2017

On-Line Adaptive Dynamic Programming for Feedback Control

Xiangnan Zhong
University of Rhode Island, xzhong@ele.uri.edu

ON-LINE ADAPTIVE DYNAMIC PROGRAMMING FOR FEEDBACK CONTROL

BY

XIANGNAN ZHONG

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2017

DOCTOR OF PHILOSOPHY DISSERTATION

OF

XIANGNAN ZHONG

APPROVED:

Dissertation Committee:

Major Professor Haibo He

Yan Sun

Lisa DiPippo

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2017

ABSTRACT

Stability analysis and controller design are among the most important issues in feedback control problems. Usually, controller design for linear system can be obtained by solving the Riccati equation. However, when comes to the nonlinear control problem, Riccati equation becomes the well-known Hamilton-Jacobi-Bellman (HJB) equation which is difficult to tackle directly. Fortunately, adaptive dynamic programming (ADP) has been widely recognized as one of the "core methodologies" to achieve optimal control in stochastic process in a general case to achieve brain-like intelligent control. Extensive efforts and promising results have been achieved over the past decades. The achievements cover a large variety of problems, including system stability, convergence analysis, controller design, optimal control, state prediction, etc.

This dissertation investigates the on-line ADP techniques for the feedback control systems and provides novel methods to solve several existing problems in this field. Specifically, the improvement and the original contribution of this dissertation can be summarized from algorithms, architectures, and applications, respectively.

From the algorithms, an event-triggered ADP method is provided by sampling the efficient states rather than the entire system states generated during the learning process. The designed control law is only updated according to the sampled states to reduce the computation cost. In order to guarantee the sampled states are efficient, the theoretical analysis is provided to generate an event threshold to make sure the stability of the system during the event triggered learning process. It is said that only when the difference between the sampled and the current states is larger than the threshold, the system samples the state from the environment and updates the control law according to the sample states. This idea is further developed in the partially observable environment and the event threshold is designed only based on the observed feedback. A neural-network-based observer is designed to recover the internal states from the partially observable

ones. Both the observer and the control law are updated aperiodically based on the sampled system outputs. In this way, the computation and the transmission load can be significantly reduced. From the simulation results, it is shown that the event-triggered ADP method can achieve competitive performance at the same time.

From the architectures, a new framework, named "goal representation adaptive dynamic programming (GrADP)", is proposed and introduced in this dissertation. It is regarded as the foundation of building intelligent systems through internal reward learning, goal representation and state-action association. Unlike the traditional ADP design with an action network and a critic network, this new approach integrates an additional goal network, such that to build a general internal reinforcement signal. Unlike the traditional fixed or predefined reinforcement signal, this new design can adaptively update the internal reinforcement representation over time and thus facilitate the system's learning and optimization to accomplish the ultimate goals. This dissertation for the first time provides the theoretical foundation of the GrADP design. It is shown that the designed internal reinforcement signal can give the agent more information by considering more distance lookahead, and therefore, this signal is more efficient.

From the applications, this dissertation designs the ADP method for a class of Markov jump systems (MJSs) to find the optimal control law even though the system state keeps jumping among several subsystems. This dissertation also shows that the control law obtained from the learning process can quickly converge to the optimal solutions which verifies the effectiveness of the proposed method.

ACKNOWLEDGMENTS

Five years on my Ph.D. study in the University of Rhode Island (URI) has been a period of intense learning for me, not only in the scientific arena, but also on a personal level. Writing this dissertation has had a big impact on me. I would like to acknowledge the people who have supported and helped me so much throughout this period.

I would first like express my deepest gratitude to my advisor, Prof. Haibo He, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. He continually and convincingly conveyed a spirit of adventure in regard to research, and an excitement in regard to teaching. Without his guidance and persistent help, this dissertation would not have been possible.

I would also like to sincerely thank Prof. Yan Sun, Prof. Richard Vaccaro, Prof. Lisa DiPippo, and Prof. P. V. August for serving as my dissertation defense committee. They gave me many precious comments and guidance in this process which provided me a different view of my research work and myself. Also extremely thankful to Prof. Huaguang Zhang, Prof. Zhanshan Wang, and Dr. Danil Prokhorov for close collaborations and suggestions all the way along this research direction.

I am grateful to the Computational Intelligence and Self-Adaptive System (CISA) group in URI during my Ph.D. period: Dr. Ding Wang, Dr. Chaoxu Mu, Dr. Siyao Fu, Dr. Yufei Tang, Dr. Bo Tang, Jun Yan, Lu Dong and Jing Wang for their inputs, valuable discussions and accessibility of my research work. Also thanks all the faculty members, staff, visiting scholars, and graduate students in the Department of Electrical, Computer, and Biomedical Engineering for their friendship and support in these five years.

Grateful acknowledgement is expressed to my beloved parents, Liangcai Zhong and Shu Chen, for being my inspirations and motivation to always do the best in everything I do; for always showing the world and making me feel how much they are proud of me; for their teachings and guidance which made me become as to what I am now.

Also many thanks to my grandparents: Futang Chen and Xiangyu Shu, Bingwen Zhang and Guifang Yu, who always care about me, my uncle: Mo Chen, my aunt: Haiping Xu, my cousin: Chen Chen, and all my family for their unconditional supports and patience. Thank you for being so understanding and supportive.

My final words go equally to my husband, Zhen Ni, for his inestimable moral support and his infinite warmth and tenderness. Thank you for always being my side whenever and wherever I need. Thank you for completing my life.

This work was supported in part by the National Science Foundation under Grant ECCS 1053717, Grant CMMI 1526835, Army Research Office under Grant W911NF-12-1-0378, NSF-DFG Collaborative Research on Autonomous Learning (a supplement grant to CNS 1117314), and in part by the National Aeronautics and Space Administration under Grant NNX15AK54A.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Motivation and Inspirations

When we first think about the nature of learning, we probably start with the idea that we learn by interacting with our environment. Actually, we learn from our childhood. When an infant plays, waves its arms, or looks about, there is no explicit teacher. However, it does have a direct connection with its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals. Throughout our lives, such interactions are undoubtedly a major source of knowledge about our environment and ourselves. Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence.

Brain intelligence and animal intelligence are very important biological inspiration to develop truly self-adaptive systems to such a level of intelligence in certain perspectives. With the recent developments of brain research and modern technologies, scientists and engineers will hopefully find efficient ways to build complex systems that are highly adaptive, robust, and fault tolerant to uncertain and unstructured environment. However, although many important fundamental research as well as critical engineering applications have been successfully developed, there is still a long way to go to achieve truly brain-like general-purpose intelligent machines. In other words, understanding of brain intelligence and developing self-adaptive systems to potentially mimic certain level of intelligence is still one of the greatest unsolved scientific challenges [1, 2]. One of the key fundamental challenges is how to design intelligent systems to be able to "learn to optimize" and "learn to predict" over time to achieve goals. In this dissertation, an online learning system with an emphasis on the latest data-driven adaptive dynamic programming (ADP) architecture is developed for the feedback control prob-

lems to improve the control performance.

In the general control problems, learning controller design for nonlinear systems is a difficult and challenging topic because it often requires solving the Hamilton-Jacobi-bellman (HJB) equation rather than the Riccati equation. Fortunately, ADP technique gives us an opportunity to obtain the approximate solutions of the HJB equation [3, 4]. In recent years, ADP method has attracted significantly increasing attention and it has been widely recognized as one of the "core methodologies" to achieve optimal control for intelligent systems in a general case [1, 5, 6]. Extensive efforts have been dedicated to developing ADP method from both theoretical researches and real-world applications [7, 8]. Although promising results have been achieved, there still exist several major challenges when we design the ADP method for the control problems. For instance, the high computation during the learning process, how we can provide an evaluative feedback in terms of the reinforcement signal, the requirement of the full system states, and robustness. In this dissertation, I provide the detailed solutions for several certain major challenges in this field.

1.2 Significance of the Research

Generally, every living organism interacts with its environment and uses those interactions to improve its own actions in order to survive and increase. In this process, the living organism modifies actions based on the interactions with the environment [9]. We call this process as reinforcement learning or ADP. There are many types of learning in the computational intelligent society, including supervised learning and unsupervised learning, etc. Reinforcement learning and ADP refer to an actor or agent that interacts with its environment and modifies its actions, or control policies, based on stimuli received in response to its actions. This is based on evaluative information from the environment and could be called action-based learning. ADP implies a cause and effect relationship between actions and rewards or punishments. It implied goal directed be-

havior at least insofar as the agent has an understanding of reward versus lack of reward or punishment. The rewards or punishments define the goal of the task. Optimal actions may be based on minimum fuel, minimum energy, minimum risk, maximum reward, and so on.

Motivated by the human-level intelligence, this idea is introduced into the machine training process [10]. The machines interact with the environment to modify the action in order to achieve the goals. When we train the machines based on the reinforcement learning or ADP scheme, we want the machines become intelligent and act as human beings. During the learning process, we tell the machines which is good or which is bad. This means we need to let the machines know the effect of their actions or situations. In this way, after trial-and-error learning, the machines can achieve human-level intelligence.

1.3 Research Objective

The objective of this dissertation is focused on designing on-line intelligent learning systems that are capable to learn to optimize the decision-making process in an unknown environment for the feedback control problems. It is very important to analyze different types of challenges and develop the detailed solutions for them onto the various critical engineering applications:

- ADP is usually relying on the periodic transmission of the sampled data and computation of the control law. This periodic data abstraction is advantageous from the design standpoint. It permits real-time system engineers and control system engineers to pursue their design objectives in relative isolation from each other. However, such algorithm may bring huge number of the transmitted data and subsequently tremendous computation [11], [12]. This is clearly a disadvantage when the computation bandwidth or sensor power resources are constrained [13], [14], [15]. In this dissertation, an event-triggered ADP control method has been pro-

vided for its capability of computation efficiency. In the proposed event-triggered control algorithm, the controller is only updated when an event is triggered, and thus the computation is significantly saved.

- Usually, when we develop the ADP method to solve the Bellman equation, it requires careful evaluate the benefits and cost not only the immediate action but also the choices we may have in the future [16, 17, 18]. In order to achieve this goal, the designed method needs a complete set of system information/states to achieve the online optimal decision-making. However, in many practical situations, the measured input/output data can only represent part of the system internal information. In this dissertation, I further develop the event triggered ADP method in the partially observable environment. Only the reduced information is applied to design the adaptive observer and neural-network-based controller. The goal is to obtain the competitive results with limited information.

- Since the goal representation design could be able to learn proper internal reward through the interaction with the environment adaptively, rather than a fixed reward formula all the way over time, I further integrate the goal representation technique into the GDHP design and propose an advanced method which is goal representation GDHP (Gr-GDHP). In the proposed method, the goal network not only provides the internal reinforcement signal, but also generates its partial derivatives with respect to the system variables and control action. The objective of this research is to improve the control performance comparing with the traditional ADP methods and other goal representation ADP structure.

- This dissertation also develops an adaptive learning method for a class of unknown nonlinear Markov jump systems based on ADP technique. Unlike the traditional method, such as the linear matrix inequality (LMI) technique, the proposed in-

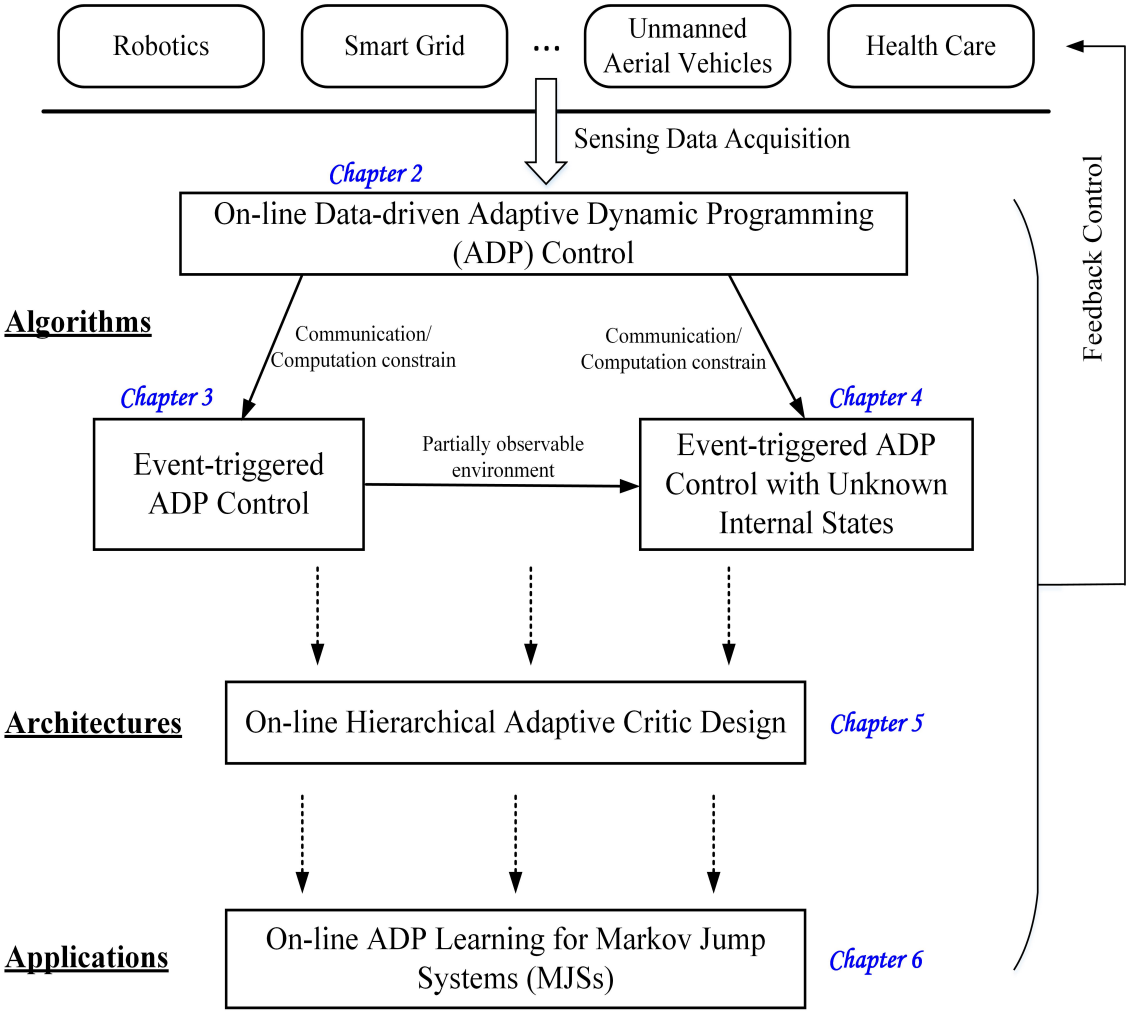

Figure 1. Dissertation Organization.

telligent approach includes the adaptive and learning capability of the system dynamics, indicating that this approach can still find the near optimal controller even if the system parameters are changed.

1.4 Dissertation Organization

The rest of this dissertation is shown in Figure 1. It includes three parts of the on-line ADP design for feedback control: Algorithms, Architectures, and Applications. We focus on the problems exist in the ADP control field and develop novel algorithms and architectures to improve the control performance or save the computation resources.

Then, the developed methods are demonstrated in a kind of Markov Jump Systems to show the performance.

Chapter 2 provides the background of my research and literature review in current community. It further provides the ADP development in feedback control and the advantages of on-line learning.

Chapter 3 focuses on an event-triggered ADP control method. The triggered threshold is designed to guarantee the stability of the developed method. Competitive results are obtained with reduced sampling states.

Chapter 4 further investigates the event-triggered ADP scheme and develops it in the partially observable environment. An observer is designed based on the neural network techniques to recover the entire states from the system input/output data. Both the observer and the controller are only updated when a specific event is triggered.

Chapter 5 studies the ADP control from the architecture side and presents a new internal goal representation design based on the traditional adaptive critic design. An additional goal network is integrated into the structure to facilitate the performance. Then, a goal representation HDP method is developed with the explicit on-line learning process.

Chapter 6 develops the ADP method for a class of Markov Jump Systems which have the powerful modeling capability for power systems, network control systems, manufacturing systems among others.

Chapter 7 concludes the dissertation and also discusses the future directions of this on-line ADP method for feedback control.

List of References

[1] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200–212, 2009.

[2] P. J. Werbos, "Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design," *IEEE Int. Symposium on Approximate Dynamic Programming*

and Reinforcement Learning (ADPRL07), pp. 209–216, 2007.

[3] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.

[4] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, London, UK, 1999.

[5] P. J. Werbos, "ADP: The key direction for future research in intelligent control and understanding brain intelligence," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 898–900, 2008.

[6] X. Zhong, H. He, H. Zhang, and Z. Wang, "A neural network based online learning and control approach for markov jump systems," *Neurocomputing*, vol. 149, pp. 116–123, 2015.

[7] X. Zhong, H. He, and D. V. Prokhorov, "Robust controller design of continuous-time nonlinear system using neural network," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013.

[8] X. Zhong, Z. Ni, Y. Tang, and H. He, "Data-driven partially observable dynamic processes using adaptive dynamic programming," in *Proc. IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2014, pp. 1–8.

[9] F. L. Lewis, and D. Vrabie., "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Sys. Mag.*, vol. 9, no. 3, pp. 32–50, 2009.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

[11] W. Heemels, M. Donkers, and A. Teel, "Periodic event-triggered control for linear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 847–861, 2013.

[12] P. Tallapragada and N. Chopra, "On event triggered tracking for nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2343–2348, 2013.

[13] M. Lemmon, "Event-triggered feedback in control, estimation, and optimization," in *Networked Control Systems*. Springer, 2010, pp. 293–358.

[14] E. Garcia and P. J. Antsaklis, "Model-based event-triggered control with time-varying network delays," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 1650–1655.

[15] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3270–3285.

[16] X. Zhong, H. He, H. Zhang, and Z. Wang, "Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming," *Neural Networks and Learning Systems, IEEE Transactons on*, vol. 25, no. 12, pp. 2141–2155, 2015.

[17] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, "Model-free dual heuristic dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1834–1839, 2015.

[18] X. Zhong, Z. Ni, and H. He, "A theoretical foundation of goal representation heuristic dynamic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, 2015, in press.

CHAPTER 2

On-line Data-driven Adaptive Dynamic Programming (ADP) Control

2.1 Feedback Control

The study of the means of developing control systems for human engineered systems to endow them with guaranteed performance and safety is called as feedback control theory. Included are control systems for aircraft, ships, race cars, robot systems, industrial process, building temperature and climate regulation systems, and many more.

Feedback control systems require the designed algorithms and analysis to yield guaranteed provable performance and safety margins. Feedback controller without performance, stability, and robustness guarantees will not be accepted by industry. Providing such guarantees needs to use the framework and tools through mathematics. Therefore, we should like to capture the ideas about reinforcement learning or adaptive dynamic programming in some sort of mathematical formulation. One such formulation is the framework of Markov decision processes (MDP). MDP have been extensively used to study and embody reinforcement learning systems. In MDP, the state spaces and action spaces are generally discrete (i.e. state and action take on only certain allowed discrete values). However, human engineered systems develop and move through time and generally have states and actions that reside in continuous spaces. A broad class of engineered systems can be effectively described by ordinary differential equations, since these describe the development of a system through time based on its current status as well as any inputs received, such as commands, disturbances, and so on.

This dissertation is to show the usefulness of ADP techniques for feedback control of human engineered systems. ADP techniques have been developed by computational intelligence community. Therefore, this requires bringing together ideas from two communities-control systems engineering and computational intelligence. Since ADP involves modifying the control policy based on responses from the environment, one

has the initial feeling that it should be closely related to adaptive control, a family of successful control techniques held in high regard in control systems community.

2.2 Background of Adaptive Dynamic Programming

The fundamental idea of intelligence is to learn from the interaction. Over the past decades, many researchers have explored computational approaches to learning from active interaction with the environment. Imagine that we try to train a mouse to find its way out of a specific maze. When the mouse succeeds in the test, we give it a cheese, which is its favorite food as a reward. Otherwise, we give it nothing. After trial and error learning, we find that the mouse can quickly find its way out of this maze no matter where it is. In this process, even though the mouse does not know the exact meaning of maze navigation, it knows a cheese can be given when certain position is achieved. This process can be called as learning. Nowadays, we apply this idea when we train the machines to copy intelligent human behaviors or to achieve some certain goals. In the training process, we tell the machines which is good and which is bad. This means we need to let the machines know the effect of their actions and situations. One of the popular methods we explore to achieve this goal is ADP.

In an ADP problem, the learner or decision maker, like the mouse in the first example, is called the agent. The thing it interacts with, like the maze, is called the environment. At one time step, the agent selects actions $u(t)$ to the environment. The environment responds to these actions and produces new situations $x(t)$ to the agent. At the same time, the environment also gives rise to rewards or punishments to the agent. In the ADP control problems, we unite the reward or punishment signals as reinforcement signals or utility functions which can be described as $r(t)$ or $U(x(t), u(t))$. The objective of the agent is to choose a control sequence, so that the total expected future reinforcement signals can be minimized in the long run. Therefore, by defining the interaction between an agent and its environment in terms of states, actions, and

reinforcement signals, ADP approach helps the agent to make the optimal decision.

Generally speaking, ADP can be categorized into three typical schemes: heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP), and globalized dual heuristic dynamic programming (GDHP) [1]. Specifically, the HDP design adopts a critic network to estimate the performance index function or total cost-to-go $J(t)$ in the Bellman equation. This idea is essentially similar to the temporal-difference (TD) method discussed in [2]. The detailed backpropagation rules for both the critic and the action networks of the direct HDP design were proposed in [3]. The authors further presented the stability of this method in [4, 5] where they demonstrated the theoretical analysis that the estimation errors of neural network weights were uniformly ultimately bounded (UUB) by Lyapunov stability construct. In [6], the authors provided the convergence of the value-iteration-based HDP algorithm for general discrete-time nonlinear systems. In [7], the policy iteration using adaptive dynamic programming for discrete-time nonlinear systems was also discussed and demonstrated. Many other publications on the theoretical analysis for the HDP approach were also provided and demonstrated [8, 9, 10, 11]. To overcome the limitations of scalability, Werbos went beyond a critic network approximating just the performance index function and further proposed two new methods: DHP and GDHP [12], followed by many improvements and demonstrations of such methods [13, 14]. The core idea of DHP is to use a critic network to approximate the derivatives of the performance index function with respect to the state variables. While GDHP takes advantage of both HDP and DHP by using a critic network to approximate both the value function and its derivatives [15]. In [16, 17], the authors built the GDHP controller for a class of unknown discrete-time nonlinear systems and compared the performance among the HDP, the DHP and the GDHP controllers. Various versions of ADP have been developed based on these three typical schemes, such as the action-dependent (AD) version and model-dependent version.

Recently, a series of the goal representation heuristic dynamic programming (GrHDP) was proposed to improve the online learning of the ADP design in [18, 19]. Unlike the typical ADP schemes (i.e., one critic and one action networks), the authors integrated an additional network, namely the reference/goal network, to obtain an internal reinforcement signal to facilitate the optimal learning and control. This architecture has been applied to various realistic and complex control problems. In [20], the GrHDP design was applied on the tracking control problem and further on the real-time simulink/virtual reality platform. In addition, multiple reference/goal networks design, namely the hierarchical HDP design, was proposed and verified with promising control performance [21]. More recently, the GrHDP controller was further tested on the maze navigation problems [22, 23]. The goal representation dual heuristic dynamic programming (GrDHP) was also proposed and the partial derivatives of utility function can be directly obtained through a neural network rather than engineering designs [24, 25]. In the society, many researchers also followed this trend and applied the three-network HDP framework from different aspects. The improvement from the simulation results were provided and discussed in [26, 27].

2.3 On-line Learning

Usually, in the ADP problems, we find a control law $u(t)$ to minimize the discounted total expected future reinforcement signals as

$$J^*(t) = \min_{u(t)}\{r(t) + \alpha J^*(t+1)\} \tag{1}$$

where $r(t)$ is the reinforcement signal, $0 < \alpha < 1$ is the discount factor, and $J(t)$ is the discounted total expected future reinforcement signals which is called the performance index.

Neural network techniques are applied to solve the problem. There are usually two neural networks in the traditional adaptive critic design. Figure.2 is the schematic diagram of typical HDP structure. An action network is used to provide the control action

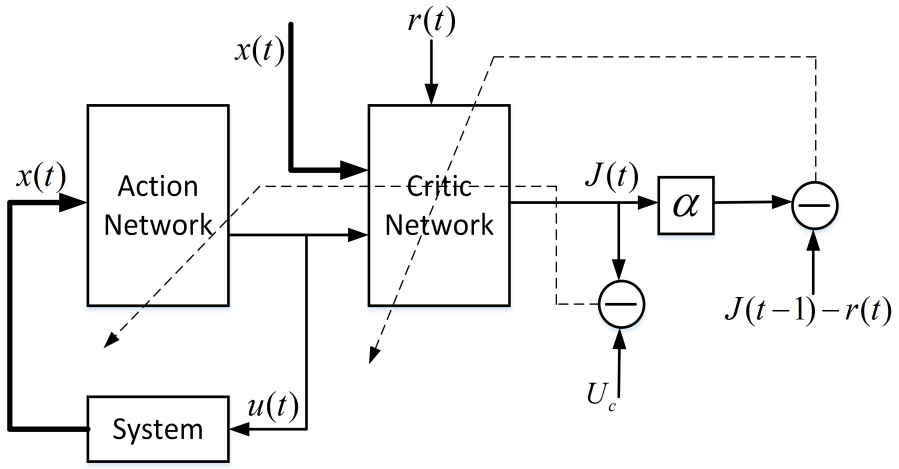

Figure 2. The schematic diagram of a typical HDP structure.

to the system, and a critic network is used to evaluate the control performance over time. For example, the action network generates the control action $u(t)$ based on the observation of the system variables $x(t)$. The critic network evaluates the performance of this control policy based on the reinforcement signal feedback $r(t)$ from the environment. Meanwhile, the performance index $J(t)$ will be approximated by the critic network. As presented in Figure. 2, the objective function of critic network will be provided by the temporal difference between current step and previous step in Bellman's equation and the performance index is applied to adjust the weights in action network.

In this dissertation, the learning process of the neural networks is conducted on-line. This means we train the weight matrices in the order of the critic network, and the action network. After the critic network weights are learned, we fix them thereafter and start to train the weights in the action network. It can be observed that this process does not require the knowledge of system functions. This is important, as the exact information of system functions is difficult to obtain for general nonlinear systems. In addition, this algorithm includes the adaptive capability, indicating that even if the parameters of the system change, the optimal controller can still be determined automatically. The on-line learning process also has the benefit that when the system functions are changed

or some disturbance happens, the method can still find the optimal control law through the on-line learning scheme.

2.4 Related Work

Taking advantage of solving the problem without the knowledge of system functions, ADP has attracted significantly increasing attention from both theoretical research and real-world applications [15, 9, 28, 29, 30, 31] over the past decades by attempting to obtain the approximate solutions of the Hamilton-Jacobi-Bellman (HJB) equation. It has been widely recognized that ADP could be one of the "core methodologies" to achieve optimal control in stochastic process in a general case to achieve brain-like intelligent control [32, 33]. Extensive efforts and promising results have been achieved over the past decades. Here we highlight a few important ADP research from the theoretical perspective that are closely related to the research presented in this dissertation. Interested readers can refer to the two important handbooks on ADP for many other successful architectures, algorithms, models, and challenging engineering applications [34, 35]. For instance, Al-Tamimi *et al* provided the convergence of the value-iteration-based ADP algorithm for general discrete-time nonlinear systems in [6]. In another paper [36], Abu-Khalaf *et al* introduced a new generalized non-quadratic function into the performance index to evaluate the performance of systems with constrained control inputs. This idea of bounded control was also related to the work presented in [37] and [38], in which the authors focused on the optimal control problem for nonlinear systems with unknown perturbation. The optimal control problem with constrained input was also solved in [39], [40] based on the ADP algorithm. In [41], the feasibility of using the solution of the optimal control problem to solve the robust control problem was provided for nonlinear system with matched uncertainties. The author further developed the results into the nonlinear system with unmatched uncertainties in [42]. Zhong *et al* used online neural network learning method to train the control law for robust control prob-

lem [11]. In [43], Wei and Liu proposed a new "θ-ADP" iterative algorithm to solve the optimal control problem of infinite horizon discrete-time nonlinear systems by finding a lower bound for parameter θ to assure the convergence of this algorithm. Motivated by these results, Liu and Wei further developed the convergence conditions for the situation that the iterative control policy and iterative performance index cannot be accurately obtained [44]. In a related work [45], an optimal scheme for unknown nonaffine nonlinear discrete-time systems by using cost function with discount factor was developed and analyzed. For the affine nonlinear system, the optimal control by using general value iteration was provided in [46]. A new iterative ADP method was proposed to solve a class of nonlinear zero-sum differential games in [47], [48] for continuous-time and discrete-time situation, respectively. Wei *et al* developed a numerical iterative ADP algorithm with convergence analysis in [49]. Moreover, the adaptive critic techniques were also applied for engine torque and air-fuel ratio control [50] and tracking control [51]. From the architecture point of view, He *et al* integrated a reference network into the classic ADP structure to adaptively establish an internal goal representation to facilitate the optimal learning and control [18], [19]. Then, they used this new structure to solve tracking control problem and obtain the effective performance [20], [52]. This GrHDP approach was also applied on the maze navigation example and compared with many other reinforcement learning approaches in [53], [54]. The hierarchical GrHDP architecture was further studied in [55], [21]. Furthermore, due to the problem of (partially) unknown system dynamics, many researchers have developed different approaches to handle such partially observable situations [56], [57]. In a similar situation, Zhang *et al* employed a model network based on recurrent neural network structure to reconstruct the unknown system dynamics for nonlinear systems [58].

When the ADP method is introduced into the feedback control system, it is important to notice the stability and the convergence of the proposed methods. The proofs

of convergence for the ADP designs have been studied for years. In [6], the authors proved the convergence of the value iterations using the HDP design and sought the optimal solution of the discrete-time HJB equation. This idea was further extended to prove the stability of the DHP and the GDHP in [45] and [16], respectively. In [44], the convergence conditions were developed for the situation that the iterative control policy and the iterative performance index cannot be accurately obtained. In [43], new ADP control designs were proposed under the specific situations and the corresponding convergence analysis was provided to show the accuracy of the proposed method. The convergence analysis of HDP method was discussed for stochastic system in [59]. The core idea in the above results was the stability of the iterative ADP algorithm. It was shown that after infinite iteration steps, the value function could converge to the optimal value and the corresponding control law could stabilize the system. Then neural network techniques were applied to approximate the optimal value function and the controller. Particularly, a pre-trained model network was required in these methods. For continuous state and action spaces, theoretical guarantee of convergence is more challenging. Several analytical frameworks were developed for ADP control designs under multiple system formulations in [60], [61]. In these papers, the control system was described as $\dot{x} = f(x) + g(x)u(x)$ and the knowledge of $g(x)$ was required for deriving the optimal controller $u(x)$.

Moreover, stability analysis was also developed based on the Lyapunov stability approach in [4], [5], [58]. A positive definite Lyapunov function was designed and the first difference of this function was derived as negative definite. Hence, the learning weights of neural networks were guaranteed to converge to the optimal values under certain conditions. In [3], the Robbins-Monro algorithms was used to find the optimal weights for each neural networks during the learning process. In the reinforcement learning field, researchers have demonstrated that the difference between the estimated

value function and the expected optimal value function can be bounded in an arbitrary small range after infinite iteration [62], [63].

List of References

[1] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

[3] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 264–276, 2001.

[4] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, 2012.

[5] L. Yang, J. Si, K. S. Tsakalis, and A. A. Rodriguez, "Direct heuristic dynamic programming for nonlinear tracking conrol with filtered tracking error," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 39, no. 6, pp. 1617–1622, 2009.

[6] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 942–949, 2008.

[7] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," in press, 2013.

[8] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.

[9] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

[10] X. Zhong, Z. Ni, Y. Tang, and H. He, "Data-driven partially observable dynamic processes using adaptive dynamic programming," in *Proc. IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2014, pp. 1–8.

[11] X. Zhong, H. He, and D. V. Prokhorov, "Robust controller design of continuous-time nonlinear system using neural network," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013.

[12] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System modeling and optimization*. Springer, 1982, pp. 762–770.

[13] F. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.

[14] J. Si, A. G. Barto, W. B. Powell, D. C. Wunsch, *et al.*, *Handbook of learning and approximate dynamic programming*. IEEE Press Los Alamitos, 2004.

[15] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 8, no. 5, pp. 997–1007, 1997.

[16] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 3, pp. 628–634, 2012.

[17] D. Liu and D. Wang, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Optimal Control of Unkonwn Nonlinear Discrete-Time Systems Using the Iterative Globalized Dual Heuristic Programming Algorithm, pp. 52–74.

[18] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon78.html#HeNF12

[19] H. He, *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.

[20] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 913–928, 2013.

[21] Z. Ni, H. He, D. Zhao, and D. V. Prokhorov, "Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–8.

[22] Z. Ni, H. He, J. Wen, and X. Xu, "Goal representation heuristic dynamic programming on maze navigation," *Neural Networks, IEEE Transactions on*, vol. 24, pp. 2038–2050, Dec. 2013.

[23] Z. Ni and H. He, "Heuristic dynamic programming with internal goal representation," *Soft Computing*, vol. 17, pp. 2101–2108, 2013.

[24] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "GrDHP: A General Utility Function Representation for Dual Heuristic Dynamic Programming," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 614–627, Mar. 2015.

[25] Z. Ni, Y. Tang, H. He, and J. Wen, "Multi-machine power system control based on dual heuristic dynamic programming," in *Proc. of 2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, Orlando, FL, pp. 1–7.

[26] X. Luo, J. Si, and Y. Zhou, "An integrated design for intensified direct heuristic dynamic programming," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*. IEEE, 2013, pp. 183–190.

[27] J. Chen and Z. Li, "A novel adaptive tropism reward ADHDP method with robust property," in *Advances in Brain Inspired Cognitive Systems*. Springer, 2013, pp. 288–295.

[28] D. Liu, Y. Zhang, and H. G. Zhang, "A self-learning call admission control scheme for CDMA cellular networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1219–1228, 2005.

[29] J. Fu, H. He, and X. Zhou, "Adaptive learning and control for mimo system based on adaptive dynamic programming," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.

[30] H. G. Zhang, Q. L. Wei, and Y. H. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 937–942, 2008.

[31] K. G. Vamvoudakis and F. L. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2010.02.018

[32] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200–212, 2009.

[33] P. J. Werbos, "Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design," *IEEE Int. Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL07)*, pp. 209–216, 2007.

[34] J. Si, A. G. Barto, W. B. Powell, and D. W. II, Eds., *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE, 2004.

[35] F. L. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE, 2012.

[36] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/automatica/automatica41.html#Abu-KhalafL05

[37] D. M. Adhyaru, I. N. Kar, and M. Gopal, "Bounded robust control of nonlinear systems using neural network-based HJB solution," *Neural Computing and Applications*, vol. 20, no. 1, pp. 91–103, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/nca/nca20.html#AdhyaruKG11

[38] D. M. Adhyaru, I. N. Kar, and M. Gopal, "Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution," *IET Control Theory and Applications*, vol. 3, no. 1, pp. 1183–1195, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/nca/nca20.html#AdhyaruKG11

[39] D. Liu, D. Wang, and X. Yang, "An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs," *Information Sciences*, vol. 220, pp. 331–342, Jan. 2013.

[40] D. Wang, D. Liu, D. Zhao, Y. Huang, and D. Zhang, "A neural-network-based iterative GDHP approach for solving a class of nonlinear optimal control problems with control constraints," *Neural Computing and Applications*, vol. 22, no. 2, pp. 219–227, 2013.

[41] F. Lin, R. D. Brandt, and J. Sun, "Robust control of nonlinear systems: Compensating for uncertainty," *International Journal of Control*, vol. 56, no. 6, pp. 1453–1459, 1992.

[42] F. Lin, "An optimal control approach to robust control design," *International Journal of control*, vol. 73, no. 3, pp. 177–186, 2000.

[43] Q. Wei and D. Liu, "Adaptive dynamic programming with stable value iteration algorithm for discrete-time nonlinear systems," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2012, pp. 1–6.

[44] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 43, no. 2, 2013.

[45] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[46] H. Li and D. Liu, "Optimal control for discrete-time affine non-linear systems using general value iteration," *IET Control Theory & Applications*, vol. 6, no. 18, pp. 2725–2736, 2012.

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

[48] D. Liu, H. Li, and D. Wang, "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm," *Neurocomputing*, vol. 110, pp. 92–100, 2013.

[49] Q. Wei and D. Liu, "Numerical adaptive learning control scheme for discrete-time non-linear systems," *IET Control Theory & Applications*, vol. 7, no. 11, pp. 1472–1486, 2013.

[50] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, "Adaptive critic learning techniques for engine torque and air–fuel ratio control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 988–993, 2008.

[51] D. Wang, D. Liu, and Q. Wei, "Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14–22, 2012.

[52] Z. Ni, X. Fang, H. He, D. Zhao, and X. Xu, "Real-time tracking control on adaptive critic design with uniformly ultimately bounded condition," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL'13), IEEE Symposium Series on Computational Intelligence (SSCI)*, Apr. 2013.

[53] Z. Ni, H. He, J. Wen, and X. Xu, "Goal representation heuristic dynamic programming on maze navigation," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 2038–2050, 2013.

[54] Z. Ni and H. He, "Heuristic dynamic programming with internal goal representation," *Soft Computing*, vol. 17, pp. 2101–2108, 2013.

[55] H. He, Z. Ni, and D. Zhao, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Learning and Optimization in Hierarchical Adaptive Critic Design, pp. 78–95.

[56] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 14–25, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tsmc/tsmcb41.html#LewisV11

[57] Z. Ni, H. He, and X. Zhong, *Frontiers of Intelligent Control and Information Processing*. World Scientific Publishing, 2014, in press, ch. Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP.

[58] X. Zhang, H. Zhang, Q. Sun, and Y. Luo, "Adaptive dynamic programming-based optimal control of unknown nonaffine nonlinear discrete-time systems with proof of convergence," *Neurocomputing*, vol. 91, pp. 48–55, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon91.html#ZhangZSL12

[59] X. Zhong, H. He, H. Zhang, and Z. Wang, "Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming," *Neural Networks and Learning Systems, IEEE Transactons on*, vol. 25, no. 12, pp. 2141–2155, 2015.

[60] M. Abu-Khalaf, F. L. Lewis, and J. Huang, "Neurodynamic programming and zero-sum games for constrained control systems," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1243–1252, 2008.

[61] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[62] A. Bernstein and N. Shimkin, "Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains," *Machine learning*, vol. 81, no. 3, pp. 359–397, 2010.

[63] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1997.

CHAPTER 3

Event-triggered ADP Control

3.1 Introduction

In literature, digital control methods are relying on the periodic transmitted data using the fixed sampling period. However, huge number of the transmitted data may cause subsequent tremendous computation, especially when the computation bandwidth or sensor power sources are constrained. In recent years, the event-triggered control method has been studied for its capability of computation efficiency [1], [2]. In the event-triggered control algorithm, the controller is only updated when an event is triggered, and thus the computation is significantly saved [3], [4], [5]. Currently, the event-triggered control methods are based on the accurate system function or model [6], [7]. In many cases, the complete knowledge of the system function is either infeasible or very difficult to obtain. Recently, neural-network-based event-triggered optimal control approaches were proposed and demonstrated with the promising performance in [8], [9], [10].

ADP techniques have been studied and adopted for seeking the solution of the Hamilton-Jacobi-Bellman (HJB) equation in recent years [11, 12, 13]. Extensive efforts and promising results have been achieved over the past decades, such as the special issue on feedback control provided well-known feedback control problems with new techniques of ADP [14]. Higher level exploration, like [15, 16], showed the deeper thinking for the future development on ADP community. In addition, ADP methods demonstrate the control capabilities in many real applications, like the power system stability/transient control in [17, 18], the looper system control in the iron and steel company in [19, 20], the engine torque and air-fuel control in [21] and among others [22, 23, 24]. Stability analysis of the ADP control on dynamic systems were provided under certain conditions in [25, 26, 27]. The performance index function and the control

law were studied and demonstrated in [28, 29]. The robust controller with the ADP technique was also presented in [30].

In this chapter, the event-triggered control technique is integrated into the ADP approach for the unknown nonlinear continuous-time system. First, the stability analysis is investigated for the event-triggered method. The event-triggered controller is then implemented with the neural network techniques. That is, we use an action network to approximate the control law based on the event-triggered sample data (with event-triggered techniques), and use a critic network to evaluate the control performance with the value function. The pseudo-code for the event-triggered algorithm is provided and the weights updating rules are subsequently derived. The weights evolution in the learning process are provided to show the achieved learned/optimal policy. The performance of both the traditional ADP approach and the proposed event-triggered ADP approach are also provided in the simulation studies for the comparative studies. From the simulation results, we know that the proposed event-triggered ADP method can achieve competitive performance with limited sampled data. Note that this method relies on the on-line learning process and the information of the system dynamics is unknown in the learning process.

3.2 Event-triggered Method Design

Consider a nonlinear continuous-time system with the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{2}$$

where $x(t) \in R^n$ denotes the system state variable with the initial state $x(0) = x_0$ and $u(t) \in R^m$ is the control input. $f(x(t))$ and $g(x(t))$ are the unknown system functions. Assume that $f(x(t)) + g(x(t))u(t)$ is Lipschitz continuous on a set $\Omega \subseteq R^n$, and $f(0) = 0,\ g(0) = 0$. In order to save resources, we design a sampled-data system which is characterized by a monotonically increasing sequence of sampling instants $\{\delta_j\}_{j=0}^{\infty}$, where $\delta_j < \delta_{j+1}$ for $j = 0, 1, 2, \cdots, \infty$. The time δ_j denotes the jth consecutive sampling

instant. The output of the sampled-data system is a sequence of the sampled states which can be described by

$$\hat{x}_j = x(\delta_j). \tag{3}$$

For simplicity, we assume that the sampled-data system has zero task delay. Define the gap function for $\forall t \in [\delta_j, \delta_{j+1})$ as

$$e_j(t) = \hat{x}_j - x(t) \tag{4}$$

which is the difference between the sampled state and the current state. It is obvious that at the beginning of the interval $[\delta_j, \delta_{j+1})$, the gap function in (4) equals to zero. After that, one expects the norm of the gap function to increase. When the value is larger than a threshold e_T, then the system state is again sampled by setting $\hat{x}_j = x(t)$, thereby forcing the gap function to zero again.

We are interested in the state-feedback controller $\gamma(\hat{x}_j)$, which maps the sampled state onto a control vector. Assume that $\gamma(\hat{x}_j)$ is a Lipschitz continuous function. The obtained control sequence $\{\gamma(\hat{x}_j)\}_{j=0}^{\infty}$ becomes a continuous-time signal through a zero-order hold (ZOH). In particular, this control signal can be seen as a piecewise constant function and within any time interval $[\delta_j, \delta_{j+1})$, the controller is $u(t) = \gamma(\hat{x}_j)$, $j = 0, 1, 2, \cdots, \infty$.

Rewrite equation (4) as $\hat{x}_j = x(t) + e_j(t)$, so that the closed loop dynamics can be described as

$$\dot{x}(t) = f(x(t)) + g(x(t))\gamma(x(t) + e_j(t)), \ \forall t \in [\delta_j, \delta_{j+1}). \tag{5}$$

Similar to the traditional ADP problem, it is desired to find a controller $u(t)$ that minimizes the performance index given as

$$\begin{aligned} V(x_0) &= \int_0^{\infty} U(x(\tau), u(\tau))d\tau \\ &= \sum_{\bigcup_j [\delta_j, \delta_{j+1}) = [0,\infty)} \int_{\delta_j}^{\delta_{j+1}} U(x(\tau), \gamma(\hat{x}_j))d\tau \end{aligned} \tag{6}$$

where $U(x(\tau), \gamma(\hat{x}_j))$ is the utility function with $U(0,0) = 0$. In this chapter, the utility function is given by

$$U(x(t), \gamma(\hat{x}_j)) = x^T(t)Qx(t) + \gamma^T(\hat{x}_j)R\gamma(\hat{x}_j) \tag{7}$$

in which Q and R are symmetric and positive definite matrices with appropriate dimensions, and they can be described by

$$Q = q \cdot q^T \qquad R = r \cdot r^T \tag{8}$$

Definition 1: A law $u(t)$ is said to be an admissible control with respect to (6) on Ω, if $u(t)$ is continuous on Ω and can stabilize system (2) for all $x_0 \in \Omega$, $u(t) = 0$ if $x(t) = 0$, and $V(x_0)$ is finite, $\forall x(t) \in \Omega$.

