UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and the Environment
Electro-Mechanical Research Group

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

Supervisor: Dr Zhan Shu
Examiner: Dr Xinggang Yan, Dr Andrew J Chipperfield

Optimal Control of Networked Systems Using Reinforcement Learning

by Xiaoru Sun

August 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT
Electro-Mechanical Research Group

Doctor of Philosophy

by Xiaoru Sun

The trend of using wireless communication channel in network control system increases a lot, because of its flexibility and mobility. Improving system performance with simple devices, such as low storage capacity sensors and low transmission power channel, is very important to ensure long life time. Hence, there is interest in system communication and controller design to optimize the information used by devices, so as to maintain overall system performance. This thesis explores an approach to co-design of communication and control.

First of all, the design of encoder and controller pair for feedback control systems over binary symmetric channels is concerned. An iterative design method based on Q-learning is proposed to obtain a pair of encoder and controller that can optimize a finite-horizon linear quadratic cost function. Three encoder strategies, memoryless encoder, memory encoder and predictive encoder, are considered. The proposed design can be implemented online, and has the potential to provide better performance. Compared with traditional control optimization method, the proposed design method is model-free, only data measured along with the system trajectories is utilized. Simulations are provided to show the effectiveness and the merits of the proposed method.

Only finite channel inputs and finite outputs is considered in previous work, while there are some infinite channel output models in practical. Hence, we studies how the generalization to infinite-output channels affected the optimization of the encoder-controller, theoretically and practically, by studying one special type of infinite output channels, namely, Gaussian channel. Since the infinite-channel outputs mainly affect the controller design, we devote to controller design, which are soft controller design, hard controller design and the combination.

From above considerations, all the research works are based on iterative design method, which means the encoder is optimized with fixed controller and the controller is optimized with fixed encoder. However, only local optimal solutions can be got by iterative design. Therefore, distributed encoder and controller design is proposed. Both encoder and controller learn independently with their own local information, and both of them can be optimized simultaneously. Obviously, the system performance is better than iterative design. In addition, distributed Q-learning can be applied into complex networked control systems.

Contents

Acknowledgements xv

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Contributions . . . 2
- 1.3 Organization . . . 4

2 Literature Review 7
- 2.1 Networked Control Systems . . . 7
  - 2.1.1 Introduction . . . 7
  - 2.1.2 Research Issues of Networked Control Systems . . . 8
- 2.2 Elements of Channel Coding . . . 9
  - 2.2.1 Source Coding . . . 9
  - 2.2.2 Communication Channels . . . 10
  - 2.2.3 Hadamard-Based Soft Decoding . . . 11
- 2.3 Reinforcement Learning Review . . . 12
  - 2.3.1 Basic Knowledge of Reinforcement Learning . . . 13
  - 2.3.2 Markov Decision Processes . . . 14
- 2.4 Three Optimal Control Design Methods . . . 14
  - 2.4.1 Dynamic Programming . . . 15
  - 2.4.2 Policy Iteration . . . 16
  - 2.4.3 Value Iteration . . . 17
- 2.5 Q-Learning Method . . . 18
  - 2.5.1 Q-Function . . . 18
  - 2.5.2 Q-Learning Algorithm . . . 19
- 2.6 Discrete-Time LQR Optimization By Q-Learning . . . 21
  - 2.6.1 Model-Based . . . 22
  - 2.6.2 Model-Free . . . 23
- 2.7 Co-Design Encoder and Controller Problem . . . 24

3 Co-Design of Encoder and Controller for Feedback Control Systems Over Binary Symmetric Channels Using Q-Learning 27
- 3.1 Introduction . . . 27
- 3.2 Preliminaries . . . 28
  - 3.2.1 System Model . . . 28
  - 3.2.2 Problem Statement . . . 30
- 3.3 Encoder Design by Q-Learning . . . 30

3.3.1 Q-Value Updating Rule . . . 30
3.3.2 Memoryless Encoder Design . . . 33
3.3.3 Memory Encoder Design . . . 34
3.3.4 Predictive Encoder Design . . . 34
3.3.5 Theoretical Aanlysis . . . 35
3.4 Controller Design by Q-Learning . . . 37
3.4.1 Q-Learning for Optimal Control . . . 38
3.4.2 Theoretical Analysis . . . 39
3.4.3 Q-Learning Design Process . . . 49
3.5 Iterative Encoder-Controller Design . . . 51
3.6 Convergence Analysis . . . 52
3.7 Numerical Results . . . 53
3.7.1 Encoder Design . . . 53
3.7.2 Controller Design . . . 55
3.7.3 Iterative Design . . . 56
3.8 Conclusions . . . 59

4 Co-Design of Encoder and Controller for Feedback Control Systems Over Gaussian Channels Using Q-Learning 61
4.1 Introduction . . . 61
4.2 Preliminaries . . . 62
4.2.1 System Model . . . 62
4.2.2 Problem Statement . . . 64
4.3 Controller Design . . . 64
4.3.1 Controller Design by Q-Learning . . . 65
4.3.2 Soft-Information Based Controller Design . . . 66
4.3.3 Hard-Information Based Controller Design . . . 71
4.3.4 Combined Soft-Hard Controller Design . . . 72
4.4 Iterative Encoder-Controller Design . . . 73
4.5 Numerical Results . . . 74
4.5.1 Controller Design . . . 75
4.5.2 Iterative Design . . . 77
4.6 Conclusions . . . 78

5 Distributed Encoder and Controller Design for Feedback Control Systems Over BSC Using Q-Learning 81
5.1 Introduction . . . 81
5.2 Distributed Q-Learning . . . 84
5.2.1 Assumption . . . 84
5.2.1.1 Measurability and Moments . . . 84
5.2.1.2 Link Failures . . . 85
5.2.1.3 Independence . . . 85
5.2.1.4 Finite Stopping Time . . . 86
5.2.2 Updating Equation of QD-Learning . . . 86
5.2.2.1 The Algorithm of QD-Learning . . . 87
5.2.2.2 Restriction of Convergence . . . 88
5.3 Examples about QD-Learning . . . 89

5.3.1 Parameters Setting 89
5.3.2 Simulations 91
5.4 Distributed Encoder-Controller Design 92
5.5 Numerical Results 95
5.6 Conclusions 98

6 Conclusions and Future Research **101**
6.1 Conclusions 101
6.2 Future Research 102

Bibliography **105**

List of Figures

2.1 A common diagram of a digital communication channel system. 9
2.2 Binary symmetric channel (BSC) model 10
2.3 Quantization over Gaussian channel. 11
2.4 The agent and environment interaction in reinforcement learning. 14

3.1 A general feedback control system over communication channel. 28
3.2 Binary symmetric channel model 29
3.3 Basic framework of Q-learning. 31
3.4 The flow chart of memoryless encoder design by Q-learning (model-free and online learning method) 33
3.5 Closed-loop control system with open-loop encoder and side information from controller to encoder. 45
3.6 The controller design process by Q-learning method 51
3.7 The flow chart of iterative encoder and controller design by Q-learning 52
3.8 The optimized encoder regions with the crossover probability $p = 0.02$ over time interval from $t = 1$ to $t = 6$. 54
3.9 The optimized encoder regions of open-loop encoder system with the crossover probability $p = 0.02$ at $t = 1$. 55
3.10 Randomly choose one state action pair (y_t, i_t) from 16 encoder regions and plot its Q-value evolutions at $t = 1$ with the crossover probability $p = 0.02$(memoryless encoder). 56
3.11 The evolution of control gain with crossover probability $p = 0.02$ 57
3.12 The evolution of weight sequence W with crossover probability $p = 0.02$ . . . 57
3.13 Performance Comparison 58
3.14 The closed loop trajectories of system state $x(t)$ obtained by Q-learning method with memoryless encoder strategy. 58

4.1 A general feedback control system over Gaussian channels (GC). 63
4.2 Soft information based controller 66
4.3 Separation of the soft controller design decoding procedure 70
4.4 Hard controller $u_t = k_t \mathbb{E}\{x_t | j_t\}$ 71
4.5 Combined soft and hard controller $u_t = k_t \mathbb{E}\{x_t | h_t, j_0^{t-1}\}$ 72
4.6 The flow chart of iterative memoryless/memory encoder and controller design by Q-learning (model-free and online learning method) 73
4.7 The evolution of control gain for soft controller design at different system times with $\sigma_w^2 = 1$ 76
4.8 The evolution of control gain for hard controller design at different system times with $\sigma_w^2 = 1$ 76

4.9 The evolution of control gain for combined soft and hard controller design at different system times with $\sigma_w^2 = 1$ . . . 77
4.10 Performance Comparison with $E_t = 1$ . . . 78
4.11 Performance Comparison with $E_t = 4$ . . . 78

5.1 The multi-agent learning framework . . . 82
5.2 State-action framework . . . 89
5.3 The set of agent network . . . 90
5.4 Centralised Q-learning (dotted lines) and distributed Q-learning (solid lines) . . . 91
5.5 Consensus among distributed Q-factors . . . 92
5.6 Changing the learning-rate $\alpha_{i,u}(t)$ . . . 93
5.7 Changing the learning-rate $\alpha_{i,u}(t)$ . . . 93
5.8 Changing the consensus-rate $\beta_{i,u}(t)$ . . . 94
5.9 The variance of one agent over episode . . . 94
5.10 The flowchart of distributed encoder and controller design . . . 96
5.11 The evolution of encoder region with $i_t = 4$ over time interval from $t = 1$ to $t = 10$ and its corresponding binary code is $[0 \quad 0 \quad 1 \quad 1]$. . . . 97
5.12 Randomly choose one state action pair (y_t, i_t) from 16 encoder regions and plot its Q-value evolutions at $t = 0$. . . . 97
5.13 Performance Comparison . . . 98

List of Algorithms

1 Policy iteration 17
2 Value iteration 18
3 General Q-learning algorithm 20
4 Discrete-Time LQR Optimization By Q-Learning 24
5 QD-learning 88

List of Tables

3.1 Q-Learning table at time $t = 0$ of states by actions that is initialized to zero. . . 31
3.2 Q-matrix updating after the initialization in TABLE 3.1. 32

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr Zhan Shu for his patience, motivation, enthusiasm, and immense knowledge. His guidances helped me in every aspect of my research study. I appreciate very much for introducing me to this joint field of control and communication, and for freedom to explore different research ideas. Your passion for science and research work has always been inspiring to me.

Further thanks to my colleagues : Shangcheng Chen, Shenglong Zhou, Xingda Yan, Mu Li, Thabiso M. Maupong, Ahmad Takiyuddin, Boriboon Deeka. My student life would be more difficult without the friendships.

My heartiest gratitude goes to my family for their continuous love, emotional and moral support.

Chapter 1

Introduction

1.1 Background

With the rapid development of communication technology, an increasing number of control systems use communication channels or networks to transmit data. Application areas include industrial automation, aerospace and medical systems, as well as consumer electronics such as home electronics and mobile phones. Although using communication networks/channels has many advantages such as lower cost, less system wiring, and more flexibility, it has also raised many issues and challenges to design, e.g. , time delay, packet loss, data rate limitation, and incomplete information pattern [Antsaklis and Baillieul, 2007, Ploplys et al., 2004, Delchamps, 1990, Zhang et al., 2014, Fagnani and Zampieri, 2003, Shu and Middleton, 2011]. How to find efficient ways of processing available information at each distributed node, as well as exchanging useful information among the nodes play a very important role.

There are two areas involved for networked embedded control systems, which are information theory and control theory. For traditional communication theory, most efforts make on optimal protocol for transmitting information, while traditional control theory provides methodologies for designing controllers to interact with the environment Cover and Thomas [2012], El Gamal and Kim [2011]. Most the research work in these two disciplines have largely been carried out separately. A traditional control system is based on an underlying assumption of perfect communication links between the plant and the controller Lee and Markus [1967], Doyle et al. [2013]. While the controller is assumed to have perfect access to the sensor observations, and the decision of the controller is available directly at the input of the actuator. Under these ideal assumptions, there is no limitation on how much data it is possible to transmit at each time instant, and there are neither delays nor transmission errors in the links between the plant and the controller [Aoki, 1967, Bertsekas et al., 1995]. Advanced mathematical tools are developed to govern the interplay among the plant, sensor, and controller under these ideal assumptions. Until recently, the research work about control system by using wireless communication channels is considered [Ploplys et al., 2004, Mazo and Tabuada, 2011, Pajic et al., 2011]. In such

networks, the sensor observations are typically quantized and transmitted over noisy links. Challenges, such as data delays and data drops, are encountered. Concerning control over non-ideal communication links, relatively little work has been performed so far. To develop methods and tools for the analysis and synthesis of feedback control over imperfect communication links is therefore of great importance.

