and $\hat{W}_2$ converge to the approximate coupled HJ solution.

b. $u_3(x), d_4(x)$ (see (4.39) and (4.40)) converge to the approximate Nash solution of the game.

Proof: Consider the weights $\tilde{W}_1$, $\tilde{W}_2$, $\tilde{W}_3$ and $\tilde{W}_4$ to be UUB as proved in Theorem 4.2.

a. The approximate coupled HJ equations are

$$H_1(x,\hat{W}_1,\hat{u}_1,\hat{d}_2) \equiv H_1(x,\hat{W}_1,\hat{W}_2) = Q_1(x) - \tfrac{1}{4}\hat{W}_1^{\,T}\nabla\phi_1(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^{\,T}(x)\hat{W}_1$$
$$+\hat{W}_1^{\,T}\nabla\phi_1(x)f(x) - \tfrac{1}{2}\hat{W}_1^{\,T}\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\hat{W}_2 + \tfrac{1}{4}\hat{W}_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\hat{W}_2 - \varepsilon_{HJ_1}$$

and

$$H_2(x,\hat{W}_2,\hat{u}_1,\hat{d}_2) \equiv H_2(x,\hat{W}_1,\hat{W}_2) = Q_2(x) - \tfrac{1}{2}\hat{W}_2^{\,T}\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^{\,T}(x)\hat{W}_1$$
$$+\tfrac{1}{4}\hat{W}_1^{\,T}\nabla\phi_1(x)g(x)R_{11}^{\;-T}R_{21}R_{11}^{\;-1}g^T(x)\nabla\phi_1^{\,T}(x)\hat{W}_1 - \varepsilon_{HJ_2} - \tfrac{1}{4}\hat{W}_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\hat{W}_2 + \hat{W}_2^{\,T}\nabla\phi_2(x)f(x)$$

After adding zero we have

$$H_1(x,\hat{W}_1,\hat{W}_2) = \tilde{W}_1^{\,T}\nabla\phi_1(x)f(x) + \tfrac{1}{4}\tilde{W}_1^{\,T}\overline{D}_1\tilde{W}_1 - \tfrac{1}{2}W_1^{\,T}\overline{D}_1\tilde{W}_1 - \tfrac{1}{4}\hat{W}_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\hat{W}_2$$
$$+\tfrac{1}{4}W_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)W_2 + \tfrac{1}{2}\hat{W}_1^{\,T}\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\hat{W}_2$$
$$-\tfrac{1}{2}W_1^{\,T}\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)W_2 - \varepsilon_{HJ_1} \qquad (4.48)$$

and

$$H_2(x,\hat{W}_1,\hat{W}_2) = \tilde{W}_2^{\,T}\nabla\phi_2(x)f(x) + \tfrac{1}{2}\hat{W}_2^{\,T}\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^{\,T}(x)\hat{W}_1$$
$$-\tfrac{1}{2}W_2^{\,T}\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^{\,T}(x)W_1$$
$$-\tfrac{1}{4}\hat{W}_1^{\,T}\nabla\phi_1(x)g(x)R_{11}^{\;-T}R_{21}R_{11}^{\;-1}g^T(x)\nabla\phi_1^{\,T}(x)\hat{W}_1 + \tfrac{1}{4}W_1^{\,T}\nabla\phi_1(x)g(x)R_{11}^{\;-T}R_{21}R_{11}^{\;-1}g^T(x)\nabla\phi_1^{\,T}(x)W_1$$
$$+\tfrac{1}{4}\tilde{W}_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\tilde{W}_2 - \tfrac{1}{2}W_2^{\,T}\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^{\,T}(x)\tilde{W}_2 - \varepsilon_{HJ_2} \qquad (4.49)$$

But

$$\hat{W}_1 = -\tilde{W}_1 + W_1 \text{ and } \hat{W}_2 = -\tilde{W}_2 + W_2 \qquad (4.50)$$

After taking norms in (4.50) and letting $\|W_1\| < W_{1\max}$ and $\|W_2\| < W_{2\max}$ one has

$$\|\hat{W}_1\| = \|-\tilde{W}_1 + W_1\| \le \|\tilde{W}_1\| + \|W_1\| \le \|\tilde{W}_1\| + W_{1\max} \qquad (4.51)$$

and

$$\|\hat{W}_2\| = \|-\tilde{W}_2 + W_2\| \le \|\tilde{W}_2\| + \|W_2\| \le \|\tilde{W}_2\| + W_{2\max} \qquad (4.52)$$

Now (4.48) becomes, by taking into account (4.51), (4.52), Facts 4.1 and $\sup_{x\in\Omega}\|\varepsilon_{HJ_1}\| < \bar{\varepsilon}_1$

$$\left\|H_1(x,\hat{W}_1,\hat{W}_2)\right\| \le b_{\phi_{1x}} b_f \|x\| \left\|\tilde{W}_1\right\|^T + \tfrac{1}{4}\left\|\tilde{W}_1\right\|^2 \left\|\bar{D}_1\right\| + \tfrac{1}{4}\left(\left\|\tilde{W}_2\right\| + W_{2\max}\right)^2 \left\|\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\|$$
$$+\tfrac{1}{4}W_{2\max}{}^2\left\|\nabla\phi_2(x)k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\|$$
$$+\tfrac{1}{2}\left(\left\|\tilde{W}_1\right\| + W_{1\max}\right)\left(\left\|\tilde{W}_2\right\| + W_{2\max}\right)\left\|\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\| + \tfrac{1}{2}W_{1\max}\left\|\tilde{W}_1\right\|\left\|\bar{D}_1\right\|$$
$$+\tfrac{1}{2}W_{1\max}W_{2\max}\left\|\nabla\phi_1(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\| + \bar{\varepsilon}_1 \qquad (4.53)$$

and same for (4.49) with $\sup\limits_{x\in\Omega}\left\|\varepsilon_{HJ_2}\right\| < \bar{\varepsilon}_2$

$$\left\|H_2(x,\hat{W}_1,\hat{W}_2)\right\| \le b_{\phi_{1x}} b_f \|x\| \left\|\tilde{W}_2\right\| + \tfrac{1}{2}\left(\left\|\tilde{W}_2\right\| + W_{2\max}\right)\left(\left\|\tilde{W}_1\right\| + W_{1\max}\right)\left\|\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^T(x)\right\|$$
$$+\tfrac{1}{2}W_{2\max}W_{1\max}\left\|\nabla\phi_2(x)g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^T(x)\right\| + \tfrac{1}{4}\left(\left\|\tilde{W}_1\right\| + W_{1\max}\right)^2\left\|\nabla\phi_1(x)g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1^T(x)\right\|$$
$$+\tfrac{1}{4}W_{1\max}{}^2\left\|\nabla\phi_1(x)g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1^T(x)\right\| + \tfrac{1}{4}\left\|\tilde{W}_2\right\|^2\left\|\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\|$$
$$+\tfrac{1}{2}W_{2\max}\left\|\tilde{W}_2\right\|\left\|\nabla\phi_2(x)k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^T(x)\right\| + \bar{\varepsilon}_2 \qquad (4.54)$$

All the signals on the right hand side of (4.53) and (4.54) are UUB. So $H_1(x,\hat{W}_1,\hat{u}_1,\hat{d}_2)$ and $H_2(x,\hat{W}_2,\hat{u}_1,\hat{d}_2)$ are UUB and convergence to the approximate coupled HJ solution is obtained.

b. According to Theorem 4.1, 4.2 and equations (3.42), (3.43) and (4.39), (4.40) $\|u_3 - u_1\|$ and $\|d_4 - d_2\|$ are UUB because $\left\|\hat{W}_3 - W_1\right\|$ and $\left\|\hat{W}_4 - W_2\right\|$ are UUB.

So the pair $u_3(x), d_4(x)$ gives the approximate Nash equilibrium solution of the game. This completes the proof.

■

Remark 4.5. The theorems show that PE is needed for proper identification of the value functions by the critic NNs, and that nonstandard tuning algorithms are required for the actor NNs to guarantee stability.

Remark 4.6. Theorems 4.2 and 4.3 show UUB of key quantities. According to the definition of vector *D* in (A.49) and to (4.53), (4.54), the error bounds depend on the NN approximation errors, nonlinear Lyapunov equation and HJ equation residuals, and the bounds $W_{1\max}, W_{2\max}$ on the unknown NN weights. By the Weierstrass Approximation Theorem and Proposition 4.2, all of these go to zero uniformly as the number of NN hidden layers increases except for $W_{1\max}, W_{2\max}$. The question of obtaining improved errors bounds in neuroadaptive control has a long and studied literature. Methods have been developed for removing $W_{1\max}, W_{2\max}$ from the UUB error bounds. See for instance the work of Ge et al (e.g. [29]). Those results could be used to extend the tuning algorithms provided here, but do not constitute a part of the novel results presented herein.

The bounds in the theorems are in fact conservative. The simulation results show that the values and Nash solutions are very closely identified.

4.5 Simulation Results

Here we present simulations of nonlinear and linear systems to show that the game can be solved ONLINE by learning in real time, using the method of this chapter. PE is needed to guarantee convergence to the Nash solution. In these simulations, exponentially decreasing probing noise is added to the control inputs to ensure PE until convergence is obtained.

4.5.1 Nonlinear system

Consider the following affine in control input nonlinear system, with a quadratic cost constructed as in [64]

$$\dot{x} = f(x) + g(x)u + k(x)d, \, x \in \mathbb{R}^2$$

where

$$f(x) = \begin{bmatrix} x_2 \\ -x_2 - \frac{1}{2}x_1 + \frac{1}{4}x_2(\cos(2x_1)+2)^2 + \frac{1}{4}x_2(\sin(4x_1^2)+2)^2 \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}, \quad k(x) = \begin{bmatrix} 0 \\ \sin(4x_1^2)+2 \end{bmatrix}.$$

Select $Q_1 = 2Q_2$, $R_{11} = 2R_{22}$ and $R_{12} = 2R_{21}$, where Q_2, R_{22} and R_{21} are identity matrices.

Select $a_1 = a_2 = a_3 = a_4 = 1$ and $F_4 = F_3 = F_2 = F_1 = 100I$ for the constants on the tuning laws, where I is an identity matrix of appropriate dimensions.

The optimal value function for the first critic (player 1) is

$$V_1^*(x) = \frac{1}{2}x_1^2 + x_2^2$$

and for the second critic (player 2) is

$$V_2^*(x) = \frac{1}{4}x_1^2 + \frac{1}{2}x_2^2 .$$

The optimal control signal for the first player is

$$u^*(x) = -2(\cos(2x_1) + 2)x_2$$

and the optimal control signal for the second player is

$$d^*(x) = -(\sin(4x_1^2) + 2)x_2 .$$

One selects the NN vector activation function for the critics as $\varphi_1(x) = \varphi_2(x) \equiv [x_1^2 \;\; x_1x_2 \;\; x_2^2]$ and uses the control algorithm presented in Theorem 4.2. Each player maintains two NN, a critic NN to estimate its current value and an action NN to estimate its current control policy.

Figure 19 shows the critic parameters for the first player, denoted by $\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3}]^T$ by using the proposed game algorithm. After convergence at about 150s one has $\hat{W}_1(t_f) = [0.5015 \quad 0.0007 \quad 1.0001]^T$. Figure 20 shows the critic parameters for the second player $\hat{W}_2 = [W_{2c1} \quad W_{2c2} \quad W_{2c3}]^T$. After convergence at about 150s one has

$$\hat{W}_2(t_f) = [0.2514 \quad 0.0006 \quad 0.5001]^T .$$

The actor parameters for the first player after 150s converge to the values of $\hat{W}_3(t_f) = [0.5015 \quad 0.0007 \quad 1.0001]^T$, and the actor parameters for the second player converge to the values of

$$\hat{W}_4(t_f) = [0.2514 \quad 0.0006 \quad 0.5001]^T.$$

Therefore the actor NN for the first player

$$\hat{u}(x) = -\tfrac{1}{2} R_{11}^{-1} \begin{bmatrix} 0 \\ \cos(2x_1)+2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.5015 \\ 0.0007 \\ 1.0001 \end{bmatrix}$$

also converged to the optimal control, and similarly for the second player

$$\hat{d}(x) = -\tfrac{1}{2} R_{22}^{-1} \begin{bmatrix} 0 \\ \sin(4x_1^2)+2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix}^T \begin{bmatrix} 0.2514 \\ 0.0006 \\ 0.5001 \end{bmatrix}.$$

The evolution of the system states is presented in Figure 21. After the probing noise is turned off, the states converge to zero. Figure 22 shows the 3-D plot of the difference between the approximated value function for player 1 and the optimal one (similarly for the second player). These errors are close to zero. Good approximations of the actual value functions are being evolved. Figure 23 shows the 3-D plot of the difference between the approximated control for the first player, by using the online algorithm, and the optimal one. This error is close to zero. Similarly for the second player.

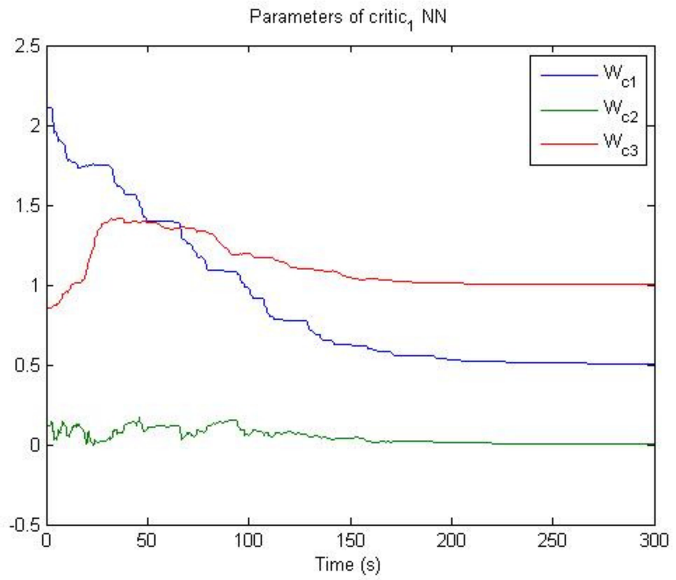

Figure 19. Convergence of the critic parameters for first player.

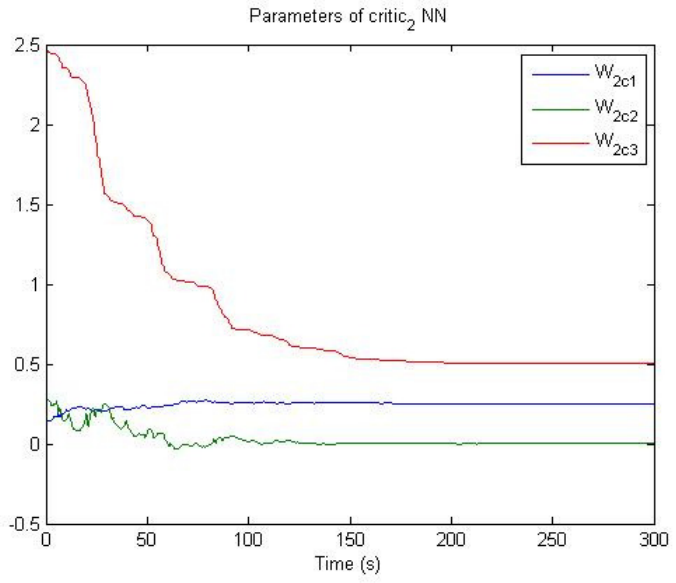

Figure 20. Convergence of the critic parameters for second player.

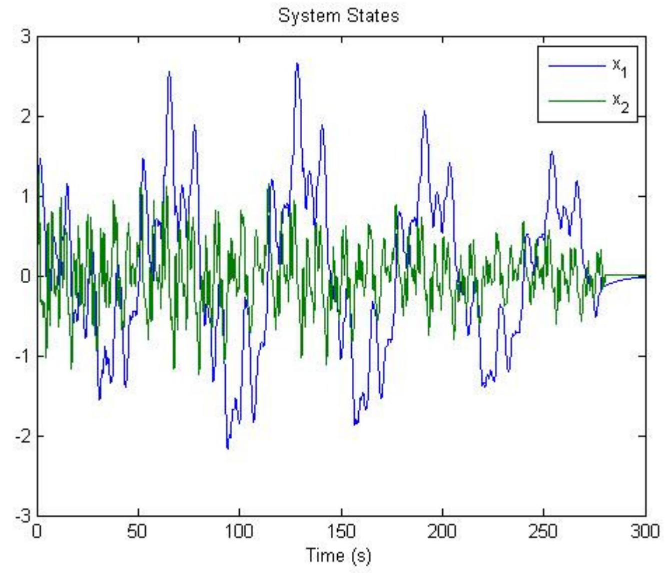

Figure 21. Evolution of the system states.

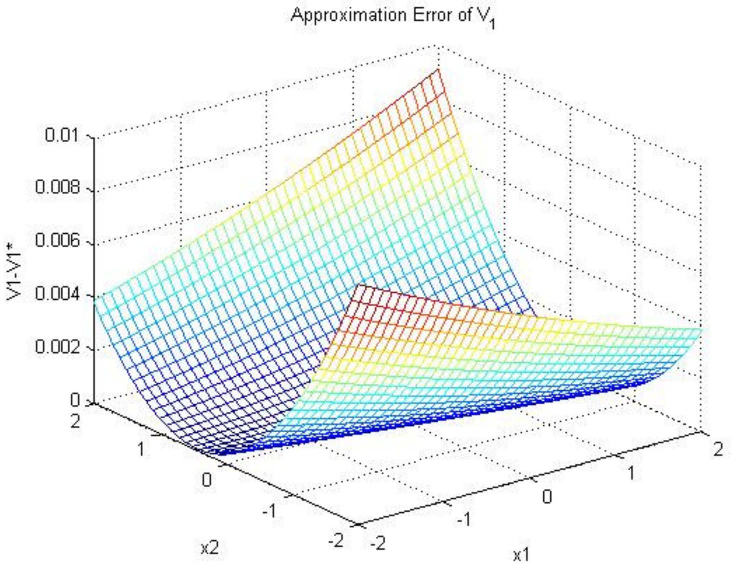

Figure 22. 3D plot of the approximation error for the value function of player 1.

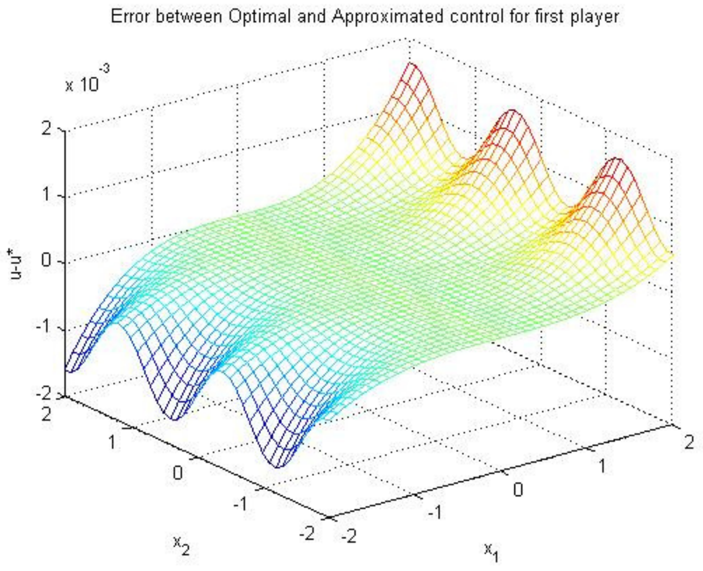

Figure 23. 3D plot of the approximation error for the control of player 1.

4.5.2 Linear system

The aim of this example is to illustrate the online algorithm with a simple example that was simulated in previous work in [39], [60] to solve coupled Riccatis.

Suppose

$$\dot{x}(t) = 2x(t) + u_1(t) + 3u_2(t)$$

with

$$J_1 = \int_0^\infty (-9x^2 + 0.64u_1^2 - u_2^2)\, dt$$

and

$$J_2 = \int_0^\infty (9x^2 + u_1^2)\, dt\ .$$

Select $a_1 = a_2 = a_3 = a_4 = 1$, $F_4 = F_3 = F_2 = F_1 = 5$ for the constants on the tuning laws. According to [60] the solution of the coupled Riccati is given by $P_1 = -1.4145$ and $P_2 = 1.5718$. By using the algorithm proposed in Theorem 4.2 the solution of the coupled Riccati is found online to be $P_1 =$ -1.4250 and $P_2 = 1.5754$. Figure 24 shows the evolution of the system state. Figures 25 and 26 show the convergence of the critic Neural Networks for each player.

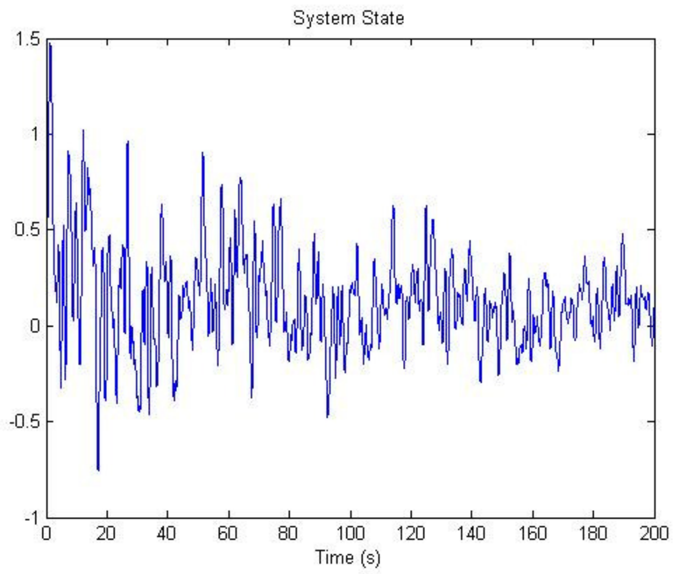

Figure 24. Evolution of the system state.

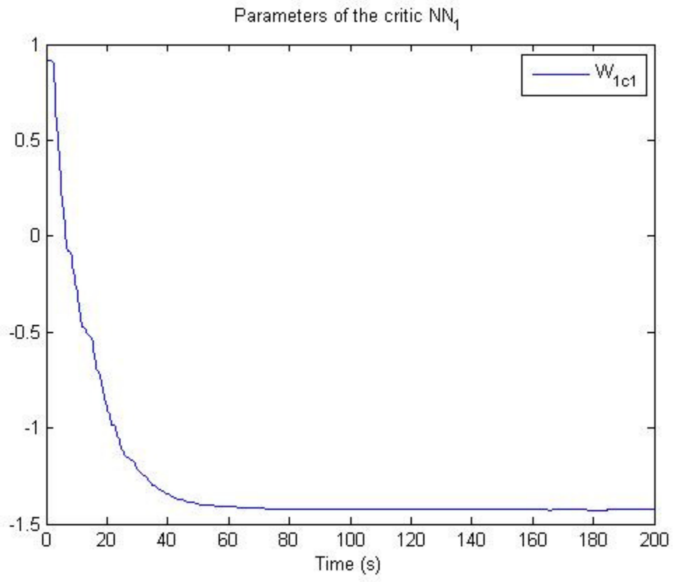

Figure 25. Convergence of the critic NN for the first player.

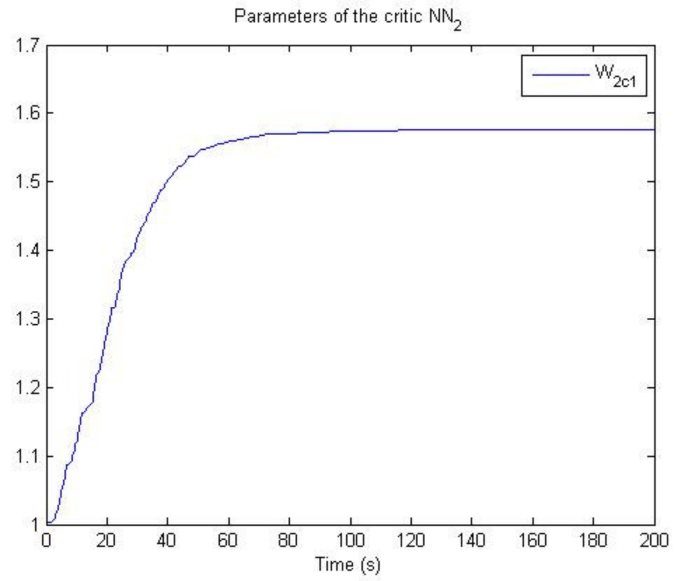

Figure 26. Convergence of the critic NN for the second player.

4.5.3 Zero-Sum Game with unstable linear system

Consider the following unstable linear system,

$$\dot{x} = \begin{bmatrix} 0 & 0.25 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 \\ 0 \end{bmatrix} d$$

with

$$J_1 = \int_0^\infty (x^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x + u^T u - 25 d^T d)\, dt \text{ and } J_2 = -J_1 .$$

One can see that this is a zero-sum game because $Q_1 = -Q_2, \quad R_{11} = -R_{21}, \quad R_{22} = -R_{12}$.

Select $a_1 = a_2 = a_3 = a_4 = 1$, $F_4 = F_3 = F_2 = F_1 = 5I$ for the constants on the tuning laws, where I is an identity matrix of appropriate dimensions. In this linear case the solution is given by the solution of the game algebraic Riccati equation (GARE) with

$$\gamma \equiv \sqrt{R_{22}} = 5 ,\ R \equiv R_{11} = 1 \text{ and } Q \equiv Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

$$A^T P + PA + Q - PBR^{-1}B^T P + \frac{1}{\gamma^2} PKK^T P = 0$$

Solving the GARE gives the parameters of the optimal critic as

$$W^* = [1.9439 \quad 1.3137 \quad 1.9656]^T$$

which are the components of the Riccati solution matrix *P*.

Since the value is quadratic in the LQR case, the critic NNs basis sets $\phi_1(x) = \phi_2(x)$ were selected as the quadratic vector in the state components $x \otimes x$ with $\otimes$ the Kronecker product. Redundant terms were removed to leave $n(n+1)/2 = 3$ components.

The two-player game algorithm is implemented as in Theorem 4.2. PE was ensured by adding a small exponenttial decreasing probing noise to the control inputs. Figure 27 shows the critic parameters for the first player, denoted by $\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3}]^T$ and Figure 28 the critic parameters for the second player, $\hat{W}_2 = [W_{2c1} \quad W_{2c2} \quad W_{2c3}]^T$, converging to the optimal values. It is clear that $\hat{W}_1 = -\hat{W}_2$. In fact after 400s the critic parameters for the first player converged to

$$\hat{W}_1(t_f) = [1.9417 \quad 1.3138 \quad 1.9591]^T$$

and the second to

$$\hat{W}_2(t_f) = [-1.9417 \quad -1.3138 \quad -1.9591]^T .$$

Finally Figure 29 shows the evolution of the states.

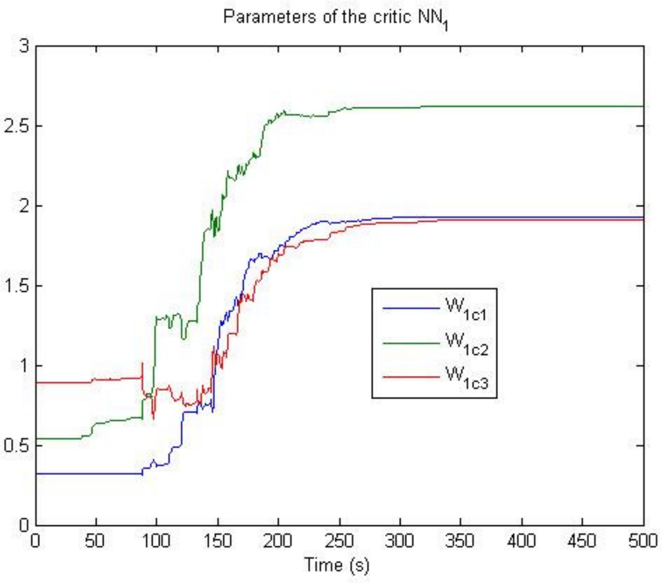

Figure 27. Convergence of the critic NN for the first player.

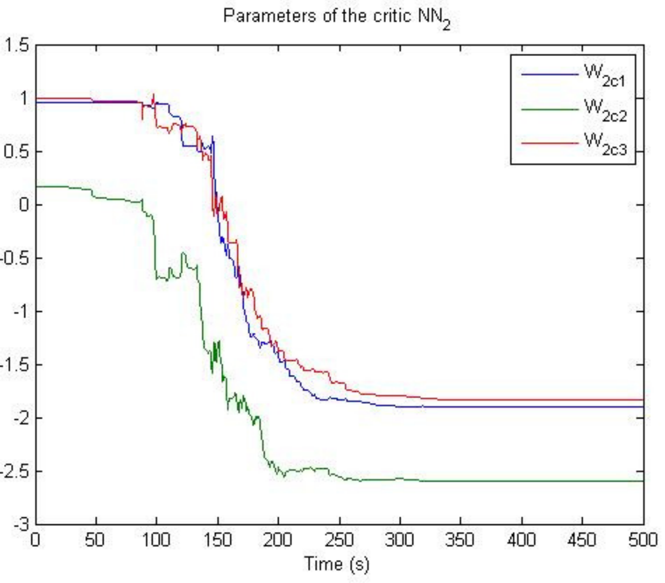

Figure 28. Convergence of the critic NN for the second player.

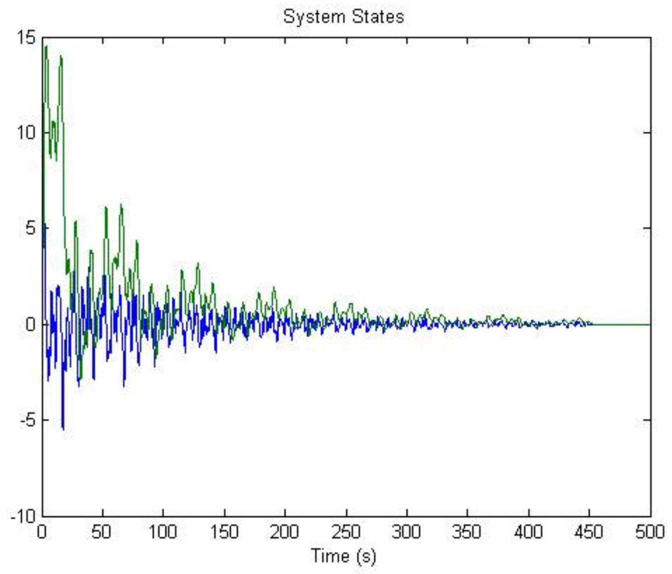

Figure 29. Evolution of the system states.

CHAPTER 5

REINFORCEMENT LEARNING FOR PARTIALLY OBSERVABLE DYNAMIC PROCESSES: ADAPTIVE DYNAMIC PROGRAMMING USING MEASURED OUTPUT DATA

5.1 Introduction

In this work, novel output feedback ADP algorithms are derived for affine in the control input linear time-invariant (LTI) deterministic systems. Such systems have as stochastic equivalent the partially observable Markov decision processes (POMDPs). In this chapter, data-based optimal control is implemented on-line using novel PI and VI ADP algorithms that require only reduced measured information available at the system outputs. These two classes of output feedback algorithms *do not require any knowledge of the system dynamics (A,B,C)* and as such are similar to Q-learning [17], [97], [99], [100] but they have an added advantage of requiring only measurements of input/output data and not the full system state. In order to ensure that the data set is sufficiently rich and linearly independent, there is a need to add (c.f. [17]) probing noise to the control input. We discuss this issue showing that probing noise leads to bias. Adding a discount factor in the cost minimizes it to an almost zero effect. This discount factor is related to adding exponential data weighting in the Kalman Filter to remove the bias effects of unmodeled dynamics [55].

The chapter is organized as follows. Section 5.2 provides the background of the optimal control problem, dynamic programming and reinforcement learning methods for the linear quadratic regulation problem. Section 5.3 introduces the new class of PI and VI algorithms by formulating the temporal difference error with respect to observed data and redefining the control sequence as the output of a dynamic polynomial controller. Section 5.4 discusses the implementation aspects of the output feedback ADP algorithms. For convergence the new algorithms require persistently exciting probing noise whose bias effect is canceled by using a discounted cost function. Section 5.5 presents simulation results obtained using the new data based ADP algorithms and is followed by concluding remarks.

5.2 Background

In this Section we give a review of optimal control, dynamic programming, and reinforcement learning methods (i.e. policy iteration (PI) and value iteration (VI)) for the linear quadratic regulator (LQR). It is pointed out that both these methods employ contraction maps to solve the Bellman equation, which is a fixed-point equation [62]. Both PI and VI methods require *full measurements of the entire state vector.* In the next section we show how to implement PI and VI using only reduced information available at the system outputs.

5.2.1 Dynamic Programming and LQR

Consider the linear time-invariant discrete-time (DT) system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{5.1}$$

with $x_k \in \mathbb{R}^n$ the state, $u_k \in \mathbb{R}^m$ the control input, and $y_k \in \mathbb{R}^p$ the measured output. Assume throughout that *(A, B)* is controllable and *(A, C)* is observable [53].

Given a stabilizing control policy $u_k = \mu(x_k)$, associate to the system the performance index

$$V^{\mu}(x_k) = \sum_{i=k}^{\infty} \left(y_i^T Q y_i + u_i^T R u_i \right) \equiv \sum_{i=k}^{\infty} r_i \tag{5.2}$$

with weighting matrices $Q = Q^T \geq 0, R = R^T > 0$ and $(A, C\sqrt{Q})$ observable. Note that $u_k = \mu(x_k)$ is the fixed policy. The utility is

$$r_k = y_k^T Q y_k + u_k^T R u_k \tag{5.3}$$

The optimal control problem [53] is to find the policy $u_k = \mu(x_k)$ that minimizes the cost (5.2) along the trajectories of the system (5.1). Due to the special structure of the dynamics and the cost this is known as the linear quadratic regulator (LQR) problem.

A difference equation equivalent to (5.2) is given by the Bellman equation

$$V^{\mu}(x_k) = y_k^T Q y_k + u_k^T R u_k + V^{\mu}(x_{k+1}) \tag{5.4}$$

The optimal cost, or value, is given by

$$V^*(x_k) = \min_{\mu} \sum_{i=k}^{\infty} \left(y_i^T Q y_i + u_i^T R u_i \right). \tag{5.5}$$

According to Bellman's optimality principle the value may be determined using the Hamilton-Jacobi-Bellman (HJB) equation

$$V^*(x_k) = \min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right) \tag{5.6}$$

and the optimal control is given by

$$\mu^*(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right) \tag{5.7}$$

For the LQR case, any value is quadratic in the state so that the cost associated to any policy $u_k = \mu(x_k)$ (not necessary optimal) is

$$V^{\mu}(x_k) = x_k^T P x_k \tag{5.8}$$

for some $n \times n$ matrix P. Substituting this into (5.4) one obtains the LQR Bellman equation

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \tag{5.9}$$

If the policy is a linear state variable feedback so that

$$u_k = \mu(x_k) = -K x_k \tag{5.10}$$

then the closed loop system is

$$x_{k+1} = (A - BK) x_k \equiv A_c x_k \tag{5.11}$$

Inserting these equations into (5.9) and averaging over all state trajectories yields the Lyapunov equation

$$0 = (A - BK)^T P(A - BK) - P + C^T QC + K^T RK \tag{5.12}$$

If the feedback *K* is stabilizing, and *(A, C)* is observable, there exists a positive definite solution to this equation. Then, the Lyapunov solution gives the value of using the state feedback *K*, i.e. solution of this equation gives the kernel *P* such that $V^{\mu}(x_k) = x_k^T P x_k$.