Equation (6) can be expanded as follows

$$\begin{aligned} V(x_0) = & \sum_{\bigcup_j [\delta_j, \delta_{j+1}) = [0, \delta_1)} \int_{\delta_j}^{\delta_{j+1}} U(x(\tau), \gamma(\hat{x}_j)) d\tau \\ & + \sum_{\bigcup_j [\delta_j, \delta_{j+1}) = [\delta_1, \infty)} \int_{\delta_j}^{\delta_{j+1}} U(x(\tau), \gamma(\hat{x}_j)) d\tau \\ = & \int_0^{\delta_1} U(x(\tau), \gamma(\hat{x}_j)) d\tau + V(x(\delta_1)). \end{aligned} \tag{9}$$

After transformation, equation (9) becomes

$$\begin{aligned} & \lim_{\delta_1 \to 0} \left[\frac{V(x(\delta_1)) - V(x_0)}{\delta_1} \right] \\ = & - \lim_{\delta_1 \to 0} \frac{1}{\delta_1} \int_0^{\delta_1} [x^T(\tau)Qx(\tau) + \gamma^T(\hat{x}_j)R\gamma(\hat{x}_j)] d\tau. \end{aligned} \tag{10}$$

Then, we obtain the infinitesimal version of (6) as

$$V_x^T(f(x(t)) + g(x(t))\gamma(\hat{x}_j, t)) + x^T(t)Qx(t)\gamma^T(\hat{x}_j, t)R\gamma(\hat{x}_j, t) = 0 \tag{11}$$

where $V_x = \frac{\partial V(x(t))}{\partial x(t)}$ is the partial derivative of the performance index with respect to the state and $\gamma(\hat{x}_j, t)$ is the continuous-time signal of the event-triggered control law $\gamma(\hat{x}_j)$. Given that $u(t) = \gamma(\hat{x}_j, t)$ is an admissible control law, if $V(x(t))$ satisfies (11) and

$Q \geq 0$, $R \geq 0$, then $V(x(t))$ is a Lyapunov function for the system (2) with the control law $u(t) = \gamma(\hat{x}_j, t)$. Note that, in order to simplify the expression, we use $\gamma(\hat{x}_j)$ to represent $\gamma(\hat{x}_j, t)$ in the following presentation.

According to Bellman's optimality equation, the optimal performance index $V^*(x(t))$ satisfies

$$\min_{\gamma(\hat{x}_j)}[V_x^{*T}(f(x(t)) + g(x(t))\gamma(\hat{x}_j)) + x^T(t)Qx(t) + \gamma^T(\hat{x}_j)R\gamma(\hat{x}_j)] = 0. \tag{12}$$

Assume that the minimum on the left-hand side of the equation (12) exists and is unique. Therefore, the optimal control $\gamma^*(\hat{x}_j)$ satisfies the first-order necessary condition, which is given by the gradient of (11) with respect to $\gamma(\hat{x}_j)$. Note that, in the event-triggered method, the controller is only updated when an event is triggered. In other words, the controller is designed based on the event-triggered sampling state $\hat{x}_j$ rather than the real state $x(t)$. Hence, we have $g(x(t)) = g(\hat{x}_j)$ and $V_x = V_{\hat{x}_j}$, where $V_{\hat{x}_j} = \frac{\partial V(\hat{x}_j)}{\partial x(t)}$ is the partial derivative of the event-triggered performance index with respect to the state. Therefore, we obtain the event-triggered optimal control as

$$u^*(t) = \gamma^*(\hat{x}_j) = -\frac{1}{2}R^{-1}g^T(\hat{x}_j)V_{\hat{x}_j}^*. \tag{13}$$

By substituting (13) into (11), we obtain the HJB equation under event-triggered method as follows

$$\begin{aligned} &V_x^{*T}f(x(t)) - \frac{1}{2}V_x^{*T}g(x(t))R^{-1}g^T(\hat{x}_j)V_{\hat{x}_j}^* \\ &+\frac{1}{4}V_{\hat{x}_j}^{*T}g(\hat{x}_j)R^{-1}g^T(\hat{x}_j)V_{\hat{x}_j}^* + x^T(t)Qx(t) = 0 \end{aligned} \tag{14}$$

with $V^*(0) = 0$.

In the next section, it will be shown that the event-triggered control (13) is admissible and can stabilize the nonlinear continuous-time system (2).

3.3 Stability Analysis of the Event-Triggered Method

Assumption: The controller $\gamma(x)$ is Lipschitz continuous with respect to the gap,

$$\|\gamma(x(t)) - \gamma(\hat{x}_j)\| = \|\gamma(x(t)) - \gamma(x(t) + e_j(t))\| \leq L\|e_j(t)\| \tag{15}$$

where L is a positive real constant.

The stability analysis of the event-triggered controller is provided as follows.

Theorem 1: Consider the nonlinear continuous-time system (2). For $\forall t \in [\delta_j, \delta_{j+1})$, the control law is given by (13) and assume $V^*(x(t))$ is the solution of the event-triggered HJB equation (14). If the triggered condition is defined as follows

$$\|e_j(t)\|^2 \leq \|e_T\|^2 = \frac{(1-\alpha^2)}{L^2\|r\|^2}\underline{\lambda}(Q)\|x(t)\|^2 + \frac{1}{L^2}\|\gamma^*(\hat{x}_j)\|^2 \tag{16}$$

where $\underline{\lambda}(Q)$ is the minimal eigenvalue of Q, $\alpha \in (0,1)$ is the designed parameter, and e_T is the threshold of the gap between the sampled and the real state, then the following conditions hold.

(1) The event-triggered control law (13) is an admissible control.

(2) The event-triggered control law (13) can asymptotically stabilize the nonlinear system (2).

Proof: Let us start with the admissibility part. From equation (13), we know when the state $\hat{x}_j = 0$, then $g(\hat{x}_j) = 0$ and hence $\gamma^*(\hat{x}_j) = 0$. The continuity assumption on $f(x(t)) + g(x(t))u(t)$ and $\gamma^*(\hat{x}_j)$ implies that $\gamma^*(\hat{x}_j)$ is continuous and the system (2) cannot jump to infinity by any one step of finite control. Moreover because $f(0) = 0$, $g(0) = 0$, when the system state $x(t)$ reaches the equilibrium state, $\gamma^*(\hat{x}_j)$ becomes zero and the state is kept at zero. Therefore, according to Definition (1), we obtain event-triggered control law $\gamma^*(\hat{x}_j)$ is an admissible control which proves the part (1).

Now we will show that $\gamma^*(\hat{x}_j)$ can asymptotically stabilize the nonlinear continuous-time system (2). Let $\gamma^*(\hat{x}_j(t))$ and $V^*(x(t))$ be the optimal event-triggered control law and the optimal performance index obtained in equation (13) and (14), respectively. From equation (6), we know $V^*(x(t))$ is a positive definite function, namely, $V^*(x(t)) > 0$ for any $x(t) \neq 0$ and $V^*(x(t)) = 0$ when $x(t) = 0$. Hence, $V^*(x(t))$ can be seen as a Lyapunov function.

With the event-triggered controller, the derivative of $V^*(x(t))$ along the system

trajectory can be obtained as,

$$\begin{aligned}\dot{V}^*(x(t)) &= \left(\frac{\partial V^*(x(t))}{\partial x(t)}\right)^T \cdot \dot{x} \\ &= V_x^{*T} f(x(t)) + V_x^{*T} g(x(t))\gamma^*(\hat{x}_j)\end{aligned} \tag{17}$$

Here, we recall the control law and the HJB equation in the traditional ADP method as

$$u^*(t) = -\frac{1}{2}R^{-1}g^T(x)V_{x(t)}^* \equiv \gamma^*(x(t)) \tag{18}$$

and

$$V_x^{*T} f(x(t)) - \frac{1}{4}V_x^{*T} g(x(t))R^{-1}g^T(x(t))V_x^* + x^T(t)Qx(t) = 0. \tag{19}$$

Therefore,

$$g^T(x(t))V_x^* = -2R\gamma^*(x(t)) \tag{20}$$

$$V_x^{*T} f(x(t)) = \frac{1}{4}V_x^{*T} g(x(t))R^{-1}g^T(x(t))V_x^* + x^T(t)Qx(t) \tag{21}$$

Substitute (20) and (21) into (17), we have

$$\begin{aligned}\dot{V}^*(x(t)) =& \frac{1}{4}V_x^{*T} g(x(t))R^{-1}g^T(x(t))V_x^* + x^T(t)Qx(t) + V_x^{*T} g(x(t))\gamma^*(\hat{x}_j) \\ =& \gamma^{*T}(x(t))R\gamma^*(x(t)) - x^T(t)Qx(t) - 2\gamma^{*T}(x(t))R\gamma^*(\hat{x}_j)\end{aligned} \tag{22}$$

Because $R = r \cdot r^T$, we obtain

$$\gamma^{*T}(x(t))R\gamma^*(x(t)) - 2\gamma^{*T}(x(t))R\gamma^*(\hat{x}_j) = \|r^T\gamma^*(x(t)) - r^T\gamma^*(\hat{x}_j)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2. \tag{23}$$

By substituting (23) into (22) and using the Lipschitz condition from Assumption 1, we have

$$\begin{aligned}\dot{V}^*(x(t)) =& \|r^T\gamma^*(x(t)) - r^T\gamma^*(\hat{x}_j)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2 - x^T(t)Qx(t) \\ \leq& L^2\|r\|^2\|e_j(t)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2 - x^T(t)Qx(t) \\ \leq& L^2\|r\|^2\|e_j(t)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2 - \underline{\lambda}(Q)\|x(t)\|^2 \\ =& -\alpha^2\underline{\lambda}(Q)\|x(t)\|^2 + \Big[-(1-\alpha^2)\underline{\lambda}(Q)\|x(t)\|^2 \\ &+ L^2\|r\|^2\|e_j(t)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2\Big]\end{aligned} \tag{24}$$

since $-x^T(t)Qx(t) \leq -\underline{\lambda}(Q)\|x(t)\|^2$, where $\underline{\lambda}(Q)$ is the minimal eigenvalue of Q.

Based on the condition (16), we know that the last three terms in (24) is guaranteed negative. Therefore, (24) can be modified as follows

$$\begin{aligned}\dot{V}^*(x(t)) \leq & (1-\alpha^2)\underline{\lambda}(Q)\|x(t)\|^2 + \|r\|^2\|\gamma^*(\hat{x}_j)\|^2 - \|r^T\gamma^*(\hat{x}_j)\|^2 - \underline{\lambda}(Q)\|x(t)\|^2 \\ = & -\alpha^2\underline{\lambda}(Q)\|x(t)\|^2 \\ < & 0\end{aligned} \tag{25}$$

for any $x(t) \neq 0$. Thus, $u^*(t) = \gamma^*(\hat{x}_j)$ can asymptotically stabilize the nonlinear continuous-time system (2). The conclusion holds. ■

From Theorem 1, we know that the controller is guaranteed stable (under certain conditions) with the event-triggered sample data. In the next section, we are applying the neural network methods to implement the event-triggered ADP approach.

3.4 Neural-network-based Event-Triggered Controller Design

In this section, an ADP approach is provided to solve the event-triggered HJB equation (14) and approximate the optimal event-triggered control law (13). The neural network techniques are employed to implement this approach. Two subsections are included. The first one shows the event-triggered online learning ADP algorithm for nonlinear continuous-time system. The neural network implementation is presented in the second subsection.

3.4.1 Event-Triggered Control Law Estimation

Set the initial triggered state as $\hat{x}_0 = x_0$. Note that if we use equation (13) to calculate the event-triggered control law, the system function $g(\hat{x}_j)$ is required which is unknown in this chapter. Hence, we provide a method to approximate the control updating equation (13). The algorithm can be described as Algorithm 1.

From Algorithm 1, it is obvious that by estimation of the control law, no system information is required during the learning process. The control law is only updated when

Algorithm 1 Event-triggered ADP Algorithm Using Only the Measured Input-Output Data.

Set $i = 0$, $j = 0$, $\hat{x}_0 = x_0$
Calculate $\mu(\hat{x}_j) = -\frac{1}{2}R^{-1}g^T(\hat{x}_j)\hat{V}_{\hat{x}_j}$
for all $i < N_{run}$ **do**
  State estimation:
  $\dot{\hat{x}} = A\hat{x} + \hat{F}_A(\hat{x}, \mu(\hat{x}_j)) + G(y - C\hat{x})$
  Policy evaluation:
  $V(\hat{x}) = \min_{\mu(\hat{x}_j)} \int_0^\infty U(\hat{x}(\tau), \mu(\hat{x}_j))d\tau$
  if $\hat{x}_j - \hat{x} = \hat{e}_j > e_T$ **then**
    Set $j = j + 1$, $\hat{x}_j = \hat{x}$
    Update $\mu(\hat{x}_j) = \arg\min_{\mu(\hat{x}_j)}\{V(\hat{x}_j)\}$
  end if
  Update system information $\dot{x} = F(x, \mu(\hat{x}_j));\ y = Cx$
  Set $i = i + 1$
end for

an event is triggered. In the next subsection, we will provide the explicit approximation process based on neural network techniques.

3.4.2 Neural-Network-based Implementation

The neural networks are employed in this subsection to approximate the event-triggered control law. The architecture of this event-triggered method is shown in Figure.3. A critic network and an action network are built to approximate the performance index and the control law of the event-triggered method, respectively. We can observe that a sampled-data system is used during this process with the sampling instants $\{\delta_j\}_{j=0}^{\infty}$. As we provided above, $\{\delta_j\}_{j=0}^{\infty}$ are obtained based on the gap function $(e_j(t))$ which is the difference between the current and the sampled state. When $e_j(t)$ is larger than the threshold e_T, the system state is sampled by $\hat{x}_j = x(\delta_j)$, and the action network is updated based on the event-triggered sample state. Then through the ZOH, the control law sequence is transformed into a continuous-time control signal. Assume that the sampling period for the discretization is $\triangle t$. We set both the critic and the action network used in this chapter be the three-layer networks. In the following part, we will

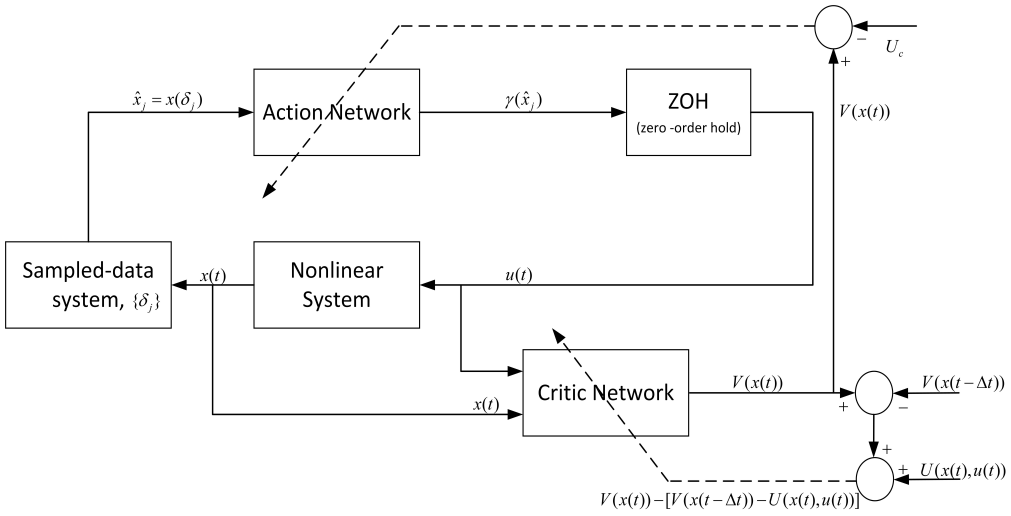

Figure 3. Architecture of the event-triggered method based on the ADP approach.

provide the online learning rules for both neural networks.

Critic Network

The critic network is used to approximate the performance index $V(x(t))$ which can be formulated as

$$V(x(t)) = \omega_{c2}^T(t)\Phi(h(t)) \tag{26}$$

where $\omega_{c2}^T(t)$ is the weight matrix between the hidden and the output layer of the critic network and $h(t) = \omega_{c1}^T[x^T(t), \gamma^T(\hat{x}_j)]$, to which ω_{c1} denotes the weight matrix between the hidden and the input layer. Note that ω_{c1} is randomly chosen as initial and is kept constantly during the implementation process in this chapter.

$\Phi(x)$ is a sigmoid function that can be described as

$$\Phi(x) = \frac{1 - e^{-x}}{1 + e^{-x}}. \tag{27}$$

The purpose of the sigmoid function is to constrain the output into $[-1, 1]$. Here the sigmoid function is applied on the hidden to output nodes.

Define the error function for the critic network by

$$e_c(t) = V(x(t)) - [V(x(t - \triangle t)) - U(x(t), \gamma^T(\hat{x}_j)] \tag{28}$$

where $\triangle t$ is the sampling period during discretization.

Therefore, to update the weight matrix is to minimize the following objective function

$$E_c(t) = \frac{1}{2} e_c^2(t). \tag{29}$$

Hence, we obtain the critic network weights adjustments for the hidden to the output layer

$$\omega_{c2}(t + \triangle t) = \omega_{c2}(t) - \beta_c \left(\frac{\partial E_c(t)}{\partial \omega_{c2}(t)} \right) \tag{30}$$

where $\beta_c > 0$ is the learning rate of the critic network. According to the chain-backpropagation rules, we derive the tuning formula as

$$\frac{\partial E_c(t)}{\partial \omega_{c2}(t)} = \frac{\partial E_c(t)}{\partial V(x(t))} \frac{\partial V(x(t))}{\partial \omega_{c2}(t)} \tag{31}$$

Action Network

The purpose of the action network is to estimate the optimal event-triggered control law. As we discussed, the action network is only updated when an event is triggered. Therefore, the estimated control law can be formulated as

$$\gamma^T(\hat{x}_j) = \Phi(\omega_{a2}^T(\delta_j) g(\delta_j)) \tag{32}$$

$$g(\delta_j) = \Phi(\omega_{a1}^T(\delta_j) \hat{x}_j) \tag{33}$$

where $\omega_{a1}(\delta_j)$ and $\omega_{a2}(\delta_j)$ are the weight matrices of the input-to-hidden and the hidden-to-output layer at the sampled time δ_j, respectively. Sigmoid function is applied on both hidden and the output side. $\hat{x}_j$ is the sampled state and is also the input of the action network. The same as above, we fix the input-to-hidden layer weight matrix $\omega_{a1}(\delta_j)$ which is chosen initially at random. Therefore, only the weight matrix $\omega_{a2}(\delta_j)$ between the hidden and the output layer is needed to be updated.

We know the objective for the action network is to minimize the total future cost,

hence we define the error function here by

$$e_a(\delta_j) = V(\hat{x}_j) - U_c \tag{34}$$

where U_c is the ultimate utility function. The value of U_c is critical in ADP design and it could be variant in different application. In this chapter, we choose $U_c = 0$.

The objective function of the action, therefore, can be written as

$$E_a(\delta_j) = \frac{1}{2} e_a^2(\delta_j) \tag{35}$$

The gradient descent method is also applied to minimize the approximation error (35) as

$$\omega_{a2}(\delta_{j+1}) = \omega_{a2}(\delta_j) - \beta_a \left(\frac{\partial E_a(\delta_j)}{\partial \omega_{a2}(\delta_j)} \right) \tag{36}$$

where $\beta_a > 0$ is the learning rate of the action network. From the chain backpropagation rule, we obtain

$$\frac{\partial E_a(\delta_j)}{\partial \omega_{a2}(\delta_j)} = \frac{\partial E_a(\delta_j)}{\partial V(\hat{x}_j)} \frac{\partial V(\hat{x}_j)}{\partial \gamma^T(\hat{x}_j)} \frac{\partial \gamma^T(\hat{x}_j)}{\partial \omega_{a2}(\delta_j)} \tag{37}$$

3.5 Simulation Studies

Consider a single link robot arm with the following dynamic function

$$\ddot{\theta}(t) = -\frac{MgH}{G} \sin(\theta(t)) - \frac{D}{G} \dot{\theta}(t) + \frac{1}{G} u(t) \tag{38}$$

where $\theta(t)$ is the angle position of robot arm, and $u(t)$ is the control input. Moreover, M is the mass of the payload, G is the moment of inertia, g is the acceleration of gravity, H is the length of the arm and D is the viscous friction, where g, H, D are the system parameters and M, G are the design parameters. Set the values of the system parameters as $g = 9.81$, $D = 2$, and $L = 0.5$, and the design parameters M and G are alterable. Assuming $x_1(t) = \theta(t)$ and $x_2(t) = \dot{\theta}(t)$, the dynamic function (38) can be rewritten by

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -\dfrac{2}{G} x_2(t) + \dfrac{1}{G} u(t) - \dfrac{4.905 M \sin(x_1(t))}{G}. \end{cases} \tag{39}$$

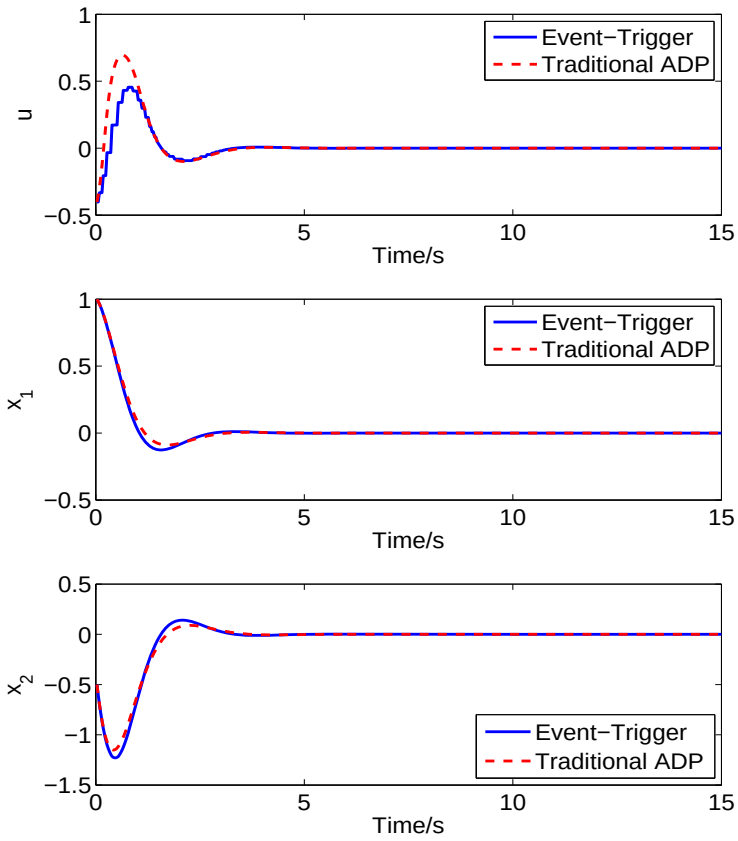

Figure 4. Comparisons of system responses by the event-triggered and the traditional ADP method with $M = 1$ and $G = 1$.

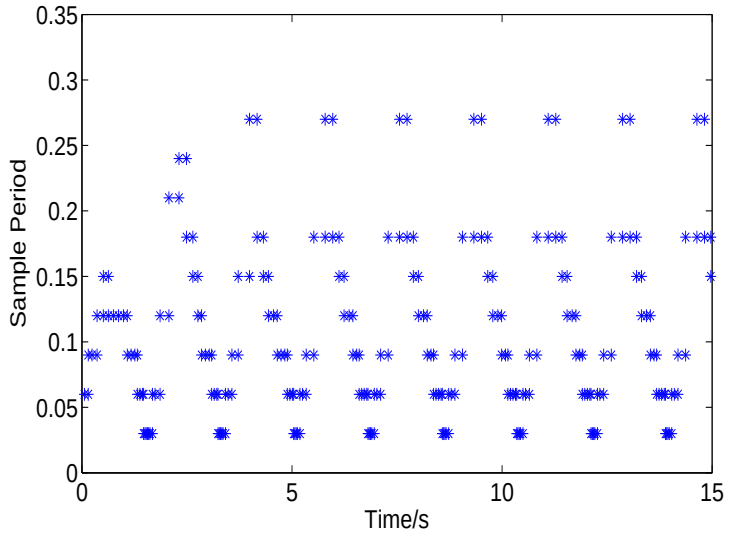

Figure 5. Inter-event instants during the learning process with $M = 1$ and $G = 1$.

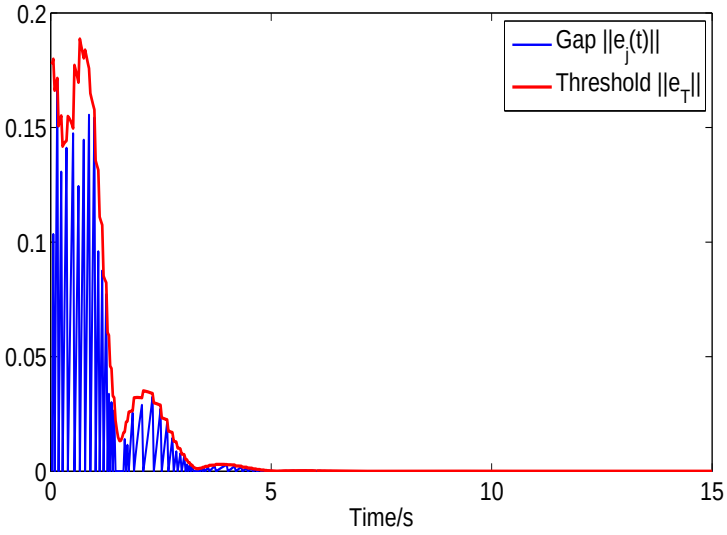

Figure 6. Response of the gap $\|e_j(t)\|$ and the threshold $\|e_T\|$ with $M = 1$ and $G = 1$.

We use the event-triggered method proposed in this chapter to solve the problem. Choose the threshold according to condition (16) with $L = 3$, $\alpha = 0.95$. Set Q, R and r are the identity matrices with appropriate dimensions. Therefore, the threshold is

$$\begin{aligned} \|e_T\|^2 &= \frac{(1-\alpha^2)}{L^2\|r\|^2}\underline{\lambda}(Q)\|x(t)\|^2 + \frac{1}{L^2}\|\gamma(\hat{x}_j)\|^2 \\ &= \frac{1-0.95^2}{9}\|x(t)\|^2 + \frac{1}{9}\|\gamma(\hat{x}_j(t))\|^2. \end{aligned} \tag{40}$$

When the gap $e_j(t) = \hat{x}_j - x(t)$ satisfies the condition $\|e_j(t)\|^2 > \|e_T\|^2$, then the system state is again sampled by setting $\hat{x}_j = x(t)$.

Two three-layer neural networks are built as the critic and the action networks. The neuron structures of the critic and the action network are $3-8-1$ (i.e., three input neuros, eight hidden neuros, and one output neuros) and $2-6-1$, respectively. Set the learning rates of both networks as $\beta_c = \beta_a = 0.01$, and the sampling period for discretization as $\triangle t = 0.03s$. The initial weights of both networks are chosen randomly within $[-1, 1]$. The initial state is set to $x_0 = [1, -0.5]$. The input of the action network is the sampled state.

In the first case, we set the design parameters as $M = 1$, $G = 1$. By employing the event-triggered method proposed in this chapter, we obtain the system responses in

Figure 4. Note that, in order to demonstrate the performance of our method, we also conduct this example under the traditional ADP method with the same initial weights which is also presented in Figure 4. From the comparison, we know that the event-triggered control law keeps the same at period $[\delta_j, \delta_{j+1})$ and is only updated when an event is triggered. The control law evolution and the state trajectories of the event-triggered method are very close to those of the traditional ADP method. This means efficiently reducing the sampled times does not influence the system performance. The sampling period during the event-triggered learning process is provided in Figure 5 which shows that the sampling period is up to $0.27s$. The relationship between the gap $\|e_j(t)\|$ and the threshold $\|e_T\|$ is shown in Figure 6. The learning trajectories of the critic and the action network weights from the hidden to the output layer is provided in Figure 7. It is obvious that the weights converge after $3s$. In particular, comparing the event-triggered and

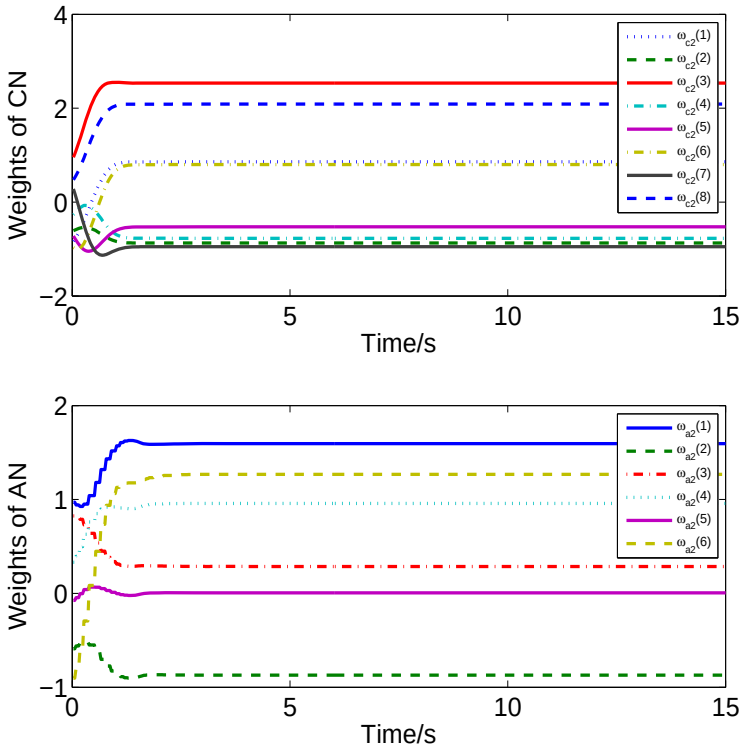

Figure 7. Learning trajectories of the critic and the action network weights from the hidden to the output layer with $M = 1$ and $G = 1$.

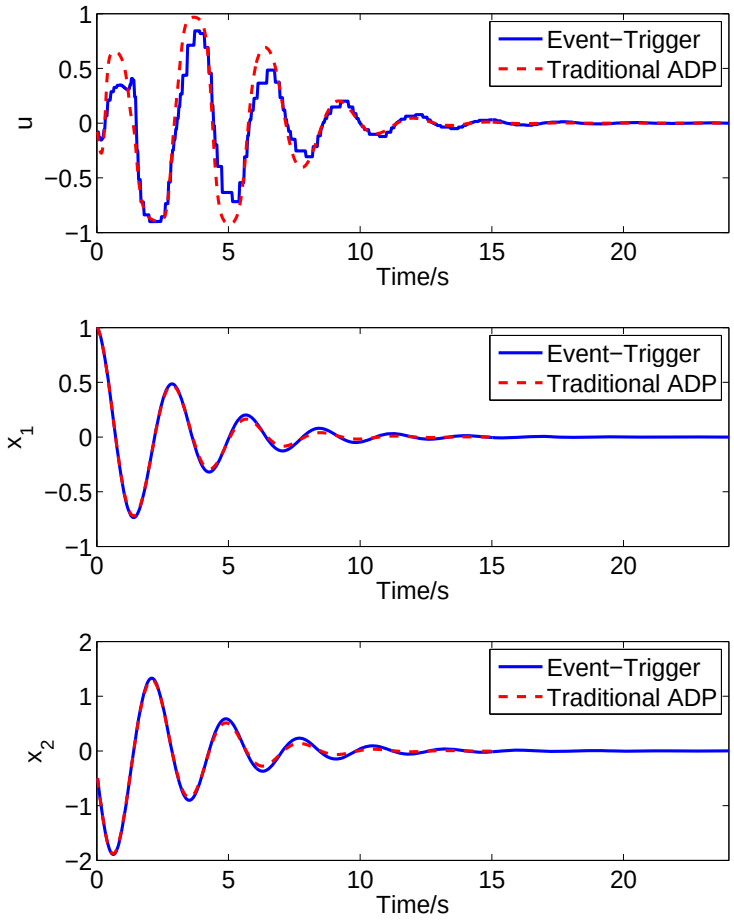

Figure 8. Comparisons of system responses by the event-triggered and the traditional ADP method with $M = 5$ and $G = 5$.

the traditional ADP method, the event-triggered controller uses 161 samples of the state while the traditional ADP controller uses 500 samples, which means the even-triggered method improved the learning process.

In the second case, we conduct the example with the design parameters $M = 5$, $G = 5$. The comparison of the system responses by the event-triggered and the traditional ADP method with the same initial weights is presented in Figure 8. We can observe that the event-triggered method also works with the high design parameters. The sampling period during the learning process of the event-triggered method is provided in Figure 9. We know the sampling period is up to $0.39s$ in this case. The relationship

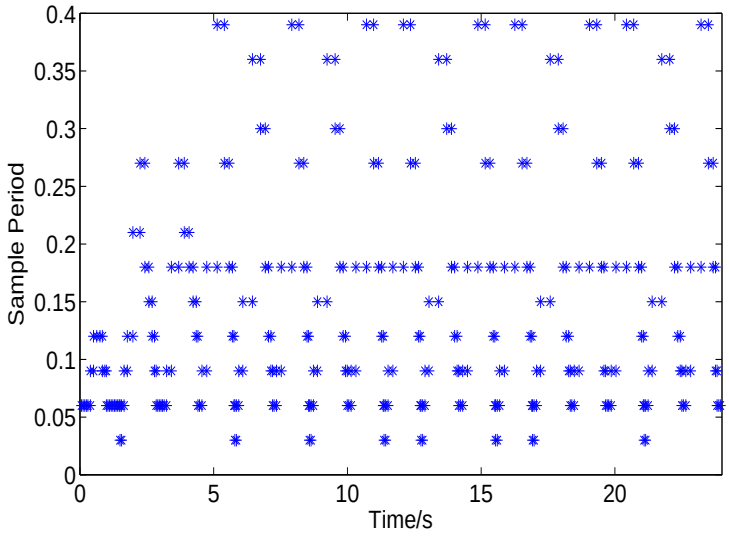

Figure 9. Inter-event instants during the learning process with $M = 5$ and $G = 5$.

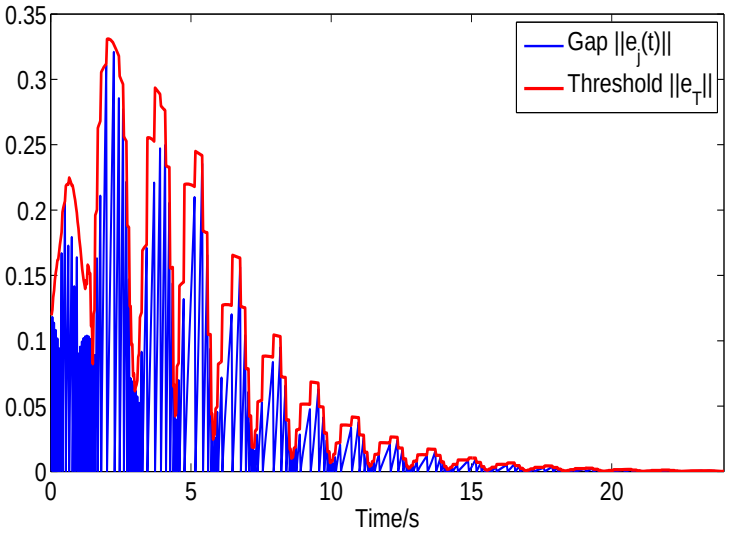

Figure 10. Response of the gap $\|e_j(t)\|$ and the threshold $\|e_T\|$ with $M = 5$ and $G = 5$.

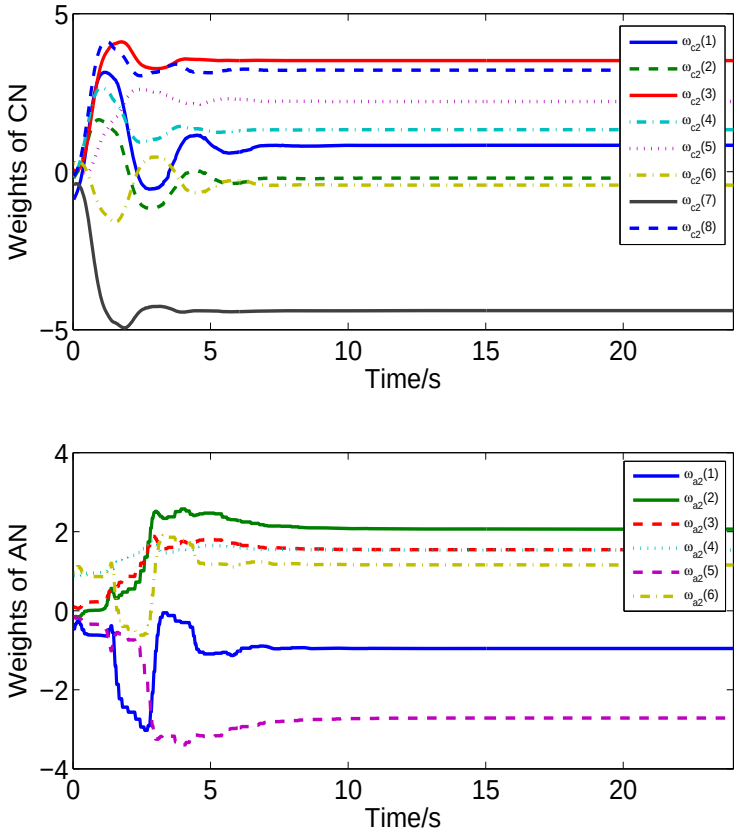

Figure 11. Learning weights of the critic and the action network from the hidden to the output layer with $M = 5$ and $G = 5$.

between the gap $\|e_j(t)\|$ and the threshold $\|e_T\|$ is shown in Figure 10. Moreover, the learning weights of the critic and the action network from the hidden to the output layer is provided in Figure 11. In this learning process, the event-triggered controller uses 291 samples of the state while the traditional ADP controller uses 800 samples. This means the proposed method can reduce the computation cost and achieve the competitive results at the same time.

Additionally, without loss of generality, we choose the values of the design parameters as $M = 1, 2, \cdots, 20$ and $G = 1, 2, \cdots, 20$. For each pair of the design parameters, we conduct the simulation based on the proposed method for 100 times. The sampling period for discretization is set as $\triangle t = 0.03s$ and each simulation lasts $25s$. This means the traditional ADP controller will use 800 samples to stabilize the system. However, from

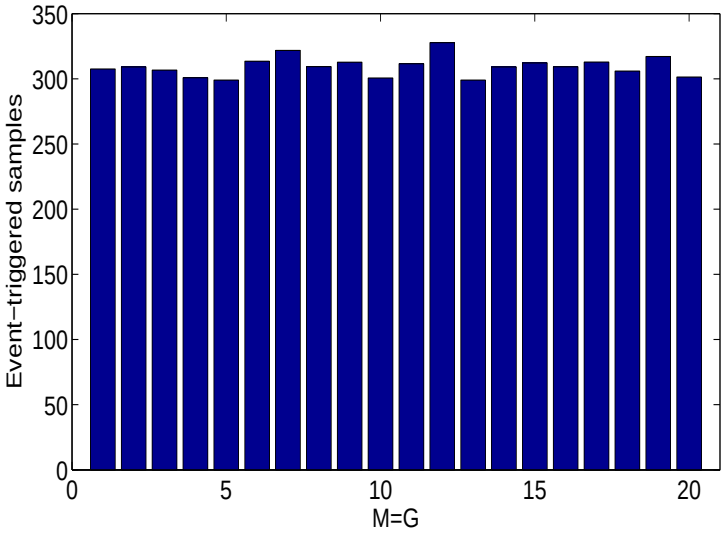

Figure 12. The average number of the samples used by the event-triggered controller for each parameters pair within $[1, 20]$.

Figure 12, we know the average number of the samples used by the event-triggered controller for each parameters pair is around 300, which is significantly less than the samples used by the traditional ADP controller. All the simulation studies indicate that the designed event-triggered ADP control method is effective.

3.6 Conclusion

In this chapter, we design an event-triggered controller for nonlinear continuous-time system using ADP approach. The system function is assumed to be unknown. The controller is updated only based on the triggered state. A zero-order hold is used to transform the control sequence into a continuous-time signal. The threshold for triggering an event is discussed and the stability of this event-triggered controller is analyzed. Neural network techniques are used to approximate the performance index and the controller in event-triggered method, respectively. The stability of the designed controller is analyzed in this chapter. The simulation results demonstrate the effectiveness of the designed controller and also verify the theoretical analysis. In the next chapter, I am going to further demonstrate its adaptive learning mechanism in the partially observable

environment.

List of References

[1] W. Heemels, M. Donkers, and A. Teel, "Periodic event-triggered control for linear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 847–861, 2013.

[2] P. Tallapragada and N. Chopra, "On event triggered tracking for nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2343–2348, 2013.

[3] M. Lemmon, "Event-triggered feedback in control, estimation, and optimization," in *Networked Control Systems*. Springer, 2010, pp. 293–358.

[4] E. Garcia and P. J. Antsaklis, "Model-based event-triggered control with time-varying network delays," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 1650–1655.

[5] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3270–3285.

[6] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered control for discrete-time systems," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 4719–4724.

[7] W. Heemels and M. Donkers, "Model-based periodic event-triggered control for linear systems," *Automatica*, 2013.

[8] A. Sahoo, H. Xu, and S. Jagannathan, "Neural network-based adaptive event-triggered control of affine nonlinear discrete time systems with unknown internal dynamics," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 6418–6423.

[9] A. Sahoo, H. Xu, and S. Jagannathan, "Neural network-based adaptive event-triggered control of nonlinear continuous-time systems," in *Intelligent Control (ISIC), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 35–40.

[10] D. Tolic, R. Fierro, and S. Ferrari, "Optimal self-triggering for nonlinear systems via approximate dynamic programming," in *Control Applications (CCA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 879–884.

[11] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press and John Wiley & Sons, 2004.

[12] F. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.

[13] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive Dynamic Programming for Control: Algorithms and Stability (Communications and Control Engineering)*. Springer, 2013.

[14] F. L. Lewis, D. Liu, and G. G. Lendaris, "Special issue on adaptive dynamic programming and reinforcement learning in feedback control," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 896–897, 2008.

[15] P. J. Werbos, "ADP: The key direction for future research in intelligent control and understanding brain intelligence," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 898–900, 2008.

[16] G. G. Lendaris, "Higher level application of adp: A next phase for the control field?" *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 901–912, 2008.

[17] W. Qiao, G. Venayagamoorthy, and R. Harley, "DHP-based wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices," in *Proc. IEEE Int. Conf. Neural Netw.*, 2007, pp. 2093–2098.

[18] S. Ray, G. K. Venayagamoorthy, B. Chaudhuri, and R. Majumder, "Comparison of adaptive critics and classical approaches based wide area controllers for a power system," *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.

[19] J. Fu, H. He, and X. Zhou, "Adaptive learning and control for mimo system based on adaptive dynamic programming," *IEEE Trans. Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.

[20] J. Fu, H. He, and Z. Ni, "Adaptive Dynamic Programming with Balanced Weights Seeking Strategy," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), IEEE Symposium Series on Computational Intelligence (SSCI)*, Paris, France, 2011.

[21] D. Liu, H. Javaherian, O. Kovalenko, and T. Huang, "Adaptive critic learning techniques for engine torque and air-fuel ratio control," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 988–993, 2008.

[22] D. Liu, Y. Zhang, and H. G. Zhang, "A self-learning call admission control scheme for CDMA cellular networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1219–1228, 2005.

[23] Y. Tang, H. He, Z. Ni, J. Wen, and X. Sui, "Reactive power control of grid-connected wind farm based on adaptive dynamic programming," *Neurocomputing*, vol. 125, pp. 125–133, Feb. 2014.

[24] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Trans. Sys. Man Cyber. Part B*, vol. 38, no. 4, pp. 1008–1013, 2008.

[25] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 943–949, 2008.

[26] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.

[27] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input contraints," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 2, pp. 425–436, 2007.

[28] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. on Neural Networks and Learning Systems*, in press.

[29] D. Liu, D. Wang, and H. Li, "Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach," *IEEE Trans. on Neural Networks and Learning Systems*, in press.

[30] X. Zhong, H. He, and D. V. Prokhorov, "Robust controller design of continuous-time nonlinear system using neural network," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013.

CHAPTER 4

Event-triggered ADP Control with Unknown Internal States

4.1 Introduction

So far, most ADP control designs are based on entire state measurements in the literature [1, 2, 3]. This is because ADP design needs to carefully evaluate the costs and benefits of the immediate action, as well as the choices which may be acted in the future [4], [5], [6, 7]. If the system feedback is imperfect or unreliable indicators of the underlying process, this evaluation will become difficult [8]. However, in many real-world applications, the likelihood to access the complete knowledge of system state is either infeasible or very difficult to obtain [9, 10]. In other words, the feedback can only represent parts of the system states in these situations. In order to achieve better performance, estimating or reconstructing the state variables needs to be considered. Over the past decades, partially observable processes have attracted significantly increasing attention from both the artificial intelligence and machine learning areas. One major idea of most existing methods is to obtain the belief state, which is a sufficient statistic of the complete information of system and is also updated after each observation [11], [12], [13]. However, intensive computational burden will be caused when we try to obtain the belief state, especially when the dimension of the system state increases (i.e., curse of dimensionality). In these years, new iterative algorithms were developed under the partially observable environment based on reinforcement learning approach [14], [15]. Many of these methods, however, were still based on parameters/probability and required solid mathematic background to apply. Recently, ADP has been applied in this field and achieved some promising results. In [16], both the policy iteration and value iteration were provided using only the input-output data to obtain an optimal controller. This idea is extended to a linear tracking problem for unknown discrete-time system in [17]. Only the reduced information of the system dynamics is used in their method. In

[18], [19], an observer was established based on neural networks to determine a mapping between the behavior of the system and the external influences.