The constraints imposed by the imperfect communication links are complex [Niu et al., 2009, Fu and Xie, 2009, Nair and Evans, 2000]. As discussed above, quantization and transmission errors are examples of crucial obstacles . The quantization deteriorates the signals transmitted between the plant and the controller. This can potentially degrade the overall system performance substantially. Although quantization in feedback control systems was studied since the dawn of control engineering, the results have mainly been restricted to treating quantization errors as additive white noise. Moreover, in almost all applications, simple quantizers, such as uniform quantizers, are employed because of practical reasons. However, for applications with extremely low data rate requirements and high communication costs, it is natural to study closer-to-optimal solutions.

Transmission errors are unavoidable in communications over unreliable media, for example in wireless networks. Therefore, robustness to transmission errors is one of the fundamental requirements of all modern communication systems [Li et al., 2009, Braslavsky et al., 2007, Tatikonda et al., 2004]. Concerning control applications, relatively little has been done to take into account imperfect communications in the overall system design. However, due to the delay sensitivity, it is not suitable to use long block codes to reduce the uncertainties, as commonly done in traditional communication systems. When facing the constraint on the codeword length, a joint design which combines the source compression and the channel protection is expected to achieve satisfactory performance. One of the main objectives of this thesis is to study the joint design of coding and control for an efficient use of the limited communication resources.

1.2 Contributions

In this thesis, we mainly consider about network control system optimization, in which the encoder and controller need to be designed to minimize the system performance. Q-learning is utilized in the design process, because it is an online learning method, and it is much more implementable compared with dynamic programming method. In addition, Q-learning is model-free learning method, only data measured along the system trajectories are utilized in system design process.

1. **Co-design of encoder and controller for feedback control systems over binary symmetric channels (BSC)**
   An iterative design based on Q-learning has been proposed to obtain a pair of encoder and controller that can optimize a finite-horizon cost function. Three encoder strategies,

memory encoder, memoryless encoder and predictive encoder have been considered in this component, and simulation results have shown the merits of the proposed method. With memory and memoryless encoder strategies, the design process is model free, while model is needed to predict estimation in predictive encoder design.

Current approaches to network control system design all require a plant model and involve offline computation. Many are SISO (single-input and single-output) and need side-information. In basic setup, the optimal control policy was obtained by solving Bellman equation (discrete-time optimization problems) or Hamilton-Jacobi-Bellman (HJB) equation (continuous-time optimization problems). However, it is difficult to get solutions for non-linearity and non-convexity system. Motivated by Lei et al. [2011], we address the issue of joint design of encoder and control for multi-input and multi-output high order system. Heuristic and numerical methods needed to deal with intractabilities. Reinforcement learning is well studied to this problem and has lots of success in other applications. Therefore, it is now proposed so particular. It has advantages of being online learning and model-free. It will be applied to multi-input and multi-out case, without requiring side-information.

2. **Co-design of encoder and controller for feedback control systems over Gaussian channel**

For BSC, the inputs and outputs of channel are finite. In practical, there are some system with continuous channel outputs. Gaussian channel is selected since it is the most important continuous alphabet channel, modelling a wide range of communication channels.

The challenge is that the trained encoder-controller can no longer be implemented as a simple look-up table. Since the infinite-channel outputs mainly affect the controller design, three types of controller are considered, in which soft-information based controller design, hard-information based controller design and the combination of soft-hard information based controller design are studied.

For soft-information based controller, the infinite channel outputs can be fully exploited. While this process is too complex to implement in practical. The hard-information-based controller is practically implementable because of the low complexity, while it has not taken into consideration all information carried by channel outputs, such a solution is expected to cause a degradation in system performance. Here, we propose a low-complexity controller which can take advantage of both the soft and hard information of channel outputs, named the combination of soft-hard information based controller. The proposed scheme has good performance.

3. **Distributed design of encoder and controller**

For iterative design, the encoder is optimized with fixed controller and the controller is optimized with fixed encoder. However, only local optimal solutions can be got by iterative design. Therefore, distributed encoder and controller design is proposed. Both encoder and controller learn independently with their own local information. Therefore,

both of them can be optimized simultaneously. Obviously, the system performance is better than iterative design. However, global optimal cant be guaranteed. In addition, the computation decreases a lot, since only partial information is used in the design process.

1.3 Organization

- Chapter 2

  First of all, this chapter presents the basic knowledge of network control systems and some research issues are given to help understanding. Since communication channel is a very important link of network control system, how to encode the real-valued vectors to discrete symbols is introduced. For the convenience of research, two communication channel model are given, named binary symmetric channel and Gaussian channel. These two channels are the most widely studied channel in coding theory and information theory because there are simple noise channel to analyze. Many problems in communication theory can be reduced to a binary symmetric channel or Gaussian channel. The difference between binary symmetric channel and Gaussian channel is the channel outputs, which are finite channel outputs of binary symmetric channel and infinite channel outputs of Gaussian channel.

  After that, basic background about reinforcement learning is given. Since Q-learning is very important for our work, it is one of reinforcement learning method. Three optimal controller design method are described, which are dynamic programming, policy iteration and value iteration. For dynamic programming, it is a very fundamental optimization method in control theory. However, there are some drawbacks, which are model-based, back-forward learning method and it is difficult to get solutions on non-linear or non-convexity systems. Two model-free learning methods are presented, which are policy iteration and value iteration method. For policy iteration method, the computation is a bit complexity and there is constraint on initial condition. While value iteration process is relatively simple since the best action is selected on each learning stage. Furthermore, no requirement is needed for initial condition. One specific value iteration method, Q-learning, is introduced to get the optimal control policy. By using model-based and model-free learning methods, the optimal control policy of linear quadratic case is given. Because the design objective of this thesis is to minimize the system linear quadratic cost. Finally, research works on co-design about encoder and controller are reviewed. Based on these works, motivation of the research works is also described.

- Chapter 3

  This Chapter is concerned with the design of encoder and controller pair for feedback control systems over binary symmetric channels. An iterative design method based on Q-learning is proposed to obtain a pair of encoder and controller that can optimize a finite-horizon linear quadratic cost function.Three encoder strategies, memoryless encoder, memory encoder and predictive encoder, are considered. The proposed design can

be implemented online, and has the potential to provide better performance. Compared with traditional control optimization method, the proposed design method in this chapter is model-free, only data measured along the system trajectories is utilized. Simulations are provided to show the effectiveness and the merits of the proposed method.

- Chapter 4
  Compared with Chapter 3, the problem of infinite channel outputs is considered over the co-design process. How the generalization to infinite-output channels affected the optimization of the encoder-controller is studied. Here, Gaussian channel has all these properties and it is easy to analysis and computer. Therefore, co-design of encoder and controller pair for feedback control systems over Gaussian channels generates. Since the encoder design has no affects by infinite channel outputs, we make more efforts on controller design. Based on the information utilized by controller design, three controller design processes are given, which are soft-information based controller design, hard-information based controller design and the combination of soft-hard information controller design. For soft-information based controller design, all the real-valued channel outputs are utilized to the design. While this process is too complex to implement in practical. To simplify the design process, hard-information controller design is proposed, which is simplified to finite channel outputs. Actually, the design process of binary symmetric channel can be utilized directly. However, only partial information of channel outputs take part in the controller design, which might deteriorate the system performance. To balance these problems, the combination of soft-hard information controller design is given, which takes these two essentials.

- Chapter 5
  As mentioned before, only local optimal solutions are obtained by iterative design method in Chapter 3 and Chapter 4. Therefore, distributed design is proposed to improve the system performance. First of all, distributed Q-learning is introduced, where the multi-agents are cooperative and non-competitive with the global objective. All the agents try to learn and evaluate the optimal stationary control strategy to minimize the one stage cost. By distributed design, the optimized encoder and controller can be obtained simultaneously. Obviously, the system performance is better than iterative design. In the future, it is better to expand the system to complex networked control system, such as multi-sensors, multi-controllers or even multi-communication-channels.

- Chapter 6
  This chapter covers some conclusions and outline future plans of the research.

Chapter 2

Literature Review

2.1 Networked Control Systems

2.1.1 Introduction

Feedback control system is very important in many domains, e.g. industrial application, aerospace, automation engineering. The functional of feedback controller is adjusting the output of the system to satisfy the ideal system performance. State-space function or frequency domain theories are utilized to solve most linear or non-linear feedback control system [Brogan, 1991, Bubnicki and Bubnicki, 2005, Bellman and Kalaba, 1965]. These techniques have been proven and applied by multiple fields, e.g. control systems of power plant, control systems of flight and space shuttle controllers. With the development of the ages, the systems and environment are much more complexity which constraints the application of feedback control system. However, with the system uncertainty and the unknown environment, the requirement of operation speed and accuracy is improved, which is a big challenge of modern feedback control system . Now the intelligent controller, adaptive and learning capabilities, is proposed to meet the requirement [Lewis et al., 1998, Lin and Lee, 1991]. In complexity system, the intelligent controller should be learned and updated adaptively to resistant the interference, model dynamics and even without model structure. In addition, some applications of feedback control systems come out multi-layers nonlinearities.

The intelligent control system endow the human cognitive abilities and learning abilities, so that it can adaptive the variable environment. Hence, the intelligent control system could improve the system performance with the complex environment. Neural networks, machine learning techniques, fuzzy logic, and so on, are proposed to utilized on intelligent control systems. While we discuss the problem based on feedback control system with communication channel , named networked control systems. Because of the communication channel, the system becomes dynamical system and even generates unexpected behaviours. In general, it is difficult to analysis

or design networked control systems by traditional control theory. Further more, the research issues on networked control systems has received considerable attention.

2.1.2 Research Issues of Networked Control Systems

Networked control systems is a feedback control system with information exchanging via communication channels. It is composed with sensors, networks, controllers, plants. Network and control system are two main researches about information theory and control theory. With the growing trends, there has been considerable interest in implementing network and control system together (Antsaklis and Baillieul [2004], Wang et al. [2018], Park et al. [2018], Sahoo et al. [2016], Hespanha et al. [2007], Yang [2006], Zhang et al. [2001]). Generally speaking, there are two research areas about networked control systems, which are control of communication networks and control over networks (networked control systems). The main research on control of communication networks is about communication and networks so that the quality of service can be improved. It is widely used in congestion areas, routing control, updating data from different points, the efficient of data transmission, networking protocol, etc.. The usage of bandwidth in (Zuberi and Shin [2000], Cena and Valenzano [2002]) have different priority for different data. This kind of control strategy focuses on ensuring the boundedness of network delay and the improvement of bandwidth resource utilization rate. The system performance is simply represented by information transmission time and reliability. However, there is no research on the impact of network control system performance.

Control over network system is also named networked control systems, which is a closed loop control system over a communication network. The task of communication network (wire or wireless) is to exchange information and control signals between sensors and controllers. Compared with traditional point-to-point wiring based control, the advantages of networked control systems are low cost, reduced the system complexity, simple installation and maintenance, high reliability. In addition, it is easy to make an intelligent decision with sharing data efficiently. In addition, the system with wireless communication channel is much more mobility and flexibility. Because of these advantages, networked control systems is widely used by telemedicine system, intelligent transportation system, aircraft and manufacturing etc.. However, the communication channel is not a reliable medium, which has an adverse effect on system performance, e.g. time delay, packet loss, data rate limitation and incomplete information pattern. These problems can be either damaged the plant, system devices or degraded the system performance. Stability is very important in feedback control system, which has the high priority. Networked control system is same as feedback control system. Due to the instinct of communication channel, e.g. bandwidth limited, data loss, stability of networked control system is even harder to maintain.Walsh et al. [2002] introduce a novel method, named try-once-discard (TOD), to analysis the stability of networked control system and proves the stability of new protocol and common protocols. Azimi-Sadjadi [2003] emphasizes the stability analysis from packet losses, in which

uncertainty threshold principle is utilized. Cloosterman et al. [2009] discusses the system instability caused by time delay. Most of researches focus on finding control strategies or control system design to make sure the system stability. Furthermore, there are some other researches are system optimization in which the system cost is minimized to improve the system performance [Shakkottai et al., 2008, Sofge and White, 1990].