To find the optimal control, insert (5.1) into (5.9) to obtain

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + (Ax_k + Bu_k)^T P(Ax_k + Bu_k) \tag{5.13}$$

To determine the minimizing control, set the derivative with respect to u_k equal to zero to obtain

$$u_k = -(R + B^T PB)^{-1} B^T PAx_k \tag{5.14}$$

whence substitution into (5.13) yields the Riccati equation

$$0 = A^T PA - P + C^T QC - A^T PB(R + B^T PB)^{-1} B^T PA \tag{5.15}$$

This is the LQR equivalent to the HJB equation (5.6).

5.2.2 Temporal difference, policy iteration, value iteration

It is well known that the optimal value and control can be determined online in real time using temporal difference reinforcement learning methods ([20], [68], [78], [86], [99]), which rely on solving online for the value that makes small the so-called Bellman temporal difference error

$$e_k = -V^{\mu}(x_k) + y_k^T Q y_k + u_k^T R u_k + V^{\mu}(x_{k+1}) \tag{5.16}$$

The temporal difference error is defined based on the Bellman equation (5.4). For use in any practical real-time algorithm, the value should be approximated by a parametric structure [15], [98], [99].

For the LQR case, the value is quadratic in the state and the Bellman temporal difference error is

$$e_k = -x_k^T P x_k + y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \tag{5.17}$$

Given a control policy, solution of this equation is equivalent to solving Lyapunov equation (5.12) and gives the kernel *P* such that $V^{\mu}(x_k) = x_k^T P x_k$.

Write (5.6) equivalently as

$$0 = \min_{u_k}\left(-V^*(x_k) + y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1})\right) \tag{5.18}$$

This is a fixed point equation. As such, it can be solved by the method of successive approximation using a contraction map. The successive approximation method resulting from this fixed point equation is known as Policy Iteration (PI), an iterative method of determining the optimal value and policy. For the LQR, PI is performed by the following two steps, based on the temporal difference error (5.17) and a policy update step based on (5.7).

Algorithm 5.1- PI

Select a stabilizing initial control policy $u_k^0 = \mu^0(x_k)$. Then, for $j = 0,1,...$ perform until convergence:

1. **Policy Evaluation.** Using the policy $u_k^j = \mu^j(x_k)$ in (5.1), solve for P^{j+1} such that

$$0 = -x_k^T P^{j+1} x_k + y_k^T Q y_k + (u_k^j)^T R u_k^j + x_{k+1}^T P^{j+1} x_{k+1} \tag{5.19}$$

2. **Policy Improvement.**

$$u_k^{j+1} = \mu^{j+1}(x_k) = \arg\min_{u_k}\left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1}\right) \tag{5.20}$$

■

The solution in the policy evaluation step is generally carried out in a least-squares sense. The initial policy is required to be stable since only then does (5.19) have a meaningful solution.

This algorithm is equivalent to the following, which uses the Lyapunov equation (5.12), instead of the Bellman equation, and (5.14).

Algorithm 5.2- PI Lyapunov Iteration Equivalent

Select a stabilizing initial control policy K^0. Then, for $j = 0,1,...$ perform until convergence:

1. **Policy Evaluation.**

$$0 = (A - BK^j)^T P^{j+1}(A - BK^j) - P^{j+1} + C^T QC + (K^j)^T RK^j \tag{5.21}$$

2. **Policy Improvement.**

$$K^{j+1} = (R + B^T P^{j+1} B)^{-1} B^T P^{j+1} A \tag{5.22}$$

■

It was shown in [15] that under general conditions, the policy μ^{j+1} obtained by (5.20) is improved over the μ^j in the sense that $V^{\mu^{j+1}}(x_k) \le V^{\mu^j}(x_k)$. It was shown by Hewer [20] that Algorithm 5.2 converges under the controllability/observability assumptions if the initial feedback gain is stabilizing.

Note that in PI Algorithm 5.2, the system dynamics *(A, B)* are required for the policy evaluation step, while in PI Algorithm 5.1 they are not. Algorithm 5.2 is performed off-line knowing the state dynamics.

On the other hand, Algorithm 5.1 is performed on-line in real-time as the data (x_k, r_k, x_{k+1}) are measured at each time step, with $r_k = y_k^T Q y_k + u_k^T R u_k$ the utility. Note that (5.19) is a scalar equation, whereas the value kernel *P* is a symmetric $n \times n$ matrix with $n(n+1)/2$ independent elements. Therefore, $n(n+1)/2$ data sets are required before (5.19) can be solved. This is a standard problem in least-squares estimation. The policy evaluation step may be performed using batch least-squares, as enough data are collected along the system trajectory, or using recursive least-squares (RLS). The dynamics *(A, B)* are not required for this, since the state x_{k+1} is measured at each step, which contains implicit information about the dynamics *(A, B)*. This procedure amounts to a stochastic approximation method that evaluates the performance of a given policy along one sample path, e.g. the system trajectory. PI Algorithm 5.1 effectively provides a method for solving the Riccati equation (5.15) online using data measured along the system trajectories. Full state measurements of x_k are required.

A second class of algorithms for on-line iterative solution of the optimal control based on the Bellman temporal difference error (5.17) is given by Value Iteration (VI) or heuristic dynamic programming (HDP) [99]. Instead of the fixed point equation in the form (5.18), which leads to policy iteration, consider (5.6),

$$V^*(x_k) = \min_{u_k}\left(y_k^T Q y_k + u_k^T R u_k + V^*(x_{k+1}) \right) \tag{5.23}$$

which is also a fixed point equation. As such, it can be solved using a contraction map by successive approximation using the Value Iteration (VI) algorithm.

Algorithm 5.3- VI

Select an initial control policy K^0. Then, for $j = 0,1,...$ perform until convergence:

1. **Value Update.**

$$x_k^T P^{j+1} x_k = y_k^T Q y_k + (u_k^j)^T R u_k^j + x_{k+1}^T P^j x_{k+1} \tag{5.24}$$

2. **Policy Improvement.**

$$u_k^{j+1} = \mu^{j+1}(x_k) = \arg\min_{u_k}\left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1} \right) \tag{5.25}$$

■

Since (5.23) is not an equation but a recursion, its implementation does not require a stabilizing policy. Therefore, the VI algorithm does not require an initial stabilizing gain. It is equivalent to a matrix recursion knowing the system dynamics *(A, B)*, which has been shown to converge in [46].

Note that the policy update steps in PI Algorithm 5.1 and in VI Algorithm 5.3 rely on the hypothesis of the value function approximation (VFA) parameterization (5.8). Both algorithms require knowledge of the dynamics *(A, B)* for the policy improvement step. In [7] it is shown that if a second approximator structure is assumed for the policy, then only the *B* matrix is needed for the policy improvement in Algorithm 5.1 or 5.3.

An approach which provides on-line real-time algorithms for solution of the optimal control problem without knowing any system dynamics is Q learning. This has been applied to both PI [17] and VI, where it is known as action dependent HDP [99].

All these methods require measurement of the full state $x_k \in \mathbb{R}^n$.

5.3 Temporal Difference, PI, and VI based on output feedback

This section presents the new results of this chapter. The Bellman error (5.17) for LQR is quadratic in the state. This can be used in a PI or VI algorithm for online learning of optimal controls as long as full measurements of state $x_k \in \mathbb{R}^n$ are available. In this section we show how to write the Bellman temporal difference error in terms *only of the observed data*, namely the input sequence u_k and the output sequence y_k.

The main result of the following equations is (5.45), (5.46) which give a temporal difference error in terms only of observed output data.

To reformulate policy iteration and value iteration in terms only of the observed data, we show how to write $V^{\mu}(x_k) = x_k^T P x_k$ as a quadratic form in terms of the input and output sequences. A surprising benefit is that there result two algorithms for reinforcement learning that *do not require any knowledge of the system dynamics (A, B, C).* That is, these algorithms have the same advantage as Q-learning in not requiring knowledge of the system dynamics, yet they have the added benefit of requiring only measurements of the available input/output data, not the full system state as required by Q learning.

5.3.1 Writing the value function in terms of available measured data

Consider the deterministic linear time invariant system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{5.26}$$

with $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, and $y_k \in \mathbb{R}^p$. Assume that *(A, B)* is controllable and *(A, C)* is observable [53]. Controllability is the property of matrices *(A, B)* and means that any initial state can be driven to any desired final state. Observability is a property of *(A, C)* and means that observations of the output y_k over a long enough time horizon can be used to reconstruct the full state x_k. Given the current time *k*, the dynamics can be written on a time horizon *[k-N, k]* as the expanded state equation

$$x_k = A^N x_{k-N} + \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-1}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{5.27}$$

$$\begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix} = \begin{bmatrix} CA^{N-1} \\ \vdots \\ CA \\ C \end{bmatrix} x_{k-N} + \begin{bmatrix} 0 & CB & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & & 0 & CB \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{5.28}$$

or by appropriate definition of variables as

$$x_k = A^N x_{k-N} + U_N \bar{u}_{k-1,k-N} \tag{5.29}$$

$$\bar{y}_{k-1,k-N} = V_N x_{k-N} + T_N \bar{u}_{k-1,k-N} \tag{5.30}$$

with

$$U_N = \begin{bmatrix} B & AB & \cdots & A^{N-1}B \end{bmatrix}$$

the controllability matrix and

$$V_N = \begin{bmatrix} CA^{N-1} \\ \vdots \\ CA \\ C \end{bmatrix} \tag{5.31}$$

the observability matrix, where $V_N \in \mathbb{R}^{pN \times n}$. T_N is the Toeplitz matrix of Markov parameters.

Vectors

$$\bar{y}_{k-1,k-N} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix} \in R^{pN}, \qquad \bar{u}_{k-1,k-N} = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} \in R^{mN}$$

are the input and output sequences over the time interval *[k-N, k-1]*. They represent the available measured data.

Since *(A, C)* is observable, there exists a *K*, the observability index, such that $rank(V_N) < n$ for $N < K$, $rank(V_N) = n$ for $N \geq K$. Note that *K* satisfies $Kp \geq n$. Let $N \geq K$. Then V_N has full column rank *n*, and there exists a matrix $M \in \mathbb{R}^{n \times pN}$ such that

$$A^N = MV_N . \tag{5.32}$$

This was used in [1] to find the optimal control through identification of the Markov parameters.

Since V_N has full column rank, its left inverse is given as

$$V_N^+ = (V_N^T V_N)^{-1} V_N^T \tag{5.33}$$

so that

$$M = A^N V_N^+ + Z(I - V_N V_N^+) \equiv M_0 + M_1 \tag{5.34}$$

for any matrix Z, with M_0 denoting the minimum norm operator and $P(R^{\perp}(V_N)) = I - V_N V_N^{+}$ the projection onto range perpendicular of V_N.

The following Lemma shows how to write the system state in terms of input/output data.

Lemma 5.1. Let the system (5.26) be observable. Then the system state is given uniquely in terms of the measured input/output sequences by

$$x_k = M_0 \bar{y}_{k-1,k-N} + \left(U_N - M_0 T_N\right)\bar{u}_{k-1,k-N} \equiv M_y \bar{y}_{k-1,k-N} + M_u \bar{u}_{k-1,k-N} \tag{5.35}$$

or

$$x_k = \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{y}_{k-1,k-N} \end{bmatrix} \tag{5.36}$$

where $M_y = M_0$ and $M_u = U_N - M_0 T_N$

with $M_0 = A^N V_N^{+}$, $V_N^{+} = (V_N^T V_N)^{-1} V_N^T$ the left inverse of the observability matrix (5.31), and $N \geq K$, where K is the observability index.

Proof: Note that $A^N x_{k-N} = MV_N x_{k-N}$, so that according to (5.30)

$$A^N x_{k-N} = MV_N x_{k-N} = M\bar{y}_{k-1,k-N} - MT_N \bar{u}_{k-1,k-N} \tag{5.37}$$

$$(M_0 + M_1)V_N x_{k-N} = (M_0 + M_1)\bar{y}_{k-1,k-N} - (M_0 + M_1)T_N \bar{u}_{k-1,k-N} \tag{5.38}$$

Note however that $M_1 V_N = 0$ so that

$$MV_N x_{k-N} = M_0 V_N x_{k-N},$$

and apply M_1 to (5.30) to see that

$$0 = M_1 \bar{y}_{k-1,k-N} - M_1 T_N \bar{u}_{k-1,k-N}, \quad \forall M_1 \; s.t. \; M_1 V_N = 0 \tag{5.39}$$

Therefore,

$$A^N x_{k-N} = M_0 V_N x_{k-N} = M_0 \bar{y}_{k-1,k-N} - M_0 T_N \bar{u}_{k-1,k-N} \tag{5.40}$$

independently of *M₁*. Then, from (5.29)

$$x_k = M_0 \bar{y}_{k-1,k-N} + \left(U_N - M_0 T_N\right) \bar{u}_{k-1,k-N} \equiv M_y \bar{y}_{k-1,k-N} + M_u \bar{u}_{k-1,k-N} \tag{5.41}$$

■

This result expresses x_k in terms of the system inputs and outputs from time *k-N* through time *k-1*. Now we will express the value function in terms of the inputs and outputs.

It is important to note that the system dynamics information (e.g. *A*, *B*, *C*) must be known to use (5.36). In fact, $M_y = M_0$ is given in (5.34) where V_N^+ depends on *A* and *C*. Also, M_u depends on M_0, U_N, T_N. U_N is given in (5.27), (5.29) in terms of *A* and *B*. T_N in (5.28), (5.30) depends on *A*, *B* and *C*.

In the next step, it is shown how to use the structural dependence in (5.35) yet avoid knowledge of *A*, *B* and *C*.

Define the vector of observed data at time *k* as

$$\bar{z}_{k-1,k-N} = \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{y}_{k-1,k-N} \end{bmatrix}. \tag{5.42}$$

Now one has

$$V^{\mu}(x_k) = x_k^T P x_k = \bar{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T \\ M_y^T \end{bmatrix} P \begin{bmatrix} M_u & M_y \end{bmatrix} \bar{z}_{k-1,k-N} \tag{5.43}$$

$$V^{\mu}(x_k) = \bar{z}_{k-1,k-N}^T \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix} \bar{z}_{k-1,k-N} \equiv \bar{z}_{k-1,k-N}^T \bar{P}\, \bar{z}_{k-1,k-N} \tag{5.44}$$

Note that $\bar{u}_{k-1,k-N} \in \mathbb{R}^{mN}$, $\bar{y}_{k-1,k-N} \in \mathbb{R}^{pN}$, $\bar{z}_{k-1,k-N} \in \mathbb{R}^{(m+p)N}$, $\bar{P} \in \mathbb{R}^{(m+p)N \times (m+p)N}$.

Equation (5.44) expresses the value function at time *k* as a quadratic form in terms of past inputs and outputs.

Note, that the inner kernel matrix $\bar{P}$ in (5.44) depends on the system dynamics *A*, *B* and *C*. In the next section it is shown how to use reinforcement learning methods to learn the kernel matrix $\bar{P}$ without knowing *A*, *B* and *C*.

5.3.2 Writing the TD error in terms of available measured data

We may now write Bellman's equation (5.9) in terms of the observed data as

$$\bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} = y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} . \tag{5.45}$$

Based on this equation, write the temporal difference error (5.17) in terms of inputs and outputs as

$$e_k = -\bar{z}_{k-1,k-N}^T \bar{P} \bar{z}_{k-1,k-N} + y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} \tag{5.46}$$

Using this TD error, the policy evaluation step of any form of reinforcement learning based on the Bellman temporal difference error (5.17) can be equivalently performed using only the measured data, not the state.

Matrix $\bar{P}$ depends on *A*, *B* and *C* through M_y and M_u. However, reinforcement learning methods allow one to learn $\bar{P}$ online without *A*, *B*, *C*, as shown next.

5.3.3 Writing the Policy Update in terms of Available Measured Data

Using Q learning, one can perform PI and VI without any knowledge of the system dynamics. Likewise, it is now shown that, using the above constructions, one can derive a form for the policy improvement step (5.20)/(5.25) that does not depend on the state or the dynamics, but only on the measured input/output data.

The policy improvement step may be written in terms of the observed data as

$$\mu(x_k) = \arg\min_{u_k} \left(y_k^T Q y_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \right) \tag{5.47}$$

$$\mu(x_k) = \arg\min_{u_k}\left(y_k^T Q y_k + u_k^T R u_k + \bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} \right) \tag{5.48}$$

Partition $\bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1}$ as

$$\bar{z}_{k,k-N+1}^T \bar{P} \bar{z}_{k,k-N+1} = \begin{bmatrix} u_k \\ \bar{u}_{k-1,k-N+1} \\ \bar{y}_{k,k-N+1} \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & P_{22} & P_{23} \\ p_y^T & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} u_k \\ \bar{u}_{k-1,k-N+1} \\ \bar{y}_{k,k-N+1} \end{bmatrix} \tag{5.49}$$

One has $p_0 \in \mathbb{R}^{m\times m}$, $p_u \in \mathbb{R}^{m\times(m(N-1))}$, $p_y \in \mathbb{R}^{m\times pN}$. Then, differentiating with respect to u_k to perform the minimization in (5.48) yields

$$0 = Ru_k + p_0 u_k + p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} \tag{5.50}$$

or

$$u_k = -(R + p_0)^{-1}\left(p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} \right) \tag{5.51}$$

This is a dynamic polynomial auto regression moving average (ARMA) controller that generates the current control input u_k in terms of previous inputs and the current and previous outputs.

Exactly as in Q learning (called by Werbos *action-dependent* learning [99]), the control input appears in the quadratic form (5.49), so that the minimization in (5.48) can be carried out in terms of the learned kernel matrix $\bar{P}$ without resort to the system dynamics. However, since (5.49) contains present and past values of the input and the output, the result is a dynamical controller in polynomial ARMA form.

We have developed the following reinforcement learning algorithms which only use the measured input/output data, and do not require measurements of the full state vector x_k.

Algorithm 5.4- PI Algorithm Using Output Feedback

Select a stabilizing initial control policy $u_k^0 = \mu^0$. Then, for $j = 0,1,...$ perform until convergence:

1. **Policy Evaluation.** Solve for $\overline{P}^{j+1}$ such that

$$0 = -\overline{z}_{k-1,k-N}^T \overline{P}^{j+1} \overline{z}_{k-1,k-N} + y_k^T Q y_k + (u_k^j)^T R u_k^j + \overline{z}_{k,k-N+1}^T \overline{P}^{j+1} \overline{z}_{k,k-N+1} \tag{5.52}$$

2. **Policy Improvement.** Partition $\overline{P}$ as in (5.49). Then define the updated policy by

$$u_k^{j+1} = \mu^{j+1}(x_k) = -(R + p_0^{j+1})^{-1} \cdot \left(p_u^{j+1} \overline{u}_{k-1,k-N+1} + p_y^{j+1} \overline{y}_{k,k-N+1} \right) \tag{5.53}$$

■

Algorithm 5.5- VI Algorithm Using Output Feedback

Select any initial control policy $u_k^0 = \mu^0$. Then, for $j = 0,1,...$ perform until convergence:

1. **Policy Evaluation.** Solve for $\overline{P}^{j+1}$ such that

$$\overline{z}_{k-1,k-N}^T \overline{P}^{j+1} \overline{z}_{k-1,k-N} = y_k^T Q y_k + (u_k^j)^T R u_k^j + \overline{z}_{k,k-N+1}^T \overline{P}^{j} \overline{z}_{k,k-N+1} \tag{5.54}$$

2. **Policy Improvement.** Partition $\overline{P}$ as in (5.49). Then define the updated policy by

$$u_k^{j+1} = \mu^{j+1}(x_k) = -(R + p_0^{j+1})^{-1} \cdot \left(p_u^{j+1} \overline{u}_{k-1,k-N+1} + p_y^{j+1} \overline{y}_{k,k-N+1} \right) \tag{5.55}$$

■

Remark 5.1. These algorithms do not require measurements of the internal state vector x_k. They only require measurements at each time step *k* of the utility $r_k = y_k^T Q y_k + u_k^T R u_k$, the inputs from time *k-N* through time *k*, and the outputs from time *k-N* through time *k*. The policy evaluation step may be implemented using standard methods such as batch least-squares or Recursive Least Squares (RLS) (see next section).

Remark 5.2. The control policy given by these algorithms in the form of (5.51), (5.53), (5.55) is a dynamic ARMA regulator in terms of past inputs and current and past outputs. As such, it optimizes a polynomial square-of-sums cost function that is equivalent to the LQR sum-of-squares cost function (5.2). This polynomial cost function could be determined using the techniques in [53].

Remark 5.3. The PI Algorithm 5.4 requires an initial stabilizing control policy, while the VI Algorithm 5.5 does not. As such, VI is suitable for control of open-loop unstable systems.

Remark 5.4. The PI algorithm requires an initial stabilizing control policy $u_k^0 = \mu^0$ that is required to be a function only of the observable data. Suppose one can find an initial stabilizing state feedback gain $u_k^0 = \mu^0(x_k) = -K^0 x_k$. Then the equivalent stabilizing output feedback ARMA controller is easy to find and is given by

$$u_k^0 = \mu^0(x_k) = -K^0 x_k = -K^0 \begin{bmatrix} M_u & M_y \end{bmatrix} \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{y}_{k-1,k-N} \end{bmatrix}$$

The next result shows that the controller (5.51) is unique.

Lemma 5.2. Define M_0, M_1 according to (5.34). Then the control sequence generated by (5.51) is independent of M_1 and depends only on M_0. Moreover, (5.51) is equivalent to

$$u_k = -(R + B^T PB)^{-1} \left(p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} \right) \tag{5.56}$$

where p_u, p_y depend only on M_0.

Proof: Write (5.50) as

$$0 = Ru_k + p_0 u_k + p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} = Ru_k + \begin{bmatrix} p_0 & p_u & | & p_y \end{bmatrix} \begin{bmatrix} \bar{u}_{k,k-N+1} \\ \bar{y}_{k,k-N+1} \end{bmatrix}$$

According to (5.44) and (5.49)

$$\begin{bmatrix} p_0 & p_u \end{bmatrix} = \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T PM_u, \quad p_y = \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T PM_y$$

therefore

$$0 = Ru_k + p_0 u_k + p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} = Ru_k + \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P\left(M_u \bar{u}_{k,k-N+1} + M_y \bar{y}_{k,k-N+1}\right)$$

According to Lemma 5.1 this is unique independently of *M₁* (see (5.38)) and equal to

$$0 = Ru_k + \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P\left(\left(U_N - M_0 T_N\right)\bar{u}_{k,k-N+1} + M_0 \bar{y}_{k,k-N+1}\right)$$

Now

$$\begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P = \left(M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix}\right)^T P .$$

However

$$M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix} = \left(U_N - (M_0 + M_1) T_N\right) \begin{bmatrix} I_m \\ 0 \end{bmatrix} = B$$

where one has used the structure of U_N, T_N . Consequently

$$0 = Ru_k + p_0 u_k + p_u \bar{u}_{k-1,k-N+1} + p_y \bar{y}_{k,k-N+1} = Ru_k + B^T P\left(\left(U_N - M_0 T_N\right)\bar{u}_{k,k-N+1} + M_0 \bar{y}_{k,k-N+1}\right)$$

which is independent of M_1 .

Now note that $p_0 = \begin{bmatrix} I_m & 0 \end{bmatrix} M_u^T P M_u \begin{bmatrix} I_m \\ 0 \end{bmatrix} = B^T P B$ as per the above. Hence (5.56) follows from (5.51).

■

Remark 5.5. Note that the controller cannot be implemented in the form (5.56), since it requires that matrix *P* be known.

5.4 Implementation, probing noise, bias and discount factors

In this section we discuss the need for probing noise to implement the above algorithms, show that this noise leads to deleterious effects such as bias, and argue how adding a discount factor in the cost (5.2) can reduce this bias to a negligible effect.

The equations in Policy Iteration and Value Iteration are solved online by standard techniques using methods such as batch least-squares or recursive least-squares. See [17] where RLS was used in Q-learning. In PI, one solves (5.52) by writing it in the form

$$stk(\bar{P}^{j+1})\left[\bar{z}_{k-1,k-N} \otimes \bar{z}_{k-1,k-N} - \bar{z}_{k,k-N+1} \otimes \bar{z}_{k,k-N+1}\right] = y_k^T Q y_k + (u_k^j)^T R u_k^j \qquad (5.57)$$

with $\otimes$ the Kronecker product and stk(.) the column stacking operator [18]. Redundant quadratic terms in the Kronecker product are combined. In VI one solves (5.54) in the form

$$stk(\bar{P}^{j+1})\left[\bar{z}_{k-1,k-N} \otimes \bar{z}_{k-1,k-N}\right] = y_k^T Q y_k + (u_k^j)^T R u_k^j + \bar{z}_{k,k-N+1}^T \bar{P}^j \bar{z}_{k,k-N+1} \qquad (5.58)$$

Both of these equations only require that the input/output data be measured. They are scalar equations, yet one must solve for the kernel matrix $\bar{P} \in \mathbb{R}^{(m+p)N \times (m+p)N}$, which is symmetric and has $\left[(m+p)N\right]\left[(m+p)N+1\right]/2$ independent terms. Therefore, one requires data samples for at least $\left[(m+p)N\right]\left[(m+p)N+1\right]/2$ time steps for solution using batch LS.

To solve the PI update equation (5.57), it is required that the quadratic vector $\left[\bar{z}_{k-1,k-N} \otimes \bar{z}_{k-1,k-N} - \bar{z}_{k,k-N+1} \otimes \bar{z}_{k,k-N+1}\right]$ be linearly independent over time, a property known as persistence of excitation (PE). To solve the VI update equation (5.58), one requires PE of the quadratic vector $\left[\bar{z}_{k-1,k-N} \otimes \bar{z}_{k-1,k-N}\right]$. It is standard practice (see [17] for instance) to inject probing noise into the control action to obtain PE, so that one puts into the system dynamics the input $\hat{u}_k = u_k + d_k$, with u_k the control computed by the PI or VI current policy and d_k a probing noise or dither, e.g. white noise.

It is well known that dither can caused biased results and mismatch in system identification. In [102] this issue is discussed and several alternative methods are presented for injecting dither into system identification schemes to obtain improved results. Unfortunately, in control applications, one has little choice about where to inject the probing noise.

To see the deleterious effects of probing noise, consider the Bellman equation (5.9) with input $\hat{u}_k = u_k + d_k$, where d_k is a probing noise. One writes

$$x_k^T \hat{P} x_k = y_k^T Q y_k + \hat{u}_k{}^T R \hat{u}_k + \hat{x}_{k+1}^T \hat{P} \hat{x}_{k+1}$$
$$x_k^T \hat{P} x_k = y_k^T Q y_k + (u_k + d_k)^T R (u_k + d_k) + (Ax_k + Bu_k + Bd_k)^T \hat{P} (Ax_k + Bu_k + Bd_k)$$

$$\begin{aligned} x_k^T \hat{P} x_k = {} & y_k^T Q y_k + u_k{}^T R u_k + d_k{}^T R d_k + u_k{}^T R d_k + d_k{}^T R u_k + (Ax_k + Bu_k)^T \hat{P} (Ax_k + Bu_k) \\ & + (Ax_k + Bu_k)^T \hat{P} B d_k + (Bd_k)^T \hat{P} (Ax_k + Bu_k) + (Bd_k)^T \hat{P} B d_k \end{aligned}$$

Now, use $trace\{AB\} = trace\{BA\}$ for commensurate matrices *A* and *B*, take expected values to evaluate correlation matrices and assume the dither at time *k* is white noise independent of u_k, x_k so that $E\{R u_k d_k{}^T\} = 0,\ E\{\hat{P} B d_k (Ax_k + Bu_k)^T\} = 0$ and the cross terms drop out. Then, averaged over repeated control runs with different probing noise sequence d_k, this equation is effectively

$$x_k^T \hat{P} x_k = y_k^T Q y_k + u_k{}^T R u_k + (Ax_k + Bu_k)^T \hat{P} (Ax_k + Bu_k) + d_k{}^T (B^T \hat{P} B + R) d_k$$

which is the undithered Bellman equation plus a term depending on the dither covariance. As such, the solution computed by PI or VI will not correspond to the actual value corresponding to the Bellman equation.

It is now argued that discounting the cost can significantly reduce the deleterious effects of probing noise. Adding a discount factor $\gamma < 1$ to the cost (5.2) results in

$$V^{\mu}(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} \left(y_i^T Q y_i + u_i^T R u_i \right) \equiv \sum_{i=k}^{\infty} \gamma^{i-k} r_i \qquad (5.59)$$

with associated Bellman equation

$$x_k^T P x_k = y_k^T Q y_k + u_k^T R u_k + \gamma x_{k+1}^T P x_{k+1} \qquad (5.60)$$

This has an HJB (Riccati) equation equivalent of

$$0 = -P + \gamma \left(A^T P A - A^T P B (R/\gamma + B^T P B)^{-1} B^T P A \right) + C^T Q C \qquad (5.61)$$

and an optimal policy of

$$u_k = -(R/\gamma + B^T P B)^{-1} B^T P A x_k \tag{5.62}$$

The benefits of discounting are most clearly seen by examining value iteration. The Value Iteration Algorithm 5.3, which with discount has (5.24) modified as

$$x_k^T P^{j+1} x_k = y_k^T Q y_k + (u_k^j)^T R u_k^j + \gamma x_{k+1}^T P^j x_{k+1} \tag{5.63}$$

corresponds to the underlying Riccati difference equation

$$P_{j+1} = \gamma\left(A^T P_j A - A^T P_j B (R/\gamma + B^T P_j B)^{-1} B^T P_j A\right) + C^T Q C \tag{5.64}$$

where each iteration is decayed by the factor $\gamma < 1$. The effects of this can best be understood by considering the Lyapunov difference equation

$$P_{j+1} = \gamma A^T P_j A + C^T Q C, \quad P_0 \tag{5.65}$$

which has solution

$$P_j = \gamma^j (A^T)^j P_0 A^j + \sum_{i=0}^{j-1} \gamma^i (A^T)^i Q A^i \tag{5.66}$$

The effect of the discount factor is thus to decay the effects of the initial conditions.

In similar vein, the discount factor decays the effects of previous probing noises and improper initial conditions in the PI and VI algorithms. In fact, adding a discount factor is closely related to adding exponential data weighting in the Kalman Filter to remove the bias effects of unmodeled dynamics [55].

5.5 Simulations Results

5.5.1 Stable linear system and policy iteration

Consider the stable linear system with quadratic cost function

$$x_{k+1} = \begin{bmatrix} 1.1 & -0.3 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$$
$$y_k = \begin{bmatrix} 1 & -0.8 \end{bmatrix} x_k$$

where Q and R in the cost function are identity matrices of appropriate dimensions. The open-loop poles are $z_1 = 0.5$, and $z_2 = 0.6$. In order to verify the correctness of the proposed algorithm, the optimal value kernel matrix P is found by solving (5.15) to be

$$P = \begin{bmatrix} 1.0150 & -0.8150 \\ -0.8150 & 0.6552 \end{bmatrix}.$$

Now by using $\bar{P} \equiv \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}$ and (5.35) one has

$$\bar{P} = \begin{bmatrix} 1.0150 & -0.8440 & 1.1455 & -0.3165 \\ -0.8440 & 0.7918 & -1.0341 & 0.2969 \\ 1.1455 & -1.0341 & 1.3667 & -0.3878 \\ -0.3165 & 0.2969 & -0.3878 & 0.1113 \end{bmatrix}.$$

Since the system is stable, we use the output feedback PI Algorithm 5.4 implemented as in equations (5.52) and (5.53). PE was ensured by adding dithering noise to the control input, and a discount (forgetting) factor $\gamma = 0.2$ was added to diminish the dither bias effects. The observer index is $K = 2$ and N is selected equal to 2.

By applying dynamic output feedback control (5.53) the system remained stable and the parameters of the $\bar{P}$ converged to the optimal ones, in fact

$$\hat{\bar{P}} = \begin{bmatrix} 1.1340 & -0.8643 & 1.1571 & -0.3161 \\ -0.8643 & 0.7942 & -1.0348 & 0.2966 \\ 1.1571 & -1.0348 & 1.3609 & -0.3850 \\ -0.3161 & 0.2966 & -0.3850 & 0.1102 \end{bmatrix}.$$

In the example, Batch Least Squares was used to solve (5.52) at each step.

Figure 30 shows the convergence of $p_0 \in \mathbb{R}, p_u \in \mathbb{R}, p_y \in \mathbb{R}^{1\times 2}$ of $\bar{P}$ to the correct values. Figure 31 shows the evolution of the system states and their convergence to zero.

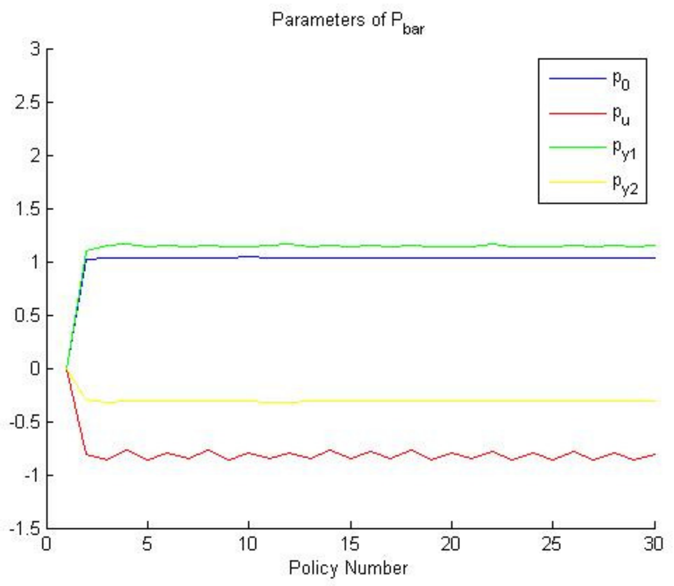

Figure 30. Convergence of p_0, p_u, p_y.

