DISCRETE-TIME CONTROL ALGORITHMS AND ADAPTIVE INTELLIGENT SYSTEMS DESIGNS

by

ASMA AZMI AL-TAMIMI

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 2007

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Professor Frank L. Lewis for his supervision and guidance during my doctoral research work. I also extend my thanks to Dr. Murad Abu-Khalaf for his close instrumental supervision during my research work at ARRI.

My thanks also go to my defense committee: Kai-Shing Yeung, Wei-Jen Lee, Dan Popa, and Kamesh Subbarao for their time, suggestions, and remarks to improve this work. Moreover, I like to thank my colleagues in the ACS group and every one else who helped me during my studies in the USA.

I would like to acknowledge the financial support of the Hashemite University. I also acknowledge the financial support by the Electrical Engineering Department, and the Automation Robotics Research Institute. The research in this dissertation was funded by the National Science Foundation ECS-0501451 and the Army Research Office W91NF-05-1-0314.

I am very indebted to my parents for their infinite support, care and attention without which I would have not been at this stage in my life. I have been greatly inspired by them and I dedicate this dissertation to my mother Khawla and father Azmi.

March 30, 2007

ABSTRACT

DISCRETE-TIME CONTROL ALGORITHMS AND ADAPTIVE INTELLIGENT SYSTEMS DESIGNS

Publication No. ______

Asma Azmi Al-Tamimi, PhD.

The University of Texas at Arlington, 2007

Supervising Professor: Frank L. Lewis

In this work, approximate dynamic programming (ADP) designs based on adaptive critic structures are developed to solve the discrete-time H_2 / H_∞ optimal control problems in which the state and action spaces are continuous. This work considers linear discrete-time systems as well as nonlinear discrete-time systems that are affine in the input. This research resulted in forward-in-time reinforcement learning algorithms that converge to the solution of the Generalized Algebraic Riccati Equation (GARE) for linear systems. For the nonlinear case, a forward-in-time reinforcement learning algorithm is presented that converges to the solution of the associated Hamilton-Jacobi Bellman equation (HJB).

The results in the linear case can be thought of as a way to solve the GARE of the well-known discrete-time H_∞ optimal control problem forward in time. Four design algorithms are developed: Heuristic Dynamic programming (HDP), Dual Heuristic dynamic programming (DHP), Action dependent Heuristic Dynamic programming (ADHDP) and Action dependent Dual Heuristic dynamic programming (ADDHP). The significance of these algorithms is that for some of them, particularly the ADHDP algorithm, a priori knowledge of the plant model is not required to solve the dynamic programming problem.

Another major outcome of this work is that we introduce a convergent policy iteration scheme based on the HDP algorithm that allows the use of neural networks to arbitrarily approximate for the value function of the discrete-time HJB equation. This online algorithm may be implemented in a way that requires only partial knowledge of the model of the nonlinear dynamical system.

The dissertation includes detailed proofs of convergence for the proposed algorithms, HDP, DHP, ADHDP, ADDHP and the nonlinear HDP. Practical numerical examples are provided to show the effectiveness of the developed optimization algorithms. For nonlinear systems, a comparison with methods based on the State-Dependent Riccati Equation (SDRE) is also presented. In all the provided examples, parametric structures like neural networks have been used to find compact representations of the value function and optimal policies for the corresponding optimal control problems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

In this dissertation, adaptive critic designs that are based on the dynamic programming principle are developed to solve H_2 / H_∞ optimal control problems for discrete-time dynamical systems. In the case of H_∞ optimal control, the zero-sum game for discrete-time linear systems is solved by creating and developing adaptive critic structures that learn to co-exist. In the H_2 optimal control case, the dynamical programming problem associated with nonlinear discrete-time dynamical systems is solved, *i.e.* solving for the value function of the corresponding HJB equation.

Approximate dynamic programming, also known as Neuro Dynamic Programming, was first proposed by Werbos [27], Barto *et. al.* [1], Widrow *et. al.* [5], Howard [28], Watkins [8], Bertsekas and Tsitsiklis [11], and others to solve optimal control problems forward-in-time. The optimal control law, *i.e.* the action network, and the value function, *i.e.* the critic network, are modeled as parametric structures, *i.e.* neural networks. This is combined with incremental optimization such as reinforcement learning to tune and improve both networks forward-in-time and hence can be implemented in actual control systems. This overcomes computational complexity associated with dynamic programming, which is an offline technique that requires a backward-in-time solution procedure [15]. Moreover, as will be discussed in the dissertation, some of the presented adaptive critic designs do not require the plant model for tuning the action network, the critic network, or both of them.

Several approximate dynamic programming schemes appear in literature. Howard [28] proposed iterations in the policy space in the framework of stochastic decision theory. In [30], Bradtke *et al.* implemented a Q-learning policy iteration method for the discrete-time linear quadratic optimal control problem. Hagen [29] discussed the relation between the Q-learning policy iteration method and model-based adaptive control with system identification. Werbos [25] classified approximate dynamic programming approaches into four main schemes: Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP), Action Dependent Heuristic Dynamic Programming (ADHDP), also known as Q-learning [8], and Action Dependent Dual Heuristic Dynamic Programming (ADDHP). In [9], Prokhorov and Wunsch developed new approximate dynamic programming schemes known as Globalized Dual Heuristic Dynamic Programming (GDHP) and Action Dependent Globalized Dual Heuristic Dynamic Programming (ADGDHP). Landelius [31] applied HDP, DHP, ADHDP and ADDHP techniques to the discrete-time linear quadratic optimal control problem and discussed their convergence showing that they are equal to iterating on the underlying Riccati equation. The current status of work on approximate dynamic programming is given in [20]. See also [11].

Reinforcement learning methods to solve game theory problems have recently appeared in [24] and [18] in the framework of Markov games where multiagent Q-learning methods are proposed and shown to converge to the Nash equilibrium under specific conditions. Unlike the work in this dissertation, these are lookup-table-based methods concerned with discrete state and action spaces.

In this dissertation, adaptive critic designs, namely HDP, DHP, ADHDP and ADDHP, are derived to solve dynamic programming problems online for discrete-time dynamical systems with continuous state and action spaces. Offline solutions of these optimal control problems based on the dynamic programming principle appear in [7], [6], [15]. An off-line neural net policy iterations solution based on the dynamic programming principle appears in [23] for the continuous-time case.

The importance of this dissertation stems from the fact adaptive critics algorithms are used to design H_2 / H_∞ controller without knowing the dynamics of linear systems, *e.g.* ADHDP algorithm, and partial knowledge of the dynamics of nonlinear systems, *e.g.* HDP algorithm. Therefore, these algorithms may be thought of as being direct adaptive optimal control architectures.

The organization of this dissertation is as follows. In Chapter 2, zero-sum games for discrete-time linear systems with quadratic infinite horizon cost are revisited. Dynamic programming is used to derive the optimal policies for both the control and the disturbance inputs along with the associated Riccati equation. Although equivalent to those found in literature [6], the derived policies are different in structure and appear in a form required for the design of adaptive critics. In Chapter 3, Heuristic Dynamic Programming (HDP) and Dual Heuristic Dynamic Programming (DHP) algorithms are proposed to solve the zero-sum game for linear systems forward-in-time. Chapter 4 extends the results of Chapter 3 to Action Dependent Heuristic Dynamic Programming

(ADHDP) and Action Dependent Dual Heuristic Dynamic Programming (ADDHP). In Chapter 5, an application of the ADHDP algorithm to the design of load frequency control systems is demonstrated. In Chapter 6 the nonlinear HDP algorithm is derived, with proofs of convergence, to solve for the HJB equation. It is also shown that the optimal controller derived through the DT HJB outperforms that using the State Dependent Riccati Equation (SDRE).

CHAPTER 2

DISCRETE-TIME H-INFINITY STATE FEEDBACK CONTROL FOR ZERO-SUM GAMES

In this chapter, the solution of the zero-sum game of a linear discrete-time system with quadratic cost derived under state feedback information structure. The policies for each of the two players, control and disturbance, are derived with the associated Riccati equation. Specific forms for both the Riccati equation and the control and disturbance policies are derived that are required for applications in ADP these forms are not same as standard forms in the existing literature. The relation between the derived policies and the associated Riccati equation with those existing in literature is discussed

Consider the following discrete-time linear system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k \\ y_k &= x_k, \end{aligned} . \tag{2.1}$$

where $x \in R^n$, $y \in R^p$, $u_k \in R^{m_1}$ is the control input and $w_k \in R^{m_2}$ is the disturbance input. Consider the infinite-horizon cost function. For any stabilizing sequence of policies u_k and w_k, one can write the infinite-horizon cost-to-go as

$$\begin{aligned} V(x_k) &= \sum\nolimits_{i=k}^{\infty} x_i^T Q x_i + u_i^T u_i - \gamma^2 w_i^T w_i \\ &= x_k^T Q x_k + u_k^T u_k - \gamma^2 w_k^T w_k + \sum\nolimits_{i=k+1}^{\infty} x_i^T Q x_i + u_i^T u_i - \gamma^2 w_i^T w_i \\ &= x_k^T Q x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V(x_{k+1}) \\ &= r(x_k, u_k, w_k) + V(x_{k+1}). \end{aligned} \tag{2.2}$$

It is desired to find the optimal control u_k^* and the worst case disturbance w_k^*, in which the infinite-horizon cost is to be minimized by player 1, u_k, and maximized by player 2, w_k. Here the class of strictly feedback stabilizing policies is considered [6].

$$V^*(x_k) = \min_u \max_w \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T u_i - \gamma^2 w_i^T w_i \tag{2.3}$$

Using the dynamic programming principle, the optimization problem in equation (2.3) and (2.2) can be written as

$$\begin{aligned} V^*(x) &= \min_u \ \max_w (r(x_k, u_k, w_k) + V^*(x_{k+})) \\ &= \max_w \ \min_u (r(x_k, u_k, w_k) + V^*(x_{k+})). \end{aligned} \tag{2.4}$$

If we assume that there exists a solution to the GARE that is strictly feedback stabilizing, then it can be shown, see[10], that the policies are in saddle-point equilibrium, *i.e. minimax* is equal to *maximin,* in the restricted class of feedback stabilizing policies under which $x_k \to 0$ as $k \to \infty$ for all $x_0 \in R^n$. See [6], p. 340), and ([7], p. 138) and [13][10]. It is known that the optimal cost is quadratic in the state, and it is given as

$$V^*(x_k) = x_k^T P x_k \tag{2.5}$$

where $P \geq 0$.

Assuming that the game has a value and is solvable, then in order to have a unique feedback saddle-point in the class of strictly feedback stabilizing policies, the inequalities in (2.6) and (2.7) should be satisfied, [7],

$$I - \gamma^{-2} E^T P E > 0 \tag{2.6}$$

$$I + B^T PB > 0 . \tag{2.7}$$

Applying the Bellman optimality principle, one has

$$\begin{aligned} V^*(x_k) &= \min_u \max_w (r(x_k, u_k, w_k) + V^*(x_{k+1})) \\ &= \min_u \max_w (x_k^T Q x_k + u_k^T u_k - \gamma \wedge 2 w_k^T w_k + x_{k+1}^T P x_{k+1}). \end{aligned} \tag{2.8}$$

Substituting (2.5) in equation(2.8) one has

$$\begin{aligned} x_k^T P x_k = \min_u \max_w (x_k^T Q x_k + u_k^T u_k - \gamma \wedge 2 w_k^T w_k \\ + (Ax_k + Bu_k + Ew_k)^T P(Ax_k + Bu_k + Ew_k). \end{aligned} \tag{2.9}$$

To maximize with respect to the disturbance w_k, one needs to apply the first order necessary condition

$$\begin{aligned} \frac{\partial V_k}{\partial w_k} &= 0 \\ &= -2\gamma^2 w_k + 2E^T P(A + Bu_k + Ew_k). \end{aligned} \tag{2.10}$$

Therefore, the disturbance can be written as

$$w_k = (\gamma^2 I - E^T PE)^{-1}(E^T PAx_k + E^T PBu_k) . \tag{2.11}$$

Similarly, to minimize with respect to the control input u_k one has

$$\begin{aligned} \frac{\partial V_k}{\partial u_k} &= 0 \\ &= 2u_k + 2B^T P(A + Bu_k + Ew_k). \end{aligned} \tag{2.12}$$

Hence, the controller can be written as

$$u_k = -(I + B^T PB)^{-1}(B^T PAx_k + B^T PEw_k) . \tag{2.13}$$

Note that applying the 2nd order sufficiency conditions for both players, one obtains (2.6) and (2.7).

Substituting equation (2.11) in (2.12)

$$u_k^* = (I + B^T PB - B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PB)^{-1} \times \\ (B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PA - B^T PA)x_k , \tag{2.14}$$

so the optimal control is a state feedback with gain

$$L = (I + B^T PB - B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PB)^{-1} \times \\ (B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PA - B^T PA). \tag{2.15}$$

Substituting the equation (2.13) in (2.10) one can find the optimal policy to the disturbance

$$w_k^* = (E^T PE - \gamma^2 I - E^T PB(I + B^T PB)^{-1} B^T PE)^{-1} \times \\ (B^T PE(I + B^T PB)^{-1} BPA - E^T PA)x_k , \tag{2.16}$$

so the optimal disturbance is a state feedback with gain

$$K = (E^T PE - \gamma^2 I - E^T PB(I + B^T PB)^{-1} B^T PE)^{-1} \times \\ (E^T PB(I + B^T PB)^{-1} BPA - E^T PA). \tag{2.17}$$

Note that the inversion matrices in (2.14) and (2.16) exists due to (2.6) and (2.7)

It is now going to be shown that the policies obtained in equations (2.15) and (2.17) are equivalent to those known in the literature [7]. The following Lemma is required.

Lemma 2.1*:* If $(I - \gamma^{-2} E^T PE)$ is invertible, then $(I - \gamma^{-2} EE^T P)$ is also invertible.

Proof: Since $(I - \gamma^{-2} E^T PE)$ is invertible then the following expression is valid

$$I + \gamma^{-2} E(I - \gamma^{-2} E^T PE)^{-1} E^T P .$$

Applying the matrix inversion lemma, [15], it can be shown that

$$I + \gamma^{-2} E(I - \gamma^{-2} E^T PE)^{-1} E^T P = (I - \gamma^{-2} EE^T P)^{-1}$$

Hence, $I - \gamma^{-2} EE^T P$ is invertible and $I - \gamma^{-2} EE^T P > 0$. □

Lemma 2.2*:* The optimal policies for control L, and disturbance K, in equation (2.15) and (2.17) respectively are equivalent to the ones that appear in [7], namely

$$\begin{gathered} L = -B^T P(I - BB^T P - \gamma^2 EE^T P)^{-1})A \\ K = -\gamma^{-2} E^T P(I - BB^T P - \gamma^2 EE^T P)^{-1})A. \end{gathered}$$

Proof: To show the control policy part, L, one can rewrite (2.15) as follows

$$\begin{gathered} L = (I + B^T P(I - E(E^T PE - \gamma^2 I)^{-1} E^T P)B)^{-1} \times \\ B^T P(E(E^T PE - \gamma^2 I)^{-1} E^T P - I)A. \end{gathered} \tag{2.18}$$

Applying the well known matrix inversion lemma, [15], on the (2.18), one has

$$L = -(I + B^T P(I - \gamma^2 EE^T P)^{-1} B)^{-1} B^T P(I - \gamma^{-2} EE^T P)^{-1} A\,. \tag{2.19}$$

Note that $(I - \gamma^2 EE^T P)$ is invertible due to lemma 2.1. Applying the matrix inversion lemma on (2.19), one has

$$L = -(I - B^T P(BB^T P + I - \gamma^2 EE^T P)^{-1} B)B^T P(I - \gamma^{-2} EE^T P)^{-1} A. \tag{2.20}$$

One can rewrite equation (2.20) as follows

$$L = -B^T P(I - (BB^T P + I - \gamma^2 EE^T P)^{-1} BB^T P) \times (I - \gamma^{-2} EE^T P)^{-1} A\,. \tag{2.21}$$

Applying the matrix inversion lemma on (2.21), one has

$$L = -B^T P(I + BB^T P - \gamma^2 EE^T P)^{-1})A\,. \tag{2.22}$$

Note that since $I - \gamma^{-2} EE^T P > 0$, then $I + BB^T P - \gamma^2 EE^T P > 0$ and concludes that equation (2.22) is equivalent to the control policy that appears on [7].

To show the control policy part, K, one can rewrite (2.17) as follows

$$K = (-\gamma^2 I + E^T P(E - B(I + B^T PB)^{-1} B^T P)E)^{-1} \times E^T P(B(I + B^T PB)^{-1} BP - I)A\,. \tag{2.23}$$

Applying the matrix inversion Lemma on (2.23), on has

$$K = -(-\gamma^2 I + E^T P(I + BB^T P)^{-1} E)^{-1} E^T P(I + BB^T P)A . \tag{2.24}$$

Applying the matrix inversion lemma on equation (2.24), one has

$$K = \gamma^{-2}(I + \gamma^{-2} E^T P(BB^T P + I - \gamma^2 EE^T P)^{-1} E)E^T P \times (I + BB^T P)^{-1} A . \tag{2.25}$$

One can rewrite (2.25) as follows

$$K = \gamma^{-2} E^T P(I + (BB^T P + I - \gamma^2 EE^T P)^{-1} \gamma^{-2} EE^T P) \times (I + BB^T P)^{-1} A . \tag{2.26}$$

Applying the matrix inversion lemma on equation (2.26), one obtains

$$K = \gamma^{-2} E^T P(I + BB^T P - \gamma^2 EE^T P)^{-1})A . \tag{2.27}$$

Note that since $I - \gamma^{-2} EE^T P > 0$, then $I + BB^T P - \gamma^2 EE^T P > 0$ and concludes that equation (2.27) is equivalent to the disturbance policy that appears in [7]. □

Next it is shown that the value function of the game $V^*(x_k) = x_k^T P x_k$ satisfies a Riccati equation. The form of the Riccati equation derived in this chapter, under state feedback information structure in order to perform ADP. It is similar to the one appearing in [3] which was derived under full information structure. Moreover, it will be shown that the Riccati equation derived in this chapter is equivalent to the work in [7] derived under the same state feedback information structure.

Note that (2.9) can be rewritten as follows

$$x_k^T P x_k = (x_k^T Q x_k + u^{*T}_k u^*_k - \gamma^2 w^{*T}_k w^*_k + (Ax_k + Bu^*_k + Ew^*_k)^T P(Ax_k + Bu^*_k + Ew^*_k)) . \tag{2.28}$$

This is equivalent

$$\begin{aligned} P &= Q + L^T L - \gamma^2 K^T K + (A + BL + EK)^T P(A + BL + EK)) \\ &= Q + L^T L - \gamma^2 K^T K + A_{cl}{}^T P A_{cl}. \end{aligned} \tag{2.29}$$

where $A_{cl} = A + BL + EK$. Equation (2.29) is the closed-loop Riccati equation.

Next it is shown upon substituting (2.15) and (2.17) in (2.29), one obtains the desired Riccati equation upon which the adaptive critic designs are based.

Lemma 2.3: Substituting the policies, (2.15) and (2.17), in (2.29) one can obtain the Riccati equation that appears in [3][34], and given by

$$P = A^T PA + Q - [A^T PB \quad A^T PE]\begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix}^{-1}\begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix}.$$

Proof: The control policy and the disturbance policy can be written as follows

$$L = D_{11}^{-1}(A_{12}A_{22}^{-1}E^T PA - B^T PA), \tag{2.30}$$

$$K = D_{22}^{-1}(A_{21}A_{11}^{-1}B^T PA - E^T PA), \tag{2.31}$$

where

$$\begin{aligned}
&D_{11}^{-1} = (I + B^T PB - B^T PE(E^T PE - \gamma^2 I)^{-1}E^T PB)^{-1} \\
&A_{12} = B^T PE \\
&A_{21} = E^T PB \\
&A_{11} = I + B^T PB \\
&A_{22} = E^T PE - \gamma^2 I \\
&D_{22}^{-1} = (E^T PE - \gamma^2 I - E^T PB(I + B^T PB)^{-1}B^T PE)^{-1}.
\end{aligned}$$

From (2.6) and (2.7), one concludes that D_{11}^{-1} and D_{22}^{-1} are invertible. Equations (2.30) and (2.31) can be written as follows

$$\begin{bmatrix} L \\ K \end{bmatrix} = -\begin{bmatrix} D_{11}^{-1} & -D_{11}^{-1}A_{12}A_{22}^{-1} \\ D_{22}^{-1}A_{21}A_{22}^{-1} & D_{22}^{-1} \end{bmatrix}\begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix}. \tag{2.32}$$

It is known that, [15],

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} D_{11}^{-1} & -D_{11}^{-1}A_{12}A_{22}^{-1} \\ D_{22}^{-1}A_{21}A_{22}^{-1} & D_{22}^{-1} \end{bmatrix}.$$

Therefore one can rewrite (2.32) as follows

$$\begin{bmatrix} L \\ K \end{bmatrix} = -\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix} = -\begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix}. \quad (2.33)$$

Equation (2.29) can be written as follows

$$\begin{aligned} P &= (A + BL + EK)^T P(A + BL + EK) + L^T L - \gamma^2 K^T K + Q \\ &= A^T PA + A^T PBL + A^T PEK + L^T B^T PA + K^T E^T PA \qquad (2.34) \\ &+ \begin{bmatrix} L^T & K^T \end{bmatrix} \begin{bmatrix} B^T PB & B^T PE \\ E^T PB & E^T PE \end{bmatrix} \begin{bmatrix} L \\ K \end{bmatrix} + \begin{bmatrix} L^T & K^T \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} L \\ K \end{bmatrix} + Q. \end{aligned}$$

Substituting (2.33) in (2.34), one has

$$\begin{aligned} P &= A^T PA + A^T PBL + A^T PEK + L^T B^T PA + K^T E^T PA \\ &- \begin{bmatrix} L^T & K^T \end{bmatrix} \begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix} \begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix} + Q \quad .(2.35) \\ &= A^T PA + A^T PBL + A^T PEK + Q. \end{aligned}$$

Equation (2.35) can be written as

$$P = A^T PA + \begin{bmatrix} A^T PB & A^T PE \end{bmatrix} \begin{bmatrix} L \\ K \end{bmatrix} + Q. \qquad (2.36)$$

Substituting (2.32) in (2.36), one has the desired Riccati equation

$$P = A^T PA + Q - [A^T PB \quad A^T PE] \begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix} \qquad (2.37)$$

It can be seen that (2.37) is the Riccati equation that appears in [3][34][32]. □

It is shown in [3] that (2.37) is equivalent to the Riccati equation that appears in [7] and [6], which is given as $P = Q + A^T P(I + (BB^T - \gamma^{-2} EE^T)P)^{-1} A$.

It is important to note that the H_2 problem is a special case of the H_∞ where in the system equation, (2.1) , $E = 0$ or in the value function,(2.2) , $\gamma \to \infty$, *i.e.* P will be the solution of the discrete-time algebraic Riccati equation DARE. One form of the DARE is

$$P = A^T PA + Q - A^T PB(I + B^T PB)^{-1} B^T PA$$

CHAPTER 3

HUERISTC DYNAMIC PROGRAMING H-INFINTY COTROL DESIGN

In this chapter, adaptive critic approximate dynamic programming designs are derived to solve the discrete-time zero-sum game in which the state and action spaces are continuous. This results in a forward-in-time reinforcement learning algorithm that converges to the Nash equilibrium of the corresponding zero-sum game. The results in this chapter can be thought of as a way to solve the Riccati equation of the well-known discrete-time H_∞ optimal control problem forward in time. Two schemes are presented, a Heuristic Dynamic Programming (HDP) and a Dual Heuristic Dynamic Programming (DHP) to solve for the value function and the co-state of the game respectively. An H_∞ autopilot design for an F-16 aircraft is presented to illustrate the results

3.1 Heuristic Dynamic Programming (HDP)

In this section, an HDP algorithm is developed to solve the discrete-time linear system zero-sum game described in chapter 2. The HDP algorithm was originally proposed in [25] to solve optimal control problems. The HDP algorithm has been applied earlier to solve the discrete-time Linear Quadratic Regulator (LQR) in optimal control theory [31]. In the HDP approach, a parametric structure is used to approximate the cost-to-go function of the current control policy. Then the certainty equivalence principle is used to improve the policy of the action network.

In this section, we extend the HDP approach to linear quadratic discrete-time zero-sum games appearing in [7], and prove the convergence of the presented algorithm.

3.1.1 Derivation of HDP for Zero-Sum Games

Consider the system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k \\ y_k &= x_k, \end{aligned} \tag{3.1}$$

and the cost-to-go function as

$$V(x_k) = \min_u \max_w \sum_{i=k}^{\infty} x_i^T Q x_i + u_i^T u_i - \gamma^2 w_i^T w_i \tag{3.2}$$

The HDP is developed to solve the zero-sum game described in chapter 2, one starts with an initial cost-to-go $V_0(x) \ge 0$ that is not necessarily optimal, and then finds $V_1(x)$ by solving equation (3.3) with $i = 0$ according to

$$V_{i+1}(x_k) = \min_{u_k} \max_{w_k} \left\{ x_k^T Q x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(x_{k+1}) \right\}. \tag{3.3}$$

Equation (3.3) is a recurrence relation that is used to solve for the optimal cost-to-go, the game value function, forward in time.

Note that since $V_i(x)$ is not initially optimal, optimal policies found using $V_i(x)$ in (3.3) use the certainty equivalence principle and are denoted as $u_i(x_k)$ and $w_i(x_k)$. Then, $V_{i+1}(x)$ is given by

$$V_{i+1}(x_k) = x_k^T Q x_k + u_i^T(x_k) u_i(x_k) - \gamma^2 w_i^T(x_k) w_i(x_k) + V_i(x_{k+1}). \tag{3.4}$$

Once $V_{i+1}(x)$ is found, one then repeats the same process for $i = 0,1,2,\ldots$. In this chapter, it is shown that $V_{i+1}(x_k) \to V^*(x_k)$ as $i \to \infty$, where $V^*(x_k)$ is the optimal value function for the game based on the solution to the GARE (2.37).

In the HDP approach, the cost-to-go function, $V_i(x)$, is generally difficult to obtain in closed-form except in special cases. Therefore, in general a parametric structure $\hat{V}(x, p_i)$, is used to approximate the actual $V_i(x)$. Similarly, parametric structures are used to obtain approximate closed-form representations of the two action networks $\hat{u}(x, L)$ and $\hat{w}(x, K)$. Since in this chapter the zero-sum game considered is linear and quadratic, it is well-known that the cost-to-go function is quadratic in the state, i.e. $V(x) = x^T P x$, and the two action networks are linear in the state. Therefore a natural choice of these parameter structures is given as

$$\hat{V}(x, p_i) = p_i^T \overline{x}, \tag{3.5}$$

$$\hat{u}(x, L_i) = L_i^T x, \tag{3.6}$$

$$\hat{w}(x, K_i) = K_i^T x, \tag{3.7}$$

where $\overline{x} = (x_1^2, \ldots, x_1 x_n, x_2^2, x_2 x_3, \ldots, x_{n-1} x_n, x_n^2)$, is the Kronecker product quadratic polynomial basis vector [21], and $p = v(P)$, where $v(\cdot)$ is a vector function that acts on $n \times n$ matrices and outputs a ${}^{n(n+1)}\!/_{2} \times 1$ column vector. The output vector of $v(\cdot)$ is constructed by stacking the columns of the squared matrix into a one-column vector with the off-diagonal elements summed as $P_{ij} + P_{ji}$, [21]. The parameter structures (3.5) (3.6) and (3.7) give an exact closed-form representation of the functions in (3.4).

It can be shown that the parameters of the action networks, L_i and K_i of (3.6) and (3.7), are found as

$$\begin{aligned} L_i = &(I + B^T P_i B - B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times \\ &(B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A), \end{aligned} \tag{3.8}$$

$$\begin{aligned} K_i = &(E^T P_i E - \gamma^2 I - E^T P_i B(I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times \\ &(E^T P_i B(I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A). \end{aligned} \tag{3.9}$$

These are greedy policy iterations that are based on the certainty equivalence principle when compared to (2.15) and (2.17), since they depend on P_i which does not necessarily solve (2.37). Note that to update the action networks, it is necessary to know the plant model A and B matrices.