2.2 Elements of Channel Coding

In networked control systems, the information after sensor (real value) should be encoded to code form that the communication channel can be accepted and transmitted. Thereafter, the channel output should be converted to real value and utilized as controller input, which can be regarded as inverse process of encoder. Fig. 2.1 is the common structure about channel coding process in communication systems. Source coding is presented in Section 2.2.1 and Section 2.2.2 introduces the communication channel utilized in this thesis. The remain is the introduction about soft decoding process.

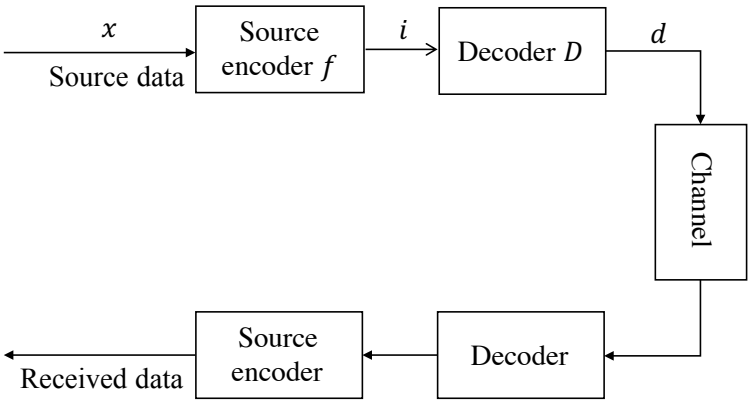

FIGURE 2.1: A common diagram of a digital communication channel system.

2.2.1 Source Coding

Source coding [Rochkind, 1975, Glasser, 1978, Kuo and Morgan, 1996] is the process that converts the information from one form to another form and the process is irreversible. Based on Fig. 2.1, source encoder encodes the source data to discrete index. In Fig. 2.1, the encoder mapping f represents the mapping of a sequence of source samples x into an integer index $i \in \mathscr{I} = \{0,1,2,3,\cdots,I\}$, named $i = f(x)$. ρ is the transmission rate. While the encoder and controller share the same $\mathscr{I}$ in this thesis, if there is no special stated. In this thesis, the encoder maps arrange of values into a region and each region is represented by a unique codeword [Gallager, 1968, Gray, 2012].

The encoder region $S_i(x)$ can be defined as

$$S_i(x) \triangleq \{x|f(x) = i\} \tag{2.1}$$

where $i \in \mathscr{I}$. $S_i(x)$ contains all source symbol x assigned the index value i. While the decoder D converts i to the estimation of x, named d. The formulation is

$$d = D(i) \tag{2.2}$$

where the set of all d compose to codebook.

2.2.2 Communication Channels

There are many types of communication channels [Ryan and Lin, 2009, El Gamal and Kim, 2011], such as BSC, Gaussian channel, fading channel and so on. In this thesis, BSC and Gaussian channel are applied in the analysis. For discrete memoryless channel, BSC is the most widely studied channel in coding theory and information theory because it is one of the simplest noisy channel to analyze. Many problems in communication theory can be reduced to a BSC. BSC has a binary input and binary output (0 or 1) with the crossover probability p [MacKay and Mac Kay, 2003].

The transmission scheme is depicted in Fig. 3.2. The crossover probability that a character is transmitted with error is labeled p. The probability without error is $1-p$, where $p(0|0) = p(1|1) = 1-p$, $p(0|1) = p(0|1) = p$.

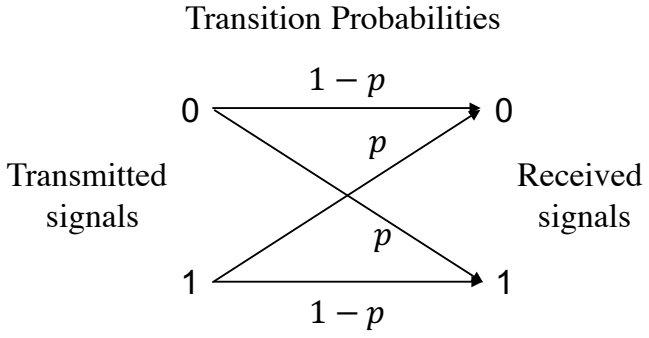

FIGURE 2.2: Binary symmetric channel (BSC) model

The Gaussian channel is the most important continuous alphabet channel, modelling a wide range of communication channels [Zaidi et al., 2014, Tatikonda et al., 2004, Tatikonda and Mitter, 2004, Freudenberg et al., 2010]. In general, a real-valued channel outputs are defined as a soft channel output [Cover and Thomas, 2012, Skoglund, 1999] and an integer-valued channel outputs are referred to as a hard channel output. In general, Gaussian channel is a very common channel and can be applied in many areas where the binary code word is added with Gaussian

noise. We choose Gaussian channel, because it provides structural and functional insights into the solution through instructive and relatively simple calculations.

For hard channel output, the design process is similar to BSC. While it is bit complex about channel output since the decoder process should be defined for real-valued channel output. The basic idea is to imitate the encoder and decoder process based on the information before the communication so that the real-valued channel output can be decoded to integer values. In this thesis, Hadamard transform [Pratt et al., 1969] is utilized to build the real-valued channel output decoding process. The following section illustrates more details about Hadamard-based soft decoding.

2.2.3 Hadamard-Based Soft Decoding

For soft channel output, how to decode the real-value to integer value is the key point. Hadamard-based decoding techniques is utilized to transform the real-valued channel output to source symbol and has been shown that the real-value can be mapped into integer values [Knagenhjelm and Agrell, 1996, Skoglund, 1999, Bao and Skoglund, 2010]. In Fig. 2.3, $P(r|i)$ is the channel error

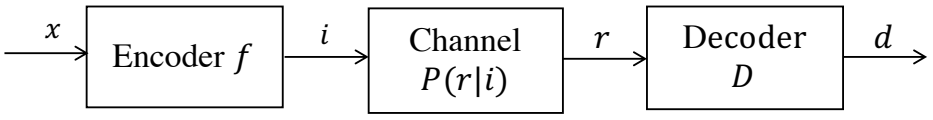

FIGURE 2.3: Quantization over Gaussian channel.

probability. The input of channel i_t is integer index $i \in \mathscr{I} = \{0,1,2,3,\cdots,2^{\rho}-1\}$ and output r is real value. The corresponding binary codeword can be represented by

$$\begin{bmatrix} b_1(l) & b_2(l) & \cdots & b_k(l) & \cdots & b_R(l) \end{bmatrix}$$

where $b_k(l), k \in \{1,2,\cdots\rho\}$ denotes the k^{th} bits of the binary codeword $b(l)$. Note that the channel input alphabet is different with the channel output alphabet in Gaussian communication system.

The following is the Hadamard-based decoding process. First, Hadamard matrix should be introduces, assumed H_n. The matrix size is $2^n * 2^n$ and it is composed by the binary elements $\{-1,1\}$. The formulation is

$$H_n = H_1 \otimes H_{n-1}, \quad H_n = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.3}$$

where $\otimes$ is Kronecker product. The l^{th} column of Hadamard matrix is denoted by $h(l)$. After that, the formulation of $h(l)$ is

$$h(l) = \begin{bmatrix} 1 \\ b_R(l) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ b_{R-1}(l) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ b_1(l) \end{bmatrix} \tag{2.4}$$

For the decoded information d, it can be expressed as

$$d = D(r) = \mathbb{E}\{x|r\} = \sum P(i=l|r)\mathbb{E}\{x|i=l\} \tag{2.5}$$

Based on Hadamard decoder process, $D(r)$ can be rewritten as

$$D(r) = \bar{T}\hat{h}(r) \tag{2.6}$$

which is a product of two matrixes. First of all, $\bar{T}$ can be calculated by

$$\begin{bmatrix}\mathbb{E}\{x|i=0\} & \mathbb{E}\{x|i=1\} & \cdots & \mathbb{E}\{x|i=2^n-1\}\end{bmatrix} = \bar{T}H \tag{2.7}$$

where $\mathbb{E}\{x|i=l\}, l \in \{1,2,\cdots,2^R-1\}$ is the conditional mean estimate and can be calculated. H is Hadamard matrix.

Then, based on [Knagenhjelm and Agrell, 1996, Skoglund, 1999], $\hat{h}(r)$ can be given as

$$\hat{h}(r) = \frac{R_{hh}\hat{P}(r)}{m_h^T\hat{P}(r)} \tag{2.8}$$

where $(\cdot)^T$ is the transposition. Furthermore, R_{hh} and m_h^T can be defined as

$$\begin{aligned} R_{hh} &\triangleq \sum_{l=0}^{2^n-1} P(i=l)h(l)h^T(l) \\ m_h &\triangleq \sum_{l=0}^{2^n-1} P(i=l)h(l) \end{aligned} \tag{2.9}$$

where $h(l)$ is the l^{th} column of the Hadamard matrix H. Note that R_{hh} and m_h is no related with the channel output r. $\hat{P}(r)$ is the only term which is affected by the channel output r, and can be given as

$$\hat{P}(r) \triangleq \mathbb{E}\{h|r, P(i) = \frac{1}{2^n}\} \tag{2.10}$$

Combined with equation (2.7), (2.32), (2.33) and (2.34), $d = D(r)$ can be obtained. It is worth to remind that $\bar{T}$ is actually the source decoding in particular and $\hat{h}(r)$ is the channel decoding.

2.3 Reinforcement Learning Review

In general case, solving Hamilton-Jacobi-Bellman (HJB) equations is the main method to get optimal controllers [Bertsekas et al., 1995, Bardi and Capuzzo-Dolcetta, 2008, Crespo and Sun, 2003, Bellman and Kalaba, 1964]. The most common method is to solve the Riccati equation, where the system model is necessary. In finite time horizon case, Riccati equation should be solved by backward method that is non-implementable in practical. For non-linear system, it is very difficult or even impossible to obtain the optimal controllers by solving nonlinear

HJB equation. In order to make the control system design more suitable to the development requirements, the concept of adaptive controller is introduced, which can optimize controller by using data measured on line and then updating control policy alternatively [Lewis et al., 2012a, Li and Krstić, 1997]. However, the optimal solution can't be guaranteed with adaptive controller. To some extent, only local optimal solution can be obtained.

Because the optimal controller designed by reinforcement learning can learn online and update adaptively, reinforcement learning [Sutton and Barto, 1998, Watkins and Dayan, 1992, Lewis and Liu, 2013] has been widely used in optimal control area. Reinforcement learning refers to the problem of a goal-directed agent interacting with an uncertain environment. The goal of an reinforcement agent is to maximize or minimize a long-term scalar reward by sensing state of environment and taking actions which affects state of environment. At each time step, reinforcement learning system receives evaluative feedback about the performance of its action, allowing it to improve the performance of subsequent actions. Several reinforcement learning methods have been developed and successfully applied in machine learning to learn optimal policies for finite-state finite-action discrete-time Markov Decision Processes (MDPs) [Monahan, 1982]. There are many reinforcement learning methods, which are dynamic programming, Monte Carlo method and temper difference learning method.

2.3.1 Basic Knowledge of Reinforcement Learning

The idea that learning by interacting with environment is the first respond of human, *i.e.* a baby can learn without being taught, such as waving arms and playing. Another example is that a student studies or plays computer games, he/she actually has a judgement about responding to their actions, and considers the influence of corresponding consequence caused by their behaviour. Parameters of environment can be detected directly over learning process and those parameters have an influence on action selections to reach learning goals. Hence, interaction among environment and action is precondition of reinforcement learning. In addition, reinforcement learning is a goal-directed learning method aiming to solve decision making problem, [Sutton and Barto, 1998, Kaelbling et al., 1996, Sutton et al., 2000, Dayan and Balleine, 2002]. The mapping from situation to actions , named learning policy, determines system performance. More specifically, it is learning from experience to maximize or minimize the total amount of reward received over time.