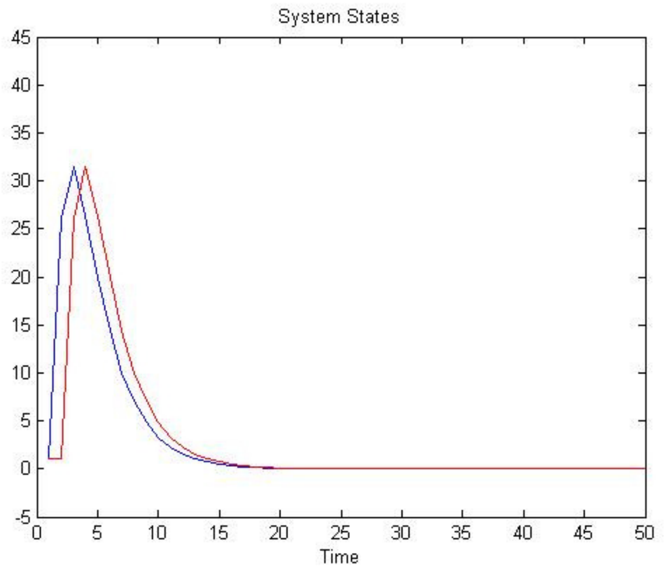

Figure 31. Evolution of the system states for the duration of the experiment.

5.5.2 Q-learning and output feedback ADP

The purpose of this example is to compare the performance of Q learning [17], [97] and Output Feedback ADP. Consider the stable linear system described before, and apply Q learning.

The Q function for this particular system is given by:

$$Q_h(x_k,u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T PA & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

$$\equiv \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} 4.9768 & -0.8615 & 2.8716 \\ -0.8615 & 1.2534 & -0.8447 \\ 2.8716 & -0.8447 & 3.8158 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

By comparing these two methods it is obvious that Q learning has a faster performance than the Output Feedback ADP since fewer parameters are being identified.

Figure 32 shows the convergence of the three parameters H_{11}, H_{12}, H_{13} of H matrix shown before to the correct values. Figure 33 shows the evolution of the system states.

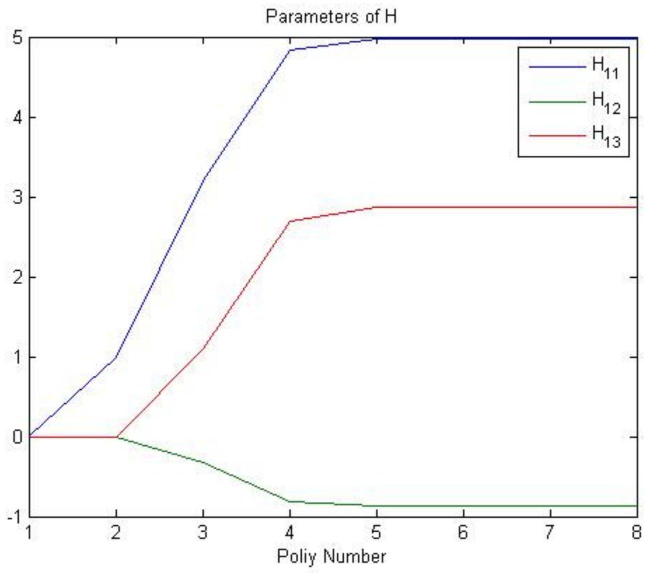

Figure 32. Convergence of H_{11}, H_{12}, H_{13}.

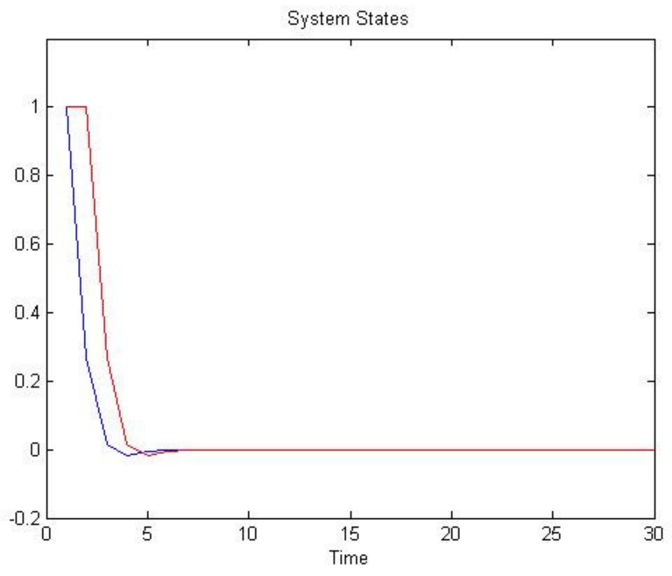

Figure 33. Evolution of the system states for the duration of the experiment.

5.5.3 Unstable linear system and Value Iteration

Consider the unstable linear system with quadratic cost function

$$x_{k+1} = \begin{bmatrix} 1.8 & -0.7700 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$$
$$y_k = \begin{bmatrix} 1 & -0.5 \end{bmatrix} x_k$$

where Q and R in the cost function are identity matrices of appropriate dimensions. The open-loop poles are $z_1 = 0.7$, and $z_2 = 1.1$ so the system is unstable. In order to verify the correctness of the proposed algorithm, the optimal value kernel matrix P is found by solving (5.15) to be

$$P = \begin{bmatrix} 1.3442 & -0.7078 \\ -0.7078 & 0.3756 \end{bmatrix}.$$

Now by using $\overline{P} \equiv \begin{bmatrix} M_u^T P M_u & M_u^T P M_y \\ M_y^T P M_u & M_y^T P M_y \end{bmatrix}$ and (5.35)one has

$$\bar{P} = \begin{bmatrix} 1.3442 & -0.7465 & 2.4582 & -1.1496 \\ -0.7465 & 0.4271 & -1.3717 & 0.6578 \\ 2.4582 & -1.3717 & 4.4990 & -2.1124 \\ -1.1496 & 0.6578 & -2.1124 & 1.0130 \end{bmatrix}.$$

Since the system is open-loop unstable, one must use value iteration, not policy iteration. The output feedback VI Algorithm 5.5 is implemented as in equations (5.54) and (5.55). PE was ensured by adding dithering noise to the control input, and a discount (forgetting) factor $\gamma = 0.2$ was used to diminish the dither bias effects. The observer index is $K = 2$ and N is selected equal to 2.

By applying dynamic output feedback control (5.55), the system is stabilized and the parameters of the $\bar{P}$ converged to the optimal ones, in fact

$$\hat{\bar{P}} = \begin{bmatrix} 1.3431 & -0.7504 & 2.4568 & -1.1493 \\ -0.7504 & 0.4301 & -1.3730 & 0.6591 \\ 2.4568 & -1.3730 & 4.4979 & -2.1120 \\ -1.1493 & 0.6591 & -2.1120 & 1.0134 \end{bmatrix}.$$

In the example, Batch Least Squares was used to solve (5.54) at each step.

Figure 34 shows the convergence of $p_0 \in \mathbb{R}, p_u \in \mathbb{R}, p_y \in \mathbb{R}^{1\times 2}$ of $\bar{P}$. Figure 35 shows the evolution of the system states, their boundedness despite the fact that the plant is initially unstable, and their convergence to zero.

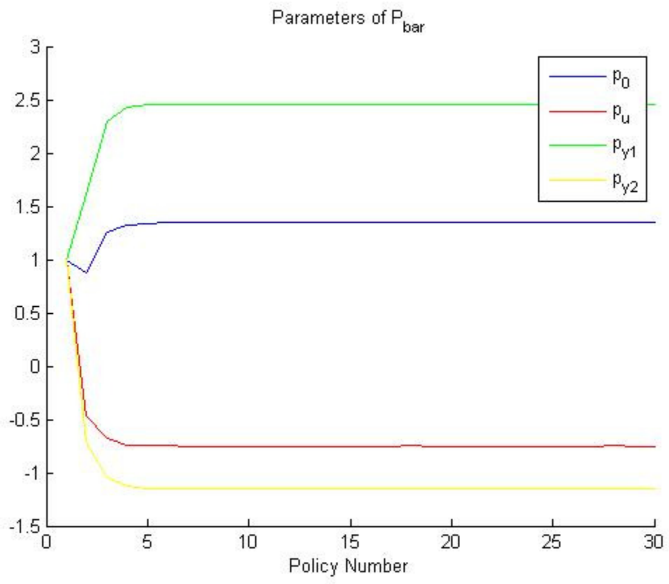

Figure 34. Convergence of p_0, p_u, p_y.

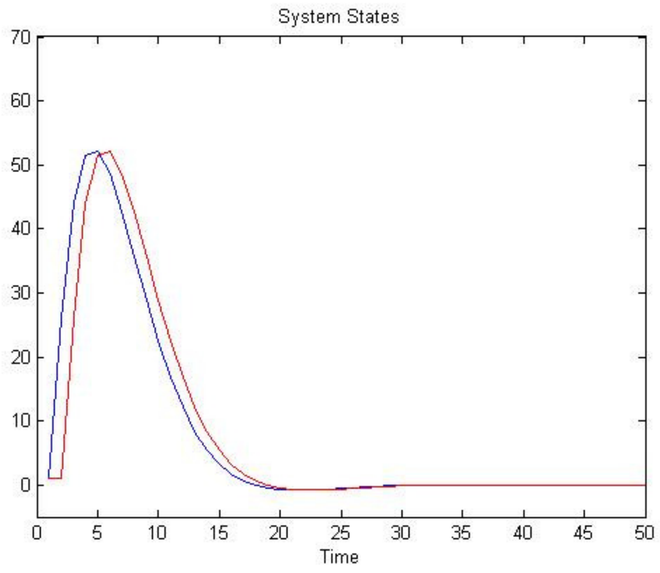

Figure 35. Evolution of the system states for the duration of the experiment.

5.6 Conclusion

In this chapter we have proposed the implementation of ADP using only measured input/output data from a dynamical system. This is known in control system theory as 'output feedback' as opposed to full state feedback, and corresponds to reinforcement learning for a class of partially observable Markov decision processes (POMDPs). Both policy iteration and value iteration algorithms are developed that require only output feedback. An added and surprising benefit is that, similar to Q learning, the system dynamics are not needed to implement these output feedback algorithms, so that they converge to the optimal controller for completely unknown systems. The system order must be known, and an upper bound on its ‘observability index’. The learned output feedback controller is given in the form of a polynomial ARMA controller that is equivalent to the optimal state variable feedback gain. This controller needs the addition of probing noise in order to be sufficiently rich. This probing noise adds some bias, and in order to avoid it, a discount factor is added in the cost.

CHAPTER 6

MULTI AGENT DIFFERENTIAL GRAPHICAL GAMES: POLICY ITERATION AND ONLINE ADAPTIVE LEARNING SOLUTION

6.1 Introduction

Distributed networks have received much attention in the last year because of their flexibility and computational performance. The ability to coordinate agents is important in many real-world tasks where it is necessary for agents to exchange information with each other. Synchronization behavior among agents is found in flocking of birds, schooling of fish, and other natural systems. Work has been done to develop cooperative control methods for consensus and synchronization ([24], [37], [66], [70], [72], [73], [74], [90]). See [65], [71] for surveys. Leaderless consensus results in all nodes converging to common value that cannot generally be controlled. We call this the cooperative regulator problem. On the other hand the problem of cooperative tracking requires that all nodes synchronize to a leader or control node [33], [56], [74], [96]. This has been called pinning control or control with a virtual leader. Consensus has been studied for systems on communication graphs with fixed or varying topologies and communication delays. This chapter brings together cooperative control, reinforcement learning, and game theory to solve multi-player differential games on communication graph topologies. There are four main contributions in this chapter. The first involves the formulation of a *graphical game* for dynamical systems networked by a communication graph. The dynamics and value function of each node depend only on the actions of that node and its neighbors. This graphical game allows for synchronization as well as Nash equilibrium solutions among neighbors. The second contribution is the derivation of coupled Riccati equations for solution of graphical games. The third contribution is a Policy Iteration algorithm for solution of graphical games that relies only on local information from neighbor nodes. It is shown that this algorithm converges to the best response policy of a node if its neighbors have fixed policies, and to the Nash solution if all nodes update their policies. The last contribution is the development of an online adaptive learning algorithm for computing the Nash equilibrium solutions of graphical games.

The chapter is organized as follows. Section 6.2 reviews synchronization in graphs and derives an error dynamics for each node that is influenced by its own actions and those of its neighbors. Section 6.3 introduces differential graphical games cooperative Nash equilibrium. Coupled Riccati equations are developed and stability and solution for Nash equilibrium are proven. Section 6.4 proposes a policy iteration algorithm for the solution of graphical games and gives proofs of convergence. Section 6.5 presents an online adaptive learning solution based on the structure of the policy iteration algorithm of Section 6.4. Finally Section 6.6 presents a simulation example that shows the effectiveness of the proposed algorithms in learning in real-time the solutions of graphical games.

6.2 Synchronization and node error dynamics

6.2.1 Graphs

Consider a graph $G=(V,\mathrm{E})$ with a nonempty finite set of *N* nodes $V=\{v_1,\cdots,v_N\}$ and a set of edges or arcs $\mathrm{E}\subseteq V\times V$. We assume the graph is simple, e.g. no repeated edges and $(v_i,v_i)\notin E,\forall i$ no self loops. Denote the connectivity matrix as $E=[e_{ij}]$ with $e_{ij}>0\ if\ (v_j,v_i)\in \mathrm{E}$ and $e_{ij}=0$ otherwise. Note $e_{ii}=0$. The set of neighbors of a node v_i is $N_i=\{v_j:(v_j,v_i)\in\mathrm{E}\}$, i.e. the set of nodes with arcs incoming to v_i. Define the in-degree matrix as a diagonal matrix $D=[d_i]$ with $d_i=\sum_{j\in N_i} e_{ij}$ the weighted in-degree of node i (i.e. i *-th* row sum of *E*). Define the graph Laplacian matrix as $L=D-E$, which has all row sums equal to zero.

A directed path is a sequence of nodes $v_0,v_1,\cdots,v_r$ such that $(v_i,v_{i+1})\in E,\ i\in\{0,1,\cdots,r-1\}$. A directed graph is strongly connected if there is a directed path from v_i to v_j for all distinct nodes $v_i, v_j \in V$. A (directed) tree is a connected digraph where every node except one, called the root, has in-degree equal to one. A graph is said to have a spanning tree if a subset of the edges forms a directed tree. A strongly connected digraph contains a spanning tree.

General directed graphs with fixed topology are considered in this chapter.

6.2.2 Synchronization and node error dynamics

Consider the *N* systems or agents distributed on communication graph *Gr* with node dynamics

$$\dot{x}_i = Ax_i + B_i u_i \tag{6.1}$$

where $x_i(t) \in \mathbb{R}^n$ is the state of node *i,* $u_i(t) \in \mathbb{R}^{m_i}$ its control input. Cooperative team objectives may be prescribed in terms of the *local neighborhood tracking error* $\delta_i \in \mathbb{R}^n$ [43]) as

$$\delta_i = \sum_{j \in N_i} e_{ij}(x_i - x_j) \quad + g_i(x_i - x_0) \tag{6.2}$$

The pinning gain $g_i \geq 0$ is nonzero for a small number of nodes *i* that are coupled directly to the leader or control node x_0, and $g_i > 0$ for at least one i [56] We refer to the nodes *i* for which $g_i \neq 0$ as the pinned or controlled nodes. Note that δ_i represents the information available to node *i* for state feedback purposes as dictated by the graph structure.

The state of the control or target node is $x_0(t) \in \mathbb{R}^n$ which satisfies the dynamics

$$\dot{x}_0 = Ax_0 \tag{6.3}$$

Note that this is in fact a *command generator* [49] and we seek to design a cooperative control command generator tracker. Note that the trajectory generator *A* may not be stable.

The **Synchronization control design problem** is to design local control protocols for all the nodes in *G* to synchronize to the state of the control node, i.e. one requires $x_i(t) \to x_0(t), \forall i$.

From (6.2), the overall error vector for network Gr is given by

$$\delta = \left(\left(L + G \right) \otimes I_n \right) \left(x - \underline{x}_0 \right) = \left(\left(L + G \right) \otimes I_n \right) \zeta \qquad (6.4)$$

where $\delta = \begin{bmatrix} \delta_1^T & \delta_2^T & \cdots & \delta_N^T \end{bmatrix}^T \in \mathbb{R}^{nN}$ and $\underline{x}_0 = \underline{I} x_0 \in \mathbb{R}^{nN}$, with $\underline{I} = \underline{1} \otimes I_n \in R^{nN \times n}$ and $\underline{1}$ the *N*-vector of ones. The Kronecker product is $\otimes$. $G \in R^{N \times N}$ is a diagonal matrix with diagonal entries equal to the pinning gains g_i. The (global) consensus or synchronization error (e.g. the disagreement vector in [66]) is

$$\zeta = \left(x - \underline{x}_0 \right) \in \mathbb{R}^{nN} \qquad (6.5)$$

The communication digraph is assumed to be strongly connected. Then, if $g_i \neq 0$ for at least one i, $\left(L + G \right)$ is nonsingular with all eigenvalues having positive real parts [43]. The next result therefore follows from (6.4) and the Cauchy Schwartz inequality and the properties of the Kronecker product [18].

Lemma 6.1. Let the graph be strongly connected and $G \neq 0$. Then the synchronization error is bounded by

$$\left\| \zeta \right\| \leq \left\| \delta \right\| / \underline{\sigma}(L + G) \qquad (6.6)$$

with $\underline{\sigma}(L + G)$ the minimum singular value of $\left(L + G \right)$, and $\delta(t) \equiv 0$ if and only if the nodes synchronize, that is

$$x(t) = \underline{I} x_0(t) \; (6.7)$$

■

Our objective now shall be to make small the local neighborhood tracking errors $\delta_i(t)$, which in view of Lemma 6.1 will guarantee synchronization.

To find the dynamics of the local neighborhood tracking error, write

$$\dot{\delta}_i = \sum_{j \in N_i} e_{ij} (\dot{x}_i - \dot{x}_j) \quad + g_i (\dot{x}_i - \dot{x}_0)$$

$$\dot{\delta}_i = \sum_{j \in N_i} e_{ij}(Ax_i + B_i u_i - (Ax_j + B_j u_j)) + g_i(Ax_i + B_i u_i - Ax_0)$$

$$\dot{\delta}_i = \sum_{j \in N_i} e_{ij}(Ax_i - Ax_j) + Ag_i(x_i - x_o) + \sum_{j \in N_i} e_{ij}(B_i u_i - B_j u_j) \; + \; g_i B_i u_i$$

$$\dot{\delta}_i = A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \tag{6.8}$$

with $\delta_i \in \mathbb{R}^n, u_i \in \mathbb{R}^{m_i}, \forall i$.

This is a dynamical system with multiple control inputs, from node *i* and all of its neighbors.

6.3 Cooperative multi-player games on graphs

We wish to achieve synchronization while simultaneously optimizing some performance specifications on the agents. To capture this, we intend to use the machinery of multi-player games [11].

6.3.1 Cooperative performance index

Define the local performance indices

$$J_i(\delta_i(0), u_i, u_{-i}) = \tfrac{1}{2}\int_0^\infty (\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii} u_i + \sum_{j \in N_i} u_j^T R_{ij} u_j)\, dt \equiv \tfrac{1}{2}\int_0^\infty L_i(\delta_i(t), u_i(t), u_{-i}(t))\, dt \tag{6.9}$$

where $u_{-i}(t)$ is the vector of the control inputs $\{u_j : j \in N_i\}$ of the neighbors of node *i*, and u_{-i} denotes $\{u_{-i}(t) : 0 \le t\}$. All weighting matrices are constant and symmetric with $Q_{ii} > 0, R_{ii} > 0, R_{ij} \ge 0$. Note that the *i*-th performance index includes only information about the inputs of node *i* and its neighbors.

For dynamics (6.8) with performance objectives (6.9), introduce the associated Hamiltonians

$$H_i(\delta_i, p_i, u_i, u_{-i}) \equiv p_i^T \left(A\delta_i + (d_i + g_i)B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j \right) + \tfrac{1}{2}\delta_i^T Q_{ii}\delta_i + \tfrac{1}{2}u_i^T R_{ii} u_i + \tfrac{1}{2}\sum_{j \in N_i} u_j^T R_{ij} u_j = 0 \tag{6.10}$$

where p_i is the co-state variable.

Necessary conditions [53] for a minimum of (6.9) are (6.1) and

$$-\dot{p}_i = \frac{\partial H_i}{\partial \delta_i} \equiv A^T p_i + Q_{ii}\delta_i \tag{6.11}$$

$$0 = \frac{\partial H_i}{\partial u_i} \Rightarrow u_i = -(d_i + g_i)R_{ii}^{-1}B_i^T p_i \tag{6.12}$$

6.3.2 Graphical games and cooperative Nash equilibrium

Interpreting the control inputs u_i, u_j as state dependent policies or strategies, the value function for node *i* corresponding to those policies is

$$V_i(\delta_i(t)) = \tfrac{1}{2}\int_t^\infty (\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii}u_i + \sum_{j\in N_i} u_j^T R_{ij}u_j)\, dt \tag{6.13}$$

Definition 6.1. Control policies $u_i, \quad \forall i$ are defined as admissible if u_i are continuous, $u_i(0) = 0$, u_i stabilize system (6.8) locally, and values (6.13) are finite.

When V_i is finite, using Leibniz' formula, a differential equivalent to this is given in terms of the Hamiltonian function by the Bellman equation

$$H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i, u_{-i}) \equiv \frac{\partial V_i}{\partial \delta_i}^T \left(A\delta_i + (d_i + g_i)B_i u_i - \sum_{j\in N_i} e_{ij}B_j u_j \right)$$
$$+\tfrac{1}{2}\delta_i^T Q_{ii}\delta_i + \tfrac{1}{2}u_i^T R_{ii}u_i + \tfrac{1}{2}\sum_{j\in N_i} u_j^T R_{ij}u_j = 0 \tag{6.14}$$

with boundary condition $V_i(0) = 0$. (The gradient is disabused here as a column vector.) That is, solution of equation (6.14) serves as an alternative to evaluating the infinite integral (6.13) for finding the value associated to the current feedback policies. It is shown in the Proof of Theorem 6.1 that (6.14) is a Lyapunov equation. According to (6.13) and (6.10) one equates $p_i = \partial V_i / \partial \delta_i$.

The control objective of agent *i* is to determine

$$V_i^*(\delta_i(t)) = \min_{u_i} \int_t^\infty \tfrac{1}{2}(\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii} u_i + \sum_{j\in N_i} u_j^T R_{ij} u_j)\, dt \tag{6.15}$$

which corresponds to Nash equilibrium.

Definition 6.2. [11] (Global Nash equilibrium) An *N-tuple* of policies $\{u_1^*, u_2^*, ..., u_N^*\}$ is said to constitute a global Nash equilibrium solution for an *N* player game if for all $i \in N$

$$J_i^* \triangleq J_i\,(u_1^*, u_2^*, ..., u_i^*, ..., u_N^*) \le J_i\,(u_1^*, u_2^*, ..., u_i, ..., u_N^*) \tag{6.16}$$

The *N-tuple* of game values $\{J_1^*, J_2^*, ..., J_N^*\}$ is known as a Nash equilibrium outcome of the *N*-player game.

The distributed multiplayer game with local dynamics (6.8) and local performance indices (6.9) should be contrasted with standard multiplayer games [2], [11] which have centralized dynamics

$$\dot{z} = Az + \sum_{i=1}^{N} B_i u_i \tag{6.17}$$

where $z \in \mathbb{R}^n$ is the state*,* $u_i(t) \in \mathbb{R}^{m_i}$ is the control input for every player, and where the performance index of each player depends on the control inputs of all other players. In the graphical games, by contrast, each node dynamics and performance index only depends on its own state, its control, and the controls of its immediate neighbors.

We want to study the distributed game on a graph defined by (6.15) with distributed dynamics (6.8). It is not clear in this scenario how global Nash equilibrium is to be achieved.

Graphical games have been studied in the computational intelligence community [40], [41], [80]. A (nondynamic) graphical game has been defined there as a tuple (G,U,v) with $G = (V,E)$ a graph with *N* nodes, action set $U = U_1 \times \cdots \times U_N$ with U_i the set of actions available to node *i*, and $v = \begin{bmatrix} v_1 & \cdots & v_N \end{bmatrix}^T$ a payoff vector, with $v_i(U_i, \{U_j : j \in N_i\}) \in R$ the payoff function of node *i*. It is important to note that *the payoff of node i only depends on its own action and* *those of its immediate neighbors*. The work on graphical games has focused on developing algorithms to find standard Nash equilibria for payoffs generally given in terms of matrices. Such algorithms are simplified in that they only have complexity on the order of the maximum node degree in the graph, not on the order of the number of players *N*. Undirected graphs are studied, and it is assumed that the graph is connected.

Our intention in this chapter is to provide online real-time adaptive methods for solving differential graphical games that are distributed in nature. That is, the control protocols and adaptive algorithms of each node are allowed to depend only information about itself and its neighbors. Moreover, as the game solution is being learned, all node dynamics are required to be stable, until finally all the nodes synchronize to the state of the control node.

The following notions are needed in the study of differential graphical games. Define $u_{-i} = \{u_j : j \in N_i\}$ as the set of policies of the neighbors of node *i*.

Definition 6.3. [80] Agent *i*'s *best response* to fixed policies u_{-i} of his neighbors is the policy u_i^* such that

$$J_i\,(u_i^*, u_{-i}) \le J_i\,(u_i, u_{-i}) \qquad (6.18)$$

for all policies u_i of agent *i*.

For centralized multi-agent games, where the dynamics is given by (6.17) and the performance index of each agent depends on the actions of all other agents, an alternative definition of Nash equilibrium is that each agent is in best response to all other agents. However, in Definition 6.3 each node *i* is only in best response to all his neighbors. If the graph is strongly connected and each node is in best response to all his neighbors then all nodes in the graph are in global Nash equilibrium

6.3.3 Stability and solution of graphical games

According to the results just established, the following assumptions are made.

Assumptions 6.1.

a. The graph is strongly connected and at least one pinning gain g_i is nonzero. Then $\left(L+G\right)$ is nonsingular.

The game is well-formed in the sense that:

b. $B_j \neq 0 \rightleftharpoons e_{ij} \in E$.

c. $R_{ij} \neq 0 \rightleftharpoons e_{ij} \in E$.

Employing the stationarity condition (6.12) [53] one obtains the control policies

$$u_i = u_i(V_i) \equiv -(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} \equiv -h_i(p_i) \tag{6.19}$$

Substituting into (6.14) yields the coupled cooperative game Hamilton-Jacobi (HJ) equations

$$\begin{aligned} &\frac{\partial V_i}{\partial \delta_i}^T A_i^c + \tfrac{1}{2}\delta_i^T Q_{ii}\delta_i + \tfrac{1}{2}(d_i+g_i)^2 \frac{\partial V_i}{\partial \delta_i}^T B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} \\ &+ \tfrac{1}{2}\sum_{j\in N_i} (d_j+g_j)^2 \frac{\partial V_j}{\partial \delta_j}^T B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^T \frac{\partial V_j}{\partial \delta_j} = 0, i \in N \end{aligned} \tag{6.20}$$

where the closed-loop matrix is

$$A_i^c = A\delta_i - (d_i+g_i)^2 B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \sum_{j\in N_i} e_{ij}(d_j+g_j) B_j R_{jj}^{-1} B_j^T \frac{\partial V_j}{\partial \delta_j}, i \in N \tag{6.21}$$

For a given V_i, define $u_i^* = u_i(V_i)$ as (6.19) given in terms of V_i. Then HJ equations (6.20) can be written as

$$H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i^*, u_{-i}^*) = 0 \tag{6.22}$$

There is one coupled HJ equation corresponding to each node, so solution of this *N*-player game problem is blocked by requiring a solution to *N* coupled partial differential equations. In the next section we show how to solve this *N*-player cooperative game online in a distributed fashion at each node, requiring only measurements from neighbor nodes, by using techniques from reinforcement learning.

For the global state δ given in (6.4) we can write the dynamics as

$$\dot{\delta} = (I_N \otimes A)\delta + (L+G) \otimes I_n diag(B_i)u \tag{6.23}$$

where u is the control given by

$$u = -diag(R_{ii}^{-1} B_i^T)\big((D+G) \otimes I_n p\big) \tag{6.24}$$

where $diag(.)$ denotes diagonal matrix of appropriate dimensions. Furthermore the global co-state dynamics are

$$-\dot{p} = \frac{\partial H}{\partial \delta} \equiv (I_N \otimes A)^T p + diag(Q_{ii})\delta \tag{6.25}$$

This is a set of coupled dynamic equations reminiscent of standard multi-player games [11] or single agent optimal control [53]. Therefore the solution can be written without any loss of generality as

$$p = \overline{P}\delta \tag{6.26}$$

for some matrix $\overline{P} > 0 \in \mathbb{R}^{nNxnN}$.

Lemma 6.2. HJ equations (6.20) are equivalent to the coupled Riccati equations

$$\delta^T \overline{P}^T \overline{A}_i \delta - \delta^T \overline{P}^T \overline{B}_i \overline{P}\delta + \tfrac{1}{2}\delta^T \overline{Q}_i \delta + \tfrac{1}{2}\delta^T \overline{P}^T \overline{R}_i \overline{P}\delta = 0 \tag{6.27}$$

or equivalently

$$(\overline{P}^T \overline{A}_{ic} + \overline{A}_{ic}{}^T \overline{P} + \overline{Q}_i + \overline{P}^T \overline{R}_i \overline{P}) = 0 \tag{6.28}$$

where $\overline{P}$ is defined by (6.26), and

$$\overline{A}_i = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & [A]^{ii} & \\ & & & 0 \end{bmatrix}, \overline{Q}_i = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & [Q_{ii}]^{ii} & \\ & & & 0 \end{bmatrix}$$

$$\bar{B}_i = \begin{bmatrix} 0 & & \\ & \left[(d_i+g_i)I_n\right]^{ii} & \left[-a_{ij}I_n\right]^{ij} \\ & & 0 \end{bmatrix} diag((d_i+g_i)B_iR_{ii}^{-1}B_i^T) \;\; \bar{A}_{ic} = \bar{A}_i - \bar{B}_i\bar{P}$$

$$\bar{R}_i = diag((d_i+g_i)B_iR_{ii}^{-1}) \begin{bmatrix} R_{i1} & & & & \\ & \ddots & & & \\ & & R_{ij} & & \\ & & & \ddots & \\ & & & & R_{ii} & \\ & & & & & R_{iN} \end{bmatrix} diag((d_i+g_i)R_{ii}^{-1}B_i^T)$$

where $[\;]^{ij}$ denotes the position of the element in the block matrix.

Proof:

Take (6.14) and write it with respect to the global state and co-state as

$$H_i \equiv \begin{bmatrix} \frac{\partial V_1}{\partial \delta_1} \\ \vdots \\ \vdots \\ \frac{\partial V_N}{\partial \delta_N} \end{bmatrix}^T \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & [A]^{ii} & \\ & & & 0 \end{bmatrix} \delta + \begin{bmatrix} \frac{\partial V_1}{\partial \delta_1} \\ \vdots \\ \vdots \\ \frac{\partial V_N}{\partial \delta_N} \end{bmatrix}^T \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & 0 & \vdots & \vdots \\ \vdots & \vdots & \left[(d_i+g_i)I_n\right]^{ii} & \left[-a_{ij}I_n\right]^{ij} \\ 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 & & & \\ & \ddots & & \\ & & B_i & \\ & & & B_N \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_i \\ u_N \end{bmatrix}$$

$$+\tfrac{1}{2}\delta^T \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & [Q_{ii}]^{ii} & \\ & & & 0 \end{bmatrix} \delta + \frac{1}{2} \begin{bmatrix} u_1 \\ \vdots \\ u_i \\ u_N \end{bmatrix}^T \begin{bmatrix} R_{i1} & & & \\ & R_{ij} & & \\ & & R_{ii} & \\ & & & R_{iN} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_i \\ u_N \end{bmatrix} = 0 \qquad (6.29)$$

By definition of the co-state one has

$$p \equiv \begin{bmatrix} \frac{\partial V_1}{\partial \delta_1} \\ \vdots \\ \vdots \\ \frac{\partial V_N}{\partial \delta_N} \end{bmatrix} = \bar{P}\delta \qquad (6.30)$$

From the control policies (6.19), (6.29) becomes (6.27), which can be written in closed-loop form as (6.28).

■

Theorem 6.1. Stability and Solution for Cooperative Nash Equilibrium.

Let $V_i > 0 \in C^1$, $i \in N$ be smooth solutions to HJ equations (6.20) and control policies u_i^*, $i \in N$ be given by (6.19) in terms of these solutions V_i. Then

a. Systems (6.8) are asymptotically stable.

b. u_i^*, u_{-i}^* are in cooperative Nash equilibrium and the corresponding game values are

$$J_i^*(\delta_i(0)) = V_i \ , i \in N \tag{6.31}$$

Proof:

a. If $V_i > 0$ satisfies (6.20) then it also satisfies (6.14). Take the time derivative to obtain

$$\dot{V}_i = \frac{\partial V_i}{\partial \delta_i}^T \dot{\delta}_i = \frac{\partial V_i}{\partial \delta_i}^T \left(A\delta_i + (d_i + g_i)B_i u_i - \sum_{j\in N_i} e_{ij} B_j u_j \right) = -\tfrac{1}{2}\left(\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii} u_i + \sum_{j\in N_i} u_j^T R_{ij} u_j \right) \tag{6.32}$$

which is negative definite since $Q_{ii} > 0$. Therefore V_i is a Lyapunov function for δ_i and systems (6.8) are asymptotically stable.

b. According to part *a*, $\delta_i(t) \to 0$ for the selected control policies. For any smooth functions $V_i(\delta_i), i \in N$, such that $V_i(0) = 0$, setting $V_i(\delta_i(\infty)) = 0$ one can write (6.9) as

$$J_i(\delta_i(0), u_i, u_{-i}) = \tfrac{1}{2}\int_0^\infty (\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii} u_i + \sum_{j\in N_i} u_j^T R_{ij} u_j)\, dt + V_i(\delta_i(0)) + \int_0^\infty \dot{V}_i dt$$

or

$$J_i(\delta_i(0), u_i, u_{-i}) = \tfrac{1}{2}\int_0^\infty (\delta_i^T Q_{ii}\delta_i + u_i^T R_{ii} u_i + \sum_{j\in N_i} u_j^T R_{ij} u_j)\, dt + V_i(\delta_i(0))$$
$$+\int_0^\infty \frac{\partial V_i}{\partial \delta_i}^T (A\delta_i + (d_i + g_i)B_i u_i - \sum_{j\in N_i} e_{ij} B_j u_j) dt$$

Now let V_i satisfy (6.20) and u_i^*, u_{-i}^* be the optimal controls given by (6.19). By completing the squares one has

$$J_i(\delta_i(0), u_i, u_{-i}) = V_i\ (\delta_i(0)) + \int_0^\infty (\tfrac{1}{2}\sum_{j\in N_i}(u_j - u_j^*)^T R_{ij}(u_j - u_j^*) + \tfrac{1}{2}(u_i - u_i^*)^T R_{ii}(u_i - u_i^*)$$
$$-\frac{\partial V_i}{\partial \delta_i}^T \sum_{j\in N_i} e_{ij} B_j (u_j - u_j^*) + \sum_{j\in N_i} u_j^{*T} R_{ij}(u_j - u_j^*))dt$$

At the equilibrium point $u_i = u_i^*$ and $u_j = u_j^*$ so

$$J_i^*(\delta_i(0), u_i^*, u_{-i}^*) = V_i\ (\delta_i(0))$$

Define

$$J_i(u_i, u_{-i}^*) = V_i\ (\delta_i(0)) + \tfrac{1}{2}\int_0^\infty (u_i - u_i^*)^T R_{ii}(u_i - u_i^*)dt$$

and $J_i^* = V_i\ (\delta_i(0))$. Then clearly J_i^* and $J_i(u_i, u_{-i}^*)$ satisfy (6.16).