After determining (3.8) and (3.9) substituting them in (3.4), one then has

$$d(x_k, p_i) = x_k^T Q x_k + (L_i x_k)^T (L_i x_k) - \gamma^2 (K_i x_k)^T (K_i x_k) + p_i^T \overline{x}_{k+1} \tag{3.10}$$

which can be thought of as the desired target function to which one needs to fit $\hat{V}(x, p_{i+1})$ in least-squares sense to find p_{i+1} such that

$$p_{i+1}^T \overline{x}_k = d(x_k, p_i) \tag{3.11}$$

The parameter vector p_{i+1} is found by minimizing the error between the target value function (3.10) and (3.11) in a least-squares sense over a compact set, Ω,

$$p_{i+1} = \arg\min_{p_{i+1}} \{ \int_{\Omega} | p_{i+1}^T \overline{x} - d(x, p_i) |^2 \, dx \} . \tag{3.12}$$

3.1.2 Online Implementation of the HDP Algorithm

The least-squares problem in (3.12) can be solved in real-time by collecting enough data points generated from $d(x_k, p_i)$ in (3.10). This requires one to have knowledge of the state information x_k, x_{k+1} as the dynamics evolve in time, and also of the reward function $r(x_k, u_k, w_k)$. This can be determined by simulation, or, in real-time applications, by observing the states on-line. Therefore, in the HDP algorithm, the model of the system is not needed to update the critic network, though it is needed to update the actions.

To satisfy the excitation condition of the least-squares problem, one needs to have the number of collected points N at least

$$N \geq n(n+1)/2,$$

where n is the number of states. Therefore, after several time steps that are enough to guarantee the excitation condition, one has the following least-squares problem

$$p_{i+1} = (XX^T)^{-1} XY, \tag{3.13}$$

where

$$X = [\bar{x}|_{x_{k-N-1}} \quad \bar{x}|_{x_{k-N-2}} \quad \cdots \quad \bar{x}|_{x_{\text{k-1}}}]$$
$$Y = [d(x_{k-N-1}, p_i) \quad d(x_{k-N-2}, p_i) \quad \cdots \quad d(x_{\text{k-1}}, p_i)]^T.$$

One can solve (3.13) recursively using the well-known recursive least-squares technique. In that case, the excitation condition is replaced by the persistency of excitation condition

$$\varepsilon_0 I \leq \frac{1}{\alpha} \sum_{m=1}^{\alpha} \bar{x}_{k-t} \bar{x}_{k-t}^T \leq \varepsilon_1 I$$

for all $k > \alpha_0$, $\alpha > \alpha_0$, with $\varepsilon_0 \leq \varepsilon_1$, ε_0 and ε_1 positive integers and $\varepsilon_0 \leq \varepsilon_1$.

The recursive least-squares algorithm is given as

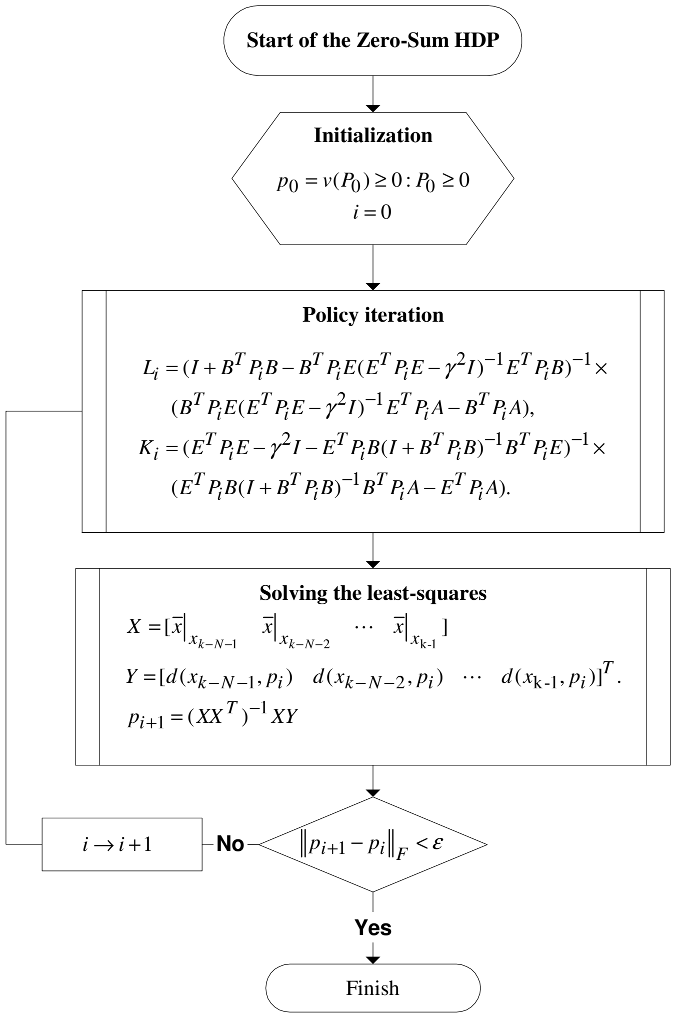
$$e_i(t) = d(x_k, p_i) - \overline{x}_k^T \, p_{i+1}(t-1)$$

$$p_{i+1}(t) = p_{i+1}(t-1) + \frac{\Gamma_i(t-1)\overline{x}_k \, e_i(t)}{1 + \overline{x}_k^T \Gamma_i(t-1)\overline{x}_k}$$

$$\Gamma_i(t) = \Gamma_i(t-1) - \frac{\Gamma_i(t-1)\overline{x}_k \, \overline{x}_k^T \Gamma_i(t-1)}{1 + \overline{x}_k^T \Gamma_i(t-1)\overline{x}_k}$$

where i is the policy update index, t is the index of the recursions of the recursive least-squares, and k is the discrete time. Γ is the covariance matrix of the recursion and $e(t)$ is the estimation error of the recursive least-squares. Note that $\Gamma_i(0)$ is a large number and $\Gamma_{i+1}(0) = \Gamma_i$.

The on-line HDP algorithm developed in this chapter is summarized in the flowchart shown in Figure 3.1. The HDP algorithm for zero-sum games follows by iterating between (3.8) (3.9) and (3.13). As will be shown show next, this will cause P_i to converge to the optimal P, when it exists, that solves the GARE associated with the discrete time zero-sum game given in (2.37). Note that the model of the system is needed in the HDP algorithm to update the actions networks only.


Start of the Zero-Sum HDP
Initialization
$p_0 = v(P_0) \geq 0 : P_0 \geq 0$
$i = 0$
Policy iteration
$L_i = (I + B^T P_i B - B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times (B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A),$
$K_i = (E^T P_i E - \gamma^2 I - E^T P_i B(I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times (E^T P_i B(I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A).$
Solving the least-squares
$X = [\bar{x}|_{x_{k-N-1}} \quad \bar{x}|_{x_{k-N-2}} \quad \cdots \quad \bar{x}|_{x_{k-1}}]$
$Y = [d(x_{k-N-1}, p_i) \quad d(x_{k-N-2}, p_i) \quad \cdots \quad d(x_{k-1}, p_i)]^T.$
$p_{i+1} = (XX^T)^{-1} XY$
$i \rightarrow i+1$
No
$\|p_{i+1} - p_i\|_F < \varepsilon$
Yes
Finish
Figure 3.1. The HDP algorithm

3.1.3 Convergence of the HDP Algorithm

Now the proof that the proposed HDP algorithm for zero-sum games converges to the optimal policies is given.

Lemma 3.1 Iterating on equations (3.8) (3.9) and (3.12) is equivalent to the iteration on the Riccati equation (2.37) associated with zero-sum games problem. That is

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}, \quad (3.14)$$

under the assumption that the system is sufficiently excited.

Proof: The least-squares problem is defined in (3.12) which is

$$p_{i+1} = \arg\min_{p_{i+1}} \{ \int_{\Omega} | p_{i+1}^T \overline{x} - d(x, p_i) |^2 \, dx \} .$$

This can be rewritten as

$$\int_{\Omega} (2\overline{x}\overline{x}^T p_{i+1} - 2\overline{x} d^T (x, p_i)) dx = 0 . \tag{3.15}$$

and implies that

$$p_{i+1} = \left(\int_{\Omega} \overline{x}\overline{x}^T dx \right)^{-1} \int_{\Omega} \overline{x} d(x, p_i) dx . \tag{3.16}$$

Under the excitation condition assumption, the inverse operator exists. Substituting (3.10) in (3.16), one has

$$p_{i+1} = \left(\int_{\Omega} \overline{x}_k \overline{x}_k^T dx \right)^{-1} \int_{\Omega} \overline{x}_k (x_k^T (Q + L_i^T L_i + (A + BL_i + EK_i)^T P_i ((A + BL_i + EK_i) x_k) dx . \tag{3.17}$$

Using the Kronecker products [21], equation (3.17) can be written as

$$\begin{aligned} p_{i+1} = & \left(\int_{\Omega} \overline{x}_k \overline{x}_k^T dx \right)^{-1} \left(\int_{\Omega} \overline{x}_k \overline{x}_k^T dx \right) \times \\ & v(Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i)) \\ & = v(Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i)), \end{aligned}$$

where v is the vectorized function in the Kronecker product.

Since the matrix P_{i+1} which reconstructed from p_{i+1} is symmetric, iteration on p_i is equivalent to the following iteration

$$P_{i+1} = Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i) . \tag{3.18}$$

Using the same steps as in Lemma 2.3 equation (3.18) can be written

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix} \quad (3.19)$$

which is equivalent to (3.14). □

Theorem 3.1: Assume that the game has a value and is solvable. If the sequence of least-squares problems in (3.12) is solvable, i.e. the corresponding excitation conditions hold, then the HDP algorithm converges to the value of the game that solves the Riccati equation (2.37) when starting with $P_0 \geq 0$.

Proof. This follows from Lemma 3.1 and from [3] where it is shown that iterating on (3.14) with $P_0 \geq 0$ converges to P that solves (2.37). □

The proof of convergence of the HDP algorithm has just been established assuming the least-squares problem (3.12) is solved completely; i.e. the excitation condition is satisfied. Note that an easy way to initialize the algorithm in Figure 3.1 is by selecting $P_0 = 0$.

3.2 Dual Heuristic Dynamic Programming (DHP)

In this section, a DHP algorithm is developed to solve the discrete-time linear system zero-sum game described in chapter 2. The DHP algorithm has been applied earlier to solve the discrete-time Linear Quadratic Regulator (LQR) in optimal control theory [31].

In the DHP approach, a parametric structure is used to approximate the co-state function, i.e. the gradient of the cost-to-go function, of the current control policies. As in the HDP case, the certainty equivalence principle is used to improve the policies of the actions networks.

In this section, the DHP is extended approach to linear quadratic discrete-time zero-sum games appearing in [7], and prove the convergence of the presented algorithm.

3.2.1 Derivation of DHP for Zero-Sum Games

Consider the system (3.1) and the cost-to-go function (3.2). In the DHP approach, the critic network approximates the co-state $\lambda^*(x_k)$ forward in time. It is known [6] that the co-state of the zero-sum game is the gradient of the game value function given as

$$\lambda^*(x_k^*) = \frac{\partial V^*(x_k^*)}{\partial x_k^*}. \tag{3.20}$$

In the zero-sum game DHP algorithm developed in this chapter, the following recurrence relation is derived to solve for the co-state forward in time.

$$\begin{aligned}
\lambda_{i+1}(x_k) &= \frac{\partial V_{i+1}(x_k)}{\partial x_k} \\
&= \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial x_k} + \left(\frac{\partial u_i(x_k)}{\partial x_k}\right)^T \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial u_i(x_k)} + \\
&\left(\frac{\partial w_i(x_k)}{\partial x_k}\right)^T \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial w_i(x_k)} + \\
&\left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} + \left(\frac{\partial u_i(x_k)}{\partial x_k}\right)^T \left(\frac{\partial x_{k+1}}{\partial u_i(x_k)}\right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} + \\
&\left(\frac{\partial w_i(x_k)}{\partial x_k}\right)^T \left(\frac{\partial x_{k+1}}{\partial w_i(x_k)}\right)^T \frac{\partial V_i(x_{k+1})}{\partial x_{k+1}},
\end{aligned} \tag{3.21}$$

The recurrence relation (3.21) is obtained by differentiating the recurrence relation on the cost-to-go function (3.4). Equation (3.21) can be rewritten as

$$\begin{aligned}\lambda_{i+1}(x_k) = {} & \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial x_k} + \\ & \left(\frac{\partial u_i(x_k)}{\partial x_k}\right)^T \left\{\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial u_i(x_k)} + \left(\frac{\partial x_{k+1}}{\partial u_i(x_k)}\right)^T \lambda_i(x_{k+1})\right\} + \\ & \left(\frac{\partial w_i(x_k)}{\partial x_k}\right)^T \left\{\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial w_i(x_k)} + \left(\frac{\partial x_{k+1}}{\partial w_i(x_k)}\right)^T \lambda_i(x_{k+1})\right\} + \\ & \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda_i(x_{k+1}).\end{aligned} \tag{3.22}$$

As was the case in (3.8) and (3.9) in the HDP case, the improvement of the actions networks requires that

$$\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial u_i(x_k)} + \left(\frac{\partial x_{k+1}}{\partial u_i(x_k)}\right)^T \lambda_i(x_{k+1}) = 0, \tag{3.23}$$

$$\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial w_i(x_k)} + \left(\frac{\partial x_{k+1}}{\partial w_i(x_k)}\right)^T \lambda_i(x_{k+1}) = 0. \tag{3.24}$$

Combining (3.22), (3.23) and (3.24), one has

$$\lambda_{i+1}(x_k) = \frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial x_k} + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda_i(x_{k+1}). \tag{3.25}$$

Hence, the DHP algorithm can be summarized as the successive iteration between (3.22) on one hand, and (3.23) and (3.24) on the other. This results in a successive-improvement of the value function derivatives sequence $\{\lambda_i \mid i = 1, 2, \ldots\}$ as the player policies are generated. In this chapter, it will be shown that the DHP algorithm converges to the co-state value function (3.20). When converged, equation (3.25) becomes

$$\lambda^*(x_k^*) = \frac{\partial r(x_k^*, u_i(x_k^*), w_i(x_k^*))}{\partial x_k^*} + \left(\frac{\partial x_{k+1}}{\partial x_k^*}\right)^T \lambda^*(x_{k+1}^*)$$

which is known as the co-state equation in Theorem 6.3 in [6].

In DHP, a parametric structure $\hat{\lambda}(x, p_i)$ is used to approximate the actual $\lambda_i(x)$. Similarly, parametric structures are used to obtain approximate closed-form representations of the two action networks $\hat{u}(x, L)$ and $\hat{w}(x, K)$. Since in this chapter the zero-sum game considered is linear and quadratic, it is well-known that co-state and the action networks are all linear in the state. Therefore a natural choice of these parameter structures is given as

$$\hat{\lambda}(x, p_i) = p_i^T \frac{\partial \overline{x}}{\partial x}, \tag{3.26}$$

$$\hat{u}(x, L_i) = L_i^T x, \tag{3.27}$$

$$\hat{w}(x, K_i) = K_i^T x, \tag{3.28}$$

where $\overline{x}$ and p are as described in equations (3.5). The parameter structures (3.26) (3.27) and (3.28) give an exact closed-form representation of the functions in (3.21) to (3.24).

Using the parameter structures (3.27) and (3.28) along with the certainty equivalence principle, it can be easily shown that the parameters of the actions networks are updates as in (3.8) and (3.9) respectively.

Substituting (3.8) and (3.9) in (3.21) given the system model (2.1), one has

$$d(x_k, P_i) = 2Qx_k + 2L_i^T u_k - 2\gamma^2 K_i^T w_k + (A + BL_i + EK_i)^T P_i x_{k+1}. \tag{3.29}$$

Using the Kroncker product notation [21], equation (3.29) can be rewritten as

$$d(x_k, p_i) = v(Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i))^T \frac{\partial \overline{x}}{\partial x}. \quad (3.30)$$

Equation (3.30) can be thought of as the desired target function to which one needs to fit $\hat{\lambda}(x, p_{i+1})$ such that $\hat{\lambda}(x, p_{i+1}) = d(x_k, p_i)$.

The parameter vector p_{i+1} is found by minimizing the error between the target value function (3.30) and (3.26) in a least-squares sense over a compact set, Ω,

$$p_{i+1} = \arg\min_{p_{i+1}} \int_{\Omega} \{| p_{i+1}^T \frac{\partial \overline{x}^T}{\partial x} - d_x(x_k, p_i) |^2\} dx . \quad (3.31)$$

3.2.2 Online Implementation of the DHP Algorithm

The least-squares problem in (3.31) can be solved in real-time by collecting enough data points generated from $d(x_k, p_i)$ in (3.29). This requires having access to the state information x_k, x_{k+1} as the dynamics evolve in time and gradients of the reward function $\partial r / \partial x_k$, $\partial r / \partial u_i$, $\partial r / \partial w_i$, as well as the plant model A and B. Therefore, in the DHP algorithm developed in this chapter for the zero-sum game, the plant model is required to update the critic network.

To satisfy the excitation condition of the least squares problem, one needs to have the number of collected points N at least $N \geq n$, where n is the number of states. Therefore, after several time steps that are enough to guarantee the excitation condition, one has the following least-squares problem

$$p_{i+1} = (XX^T)^{-1} XY , \quad (3.32)$$

where

$$X = [\left.\frac{\partial \overline{x}}{\partial x}\right|_{x_{k-N-1}} \quad \left.\frac{\partial \overline{x}}{\partial x}\right|_{x_{k-N-2}} \quad \cdots \quad \left.\frac{\partial \overline{x}}{\partial x}\right|_{x_{k-1}}]$$
$$Y = [d^T(x_{k-N-1}, p_i) \quad d^T(x_{k-N-2}, p_i) \quad \cdots \quad d^T(x_{k-1}, p_i)]^T.$$

One can solve (3.32) recursively using the well-known recursive least squares technique. In that case, the excitation condition is replaced with the persistency of excitation condition

$$\varepsilon_0 I \le \frac{1}{\alpha}\sum_{m=1}^{\alpha} \frac{\partial \overline{x}}{\partial x}\Big|_{k-t} \frac{\partial \overline{x}}{\partial x}^T \Big|_{k-t} \le \varepsilon_1 I$$

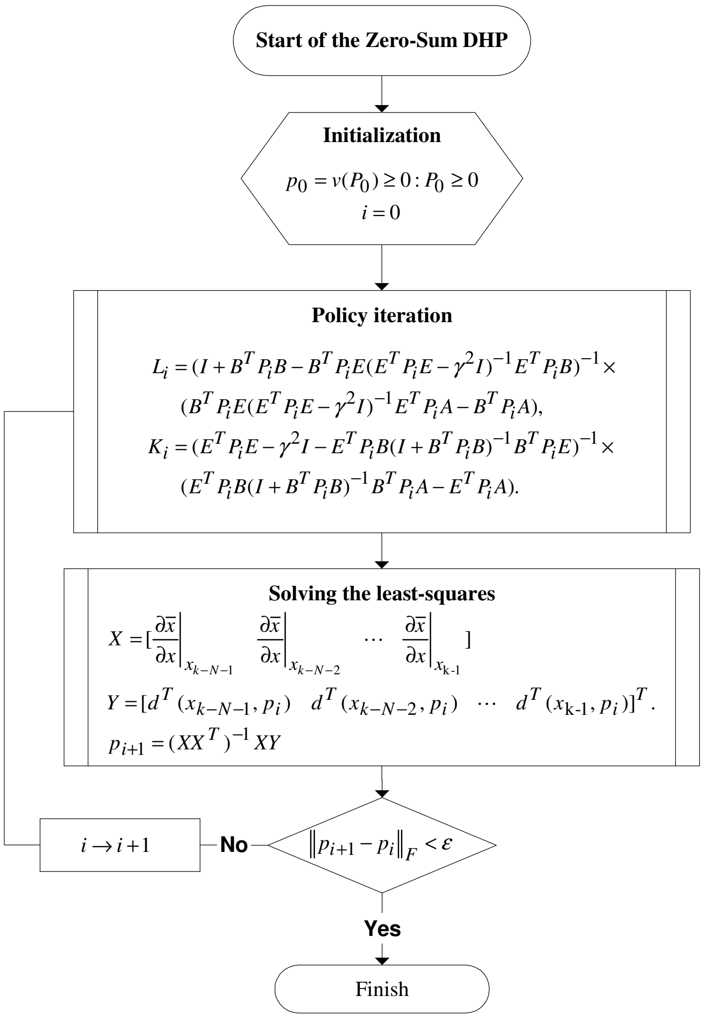
for all $k > \alpha_0$, $\alpha > \alpha_0$, with $\varepsilon_0 \le \varepsilon_1$, ε_0 and ε_1 are positive integers and $\varepsilon_0 \le \varepsilon_1$.

The recursive least-squares algorithm is given as

$$e_i(t) = d_x(x_k, p_i) - \frac{\partial \overline{x}_k^T}{\partial x} p_{i+1}(t-1)$$
$$p_{i+1}(t) = p_{i+1}(t-1) + \Gamma_i(t-1)\frac{\partial \overline{x}_k}{\partial x}\left(I + \frac{\partial \overline{x}_k^T}{\partial x}\Gamma_i(t-1)\frac{\partial \overline{x}_k}{\partial x}\right)^{-1} e_i(t)$$
$$\Gamma_i(t) = \Gamma_i(t-1) - \Gamma_i(t-1)\frac{\partial \overline{x}_k}{\partial x}\left(I + \frac{\partial \overline{x}_k^T}{\partial x}\Gamma_i(t-1)\frac{\partial \overline{x}_k}{\partial x}\right)^{-1}\frac{\partial \overline{x}_k^T}{\partial x}\Gamma_i(t-1)$$

where i is the policy update index, t is the index of the recursions of the recursive least-squares, and k is the discrete time. Γ is the covariance matrix of the recursion and $e(t)$ is the estimation error of the recursive least-squares. Note that $\Gamma_i(0)$ is a large number, and $\Gamma_{i+1}(0) = \Gamma_i$.

The developed DHP algorithm is summarized in the flowchart shown in figure 3.2.



Start of the Zero-Sum DHP
Initialization
$p_0 = v(P_0) \geq 0 : P_0 \geq 0$
$i = 0$
Policy iteration
$L_i = (I + B^T P_i B - B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times (B^T P_i E(E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A),$
$K_i = (E^T P_i E - \gamma^2 I - E^T P_i B(I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times (E^T P_i B(I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A).$
Solving the least-squares
$X = [\left.\frac{\partial \bar{x}}{\partial x}\right|_{x_{k-N-1}} \quad \left.\frac{\partial \bar{x}}{\partial x}\right|_{x_{k-N-2}} \quad \cdots \quad \left.\frac{\partial \bar{x}}{\partial x}\right|_{x_{k-1}}]$
$Y = [d^T(x_{k-N-1}, p_i) \quad d^T(x_{k-N-2}, p_i) \quad \cdots \quad d^T(x_{k-1}, p_i)]^T.$
$p_{i+1} = (XX^T)^{-1} XY$
$i \rightarrow i+1$
No
$\|p_{i+1} - p_i\|_F < \varepsilon$
Yes
Finish

Figure 3.2. The DHP algorithm

The DHP algorithm for the zero-sum game considered in this chapter follows by iterating between (3.8) (3.9) and (3.32). Next it will be shown that this will cause P_i to converge to the optimal P, when it exists, which solves the GARE associated with the discrete time zero-sum game given in (2.37) . Note that the model of the system is needed in the DHP algorithm in both the critic network and the actions networks.

3.2.3 Convergence of the DHP Algorithm

Lemma 3.2: Iterating on equation (3.8) (3.9) and (3.31) is equivalent to the iteration on the Riccati equation (2.37) associated with the zero-sum game problem. That is

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix} \quad (3.33)$$

under the assumption that the system is sufficiently excited.

Proof: The least-squares problem is defined in equation (3.31) which is

$$p_{i+1} = \arg\min_{p_{i+1}} \int_{\Omega} \{ | \, p_{i+1}^T \frac{\partial \overline{x}^T}{\partial x} - d_x(x_k, p_i) \, |^2 \} dx .$$

This can be written as

$$\int_{\Omega} \left\{ 2 \frac{\partial \overline{x}}{\partial x} \frac{\partial \overline{x}^T}{\partial x} p_{i+1} - 2 \frac{\partial \overline{x}}{\partial x} d^T(x_k, P_i) \right\} dx = 0 , \quad (3.34)$$

and that implies

$$p_{i+1} = \left(\int_{\Omega} \frac{\partial \overline{x}}{\partial x} \frac{\partial \overline{x}^T}{\partial x} dx \right)^{-1} \int_{\Omega} \frac{\partial \overline{x}}{\partial x} d^T(x_k, p_i).dx . \quad (3.35)$$

Under the excitation condition assumption, the inverse operator exists.

Substituting (3.30) in (3.35), one has

$$p_{i+1} = \left(\int_{\Omega} \frac{\partial \overline{x}^T}{\partial x} \frac{\partial \overline{x}^T}{\partial x} dx \right)^{-1} \times$$
$$\int_{\Omega} \frac{\partial \overline{x}}{\partial x} \frac{\partial \overline{x}^T}{\partial x} v(Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i)) dx,$$

which can be written as

$$p_{i+1} = v(Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i)) ,$$

where v is the vectorized function in the Kronecker product.

Since the matrix P_{i+1} reconstructed from p_{i+1} is symmetric, iteration on p_i is equivalent to the following iteration

$$P_{i+1} = Q + L_i^T L_i - \gamma^2 K_i^T K_i + (A + BL_i + EK_i)^T P_i (A + BL_i + EK_i). \tag{3.36}$$

Using the same steps as in Lemma 2.3 equation (3.36) can be written

$$P_{i+1} = A^T P_i A + Q - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}. \tag{3.37}$$

which is equivalent to (3.33). □

Theorem 3.2: Assume that the game has a value and is solvable. If the sequence of least-squares problems in (3.31), i.e. the corresponding excitation conditions hold, then the DHP algorithm converges to the value of the game that solves the Riccati equation (2.37) when starting with $P_0 \geq 0$.

Proof. This follows from Lemma 3.2 and from [3] where it is shown that iterating on (3.36) with $P_0 \geq 0$ converges to P that solves (2.37). □

The convergence proof of the DHP algorithm has just been establised assuming the least-squares problem (3.32) is solved completely; i.e. the exciting condition is satisfied. Note that an easy way to initialize the algorithm in Figure 3.2 is by selecting $P_0 = 0$.

In the next section, the developed HDP and DHP zero-sum game algorithms are used to derive suboptimal H_∞ controllers by the forward time solution technique. The practical relevance of the developed algorithms will thus become clear.

3.3 Online ADP H_∞ Autopilot Controller Design for an F-16 aircraft

In this design application, the zero-sum game that corresponds to the H_∞ controller problem is solved for an F-16 aircraft autopilot design. The H-infinity approach is used, which is enabled by the ADP procedures in this chapter. H-infinity design has been proven highly effective in the design of feedback control systems with robustness and disturbance rejection capabilities [15].

The F-16 short period dynamics has three states given as

$$x = \begin{bmatrix} \alpha \\ q \\ \delta_e \end{bmatrix}$$

where α is the angle of attack, q is the pitch rate and δ_e is the elevator deflection angle. The discrete-time plant model of this aircraft dynamics is a discretized version of the continuous-time one given in [4]. We used standard zero-order-hold discretization techniques explained in [14] and easily implemented in the MATLAB control systems toolbox to obtain the sampled data plant

$$A = \begin{bmatrix} 0.906488 & 0.0816012 & -0.0005 \\ 0.0741349 & 0.90121 & -0.000708383 \\ 0 & 0 & 0.132655 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.00150808 \\ -0.0096 \\ 0.867345 \end{bmatrix} \quad E = \begin{bmatrix} 0.00951892 \\ 0.00038373 \\ 0 \end{bmatrix}. \tag{3.38}$$

with sampling time $T = 0.1$. In this H_∞ design problem, the disturbance attenuation is $\gamma = 1$.

3.3.1 H_∞ Solution Based on the Riccati Equation

Since the ADP designs developed in this chapter to solve the H_∞ controller design problem are based on an iterative form of the Riccati equation (3.14), in Figure 3.3 the convergence of P_i to the solution of the GARE (2.37) is shown when done offline with $P_0 = 0$.