Figure 2.4 is the basic framework of reinforcement learning, which is divided two components, agent and environment. The agent, who is a learner, has responsibility to make a decision for action selection according to accumulative reward. All the other components of system can be regarded as environment. Current situation of environment might be affected by action and changes to new state. While all information will be feedback to agent, which is a closed system. More specifically, interaction happens to agent and environment at each direct time step t with $t = 1,2,3,\ldots$, named one episode. At each time step t, the agent takes an action a, $a \in A(a)$, where $A(a)$ is the action set and then the environment state will be changed from current state

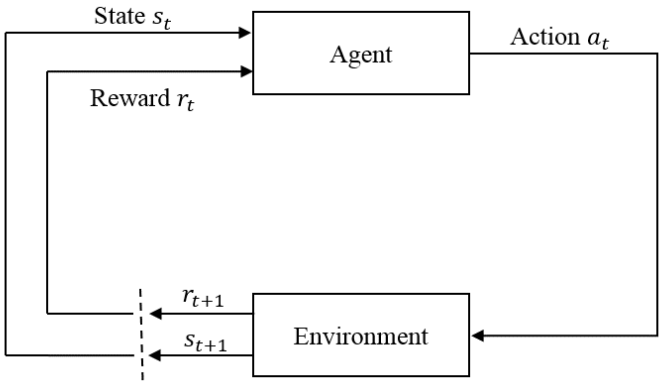

FIGURE 2.4: The agent and environment interaction in reinforcement learning.

to next-state ($s_t \rightarrow s_{t+1}$, $(s_t, s_{t+1}) \in S$), where S is a set of all possible states. Reward, which is a function or a scalar, will be generated over each time step learning process. New action should be chosen based current state and accumulated reward from beginning. This learning process will be repeating until optimal policy obtained.

2.3.2 Markov Decision Processes

Because MDPs is the fundamental formalism of reinforcement learning, it is important to discuss MDPs, Sutton and Barto [1998], Bertsekas and Tsitsiklis [1996]. First of all, let's define four sets (X, U, P, R), where X is state set and U is action set. P represents the transition probability and R is the expected immediate reward $R^u_{x,x'}$. For each state $x \in X$, the next state x' can be determined by action $u \in U$, which can be expressed as $P^u_{x,x'} = Pr\{x'|x,u\}$, $P^u_{x,x'} \in [0,1]$. Note that the Markov property means that the transition probability $P^u_{x,x'}$ is only determined by current state.

How to find the mapping from state to action is the most important problem of MDPs, named $\pi(x,u) = Pr\{u|x\}$, which can be interpreted as the action u choosing probability with current state x. In control optimization problem, such a mapping can be regarded as a feedback control gain. Therefore, a control policy determining problem can be solved by reinforcement learning. In addition, finite MDPs refers to the process with finite states and actions.

2.4 Three Optimal Control Design Methods

Dynamic programming [Bellman, 1966, Bertsekas et al., 1995, Bertsekas and Tsitsiklis, 1996] is a fundamental method of solving optimal control problem, which is a backward recursion method. Hence, it is an offline method, which can not be applied online and forward-in-time. For dynamic programming, the solution can be obtained by solving Bellman equation, which

will be described in next subsection. This thesis solves the problem with online and forward-in-time, policy iteration [Lagoudakis and Parr, 2003, Bradtke et al., 1994, Liu and Wei, 2013, Bertsekas, 2011] and value iteration [Smith and Simmons, 2004, Tamar et al., 2016, Wei et al., 2015], which are the basic method of reinforcement learning, are introduced in the following subsection.

2.4.1 Dynamic Programming

Optimal control is an extension of calculus of variations, so that how to find optimal conditions and methods for control systems design is very important. In general, finding control policies to minimiaize the system perfirmance is the key of optimal control. First of all, let's define the system state space function is

$$x(t+1) = f(x(t), u(t), t) \tag{2.11}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are state vector and control vector, f is vector of function, which is continuous and differentiable.

For optimal control problem, system performance is consisted by cost at each time step, $r_t = r_t(x_t, u_t, x_{t+1})$. Obviously, $R^u_{x,x'} = \mathbb{E}\{r_t | x_t = x, u_t = u, x_{t+1} = x'\}$, where $\mathbb{E}\{\cdot\}$ represents the expected value operator. The cost-to-go can be defined as

$$J_{t,T} = \sum_{\tau=t}^{T} \gamma^\tau r_\tau \tag{2.12}$$

where $\gamma \in [0,1]$ is discount factor, and determines the weight of cost appearing in the future. For a control policy π, the value function can be determined as

$$V^\pi_t(x) = \mathbb{E}_\pi\{J_{t,T} | x_t = t\} = \mathbb{E}_\pi\{\sum_{\tau=t}^{T} \gamma^\tau r_\tau | x_t = x\} \tag{2.13}$$

where $E_\pi\{\cdot\}$ represents the expected value of the given policy $\pi(x,u)$. Based on Markov property, the value function of policy $\pi(x,u)$ can be written as

$$\begin{aligned} V^\pi_t(x) &= \mathbb{E}_\pi\{r_t + \gamma \sum_{\tau=t+1}^{T} \gamma^{\tau+1} r_\tau | x_t = x\} \\ &= \sum_u \pi(x,u) \sum_{x'} P^u_{x,x'} [R^u_{x,x'} + \gamma \mathbb{E}_\pi\{\sum_{\tau=t+1}^{T} \gamma^{\tau+1} r_\tau | x_{t+1} = x'\}] \\ &= \sum_u \pi(x,u) \sum_{x'} P^u_{x,x'} [R^u_{x,x'} + \gamma V^\pi_{t+1}(x')] \end{aligned} \tag{2.14}$$

Note that this equation should be solved by a backward recursion, since the value function on time t should be calculated by the value function on $t+1$. Based on the optimality principle, the

optimal cost can be written as

$$\begin{aligned} V_t^*(x) &= \min_{\pi} V_t^{\pi}(x) \\ &= \min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P_{x,x'}^{u} [R_{x,x'}^{u} + \gamma V_{t+1}^{\pi}(x')] \end{aligned} \tag{2.15}$$

Based on Bellman [1966], the control policies from $t+1$ to T are already optimized which has not affected by previous or current control policy. Therefore, equation (2.15) can be written as

$$V_t^*(x) = \min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P_{x,x'}^{u} [R_{x,x'}^{u} + \gamma V_{t+1}^{*}(x')] \tag{2.16}$$

The optimal control policy is

$$\pi^*(x,u) = \arg\min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P_{x,x'}^{u} [R_{x,x'}^{u} + \gamma V_{t+1}^{*}(x')] \tag{2.17}$$

2.4.2 Policy Iteration

The above shows that dynamic programming can be solved by backward recursion, which is an off-line methods. Furthermore, the complete knowledge of system dynamic should be given. Hence, there are still lots limitations about dynamic programming. While the optimal control policy obtained by reinforcement learning method is an online learning method and the policy can be improved by data measured along the system trajectories. Both of these method are based o Bellman equation [Sutton and Barto, 1998, Lewis and Liu, 2013, Bertsekas and Tsitsiklis, 1996, Bertsekas et al., 1995, Barto et al., 1983].

In general, there are two important steps of reinforcement learning technique, which are policy evaluation and policy improvement. For policy evaluation, the value function (2.14) can be updated with the given current policy $\pi(x,u)$. While the policy can be obtained based on the updated value, which might be better or at least no worse. This step is policy improvement. In reinforcement learning algorithm, these two processes are alternatively updated until the optimal policy got.

The following in this section discusses one of reinforcement learning method, named policy iteration. In policy iteration process, the value function can be determined with current policy, while the policy is improved based on the updated value function. This process is continued until the value or the policy is stable.

Once the policy π has been improved, the new policy π' will be given, which is better than or equal to π. Then the new policy can be used to compute $V^{\pi'}$, which yield the new update policy π''. It can be expressed as

$$\pi_0 \xrightarrow{\mathbb{E}} V_0^{\pi} \xrightarrow{I} \pi_1 \xrightarrow{\mathbb{E}} V_1^{\pi} \xrightarrow{I} \pi_2 \xrightarrow{\mathbb{E}} V_2^{\pi} \xrightarrow{I} \cdots \xrightarrow{I} \pi^* \xrightarrow{\mathbb{E}} V^* \tag{2.18}$$

where $\mathbb{E}$ represents policy evaluation, I represents policy improvement. π^* and V^* are optimal policy and value.

The algorithm about policy iteration is given in Algorithm 1. δ is the error between current value and estimated value. When the error δ is sufficiently small, the policy evaluation part is completed and move to policy improvement part. It is clearly to see that this process is forward-in-time, so as to other algorithms of reinforcement learning.

Algorithm 1 Policy iteration

Intilization Arbitrary choose initial policy $\pi(x,u)$
Policy Evaluation
repeat
  $\delta \leftarrow 0$
  loop
    For each $x \in X$:
    $v \leftarrow V(x)$
    $V(x) \leftarrow \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V^*(x')]$
    $\delta \leftarrow \max(\delta, |v - V(x)|)$
  end loop
until $\delta < \theta$ (a small positive number)
Policy Improvement
policy-stable $\leftarrow$ true
loop
  for each $x \in X$:
  $b \leftarrow \pi(x,u)$
  $\pi(x,u) \leftarrow \arg\min_u \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V^*(x')]$
  if $b \neq \pi(x,u)$ **then**
    policy stable $\leftarrow$ false
  end if
  if policy stable, then stop **then**
    go to policy evaluation
  end if
end loop

2.4.3 Value Iteration

It is known that policy evaluation is an iterative computation process, where the convergence speed decreases a lot. If policy evaluation has been done, the convergence exactly to V^π would be very fast. There is an important special case that policy evaluation is stopped after just one sweep (one backup of each state) which is known as value iteration. More specifically, value iteration combines the policy improvement and truncate policy evaluation steps:

$$\begin{aligned} V_{k+1}(x) &= \min_u \mathbb{E}\{r_{t+1} + \gamma V_k(x_{t+1}) | x_t = x, u_t = u\} \\ &= \min_u \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V_k(x')] \end{aligned} \tag{2.19}$$

where $x \in X$. k is value iteration step index.

The algorithm of Value iteration is shown in Algorithm 2. The value updating equation (2.19) is a simple one-step recursion equation, not like policy evaluation of policy iteration algorithm. Compared with policy iteration, which converges with a conditions in a finite number of steps, value iteration has no constraints [Sutton and Barto, 1998, Bertsekas and Tsitsiklis, 1996].

Algorithm 2 Value iteration

Intialisation Choose V_0 arbitrary
repeat
  $\delta \to 0$
  loop
    For each $s \in S$:
    $v \to V(x)$
    $V(x) \to \min_u \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V(x')]$
    $\delta \to \max(\delta, |v - V(x)|)$
  end loop
until $\delta < \theta$, θ is efficiently small positive number.
Output a deterministic policy π, such that
$\pi(x,u) = \arg\min_u \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V(x')]$

Both policy iteration and value iteration are widely used, and it is not clear which one is better. While both of them usually converge faster than their theoretical case, especially when a good initial value function or policy is given.

2.5 Q-Learning Method

2.5.1 Q-Function

Q-function can be interpreted as quality function, also named state-action value function [Watkins, 1989, Watkins and Dayan, 1992]. Based on value function (2.15), optimal Q-function can be written as

$$\begin{aligned} Q^* &= \sum_{x'} P^u_{x,x'}[R^u_{x,x'} + \gamma V^*(x')] \\ &= \mathbb{E}_\pi\{r_t + \gamma V^*(x') | x_t = x, u_t = u\} \end{aligned} \tag{2.20}$$

where x' is next state and x is current state.