■

6.4 Policy iteration solution for cooperative multi-player games

6.4.1 Best response

Theorem 6.1 reveals that the systems are in cooperative Nash equilibrium if, for all $i \in N$ node *i* selects his best response policy to his neighbors policies and the graph is strongly connected. Define the best response HJ equation as the Bellman equation (6.14) with control $u_i = u_i^*$ given by (6.19) and arbitrary policies $u_{-i} = \{u_j : j \in N_i\}$

$$0 = H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}, u_i^*, u_{-i}) \equiv \frac{\partial V_i}{\partial \delta_i}^T A_i^c + \tfrac{1}{2}\delta_i^T Q_{ii}\delta_i + \tfrac{1}{2}(d_i + g_i)^2 \frac{\partial V_i}{\partial \delta_i}^T B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \tfrac{1}{2}\sum_{j\in N_i} u_j^T R_{ij} u_j \qquad (6.33)$$

where the closed-loop matrix is

$$A_i^c = A\delta_i - (d_i + g_i)^2 B_i R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} - \sum_{j\in N_i} e_{ij} B_j u_j \qquad (6.34)$$

Theorem 6.2. Solution for Best Response Policy

Given fixed neighbor policies $u_{-i}=\{u_j : j\in N_i\}$, assume there is an admissible policy u_i. Let $V_i>0\in C^1$ be a smooth solution to the best response HJ equation (6.33) and let control policy u_i^* be given by (6.19) in terms of this solution V_i. Then

a. System (6.8) is asymptotically stable.

b. u_i^* is the best response to the fixed policies u_{-i} of its neighbors.

Proof:

a. $V_i>0$ satisfies (6.33). Proof follows Theorem 6.1, part a.

b. According to part *a*, $\delta_i(t)\to 0$ for the selected control policies. For any smooth functions $V_i(\delta_i), i\in N$, such that $V_i(0)=0$, setting $V_i(\delta_i(\infty))=0$ one can write (6.9) as

$$J_i(\delta_i(0),u_i,u_{-i})=\tfrac{1}{2}\int_0^\infty(\delta_i^T Q_{ii}\delta_i+u_i^T R_{ii}u_i+\sum_{j\in N_i}u_j^T R_{ij}u_j)\,dt$$
$$+V_i(\delta_i(0))+\int_0^\infty\frac{\partial V_i}{\partial\delta_i}^T(A\delta_i+(d_i+g_i)B_iu_i-\sum_{j\in N_i}e_{ij}B_ju_j)dt$$

Now let V_i satisfy (6.33), u_i^* be the optimal controls given by (6.19) and u_{-i} be arbitrary policies. By completing the squares one has

$$J_i(\delta_i(0),u_i,u_{-i})=V_i\;(\delta_i(0))+\int_0^\infty\tfrac{1}{2}(u_i-u_i^*)^T R_{ii}(u_i-u_i^*)dt$$

The agents are in best response to fixed policies u_{-i} when $u_i=u_i^*$ so

$$J_i(\delta_i(0),u_i^*,u_{-i})=V_i\;(\delta_i(0))$$

Then clearly $J_i(\delta_i(0),u_i,u_{-i})$ and $J_i(\delta_i(0),u_i^*,u_{-i})$ satisfy (6.18).

■

6.4.2 Policy Iteration for solution of graphical games

The following algorithm for the *N*-player distributed games is motivated by the structure of policy iteration algorithms in reinforcement learning [15], [86], which rely on repeated policy evaluation (e.g. solution of (6.14)) and policy improvement (solution of (6.19)). These two steps are repeated until the policy improvement step no longer changes the present policy. If the algorithm converges for every i, then it converges to the solution to HJ equations (6.20), and hence provides the distributed Nash equilibrium. One must note that the costs can be evaluated only in the case of admissible control policies, admissibility being a condition for the control policy which initializes the algorithm.

Algorithm 6.1. Policy Iteration (PI) Solution for *N*-player distributed games.

Step 0: Start with admissible initial policies $u_i^0, \forall i$.

Step 1: (Policy Evaluation) Solve for V_i^k using (6.14)

$$H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}^k, u_i^k, u_{-i}^k) = 0, \forall i = 1, \ldots, N \tag{6.35}$$

Step 2: (Policy Improvement) Update the *N*-tuple of control policies using

$$u_i^{k+1} = \arg\min_{u_i} H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}^k, u_i, u_{-i}^k), \forall i = 1, \ldots, N$$

which explicitly is

$$u_i^{k+1} = -(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i}^k, \forall i = 1, \ldots, N. \tag{6.36}$$

Go to step 1.

On convergence End

■

The following two theorems prove convergence of the policy iteration algorithm for distributed games for two different cases. The two cases considered are the following, i) *only* agent *i* updates its policy and ii) all the agents update their policies.

Theorem 6.3. Convergence of Policy Iteration algorithm when only *i^th^* agent updates its policy and all players u_{-i} in the neighborhood do not change. Given fixed neighbors policies u_{-i}, assume there exists an admissible policy u_i. Assume that agent *i* performs Algorithm 6.1 and its neighbors do not update their control policies. Then the algorithm converges to the best response u_i to policies u_{-i} of the neighbors and to the solution V_i to the best response HJ equation (6.33).

Proof:

It is clear that

$$H_i^o(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_{-i}^k) \equiv \min_{u_i} H_i(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_i^k, u_{-i}^k) = H_i(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_i^{k+1}, u_{-i}^k) \qquad (6.37)$$

Let $H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}^k, u_i^k, u_{-i}^k) = 0$ from (6.35) then according to (6.37) it is clear that

$$H_i^o(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_{-i}^k) \le 0 \qquad (6.38)$$

Using the next control policy u_i^{k+1} and the current policies u_{-i}^k one has the orbital derivative [47]

$$\dot{V}_i^k = H_i(\delta_i, \frac{\partial V_i^k}{\partial \delta_i}, u_i^{k+1}, u_{-i}^k) - L_i(\delta_i, u_i^{k+1}, u_{-i}^k)$$

From (6.37) and (6.38) one has

$$\dot{V}_i^k = H_i^0(\delta_i, \frac{\partial V_i}{\partial \delta_i}^k, u_{-i}^k) - L_i(\delta_i, u_i^{k+1}, u_{-i}^k) \le -L_i(\delta_i, u_i^{k+1}, u_{-i}^k) \qquad (6.39)$$

Because only agent *i* update its control it is true that $u_{-i}^{k+1} = u_{-i}^k$ and

$$H_i(\delta_i, \frac{\partial V_i^{k+1}}{\partial \delta_i}, u_i^{k+1}, u_{-i}^k) = 0 .$$

But since $\dot{V}_i^{k+1} = -L_i(\delta_i, u_i^{k+1}, u_{-i}^{k+1})$, from (6.39) one has

$$\dot{V}_i^k = H_i^0(\delta_i, \frac{\partial V_i}{\partial \delta_i}^k, u_{-i}^k) - L_i(\delta_i, u_i^{k+1}, u_{-i}^k) \le -L_i(\delta_i, u_i^{k+1}, u_{-i}^k) = \dot{V}_i^{k+1} \qquad (6.40)$$

So that $\dot{V}_i^k \le \dot{V}_i^{k+1}$ and by integration it follows that

$$V_i^{k+1} \le V_i^k \qquad (6.41)$$

Since $V_i^* \le V_i^k$, the algorithm converges, to V_i^*, to the best response HJ equation (6.33).

■

The next result concerns the case where all nodes update their policies at each step of the algorithm. Define the relative control weighting as $\rho_{ij} = \bar{\sigma}(R_{jj}^{-1}R_{ij})$, where $\bar{\sigma}(R_{jj}^{-1}R_{ij})$ is the maximum singular value of $R_{jj}^{-1}R_{ij}$.

Theorem 6.4. Convergence of Policy Iteration algorithm when all agents update their policies. Assume all nodes *i* update their policies at each iteration of PI. Then for small enough edge weights e_{ij} and ρ_{ij}, u_i converges to the global Nash equilibrium and for all i**,** and the values converge to the optimal game values $V_i^k \to V_i^*$.

Proof:

It is clear that

$$H_i(\delta_i, \frac{\partial V_i}{\partial \delta_i}^{k+1}, u_i^{k+1}, u_{-i}^{k+1}) \equiv H_i^0(\delta_i, \frac{\partial V_i}{\partial \delta_i}^{k+1}, u_{-i}^k)$$

$$+\tfrac{1}{2}\sum_{j\in N_i}(u_j^{k+1}-u_j^k)^T R_{ij}(u_j^{k+1}-u_j^k) + \sum_{j\in N_i} u_j^{kT} R_{ij}(u_j^{k+1}-u_j^k) + \frac{\partial V_i}{\partial \delta_i}^{k+1T} \sum_{j\in N_i} e_{ij}B_j(u_j^k - u_j^{k+1})$$

and so

$$\dot{V}_i^{k+1} = -L_i(\delta_i, u_i^{k+1}, u_{-i}^{k+1}) = -L_i(\delta_i, u_i^{k+1}, u_{-i}^k) + \tfrac{1}{2}\sum_{j\in N_i}(u_j^{k+1}-u_j^k)^T R_{ij}(u_j^{k+1}-u_j^k)$$

$$+\frac{\partial V_i}{\partial \delta_i}^{k+1T} \sum_{j\in N_i} e_{ij}B_j(u_j^k - u_j^{k+1}) + \sum_{j\in N_i} u_j^{kT} R_{ij}(u_j^{k+1}-u_j^k)$$

Therefore,

$$\dot{V}_i^k \le \dot{V}_i^{k+1} - \tfrac{1}{2}\sum_{j\in N_i}(u_j^{k+1}-u_j^k)^T R_{ij}(u_j^{k+1}-u_j^k) + \frac{\partial V_i}{\partial \delta_i}^{k+1T}\sum_{j\in N_i} e_{ij}B_j(u_j^{k+1}-u_j^k) - \sum_{j\in N_i} u_j^{kT}R_{ij}(u_j^{k+1}-u_j^k)$$

A sufficient condition for $\dot{V}_i^k \le \dot{V}_i^{k+1}$ is

$$\tfrac{1}{2}\Delta u_j^T R_{ij}\Delta u_j - (p_i^{k+1})^T e_{ij}B_j\Delta u_j + u_j^{kT}R_{ij}\Delta u_j > 0$$

or

$$\tfrac{1}{2}\Delta u_j^T R_{ij}\Delta u_j - e_{ij}(p_i^{k+1})^T B_j\Delta u_j - (d_j+g_j)(p_j^{k-1})B_j^T R_{jj}^{-1}R_{ij}\Delta u_j > 0,$$

$$\tfrac{1}{2}\underline{\sigma}(R_{ij})\left\|\Delta u_j\right\| > e_{ij}\left\|p_i^{k+1}\right\|\cdot\left\|B_j\right\| + (d_j+g_j)\rho_{ij}\left\|p_j^{k-1}\right\|\cdot\left\|B_j\right\|$$

where $\Delta u_j = (u_j^{k+1}-u_j^k)$, p_i the co-state and $\underline{\sigma}(R_{ij})$ is the minimum singular value of R_{ij}.

This holds if $e_{ij}=0, \rho_{ij}=0$. By continuity, it holds for small values of e_{ij}, ρ_{ij}.

■

This proof indicates that for the PI algorithm to converge, the neighbors' controls should not unduly influence the *i*-th node dynamics (6.8), and the *j*-th node should weight its own control u_j in its performance index J_j relatively more than node *i* weights u_j in J_i. These requirements are consistent with selecting the weighting matrices to obtain proper performance in the simulation examples.

An alternative condition for convergence in Theorem 6.4 is that the norm $\left\|B_j\right\|$ should be small. This is similar to the case of weakly coupled dynamics in multi-player games in [11].

6.5 Online solution of multi-agent cooperative games using neural networks

In this section an online algorithm for solving cooperative Hamilton-Jacobi equations (6.20) based on Chapter 5 is presented. This algorithm uses the structure in the PI Algorithm 6.1 to develop an actor/critic architecture for approximate online solution of (6.20). The algorithm uses two approximator structures at each node, which are taken here as neural networks (NN) [4], [15], [86]. One critic NN is used at each node for value function approximation, and one actor NN at each node to approximate the control policy (6.36). The critic NN seeks to solve Bellman equation (6.35). We give tuning laws for the actor NN and the critic NN such that equations (6.35) and (6.36) are solved simultaneously online for each node. Then, the solutions to the coupled HJ equations (6.20) are determined. Though these coupled HJ equations are difficult to solve, and may not even have analytic solutions, we show how to tune the NN so that the approximate solutions are learned online. The next assumption is made.

Assumption 6.2. For each admissible control policy the nonlinear Bellman equations (6.14), (6.35) have smooth solutions $V_i \geq 0$.

To solve the Bellman equations (6.35), approximation is required of both the value functions V_i and their gradients $\partial V_i / \partial \delta_i$. This requires approximation in Sobolev space [4].

6.5.1 Critic neural network

According to the Weierstrass higher-order approximation theorem [4] there are NN weights W_i such that the smooth value functions V_i are approximated using a critic NN as

$$V_i(\delta_i) = W_i^T \phi_i + \varepsilon_i \tag{6.42}$$

with $\phi_i : \mathbb{R}^n \to \mathbb{R}^h$ the critic NN activation function vectors and *h* the number of neurons in the critic NN hidden layer. The NN approximation error ε_i converges to zero uniformly as $h \to \infty$.

Assuming current weight estimates $\hat{W}_i$, the outputs of the critic NN are given by

$$\hat{V}_i = \hat{W}_i^T \phi_i \tag{6.43}$$

Then, the Bellman equation (6.35) can be approximated at each step *k* as

$$\begin{aligned} H_i(\delta_i, \hat{W}_i\ , u_i, u_{-i}) &= \delta_i^T Q_{ii} \delta_i + u_i^T R_{ii} u_i + \sum_{j \in N_i} u_j^T R_{ij} u_j \\ &+ \hat{W}_i^T \frac{\partial \varphi_i}{\partial \delta_i} (A\delta_i + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_{ij} B_j u_j) = e_{H_i} \end{aligned} \tag{6.44}$$

It is desired to select $\hat{W}_i$ to minimize the square residual error

$$E_1 = \tfrac{1}{2} e_{H_i}^T e_{H_i} \tag{6.45}$$

Then $\hat{W}_i \to W_i$ which solves (6.44) in a least-squares sense and e_{H_i} becomes small. Theorem 6.5 gives a tuning law for the critic weights that achieves this.

6.5.2 Actor neural network and online solution

Define the control policy in the form of an action neural network which computes the control input (6.36) in the structured form

$$\hat{u}_{i+N} = -\tfrac{1}{2}(d_i + g_i)R_{ii}^{-1}B_i^T \frac{\partial \varphi_i}{\partial \delta_i}^T \hat{W}_{i+N} \tag{6.46}$$

where $\hat{W}_{i+N}$ denote the current estimated values of the ideal NN weights W_i. Define the actor NN estimation errors as $\tilde{W}_i = W_i - \hat{W}_i$ and $\tilde{W}_{i+N} = W_i - \hat{W}_{i+N}$.

The next results show how to tune the critic NN and actor NN in real time at each node so that equations (6.35) and (6.36) are simultaneously solved, while closed-loop system stability is also guaranteed. Simultaneous solution of (6.35) and (6.36) guarantees that the coupled HJ equations (6.20) are solved for each node *i*. System (6.8) is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset R^n$ so that for all $x_0 \in S$ there exists a bound *B* and a time $T(B, x_0)$ such that $\|x(t)\| \le B$ for all $t \ge t_0 + T$.

Select the tuning law for the *i*th critic NN as

$$\begin{aligned}\dot{\hat{W}}_i = -a_i \frac{\partial E_1}{\partial \hat{W}_i} = -a_i \frac{\sigma_i}{(1+\sigma_i^{\mathrm{T}}\sigma_i)^2}[\sigma_i^{\mathrm{T}}\hat{W}_i + \delta_i^T Q_{ii}\delta_i + \tfrac{1}{4}\hat{W}_{i+N}^T \bar{D}_i \hat{W}_{i+N} \\ + \tfrac{1}{4}\sum_{j\in N_i}(d_j+g_j)^2 \hat{W}_{j+N}^T \frac{\partial \varphi_j}{\partial \delta_j} B_j R_{jj}^{-T} R_{ij} R_{jj}^{-1} B_j^T \frac{\partial \varphi_j}{\partial \delta_j}^T \hat{W}_{j+N}]\end{aligned} \tag{6.47}$$

where $\sigma_{i+N} = \dfrac{\partial \varphi_i}{\partial \delta_i}(A\delta_i + (d_i+g_i)B_i\hat{u}_{i+N} - \sum_{j\in N_i} e_{ij}B_j\hat{u}_{j+N})$

and the tuning law for the *i*th actor NN as

$$\begin{aligned}\dot{\hat{W}}_{i+N} = -a_{i+N}\{(F_{i+1}\hat{W}_{i+N} - F_i\bar{\sigma}_{i+N}^T\hat{W}_i) - \frac{1}{4}\bar{D}_i\hat{W}_{i+N}\frac{\bar{\sigma}_{i+N}}{m_{si}}^T\hat{W}_i \\ -\frac{1}{4}\hat{W}_{i+N}{}^T\sum_{\substack{j\in N_i \\ j\neq i}}(d_j+g_j)^2\hat{W}_j\frac{\bar{\sigma}_{i+N}}{m_{s_i}}^T\frac{\partial\varphi_j}{\partial\delta_j}B_jR_{jj}^{-T}R_{ij}R_{jj}^{-1}B_j{}^T\frac{\partial\varphi_j}{\partial\delta_j}^T\}\end{aligned} \qquad (6.48)$$

where $\bar{D}_i(x)\equiv\frac{\partial\varphi_i}{\partial\delta_i}B_iR_{ii}{}^{-1}B_i{}^T\frac{\partial\varphi_i}{\partial\delta_i}^T$, $m_{s_i}\equiv(\sigma_{i+N}{}^T\sigma_{i+N}+1)$, $a_i>0,\ldots a_{i+N}>0$ and $F_i>0,\ldots F_{i+N}>0$ are tuning parameters.

Theorem 6.5. Online Cooperative Games.

Let the error dynamics be given by (6.8), and consider the cooperative game formulation in (6.15). Let the critic NN at each node be given by (6.43) and the control input be given for each node by actor NN (6.46). Let the tuning law for the *i*th critic NN be provided by (6.47) and the tuning law for the *i*th actor NN be provided by (6.48) and assume $\bar{\sigma}_{i+N}=\sigma_{i+N}/(\sigma_{i+N}{}^T\sigma_{i+N}+1)$ is persistently exciting and $Q_{ii}>0$. Then the closed-loop system state, the critic NN errors $\tilde{W}_i$, and the actor NN errors $\tilde{W}_{i+N}$ are uniformly ultimately bounded.

Proof: The proof is similar to the previous chapter.

■

Remark 6.1. Theorem 6.5 provides algorithms for tuning the actor/critic networks of the *N* agents at the same time to guarantee stability and make the system errors $\delta_i(t)$ small and the NN approximation errors bounded. Small errors guarantee synchronization of all the node trajectories.

Remark 6.2. Persistence of excitation is needed for proper identification of the value functions by the critic NNs, and nonstandard tuning algorithms are required for the actor NNs to guarantee stability. It is important to notice that the actor tuning law of every agent needs information of the critic weights of all his neighbors.

Remark 6.3. NN usage suggests starting with random, nonzero control NN weights in (6.46) in order to converge to the coupled Riccatis solution. However, it should be noted that convergence is more sensitive to the persistence of excitation in the control inputs than to the

NN weight initialization. If the proper persistence of excitation is not selected, the control weights may not converge to the correct values.

Remark 6.4. The issue of which inputs to use for the critic and actor NNs needs to be addressed. According to the dynamics (6.8), the value functions (6.13), and the control inputs (6.19), the NN inputs at node i should consist of its own state, the states of its neighbors, and the co-states of its neighbors. However, in view of (6.26) the co-states are functions of the states. In view of the approximation capabilities of NN, it is found in simulations that it is suitable to use as the NN inputs at node i its own state and the states of its neighbors.

The next result shows that the tuning laws given in Theorem 6.5 guarantee approximate solution to the coupled HJ equations (6.20) and convergence to the Nash equilibrium.

Theorem 6.6. Convergence to Cooperative Nash Equilibrium.

Suppose the hypotheses of Theorem 6.5 hold. Then:

a. $H_i(\delta_i, \hat{W}_i\ , \hat{u}_i, \hat{u}_{-i}), \quad \forall i \in N$ are uniformly ultimately bounded, where

$$\hat{u}_i = -\tfrac{1}{2}(d_i + g_i) R_{ii}^{-1} B_i^T \frac{\partial \varphi_i}{\partial \delta_i}^T \hat{W}_i$$

That is, $\hat{W}_i$ converge to the approximate cooperative coupled HJ-solution.

b. $\hat{u}_{i+N}$ converge to the approximate cooperative Nash equilibrium (Definition 6.2) for every i .

Proof: The proof is similar to the one in chapter 5 but is done only with respect to the neighbors (local information) of each agent and not with respect to all agents.

Consider the weights $\hat{W}_i, \hat{W}_{i+N}$ to be UUB as proved in Theorem 6.5.

a. The approximate coupled HJ equations are $H_i(\delta_i, \hat{W}_i\ , \hat{u}_i, \hat{u}_{-i}), \quad \forall i \in N$.

$$H_i(\delta_i, \hat{W}_i\ , \hat{u}_i, \hat{u}_{-i}) \equiv H_i(\delta_i, \hat{W}_i\ \hat{W}_{-i}) = \delta_i^T Q_{ii} \delta_i + \hat{W}_i^T \frac{\partial \varphi_i}{\partial \delta_i} A \delta_i$$
$$-\tfrac{1}{4}(d_i + g_i)^2 \hat{W}_i^T \frac{\partial \varphi_i}{\partial \delta_i} B_i R_{ii}^{-1} B_i^T \frac{\partial \varphi_i}{\partial \delta_i}^T \hat{W}_i$$

$$+\tfrac{1}{4}\sum_{j\in N_i}(d_j+g_j)^2\hat{W}_j^T\frac{\partial\varphi_j}{\partial\delta_j}B_jR_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\hat{W}_j+\tfrac{1}{2}\hat{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}\sum_{j\in N_i}e_{ij}B_jR_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\hat{W}_j-\varepsilon_{HJ_i}$$

where $\varepsilon_{HJ_i}, \forall i$ are the residual errors due to approximation.

After adding zero we have

$$H_i(\delta_i,\hat{W}_i\,\hat{W}_{-i})=-\tilde{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}A\delta_i-\tfrac{1}{4}(d_i+g_i)^2\tilde{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}B_iR_{ii}^{-1}B_i^T\frac{\partial\varphi_i}{\partial\delta_i}^T\tilde{W}_i$$

$$+\tfrac{1}{2}(d_i+g_i)^2W_i^T\frac{\partial\varphi_i}{\partial\delta_i}B_iR_{ii}^{-1}B_i^T\frac{\partial\varphi_i}{\partial\delta_i}^TW_i+\tfrac{1}{2}(d_i+g_i)^2W_i^T\frac{\partial\varphi_i}{\partial\delta_i}B_iR_{ii}^{-1}B_i^T\frac{\partial\varphi_i}{\partial\delta_i}^T\hat{W}_i$$

$$-\tfrac{1}{4}\sum_{j\in N_i}(d_j+g_j)^2\tilde{W}_j^T\frac{\partial\varphi_j}{\partial\delta_j}B_jR_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\tilde{W}_j+\tfrac{1}{2}\sum_{j\in N_i}(d_j+g_j)^2W_j^T\frac{\partial\varphi_j}{\partial\delta_j}B_jR_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^TW_j$$

$$+\tfrac{1}{2}\sum_{j\in N_i}(d_j+g_j)^2W_j^T\frac{\partial\varphi_j}{\partial\delta_j}B_jR_{jj}^{-1}R_{ij}R_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\hat{W}_j+\hat{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}\sum_{j\in N_i}e_{ij}B_jR_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\hat{W}_j-\varepsilon_{HJ_i}$$

$$-\tfrac{1}{2}\tilde{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}\sum_{j\in N_i}e_{ij}B_jR_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\tilde{W}_j-\tfrac{1}{2}W_i^T\frac{\partial\varphi_i}{\partial\delta_i}\sum_{j\in N_i}e_{ij}B_jR_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^T\hat{W}_j$$

$$-\tfrac{1}{2}\hat{W}_i^T\frac{\partial\varphi_i}{\partial\delta_i}\sum_{j\in N_i}e_{ij}B_jR_{jj}^{-1}B_j^T\frac{\partial\varphi_j}{\partial\delta_j}^TW_j \tag{6.49}$$

But

$$\hat{W}_i=-\tilde{W}_i+W_i,\quad \forall i. \tag{6.50}$$

After taking norms in (6.50) and letting $\|W_i\|<W_{i\max}$ one has

$$\|\hat{W}_i\|=\|-\tilde{W}_i+W_i\|\le\|\tilde{W}_i\|+\|W_i\|\le\|\tilde{W}_i\|+W_{i\max}$$

Now (6.49) with $\sup\|\varepsilon_{HJ_i}\|<\bar{\varepsilon}_i$ becomes

$$\left\|H_i(\delta_i,\hat{W}_i\ \hat{W}_{-i})\right\| \le \left\|\tilde{W}_i\right\|\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\| A\left\|\delta_i\right\| + \tfrac{1}{4}(d_i+g_i)^2\left\|\tilde{W}_i\right\|^2\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|^2\left\|B_i\right\|^2\left\|R_{ii}^{-1}\right\|$$

$$+\tfrac{1}{2}(d_i+g_i)^2\left\|W_{i\max}\right\|^2\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|^2\left\|B_i\right\|^2\left\|R_{ii}^{-1}\right\| + \tfrac{1}{2}(d_i+g_i)^2\left\|W_{i\max}\right\|\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|^2\left\|B_i\right\|^2\left\|R_{ii}^{-1}\right\|\left(\left\|\tilde{W}_i\right\|+W_{i\max}\right)$$

$$+\tfrac{1}{4}\sum_{j\in N_i}(d_j+g_j)^2\left\|\tilde{W}_j\right\|^2\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|^2\left\|B_j\right\|^2\left\|R_{jj}^{-1}R_{ij}R_{jj}^{-1}\right\|$$

$$+\tfrac{1}{2}\sum_{j\in N_i}(d_j+g_j)^2\left\|W_{j\max}\right\|^2\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|^2\left\|B_j\right\|^2\left\|R_{jj}^{-1}R_{ij}R_{jj}^{-1}\right\|$$

$$+\tfrac{1}{2}\sum_{j\in N_i}(d_j+g_j)^2\left\|W_{j\max}\right\|\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|^2\left\|B_j\right\|^2\left\|R_{jj}^{-1}R_{ij}R_{jj}^{-1}\right\|\left(\left\|\tilde{W}_j\right\|+W_{j\max}\right)$$

$$+\left(\left\|\tilde{W}_i\right\|+W_{i\max}\right)\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|\sum_{j\in N_i}e_{ij}\left\|B_j\right\|^2\left\|R_{jj}^{-1}\right\|\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|\left(\left\|\tilde{W}_j\right\|+W_{j\max}\right)$$

$$+\tfrac{1}{2}\left\|\tilde{W}_i\right\|\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|\sum_{j\in N_i}e_{ij}\left\|B_j\right\|^2\left\|R_{jj}^{-1}\right\|\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|\left\|\tilde{W}_j\right\|$$

$$+\tfrac{1}{2}\left\|W_{i\max}\right\|\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|\sum_{j\in N_i}e_{ij}\left\|B_j\right\|^2\left\|R_{jj}^{-1}\right\|\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|\left(\left\|\tilde{W}_j\right\|+W_{j\max}\right)$$

$$+\tfrac{1}{2}\left(\left\|\tilde{W}_i\right\|+W_{i\max}\right)\left\|\frac{\partial\varphi_i}{\partial\delta_i}\right\|\sum_{j\in N_i}e_{ij}\left\|B_j\right\|^2\left\|R_{jj}^{-1}\right\|\left\|\frac{\partial\varphi_j}{\partial\delta_j}\right\|W_{j\max}$$

$$+\bar{\varepsilon}_2 \tag{6.51}$$

All the signals on the right hand side of (6.51) are UUB and convergence to the approximate coupled HJ solution is obtained for every agent.

b. According to Theorem 6.5, $\left\|\hat{W}_{i+N}-W_i\right\|, \forall i$ are UUB. Then it is obvious that $\hat{u}_{i+N}, \forall i$ give the approximate cooperative Nash equilibrium (Definition 6.2).

∎

6.6 Simulations

This section will show the effectiveness of the online approach described in Theorem 6.5 for two different cases.

Consider the three-node strongly connected digraph structure shown in Figure 36 with a leader node connected to node 3. The edge weights and the pinning gains are taken equal to 1. Every node is a second order system.

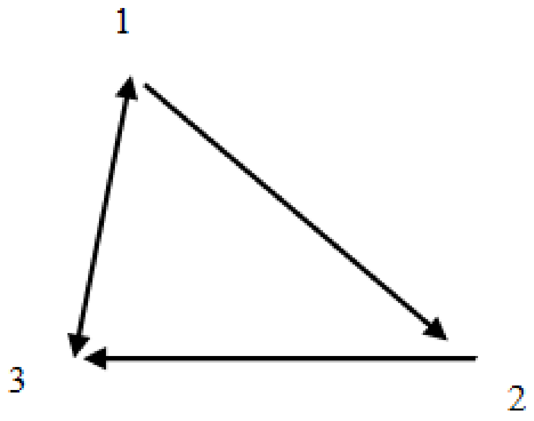

Figure 36. Communication Graph.

Select the weight matrices in (6.9) as

$Q_{11} = Q_{22} = Q_{33} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R_{11} = 4, R_{12} = 1, R_{13} = -1,\ R_{31} = -4, R_{22} = 9, R_{23} = 1, R_{33} = 9, R_{32} = 1, R_{21} = 1$ and $d_1 = d_2 = 1, d_3 = 2$.

Since the value is quadratic in the Linear Quadratic Regulator (LQR) case, the critic NNs basis sets were selected as the quadratic vector in the agent's components and the neighbors' components. The systems below are of second order then we can write for every agent

$$\delta_1 = \begin{bmatrix} \delta_{11} & \delta_{12} \end{bmatrix}, \delta_2 = \begin{bmatrix} \delta_{21} & \delta_{22} \end{bmatrix}, \delta_3 = \begin{bmatrix} \delta_{31} & \delta_{32} \end{bmatrix}$$

Thus the NN activation functions are

$$\phi_1(\delta_1,0,\delta_3)=\begin{bmatrix}\delta_{11}^2 & \delta_{11}\delta_{12} & \delta_{12}^2 & 0 & 0 & 0 & \delta_{31}^2 & \delta_{31}\delta_{32} & \delta_{32}^2\end{bmatrix}$$

$$\phi_1(\delta_1,\delta_2,0)=\begin{bmatrix}\delta_{11}^2 & \delta_{11}\delta_{12} & \delta_{12}^2 & \delta_{21}^2 & \delta_{21}\delta_{22} & \delta_{21}^2 & 0 & 0 & 0\end{bmatrix}$$

$$\phi_3(\delta_1,\delta_2,\delta_3)=\begin{bmatrix}\delta_{11}^2 & \delta_{11}\delta_{12} & \delta_{12}^2 & \delta_{21}^2 & \delta_{21}\delta_{22} & \delta_{22}^2 & \delta_{31}^2 & \delta_{31}\delta_{32} & \delta_{32}^2\end{bmatrix}$$

6.6.1 Velocity and Position regulated to zero

For the graph structure shown above consider the following node dynamics

$$\dot{x}_1=\begin{bmatrix}-2 & 1\\ -4 & -1\end{bmatrix}x_1+\begin{bmatrix}2\\ 1\end{bmatrix}u_1$$
$$\dot{x}_2=\begin{bmatrix}-2 & 1\\ -4 & -1\end{bmatrix}x_2+\begin{bmatrix}2\\ 3\end{bmatrix}u_2$$
$$\dot{x}_3=\begin{bmatrix}-2 & 1\\ -4 & -1\end{bmatrix}x_3+\begin{bmatrix}2\\ 2\end{bmatrix}u_3$$

The graphical game is implemented as in Theorem 6.5. Persistence of excitation was ensured by adding a small exponential decreasing probing noise to the control inputs. Figure 37 shows the critic parameters for every agent. Figure 38 shows the evolution of the states for the duration of the experiment.

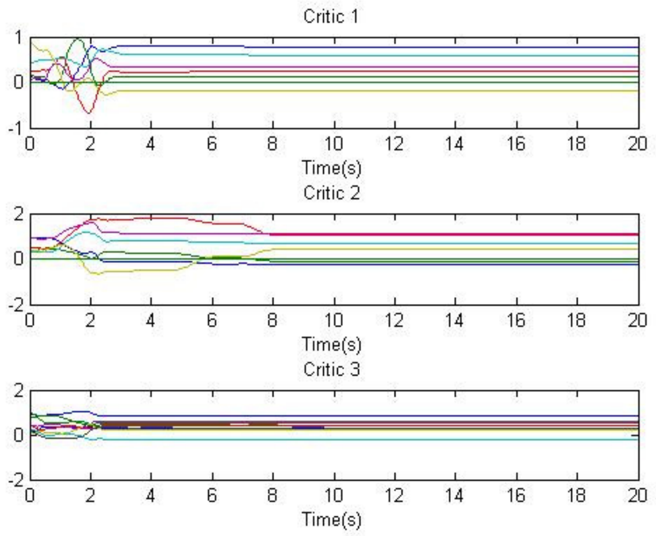

Figure 37. Critic parameters.