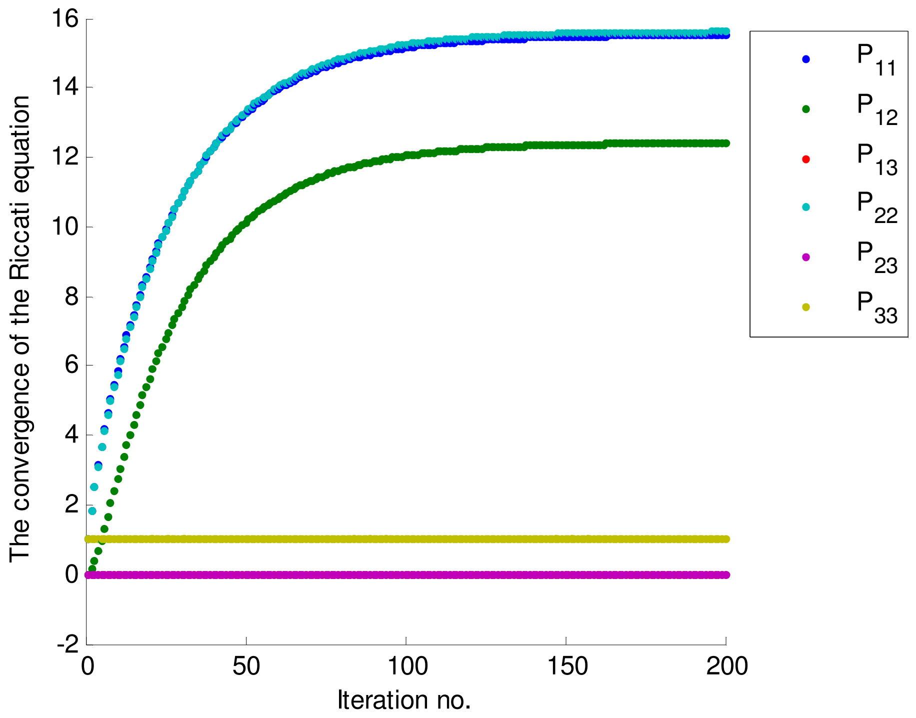


Figure 3.3. The convergence of P_i by iterating on Riccati equation

It is noticed from Figure 3.3 that for the discretized aircraft dynamics(3.38), P_i converges after at least 100 iterations with $\gamma = 1$ to

$$P = \begin{bmatrix} 15.5109 & 12.4074 & -0.0089 \\ 12.4074 & 15.5994 & -0.0078 \\ -0.0089 & -0.0078 & 1.0101 \end{bmatrix} \tag{3.39}$$

which solves the GARE (2.37). Note that $P \geq 0$ and hence from [7] this implies that

$$\frac{\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^{*T} u_k^*}{\sum_{k=0}^{\infty} w_k^T w_k} \leq \gamma^2 \tag{3.40}$$

for all finite energy disturbances, i.e.

$$\sum_{k=0}^{\infty} w_k^T w_k \ ,$$

are bounded, and hence $u^*(x_k)$ has the well-known robustness and disturbance rejection capabilities of H_∞ control.

Next, the ADP algorithms developed in this chapter are used to design an H_∞ controller for the discretized aircraft dynamics (3.38) with $\gamma = 1$ in forward time.

3.3.2 HDP based H_∞ Autopilot Controller Design

In this part, the HDP algorithm developed in Section 3.1 of this chapter is applied to solve for the H_∞ autopilot controller in forward time. The recursive least-squares algorithm is used to tune the parameters of the critic network on-line. The parameters of the actions networks are updated according to (3.8) and (3.9). It is important to mention that using the LS to tune the parameter will cause faster convergence than using the RLS.

In this HDP design, the states of the aircraft are initialized to be $x_0 = \begin{bmatrix} 4 & 2 & 5 \end{bmatrix}$. Any values could be selected. The parameters of the critic network and the actions networks are initialized to zero. Following this initialization step, the aircraft dynamics are run forward in time and tuning of the parameter structures is performed by observing the states on-line.

In Figures 3.4 and 3.5, the states and the inputs to the aircraft are shown with respect to time. In order to maintain the excitation condition, one can use several standard schemes, including covariance resetting, state resetting, or injection of a small probing noise signal. In this example, state resetting is used and the states are re-initialized to $x_0 = [4 \quad 2 \quad 5]$ periodically to prevent them from converging to zero. Hence the persistency of excitation condition required for the convergence of the recursive least-squares tuning, i.e. avoiding the parameter drift problem, will hold. State re-initialization has appeared recently in [19] to solve the HJB equation associated with continuous-time optimal control problems.

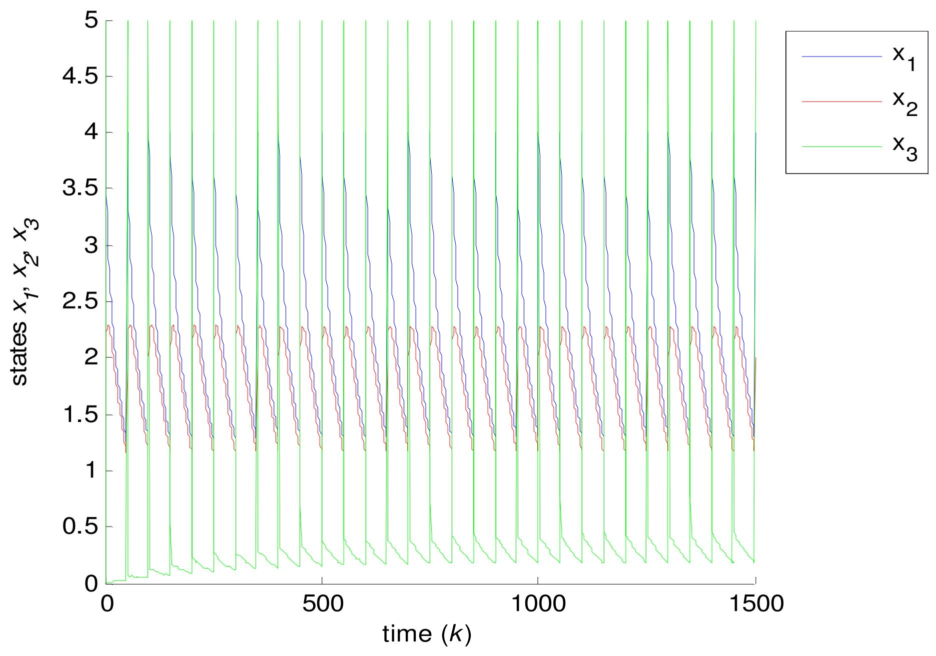


Figure 3.4. States trajectories with re-initialization for the HDP algorithm.

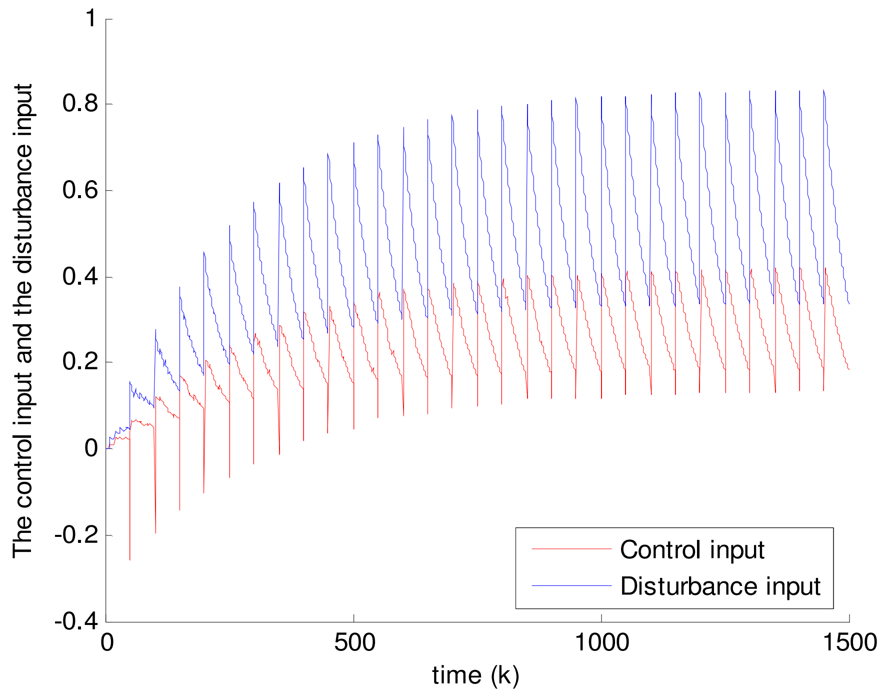


Figure 3.5. The control and disturbance in the HDP

In Figures 3.6, 3.7 and 3.8, the convergence of the parameters of the critic network, and the actions networks is shown. As expected, the parameters of the critic network converge to P in (3.39) that solves the GARE equation. It takes the critic network 1500 time steps to converge to P. The reason for this is that 10 readings are required to tune the critic network at each update to solve for each P_i. Since as shown in Figure 3.3, the action networks require to be updated at least 100 times, this implies that the over all time steps required for the convergence of the HDP algorithm are about 1000 time steps.

It is important to realize that state is used resetting here to determine the optimal solution for the game problem, as given by the converged critic network parameters in Figure 3.6 and action network parameters in Figures 3.7 and 3.8. State resetting provides the excitation conditions needed to get parameter convergence. Once these parameters are known, the H_∞ controller has been found. Then, one can use the parameters of the control action network as the final parameters of the controller in any on-line control runs, without having to deliberately insert any excitation signals to the system.

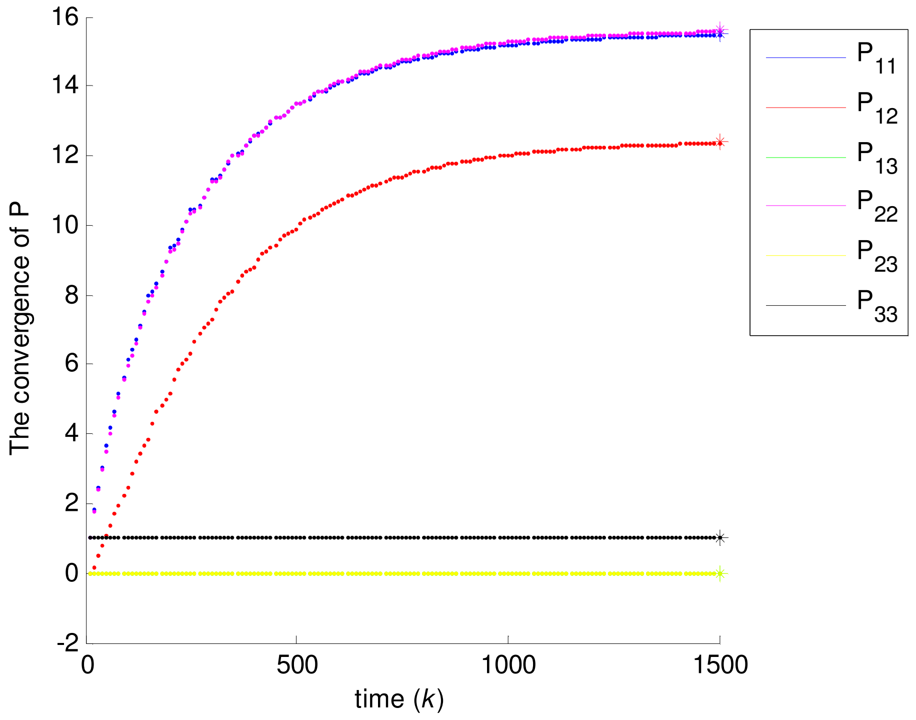


Figure 3.6. Convergence of the critic network parameters in the HDP.

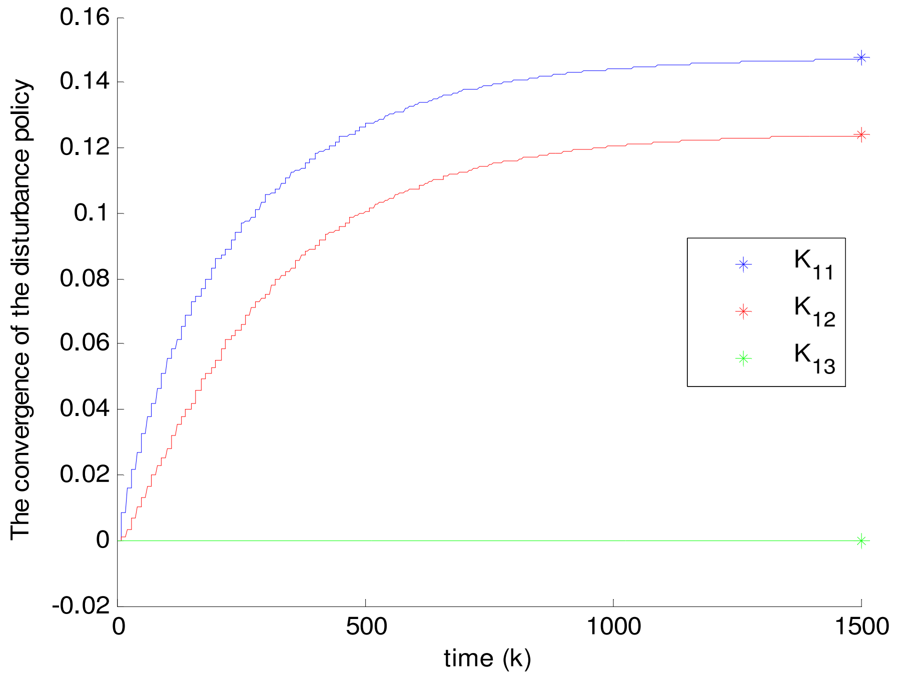


Figure 3.7. Convergence of the disturbance action network parameters in the HDP.

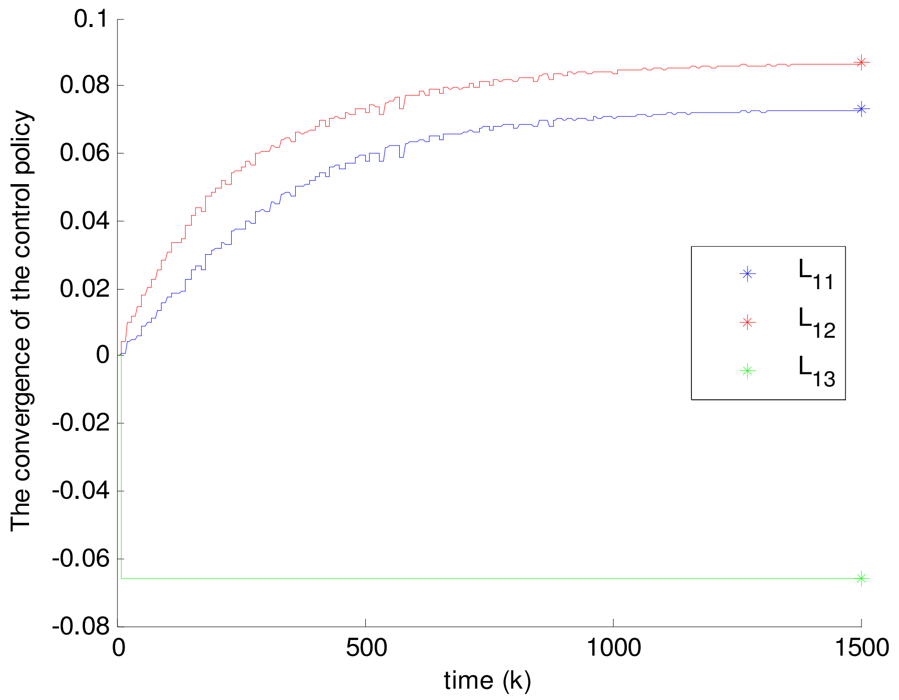


Figure 3.8. Convergence of the control action network parameters in the HDP.

Next, the DHP algorithm developed in Section 3.4 is applied to this aircraft design problem.

3.3.3 DHP based H_∞ Autopilot Controller Design

In this part, the DHP algorithm developed in Section 3.2 of this chapter is applied to solve for the H_∞ autopilot controller in forward time. The recursive least-squares algorithm is used to tune the parameters of the critic network. The parameters of the actions networks are updated according to (3.8) and (3.9).

In this DHP design, the states of the aircraft are initialized to be $x_0 = \begin{bmatrix} 4 & 2 & 5 \end{bmatrix}$. The parameters of the critic network and the actions networks are initialized to zero. Following this initialization step, the aircraft dynamics are run forward in time and tuning of the parameter structures happen by observing the states on-line.

In Figures 3.9 and 3.10, the states and the inputs to the aircraft are shown with respect to time. Note that the states are re-initialized to $x_0 = \begin{bmatrix} 4 & 2 & 5 \end{bmatrix}$ to prevent them from converging to zero. Hence the persistency of excitation condition required for the convergence of the recursive least-squares tuning, i.e. avoiding the parameter drift problem, will hold.

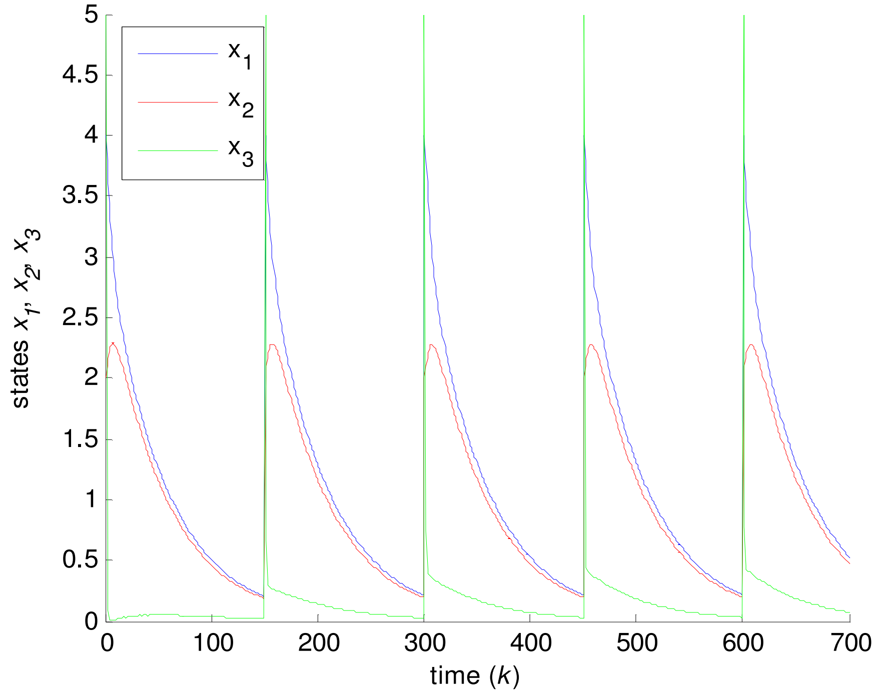


Figure 3.9. States trajectories with re-initialization for the DHP algorithm

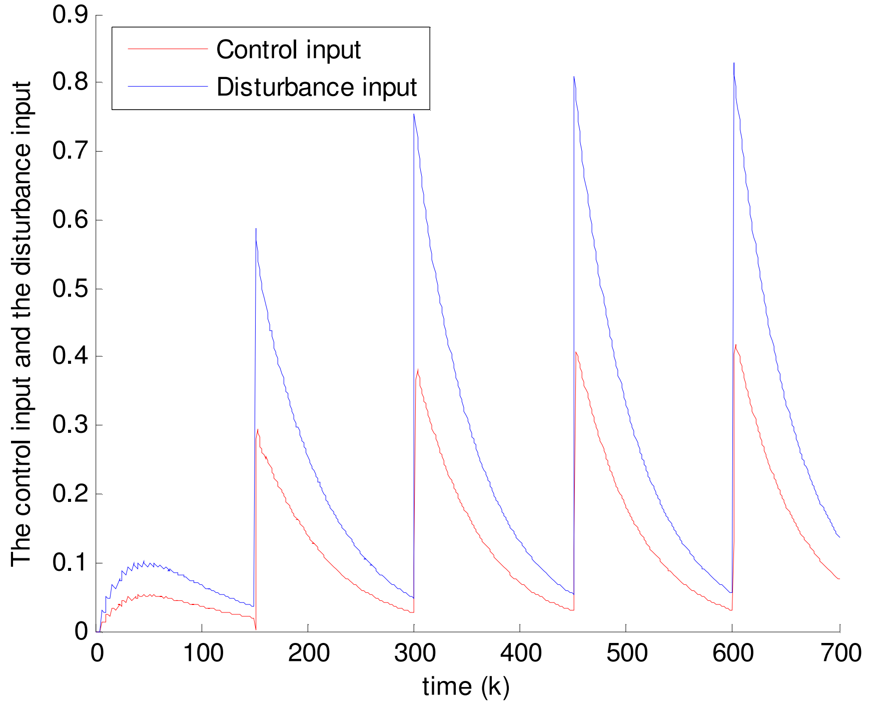


Figure 3.10. The control and disturbance in the DHP

In Figures 3.11, 3.12 and 3.13, the convergence of the parameters of the critic network, and the action networks is shown. As expected, the parameters of the critic network converge to P in (3.39) that solves the GARE equation. It takes the critic network 600 time steps to converge to P. The reason for this is that 6 readings are required to tune the critic network at to solve for each P_i. Since as shown in Figure 3, the action networks require to be updated at least 100 times, this implies that the over all time steps required for the convergence of the DHP algorithm are about 600 time steps.

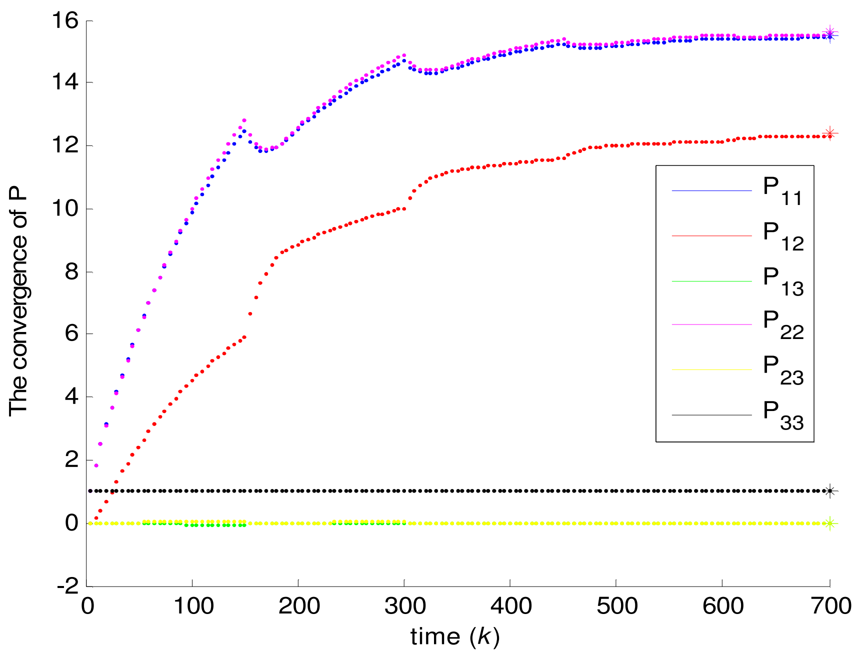


Figure 3.11. Convergence of the critic network parameters in the DHP.

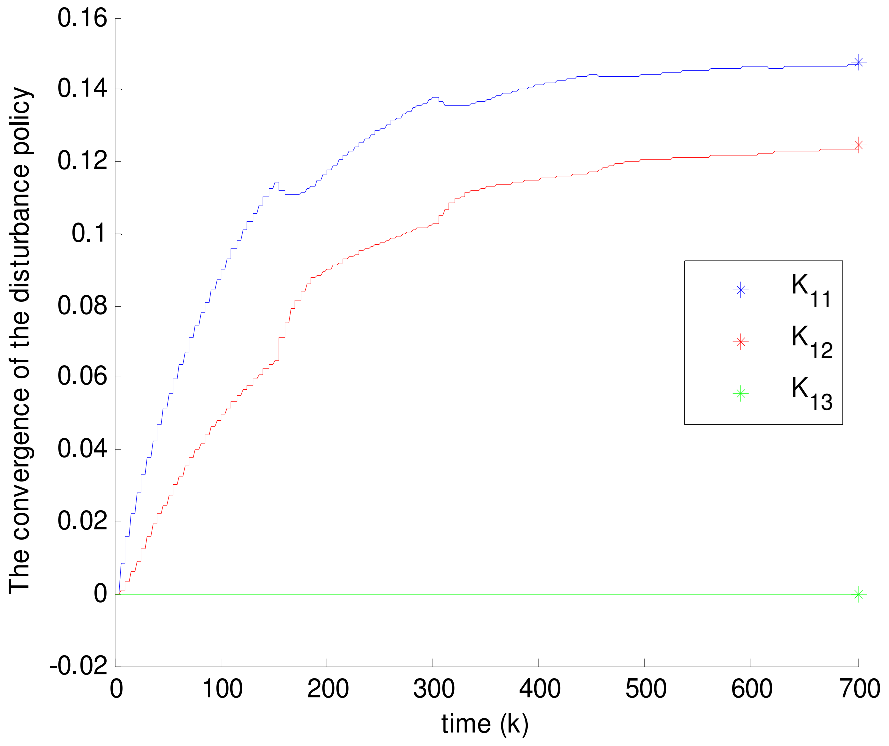


Figure 3.12. Convergence of the disturbance action network parameters in the DHP.

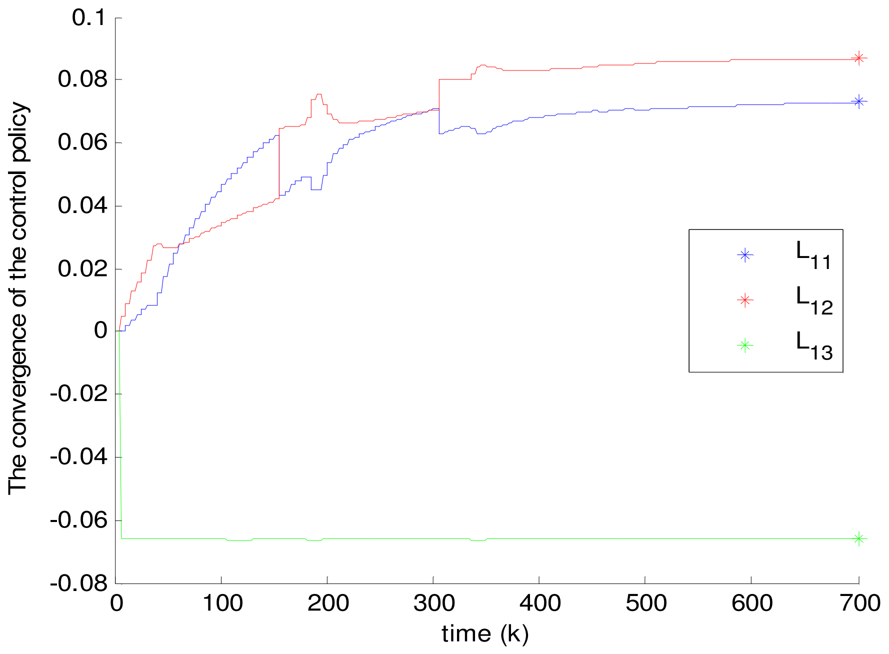


Figure. 3.13. Convergence of the control action network parameters in the DHP.

It is clear that the in the DHP algorithm, the parameters of the critic network converge to the solution of the GARE faster than the case in the HDP algorithm. This is because in DHP one has vector gradient information available for tuning, not only scalar information as in HDP. That is, in DHP the target value for the action network is a vector, while in HDP it is a scalar.

State resetting was used here to provide the excitation conditions needed to get parameter convergence in the critic and action networks. Once these parameters are known, the H-infinity controller has been found. Then, one can use the parameters of the control action network as the final parameters of the controller in any on-line control runs, without having to deliberately insert any excitation signals to the system.

3.4 Conclusion

In this chapter two on-line Approximate Dynamic Programming techniques are introduced to solve the discrete-time zero-sum game problem with continuous state and action spaces. Two of the ADP techniques, namely Heuristic Dynamic Programming, and Dual Heuristic Dynamic Programming are discussed. The derivation of the policies and the convergence of the HDP and DHP are provided. It is clear that the convergence to the optimal solution in the DHP algorithm is faster than the HDP, as gradient information, a vector, as used in DHP provides more information than scalar function information as used in HDP, therefore the number of points needed to solve the least-squares problem in the DHP is less than that in HDP. On the other hand, in the HDP algorithm the system model is needed only to tune the action networks, while in the

DHP algorithm the system model is needed to tune both the critic network and the actions networks.

The results presented herein are directly applicable in practice since they provide means to solve the H-infinity control problem, which is highly effective in feedback control systems design. A provided aircraft design example makes the point. It is interesting to see that when designing the H-infinity controller in forward time, one needs to provide an input signal that acts as a disturbance that is tuned to be the worst case disturbance in forward time.

Once the H-infinity controller is found, one can use the parameters of the control action network as the final parameters of the controller, without having to deliberately inserting any disturbance signal to the system. Disturbance is from now is determined by the nature of the process and the surrounding environment.

The results in this chapter can be summarized as a way to solve the linear quadratic discrete-time zero-sum game forward in time. The results presented here will be extended to the Q-learning case and other Action Dependent Heuristic Dynamic Programming (ADHDP) techniques in the next chapter.