Because of the integration of an observer, the computation of ADP control design increases. Generally, the observer-based ADP methods rely on the periodic transmitted data with the fixed sampling period. This may bring huge number of the transmitted data and cause subsequent tremendous computation. This disadvantage becomes severe when the computation bandwidth or sensor power resources are constrained. In recent years, the event-triggered control method [20, 21, 22, 23] is introduced in ADP design. Different from the traditional method, the event-triggered method only transmits the system data and updates the control law when a specific event is triggered. In this way, the transmission load and computation burden are significantly reduced. The authors in [24] for the first time online solved an event-triggered controller for a nonlinear system with guaranteed performance and without any linearizing process. In [25], a near optimal event-triggered condition of a nonlinear discrete-time system in affine form was provided. The authors extended this idea on the multi-input multi-output continuous-time system in [26] and provided the corresponding neural-network-based event-triggered condition.

In this chapter, a novel event-triggered ADP control method for the nonlinear continuous-time system with unknown internal states is proposed. In this situation, the measured input/output data can only represent parts of the system internal states. A neural-network-based observer is developed to recover the entire states from the system feedback. Then, a triggering condition is designed to make sure the control stability with the reduced information. A critic network is established to approximate the performance index and help calculate the control law. Note that, in this chapter, both the observer and the control law are updated aperiodically according to the triggering condition. This means the observer and the control law are updated only when an specific event is trig-

gered and held as constant otherwise. The stability analysis for the closed-loop system is presented based on the Lyapunov construct for both the continuous and the jump dynamics. Comparing with [27], the proposed method only uses the triggered samples to update the observer and the control law, which reduces the transmission load and computation burden. Comparing with the works in [19], [18], the proposed method can recover the details of what actually happened inside the partially observable dynamic processes.

4.2 Problem Statement

Consider the nonlinear continuous-time system given as

$$\begin{aligned} \dot{x}(t) =& f(x(t)) + g(x(t))u(t) \\ y(t) =& Cx(t) \end{aligned} \tag{41}$$

where $x(t) \in \mathbb{R}^n$ is the state vector with the initial state $x(0) = x_0$, $u(t) \in \mathbb{R}^m$ is the control input vector, $y(t) \in \mathbb{R}^p$ is the output vector, $f(x(t)) : \mathbb{R}^n \to \mathbb{R}^n$ and $g(x(t)) : \mathbb{R}^n \to \mathbb{R}^m$ are the unknown continuous-time state functions, and $f(0) = 0$. Assume that $f + gu$ is Lipschitz continuous on a set $\Omega \subseteq \mathbb{R}^n$ containing the origin. $C \in \mathbb{R}^{p*n}$ is the known output matrix.

Generally, the digital communication network is used to connect system, sensor, controller, and actuator in practical applications. Consider the limitation of the computation bandwidth or sensor power resources, an aperiodic updating and transmission rule for control action and system states is designed. In order to achieve this goal, a sampled-data system is introduced, which is characterized by a monotonically increasing sequence of sampling instants $\{\delta_j\}_{j=0}^{\infty}$, where $\delta_j < \delta_{j+1}$ for $j = 0, 1, 2, \cdots, \infty$. The time instant δ_j denotes the jth consecutive sampling instant. The output of the sample-data system is a sequence of the sampled states which can be denoted as

$$\hat{x}_j = x(\delta_j) \tag{42}$$

For simplicity, we assume that the sampled-data system has zero task delay.

Assumption 1 [27]: The nonlinear continuous-time system described in (41) is controllable and observable. Here, the system output, $y(t)$, is considered measured.

Therefore, a stabilizing controller can be guaranteed to be designed due to the controllability and the internal state can be ensured to be estimated from output measurement because of the observability. The control objective is to determine a feedback control law $u(t) = \mu(x(t))$ which minimizes the following infinite-horizon performance index

$$\begin{aligned} V(x_0) &= \int_0^\infty \Big(y^T(\tau) Q y(\tau) + u^T(\tau) R u(\tau) \Big) d\tau \\ &= \int_0^\infty U(y(\tau), u(\tau)) d\tau \end{aligned} \tag{43}$$

where $U(y(t), u(t)) = y^T(t) Q y(t) + u^T(t) R u(t)$ is the utility function with $U(0, 0) = 0$. Note that Q and R are symmetric positive definite matrices with appropriate dimensions. Here, the state-feedback control law is designed as $u(t) = \mu(\hat{x}_j, t)$, which maps the sampled state, rather than the continuous state in literature, onto a control vector. Therefore, the control signal $\mu(\hat{x}_j, t)$ is a piecewise constant function and consists of the control sequence $\{\mu(\hat{x}_j)\}_{j=0}^{\infty}$. In particular, $\{\mu(\hat{x}_j)\}_{j=0}^{\infty}$ becomes a continuous-time signal $\mu(\hat{x}_j, t)$ through a zero-order hold (ZOH).

Let us recall the performance index in the traditional ADP method (time-triggered case),

$$\begin{aligned} V(x_0) &= \int_0^\infty U(Cx(\tau), \mu(x(\tau))) d\tau \\ &= \int_0^t U(Cx(\tau), \mu(x(\tau))) d\tau + V(x(t)) \end{aligned} \tag{44}$$

If the performance index (44) is continuously differentiable, then after transformation, we obtain

$$\lim_{t \to 0} \left[V(x(t)) - V(x_0) \right] / t = - \lim_{t \to 0} \frac{1}{t} \int_0^t U(Cx(\tau), \mu(x(\tau))) d\tau \tag{45}$$

Therefore, the infinitesimal version of (44) is as

$$V_x^{*T}(f(x(t)) + g(x(t))\mu(x(t))) + U(Cx(t), \mu(x(t))) = 0 \tag{46}$$

where $V_x^* = \partial V^*(x(t))/\partial x(t)$ is the partial derivatives of the optimal performance index $V^*(x(t))$ with respect to $x(t)$.

Assume that the minimum of the left-hand side of (46) exists and is unique [28]. Therefore, the optimal control $\mu^*(x(t))$ satisfies the first-order necessary condition, which is given by the gradient of (46) with respect to $\mu(x(t))$. Hence, the optimal control law for the time-triggered case can be described as,

$$u^*(t) = \mu^*(x(t)) = -\frac{1}{2}R^{-1}g^T(x(t))V_x^* \tag{47}$$

In this event-triggered control design, the controller is only updated when an event is triggered. This means the controller is designed based on the sampled state $\hat{x}_j$ instead of the current state $x(t)$. Therefore, we obtain the event-triggered control law as

$$u^*(t) = \mu^*(\hat{x}_j, t) = -\frac{1}{2}R^{-1}g^T(\hat{x}_j)V_{\hat{x}_j}^* \tag{48}$$

where $V_{\hat{x}_j}^* = \partial V^*(\hat{x}_j)/\partial \hat{x}_j$. Note that, $\mu(\hat{x}_j)$ is used to represent $\mu(\hat{x}_j, t)$ in order to simplify the expression in the following presentation. By applying event-triggered control law (48) into (46), the event-triggered HJB equation can be obtained,

$$\begin{aligned} H(x(t), \mu^*(\hat{x}_j), V_x^*) =& V_x^{*T}(f(x(t)) - \frac{1}{2}g(x(t))g^T(\hat{x}_j)V_{\hat{x}_j}^*) \\ &+ \frac{1}{4}V_{\hat{x}_j}^{*T}g(\hat{x}_j)g^T(\hat{x}_j)V_{\hat{x}_j}^* + x^T(t)C^TQCx(t) \end{aligned} \tag{49}$$

By developing the event-triggered ADP method, the transmission load and computation burden can be significantly relaxed. However, we can observe that the system internal states $x(t)$, $\hat{x}_j$ are used in (48) and (49) to calculate the event-triggered controller and HJB equation. Since the knowledge of the system functions is completely unknown and the measured output can only represent parts of the system internal states,

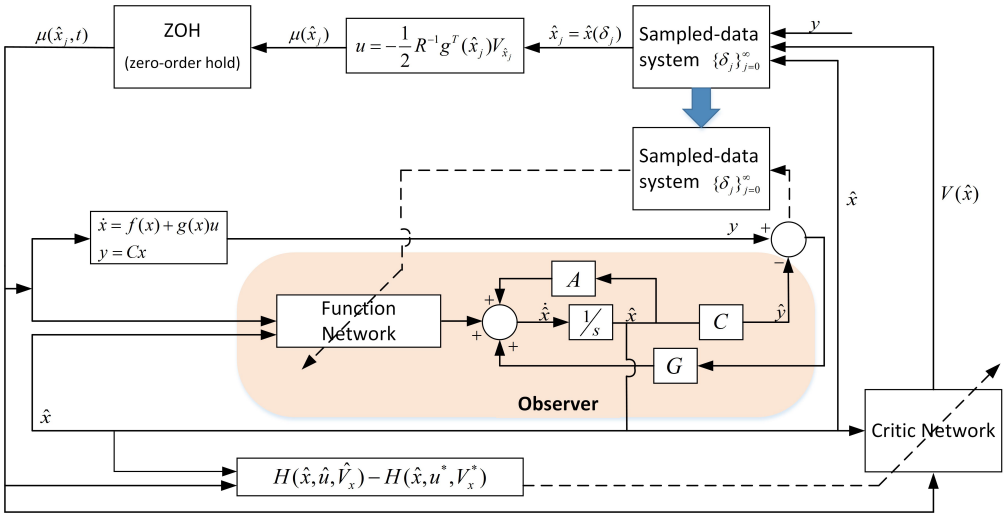

Figure 13. Block diagram of the nonlinear continuous-time system control with only the input-output data.

the existing ADP methods cannot be applied directly in this situation. In the next section, an event-triggered ADP control method using only the system input-output data will be provided. Note that, in order to simplify the presentation, the time index t is omitted in the following statement.

4.3 Event-triggered Controller Design Using Only the Input-Output Data

The general architecture of event-triggered ADP control using only the system input-output data is shown in Figure 13. First, because of the unavailability of the system internal state vectors and the system functions, a neural-network-based observer is designed to reconstruct both the state vector x and the control coefficient function $g(x)$ through an online manner. Therefore, the proposed observer design relaxes the requirement of an explicit identifier for $g(x)$ or an action network for $\mu(x)$. Then, the ADP framework is applied to approximate the performance index and calculate the optimal control vector. The critic network is established to estimate the performance index and it is trained online with a corresponding error term minimized overtime. Moreover, it is important to note that a sampled-data system is introduced with a sequence of sampling

instants $\{\delta_j\}_{j=0}^{\infty}$ for both the neural network observer and the controller. This means both the observer and the controller are updated only when an specific event is triggered. The corresponding triggering condition is also provided. Due to the limitation of the communication bandwith and sensor power resources, this can significantly reduce the huge number of the transmitted data and subsequently tremendous computation.

In the following part, I will explicitly present the event-triggered ADP design using only the system input and output data. Specifically, in the first subsection, the triggering condition is derived for the sampled-data system. The corresponding stability analysis is also provided. A neural-network-based observer is designed in the second subsection, so that the control scheme can be developed using only the input and the output data measured during the operation of the system. A proof is also provided in this subsection to guarantee the stability of the observer and the accuracy of its estimation during the continuous and the jump dynamics. In the third subsection, neural network techniques are used to implement the proposed method. The weights updating rules for the critic network are also provided. Finally, the stability of the close-loop system is demonstrated using the Lyapunov theory for both dynamics. It is shown that the system state and parameter estimations are proved bounded, even when the trigger occurs.

4.3.1 Event-triggered Regulator Design

Note that, since the internal state is unknown, an observer is designed to recover the system state. Therefore, the sampled states should be described as

$$\hat{x}_j = \hat{x}(\delta_j) \tag{50}$$

where $\hat{x}(\delta_j)$ is the estimated state at the sampled instants.

Now, define the gap function for $\forall t \in [\delta_j, \delta_{j+1})$as

$$e_{y_j}(t) = C\hat{x}_j - y(t) \tag{51}$$

which is the difference between the term $C\hat{x}_j$ and the current system output.

Assumption 2: The controller is Lipschitz continuous with respect to the gap,

$$\|\mu(x(t)) - \mu(\hat{x}_j)\| \leq L\|e_{x_j}\| \tag{52}$$

where L is a positive real constant, and $e_{x_j} = \hat{x}_j - x(t)$.

Theorem 1: If there exists a positive definite function $V(x)$ that satisfies the HJB equation (49) with $V(0) = 0$, and the control law is given in (48) with the triggering condition

$$\|e_{y_j}\|^2 \leq \frac{(1-\alpha^2)\underline{\lambda}(Q)\|C\|^2\|y\|^2 + \|C\|^2\|r^T\mu(x_j)\|^2}{L^2\|r\|^2} \tag{53}$$

then the close-loop system can be asymptotically stabilized, where $\alpha \in (0,1)$ is the designed parameter.

Proof: With the event-triggered control law (48), the orbital derivative of $V^*(x)$ along the system trajectory can be given as

$$\begin{aligned}\dot{V}^*(x) =& \Big(\frac{\partial V^*(x)}{\partial x}\Big)^T \dot{x} \\ =& V_x^* f(x) + V_x^{*T} g(x)\mu^*(\hat{x}_j)\end{aligned} \tag{54}$$

Here, consider the optimal control law and HJB equation in the traditional ADP method as

$$u^* = \mu^*(x) = -\frac{1}{2}R^{-1}g^T(x)V_x^* \tag{55}$$

and

$$V_x^{*T} f(x) - \frac{1}{4}V_x^{*T} g(x)R^{-1}g^T(x)V_x^* + y^T Q y = 0. \tag{56}$$

Therefore, we have

$$g^T(x)V_x^* = -2R\mu^*(x) \tag{57}$$

$$V_x^{*T} f(x) = \frac{1}{4}V_x^{*T} g(x)R^{-1}g^T(x)V_x^* - y^T Q y \tag{58}$$

Substitute (57) and (58) into (54), we obtain

$$\begin{aligned}\dot{V}^*(x) =& \frac{1}{4}V_x^{*T}g(x)R^{-1}g^T(x)V_x^* - y^TQy - 2\mu^{*T}(x)R\mu^*(\hat{x}_j) \\ =& \mu^{*T}(x)R\mu^*(x) - 2\mu^{*T}(x)R\mu^*(\hat{x}_j) - y^TQy\end{aligned} \tag{59}$$

Since R is a symmetric positive definite matrix, we can describe R as $R = r \cdot r^T$. Therefore, we have

$$\mu^{*T}(x)R\mu^*(x) - 2\mu^{*T}(x)R\mu^*(\hat{x}_j) = \|r^T\mu^*(x) - r^T\mu^*(\hat{x}_j)\|^2 - \|r^T\mu^*(\hat{x}_j)\|^2 \tag{60}$$

By using the Lipschitz condition in Assumption 2, we can write

$$\begin{aligned}\dot{V}^*(x) =& \|r^T\mu^*(x) - r^T\mu^*(\hat{x}_j)\|^2 - \|r^T\mu^*(\hat{x}_j)\|^2 - y^TQy \\ \leq& - \|r^T\mu^*(\hat{x}_j)\|^2 + L^2\|r\|^2\|e_{x_j}\|^2 - \underline{\lambda}(Q)\|y\|^2 \\ =& - \alpha^2\underline{\lambda}(Q)\|y\|^2 + \Big[- (1-\alpha^2)\underline{\lambda}(Q)\|y\|^2 + L^2\|r\|^2\|e_{x_j}\|^2 - \|r^T\mu^*(\hat{x}_j)\|^2\Big]\end{aligned} \tag{61}$$

We know when the following inequality is satisfied,

$$\|e_{x_j}\|^2 \leq \frac{(1-\alpha^2)\underline{\lambda}(Q)\|y\|^2 + \|r^T\mu(\hat{x}_j)\|^2}{L^2\|r\|^2} \tag{62}$$

we have $\dot{V}^*(x) < 0$.

Due to the unavailability of the current internal state, we obtain an equivalent condition (53) from (51). This is to say, when (53) is satisfied, we have $\dot{V}^*(x) < 0$. Thus, in this way, $u^* = \mu^*(\hat{x}_j)$ can asymptotically stabilize the nonlinear continuous-time system (41). The conclusion holds. ■

It can be seen that the controller is guaranteed stable with the event-triggered sample data. The sampled-data system will continuously monitor the triggering condition (53). When a violation is about to occur, the sampled-data system will be triggered to sample the estimated system state, and according to the new sampled data, both the observer and the controller will be updated again.

4.3.2 Neural-network-based Observer Design

In this subsection, a neural-network-based observer is established to reconstruct the system state x and the control coefficient function $g(x)$. Consider system (41) with the event-triggered control law $\mu(\hat{x}_j)$. Choose a Hurwitz matrix A, such that the pair (C, A) is observable. The system dynamics (41) can be reformulated as

$$\begin{aligned} \dot{x} =& Ax + F_A(x) + g(x)\mu(\hat{x}_j) \\ y =& Cx \end{aligned} \tag{63}$$

where $F_A(x) = f(x) - Ax$. In order to reconstruct the state, the nonlinearity of the system should be identified. Since x is restricted to a compact set of $x \in \mathbb{R}^n$, the unknown system function can be described as a multilayer neural network with sufficiently large number of hidden layer neurons [29], then

$$\begin{aligned} F_A(x) + g(x)\mu(\hat{x}_j) =& \omega_{o2F}^{*T}\Phi_F(x) + \omega_{o2g}^{*T}\Phi_g(x)\mu(\hat{x}_j) + \varepsilon_F(x) + \varepsilon_g(x)\mu(\hat{x}_j) \\ =& [\omega_{o2F}^{*T}, \omega_{o2g}^{*T}] \begin{bmatrix} \Phi_F(x) & 0 \\ 0 & \Phi_g(x) \end{bmatrix} \begin{bmatrix} 1 \\ \mu(\hat{x}_j) \end{bmatrix} \\ & + [\varepsilon_F(x), \varepsilon_g(x)] \begin{bmatrix} 1 \\ \mu(\hat{x}_j) \end{bmatrix} \\ =& \omega_{o2}^{*T}\Phi(x) + \varepsilon(x) \end{aligned} \tag{64}$$

where ω_{o2}^* is the ideal weights of the neural network output layer, $\|\varepsilon(x)\| \leq \varepsilon_M$ is the bounded neural network approximation error, $\Phi(\cdot)$ is the bounded sigmoid function that can be expressed as

$$\|\Phi(\cdot)\| = \left\| \frac{1 - e^{-(\cdot)}}{1 + e^{-(\cdot)}} \right\| \leq \Phi_M \tag{65}$$

It is assumed that the ideal weights are bounded as $\|\omega_{o2}^*\| \leq \omega_{o2M}$. Moreover, we have

$$\omega_{o2}^* = [\omega_{o2F}^*, \omega_{o2g}^*] \tag{66}$$

$$\Phi(x) = \begin{bmatrix} \Phi_F(x) & 0 \\ 0 & \Phi_g(x) \end{bmatrix} \begin{bmatrix} 1 \\ \mu(\hat{x}_j) \end{bmatrix} \tag{67}$$

$$\varepsilon(x) = [\varepsilon_F(x), \varepsilon_g(x)] \begin{bmatrix} 1 \\ \mu(\hat{x}_j) \end{bmatrix} \tag{68}$$

Hence, the system states can be identified by updating the corresponding neural network weights. Since the ideal weights ω_{o2}^* are unknown, a neural network, which is called the function network in this chapter, is established to identify the nonlinearity by using the current estimates $\hat{\omega}_{o2}$ of the ideal weights ω_{o2}^*,

$$\hat{F}_A(\hat{x}) + \hat{g}(\hat{x})\mu(\hat{x}_j) = \hat{\omega}_{o2}^T \Phi(\hat{x}) \tag{69}$$

It is important to note that in order to save the resource, the function network weights are only updated when an event is triggered, i.e.,

$$\hat{\omega}_{o2j} = \hat{\omega}_{o2}(\delta_j) \tag{70}$$

Then, (69) becomes

$$\hat{F}_A(\hat{x}) + \hat{g}(x)\mu(\hat{x}_j) = \hat{\omega}_{o2j}^T \Phi(\hat{x}_j) \tag{71}$$

Hence, I design the following neural-network-based observer which is assumed to be of the Luenberger like structure

$$\begin{aligned} \dot{\hat{x}} =& A\hat{x} + \hat{\omega}_{o2j}^T \Phi(\hat{x}_j) + G(y - \hat{y}) \\ \hat{y} =& C\hat{x} \end{aligned} \tag{72}$$

where $\hat{x}$ and $\hat{y}$ are the estimated state and output of the observer, respectively, $\hat{x}_j$ is the estimated sampled state, and $G \in \mathbb{R}^{n*m}$ is the observer gain. Here, $\Phi(\hat{x}_j) = \Phi(\omega_{o1}\hat{X}_{oj})$, in which $\hat{X}_{oj} = [\hat{x}_j, \mu(\hat{x}_j)]$ is the input of the function network, and ω_{o1} is the weights of the function network hidden layer. Now, define the state estimation error as

$$\begin{aligned} \dot{\tilde{x}} =& \dot{x} - \dot{\hat{x}} \\ =& Ax + \omega_{o2}^{*T}\Phi(x) - A\hat{x} - \hat{\omega}_{o2j}^T \Phi(\hat{x}_j) - G(y - \tilde{y}) + \varepsilon(x) \end{aligned} \tag{73}$$

By adding and subtracting $\omega_{o2}^{*T}\Phi(\hat{x}_j)$ from (73), such error dynamics become

$$\dot{\tilde{x}} = A_c\tilde{x} + \tilde{\omega}_{o2j}^T \Phi(\hat{x}_j) + \xi(x) \tag{74}$$

where $\tilde{\omega}_{o2j} = \omega^*_{o2} - \hat{\omega}_{o2j}$ is the neural network estimation error, $A_c = A - GC$ is a Hurwitz matrix, and $\xi(x) = \omega^{*T}_{o2}[\Phi(x) - \Phi(\hat{x}_j)] + \varepsilon(x)$ is a bounded disturbance term. This means, $\|\xi(x)\| \leq \xi_M$ for some positive constant, due to the boundedness of the sigmoid function and the ideal neural network weights ω^*_{o2}.

Note that, in this study, the input-to-hidden layer weights ω_{o1} are randomly chosen and kept constantly during the training process. Therefore, the goal now should be to find the updating rule for the hidden-to-output layer weights $\hat{\omega}_{o2j}$. Adjusting the weights of the function network is to minimize the squared error

$$E_o = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}\tilde{y}^2 \tag{75}$$

where $\tilde{y} = y - \hat{y}$. Since the updating law for the observer will have an aperiodic nature, it has to be updated only at the trigger instants and held constant otherwise. Therefore, it can be described as the following updating laws: when an event is not triggered, we have

$$\dot{\hat{\omega}}_{o2j} = 0, \quad \text{for } \delta_{j-1} \leq t < \delta_j \tag{76}$$

and when an event is triggered, the jump equation to calculate $\hat{\omega}_{o2j}$ is given by

$$\begin{aligned} \hat{\omega}^+_{o2j} =& \hat{\omega}_{o2j} - \beta_o \frac{\partial E_o}{\partial \hat{\omega}_{o2j}} - \rho\|\tilde{y}\|\hat{\omega}_{o2j} \\ =& \hat{\omega}_{o2j} - \beta_o \frac{\partial E_o}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial \hat{\omega}_{o2j}} - \rho\|\tilde{y}\|\hat{\omega}_{o2j} \quad \text{for } t = \delta_j \end{aligned} \tag{77}$$

where $\beta_o > 0$ is the learning rate of the function network and $\rho > 0$ is a small positive number. Note that the second term in (77) is the backpropagation term and the third term is the e-modification term for incorporating damping. We have $\frac{\partial E_o}{\partial \tilde{y}} = \tilde{y}$ and $\frac{\partial \tilde{y}}{\partial \hat{x}} = C^T$ according to (75) and (63), respectively. The updating rule can thus be achieved for solving the gradient $\frac{\partial \hat{x}}{\partial \hat{\omega}_{o2j}}$. To solve this problem, we apply the static approximation of the gradient by setting $\dot{\hat{x}} = 0$ in (72). Then, after transformation, we obtain

$$\frac{\partial \hat{x}}{\partial \omega_{o2j}} = -A_c^{-T}\Phi(\hat{x}_j) \tag{78}$$

Hence, we have the updating rule for the function network at the trigger instants as

$$\hat{\omega}_{o2j}^{+} = \hat{\omega}_{o2j} - \beta_o(\tilde{y}^T C A_c^{-1})^T \Phi(\hat{x}_j) - \rho\|\tilde{y}\|\hat{\omega}_{o2j} \quad \text{for } t = \delta_j \tag{79}$$

In order to guarantee the stability of the neural-network-based observer and the accuracy of the estimation, the boundedness of the observer error should be provided for both the continuous and the jump dynamics.

Theorem 2: Consider the nonlinear continuous-time system given by (41) with the event-triggered neural-network-based observer given by (72). If the tuning laws for the function network of the observer is provided as (76) and (79) for different time instants, then the state estimation error $\tilde{x}$ and weight estimation errors $\tilde{\omega}_{o2j} = \omega_{o2j}^{*} - \hat{\omega}_{o2j}$ are uniformly ultimately bounded (UUB).

Proof: Since the observer is updated only when the event is triggered, we have to consider both the continuous and the jump dynamics separately. Initially, we will consider the following Lyapunov function L_o

$$L_o = \frac{1}{2}\tilde{x}^T P \tilde{x} + \frac{1}{2} tr(\tilde{\omega}_{o2j}^T \tilde{\omega}_{o2j}) \tag{80}$$

where $\tilde{x}$ is the state estimation error given by (74) and $\tilde{\omega}_{o2j}$ is the weight estimation error. P is a positive definite matrix that satisfies

$$A_c^T P + P A_c = -M \tag{81}$$

where M is a positive definite matrix.

For the continuous dynamics of the observer model, by taking the time derivative of (80) with respect to the close-loop system trajectories, the second term has a zero derivative due to the function network continuous dynamics (76). Therefore,

$$\begin{aligned} \dot{L}_o =& \frac{1}{2}\dot{\tilde{x}}^T P \tilde{x} + \frac{1}{2}\tilde{x}^T P \dot{\tilde{x}} \\ =& \frac{1}{2}\Big(A_c\tilde{x} + \tilde{\omega}_{o2j}^T \Phi(\hat{x}_j) + \xi(x)\Big)^T P\tilde{x} + \frac{1}{2}\tilde{x}^T P\Big(A_c\tilde{x} + \tilde{\omega}_{o2j}^T \Phi(\hat{x}_j) + \xi(x)\Big) \end{aligned} \tag{82}$$

By using some polynomial adjustments and (81), equation (82) can be rewritten as

$$\begin{aligned}\dot{L}_o &= -\frac{1}{2}\tilde{x}^T M\tilde{x} + \tilde{x}^T P\big(\tilde{\omega}_{o2j}^T\Phi(\hat{x}_j) + \xi(x)\big)\\ &\leq -\frac{1}{2}\underline{\lambda}(M)\|\tilde{x}\|^2 + \|\tilde{x}\|\|P\|\big(\|\tilde{\omega}_{o2}\|\Phi_M + \xi_M\big)\\ &\leq -\frac{1}{2}\underline{\lambda}(M)\|\tilde{x}\|^2 + \big(2\omega_{oM}\Phi_M\|P\| + \|P\|\xi_M\big)\|\tilde{x}\|\end{aligned} \tag{83}$$

where $\underline{\lambda}(M)$ is the minimal eigenvalue of M. Hence, in order to guarantee the negativeness of the time derivative $\dot{L}_o$ at the continuous dynamics, the following condition on the state estimation error should hold,

$$\|\tilde{x}\| \geq \frac{4\omega_M\Phi_M\|P\| + 2\xi_M\|P\|}{\underline{\lambda}(M)} = d. \tag{84}$$

According to the Lyapunov extension theorem, as long as condition (84) is satisfied, it demonstrates that the state and the weights estimation errors are UUB.

Note that $\dot{L}_o$ for continuous dynamics is negative definite under the condition (84), which means $\tilde{x}$ is UUB outside the ball with radius d described as $X = \{\tilde{x}|\|\tilde{x}\| > d\}$. The size of the estimation error bound d can be kept arbitrarily small by proper selection of the parameters.

Next we have to consider the jump dynamics. The function network weights are updated at these instants. For that reason, we consider the following form

$$\begin{aligned}\Delta L_o =& \frac{1}{2}(\tilde{x}^T)^+P\tilde{x}^+ - \frac{1}{2}\tilde{x}^T P\tilde{x}\\ &+ \frac{1}{2}tr\big((\tilde{\omega}_{o2j}^T)^+\tilde{\omega}_{o2j}^+\big) - \frac{1}{2}tr\big(\tilde{\omega}_{o2j}^T\tilde{\omega}_{o2j}\big), \quad t = \delta_j\end{aligned} \tag{85}$$

Since we have proved that the state estimation error is asymptotically stable, there exists

$$\frac{1}{2}(\tilde{x}^T)^+P\tilde{x}^+ \leq \frac{1}{2}\tilde{x}^T P\tilde{x} \tag{86}$$

Therefore, the problem becomes to find a bound for the following term,

$$\Delta L_{o1}(\tilde{\omega}_{o2j}) = \frac{1}{2}tr\big((\tilde{\omega}_{o2j}^T)^+\tilde{\omega}_{o2j}^+\big) - \frac{1}{2}tr\big(\tilde{\omega}_{o2j}^T\tilde{\omega}_{o2j}\big), \quad t = \delta_j \tag{87}$$

Consider (79), we obtain,

$$\begin{aligned}\tilde{\omega}_{o2j}^{+} =&\omega_{o2j}^{*} - \hat{\omega}_{o2j}^{+} \\ =&\tilde{\omega}_{o2j} + \beta_o(\tilde{y}^T C A_c^{-1})^T \Phi(\hat{x}_j) + \rho\|\tilde{y}\|\hat{\omega}_{o2j}\end{aligned} \tag{88}$$

Substituting (88) into (87) and after some mathematical manipulation, the first difference $\Delta L_{o1}(\tilde{\omega}_{o2j})$ becomes

$$\begin{aligned}\Delta L_{o1}(\tilde{\omega}_{o2j}) =&tr\Big(\tilde{\omega}_{o2j}^T\big(\beta_o(\tilde{y}^T C A_c^{-1})^T \Phi(\hat{x}_j) \\ &+ \rho\|\tilde{y}\|\hat{\omega}_{o2j}\big)\Big) + \Big\|\beta_o(\tilde{y}^T C A_c^{-1})^T \Phi(\hat{x}_j) + \rho\|\tilde{y}\|\hat{\omega}_{o2j}\Big\|^2 \\ =&tr\Big(\tilde{\omega}_{o2j}^T \beta_o A_c^{-T} C^T \tilde{y}\Phi(\hat{x}_j) + \rho\|\tilde{y}\|\tilde{\omega}_{o2j}^T \omega_{o2j}^{*} - \rho\|\tilde{y}\|\tilde{\omega}_{o2j}^T\tilde{\omega}_{o2j}\Big) \\ &+ \|\beta_o A_c^{-T} C^T \tilde{y}\Phi(\hat{x}_j)\|^2 + 2\Phi^T(\hat{x}_j)\tilde{y}^T(\beta_o A_c^{-T} C^T)^T \rho\|\tilde{y}\|\hat{\omega}_{o2j} \\ &+ \rho^2\|\tilde{y}\|^2\hat{\omega}_{o2j}^T\hat{\omega}_{o2j} \\ \leq& -\rho\|C\|\|\tilde{x}\|\|\tilde{\omega}_{o2j}\|^2 + \|m\|\|\tilde{x}\|\|\tilde{\omega}_{o2j}\|\Phi_M + \rho\|C\|\|\tilde{x}\|\|\tilde{\omega}_{o2j}\|\omega_{oM} \\ &+ \|m\|^2\|\tilde{x}\|^2\Phi_M^2 + 2\rho\|C\|\|m\|\|\tilde{x}\|^2\Phi_M\|\hat{\omega}_{o2j}\| + \rho^2\|C\|^2\|\tilde{x}\|^2\omega_{oM}^2\end{aligned} \tag{89}$$

where $m = \beta_o A_c^{-T} C^T C$. By completing the square of $\|\tilde{\omega}_{o2j}\|$, formula (89) becomes

$$\begin{aligned}\Delta L_{o1}(\tilde{\omega}_{o2j}) \leq& -\frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\omega_M - \|\tilde{\omega}_{o2j}\|\Big)^2\|\tilde{x}\| - \Big(\rho - \frac{1}{2}\Big)\|\tilde{\omega}_{o2j}\|^2\|\tilde{x}\| \\ &+ \frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\|\omega_{oM}\|\Big)^2\|\tilde{x}\| \\ &+ \Big(\|m\|^2\Phi_M^2 + 2\rho\|m\|\|C\|\|\omega_M\|^2\Phi_M + \rho^2\|C\|^2\omega_{oM}^2\Big)\|\tilde{x}\|^2\end{aligned} \tag{90}$$

Since $\|\tilde{x}\|$ is guaranteed positive, then $\Delta L_o(\tilde{\omega}_{o2j}) \leq 0$ is equivalent to the following condition holding,

$$\begin{aligned}&-\frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\omega_M - \|\tilde{\omega}_{o2j}\|\Big)^2 - \Big(\rho - \frac{1}{2}\Big)\|\tilde{\omega}_{o2j}\|^2 \\ &+ \frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\|\omega_{oM}\|\Big)^2 + \Big(\|m\|^2\Phi_M^2 + 2\rho\|m\|\|C\|\|\omega_M\|^2\Phi_M \\ &+ \rho^2\|C\|^2\omega_{oM}^2\Big)\|\tilde{x}\| \leq 0\end{aligned} \tag{91}$$

By defining

$$\begin{aligned}\gamma^2 = \frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\|\omega_{oM}\|\Big)^2 + \Big(&\|m\|^2\Phi_M^2 \\ &+ 2\rho\|m\|\|C\|\|\omega_M\|^2\Phi_M + \rho^2\|C\|^2\omega_{oM}^2\Big)\|\tilde{x}\|\end{aligned} \tag{92}$$

condition (91) becomes

$$-\frac{1}{2}\Big(\|m\|\Phi_M + \rho\|C\|\omega_M - \|\tilde{\omega}_{o2j}\|\Big)^2 - \big(\rho - \frac{1}{2}\big)\|\tilde{\omega}_{o2j}\|^2 + \gamma^2 \leq 0 \tag{93}$$

Note that because the boundedness of the state estimation error has been proved, there exists a bound for γ^2. Therefore, we can prove that the jump dynamics are UUB as long as the following conditions satisfied

$$\rho > \frac{1}{2} \tag{94}$$

$$\|\tilde{\omega}_{o2j}\| \geq \sqrt{\frac{\gamma^2}{\left(\rho - \frac{1}{2}\right)}} \tag{95}$$

Hence, the system states estimation error and the neural network weight estimation errors are UUB in both the continuous and the jump dynamics. This completes the proof. ■

4.3.3 Optimal Event-triggered Control Scheme Design

Neural network technique is applied in this subsection to implement the proposed event-triggered ADP method. A critic network is built to approximate the performance index which can be formulated as

$$V^*(x) = \omega_{c2}^{*T}\Phi(m(x)) + \varepsilon_c(x) \tag{96}$$

where ω_{c2}^* is the optimal weights between the hidden and the output layer of the critic network, $m(x) = \omega_{c1}^{*T}X_c$ to which ω_{c1}^* is the optimal input-to-hidden layer weights, $X_c = [x^T, \mu^T(x)]^T$, and $\|\varepsilon_c(x)\| \leq \varepsilon_{cM}$ is the bounded critic network error.

According to (96), the performance index $V^*(x)$ in the event-triggered control scheme can be approximated as

$$\hat{V}(\hat{x}) = \hat{\omega}_{c2}^T\Phi(m(\hat{x})) \tag{97}$$

where $\hat{V}(\hat{x})$ represent the estimated performance index, $\hat{\omega}_{c2}^T$ is the approximated hidden-to-output layer weights of the critic network, and $m(\hat{x}) = \hat{\omega}_{c1}^T \hat{X}_c$ to which $\hat{\omega}_{c1}$ is the estimated input-to-hidden layer weights of critic network and $\hat{X}_c = [\hat{x}, \mu(\hat{x}_j)]$ is the input of the critic network. We fix the input-to-hidden layer weights as ω_{c1}, which are chosen randomly at initial. Therefore, only the hidden-to-output layer weights $\hat{\omega}_{c2}$ need to be updated.

Define the error function for the critic network as

$$\begin{aligned} e_c =& H(\hat{x}, \mu(\hat{x}_j), \hat{V}_x) - H(x, u^*, V_x^*)) \\ =& \Big((\frac{\partial \Phi(m(\hat{x}))}{\partial \hat{x}})^T \hat{\omega}_{c2} \Big)^T \dot{\hat{x}} + U(\hat{x}, \mu(\hat{x}_j)) \end{aligned} \tag{98}$$

We know that $H(x, u^*, V_x^*)) = 0$ from (49). Adjusting the weights of the critic network is to minimize the objective function

$$E_c = \frac{1}{2} e_c^2 \tag{99}$$

Therefore, the hidden-to-output layer weights of the critic network can be updated as

$$\begin{aligned} \dot{\hat{\omega}}_{c2} =& -\beta_c \frac{\partial E_c}{\partial \hat{\omega}_{c2}} = -\beta_c \frac{\partial E_c}{\partial e_c} \frac{\partial e_c}{\partial \hat{\omega}_{c2}} \\ =& -\beta_c \frac{\kappa}{(\kappa^T \kappa + 1)^2} (\hat{\omega}_{c2}^T \kappa + U(\hat{x}, \mu(\hat{x}_j)))^2 \end{aligned} \tag{100}$$

where $\kappa = (\frac{\partial \Phi(m(\hat{x}))}{\partial \hat{x}})^T \dot{\hat{x}}$, and $\beta_c > 0$ is the learning rate of the critic network.

The control law is only updated when the triggering condition (53) is violated. Since the design of the neural-network-based observer can reconstruct both the system internal state and the control coefficient function, the control law can be directly calculated as

$$\mu(\hat{x}_j) = -\frac{1}{2} R^{-1} g^T(\hat{x}_j) \hat{V}_{\hat{x}_j} \tag{101}$$

where $\hat{V}_{\hat{x}_j}$ is the partial derivative of the estimated performance index with respect to

the sampled state $\hat{x}_j$. According to (97), $\hat{V}_{\hat{x}_j}$ can be formulated as

$$\begin{aligned}\hat{V}_{\hat{x}_j} =& \frac{\partial \hat{V}(\hat{x}_j)}{\partial \hat{x}_j} \\ =& \frac{\partial \hat{V}(\hat{x}_j)}{\partial \Phi(m(\hat{x}_j))} \frac{\partial \Phi(m(\hat{x}_j))}{\partial m(x(\hat{x}_j))} \frac{\partial m(x(\hat{x}_j))}{\partial \hat{x}_j} \\ =& \frac{1}{2} \hat{\omega}_{c2j}^T (1 - \Phi^2(m(\hat{x}_j))) \omega_{c1(\hat{x}_j)}\end{aligned} \tag{102}$$

to which $\omega_{c1}(\hat{x}_j)$ is the fixed weights of $\hat{x}$ component for the input to the hidden layer of the critic network at the jump instant δ_j.

Also, considering (69), $g(\hat{x}_j)$ can be described by

$$\begin{aligned}g(\hat{x}_j) =& \frac{\partial (F_A(\hat{x}_j) + g(\hat{x}_j)\mu(\hat{x}_j))}{\partial \mu(\hat{x}_j)} \\ =& \frac{\partial (F_A(\hat{x}_j) + g(\hat{x}_j)\mu(\hat{x}_j))}{\partial \Phi(\hat{x}_j)} \frac{\partial \Phi(\hat{x}_j)}{\partial \mu(\hat{x}_j)} \\ =& \frac{1}{2} \hat{\omega}_{o2j}^T (1 - \Phi^2(\hat{x}_j)) \omega_{o1}(\mu(\hat{x}_j))\end{aligned} \tag{103}$$

where $\omega_{o1}(\mu(\hat{x}_j))$ is the input-to-hidden layer weights of $\mu(\hat{x}_j)$ component for the function network at jump instant δ_j. Because the control law is only updated when the triggering condition (53) is violated, we then have the following description,

$$u(t) = \begin{cases} \mu(\hat{x}_{j-1}), & \text{Event is not triggered, } \delta_{j-1} \le t < \delta_j \\ -\frac{1}{2} R^{-1} g^T(\hat{x}_j) \hat{V}_{\hat{x}_j}, & \text{Event is triggered, } t = \delta_j \end{cases} \tag{104}$$

The algorithm of the proposed event-triggered ADP control using the measurable input-output data is provided in Algorithm 3.

4.3.4 Stability Analysis of the closed-loop system

In this subsection, the stability analysis for the close-loop system will be investigated. A Lyapunov function candidate is considered as a combination of the Lyapunov functions for the neural-network-based observer and the designed control law. Both of them have two dynamics. The following theorem provides the stability of the whole system.

Algorithm 2 Event-triggered ADP control design Using Only the Measurable Input-Output Data.

Set $i = 0$, $j = 0$, $\hat{x}_0 = x_0$
Calculate $\mu(\hat{x}_j) = -\frac{1}{2}R^{-1}g^T(\hat{x}_j)\hat{V}_{\hat{x}_j}$
Initialize all the neural network weights
for all $i < N_{run}$ **do**
  State estimation:
  $\dot{\hat{x}} = A\hat{x} + \hat{\omega}_{o2j}^T\Phi(\hat{x}_j) + G(y - C\hat{x})$
  Policy evaluation:
  $V(\hat{x}) = \min_{\mu(\hat{x}_j)} \int_0^\infty U(\hat{x}(\tau), \mu(\hat{x}_j))d\tau$
  if $\hat{x}_j - \hat{x} = \hat{e}_j > e_T$ **then**
    Set $j = j + 1$, $\hat{x}_j = \hat{x}$
    Update $\hat{\omega}_{o2j}$ according to (79)
    Update $\mu(\hat{x}_j) = \arg\min_{\mu(\hat{x}_j)}\{V(\hat{x}_j)\}$
  end if
  Update system information $\dot{x} = F(x, \mu(\hat{x}_j));\ y = Cx$
  Set $i = i + 1$
end for

Theorem 3: Consider the nonlinear continuous-time system (41) with the event-triggered observer (72) and control law (104). The tuning laws for the impulsive observer and the continuous critic network are provided by (76), (79) and (100), respectively. Then, the system state x, sampled state $\hat{x}_j$, observer error $\tilde{x}$, function network weights estimation error $\tilde{\omega}_{o2}$, and the critic network weights estimation error $\tilde{\omega}_{c2}$ are all UUB given the following triggering condition:

$$\|e_{y_j}\|^2 \leq \frac{(1-\alpha^2)\underline{\lambda}(Q)\|C\|^2\|y\|^2 + \|C\|^2\|r^T\mu(x_j)\|^2}{L^2\|r\|^2} \tag{105}$$

where $\alpha \in (0, 1)$.

Proof: The proof of the boundedness is carried out in two parts, which are the continuous and the jump dynamics, respectively. The objective is to prove that both dynamics of the impulsive close-loop model are UUB. First, let us consider the following

Lyapunov function,

$$\begin{aligned} L_{cl} =& \frac{1}{2}\tilde{x}^T P\tilde{x} + \frac{1}{2}tr(\tilde{\omega}_{o2}^T\tilde{\omega}_{o2}) + V^*(x) + V^*(\hat{x}_j) + \frac{\beta_c^{-1}}{2}tr(\tilde{\omega}_{c2}^T\tilde{\omega}_{c2}) \\ =& L_o + L_c, \quad t \in (\delta_j, \delta_{j+1}] \end{aligned} \tag{106}$$

where

$$L_o = \frac{1}{2}\tilde{x}^T P\tilde{x} + \frac{1}{2}tr(\tilde{\omega}_{o2}^T\tilde{\omega}_{o2}) \tag{107}$$

$$L_c = V^*(x) + V^*(\hat{x}_j) + \frac{\beta_c^{-1}}{2}tr(\tilde{\omega}_{c2}^T\tilde{\omega}_{c2}) \tag{108}$$

and $V^*(x)$ and $V^*(\hat{x}_j)$ are the optimal performance index for the continuous and event-triggered sampled system.