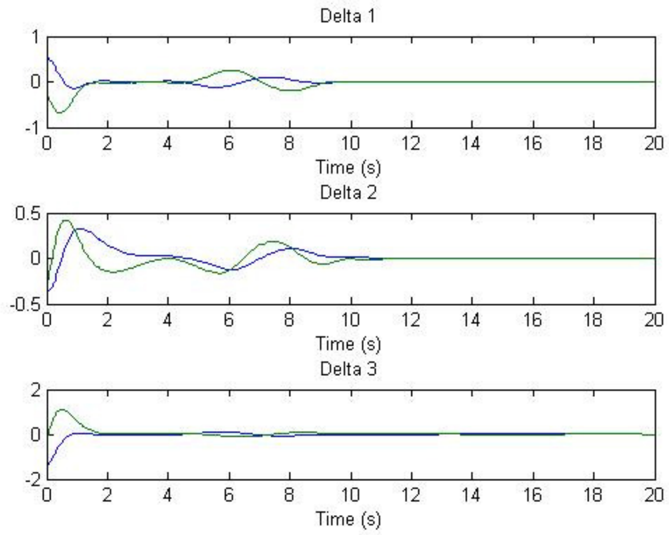

Figure 38. Evolution of the system states.

6.6.2 All nodes synchronize to the leader node

For the graph structure shown above consider the following node dynamics

$$\dot{x}_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x_1 + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u_1$$
$$\dot{x}_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x_2 + \begin{bmatrix} 2 \\ 3 \end{bmatrix} u_2$$
$$\dot{x}_3 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x_3 + \begin{bmatrix} 2 \\ 2 \end{bmatrix} u_3$$

The graphical game is implemented as in Theorem 6.5. Persistence of excitation was ensured by adding a small exponential decreasing probing noise to the control inputs. Figure 39 shows the critic parameters for every agent and figure 40 shows the synchronization of all the agents to the leader behavior.

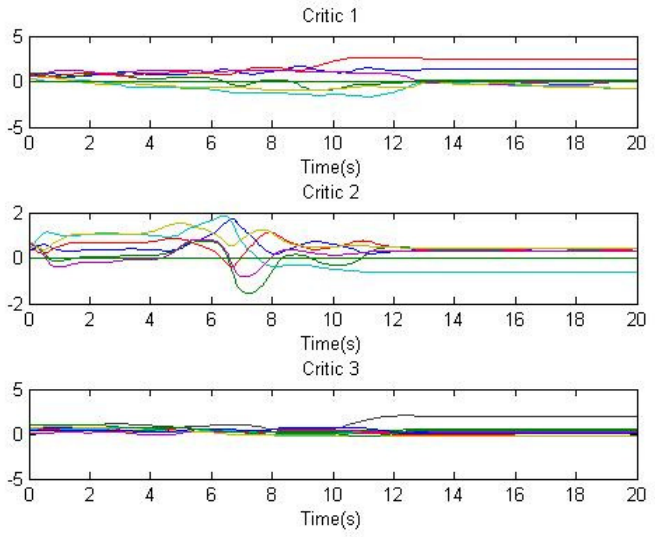

Figure 39. Critic parameters.

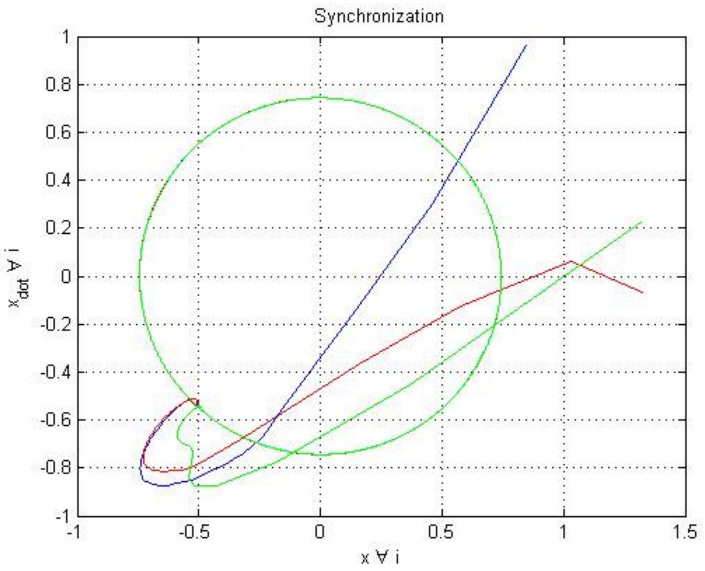

Figure 40. Synchronization of all the agents to the leader.

CHAPTER 7

ONLINE ADAPTIVE LEARNING OF OPTIMAL CONTROL SOLUTIONS USING INTEGRAL REINFORCEMENT LEARNING

7.1 Introduction

In this chapter we use Reinforcement Learning (RL) methods, specifically a new Integral Reinforcement Learning (IRL) approach, to provide an online learning solution to optimal control problem that does not require knowledge of the system drift dynamics. Integral reinforcement learning (IRL) was first introduced in [91], [92] to provide a practical means of applying reinforcement learning to continuous-time systems. The algorithm that we introduce herein is conceptually based on the Policy Iteration (PI) technique of chapter 2.

This chapter is concerned with developing approximate online solutions, based on PI, for the infinite horizon optimal control problem for continuous-time (CT) nonlinear systems. We present an online integral reinforcement algorithm that combines the advantages of the algorithm in chapter 2 and [92]. These include *simultaneous tuning* of both actor and critic neural networks (i.e. both neural networks are tuned at the same time) and no need for the drift term in the dynamics [92]. Simultaneous tuning idea was first introduced by [98], [99] and has been the idea of recent chapters in the area, however in most of these chapters the authors either designed model-based controllers [21] or used dynamic neural networks to identify the nonlinear plant [16]. Our algorithm avoids partial knowledge of the plant and uses only two neural networks by designing a hybrid controller as in [92].

The contributions in this chapter are i) provide a new online continuous time algorithm that converge to the solution of HJB and Bellman equation without solving them, ii) partial need of dynamics, and iii) update actor and critic neural networks simultaneously in real time.

The chapter is organized as follows. Section 7.2 provides the formulation of the optimal control problem followed by the general description of neural network value function approximation (VFA). Section 7.3 introduces the online synchronous integral reinforcement learning algorithm for the actor and critic networks based on PI. Results for convergence and stability are given. Section 7.4 presents simulation examples that show the effectiveness of the online integral reinforcement learning algorithm.

7.2 Optimal control problem and the policy iteration algorithm

7.2.1 Optimal control and the continuous time HJB equation

Let the system dynamics be described by the differential equation

$$\dot{x}(t) = f(x(t)) + g(x(t))\,u(x(t))\,;\, x(0) = x_0 \tag{7.1}$$

with state $x(t) \in \mathbb{R}^n$, $f(x(t)) \in \mathbb{R}^n$, $g(x(t)) \in \mathbb{R}^{n\times m}$ and control input $u(t) \in U \subset \mathbb{R}^m$. We assume that $f(0)=0,\quad g(0)=0$, $f(x)+g(x)u$ is Lipschitz continuous on a set $\Omega \subseteq \mathbb{R}^n$ that contains the origin. We assume 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, and that $f(x)+g(x)u$ is Lipschitz continuous on Ω.

Define the infinite horizon integral cost

$$V(x_0) = \int_0^\infty r(x(\tau),u(\tau))d\tau \tag{7.2}$$

where $r(x,u) = Q(x) + u^T R u$ 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 7.1. [4] (Admissible policy) A control policy $\mu(x)$ is defined as admissible with respect to (7.2) on Ω, denoted by $\mu \in \Psi(\Omega)$, if $\mu(x)$ is continuous on Ω, $\mu(0)=0$, $\mu(x)$ stabilizes (7.1) on Ω and $V(x_0)$ is finite $\forall x_0 \in \Omega$.

For any admissible control policy $\mu \in \Psi(\Omega)$, if the associated cost function

$$V^{\mu}(x_0) = \int_0^{\infty} r(x(\tau), \mu(x(\tau)))d\tau \tag{7.3}$$

is C^1 , then an infinitesimal version of (7.3)is

$$0 = r(x, \mu(x)) + V_x^{\mu T}(f(x) + g(x)\mu(x)), \; V^{\mu}(0) = 0 \tag{7.4}$$

where V_x^{μ} denotes the partial derivative of the value function V^{μ} with respect to x . (Note that the value function does not depend explicitly on time). Equation (7.4) is a Lyapunov equation for nonlinear systems which, given a controller $\mu(x) \in \Psi(\Omega)$, can be solved for the value function $V^{\mu}(x)$ associated with it. Given that $\mu(x)$ is an admissible control policy, if $V^{\mu}(x)$ satisfies (7.4) , with $r(x, \mu(x)) \geq 0$, then $V^{\mu}(x)$ is a Lyapunov function for the system (7.1) with control policy $\mu(x)$.

The optimal control problem can now be formulated: Given the continuous-time system (7.1), the set $\mu \in \Psi(\Omega)$ of admissible control policies and the infinite horizon cost functional (7.2), find an admissible control policy such that the cost index (7.2) associated with the system (7.1) is minimized.

Define the Hamiltonian of the problem

$$H(x, u, \nabla V_x) = r(x(t), u(t)) + \nabla V_x^T (f(x(t)) + g(x(t))u(t)) \tag{7.5}$$

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

$$0 = \min_{u \in \Psi(\Omega)} [H(x, u, \nabla V_x^*)] \tag{7.6}$$

where $\nabla V_x \equiv \dfrac{\partial V}{\partial x}$ is disabused here as a column vector.

Assuming that the minimum on the right hand side of (7.6) exists and is unique then the optimal control function for the given problem is

$$u^*(x) = -R^{-1} g^T(x) \nabla V_x^* \tag{7.7}$$

Inserting this optimal control policy in the Hamiltonian we obtain the formulation of the HJB equation in terms of V_x^*

$$0 = Q(x) + \nabla V_x^{*T} f(x) - \frac{1}{4} \nabla V_x^{*T} g(x) R^{-1} g^T(x) \nabla V_x^*, V^*(0) = 0 \tag{7.8}$$

This is a necessary and sufficient condition for the optimal value function [53]. 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 (7.8)for the value function and then substitute the solution in (7.7) to obtain the optimal control. However, solving the HJB equation is generally difficult as it is a nonlinear differential equation, quadratic in the cost function, which also requires complete knowledge of the system dynamics (*i.e.* the system dynamics described by the functions $f(x), g(x)$ need to be known). The next section provides the policy iteration algorithm and the value function approximation of the critic network.

7.2.2 Policy iteration

Policy iteration is an iterative method of reinforcement learning [16] for solving (7.8), and consists of policy improvement based on (7.7) and policy evaluation based on (7.4).

In the actor/critic structure the Critic and the Actor functions are approximated by neural networks, and the PI algorithm consists in tuning alternatively each of the two neural networks.

The critic neural network is tuned to evaluate the performance of the current control policy.

Policy Iteration Algorithm:

Step 1. Given policies $\mu^{(i)}(x)$, solve for the value $V^{\mu^{(i)}}(x(t))$ using

$$\begin{aligned} &0 = r(x,\mu^{(i)}(x)) + (\nabla V_x^{\mu^{(i)}})^T (f(x) + g(x)\mu^{(i)}(x)) \\ &V^{\mu^{(i)}}(0) = 0 \end{aligned} \tag{7.9}$$

Step 2. Update the control policy using

$$\mu^{(i+1)} = \underset{u \in \Psi(\Omega)}{\arg\min}[H(x,u,\nabla V_x^{(i)})] \tag{7.10}$$

which explicitly is

$$\mu^{(i+1)}(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla V_x^{(i)} \tag{7.11}$$

To ensure convergence of the PI algorithm an initial admissible policy $\mu^{(0)}(x(t)) \in \Psi(\Omega)$ is required. It is in fact required by the desired completion of the first step in the policy iteration: i.e. finding a value associated with that initial policy (which needs to be admissible to have a finite value and for the nonlinear Lyapunov equation to have a solution). The algorithm then converges to the optimal control policy $\mu^* \in \Psi(\Omega)$ with corresponding cost $V^*(x)$. Proofs of convergence of the PI algorithm have been given in several references. See [4], [13], [15], [30], [35], [63], [92].

Policy iteration is a Newton method. In the linear time-invariant case, it reduces to the Kleinman algorithm [44] for solution of the Riccati equation, a familiar algorithm in control systems. Then, (7.9) become a Lyapunov equation.

A major problem with this formulation of PI for CT systems is that the full system dynamics must be known as both *f(x)* and *g(x)* appear in the Bellman equation (7.9).

7.2.3 Value function approximation (VFA)

A practical method for implementing PI for CT systems is presented in this section. This involves two aspects: value function approximation (VFA) and integral reinforcement learning (IRL). The critic NN is based on value function approximation (VFA). Thus, assume there exist weights W_1 such that the value $V(x)$ is approximated by a neural network as

$$V(x) = W_1^T \phi(x) + \varepsilon(x) \tag{7.12}$$

where $\phi(x) : \mathbb{R}^n \to \mathbb{R}^N$ is the activation functions vector, *N* the number of neurons in the hidden layer, and $\varepsilon(x)$ the NN approximation error. It is known that $\varepsilon(x)$ is bounded by a constant on a compact set. Select the activation functions to provide a *complete* basis set such that $V(x)$ and its derivative

$$\frac{\partial V}{\partial x} = \nabla \phi^T W_1 + \frac{\partial \varepsilon}{\partial x} \tag{7.13}$$

are uniformly approximated. According to the Weierstrass higher-order approximation theorem [4], such a basis exists if $V(x)$ is sufficiently smooth. This means that, as the number of hidden-layer neurons $N \to \infty$, the approximation error $\varepsilon \to 0$ uniformly.

7.2.4 Integral reinforcement learning

The PI algorithm given above requires full system dynamics, since both *f(x)* and *g(x)* appear in the Bellman equation (7.9). In order to find an equivalent formulation of the Bellman equation that does not involve the dynamics, we note that for any time t_0 and time interval *T* the value function (7.3) satisfies

$$V^{\mu}(x_{t_0}) = \int_{t_0-T}^{t_0} r(x(\tau), \mu(x(\tau)))d\tau + V^{\mu}(x_{t_0-T}) \tag{7.14}$$

In [92] it is shown that (7.14) and (7.9) are equivalent, i.e., they both have the same solution. Therefore, (7.14) can be seen as a Bellman equation for CT systems. Note that this form does not involve the system dynamics. We call this the integral reinforcement learning (IRL) form of the Bellman equation.

Therefore, by using a critic NN for VFA, the Bellman error based on (7.14) becomes [92]

$$\int_{t-T}^{t} \left(Q(x)+\mu^T R\mu\right) d\tau + W_1^T \phi(x(t)) - W_1^T \phi(x(t-T)) = \varepsilon_B \tag{7.15}$$

We define the integral reinforcement as

$$p = \int_{t-T}^{t} \left(Q(x)+\mu^T R\mu\right) d\tau \tag{7.16}$$

Now (7.15) can be written as

$$\varepsilon_B - p = W_1^T \Delta\phi(x(t)) \tag{7.17}$$

where $\Delta\phi(x(t)) \equiv \phi(x(t)) - \phi(x(t-T))$.

Under the Lipschitz assumption on the dynamics, this residual error is bounded on a compact set. Moreover, in [4] it has been shown that, under certain assumptions, as the number of hidden layer neurons $N \to \infty$, one has $\varepsilon_B \to 0$.

7.3 Online integral reinforcement learning algorithm with synchronous tuning of actor and critic neural networks

Standard PI algorithms for CT systems are offline methods that require complete knowledge on the system dynamics to obtain the solution (*i.e.* the functions $f(x), g(x)$ in (7.1) need to be known). In order to change the offline character of PI for CT systems, and thus make it consistent with online learning mechanisms in the mammal brain, we present an adaptive learning algorithm that uses simultaneous continuous-time tuning for the actor and critic neural networks and does not need the drift term $f(x)$ in the dynamics. We term this *online integral reinforcement learning algorithm*.

7.3.1 Critic NN and Bellman equation solution

The weights of the critic NN, W_1, which solve (7.15) are unknown. Then the output of the critic neural network is

$$\hat{V}(x) = \hat{W}_1^T \phi(x) \tag{7.18}$$

where $\hat{W}_1$ are the current known values of the critic NN weights. Recall that $\phi(x) : \mathbb{R}^n \to \mathbb{R}^N$ is the activation functions vector, with *N* the number of neurons in the hidden layer. The approximate Bellman error is then

$$\int_{t-T}^{t} \left(Q(x) + u^T Ru\right) d\tau + \hat{W}_1^T \phi(x(t)) - \hat{W}_1^T \phi(x(t-T)) = e_1 \tag{7.19}$$

which according to (7.16) can be written as

$$\hat{W}_1^T \Delta\phi(x(t)) = e_1 - p \tag{7.20}$$

It is desired to select $\hat{W}_1$ to minimize the squared residual error

$$E_1 = \tfrac{1}{2} e_1^T e_1 \tag{7.21}$$

Then $\hat{W}_1(t) \to W_1$. We select the tuning law for the critic weights as the normalized gradient descent algorithm

$$\dot{\hat{W}}_1 = -a_1 \frac{\Delta\phi(x(t))^T}{\left(1 + \Delta\phi(x(t))^T \Delta\phi(x(t))\right)^2} \left[\int_{t-T}^{t} \left(Q(x) + u^T Ru\right) d\tau + \Delta\phi(x(t))^T \hat{W}_1 \right] \tag{7.22}$$

Note that the data required in this tuning algorithm at each time are $\left(\Delta\phi(t), p(t)\right)$.

Define the critic weight estimation error $\tilde{W}_1 = W_1 - \hat{W}_1$ and substitute (7.15) in (7.22) and, with the notation $\overline{\Delta\phi}(t) = \Delta\phi(t)/(\Delta\phi(t)^{\mathrm{T}}\Delta\phi(t)+1)$ and $m_s = 1 + \Delta\phi(t)^T \Delta\phi(t)$, we obtain the dynamics of the critic weight estimation error as

$$\dot{\tilde{W}}_1 = -a_1 \overline{\Delta\phi}(t)\,\overline{\Delta\phi}(t)^{\mathrm{T}}\tilde{W}_1 + a_1\overline{\Delta\phi}(t)\frac{\varepsilon_B}{m_s} \tag{7.23}$$

Though it is traditional to use critic tuning algorithms of the form (7.22), it is not generally understood when convergence of the critic weights can be guaranteed. In this chapter, we address this issue in a formal manner. To guarantee convergence of $\hat{W}_1$ to W_1, the next Persistence of Excitation (PE) assumption is required.

Note that:

$$\Delta\phi(x(t)) = \int_{t-T}^{t} \nabla\phi(x)\dot{x}d\tau = \int_{t-T}^{t} \nabla\phi(f+gu)\,d\tau \tag{7.24}$$

It is obvious to see from (7.20) that the regression vector $\overline{\Delta\phi}(t)$ must be persistently exciting to solve for $\hat{W}_1$ in a least squares sense.

Persistence of Excitation (PE) Assumption. Let the signal $\overline{\Delta\phi}(t)$ be persistently exciting over the interval $[t-T,t]$, *i.e.* there exist constants $\beta_1 > 0$, $\beta_2 > 0$, $\mathrm{T} > 0$ such that, for all *t*,

$$\beta_1 \mathrm{I} \le S_0 \equiv \int_{t-T}^{t} \overline{\Delta\phi}(\tau)\overline{\Delta\phi}^{\mathrm{T}}(\tau)d\tau \le \beta_2 \mathrm{I} \tag{7.25}$$

Remark 7.1. Note that, as $N \to \infty$, $\varepsilon_B \to 0$ uniformly [4].

7.3.2 Action NN and online adaptive optimal control

The policy improvement step in PI is given by substituting (7.13)into (7.7)as

$$u(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \phi^T W_1 \tag{7.26}$$

with critic weights W_1 unknown. Therefore, define the control policy in the form of an action neural network which computes the control input in the structured form

$$u_2(x) = -\tfrac{1}{2} R^{-1} g^T(x) \nabla \phi^T \hat{W}_2 \tag{7.27}$$

where $\hat{W}_2$ denotes the current known values of the actor NN weights.

Based on (7.8) and (7.15), define the approximate HJB equation

$$\int_{t-\mathrm{T}}^{t} \left(-Q(x) - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \varepsilon_{HJB}(x) \right) d\tau = W_1^T \Delta\phi(x(t)) \tag{7.28}$$

with the notation $\overline{D}_1(x) = \nabla\phi(x) g(x) R^{-1} g^T(x) \nabla\phi^T(x)$, where W_1 denotes the ideal unknown weights of the critic and actor neural networks which solve the HJB.

We now present the main Theorems, which provide the tuning laws for the actor and critic neural networks that guarantee convergence to the optimal controller along with closed-loop stability. The next notion of practical stability is needed.

Definition 7.2. [50] (UUB) A time signal $\zeta(t)$ is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset \mathbb{R}^n$ so that for all $\zeta(0) \in S$ there exists a bound *B* and a time $T(B, \zeta(0))$ such that $\|\zeta(t)\| \le B$ for all $t \ge t_0 + T$.

Theorem 7.1. Let tuning for the critic NN be provided by

$$\dot{\hat{W}}_1 = -a_1 \frac{\Delta\phi(x(t))^T}{\left(1 + \Delta\phi(x(t))^T \Delta\phi(x(t))\right)^2} \left(\Delta\phi(x(t))^T \hat{W}_1 + \int_{t-T}^{t} \left(Q(x) + \frac{1}{4} \hat{W}_2^T \overline{D}_1 \hat{W}_2 \right) d\tau \right) \tag{7.29}$$

where $\Delta\phi(x(t)) = \int\limits_{t-T}^{t} \nabla\phi(f + gu_2)\, d\tau$ and assume that $\overline{\Delta}\phi(t)$ is persistently exciting (which means u_2 is persistently exciting). Let the actor NN be tuned as

$$\dot{\hat{W}}_2 = -a_2\left(F_2\hat{W}_2 - F_1\Delta\phi(x(t))^T\hat{W}_1\right) - \tfrac{1}{4}a_2\overline{D}_1(x)\hat{W}_2\frac{\Delta\phi(x(t))^T}{\left(1+\Delta\phi(x(t))^T\Delta\phi(x(t))\right)^2}\hat{W}_1 \qquad (7.30)$$

Then the closed-loop system state is UUB, the critic parameter error $\tilde{W}_1 = W_1 - \hat{W}_1$ and the actor parameter error $\tilde{W}_2 = W_1 - \hat{W}_2$ are UUB.

Proof:

The convergence proof is based on Lyapunov analysis. We consider the Lyapunov function

$$L(t) = V(x) + \frac{1}{2}tr(\tilde{W}_1^T a_1^{-1}\tilde{W}_1) + \frac{1}{2}tr(\tilde{W}_2^T a_2^{-1}\tilde{W}_2) \qquad (7.31)$$

With the chosen tuning laws one can then show that the errors $\tilde{W}_1$ and $\tilde{W}_2$ are UUB and convergence is obtained. The chapter has the same form with the one in chapter 2.

■

Theorem 7.2. Optimal solution. Suppose the hypotheses of Theorem 7.1 hold. Then:

a. $H(\hat{u},\hat{W}_1,x) \equiv \int\limits_{t-T}^{t}\left(Q(x) + \hat{u}^T R\hat{u} - \varepsilon_{HJB}\right)d\tau + \hat{W}_1^T\phi(x(t)) - \hat{W}_1^T\phi(x(t-T))$

is UUB, where $\hat{u} = -\tfrac{1}{2}R^{-1}g^T(x)\nabla\phi^T(x)\hat{W}_1$. That is, $\hat{W}_1$ converge to the approximate HJB solution.

b. $\hat{u}_2(x)$ converges to the optimal solution, where $\hat{u}_2 = -\tfrac{1}{2}R^{-1}g^T(x)\nabla\phi^T(x)\hat{W}_2$.

Proof:

a. Consider the weights $\tilde{W}_1$, $\tilde{W}_2$ to be UUB as proved in Theorem 7.1.

$$H(\hat{u},\hat{W}_1,x) \equiv \int_{t-T}^{t} \left(Q(x)+\hat{u}^T R\hat{u}-\varepsilon_{HJB}\right) d\tau + \hat{W}_1^T \phi(x(t)) - \hat{W}_1^T \phi(x(t-T))$$

After adding zero we have

$$H(\hat{u},\hat{W}_1,x) \equiv \int_{t-T}^{t} \left(\tfrac{1}{4}\tilde{W}_1^T \bar{D}_1(x)\tilde{W}_1 - \tfrac{1}{2}\tilde{W}_1^T \bar{D}_1(x)W_1 - \varepsilon_{HJB}\right) d\tau - \tilde{W}_1^T \varphi(x(t)) + \tilde{W}_1^T \varphi(x(t-T))$$

By taking norms in both sides and taking into account that $\sup_{x\in\Omega}\left\|\varepsilon_{HJB}\right\| < \bar{\varepsilon}$ and letting $\left\|W_1\right\| < W_{1\max}$.

$$\left\|H(\hat{u},\hat{W}_1,x)\right\| \equiv \int_{t-T}^{t} \left(\tfrac{1}{4}\left\|\tilde{W}_1\right\|^2 \left\|\bar{D}_1(x)\right\| + \tfrac{1}{2}\left\|\tilde{W}_1\right\| W_{1\max} \left\|\bar{D}_1(x)\right\| + \bar{\varepsilon}\right) d\tau + \left\|\tilde{W}_1\right\| (\left\|\varphi(x(t))\right\| + \left\|\varphi(x(t-T))\right\|) \quad (7.32)$$

All the signals on the right hand side of are UUB. So $H(\hat{u},\hat{W}_1,x)$ is UUB and convergence to the approximate HJB solution is obtained.

b. According to Theorem 7.1 and equations (7.26)and, (7.27), $\left\|u_2 - u\right\|$ is UUB because $\left\|\hat{W}_2 - W_1\right\|$ is UUB.

So $u_2(x)$ gives the optimal solution.

This completes the proof.

■

Remark 7.2. The positive tuning parameters F_1, F_2 are selected appropriately to ensure stability.

7.4 Simulation results

To support the new synchronous online integral reinforcement learning algorithm for CT systems, we offer two simulation examples, one linear and one nonlinear. In both cases we observe convergence to the actual optimal value function and control. In these simulations, exponentially decreasing noise is added to the control inputs to ensure PE until convergence is obtained.

7.4.1 Linear system example

Consider the continuous-time F16 aircraft plant with quadratic cost function used in [85]

$$\dot{x} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

where Q and R in the cost function are identity matrices of appropriate dimensions and $T = 0.01$. In this linear case the solution of the HJB equation is given by the solution of the algebraic Riccati equation (ARE). Since the value is quadratic in the LQR case, the critic NN basis set $\phi(x)$ was selected as the quadratic vector in the state components. Solving the ARE gives the parameters of the optimal critic as $W_1^* = [1.4245 \quad 1.1682 \quad -0.1352 \quad 1.4349 \quad -0.1501 \; 0.4329]^T$.

The integral reinforcement algorithm is implemented as in Theorem 7.1. PE was ensured by adding a small probing noise to the control input. Figure 41 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \; W_{c4} \quad W_{c5} \quad W_{c6}]^T$$

converging to the optimal values. In fact after 250s the critic parameters converged to

$$\hat{W}_1(t_f) = [1.4279 \quad 1.0193 \quad -0.1473 \quad 1.4462 \quad -0.1376 \; 0.4330]^T$$

The actor parameters after 300s converge to the values of

$$\hat{W}_2(t_f) = [1.4279 \quad 1.0193 \quad -0.1473 \quad 1.4462 \quad -0.1376 \; 0.4330]^T.$$

The actor NN is given by

$$\hat{u}_2(x) = -\tfrac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & 0 & x_1 \\ 0 & 2x_2 & 0 \\ 0 & x_3 & x_2 \\ 0 & 0 & 2x_3 \end{bmatrix}^T \hat{W}_2(t_f)$$

The evolution of the system states is presented in Figure 42. One can see that after 250s convergence of the NN weights in both critic and actor has occurred. This shows that the probing noise effectively guaranteed the PE condition.

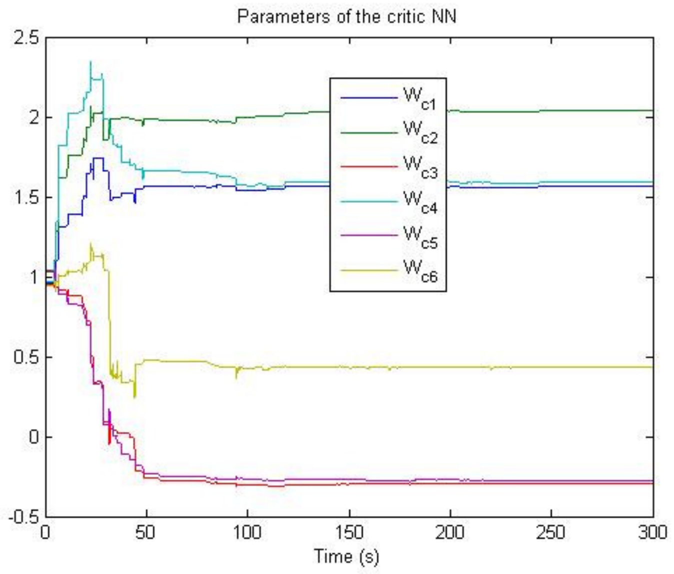

Figure 41. Convergence of the critic parameters to the parameters of the optimal critic.

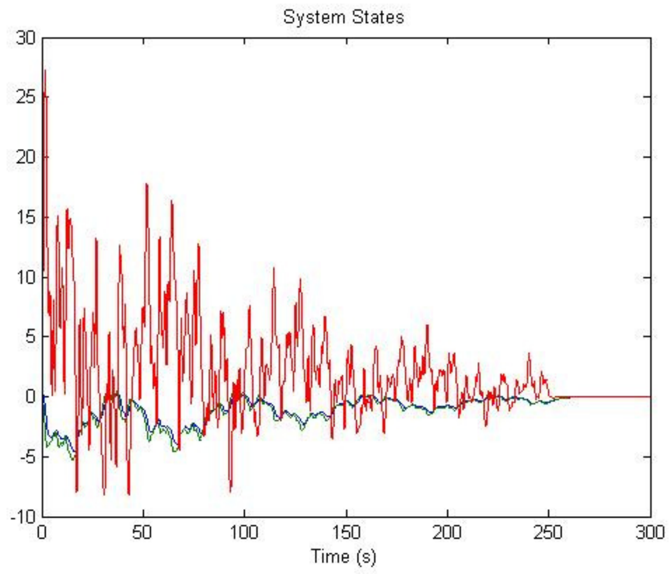

Figure 42. Evolution of the system states for the duration of the experiment.

7.4.2 Nonlinear system example

Consider the following affine in control input nonlinear system, with a quadratic cost constructed as in [64]

$$\dot{x} = f(x) + g(x)u,\ x \in R^2$$

where

$$f(x) = \begin{bmatrix} -x_1 + x_2 \\ -x_1{}^3 - x_2 - \dfrac{x_1^2}{x_2} + 0.25x_2(\cos(2x_1 + x_1^3) + 2)^2 \end{bmatrix}, \quad g(x) = \begin{bmatrix} 0 \\ \cos(2x_1 + x_1^3) + 2 \end{bmatrix},$$

One selects $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 1$ and $T = 0.01$.

The optimal value function is

$$V^*(x) = \frac{1}{4}x_1{}^4 + \frac{1}{2}x_2{}^2$$

the optimal control signal is

$$u^*(x) = -\tfrac{1}{2}(\cos(2x_1 + x_1^3) + 2)x_2$$

One selects the critic NN vector activation function as

$$\phi(x) = [x_1{}^2 \quad x_2^2 \quad x_1{}^4 \ x_2{}^4]$$

Figure 43 shows the critic parameters, denoted by

$$\hat{W}_1 = [W_{c1} \quad W_{c2} \quad W_{c3} \quad W_{c4}]^T$$

After the simulation by using the integral reinforcement learning algorithm we have

$$\hat{W}_1(t_f) = [0.0033 \quad 0.4967 \quad 0.2405 \quad 0.0153]^T$$

The actor parameters after 80s converge to the values of

$$\hat{W}_2(t_f) = [0.0033 \quad 0.4967 \quad 0.2405 \quad 0.0153]^T.$$

The actor NN is given by

$$\hat{u}_2(x) = -\frac{1}{2}\begin{bmatrix} 0 \\ \cos(2x_1 + x_1^3) + 2 \end{bmatrix}^T \begin{bmatrix} 2x_1 & 0 & 4x_1^3 & 0 \\ 0 & 2x_2 & 0 & 4x_2^3 \end{bmatrix} \hat{W}_2(t_f)$$

The evolution of the system states is presented in Figure 44. One can see that after 80s convergence of the NN weights in both critic and actor has occurred. This shows that the probing noise effectively guaranteed the PE condition.

Figure 45 shows the 3-D plot of the difference between the approximated value function, by using the online algorithm, and the optimal one. This error is close to zero. Good approximation of the actual value function is being evolved. Figure 46 shows the 3-D plot of the difference between the approximated control, by using the online integral reinforcement learning algorithm, and the optimal one. This error is close to zero.

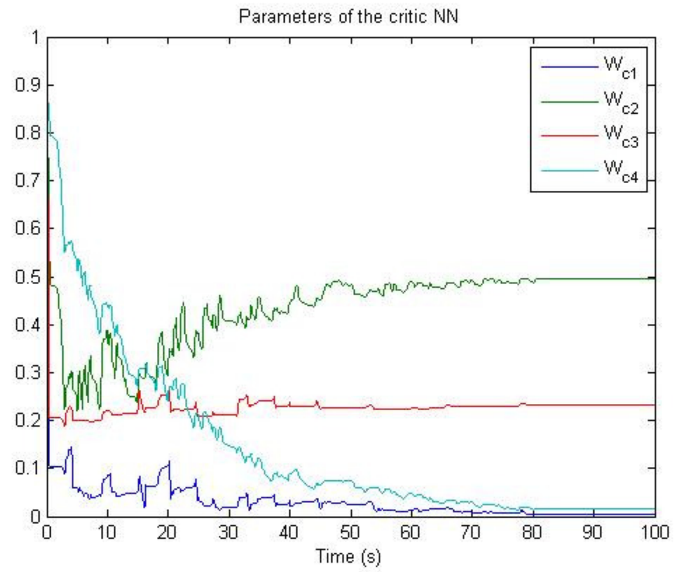

Figure 43. Convergence of the critic parameters to the parameters of the optimal critic.

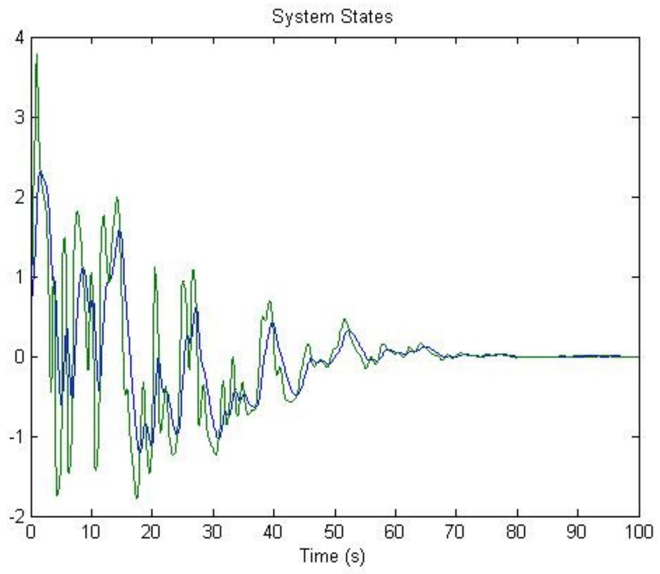

Figure 44. Evolution of the system states for the duration of the experiment.