CHAPTER 4

ACTION DEPENDENT HEURISTIC DYNAMIC PROGRAMMING H-INFINTY CONTROL DESIGN

In this chapter, adaptive critic approximate dynamic programming designs are derived to solve the discrete-time zero-sum game in which the state and action spaces are continuous. In which, the concept of the Q-function to the zero-sum games that continuous in the action and state spaces for linear discrete-time quadratic games is developed. This results in a forward-in-time reinforcement learning algorithm that converges to the Nash equilibrium of the corresponding zero-sum game. The results in this chapter can be thought of as a way to solve the Riccati equation of the well-known discrete-time H_∞ optimal control problem forward in time. Two designs are presented. An Action Dependent Heuristic Dynamic Programming (ADHDP) algorithm to solve for the Q-function of the associated zero-sum game is presented, and this is a model free design, *i.e.* they system dynamics is not needed, which can be thought as adaptive control design. In a second algorithm, an Action Dependent Dual Heuristic Dynamic Programming (ADDHP) is developed to solve the zero-sum game. Proofs of convergence for both forward dynamic programming schemes are presented.

4.1 Q-Function Setup for Discrete-Time Linear Quadratic Zero-sum Games

In this section, the discrete-time linear quadratic zero-sum game appearing in H_∞ optimal control problems under full state measurement information structure is considered. This problem has been solved in the literature using the dynamic programming principle and results in a backward-in-time recurrence relation for the game value function. In this chapter, the Bellman's optimality principle for the zero-sum-game is formulated using the concept of Q-functions [8][26][25] instead of the standard value functions used elsewhere. Using Q-functions will allow us in the next section to apply forward-in-time dynamic programming and the result is a model-free tuning algorithm that is relevant to Adaptive control theory.

Consider the following discrete-time linear system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k \\ y_k &= x_k, \end{aligned} \tag{4.1}$$

where $x \in R^n$, $y \in R^p$, $u_k \in R^{m_1}$ is the control input and $w_k \in R^{m_2}$ is the disturbance input. Also consider the infinite-horizon value function

$$V^*(x_k) = \min_u \max_w \sum_{i=k}^{\infty} \left[x_i^T R x_i + u_i^T u_i - \gamma^2 w_i^T w_i \right] \tag{4.2}$$

for a prescribed fixed value of γ. In the H-infinity control problem, γ is an upper bound on the desired L_2 gain disturbance attenuation [7][34].

It is desired to find the optimal control u_k^* and the worst case disturbance w_k^*, in which the infinite-horizon cost is to be minimized by player 1, u_k, and maximized by player 2, w_k. Here the class of strictly feedback stabilizing policies is considered[6].

Using the dynamic programming principle, the optimization problem in equation (4.1) and (4.2) can be written as

$$\begin{aligned} V^*(x_k) &= \min_{u_k} \max_{w_k} (r(x_k, u_k, w_k) + V^*(x_{k+1})) \\ &= \max_{w_k} \min_{u_k} (r(x_k, u_k, w_k) + V^*(x_{k+1})). \end{aligned} \tag{4.3}$$

Note that the *minimax* is equal to *maximin* since the linear system (4.1) is affine in input and the cost is quadratic. Assuming that the game has a value and is solvable, then in order to have a unique feedback saddle-point in the class of strictly feedback stabilizing policies, then the value function is quadratic in the state

$$V^*(x_k) = x_k^T P x_k \tag{4.4}$$

where $P \geq 0$ and the GARE solution and given as

$$P = A^T PA + R - [A^T PB \quad A^T PE] \begin{bmatrix} I + B^T PB & B^T PE \\ E^T PB & E^T PE - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T PA \\ E^T PA \end{bmatrix} \tag{4.5}$$

so the inequalities in (4.6) and (4.7) should be satisfied, [7],

$$I - \gamma^{-2} E^T PE > 0, \tag{4.6}$$

$$I + B^T PB > 0 \tag{4.7}$$

Q-functions have been applied to zero-sum games in the context of Markov Decision Problems [24]. In this chapter, we extend the concept of Q-functions to zero-sum games that are continuous in the state and action space as in (4.3). The optimal Q-function, Q^*, of the zero-sum game is then defined to be

$$\begin{aligned} Q^*(x_k, u_k, w_k) &= r(x_k, u_k, w_k) + V^*(x_{k+1}) \\ &= \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix} H \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix}^T, \end{aligned} \tag{4.8}$$

where H is the matrix associated with the solution of the GARE P, and is given as

$$\begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix} = r(x_k, u_k, w_k) + V^*(x_{k+1})$$

$$= x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + x_{k+1}^T P x_{k+1}$$

$$= \begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix}^T \begin{bmatrix} R & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix} + \begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix}^T \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} P \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \\ w_k \end{bmatrix}$$

where $u(x_k) = Lx_k$, and $w(x_k) = Kx_k$ so H can be written as

$$\begin{pmatrix} H_{xx} & H_{xu} & H_{xw} \\ H_{ux} & H_{uu} & H_{uw} \\ H_{wx} & H_{wu} & H_{ww} \end{pmatrix} = G + \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}^T H \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}$$

$$= G + \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} \begin{bmatrix} I & L^T & K^T \end{bmatrix} H \begin{bmatrix} I \\ L \\ K \end{bmatrix} \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix}^T \qquad (4.9)$$

$$= \begin{bmatrix} A^T PA + R & A^T PB & A^T PE \\ B^T PA & B^T PB + I & B^T PE \\ E^T PA & E^T PB & E^T PE - \gamma^2 I \end{bmatrix}$$

where

$$G = \begin{bmatrix} R & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & -\gamma^2 I \end{bmatrix},$$

and

$$P = \begin{bmatrix} I & L^T & K^T \end{bmatrix} H \begin{bmatrix} I \\ L \\ K \end{bmatrix} \qquad (4.10)$$

with L, K are the optimal strategies

The optimal Q-function $Q^*(x_k, u_k, w_k)$ is equal to the value function $V^*(x_k)$ when the policies u_k, w_k are equal to the optimal policies, this can be written as

$$\begin{aligned} V^*(x_k) &= \min_{u_k} \max_{w_k} Q^*(x_k, u_k, w_k) \\ &= \min_{u_k} \max_{w_k} \left[x_k^T \quad u_k^T \quad w_k^T \right] H \left[x_k^T \quad u_k^T \quad w_k^T \right]^T . \end{aligned} \tag{4.11}$$

Combining (4.11) with (4.8), one obtains the following recurrence relation

$$\begin{aligned} \min_{u_k} \max_{w_k} Q^*(x_k, u_k, w_k) &= \min_{u_k} \max_{w_k} \left\{ r(x_k, u_k, w_k) + \min_{u_{k+1}} \max_{w_{k+1}} Q^*(x_{k+1}, u_{k+1}, w_{k+1}) \right\} \\ &= \max_{w_k} \min_{u_k} \left\{ r(x_k, u_k, w_k) + \max_{w_{k+1}} \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}, w_{k+1}) \right\} \\ &= \max_{w_k} \min_{u_k} Q^*(x_k, u_k, w_k). \end{aligned} \tag{4.12}$$

To maximize with respect to the disturbance w_k, one needs to apply the first order necessary condition

$$\begin{aligned} 0 &= \frac{\partial Q_k^*}{\partial w_k} \\ 0 &= 2H_{wx} x_k + 2H_{wu} u_k + 2H_{ww} w_k \end{aligned} \tag{4.13}$$

Therefore, the disturbance can be written as

$$w_k = -H_{ww}^{-1}(H_{wx} x + H_{wu} u) . \tag{4.14}$$

Similarly, to minimize with respect to the control input u_k one has

$$\begin{aligned} 0 &= \frac{\partial Q_k^*}{\partial u_k} \\ 0 &= 2H_{ux} x_k + 2H_{uw} w_k + 2H_{uu} u_k \end{aligned} \tag{4.15}$$

Hence, the controller can be written as

$$u_k = -H_{uu}^{-1}(H_{ux} x_k + H_{uw} w_k) . \tag{4.16}$$

Note that applying the second order sufficiency conditions for both players one obtains

$$\begin{aligned} H_{uu} &> 0 \\ H_{ww} &< 0 \end{aligned}$$

which implies (4.6) and (4.7).

Substituting equation (4.14) in (4.15) one has

$$u_k^* = (H_{uu} - H_{uw}H_{ww}^{-1}H_{wu})^{-1}(H_{uw}H_{ww}^{-1}H_{wx} - H_{ux})x_k\,, \tag{4.17}$$

so the optimal control is a state feedback with gain

$$L = (H_{uu} - H_{uw}H_{ww}^{-1}H_{wu})^{-1}(H_{uw}H_{ww}^{-1}H_{wx} - H_{ux})\,. \tag{4.18}$$

Substituting the equation (4.16) in (4.13) one can find the optimal policy to the disturbance

$$w_k^* = (H_{ww} - H_{wu}H_{uu}^{-1}H_{uw})^{-1}(H_{wu}H_{uu}^{-1}H_{ux} - H_{wx})x_k\,, \tag{4.19}$$

so the optimal disturbance is a state feedback with gain

$$K = (H_{ww} - H_{wu}H_{uu}^{-1}H_{uw})^{-1}(H_{wu}H_{uu}^{-1}H_{ux} - H_{wx})\,. \tag{4.20}$$

Equation (4.18) and (4.20) depend only on the H matrix, and they are the main equations needed in the algorithm to be proposed to find the control and disturbance gains. The system model is not needed.

In the convergence proof, different expressions for L and K are required. One can use (4.9) to obtain the gains (4.18) and (4.20) in terms of the P matrix

$$\begin{aligned} L = {} & (I + B^T PB - B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PB)^{-1} \times \\ & (B^T PE(E^T PE - \gamma^2 I)^{-1} E^T PA - B^T PA) \end{aligned} \tag{4.21}$$

$$\begin{aligned} K = {} & (E^T PE - \gamma^2 I - E^T PB(I + B^T PB)^{-1} B^T PE)^{-1} \times \\ & (E^T PB(I + B^T PB)^{-1} B^T PA - E^T PA). \end{aligned} \tag{4.22}$$

Note that the inverse matrices in (4.21) and (4.22) exist due to (4.6) and (4.7). The policies (4.21) and (4.22) can be derived directly from (4.3) and requires the knowledge of the system model matrices, A, B and E unlike the policies derived in (4.18) and (4.20) which depends on H only. Hence, as will be seen in the next section, this will allow the development of a model-free online tuning algorithm.

Now, it will be shown how to develop the ADHDP and ADDHP algorithm using these constructions.

4.2 Action dependent Heuristic Dynamic programming (ADHDP)

In this section, an Q-Learning algorithm ADHDP to solve the discrete-time linear quadratic zero-sum game described in chapter 2 is developed. The Q-Learning algorithm was originally proposed in [8][25] to solve optimal control problems. The Q-Learning algorithm has been applied earlier to solve the discrete-time Linear Quadratic Regulator (LQR) in optimal control theory [31]. In the Q-Learning approach, a parametric structure is used to approximate the Q-function of the current control policy. Then the certainty equivalence principle is used to improve the policy of the action network. It can be thought of as a Q-learning algorithm in continuous state and action spaces.

In this section, the Q-Learning approach is extended to discrete-time linear quadratic zero-sum games appearing in [7], an the convergence proof of the presented algorithm is provided. This can be thought of as a Q-learning for zero-sum games that have continuous state and action spaces.

4.2.1 Derivation of the ADHDP for zero-sum games

In the Q-Learning, one starts with an initial Q-function $Q_0(x,u,w) \geq 0$ that is not necessarily optimal, and then finds $Q_1(x,u,w)$ by solving equation (4.23) with $i=0$ as

$$\begin{aligned} &Q_{i+1}(x_k,u_k,w_k) = \\ &\qquad \left\{ x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + \min_{u_{k+1}} \max_{w_{k+1}} Q_i(x_{k+1},u_{k+1},w_{k+1}) \right\}, \\ &\qquad = \left\{ x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(x_{k+1}) \right\} \\ &\qquad = \left\{ x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(Ax_k + Bu_k + Ew_k) \right\} \end{aligned} \tag{4.23}$$

then applying the following incremental optimization on the Q function as

$$\min_{u_k} \max_{w_k} Q_{i+1}(x_k,u_k,w_k) = \min_{u_k} \max_{w_k} \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix} H_{i+1} \begin{bmatrix} x_k^T & u_k^T & w_k^T \end{bmatrix}^T$$

According to(4.18) and (4.20) the corresponding state feedback policy updates are given by

$$\begin{aligned} L_i &= (H_{uu}^i - H_{uw}^i H_{ww}^{i\;-1} H_{wu}^i)^{-1} (H_{uw}^i H_{ww}^{i\;-1} H_{wx}^i - H_{ux}^i), \\ K_i &= (H_{ww}^i - H_{wu}^i H_{uu}^{i\;-1} H_{uw}^i)^{-1} (H_{wu}^i H_{uu}^{i\;-1} H_{ux}^i - H_{wx}^i). \end{aligned} \tag{4.24}$$

with

$$\begin{aligned} u_i(x_k) &= L_i x_k \\ w_i(x_k) &= K_i x_k \end{aligned} \tag{4.25}$$

Note that since $Q_i(x,u,w)$ is not initially optimal, the improved policies $u_i(x_k)$ and $w_i(x_k)$ use the certainty equivalence principle. Note that to update the action networks, the plant model A , B and E matrices are not needed.

This is a greedy policy iteration method that is based on the Q-function. In chapter 3, a greedy policy updates on V is shown and this can now be recovered from (4.23) as

$$\begin{aligned}\min_{u_k}\max_{w_k} Q_{i+1}(x_k,u_k,w_k) &= V_{i+1}(x_k)\\ &= \min_{u_k}\max_{w_k}\left\{x_k^T R x_k + u_k^T u_k - \gamma^2 w_k^T w_k + V_i(Ax_k + Bu_k + Ew_k)\right\}.\end{aligned}$$

Note that in equation (4.23), the Q-function is given for any policy u and w. To develop solutions to (4.23) forward in time, one can substitute (4.25) in (4.23) to obtain the following recurrence relation on

$$\begin{aligned}Q_{i+1}(x_k,u_i(x_k),w_i(x_k)) = x_k^T R x_k + u_i^T(x_k)u_i(x_k) - \gamma^2 w_i^T(x_k)w_i(x_k) +\\ \begin{bmatrix} x_{k+1}^T & u_i^T(x_{k+1}) & w_i^T(x_{k+1})\end{bmatrix} H_i \begin{bmatrix} x_{k+1}^T & u_i^T(x_{k+1}) & u_i^T(x_{k+1})\end{bmatrix}\end{aligned} \tag{4.26}$$

that is used to solve for the optimal Q-function forward in time.

The idea to solve for Q_{i+1}, then once determined, one repeats the same process for $i = 0,1,2,\ldots$. In this chapter, it is shown that $Q_{i+1}(x_k,u_i(x_k),w_i(x_k) \to Q^*(x_k,u_{k,}w_k)$ as $i \to \infty$, which means $H_i \to H$, $L_i \to L$ and $K_i \to L$.

In the ADHDP approach, the Q-function is generally difficult to obtain in closed-form except in special cases like the linear system (4.1) . Therefore, in general, a parametric structure is used to approximate the actual $Q_i(x,u,w)$. Similarly, parametric structures are used to obtain approximate closed-form representations of the two action networks $\hat{u}(x,L)$ and $\hat{w}(x,K)$. Since in this chapter linear quadratic zero-sum games are considered, the Q-function is quadratic in the state and the policies, *i.e.* (4.8). Moreover, the two action networks are linear in the state, *i.e.* (4.17) and (4.19).

A natural choice of these parameter structures is given as

$$\hat{u}_i(x) = L_i x, \tag{4.27}$$

$$\hat{w}_i(x) = K_i x, \tag{4.28}$$

$$\begin{aligned}\hat{Q}(\bar{z},h_i) &= z^T H_i z \\ &= h_i^T \bar{z}\end{aligned}, \tag{4.29}$$

where $z = \begin{bmatrix} x^T & u^T & w^T \end{bmatrix}^T$ $z \in R^{n+m_1+m_2=q}$, $\bar{z} = (z_1^2, \ldots, z_1 z_q, z_2^2, z_2 z_3, \ldots, z_{q-1} z_q, z_q^2)$ is the Kronecker product quadratic polynomial basis vector [21], and $h = v(H)$ with $v(\cdot)$ a vector function that acts on $q \times q$ matrices and gives a ${}^{q(q+1)}\!/_2 \times 1$ column vector. The output of $v(\cdot)$ is constructed by stacking the columns of the squared matrix into a one-column vector with the off-diagonal elements summed as $H_{ij} + H_{ji}$, [21]. In the linear case, the parametric structures (4.27) (4.28) and (4.29) give an exact closed-form representation of the functions in (4.26). Note that (4.27) and (4.28) are updated using (4.23).

To solve for Q_{i+1} in (4.26), the right hand side of (4.26) is written as

$$\begin{aligned} d(z_k(x_k), H_i) = x_k^T R x_k + \hat{u}_i(x_k)^T \hat{u}_i(x_k) - \gamma^2 \hat{w}_i(x_k)^T \hat{w}_i(x_k) + \\ Q_i(x_{k+1}, \hat{u}_i(x_{k+1}), \hat{w}_i(x_{k+1})) \end{aligned} \tag{4.30}$$

which can be thought of as the desired target function to which one needs to fit $\hat{Q}(z, h_{i+1})$ in least-squares sense to find h_{i+1} such that

$$h_{i+1}^T \bar{z}(x_k) = d(\bar{z}(x_k), h_i)\,. \tag{4.31}$$

The parameter vector h_{i+1} is found by minimizing the error between the target value function (4.30) and (4.29) in a least-squares sense over a compact set Ω,

$$h_{i+1} = \arg\min_{h_{i+1}} \{ \int_{\Omega} | h_{i+1}^T \bar{z}(x_k) - d(\bar{z}(x_k), h_i) |^2 \, dx_k \}\,. \tag{4.32}$$

Solving the least-squares problem one obtains

$$h_{i+1} = \left(\int_{\Omega} \overline{z}(x_k)\overline{z}(x_k)^T dz \right)^{-1} \int_{\Omega} \overline{z}(x_k) d(\overline{z}(x_k), h_i) dx \tag{4.33}$$

Note however that $z(x_k)$ is

$$\begin{aligned} z(x_k) &= \begin{bmatrix} x_k^T & (\hat{u}_i(x_k))^T & (\hat{w}_i(x_k))^T \end{bmatrix}^T \\ &= \begin{bmatrix} x_k^T & (L_i x_k)^T & (K_i x_k)^T \end{bmatrix}^T \quad , \\ &= \left(x_k^T \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix}^T \right)^T \end{aligned} \tag{4.34}$$

from (4.34) one can note that $\hat{u}_i$ and $\hat{w}_i$ are linearly dependent on x_k, see (4.27) and (4.28), therefore

$$\int_{\Omega} \overline{z}(x_k)\overline{z}(x_k)^T dx_k$$

is never invertible, which means that the least-squares problem (4.32), (4.33) will never be solvable. To overcome this problem one, exploration noise is added to both inputs in (4.25) to obtain

$$\begin{aligned} \hat{u}_{ei}(x_k) &= L_i x_k + n_{1k} \\ \hat{w}_{ei}(x_k) &= K_i x_k + n_{2k} \end{aligned} , \tag{4.35}$$

where $n_1(0,\sigma_1)$ and $n_2(0,\sigma_2)$ are zero-mean exploration noise with variances σ_1^2 and σ_2^2 respectively, therefore $z(x_k)$ in (4.34) becomes

$$z(x_k) = \begin{bmatrix} x_k \\ \hat{u}_{ei}(x_k) \\ \hat{w}_{ei}(x_k) \end{bmatrix} = \begin{bmatrix} x_k \\ L_i x_k + n_{1k} \\ K_i x_k + n_{2k} \end{bmatrix} = \begin{bmatrix} x_k \\ L_i x_k \\ K_i x_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ n_{1k} \\ n_{2k} \end{bmatrix}.$$

Evaluating (4.31) at several points $p1, p2, p3, \ldots \in \Omega$, one has

$$h_{i+1} = (ZZ^T)^{-1} ZY \tag{4.36}$$

with

$$Z = [\bar{z}(p1) \quad \bar{z}(p2) \quad \cdots \quad \bar{z}(pN)]$$
$$Y = [d(z(p1),h_i) \quad d(z(p2),h_i) \quad \cdots \quad d(z(pN),h_i)]^T .$$

It is not enough to add the noise to the control and disturbance inputs, In order the algorithm to converge to optimal solution, the target in equation (4.30) is modified to become

$$\begin{aligned} d(z_k(x_k),H_i) = x_k^T R x_k + \hat{u}_{ei}(x_k)^T \hat{u}_{ei}(x_k) - \gamma^2 \hat{w}_{ei}(x_k)^T \hat{w}_{ei}(x_k) + \\ Q_i(x_{k+1}, \hat{u}_i(x_{k+1}), \hat{w}_i(x_{k+1}) \end{aligned} \tag{4.37}$$

with $\hat{u}_i$ and $\hat{w}_i$ used for Q_i instead of $\hat{u}_{ei}$ and $\hat{w}_{ei}$. The invertiblity of the matrix in (4.36) is therefore guaranteed by the excitation condition. This can be written as

$$\begin{aligned} d(z_k(x_k),H_i) = x_k^T R x_k + \hat{u}_{ei}(x_k)^T \hat{u}_{ei}(x_k) - \gamma^2 \hat{w}_{ei}(x_k)^T \hat{w}_{ei}(x_k) + \\ \left[x_{k+1}^T \quad (L_i x_{k+1})^T \quad (K_i x_{k+1})^T \right] H_i \left[x_{k+1}^T \quad (L_i x_{k+1})^T \quad (K_i x_{k+1})^T \right]^T \end{aligned}$$

where

$$x_{k+1} = Ax_k + B\hat{u}_{ei}(x_k) + E\hat{w}_{ei}$$

4.2.2 Online implementation of the ADHDP Algorithm

The least-squares problem in (4.36) can be solved in real-time by collecting enough data points generated from $d(z_k, h_i)$ in (4.37). This requires one to have knowledge of the state information x_k, x_{k+1} as the dynamics evolve in time, and also of the reward function $r(z_k) = x_k^T R x_k + \hat{u}_{ei}(x_k)^T \hat{u}_{ei}(x_k) - \gamma^2 \hat{w}_{ei}(x_k)^T \hat{w}_{ei}(x_k)$ and Q_i. This can be determined by simulation, or in real-time applications, by observing the states on-line. Therefore, in the Q-Learning algorithm, the model of the system is not needed to update the critic network and the action network. This results in a model-free tuning algorithm suitable for adaptive control application.

To satisfy the excitation condition of the least-squares problem, one needs to have the number of collected points N at least $N \geq q(q+1)/2$,where $q = n + m_1 + m_2$ is the number of states and both policies, control and disturbance. In online implementation of the least-squares problem, Y and Z matrices are obtained in real-time as

$$\begin{aligned} Z &= \left[\bar{z}(x_{k-N-1}) \quad \bar{z}(x_{k-N-2}) \quad \cdots \quad \bar{z}(x_{k-1}) \right] \\ Y &= \left[d(\bar{z}(x_{k-N-1}), h_i) \quad d(\bar{z}(x_{k-2}), h_i) \quad \cdots \quad d(\bar{z}(x_{k-1}), h_i) \right]^T . \end{aligned} \tag{4.38}$$

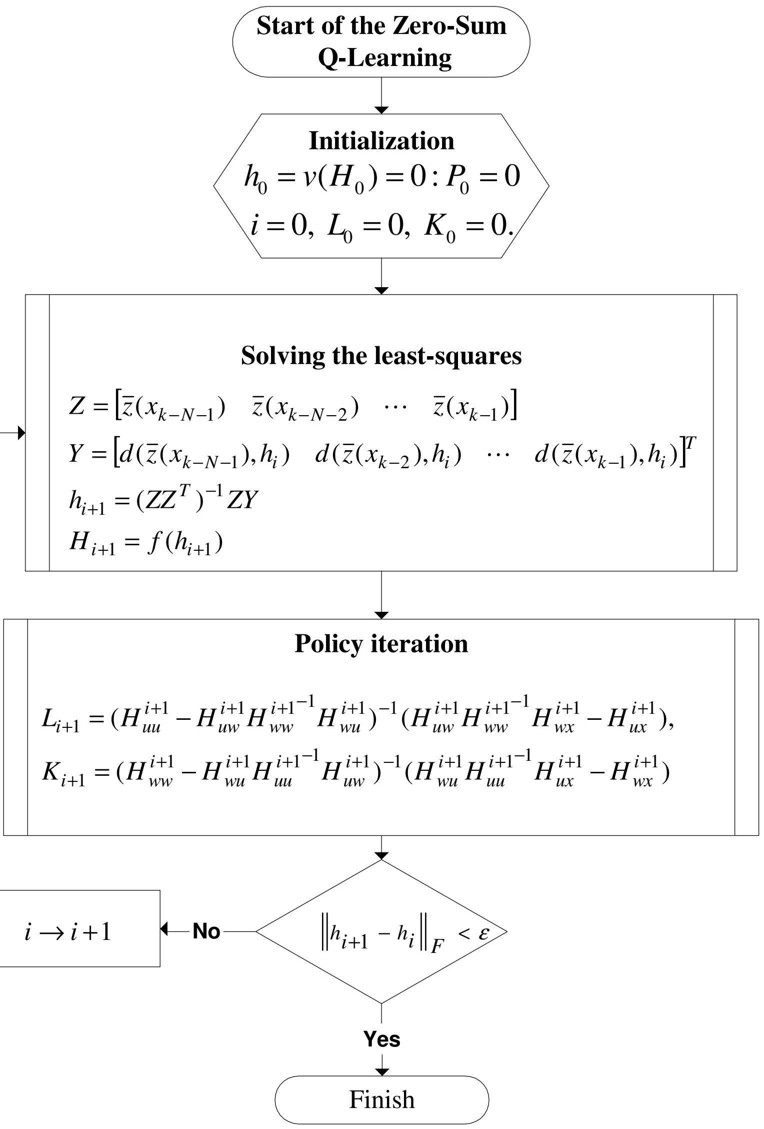
One can also solve (4.38) recursively using the well-known recursive least-squares technique. In that case, the excitation condition is replaced by the persistency of excitation condition

$$\varepsilon_0 I \leq \frac{1}{\alpha} \sum_{m=1}^{\alpha} \bar{z}_{k-t} \bar{z}_{k-t}^T \leq \varepsilon_1 I$$

for all $k > \alpha_0$, $\alpha > \alpha_0$, with $\varepsilon_0 \leq \varepsilon_1$, ε_0 and ε_1 positive integers and $\varepsilon_0 \leq \varepsilon_1$. The recursive least-squares algorithm is given as

$$\begin{aligned} e_i(t) &= d(z_k, h_i) - \bar{z}_k^T h_{i+1}(t-1) \\ h_{i+1}(t) &= h_{i+1}(t-1) + \frac{\Gamma_i(t-1)\bar{z}_k e_i(t)}{1 + \bar{z}_k^T \Gamma_i(t-1)\bar{z}_k} \\ \Gamma_i(t) &= \Gamma_i(t-1) - \frac{\Gamma_i(t-1)\bar{z}_k \bar{z}_k^T \Gamma_i(t-1)}{1 + \bar{z}_k^T \Gamma_i(t-1)\bar{z}_k} \end{aligned}$$

where i is the policy update index, t is the index of the recursions of the recursive least-squares, and k is the discrete time. Γ is the covariance matrix of the recursion and $e(t)$ is the estimation error of the recursive least-squares. Note that $\Gamma_i(0)$ is a large number and $\Gamma_{i+1}(0) = \Gamma_i$. The on-line Q-Learning algorithm (ADHDP) developed in this chapter is summarized in the flowchart shown in Figure 4.1.



Start of the Zero-Sum Q-Learning
Initialization
$h_0 = v(H_0) = 0 : P_0 = 0$
$i = 0,\ L_0 = 0,\ K_0 = 0.$
Solving the least-squares
$Z = \left[\bar{z}(x_{k-N-1}) \quad \bar{z}(x_{k-N-2}) \quad \cdots \quad \bar{z}(x_{k-1})\right]$
$Y = \left[d(\bar{z}(x_{k-N-1}), h_i) \quad d(\bar{z}(x_{k-2}), h_i) \quad \cdots \quad d(\bar{z}(x_{k-1}), h_i)\right]^T$
$h_{i+1} = (ZZ^T)^{-1} ZY$
$H_{i+1} = f(h_{i+1})$
Policy iteration
$L_{i+1} = (H_{uu}^{i+1} - H_{uw}^{i+1} H_{ww}^{i+1^{-1}} H_{wu}^{i+1})^{-1} (H_{uw}^{i+1} H_{ww}^{i+1^{-1}} H_{wx}^{i+1} - H_{ux}^{i+1}),$
$K_{i+1} = (H_{ww}^{i+1} - H_{wu}^{i+1} H_{uu}^{i+1^{-1}} H_{uw}^{i+1})^{-1} (H_{wu}^{i+1} H_{uu}^{i+1^{-1}} H_{ux}^{i+1} - H_{wx}^{i+1})$
$\left\| h_{i+1} - h_i \right\|_F < \varepsilon$
No
$i \to i+1$
Yes
Finish

Figure 4.1. The ADHDP algorithm.