For the continuous dynamics of the impulsive model, we take the time derivative of (106). $\dot{L}_o$ is provided in (83). Now $\dot{L}_c$ needs to be considered. Note that the second term in (108) has a zero derivative. Hence, we obtain,

$$\dot{L}_c = \frac{\partial V^{*T}(x)}{\partial x}\dot{x} + \beta_c^{-1}tr(\tilde{\omega}_{c2}^T\dot{\tilde{\omega}}_{c2}) \tag{109}$$

where $\tilde{\omega}_{c2} = \omega_{c2}^* - \hat{\omega}_{c2}$, and

$$\begin{aligned} \dot{\tilde{\omega}}_{c2} =& \beta_c\frac{\kappa}{(\kappa^T\kappa+1)^2}\left(\hat{\omega}_{c2}^T\kappa + U(\hat{x}, \mu(\hat{x}_j))\right)^2 \\ =& -\beta_c\frac{\kappa\kappa^T}{(\kappa^T\kappa+1)^2}\tilde{\omega}_{c2} + \beta_c\frac{\kappa}{(\kappa^T\kappa+1)^2}\left(\kappa^T\omega_{c2}^* + U(\hat{x}, \mu(\hat{x}_j))\right) \\ =& -\beta_c\frac{\kappa\kappa^T}{(\kappa^T\kappa+1)^2}\tilde{\omega}_{c2} + \beta_c\frac{\kappa}{(\kappa^T\kappa+1)^2}\sigma_c \end{aligned} \tag{110}$$

where $\sigma_c = -\frac{\partial\varepsilon_c}{\partial\hat{x}}\dot{\hat{x}}$

Now, we will consider the following two terms separately,

$$\dot{L}_{c1}(V^*) = \frac{\partial V^{*T}(x)}{\partial x}\dot{x} \tag{111}$$

$$\dot{L}_{c2}(\tilde{\omega}_{c2}) = \beta_c^{-1}tr(\tilde{\omega}_{c2}^T\dot{\tilde{\omega}}_{c2}) \tag{112}$$

Then, (111) can be rewritten as

$$\begin{aligned}\dot{L}_{c1}(V^*) =& \frac{\partial V^{*T}(x)}{\partial x}\big(f(x)+g(x)\mu(\hat{x}_j)\big)\\ =& \frac{\partial V^{*T}(x)}{\partial x}f(x)+\frac{\partial V^{*T}(x)}{\partial x}g(x)\mu(\hat{x}_j)\end{aligned} \tag{113}$$

Consider equations (57) and (58), we obtain

$$\begin{aligned}\dot{L}_{c1}(V^*) =& \frac{1}{4}V_x^{*T}g(x)R^{-1}g^T(x)V_x^* - y^TQy - 2\mu^{*T}(x)R\mu^*(\hat{x}_j)\\ =& \mu^{*T}(x)R\mu^*(x) - 2\mu^{*T}(x)R\mu^*(\hat{x}_j) - y^TQy\end{aligned} \tag{114}$$

where $R = r\cdot r^T$ is a symmetric positive definite matrix. Therefore, we have

$$\begin{aligned}&\mu^{*T}(x)R\mu^*(x) - 2\mu^{*T}(x)R\mu^*(\hat{x}_j)\\ &\quad = \|r^T\mu^*(x) - r^T\mu^*(\hat{x}_j)\|^2 - \|r^T\mu^*(\hat{x}_j)\|^2\end{aligned} \tag{115}$$

By using the Lipschitz condition in Assumption 2, we have

$$\begin{aligned}\dot{L}_{c1}(V^*) \le& -\|r^T\mu^*(\hat{x}_j)\|^2 + L^2\|r\|^2\|e_{x_j}\|^2 - \underline{\lambda}(Q)\|y\|^2\\ =& -\alpha^2\underline{\lambda}(Q)\|y\|^2 + \Big[-(1-\alpha^2)\underline{\lambda}(Q)\|y\|^2\\ &+ L^2\|r\|^2\|e_{x_j}\|^2 - \|r^T\mu^*(\hat{x}_j)\|^2\Big]\end{aligned} \tag{116}$$

Considering the triggering condition (105), we have

$$\dot{L}_{c1}(V^*) \le -\alpha^2\underline{\lambda}(Q)\|y\|^2 \tag{117}$$

Next, for the term $\dot{L}_{c2}(\tilde{\omega}_{c2})$ in (112), we obtain

$$\begin{aligned}\dot{L}_{c2}(\tilde{\omega}_{c2}) =& \beta_c^{-1}tr\Big(-\beta_c\tilde{\omega}_{c2}^T\frac{\kappa\kappa^T}{(\kappa^T\kappa+1)^2}\tilde{\omega}_{c2} + \beta_c\tilde{\omega}_{c2}^T\frac{\kappa}{(\kappa^T\kappa+1)^2}\sigma_c\Big)\\ \le& -\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2\|\tilde{\omega}_{o2}\|^2\\ &+\frac{1}{2\beta_c}\Big(\beta_c^2\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2\|\tilde{\omega}_{o2}\|^2 + \frac{\sigma_c^2}{\|\kappa^T\kappa+1\|^2}\Big)\\ \le& -\left(1-\frac{\beta_c}{2}\right)\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2\|\tilde{\omega}_{o2}\|^2 + \frac{\sigma_c^2}{2\beta_c}\end{aligned} \tag{118}$$

It is important to note that the gradients of the critic network error is upper bounded, i.e., $\sigma_c \leq \sigma_{cM}$. Hence, we have

$$\dot{L}_{c2}(\tilde{\omega}_{c2}) \leq -\left(1-\frac{\beta_c}{2}\right)\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2 \|\tilde{\omega}_{o2}\|^2 + \frac{\sigma_{cM}^2}{2\beta_c} \tag{119}$$

Based on (83), (117) and (119), then $\dot{L}_{cl}$ becomes

$$\begin{aligned}\dot{L}_{cl} \leq & -\frac{1}{2}\underline{\lambda}(M)\|\tilde{x}\|^2 + \left(2\omega_{oM}\Phi_M\|P\| + \|P\|\xi_M\right)\|\tilde{x}\| \\ & - \alpha^2\underline{\lambda}(Q)\|y\|^2 - \left(1-\frac{\beta_c}{2}\right)\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2 \|\tilde{\omega}_{o2}\|^2 + \frac{\sigma_{cM}^2}{2\beta_c}\end{aligned} \tag{120}$$

Therefore, if the following conditions are satisfied,

$$\beta_c < 2 \tag{121}$$

$$\|\tilde{x}\| \geq \frac{4\omega_M\Phi_M\|P\| + 2\xi_M\|P\|}{\underline{\lambda}(M)} \tag{122}$$

$$\|\tilde{\omega}_{o2}\| \geq \sqrt{\frac{\sigma_{cM}^2/2\beta_c}{\left(1-\frac{\beta_c}{2}\right)\left\|\frac{\kappa\kappa^T}{\kappa^T\kappa+1}\right\|^2}} \tag{123}$$

then $\dot{L}_{cl} < 0$. This means the continuous dynamics of the impulsive model are UUB.

Now, we will consider the boundedness of the jump dynamics. The first difference of the Lyapunov function is shown as follows,

$$\begin{aligned}\Delta L_{cl} =& V^*(x^+) - V^*(x) + V^*(\hat{x}_j^+) - V^*(\hat{x}_j) \\ & + \beta_c^{-1}tr\left((\tilde{\omega}_{c2}^+)^T\tilde{\omega}_{c2}^+\right) - \beta_c^{-1}tr\left(\tilde{\omega}_{c2}^T\tilde{\omega}_{c2}\right) \\ & + \frac{1}{2}(\tilde{x}^T)^+P\tilde{x}^+ - \frac{1}{2}\tilde{x}^TP\tilde{x} + \frac{1}{2}tr\left((\tilde{\omega}_{o2}^T)^+\tilde{\omega}_{o2}^+\right) - \frac{1}{2}tr\left(\tilde{\omega}_{o2}^T\tilde{\omega}_{o2}\right) \\ =& \Delta L_c + \Delta L_o, \quad t = \delta_j\end{aligned} \tag{124}$$

where ΔL_o is defined in (85), which is UUB under the conditions (94) and (95). Now, we consider the boundedness of ΔL_c which is defined as

$$\begin{aligned}\Delta L_c =& V^*(x^+) - V^*(x) + V^*(\hat{x}_j^+) - V^*(\hat{x}_j) \\ & + \beta_c^{-1}tr\left((\tilde{\omega}_{c2}^+)^T\tilde{\omega}_{c2}^+\right) - \beta_c^{-1}tr\left(\tilde{\omega}_{c2}^T\tilde{\omega}_{c2}\right)\end{aligned} \tag{125}$$

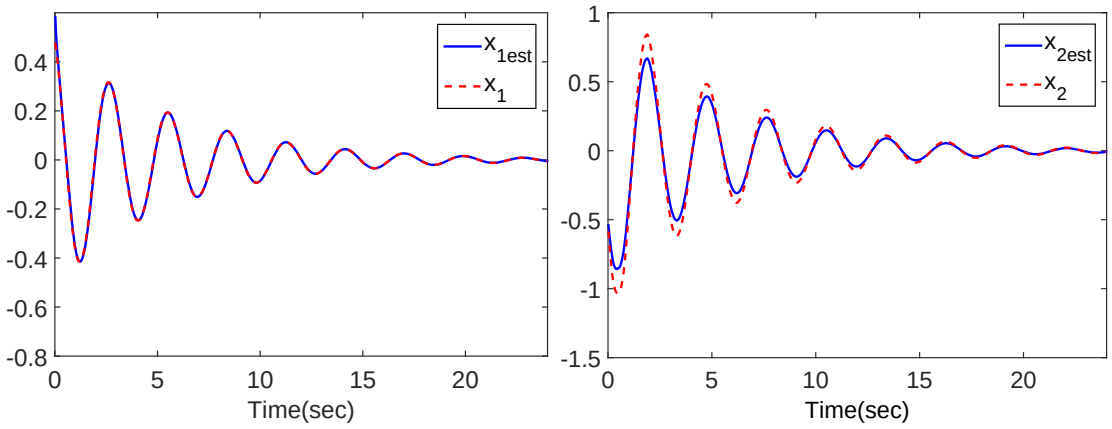

Figure 14. System responses with the event-triggered observer and ADP controller

Since the states and the critic network estimation error are UUB from the first part of the proof, there exists $V^*(x^+) \leq V^*(x)$ and $tr\big((\tilde{\omega}_{c2}^+)^T\tilde{\omega}_{c2}^+\big) \leq tr\big(\tilde{\omega}_{c2}^T\tilde{\omega}_{c2}\big)$ at the jump instants $t = \delta_j$. Moreover, for the sampled data, because during the jump instants, one has $\hat{x}^+ = \hat{x}_j^+$ and we have proved that the state estimation error is UUB, then $V^*(\hat{x}_j^+) \leq V^*(\hat{x}_j)$. Therefore, we have $\Delta L_c < 0$, then $\Delta L_{cl} < 0$. This means the jump dynamics of the close-loop system is also UUB. This completes the proof. ■

4.4 Simulation Studies

Consider a single link robot arm system giving by

$$\ddot{\theta}(t) = -\frac{MgH}{G}\sin(\theta(t)) - \frac{D}{G}\dot{\theta}(t) + \frac{1}{G}u(t) \tag{126}$$

where

$g = 9.81$, is the acceleration of gravity;

$H = 0.5$, is the length of the arm;

$D = 2$, is the viscous friction;

$M = 10$, is the mass of the payload;

$G = 10$, is the moment of inertia;

$\theta(t)$ is the angle position of robot arm;

$u(t)$ is the control input.

Assume that only the angle position $\theta(t)$ of the robot arm is observable. Defining $x_1(t) = \theta(t)$ and $x_2(t) = \dot{\theta}(t)$, the dynamic function (126) can be described as

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\dfrac{2}{10}x_2 + \dfrac{1}{10}u - \dfrac{49.05\sin(x_1)}{10} \\ y = x_1. \end{cases} \tag{127}$$

We can clearly observe that $y = x_1$ is the system measurable feedback in (127). This means the output matrix is $C = [1, 0]$ in this case.

Apply the proposed event-triggered ADP method to solve the problem. In order to recover the internal system state, an observer is built with the following parameters,

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -0.4 \end{bmatrix}; \qquad G = \begin{bmatrix} 10 & -1 \end{bmatrix}^T. \tag{128}$$

The designed observer includes a three-layer function network with the neuron structure as $3 - 6 - 2$ (i.e., three input neurons, six hidden neurons, and two output neurons). Based on the estimated internal state from the observer, a critic network is

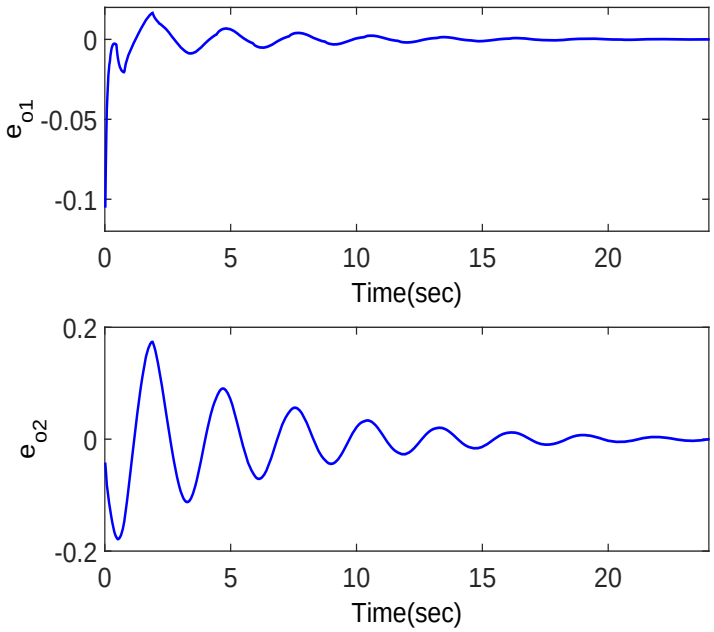

Figure 15. Errors between the estimated state and the true state.

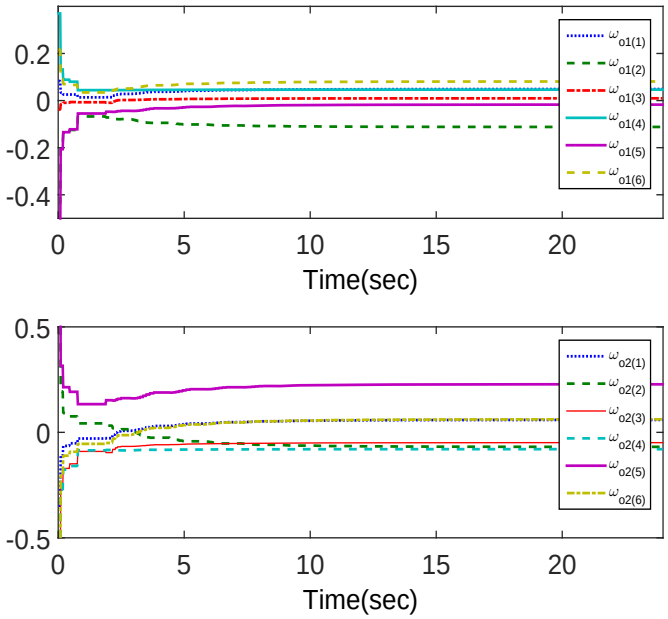

Figure 16. Trajectory of the weights in function network.

established to approximate the performance index and help to obtain the control law. The neuron structure of the critic network is $3-8-1$.

Choose the triggering condition as (53) with $L = 3$, $\alpha = 0.95$. Set Q, r as the identity

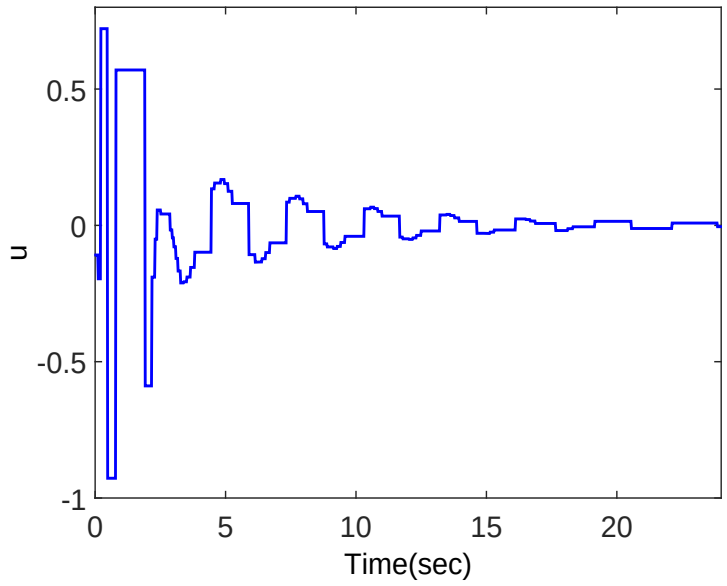

Figure 17. Trajectory of the event-triggered control law.

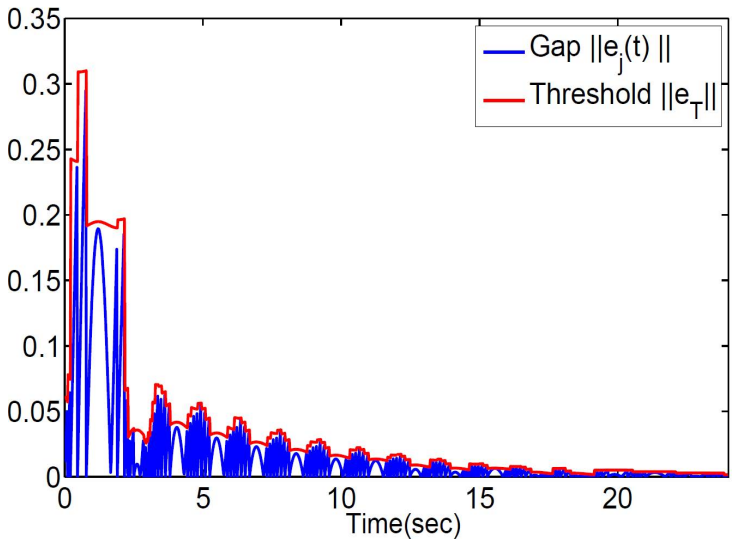

Figure 18. Comparison of the gap $\|e_{y_j}\|$ and the threshold $\|e_T\|$.

matrix with appropriate dimensions. Therefore, we have the triggering condition for this case as

$$\|e_{y_j}\|^2 \leq \frac{(1-0.95^2)\|C\|^2\|y\|^2 + \|C\|^2\|r^T\mu(x_j)\|^2}{3^2} \tag{129}$$

The trigger instants are decided according to (129). When the gap $e_{y_j} = C\hat{x}_j - y$ violates condition (129), the system state is sampled again by setting $\hat{x}_j = \hat{x}(t)$. The event-triggered observer and control law are updated again according to the sampled state.

Set the initial learning rates for both the function and the critic network as $\beta_o = \beta_c = 0.1$. Learning rates are decreased by 0.05 every five time steps until they reach $\beta_o = \beta_c = 0.005$ and stay thereafter. The initial weights of both networks are chosen randomly within $[-1, 1]$. The initial state is set to $x_0 = [0.5, -0.5]^T$. The sampling period for discretization is chosen as $0.03s$.

By employing the event-triggered ADP control method proposed in this chapter, we stabilize the partially observable system (127) only using the system input-output data. The trajectories of the system estimated state and true state are provided in Figure 14. It can be seen that the estimated state $\hat{x}_1$ and $\hat{x}_2$ can quickly approach the true state x_1 and x_2, respectively. This means the designed observer can recover the system internal state

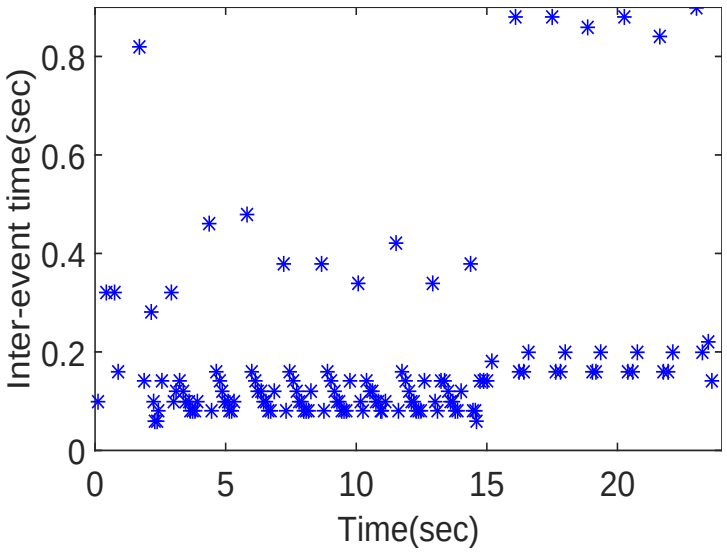

Figure 19. Inter-event time during the learning process.

from the output feedback, even with the reduced sampled data. The errors between the estimated state and the true state are provided in Figure 15. The learning process of the function network weights are shown in Figure 16. It is clearly that the weights updating law is aperiodic and only based on the sampled data. The observer is online training.

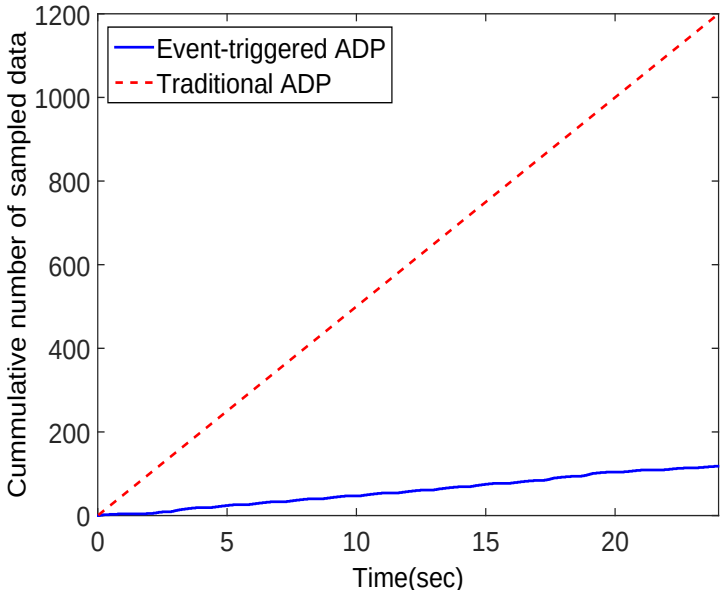

Figure 20. Cumulative number of the sampled data for both the event-triggered ADP method and traditional ADP method.

The trajectory for the event-triggered control law in this process is shown in Figure 17. We can observe that the control law is a piecewise signal. This means the control law keeps the same at period $[\delta_j, \delta_{j+1})$ and is only updated when an event is triggered. The relationship between the gap $\|e_{y_j}\|$ and the threshold is shown in Figure 18. It can be clearly observed that the gap $\|e_{y_j}\|$ is always smaller than the threshold to make sure the close-loop system is stable. The inter-event time between two consecutive transmissions are shown in Figure 19. We know the inter-event time exists and is up to $0.9s$ in this case. Finally, the cumulative number of the sampled data during the control process for both the proposed event-triggered ADP method and the traditional ADP method in [27] are provided in Figure 20. The event-triggered ADP method uses 118 samples while the traditional ADP method needs 1200 sample data. This means by efficiently reducing the sampled instants, the performance of the control method will not be influenced.

4.5 Summary

An event-triggered ADP control method is proposed in this chapter for nonlinear continuous-time system using only the input-output data. A neural-network-based observer is established to reconstruct the system internal states and the control coefficient function. Neural network techniques are applied to approximate the performance index and help calculate the control law. In order to save the computation resource and transmission load, both the designed observer and the controller are only updated when an event is triggered. The stability of the close-loop system is analyzed by Lyapunov construct for both the continuous and the jump dynamics. The simulation results demonstrate the effectiveness of the proposed method and also verify the theoretical analysis.

List of References

[1] X. Zhong, H. He, H. Zhang, and Z. Wang, "Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming," *Neural Networks and Learning Systems, IEEE Transactons on*, vol. 25, no. 12, pp. 2141–2155, 2015.

[2] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, "Model-free dual heuristic dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1834–1839, 2015.

[3] X. Zhong, Z. Ni, and H. He, "A theoretical foundation of goal representation heuristic dynamic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, 2015, in press.

[4] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press and John Wiley & Sons, 2004.

[5] V. Ugrinovskii* and H. R. Pota, "Decentralized control of power systems via robust control of uncertain markov jump parameter systems," *International Journal of Control*, vol. 78, no. 9, pp. 662–677, 2005.

[6] S. C. Lee, "Maintenance strategies for manufacturing systems using markov models," Ph.D. dissertation, The University of Michigan, 2010.

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

[8] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *arXiv preprint arXiv:1106.0234*, 2011.

[9] F. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.

[10] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive Dynamic Programming for Control: Algorithms and Stability (Communications and Control Engineering)*. Springer, 2013.

[11] T. Smith and R. Simmons, "Heuristic search value iteration for pomdps," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 520–527.

[12] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, vol. 3, 2003, pp. 1025–1032.

[13] H. Zhang, "Partially observable markov decision processes: A geometric technique and analysis," *Operations Research*, vol. 58, no. 1, pp. 214–228, 2010.

[14] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," vol. 7. MIT Press, 1995, p. 345.

[15] E. Saad, "Reinforcement learning in partially observable markov decision processes using hybrid probabilistic logic programs," *arXiv preprint arXiv:1011.5951*, 2010.

[16] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 14–25, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tsmc/tsmcb41.html#LewisV11

[17] B. Kiumarsi, F. Lewis, M.-B. Naghibi-Sistani, and A. Karimpour, "Optimal tracking control of unknown discrete-time linear systems using input–output measured data," *Cybernetics, IEEE Transactions on*, 2015, in press.

[18] Z. Ni, H. He, and X. Zhong, *Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP*. World Scientific Publishing, Singpore, ch. Frontiers of Intelligent Control and Information Processing.

[19] X. Zhong, Z. Ni, Y. Tang, and H. He, "Data-driven partially observable dynamic processes using adaptive dynamic programming," in *Proc. IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2014, pp. 1–8.

[20] W. Heemels, M. Donkers, and A. Teel, "Periodic event-triggered control for linear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 847–861, 2013.

[21] P. Tallapragada and N. Chopra, "On event triggered tracking for nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2343–2348, 2013.

[22] M. Lemmon, "Event-triggered feedback in control, estimation, and optimization," in *Networked Control Systems*. Springer, 2010, pp. 293–358.

[23] J. Zhang and G. Feng, "Event-driven observer-based output feedback control for linear systems," *Automatica*, vol. 50, no. 7, pp. 1852–1859, 2014.

[24] K. G. Vamvoudakis, "Event-triggered optimal adaptive control algorithm for continuous-time nonlinear systems," *IEEE/CAA JOURNAL OF AUTOMATICA SINICA*, vol. 1, no. 3, pp. 282–293, 2014.

[25] A. Sahoo, H. Xu, and S. Jagannathan, "Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming," *Neural Networks and Learning System, IEEE Transactions on*, 2015, in press.

[26] A. Sahoo, H. Xu, and S. Jagannathan, "Neural network-based event-triggered state feedback control of nonlinear continuous-time systems," *Neural Networks and Learning System, IEEE Transactions on*, 2015, in press.

[27] D. Liu, H. Li, and D. Wang, "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm," *Neurocomputing*, vol. 110, pp. 92–100, 2013.

[28] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[29] H. A. Talebi, F. Abdollahi, R. V. Patel, and K. Khorasani, *Neural Network-Based State Estimation of Nonlinear Systems*. Springer, New York, 2010. [Online]. Available: http://books.google.com/books?id=TzdCJdjrp3YC

CHAPTER 5

On-line Hierarchical Adaptive Critic Design

5.1 Introduction

Learning controller design for nonlinear systems is a difficult and challenging topic because it often requires solving the Hamilton-Jacobi-Bellman (HJB) equation rather than the Riccati equation. Fortunately, ADP techniques give us an opportunity to obtain the approximate solutions of the HJB equation [1, 2]. Generally specking, ADP can be categorized into three typical schemes: heuristic dynamic programming (HDP), dual HDP (DHP), and globalized DHP (GDHP) [3]. Various versions of ADP have been developed based on these three typical schemes, such as the action-dependent (AD) version and model-dependent version. Recently, a series of the goal representation heuristic dynamic programming (GrHDP) was proposed to improve the online learning of the ADP design in [4, 5]. Unlike the typical ADP schemes (i.e., one critic and one action networks), the authors integrated an additional network, namely the reference/goal network, to obtain an internal reinforcement signal to facilitate the optimal learning and control. This architecture has been applied to various realistic and complex control problems. In [6], the GrHDP design was applied on the tracking control problem and further on the real-time simulink/virtual reality platform. In addition, multiple reference/goal networks design, namely the hierarchical HDP design, was proposed and verified with promising control performance [7]. More recently, the GrHDP controller was further tested on the maze navigation problems [8, 9]. On the other hand, many researchers also followed this trend and applied the three-network HDP framework from different aspects. The improvement from the simulation results were provided and discussed in [10, 11]. This concept of goal representation was later introduced into the DHP structure, where the requiement of partial derivatives of the reinforcement signal were provided by the goal network. Literature [12] showed that the control performance

of this goal representation DHP (GrDHP) was improved on certain nonlinear numerical examples and a power system example.

Previous studies on the GrADP design were focused on the feasibility in implementation and simulation. Although the impressive performance has been achieved in the GrADP design, there is no rigorous theoretical proof of convergence in terms of both the internal reinforcement signal and the performance index under general conditions. In this chapter, I focus on this direction and provide a theoretical analysis of the GrADP design. Specifically, the theoretical foundation of the internal reinforcement signal is provided. It is shown that the designed internal reinforcement signal has the information of the future external reinforcement signals, which gives the agent a more effective information about the control action. Furthermore, a rigorous theoretical convergence analysis for the GrADP design is developed. It is proved that the internal reinforcement signal has an upper bound and the performance index in this method can monotonically non-decrease towards its optimal value. Furthermore, considering the advantage of the goal representation ADP (GrADP) design, in this chapter, we follow this trend by integrating the goal representation technique into the GDHP design and propose an advanced method which is goal representation GDHP (Gr-GDHP). In the proposed method, the goal network not only provides the internal reinforcement signal, but also generates its partial derivatives with respect to the system variables and control action. Therefore, the critic network can obtain both the adaptive (internal) reward and its derivatives within the Gr-GDHP structure to realize its objective function. Furthermore, we define the output of the goal network, including the internal reinforcement signal and its derivatives, as parts of the critic network's input to closely connect these two networks and help approximate the performance index and its derivatives. This chapter starts with the fundamental idea of goal representation design. Then, the goal representation ADP design ladder is presented. The foundation of the GrHDP and GrDHP

methods is provided as the background. After that, comparing with these two previous design, the key idea of the Gr-GDHP method is discussed. The algorithm for the proposed method is also provided. The architecture of the Gr-GDHP method is shown based on the neural networks with explicit neural-network-based learning process. We compare the simulation results of the Gr-GDHP method with the GrHDP, the GrDHP, and the traditional GDHP methods. It is shown that the proposed method can achieve better performance comparing with other ADP designs.

5.2 Goal Representation Design

Consider a nonlinear discrete-time system of the form

$$x(t+1) = F[x(t), u(t)] \tag{130}$$

where $x(t) \in R^n$ denotes the system state vector, and $u(t) \in R^m$ is the control input. Let $x(0)$ be the initial state. Assume that $F[x(t), u(t)]$ is Lipschitz continuous on a set $\Omega \subseteq R^n$ and $F[0, 0] = 0$. Therefore, $x(t) = 0$ is the equilibrium state of the system.

In the GrHDP method, an internal reinforcement signal $s(t)$ is designed to help stabilize the system which can be described as

$$\begin{aligned} s(t) =& r(t) + \alpha r(t+1) + \alpha^2 r(t+2) + \cdots \\ =& r(t) + \alpha\Big(r(t+1) + \alpha r(t+2) + \cdots\Big) \\ =& r(t) + \alpha s(t+1) \end{aligned} \tag{131}$$

where $0 < \alpha < 1$ is the discount factor, and $r(t)$ is the external reinforcement signal which is positive definite. In this chapter, the external reinforcement signal is chosen as the quadratic form

$$r(t) = x^T(t) Q x(t) + u^T(t) R u(t) \tag{132}$$

where Q and R are positive definite matrices with appropriate dimensions.

Compare with the traditional ADP design which only provides a single external reinforcement signal to the agent. We can observe that the designed internal reinforcement signal has the information of the future external reinforcement signals. This means the internal reinforcement signal $s(t)$ gives us more information by considering more distant lookahead. In other words, for each state visited, the internal reinforcement signal looks forward in time to the future information and therefore this signal is more effective.

The major difference between the GrADP and ADP method is provided in Figure 21. Comparing with the traditional ADP method which uses the external reinforcement signal $r(t)$ to provide the information of the control action directly, the GrADP method designs an internal reinforcement signal $s(t)$ to represent the performance of the control action based on the information of $r(t)$.

The control action we are desired is to minimize the performance index function $V(t)$ which is given based on $s(t)$,

$$J(t) = s(t) + \gamma J(t+1) \tag{133}$$

where $0 < \gamma < 1$ is the discount factor. Note that γ and α are not necessary the same. Furthermore, since (132) is positive definite, then according to (131) and (133), we know that $s(t)$ and $J(t)$ are positive definite.

From [13], we know that the designed controllers need not only to stabilize the system but also to guarantee that the performance index (133) is finite, which means the control must be admissible.

Definition 1: A law $u(t)$ is said to be an admissible control with respect to (133) on Ω, if $u(t)$ is continuous on Ω and stabilize system (218) for all $x(0) \in \Omega$, $u(t) = 0$ if $x(t) = 0$, and $\forall x(t) \in \Omega$, $J(t)$ is finite.

Here, assume that there exists the optimal solution for (133). Based on Bellman's optimality principle, the optimal performance index $J^*(t)$ satisfies the discrete-time

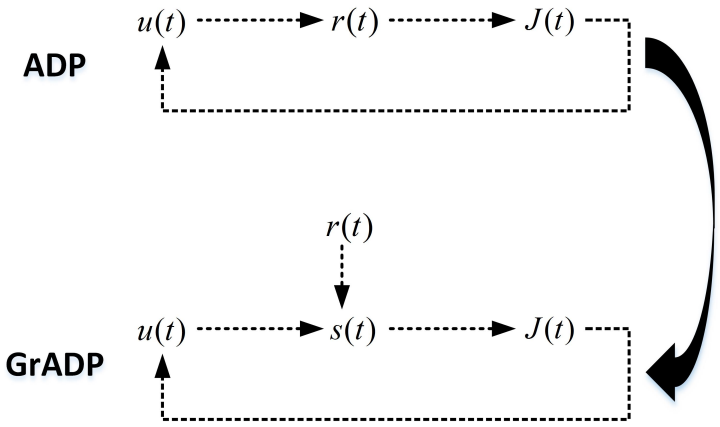

Figure 21. The concept diagram of learning and feedback evaluation process: In contrast with the traditional ADP design, the GrADP method has the internal reinforcement signal in the loop which includes the information of future external reinforcement signal.

HJB equation

$$J^*(t) = \min_{u(t)}\{s(t) + \gamma J^*(t+1)\} \tag{134}$$

where

$$s(t) = r(t) + \alpha s(t+1) \tag{135}$$

is the internal reinforcement signal.

Therefore, the optimal control law can be described as

$$u^*(t) = arg\min_{u(t)}\{s(t) + \gamma J^*(t+1)\}. \tag{136}$$

From (134) we know the control action decides what is the best strategy to combine the internal reinforcement signals, and also the internal reinforcement signal has the information of future external reinforcement signals. Assume that the agent is standing on a given state, first calculating the internal reinforcement signals $s(t)$ for all the possible control actions to provide the adaptive and effective information, then determining which is the optimal control action according to the discounted cumulative internal reinforcement signals. Therefore, the ultimate goal is to find the control action to minimize

the performance index $J(t)$. In the next section, the GrHDP algorithm is provided to solve the performance index $J^*(t)$ of the HJB equation (134) and the corresponding convergence analysis is also presented.

5.3 GrADP Control and the Theoretical Analysis

In this section, the GrADP algorithm is first adopted to approximate $s(t)$, $J(t)$ and $u(t)$, respectively. Then the convergence analysis of this algorithm is also presented. Note that since $r(t)$ is a function of $x(t)$ and $u(t)$, we use $r(x(t), u(t))$ to represent this external reinforcement signal to facilitate the proof.

5.3.1 GrADP Algorithm

Set the initial values as $s^0 = 0$ and $J^0 = 0$. The control action is calculated by

$$u^i(t) = arg \min_{u(t)}\{s(t) + \gamma J^i(t+1)\}, \tag{137}$$

where i is the iteration index and t is the time index. Once the control action $u^i(t)$ is determined, the internal reinforcement signal can be updated by

$$s^{i+1}(t) = r(x(t), u^i(t)) + \alpha s^i(t+1). \tag{138}$$

Then, the performance index is determined by

$$\begin{aligned} J^{i+1}(t) &= \min_{u(t)}\{s(t) + \gamma J^i(t+1)\} \\ &= s^{i+1}(t) + \gamma J^i(t+1) \\ &= r(x(t), u^i(t)) + \alpha s^i(t+1) + \gamma J^i(t+1). \end{aligned} \tag{139}$$

The GrADP algorithm, therefore, is a form of incremental optimization of iterating (137), (138) and (139).

Note that, the GrADP algorithm is an incremental optimization process which is implemented forward in time and online. In the next subsection, the convergence proof

of the GrADP approach is provided, including the boundedness of the internal reinforcement signal and the convergence of the performance index. The existence of the corresponding control law is also presented.

5.3.2 Convergence analysis of the GrADP approach

Before I present the proof of convergence for the GrADP algorithm, two important lemmas are given as follows

Lemma 1: Consider sequences which are updated as

$$\phi^{i+1}(t) = r(x(t), \eta^i(t)) + \alpha\phi^i(t+1) \tag{140}$$

and

$$\delta^{i+1}(t) = \phi^{i+1}(t) + \gamma\delta^i(t+1), \tag{141}$$

where $\eta^i(t)$ is any stabilizing and admissible control law sequence, $r(x(t), \eta^i(t)) = x^T(t)Qx(t) + \eta^{iT}(t)R\eta^i(t)$. Define $u^i(t)$, $s^i(t)$ and $J^i(t)$ as in (137), (138) and (139), respectively. If $\phi^0 = s^0 = 0$, $\delta^0 = J^0 = 0$ then, $J^i(t) \leq \delta^i(t)$.

Proof: From (138), we have

$$\begin{aligned} s^{i+1}(t) &= r(x(t), u^i(t)) + \alpha s^i(t+1) \\ &= \sum_{k=t}^{i+t} \alpha^{k-t} r(x(k), u^{i+t-k}(k)). \end{aligned} \tag{142}$$

Hence, according to (139), we can rewrite the performance index as

$$\begin{aligned} J^{i+1}(t) &= \sum_{k=t}^{i+t} \alpha^{k-t} r(x(k), u^{i+t-k}(k)) + \gamma J^i(t+1) \\ &= \sum_{k=t}^{i+t} \alpha^{k-t} \Big(x^T(t)Qx(t) + u^{iT}(t)Ru^i(t) \Big) + \gamma J^i(t+1). \end{aligned} \tag{143}$$

Consider (140) and (141), the sequences $\phi^i(t)$ and $\delta^i(t)$ can be also described as

$$\phi^{i+1}(t) = \sum_{k=t}^{i+t} \alpha^{k-t} r(x(k), \eta^{i+t-k}(k)) \tag{144}$$

$$\delta^{i+1}(t) = \sum_{k=t}^{i+t} \alpha^{k-t} \Big(x^T(t)Qx(t) + \eta^{iT}(t)R\eta^i(t) \Big) + \gamma\delta^i(t+1). \tag{145}$$

Consider (143) and (145). Because $u^i(t)$ minimizes the right-hand side of (143), and $\eta^i(t)$ is any stabilizing and admissible control law sequence, then considering $\phi^0 = s^0 = 0$, $\delta^0 = J^0 = 0$, we have $J^i(t) \leq \delta^i(t)$ by induction, which completes the proof. ■

Next lemma provides the existence of the admissible control.

Lemma 2: Let the internal reinforcement signal sequence $s^i(t)$ and the performance index sequence $J^i(t)$ be defined as in (138) and (139), respectively. If the system (130) is controllable, then the admissible control law exists.

Proof: Since sequences $J^i(t)$ and $s^i(t)$ are positive definite, both of them attain minimal values at $x(t) = 0$. Thus, $\frac{\partial J^i(t)}{\partial x(t)}$ and $\frac{\partial s^i(t)}{\partial x(t)}$ should vanish there, which indicates that $u(t) = 0$ if $x(t) = 0$. The continuity assumption on $F[x(t), u(t)]$ implies that there exists a continuous control law and the system (130) cannot jump to infinity by any one step of finite control. Moreover, since $F[0, 0] = 0$, the control input becomes zero and the state is kept at zero when the system reaches the equilibrium state. This means the external reinforcement signal $r(x(t), u(t))$ becomes zero when the system is stable ($r(0, 0) = 0$). Hence, according to the definition of $s(t)$ and $J(t)$, we know that both $s(t)$ and $J(t)$ are finite. According to Definition 1, we know the admissible control law exists for system (130), which proves the conclusion. ■

Now, we present our main results.

Theorem 1: Define the sequences $s^i(t)$ and $J^i(t)$ as in (138) and (139), respectively. The sequence $u^i(t)$ is determined by (137). If the system is controllable and $s^0 = J^0 = 0$, then the following conclusions hold.

(1) $J^i(t)$ is a monotonically non-decreasing sequence, i.e., $J^i(t) \leq J^{i+1}(t)$.

(2) There exists $0 \leq s^i(t) \leq P(x(t))$, $0 \leq J^i(t) \leq W(x(t))$, where $P(x(t))$ and $W(x(t))$ are the upper bounds for sequences $s^i(t)$ and $J^i(t)$, respectively.

(3) When $i \to \infty$, then $J^i(t) \to J^*(t)$, $u^i(t) \to u^*(t)$. This implies the sequence $J^i(t)$ can converge to the solution of the discrete-time HJB equation (134).

Proof: From Lemma 1, we know if $\phi^0 = s^0 = 0$, $\delta^0 = J^0 = 0$, the sequence $\delta^i(t)$ defined in (141) has the following property

$$J^i(t) \leq \delta^i(t). \tag{146}$$

In the following part, it will be proved $J^{i+1}(t) \geq \delta^i(t)$ by mathematical induction. Because $\phi^0 = s^0 = 0$ and $\delta^0 = J^0 = 0$, it follows

$$J^1(t) - \delta^0(t) = r(x(t), u^0(t)) \geq 0 \tag{147}$$

which means $J^{i+1}(t) \geq \delta^i(t)$ holds for $i = 0$.

Now, assume that there exists $J^i(t) \geq \delta^{i-1}(t)$ for the $(i-1)^{th}$ iteration step. Set the stabilizing and admissible control law $\eta^{i-1}(t) = u^i(t)$ and the summation of external reinforcement signal $\phi^i(t+1) = s^i(t+1)$. We obtain

$$\delta^i(t) = r(x(t), u^i(t)) + s^i(t+1) + \gamma\delta^{i-1}(t+1). \tag{148}$$

By subtracting (148) from (139), it yields

$$J^{i+1}(t) - \delta^i(t) = \gamma\Big(J^i(t+1) - \delta^{i-1}(t+1)\Big) \geq 0. \tag{149}$$

This completes the proof of $J^{i+1}(t) \geq \delta^i(t)$. Combining this with (146), we obtain $J^i(t) \leq \delta^i(t) \leq J^{i+1}(t)$ for any $i = 0, 1, 2, \cdots$. This means, $J^i(t) \leq J^{i+1}(t)$. Therefore, the sequence $J^i(t)$ is a monotonically non-decreasing sequence. This completes the proof of part (1).

The second part of the theorem follows by realizing that the sequence $J^i(t)$ is positive and monotonically non-decreasing. Hence, we can conclude that

$$0 \leq J^i(t) \leq J^\infty(t). \tag{150}$$

Now, it will be proved there exist an upper bound for this sequence. Set $\mu(t)$ be any stabilizing and admissible control and $\theta^0 = s^0 = 0$, where $\theta^i(t)$ is updated as

$$\theta^{i+1}(t) = r(x(t), \mu(t)) + \alpha\theta^i(t+1). \tag{151}$$

Define $\theta^0 = 0$, and equation (151) can be written as

$$\begin{aligned}
\theta^{i+1}(t) &= r(x(t),\mu(t)) + \alpha\theta^i(t+1) \\
&= r(x(t),\mu(t)) + \alpha r(x(t+1),\mu(t+1)) + \alpha^2\theta^{i-1}(t+2) \\
&\vdots \\
&= r(x(t),\mu(t)) + \alpha r(x(t+1),\mu(t+1)) \\
&\quad + \cdots + \alpha^i r(x(t+i),\mu(t+i)) + \alpha^{i+1}\theta^0(t+i+1) \\
&= \sum_{k=0}^{i} \alpha^k r(x(t+k),\mu(t+k)) \\
&= \sum_{k=t}^{i+t} \alpha^{k-t} r(x(k),\mu(k)) \\
&\le \sum_{k=t}^{\infty} \alpha^{k-t} r(x(k),\mu(k)).
\end{aligned} \tag{152}$$

Since $\mu(t)$ is an admissible stabilizing control, $x(t) \to 0$ when $t \to 0$, and $\forall i$ such that

$$\theta^{i+1}(t) \le \sum_{k=t}^{\infty} \alpha^{k-t} r(x(k),\mu(k)). \tag{153}$$

By setting $\mu(t) = u^{i-1}(t)$ and $\theta^{i-1}(t+1) = s^{i-1}(t+1)$, it follows that $\forall i$,

$$s^{i+1}(t) \le \sum_{k=t}^{\infty} \alpha^{k-t} r(x(k),\mu(k)). \tag{154}$$

Define $P(t) = \sum_{k=t}^{\infty} \alpha^{k-t} r(x(k),\mu(k))$, and hence $s^i(t) \le P(t)$. This completes the conclusion that $P(t)$ is an upper bound of sequence $s^i(t)$. Therefore, $0 \le s^i(t) \le P(t)$.