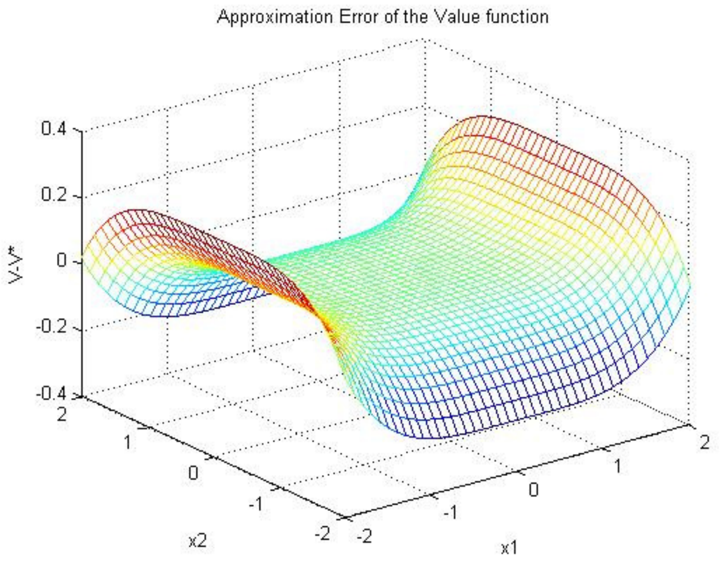

Figure 45. Error between the optimal and approximated value function.

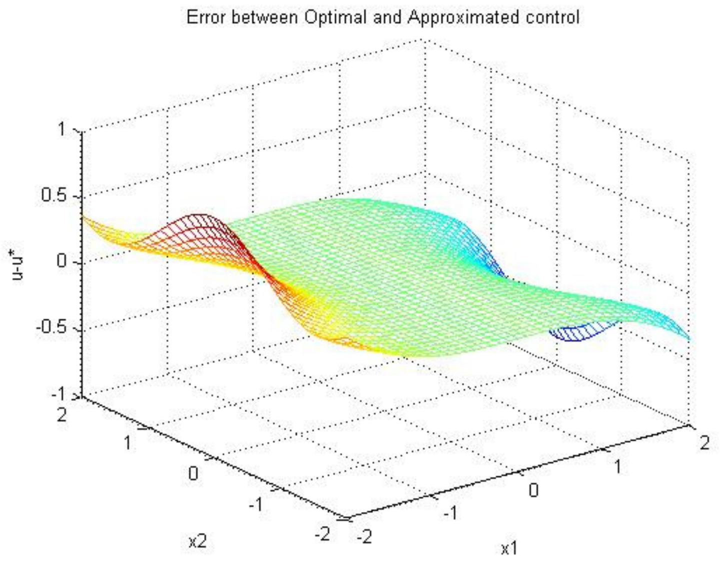

Figure 46. Error between the optimal and the approximated control input.

7.5 Conclusion

In this chapter we have proposed a new adaptive algorithm which solves the continuous-time optimal control problem for affine in the inputs nonlinear systems. The importance of this algorithm relies on the partial need of dynamics, only $g(x)$ is needed, the simultaneous tuning of the actor and critic neural networks and the convergence to HJB and Bellman equation solution without solving these equations.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this thesis have been developed online learning algorithms which use reinforcement learning ideas to solve games and optimal control problems by tuning simultaneously all the critic and actor neural networks and by measuring the states or input/output data.

This thesis considers three classes of dynamical systems. The first one is the linear continuous-time system, the second one is the affine in control nonlinear continuous-time system and the third one is the linear discrete-time system.

The new results presented herein are:

1. An online adaptive optimal controller to solve the continuous-time infinite horizon optimal control problem by tuning the critic and actor neural networks simultaneously. The algorithm converges to the solution of Hamilton-Jacobi-Bellman equation without solving it.

2. An online gaming algorithm to solve the zero-sum game problem, by tuning the critic, actor and disturbance neural networks simultaneously. The algorithm converges to the solution of Hamilton-Jacobi-Isaacs equation without solving it.

3. An online adaptive control algorithm based on policy iteration to solve the continuous-time multi player non zero sum game with infinite horizon for linear and nonlinear systems. Every player has his own actor/critic structure and the algorithm tunes all *N* actor/critic neural networks at the same time. The algorithm converges to the solution of coupled Hamilton-Jacobi equations and coupled Riccati equations for nonlinear and linear systems respectively without solving them.

4. Reinforcement learning methods which require only output feedback and yet converge to an optimal controller.

5. Policy iteration and online learning solution is developed for graphical games.

6. Online adaptive learning algorithm (synchronous policy iteration) that uses integral reinforcement learning and does not need any knowledge on the drift dynamics.

Convergence and stability proofs are provided for all the aforementioned algorithms.

The following are some of the directions for continuation of this work

1. Extend the idea of online adaptive learning algorithm with integral reinforcement learning to zero and non-zero sum games.

2. Adaptive optimal controllers for nonlinear systems that use only output feedback.

APPENDIX A

PROOFS

Proof for Technical Lemma 2.2 Part a:

This is a more complete version of results in [36], [87].

Set $\varepsilon_H = 0$ in (2.26). Take the Lyapunov function

$$L = \frac{1}{2}\tilde{W}_1^T a_1^{-1}\tilde{W}_1 \tag{A.1}$$

The derivative is

$$\dot{L} = -\tilde{W}_1^T \bar{\sigma}_1 \bar{\sigma}_1^{\mathrm{T}} \tilde{W}_1$$

Integrating both sides

$$L(t+T) - L(t) = -\int_t^{t+T} \tilde{W}_1^{\mathrm{T}} \sigma_1(\tau)\sigma_1^{\mathrm{T}}(\tau)\tilde{W}_1 d\tau$$

$$L(t+T) = L(t) - \tilde{W}_1^{\mathrm{T}}(t)\int_t^{t+T} \Phi^T(\tau,t)\sigma_1(\tau)\sigma_1^{\mathrm{T}}(\tau)\Phi(\tau,t)d\tau\, \tilde{W}_1(t)$$

$$= L(t) - \tilde{W}_1^{\mathrm{T}}(t) S_1 \tilde{W}_1(t) \le (1-2a_1\beta_3)L(t)$$

So

$$L(t+\mathrm{T}) \le (1-2a_1\beta_3)L(t) \tag{A.2}$$

Define $\gamma = (1-2a_1\beta_3)$. By using norms we write (A.2) in terms of $\tilde{W}_1$ as

$$\frac{1}{2a_1}\left\|\tilde{W}(t+T)\right\|^2 \le \sqrt{(1-2a_1\beta_3)}\,\frac{1}{2a_1}\left\|\tilde{W}(t)\right\|^2$$

$$\left\|\tilde{W}(t+T)\right\| \le \sqrt{(1-2a_1\beta_3)}\left\|\tilde{W}(t)\right\|$$

$$\left\|\tilde{W}(t+T)\right\| \le \gamma\left\|\tilde{W}(t)\right\|$$

Therefore

$$\|\tilde{W}(k\mathrm{T})\| \le \gamma^k \,\|\tilde{W}(0)\| \tag{A.3}$$

i.e. $\tilde{W}(t)$ decays exponentially. To determine the decay time constant in continuous time, note that

$$\|\tilde{W}(k\mathrm{T})\| \le e^{-\alpha kT}\|\tilde{W}(0)\| \tag{A.4}$$

where $e^{-\alpha kT} = \gamma^k$.

Therefore the decay constant is

$$\alpha = -\frac{1}{\mathrm{T}}\ln(\gamma) \Leftrightarrow \alpha = -\frac{1}{\mathrm{T}}\ln(\sqrt{1-2a_1\beta_3}). \tag{A.5}$$

This completes the proof.

■

Proof for Technical Lemma 2.2 Part b:

Consider the system

$$\begin{cases} \dot{x}(t) = B(t)u(t) \\ y(t) = C^T(t)x(t) \end{cases} \tag{A.6}$$

The state and the output are

$$\begin{cases} x(t+T) = x(t) + \int_t^{t+T} B(\tau)u(\tau)d\tau \\ y(t+T) = C^T(t+T)x(t+T) \end{cases} \tag{A.7}$$

Let $C(t)$ be PE, so that

$$\beta_1 I \le S_C \equiv \int_t^{t+T} C(\lambda)C^T(\lambda)d\lambda \le \beta_2 I\ . \tag{A.8}$$

Then,

$$y(t+T) = C^T(t+T)x(t) + \int_t^{t+T} C^T(t+T)B(\tau)u(\tau)d\tau$$

$$\int_t^{t+T} C(\lambda)\left(y(\lambda) - \int_t^{\lambda} C^T(\lambda)B(\tau)u(\tau)d\tau \right) d\lambda =$$

$$\int_t^{t+T} C(\lambda)C^T(\lambda)x(t)d\lambda$$

$$\int_t^{t+T} C(\lambda)\left(y(\lambda) - \int_t^{\lambda} C^T(\lambda)B(\tau)u(\tau)d\tau \right) d\lambda = S_C x(t)$$

$$x(t) = S_C^{-1}\left\{ \int_t^{t+T} C(\lambda)\left(y(\lambda) - \int_t^{\lambda} C^T(\lambda)B(\tau)u(\tau)d\tau \right) d\lambda \right\}$$

Taking the norms in both sides yields

$$\| x(t) \| \le \| S_C^{-1} \int_t^{t+T} C(\lambda)y(\lambda)d\lambda \| + \| S_C^{-1} \left\{ \int_t^{t+T} C(\lambda)\left(\int_t^{\lambda} C^T(\lambda)B(\tau)u(\tau)d\tau \right) d\lambda \right\} \|$$

$$\| x(t) \| \le (\beta_1 \mathrm{I})^{-1} (\int_t^{t+T} C(\lambda)C^T(\lambda)d\lambda)^{\frac{1}{2}} (\int_t^{t+T} y(\lambda)^T y(\lambda)d\lambda)^{\frac{1}{2}}$$

$$+ \left\| S_C^{-1} \right\| \left\{ \int_t^{t+\mathrm{T}} \left\| C(\lambda)C^T(\lambda) \right\| d\lambda \int_t^{t+\mathrm{T}} \| B(\tau)u(\tau) \| d\tau \right\}$$

$$\| x(t) \| \le \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1} y_{\max} + \frac{\delta\beta_2}{\beta_1} \int_t^{t+T} \| B(\tau) \| \cdot \| u(\tau) \| d\tau \qquad \text{(A.9)}$$

where δ is a positive constant of the order of 1. Now consider

$$\dot{\tilde{W}}_1(t) = a_1 \bar{\sigma}_1 u. \qquad \text{(A.10)}$$

Note that setting $u = -y + \frac{\varepsilon_H}{m_s}$ with output given $y = \bar{\sigma}_1^T \tilde{W}_1$ turns (A.10) into (2.26). Set $B = a_1 \bar{\sigma}_1,\ C = \bar{\sigma}_1, x(t) = \tilde{W}_1$ so that (A.6) yields (A.10). Then,

$$\left\| u \right\| \le \left\| y \right\| + \left\| \frac{\varepsilon_H}{m_s} \right\| \le y_{\max} + \varepsilon_{\max} \qquad \text{(A.11)}$$

since $\left\| m_s \right\| \ge 1$. Then,

$$N \equiv \int_t^{t+T} \| B(\tau) \| \cdot \| u(\tau) \| d\tau = \int_t^{t+T} \| a_1 \bar{\sigma}_1(\tau) \| \cdot \| u(\tau) \| d\tau$$

$$\leq a_1(y_{\max} + \varepsilon_{\max}) \int_t^{t+T} \| \bar{\sigma}_1(\tau) \| \, d\tau$$

$$\leq a_1(y_{\max} + \varepsilon_{\max}) \left[\int_t^{t+T} \| \bar{\sigma}_1(\tau) \|^2 \, d\tau \right]^{1/2} \left[\int_t^{t+T} 1 d\tau \right]^{1/2}$$

By using (A.8),

$$N \leq a_1(y_{\max} + \varepsilon_{\max}) \sqrt{\beta_2 \mathrm{T}} \qquad \text{(A.12)}$$

Finally (A.9) and (A.12) yield,

$$\tilde{W}_1(t) \leq \frac{\sqrt{\beta_2 \mathrm{T}}}{\beta_1} \left\{ \left[y_{\max} + \delta \beta_2 a_1 \left(\varepsilon_{\max} + y_{\max} \right) \right] \right\}. \qquad \text{(A.13)}$$

This completes the proof.

■

Proof of Theorem 2.2.

The convergence proof is based on Lyapunov analysis. We consider the Lyapunov function

$$L(t) = V(x) + \frac{1}{2} tr(\tilde{W}_1^T a_1^{-1} \tilde{W}_1) + \frac{1}{2} tr(\tilde{W}_2^T a_2^{-1} \tilde{W}_2). \qquad \text{(A.14)}$$

With the chosen tuning laws one can then show that the errors $\tilde{W}_1$ and $\tilde{W}_2$ are UUB and convergence is obtained.

Hence the derivative of the Lyapunov function is given by

$$\dot{L}(x) = \dot{V}(x) + \tilde{W}_1^T \alpha_1^{-1} \dot{\tilde{W}}_1 + \tilde{W}_2^T \alpha_2^{-1} \dot{\tilde{W}}_2 = \dot{L}_V(x) + \dot{L}_1(x) + \dot{L}_2(x) \qquad \text{(A.15)}$$

First term is,

$$\dot{V}(x) = W_1^T \left(\nabla \phi_1 f(x) - \frac{1}{2} \overline{D}_1(x) \hat{W}_2 \right) + \nabla \varepsilon^T(x) \left(f(x) - \frac{1}{2} g(x) R^{-1} g^T(x) \nabla \phi_1^T \hat{W}_2 \right).$$

Then

$$\dot{V}(x) = W_1^T \left(\nabla \phi_1 f(x) - \frac{1}{2} \overline{D}_1(x) \hat{W}_2 \right) + \varepsilon_1(x)$$

$$= W_1^T \nabla\phi_1 f(x) + \frac{1}{2} W_1^T \overline{D}_1(x)\left(W_1 - \hat{W}_2\right) - \frac{1}{2} W_1^T \overline{D}_1(x) W_1 + \varepsilon_1(x)$$

$$= W_1^T \nabla\phi_1 f(x) + \frac{1}{2} W_1^T \overline{D}_1(x)\tilde{W}_2 - \frac{1}{2} W_1^T \overline{D}_1(x) W_1 + \varepsilon_1(x)$$

$$= W_1^T \sigma_1 + \frac{1}{2} W_1^T \overline{D}_1(x)\tilde{W}_2 + \varepsilon_1(x)$$

where

$$\varepsilon_1(x) \equiv \dot{\varepsilon}(x) = \nabla\varepsilon^T(x)\left(f(x) - \frac{1}{2} g(x) R^{-1} g^T(x) \nabla\phi_1^T(x)\hat{W}_2\right).$$

From the HJB equation

$$W_1^T \sigma_1 = -Q(x) - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \varepsilon_{HJB}(x).$$

Then

$$\dot{L}_V(x) = -Q(x) - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \frac{1}{2} W_1^T \overline{D}_1(x)\tilde{W}_2 + \varepsilon_{HJB}(x) + \varepsilon_1(x) \equiv \dot{\overline{L}}_V(x) + \frac{1}{2} W_1^T \overline{D}_1(x)\tilde{W}_2 + \varepsilon_1(x). \tag{A.16}$$

Second term is,

$$\dot{L}_1 = \tilde{W}_1^T \alpha_1^{-1} \dot{\tilde{W}}_1$$

$$= \tilde{W}_1^T \alpha_1^{-1} \alpha_1 \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2}\left(\sigma_2^T \hat{W}_1 + Q(x) + \frac{1}{4}\hat{W}_2^T \overline{D}_1 \hat{W}_2\right)$$

$$= \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2}(\sigma_2^T \hat{W}_1 + Q(x) + \frac{1}{4}\hat{W}_2^T \overline{D}_1(x)\hat{W}_2 - Q(x) - \sigma_1^T W_1 - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \varepsilon_{HJB}(x))$$

$$= \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2}(\sigma_2^T(x)\hat{W}_1 - \sigma_1^T(x) W_1 + \frac{1}{4}\hat{W}_2^T \overline{D}_1(x)\hat{W}_2 - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \varepsilon_{HJB}(x))$$

$$= \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2}(-\tilde{W}_1^T \left(\nabla\phi_1^T(x)\right)^T f(x) - \frac{1}{2}\hat{W}_2^T \overline{D}_1(x)\hat{W}_1 + \frac{1}{2} W_1^T \overline{D}_1(x) W_1 + \frac{1}{4}\hat{W}_2^T \overline{D}_1(x)\hat{W}_2$$
$$- \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \varepsilon_{HJB}(x))$$

$$= \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2}(-f(x)^T \nabla\phi_1^T(x)\tilde{W}_1 + \frac{1}{2}\hat{W}_2^T \overline{D}_1(x)\tilde{W}_1 + \frac{1}{4}\tilde{W}_2^T \overline{D}_1(x)\tilde{W}_2 + \varepsilon_{HJB}(x)).$$

$$\dot{L}_1 = \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T\sigma_2+1\right)^2}(-\sigma_2^T\tilde{W}_1 + \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)\tilde{W}_2 + \varepsilon_{HJB}(x))$$

$$= \dot{\overline{L}}_1 + \frac{1}{4}\tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T\sigma_2+1\right)^2}\tilde{W}_2^T\overline{D}_1(x)\tilde{W}_2 \qquad \text{(A.17)}$$

where

$$\dot{\overline{L}}_1 = \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T\sigma_2+1\right)^2}\left(-\sigma_2^T\tilde{W}_1 + \varepsilon_{HJB}(x)\right)$$

$$= \tilde{W}_1^T\overline{\sigma}_2\left(-\sigma_2^T\tilde{W}_1 + \frac{\varepsilon_{HJB}(x)}{m_s}\right).$$

Finally by adding the terms (A.16) and (A.17)

$$\dot{L}(x) = -Q(x) - \frac{1}{4}W_1^T\overline{D}_1(x)W_1 + \frac{1}{2}W_1^T\overline{D}_1(x)\tilde{W}_2 + \varepsilon_{HJB}(x) + \varepsilon_1(x)$$
$$+\tilde{W}_1^T\frac{\sigma_2}{\left(\sigma_2^T\sigma_2+1\right)^2}(-\sigma_2^T\tilde{W}_1 + \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)\tilde{W}_2 + \varepsilon_{HJB}(x)) + \tilde{W}_2^T\alpha_2^{-1}\dot{\tilde{W}}_2$$

$$\dot{L}(x) = \dot{\overline{L}}_V + \dot{\overline{L}}_1 + \varepsilon_1(x) - \tilde{W}_2^T\alpha_2^{-1}\dot{\hat{W}}_2 + \frac{1}{2}\tilde{W}_2^T\overline{D}_1(x)W_1 + \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)W_1\frac{\overline{\sigma}_2^T}{m_s}\tilde{W}_1 - \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)W_1\frac{\overline{\sigma}_2^T}{m_s}W_1$$
$$+\frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)\tilde{W}_2\frac{\overline{\sigma}_2^T}{m_s}W_1 + \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)\hat{W}_2\frac{\overline{\sigma}_2^T}{m_s}\hat{W}_1 \qquad \text{(A.18)}$$

where $\overline{\sigma}_2 = \dfrac{\sigma_2}{\sigma_2^T\sigma_2+1}$ and $m_s = \sigma_2^T\sigma_2 + 1$.

In order to select the update law for the action neural network, write (A.18) as

$$\dot{L}(x) = \dot{\overline{L}}_V + \dot{\overline{L}}_1 + \varepsilon_1(x) - \tilde{W}_2^T\left[\alpha_2^{-1}\dot{\hat{W}}_2 - \tfrac{1}{4}\overline{D}_1(x)\hat{W}_2\frac{\overline{\sigma}_2^T}{m_s}\hat{W}_1\right] + \frac{1}{2}\tilde{W}_2^T\overline{D}_1(x)W_1$$
$$+\frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)W_1\frac{\overline{\sigma}_2^T}{m_s}\tilde{W}_1 - \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)W_1\frac{\overline{\sigma}_2^T}{m_s}W_1 + \frac{1}{4}\tilde{W}_2^T\overline{D}_1(x)W_1\frac{\overline{\sigma}_2}{m_s}\tilde{W}_2$$

and we define the actor tuning law as

$$\dot{\hat{W}}_2 = -\alpha_2\left\{\left(F_2\hat{W}_2 - F_1\overline{\sigma}_2^T\hat{W}_1\right) - \frac{1}{4}\overline{D}_1(x)\hat{W}_2 m^T\hat{W}_1\right\}. \qquad \text{(A.19)}$$

This adds to $\dot{L}$ the terms

$$\begin{aligned}
&\tilde{W}_2^T F_2 \hat{W}_2 - \tilde{W}_2^T F_1 \bar{\sigma}_2^T \hat{W}_1 \\
&= \tilde{W}_2^T F_2 (W_1 - \tilde{W}_2) - \tilde{W}_2^T F_1 \bar{\sigma}_2^T (W_1 - \tilde{W}_1) \\
&= \tilde{W}_2^T F_2 W_1 - \tilde{W}_2^T F_2 \tilde{W}_2 - \tilde{W}_2^T F_1 \bar{\sigma}_2^T W_1 + \tilde{W}_2^T F_1 \bar{\sigma}_2^T \tilde{W}_1 .
\end{aligned}$$

Overall

$$\begin{aligned}
\dot{L}(x) &= -Q(x) - \frac{1}{4} W_1^T \bar{D}_1(x) W_1 + \varepsilon_{HJB}(x) + \tilde{W}_1^T \bar{\sigma}_2 \left(-\bar{\sigma}_2^T \tilde{W}_1 + \frac{\varepsilon_{HJB}(x)}{m_s} \right) + \varepsilon_1(x) \\
&+ \frac{1}{2} \tilde{W}_2^T \bar{D}_1(x) W_1 + \frac{1}{4} \tilde{W}_2^T \bar{D}_1(x) W_1 \frac{\bar{\sigma}_2^T}{m_s} \tilde{W}_1 - \frac{1}{4} \tilde{W}_2^T \bar{D}_1(x) W_1 \frac{\bar{\sigma}_2^T}{m_s} W_1 + \frac{1}{4} \tilde{W}_2^T \bar{D}_1(x) W_1 \frac{\bar{\sigma}_2}{m_s} \tilde{W}_2 \quad \text{(A.20)} \\
&+ \tilde{W}_2^T F_2 W_1 - \tilde{W}_2^T F_2 \tilde{W}_2 - \tilde{W}_2^T F_1 \bar{\sigma}_2^T W_1 + \tilde{W}_2^T F_1 \bar{\sigma}_2^T \tilde{W}_1
\end{aligned}$$

Now it is desired to introduce norm bounds. It is easy to show that under the Facts

$$\|\varepsilon_1(x)\| < b_{\varepsilon_x} b_f \|x\| + \tfrac{1}{2} b_{\varepsilon_x} b_g^2 b_{\phi_x} \sigma_{\min}(R) \left(\|W_1\| + \|\tilde{W}_2\| \right)$$

Also, since $Q(x) > 0$ there exists q such that $x^T q x < Q(x)$ for $x \in \Omega$. It is shown in [4] that ε_{HJB} converges to zero uniformly as N increases.

Select $\varepsilon > 0$ and $N_0(\varepsilon)$ such that $\sup_{x \in \Omega} \|\varepsilon_{HJB}\| < \varepsilon$. Then, assuming $N > N_0$ and writing in terms of $\tilde{Z} = \begin{bmatrix} x \\ \bar{\sigma}_2^{\mathrm{T}} \tilde{W}_1 \\ \tilde{W}_2 \end{bmatrix}$ (A.20) becomes

$$\begin{aligned}
\dot{L} &< \frac{1}{4} \|W_1\|^2 \|\bar{D}_1(x)\| + \varepsilon + \tfrac{1}{2} \| W_1 \| b_{\varepsilon_x} b_{\varphi_x} b_g{}^2 \sigma_{\min}(R) \\
&- \tilde{Z}^T \begin{bmatrix} qI & 0 & 0 \\ 0 & I & \left(-\tfrac{1}{2} F_1 - \dfrac{1}{8 m_s} \bar{D}_1 W_1 \right)^T \\ 0 & -\tfrac{1}{2} F_1 - \left(\dfrac{1}{8 m_s} \bar{D}_1 W_1 \right) & F_2 - \tfrac{1}{8} \left(\bar{D}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{D}_1 \right) \end{bmatrix} \tilde{Z}
\end{aligned}$$

$$+\tilde{Z}^T \begin{bmatrix} b_{\varepsilon_x} b_f \\ \dfrac{\varepsilon}{m_s} \\ (\frac{1}{2}\bar{D}_1 + F_2 - F_1\bar{\sigma}_2^T - \frac{1}{4}\bar{D}_1 W_1 m^T)W_1 + \frac{1}{2} b_{\varepsilon_x} b_g^2 b_{\phi_x} \sigma_{\min}(R) \end{bmatrix} \tag{A.21}$$

Define $$M = \begin{bmatrix} qI & 0 & 0 \\ 0 & I & \left(-\frac{1}{2}F_1 - \dfrac{1}{8m_s}\bar{D}_1 W_1\right)^T \\ 0 & -\frac{1}{2}F_1 - \left(\dfrac{1}{8m_s}\bar{D}_1 W_1\right) & F_2 - \frac{1}{8}\left(\bar{D}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{D}_1\right) \end{bmatrix} \tag{A.22}$$

$$d = \begin{bmatrix} b_{\varepsilon_x} b_f \\ \dfrac{\varepsilon}{m_s} \\ (\frac{1}{2}\bar{D}_1 + F_2 - F_1\bar{\sigma}_2^T - \frac{1}{4}\bar{D}_1 W_1 m^T)W_1 + \frac{1}{2} b_{\varepsilon_x} b_g^2 b_{\phi_x} \sigma_{\min}(R) \end{bmatrix}$$

$$c = \frac{1}{4}\|W_1\|^2 \left\|\bar{D}_1(x)\right\| + \varepsilon + \tfrac{1}{2}\| W_1 \| b_{\varepsilon_x} b_{\varphi_x} b_g{}^2 \sigma_{\min}(R)$$

Let the parameters be chosen such that $M > 0$. Now (A.21) becomes

$$\dot{L} < -\left\|\tilde{Z}\right\|^2 \sigma_{\min}(M) + \|d\| \left\|\tilde{Z}\right\| + c + \varepsilon$$

Completing the squares, the Lyapunov derivative is negative if

$$\left\|\tilde{Z}\right\| > \frac{\|d\|}{2\sigma_{\min}(M)} + \sqrt{\frac{d^2}{4\sigma_{\min}^2(M)} + \frac{c+\varepsilon}{\sigma_{\min}(M)}} \equiv B_Z. \tag{A.23}$$

It is now straightforward to demonstrate that if *L* exceeds a certain bound, then, $\dot{L}$ is negative. Therefore, according to the standard Lyapunov extension theorem [42], [50] the analysis above demonstrates that the state and the weights are UUB.

This completes the proof.

■

Proofs for Chapter 3

Proof for Theorem 3.2: The convergence proof is based on Lyapunov analysis.

We consider the Lyapunov function

$$L(t) = V(x) + \frac{1}{2}tr(\tilde{W}_1^T a_1^{-1}\tilde{W}_1) + \frac{1}{2}tr(\tilde{W}_2^T a_2^{-1}\tilde{W}_2) + \frac{1}{2}tr(\tilde{W}_3^T a_3^{-1}\tilde{W}_3). \tag{A.24}$$

The derivative of the Lyapunov function is given by

$$\dot{L}(x) = \dot{V}(x) + \tilde{W}_1^T \alpha_1^{-1}\dot{\tilde{W}}_1 + \tilde{W}_2^T \alpha_2^{-1}\dot{\tilde{W}}_2 + \tilde{W}_3^T \alpha_3^{-1}\dot{\tilde{W}}_3 \tag{A.25}$$

First term is,

$$\dot{V}(x) = W_1^T\left(\nabla\phi_1 f(x) - \frac{1}{2}\overline{D}_1(x)\hat{W}_2 + \frac{1}{2\gamma^2}\overline{E}_1(x)\hat{W}_3\right) + \nabla\varepsilon^T(x)\left(f(x) - \frac{1}{2}g(x)R^{-1}g^T(x)\nabla\phi_1^T\hat{W}_2 + \frac{1}{2\gamma^2}kk^T\nabla\phi_1^T\hat{W}_3\right)$$

Then

$$\begin{aligned}
\dot{V}(x) &= W_1^T\left(\nabla\phi_1 f(x) - \frac{1}{2}\overline{D}_1(x)\hat{W}_2 + \frac{1}{2\gamma^2}\overline{E}_1(x)\hat{W}_3\right) + \varepsilon_1(x) \\
&= W_1^T\nabla\phi_1 f(x) + \frac{1}{2}W_1^T\overline{D}_1(x)\left(W_1 - \hat{W}_2\right) - \frac{1}{2}W_1^T\overline{D}_1(x)W_1 - \frac{1}{2\gamma^2}W_1^T\overline{E}_1(x)\left(W_1 - \hat{W}_3\right) + \frac{1}{2\gamma^2}W_1^T\overline{\mathrm{E}}_1(x)W_1 + \varepsilon_1(x) \\
&= W_1^T\nabla\phi_1 f(x) + \frac{1}{2}W_1^T\overline{D}_1(x)\tilde{W}_2 - \frac{1}{2}W_1^T\overline{D}_1(x)W_1 - \frac{1}{2\gamma^2}W_1^T\overline{E}_1(x)\tilde{W}_3 + \frac{1}{2\gamma^2}W_1^T\overline{\mathrm{E}}_1(x)W_1 + \varepsilon_1(x) \\
&= W_1^T\sigma_1 + \frac{1}{2}W_1^T\overline{D}_1(x)\tilde{W}_2 - \frac{1}{2\gamma^2}W_1^T\overline{E}_1(x)\tilde{W}_3 + \varepsilon_1(x)
\end{aligned}$$

where $\varepsilon_1(x) \equiv \dot{\varepsilon}(x) = \nabla\varepsilon^T(x)(f(x) - \frac{1}{2}g(x)R^{-1}g^T(x)\nabla\phi_1^T\hat{W}_2 + \frac{1}{2\gamma^2}kk^T\nabla\phi_1^T\hat{W}_3)$.

From the HJI equation $W_1^T\sigma_1 = -h^T h - \frac{1}{4}W_1^T\overline{D}_1(x)W_1 + \frac{1}{4\gamma^2}W_1^T\overline{E}_1(x)W_1 + \varepsilon_{HJI}(x)$.

Then

$$\dot{L}_V(x) = -h^T h - \frac{1}{4}W_1^T\overline{D}_1(x)W_1 + \frac{1}{4\gamma^2}W_1^T\overline{D}_1(x)W_1 + \frac{1}{2}W_1^T\overline{E}_1(x)\tilde{W}_2 - \frac{1}{2\gamma^2}W_1^T\overline{E}_1(x)\tilde{W}_3 + \varepsilon_{HJI}(x) + \varepsilon_1(x)$$

$$\equiv \dot{\overline{L}}_V(x) + \frac{1}{2}W_1^T\overline{D}_1(x)\tilde{W}_2 - \frac{1}{2\gamma^2}W_1^T\overline{E}_1(x)\tilde{W}_3 + \varepsilon_1(x). \tag{A.26}$$

where $\dot{\overline{L}}_V(x) = -h^T h - \frac{1}{4}W_1^T\overline{D}_1(x)W_1 + \frac{1}{4\gamma^2}W_1^T\overline{D}_1(x)W_1 + \varepsilon_{HJI}(x) + \varepsilon_1(x)$

Second term is,

$$\dot{L}_1 = \tilde{W}_1^T \alpha_1^{-1} \dot{\tilde{W}}_1 = \tilde{W}_1^T \alpha_1^{-1} \alpha_1 \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2} (\sigma_2^T \hat{W}_1 + Q(x) + \frac{1}{4} \hat{W}_2^T \overline{D}_1 \hat{W}_2 - \frac{1}{4\gamma^2} \hat{W}_3^T \overline{\mathrm{E}}_1 \hat{W}_3)$$

$$\dot{L}_1 = \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2} (-\sigma_2^T \tilde{W}_1 + \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) \tilde{W}_2 - \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{\mathrm{E}}_1 \tilde{W}_3 + \varepsilon_{HJI}(x))$$

$$= \dot{\overline{L}}_1 + \frac{1}{4} \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2} \left(\tilde{W}_2^T \overline{D}_1(x) \tilde{W}_2 - \frac{1}{\gamma^2} \tilde{W}_3^T \overline{\mathrm{E}}_1 \tilde{W}_3 \right) \tag{A.27}$$

where $\dot{\overline{L}}_1 = \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2} \left(-\sigma_2^T \tilde{W}_1 + \varepsilon_{HJI}(x) \right) = \tilde{W}_1^T \overline{\sigma}_2 \left(-\sigma_2^T \tilde{W}_1 + \frac{\varepsilon_{HJI}(x)}{m_s} \right)$.