The ADHDP algorithm for zero-sum games follows by iterating between (4.23) and (4.38). In the remaining of this section, it will be shown that this policy iteration technique will cause Q_i to converge to the optimal Q^*. Note that the model of the system is not required to update the actions networks and the critic network.

4.2.3 Convergence of the ADHDP Algorithm

The proof of convergence for the proposed Q-Learning algorithm, *i.e.* ADHDP algorithm, for zero-sum games converges to the optimal policies is provided.

Lemma 4.1 Iterating on equations (4.23), and (4.38) is equivalent to

$$H_{i+1} = G + \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}. \tag{4.39}$$

Proof: Since equation (4.37) is equivalent to

$$d(\overline{z}_k(x_k), h_i) = \overline{z}_k^T \times v\left(G + \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix} \right),$$

then using the Kronecker products [21], the least-squares (4.38) becomes

$$h_{i+1} = \underbrace{(ZZ^T)^{-1}(ZZ)}_{I} \times v\left(G + \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix} \right).$$

where v is the vectorized function in Kronecker products.

Since the matrix H_{i+1} reconstructed from h_{i+1} is symmetric, iterating on h_i is equivalent to

$$H_{i+1} = G + \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix} \qquad \square$$

Lemma 4.2 The matrices H_{i+1}, L_{i+1} and K_{i+1} can be written as

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}. \tag{4.40}$$

$$\begin{aligned} L_{i+1} &= (I + B^T P_i B - B^T P_i E (E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times \\ &\quad (B^T P_i E (E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A), \end{aligned} \tag{4.41}$$

$$\begin{aligned} K_{i+1} &= (E^T P_i E - \gamma^2 I - E^T P_i B (I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times \\ &\quad (E^T P_i B (I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A). \end{aligned} \tag{4.42}$$

where P_i is given as

$$P_i = \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix}. \tag{4.43}$$

Proof: Equation (4.39) can be written as

$$\begin{aligned} H_{i+1} &= G + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix} \\ &= G + \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix} \begin{bmatrix} A & B & E \end{bmatrix}. \end{aligned}$$

Since

$$P_i = \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix},$$

then it follows that

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}.$$

Using equations (4.24) and (4.40), one obtains (4.41) and (4.42). □

Lemma 4.3: Iterating on H_i is similar to iterating on P_i as

$$P_{i+1} = A^T P_i A + R - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix} \tag{4.44}$$

with P_i defined as in (4.43).

Proof: From (4.43), one has

$$P_{i+1} = \begin{bmatrix} I & L_{i+1}^T & K_{i+1}^T \end{bmatrix} H_{i+1} \begin{bmatrix} I \\ L_{i+1} \\ K_{i+1} \end{bmatrix},$$

and using (4.40), one obtains

$$\begin{aligned} P_{i+1} &= \begin{bmatrix} I & L_{i+1}^T & K_{i+1}^T \end{bmatrix} \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix} \begin{bmatrix} I \\ L_{i+1} \\ K_{i+1} \end{bmatrix} \\ &= R + L_{i+1}^T L_{i+1} - \gamma^2 K_{i+1}^T K_{i+1} + (A^T + L_{i+1}^T B^T + K_{i+1}^T E^T) P_i (A + BL_{i+1} + EK_{i+1}) \end{aligned} \tag{4.45}$$

Using (4.41), and (4.42), one has

$$P_{i+1} = A^T P_i A + R - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}. \qquad □$$

Theorem 4.1: Assume that the linear quadratic zero-sum game is solvable and has a value under the state feedback information structure. Then, iterating on equation (4.39) in Lemma 4.1, with $H_0 = 0$, $L_0 = 0$ and $K_0 = 0$ converges with $H_i \to H$, where H is corresponds to $Q^*(x_k, u_k w_k)$ and

$$xPx = \min_u \max_w Q^*(x,u,w) = \max_w \min_u Q^*(x,u,w)$$

with P solving the GARE (4.5).

Proof: In [3] it is shown that iterating on the algebraic Riccati equation (4.44) with $P_0 = 0$ converges to P that solves (4.5). Since Lemma 4.3 shows that iterating on H_i matrix is equivalent to iterating on P_i, then as $i \to \infty$

$$H_i \to \begin{bmatrix} A^TPA+R & A^TPB & A^TPE \\ B^TPA & B^TPB+I & B^TPE \\ E^TPA & E^TPB & E^TPE-\gamma^2 I \end{bmatrix}.$$

hence from (4.9), and since from (4.43) $H_0 = 0$, $L_0 = 0$ and $K_0 = 0$ implies that $P_0 = 0$, one concludes that $Q_i \to Q^*$. □

The proof of convergence of the ADHDP has been established assuming the least-squares problem (4.38) is solved completely; i.e. the excitation condition is satisfied. Note that this implies that ADHDP can be interpreted as solving the algebraic Riccati equation of the zero-sum game without requiring the plant model.

4.3 Action Dependent Dual Heuristic Dynamic Programming (ADDHP)

In this section, an ADDHP algorithm is developed to solve the discrete-time linear system zero-sum game described in chapter 2. The ADDHP algorithm has been applied earlier to solve the discrete-time Linear Quadratic Regulator (LQR) in optimal control theory [31]. In the ADDHP approach, a parametric structure is used to approximate the co-state function, *i.e.* the gradient of the cost-to-go function, of the current control policies. As in the ADHDP case, the certainty equivalence principle is used to improve the policies of the actions networks.

In this section, we extend the ADDHP approach to linear quadratic discrete-time zero-sum games appearing in [7], and prove the convergence of the presented algorithm.

4.3.1 Derivation of the ADDHP algorithm

Consider the system (4.1) and the cost-to-go function (4.2). In the ADDHP approach, the critic network approximates the Q co-state $\lambda\ (z_k)$ forward in time. In the zero-sum game ADDHP algorithm developed in this chapter, the following recurrence relation is derived to solve for the Q co-state forward in time.

$$\begin{aligned}
\lambda_{i+1}(z_k) &= \frac{\partial Q_{i+1}(z_k)}{\partial z_k} \\
&= \begin{bmatrix} \left(\dfrac{\partial Q_{i+1}(z_k)}{\partial x_k}\right)^T & \left(\dfrac{\partial Q_{i+1}(z_k)}{\partial u_k}\right)^T & \left(\dfrac{\partial Q_{i+1}(z_k)}{\partial w_k}\right)^T \end{bmatrix}^T \\
&= \begin{bmatrix} \dfrac{\partial r(z_k)}{\partial x_k} + \left(\dfrac{\partial Q_i(z_{k+1})}{\partial z_{k+1}}\right)^T \left(\dfrac{\partial z_{k+1}}{\partial x_k}\right) \\ \dfrac{\partial r(z_k)}{\partial u_k} + \left(\dfrac{\partial Q_i(z_{k+1})}{\partial z_{k+1}}\right)^T \left(\dfrac{\partial z_{k+1}}{\partial u_k}\right) \\ \dfrac{\partial r(z_k)}{\partial u_k} + \left(\dfrac{\partial Q_i(z_{k+1})}{\partial z_{k+1}}\right)^T \left(\dfrac{\partial z_{k+1}}{\partial w_k}\right) \end{bmatrix}
\end{aligned} \tag{4.46}$$

The recurrence relation (4.46) is obtained by differentiating the recurrence relation on the Q- function, which given as follows

$$\begin{aligned}
Q_{i+1}(x_k, u_i(x_k), w_i(x_k)) = x_k^T R x_k + u_i^T(x_k) u_i(x_k) - \gamma^2 w_i^T(x_k) w_i(x_k) + \\
\begin{bmatrix} x_{k+1}^T & u_i^T(x_{k+1}) & w_i^T(x_{k+1}) \end{bmatrix} H_i \begin{bmatrix} x_{k+1}^T & u_i^T(x_{k+1}) & u_i^T(x_{k+1}) \end{bmatrix}
\end{aligned}$$

Let $z = \begin{bmatrix} x^T & u^T & w^T \end{bmatrix}^T$, Equation (4.46) can be rewritten as

$$\lambda_{i+1}(z_k) = \begin{bmatrix} \dfrac{\partial r(z_k)}{\partial x_k} + \lambda_i^T(z_{k+1})\left(\dfrac{\partial z_{k+1}}{\partial x_k}\right) \\ \dfrac{\partial r(z_k)}{\partial u_k} + \lambda_i^T(z_{k+1})\left(\dfrac{\partial z_{k+1}}{\partial u_k}\right) \\ \dfrac{\partial r(z_k)}{\partial w_k} + \lambda_i^T(z_{k+1})\left(\dfrac{\partial z_{k+1}}{\partial w_k}\right) \end{bmatrix}. \tag{4.47}$$

The improvement of the actions networks requires that

$$\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial u_i(x_k)} + \lambda_i^T(z_{k+1})\left(\frac{\partial z_{k+1}}{\partial u_k}\right) = 0, \tag{4.48}$$

$$\frac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial w_i(x_k)} + \lambda_i^T(z_{k+1})\left(\frac{\partial z_{k+1}}{\partial w_k}\right) = 0. \tag{4.49}$$

Combining (4.47), (4.48) and (4.49), one has

$$\lambda_{i+1}(z_k) = \begin{pmatrix} \dfrac{\partial r(x_k, u_i(x_k), w_i(x_k))}{\partial x_k} + \lambda_i^T(z_{k+1})\left(\dfrac{\partial z_{k+1}}{\partial x_k}\right) \\ 0 \\ 0 \\ \end{pmatrix}. \tag{4.50}$$

Hence, the ADDHP algorithm can be summarized as the successive iteration between (4.47) on one hand, and (4.48), (4.49) on the other. This results in a successive-improvement of the Q function derivatives sequence $\{\lambda_i \mid i = 1, 2, \ldots\}$ as the player policies are generated. In this chapter, it will be shown that the ADDHP algorithm converges. When converged, equation (4.50) becomes

$$\lambda^*(z_k^*) = \begin{bmatrix} \dfrac{\partial r(x_k^*, u_i(x_k^*), w_i(x_k^*))}{\partial x_k^*} + \lambda^{T*}(x_{k+1})\left(\dfrac{\partial x_{k+1}}{\partial x_k^*}\right) \\ 0 \\ 0 \\ \end{bmatrix},$$

the top element is known as the co-state equation in Theorem 6.3 in [6].

In ADDHP, a parametric structure $\hat{\lambda}(z, h_i)$ is used to approximate the actual $\lambda_i(z)$. Similarly, parametric structures are used to obtain approximate closed-form representations of the two action networks $\hat{u}(x, L)$ and $\hat{w}(x, K)$. Since in this chapter the zero-sum game considered is linear and quadratic, it is well-known that Q co-state and the action networks are all linear in the state. Therefore a natural choice of these parameter structures is given as

$$\hat{\lambda}(z, h_i) = h_i^T \frac{\partial \overline{z}}{\partial z}, \tag{4.51}$$

$$\hat{u}_i(x) = L_i x, \tag{4.52}$$

$$\hat{w}_i(x) = K_i x, \tag{4.53}$$

where $z = \begin{bmatrix} x^T & u^T & w^T \end{bmatrix}^T$ $z \in R^{n+m_1+m_2=q}$, $\overline{z} = (z_1^2, \ldots, z_1 z_q, z_2^2, z_2 z_3, \ldots, z_{q-1} z_q, z_q^2)$ is the Kronecker product quadratic polynomial basis vector [21], and $h = v(H)$ with $v(\cdot)$ a vector function that acts on $q \times q$ matrices and gives a ${}^{q(q+1)}\!/_2 \times 1$ column vector. The output of $v(\cdot)$ is constructed by stacking the columns of the squared matrix into a one-column vector with the off-diagonal elements summed as $H_{ij} + H_{ji}$, [21].. The parametric structures (4.51) (4.52) and (4.53) give an exact closed-form representation of the functions in (4.47) to (4.50).

Using the parameter structures (4.52) and (4.52) along with the certainty equivalence principle, it can be easily shown form (4.18) and (4.20) that the parameters of the actions networks are updates as.

$$\begin{aligned} L_i &= (H^i_{uu} - H^i_{uw} H^{i\ -1}_{ww} H^i_{wu})^{-1} (H^i_{uw} H^{i\ -1}_{ww} H^i_{wx} - H^i_{ux}), \\ K_i &= (H^i_{ww} - H^i_{wu} H^{i\ -1}_{uu} H^i_{uw})^{-1} (H^i_{wu} H^{i\ -1}_{uu} H^i_{ux} - H^i_{wx}). \end{aligned} \tag{4.54}$$

Substituting (4.54) in (4.51) given the system model (4.1) , one has

$$d(z_k, P_i) = 2Gz_k + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i z_{k+1}. \tag{4.55}$$

Using the Kroncker product notation [21], equation (4.34) can be rewritten as

$$d(z_k, h_i) = v(G + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix})^T \frac{\partial \overline{z}}{\partial z}. \tag{4.56}$$

Equation (4.56) can be thought of as the desired target function to which one needs to fit $\hat{\lambda}(z, h_{i+1})$ such that $\hat{\lambda}(z, h_{i+1}) = d(z_k, h_i)$.

The parameter vector h_{i+1} is found by minimizing the error between the target value function (4.56) and (4.51) in a least-squares sense over a compact set, Ω,

$$h_{i+1} = \arg\min_{h_{i+1}} \int_{\Omega} \{| h^T_{i+1} \frac{\partial \overline{z}^T(x_k)}{\partial z} - d(z_k(x_k, h_i) |^2\} dx. \tag{4.57}$$

This least square problem can be solved as

$$h_{i+1} = \left(\int_{\Omega} \frac{\partial \overline{z}}{\partial z} \frac{\partial \overline{z}^T}{\partial z} dx \right)^{-1} \times$$
$$\int_{\Omega} \frac{\partial \overline{z}}{\partial z} \frac{\partial \overline{z}^T}{\partial z} v(G + \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}))dx, \quad , \tag{4.58}$$

note that z is

$$\begin{aligned} z &= \left[x_k^T \quad (\hat{u}_i(x_k))^T \quad (\hat{w}_i(x_k))^T \right]^T \\ &= \left[x_k^T \quad (L_i x_k)^T \quad (K_i x_k)^T \right]^T \quad , \\ &= \left(x_k^T \left[I \quad L_i^T \quad K_i^T \right]^T \right)^T \end{aligned} \tag{4.59}$$

from (4.59) one can note that $\hat{u}_i$ and $\hat{w}_i$ are linearly dependent on x_k, so $\overline{zz}^T$ will never be invertible, which means that the least-square problem (4.57), (4.58) will never be solvable.

To solve this problem one can we redefine z as

$$\begin{aligned} z &= \left[x_k^T \quad (\hat{u}_{ei}(x_k))^T \quad (\hat{w}_{ei}(x_k))^T \right] \\ &= \left[x_k^T \quad (L_i x_k + n_{1k})^T \quad (K_i x_k + n_{2k})^T \right]^T, \end{aligned}$$

where

$$\begin{aligned} \hat{u}_{ei}(x_k) &= L_i x_k + n_{1k} \\ \hat{w}_{ei}(x_k) &= K_i x_k + n_{2k} \end{aligned}, \tag{4.60}$$

Evaluation (4.58) at several points $p1, p2, p3, \ldots \in \Omega$, one has

$$h_{i+1} = (ZZ^T)^{-1} ZY \tag{4.61}$$

with

$$Z = [\frac{\partial \overline{z}(p1)}{\partial z} \quad \frac{\partial \overline{z}(p2)}{\partial z} \quad \cdots \quad \frac{\partial \overline{z}(pN)}{\partial z}]$$
$$Y = [d(z(p1), h_i) \quad d(z(p2), h_i) \quad \cdots \quad d(z(pN), h_i)]^T.$$

The invertiblity of the matrix in (4.60) is therefore guaranteed by the excitation condition

4.3.1 Online Implementation of the ADDHP algorithm

The least-squares problem in (4.57) can be solved in real-time by collecting enough data points generated from $d(z_k, h_i)$ in (4.56). This requires having access to the state and the policies information z_k, z_{k+1} as the dynamics evolve in time and gradients of the reward function $\partial r/\partial x_k$, $\partial r/\partial u_i$, $\partial r/\partial w_i$, as well as the plant model A and B. Therefore, in the ADDHP algorithm developed in this chapter for the zero-sum game, the plant model is required to update the critic network.

To satisfy the excitation condition of the least squares problem, one needs to have the number of collected points N at least $N \geq q$. Therefore, after several time steps that are enough to guarantee the excitation condition, one has the following least-squares problem

$$h_{i+1} = (ZZ^T)^{-1} ZY, \qquad (4.62)$$

where

$$Z = [\left.\frac{\partial \overline{z}}{\partial z}\right|_{z_{k-N-1}} \quad \left.\frac{\partial \overline{z}}{\partial z}\right|_{z_{k-N-2}} \quad \cdots \quad \left.\frac{\partial \overline{z}}{\partial z}\right|_{z_{\mathrm{k}\text{-}1}} \quad]$$
$$Y = [d^T(z_{k-N-1}, h_i) \quad d^T(z_{k-N-2}, h_i) \quad \cdots \quad d^T(z_{\mathrm{k}\text{-}1}, h_i)]^T.$$

One can solve (4.62) recursively using the well-known recursive least squares technique. In that case, the excitation condition is replaced with the persistency of excitation condition

$$\varepsilon_0 I \le \frac{1}{\alpha}\sum_{m=1}^{\alpha}\frac{\partial \overline{z}}{\partial \overline{x}}\Big|_{k-t}\frac{\partial \overline{z}}{\partial z}^T\Big|_{k-t} \le \varepsilon_1 I$$

for all $k > \alpha_0$, $\alpha > \alpha_0$, with $\varepsilon_0 \le \varepsilon_1$, ε_0 and ε_1 are positive integers and $\varepsilon_0 \le \varepsilon_1$.

The recursive least-squares algorithm is given as

$$e_i(t) = d_x(x_k, p_i) - \frac{\partial \overline{z}_k^T}{\partial z} p_{i+1}(t-1)$$

$$p_{i+1}(t) = p_{i+1}(t-1) + \Gamma_i(t-1)\frac{\partial \overline{z}_k}{\partial z}\left(I + \frac{\partial \overline{x}_k^T}{\partial x}\Gamma_i(t-1)\frac{\partial \overline{x}_k}{\partial x}\right)^{-1} e_i(t)$$

$$\Gamma_i(t) = \Gamma_i(t-1) - \Gamma_i(t-1)\frac{\partial \overline{z}_k}{\partial z}\left(I + \frac{\partial \overline{z}_k^T}{\partial z}\Gamma_i(t-1)\frac{\partial \overline{z}_k}{\partial z}\right)^{-1}\frac{\partial \overline{z}_k^T}{\partial z}\Gamma_i(t-1)$$

where i is the policy update index, t is the index of the recursions of the recursive least-squares, and k is the discrete time. Γ is the covariance matrix of the recursion and $e(t)$ is the estimation error of the recursive least-squares. Note that $\Gamma_i(0)$ is a large number, and $\Gamma_{i+1}(0) = \Gamma_i$.

The developed ADDHP algorithm is summarized in the flowchart shown in Figure 4.2.

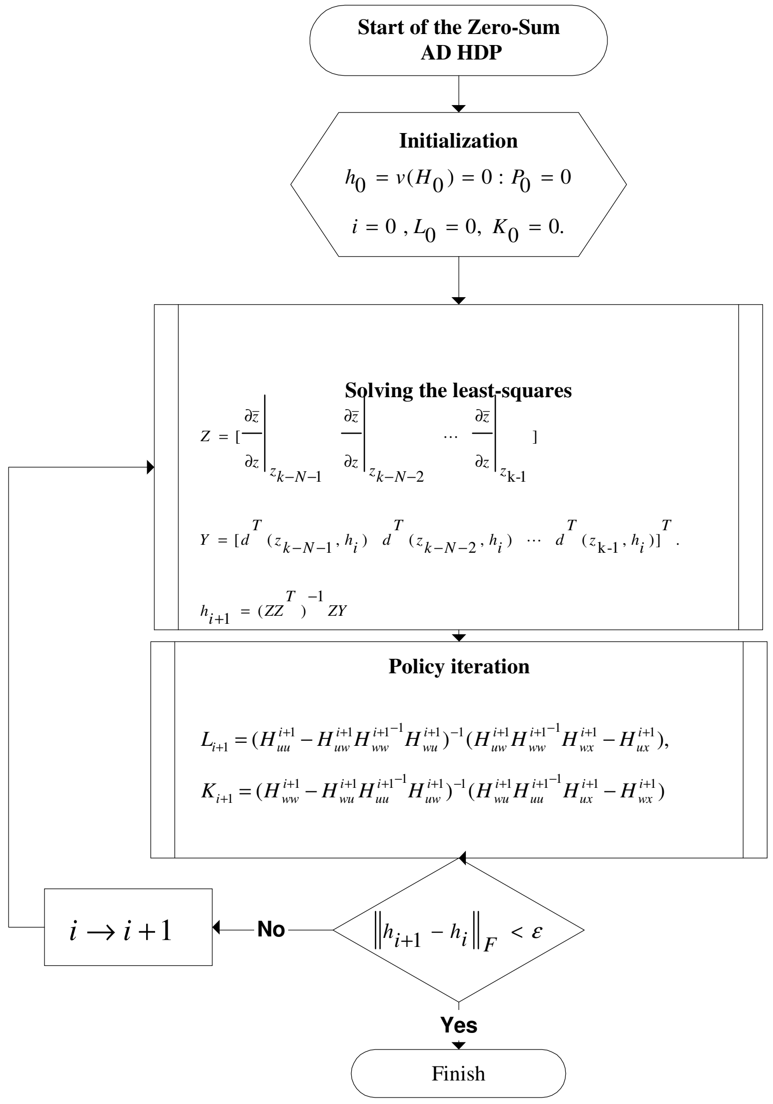

Start of the Zero-Sum AD HDP
Initialization
$h_0 = v(H_0) = 0 : P_0 = 0$
$i = 0, L_0 = 0, K_0 = 0.$
Solving the least-squares
$Z = [\frac{\partial \bar{z}}{\partial z}\Big|_{z_{k-N-1}} \frac{\partial \bar{z}}{\partial z}\Big|_{z_{k-N-2}} \cdots \frac{\partial \bar{z}}{\partial z}\Big|_{z_{k-1}}]$
$Y = [d^T(z_{k-N-1}, h_i)\ d^T(z_{k-N-2}, h_i)\ \cdots\ d^T(z_{k-1}, h_i)]^T.$
$h_{i+1} = (ZZ^T)^{-1} ZY$
Policy iteration
$L_{i+1} = (H_{uu}^{i+1} - H_{uw}^{i+1} H_{ww}^{i+1^{-1}} H_{wu}^{i+1})^{-1} (H_{uw}^{i+1} H_{ww}^{i+1^{-1}} H_{wx}^{i+1} - H_{ux}^{i+1}),$
$K_{i+1} = (H_{ww}^{i+1} - H_{wu}^{i+1} H_{uu}^{i+1^{-1}} H_{uw}^{i+1})^{-1} (H_{wu}^{i+1} H_{uu}^{i+1^{-1}} H_{ux}^{i+1} - H_{wx}^{i+1})$
$\|h_{i+1} - h_i\|_F < \varepsilon$
No
$i \to i+1$
Yes
Finish

Figure 4.2.The ADDHP algorithm

The ADDHDP algorithm for zero-sum games follows by iterating between (4.54) and (4.62). In the remaining of this section, it will be shown that this policy iteration technique will cause Q_i to converge to the optimal Q^*. Note that the model of the system is required to update the critic network.

4.3.3 Convergence of the ADDHP Algorithm

Now the convergence proof of the proposed ADDHP algorithm for zero-sum games is presented.

Lemma 4.4: Iterating on equations (4.54), and (4.62) is equivalent to

$$H_{i+1} = G + \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_iA & L_iB & L_iE \\ K_iA & K_iB & K_iE \end{bmatrix}, \tag{4.63}$$

under the assumption that the system is sufficiently excited.

Proof: The least-squares problem is defined in equation (4.62) which is

$$h_{i+1} = (ZZ^T)^{-1} ZY .$$

Under the excitation condition assumption, the inverse operator exists. Substituting (4.56) in (4.62), one has

$$h_{i+1} = (ZZ^T)^{-1} ZZ^T v(G + \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}), \tag{4.64}$$

which can be written as

$$h_{i+1} = v(G + \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ LA & LB & LE \\ KA & KB & KE \end{bmatrix}) ,$$

where v is the vectorized function in the Kronecker product.

Since the matrix H_{i+1} which reconstructed from h_{i+1} is symmetric, iteration on h_i is equivalent to the following iteration

$$H_{i+1} = G + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}, \tag{4.65}$$

which is equivalent to (4.63). □

Lemma 4.5 The matrices H_{i+1}, L_{i+1} and K_{i+1} can be written as

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}. \tag{4.66}$$

$$\begin{aligned} L_{i+1} = {} & (I + B^T P_i B - B^T P_i E (E^T P_i E - \gamma^2 I)^{-1} E^T P_i B)^{-1} \times \\ & (B^T P_i E (E^T P_i E - \gamma^2 I)^{-1} E^T P_i A - B^T P_i A), \end{aligned} \tag{4.67}$$

$$\begin{aligned} K_{i+1} = {} & (E^T P_i E - \gamma^2 I - E^T P_i B (I + B^T P_i B)^{-1} B^T P_i E)^{-1} \times \\ & (E^T P_i B (I + B^T P_i B)^{-1} B^T P_i A - E^T P_i A). \end{aligned} \tag{4.68}$$

where P_i is given as

$$P_i = \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix}. \tag{4.69}$$

Proof: Equation (4.66) can be written as

$$\begin{aligned} H_{i+1} &= G + \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix}^T H_i \begin{bmatrix} A & B & E \\ L_i A & L_i B & L_i E \\ K_i A & K_i B & K_i E \end{bmatrix} \\ &= G + \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix} \begin{bmatrix} A & B & E \end{bmatrix}. \end{aligned}$$

Since

$$P_i = \begin{bmatrix} I & L_i^T & K_i^T \end{bmatrix} H_i \begin{bmatrix} I \\ L_i \\ K_i \end{bmatrix},$$

then it follows that

$$H_{i+1} = \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}.$$

Using equations (4.54) and (4.66), one obtains (4.67) and (4.68). □

Lemma 4.6: Iterating on H_i is similar to iterating on P_i as

$$P_{i+1} = A^T P_i A + R - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix} \tag{4.70}$$

with P_i defined as in (4.69).

Proof: From (4.69), one has

$$P_{i+1} = \begin{bmatrix} I & L_{i+1}^T & K_{i+1}^T \end{bmatrix} H_{i+1} \begin{bmatrix} I \\ L_{i+1} \\ K_{i+1} \end{bmatrix},$$

and using (4.66), one obtains

$$\begin{aligned} P_{i+1} &= \begin{bmatrix} I & L_{i+1}^T & K_{i+1}^T \end{bmatrix} \begin{bmatrix} A^T P_i A + R & A^T P_i B & A^T P_i E \\ B^T P_i A & B^T P_i B + I & B^T P_i E \\ E^T P_i A & E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix} \begin{bmatrix} I \\ L_{i+1} \\ K_{i+1} \end{bmatrix} \\ &= R + L_{i+1}^T L_{i+1} - \gamma^2 K_{i+1}^T K_{i+1} + (A^T + L_{i+1}^T B^T + K_{i+1}^T E^T) P_i (A + BL_{i+1} + EK_{i+1}) \end{aligned} \tag{4.71}$$

Using (4.41) and (4.42), one has

$$P_{i+1} = A^T P_i A + R - [A^T P_i B \quad A^T P_i E] \begin{bmatrix} I + B^T P_i B & B^T P_i E \\ E^T P_i B & E^T P_i E - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^T P_i A \\ E^T P_i A \end{bmatrix}.$$ □

Theorem 4.2: Assume that the linear quadratic zero-sum game is solvable and has a value under the state feedback information structure. Then, iterating on equation (4.63), with $H_0 = 0$, $L_0 = 0$ and $K_0 = 0$ converges. Moreover $Q_i \to Q^*$ and

$$xPx = \min_u \max_w Q^*(x,u,w) = \max_w \min_u Q^*(x,u,w)$$

with P solving the algebraic Riccati equation (4.5).

Proof: In [3] it is shown that iterating on the algebraic Riccati equation (4.70) with $P_0 = 0$ converges to P that solves (4.5). Since Lemma 4.6 shows that iterating on H_i matrix is equivalent to iterating on P_i, then as $i \to \infty$

$$H_i \to \begin{bmatrix} A^T PA + R & A^T PB & A^T PE \\ B^T PA & B^T PB + I & B^T PE \\ E^T PA & E^T PB & E^T PE - \gamma^2 I \end{bmatrix}.$$

hence from (4.69) $H_0 = 0$, $L_0 = 0$ and $K_0 = 0$ implies that $P_0 = 0$, one concludes that $Q_i \to Q^*$. □

The convergence proof has been just established for the ADDHP algorithm assuming the least-squares problem (4.62) is solved completely; *i.e.* the exciting condition is satisfied. Note that an easy way to initialize the algorithm in Figure 2 is by selecting $H_0 = 0$, $u_0 = 0$ and $w_0 = 0$.