In the following part, it will be proved that there also exists an upper bound for the sequence $J^i(t)$. We rewrite (139) as

$$\begin{aligned}
J^{i+1}(t) &= s^{i+1}(t) + \gamma J^i(t+1) \\
&= s^{i+1}(t) + \gamma s^i(t+1) + \gamma^2 J^{i-1}(t+2) \\
&\vdots \\
&= s^{i+1}(t) + \gamma s^i(t+1) + \cdots + \gamma^i s^1(t+i) + \gamma^{i+1} J^0(t+i+1) \\
&= \sum_{m=t}^{t+i} \gamma^{m-t} s^{i+t-m}(m).
\end{aligned} \tag{155}$$

Since $s^i(t) \le P(t)$, it follows that

$$J^{i+1}(t) \le \sum_{m=t}^{\infty} \gamma^{m-t} P(m). \tag{156}$$

Define $W(t) = \sum_{m=t}^{\infty} \gamma^{m-t} P(m)$, such that $0 \le J^i(t) \le W(t)$. Hence, the proof of part (2) is completed. Note that both $P(t)$ and $W(t)$ are determined by the admissible stabilizing control law $\mu(t)$. This means when $t \to \infty$, $\mu(t) \to 0$ and $x(t) \to 0$. Hence, $\lim_{t\to\infty} r(x(t), \mu(t)) = 0$, indicating that $P(t)$ and $W(t)$ are finite values.

For part (3), define a sequence $\varphi^i(t)$ with the following update rule

$$\varphi^{i+1}(t) = \theta^{i+1}(t) + \gamma \varphi^i(t+1), \tag{157}$$

where $\varphi^0 = J^0 = 0$. From Lemma 1, we known $J^i(t) \le \varphi^i(t)$ by setting $\phi^i(t) = \theta^i(t)$ and $\delta^i(t) = \varphi^i(t)$. After some derivation, (157) can be rewritten as

$$\begin{aligned} \varphi^{i+1}(t) &= \theta^{i+1}(t) + \gamma\theta^i(t+1) + \gamma^2\theta^{i-1}(t+2) + \cdots \\ &= \sum_{m=0}^{i} \gamma^{m+t} \Big(\sum_{k=t}^{i+t} \alpha^{k-t} r\big(x(k+m), \mu(k+m)\big) \Big) \end{aligned} \tag{158}$$

Let $i \to \infty$, it follows

$$J^{\infty}(t) \le \varphi^{\infty}(t) \le \sum_{m=0}^{\infty} \gamma^{m+t} \Big(\sum_{k=t}^{\infty} \alpha^{k-t} r\big(x(k+m), \mu(k+m)\big) \Big). \tag{159}$$

If $\mu(t) = u^*(t)$, then

$$J^{\infty}(t) \le \sum_{m=0}^{\infty} \gamma^{m+t} \Big(\sum_{k=t}^{\infty} \alpha^{k-t} r\big(x(k+m), u^*(k+m)\big) \Big) = J^*(t). \tag{160}$$

On the other hand, consider the definition of the upper bound $W(t)$. Because $\mu(t)$ is defined as an admissible control, setting $\mu(t)$ is the control input of the infinite step, it follows,

$$J^{\infty}(t) = W(t) \ge J^*(t). \tag{161}$$

From (160) and (161), we know $J^{\infty}(t) = J^*(t)$, which means $J^i(t)$ converges to the optimal value $J^*(t)$. This completes the result that $J^{\infty}(t) = J^*(t)$.

Now let us consider the convergence of the control action. According to equation (137), we obtain

$$u^{\infty}(t) = arg \min_{u(t)}\{s(t) + \gamma J^{\infty}(t+1)\} \tag{162}$$

$$u^{*}(t) = arg \min_{u(t)}\{s(t) + \gamma J^{*}(t+1)\}. \tag{163}$$

Therefore, we can observe that if $\lim_{i\to\infty} J^{i}(t) = J^{*}(t)$ hold, then we have $\lim_{i\to\infty} u^{i}(x(t)) = u^{*}(x(t))$. The conclusion holds. ■

Theorem 1 proves that the performance index sequence $J^{i}(t)$ is a monotonically non-decreasing sequence, and both the internal reinforcement signal sequence $s^{i}(t)$ and the performance index sequence $J^{i}(t)$ exist upper bounds. This means $s(t)$ and $J(t)$ cannot go infinity. Moreover, the performance index sequence and the control law sequence can converge to their optimal value, respectively, after certain iteration steps. This implies that we can use this algorithm to approximate the solution of the discrete-time HJB equation (134). Next section presents the neural-network-based implementation of the GrADP approach.

5.4 Goal representation ADP ladder

The family of the GrADP method is discussed in this section, including the GrHDP, GrDHP, and Gr-GDHP.

5.4.1 Goal Representation Design in HDP

In literature work [4, 5], the GrHDP method is discussed. It is shown that the goal network is integrated into the traditional HDP design to generate an internal reinforcement signal $s(t)$ to help the control process. This $s(t)$ signal is adaptive and learn from the external reinforcement signal $r(t)$. Therefore, the output of the goal network can be described as

$$s(t) = r(t) + \alpha s(t+1) \tag{164}$$

where $0 < \alpha < 1$ is the discount factor. Then, we include $s(t)$ within the inputs of the critic network to closely connect the goal network and the critic network. Therefore, the output of the critic network, performance index $J(t)$, can be provided as

$$J(t) = s(t) + \gamma J(t+1) \tag{165}$$

where $0 < \gamma < 1$ is the discount factor. Notice that γ and α are not necessary the same. In the GrHDP design, our goal is to seek an optimal control action $u(t)$, so that the performance index $J(t)$ in (165) can be minimized. Various complex and realistic control cases have been tested on this architecture with numerous impressive performance [6, 8, 7]. Furthermore, the theoretical foundation of this method has been provided in [14]. It is shown that the internal reinforcement signal $s(t)$ exists an upper bound and the performance index $J(t)$ can converge to its optimal value.

5.4.2 General Utility Function Representation in DHP

Later, this general representation of the utility function is introduced into the DHP approach, and thus GrDHP method has been proposed [12]. In the GrDHP method, we adopt the goal network to estimate the partial derivatives of $s(t)$ with respect to the vector $Y(t) = [x^T(t), u^T(t)]^T$, where $x(t)$ denotes the system state and $u(t)$ is the control action. Therefore, the output of the goal network in GrDHP method becomes

$$g(t) = \frac{\partial s(t)}{\partial Y(t)} \tag{166}$$

The derivatives of the performance index are provided by the critic network as

$$\lambda(t) = \frac{\partial J(t)}{\partial Y(t)} = \frac{\partial s(t)}{\partial Y(t)} + \gamma \frac{\partial J(t+1)}{\partial Y(t)} \tag{167}$$

Since now the critic network is trained to estimate the high quality of $\partial J(t)/\partial u(t)$, our goal becomes to minimize $\partial J(t)/\partial u(t)$ to find the optimal control action [15].

Follow our previous work, in this chapter, we introduce the goal representation concept into the traditional GDHP method, and propose a Gr-GDHP design. A goal

network is established to provide both the adaptive internal reinforcement signal and its derivatives to help the performance index estimation in critic network and the decision making in action network.

5.4.3 Gr-GDHP Design

Considering the advantage of both GrHDP and GrDHP, we build a goal network in Gr-GDHP to estimate both the internal reinforcement signal $s(t)$ in (164) and its derivatives $g(t)$ in (166). Moreover, a critic network is also applied to estimate both the performance index $J(t)$ in (165) and its derivatives $\lambda(t)$ in (167).

Notice that, in our design, both $s(t)$ and $g(t)$ can not only provide an adaptive representation of the utility function for the critic network, but also help estimate the cost-to-go in detail since $s(t)$ and $g(t)$ work as parts of the critic network's inputs.

A model network is also applied in this design to predict the future system states. Therefore, we can obtain all the predicted internal reinforcement signal, performance index, and their partial derivatives at the current time step. Subsequently, the temporal difference (TD) errors for all the $s(t)$, $g(t)$, $J(t)$, and $\lambda(t)$ between the current and the next time step can be achieved. Furthermore, the designed model network needs to be considered into the backpropagation paths of the neural network learning process, since the critic and the goal networks are connected with the action network through it.

The algorithm of the Gr-GDHP design is presented in Algorithm 3.

5.5 Learning Process of Gr-GDHP Approach

In this section, we will provide the explicit procedures on the implementation of the proposed Gr-GDHP approach. The architecture of this approach is shown in Figure 22. All the neural networks applied in this method are established based on the Multilayer perceptron (MLP) neural network technique. We can observe that the goal network's outputs are included in the critic network's inputs, which provides more information

Algorithm 3 Outline of the Implementation for Gr-GDHP Control Design.

Initialize all the neural network weights.

Set t = 0.

for all $t < N_{run}$ **do**

1. Obtain $u(t)$ through the action network.

2. Observe $x(t)$ from the environment or system.

3. Calculate $\hat{s}(t)$, $\hat{g}(t)$, $\hat{J}(t)$, and $\hat{\lambda}(t)$ through the goal network and the critic network, respectively.

4. Predict $\hat{x}(t+1)$ through the model network with the inputs $x(t)$ and $u(t)$.

5. The predicted $\hat{u}(t+1)$ is obtained through the action network with the input $\hat{x}(t+1)$.

6. The predicted $\hat{s}(t+1)$ and $\hat{g}(t+1)$ are obtained through the goal network with the inputs $\hat{x}(t+1)$ and $\hat{u}(t+1)$.

7. The external reinforcement signal $r(t)$ is given based on the current state. If the control fails, start over from the beginning.

8. The TD errors are obtained between $\hat{s}(t)$, $\hat{g}(t)$ and $\hat{s}(t+1)$, $\hat{g}(t+1)$, respectively. The goal network weights are updated accordingly.

9. The predicted $\hat{J}(t+1)$ and $\hat{\lambda}(t+1)$ are obtained through the critic network with the inputs $\hat{x}(t+1)$, $\hat{u}(t+1)$, $\hat{s}(t+1)$, and $\hat{g}(t+1)$.

10. The TD errors are obtained between $\hat{J}(t)$, $\hat{\lambda}(t)$ and $\hat{J}(t+1)$, $\hat{\lambda}(t+1)$, respectively. The critic network weights are updated accordingly.

11. The action network weights are tuned according to the outputs of the goal and the critic networks.

12. System information is updated based on the newly obtained $u(t)$.

13. Set $t = t + 1$.

end for

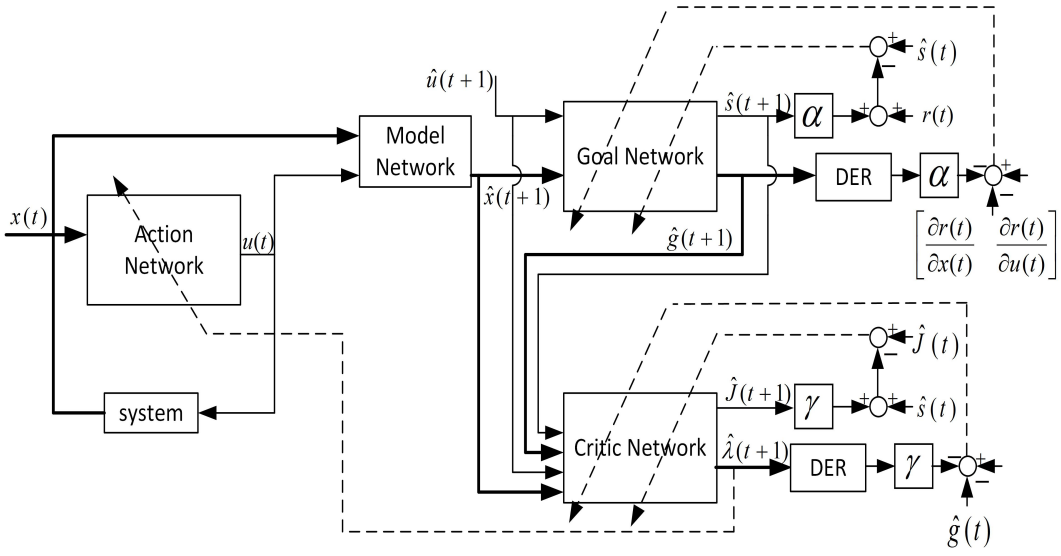

Figure 22. Architecture of the Gr-GDHP approach.

to approximate the performance index. The predicted control action $\hat{u}(t+1)$ will be generated from the action network with input $\hat{x}(t+1)$, which is estimated by the model network. In this section, the weights updating rules of the action, the critic, and the goal networks are provided, respectively. The training process of the model network is also described, since the future information is required in this method.

5.5.1 State Prediction

A model network [16, 17] is established in this Gr-GDHP method to predict the future information of the system. Therefore, the predicted value of the internal reinforcement signal, the performance index, and their derivatives can be achieved at current time step based on the predicted system state. The function approximation structure is designed as a three-layer neural network. The weights between the input and the hidden layers are denoted by ω_{m1} and the weights between the hidden and the output layers are set as ω_{m2}. In this chapter, we randomly chose the input-to-hidden layer weights ω_{m1} at initial and keep as constant thereafter. Therefore, only the output layer weights ω_{m2} are proposed to be tuned during the learning process. The design of the model network is similar to the RVFL nets in [18].

Define the identification scheme as:

$$\hat{x}(t+1) = \omega_{m2}^T(t)\sigma\Big(\omega_{m1}^T Y(t)\Big) \tag{168}$$

where $Y(t) = [x^T(t), u^T(t)]^T$ is the input of the model network, $x(t)$ is the system state, $u(t)$ is the control action, $\hat{x}(t+1)$ is the predicted state of next time step, and $\sigma(\cdot)$ is the bounded activation function. Here, we define $\sigma(\cdot)$ as the sigmoid function

$$\sigma(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{169}$$

Since the neural networks have the property of universal approximation, the above identified system can be described as the neural network representation

$$x(t+1) = \omega_{m2}^{*T}\sigma\Big(\omega_{m1}^T Y(t)\Big) + \delta(t) \tag{170}$$

where ω_{m2}^{*T} is the ideal hidden-to-output weights, and $|\delta(t)| \leq \delta_M$ is the approximation error of neural network.

The identification error is defined as

$$e(t+1) = \hat{x}(t+1) - x(t+1) \tag{171}$$

Then, considering (168) and (170), we have

$$\begin{aligned} e(t+1) =& \omega_{m2}^T(t)\sigma\Big(\omega_{m1}^T Y(t)\Big) - \omega_{m2}^{*T}\sigma\Big(\omega_{m1}^T Y(t)\Big) - \delta(t) \\ =& \tilde{\omega}_{m2}(t)\sigma\Big(\omega_{m1}^T Y(t)\Big) - \delta(t) \end{aligned} \tag{172}$$

where $\tilde{\omega}_{m2} = \omega_{m2}(t) - \omega_{m2}^*$ is the errors of the weights.

Hence, the weights in the system identification process are updated to minimize the following objective function

$$E_{m2} = \frac{1}{2} e^T(t+1)e(t+1) \tag{173}$$

Using the gradient descent method to minimize (173), we have the weights updating law for the model network as

$$\omega_{m2}(t) = \omega_{m2}(t) + \Delta\omega_{m2}(t) \tag{174}$$

where

$$\begin{aligned} \Delta\omega_{m2}(t) =& -\beta_m \left[\frac{\partial E_{m2}(t)}{\partial \omega_{m2}(t)}\right] \\ =& -\beta_m \left[\sigma\Big(\omega_{m1}^T Y(t)\Big) \cdot (\hat{x}(t+1) - x(t+1))\right] \end{aligned} \tag{175}$$

Notice that the training process of the model network is offline in this chapter. This means when the model network weights are well trained, we fixed them and start to online train the weights of the goal, the critic, and the action networks.

5.5.2 General Utility Function Representation Design

Compared with the traditional GDHP design that assigns an instant reward signal, which is called the external reinforcement signal in this chapter, from the environment,

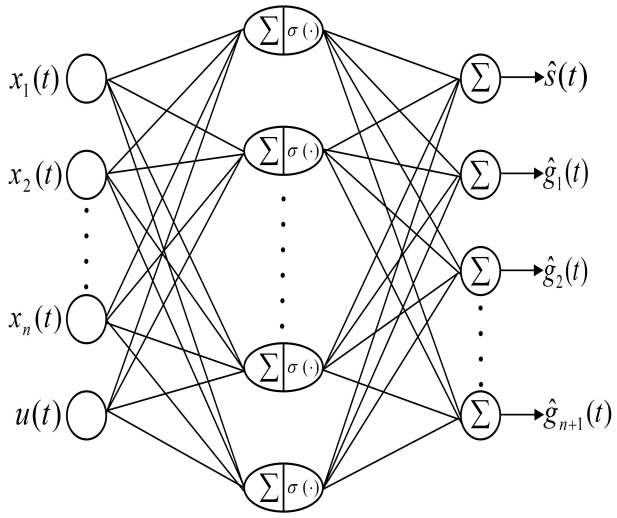

Figure 23. The schematic archtecture of the goal network.

our proposed Gr-GDHP method designs an internal reinforcement signal including the information of future external reinforcement signals. Specifically, a goal network is integrated to provide a more effective internal reinforcement signal to represent the performance of the control action. This signal is adaptive and learned from the external reinforcement signal $r(t)$. The goal network, in the Gr-GDHP design, aims to approximate both $s(t)$ and its partial derivatives $g(t)$ with respect to the vector $Y(t)$. Then, we can describe $g(t)$ as

$$g(t) = \frac{\partial s(t)}{\partial Y(t)} = \left[\frac{\partial s(t)}{\partial x(t)}, \frac{\partial s(t)}{\partial u(t)}\right]^T \tag{176}$$

Notice that $\partial(\cdot)/\partial(\star)$ means the partial derivations of the scalar $(\cdot)$ with respect to the components in $(\star)$.

The goal network is established as a three-layer neural network architecture, as shown in Figure 23. It can be observed that the vector $Y(t)$ is applied as the input of the goal network. Therefore, the output of the goal network becomes

$$\begin{bmatrix}\hat{s}(t)\\ \hat{g}(t)\end{bmatrix} = \begin{bmatrix}\omega_{g2}^{aT}(t)\\ \omega_{g2}^{bT}(t)\end{bmatrix} \cdot \sigma\left(\omega_{g1}^T(t)Y(t)\right) \tag{177}$$

where $\omega_{g2}(t) = [\omega_{g2}^a(t), \omega_{g2}^b(t)]$ is the hidden-to-output layer weights and $\omega_{g1}(t)$ is the input-to-hidden layer weights of the goal network. The sigmoid function $\sigma(\cdot)$ is defined

the same as in (169). Here, we only adopt the sigmoid function on the input-to-hidden layer nodes. Therefore, we can obtain

$$\hat{s}(t) = \omega_{g2}^{aT}(t)\sigma\left(\omega_{g1}^{T}(t)Y(t)\right) \tag{178}$$

$$\hat{g}(t) = \omega_{g2}^{bT}(t)\sigma\left(\omega_{g1}^{T}(t)Y(t)\right) \tag{179}$$

The target functions can be written as

$$s(t) = r(t) + \alpha\hat{s}(t+1) \tag{180}$$

and

$$g(t) = \frac{\partial r(t)}{\partial Y(t)} + \alpha\frac{\partial\hat{s}(t+1)}{\partial Y(t)} \tag{181}$$

Here, we use the predicted internal reinforcement signal $\hat{s}(t+1)$ instead of $s(t+1)$ in (164), since in the implementation process, the accurate value of $s(t+1)$ is difficult to achieve.

Comparing (178), (179) with (180), (181), respectively, the approximation error functions of the goal network is defined as

$$e_{g1}(t) = \hat{s}(t) - s(t) = \hat{s}(t) - \alpha\hat{s}(t+1) - r(t) \tag{182}$$

$$e_{g2}(t) = \frac{\partial\hat{s}(t)}{\partial Y(t)} - \alpha\frac{\partial\hat{s}(t+1)}{\partial Y(t)} - \frac{\partial r(t)}{\partial Y(t)} \tag{183}$$

Hence, the following objective function needs to be minimized in order to update the goal network weights:

$$E_g(t) = \eta_1 E_{g1}(t) + \eta_2 E_{g2}(t) \tag{184}$$

where

$$E_{g1}(t) = \frac{1}{2}e_{g1}^2(t), \quad E_{g2}(t) = \frac{1}{2}e_{g2}^2(t) \tag{185}$$

in which η_1 and η_2 are the positive parameters that adjust how GrHDP and GrDHP are combined in Gr-GDHP.

$$DER = \frac{\partial \hat{Y}(t+1)}{\partial Y(t)} = \begin{bmatrix} \frac{\partial \hat{x}(t+1)}{\partial u(t)} \cdot \frac{\partial u(t)}{\partial x(t)} + \frac{\partial \hat{x}(t+1)}{\partial x(t)} & \frac{\partial \hat{x}(t+1)}{\partial u(t)} \\ \frac{\partial \hat{u}(t+1)}{\partial \hat{x}(t+1)} \cdot \left(\frac{\partial \hat{x}(t+1)}{\partial u(t)} \cdot \frac{\partial u(t)}{\partial x(t)} + \frac{\partial \hat{x}(t+1)}{\partial x(t)} \right) & \frac{\partial \hat{u}(t+1)}{\partial \hat{x}(t+1)} \cdot \frac{\partial \hat{x}(t+1)}{\partial u(t)} \end{bmatrix} \tag{190}$$

Then, the weights updating rule is obtained based on the gradient decent method,

$$\omega_g(t+1) = \omega_g(t) + \Delta\omega_g(t) \tag{186}$$

where

$$\begin{aligned} \Delta\omega_g(t) &= -\beta_g \left[\frac{\partial E_g(t)}{\partial \omega_g(t)} \right] \\ &= -\beta_g \left[\eta_1 \frac{\partial E_{g1}(t)}{\partial \omega_g(t)} + \eta_2 \frac{\partial E_{g2}(t)}{\partial \omega_g(t)} \right] \\ &= -\beta_g \left[\eta_1 \left(\hat{s}(t) - \alpha \hat{s}(t+1) - r(t) \right) \frac{\partial \hat{s}(t)}{\partial \omega_g(t)} \right. \\ &\left. + \eta_2 \left(\frac{\partial \hat{s}(t)}{\partial Y(t)} - \alpha \frac{\partial \hat{s}(t+1)}{\partial Y(t)} - \frac{\partial r(t)}{\partial Y(t)} \right) \frac{\partial^2 \hat{s}(t)}{\partial Y(t) \partial \omega_g(t)} \right] \end{aligned} \tag{187}$$

in which β_g is the learning rate of the goal network. Here, we use $\omega_g(t)$ to express both $\omega_{g1}(t)$ and $\omega_{g2}(t)$. Note that, in the implementation process, equation (187) needs to be calculated in a component-by-component fashion.

Note that, in (187)

$$\frac{\partial \hat{s}(t)}{\partial Y(t)} = \hat{g}(t) \tag{188}$$

Thus, terms $\hat{s}(t)$ and $\partial \hat{s}(t)/\partial Y(t)$ can be directly obtained from the output of the goal network. Since we apply the model network to predict the future system state, then $\hat{J}(t+1)$, $\hat{g}(t+1)$, and $\hat{u}(t+1)$ can be obtain subsequently. Hence, we have

$$\frac{\partial \hat{s}(t+1)}{\partial Y(t)} = \frac{\partial \hat{s}(t+1)}{\partial \hat{Y}(t+1)} \cdot \frac{\partial \hat{Y}(t+1)}{\partial Y(t)} \tag{189}$$

The second term $\partial \hat{Y}(t+1)/\partial Y(t)$ can be achieved from the model network as (190). Thus, we have

$$\frac{\partial \hat{s}(t+1)}{\partial Y(t)} = \hat{g}(t+1) \cdot DER \tag{191}$$

Substituting (191) into (187), we can observe that only the terms $\partial\hat{s}(t+1)/\partial\omega_g(t)$ and $\partial^2\hat{s}(t)/\partial Y(t)\partial\omega_g(t)$ need to be solved. Note that in this chapter, both the input-to-hidden and the hidden-to-output layer weights $\omega_{g1}(t)$, $\omega_{g2}(t)$ are tuned during the learning process. Now, we will derive these two terms for both layers separately.

(1). $\Delta\omega_{g1}(t)$: Adjustment for the input-to-hidden layer weights of the goal network.

$$\begin{aligned}\frac{\partial\hat{s}(t)}{\partial\omega_{g1}(t)} &= \frac{\partial\hat{s}(t)}{\partial\sigma_g(t)}\cdot\frac{\partial\sigma_g(t)}{\partial\omega_{g1}(t)} \\ &= \frac{1}{2}\omega_{g2}^{aT}(t)\cdot\left(1-\sigma_g^2(t)\right)\cdot Y(t)\end{aligned} \tag{192}$$

$$\begin{aligned}\frac{\partial^2\hat{s}(t)}{\partial Y(t)\partial\omega_{g1}(t)} &= \frac{\partial\hat{g}(t)}{\partial\omega_{g1}(t)} \\ &= \frac{\partial\hat{g}(t)}{\partial\sigma_g(t)}\cdot\frac{\partial\sigma_g(t)}{\partial\omega_{g1(t)}} \\ &= \frac{1}{2}\omega_{g2}^{bT}(t)\cdot\left(1-\sigma_g^2(t)\right)\cdot Y(t)\end{aligned} \tag{193}$$

where $\sigma_g(t) = \sigma\left(\omega_{g1}^T(t)Y(t)\right)$.

(2). $\Delta\omega_{g2}(t)$: Adjustment for the hidden-to-output layer weights of the goal network.

$$\frac{\partial\hat{s}(t)}{\partial\omega_{g2}(t)} = \left[\frac{\partial\hat{s}(t)}{\partial\omega_{g2}^a(t)}, \frac{\partial\hat{s}(t)}{\partial\omega_{g2}^b(t)}\right] \tag{194}$$

$$\frac{\partial^2\hat{s}(t)}{\partial Y(t)\partial\omega_{g2}(t)} = \left[\frac{\partial\hat{g}(t)}{\partial\omega_{g2}^a(t)}, \frac{\partial\hat{g}(t)}{\partial\omega_{g2}^b(t)}\right] \tag{195}$$

5.5.3 Learning Process of Critic Network

The performance index $J(t)$ and its partial derivatives with respect to the vector $Y(t)$ are estimated by the critic network. The partial derivatives of the performance index can be defined as $\lambda(t) = \partial J(t)/\partial Y(t)$. The critic network is built as a three-layer neural network architecture and its schematic diagram is shown in Figure 24. It can be seen that we include the goal network's outputs $\hat{s}(t)$ and $\hat{g}(t)$ as parts of the critic network's input and aim to help the performance index approximation. In this way, the

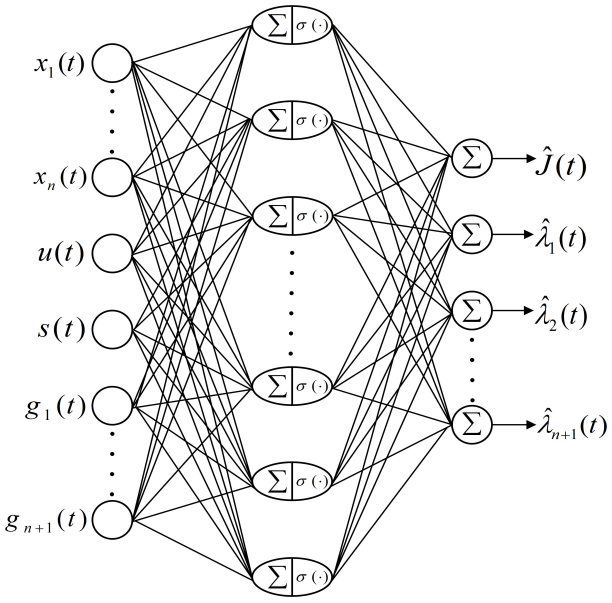

Figure 24. The schematic archtecture of the critic network.

critic and the goal networks are closely connected. The input of the critic network is defined as $Y_c = [x^T(t), u^T(t), \hat{s}(t), \hat{g}^T(t)]^T$. We also adopt the sigmoid function on the hidden layer nodes. Set the input-to-hidden layer weights as $\omega_{c1}(t)$ and the hidden-to-output layer weights as $\omega_{c2}(t) = [\omega_{c2}^a(t), \omega_{c2}^b(t)]$. Then, we have the output of the critic network:

$$\begin{bmatrix} \hat{J}(t) \\ \hat{\lambda}(t) \end{bmatrix} = \begin{bmatrix} \omega_{c2}^{aT}(t) \\ \omega_{c2}^{bT}(t) \end{bmatrix} \sigma\left(\omega_{c1}^T Y_c(t)\right) \tag{196}$$

The objective function for critic network can be described as

$$E_c(t) = \eta_1 E_{c1}(t) + \eta_2 E_{c2}(t) \tag{197}$$

where

$$E_{c1}(t) = \frac{1}{2}\left(\hat{J}(t) - \gamma\hat{J}(t+1) - \hat{s}(t)\right)^2 \tag{198}$$

$$E_{c2}(t) = \frac{1}{2}\left(\frac{\partial\hat{J}(t)}{\partial Y(t)} - \gamma\frac{\partial\hat{J}(t+1)}{\partial Y(t)} - \frac{\partial\hat{s}(t)}{\partial Y(t)}\right)^2 \tag{199}$$

Here, instead of the predefined (external) reinforcement signal $r(t)$ in literature, we provide an adaptive internal reinforcement signal $\hat{s}(t)$ for the critic network.

The gradient decent rule is employed to minimize the objective function in (197). Then, the weights updating rule of the critic network is:

$$\omega_c(t+1) = \omega_c(t) + \Delta\omega_c(t) \tag{200}$$

where

$$\begin{aligned}\Delta\omega_c(t) &= -\beta_c\left[\frac{\partial E_c(t)}{\partial\omega_c(t)}\right] \\ &= -\beta_c\left[\eta_1\frac{\partial E_{c1}(t)}{\partial\omega_c(t)} + \eta_2\frac{\partial E_{c2}(t)}{\partial\omega_c(t)}\right] \\ &= -\beta_c\left[\eta_1\Big(\hat{J}(t) - \gamma\hat{J}(t+1) - \hat{s}(t)\Big)\frac{\partial\hat{J}(t)}{\partial\omega_c(t)}\right. \\ &\left.+\eta_2\left(\frac{\partial\hat{J}(t)}{\partial Y(t)} - \gamma\frac{\partial\hat{J}(t+1)}{\partial Y(t)} - \frac{\partial\hat{s}(t)}{\partial Y(t)}\right)\frac{\partial^2\hat{J}(t)}{\partial Y(t)\partial\omega_c(t)}\right]\end{aligned} \tag{201}$$

in which β_c is the learning rate of the critic network. Here, we use $\omega_c(t)$ to express both $\omega_{c1}(t)$ and $\omega_{c2}(t)$.

It can be observed that

$$\frac{\partial\hat{J}(t)}{\partial Y(t)} = \hat{\lambda}(t); \quad \frac{\partial\hat{s}(t)}{\partial Y(t)} = \hat{g}(t) \tag{202}$$

Hence, we can obtain these two terms in (201) from the goal and the critic networks directly. Based on the model network, we have

$$\frac{\partial\hat{J}(t+1)}{\partial Y(t)} = \hat{\lambda}(t+1)\cdot DER \tag{203}$$

Now, we will derive the adjustment for the hidden and the output layers weights, respectively.

(1). $\Delta\omega_{c1}(t)$: Adjustment for the input-to-hidden layer weights of the critic network.

$$\frac{\partial\hat{J}(t)}{\partial\omega_{c1}(t)} = \frac{1}{2}\omega_{c2}^{aT}(t)\cdot\Big(1-\sigma_c^2(t)\Big)\cdot Y_c(t) \tag{204}$$

$$\begin{aligned}\frac{\partial^2\hat{J}(t)}{\partial Y(t)\partial\omega_{c1}(t)} &= \frac{\partial\hat{\lambda}(t)}{\partial\omega_{c1}(t)} \\ &= \frac{1}{2}\omega_{c2}^{bT}(t)\cdot\Big(1-\sigma_c^2(t)\Big)\cdot Y_c(t)\end{aligned} \tag{205}$$

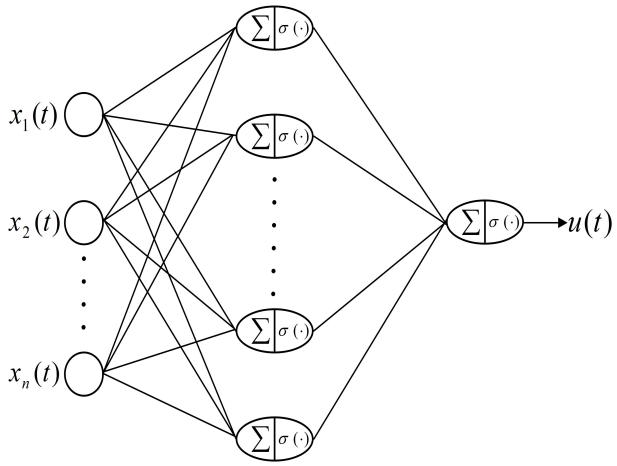

Figure 25. The schematic archtecture of the action network.

where $\sigma_c(t) = \sigma\left(\omega_{c1}^T(t)Y_c(t)\right)$

(2). $\Delta\omega_{c2}(t)$: Adjustment for the hidden-to-output layer weights of the critic network.

$$\frac{\partial \hat{J}(t)}{\partial \omega_{c2}(t)} = \left[\frac{\partial \hat{J}(t)}{\partial \omega_{c2}^a(t)}, \frac{\partial \hat{J}(t)}{\partial \omega_{c2}^b(t)}\right] \tag{206}$$

$$\frac{\partial^2 \hat{J}(t)}{\partial Y(t)\partial \omega_{c2}(t)} = \left[\frac{\partial \hat{\lambda}(t)}{\partial \omega_{c2}^a(t)}, \frac{\partial \hat{\lambda}(t)}{\partial \omega_{c2}^b(t)}\right] \tag{207}$$

5.5.4 Learning Process of Action Network

The optimal control action $u(t)$ is approximated by the action network. Recall the GrHDP design [4], this goal is achieved by minimizing the performance index or total future cost. Although the performance index $J(t)$ is one of the outputs of the critic network in the proposed Gr-GDHP design, we also have high quality estimation of $\partial J(t)/\partial s(t)$ at the critic's outputs. Therefore, similar to the GrDHP design [12], we use $\partial J(t)/\partial s(t)$ instead to perform the training of the action network [15]. Hence, the approximation error for the action network can be defined as

$$e_a(t) = \frac{\partial \hat{s}(t)}{\partial u(t)} + \gamma \frac{\partial \hat{J}(t+1)}{\partial u(t)} \tag{208}$$

$$E_a(t) = \frac{1}{2}e_a^2(t) \tag{209}$$

The estimated control law can be formulated as

$$u(t) = \sigma\Big(\omega_{a2}^{T}(t)m(t)\Big) \tag{210}$$

$$m(t) = \sigma\Big(\omega_{a1}^{T}(t)x(t)\Big) \tag{211}$$

where $\omega_{a1}(t)$ and $\omega_{a2}(t)$ are the action network weights of the input-to-hidden and hidden-to-output layers, respectively. The input of the action network is the system state $x(t)$.

The weights updating law is derived based on the gradient descent method as

$$\omega_a(t+1) = \omega_a(t) + \Delta\omega_a(t) \tag{212}$$

where

$$\begin{aligned}\Delta\omega_a(t) &= -\beta_a\left[\frac{\partial E_a(t)}{\partial\omega_a(t)}\right] \\ &= -\beta_a\left[\left(\frac{\partial\hat{s}(t)}{\partial u(t)} + \gamma\frac{\partial\hat{J}(t+1)}{\partial u(t)}\right)\left(\frac{\partial^2\hat{s}(t)}{\partial u(t)\partial\omega_a(t)} + \gamma\frac{\partial^2\hat{J}(t+1)}{\partial u(t)\partial\omega_a(t)}\right)\right]\end{aligned} \tag{213}$$

In this Gr-GDHP design, the goal and the critic networks are built to estimate the internal reinforcement signal $s(t)$, performance index $J(t)$, and their derivatives $g(t)$ and $\lambda(t)$. Thus, the second derivatives $\partial^2 s(t)/\partial Y(t)\partial\omega_g(t)$ and $\partial^2 J(t)/\partial Y(t)\partial\omega_c(t)$ are conveniently obtained through backpropagation. In general, the use of both the optimization criterion and its derivatives is regarded as being more critical to seek the optimal solution [15]. Furthermore, the training process of Gr-GDHP is in the order of the goal network, the critic network, and the action network. Specifically, after the weights $\omega_{g1}(t)$, $\omega_{g2}(t)$ of the goal network are learned, we fix them thereafter and start to train the weights $\omega_{c1}(t)$, $\omega_{c2}(t)$ of the critic network. Then, we fix $\omega_{c1}(t)$, $\omega_{c2}(t)$ and start to train the weights $\omega_{a1}(t)$, $\omega_{a2}(t)$ of the action network. For each time step, the designed neural networks have their own internal cycles N_g, N_c, N_a and training error thresholds T_g, T_c, T_a. For instance, the goal network is trained until the squared error under the threshold T_g, otherwise it is trained at most N_g cycles in each time step.

5.6 Simulation Studies

In this section, we provide two case studies to demonstrate the effectiveness of the proposed Gr-GDHP method. We also compare the results with the GrDHP, GrHDP and GDHP methods. The comparison shows that the proposed Gr-GDHP method can improve the control performance.

5.6.1 Nonlinear System

Consider the following nonlinear system

$$\begin{cases} x_1(t+1) = -\sin\left(0.5x_1(t)\right) \\ x_2(t+1) = -\sin\left(x_1(t)\right)\cos\left(0.2x_1(t) + 0.8x_2(t)\right) + u(t) \end{cases} \tag{214}$$

where $x(t) = [x_1(t), x_2(t)]^T$ is the system state vector and $u(t)$ is the control action. The external reinforcement signal is chosen as $r(t) = x^T(t)x(t) + u^T(t)u(t)$. We can observe that $x(t) = 0$ is an equilibrium state of the system.

We apply the proposed Gr-GDHP method to stabilize the nonlinear system (214). In the system identification process, a three-layer neural network is established as the model network with the structure $3 - 6 - 2$ (i.e., the network has three input nodes, six hidden nodes, and two output nodes). The initial weights of the input-to-hidden and the hidden-to-output layer are chosen randomly within $[-1, 1]$ and the learning rate of the model network is set as $\beta_m = 0.01$. After training, the model network can successfully predict the unknown future state of the nonlinear system (214). Then, the offline training process is finished and we fix the well-trained model network weights.

Three neural networks are built as the goal, the critic, and the action networks with three-layer structure. Specifically, the structures of these three neural networks are $3 - 6 - 4$, $7 - 10 - 4$, and $2 - 5 - 1$, respectively. All the initial weights are randomly chosen within $[-0.5, 0.5]$. The weights training processes are based on the analysis in Section 6.3. The learning rates of the goal, the critic, and the action networks are set as $\beta_g = \beta_c = \beta_a = 0.01$. In this case study, we define the number of internal cycles

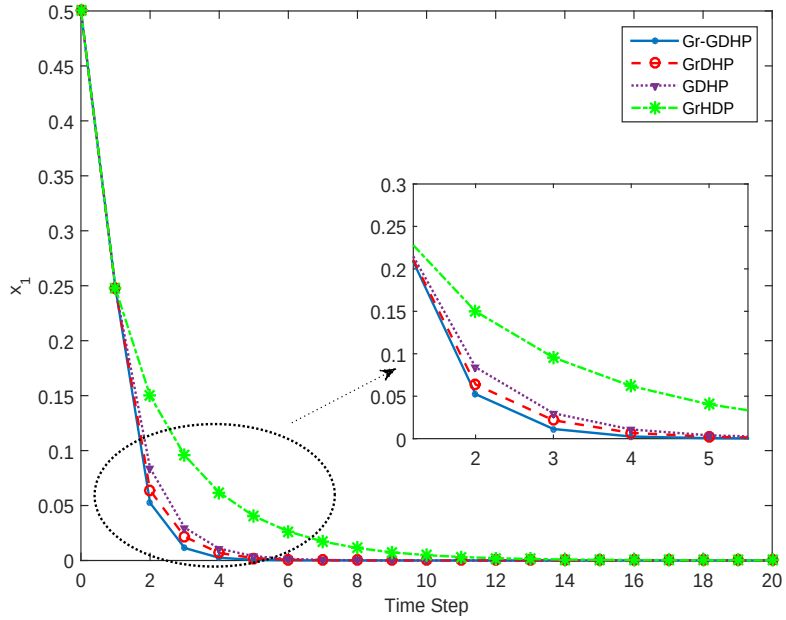

Figure 26. Comparison of the state trajectory x_1 on the nonlinear system with Gr-GDHP, GrDHP, GrHDP and GDHP methods.

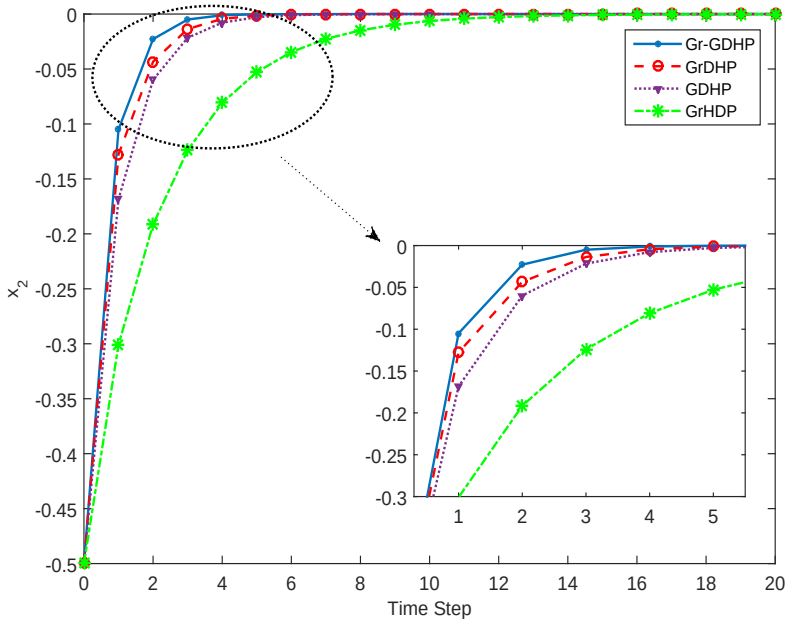

Figure 27. Comparison of the state trajectory x_2 on the nonlinear system with Gr-GDHP, GrDHP, GrHDP and GDHP methods.

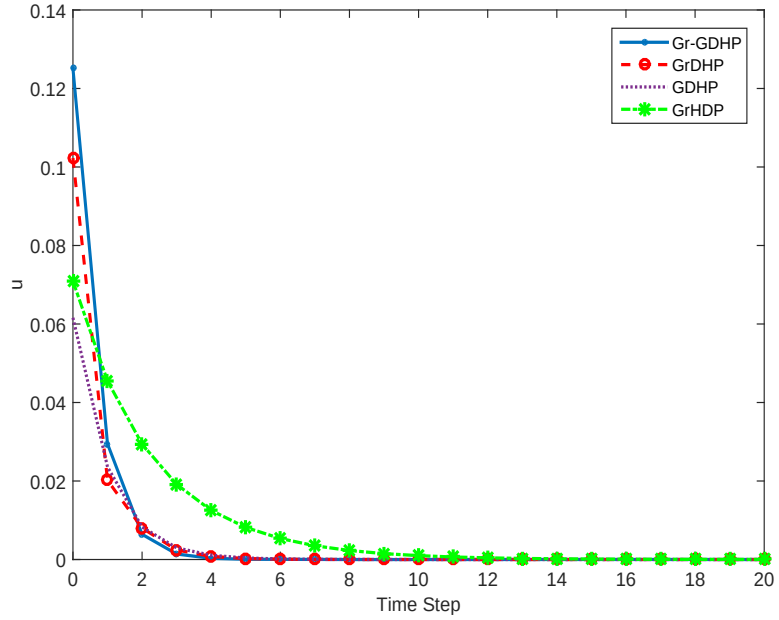

Figure 28. Comparison of the control input u on the nonlinear system with Gr-GDHP, GrDHP, GrHDP and GDHP methods.

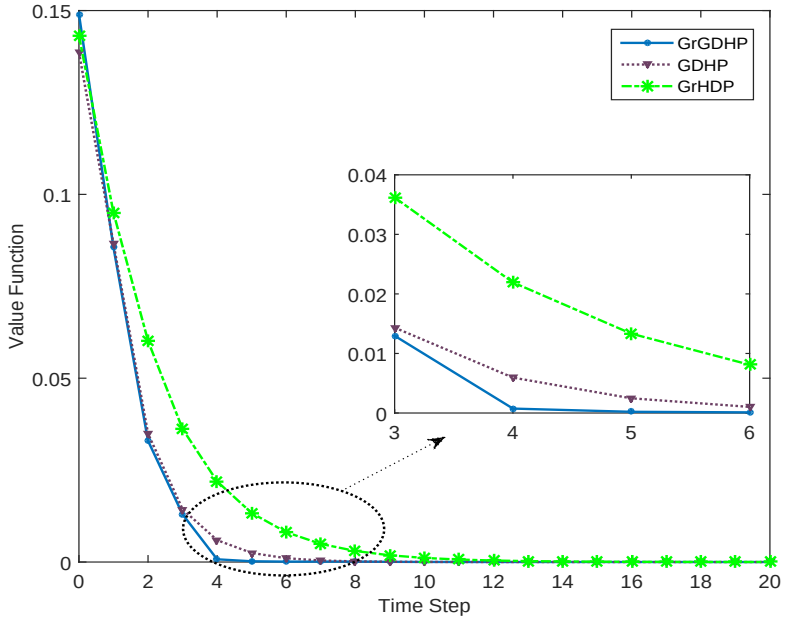

Figure 29. The trajectories of the performance index $\hat{J}(t)$ of the Gr-GDHP, GrHDP and GDHP methods.

as N_g = 150, N_c = 100, N_a = 200 and the error threshold as T_g = T_c = T_a = $1e-6$. The discounted factors are set as α = γ = 0.95. The weighted parameters are set as η_1 = η_2 = 0.5.