Obviously, Q-function is a function about current state x_t and action u_t, which equals to the expected returns for choosing an action u on the fixed state x and policy π. The Bellman optimality equation can be simplified as

$$V^*(x) = \min_u Q^*(x,u) \tag{2.21}$$

$$u^*_t(x) = \arg\min_u Q^*(x,u) \tag{2.22}$$

Since $V^{\pi}(x) = Q^{\pi}(x, \pi(x,u))$, Q-function can be rewritten as

$$Q^{\pi}(x,u) = \sum_{x}^{x'} P^{u}_{x,x'}[R^{u}_{x,x'} + \gamma Q^{\pi}(x', \pi(x', u'))] \tag{2.23}$$

Compared with value function, Q-function is a function about current state and action. Hence, Q-function can be viewed as a matrix with state column and action row in finite MDPs, which means Q-matrix is a lookup table for each state-action pair. It has beed discussed that the optimal policy obtained by dynamic programming (2.15) should use the system dynamics (system model). While the minimization in (2.21) and (2.22) only utilize the information from Q-function.

Each value of Q-matrix correspond to one state-action pair, where the best control action can be selected with a fixed state by using (2.22). Furthermore, Q-function can be estimated online with data measured along the system trajectories, in which the system dynamic, named the transition probabilities, is no need. Algorithms of policy iteration and value iteration with Q-function are similar to algorithm 1 and 2. These algorithms can be applied on control optimization problem, which means that the control policies obtained by policy iteration and value iteration can converge to optimal solutions.

2.5.2 Q-Learning Algorithm

Temporal Difference (TD) learning is one of very important reinforcement learning method, which is a combination of Monte Carlo method and dynamic programming [Sutton and Barto, 1998]. Both Monte Carlo method and TD method learn from experience without a exact model of their environment. Whereas, in terms of dynamic programming, both of them update estimates rely partly on an existing estimate, that is, updating every time-step without waiting until the end of the episode. TD method only needs to wait until the next time step to determine the increment for current estimated value function $V(x_t)$. At time $t+1$, it immediately generates a target and makes an updating based on the reward r_{t+1} and the estimate $V(x_{t+1})$. The simplest TD method, known as TD(0), is

$$V(x_t) \leftarrow V(x_t) + \alpha[r_{t+1} + \gamma V(x_{t+1}) - V(x_t)] \tag{2.24}$$

where the target for the TD update is $r_{t+1} + \gamma V_t(x_{t+1})$. α is learning rate and γ is discount factor. This idea is the basis of the important development in the field of reinforcement learning, which minimizes the error between the estimation value and current value. With the temporal difference error $r_{t+1} + \gamma V(x_{t+1}) - V(x_t) = 0$, the policy obtained can converge.

There are two important algorithms of TD learning method, which are state-action-reward-state-action (SARSA) [Sutton, 1996, Sprague and Ballard, 2003] and Q-learning. Since all our work are based on Q-learning, the following illustrates Q-learning algorithm. Q-Learning is defined

as an off-policy TD control algorithm, which is one of the most important breakthrough in reinforcement learning [Watkins and Dayan, 1992]. During each interaction between the agent and the environment, the agent choose an action u based on the current state x. After that, the state of the environment will change to a new state x_{t+1}, and the agent will get a feedback (reward) from the environment simultaneously. Thereafter, the action value function $Q(x,u)$ will be updated. The agent chooses actions based on the maximum Q-value in the current state.

The core of the Q-learning algorithm is value iteration update of value function. The Q-value for each state-action pair is initially chosen by the agent. The general form of temporal difference equation is

$$NewEstimate = OldEstimate + StepSize[Target - OldEstimate].$$

Then, it is updated each time, an action is issued and a reward is observed, based on the following expression:

$$Q(x_t,u_t) \leftarrow Q(x_t,u_t) + \alpha[r_{t+1} + \gamma \min_u Q(x_{t+1},u) - Q(x_t,u_t)] \tag{2.25}$$

where α is the learning rate, named step-size parameter, which is used in the incremental method to describe changes during time step shift. Normally, $\alpha \in [0,1]$, and satisfies the condition: $\sum_t^\infty \alpha_t = \infty$, $\sum_t^\infty a_t^2 < \infty$, while r_{t+1} is the reward given at time $t+1$.

γ is a parameter, named discount rate ($\gamma \in (0,1)$), which determines the present value of future rewards. If $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence is bounded. If $\gamma = 0$, the objective is to learn how to choose action u_t to minimize the immediate reward r_t.

As a model-free learning algorithm, it is not necessary for the agent to have any prior information about the system, such as the transition probability from one state to the next. Thus, it is a highly adaptive and flexible algorithm. The algorithm about general Q-learning method is illustrated by algorithm 3.

Algorithm 3 General Q-learning algorithm

Initialised — $Q(s,a)$ arbitrary
for $n = 1 : N$ **do**
  Initialise s
  for $t = 1 : T$ (for each time step of episode) **do**
    Choose u_t from x_t using policy derived from Q, $u_t = \arg\min_{u_t} Q(x_t,u_t)$.
    Take action u_t, and then observe the reward r_t, the next state x_{t+1}
    $Q(x_t,u_t) \leftarrow Q(x_t,u_t) + \alpha[r_t + \gamma \min_u Q(x_{t+1},u) - Q(x_t,u_t)]$
    $x_t \leftarrow x_{t+1}$
  end for
end for

where n represents one episode, that is the learning iteration times. t here is each time step of one episode. While Q-value updating function in optimal control can be written as $error = r(x_t,u_t) + \gamma \min_u Q(x_{t+1},u) - Q(x_t,u_t)$. In general, TD method updating the estimated value by minimizing the TD error.

Control optimization problem can also be obtained by Q-learning algorithm. Hence, the Bellman equation can be replaced by

$$V(x_t) = r(x_t) + \gamma V^{\pi}(x_{k+1}) \tag{2.26}$$

which satisfies for each observed data set (x_t, u_t, x_{t+1}). x_t is current state, u_t is current action and next state is x_{t+1}.

To applied Q-learning into policy iteration algorithm, the policy evaluation updating equation is

$$V_{k+1}(x_t) = r(x_t, \pi_k(x_t)) + \gamma V_{k+1}(x_{t+1}) \tag{2.27}$$

The policy improvement function is

$$\pi_{k+1}(x_t) = \arg\min_{\pi(\cdot)}(r(x_t, \pi(x_t)) + \gamma V_{k+1}(x_{t+1})) \tag{2.28}$$

This algorithm should be selected admissible control policy $\pi_0(x_t)$. Since k is the iteration times, the algorithm starts with $k = 0$ and iterates on k until convergence.

For value iteration using TD learning method, the policy improvement process is similar with policy iteration algorithm. While the policy evaluation procedure is

$$V_{k+1}(x_t) = r(x_t, \pi_k(x_t)) + \gamma V_k(x_{t+1}) \tag{2.29}$$

Because this is a recursive equation, there is no requirement on initial condition. While policy evaluation (2.27) of policy iteration is a fixed point equation, the initial condition should be required to ensure the solution.

2.6 Discrete-Time LQR Optimization By Q-Learning

Consider the discrete-time linear quadratic regulator (LQR) problem, where the MDPs is deterministic and satisfies the state space function is

$$x_{t+1} = Ax_t + Bu_t \tag{2.30}$$

where the time index is t. $x_t \in X = \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$. The system performance with LQ cost is

$$J_t = \frac{1}{2}\sum_{\tau=t}^{T} r_\tau = \frac{1}{2}\sum_{\tau=t}^{T}(x_\tau^T D x_\tau + u_\tau^T E u_\tau) \tag{2.31}$$

where T is time horizon. D matrix is the weight matrix of state x_t, which differs from value function $V(\cdot)$. Similarly E is the constraints of u_t. Both D matrix and E matrix are positive definite. The associated value function is

$$V(x_t) = \frac{1}{2}(x_t^T D x_t + u_t^T E u_t) + V(x_{t+1}) \tag{2.32}$$

where $V(0) = 0$. This equation can be viewed as Bellman equation for LQR.

For LQR, the value is considered as quadratic, which can be formulated as

$$V_t(x_t) = \frac{1}{2} x_t^T P_t x_t \tag{2.33}$$

Combining equation (2.32) and equation (2.33),

$$\begin{aligned} 2V(x_t) &= x_{t+1}^T P_{t+1} x_{t+1} + x_t^T D x_t + u_t^T E u_t \\ &= x_t^T D x_t + u_t^T E u_t + (Ax_t + Bu_t)^T P_{t+1} (Ax_t + Bu_t) \end{aligned} \tag{2.34}$$

For LQR case, the control policy can be assumed as constant value. That is $u_t = \mu(x_t) = -Kx_t$. Hence, equation 2.34 can be written as

$$\begin{aligned} 2V(x_t) &= x_t^T P_t x_t \\ &= x_t^T D x_t + x_t^T K^T E K x_t + x_t^T (A - BK)^T P_{t+1} (A - BK) x_t \\ &= x_t^T (D + K^T E K + (A - BK)^T P_{t+1} (A - BK)) x_t \end{aligned} \tag{2.35}$$

Obviously, it holds

$$(D + K^T E K + (A - BK)^T P_{t+1} (A - BK)) - P_t = 0 \tag{2.36}$$

which is a Lyapunov equation. In general, optimal control design by the Lyapunov equation is the standard procedure in control systems theory.

2.6.1 Model-Based

Q-function has been defined in Section 2.5.1. For discrete-time LQR case, the Q-function is

$$Q(x_t, u_t) = \frac{1}{2} (x_t^T D x_t + u_t^T E u_t) + V_{t+1}(x_{t+1}) \tag{2.37}$$

Since the control policy $u_t = \mu(x_t) = -Kx_t$, Q-function can be written as

$$Q(x_t, u_t) = x_t^T D x_t + u_t^T E u_t + (Ax_t + Bu_t)^T P_{t+1} (Ax_t + Bu_t) \tag{2.38}$$

where P_{t+1} is the Riccati solution in finite time horizon. Q-function can be formulated by matrix, which is

$$Q(x_t, u_t) = \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} A^T P_{t+1} A + D & B^T P_{t+1} A \\ A^T P_{t+1} B & B^T P_{t+1} B + E \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \tag{2.39}$$

Applying $\frac{\partial Q(x_t, u_t)}{\partial u_t} = 0$, the optimal control is

$$u_t = -(B^T P_{t+1} B + E)^{-1} B^T P_{t+1} A x_t \tag{2.40}$$

This method need to know the system dynamics to obtain the optimal control policy.

2.6.2 Model-Free

This section describes the optimal control policy obtained without system model. First of all, Define the Q-value function as

$$\begin{aligned} Q(x_t,u_t) &= \frac{1}{2}\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T G \begin{bmatrix} x_t \\ u_t \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \begin{bmatrix} G_{xx} & G_{xu} \\ G_{ux} & G_{uu} \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \end{aligned} \tag{2.41}$$

where G is kernel matrix. The optimal control policy is

$$u_t = -G_{uu}^{-1} G_{ux} x_t \tag{2.42}$$

Obviously, the optimal control policy can only be obtained by kernel matrix G, which can be utilized online with using data measured along the system trajectories. The following discusses how to get kernel matrix G. Based on the Weierstrass higher order approximation theorem, there is a dense basis set $\phi(x_t,u_t)$ such that

$$\begin{aligned} Q(x_t,u_t) &= Q(z_t) \\ &= \frac{1}{2} z_t^T G z_t \\ &= W^T \phi(z_t) \end{aligned} \tag{2.43}$$

where $z_t = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T$. W is the vector of the elements of matrix G. The basis vector $\phi(z_t)$ is composed with quadratic terms of z_t. Since the size of G matrix is $(n+m)\times(n+m)$, the vector W is the upper triangle of G. Hence, there are $\frac{(n+m)(n+m+1)}{2}$ elements of W.

Q-learning Bellman equation of value iteration can be written as

$$W_{k+1}^T \phi(z_t) = x_t^T D x_t + u_t^T E u_t + W_k^T \phi(z_{t+1}) \tag{2.44}$$

This is a recursive equation and the only unknown term is W_{k+1}. It can be solved online using methods from adaptive control such as least square method, gradient method or recursive least square method (RLS). Therefore, Q-learning is implemented by algorithm 4.