By adding the terms of (A.26) and (A.27) we have

$$\dot{L}(x) = -Q(x) - \frac{1}{4} W_1^T \overline{D}_1(x) W_1 + \frac{1}{4\gamma^2} W_1^T \overline{E}_1(x) W_1 + \frac{1}{2} W_1^T \overline{D}_1(x) \tilde{W}_2 - \frac{1}{2\gamma^2} W_1^T \overline{E}_1(x) \tilde{W}_3 + \varepsilon_{HJI}(x) + \varepsilon_1(x)$$

$$+ \tilde{W}_1^T \frac{\sigma_2}{\left(\sigma_2^T \sigma_2 + 1\right)^2} (-\sigma_2^T \tilde{W}_1 + \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) \tilde{W}_2 - \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{\mathrm{E}}_1 \tilde{W}_3 + \varepsilon_{HJI}(x)) + \tilde{W}_2^T \alpha_2^{-1} \dot{\tilde{W}}_2 + \tilde{W}_3^T \alpha_3^{-1} \dot{\tilde{W}}_3$$

$$\dot{L}(x) = \dot{\overline{L}}_1 + \dot{\overline{L}}_V(x) + \varepsilon_1(x) - \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) W_1 \frac{\overline{\sigma}_2^T}{m_s} W_1 + \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) W_1 \frac{\overline{\sigma}_2^T}{m_s} \tilde{W}_1$$

$$+ \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) \tilde{W}_2 \frac{\overline{\sigma}_2^T}{m_s} W_1 + \frac{1}{4} \tilde{W}_2^T \overline{D}_1(x) \hat{W}_2 \frac{\overline{\sigma}_2^T}{m_s} \hat{W}_1 - \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{E}_1(x) W_1 \frac{\overline{\sigma}_2^T}{m_s} \tilde{W}_1$$

$$- \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{E}_1(x) \tilde{W}_3 \frac{\overline{\sigma}_2^T}{m_s} W_1 - \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{E}_1(x) \hat{W}_3 \frac{\overline{\sigma}_2^T}{m_s} \hat{W}_1 + \frac{1}{2} \tilde{W}_2^T \overline{D}_1(x) W_1 - \frac{1}{2\gamma^2} \tilde{W}_3^T \overline{\mathrm{E}}_1(x) W_1$$

$$+ \frac{1}{4\gamma^2} \tilde{W}_3^T \overline{E}_1(x) W_1 \frac{\overline{\sigma}_2^T}{m_s} W_1 - \tilde{W}_2^T \alpha_2^{-1} \dot{\hat{W}}_2 - \tilde{W}_3^T \alpha_3^{-1} \dot{\hat{W}}_3 \tag{A.28}$$

where $\overline{\sigma}_2 = \frac{\sigma_2}{\sigma_2^T \sigma_2 + 1}$ and $m_s = \sigma_2^T \sigma_2 + 1$.

In order to select the update law for the action neural networks, write (A.28) as

$$\dot{L}(x) = \dot{\bar{L}}_V + \dot{\bar{L}}_1 + \varepsilon_1(x) - \tilde{W}_2^T\left[\alpha_2^{-1}\dot{\hat{W}}_2 - \tfrac{1}{4}\bar{D}_1(x)\hat{W}_2\frac{\bar{\sigma}_2^T}{m_s}\hat{W}_1\right] - \tilde{W}_3^T\left[\alpha_3^{-1}\dot{\hat{W}}_3 + \tfrac{1}{4\gamma^2}\bar{E}_1(x)\hat{W}_3\frac{\bar{\sigma}_2^T}{m_s}\hat{W}_1\right]$$
$$+\frac{1}{2}\tilde{W}_2^T\bar{D}_1(x)W_1 + \frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}\tilde{W}_1 - \frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}W_1 + \frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2}{m_s}\tilde{W}_2$$
$$-\frac{1}{2\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1 - \frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}\tilde{W}_1 + \frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}W_1 - \frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2}{m_s}\tilde{W}_3$$

Now define the actor tuning law as

$$\dot{\hat{W}}_2 = -\alpha_2\left\{\left(F_2\hat{W}_2 - F_1\bar{\sigma}_2^T\hat{W}_1\right) - \frac{1}{4}\bar{D}_1(x)\hat{W}_2 m^T\hat{W}_1\right\} \tag{A.29}$$

and the disturbance tuning law as

$$\dot{\hat{W}}_3 = -\alpha_3\left\{\left(F_4\hat{W}_3 - F_3\bar{\sigma}_2^T\hat{W}_1\right) + \frac{1}{4\gamma^2}\bar{E}_1(x)\hat{W}_3 m^T\hat{W}_1\right\}. \tag{A.30}$$

This adds to $\dot{L}$ the terms

$$\tilde{W}_2^T F_2 W_1 - \tilde{W}_2^T F_2\tilde{W}_2 - \tilde{W}_2^T F_1\bar{\sigma}_2^T W_1 + \tilde{W}_2^T F_1\bar{\sigma}_2^T\tilde{W}_1 + \tilde{W}_3^T F_4 W_1 - \tilde{W}_3^T F_4\tilde{W}_3 - \tilde{W}_3^T F_3\bar{\sigma}_2^T W_1 + \tilde{W}_3^T F_3\bar{\sigma}_2^T\tilde{W}_1$$

Overall

$$\dot{L}(x) = -Q(x) - \frac{1}{4}W_1^T\bar{D}_1(x)W_1 + \frac{1}{4\gamma^2}W_1^T\bar{E}_1(x)W_1 + \varepsilon_{HJI}(x) + \tilde{W}_1^T\bar{\sigma}_2\left(-\bar{\sigma}_2^T\tilde{W}_1 + \frac{\varepsilon_{HJI}(x)}{m_s}\right) + \varepsilon_1(x)$$
$$+\frac{1}{2}\tilde{W}_2^T\bar{D}_1(x)W_1 + \frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}\tilde{W}_1 - \frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}W_1$$
$$+\frac{1}{4}\tilde{W}_2^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_2}{m_s}\tilde{W}_2 - \frac{1}{2\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1 - \frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2}{m_s}\tilde{W}_3$$
$$-\frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}\tilde{W}_1 + \frac{1}{4\gamma^2}\tilde{W}_3^T\bar{E}_1(x)W_1\frac{\bar{\sigma}_2^T}{m_s}W_1$$
$$+\tilde{W}_2^T F_2 W_1 - \tilde{W}_2^T F_2\tilde{W}_2 - \tilde{W}_2^T F_1\bar{\sigma}_2^T W_1 + \tilde{W}_2^T F_1\bar{\sigma}_2^T\tilde{W}_1 + \tilde{W}_3^T F_4 W_1 - \tilde{W}_3^T F_4\tilde{W}_3 - \tilde{W}_3^T F_3\bar{\sigma}_2^T W_1 + \tilde{W}_3^T F_3\bar{\sigma}_2^T\tilde{W}_1$$

(A.31)

Now it is desired to introduce norm bounds. It is easy to show that under the Facts 3.1

$$\|\varepsilon_1(x)\| < b_{\varepsilon_x} b_f\|x\| + \tfrac{1}{2}b_{\varepsilon_x} b_g^2 b_{\phi_x}\sigma_{\min}(R)\left(\|W_1\| + \|\tilde{W}_2\|\right) + \tfrac{1}{2\gamma^2}b_{\varepsilon_x} b_k^2 b_{\phi_x}\left(\|W_1\| + \|\tilde{W}_3\|\right)$$

Also since $Q(x) > 0$ there exists *q* such that $x^T qx < Q(x)$ locally. It is shown in [3], [5] that ε_{HJI} converges to zero uniformly as N increases.

Select $\varepsilon > 0$ and $N_0(\varepsilon)$ such that $\sup\left\|\varepsilon_{HJI}\right\| < \varepsilon$. Then assuming $N > N_0$ and writing in

terms of $\tilde{Z} = \begin{bmatrix} x \\ \bar{\sigma}_2{}^{\mathrm{T}}\tilde{W}_1 \\ \tilde{W}_2 \\ \tilde{W}_3 \end{bmatrix}$, (A.31) becomes

$$\dot{L} < \frac{1}{4}\left\|W_1\right\|^2 \left\|\bar{D}_1(x)\right\| + \frac{1}{4\gamma^2}\left\|W_1\right\|^2 \left\|\bar{E}_1(x)\right\| + \varepsilon + \tfrac{1}{2}\|W_1\| b_{\varepsilon_x} b_{\varphi_x} b_g{}^2 \sigma_{\min}(R) + \tfrac{1}{2\gamma^2}\|W_1\| b_{\varepsilon_x} b_k^2 b_{\phi_x}$$

$$-\tilde{Z}^T \begin{bmatrix} qI & 0 & 0 & 0 \\ 0 & I & \left(\tfrac{1}{2}F_1 - \dfrac{1}{8m_s}\bar{D}_1 W_1\right)^T & \tfrac{1}{2}F_3 + \left(\dfrac{1}{8\gamma^2 m_s}\bar{\mathrm{E}}_1 W_1\right) \\ 0 & \tfrac{1}{2}F_1 - \left(\dfrac{1}{8m_s}\bar{D}_1 W_1\right) & F_2 - \tfrac{1}{8}\left(\bar{D}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{D}_1\right) & 0 \\ 0 & \tfrac{1}{2}F_3 + \left(\dfrac{1}{8\gamma^2 m_s}\bar{\mathrm{E}}_1 W_1\right) & 0 & F_4 + \tfrac{1}{8\gamma^2}\left(\bar{\mathrm{E}}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{\mathrm{E}}_1\right) \end{bmatrix} \tilde{Z}$$

$$+\tilde{Z}^T \begin{bmatrix} b_{\varepsilon_x} b_f \\ \dfrac{\varepsilon}{m_s} \\ (\tfrac{1}{2}\bar{D}_1 + F_2 - F_1\bar{\sigma}_2^T - \tfrac{1}{4}\bar{D}_1 W_1 m^T)W_1 + \tfrac{1}{2}b_{\varepsilon_x} b_g^2 b_{\phi_x}\sigma_{\min}(R) \\ (-\tfrac{1}{2\gamma^2}\bar{\mathrm{E}}_1 + F_4 - F_3\bar{\sigma}_2^T + \tfrac{1}{4\gamma^2}\bar{\mathrm{E}}_1 W_1 m^T)W_1 + \tfrac{1}{2\gamma^2}b_{\varepsilon_x} b_k^2 b_{\phi_x} \end{bmatrix} \qquad \text{(A.32)}$$

Define

$$M = \begin{bmatrix} qI & 0 & 0 & 0 \\ 0 & I & \left(\tfrac{1}{2}F_1 - \dfrac{1}{8m_s}\bar{D}_1 W_1\right)^T & \tfrac{1}{2}F_3 + \left(\dfrac{1}{8\gamma^2 m_s}\bar{\mathrm{E}}_1 W_1\right) \\ 0 & \tfrac{1}{2}F_1 - \left(\dfrac{1}{8m_s}\bar{D}_1 W_1\right) & F_2 - \tfrac{1}{8}\left(\bar{D}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{D}_1\right) & 0 \\ 0 & \tfrac{1}{2}F_3 + \left(\dfrac{1}{8\gamma^2 m_s}\bar{\mathrm{E}}_1 W_1\right) & 0 & F_4 + \tfrac{1}{8\gamma^2}\left(\bar{\mathrm{E}}_1 W_1 m^{\mathrm{T}} + m W_1^T \bar{\mathrm{E}}_1\right) \end{bmatrix} \qquad \text{(A.33)}$$

$$d = \begin{bmatrix} b_{\varepsilon_x} b_f \\ \dfrac{\varepsilon}{m_s} \\ (\tfrac{1}{2}\bar{D}_1 + F_2 - F_1 \bar{\sigma}_2^T - \tfrac{1}{4}\bar{D}_1 W_1 m^T) W_1 + \tfrac{1}{2} b_{\varepsilon_x} b_g^2 b_{\phi_x} \sigma_{\min}(R) \\ (-\tfrac{1}{2\gamma^2}\bar{\mathrm{E}}_1 + F_4 - F_3 \bar{\sigma}_2^T + \tfrac{1}{4\gamma^2}\bar{\mathrm{E}}_1 W_1 m^T) W_1 + \tfrac{1}{2\gamma^2} b_{\varepsilon_x} b_k^2 b_{\phi_x} \end{bmatrix}$$

$$c = \frac{1}{4}\|W_1\|^2 \left\|\bar{D}_1(x)\right\| + \frac{1}{4\gamma^2}\|W_1\|^2 \left\|\bar{E}_1(x)\right\| + \tfrac{1}{2} \| W_1 \| b_{\varepsilon_x} b_{\varphi_x} b_g{}^2 \sigma_{\min}(R) + \tfrac{1}{2\gamma^2} \| W_1 \| b_{\varepsilon_x} b_k^2 b_{\phi_x}$$

Let the parameters F_1, F_2, F_3 and F_4 be chosen such that $M > 0$. To justify this, the matrix *M* is written in compact form as

$$M = \begin{bmatrix} q & 0 & 0 \\ 0 & I & M_{23} \\ 0 & M_{32} & M_{33} \end{bmatrix} \tag{A.34}$$

where in order to be positive definite the following properties must hold

a) $q > 0$

b) $I > 0$

c) Shur complement for I is

$$D_{22} = I - M_{23} M_{33}^{-1} M_{32} > 0 \tag{A.35}$$

D_{22} can be made positive definite by selecting $F_2 \gg F_1,\ \ F_4 \gg F_3$, since W_1 is bounded above by $W_{\max}$.

Now (A.32) becomes

$$\dot{L} < -\left\|\tilde{Z}\right\|^2 \sigma_{\min}(M) + \|d\|\left\|\tilde{Z}\right\| + c + \varepsilon$$

Completing the squares, the Lyapunov derivative is negative if

$$\left\|\tilde{Z}\right\| > \frac{\|d\|}{2\sigma_{\min}(M)} + \sqrt{\frac{d^2}{4\sigma_{\min}^2(M)} + \frac{c+\varepsilon}{\sigma_{\min}(M)}} \equiv B_Z. \tag{A.36}$$

It is now straightforward to demonstrate that if *L* exceeds a certain bound, then, $\dot{L}$ is negative. Therefore, according to the standard Lyapunov extension theorem [50] the analysis above demonstrates that the state and the weights are UUB.

To show this from (A.24), one has,

$$\sigma_{\min}(P)\|x\|^2+\frac{1}{2a_1}\|\tilde{W}_1\|^2+\frac{1}{2a_2}\|\tilde{W}_2\|^2+\frac{1}{2a_3}\|\tilde{W}_3\|^2\le L\le\sigma_{\max}(P)\|x\|^2+\frac{1}{2a_1}\|\tilde{W}_1\|^2+\frac{1}{2a_2}\|\tilde{W}_2\|^2+\frac{1}{2a_3}\|\tilde{W}_3\|^2 \quad \text{(A.35)}$$

$$\tilde{Z}^T\underbrace{\begin{bmatrix}\sigma_{\min}(P) & & & \\ & \dfrac{1}{2\|\bar{\sigma}_2\|^2 a_1} & & \\ & & 1/2a_2 & \\ & & & 1/2a_3\end{bmatrix}}_{S_1}\tilde{Z}\le L\le\tilde{Z}^T\underbrace{\begin{bmatrix}\sigma_{\max}(P) & & & \\ & \dfrac{1}{2\|\bar{\sigma}_2\|^2 a_1} & & \\ & & 1/2a_2 & \\ & & & 1/2a_3\end{bmatrix}}_{S_2}\tilde{Z} \quad \text{(A.37)}$$

Equation (A.37) is equivalent to

$$\tilde{Z}^T\sigma_{\min}(S_1)\tilde{Z}\le L\le\tilde{Z}^T\sigma_{\max}(S_2)\tilde{Z}$$

Then

$$\sigma_{\min}(S_1)\|\tilde{Z}\|^2\le L\le\sigma_{\max}(S_2)\|\tilde{Z}\|^2 .$$

Therefore,
$$L>\sigma_{\max}(S_2)\left(\frac{\|d\|}{2\sigma_{\min}(M)}+\sqrt{\frac{\|d\|^2}{4\sigma_{\min}^2(M)}+\frac{c+\bar{\varepsilon}_1+\bar{\varepsilon}_2}{\sigma_{\min}(M)}}\right)^2 \quad \text{(A.38)}$$

implies (A.36).

Note that condition (A.36) holds if the norm of any component of $\tilde{Z}$ exceeds the bound, i.e. specifically $x>B_Z$ or $\bar{\sigma}_2{}^{\mathrm{T}}\tilde{W}_1>B_Z$ or $\tilde{W}_2>B_Z$ or $\tilde{W}_3>B_Z$ [42].

Now consider the error dynamics and the output as in Technical Lemmas 3.1, 3.2 and assume $\bar{\sigma}_2$ is persistently exciting

$$\dot{\tilde{W}}_1 = -a_1\bar{\sigma}_2\bar{\sigma}_2^{\mathrm{T}}\tilde{W}_1 + a_1\bar{\sigma}_2\frac{\varepsilon_{HJI}}{m_s} + \frac{a_1}{4m_s^2}\tilde{W}_2^T\bar{D}_1(x)\tilde{W}_2 - \frac{a_1}{4\gamma^2 m_s^2}\tilde{W}_3^T\bar{\mathrm{E}}_1\tilde{W}_3$$

$$y = \bar{\sigma}_2^T\tilde{W}_1\,. \tag{A.39}$$

Then Theorem 3.1 is true with

$$\left\|\frac{\sigma_2^T}{m_s}\tilde{W}_1\right\| > \varepsilon_{\max} > \frac{1}{4}\left\|\tilde{W}_2\right\|^2\left\|\frac{\bar{D}_1}{m_s}\right\| - \frac{1}{4\gamma^2}\left\|\tilde{W}_3\right\|^2\left\|\frac{\bar{\mathrm{E}}_1}{m_s}\right\| + \varepsilon\left\|\frac{1}{m_s}\right\| \tag{A.40}$$

This provides an effective practical bound for $\left\|\bar{\sigma}_2^T\tilde{W}_1\right\|$.

This completes the proof.

■

Proofs for Chapter 4

Proof for Theorem 4.2: The convergence proof is based on Lyapunov analysis.

We consider the Lyapunov function

$$L(t) = V_1(x) + V_2(x) + \frac{1}{2}\tilde{W}_1^T a_1^{-1}\tilde{W}_1 + \frac{1}{2}\tilde{W}_2^T a_2^{-1}\tilde{W}_2 + \frac{1}{2}\tilde{W}_3^T a_3^{-1}\tilde{W}_3 + \frac{1}{2}\tilde{W}_4^T a_4^{-1}\tilde{W}_4. \tag{A.41}$$

where $V_1(x)$ and $V_2(x)$ are the approximate solutions to (4.10) and are given by (4.25) and (4.26) respectively.

The time derivative of the Lyapunov function is given by

$$\dot{L}(x) = \dot{V}_1(x) + \dot{V}_2(x) + \tilde{W}_1^T a_1^{-1}\dot{\tilde{W}}_1 + \tilde{W}_2^T a_2^{-1}\dot{\tilde{W}}_2 + \tilde{W}_3^T a_3^{-1}\dot{\tilde{W}}_3 + \tilde{W}_4^T a_4^{-1}\dot{\tilde{W}}_4 \tag{A.42}$$

Next we will evaluate each one of the terms of $\dot{L}(x)$. First term is, differentiating (4.25), and adding and subtracting $\frac{1}{2}W_1^T\bar{E}_2(x)W_2$ and $\frac{1}{2}W_1^T\bar{D}_1(x)W_1$

$$\begin{aligned}\dot{V}_1(x) &= W_1^T\left(\nabla\phi_1 f(x) - \frac{1}{2}\bar{D}_1(x)\hat{W}_3 - \frac{1}{2}\bar{E}_2(x)\hat{W}_4\right) + \nabla\varepsilon_1^T(x)\left(f(x) - \frac{1}{2}g(x)R_{11}^{-1}g^T(x)\nabla\phi_1^T\hat{W}_3 - \frac{1}{2}kR_{22}^{-1}k^T\nabla\phi_2^T\hat{W}_4\right)\\ &= W_1^T\nabla\phi_1 f(x) + \frac{1}{2}W_1^T\bar{D}_1(x)(W_1 - \hat{W}_3) + \frac{1}{2}W_1^T\bar{E}_2(x)(W_2 - \hat{W}_4) - \frac{1}{2}W_1^T\bar{D}_1(x)W_1 - \frac{1}{2}W_1^T\bar{E}_2(x)W_2 + \dot{\varepsilon}_1(x)\\ &= W_1^T\sigma_1 + \frac{1}{2}W_1^T\bar{D}_1(x)\tilde{W}_3 + \frac{1}{2}W_1^T\bar{E}_2(x)\tilde{W}_4 + \dot{\varepsilon}_1(x)\end{aligned}$$

where

$\bar{E}_2(x) \equiv \nabla\phi_1(x)kR_{22}{}^{-1}k^T\nabla\phi_2{}^T(x)$, $\sigma_1 = \nabla\phi_1(x)(f + gu_1 + kd_2)$, $\bar{D}_1(x) \equiv \nabla\phi_1(x)g(x)R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T(x)$ and

$$\dot{\varepsilon}_1(x) = \nabla\varepsilon_1{}^T(x) \quad \left(f(x) - \frac{1}{2}g(x)R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\hat{W}_3 - \frac{1}{2}kR_{22}{}^{-1}k^T\nabla\phi_2{}^T\hat{W}_4 \right) \tag{A.43}$$

From (4.23) we have

$$W_1{}^T\sigma_1 = -Q_1(x) - \tfrac{1}{4}W_1{}^T\nabla\varphi_1 g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1{}^TW_1 - \tfrac{1}{4}W_2{}^T\nabla\varphi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varphi_2{}^TW_2 + \varepsilon_{HJ_1}$$

Similarly for the second term

$$\dot{V}_2(x) = W_2{}^T\sigma_2 + \frac{1}{2}W_2{}^T\bar{E}_1(x)\tilde{W}_3 + \frac{1}{2}W_2{}^T\bar{D}_2(x)\tilde{W}_4 + \dot{\varepsilon}_2(x)$$

where $\bar{E}_1(x) \equiv \nabla\phi_2(x)g(x)R_{11}{}^{-1}g(x)^T\nabla\phi_1{}^T(x)$, $\sigma_2 = \nabla\phi_2(x)(f + gu_1 + kd_2)$ $\bar{D}_2(x) \equiv \nabla\phi_2(x)kR_{22}{}^{-1}k^T\nabla\phi_2{}^T(x)$

and

$$\dot{\varepsilon}_2(x) = \nabla\varepsilon_2{}^T(x) \quad \left(f(x) - \frac{1}{2}g(x)R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\hat{W}_3 - \frac{1}{2}kR_{22}{}^{-1}k^T\nabla\phi_2{}^T\hat{W}_4 \right) \tag{A.44}$$

From (4.24)

$$W_2{}^T\sigma_2 = -Q_2(x) - \tfrac{1}{4}W_1{}^T\nabla\varphi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\varphi_1{}^TW_1 - \tfrac{1}{4}W_2{}^T\nabla\varphi_2 k(x)R_{22}^{-1}k^T(x)\nabla\varphi_2{}^TW_2 + \varepsilon_{HJ_2}$$

Then we add $\dot{V}_1(x)$ and $\dot{V}_2(x)$

$$\begin{aligned}
\dot{L}_v \equiv \dot{V}_1(x) + \dot{V}_2(x) = {} & -Q_1(x) - \tfrac{1}{4}W_1{}^T\nabla\varphi_1 g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1{}^TW_1 \\
& - \tfrac{1}{4}W_2{}^T\nabla\varphi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varphi_2{}^TW_2 + \varepsilon_{HJ_1} + \frac{1}{2}W_1{}^T\bar{D}_1(x)\tilde{W}_3 + \frac{1}{2}W_1{}^T\bar{E}_2(x)\tilde{W}_4 + \dot{\varepsilon}_1(x) \\
& - Q_2(x) - \tfrac{1}{4}W_1{}^T\nabla\varphi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\varphi_1{}^TW_1 - \tfrac{1}{4}W_2{}^T\nabla\varphi_2 k(x)R_{22}^{-1}k^T(x)\nabla\varphi_2{}^TW_2 + \varepsilon_{HJ_2} \\
& + \frac{1}{2}W_2{}^T\bar{E}_1(x)\tilde{W}_3 + \frac{1}{2}W_2{}^T\bar{D}_2(x)\tilde{W}_4 + \dot{\varepsilon}_2(x)
\end{aligned}$$

$$\dot{L}_v \equiv \dot{\bar{L}}_v(x) + \frac{1}{2}W_1{}^T\bar{D}_1(x)\tilde{W}_3 + \frac{1}{2}W_1{}^T\bar{E}_2(x)\tilde{W}_4 + \frac{1}{2}W_2{}^T\bar{E}_1(x)\tilde{W}_3 + \frac{1}{2}W_2{}^T\bar{D}_2(x)\tilde{W}_4 + \dot{\varepsilon}_1(x) + \dot{\varepsilon}_2(x) \tag{A.45}$$

where

$$\dot{\bar{L}}_v = -Q_1(x) - \tfrac{1}{4}W_1^T\nabla\varphi_1 g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1^T W_1 - \tfrac{1}{4}W_2^T\nabla\varphi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varphi_2^T W_2 + \varepsilon_{HJ_1}$$
$$-Q_2(x) - \tfrac{1}{4}W_1^T\nabla\varphi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\varphi_1^T W_1 - \tfrac{1}{4}W_2^T\nabla\varphi_2 k(x)R_{22}^{-1}k^T(x)\nabla\varphi_2^T W_2 + \varepsilon_{HJ_2}$$

Using the tuning law (4.42) for the first critic and the definitions for the parameter errors (4.41), the third term becomes

$$\dot{L}_1 = \tilde{W}_1^T\frac{\sigma_3}{(\sigma_3^T\sigma_3+1)^2}(-\sigma_3^T\tilde{W}_1 + \tfrac{1}{2}W_1^T\bar{E}_2(x)\tilde{W}_4 + \tfrac{1}{4}\tilde{W}_3^T\bar{D}_1(x)\tilde{W}_3 + \tfrac{1}{4}\tilde{W}_4^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4$$
$$-\tfrac{1}{2}W_2^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4 + \varepsilon_{HJ_1}(x))$$

Finally by rearranging and grouping the terms

$$\dot{L}_1 = \dot{\bar{L}}_1 + \frac{1}{4}\tilde{W}_1^T\frac{\sigma_3}{(\sigma_3^T\sigma_3+1)^2}[\tilde{W}_3^T\bar{D}_1(x)\tilde{W}_3 + \tilde{W}_4^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4$$
$$+2(W_1^T\bar{E}_2(x)\tilde{W}_4 - W_2^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4)] \qquad \text{(A.46)}$$

where $\dot{\bar{L}}_1 = \tilde{W}_1^T\bar{\sigma}_3(-\bar{\sigma}_3^T\tilde{W}_1 + \frac{\varepsilon_{HJ_1}(x)}{m_{s_1}})$, $\bar{\sigma}_3 = \frac{\sigma_3}{\sigma_3^T\sigma_3+1}$ and $m_{s_1} = \sigma_3^T\sigma_3+1$.

Similarly by using the tuning law (4.43) for the second critic and the definitions for the parameter errors (4.41), the fourth term becomes

$$\dot{L}_2 = \dot{\bar{L}}_2 + \frac{1}{4}\tilde{W}_2^T\frac{\sigma_4}{(\sigma_4^T\sigma_4+1)^2}[\tilde{W}_4^T\bar{D}_2(x)\tilde{W}_4 + \tilde{W}_3^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T\tilde{W}_3$$
$$+2(\tilde{W}_3^T\bar{E}_1W_2 - \tilde{W}_3^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T W_1)] \qquad \text{(A.47)}$$

where $\dot{\bar{L}}_2 = \tilde{W}_2^T\bar{\sigma}_4(-\bar{\sigma}_4^T\tilde{W}_2 + \frac{\varepsilon_{HJ_2}(x)}{m_{s_2}})$, $\bar{\sigma}_4 = \frac{\sigma_4}{\sigma_4^T\sigma_4+1}$ and $m_{s_2} = \sigma_4^T\sigma_4+1$.

Finally we need to add the terms of (A.45), (A.46) and (A.47), but in order to select the update laws for the action NNs, we group together the terms of $\tilde{W}_3$ and $\tilde{W}_4$ that are multiplied with the estimated values (two last terms)

$$
\begin{aligned}
\dot{L}(x) = {} & \dot{\bar{L}}_v(x) + \dot{\bar{L}}_1 + \dot{\bar{L}}_2 + \dot{\varepsilon}_1(x) + \dot{\varepsilon}_2(x) + \frac{1}{2}W_2{}^T\bar{D}_2(x)\tilde{W}_4 + \frac{1}{2}W_1{}^T\bar{D}_1(x)\tilde{W}_3 + \frac{1}{2}W_1{}^T\bar{E}_2(x)\tilde{W}_4 + \frac{1}{2}W_2{}^T\bar{E}_1(x)\tilde{W}_3 \\
& + \frac{1}{2}W_1{}^T\bar{E}_2(x)\tilde{W}_4\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\tilde{W}_1 + \frac{1}{2}W_2{}^T\bar{E}_1\tilde{W}_3\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\tilde{W}_2 - \frac{1}{2}W_2{}^T\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\tilde{W}_4\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\tilde{W}_1 \\
& - \frac{1}{2}W_1{}^T\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\tilde{W}_3\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\tilde{W}_2 + \frac{1}{4}\tilde{W}_3{}^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\tilde{W}_1 - \frac{1}{4}\tilde{W}_3{}^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_3{}^T}{m_{s_1}}W_1 \\
& + \frac{1}{4}\tilde{W}_3{}^T\bar{D}_1(x)\tilde{W}_3\frac{\bar{\sigma}_3{}^T}{m_{s_1}}W_1 \quad - \frac{1}{4}\tilde{W}_4{}^T\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T W_2\frac{\bar{\sigma}_3{}^T}{m_{s_1}}W_1 \\
& + \frac{1}{4}\tilde{W}_4{}^T\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T W_2\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\tilde{W}_1 \\
& + \frac{1}{4}\tilde{W}_4{}^T\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\tilde{W}_4\frac{\bar{\sigma}_3{}^T}{m_{s_1}}W_1 + \frac{1}{4}\tilde{W}_4{}^T\bar{D}_2(x)W_2\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\tilde{W}_2 - \frac{1}{4}\tilde{W}_4{}^T\bar{D}_2(x)W_2\frac{\bar{\sigma}_4{}^T}{m_{s_2}}W_2 \\
& + \frac{1}{4}\tilde{W}_4{}^T\bar{D}_2(x)\tilde{W}_4\frac{\bar{\sigma}_4{}^T}{m_{s_2}}W_2 + \frac{1}{4}\tilde{W}_3{}^T\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T W_1\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\tilde{W}_2 \\
& - \frac{1}{4}\tilde{W}_3{}^T\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T W_1\frac{\bar{\sigma}_4{}^T}{m_{s_2}}W_2 + \frac{1}{4}\tilde{W}_3{}^T\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\tilde{W}_3\frac{\bar{\sigma}_4{}^T}{m_{s_2}}W_2 \\
& - \tilde{W}_3{}^T[a_3^{-1}\dot{\hat{W}}_3 - \frac{1}{4}\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\hat{W}_3\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\hat{W}_2 - \frac{1}{4}\bar{D}_1(x)\hat{W}_3\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\hat{W}_1] \\
& - \tilde{W}_4{}^T[a_4^{-1}\dot{\hat{W}}_4 - \frac{1}{4}\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\hat{W}_4\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\hat{W}_1 - \frac{1}{4}\bar{D}_2(x)\hat{W}_4\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\hat{W}_2]
\end{aligned}
$$

In order for the last two terms to be zero we define the actor tuning law for the first player as

$$
\dot{\hat{W}}_3 = -a_3\{(F_2\hat{W}_3 - F_1\bar{\sigma}_3^T\hat{W}_1) - \frac{1}{4}\bar{D}_1(x)\hat{W}_3\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\hat{W}_1 - \frac{1}{4}\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T\hat{W}_3\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\hat{W}_2\}
$$

and the second player’s actor tuning law is defined as

$$
\dot{\hat{W}}_4 = -a_4\{(F_4\hat{W}_4 - F_3\bar{\sigma}_4^T\hat{W}_2) - \frac{1}{4}\bar{D}_2(x)\hat{W}_4\frac{\bar{\sigma}_4{}^T}{m_{s_2}}\hat{W}_2 - \frac{1}{4}\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\hat{W}_4\frac{\bar{\sigma}_3{}^T}{m_{s_1}}\hat{W}_1\}
$$

But this adds to $\dot{L}$ the following terms

$$
\tilde{W}_3^T F_2 W_1 - \tilde{W}_3^T F_2\tilde{W}_3 - \tilde{W}_3^T F_1\bar{\sigma}_3^T W_1 + \tilde{W}_3^T F_1\bar{\sigma}_3^T\tilde{W}_1 + \tilde{W}_4^T F_4 W_2 - \tilde{W}_4^T F_4\tilde{W}_4 - \tilde{W}_4^T F_3\bar{\sigma}_4^T W_2 + \tilde{W}_4^T F_3\bar{\sigma}_4^T\tilde{W}_2
$$

Overall

$$
\begin{aligned}
\dot{L}(x) = & -Q_1(x) - \tfrac{1}{4}W_1^T\nabla\varphi_1\, g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1^T W_1 - \tfrac{1}{4}W_2^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T W_2 + \varepsilon_{HJB_1} \\
& - Q_2(x) - \tfrac{1}{4}W_1^T\nabla\phi_1\, g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T W_1 - \tfrac{1}{4}W_2^T\nabla\phi_2 k(x)R_{22}^{-1}k^T(x)\nabla\phi_2^T W_2 + \varepsilon_{HJB_2} \\
& + \tilde{W}_1^T\bar{\sigma}_3\,(-\sigma_3^T\tilde{W}_1 + \frac{\varepsilon_{HJ_1}(x)}{m_{s_1}}) + \dot{\varepsilon}_1(x) + \tilde{W}_2^T\bar{\sigma}_4\,(-\sigma_4^T\tilde{W}_2 + \frac{\varepsilon_{HJ_2}(x)}{m_{s_2}}) + \dot{\varepsilon}_2(x) \\
& + \frac{1}{2}W_1^T\bar{D}_1(x)\tilde{W}_3 + \frac{1}{2}W_1^T\bar{E}_2(x)\tilde{W}_4 + \frac{1}{2}W_2^T\bar{E}_1(x)\tilde{W}_3 + \frac{1}{2}W_2^T\bar{D}_2(x)\tilde{W}_4 \\
& + \frac{1}{2}W_1^T\bar{E}_2(x)\tilde{W}_4\frac{\bar{\sigma}_3^T}{m_{s_1}}\tilde{W}_1 + \frac{1}{2}W_2^T\bar{E}_1\tilde{W}_3\frac{\bar{\sigma}_4^T}{m_{s_2}}\tilde{W}_2 - \frac{1}{2}W_2^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4\frac{\bar{\sigma}_3^T}{m_{s_1}}\tilde{W}_1 \\
& - \frac{1}{2}W_1^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T\tilde{W}_3\frac{\bar{\sigma}_4^T}{m_{s_2}}\tilde{W}_2 + \frac{1}{4}\tilde{W}_3^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_3^T}{m_{s_1}}\tilde{W}_1 - \frac{1}{4}\tilde{W}_3^T\bar{D}_1(x)W_1\frac{\bar{\sigma}_3^T}{m_{s_1}}W_1 \\
& + \frac{1}{4}\tilde{W}_3^T\bar{D}_1(x)\tilde{W}_3\frac{\bar{\sigma}_3^T}{m_{s_1}}W_1 + \frac{1}{4}\tilde{W}_4^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T W_2\frac{\bar{\sigma}_3^T}{m_{s_1}}\tilde{W}_1 \\
& - \frac{1}{4}\tilde{W}_4^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T W_2\frac{\bar{\sigma}_3^T}{m_{s_1}}W_1 + \frac{1}{4}\tilde{W}_4^T\nabla\phi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\phi_2^T\tilde{W}_4\frac{\bar{\sigma}_3^T}{m_{s_1}}W_1 \\
& + \frac{1}{4}\tilde{W}_4^T\bar{D}_2(x)W_2\frac{\bar{\sigma}_4^T}{m_{s_2}}\tilde{W}_2 - \frac{1}{4}\tilde{W}_4^T\bar{D}_2(x)W_2\frac{\bar{\sigma}_4^T}{m_{s_2}}W_2 \\
& + \frac{1}{4}\tilde{W}_4^T\bar{D}_2(x)\tilde{W}_4\frac{\bar{\sigma}_4^T}{m_{s_2}}W_2 + \frac{1}{4}\tilde{W}_3^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T W_1\frac{\bar{\sigma}_4^T}{m_{s_2}}\tilde{W}_2 \\
& - \frac{1}{4}\tilde{W}_3^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T W_1\frac{\bar{\sigma}_4^T}{m_{s_2}}W_2 + \frac{1}{4}\tilde{W}_3^T\nabla\phi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\phi_1^T\tilde{W}_3\frac{\bar{\sigma}_4^T}{m_{s_2}}W_2 \\
& + \tilde{W}_3^T F_2 W_1 - \tilde{W}_3^T F_2\tilde{W}_3 - \tilde{W}_3^T F_1\bar{\sigma}_3^T W_1 + \tilde{W}_3^T F_1\bar{\sigma}_3^T\tilde{W}_1 + \tilde{W}_4^T F_4 W_2 - \tilde{W}_4^T F_4\tilde{W}_4 - \tilde{W}_4^T F_3\bar{\sigma}_4^T W_2 + \tilde{W}_4^T F_3\bar{\sigma}_4^T\tilde{W}_2 \qquad \text{(A.48)}
\end{aligned}
$$

Now it is desired to introduce norm bounds. It is easy to show that under the Facts 4.1 equations (A.43) and (A.44) become

$$\left\|\dot{\varepsilon}_1(x)\right\| < b_{\varepsilon_{1x}} b_f \left\|x\right\| + \tfrac{1}{2} b_{\varepsilon_{1x}} b_g^2 b_{\phi_{1x}} \sigma_{\min}(R_{11})\left(\left\|W_1\right\| + \left\|\tilde{W}_3\right\|\right) + \tfrac{1}{2} b_{\varepsilon_{1x}} b_k^2 b_{\phi_{2x}} \sigma_{\min}(R_{22})\left(\left\|W_2\right\| + \left\|\tilde{W}_4\right\|\right)$$

$$\left\|\dot{\varepsilon}_2(x)\right\| < b_{\varepsilon_{2x}} b_f \left\|x\right\| + \tfrac{1}{2} b_{\varepsilon_{2x}} b_g^2 b_{\phi_{1x}} \sigma_{\min}(R_{11})\left(\left\|W_1\right\| + \left\|\tilde{W}_3\right\|\right) + \tfrac{1}{2} b_{\varepsilon_{2x}} b_k^2 b_{\phi_{2x}} \sigma_{\min}(R_{22})\left(\left\|W_2\right\| + \left\|\tilde{W}_4\right\|\right)$$

Also since $Q_1(x) > 0$ and $Q_2(x) > 0$ there exists q_1 and q_2 such that $x^T q_1 x < Q_1(x)$ and $x^T q_2 x < Q_2(x)$ for $x \in \Omega$.