In order to verify the effectiveness of the Gr-GDHP method, we compare this method with three other ADP methods in literature, which are GDHP in [15], GrHDP in [4], and GrDHP in [12]. The initial state is set as $x(0) = [0.5, -0.5]^T$. We apply the optimal controller designed by these four methods and compare the performance. The compared state trajectories of these four methods are shown in Figure 26 and Figure 27. The comparison of the control input $u(t)$ of these four methods is provided in Figure 28. It can be observed that all the methods can stabilize the system. Moreover, the proposed Gr-GDHP method can drive the system states to converge to the equilibrium points faster than other three methods. Furthermore, the trajectories of the performance index $J(t)$ of the Gr-GDHP, GrHDP, and GDHP methods are shown in Figure 29. Note that since the performance index $\hat{J}(t)$ cannot be directly obtained in the GrDHP design, we only compare the trajectories of $\hat{J}(t)$ for the other three methods here. From Figure 29, we can observe that the proposed Gr-GDHP method can minimize the performance index faster than the other methods. These simulation results demonstrate that the proposed Gr-GDHP design has better performance than the GrHDP, GrDHP, and GDHP designs.

5.6.2 Ball-and-beam balancing system

In this case study, the effectiveness of the proposed Gr-GDHP method is further investigated on the ball-and-beam balancing system [6, 19], which is shown in Figure 30. The motion equations can be described as:

$$\left(m + \frac{I_b}{r^2}\right)\ddot{x}' + \left(mr^2 + I_b\right)\frac{1}{r}\ddot{\theta} - mx'\dot{\theta}^2 = mg(\sin\theta) \tag{215}$$

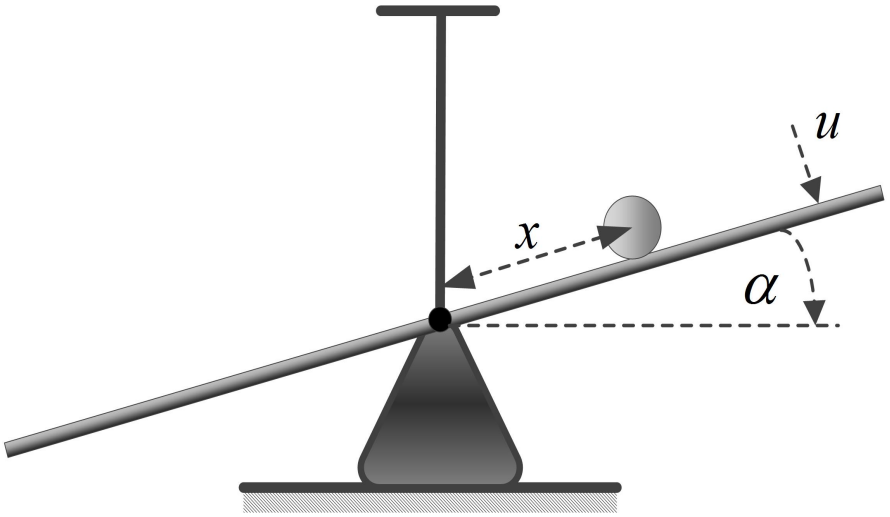

Figure 30. Schematics of the ball-and-beam balancing system.

$$\begin{aligned}&\left[m(x')^2 + I_b + I_\omega\right]\ddot{\theta} + \left(2m\dot{x}'x' + bl^2\right)\dot{\theta} + Kl^2\theta \\ &\quad + \left(mr^2 + I_b\right)\frac{1}{r}\ddot{x}' - mgx'(\cos\theta) = ul(\cos\theta)\end{aligned} \tag{216}$$

where

$b = 1N_s/m$, is the friction coefficient of the drive mechanics;

$m = 0.0162kg$, is the mass of the ball;

$g = 9.8m/s^2$, is the acceleration due to gravity;

$r = 0.02m$, is the roll radius of the ball;

$K = 0.001N/m$, is the stiffness of the drive mechanics;

$I_b = 4.32 \times 10^{-5}kg \cdot m^2$, is the inertia moment of the ball;

$I_\omega = 0.14025kg \cdot m^2$, is the inertia moment of the beam;

$l = 0.48m$, is the radius of force application;

$l_\omega = 0.5m$, is the radius of the beam;

x', is the position of the ball;

θ, is the angle of the beam to the horizontal axis;

u, is the force of the drive mechanics.

To formulate the system dynamics, assume $x_1 = x'$ is the position of the ball, $x_2 = \dot{x}'$

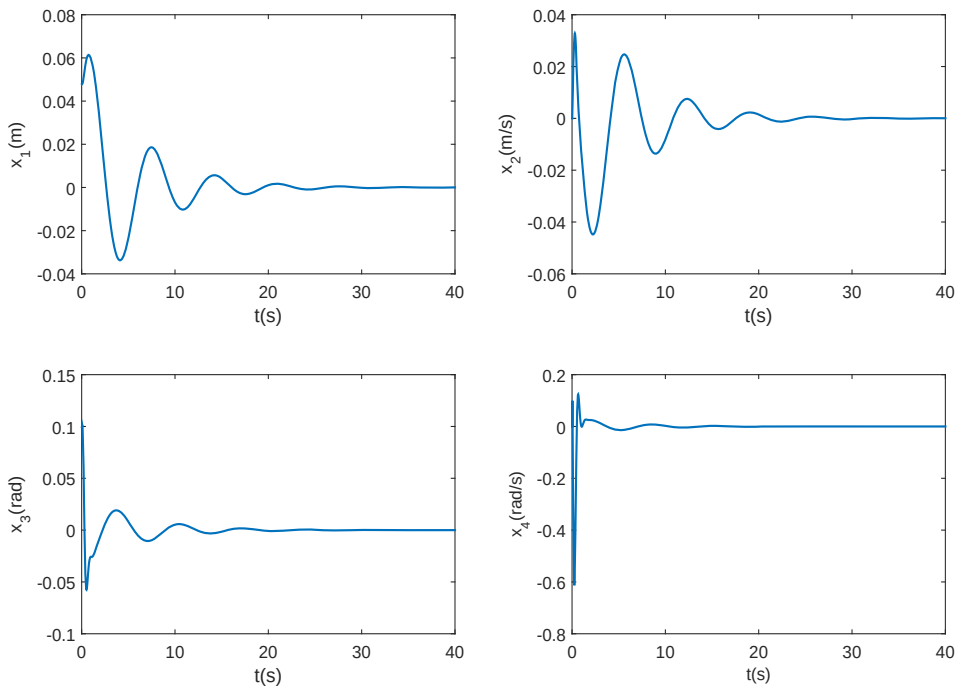

Figure 31. System responses of a typical successful trial without noise in the first 40 seconds.

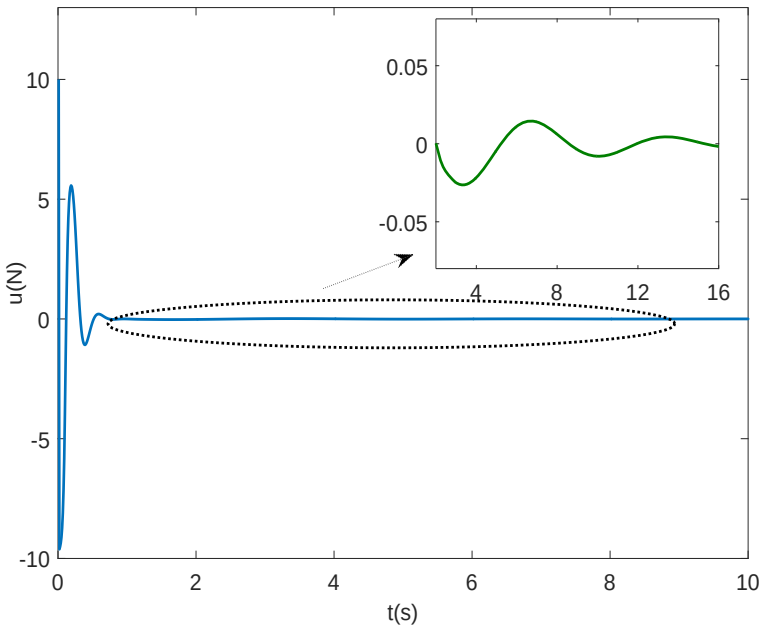

Figure 32. Typical trajectory of control action in the first 20 seconds in a typical successful trial without noise.

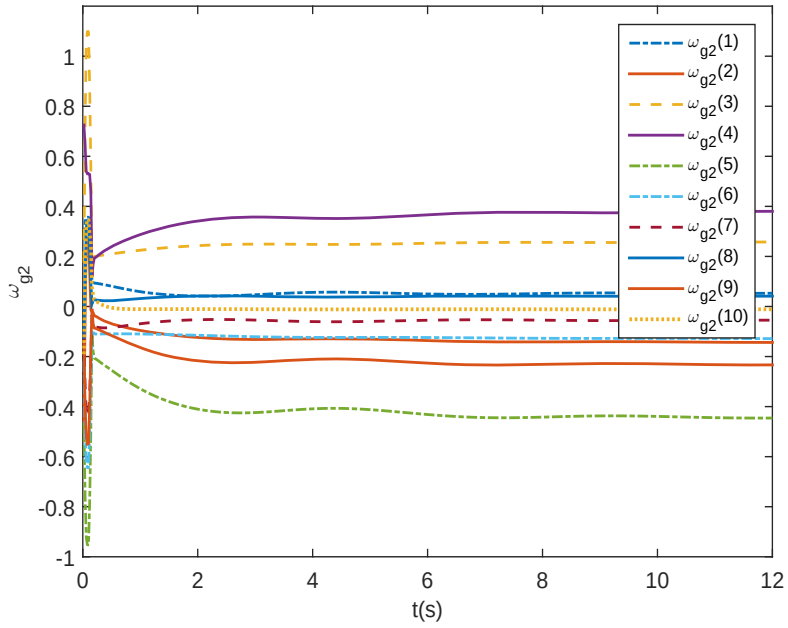

Figure 33. The weights evolution in goal network from ten hidden layer nodes to the first output layer node of a typical successful trial without noise in the first 12 seconds.

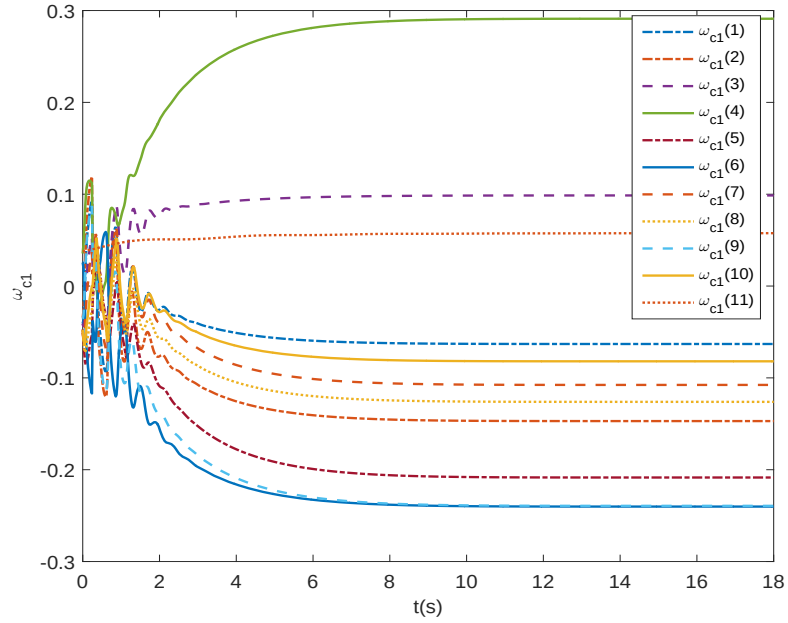

Figure 34. The weights evolution in critic network from eleven input layer nodes to the first hidden layer node of a typical successful trial without noise in the first 18 seconds.

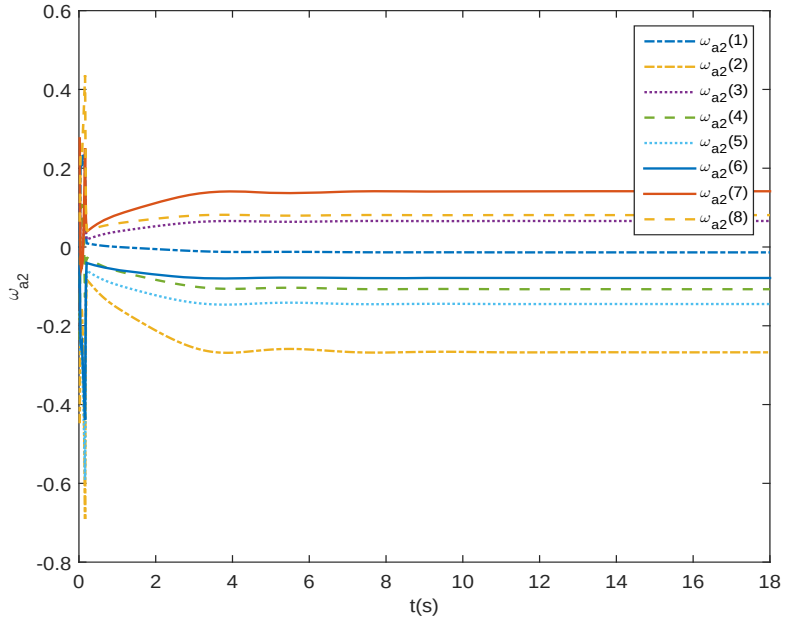

Figure 35. The weights evolution in action network from eight hidden layer nodes to the output layer node of a typical successful trial without noise in the first 18 seconds.

is the velocity of the ball, $x_3 = \theta$ is the angle of the beam to the horizontal axis, and $x_4 = \dot{\theta}$ is the angular velocity of the beam. Therefore, after substituting the physical value of each parameter, we obtain the following corresponding nonlinear state space function:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 1.717 \sin(x_3) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -0.241 x_4 + 0.157 x_1 \cos(x_3) + 0.5 \cos(x_3) \cdot u \end{cases} \tag{217}$$

Our goal is to balance the ball on the beam for a certain period of time. If any trial of the run can last 6000 time steps, we consider it as a successful run. We run the simulations 100 times with a maximum of 1000 consecutive trials in each run. This means the designed controller needs to balance the ball within some certain range for 6000 time steps within 1000 consecutive trials to be regarded as a successful run. The ball is considered fallen if the position of the ball on the track is out of $[-0.48m, 0.48m]$,

or if the angle of the beam to the horizontal axis is out of $[-0.24rad, 0.24rad]$. The force u is starting from u = $10N$ which is applied on the system at the beginning of each learning process. In our current simulation, the sampling period is chosen as T = $0.02s$.

We apply the proposed Gr-GDHP method to design the controller. Three neural networks are established as the goal, the critic, and the action networks. Since there are four system states in this case, the multilayer perceptron structures for the goal, the critic, and the action network are chosen as $5-10-6$, $11-18-6$, and $4-8-1$, respectively. Set the external reinforcement signal as $r(t)$ = -1 when the ball is fallen. Otherwise, we define $r(t)$ = 0. The parameters used in this example are summarized in Table 1. Note that the learning rate $\beta_g(t)$ is initialized as $\beta_g(0)$ and will dropped 0.05 every 10 steps until it reach $\beta_g(f)$ and stay thereafter. Here, we assume that the learning rates of the critic and the action networks have the same settings as those of the goal network.

Table 1. Summary of the parameters used in the case study A

Parameter	$\beta_g(0)$	$\beta_c(0)$	$\beta_a(0)$	N_g	N_c
Value	0.3	0.3	0.3	80	80
Parameter	N_a	$\beta_g(f)$	$\beta_c(f)$	$\beta_a(f)$	T_g
Value	100	0.005	0.005	0.005	$1e-6$
Parameter	T_c	T_a	α/γ	$*$	$*$
Value	$1e-6$	$1e-6$	0.95	$*$	$*$

where

$\beta_g(0)$ is the initial learning rate of the goal network;

$\beta_c(0)$ is the initial learning rate of the critic network;

$\beta_a(0)$ is the initial learning rate of the action network;

N_g is the internal cycle of the goal network;

N_c is the internal cycle of the critic network;

N_a is the internal cycle of the action network;

$\beta_g(f)$ is the lower bound of the learning rate of the goal network;

$\beta_c(f)$ is the lower bound of the learning rate of the critic network;

$\beta_a(f)$ is the lower bound of the learning rate of the action network;

T_g is the training error threshold for the goal network;

T_c is the training error threshold for the critic network;

T_a is the training error threshold for the action network;

α, γ are the discount factors.

The action network will provide a control force u to balance the ball on the beam. The initial values of the ball position x_1 and the beam angle x_3 are randomly chosen in $[-0.2m, 0.2m]$ and $[-0.15rad, 0.15rad]$, respectively. The initial values of the ball velocity x_2 and the angular velocity x_4 are set as zero. In order to make the problem more realistic, we considered both the sensor and actuator noise. Specifically, the sensor noise is added to the state measurements and the actuator noise is added to the output of the action network. For instance, if the noise level is 5%, we implement the uniform noise through $x/u + 0.05 \cdot x/u \cdot random(-1, 1)$.

Table 2. Comparison of the statistical simulation results on the ball-and-beam balancing system with the Gr-GDHP and the GDHP controller

Noise type	$Gr-GDHP$	$GDHP$
Noise free	4.1	11.4
Uniform 5% $a.^*$	5.2	11.2
Uniform 10% a	5.7	18.6
Uniform 5% $x.^+$	4.9	15.9
Uniform 10% $x.$	5.5	21.5

$a.^*$: actuator sensor noise
$x.^+$: position sensor noise on x_3

We also apply the GDHP method to do the same example for 100 times and compare the statistical results with the proposed Gr-GDHP method. The required average numbers of trials to success are provided in Table 2. For this ball-and-beam balancing case, both the Gr-GDHP and the GDHP methods can achieve 100% successful rate under various noise conditions. However, comparing with the GDHP method, the proposed

method needs less average number of trials to successfully learn balancing the ball under the same noise type. This indicates that the Gr-GDHP method can improve the performance by requiring less number of trials to balance the ball on the beam. Furthermore, different types of noise does not affect the required average number of trials. It indicates that the proposed approach is very robust. We also study the computational cost per time step in the successful run for both methods. We account for the backpropagation training time in every time step and calculate the average value. After calculation, we obtain that the average computational cost in one backpropagation training for the Gr-GDHP method is $0.0167s$, while for the traditional GDHP method is $0.0118s$. This indicates that the proposed Gr-GDHP method can achieve better performance with competitive computational cost.

Furthermore, we provide the typical trajectories of the state variables in the first $40s$ (2000 time steps) of a typical successful trial without noise in Figure 31. It can be clearly observed that the ball can keep staying in the middle of the beam after several seconds. The corresponding evolution of the control action in the first $20s$ (1000 time steps) is shown in Figure 32. The control starts from $u(0) = 10$ and converges to zero in the end. Figure 33 provides the weights trajectories in the goal network from ten hidden layer nodes to the first output layer node of a typical successful trail during the first $12s$ (600 time steps). The weights evolution in the critic network from eleven input layer nodes to the first hidden layer node during the first $18s$ (900 time steps) is provided in Figure 34. Moreover, Figure 35 shows the weights trajectories in the action network from eight hidden layer nodes to the output layer node during the first $18s$ (900 time steps). From these results, we know the neural network weights start from small values round zero and converge after a few seconds. These simulation results indicate the promising performance of the proposed Gr-GDHP approach in the learning and control process.

5.7 Discussions

In recent years, goal representation adaptive dynamic programming (GrADP) design has been developed to improve the online learning and control performance of the traditional ADP methods. Specifically, by integrating an additional neural network, goal network, into the traditional adaptive critic design, the GrHDP method can obtain an internal reinforcement signal $s(t)$ to facilitate the optimal learning process. This $s(t)$ includes the information of the future external reinforcement signals, which means the internal reinforcement signal can give us more information by considering more distant lookahead [14]. The designed $s(t)$ also inputs to the critic network to help estimate the performance index $J(t)$. This architecture has been applied to various simulation studies and many realistic complex applications [5, 6, 8, 9, 7]. Promising capability and impressive performance were achieved. The theoretical foundation of the GrHDP method is provided in [14]. Later, based on the GrHDP design, the goal representation DHP design has been proposed and tested on numerous challenging applications including the multimachine power system control problem [12]. In the GrDHP design, the goal and the critic network build a representation for the partial derivatives of $s(t)$ and $J(t)$, respectively. The derivatives of the internal reinforcement signal $g(t)$ (i.e., $g(t) = \partial s(t)/\partial Y(t)$, $Y(t) = [x(t), u(t)]^T$) is generated internal within the GrDHP design to help the learning process. This method has been applied on many complex applications and showed the better performance comparing to the traditional DHP design.

In this chapter, we follow our previous work and develop the Gr-GDHP method by building the general mapping from the system states and actions to the signals $s(t)$ and $J(t)$, as well as their derivatives $g(t)$ and $\lambda(t)$. Note that the Gr-GDHP design is not an easy combination of the GrHDP and GrDHP design. The goal and the critic networks in Gr-GDHP directly estimate not only the internal reinforcement signal $s(t)$, and performance index $J(t)$, but also their derivatives $g(t)$ and $\lambda(t)$, respectively. Specifically,

the outputs of the goal network, which are the signal $s(t)$ and its derivatives $g(t)$, can directly provide information of the error function for the critic network. Moreover, $s(t)$ and $g(t)$ are also set as parts of the inputs for the critic network to support the approximation of performance index $J(t)$ and its derivatives $\lambda(t)$. Knowing $s(t)$ and $J(t)$, as well as their derivatives is important in the problem where the availability of the information associated with $s(t)$ and $J(t)$ themselves is as important as knowledge of the slope of $s(t)$ and $J(t)$, respectively [15]. Furthermore, any adjustment of combination for the values of $s(t)$ and $J(t)$ or their derivatives $g(t)$ and $\lambda(t)$ can be accommodated by selecting the weighted parameters η_1 and η_2 in (184) and (197).

A model network is built for the Gr-GDHP method in this chapter. The design of the model network here is similar with the model network established in the traditional ADP methods [15, 20, 12]. The goal of the model network is to predict the future system state $\hat{x}(t+1)$, and then we can obtain the subsequent internal reinforcement signal and the performance index for the next time step. Following the idea in literature, the training process of the model network is offline in this chapter. In this way, we can compare the performance of the proposed method with the existing ADP methods in literature. Moreover, in this design, we set the discount factors for both the internal reinforcement signal and the performance index as $0 < \alpha < 1$, and $0 < \gamma < 1$ to make sure $s(t)$ and $J(t)$ are finite [15]. Note that this is not a necessary condition for the finite horizon problem. For certain designs of the external reinforcement signal $r(t)$, like the quadratic form, the finite horizon can also be satisfied. In these situations, we can set the discount factors equal to 1.

This chapter integrates an internal reinforcement signal $s(t)$ into the traditional GDHP design. By containing the information of future external reinforcement signals, $s(t)$ can facilitates the learning process. Since a new method is proposed, it is very important to consider its stability. In [14], we developed the convergence and the sta-

bility analysis of the GrHDP method with a rigorous theoretical proof. It was proved that the internal reinforcement signal had an upper bound and the performance index in the GrHDP design could converge to its optimal value. The existence of the admissible control in this learning process was also provided. Then, we went further to consider the stability of the GrDHP design and presented a theoretical foundation for the GrDHP in terms of the partial derivatives for both the internal reinforcement signal and the performance index [21]. It is shown that the partial derivatives of the internal reinforcement signal are bounded signals. And the partial derivatives of the performance index can converge to their optimal values when the number of iteration step went to infinity. Since the Gr-GDHP method is a weighted combination of the GrHDP and GrDHP methods, it is expected that there exists upper bounds for the internal reinforcement signal $s(t)$ and its derivatives $g(t)$. Moreover, $J(t)$ and $\lambda(t)$ are expected to converge to their optimal values respectively.

The proposed Gr-GDHP method is studied on two simulation examples to test its performance in this chapter. In the first case study, we apply this method on a nonlinear system and compare the results with other existing ADP methods, i.e., GDHP, GrHDP, and GrDHP. From the results, we can observe that all the methods can stabilize the system, and the proposed Gr-GDHP method has a faster speed comparing with other methods. Then, this method has been tested on a more complex example, the ball-and-beam balancing system. The comparison of the proposed method with the traditional GDHP method is provided. Generally, the goal of this case is to balance the ball on the beam for a certain period of time. The simulation results show that both of these two methods can achieve 100% successful rate under various noise conditions. Moreover, the proposed method performs better results in terms of the number of trails to successfully learn balancing the ball under the same noise type.

5.8 Summary

This chapter presents an advanced ADP method, which is the Gr-GDHP control design. Starting from the general formulation of the Gr-GDHP method, a neural-network-based architecture is proposed to implement this approach. Then, the explicit learning process of the goal, the critic, and the action networks is discussed, respectively. The weights updating rules of both the input-to-hidden layer and the hidden-to-output layer are also provided. A nonlinear system and a ball-and-beam balancing system are applied to verify the proposed method. The simulation results demonstrate the effective control performance of the proposed Gr-GDHP method comparing with other ADP designs.

List of References

[1] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.

[2] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, London, UK, 1999.

[3] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.

[4] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon78.html#HeNF12

[5] H. He, *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.

[6] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 913–928, 2013.

[7] Z. Ni, H. He, D. Zhao, and D. V. Prokhorov, "Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–8.

[8] Z. Ni, H. He, J. Wen, and X. Xu, "Goal representation heuristic dynamic programming on maze navigation," *Neural Networks, IEEE Transactions on*, vol. 24, pp. 2038–2050, Dec. 2013.

[9] Z. Ni and H. He, "Heuristic dynamic programming with internal goal representation," *Soft Computing*, vol. 17, pp. 2101–2108, 2013.

[10] X. Luo, J. Si, and Y. Zhou, "An integrated design for intensified direct heuristic dynamic programming," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*. IEEE, 2013, pp. 183–190.

[11] J. Chen and Z. Li, "A novel adaptive tropism reward ADHDP method with robust property," in *Advances in Brain Inspired Cognitive Systems*. Springer, 2013, pp. 288–295.

[12] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "Grdhp: A general utility function representation for dual heuristic dynamic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 3, pp. 614–627, 2015.

[13] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 942–949, 2008.

[14] X. Zhong, Z. Ni, and H. He, "A theoretical foundation of goal representation heuristic dynamic programming," ieee trans. on neural networks and learning systems," in press, 2015.

[15] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 8, no. 5, pp. 997–1007, 1997.

[16] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 3, pp. 628–634, 2012.

[17] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Dual heuristic programming excitation neurocontrol for generators in a multimachine power system," *Industry Applications, IEEE Transactions on*, vol. 39, no. 2, pp. 382–394, 2003.

[18] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *Neural Networks, IEEE Transactions on*, vol. 6, no. 6, pp. 1320–1329, 1995.

[19] T.-L. Chien, C.-C. Chen, Y.-C. Huang, and W.-J. Lin, "Stability and almost disturbance decoupling analysis of nonlinear system subject to feedback linearization and feedforward neural network controller," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1220–1230, 2008.

[20] D. Liu and D. Wang, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Optimal Control of Unkonwn Nonlinear Discrete-Time Systems Using the Iterative Globalized Dual Heuristic Programming Algorithm, pp. 52–74.

[21] X. Zhong, Z. Ni, and H. He, "Convergence analysis of grdhp-based optimal control for discrete-time nonlinear system," in *Neural Networks (IJCNN), The 206 International Joint Conference on*. IEEE, 2016, pp. 1–8.

CHAPTER 6

On-line ADP Learning for Markov Jump Systems (MJSs)

6.1 Introduction

Markov jump systems (MJSs) have witnessed extensive studies in recent years because of their powerful modeling capability for power systems, network control systems, and manufacturing systems [1], [2]. These systems include abrupt variations in their structures due to sudden environmental disturbances and subsystems interconnection variations. Therefore these systems are inherently vulnerable to component failure or repairs and hard to be modeled. Due to the wide spectrum of applications of MJSs, there has been extensive researches in the stability analysis [3], controller design [4], [5], and filtering [6], [7]. The study of MJSs has attracted considerable attention in recent years. Most of the results of MJSs are obtained under the full information of system dynamics, but in many practical situation, the system dynamics cannot easily be obtained exactly. In order to solve this problem, in literature, Chen *et al* designed a memoryless state feedback controller for uncertain MJSs to guarantee the closed-loop cost function value was not more than a specific level of performance for any admissible uncertainties [5]. In [8], an optimal estimator for the current state was designed according to current and past observations to overcome the system parameters varying. Farias *et al* introduced the ADP method into the stochastic control problem in [9] and approximated the optimal control law via linear programming. In [10], they defined this algorithm as approximate linear programming and provided the detailed theoretical and simulation results.

This chapter develops an adaptive learning method for a class of unknown discrete-time nonlinear MJSs based on adaptive dynamic programming (ADP) technique. Specifically, we propose an optimal control scheme to convert the MJSs control problem with multiple subsystems into a single objective optimal control problem. That

is, the performance index functions of all the subsystems in MJSs are combined into one performance index function depending on the Markov chain and the weighted sum technique. The ADP technique is introduced into the field of MJSs to solve this kind of problem. Unlike the traditional method, such as the linear matrix inequality (LMI) technique, our approach based on ADP technique includes the adaptive and learning capability of the system dynamics, indicating that our approach can still find the near optimal controller even if the system parameters change. The theoretical analysis is developed in this chapter which is focused on the stability of the proposed ADP approach for MJSs. The convergence of the proposed performance index function and the existence of the corresponding control law are provided. These are also verified by the simulation studies.

6.2 Problem Statement

Consider the unknown discrete-time nonlinear Markov jump systems (MJSs) of the following form

$$x_{k+1} = f_i(x_k) + g_i(x_k)u_k \tag{218}$$

where $x_k \in R^n$ denotes the system state with the initial value x_0, $u_k \in R^l$ is the system input, and i is the simplified notation of a discrete-time Markov chain $\{r_k\}$, of which taking values in a finite state space $S = \{1, 2, \cdots, m\}$, where m is the number of the subsystems. Assume that $f + gu$ is Lipschitz continuous on a set $\Omega \subseteq R^n$ containing the origin. $f_i(x_k)$ and $g_i(x_k)$ are the unknown discrete-time state functions and $f_i(0) = 0$, $g_i(0) = 0$, which means the system state $x_k = 0$ is an equilibrium point of system (218) under the control $u_k = 0$.

Define the transition probability matrix for discrete-time MJSs as

$$\mathrm{H} = \begin{pmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1m} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{m1} & \pi_{m2} & \cdots & \pi_{mm} \end{pmatrix}. \tag{219}$$

The elements in (219) can be expressed by

$$\pi_{ab} = \mathbf{Pr}\{r_{k+1} = b | r_k = a\} \tag{220}$$

which denotes the probability of the next system mode b, given the current mode a. Therefore, we can easily obtain that $\pi_{ab} \geq 0$, $\forall a, b \in S$ and for each subsystem a, $\sum_{b=1}^{m} \pi_{ab} = 1$.

Assume that the MJSs (218) are completely controllable and bounded on $\Omega \in R^n$. The performance index function for each subsystem can be described by

$$J_i(x_k) = \sum_{z=k}^{\infty} U_i(x_z, u_z) \tag{221}$$

where the utility function can be chosen as

$$U_i(x_k, u_k) = Q_i(x_k) + u_k^T R_i u_k \tag{222}$$

in which $Q_i(x_k)$ and R_i are positive definite. This means $U_i(x_k, u_k)$ is positive definite, i.e., if and only if $x_k = 0$ and $u_k = 0$, $U_i(0, 0) = 0$, otherwise, $U_i(x_k, u_k) > 0$.

An equivalent equation to (221) is given by the Bellman equation

$$\begin{aligned} J_i(x_k) &= U_i(x_k, u_k) + \sum_{z=k+1}^{\infty} U_i(x_z, u_z) \\ &= U_i(x_k, u_k) + J_i(x_{k+1}) \end{aligned} \tag{223}$$

where $i \in S$.

The purpose of this chapter is to find the optimal control law u_k^*, so as to minimize the performance index function of the whole MJSs and stabilize the MJSs. However, due to the existence of the transition probabilities, we cannot just add all the performance index functions of the subsystems together as the final one for the MJSs. Here, we reconstruct the performance index function (221) of the subsystems by using the

transition probability matrix (219) as follows

$$\begin{cases} J_I(x_k) = \pi_{11}J_1(x_k) + \pi_{12}J_2(x_k) + \cdots + \pi_{1m}J_m(x_k) \\ J_{II}(x_k) = \pi_{21}J_1(x_k) + \pi_{22}J_2(x_k) + \cdots + \pi_{2m}J_m(x_k) \\ \qquad\qquad \vdots \\ J_M(x_k) = \pi_{m1}J_1(x_k) + \pi_{m2}J_2(x_k) + \cdots + \pi_{mm}J_m(x_k) \end{cases}. \tag{224}$$

In this way, we transform the MJSs control problem into a multiple objectives optimal control problem. Using the weighted sum technique, we convert the above multi-objective optimal control problem into a single-objective optimisation problem. The performance index function can be rewritten as

$$J(x_k) = \omega_1 J_I(x_k) + \omega_2 J_{II}(x_k) + \cdots + \omega_m J_M(x_k) \tag{225}$$

where $\omega_i > 0$ is the weight vector and $\sum_{i=1}^{m} \omega_i = 1$.

Therefore, the control vector u_k needs to be found to minimize the performance index function (225). Note that, for optimal control problems, the designed control law must not only stabilize the systems on the compact set Ω, but also guarantee that (225) is finite, which means the control must be admissible.

Definition 1 ([11, 12]): (Admissible Controls). A law u_k is said to be an admissible control with respect to (225) on Ω, if u_k is continuous on Ω and can stabilize system (218) for all $x_0 \in \Omega$, $u_k = 0$ if $x_k = 0$, and $\forall x_k \in \Omega$, $J(x_k)$ is finite.

Equation (225) can be extended as

$$\begin{aligned} J(x_k) &= \omega_1 J_I(x_k) + \omega_2 J_{II}(x_k) + \cdots + \omega_m J_M(x_k) \\ &= \omega_1 \left(\pi_{11}J_1(x_k) + \pi_{12}J_2(x_k) + \cdots + \pi_{1m}J_m(x_k)\right) \\ &\quad + \omega_2 \left(\pi_{21}J_1(x_k) + \pi_{22}J_2(x_k) + \cdots + \pi_{2m}J_m(x_k)\right) \\ &\quad + \cdots + \omega_m \left(\pi_{m1}J_1(x_k) + \cdots + \pi_{mm}J_m(x_k)\right) \end{aligned} \tag{226}$$

Then, we can further obtain,

$$\begin{aligned} J(x_k) &= (\omega_1\pi_{11} + \omega_2\pi_{21} + \cdots + \omega_m\pi_{m1})J_1(x_k) \\ &\quad + (\omega_1\pi_{12} + \omega_2\pi_{22} + \cdots + \omega_m\pi_{m2})J_2(x_k) \\ &\quad + \cdots + (\omega_1\pi_{1m} + \cdots + \omega_m\pi_{mm})J_m(x_k) \\ &= D_1J_1(x_k) + D_2J_2(x_k) + \cdots + D_mJ_m(x_k) \\ &= \sum_{i=1}^{m} D_iJ_i(x_k) \\ &= \sum_{i=1}^{m} D_i\left(\sum_{z=k}^{\infty}\left(Q_i(x_z) + u_z^TR_iu_z\right)\right) \\ &= \sum_{i=1}^{m}\sum_{z=k}^{\infty}\left(D_i\left(Q_i(x_z) + u_z^TR_iu_z\right)\right) \end{aligned} \tag{227}$$

where $D_i = \sum_{j=1}^{m}\omega_j\pi_{ji} > 0$. Hence, equation (226) is positive definite, i.e. the obtained performance index function $J(x_k)$ is positive definite. Hence, this performance index function serves as a Lyapunov function. Equation (226) can be rewritten as

$$\begin{aligned} J(x_k) &= \sum_{i=1}^{m}(D_i(Q_i(x_k) + u_k^TR_iu_k)) + \sum_{i=1}^{m}\sum_{z=k+1}^{\infty} D_i(Q_i(x_z) + u_z^TR_iu_z) \\ &= \sum_{i=1}^{m} D_iU_i(x_k, u_k) + \sum_{i=1}^{m} D_iJ_i(x_{k+1}) \\ &= D^TT(x_k, u_k) + J(x_{k+1}) \end{aligned} \tag{228}$$

where

$D = (D_1, D_2, \cdots, D_m)^T$,

$T(x_k, u_k) = (U_1(x_k, u_k), U_2(x_k, u_k), \cdots, U_m(x_k, u_k))^T$.

Let us define a stochastic operator P by

$$PJ(x_k) = \min_{u_k}\{D^TT(x_k, u_k) + J(x_{k+1})\} \tag{229}$$

where the minimization is carried out component-wise. Adaptive dynamic programming involves solution of Bellman's equation

$$J(x_k) = PJ(x_k). \tag{230}$$

According to Bellman's optimality principle, the unique solution $J^*(x_k)$ of (230) is the optimal performance index function and satisfies the discrete-time HJB equation

$$J^*(x_k) = \min_{u_k}\{D^T T(x_k, u_k) + J^*(x_{k+1})\}. \tag{231}$$

Here, we assume that the minimum on the right-hand side of the equation (231) exists and is unique [13]. Therefore, the optimal control u_k^* satisfies the first-order necessary condition, which is given by the gradient of the right-hand side of (231) with respect to u_k as

$$\frac{\partial(D^T T(x_k, u_k))}{\partial u_k} + \left(\frac{\partial x_{k+1}}{\partial u_k}\right)^T \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} = 0 \tag{232}$$

and therefore the optimal control law is obtained by

$$\begin{aligned} u_k^* &= \arg\min\{D^T T(x_k, u_k) + J^*(x_{k+1})\} \\ &= -\frac{1}{2}(\sum_{i=1}^{m} D_i R_i)^{-1} g_i^T(x_k) \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{233}$$

where $J^*(x_k)$ is solved in the following HJB equation

$$\begin{aligned} J^*(x_k) = \sum_{i=1}^{m} D_i Q_i(x_k) + \frac{1}{4}\left(g(x_k)^T \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}}\right)^T \\ \cdot \left(\sum_{i=1}^{m} D_i R_i\right)^{-1} g^T(x_k) \frac{\partial J^*(x_{k+1})}{\partial x_{k+1}} + J^*(x_{k+1}). \end{aligned} \tag{234}$$

6.3 Optimal Control for unknown MJSs

In this section, ADP approach is proposed to approximate the optimal performance index function and control law for MJSs. Two subsections are included. The first one proposes an ADP algorithm for discrete-time nonlinear MJSs to estimate the HJB equation and solve the optimal control law according to the obtained performance index function (228). The corresponding stability analysis is given in the second subsection, including the convergence of the obtained performance index function for MJSs and the existence of the optimal control input.

6.3.1 ADP Algorithm to Approximate the Optimal Control for MJSs

In this ADP algorithm, we start with an initial performance index function $J^{(0)}(x) = 0$. Then we solve for the control law $u_k^{(0)}$ as

$$u_k^{(0)} = \arg\min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(0)}(x_{k+1}) \right\}. \tag{235}$$

According to $u_k^{(0)}$, iteration on the performance index function is performed by computing

$$\begin{aligned} J^{(1)}(x_k) &= \min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(0)}(x_{k+1}) \right\} \\ &= D^T T(x_k, u_k^{(0)}) + J^{(0)}(x_{k+1}). \end{aligned} \tag{236}$$

Because $J^{(0)}(x) = 0$, then it follows

$$J^{(1)}(x_k) = D^T T(x_k, u_k^{(0)}). \tag{237}$$

Based on (237), we can obtain the following iteration equations

$$u_k^{(1)} = \arg\min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(1)}(x_{k+1}) \right\}, \tag{238}$$

$$\begin{aligned} J^{(2)}(x_k) &= \min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(1)}(x_{k+1}) \right\} \\ &= D^T T(x_k, u_k^{(1)}) + J^{(1)}(x_{k+1}). \end{aligned} \tag{239}$$

The ADP algorithm, therefore, is obtained by iterating between a sequence of action laws $u_k^{(n)}$

$$\begin{aligned} u_k^{(n)} &= \arg\min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(n)}(x_{k+1}) \right\} \\ &= \arg\min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(n)}(f_i(x_k) + g_i(x_k)u_k) \right\} \end{aligned} \tag{240}$$

and a sequence of performance index functions $J^{(n)}(x_k)$

$$\begin{aligned} J^{(n+1)}(x_k) &= \min_{u_k} \left\{ D^T T(x_k, u_k) + J^{(n)}(x_{k+1}) \right\} \\ &= D^T T(x_k, u_k^{(n)}) + J^{(n)}(f_i(x_k) + g_i(x_k)u_k) \end{aligned} \tag{241}$$

where k is the time index, i is the index of the active subsystem at time step k, and n is the iteration index.

Note that, in the ADP algorithm, we do not need to start from an optimal performance index function which is difficult to find for general nonlinear jump systems. It is an incremental optimization process which is implemented forward in time and online. Moreover, this process is adaptive as it does not require the knowledge of system functions. In the next subsection, it is shown that $J^{(n)}(x_k)$ and $u_k^{(n)}$ converge to the optimal performance index function and to the corresponding optimal control law, respectively.

6.3.2 Convergence Analysis of the Proposed ADP Approach

In order to prove the convergence of the proposed ADP approach for discrete-time nonlinear MJSs, let us start with the following lemmas which are important in the convergence analysis.

Lemma 2: Let $\eta_k^{(n)}$ be any stabilizing and admissible control law and $\Phi^{(0)}(x) = J^{(0)}(x) = 0$, where $\Phi^{(n)}(x_k)$ is updated as

$$\Phi^{(n+1)}(x_k) = D^T T(x_k, \eta_k^{(n)}) + \Phi^{(n)}(x_{k+1}) \tag{242}$$

where,

$T(x_k, \eta_k^{(n)}) = (U_1(x_k, \eta_k^{(n)}), U_2(x_k, \eta_k^{(n)}) \cdots, U_l(x_k, \eta_k^{(n)}))^T$,

$U_i(x_k, \eta_k^{(n)}) = Q_i(x_k) + \eta_k^{(n)T} R_i \eta_k^{(n)}$, $i \in S$.

Then, $J^{(n)}(x_k) \leq \Phi^{(n)}(x_k)$.

Lemma 2 can easily be proved because $J^{(n)}(x_k)$ is the result when control u_k minimizes the right-hand side of (242).

Lemma 3: Define the performance index function sequence for discrete-time MJSs as in (241). If the MJSs (218) are controllable and $J^{(0)}(x) = 0$. Then, it follows that $J^{(n)}(x_k)$ is a monotonically non-decreasing sequence, i.e., $\forall n$, $J^{(n)}(x_k) \leq J^{(n+1)}(x_k)$.

Proof: From Lemma 2, we know if $\Phi^{(0)}(x) = J^{(0)}(x) = 0$, then the new sequence

$\Phi^{(n)}(x_k)$ defined in equation (242) has the following property

$$J^{(n)}(x_k) \leq \Phi^{(n)}(x_k). \tag{243}$$

Because $\eta_k^{(n)}$ is an arbitrary and stabilizing sequence, assume $\eta_k^{(n-1)} = u_k^{(n)}$, such that

$$\begin{aligned} \Phi^{(n)}(x_k) &= D^T T(x_k, \eta_k^{(n-1)}) + \Phi^{(n-1)}(x_{k+1}) \\ &= D^T T(x_k, u_k^{(n)}) + \Phi^{(n-1)}(x_{k+1}). \end{aligned} \tag{244}$$

In the following part, we prove $J^{(n+1)}(x_k) \geq \Phi^{(n)}(x_k)$ by mathematical induction. Let us start with $n = 0$. We know that $J^{(0)}(x_k) = \Phi^{(0)}(x_k) = 0$, then

$$J^{(1)}(x_k) - \Phi^{(0)}(x_k) = D^T T(x_k, u_k^{(0)}) \geq 0. \tag{245}$$

Thus, for $n = 0$, we obtain $J^{(1)}(x_k) \geq \Phi^{(0)}(x_k)$.