The termination condition is no further updating Q-function or control policy at each step k. This algorithm can be applied to adaptive control problems. Note that this algorithm is model-free, that is to say only data measured along the system trajectories can be utilized online learning process. Persistence of excitation is considered during applying RLS to equation 2.44.

Algorithm 4 Discrete-Time LQR Optimization By Q-Learning

Initialized
Arbitrary choose an initial control policy $u_t = -K^0 x_t$
Arbitrary choose W_0 and the iteration index $k = 0$
loop
Get the data set $(x_t, u_t, x_{t+1}, u_{t+1})$ with $u_t = K^k x_t$, $u_{t+1} = -K^k x_{t+1}$.
Calculate the basis vector $\phi(z_t)$ and $\phi(z_{t+1})$.
One step update vector W by applying RLS to equation 2.44.
Repeat updating vector W until RLS converges.
Reduction the kernel matrix G with W_{j+1},
Update the control policy $u_t = -G_{uu}^{-1} G_{ux} x_t$
$k = k + 1$
end loop

Many researches on controller design by reinforcement learning comes out recently. Bradtke et al. [1994], Lewis and Vrabie [2009], Lewis et al. [2012b] shows that the optimal control gain can be obtained by Q-learning in infinite time horizon case, in which the optimized control gain can converge to the solution obtained by algebra Riccati equation (ARE) and has been proved. Furthermore, Luo et al. [2014] discusses Q-learning for optimal control of continuous-time systems. Zhao et al. [2015, 2014] illustrates control gain obtained by Q-learning with finite time horizon case. The key difference is the time dependent basis function. However, it is very difficult to decide the time-dependent basis function, which is decided by trail and error. To overcome this problem, the controller design in this paper combines the finite time horizon Q-learning technique and infinite time horizon technique. More specifically, there is a separately Q-learning process on each time step of finite time horizon. With the system running, Q-value matrix is updated separately on each time step. The optimized control policy can be obtained on each time step after value iteration and they are close to the solution obtained by ARE. Note the basis function is different on each time step.

2.7 Co-Design Encoder and Controller Problem

Over the decades, a lot of effort has been devoted to stabilization and optimal control design over communication channels and networks (Delchamps [1990], Zhang et al. [2014], Fagnani and Zampieri [2003], Shu and Middleton [2011]). However, due to distributed architecture and incomplete information pattern, analytic design is extremely difficult, and this motivates the research on heuristic and numerical methods. In Lei et al. [2011], an iterative design method was proposed to find encoder and controller mapping that can optimize a finite-horizon linear quadratic cost function. The basic idea is to design optimal encoder for a fixed controller and then design optimal controller for a fixed encoder. Different system models are given, which are full side-information system model, partial side-information system model and no

side-information system model. For full side-information system model, a certainty equivalence controller is given, and all other system models utilize this optimized certainty equivalence controller. For the encoder design component, it is easy to get solution on full side-information model, and most results shown in the paper are obtained from full side-information. It is unimplementable in practical since it is an ideal model. While the computation of partial side-information and no side-information system growth exponential with time horizon. It should be emphasised that the design method in this paper is model-based and off-line training method. Furthermore, only scalar case is considered in Lei et al. [2011]. The design of an encoder-controller pair for control performance optimization has been considered in Freudenberg and Middleton [2009], Middleton et al. [2009]. Fagnani and Zampieri [2003] is an analysis paper about linear scalar systems with a stabilizing quantized feedback control. For the recent works, separated design of encoder and controller was proposed in Rabi et al. [2016], because of dual effect between two agents, controls-forgetting encoders or certainty equivalence controls were assumed. However, these papers only showed on scalar case, which means single-input and single-output are discussed.

Similar idea together with the approximate dynamic programming (ADP) was employed to design vector quantizer in Bao et al. [2010], in which controller design is same with Lei et al. [2011] and ADP-based encoder design was given. It is clearly shown that the system parameters were utilized in the ADP-based encoder design in Bao et al. [2010]. Thereafter,Shirazinia et al. [2015] studies a joint source-channel vector quantization problem, the optimality is defined as minimizing the end-to-end mean square error. Iterative design method is also utilized to find encoder-controller pairs. While system information is necessary. Similar case appeared on Shirazinia et al. [2014], which is another application on compressed sensing. However, it is very rare to know the system parameters in practical.

Motivated by some earlier works, we address the issue of joint design of encoder and controller for multi-input and multi-output case by using Q-learning method in this thesis. A significant advantage of Q-learning is that no model information is required, and it has the potential to provide better performance than existing methods, as will be shown later. Since the encoder and controller is optimized based on the data grabbed during system running, the design complexity decreases a lot compared with current research about co-design encoder and controller. In addition, the method of encoder and controller design in this paper is considered as online learning method, which is more implementable. Note that, this thesis emphasize the control over low-rate noise channel, the effects of delay is neglected. Last but not least, system model considered in this thesis don't need the help by side-information.

Chapter 3

Co-Design of Encoder and Controller for Feedback Control Systems Over Binary Symmetric Channels Using Q-Learning

This Chapter is concerned with the design of encoder and controller pair for feedback control systems over binary symmetric channels. An iterative design method based on Q-learning is proposed to obtain a pair of encoder and controller that can optimize a finite-horizon linear quadratic cost function. Three encoder strategies, memoryless encoder, memory encoder and predictive encoder, are considered. The proposed design can be implemented online, and has the potential to provide better performance. Compared with traditional control optimization method, the proposed design method is model-free, only data measured along the system trajectories is utilized. Simulations are provided to show the effectiveness and the merits of the proposed method.

3.1 Introduction

With the rapid development of communication technology, an increasing number of control systems use communication channels or networks to transmit data. Although using communication networks/channels has many advantages such as lower cost, less system wiring, and more flexibility, it has also raised many issues and challenges to design, e.g. time delay, packet loss, data rate limitation, and incomplete information pattern [Antsaklis and Baillieul, 2007, Ploplys et al., 2004, Delchamps, 1990, Zhang et al., 2014, Shu and Middleton, 2011].

Due to distributed architecture and incomplete information pattern, analytic design is extremely difficult, and this motivates the research on heuristic and numerical methods. Most of research

work about co-design encoder and controller system focus on system stability, such as Fagnani and Zampieri [2003], Montestruque and Antsaklis [2004], Delchamps [1990]. Not too much work make efforts on feedback control system optimization with wireless communication channel, where model is necessary in these design process and even only scalar case is considered. Motivated by some earlier works, we address the issue of joint design of encoder and controller for multi-input and multi-output case by using Q-learning method in this Chapter. A significant advantage of Q-learning is that no model information is required, and it has the potential to provide better performance than existing methods, as will be shown later. Since the encoder and controller is optimized based on the data grabbed during system running, the design complexity decreases a lot compared with current research about co-design encoder and controller. In addition, the method of encoder and controller design is considered as online learning method, which is more implementable. Note that, this design problem emphasize the control over low-rate noise channel, the effects of delay is neglected. Last but not least, there is no constraint on side-information, which is the information send back from controller to encoder.

3.2 Preliminaries

In this section, we first introduce a general feedback control system with encoder and controller over communication channel in Section 3.2.1. In Section 3.2.2, the design problem is formulated.

3.2.1 System Model

Consider a feedback control system shown in Fig. 3.1. The system state-space equation is

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + v_t \\ y_t &= Cx_t + e_t \end{aligned} \tag{3.1}$$

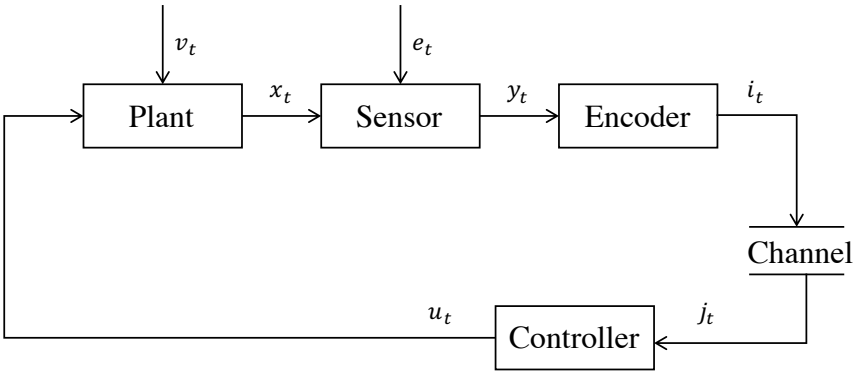

FIGURE 3.1: A general feedback control system over communication channel.

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ are the state, the control and the measurement, respectively. The matrix (A,B) is controllable and (A,C) is observable. v_t and e_t are process noise and measurement noise, which are independent and identically distributed. Furthermore, the initial state x_0, v_t and e_t are assumed as Gaussian distribution with mean zero.

The encoder is a mapping from the various measurement values y_t to a discrete set of symbols. The index of each symbol is represented by i_t and assumed to take values in the set $\mathscr{I} = \{1, \cdots, I\}$, where L_i is determined by the transmission rate ρ as $\rho = \log_2 I$. The encoder mapping f_t is expressed as

$$i_t = f_t(y_0^t) \tag{3.2}$$

where y_0^t is $y_0, y_1, \cdots, y_t$.

The symbol is transmitted through the channel, and then picked up by the controller. Due to the noise or other factors, the picked symbol may not be the same as the original one, and this is described by the following channel mapping function.

The discrete index i_t is then encoded to binary codes that contains digital 0 and 1 only. The binary codes are sent over the channel and then decoded to $j_t \in \mathscr{J} = \{1, \cdots, J\}$. The channel mapping can be described as function:

$$j_t = p(i_t) \tag{3.3}$$

where j_t has the same code book as i_t. The transition probability function $p(j_t|i_t)$ is used to characterize the channel.

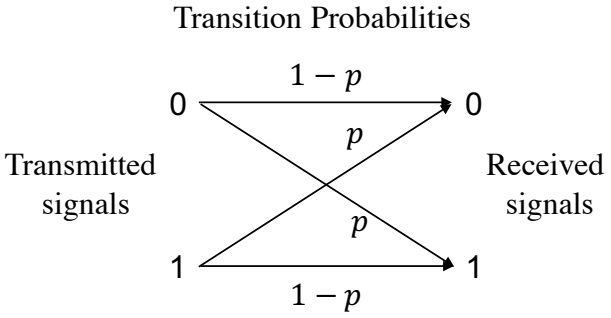

FIGURE 3.2: Binary symmetric channel model

For discrete memoryless channel, binary symmetric channel (BSC) is the most widely studied channel in coding theory and information theory because it is one of the simplest noisy channel to analyze. Many problems in communication theory can be reduced to a BSC. BSC has a binary input and binary output (0 or 1) with the crossover probability p [MacKay and Mac Kay, 2003]. The transmission scheme is depicted in Fig. 3.2. The crossover probability that a character is transmitted with error is labeled p. Hence, the probability without error is $1-p$, where $p(0|0) = p(1|1) = 1-p$, $p(0|1) = p(0|1) = p$.

The controller picks up the channel output j_t, and then generates control signal u_t, namely

$$u_t = g_t(j_t) \tag{3.4}$$

3.2.2 Problem Statement

The design objective is to find a pair of encoder mapping set $f_0^*, f_1^*, \cdots, f_{T-1}^*$ and controller mapping $g_0^*, g_1^*, \cdots, g_{T-1}^*$ that can optimize the following cost function

$$\mathbb{E}\{J_{tot}\} \triangleq \mathbb{E}\{\sum_{t=0}^{T-1}(x_{t+1}^T D x_{t+1} + u_t^T E u_t)\} \tag{3.5}$$

where matrix D and E are weight sequence of state and control, which are symmetric and positive definite.

3.3 Encoder Design by Q-Learning

The objective of this section is to find encoder mappings at each time step with fixed controller so as to minimizing the expected overall linear quadratic cost (3.5). More specifically, the optimized encoder mapping set $f_0^*, f_1^*, \cdots, f_{T-1}^*$ can be obtained by the Q-learning method.

3.3.1 Q-Value Updating Rule

Q-learning method [Watkins, 1989, Sutton and Barto, 1998] is one of reinforcement learning thechnique, with which the agent tries an action at one specific state, and do evaluation based on the reward or penalty it received. By trying all actions in all states repeatedly, the optimal policy can be got by long-term discounted reward.