Select $\bar{\varepsilon}_1 > 0$, $\bar{\varepsilon}_2 > 0$ and $K_0(\bar{\varepsilon}_1)$, $K_0(\bar{\varepsilon}_2)$ such that $\sup_{x\in\Omega}\left\|\varepsilon_{HJ_1}\right\| < \bar{\varepsilon}_1$ and $\sup_{x\in\Omega}\left\|\varepsilon_{HJ_2}\right\| < \bar{\varepsilon}_2$. Then

assuming $K > K_0$ and writing in terms of $\tilde{Z} = \begin{bmatrix} x \\ \bar{\sigma}_3^{\mathrm{T}}\tilde{W}_1 \\ \bar{\sigma}_4^{\mathrm{T}}\tilde{W}_2 \\ \tilde{W}_3 \\ \tilde{W}_4 \end{bmatrix}$ and known bounds (under Facts 4.1),

(A.48) becomes

$$\begin{aligned}
\dot{L}(x) = {} & \tfrac{1}{4}\|W_1\|^2\left\|\nabla\varphi_1 g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1{}^T\right\| + \tfrac{1}{4}\|W_2\|^2\left\|\nabla\varphi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varphi_2{}^T\right\| + \bar{\varepsilon}_1 \\
& + \tfrac{1}{4}\|W_1\|^2\left\|\nabla\varphi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\varphi_1{}^T\right\| + \tfrac{1}{4}\|W_2\|^2\left\|\nabla\varphi_2 k(x)R_{22}^{-1}k^T(x)\nabla\varphi_2{}^T\right\| + \bar{\varepsilon}_2 \\
& + \tfrac{1}{2}b_{\varepsilon_{x_1}}b_g^2 b_{\phi_{1x}}\sigma_{\min}(R_{11})\|W_1\| + \tfrac{1}{2}b_{\varepsilon_{x_1}}b_k^2 b_{\phi_{2x}}\sigma_{\min}(R_{22})\|W_2\| + \tfrac{1}{2}b_{\varepsilon_{x2}}b_g^2 b_{\phi_{1x}}\sigma_{\min}(R_{11})\|W_1\| + \tfrac{1}{2}b_{\varepsilon_{x2}}b_k^2 b_{\phi_{2x}}\sigma_{\min}(R_{22})\|W_2\|
\end{aligned}$$

$$-\tilde{Z}^T\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} \end{bmatrix}\tilde{Z} + \tilde{Z}\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \qquad \text{(A.49)}$$

Where the components of the matrix *M* are given by

$$\begin{aligned}
& m_{11} = q_1 + q_2 \\
& m_{22} = m_{33} = I \\
& m_{44} = F_2 - \frac{1}{8}(\bar{D}_1 W_1 m_1{}^T + m_1 W_1{}^T\bar{D}_1 + \nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T W_2 m_2{}^T \\
& \qquad + m_2 W_2{}^T\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T)
\end{aligned}$$

$$\begin{aligned}
& m_{55} = F_4 - \frac{1}{8}(\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T W_1 m_1{}^T \\
& \qquad + m_1 W_1{}^T\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T + m_2 W_2{}^T\bar{D}_2(x) + \bar{D}_2(x)W_2 m_2{}^T)
\end{aligned}$$

$$m_{42} = -\frac{1}{2}F_1 - \frac{1}{8m_{s_1}}\bar{D}_1 W_1 = m_{24}{}^T$$

$$m_{52} = -\frac{1}{8m_{s_1}}\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T W_2 - \frac{1}{4m_{s_1}}\bar{E}_2 W_1 = m_{25}{}^T$$

$$m_{43} = -\frac{1}{4m_{s_2}}\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T W_1 - \frac{1}{8m_{s_2}}\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^T W_1 = m_{34}{}^T$$

$$m_{53} = -\frac{1}{2}F_3 - \frac{1}{8m_{s_2}}\bar{D}_2(x)W_2 = m_{53}{}^T$$

$$m_{12} = m_{21} = m_{31} = m_{13} = m_{32} = m_{23} = m_{41} = m_{14} = m_{51} = m_{15} = m_{54} = m_{45} = 0$$

and the components of vector *D* are given

$$d_1 = (b_{\varepsilon_{x1}} + b_{\varepsilon_{x2}})b_f$$

$$d_2 = \frac{\varepsilon_{HJ_1}}{m_{s_1}}$$

$$d_3 = \frac{\varepsilon_{HJ_2}}{m_{s_2}}$$

$$\begin{aligned} d_4 = {} & F_2W_1 - F_1\bar{\sigma}_3^T W_1 + \frac{1}{2}\bar{D}_1W_1 + \frac{1}{2}\bar{E}_1W_2 + \tfrac{1}{2}b_{\varepsilon_{x1}} b_g^2 b_{\phi_{1x}} \sigma_{\min}(R_{11}) - \frac{1}{4}\bar{D}_1W_1m_1{}^TW_1 \\ & - \frac{1}{4}\nabla\phi_1 g(x)R_{11}{}^{-T}R_{21}R_{11}{}^{-1}g^T(x)\nabla\phi_1{}^TW_1m_2{}^TW_2 \end{aligned}$$

$$\begin{aligned} d_5 = {} & F_4W_2 - F_3\bar{\sigma}_4^T W_2 + \frac{1}{2}\bar{D}_2W_2 + \frac{1}{2}\bar{E}_2W_1 \\ & + \tfrac{1}{2}b_{\varepsilon_{x1}} b_k^2 b_{\phi_{2x}} \sigma_{\min}(R_{22}) - \frac{1}{4}\bar{D}_2(x)W_2m_2{}^TW_2 - \frac{1}{4}\nabla\phi_2 k(x)R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^TW_2m_1{}^TW_1 \end{aligned}$$

Define

$$\begin{aligned} c = {} & \tfrac{1}{4}\|W_1\|^2 \left\|\nabla\varphi_1 g(x)R_{11}^{-1}g^T(x)\nabla\varphi_1{}^T\right\| + \tfrac{1}{4}\|W_2\|^2\left\|\nabla\varphi_2 k(x)R_{22}^{-T}R_{12}R_{22}^{-1}k^T(x)\nabla\varphi_2{}^T\right\| \\ & + \tfrac{1}{4}\|W_1\|^2\left\|\nabla\varphi_1 g(x)R_{11}^{-T}R_{21}R_{11}^{-1}g^T(x)\nabla\varphi_1{}^T\right\| + \tfrac{1}{4}\|W_2\|^2\left\|\nabla\varphi_2 k(x)R_{22}^{-1}k^T(x)\nabla\varphi_2{}^T\right\| \\ & + \tfrac{1}{2}b_{\varepsilon_{x1}} b_g^2 b_{\phi_{1x}}\sigma_{\min}(R_{11})\|W_1\| + \tfrac{1}{2}b_{\varepsilon_{x1}} b_k^2 b_{\phi_{2x}}\sigma_{\min}(R_{22})\|W_2\| + \tfrac{1}{2}b_{\varepsilon_{x2}} b_g^2 b_{\phi_{1x}}\sigma_{\min}(R_{11})\|W_1\| + \tfrac{1}{2}b_{\varepsilon_{x2}} b_k^2 b_{\phi_{2x}}\sigma_{\min}(R_{22})\|W_2\| \end{aligned}$$

According to Facts 4.1 *c* is bounded by $c_{\max}$, and D is bounded by $D_{\max}$ which can be expressed in terms of the bounds given there.

Let the parameters be chosen such that $M > 0$. To justify this, the matrix *M* is written in compact form as

$$M = \begin{bmatrix} q_1 + q_2 & 0 & 0 \\ 0 & I_2 & M_{23} \\ 0 & M_{32} & M_{33} \end{bmatrix} \qquad \text{(A.50)}$$

where in order to be positive definite the following properties (Lewis, Syrmos, 1995) must hold

a) $q_1 + q_2 > 0$

b) $I_2 > 0$

c) Shur complement for I_2 is $D_{22} = I_2 - M_{23} M_{33}^{-1} M_{32} > 0$ which hold after proper selection of F_1, F_2, F_3 and F_4.

d) Schur complement for M_{33} is $D_{33} = M_{33} - M_{32} I_2^{-1} M_{23} > 0$ which hold after proper selection of F_1, F_2, F_3 and F_4.

Now (A.49) becomes

$$\dot{L} < -\left\|\tilde{Z}\right\|^2 \sigma_{\min}(M) + D_{\max}\left\|\tilde{Z}\right\| + c_{\max} + \bar{\varepsilon}_1 + \bar{\varepsilon}_2$$

Completing the squares, the Lyapunov derivative is negative if

$$\left\|\tilde{Z}\right\| > \frac{D_{\max}}{2\sigma_{\min}(M)} + \sqrt{\frac{D_{\max}^2}{4\sigma_{\min}^2(M)} + \frac{c_{\max} + \bar{\varepsilon}_1 + \bar{\varepsilon}_2}{\sigma_{\min}(M)}} \equiv B_Z. \tag{A.51}$$

It is now straightforward to demonstrate that if *L* exceeds a certain bound, then, $\dot{L}$ is negative. Therefore, according to the standard Lyapunov extension theorem the analysis above demonstrates that the state and the weights are UUB [42], [52].

Note that condition (A.51) holds if the norm of any component of $\tilde{Z}$ exceeds the bound, i.e. specifically $x > B_Z$ or $\bar{\sigma}_3{}^{\mathrm{T}}\tilde{W}_1 > B_Z$ or $\bar{\sigma}_4{}^{\mathrm{T}}\tilde{W}_2 > B_Z$ or $\tilde{W}_3 > B_Z$ or $\tilde{W}_4 > B_Z$.

Now consider the error dynamics and the output as in Chapter 2 and assume $\bar{\sigma}_3$ and $\bar{\sigma}_4$ are persistently exciting. Substituting (4.23), (4.24) in (4.31) and (4.32) respectively we obtain the error dynamics

$$\dot{\tilde{W}}_1 = -a_1\bar{\sigma}_3\bar{\sigma}_3{}^{\mathrm{T}}\tilde{W}_1 + a_1\bar{\sigma}_3\frac{\varepsilon_{HJ_1}}{m_{s_1}} + \frac{a_1}{4m_{s_1}{}^2}\tilde{W}_3{}^T\bar{D}_1(x)\tilde{W}_3 + \frac{a_1}{4m_s{}^2}\tilde{W}_4{}^T\nabla\phi_2 R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\tilde{W}_4$$
$$+\frac{a_1}{2m_s{}^2}(W_1{}^T\bar{E}_2(x)\tilde{W}_4 - W_2{}^T\nabla\phi_2 R_{22}{}^{-T}R_{12}R_{22}{}^{-1}k^T(x)\nabla\phi_2{}^T\tilde{W}_4)\ \ y_1 = \bar{\sigma}_3{}^{\mathrm{T}}\tilde{W}_1$$

and

$$\dot{\tilde{W}}_2 = -a_2\bar{\sigma}_4\bar{\sigma}_4^{\mathrm{T}}\tilde{W}_2 + a_2\bar{\sigma}_4\frac{\varepsilon_{HJ_2}}{m_{s_2}} + \frac{a_2}{4m_{s_2}^{\ 2}}\tilde{W}_4^{\ T}\bar{D}_2(x)\tilde{W}_4 + \frac{a_2}{4m_{s_2}^{\ 2}}\tilde{W}_3^{\ T}\nabla\phi_1 g(x)R_{11}^{\ -T}R_{21}R_{11}^{\ -1}g^T(x)\nabla\phi_1^{\ T}\tilde{W}_3$$
$$+\frac{a_2}{2m_{s_2}^{\ 2}}(\tilde{W}_3^{\ T}\bar{E}_1 W_2 - \tilde{W}_3^{\ T}\nabla\phi_1 g(x)R_{11}^{\ -T}R_{21}R_{11}^{\ -1}g^T(x)\nabla\phi_1^{\ T}W_1)\ \ y_2 = \bar{\sigma}_4^{\mathrm{T}}\tilde{W}_2$$

Then Theorem 4.1 is true with

$$\left\|\frac{\sigma_3^{\ T}}{m_{s_1}}\tilde{W}_1\right\| > \varepsilon_{\max_1} > \frac{1}{4}\left\|\tilde{W}_3\right\|^2\left\|\frac{\bar{D}_1}{m_{s_1}}\right\| + \frac{1}{2}\left\|W_1\right\|\left\|\frac{\bar{E}_2(x)}{m_{s_1}}\right\|\left\|\tilde{W}_4\right\| + \frac{1}{4}\left\|\tilde{W}_4\right\|^2\left\|\frac{\nabla\phi_2 k(x)R_{22}^{\ -T}R_{12}R_{22}^{\ -1}k^T(x)\nabla\phi_2^{\ T}}{m_{s_1}}\right\| + \bar{\varepsilon}_1\left\|\frac{1}{m_{s_1}}\right\|$$
$$-\frac{1}{2}\left\|W_2\right\|\left\|\frac{\nabla\phi_2 k(x)R_{22}^{\ -T}R_{12}R_{22}^{\ -1}k^T(x)\nabla\phi_2^{\ T}}{m_{s_1}}\right\|\left\|\tilde{W}_4\right\|$$

and

$$\left\|\frac{\sigma_4^{\ T}}{m_{s_2}}\tilde{W}_2\right\| > \varepsilon_{\max_2} > \frac{1}{4}\left\|\tilde{W}_4\right\|^2\left\|\frac{\bar{D}_2}{m_{s_2}}\right\| + \frac{1}{2}\left\|W_2\right\|\left\|\frac{\bar{E}_1(x)}{m_{s_2}}\right\|\left\|\tilde{W}_3\right\| + \frac{1}{4}\left\|\tilde{W}_3\right\|^2\left\|\frac{\nabla\phi_1 g(x)R_{11}^{\ -T}R_{21}R_{11}^{\ -1}g^T(x)\nabla\phi_1^{\ T}}{m_{s_2}}\right\| + \bar{\varepsilon}_2\left\|\frac{1}{m_{s_2}}\right\|$$
$$-\frac{1}{2}\left\|W_1\right\|\left\|\frac{\nabla\phi_1 g(x)R_{11}^{\ -T}R_{21}R_{11}^{\ -1}g^T(x)\nabla\phi_1^{\ T}}{m_{s_2}}\right\|\left\|\tilde{W}_3\right\|$$

This provides effectival bounds for $\left\|\bar{\sigma}_3^{\ T}\tilde{W}_1\right\|$ and $\left\|\bar{\sigma}_4^{\ T}\tilde{W}_2\right\|$.

This completes the proof.

■

REFERENCES

[1] Aangenent W., Kostic D., Jager B. D. van de Molengaft Rene, Steibuch M. (2005). Data-based Optimal Control, *Proc. Of America Control Conference*, Portland, OR, 1460-1465.

[2] Abou-Kandil, Freiling G., Ionescu V., & Jank G. (2003). *Matrix Riccati Equations in Control and Systems Theory*, Birkhäuser.

[3] Abu-Khalaf M., Lewis F. L. (2008). Neurodynamic Programming and Zero-Sum Games for Constrained Control Systems, *IEEE Transactions on Neural Networks*, 19(7), 1243-1252.

[4] Abu-Khalaf M. & Lewis F. L. (2005). Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach, *Automatica*, 41(5), 779-791.

[5] Abu-Khalaf M., Lewis F. L., Huang Jie (2006). Policy Iterations on the Hamilton-Jacobi- Isaacs Equation for H_∞ State Feedback Control With Input Saturation, *IEEE Transactions on Automatic Control*, 51(12), 1989-1995.

[6] Adams R & Fournier J. (2003). *Sobolev spaces*, New York: Academic Press.

[7] Al-Tamimi A., Lewis F.L., & Abu-Khalaf M. (2008). Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 38, 943-949.

[8] Baird L. C. III (1994). Reinforcement Learning in Continuous Time: Advantage Updating, *Proc. of ICNN*, Orlando FL.

[9] Ball J., Helton W. (1996). Viscosity solutions of Hamilton-Jacobi equations arising in nonlinear H_∞ -control. *J. Math Syst., Estimat., Control*, 6(1), 1-22.

[10] Bardi M., Capuzzo-Dolcetta I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Boston, MA: Birkhäuser.

[11] Başar T., & Olsder G. J. (1999). *Dynamic Noncooperative Game Theory*, 2nd ed. Philadelphia, PA: SIAM.

[12] Başar T. & Bernard P. (1995). H_∞ *Optimal Control and Related Minimax Design Problems*. Boston, MA: Birkhäuser.

[13] Beard R., Saridis G, Wen J. (1997). Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation, *Automatica*, 33(12), 2159–2177.

[14] Bennighof J.K., Chang S.M., & Subramaniam M. (1993). Minimum Time Pulse Response Based Control of Flexible Structures, *J. Guid., Control and Dyn.*, 16(5), 874–881.

[15] Bertsekas D. P. & Tsitsiklis J. N. (1996). *Neuro-Dynamic Programming*, Athena Scientific, MA.

[16] Bhasin S., Johnson M., Dixon W. E. (2010). A model free robust policy iteration algorithm for optimal control of nonlinear systems, *Proc. 49th IEEE Conference on Decision and Control*, 3060-3065.

[17] Bradtke S. J., Ydstie B. E., Barto A. G. (1994). Adaptive linear quadratic control using policy iteration, *Proceedings of the American Control Conference*, 3475-3476.

[18] Brewer J. W. (1978). Kronecker Products and Matrix Calculus in System Theory, *IEEE Trans. On Circuits and Systems*, 25, 772-781.

[19] Busoniu L., Babuska R., De Schutter B. (2008). A comprehensive survey of multi-agent reinforcement learning, *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, 38(2), 156-172.

[20] Cao X. (2007). *Stochastic Learning and Optimization*, Springer-Verlag, Berlin.

[21] Dierks T., & Jagannathan S. (2010). Optimal control of affine nonlinear continuous-time systems using an online Hamilton-Jacobi-Isaacs Formulation, *Proc. IEEE Conf. Decision and Control*, 3048-3053.

[22] Doya K. (2000). Reinforcement Learning In Continuous Time and Space, *Neural Computation*, 12(1), 219-245.

[23] Doya K., Kimura H., Kawato M. (2001). Neural Mechanisms of Learning and Control. *IEEE Control Syst. Mag.*, 21(4), 42-54.

[24] Fax J., & Murray R. (2004). Information flow and cooperative control of vehicle formations, *IEEE Trans. Automatic Control*, 49(9), 1465-1476.

[25] Feng Y., Anderson B. D., Rotkowitz M. (2009) A game theoretic algorithm to compute local stabilizing solutions to HJBI equations in nonlinear H∞ control, *Automatica*, 45(4), 881-888.

[26] Finlayson B. A. (1990). *The method of weighted residuals and variational principles*. New York: Academic Press.

[27] Freiling G., Jank G., Abou-Kandil H. (2002). On global existence of Solutions to Coupled Matrix Riccati equations in closed loop Nash Games, *IEEE Transactions on Automatic Control*, 41(2), 264- 269.

[28] Gajic Z., & Li T-Y. (1988). Simulation results for two new algorithms for solving coupled algebraic Riccati equations, *Third Int. Symp. On Differential Games*, Sophia, Antipolis, France.

[29] Ge S. S., & Wang C. (2004). Adaptive neural control of uncertain MIMO nonlinear systems, *IEEE Trans. Neural Networks*, 15(3), 674-692.

[30] Hanselmann T., Noakes L., & Zaknich A. (2007). Continuous-Time Adaptive Critics, *IEEE Trans on Neural Networks*, 18(3), 631-647.

[31] 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, 382- 384.

[32] Hjalmarsson H. & Gevers M. (1998). Iterative Feedback Tuning: Theory and Applications. *IEEE Control Syst. Mag.*, 18(4), 26–41.

[33] Hong Y., Hu J., & Cao L. (2006). Tracking control for multi-agent consensus with an active leader and variable topology, *Automatica*, 42(7), 1177-1182.

[34] Hornik K., Stinchcombe M., & White H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, *Neural Networks*, 3, 551–560.

[35] Howard R. A. (1960). *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA.

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

[37] Jadbabaie A., Lin J., & Morse A. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules, *IEEE Trans. Automatic Control*, 48(6), 988-1001.

[38] Johnson M., Hiramatsu T., Fitz-Coy N., & Dixon W. E. (2010), Asymptotic Stackelberg Optimal Control desing for an Uncertain Euler Lagrange System, *IEEE Conference on Decision and Contro, 6686-6691.*

[39] Jungers M., De Pieri E., Abou-Kandil H., (2007). Solving coupled algebraic Riccati equations from closed-loop Nash strategy, by lack of trust approach, *International Journal of Tomography and Statistics*, 7(F07), 49-54.

[40] Kakade S., Kearns M., Langford J., & Ortiz L. (2003). Correlated equilibria in graphical games, *Proc. 4th ACM Conference on Electronic Commerce*, 42-47.

[41] Kearns M., Littman M., & Singh S. (2001). Graphical models for game theory, *Proc 17th Annual Conference on Uncertainty in Artifical Intelligence*, 253-260.

[42] Khalil H. K., (1996). *Nonlinear Systems*. Prentice-Hall.

[43] Khoo S., Xie L., & Man Z. (2009). Robust Finite-Time Consensus Tracking algorithm for Multi-robot Systems, *IEEE Transactions on Mechatronics*, 14, 219-228.

[44] Kleinman D. (1968). On an Iterative Technique for Riccati Equation Computations, *IEEE Trans. on Automatic Control*, *13*(1), 114–115.

[45] Krstic M. & Deng H. (1998). Stabilization of Nonlinear Uncertain Systems, Springer.

[46] Lancaster P. & Rodman L. (1995). Algebraic Riccati Equations, Oxford Science Publications, Oxford University Press, New York.

[47] Leake R. J., & Liu Ruey-Wen (1967). Construction of Suboptimal Control Sequences, *J. SIAM Control*, 5(1), 54-63.

[48] Lecchini A., Campi M. C., & Savaresi S. M. (2002). Virtual Reference Feedback Tuning for Two Degree of Freedom Controllers, *Internat. J. Adapt. Control Sig. Process*, 16, 355-371.

[49] Lewis F. (1992). *Applied Optimal Control and Estimation: Digital Design and Implementation*, New Jersey: Prentice-Hall.

[50] Lewis F.L., Jagannathan S., Yesildirek A. (1999). *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis.

[51] Lewis F.L., Lendaris G., & Liu D. (2008). Special Issue on Approximate Dynamic Programming and Reinforcement Learning for Feedback Control, *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 38(4).

[52] Lewis F.L., Liu. K., & Yesildirek A. (1995). Neural Net Controller with Guaranteed Tracking Performance, *IEEE Trans on Neural Networks*, 6(3), 703-715.

[53] Lewis F. L., & Syrmos V. L. (1995). *Optimal Control*, John Wiley.

[54] Lewis F.L., Vrabie D., (2009). Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control, *IEEE Circuits and Systems Magazine*, 9(3), 32-50.

[55] Lewis F.L., Xie Lihua, Popa D.O. (2007). Optimal and Robust Estimation, CRC Press.

[56] Li X., Wang X., & Chen G. (2004). Pinning a complex dynamic network to its equilibrium, *IEEE Trans. Circuits Syst. I, Reg. Papers*, 51(10), 2074-2087.

[57] Li Z. H. & Krstic M. (1997). Optimal design of adaptive tracking controllers for nonlinear systems, *Proc. of ACC*, 1191-1197.

[58] Lim R. K., Phan M. Q., & Longman R. W. (1998). State-Space System Identification with Identified Hankel Matrix, Department of Mechanical and Aerospace Engineering Technical Report No. 3045, Princeton University.

[59] Lim R. K., Phan M. Q. (1997). Identification of a Multistep-Ahead Observer and Its Application to Predictive Control, *J. Guid., Control and Dyn.*, 20(6), 1200-1206.

[60] Limebeer D. J. N., Anderson B. D. O., & Hendel H., (1994). A Nash game approach to mixed H_2 / H_∞ control, *IEEE Transactions on Automatic Control*, 39(1), 69-82.

[61] Littman M. L. (2001). Value function reinforcement learning in Markov games, *Journal of Cognitive Systems Research 1.*

[62] Mehta P. & Meyn S. (2009). Q-learning and Pontryagin's minimum principle, preprint.

[63] Murray J. J., Cox C. J., Lendaris G. G., & Saeks R. (2002). Adaptive Dynamic Programming, *IEEE Trans. on Systems, Man and Cybernetics*, 32(2), 140-153.

[64] Nevistic V. & Primbs J. A. (1996). Constrained nonlinear optimal control: a converse HJB approach, California Institute of Technology, Pasadena, CA 91125, Tech Rep. CIT-CDS 96-021.

[65] Olfati-Saber R., Fax J, Murray R. (2007). Consensus and cooperation in netwowrked multi-agent systems, *Proc. IEEE,* 95(1), 215-233.

[66] Olfati-Saber R., Murray R. M. (2004). Consensus Problems in Networks of Agents with Switching Topoligy and Time-delays, *IEEE Transactions of Automatic Contro,* 49, 1520-1533.

[67] Phan M. Q., Lim R. K., & Longman, R. W. (1998). Unifying Input-Output and State-Space Perspectives of Predictive Control, Department of Mechanical and Aerospace Engineering Technical Report No. 3044, Princeton University.

[68] Powell W. (2007). Approximate Dynamic Programming: Solving the Curses of Dimensionality. J. Wiley & Sons.

[69] Prokhorov D., & Wunsch D. (1997). Adaptive critic designs, *IEEE Trans. on Neural Networks*, 8(5), 997-1007.

[70] Qu Z. (2009). *Cooperative Control of Dynamical systems: Applications to Autonomous Vehicles*, New York: Springer-Verlag.

[71] Ren W., Beard R., Atkins E. (2005). A survey of consensus problems in multi-agent coordination, *Proc Amer. Control Conference*, 1859-1864.

[72] Ren W., Beard R. (2005). Consensus seeking in multi-agent systems under dynamically changing interaction topologies, *IEEE Trans. Autom. Control*, 50(5), 655-661.

[73] Ren W., Beard R. (2008). *Distributed Consensus in Multi-vehicle Cooperative Control*, Springer.

[74] Ren W., Moore K., Chen Y. (2007). Hign-order and model reference consensus algorithms in coperative control of multi-vehicle systems, *J. Dynam. Syst., Meas., Control*, 129(5), 678-688.

[75] Safonov M. G. & Tsao T. C. (1997). The Unfalsified Control Concept and Learning, *IEEE Trans. Autom. Control*, 42(6), 843–847.

[76] Sandberg E. W. (1998). Notes on uniform approximation of time-varying systems on finite time intervals, *IEEE Transactions on Circuits and Systems—1: Fundamental Theory and Applications*, 45(8), 863-865.

[77] Van der Schaft A. J. (1992). *L*2-gain analysis of nonlinear systems and nonlinear state feedback *H*∞ control, *IEEE Transactions on Automatic Control*, 37(6), 770–784.

[78] 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.

[79] Schultz W., Dayan P., Montague P. Read (1997). A Neural Substrate of Prediction and Reward", *Science*, 275, 1593-1599.

[80] Shoham Y., Leyton-Brown K. (2009). *Multiagent Systems: Algorithmic, Game Theoretic, and Logical Foundations*, Cambridge University Press.

[81] Si J., Barto A., Powel W., & Wunsch D. (2004). *Handbook of Learning and Approximate Dynamic Programming*, John Wiley, New Jersey.

[82] Skelton R. E. & Shi G. (2000). Markov Data-Based LQG Control, *J. of Dyn. Systems, Meas., and Control*, 122, 551–559.

[83] Sontag E.D. & Sussmann H.J.(1995). Nonsmooth control-Lyapunov functions, *IEEE Proc. CDC95*, 2799-2805.

[84] Spall J.C. & Criston J.A. (1998). Model-free control of nonlinear stochastic systems with discrete-time measurements, *IEEE Trans. Autom.Control*, 43(9), 1198–1210.

[85] Stevens B., & Lewis F. L. (2003). *Aircraft Control and Simulation*, 2nd edition, John Willey, New Jersey.

[86] Sutton R. S., & Barto A. G. (1998) *Reinforcement Learning – An Introduction*, MIT Press, Cambridge, Massachusetts.

[87] Tao G. (2003) *Adaptive Control Design and Analysis*. Adaptive and Learning Systems for Signal Processing, Communications and Control Series, Hoboken, N.J.: Wiley-Interscience.

[88] Tijs S (2003). *Introduction to Game Theory*, Hindustan Book Agency, India.

[89] Toussaint R. L., Boissy J. C., Norg M. L., Steinbuch M., & Bosgra O.H. (1998). Suppressing Non-periodically Repeating Disturbances In Mechanical Servo Systems." *Proc. IEEE Conf. Dec. Control*, Tampa, Florida, 2541–2542.

[90] Tsitsiklis J. (1984). *Problems in decentralized decision making and computation*, Ph.D. dissertation, Dept. Elect. Eng. And Comput. Sci., MIT, Cambridge, MA.

[91] Vrabie D. (2009). *Online Adaptive Optimal Control for Continuous Time Systems*. Ph.D. Thesis, Dept. of Electrical Engineering, Univ. Texas at Arlington, Arlington, TX, USA.

[92] Vrabie, D., Pastravanu, O., Lewis, F. L., & Abu-Khalaf, M. (2009). Adaptive Optimal Control for continuous-time linear systems based on policy iteration, *Automatica*, 45(2), 477-484.

[93] Vrabie D., Vamvoudakis K., & Lewis F. (2009). Adaptive optimal controllers based on generalized policy iteration in a continuous-time framework, *Proc. of the IEEE Mediterranean Conf. on Control and Automation*, 1402-1409.

[94] Vrancx P., Verbeeck K., Nowe A. (2008). Decentralized learning in markov games, *IEEE Transactions on Systems, Man and Cybernetics*, 39(4), 976-981.

[95] Wang F.Y., Zhang H., Liu D. (2009). Adaptive dynamic programming: an introduction, *IEEE Computational Intelligence Magazine*, 39-47.

[96] Wang X., Chen G. (2002). Pinning control of scale-free dynamical networks, *Physica A*, 310(3-4), 521-531.

[97] Watkins C. (1989). Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, Cambridge, England.

[98] Werbos P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*, Ph.D. Thesis.

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

[100] Werbos P. J. (1989). Neural networks for control and system identification, *IEEE Proc. CDC,* 1, 260-265.

[101] White D.A., Sofge D.A (Eds.) (1992). *Handbook of intelligent control*, Van Nostrand Reinhold, New York.

[102] Widrow B. and Walach E. (2008). *Adaptive Inverse Control - A Signal Processing Approach,* Wiley, Hoboken, NJ.

BIOGRAPHICAL INFORMATION

Kyriakos G. Vamvoudakis was born in Athens Greece. He received the Diploma in Electronic and Computer Engineering from the Technical University of Crete, Greece in 2006 with highest honors, the M.Sc. degree in Electrical Engineering from The University of Texas at Arlington in 2008 and the Ph.D. in 2011 from the same department. He has been working as a research assistant at the Automation and Robotics Research Institute, The University of Texas at Arlington. He is coauthor of 2 book chapters, and 25 technical publications. His current research interests include approximate dynamic programming, game theory, neural network feedback control, optimal control, adaptive control and systems biology. He is a member of Tau Beta Pi, Eta Kappa Nu and Golden Key honor societies and is listed in *Who's Who in the world* and *Who's Who in Science and Engineering*. He received the Best Paper Award for Autonomous/Unmanned Vehicles at the 27th Army Science Conference in 2010. He also received the Best Researcher/Student Award, UTA Automation & Robotics Research Institute in 2010. He has co-organized special sessions for several international conferences. Mr. Vamvoudakis is a registered Electrical/Computer engineer (PE) and member of Technical Chamber of Greece.