Now, we assume it holds for the $(n-1)$th iteration step, i.e.,

$$J^{(n)}(x_k) - \Phi^{(n-1)}(x_k) \geq 0. \tag{246}$$

By subtracting (244) from (241), it follows

$$\begin{aligned} & J^{(n+1)}(x_k) - \Phi^{(n)}(x_k) \\ &= D^T T(x_k, u_k^{(n)}) + J^{(n)}(x_{k+1}) - (D^T T(x_k, u_k^{(n)}) + \Phi^{(n-1)}(x_{k+1})) \\ &= J^{(n)}(x_{k+1}) - \Phi^{(n-1)}(x_{k+1}) \geq 0 \end{aligned} \tag{247}$$

which completes the proof of $J^{(n+1)}(x_k) \geq \Phi^{(n)}(x_k)$.

On the other side, we obtain $J^{(n)}(x_k) \leq \Phi^{(n)}(x_k)$ from (243), hence $J^{(n)}(x_k) \leq \Phi^{(n)}(x_k) \leq J^{(n+1)}(x_k)$ for any $n = 0, 1, 2, \cdots$, which is $J^{(n)}(x_k) \leq J^{(n+1)}(x_k)$ for any iteration step, i.e., $J^{(n)}(x_k)$ is a monotonically non-decreasing sequence. The conclusion holds. ■

From Lemma 3, we know the performance index function sequence (241) for MJSs is monotonically non-decreasing. Now, we present our main theorem.

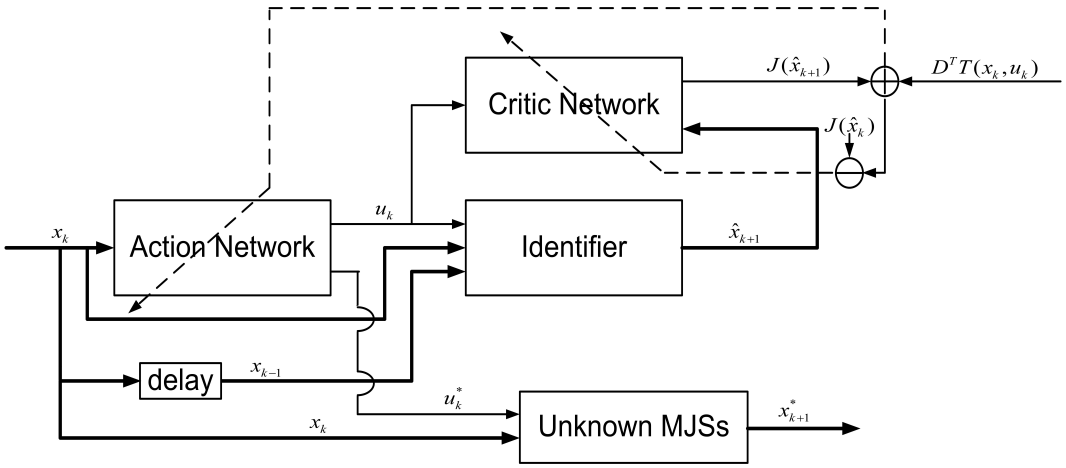

Figure 36. The neural network structure of the proposed ADP approach.

Theorem 2: Let the sequences $J^{(n)}(x_k)$ and $u_k^{(n)}$ be defined as in (241) and (240), respectively. If the MJSs (218) are controllable, then the following conditions hold.

(1) The admissible control law exists for MJSs (218).

(2) There exists an upper bound $C(x_k)$ such that $0 \leq J^{(n)}(x_k) \leq J^{\infty}(x_k) \leq C(x_k)$.

(3) The performance index function sequence can converge to the optimal value $J^*(x_k)$, and $\forall k$, u_k^{∞} is an asymptotically stable control law for MJSs (218), i.e., $u_k^{\infty} = u_k^*$.

Proof: Let us start with the admissibility part. Since $J^{(n)}(x_k)$ is positive definite, it attains a minimum at $x_k = 0$, and thus $dJ^{(n)}(x_k)/dx_k$ should vanish there. This implies that $u_k = 0$ if $x_k = 0$. The continuity assumption on $f + gu$ implies that there exists continuous control law and the system (218) cannot jump to infinity by any one step of finite control. And because $f_i(0) = g_i(0) = 0$, when the system state x_k reaches the equilibrium state, the control input becomes zero and the state of MJSs is kept at zero. According to Definition 1, the admissible control law exists for MJSs (218) which proves part (1).

The second part of the theorem follows by realizing that the elements in the obtained performance index function sequence $J^{(n)}(x_k)$ for MJSs are all positive values from equation (226). Therefore, by using Theorem 2 and equation (226), the left-hand

side of the conclusion $0 \leq J^{(n)}(x_k) \leq J^{\infty}(x_k)$ holds. Now, we prove this positive sequence has an upper bound.

Define μ_k as any stabilizing and admissible control law and let $\mu_k = \eta_k^{(n)}$. Therefore, the new sequence based on μ_k is updated as

$$\Lambda^{(n+1)}(x_k) = D^T T(x_k, \mu_k) + \Lambda^{(n)}(x_{k+1}) \tag{248}$$

where $\Lambda^{(0)}(x) = J^{(0)}(x) = 0$.

Motivated by the research in [11] and [14], we obtain the following equations

$$\begin{aligned}
&\Lambda^{(n+1)}(x_k) \\
&= D^T T(x_k, \mu_k) + \Lambda^{(n)}(x_{k+1}) \\
&= D^T T(x_k, \mu_k) + D^T T(x_{k+1}, \mu_{k+1}) + \Lambda^{(n-1)}(x_{k+2}) \\
&\vdots \\
&= D^T T(x_k, \mu_k) + D^T T(x_{k+1}, \mu_{k+1}) + \cdots + D^T T(x_{k+n}, \mu_{k+n}) + \Lambda^{(0)}(x_{k+n+1}).
\end{aligned} \tag{249}$$

Because $\Lambda^{(0)}(x) = 0$, it follows that

$$\Lambda^{(n+1)}(x_k) = \sum_{t=0}^{n} D^T T(x_{k+t}, \mu_{k+t}) = \sum_{t=k}^{n+k} D^T T(x_t, \mu_t). \tag{250}$$

Letting $n \to \infty$, $\lim_{n\to\infty} \Lambda^{(n+1)}(x_k) = \Lambda^{\infty}(x_k)$, equation (250) becomes

$$\Lambda^{\infty}(x_k) = \sum_{t=k}^{\infty} D^T T(x_t, \mu_t). \tag{251}$$

Assume $\eta_k^{(n)} = \mu_k$ and $\Phi^{(n)}(x_k) = \Lambda^{(n)}(x_k)$, such that $J^{(n)}(x_k) \leq \Lambda^{(n)}(x_k)$ obtained from Lemma 2. It can be rewritten as $J^{\infty}(x_k) \leq \Lambda^{\infty}(x_k)$ when $n \to \infty$. Combining this with (251), it follows

$$J^{\infty}(x_k) \leq \Lambda^{\infty}(x_k) = \sum_{t=k}^{\infty} D^T T(x_t, \mu_t). \tag{252}$$

Define $C(x_k) = \sum_{t=k}^{\infty} D^T T(x_t, \mu_t)$, such that (252) can be rewritten as $J^{\infty}(x_k) \leq C(x_k)$. Hence, the proof of part (2) is completed. Note that $C(x_k)$ is a function and determined by an admissible stabilizing law μ_k which means $C(x_k)$ is a finite value.

For part (3), consider the definition of the upper bound $C(x_k)$. Because μ_k is defined as an admissible control, if μ_k is the control input of the infinite step, it follows that

$$J^{\infty}(x_k) = C(x_k) \geq J^{*}(x_k). \tag{253}$$

On the other hand, since $J^{(n)}(x_k) \leq \Lambda^{(n)}(x_k)$, which can be rewritten as $J^{\infty}(x_k) \leq \Lambda^{\infty}(x_k) = \sum_{t=k}^{\infty} D^T T(x_t, \mu_t)$, we obtain

$$J^{\infty}(x_k) \leq \sum_{t=k}^{\infty} D^T T(x_t, u_t^*) \tag{254}$$

by setting $\mu_k = u_k^*$, which means $J^{\infty}(x_k) \leq J^{*}(x_k)$. From (253), we know $J^{*}(x_k) \leq J^{\infty}(x_k)$. Hence, $J^{\infty}(x_k) = J^{*}(x_k)$, i.e., $J^{(n)}(x_k)$ converge to the optimal value $J^{*}(x_k)$.

Then the convergence of the corresponding control law sequence $u_k^{(n)}$ is provided as follows.

From equation (226), we know the performance index function (228) for MJSs is positive definite. We can further write that

$$J^{\infty}(x_{k+1}) - J^{\infty}(x_k) = -D^T T(x_k, u_k^{\infty}). \tag{255}$$

Since $D^T T(x_k, u_k^{\infty})$ is positive definite, we obtain the above equation (255) is negative definite. Therefore, $J^{\infty}(x_k)$ can be seen as a kind of Lyapunov function for an admissible control u_k^{∞}. Besides, because (255) is negative definite, u_k^{∞} can make the MJSs (218) asymptotically stable. As $J^{\infty}(x_k) = J^{*}(x_k)$, it follows

$$J^{\infty}(x_{k+1}) - J^{\infty}(x_k) = J^{*}(x_{k+1}) - J^{*}(x_k). \tag{256}$$

Consider (231) and (255), equation (256) becomes

$$-D^T T(x_k, u_k^{\infty}) = -D^T T(x_k, u_k^{*}). \tag{257}$$

Hence the conclusion $u_k^{\infty} = u_k^{*}$ is proved which completes the proof. ■

From Theorem 2, we know the obtained performance index function sequence $J^{(n)}(x_k)$ for discrete-time nonlinear MJSs monotonically non-decreases to the optimal value $J^*(x_k)$ for each x_k and the corresponding admissible control input exists to asymptotically stabilize the MJSs, i.e., when $n \to \infty$, $J^{(n)}(x_k) \to J^*(x_k)$ and $u_k^{(n)} \to u_k^*$.

6.4 Design of the Proposed ADP Approach

In this section, we use the technique of neural networks to approximate the obtained performance index function sequence (241) and the control law sequence (240). The implementation process is provided in Figure 36. We can see the unknown MJS is replaced by the state identifier which is introduced in Section 5.3. Two neural networks, the critic and the action network, are used iteratively to estimate the optimal values of the performance index function and the control law. The detailed implementation process based on the actor-critic networks is presented as follows.

6.4.1 Critic Network

The purpose of the critic network is to approximate the performance index function sequence $J^{(n)}(x_k)$ of the proposed MJSs. A three-layer neural network is built as this function approximation structure. Set the weight matrix between the input and the hidden layer as W_{c1}, and the weight matrix between the hidden and the output layer as W_{c2}. Therefore, the output of the critic network can be defined as

$$\hat{J}^{(n)}(x_k) = W_{c2}^{(n)T}\Psi(y_k) \tag{258}$$

where $\Psi(\cdot)$ is the activation function defined as

$$\Psi(\cdot) = \frac{1 - e^{-(\cdot)}}{1 + e^{-(\cdot)}}. \tag{259}$$

and $y_k = W_{c1}^T[x_k^T, u_k^T]^T$.

Based on equation (241), the target performance index function is

$$J^{(n)}(x_k) = D^T T(x_k, u_k^{(n-1)}) + \hat{J}^{(n-1)}(x_{k+1}). \tag{260}$$

So, the output error of the critic network is

$$\begin{aligned} e_c^{(n)}(k) &= \hat{J}^{(n)}(x_k) - J^{(n)}(x_k) \\ &= \hat{J}^{(n)}(x_k) - \hat{J}^{(n-1)}(x_{k+1}) - D^T T(x_k, u_k^{(n-1)}). \end{aligned} \tag{261}$$

To update the weight matrix is to minimize the following performance measure

$$E_c^{(n)}(k) = \frac{1}{2} e_c^{(n)2}. \tag{262}$$

According to the gradient decent rules, the update scheme of the critic network is as follows

$$\begin{aligned} W_{c2}^{(n+1)} &= W_{c2}^{(n)} - \beta_c \left(\frac{\partial E_c^{(n)}(k)}{\partial W_{c2}^{(n)}} \right) \\ &= W_{c2}^{(n)} - \beta_c \left(\frac{\partial E_c^{(n)}(k)}{\partial e_c^{(n)}(k)} \cdot \frac{\partial e_c^{(n)}(k)}{\partial W_{c2}^{(n)}} \right) \\ &= W_{c2}^{(n)} - \beta_c \Psi(y_k) e_c^{(n)T}(k) \end{aligned} \tag{263}$$

where $\beta_c > 0$ is the learning rate of the critic network.

6.4.2 Action Network

The control law sequence $u_k^{(n)}$ is estimated by the action network. Consider a three-layer neural network architecture as this function approximation structure. Denote the weight matrix between the input and the hidden layer as a constant matrix W_{a1}, and the weight matrix between the hidden and the output layer as W_{a2}. Then, the estimated control law can be formulated as

$$\hat{u}_k^{(n)} = \Psi \left(W_{a2}^{(n)T} \Psi^{(n)}(t_k) \right) \tag{264}$$

where $t_k = W_{a1}^T x_k$, and the definition of $\Psi(t_k)$ is the same as $\Psi(y_k)$ in the critic network part

Since $u_k^{(n)}$, given by equation (240), is the target of the output of the action network, define the output error as

$$e_a^{(n)}(k) = \hat{u}_k^{(n)} - u_k^{(n)}. \tag{265}$$

The weight matrix in this process is updated to minimize the following performance measure

$$E_a^{(n)}(k) = \frac{1}{2} e_a^{(n)T}(k) e_a^{(n)}(k). \tag{266}$$

Then, we can derive gradient decent rules to train

$$W_{a2}^{(n+1)} = W_{a2}^{(n)} - \beta_a \left(\frac{\partial E_a^{(n)}(k)}{\partial W_{a2}^{(n)}} \right) \tag{267}$$

where $\beta_a > 0$ is the learning rate of the action network, and

$$\begin{aligned} \frac{\partial E_a^{(n)}(k)}{\partial W_{a2}^{(n)}} &= \frac{\partial E_a^{(n)}(k)}{\partial e_a^{(n)}(k)} \cdot \frac{\partial e_a^{(n)}(k)}{\partial \hat{u}_k^{(n)}} \cdot \frac{\partial \hat{u}_k^{(n)}}{\partial W_{a2}^{(n)}} \\ &= \Psi^{(n)}(t_k) \cdot \frac{1}{2} \left(1 - \Psi(u_a^{(n)}(k)) \right) \cdot e_a^{(n)T}(k). \end{aligned} \tag{268}$$

Note that during this training procedure, the input-to-hidden layer weight matrices W_{c1} and W_{a1} are chosen randomly at initial and only the hidden-to-output layer weight matrices W_{c2} and W_{a2} are proposed to be updated.

Remark: The training procedure above is to obtain the performance index function and the control law sequence. It is very important that the whole system would remain stable while both the action and the critic network undergo adaption, which means one should make sure the convergence of the networks' weights. So far, many papers study the neural-network-based ADP technique. Some of them prove the convergence of iterative performance index function and control law and then neural networks are just used to implement this process [11], [15], [14]. While the others prove the convergence in another way which is the convergence analysis of the neural network weights and the state [16], [17], [18]. In this chapter, we use the first method to prove that our proposed method is convergent including the performance index function and the control law. Then the actor-critic networks are used to implement this method. Detailed analysis on neural network training algorithm can be found in [16] where Liu *et al* provides a theorem to show that the training errors of the neural network weights in the ADP are uniformly ultimately bounded (UUB) by using the Lyapunov stability construct.

6.5 Simulation Studies

In this section, we provide three examples to demonstrate the effectiveness of the neural-network-based ADP approach for MJSs. Specifically, the first example solves a two-mode linear MJS and we compare the results with the theoretical solution of the HJB equation. A two-mode nonlinear MJS is considered in the second example and without loss of generality, we also consider two kinds of arbitrary selections of the system functions. In the third example, we consider a single link robot arm which is very popular in Markov jump problems. Four jumping modes are considered in this case.

6.5.1 Linear System

We start with the following discrete-time linear Markov jump system with two jumping modes

$$x_{k+1} = A_i x_k + B_i u_k \tag{269}$$

where $x_k \in R^n$ and $u_k \in R^m$.

The dynamics in each mode can be described as

$$\begin{aligned} mode\ 1 \quad & A_1 = \begin{pmatrix} -0.5 & 0.1 \\ 0.1 & 0.6 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0.1 \\ 0 \end{pmatrix} \\ mode\ 2 \quad & A_2 = \begin{pmatrix} 0.6 & -0.2 \\ 0.1 & 0.1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 \\ 0.6 \end{pmatrix}. \end{aligned} \tag{270}$$

The transition probability matrix is

$$\mathrm{H} = \begin{pmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{pmatrix}. \tag{271}$$

Assume that the system functions and dynamics are unknown. Based on the identifier design approach proposed in Section 5.3, the state identifiers for two subsystems are built with the maximal time step 250. The initial weights of the identifiers are randomly chosen in [-1,1], and the learning rate is set to $\beta = 0.01$. Then the identification

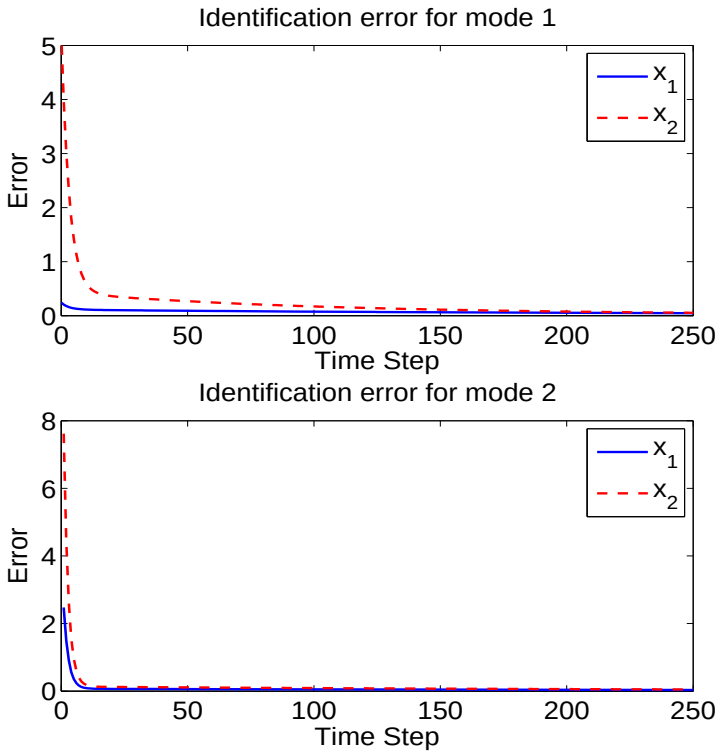

Figure 37. Identification errors for mode 1 and mode 2 of the linear MJS.

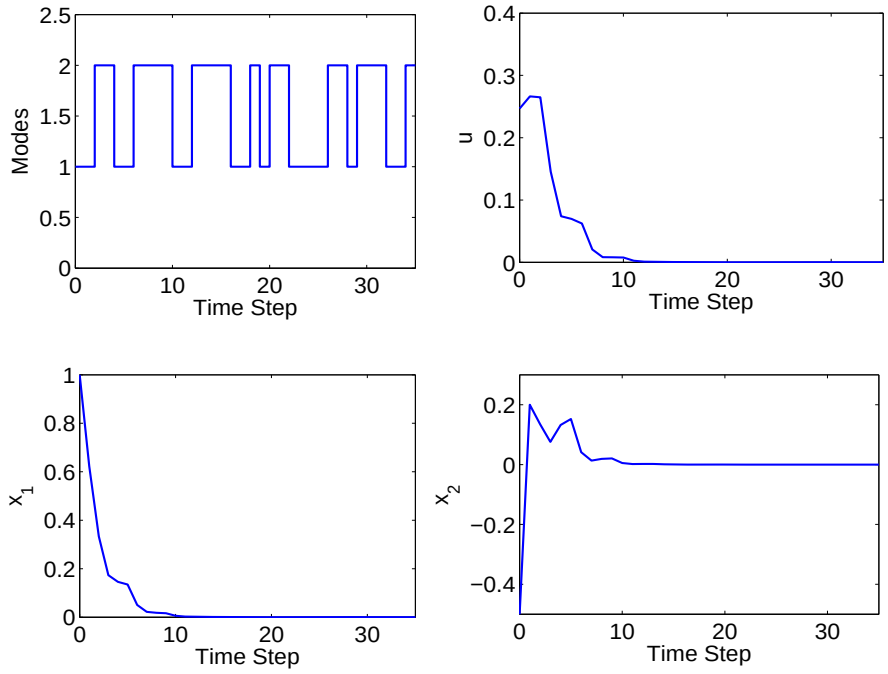

Figure 38. Active jumping mode and system responses with the ADP controller.

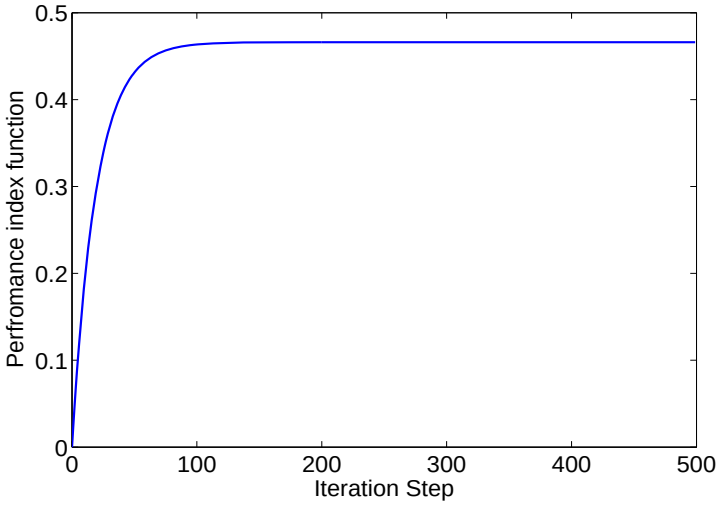

Figure 39. Performance index function trajectory of the linear MJS.

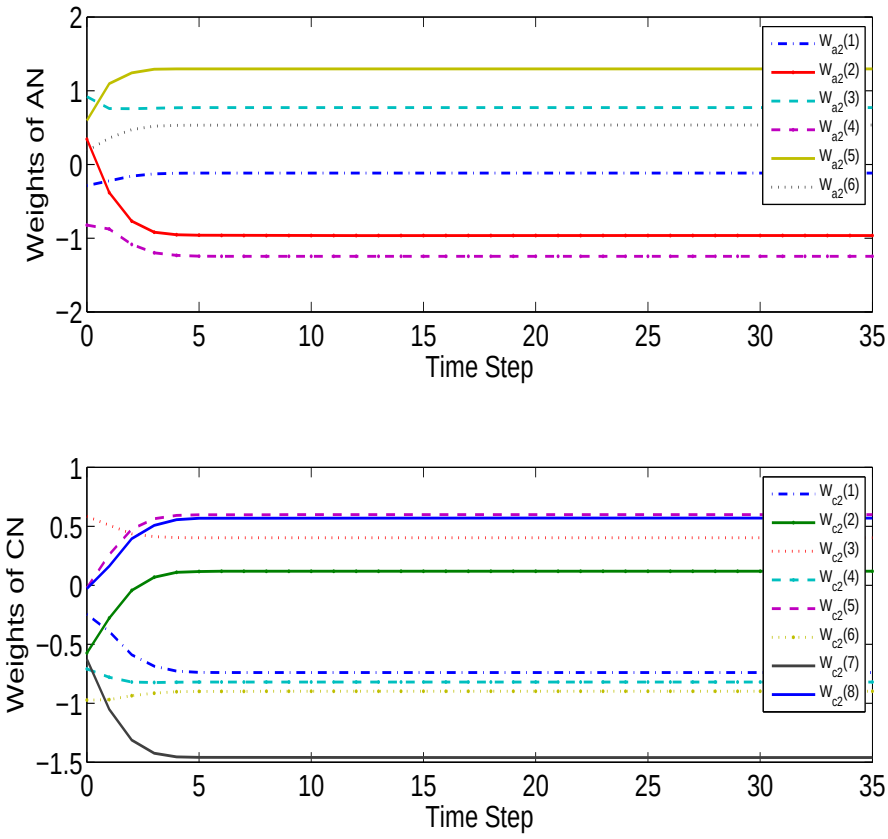

Figure 40. Weights trajectories of the action and the critic network from the hidden to the output layer.

errors for both subsystems are provided in Figure 37. We can observe that both errors converge to zero asymptotically, which means these two identifiers can approximate the states effectively.

With the built identifiers, we define the performance index function for each mode of this linear Markov jump system as linear quadratic form $J_i(x_k) = x_k^T Q_i x_k + u_k^T R_i u_k$, $i \in \{1, 2\}$, where Q_i and R_i are the identity matrices with appropriate dimensions. In this situation, set the weight vector as $\omega = [0.3, 0.7]^T$. Combining this with the knowledge of equation (271), we convert this two-mode MJS control problem into a single-objective optimal control problem according to equation (228). Therefore, the performance index function for the whole MJS can be described as

$$J(x_k) = (0.3 * 0.2 + 0.7 * 0.8)J_1(x_k) + (0.3 * 0.4 + 0.7 * 0.6)J_2(x_k). \tag{272}$$

For the design of the controller, we choose the initial state as $x_0 = [1, -0.5]^T$. Two three-layer neural networks are built as the critic and the action network and the numbers of the hidden layer nodes are set to $N_{hc} = 8$, $N_{ha} = 6$, respectively. The learning rates of both the action and the critic network are set as $\beta_c = \beta_a = 0.01$. The initial weights of both networks are set randomly within $[-1, 1]$.

The active jumping mode and the system responses of training are shown in Figure 38. We can clearly observe that the system randomly jumps between two modes and the state variables converge to zero even though the mode randomly jumps between mode 1 and mode 2. Moreover, when the system reaches the stability (after 10 time steps), the state variables do not change even though the modes still jump randomly. The trajectory of the performance index function sequence at time step $k = 0$ for MJS (270) is provided in Figure 39, indicating that the obtained performance index function sequence is monotonically non-decreasing and can stay at its optimal value during this process, just like the theoretical analysis in Section 6.3.2. Weights of both the action and the critic network from hidden to output layer are shown in Figure 40.

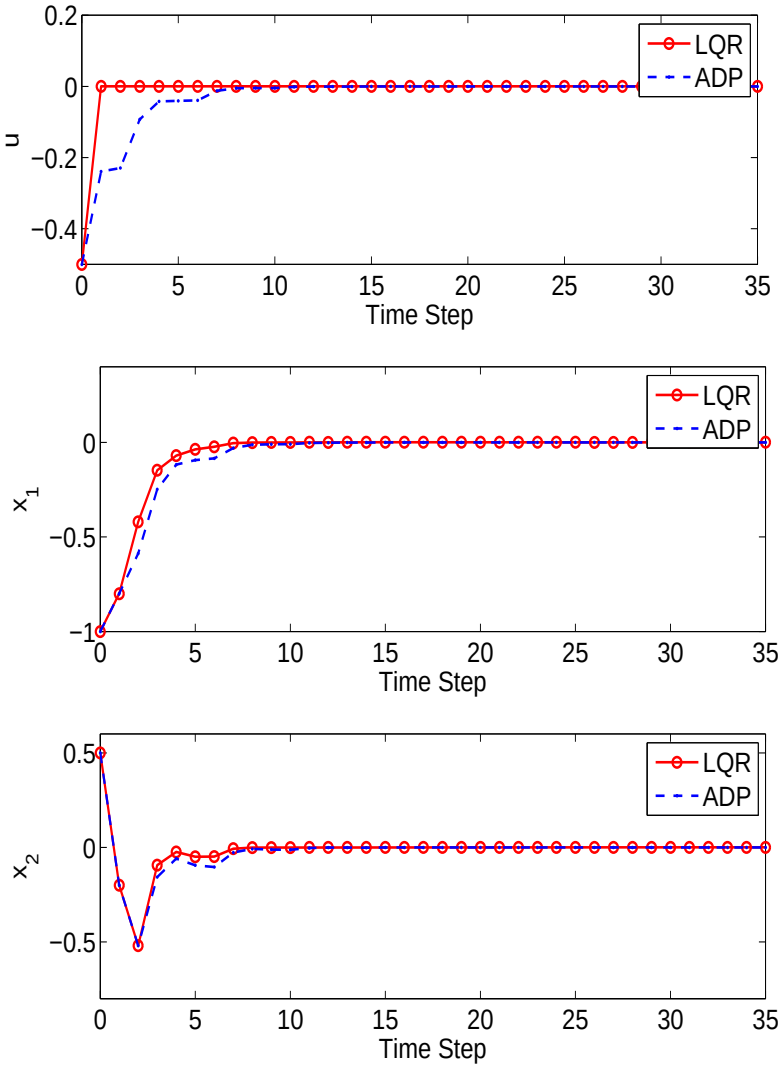

Figure 41. Comparisons of system responses of the ADP and the LQR controller.

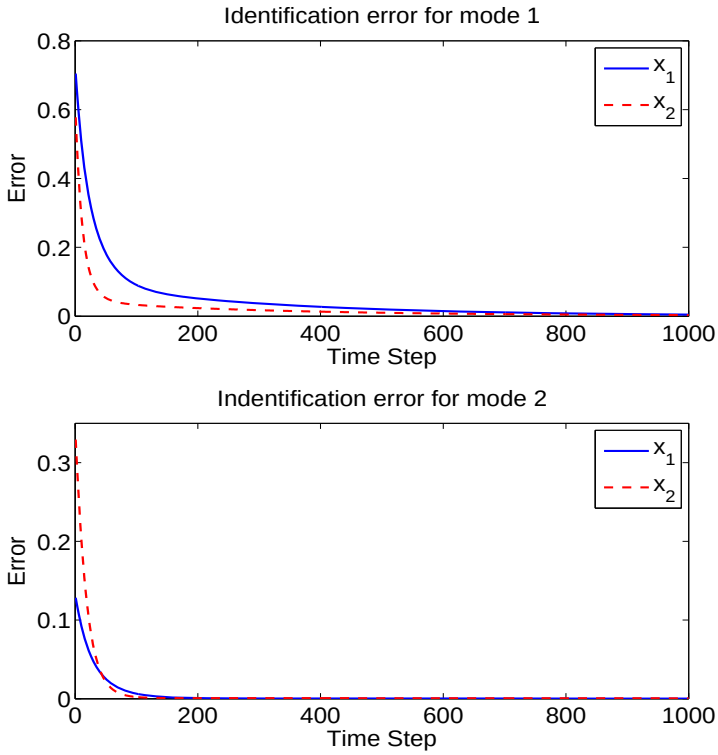

Figure 42. Identification errors for mode 1 and mode 2 of the nonlinear MJS.

Furthermore, in order to demonstrate the effectiveness of this method, we compare this ADP controller with the standard linear quadratic regulator (LQR) controller which is the exact solution of the HJB. We fix the optimal weights of the critic and the action network obtained above and test the performance of the controller. The system responses of both controllers are provided in Figure 41 including both the state variables and the control law trajectories of the two controllers. It can be seen that the system responses of the designed ADP controller can converge to those of the LQR controller, which means the training process of the proposed ADP method can obtain the performance of the optimal control solution. The simulation results reveal that the proposed neural-network-based ADP approach is effective for the linear MJS with unknown discrete-time dynamics and can obtain satisfactory.

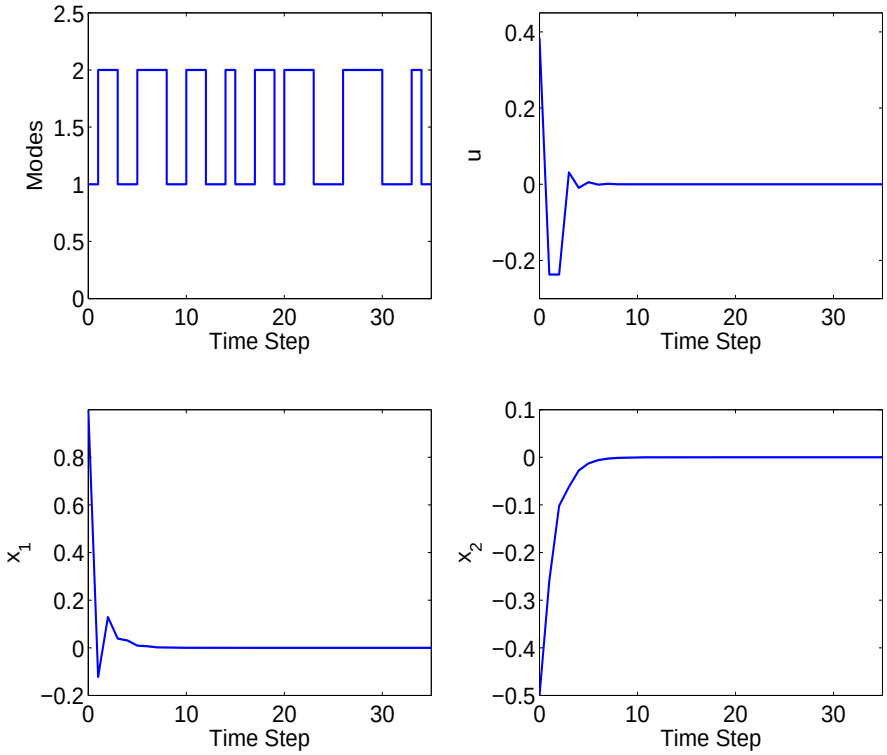

Figure 43. Active jumping mode and system responses with the ADP controller.

6.5.2 Nonlinear System

Now, we turn to the nonlinear discrete-time Markov jump system with two jumping modes. The system function can be described as follows

$$\begin{aligned} &mode\ 1 \quad \begin{cases} x_{1(k+1)} = -x_{1(k)} + x_{1(k)}\cos(x_{1(k)}x_{2(k)}) \\ x_{2(k+1)} = -\sin(x_{1(k)} + u_k) \end{cases} \\ &mode\ 2 \quad \begin{cases} x_{1(k+1)} = -\sin(0.5x_{2(k)}) \\ x_{2(k+1)} = -\sin(0.9x_{1(k)})\cos(x_{2(k)} + u_k). \end{cases} \end{aligned} \tag{273}$$

The transition probability matrix is

$$\mathrm{H} = \begin{pmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{pmatrix}. \tag{274}$$

Assume the system has unknown state dynamics. According to the identification approach presented in Section 5.3, a three-layer neural network is trained to approximate

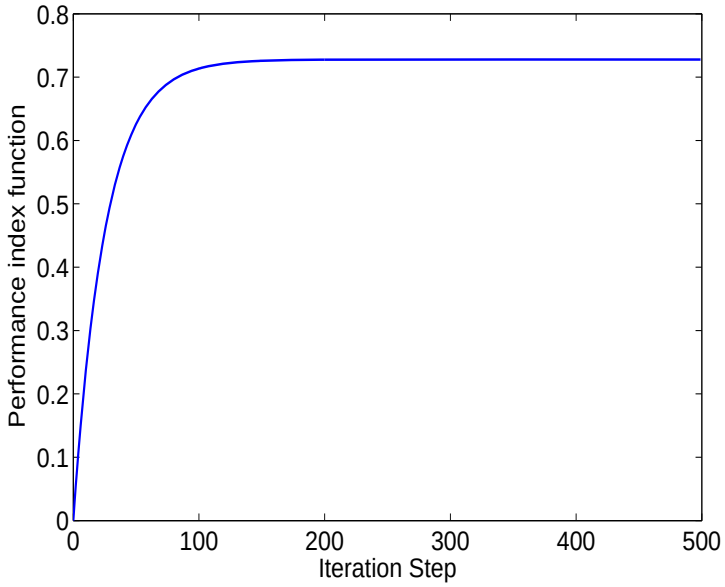

Figure 44. Performance index function trajectory of the nonlinear MJS.

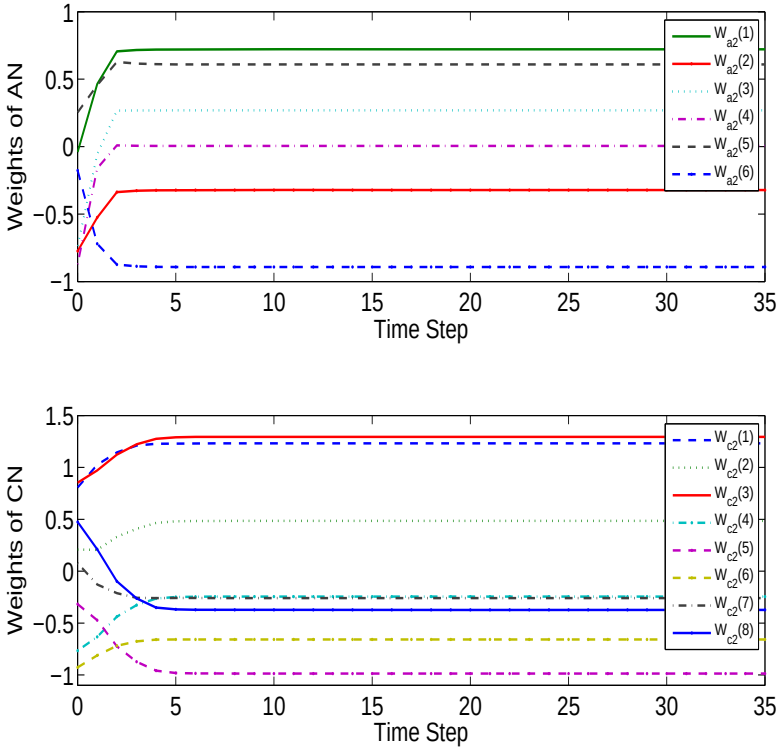

Figure 45. Weights trajectories of the action and the critic network from the hidden to the output layer.

the state of next step. The learning rate is set to $\beta = 0.01$, and the initial weights of the identifiers are chosen randomly in $[-1, 1]$. The identification errors of both subsystems are provided in Figure 42. It can be seen that the identification errors can converge to zero, indicating that the designed identifiers can accurately estimate the system state .

With these two identifiers, an ADP controller is designed to stabilize this MJS. Two three-layer neural networks are built as the critic and the action network with $N_{hc} = 8$ and $N_{ha} = 6$. The learning rates of both networks are set to $\beta_c = \beta_a = 0.01$ and the initial weights of both networks are chosen randomly within $[-1, 1]$. The initial state is set to $x_0 = [1, -0.5]$. The weight vector is set to $\omega = [0.6, 0.4]$. Therefore, we can obtain the performance index function for MJS (273) as follows

$$\begin{aligned} J(x_k) = &\big(0.6 * 0.7 + 0.4 * 0.3\big) J_1(x_k) \\ &+ \big(0.6 * 0.2 + 0.4 * 0.8\big) J_2(x_k) \end{aligned} \tag{275}$$

where $J_i(x_k) = x_k^T Q_i x_k + u_k^T R_i u_k$, $i \in \{1, 2\}$, and Q_i and R_i are the identity matrices with appropriate dimensions.

System performances of the designed ADP controller are shown in Figure 43 which illustrates the active mode of each time step, the state responses and the control input trajectory during the training process. We can observe that the system state variables reach the equilibrium point at the 10th time step and then stay at the equilibrium values even though the active mode keep changing. The performance index function sequence at time step $k = 0$ is provided in Figure 44 which consistent with the analysis in Section 6.3.2 by noticing that it is monotonically non-decreasing. The learning weights of both the critic and the action network from the hidden to the output layer are provided in Figure 45.

Additionally, without loss of generality, we consider an arbitrary selection of the system functions in the following two cases.

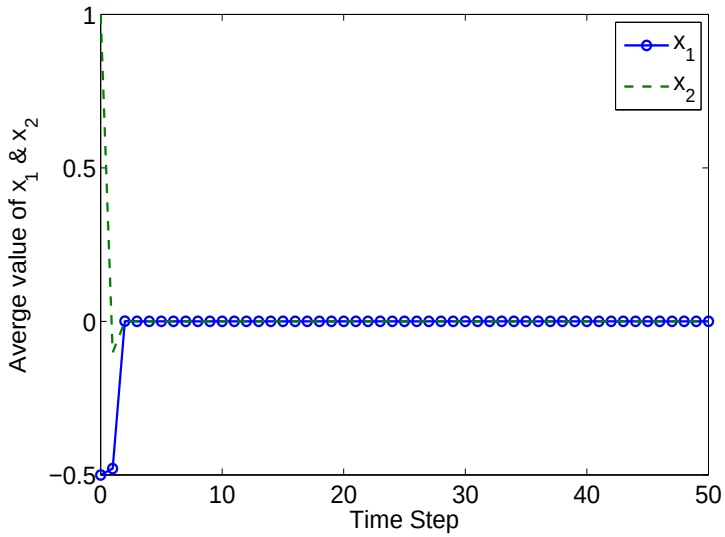

Figure 46. Average state trajectories of 10,000 round.

(1) Consider a set of nonlinear systems

$$\begin{cases} x_{1(k+1)} = \lambda_1 \sin(\lambda_2 x_{2(k)}) \\ x_{2(k+1)} = \lambda_3 \sin(\lambda_4 x_{1(k)}) \cos(\lambda_5 x_{2(k)} + \lambda_6 u_k) \end{cases} \tag{276}$$

where $\lambda_1 \sim \lambda_6$ are the designed parameters which are adjustable. For convenience, we assume $\lambda_1 \in [-1, 1]$, $\lambda_2 \in [-1, 1]$, $\lambda_3 \in [-1, 1]$, $\lambda_4 \in [-1, 1]$, $\lambda_5 \in [-100, 100]$, and $\lambda_6 \in [-100, 100]$. As we know, different sets of designed parameters ($\lambda_1 \sim \lambda_6$) come with different system functions. Therefore, we randomly choose the parameters within their boundaries, respectively, for each time step and let the system jump among these different functions for 50 time steps. Set the initial state as $x_0 = [-0.5, 1]$ and choose randomly the initial weights of both the critic and the action network within $[-1, 1]$. The performance index function is defined as $J(x_k) = x_k^T x_k + u_k^T u_k$ in this case. Then, the values of state variables are collected and the root mean square error (RMSE) is measured based on each round.

We repeat this process for 10,000 times and plot the average state trajectories of these 10,000 rounds of x_1 and x_2 which are shown in Figure 46. The histogram of the 50th state values and the state RMSE for these 10,000 rounds of x_1 and x_2 are provided

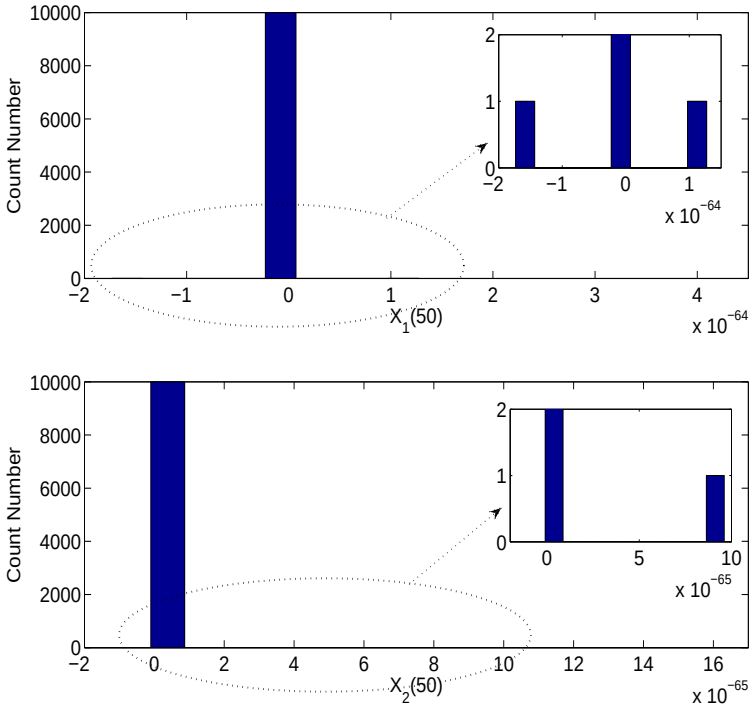

Figure 47. Histogram of the state values of 50th time step of 10,000 round.

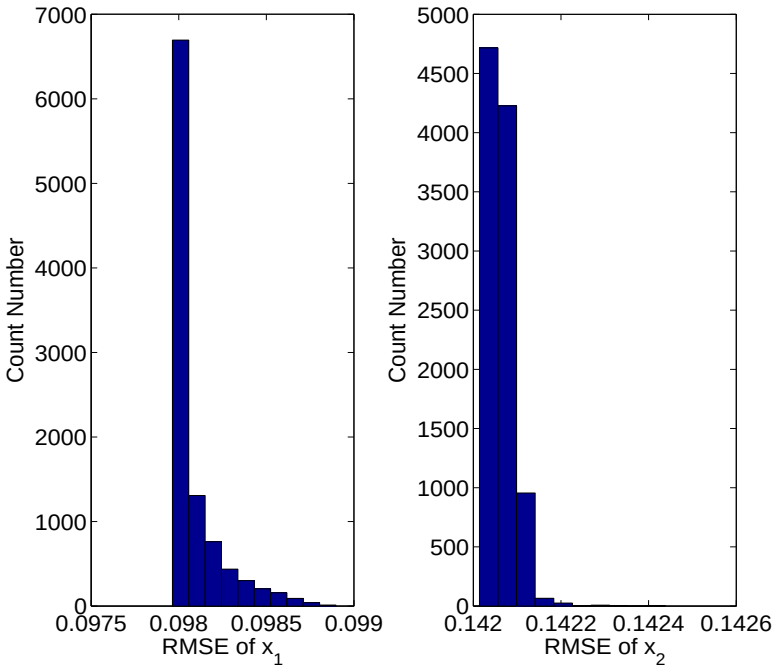

Figure 48. Histogram of RMSE for x_1 and x_2 of 10,000 round.

in Figure 47 and Figure 48. The results show that the RMSE for x_1 and x_2 can focus on a small range of errors, which means the states can converge regardless of the parameters changes within their boundaries. Moreover, from Figure 47, we know almost all the states of the 50th time step are located at a small neighbour of zero (equivalent point). Therefore, all the rounds in this situation achieve the desired performance which means the rate of success is one hundred percent.