Under the framework of Q-learning, encoder is modelled as the agent, all the other components are viewed as the environment (see Fig. 3.3). The input of encoder y_t is agent state during Q-learning process and the discrete index $i_t \in \mathscr{I}$ represents the agent action. The encoder mapping from y_t to i_t can be regarded as a look-up table. More specifically, each y_t can be encoded to an index i_t and the index is chosen that can minimize the system performance.

Q-matrix, to be the brain of agent, representing the memory of what agent has learned through experience. Each Q-value in the Q-matrix is the accumulative reward of the corresponding state y_t and action i_t. The rows of matrix Q are states of agent, and the columns represent the possible actions. Since the problem proposed is finite-time horizon case, the encoder mapping is different at different time t. Hence, the purpose is to find an optimized encoder mappings over the interval $t = 0$ to $T-1$, named $f_0^*, f_1^*, \cdots, f_{T-1}^*$. The Q-matrix also can be expressed as $Q_0, Q_1, \cdots, Q_{T-1}$, which means Q-matrix is updated independently at each time t and each encoder mapping

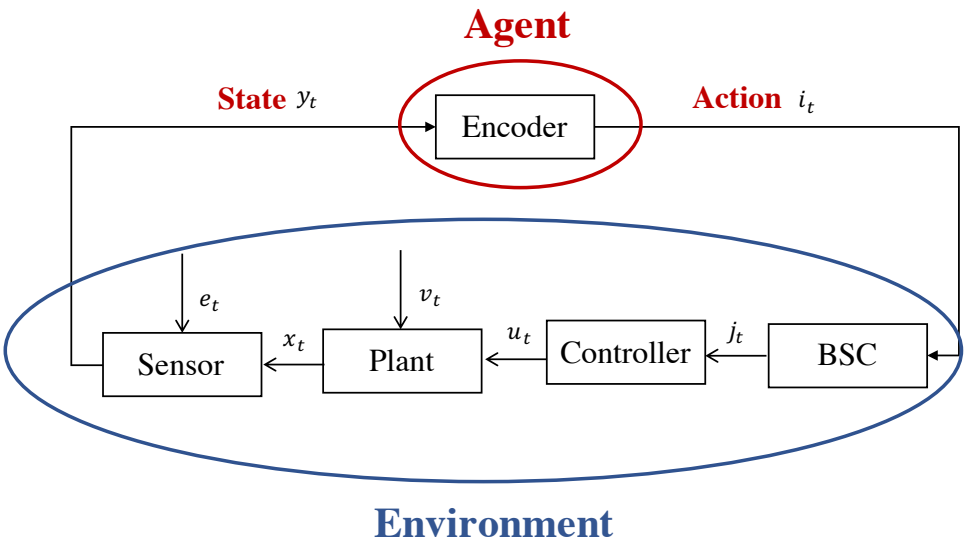

FIGURE 3.3: Basic framework of Q-learning.

f_t can be derived based on its corresponding Q-matrix Q_t. Therefore, Q-matrix is viewed as 3-dimensional matrix, the three dimensions are composed by state y_t, action i_t and system time t, named $Q(y_t, i_t, t)$.

Q-table			Actions i_0				
			1	2	3	$\cdots$	$2^{\rho n}$
States	1	$y_0^{(1)}$	0	0	0	$\cdots$	0
	2	$y_0^{(2)}$	0	0	0	$\cdots$	0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	l	$y_0^{(l)}$	0	0	0	$\cdots$	0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	L	$y_0^{(L)}$	0	0	0	$\cdots$	0

TABLE 3.1: Q-Learning table at time $t = 0$ of states by actions that is initialized to zero.

The following is how to build the Q-matrix on time $t = 0$ and its updating process. First of all, x_0 follows Gaussian distribution with mean 0, it is easy to generate a sample set X_0 of x_0, so as to sample set Y_0 of y_0 . Each element of the sample set is regarded as an agent state, which is one row of Q-matrix. The sample size is the number of Q-matrix row. Then, all options of encoder index i_t are columns of Q-matrix. The number of Q-matrix columns are determined by system dimensions n and channel transmission rate ρ, which is $2^{\rho n}$.

Since the agent starts out knowing nothing, the matrix Q is initialized to zero (see TABLE 3.1). l represents the index of sample Y_0 with size L and $y_0^{(l)}$ is the l^{th} element of Y_0. Obviously, the sample set of Y_t at each time step has same sample size. In basic Q-learning framework, l represents each step of one episode. η is defined as the iteration times.

Each cell is updated through training. The updating equation at time t is

$$Q(y_t^{(l)}, i_t^{(l)}, t) \leftarrow Q(y_t^{(l)}, i_t^{(l)}, t) + \alpha[r_t^{(l)} + \gamma \min_i Q(y_t^{(l+1)}, i, t) - Q(y_t^{(l)}, i_t^{(l)}, t)] \tag{3.6}$$

where γ is discount factor with $\gamma \in [0,1]$. α is the learning rate, named step-size parameter, which is used in the incremental method to describe changes during time step shift. Normally, $\alpha \in [0,1]$, and satisfies the condition: $\sum_t^\infty \alpha_t = \infty$, $\sum_t^\infty a_t^2 < \infty$.

With l^{th} sample value of Y_t and action i_t, the corresponding Q-value of Q-matrix on time t is $Q(y_t^{(l)}, i_t^{(l)}, t)$. While $\min_i Q(y_t^{(l+1)}, i, t)$ is the minimal Q-value with fixed state $y_t^{(l+1)}$ and the corresponding action is i.

Since the inside loop is from $y_t^{(1)}$ to $y_t^{(L)}$, the rule of next state calculation is simply moving one by one, such as from $y_t^{(l)}$ to $y_t^{(l+1)}$. While the reward $r_t^{(l)}$ is different among different encoder designs and it will specified in the following sections.

The rule of i_t selection is based on the minimal Q-value in Q-matrix, which can be formulated as

$$i_t^{(l)} = \arg\min_i Q(y_t^{(l)}, i, t) \tag{3.7}$$

where the corresponding column index of minimal Q-value $\min_i Q(y_t^{(l)}, i, t)$ on fixed state $y_t^{(l)}$ is chosen as action. Obviously, the updated Q-value in Q-matrix is $Q(y_t^{(l)}, i_t, t)$.

Q-table			Actions i_0				
			1	2	3	$\cdots$	$2^{\rho n}$
States	1	$y_0^{(1)}$	$Q(y_0^{(1)}, 1, 0)$	0	0	$\cdots$	0
	2	$y_0^{(2)}$	$\min_i Q(y_0^{(2)}, i, 0)$				
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	l	$y_0^{(l)}$	0	0	0	$\cdots$	0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	L	$y_0^{(L)}$	0	0	0	$\cdots$	0

TABLE 3.2: Q-matrix updating after the initialization in TABLE 3.1.

There is an example at $t = 0$. The initial Q-matrix at $t = 0$ has been given in TABLE 3.1. After that, the inside loop starts from $y_0^{(1)}$. Based on the action section equation (3.7), we choose $i_0^{(1)} = 1$ as i_0 on $y_0^{(1)}$ of Q-matrix at $t = 0$. Note that multiple values are same in same row and all of them are the minimal value, the action i_t is selected by the first value. The updated Q-value is $Q(y_0^{(1)}, 1, 0)$ in this step. While the next state is $y_0^{(2)}$. Hence, the estimation value $\min_i Q(y_0^{(2)}, i, 0)$ is decided by the second row, which means the minimal Q-value is chosen on the next state $y_0^{(2)}$. In TABLE 3.2, the updated Q-value $Q(y_0^{(1)}, 1, 0)$ and the estimated $\min_i Q(y_0^{(2)}, i, 0)$ are marked by red text.

In TABLE 3.2, the updating equation is $Q(y_0^{(1)}, i_0^{(1)}, 0) \leftarrow Q(y_0^{(1)}, i_0^{(1)}, 0) + \alpha[r_0^{(1)} + \gamma \min_i Q(y_0^2, i, 0) - Q(y_0^{(1)}, i_0^{(1)}, 0)]$. To realize online learning, the next updating should follow on the system state

$t = t$, in which the Q-value can be expressed as $Q(y_t^{(l)}, i_t^{(l)}, t)$. The learning process with system time step continue forward until $t = T - 1$.

By that analogy, the learning starts from $t = 0$ again and the updating Q-value is $Q(y_0^{(2)}, i_0^{(2)}, 0)$ until one episode end that is $l = L$. Thereafter, the encoder learns more through further episodes, it will finally reach convergence values for Q-matrix on each time step t.

The convergence of Q-learning has been proven by Watkins and Dayan [1992], Watkins [1989], Sutton and Barto [1998]. Once these Q-matrixes get close enough to a state of convergence, the encoder can make the optimal decision to choose an i_t for Y_t. The encoder mapping can be got by Q-matrix on each time t. In the remainder of this section, we propose three encoding strategies according to different rewards.

3.3.2 Memoryless Encoder Design

For memoryless encoder, only data measured along system trajectory at time t are utilized in the design process, which can be regarded as model free. The encoder policy can be described as

$$i_t^* = \arg\min_{i \in \mathscr{I}} \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t) | y_t, i_t = i\} \tag{3.8}$$

The reward r_t in Q-learning process is defined as $x_{t+1}^T D x_{t+1} + u_t^T E u_t$, since the selection of i_t affects u_t and x_{t+1}. Note that the data utilized in the design process are x_t, u_t and x_{t+1}.

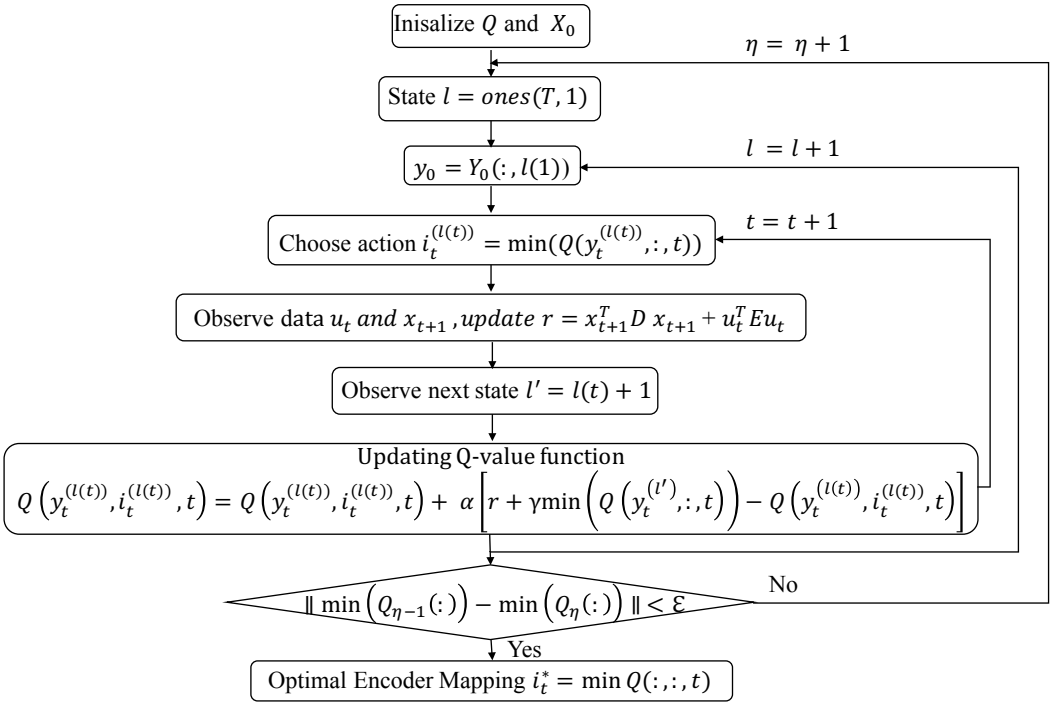

FIGURE 3.4: The flow chart of memoryless encoder design by Q-learning (model-free and online learning method)

The design process is shown in Fig. 3.4. The sample set Y_t can be obtained by online data. Because of the feedback system, with initial Y_0, it is easy to get $Y_1, Y_2, \cdots, Y_{T-1}$. l is an index vector Y_t in Q-learning with size $T * 1$. Next state calculation in Q-learning simply plus one on current state. For inside loop, it follows system time t, where Q-value function is updated separately on each time t. Hence, it is an online learning process. Thereafter, the outside two loops are standard Q-learning framework.