In the next section, the developed ADHDP and ADDHP algorithms are used to derive suboptimal H_∞ controllers by the forward time solution technique. The practical relevance of the developed algorithms will thus become clear.

4.4 Online ADP H_∞ Autopilot Controller Design for an F-16 Aircraft

In this design application, the zero-sum game that corresponds to the H_∞ controller problem is solved for an F-16 aircraft autopilot design. The H-infinity approach is used, which is enabled by the ADP procedures in this chapter. H-infinity design has been proven highly effective in the design of feedback control systems with robustness and disturbance rejection capabilities [15].

The F-16 short period dynamics has three states given as

$$x = \begin{bmatrix} \alpha \\ q \\ \delta_e \end{bmatrix}$$

where α is the angle of attack, q is the pitch rate and δ_e is the elevator deflection angle. The discrete-time plant model of this aircraft dynamics is a discretized version of the continuous-time one given in [4]. We used standard zero-order-hold discretization techniques explained in [14] and easily implemented in the MATLAB control systems toolbox to obtain the sampled data plant

$$A = \begin{bmatrix} 0.906488 & 0.0816012 & -0.0005 \\ 0.0741349 & 0.90121 & -0.000708383 \\ 0 & 0 & 0.132655 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.00150808 \\ -0.0096 \\ 0.867345 \end{bmatrix} \quad E = \begin{bmatrix} 0.00951892 \\ 0.00038373 \\ 0 \end{bmatrix}. \tag{4.72}$$

with sampling time $T = 0.1$. In this H_∞ design problem, the disturbance attenuation is $\gamma = 1$.

4.4.1 H_∞ Solution Based on the Riccati Equation

Since the ADP designs developed in this chapter to solve the H_∞ controller design problem are based on an iterative form of the Riccati equation, in Figure 4.3 the convergence of P_i to the solution of the GARE (4.5) is shown when done offline with $P_0 = 0$.

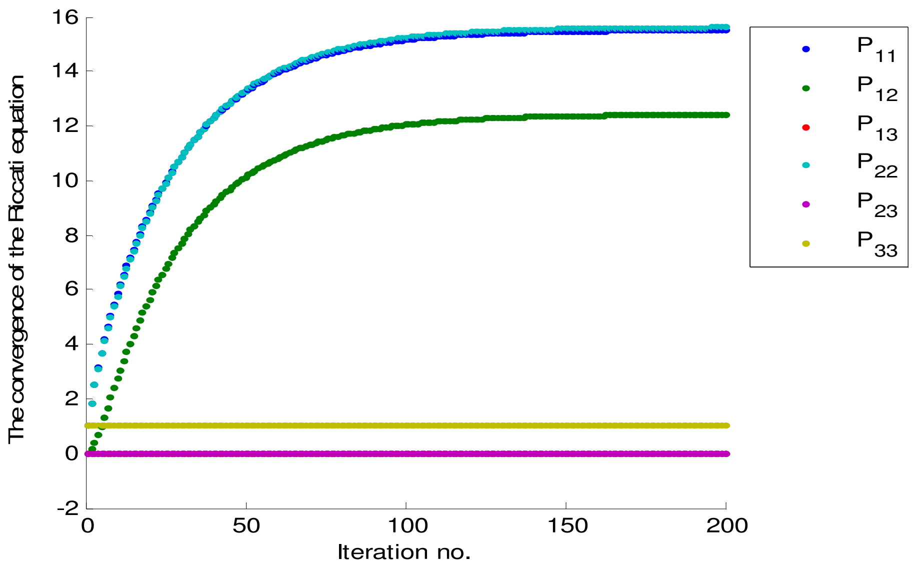


Figure 4.3. The convergence of P_i by iterating on Riccati equation

It is noticed from Figure 4.3 that for the discretized aircraft dynamics (4.72), P_i converges after at least 100 iterations with $\gamma = 1$ to

$$P = \begin{bmatrix} 15.5109 & 12.4074 & -0.0089 \\ 12.4074 & 15.5994 & -0.0078 \\ -0.0089 & -0.0078 & 1.0101 \end{bmatrix} \tag{4.73}$$

the policies that associated with the are

$$L = \begin{bmatrix} 0.0733 & 0.0872 & -0.0661 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.1476 & 0.1244 & 0 \end{bmatrix}$$

Note that $P \geq 0$ and hence from [7] this implies that

$$\frac{\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^{*T} u_k^*}{\sum_{k=0}^{\infty} w_k^T w_k} \leq \gamma^2 \tag{4.74}$$

for all finite energy disturbances, *i.e.*

$$\sum_{k=0}^{\infty} w_k^T w_k$$

bounded, and hence $u^*(x_k)$ has the well-known robustness and disturbance rejection capabilities of H_∞ control.

Next, the ADP algorithms developed in this chapter are used to design an H_∞ controller for the discretized aircraft dynamics (4.72) with $\gamma = 1$ in forward time.

4.4.2 ADHDP based H_∞ Autopilot Controller Design

In this part, the ADHDP algorithm developed in Section 4.2 of this c is applied to solve for the H_∞ autopilot controller forward in time. The recursive least-squares algorithm is used to tune the parameters of the critic network on-line. The parameters of the actions networks are updated according to (4.23).

In this ADHDP design, the states of the aircraft are initialized to be $x_0 = \begin{bmatrix} 10 & 5 & -2 \end{bmatrix}$. Any values could be selected. The parameters of the critic network and the actions networks are initialized to zero. Following this initialization step, the aircraft dynamics are run forward in time and tuning of the parameter structures is performed by observing the states on-line.

In Figures 4.4 and 4.5, the states and the inputs to the aircraft are shown with respect to time. In order to maintain the excitation condition, one can use several standard schemes, including covariance resetting, state resetting, or injection of a small probing noise signal. In this example, probing noise is injected to the control and disturbance policies. Hence the persistency of excitation condition required for the convergence of the recursive least-squares tuning, i.e. avoiding the parameter drift problem, will hold.

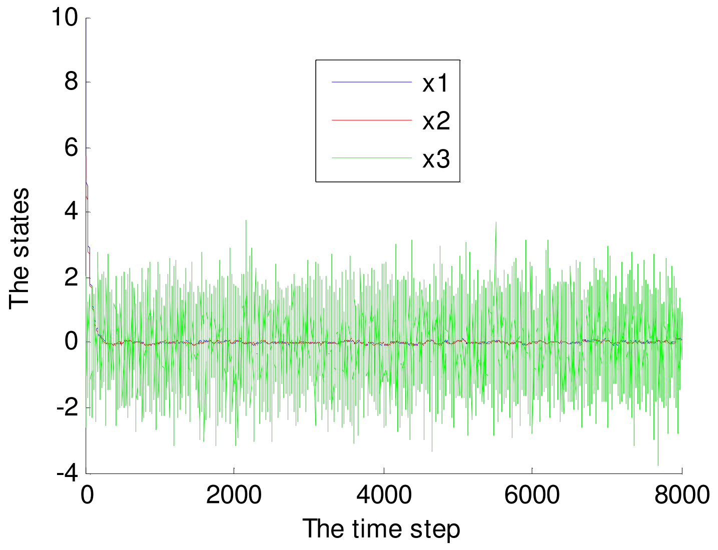


Figure 4.4. States trajectories of the ADHDP algorithm.

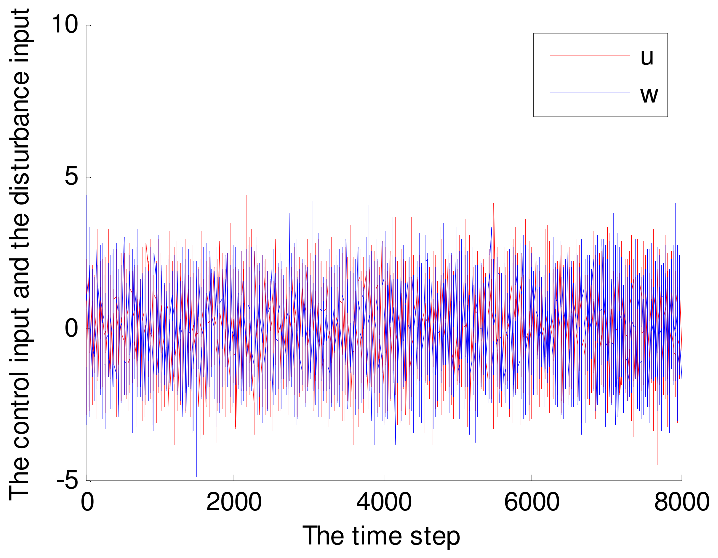


Figure 4.5. The control and disturbance in the ADHDP.

In Figures 4.6, 4.7 and 4.8, the convergence of P, which is found from the parameters (H_i) of the critic network and action networks as in (4.10), and the actions networks is shown. As expected, P_i converges to P in (4.73) that solves the GARE equation. It takes the critic network 4000 time steps to converge to P. The reason for this is that 40 readings are required to tune the critic network at each update to solve for each H_i. Since as shown in Figure 4.3, the action networks require to be updated at least 100 times, this implies that the over all time steps required for the convergence of the ADHDP algorithm are about 4000 time steps. It is important to note that if the problem is solved using the least-square less time step is needed for the algorithm to converge to the solution.

It is important to realize that here we used probing noise, see (4.57), which is injected to the control and disturbance inputs, to determine the optimal solution for the game problem, as given by the converged P associated with the converged critic network parameters in Figure 4.6 and action network parameters in Figures 4.7 and 4.8. Probing noise provides the excitation conditions needed to get parameter convergence. Once these parameters are known, the H_∞ controller has been found. Then, one can use the parameters of the control action network as the final parameters of the controller in any on-line control runs, without having to deliberately insert any excitation signals to the system.

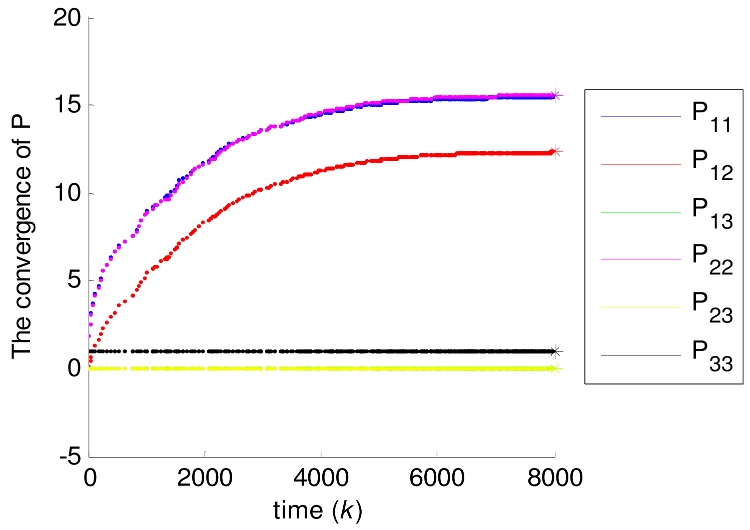


Figure 4.6. Convergence of P_i. in the ADHDP.

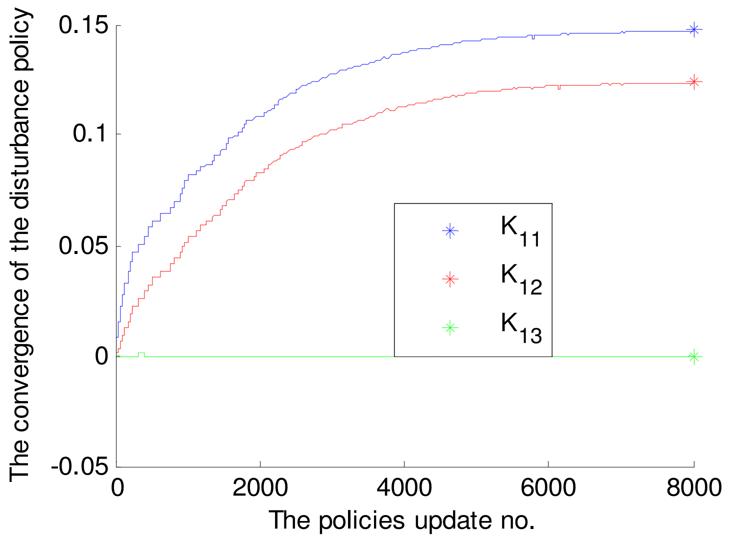


Figure 4.7. Convergence of the disturbance action network parameters in the ADHDP.

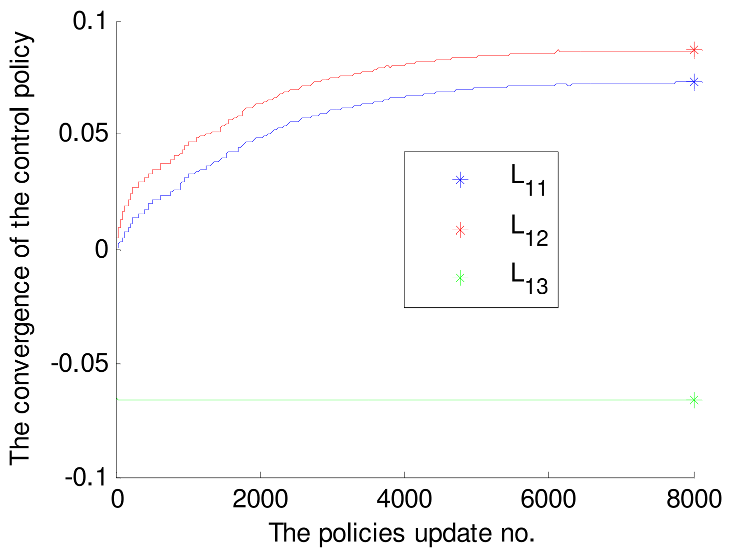


Figure.4.8. Convergence of the control action network parameters in the ADHDP.

Next, the ADDHP algorithm developed in Section 4 is applied to this aircraft design problem.

4.4.3 ADDHP based H_∞ Autopilot Controller Design

In this part, the ADDHP algorithm developed in Section 4.3 of this chapter is applied to solve for the H_∞ autopilot controller in forward time. The recursive least-squares algorithm is used to tune the parameters of the critic network. The parameters of the actions networks are updated according to (4.60).

In this ADDHP design, the states of the aircraft are initialized to be $x_0 = [10 \quad 5 \quad -2]$. The parameters of the critic network and the actions networks are initialized to zero. Following this initialization step, the aircraft dynamics are run forward in time and tuning of the parameter structures happen by observing the states on-line.

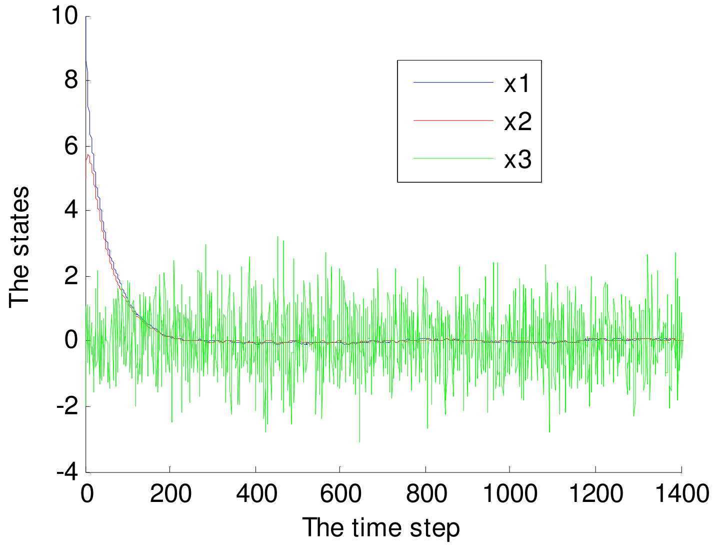


Figure 4.9. States trajectories in the ADDHP.

In Figures 4.9 and 4.10, the states and the inputs to the aircraft are shown with respect to time. Note that the probing noise are used to inject the inputs, see (4.60), so the persistency of excitation condition required for the convergence of the recursive least-squares tuning, i.e. avoiding the parameter drift problem, will hold.

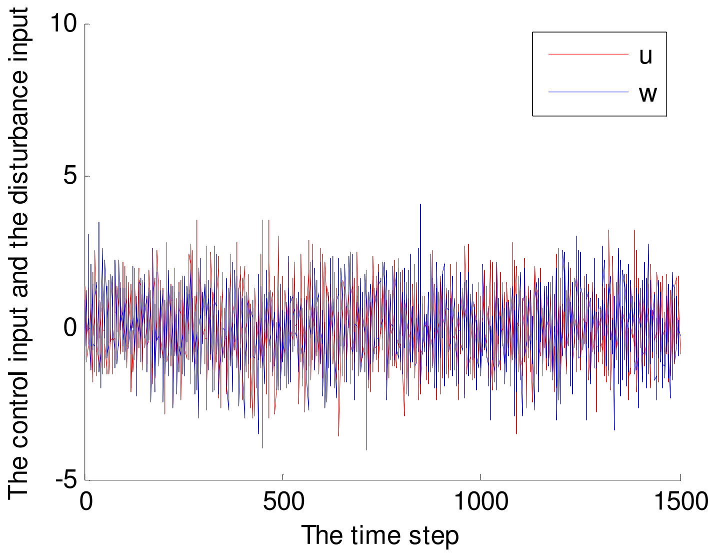


Figure 10. The control and disturbance in the ADDHP.

In Figures 4.11, 4.12 and 4.13, the convergence of P , which is found from the parameters of the critic network and the actions networks as in (4.10). and the action networks is shown. As expected, P_i converge to P in (4.73) that solve the GARE equation. It takes the critic network 700 time steps to converge to P_i The reason for this is that 7 readings are required to tune the critic network at to solve for each H_i. Since as shown in Figure 4.3, the action networks require to be updated at least 100 times, this implies that the over all time steps required for the convergence of the ADDHP algorithm are about 700 time steps.

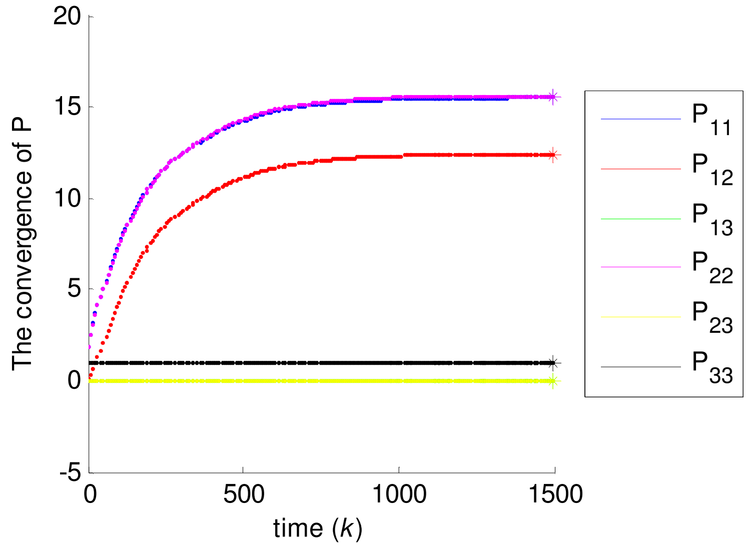


Figure 4.11. Convergence of the P_i in the ADDHP.

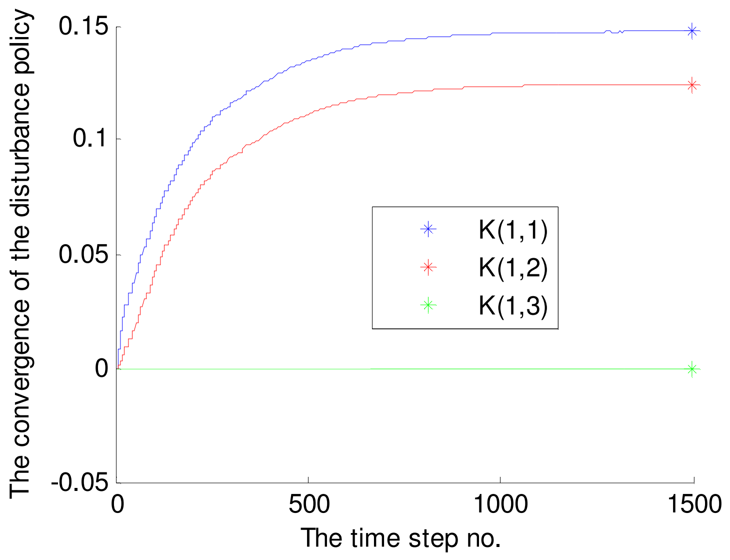


Figure 4.12. Convergence of the disturbance action network parameters in the ADDHP.

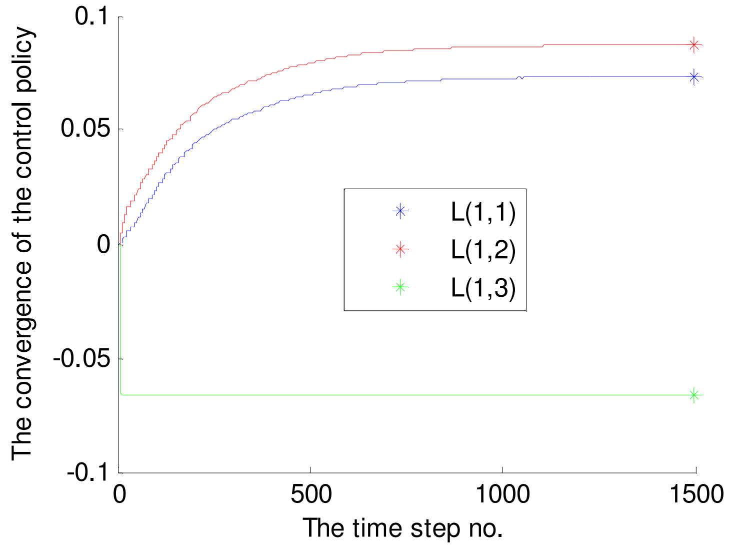


Figure 4.13. Convergence of the control action network parameters in the ADDHP.

It is clear that in the ADDHP algorithm, the parameters of the critic network converge to the solution of the GARE faster than the case in the ADHDP algorithm. This is because in ADDHP one has vector gradient information available for tuning, not only scalar information as in ADHDP. That is, in ADDHP the target value for the action network is a vector, while in ADHDP it is a scalar.

Probing noise was used here to provide the excitation conditions needed to get parameter convergence in the critic and action networks. Once these parameters are known, the H-infinity controller has been found. Then, one can use the parameters of the control action network as the final parameters of the controller in any on-line control runs, without having to deliberately insert any excitation signals to the system

4.5 Conclusion

In this chapter, two on-line Approximate Dynamic Programming techniques is introduced to solve the discrete-time zero-sum game problem with continuous state and action spaces. Two of the ADP techniques, namely Action Dependent Heuristic Dynamic Programming, and Action Dependent Dual Heuristic Dynamic Programming are discussed. The derivation of the policies and the convergence of the ADHDP and ADDHP are provided. It is clear that the convergence to the optimal solution in the ADDHP algorithm is faster than the ADHDP, as gradient information, a vector, as used in ADDHP provides more information than scalar function information as used in ADHDP, therefore the number of points needed to solve the least-squares problem in the ADDHP is less than that in ADHDP. On the other hand, in the ADHDP algorithm the system model is not needed to tune the action networks nor the critic network, while in the ADDHP algorithm the system model is needed to tune the critic network.

The results presented herein are directly applicable in practice since they provide means to solve the H-infinity control problem, which is highly effective in feedback control systems design. A provided aircraft design example makes the point. It is interesting to see that when designing the H-infinity controller in forward time, one needs to provide an input signal that acts as a disturbance that is tuned to be the worst case disturbance in forward time.

Once the H-infinity controller is found, one can use the parameters of the control action network as the final parameters of the controller, without having to deliberately inserting any disturbance signal to the system. Disturbance is from now is determined by the nature of the process and the surrounding environment.

The results in this chapter can be summarized as a way to solve the linear quadratic discrete-time zero-sum game forward in time without knowing the dynamical model.

CHAPTER 5

APPLICATION OF THE ADHDP FOR THE POWER SYTEM AND SYTEM IDENTIFICATION

5.1 Power System Model Plant

Power system generators are complex nonlinear systems [33],[17]. However during normal operation the system load, which causes the nonlinearity, has only small variations. Linear models can be used to represent the system dynamics around an operating point specified by a constant load value. Although this assumption seems to have simplified the design problem of a load-frequency controller for the system, a problem rises from the fact that in an actual plant the parameter values are not precisely known.

The ADHDP algorithm in this chapter is used to find an H_∞ controller for Discrete-time (DT) power system without knowing the system dynamics. In this part the simulation is done by using the Matlab, so for simulation purposes a system dynamics should be adapted to simulate the plant model. The DT model is obtained from the continuous-time (CT) model of the system by using the zero-order hold (ZOH) technique. The CT system model is adapted from [33], and it is given as

$$\dot{x} = Ax(t) + Bu(t) + E\Delta P_d(t) \qquad (5.1)$$

where

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

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

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

$$E^T = \begin{bmatrix} 1 - K_p/T_p & 0 & 0 & 0 \end{bmatrix}$$

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

The system parameter ranges as specified in [33] are:

$$\begin{aligned} &1/T_p \in [0.033, 0.1] \\ &K_p/T_p \in [4, 12] \\ &1/T_T \in [2.564, 4.762] \\ &1/T_G \in [9.615, 17.857] \\ &1/RT_G \in [3.081, 10.639] \end{aligned}$$

Though the ranges of the system parameters are known, the exact values are not known, so the system model for power system usually has a certain degree of uncertainty. Therefore, the goal of this chapter is to design an H_∞ load-frequency controller without knowing the system dynamics.

5.2 H-Infinity Control Design Using ADHDP Algorithm

H_∞ Controllers have been proven to be highly effective in the design of feedback control systems with robustness and disturbance rejection capabilities. The presented H_∞ controller design is a model-free online tuning design that is based on the Q-learning method presented in this chapter 4.

In this section, the ADHDP algorithm proposed in chapter 4 will be used to design an load-frequency H_∞ controller for single generator. A comparison between the designed controller in this chapter and the controller in [33] will be provided.

The adapted power system appears in [33]. The system in [33] is in continuous-time (CT) representation. Our proposed control design is for discrete-time (DT), so zero-order holder is used to discretize the CT system. Note the system model is needed to simulate the system response with the applied control law, not for tuning the control input.

The CT system matrices with nominal value is given as

$$Ac = \begin{bmatrix} -0.0665 & 8.000 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.60 & 0 & 0 & 0 \end{bmatrix}$$

$$Bc = \begin{bmatrix} 0 \\ 0 \\ 13.736 \\ 0 \end{bmatrix} \qquad Ec = \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

so the DT system matrices with sampling time $T = .1$ sec. is give as

$$Ad = \begin{bmatrix} 0.9704 & 0.6629 & 0.0849 & -0.0446 \\ -0.0762 & 0.6724 & 0.1584 & -0.1462 \\ -0.3954 & -0.1663 & 0.2367 & -0.7403 \\ 0.0594 & 0.0212 & 0.0019 & 0.9993 \end{bmatrix} \tag{5.2}$$

$$Bd = \begin{bmatrix} 0.0446 \\ 0.1462 \\ 0.7403 \\ 0.0007 \end{bmatrix} \qquad Ed = \begin{bmatrix} -0.7924 \\ 0.0230 \\ 0.1893 \\ -0.0239 \end{bmatrix}$$

It is important to note that the ADHDP algorithm the system matrices are not needed to design the controller. It is used only to simulate the plant model.