(2) Consider the following nonlinear systems

$$\begin{aligned} mode\ 1 \quad & \begin{cases} x_{1(k+1)} = p_1 x_{1(k)} + x_{1(k)} \cos\left(p_2 x_{1(k)} x_{2(k)}\right) \\ x_{2(k+1)} = p_3 \sin\left(p_4 x_{1(k)} + u_k\right) \end{cases} \\ mode\ 2 \quad & \begin{cases} x_{1(k+1)} = p_5 \sin\left(p_6 x_{2(k)}\right) \\ x_{2(k+1)} = p_7 \sin\left(p_8 x_{1(k)}\right) \cos\left(p_9 x_{2(k)} + p_{10} u_k\right) \end{cases} \end{aligned} \tag{277}$$

where $p_1 \sim p_{10}$ are the designed parameters which are chosen within their boundaries. Moreover, we assume $p_1 \in [-1, 0]$, $p_2 \in [-100, 100]$, $p_3 \in [-1, 1]$, $p_4 \in [-100, 100]$, $p_5 \in [-1, 1]$, $p_6 \in [-1, 1]$, $p_7 \in [-1, 1]$, $p_8 \in [-1, 1]$, $p_9 \in [-100, 100]$, and $p_{10} \in [-100, 100]$. We can clearly observe that system (273) is the above MJS with specific set of the designed parameters. Without loss of generality, we randomly choose a set of these parameters ($p_1 \sim p_{10}$) within their boundaries at the beginning of each round. In other words, the system jumps between two arbitrary functions of selection in one run. The transition probability matrix is defined as

$$\mathrm{H} = \begin{pmatrix} a & 1-a \\ 1-b & b \end{pmatrix} \tag{278}$$

where $0 < a < 1$ and $0 < b < 1$ are the transition probabilities which are randomly chosen at initial. We repeat this process for 10,000 times. And, for each round, the initial state is set to $x_0 = [-0.5, 1]$. System jumps every two time steps and continue for 50 steps. The state value of each time step is collected and the RMSE for system state is measured according to each set of the parameters. Figure 49 gives the average trajectories of x_1

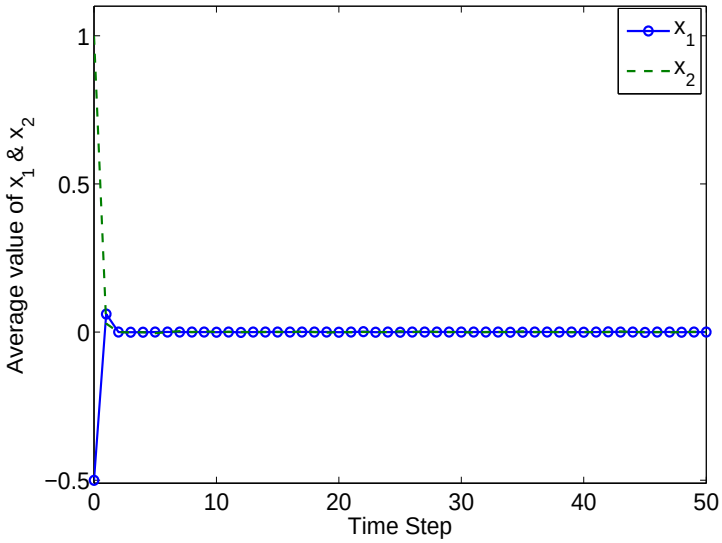

Figure 49. Average state trajectories of 10,000 round.

and x_2 of 10,000 rounds. The histograms of the state value of the 50th time step and of RMSE for x_1 and x_2 are provided in Figure 50 and Fig.51, respectively. From the results, we know most of the jumping process can converge to zero (the equivalent point). In this chapter, we define the desired performance if the state trajectories converge to the range of $[-0.01, 0.01]$. From Fig.50, we obtain the number of the unsuccessful round is 287, indicating that successful rate in this case is 97.13%.

6.5.3 Single Link Robot Arm

In this subsection, we consider a single link robot arm to illustrate the effectiveness of the proposed design approach. This model is very popular in Markov jump problems (see reference [19], [20] and [21]). Comparing to these papers, our approach does not require the knowledge of system functions. This is very important, because if the parameters of the system modes are changed, we do not need to recalculate the controller.

The dynamic function of the single link robot arm is given by

$$\ddot{\theta}(t) = -\frac{MgL}{G}\sin(\theta(t)) - \frac{D}{G}\dot{\theta}(t) + \frac{1}{G}u(t) \tag{279}$$

where $\theta(t)$ is the angle position of the robot arm, and $u(t)$ is the control input. Moreover,

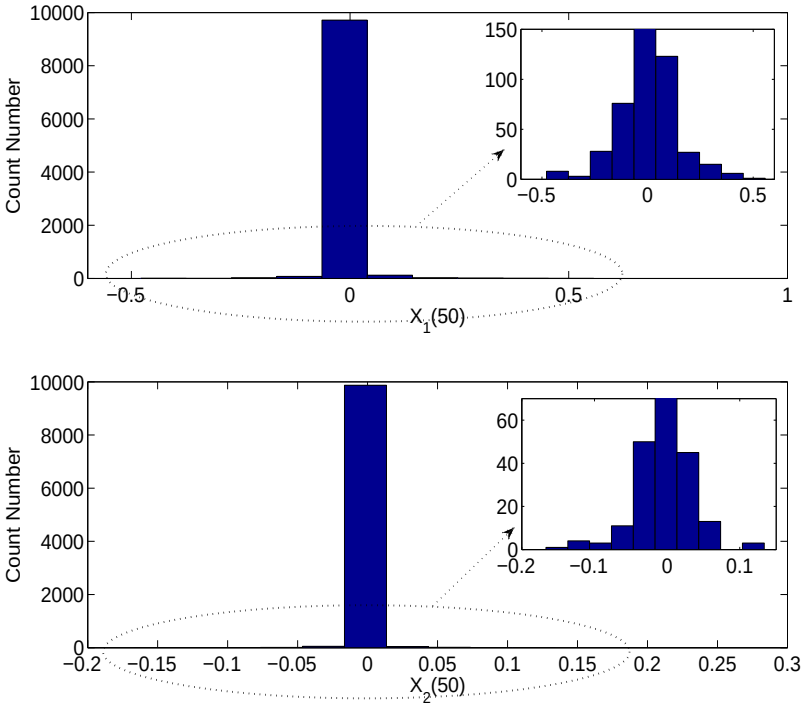

Figure 50. Histogram of the state values of 50th time step of 10,000 round.

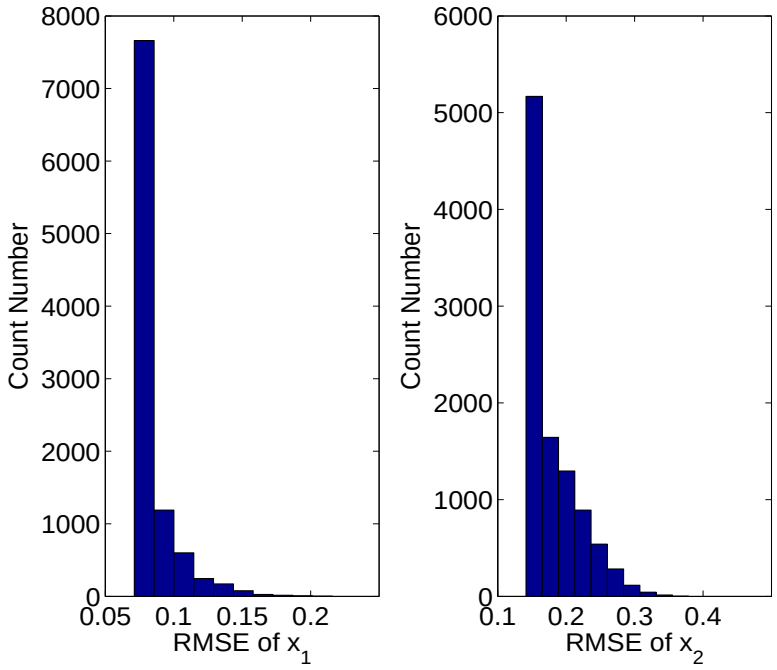

Figure 51. Histogram of RMSE for x_1 and x_2 of 10,000 round.

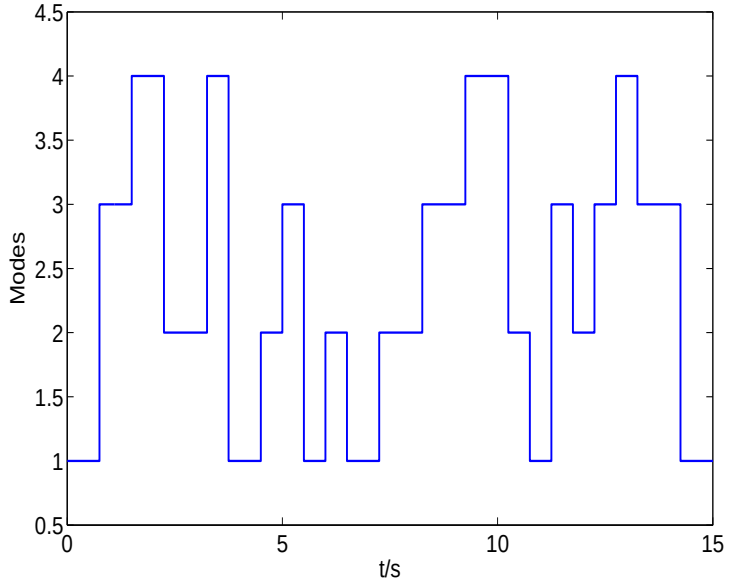

Figure 52. Jumping mode evolution r of the robot arm system.

M is the mass of the payload, G is the moment of the inertia, g is the acceleration of gravity, L is the length of the arm, and D is the viscous friction. According to [19], the values of the system parameters are given by $g = 9.81$, $D = 2$, and $L = 0.5$, respectively. This process is a Markov jump process because the parameters of M and G have four different modes. Assuming $x_1(t) = \theta(t)$ and $x_2(t) = \dot{\theta}(t)$, the dynamic function (279) can be represented by

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -\dfrac{2}{G(r)}x_2(t) + \dfrac{1}{G(r)}u(t) - \dfrac{4.905M(r)\sin(x_1(t))}{G(r)} \end{cases} \tag{280}$$

where $r = \{1, 2, 3, 4\}$, $G(r)$ and $M(r)$ are dependent on jumping mode r. In this chapter, we set $G(1) = 1$, $G(2) = 5$, $G(3) = 10$, $G(4) = 15$, and $M(1) = 1$, $M(2) = 5$, $M(3) = 10$, $M(4) = 15$. The transition probability matrix is described as

$$\mathrm{H} = \begin{pmatrix} 0.2 & 0.1 & 0.4 & 0.3 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.1 & 0.3 & 0.3 & 0.3 \\ 0.4 & 0.4 & 0.1 & 0.1 \end{pmatrix}. \tag{281}$$

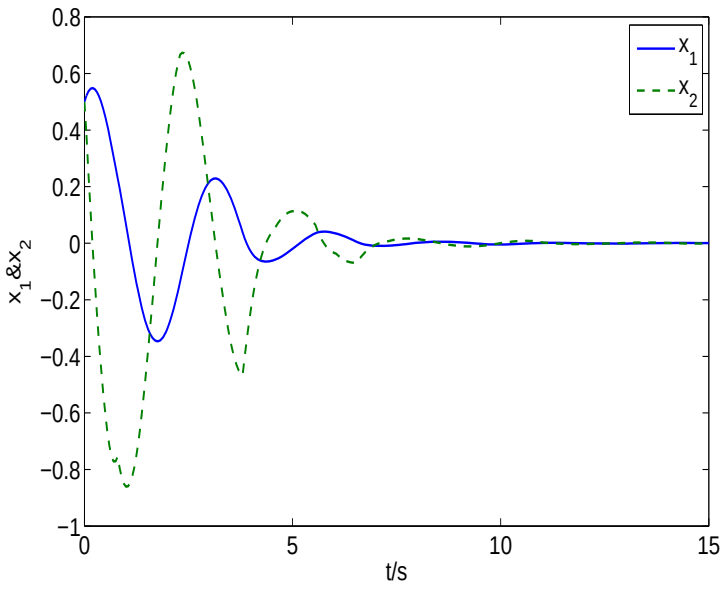

Figure 53. State trajectories of the robot arm system under jumping mode r.

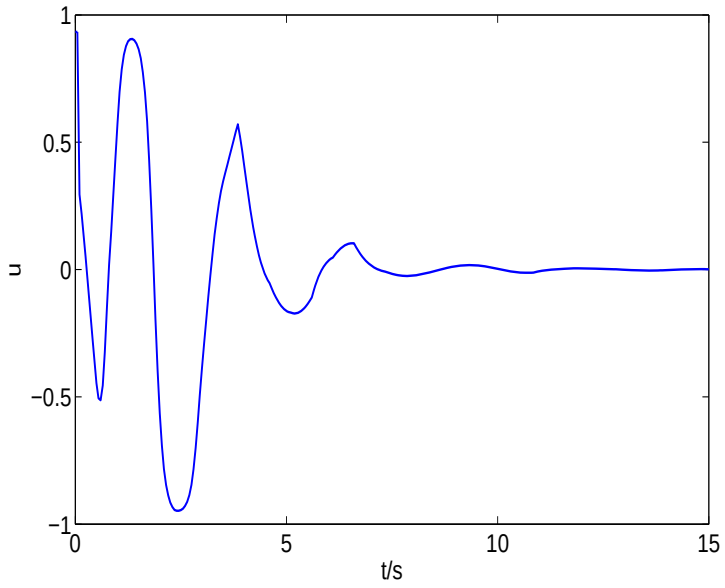

Figure 54. Control law of the robot arm system under jumping mode r.

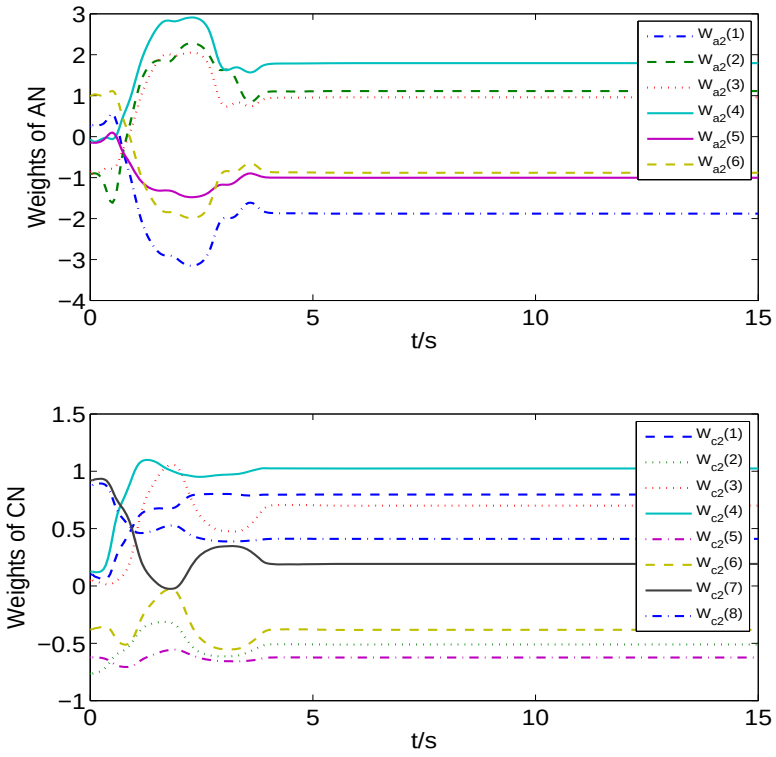

Figure 55. Weights trajectories of the action and the critic network from the hidden to the output layer.

In our current simulation, the sampling period is chosen as $T = 0.05s$. Two three-layer neural networks are built as the critic and the action network. The hidden neurons of these two networks are chosen as $N_{hc} = 8$ and $N_{ha} = 6$. The learning rates of both networks are set to $\beta_c = \beta_a = 0.01$ and the initial weights of both networks are chosen randomly within $[-1, 1]$. We set the initial state of the system to $x_0 = [0.5, 0.5]$. The system parameters are jumping randomly among four modes which can be clearly observed in Fig.52. The state trajectories and the control law of the robot arm system under jumping mode r are provided in Fig.53 and Fig.54, respectively. The weights learning process of the critic and the action network are showed in Fig.55. We know from the results that this MJSs can converge to its stable state under the designed control law. The simulation results reveal that the proposed control method can be applied to nonlinear MJSs with high jumping modes and obtain satisfying performance.

6.6 Summary

This chapter proposes an optimal control method for a class of discrete-time nonlinear MJSs with unknown dynamics. An identifier is designed to approximate the state variables for unknown systems, and an ADP-based approach is proposed to control this kind of jump systems by transforming MJSs control problem into a single objective optimal control problem. The convergence of the performance index function and the existence of the admissible control in this situation are proved in detail. Neural network techniques are applied to implement the proposed ADP method. Three simulation studies are used to demonstrate the performance of the proposed optimal control method.

List of References

[1] V. Ugrinovskii* and H. R. Pota, "Decentralized control of power systems via robust control of uncertain markov jump parameter systems," *International Journal of Control*, vol. 78, no. 9, pp. 662–677, 2005.

[2] S. C. Lee, "Maintenance strategies for manufacturing systems using markov models," Ph.D. dissertation, The University of Michigan, 2010.

[3] L. Zhang and E. K. Boukas, "Stability and stabilization of markovian jump linear systems with partly unknown transition probabilities," *Automatica*, vol. 45, no. 2, pp. 463–468, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/automatica/automatica45.html#ZhangB09

[4] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched lyapunov function approach," *IEEE Trans. Automat. Contr.*, vol. 47, no. 11, pp. 1883–1887, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac47.html#DaafouzRI02

[5] W. Chen, J. Xu, and Z. Guan, "Guaranteed cost control for uncertain markovian jump systems with mode-dependent time-delays," *IEEE Trans. Automat. Contr.*, vol. 48, no. 12, pp. 2270–2277, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac48.html#ChenXG03

[6] A. P. C. Gonalves, A. R. Fioravanti, and J. C. Geromel, "H filtering of discrete-time markov jump linear systems through linear matrix inequalities," *IEEE Trans. Automat. Contr.*, vol. 54, no. 6, pp. 1347–1351, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac54.html#GoncalvesFG09

[7] M. S. Mahmoud, P. Shi, and A. Ismail, "Robust kalman filtering for discrete-time markovian jump systems with parameter uncertainty," *J. Comput.*

Appl. Math., vol. 169, no. 1, pp. 53–69, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.cam.2003.11.002

[8] I. Matei and J. S. Baras, "Optimal state estimation for discrete-time markovian jump linear systems, in the presence of delayed output observations," *IEEE Trans. Automat. Contr.*, vol. 56, no. 9, pp. 2235–2240, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac56.html#MateiB11

[9] D. Farias and B. V. Roy, "Approximate dynamic programming via linear programming," in *Advances in Neural Information Processing Systems*, 2001, pp. 689–695.

[10] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[11] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 942–949, 2008.

[12] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/automatica/automatica41.html#Abu-KhalafL05

[13] K. G. Vamvoudakis and F. L. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2010.02.018

[14] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[15] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

[16] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, 2012.

[17] X. Zhang, H. Zhang, Q. Sun, and Y. Luo, "Adaptive dynamic programming-based optimal control of unknown nonaffine nonlinear discrete-time systems with proof of convergence," *Neurocomputing*, vol. 91, pp. 48–55, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon91.html#ZhangZSL12

[18] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 913–928, 2013.

[19] X. Luan, F. Liu, and P. Shi, "Neural-network-based finite-time h_∞ control for extended markov jump nonlinear systems," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 7, pp. 554–567, 2010.

[20] H. Wu and K. Cai, "Mode-independent robust stabilization for uncertain Markovian jump nonlinear systems via fuzzy control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 3, pp. 509–519, 2005.

[21] R. Palm and D. Driankov, "Fuzzy switched hybrid systems-modeling and identification," in *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*. IEEE, 1998, pp. 130–135.

CHAPTER 7

Conclusions and Future Research Directions

7.1 Conclusions

This dissertation research is motivated by brain-intelligence and animal intelligence, and integrates the computational intelligence into the feedback control field. By bring together the ideas from both the control systems engineering and the computational intelligence communities, this dissertation provides detailed solutions for some problems in ADP-based feedback control. This dissertation improve the performance from three aspects: algorithms, architectures, and applications. First, consider the huge computation and communication load in the ADP learning process, a novel event-triggered ADP control method is developed in chapter 3 by providing an event threshold to generate the sampled instance and guarantee the stability. The computation burden can be reduced in this way, and the competitive control performance is achieved at the same time. Next, this idea is further developed in the partially observable environment. Since the ADP design requires the full information of the system states, this is an important and challenge problem that need to be solved. An on-line learning neural-network-based observer is established in chapter 4 to recover the entire system state from the partially observable ones. Both the designed observer and the controller are only updated when an event is triggered. Therefore, the computation burden is reduced with limited state information. Then, the ADP architecture is considered. A new internal goal representation architecture is developed based on the traditional ADP structure. An additional goal network is integrated into the adaptive critic design to facilitate the learning process. Chapter 6 designs this ADP method for a class of MJSs which is a popular modeling systems in practical situations. The theoretical analysis is also provided to show the stability and convergence of the proposed methods. Simulation results are applied to demonstrate the control performance.

7.2 Original Contributions

The contributions of this dissertation are presented from two aspects: feedback control and computational intelligence. New algorithms and architectures are developed based on ADP techniques for the feedback control systems to improve the control performance. Specifically, the original contributions of this dissertation are summarized as following:

- Event-triggered ADP control method has been developed to reduce the computation and transmission load. The key idea is how to choose the sample instances to make sure the sampled data are evaluative. To this end, an event threshold has been designed to guarantee the stability of the entire feedback control systems. It is said that only when the difference between the sampled and the current states is larger than this threshold, the state is sampled and controller is updated according to the sample data.

- Event-triggered ADP control has been successfully designed in the partially observable environment. Since the ADP control method requires the full system state during the learning process, it becomes challenge when the method is developed with only the input/output data. In this dissertation, an observer has been established to recover the entire system state from the partially observable ones. Besides, in order to save the resources, both the controller and the observer are updated when an event is triggered.

- From the architecture side, goal representation design has been integrated into the traditional adaptive critic structure. The Gr-GDHP method has been proposed with explicit description. Simulation results verify the improved learning control and optimization performance. Furthermore, the results have been compared with the traditional GDHP method and other goal representation methods (i.e., GrHDP and GrDHP) to show the improvement of this method.

- Consider the significant increasing attention of MJSs in practice. The ADP method has been developed for MJSs by combining the performance index functions for multiple subsystems into one major performance index. The simulation examples have been shown that the learning-based results can converge to the optimal solutions.

7.3 Future Research Directions

The dissertation provides the comprehensive study for the on-line ADP for feedback control from three aspects: algorithms, architectures, and applications. Promising results, including both simulation results and theoretical results, are provided to demonstrate the improved learning control performance. Consider the challenges in this field, there are still many opportunities to conduct further research along this directions:

- As the event-triggered ADP method demonstrates quite competitive comparing with the traditional method with limited sampled states, it is desirable to investigate the trade-off between the control performance and the computation reduction. The optimal event threshold for each specific situation is also a interesting aspect in this field. Moreover, since the event-triggered ADP method replies on the reduced information, it will becomes challenge when the disturbance or the delay happens during the learning process. The solutions for these problems are desired to be achieved.

- Deep learning has become one of the biggest topics in machine learning field, and deep reinforcement learning has also become one of the frontier topics by taking the advantage of deep network learning principle. It is very interesting to integrate the deep learning into the ADP method. In this way, the intelligent method can be designed directly from raw data, such as images and videos. Generally, there are two ways to achieve this goal. One is to develop the convolutional neural network

for the critic and the action networks. The other way is to develop the deep neural network to derive the efficient representations of the environment from the high-dimensional sensory input. Then, the ADP method is designed directly based on the low-dimensional representations.

- In current literature, the ADP control designs are most focusing on the computer simulation. Many of the physical control systems need dedicated and high-speed embedded systems to support. It is a nature movement if this research can also be applied in several high-speed embedded system, such as FPGA and GPU boards. One step ahead this direction could make the engineering intelligence more close to reality.

- Nowadays, multi-agent control becomes a hot topic in this field, especially in robotics, UAV, tracking systems, among others. Multi-agent control is a group of autonomous agents, coordinating with each other through communication or sensing networks. Such control can perform certain challenging task which cannot be well accomplished by a single agent. In many practical situations, the complete information of multi-agent system functions is either infeasible or very difficult to obtain. However, ADP method gives us an opportunity to achieve the control performance with only the system states and control inputs. It is desired to design the distributed learning-based algorithms based on ADP or reinforcement learning not only to make all the agents reach synchronization/ achieve the goals but also minimize the energy cost under communication digraphs.

Intelligent feedback control system design is one of most exciting research topics in today's society. With the modern technologies, neuroscience, and fundamental research of computational intelligence, our human beings very hopefully achieve the truly engineering intelligent systems. This dissertation provides a comprehensive study

of an on-line learning-based method, i.e., ADP control method, for feedback control systems, including designed algorithms inspiration, new architectures based on the traditional adaptive critic designs, applications of the designed methods and theoretical assurance as well as implementation-level pseudocode algorithms. Hopefully, the dissertation could contribute to the development of this most exciting and ambitious research topics in the field.

BIBLIOGRAPHY

Abu-Khalaf, M. and Lewis, F. L., "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/automatica/automatica41.html#Abu-KhalafL05

Abu-Khalaf, M., Lewis, F. L., and Huang, J., "Neurodynamic programming and zero-sum games for constrained control systems," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1243–1252, 2008.

Adhyaru, D. M., Kar, I. N., and Gopal, M., "Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution," *IET Control Theory and Applications*, vol. 3, no. 1, pp. 1183–1195, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/nca/nca20.html#AdhyaruKG11

Adhyaru, D. M., Kar, I. N., and Gopal, M., "Bounded robust control of nonlinear systems using neural network-based HJB solution," *Neural Computing and Applications*, vol. 20, no. 1, pp. 91–103, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/nca/nca20.html#AdhyaruKG11

Al-Tamimi, A., Lewis, F. L., and Abu-Khalaf, M., "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 943–949, 2008.

Al-Tamimi, A., Lewis, F. L., and Abu-Khalaf, M., "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 942–949, 2008.

Bernstein, A. and Shimkin, N., "Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains," *Machine learning*, vol. 81, no. 3, pp. 359–397, 2010.

Chen, J. and Li, Z., "A novel adaptive tropism reward ADHDP method with robust property," in *Advances in Brain Inspired Cognitive Systems*. Springer, 2013, pp. 288–295.

Chen, W., Xu, J., and Guan, Z., "Guaranteed cost control for uncertain markovian jump systems with mode-dependent time-delays," *IEEE Trans. Automat. Contr.*, vol. 48, no. 12, pp. 2270–2277, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac48.html#ChenXG03

Chien, T.-L., Chen, C.-C., Huang, Y.-C., and Lin, W.-J., "Stability and almost disturbance decoupling analysis of nonlinear system subject to feedback linearization and feedforward neural network controller," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1220–1230, 2008.

Daafouz, J., Riedinger, P., and Iung, C., "Stability analysis and control synthesis for switched systems: a switched lyapunov function approach," *IEEE Trans. Automat. Contr.*, vol. 47, no. 11, pp. 1883–1887, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac47.html#DaafouzRI02

de Farias, D. P. and Van Roy, B., "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

Eqtami, A., Dimarogonas, D. V., and Kyriakopoulos, K. J., "Event-triggered control for discrete-time systems," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 4719–4724.

F. L. Lewis, and D. Vrabie., "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Sys. Mag.*, vol. 9, no. 3, pp. 32–50, 2009.

Farias, D. and Roy, B. V., "Approximate dynamic programming via linear programming," in *Advances in Neural Information Processing Systems*, 2001, pp. 689–695.

Fu, J., He, H., and Zhou, X., "Adaptive learning and control for mimo system based on adaptive dynamic programming," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.

Fu, J., He, H., and Ni, Z., "Adaptive Dynamic Programming with Balanced Weights Seeking Strategy," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), IEEE Symposium Series on Computational Intelligence (SSCI)*, Paris, France, 2011.

Fu, J., He, H., and Zhou, X., "Adaptive learning and control for mimo system based on adaptive dynamic programming," *IEEE Trans. Neural Networks*, vol. 22, no. 7, pp. 1133–1148, 2011.

Garcia, E. and Antsaklis, P. J., "Model-based event-triggered control with time-varying network delays," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 1650–1655.

Gonalves, A. P. C., Fioravanti, A. R., and Geromel, J. C., "H filtering of discrete-time markov jump linear systems through linear matrix inequalities," *IEEE Trans. Automat. Contr.*, vol. 54, no. 6, pp. 1347–1351, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac54.html#GoncalvesFG09

Hauskrecht, M., "Value-function approximations for partially observable markov decision processes," *arXiv preprint arXiv:1106.0234*, 2011.

He, H., *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.

He, H., Ni, Z., and Fu, J., "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon78.html#HeNF12

He, H., Ni, Z., and Zhao, D., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Learning and Optimization in Hierarchical Adaptive Critic Design, pp. 78–95.

He, P. and Jagannathan, S., "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input contraints," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 2, pp. 425–436, 2007.

Heemels, W. and Donkers, M., "Model-based periodic event-triggered control for linear systems," *Automatica*, 2013.

Heemels, W., Donkers, M., and Teel, A., "Periodic event-triggered control for linear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 847–861, 2013.

Heemels, W., Johansson, K. H., and Tabuada, P., "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3270–3285.

Igelnik, B. and Pao, Y.-H., "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *Neural Networks, IEEE Transactions on*, vol. 6, no. 6, pp. 1320–1329, 1995.

Jaakkola, T., Singh, S. P., and Jordan, M. I., "Reinforcement learning algorithm for partially observable markov decision problems," vol. 7. MIT Press, 1995, p. 345.

Kiumarsi, B., Lewis, F., Naghibi-Sistani, M.-B., and Karimpour, A., "Optimal tracking control of unknown discrete-time linear systems using input–output measured data," *Cybernetics, IEEE Transactions on*, 2015, in press.

Lee, S. C., "Maintenance strategies for manufacturing systems using markov models," Ph.D. dissertation, The University of Michigan, 2010.

Lemmon, M., "Event-triggered feedback in control, estimation, and optimization," in *Networked Control Systems*. Springer, 2010, pp. 293–358.

Lendaris, G. G., "Higher level application of adp: A next phase for the control field?" *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 901–912, 2008.

Lewis, F. and Liu, D., Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013.

Lewis, F. L., Jagannathan, S., and Yesildirek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis, London, UK, 1999.

Lewis, F. L., Liu, D., and Lendaris, G. G., "Special issue on adaptive dynamic programming and reinforcement learning in feedback control," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 896–897, 2008.

Lewis, F. L. and Liu, D., Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE, 2012.

Lewis, F. L. and Vamvoudakis, K. G., "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 1, pp. 14–25, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tsmc/tsmcb41.html#LewisV11

Li, H. and Liu, D., "Optimal control for discrete-time affine non-linear systems using general value iteration," *IET Control Theory & Applications*, vol. 6, no. 18, pp. 2725–2736, 2012.

Lin, F., "An optimal control approach to robust control design," *International Journal of control*, vol. 73, no. 3, pp. 177–186, 2000.

Lin, F., Brandt, R. D., and Sun, J., "Robust control of nonlinear systems: Compensating for uncertainty," *International Journal of Control*, vol. 56, no. 6, pp. 1453–1459, 1992.

Liu, D., Javaherian, H., Kovalenko, O., and Huang, T., "Adaptive critic learning techniques for engine torque and air-fuel ratio control," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 988–993, 2008.

Liu, D. and Wei, Q., "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.

Liu, D., Zhang, Y., and Zhang, H. G., "A self-learning call admission control scheme for CDMA cellular networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1219–1228, 2005.

Liu, D., Javaherian, H., Kovalenko, O., and Huang, T., "Adaptive critic learning techniques for engine torque and air–fuel ratio control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 988–993, 2008.

Liu, D., Li, H., and Wang, D., "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm," *Neurocomputing*, vol. 110, pp. 92–100, 2013.

Liu, D. and Wang, D., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2013, ch. Optimal Control of Unkonwn Nonlinear Discrete-Time Systems Using the Iterative Globalized Dual Heuristic Programming Algorithm, pp. 52–74.

Liu, D., Wang, D., and Li, H., "Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach," *IEEE Trans. on Neural Networks and Learning Systems*, in press.

Liu, D., Wang, D., and Yang, X., "An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs," *Information Sciences*, vol. 220, pp. 331–342, Jan. 2013.

Liu, D., Wang, D., Zhao, D., Wei, Q., and Jin, N., "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 3, pp. 628–634, 2012.

Liu, D. and Wei, Q., "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. on Cybernetics*, vol. 43, no. 2, pp. 779–789, 2013.

Liu, D. and Wei, Q., "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 43, no. 2, 2013.

Liu, D. and Wei, Q., "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. on Neural Networks and Learning Systems*, in press.

Liu, D. and Wei, Q., "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," in press, 2013.

Liu, F., Sun, J., Si, J., Guo, W., and Mei, S., "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, 2012.

Lu, C., Si, J., and Xie, X., "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Trans. Sys. Man Cyber. Part B*, vol. 38, no. 4, pp. 1008–1013, 2008.

Luan, X., Liu, F., and Shi, P., "Neural-network-based finite-time h_∞ control for extended markov jump nonlinear systems," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 7, pp. 554–567, 2010.

Luo, X., Si, J., and Zhou, Y., "An integrated design for intensified direct heuristic dynamic programming," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*. IEEE, 2013, pp. 183–190.

Mahmoud, M. S., Shi, P., and Ismail, A., "Robust kalman filtering for discrete-time markovian jump systems with parameter uncertainty," *J. Comput. Appl. Math.*, vol. 169, no. 1, pp. 53–69, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.cam.2003.11.002

Matei, I. and Baras, J. S., "Optimal state estimation for discrete-time markovian jump linear systems, in the presence of delayed output observations," *IEEE Trans. Automat. Contr.*, vol. 56, no. 9, pp. 2235–2240, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tac/tac56.html#MateiB11

Mitchell, T. M., *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1997.

Ni, Z., Fang, X., He, H., Zhao, D., and Xu, X., "Real-time tracking control on adaptive critic design with uniformly ultimately bounded condition," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL'13), IEEE Symposium Series on Computational Intelligence (SSCI)*, Apr. 2013.

Ni, Z. and He, H., "Heuristic dynamic programming with internal goal representation," *Soft Computing*, vol. 17, pp. 2101–2108, 2013.

Ni, Z. and He, H., "Heuristic dynamic programming with internal goal representation," *Soft Computing*, vol. 17, pp. 2101–2108, 2013.

Ni, Z., He, H., and Wen, J., "Adaptive learning in tracking control based on the dual critic network design," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 913–928, 2013.

Ni, Z., He, H., Wen, J., and Xu, X., "Goal representation heuristic dynamic programming on maze navigation," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 2038–2050, 2013.

Ni, Z., He, H., Wen, J., and Xu, X., "Goal representation heuristic dynamic programming on maze navigation," *Neural Networks, IEEE Transactions on*, vol. 24, pp. 2038–2050, Dec. 2013.

Ni, Z., He, H., Zhao, D., and Prokhorov, D. V., "Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–8.

Ni, Z., He, H., Zhao, D., Xu, X., and Prokhorov, D. V., "Grdhp: A general utility function representation for dual heuristic dynamic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 3, pp. 614–627, 2015.

Ni, Z., He, H., Zhao, D., Xu, X., and Prokhorov, D. V., "GrDHP: A General Utility Function Representation for Dual Heuristic Dynamic Programming," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 614–627, Mar. 2015.

Ni, Z., He, H., and Zhong, X., *Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP*. World Scientific Publishing, Singpore, ch. Frontiers of Intelligent Control and Information Processing.

Ni, Z., He, H., and Zhong, X., *Frontiers of Intelligent Control and Information Processing*. World Scientific Publishing, 2014, in press, ch. Experimental Studies on Data-Driven Heuristic Dynamic Programming for POMDP.

Ni, Z., He, H., Zhong, X., and Prokhorov, D. V., "Model-free dual heuristic dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1834–1839, 2015.

Ni, Z., Tang, Y., He, H., and Wen, J., "Multi-machine power system control based on dual heuristic dynamic programming," in *Proc. of 2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, Orlando, FL, pp. 1–7.

Palm, R. and Driankov, D., "Fuzzy switched hybrid systems-modeling and identification," in *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*. IEEE, 1998, pp. 130–135.

Pineau, J., Gordon, G., Thrun, S., *et al.*, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, vol. 3, 2003, pp. 1025–1032.

Powell, W. B., *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.

Prokhorov, D. V. and Wunsch, D. C., "Adaptive critic designs," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 8, no. 5, pp. 997–1007, 1997.

Qiao, W., Venayagamoorthy, G., and Harley, R., "DHP-based wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices," in *Proc. IEEE Int. Conf. Neural Netw.*, 2007, pp. 2093–2098.

Ray, S., Venayagamoorthy, G. K., Chaudhuri, B., and Majumder, R., "Comparison of adaptive critics and classical approaches based wide area controllers for a power system," *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.

Saad, E., "Reinforcement learning in partially observable markov decision processes using hybrid probabilistic logic programs," *arXiv preprint arXiv:1011.5951*, 2010.

Sahoo, A., Xu, H., and Jagannathan, S., "Neural network-based adaptive event-triggered control of affine nonlinear discrete time systems with unknown internal dynamics," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 6418–6423.

Sahoo, A., Xu, H., and Jagannathan, S., "Neural network-based adaptive event-triggered control of nonlinear continuous-time systems," in *Intelligent Control (ISIC), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 35–40.

Sahoo, A., Xu, H., and Jagannathan, S., "Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming," *Neural Networks and Learning System, IEEE Transactions on*, 2015, in press.

Sahoo, A., Xu, H., and Jagannathan, S., "Neural network-based event-triggered state feedback control of nonlinear continuous-time systems," *Neural Networks and Learning System, IEEE Transactions on*, 2015, in press.

Si, J., Barto, A. G., Powell, W. B., and Wunsch, D. C., Eds., *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press and John Wiley & Sons, 2004.

Si, J., Barto, A. G., Powell, W. B., and II, D. W., Eds., *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE, 2004.

Si, J., Barto, A. G., Powell, W. B., Wunsch, D. C., *et al.*, *Handbook of learning and approximate dynamic programming*. IEEE Press Los Alamitos, 2004.

Si, J. and Wang, Y.-T., "Online learning control by association and reinforcement," *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 264–276, 2001.

Smith, T. and Simmons, R., "Heuristic search value iteration for pomdps," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 520–527.

Sutton, R. S. and Barto, A. G., *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

Talebi, H. A., Abdollahi, F., Patel, R. V., and Khorasani, K., *Neural Network-Based State Estimation of Nonlinear Systems*. Springer, New York, 2010. [Online]. Available: http://books.google.com/books?id=TzdCJdjrp3YC

Tallapragada, P. and Chopra, N., "On event triggered tracking for nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2343–2348, 2013.

Tang, Y., He, H., Ni, Z., Wen, J., and Sui, X., "Reactive power control of grid-connected wind farm based on adaptive dynamic programming," *Neurocomputing*, vol. 125, pp. 125–133, Feb. 2014.

Tolic, D., Fierro, R., and Ferrari, S., "Optimal self-triggering for nonlinear systems via approximate dynamic programming," in *Control Applications (CCA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 879–884.

Ugrinovskii*, V. and Pota, H. R., "Decentralized control of power systems via robust control of uncertain markov jump parameter systems," *International Journal of Control*, vol. 78, no. 9, pp. 662–677, 2005.

Vamvoudakis, K. G., "Event-triggered optimal adaptive control algorithm for continuous-time nonlinear systems," *IEEE/CAA JOURNAL OF AUTOMATICA SINICA*, vol. 1, no. 3, pp. 282–293, 2014.

Vamvoudakis, K. G. and Lewis, F. L., "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

Vamvoudakis, K. G. and Lewis, F. L., "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2010.02.018

Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., "Dual heuristic programming excitation neurocontrol for generators in a multimachine power system," *Industry Applications, IEEE Transactions on*, vol. 39, no. 2, pp. 382–394, 2003.

Vrabie, D., Pastravanu, O., Abu-Khalaf, M., and Lewis, F. L., "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

Wang, D., Liu, D., and Wei, Q., "Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14–22, 2012.

Wang, D., Liu, D., Wei, Q., Zhao, D., and Jin, N., "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

Wang, D., Liu, D., Zhao, D., Huang, Y., and Zhang, D., "A neural-network-based iterative GDHP approach for solving a class of nonlinear optimal control problems with control constraints," *Neural Computing and Applications*, vol. 22, no. 2, pp. 219–227, 2013.

Wang, F. Y., Zhang, H., and Liu, D., "Adaptive dynamic programming: An introduction," *IEEE Comput. Intel. Mag.*, vol. 4, no. 2, pp. 39–47, 2009.

Wei, Q. and Liu, D., "Adaptive dynamic programming with stable value iteration algorithm for discrete-time nonlinear systems," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2012, pp. 1–6.

Wei, Q. and Liu, D., "Numerical adaptive learning control scheme for discrete-time nonlinear systems," *IET Control Theory & Applications*, vol. 7, no. 11, pp. 1472–1486, 2013.

Werbos, P. J., "Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design," *IEEE Int. Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL07)*, pp. 209–216, 2007.

Werbos, P. J., "ADP: The key direction for future research in intelligent control and understanding brain intelligence," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 38, no. 4, pp. 898–900, 2008.

Werbos, P. J., "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200–212, 2009.

Werbos, P. J., "Applications of advances in nonlinear sensitivity analysis," in *System modeling and optimization*. Springer, 1982, pp. 762–770.

Wu, H. and Cai, K., "Mode-independent robust stabilization for uncertain Markovian jump nonlinear systems via fuzzy control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 3, pp. 509–519, 2005.

Yang, L., Si, J., Tsakalis, K. S., and Rodriguez, A. A., "Direct heuristic dynamic programming for nonlinear tracking conrol with filtered tracking error," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 39, no. 6, pp. 1617–1622, 2009.

Zhang, H., Liu, D., Luo, Y., and Wang, D., *Adaptive Dynamic Programming for Control: Algorithms and Stability (Communications and Control Engineering)*. Springer, 2013.

Zhang, H. G., Luo, Y. H., and Liu, D., "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

Zhang, H. G., Wei, Q. L., and Luo, Y. H., "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 937–942, 2008.

Zhang, H., "Partially observable markov decision processes: A geometric technique and analysis," *Operations Research*, vol. 58, no. 1, pp. 214–228, 2010.

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

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

Zhang, J. and Feng, G., "Event-driven observer-based output feedback control for linear systems," *Automatica*, vol. 50, no. 7, pp. 1852–1859, 2014.

Zhang, L. and Boukas, E. K., "Stability and stabilization of markovian jump linear systems with partly unknown transition probabilities," *Automatica*, vol. 45, no. 2, pp. 463–468, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/automatica/automatica45.html#ZhangB09

Zhang, X., Zhang, H., Sun, Q., and Luo, Y., "Adaptive dynamic programming-based optimal control of unknown nonaffine nonlinear discrete-time systems with proof of convergence," *Neurocomputing*, vol. 91, pp. 48–55, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijon/ijon91.html#ZhangZSL12

Zhong, X., He, H., and Prokhorov, D. V., "Robust controller design of continuous-time nonlinear system using neural network," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013.

Zhong, X., He, H., Zhang, H., and Wang, Z., "Optimal control for unknown discrete-time nonlinear markov jump systems using adaptive dynamic programming," *Neural Networks and Learning Systems, IEEE Transactons on*, vol. 25, no. 12, pp. 2141–2155, 2015.

Zhong, X., Ni, Z., Tang, Y., and He, H., "Data-driven partially observable dynamic processes using adaptive dynamic programming," in *Proc. IEEE Symposium of Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2014, pp. 1–8.

Zhong, X., He, H., Zhang, H., and Wang, Z., "A neural network based online learning and control approach for markov jump systems," *Neurocomputing*, vol. 149, pp. 116–123, 2015.

Zhong, X., Ni, Z., and He, H., "A theoretical foundation of goal representation heuristic dynamic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, 2015, in press.

Zhong, X., Ni, Z., and He, H., "Convergence analysis of grdhp-based optimal control for discrete-time nonlinear system," in *Neural Networks (IJCNN), The 206 International Joint Conference on*. IEEE, 2016, pp. 1–8.

Zhong, X., Ni, Z., and He, H., "A theoretical foundation of goal representation heuristic dynamic programming," ieee trans. on neural networks and learning systems," in press, 2015.