In Figure 5.1, the convergence of the value $P_i \rightarrow P$ that solve the GARE is provided

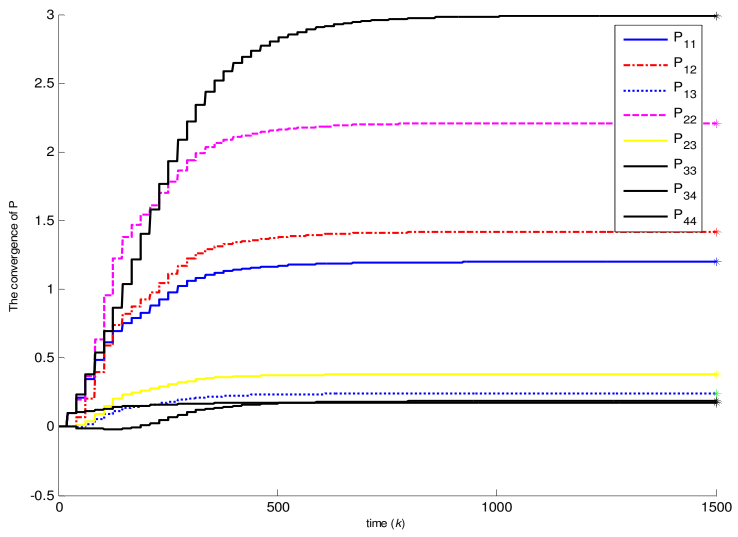


Figure 5.1 The convergence of $P_i \rightarrow P$

In figure 5.2, the convergence of the control policy to the optimal policy is shown.

$$L_i \to \begin{bmatrix} -1.6299 & -2.9620 & -0.6958 & -0.9747 \end{bmatrix} \tag{5.3}$$

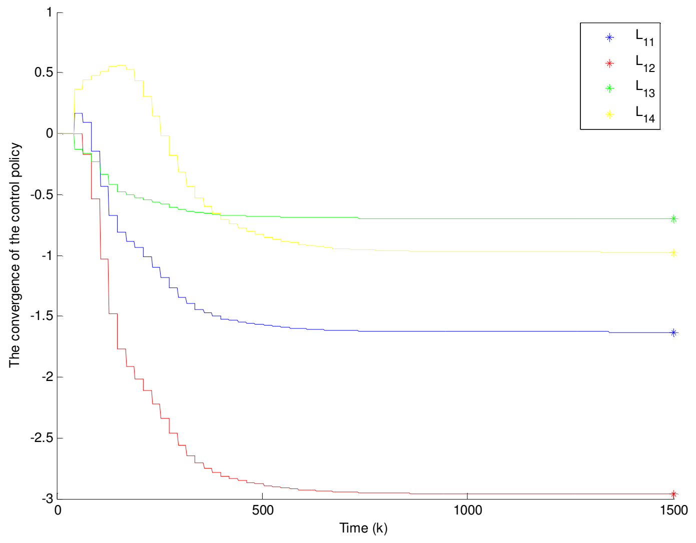


Figure 5.2. The convergence of the control policy

In the next figures, the system will be run using the H_∞ controller designed in this section, and it will be compared with the controller designed in [33]

Figure 5.3 shows the system state trajectories when applying the controller found using the ADHDP algorithm. The simulation will be run applying constant disturbance 0.1 p.u .

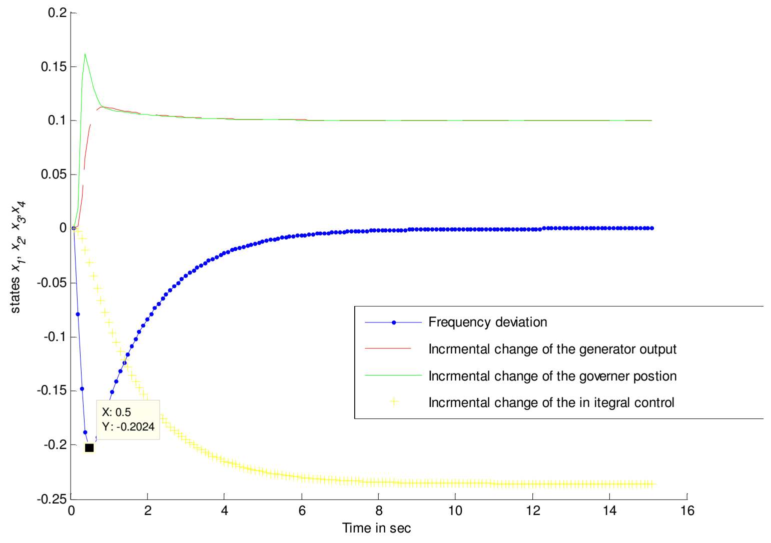


Figure 5.3. The states trajectories for the system with the H_∞ controller

Figure 5.4 shows the system state trajectories when using the controller found in [33]. The simulation will be run applying constant disturbance 0.1 p.u, where

$$L = [1.839\ 4.762\ 1.516\ 1.658] \tag{5.4}$$

From Figure 5.3 and 5.4 , one can notice that the maximum frequency deviation when using the H_∞ controller is less than the maximum frequency deviation when using the controller in [33] by 19.3%

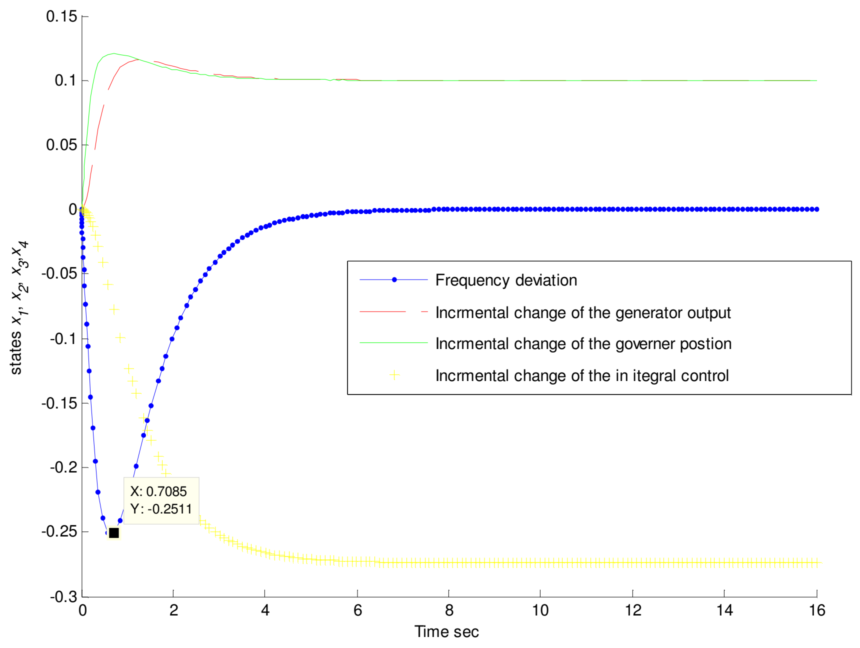


Figure 5.4 The states trajectories for the system with the controller designed in [33]

Next the same simulation will be run by applying a spike disturbance.

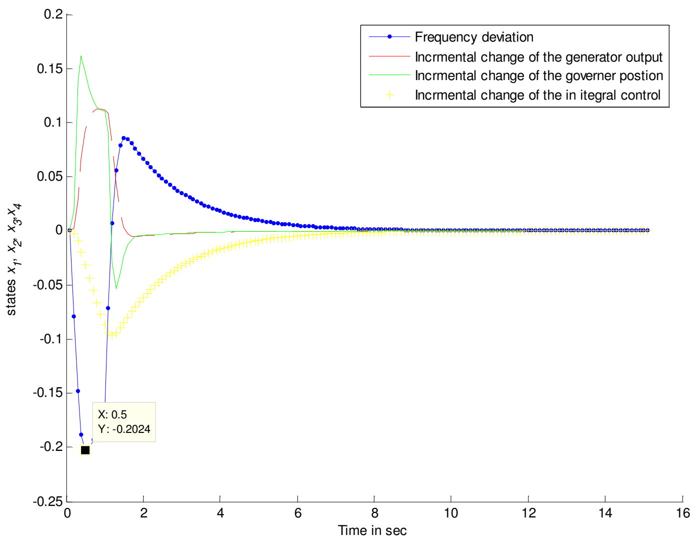


Figure 5.5. The states trajectories for the system with the H_∞ controller

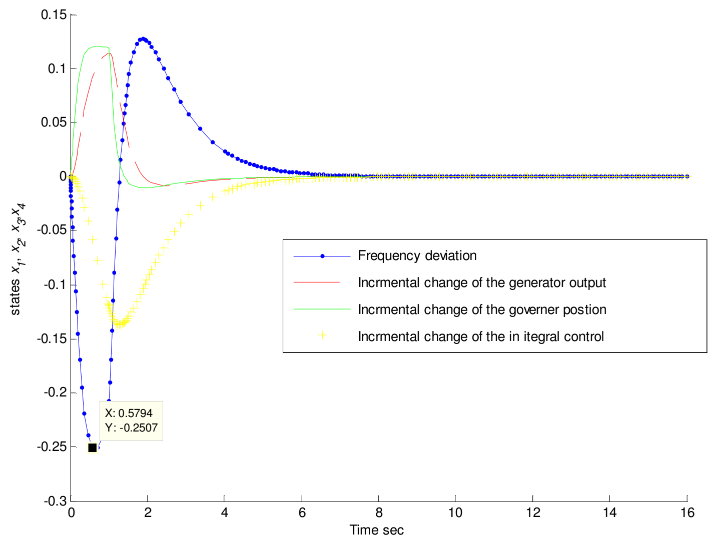


Figure 5.6. The states trajectories for the system with controller designed in [33].

From Figure 5.5 and 5.6, one can notice that the maximum frequency deviation when using the H_∞ control is less than the maximum frequency deviation when using the controller in [33] by 19.4%.

5.3 System Identification

In this section dynamical system identification will be discussed. The information used in the ADHDP algorithm will be used to identify the dynamical system.

In general, any DT linear system can be described as

$$x_{k+1} = Ax_k + Bu_k + Ew_k$$

where $A \in R^{nxn}$, $B \in R^{m1}$, $E \in R^{m2}$. The number of unknowns are equal to $(n \times (n + m1 + m2))$, and as the system is linear one can solve for the unknowns' by collecting at least

$$q = n + m1 + m2 + 1 \tag{5.5}$$

measurements. Every measurement will give n excited information, so the total number of equations will be $(n \times (n + m1 + m2))$, the same as the number of the unknowns. The measurements are collected as follows

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \cdot \\ \cdot \\ \cdot \\ x_{k+q} \end{bmatrix} = \begin{bmatrix} Ax_k + Bu_k + Ew_k \\ Ax_{k+1} + Bu_{k+1} + Ew_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ Ax_{k+q-1} + Bu_{k+q-1} + Ew_{k+q-1} \end{bmatrix}$$

where k is any arbitrary sampling time. One can rewrite the above equation as

$$\begin{bmatrix} x_{k+1}^T \\ x_{k+2}^T \\ \cdot \\ \cdot \\ \cdot \\ x_{k+q}^T \end{bmatrix} = \begin{bmatrix} x_k^T & u_k & w_k \\ x_{k+1}^T & u_{k+1} & w_{k+1} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{k+q-1}^T & u_{k+q-1} & w_{k+q-1} \end{bmatrix} \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix}$$

Solving the least-square problem one has

$$\left(\begin{bmatrix} x_k^T & u_k & w_k \\ x_{k+1}^T & u_{k+1} & w_{k+1} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{k+q-1}^T & u_{k+q-1} & w_{k+q-1} \end{bmatrix}^T \begin{bmatrix} x_k^T & u_k & w_k \\ x_{k+1}^T & u_{k+1} & w_{k+1} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{k+q-1}^T & u_{k+q-1} & w_{k+q-1} \end{bmatrix} \right)^{-1} \begin{bmatrix} x_k^T & u_k & w_k \\ x_{k+1}^T & u_{k+1} & w_{k+1} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_{k+q-1}^T & u_{k+q-1} & w_{k+q-1} \end{bmatrix}^T \begin{bmatrix} x_{k+1}^T \\ x_{k+2}^T \\ \cdot \\ \cdot \\ \cdot \\ x_{k+q}^T \end{bmatrix} = \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} \tag{5.6}$$

Note that the states measurements are the same as the one are used to tune the ADHDP algorithm, and as the noise is injected in the control input the measurements are linearly independent, so the least-square problem appears in (5.6) is solvable.

Now the measurements used to find the H_∞ controller in the previous section will be used to identify the dynamical system. The number of unknowns for the system described in (5.1) are equal to $(4\times(4+1+1))$, so one needs at least 6+1 measurements which will give 6×4 information. The information is collected starting form the sampling time $k=500$.

$$\begin{bmatrix} 0.3372 & -0.0894 & -0.2239 & 0.0203 \\ 0.3503 & -0.1490 & -0.3220 & 0.0412 \\ 0.2889 & -0.1717 & -0.1652 & 0.0604 \\ 0.2542 & -0.1576 & -0.1036 & 0.0767 \\ 0.2548 & -0.1582 & -0.1933 & 0.0920 \\ 0.1427 & -0.1524 & -0.1017 & 0.1039 \end{bmatrix} =$$

$$\begin{bmatrix} 0.2161 & -0.0136 & -0.3501 & 0.0034 & -0.0207 & -0.2110 \\ 0.3372 & -0.0894 & -0.2239 & 0.0203 & -0.1479 & -0.1374 \\ 0.3503 & -0.1490 & -0.3220 & 0.0412 & 0.0981 & -0.0915 \\ 0.2889 & -0.1717 & -0.1652 & 0.0604 & 0.1210 & -0.1250 \\ 0.2542 & -0.1576 & -0.1036 & 0.0767 & -0.0105 & -0.1581 \\ 0.2548 & -0.1582 & -0.1933 & 0.0920 & 0.1219 & -0.0194 \end{bmatrix} \begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix}$$

$$\begin{bmatrix} A^T \\ B^T \\ E^T \end{bmatrix} = \begin{bmatrix} 0.9704 & -0.0762 & -0.3954 & 0.0594 \\ 0.6629 & 0.6724 & -0.1663 & 0.0212 \\ 0.0849 & 0.1584 & 0.2367 & 0.0019 \\ -0.0446 & -0.1462 & -0.7403 & 0.9993 \\ 0.0446 & 0.1462 & 0.7403 & 0.0007 \\ -0.7924 & 0.0230 & 0.1893 & -0.0239 \end{bmatrix} \tag{5.7}$$

$$A = \begin{bmatrix} 0.9704 & 0.6629 & 0.0849 & -0.0446 \\ -0.0762 & 0.6724 & 0.1584 & -0.1462 \\ -0.3954 & -0.1663 & 0.2367 & -0.7403 \\ 0.0594 & 0.0212 & 0.0019 & 0.9993 \end{bmatrix} B = \begin{bmatrix} 0.0446 \\ 0.1462 \\ 0.7403 \\ 0.0007 \end{bmatrix}. E = \begin{bmatrix} -0.7924 \\ 0.0230 \\ 0.1893 \\ -0.0239 \end{bmatrix} \tag{5.8}$$

As expected the values obtained from (5.7) are the same as in (5.2).

5.4 Conclusion

In this chapter the on-line ADP technique based on Q-learning appears in chapter 4 is used to find H_∞ load-frequency power system controller. The tuning algorithm is used in this chapter is ADHDP. In the ADHDP algorithm the system model is not needed to tune the action networks, *i.e.* the controller, nor the critic network. The results in this chapter can be summarized as a model-free approach to solve the linear quadratic discrete-time zero-sum game forward in time.

In this chapter the measurements used to tune the controller also used to identify the system dynamics. As shown one needs only a few measurements which are equal to equation (5.5). The noise injection allows the measurements to be excited, and solve for the least square problem.

It is interesting to see that when designing the H_∞ controller in forward time, one needs to provide an input signal that acts as a disturbance that is tuned to be the worst case disturbance in forward time. Once the H_∞ controller is found, one can use the parameters of the control action network as the final parameters of the controller, without having to deliberately inserting any disturbance signal to the system.

CHAPTER 6

NONLINEAR HEURISTIC DYNAMIC PROGRAMMING OPTIMAL CONTROL DESIGN

In this chapter, a policy iteration scheme based on approximate dynamic programming (ADP), namely Heuristic Dynamic Programming (HDP), is used to solve for the optimal control policy and the value function of the Hamilton Jacobi Bellman equation (HJB) that appears in infinite-horizon discrete-time (DT) nonlinear optimal control. Two neural networks are used; the first is used to approximate the value function while a second network is used to approximate the optimal control policy. A significant result of this cahpter is that a rigorous proof of convergence of the HDP algorithm is provided when applied to input-affine nonlinear discrete-time systems with continuous state and action spaces. Furthermore, because of the use of a neural network to approximate the action policy, complete knowledge of the plant model does not become a requirement. Two examples are included to illustrate the developed theory

6.1 The Discrete-Time HJB Equation

Consider an affine in input nonlinear dynamical-system of the form

$$x_{k+1} = f(x_k) + g(x_k)u(x_k) \tag{6.1}$$

where $x \in \mathbb{R}^n$, $f(x) \in \mathbb{R}^n$, $g(x) \in \mathbb{R}^{n\times m}$ and the input $u \in \mathbb{R}^m$. Assume that the system (6.1) is stabilizable on compact set $\Omega \in \mathbb{R}^n$.

Definition 1. Stabilizable system: A nonlinear dynamical system is defined to be stabilizable on a compact set Ω, if there exist a control input $u \in \mathbb{R}^m$ such that, if the states start from anywhere in the region Ω, the state $x \to 0$ as $t \to \infty$

It is desired to find $u(x_k)$ which minimize the infinite-horizon cost function given as

$$V\ (x_k) = \sum_{n=k}^{\infty} x_n^T Q x_n + u^T (x_n) R u\ (x_n) \tag{6.2}$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are positive definite matrices. Hence, the class of controllers need to be stable and guarantee that (6.2) is finite, *i.e.* admissible controls [2].

Definition 2. Admissible Control: A control $u(x_k)$ is defined to be admissible with respect to (6.2) on a compact set Ω if $u(x_k)$ is continuous on Ω, $u(0) = 0$, u stabilizes (6.1) on Ω, and $\forall x_0 \in \Omega\ V(x_0)$ is finite.

Equation (6.2) can be written as

$$\begin{aligned} V\ (x_k) &= x_k^T Q x_k + u_k^T R u_k + \sum_{i=k+1}^{\infty} x_i^T Q x_i + u_i^T R u_i \\ &= x_k^T Q x_k + u_k^T R u_k + V(x_{k+1}) \end{aligned} \tag{6.3}$$

From Bellman optimality principle, the discrete-time HJB equation comes out to be

$$V^*(x_k) = \min_{u_k} (x_k^T Q x_k + u_k^T R u_k + V^*(x_{k+1})) \tag{6.4}$$

Note that the discrete-time HJB equation develops backward-in time. It is known that for the infinite-horizon optimization case, the value function is time-invariant and hence the discrete-time HJB in infinite-horizon becomes

$$V^*(x_k) = \min_{u_k}(x_k^T Q x_k + u_k^T R u_k + V^*(x_{k+1})) \tag{6.5}$$

The optimal control u^* satisfies the first order necessity condition for the gradient of right hand side of (6.5) with respect to u

$$\frac{\partial V^*(x_k)}{\partial u_k} = \frac{\partial(x_k^T Q x_k + u_k^T \; R u_k)}{\partial u_k} + \frac{\partial x_{k+1}}{\partial u_k}\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} = 0 \tag{6.6}$$

and therefore

$$u^*(x_k) = \frac{1}{2} R^{-1} g(x_k)^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \tag{6.7}$$

Substituting (6.7) in (6.5), one may write the discrete-time HJB

$$V^*(x_k) = x_k^T Q x_k + \frac{1}{4}\frac{\partial V^{*T}(x_{k+1})}{\partial x_{k+1}} g(x_k) R^{-1} g(x_k)^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} + V^*(x_{k+1}) \tag{6.8}$$

where $V^*(x)$ is the value function corresponding to the of the optimal control policy $u^*(x)$.

In the next section the nonlinear HDP algorithm is applied to solve for the value function V^* of the HJB equation (6.8) and present a convergence proof of the HDP policy iteration algorithm.

6.2 The Nonlinear HDP Algorithm

In the HDP algorithm, one starts with an initial cost function $V_0(x) = 0$, which is not necessarily the value function, and then finds a control policy u_0 as follows

$$u_o(x_k) = \arg\min_u(x_k^T Q x_k + u^T R u + V_0(x_{k+1})) \tag{6.9}$$

Once the policy u_0 is determined, one updates the cost function by computing V_1

$$\begin{aligned} V_1(x_k) &= x_k^T Q x_k + u_0^T(x_k) R u_0(x_k) + V_0(f(x_k) + g(x_k) u_0(x_k)) \\ &= x_k^T Q x_k + u_0^T(x_k) R u_0(x_k) + V_0(x_{k+1}) \end{aligned} \tag{6.10}$$

The HDP policy iterations scheme therefore requires iterating between a sequence of policies $u_i(x)$ determined by

$$u_i(x_k) = \arg\min_u (x_k^T Q x_k + u^T(x_k) R u(x_k) + V_i(x_{k+1})) \tag{6.11}$$

and a sequence of costs $V_i(x) \geq 0$ where

$$\begin{aligned} V_{i+1}(x_k) &= \min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1})) \\ &= x_k^T Q x_k + u_i^T(x_k) R u_i(x_k) + V_i(f(x_k) + g(x_k) u_i(x_k)) \end{aligned} \tag{6.12}$$

with i is an index representing iterations on the control policy while k is the time index. The result is an incremental optimization that is implemented forward in-time.

6.3 Convergence of the HDP Algorithm

In this section, the nonlinear case is considered as the proof of convergence is presented for the iteration between (6.11) and (6.12), that is $V_i \Rightarrow V^*$ and the control policy $u_i \Rightarrow u^*$ as $i \Rightarrow \infty$.

Lemma 1 Let μ_i be any arbitrary sequence of control policies, and u_i be the policies as in (6.11). Let V_i be as in (6.12) and Λ_i as

$$\Lambda_{i+1}(x_k) = x_k Q x_k + \mu_i^T R \mu_i + \Lambda_i(x_{k+1}). \tag{6.13}$$

If $V_0 = \Lambda_0 = 0$, then $V_i \leq \Lambda_i \ \forall i$.

Proof: It is straightforward from the fact that V_{i+1} is a result of minimizing the right hand side of equation (6.11) with respect to the control input u, while Λ_i is a result of any arbitrary control input. □

Lemma 2 Let the sequence $\{V_i\}$ be defined as in (6.12). If the system is stabilizable, then there is an upper bound Y such that $0 \le V_i \le Y \quad \forall i$.

Proof : Let $\eta(x_k)$ be any stabilizing and admissible control input, and Let $V_0 = Z_0 = 0$ where V_i is updated as (6.12) and Z_i is updated as

$$Z_{i+1}(x_k) = x_k Q x_k + \eta^T(x_k) R \eta(x_k) + Z_i(x_{k+1}). \tag{6.14}$$

It follows that the difference

$$\begin{aligned} Z_{i+1}(x_k) - Z_i(x_k) &= Z_i(x_{k+1}) - Z_{i-1}(x_{k+1}) \\ &= Z_{i-1}(x_{k+2}) - Z_{i-2}(x_{k+2}) \\ &= Z_{i-2}(x_{k+3}) - Z_{i-3}(x_{k+3}) \\ &\;. \\ &\;. \\ &\;. \\ &= Z_1(x_{k+i}) - Z_0(x_{k+i}) \end{aligned} \tag{6.15}$$

Then (6.15) can be written as

$$Z_{i+1}(x_k) - Z_i(x_k) = Z_1(x_{k+i}) - Z_0(x_{k+i}),$$

Since $Z_0(x_k) = 0$, so one has

$$\begin{aligned} Z_{i+1}(x_k) &= Z_1(x_{k+i}) + Z_i(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_{i-1}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-1}) + Z_{i-2}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-2}) + ... + Z_1(x_k) \end{aligned} \tag{6.16}$$

so equation (6.16) can be written as

$$\begin{aligned} Z_{i+1}(x_k) &= \sum_{j=0}^{i} Z_1(x_{k+j}) \\ &= \sum_{j=0}^{i} (x_{k+j}^T Q x_{k+j} + \eta^T(x_{k+j}) R \eta(x_{k+j})) \\ &\le \sum_{j=0}^{\infty} (x_{k+j}^T Q x_{k+j} + \eta^T(x_{k+j}) R \eta(x_{k+j})) \end{aligned} \tag{6.17}$$

Note that the system is stable, *i.e.* $x_k \to 0$ as $k \to \infty$, as the control input $\eta(x_k)$ is stabilizable and admissible, then

$$\forall i: \quad Z_{i+1}(x_k) \le \sum_{i=0}^{\infty} Z_1(x_{k+i}) \le Y$$

Form Lemma 1, one has

$$\forall i: \quad V_{i+1}(x_k) \le Z_{i+1}(x_k) \le Y \qquad ■$$

Now Lemma 1 and Lemma 2 will be used in the next main theorem.

Theorem 1 Define the sequence $\{V_i\}$ as in (6.12), with $V_0 = 0$. Then $\{V_i\}$ is a nondecreasing sequence in which $V_{i+1}(x_k) \ge V_i(x_k)\ \forall i$, and converges to the value function of the DT HJB, *i.e.* $V_i \Rightarrow V^*$ as $i \Rightarrow \infty$.

Proof: Let $V_0 = \Phi_0 = 0$ where V_i is updated as in (6.12) and, and Φ_i is updated as

$$\Phi_{i+1}(x_k) = (x_k Q x_k + u_{i+1}^T R u_{i+1} + \Phi_i(x_{k+1})) \tag{6.18}$$

with the policies u_i as in (6.11). We will first prove by induction that $\Phi_i(x_k) \le V_{i+1}(x_k)$.

Note that

$$V_1(x_k) - \Phi_0(x_k) = x_k^T Q x_k \ge 0$$
$$V_1(x_k) \ge \Phi_0(x_k)$$

Assume that $V_i(x_k) \ge \Phi_{i-1}(x_k)\ \ \forall x_k$. Since

$$\Phi_i(x_k) = x_k Q x_k + u_i^T R u_i + \Phi_{i-1}(x_{k+1})$$

$$V_{i+1}(x_k) = x_k Q x_k + u_i^T R u_i + V_i(x_{k+1}),$$

then

$$V_{i+1}(x_k) - \Phi_i(x_k) = V_i(x_{k+1}) - \Phi_{i-1}(x_{k+1}) \ge 0,$$

and therefore

$$\Phi_i(x_k) \le V_{i+1}(x_k)\,. \tag{6.19}$$

From Lemma 1 $V_i(x_k) \le \Phi_i(x_k)$ and therefore

$$\begin{aligned} &V_i(x_k) \le \Phi_i(x_k) \le V_{i+1}(x_k) \\ &V_i(x_k) \le V_{i+1}(x_k) \end{aligned}$$

hence proving that $\{V_i\}$ is a nondecreasing sequence bounded from above as shown in Lemma 2. Hence $V_i \to V^*$ as $i \to \infty$. □

Corollary: As the approximated cost function converges to the optimal value, *i.e.* $V_i \to V^*$, the control policy will converge to the optimal value, *i.e.* $u_i \to u^*$, and it will be equal to (6.7).

6.4 Neural Network Approximation

In this section, it is shown how to implement the HDP policy iterations algorithm with parametric structures like neural networks. It is well known that neural networks can be used to approximate smooth functions on prescribed compact sets.

Neural networks are natural for this application. Therefore, to solve (6.12), $V_i(x)$ and $u_i(x)$ are approximated by

$$\hat{V}_i(x) = \sum_{j=1}^{L} w_{vi}^j \phi_j(x) = W_{Vi}^T \boldsymbol{\phi}(x) \tag{6.20}$$

$$\hat{u}_i(x) = \sum_{j=1}^{L} w_{ui}^j \sigma_j(x) = W_{ui}^T \boldsymbol{\sigma}(x) \tag{6.21}$$

which are a neural networks with the activation functions $\sigma_j(x), \phi_j(x) \in C^1(\Omega)$, with $\sigma_j(0) = \phi_j(0) = 0$.

The neural network weights that approximate the cost function as in (6.12) are w_{vi}^{j}. L is the number of hidden-layer neurons. $\boldsymbol{\phi}(x) \equiv \left[\phi_1(x)\,\phi_2(x)\cdots\phi_L(x)\right]^T$ is the vector activation function, $W_{Vi} \equiv \left[w_{vi}^{1}\;w_{vi}^{2}\cdots w_{vi}^{L}\right]^T$ is the vector weight.

The weights are tuned to minimize the residual error between (6.20) and the target function defined in equation (6.22) in a least-squares sense over a set of points sampled from a compact set Ω .

$$\begin{aligned} d(\boldsymbol{\phi}(x_k), W_{Vi}^T, W_{ui}^T) &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1}) \\ &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \boldsymbol{\phi}(x_{k+1}) \end{aligned} \tag{6.22}$$

The least square problem can be defined as

$$\left(d(x) - \sum_{j=1}^{L} w_{vi+1}^{j} \sigma_j(x), u \right) = e_L(x)\,. \tag{6.23}$$

Note that in equation (6.23) the relation between the weight W_{Vi+1} and the target function is explicit.

To find the least squares solution, the method of weighted residuals is used [16]. The weights, W_{Vi+1}, are determined by projecting the residual error onto $de_L(x)/dW_{Vi+1}$ and setting the result to zero $\forall x \in \Omega$ using the inner product, i.e.

$$\left\langle \frac{de_L(x)}{dW_{Vi+1}}, e_L(x) \right\rangle = 0\,, \tag{6.24}$$

where $\langle \mathrm{f,g} \rangle = \int_{\Omega} f g^T dx$ is a Lebesgue integral. One has

$$0 = \int_{\Omega} \phi(x_k)(\phi^T(x_k) W_{Vi+1} - d^T(\phi(x_k), W_{Vi}^T, W_{ui}^T) dx_k \tag{6.25}$$

Therefore a unique solution for W_{Vi+1} exists and is computed as

$$W_{Vi+1} = \left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1} \int_{\Omega} \phi(x_k) d^T(\phi(x_k), W_{Vi}^T, W_{ui}^T) dx \tag{6.26}$$

Assumption 1. $\{\phi_j(x)\}^L$ is linearly independent on the compact set Ω.

From assumption 1, $\left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1}$ is full rank, which mean it is invertible and a unique solution of (6.26) exists.

Similarly, a neural network is used to find the parameters of the control policy $\hat{u}_i(x_k, W_{ui})$. The neural network weights that approximate the control policy function as in (6.11) are w_{ui}^j. L is the number of hidden-layer neurons. $\boldsymbol{\sigma}(x) \equiv [\sigma_1(x)\, \sigma_2(x) \cdots \sigma_L(x)]^T$ is the vector activation function, $W_{ui} \equiv \left[w_{ui}^1\, w_{ui}^2 \cdots w_{ui}^L \right]^T$ is the vector weight They are found by solving for

$$W_{ui} = \arg\min_{\alpha} \left. \begin{pmatrix} x_k^T Q x_k + \hat{u}^T(x_k, \alpha) R \hat{u}(x_k, \alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k, \alpha)) \end{pmatrix} \right|_{\Omega} \tag{6.27}$$

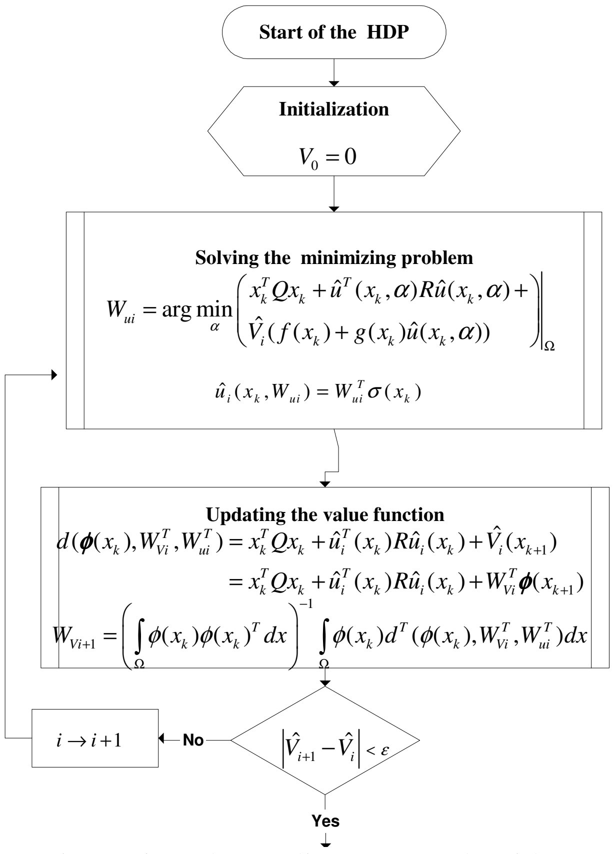
Note that the relation between the control weights W_{ui} in (6.27) is implicit. That is one can use a gradient steepest decent algorithm on a training set constructed from Ω to update the weights as

$$W_{ui(j+1)} = W_{ui(j)} - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_{i(j)}^T R \hat{u}_{i(j)} + \hat{V}_i(x_{k+1}))}{\partial W_{ui(j)}} \tag{6.28}$$

where α is a positive stepsize. Equation (6.28) can be written as

$$W_{ui}^{j+1} = W_{ui}^{j} -$$
$$\alpha\sigma(x_k)(2R\hat{u}_{i(j)} + g(x_k)^T \frac{\partial \phi(x_{k+1})}{\partial x_{k+1}} W_{Vi})^T$$

where $x_{k+1} = f(x_k) + g(x_k)\hat{u}(x_k, W_{ui}^j)$. The weights $W_{ui}^j \Rightarrow W_{ui}$ as $j \Rightarrow \infty$, which satisfies (6.27). Note that one can use different gradient methods like Newton's method and Levenberg-Marquardt method.



Start of the HDP
Initialization
$V_0 = 0$
Solving the minimizing problem
$W_{ui} = \arg\min_{\alpha} \left(\begin{matrix} x_k^T Q x_k + \hat{u}^T(x_k,\alpha) R \hat{u}(x_k,\alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k,\alpha)) \end{matrix} \right) \Big|_{\Omega}$
$\hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k)$
Updating the value function
$d(\phi(x_k), W_{Vi}^T, W_{ui}^T) = x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1})$
$= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \phi(x_{k+1})$
$W_{Vi+1} = \left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1} \int_{\Omega} \phi(x_k) d^T(\phi(x_k), W_{Vi}^T, W_{ui}^T) dx$
$\left|\hat{V}_{i+1} - \hat{V}_i\right| < \varepsilon$
No
$i \rightarrow i+1$
Yes

Figure 6.1. The nonlinear HDP algorithm.

In Figure 6.1, the flow chart of the HDP iteration is shown. Note that because of the neural network used to approximate the control policy the internal dynamics, *i.e.* $f(x_k)$, is not needed.

6.5 Discrete-time Nonlinear System Example

In this section, two examples are provided to demonstrate the solution of the DT HJB equation. The first example will be a linear dynamical system, which is a special case of the nonlinear case. The second example is for a DT nonlinear system. MATLAB simulation is used to implement some of the functions discussed in this section.

6.5.1 Linear system example

Consider the linear system

$$x_{k+1} = Ax_k + Bu \tag{6.29}$$

It is known that the solution of the optimal control problem for the linear system is quadratic in the state and given as

$$V^*(x_k) = x_k^T P x_k$$

where P is the solution of the ARE. This example is taken from [4], a linearized model of the short-period dynamics of an advanced (CCV-type) fighter aircraft. The state vector is

$$x = [\alpha \quad q \quad \gamma \quad \delta_e \quad \delta_f]^T$$

where the state components are, respectively, angle of attack, pitch rate, flight-path, elevator deflection and flaperon deflection. The control input are the elevator and the flaperon and given as

$$u = [\delta_{ec} \quad \delta_{fc}]^T$$

The plant model is discretized version of a continues model given in [4]

$$A=\begin{bmatrix} 1.0722 & 0.0954 & 0 & -0.0541 & -0.0153 \\ 4.1534 & 1.1175 & 0 & -0.8000 & -0.1010 \\ 0.1359 & 0.0071 & 1.0 & 0.0039 & 0.0097 \\ 0 & 0 & 0 & 0.1353 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix}$$

$$B=\begin{bmatrix} -0.0453 & -0.0175 \\ -1.0042 & -0.1131 \\ 0.0075 & 0.0134 \\ 0.8647 & 0 \\ 0 & 0.8647 \end{bmatrix}$$

Note that is a multi input unstable plant. The ARE solution for the given linear system is

$$P=\begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix} \quad (6.30)$$

and the optimal control $u_k^* = Lx_k$, where L is the optimal policy

$$L=\begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\ -0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix} \quad (6.31)$$

The ARE solution in (6.30) and the optimal control policy in (6.31) is given to be later compared with the results of the nonlinear HDP algorithm.

In the nonlinear HDP algorithm, the control is approximated as follows

$$\hat{u}_i = W_{ui}^T \sigma(x_k) \quad (6.32)$$

where W_u is the weights, and the $\sigma(x_k)$ is the basis. The basis is given as

$$\sigma^T(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$$

and the weights are

$$W_u^T = \begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix}$$

The choice of the control policy weights is done such that it will exactly approximate the control policy, as it is known it is linear on the state. The control weights should converge to

$$\begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix} = -\begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{21} & L_{22} & L_{23} & L_{24} & L_{25} \end{bmatrix}$$

The approximation of the value function is given as

$$\hat{V}_{i+1}(x_k, W_{Vi+1}) = W_{Vi+1}^T \phi(x_k) \tag{6.33}$$

where W_V is the weight of the neural network and $\phi(x_k)$ is the neuron vector

$$\phi^T(x) =$$
$$\begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_1x & x_2^2 & x_2x_3 & x_4x_2 & x_2x_5 & x_3^2 & x_3x_4 & x_3x_5 & x_4^2 & x_4x_5 & x_5^2 \end{bmatrix}$$

and the weights are given as

$$W_V^T = \begin{bmatrix} w_v^1 & w_v^2 & w_v^3 & w_v^4 & w_v^5 & w_v^6 & w_v^7 & w_v^8 & w_v^9 & w_v^{10} & w_v^{11} & w_v^{12} & w_v^{13} & w_v^{14} & w_v^{15} \end{bmatrix}$$

As it is known that the cost function will be quadratic in the states, the natural choice of the cost function weights as in equation (6.33). Note that the Kronecker product is used in (6.33) to approximate the cost function. In the simulation the weights of the value function are related to the P matrix given in (6.30) as follows

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} \end{bmatrix} = \begin{bmatrix} w_v^1 & 0.5w_v^2 & 0.5w_v^3 & 0.5w_v^4 & 0.5w_v^5 \\ 0.5w_v^2 & w_v^6 & 0.5w_v^7 & 0.5w_v^8 & 0.5w_v^9 \\ 0.5w_v^3 & 0.5w_v^7 & w_v^{10} & 0.5w_v^{11} & 0.5w_v^{12} \\ 0.5w_v^4 & 0.5w_v^8 & 0.5w_v^{11} & w_v^{13} & 0.5w_v^{14} \\ 0.5w_v^5 & 0.5w_v^9 & 0.5w_v^{12} & 0.5w_v^{14} & w_v^{15} \end{bmatrix}$$

The value function weights converge to

$W_V^T = [55.5411 \quad 15.2789 \quad 31.3032 \quad -9.3255 \quad -1.4536 \quad 2.3142 \quad 2.9234 \quad -1.6594 \quad -0.2430$

$24.8262 \quad -1.3076 \quad 0.0920 \quad 1.5388 \quad 0.1564 \quad 1.0240]$

The control weights converge to

$$W_u = \begin{bmatrix} 4.1068 & 0.7164 & 0.3756 & -0.5274 & -0.0707 \\ 0.6330 & 0.1005 & -0.1216 & -0.0653 & -0.0798 \end{bmatrix}$$

Note that the value function weights converge to the solution of the ARE (6.30), also the control policy weights converge to the optimal policy (6.31) as expected.

6.5.2 Nonlinear System Example

Consider the following affine in input nonlinear system

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{6.34}$$

where

$$f(x_k) = \begin{bmatrix} 0.2x_k(1)\exp(x_k^2(2)) \\ .3x_k^3(2) \end{bmatrix} \qquad g(x_k) = \begin{bmatrix} 0 \\ -.2 \end{bmatrix}$$

To approximation of the value function is given as

$$\hat{V}_{i+1}(x_k, W_{Vi+1}) = W_{Vi+1}^T \phi(x_k)$$

and the control input is approximated as

$$\hat{u}_i = W_{ui}^T \sigma(x_k)$$

The neuron vector of the Neural network that approximates the value function

$$\phi(x) = [x_1^2 \quad x_1x_2 \quad x_2^2 \quad x_1^4 \quad x_1^3x_2$$
$$x_1^2x_2^2 \quad x_1x_2^3 \quad x_2^4 \quad x_1^6 \quad x_1^5x_2 \quad x_1^4x_2^2$$
$$x_1^3x_2^3 \quad x_1^2x_2^4 \quad x_1x_2^5 \quad x_2^6]$$

and the weights are given as

$$W_V^T = \left[w_v^1 \quad w_v^2 \quad w_v^3 \quad w_v^4 \quad ..... \quad w_v^{15} \right].$$

The neuron vector of the neural network that approximates the control is given as

$$\sigma^T(x) = [x_1 \quad x_2 \quad x_1^3 \quad x_1^2x_2 \quad x_1\, x_2^2$$
$$x_2^3 \quad x_1^5 \quad x_1^4x_2 \quad x_1^3x_2^2 \quad x_1^2x_2^3$$
$$x_1\, x_2^4 \quad x_2^5]$$

and the policy weights are given as

$$W_u^T = \left[w_u^1 \quad w_u^2 \quad w_u^3 \quad w_u^4 \quad ..... \quad w_u^{12} \right]$$

The result of the algorithm is compared to the discrete-time State Dependent Riccati Equation (SDRE).

The training sets is $x_1 \in [-2,2], x_2 \in [-1,1]$. The value function weights converged to the following

$$W_V^T = [1.0382 \;\; 0 \;\; 1.0826 \;\; .0028 \;\; -0 \;\; -.053 \;\; 0 \; -.2792$$
$$-.0004 \;\; 0 \;\; -.0013 \;\; 0 \;\; .1549 \;\; 0 \;\; .3034]$$

and the control weights converged to

$$W_u^T = [\,0 \;\; -.0004 \;\; 0 \;\; 0 \;\; 0 \;\; .0651 \;\; 0 \;\; 0 \;\; 0 \;\; -.0003 \;\; 0 \;\; -.0046]$$

In the next figures, a comparison between the results obtained using the SDRE and the HDP based method is shown. Figure 6.2 and 6.3 show the states trajectories for the system for both methods.

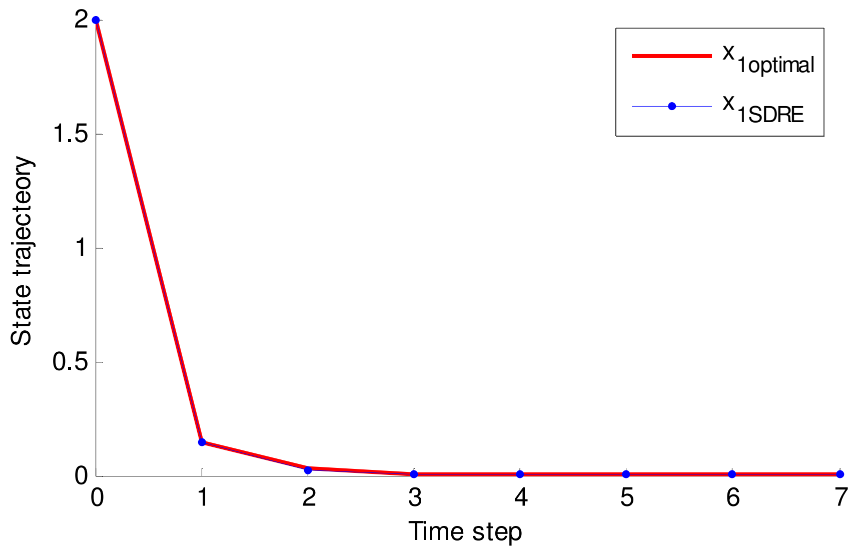


Figure 6.2. The states trajectories (x_1) for both methods

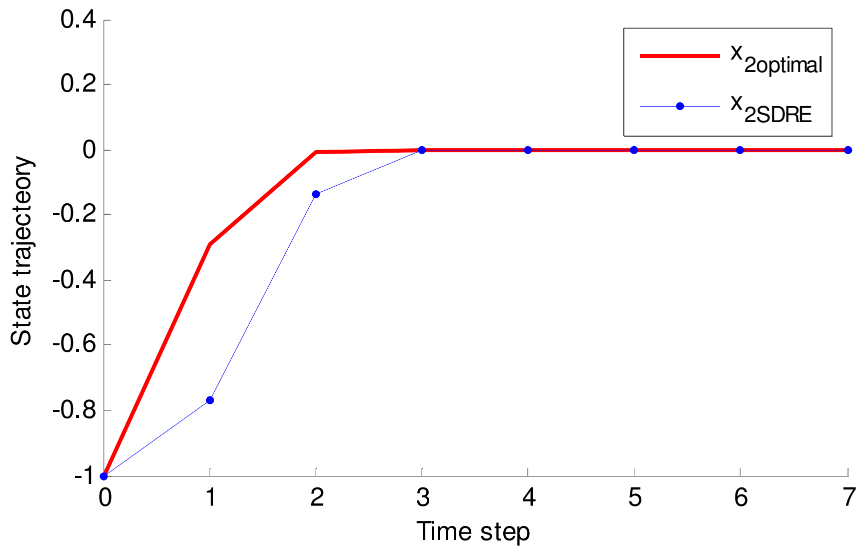


Figure 6.3. The states trajectories (x_2) for both methods

In Figure 6.4, the cost function of the SDRE solution and the cost function of the proposed algorithm in this chapter are compared. It is clear from the simulation that the cost function for the control policy derived from the HDP method is lower than the one obtained from the SDRE method.

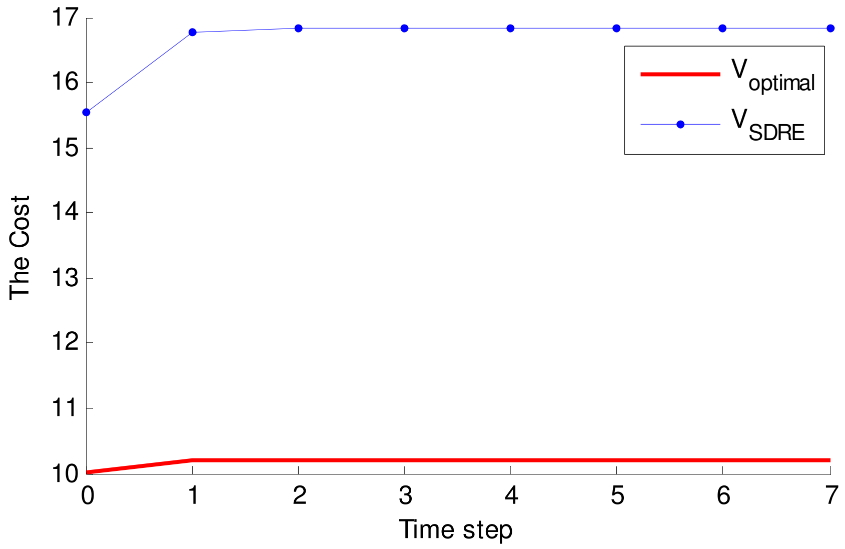


Figure 6.4. The cost function for both methods

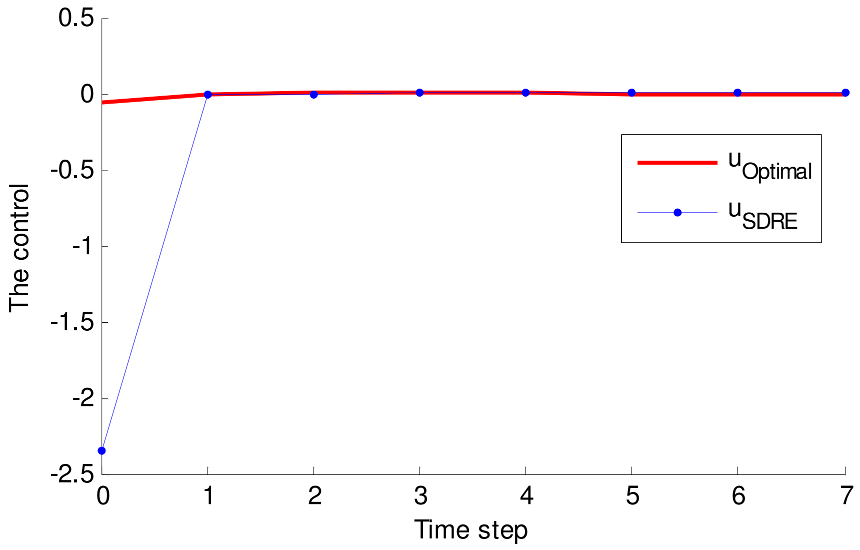


Figure 6.5. The control input for both methods

In figure 6.5, the control signals for both methods are shown.

6.5 Conclusion

A rigorous computationally effective algorithm to find the discrete-time nonlinear optimal state feedback control laws by solving the corresponding DT HJB equation. The algorithm proposed in this chapter namely nonlinear Heuristic Dynamic programming (HDP) is used to find the optimal controller. The main contribution in this chapter is the proof of convergence for the nonlinear HDP algorithm to the value function of DT HJB.

Neural networks are used as parametric structures to approximate the critics, *i.e.* $\hat{V}_i$, and the actors networks, *i.e.* $\hat{u}_i$. In the simulation part it is shown that the linear system critic network converges to the solution of the ARE, and the actor network converges to the optimal policy. In the nonlinear example, it is shown that the optimal controller obtained using the nonlinear HDP outperforms that derived using the discrete-time SDRE method.

It is important to mention that in the nonlinear HDP algorithm the full information of the system dynamic is not needed, as a result of the neural network used to approximate the control policy.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this dissertation, H_2 / H_∞ controllers have been designed using Approximate Dynamic Programming techniques, also known as Neuro Dynamic Programming. The Approximate Dynamic Programming techniques allow us to solve the dynamical programming problem forward-in-time. This allows us to perform on-line control policy tuning, that is direct adaptive optimal control.

This dissertation considers two classes of dynamical systems. The first one is the linear discrete-time system and the second one is the affine-in-input nonlinear discrete-time system.

Four on-line Approximate Dynamic Programming techniques are introduced in this dissertation to solve the discrete-time zero-sum game for linear systems, namely HDP, DHP, AHDP and ADDHP. The derivation of the policies and the proofs of convergence for the four algorithms are provided. The results can be though of as a direct H_∞ adaptive optimal control. The results presented herein are directly applicable in practice since they provide means to solve the H_∞ control problem, which is highly effective in feedback control systems design. It is interesting to see that when designing the H_∞ controller in forward time, one needs to provide an input signal that acts as a disturbance that is tuned to be the worst case disturbance in forward time. Among the proposed four algorithms, the most important is the ADHDP algorithm, where the dynamics of the linear system is not needed at all for tuning. This direct adaptive optimal control scheme will converge to the solution of the associated GARE, *i.e.* the optimal H_∞ control policy.

Note that if $\gamma \to \infty$ in the GARE, one obtains the special case Algebraic Riccati Equation (ARE). In other words, the four proposed algorithms for linear systems can solve the Linear Quadratic Regulator problem (LQR).

In the nonlinear affine-in-input dynamical systems, one ADP technique, namely HDP, is proposed to find the discrete-time nonlinear optimal state feedback control policy by solving the corresponding DT HJB equation. One main contribution in this research is the proof of convergence for the application of the HDP algorithm to nonlinear systems and it convergence to the solution of the DT HJB.

Neural networks are used as parametric structures to approximate the critics, *i.e.* $\hat{V}_i$, and the actors networks, *i.e.* $\hat{u}_i$ throughout the dissertation. It is important to mention that in the nonlinear HDP algorithm appearing in Chapter 6, only partial knowledge of the system's model is required as a result of using a neural network to approximate the action network, *i.e.* control policy.

The HDP algorithm mentioned in this dissertation can be extended to solve zero-sum games for nonlinear dynamical systems, *i.e.* discrete-time Hamilton-Jacobi-Isaacs equation (DT HJI). Additionally, the work in this dissertation could be a basis for a future work to extend Q-learning to nonlinear dynamical systems that are continuous in state and action space. This is advantageous as it allows solving the DT HJB/DT HJI without knowing the nonlinear model of the dynamical systems.

REFERENCES

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike elements that can solve difficult learning control problems,” *IEEE Trans. yst., Man, Cybern.,* vol. SMC-13, pp. 835–846, 1983.

[2] Abu-Khalaf, M., F. L. Lewis, “Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach,” Automatica, vol. 41, pp. 779 – 791, 2005

[3] Anton A. Stoorvogle, Arie J. T. M Weeren, “The discrete-time Riccati Eqaution Related to the H_∞ Control Problem,” IEEE Trans. Automat. Control, vol. 39, no. 3, pp. 686-691. March, 1994.

[4] B. Stevens, F. L. Lewis, Aircraft Control and Simulation, 2nd edition, John Wiley, New Jersey, 2003

[5] B. Widrow, N. Gupta, and S. Maitra, “Punish/reward: Learning with a critic in adaptive threshold systems,” *IEEE Trans. Syst., Man, ybern.,* vol. SMC-3, pp. 455–465, 1973.

[6] Başar, T., Greet Jan Olsder, Dynamic Noncooperative Game Theory, SIAN, 1999.

[7] Başar, T., P. Bernhard, H_∞ Optimal Control and Related Minimax Design Problems, Birkhäuser, 1995.

[8] C. Watkins, *Learning from Delayed Rewards*, Ph.D. Thesis, Cambridge University, Cambridge, England, 1989.

[9] D. Prokhorov and D. Wunsch, “Adaptive critic designs,” *IEEE Transactions on Neural Networks,* vol. 8, no. 5, September 1997.

[10] D.H.Jacobson,” On values and strategies for infinite-time linear quadratic games,” IEEE TAC, Vol 22, No 3, pp 490-491, 1977

[11] D.P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, MA, 1996.

[12] David Kleinman, “ Stabilizing a discrete, Constant, Linear System with Application to iterative Methods for Solving the Riccati Equation,” IEEE Trans. Automat. Control, pp. 252-254, June 1974.

[13] E. F. Mageirou, “Value and strategies for infinite time linear quadratic games,” IEEE TAC, Vol 21, No 4, pp 547-550, 1976.

[14] F. L. Lewis, *Applied Optimal Control and Estimation,* Prentice-Hall, New Jersey, 1992

[15] F. L. Lewis, Vassilis L. Syrmos, Optimal Control, Jhon Wiley and Sons, 1995

[16] Finlayson, B. A.. The Method of Weighted Residuals and Variational Principles. Academic Press, New York, 1972

[17] H.Jiang, J.F.Dorsey, Z.Qu, J.Bond, J.M.McCalley, ”Global robust adaptive control of power systems,” IEE Proc.-Gener. Transmit. Distib., Vol. 141, No5, September 1994.

[18] J. Hu and M. P. Wellman. “Multiagent reinforcement learning: Theoretical framework and an algorithm”, in the 15th Intl Conference on Machine Learning, pages 242--250, 1998.

[19] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, “Adaptive Dynamic Programming,” IEEE Trans. on Sys., Man. and Cyb., Vol. 32, No. 2, pp 140-153, 2002.

[20] J. Si, A. Barto, W. Powell, D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, IEEE Press, USA, 2004.

[21] John W.Brewer,”Kronecker Products and Matrix Calculus in System Theory,”IEEE Trans. on Circuit and System, Vol. CAS-25, No. 9, 1978.

[22] K.S. Narendra and F.L. Lewis, Special Issue on Neural Network feedback Control, Automatica, vol. 37, no. 8, Aug. 2001.

[23] M. Abu-Khalaf, F. L. Lewis, and J. Huang, “Hamilton-Jacobi-Isaacs formulation for constrained input nonlinear systems,” in 43rd IEEE Conference on Decision and Control, 2004, pp**.** 5034 - 5040 Vol.5, Bahamas, 2004.

[24] M. L. Littman, “Value-function reinforcement learning in Markov games,” *Journal of Cognitive Systems* Research, vol 2., pp. 55-66, 2002.

[25] P.J. Werbos, “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, 1992.

[26] P.J. Werbos, “Neural networks for control and system identification,” Proc. IEEE Conf. Decision and Control, Fla., 1989.

[27] P.J. Werbos., "A menu of designs for reinforcement learning over time," , Neural Networks for Control, pp. 67-95, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press, 1991.

[28] R. Howard, *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA, 1960.

[29] Stephan ten Hagen, Ben Krose, "Linear quadratic Regulation using Reinforcement Learning," Belgian_Dutch Conference on Mechanical Learning, pp. 39-46, 1998.

[30] Steven J. Bradtke, B. Erik Ydestie, Andrew G. Barto, "Adaptive linear quadratic control using policy iteration," Proceedings of the American Control Conference , pp. 3475-3476, Baltmore, Myrland, June, 1994.

[31] Tomas Landelius, Reinforcement Learning and Distributed Local Model Synthesis, PhD Chapter, Linkoping University, Sweden, 1997.

[32] W. H Kwon and S. Han, *Receding Horizon Control*, Springer-Verlag, London, 2005.

[33] Wang, Y., R. Zhou, C. Wen, "Robust load-frequency controller design for power systems", IEE Proc.-C, Vol. 140, No. I , 1993

[34] Wei Lin and Christopher I. Byrnes, "Dissipative, L_2-Gain and H_∞ Control for Discrete-Time Nonlinear System," Proceeding of The American Control conference, Maryland, June, 1994.

BIOGRAPHICAL INFORMATION

Asma Al-Tamimi was born in Amman, Jordan in 1976. She did her high school studies at Ajnadeen high school in Zarqa. She received her Bachelor's Degree in Electromechanicl Engineering from Al-Balqa University in Amman, Jordan in 1999. She then joined The University of Texas at Arlington from which she received the Master's of Science and PhD in Electrical Engineering in 2003 and 2007 respectively.