The algorithm will be terminated with a small enough error ε between current Q-value and previous Q-value. In Q-value matrix, the optimized encoder mapping likes a looking-up table, which means the output of encoder i_t is chosen based on the corresponding minimal Q-value on Y_t. Hence, the encoder mapping is from sample set Y_t to the optimized encoder index set obtained from minimal Q-value matrix.

3.3.3 Memory Encoder Design

Memory encoder, which memorizes the previous information, is considered. At time t, the data used by encoder design is x_0^t and u_0^t, where x_0^t and u_0^t means the system state x and control action u record over the time interval from 0 to t.

The encoder policy updating equation is

$$i_t^* = \arg\min_{i \in \mathscr{I}} \mathbb{E}\{\sum_{\tau=0}^{t} (x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau) | y_t, i_t = i\} \tag{3.9}$$

The reward r_t is defined as $\sum_{\tau=0}^{t} x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau$, which considers the previous data measured along the system trajectories. The Q-value updating equation is same with memoryless encoder design, named equation (3.6). The design process is still model free, and similar to the procedures presented in section 3.3.2.

3.3.4 Predictive Encoder Design

Predictive encoder means the future estimated information can be utilized in the encoder design. Different from previous two strategies, system model is needed to estimate the system state in the future, making the design model-based. The encoder policy can be described as

$$i_t^* = \arg\min_{i \in \mathscr{I}} \mathbb{E}\{\sum_{\tau=t}^{T-1} (x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau) | y_t, i_t = i\} \tag{3.10}$$

According to the design objective equation (3.10), the reward involves future information, and thus can be defined as $\sum_{\tau=t}^{T-1} (x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)$. Obviously, this is an off-line learning process. The Q-value updating equation is same with memoryless encoder design, named equation (3.6). Because of the future estimation, the encoder should be updated with fixed controller

and fixed encoder $f_0, f_1, \cdots, f_{t-1}, f_{t+1}, f_{t+2}, \cdots, f_{T-1}$. While all the other design procedure is similar to the procedures presented in section 3.3.2.

3.3.5 Theoretical Aanlysis

In this section, we presents the theoretical analysis about encoder design. The following proof is based on the system assumption and the optimal design criterion.

Theorem 1. Given the system (3.1) and the memoryless channel (3.3). To minimize the LQ cost (3.5),

1. With fixed controller, the encoder mapping f_t of memoryless encoder can be obtained by equation (3.8);
2. With fixed controller, the encoder mapping f_t of memory encoder can be obtained by equation (3.9);
3. With fixed controller and fixed encoder components $f_0, f_1, \cdots, f_{t-1}, f_{t+1}, f_{t+2}, \cdots, f_{T-1}$, the encoder mapping f_t of predictive encoder can be obtained by equation (3.10);

Proof. 1) The encoder mapping f_t has an influence on the cost function (3.5) by producing i_t with y_t. More specifically, the system states and control commands dependent on i_t. Based on the principle of dynamic programming, the overall cost is minimized with the minimization immediate cost at each time step t.

Let $S_i(y_t)$ denotes all corresponding y_t of set Y_t such that $i_t = i$, named encoder regions. Hence, the optimal encoder mapping is equivalent to specifying the set of $S_i(y_t)$ such that $\mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i\}$ can be minimized over $i \in \mathscr{I}$ with given y_t. The encoder regions of memoryless encode can be given as

$$\begin{aligned} S_i(y_t) &= \{y \in Y_t | \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y, i_t = i^*\} = \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i^*\}\} \\ &= \{y \in Y_t | \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y, i_t = i^*\} \leq \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i\}\} \end{aligned} \tag{3.11}$$

where $\forall i \in \mathscr{I}$.

Let $F = [f_0, f_1, \cdots, f_{T-1}]$. Based on system performance equation (3.5),

$$\begin{aligned} \arg\min_F \mathbb{E}\{\sum_{t=0}^{T-1} (x_{t+1}^T D x_{t+1} + u_t^T E u_t)\} &= \arg\min_F \sum_{t=0}^{T-1} \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)\} \\ &= \arg\min_F \sum_{t=0}^{T-1} \mathbb{E}\{\mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t\}\} \\ &= \sum_{t=0}^{T-1} \mathbb{E}\{\arg\min_{i \in \mathscr{I}} \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i\}\} \end{aligned} \tag{3.12}$$

Based on equation (3.11) and (3.12), the encoder mapping can be given as

$$i_t^* = \arg\min_{i\in\mathscr{I}} \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i\}$$

For memoryless encoder design, encoder regions $S_i(y_t)$ can also be divided by i_t. The proof about part 1) of Theorem (1) is completed.

2) Since i_t only affects the current time step decision or the future terms,

$$\mathbb{E}\{\sum_{\tau=0}^{t-1}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i_t^*\} = \mathbb{E}\{\sum_{\tau=0}^{t-1}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i\} \tag{3.13}$$

where $\forall i \in \mathscr{I}$.

Adding equation (3.13) and equation (3.11) together,

$$\begin{aligned}
&\mathbb{E}\{\sum_{\tau=0}^{t}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i_t^*\} \\
=&\mathbb{E}\{\sum_{\tau=0}^{t-1}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i_t^*\} + \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i_t^*\} \\
\leq&\mathbb{E}\{\sum_{\tau=0}^{t-1}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i\} + \mathbb{E}\{(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t = i\} \\
=&\mathbb{E}\{\sum_{\tau=0}^{t}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i\}
\end{aligned} \tag{3.14}$$

where $i \in \mathscr{I}$.

The encoder regions of memory encode can be given as

$$\begin{aligned}
S_i(y_t) =&\{y \in Y_t | \mathbb{E}\{\sum_{\tau=0}^{t}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y, i_t = i^*\} \\
=&\mathbb{E}\{\sum_{\tau=0}^{t}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i^*\}\}
\end{aligned} \tag{3.15}$$

$$i_t^* = \arg\min_{i\in\mathscr{I}} \mathbb{E}\{\sum_{\tau=0}^{t}(x_{\tau+1}^T D x_{\tau+1} + u_\tau^T E u_\tau)|y_t, i_t = i\}$$

The proof about part 2) of Theorem (1) is completed.

3) $\mathbb{E}\{J_{tot}\}$ can be written as $\mathbb{E}\{J_{tot}\} = \mathbb{E}\{\mathbb{E}\{J_{tot|y_t,i_t=i}\}\}$,

$$\mathbb{E}\{J_{tot}\} = \mathbb{E}\{\mathbb{E}\{\sum_{t=0}^{T-1}(x_{t+1}^T D x_{t+1} + u_t^T E u_t)|y_t, i_t\}\} \tag{3.16}$$

$$\begin{aligned}
&\mathbb{E}\{\sum_{t=0}^{T-1}(x_{t+1}^T Dx_{t+1}+u_t^T Eu_t)|y_t,i_t\}\\
=&\mathbb{E}\{\sum_{\tau=0}^{t-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y_t,i_t\}+\mathbb{E}\{(x_{t+1}^T Dx_{t+1}+u_t^T Eu_t)|y_t,i_t\}\\
&+\mathbb{E}\{\sum_{\tau=t+1}^{T-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y_t,i_t\}
\end{aligned} \tag{3.17}$$

Since $f_0, f_1, \cdots, f_{t-1}$ is given, subtracting equation (3.13) from equation (3.16) ,

$$\begin{aligned}
&\mathbb{E}\{\sum_{\tau=t+1}^{T-1}(x_\tau^T Dx_\tau+u_{\tau-1}^T Eu_{\tau-1})|y_t,i_t^*\}\\
=&\mathbb{E}\{(x_{t+1}^T Dx_{t+1}+u_t^T Eu_t)|y_t,i_t^*\}+\mathbb{E}\{\sum_{\tau=t+1}^{T-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y_t,i_t^*\}\\
\leq&\mathbb{E}\{(x_{t+1}^T Dx_{t+1}+u_t^T Eu_t)|y_t,i_t=i\}+\mathbb{E}\{\sum_{\tau=t+1}^{T-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y_t,i_t=i\}
\end{aligned} \tag{3.18}$$

where $\forall i \in \mathscr{I}$.

The encoder regions of predictive encode can be given as

$$S_i(y)=\{y|\mathbb{E}\{\sum_{\tau=t}^{T-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y,i_t^*\}\leq\mathbb{E}\{\sum_{\tau=t}^{T-1}(x_{\tau+1}^T Dx_{\tau+1}+u_\tau^T Eu_\tau)|y_t,i_t^*\}\} \tag{3.19}$$

The proof about part 3) of Theorem (1) is completed. □

In general, the encoder regions $S_i(y_t)$ represents intervals in scalar systems and regions in second order system and so on.

3.4 Controller Design by Q-Learning

This section presents the results on controller design using Q-learning. Generally, Hamilton-Jacobi-Bellman (HJB) equations, such as Riccati equation for linear systems, are essential to optimal control design. However, it is difficult to obtain a solution of HJB equation when there is information loss in the system. Based on the techniques presented in Lewis and Vrabie [2009], Lewis et al. [2012b], Bradtke et al. [1994], the controller in the proposed system can be optimized without solving the HJB equation.

Noting that the mathematical system model (3.1) is unknown. Q-learning method learns to the optimal control policy from real system data directly. It is important to point out that there is no requirement on initial stabilizing control gain.

3.4.1 Q-Learning for Optimal Control

Let the system state $x_t \in \mathscr{X}_t$ and control command $g_t(x_t)$, the system is stabilizable on $\mathscr{X}_t$. The design objective of optimal control is to find controller mapping set g_0^*, g_1^*, $\cdots$, g_{T-1}^*, such that the closed control system asymptotically stable. The linear quadratic cost-to-go function can be written as

$$V_{g_t}(x_t) = \sum_{\tau=t}^{T-1} (x_\tau^T D x_\tau + u_\tau^T E u_\tau) \tag{3.20}$$

Since the process of controller mapping g_0 need to use the information of x_0 and there is no need to optimize g_T, value function is reformulated by x_t and u_t.

The optimal control policy g_t^* can be defined as

$$g_t^*(x_t) \triangleq \arg\min_{g_t} V_{g_t}(x_t) \tag{3.21}$$

Note that the optimal control policy is different with different time t.

Definition 3.4.1 (Admissible Control). For a given system (3.1) with $x_t \in \mathscr{X}_t$, a control policy $g_t(x_t)$ is said to be admissible with respect to cost function (3.5) on $\mathscr{X}_t$, denote to $g_t(x_t) \in \mathscr{U}(\mathscr{X}_t)$, if u_t is continuous on $\mathscr{X}_t$ and $V_{g_t}(x_t) < \infty$, $\forall x_t \in \mathscr{X}_t$.

For any admissible control policy $g_t(x_t) \in \mathscr{U}(\mathscr{X}_t)$, its cost-to-go function is

$$\begin{aligned} V_{g_t}(x_t) &= \sum_{t}^{T-1} (x_t^T D x_t + u_t^T E u_t) \\ &= x_t^T D x_t + u_t^T E u_t + \sum_{\tau=t+1}^{T-1} (x_\tau^T D x_\tau + u_\tau^T E u_\tau) \\ &= R(x_t, u_t) + V_{g_t}(x_{t+1}) \end{aligned} \tag{3.22}$$

where $R(x_t, u_t) = x_t^T D x_t + u_t^T E u_t$ represents instant reward. After that, it is important to define the Hamilton function of system (3.1) as

$$H(x_t, h_t, V) = V(x_{t+1}) - V(x_t) + R(x_t, h_t) \tag{3.23}$$

where h_t represents control command. Based on equation (3.22) and (3.23), the Hamilton function of the admissible control command $g_t(x_t) \in \mathscr{U}(\mathscr{X}_t)$ and its cost function V_{g_t} is

$$\begin{aligned} H(x_t, u_t, V) &= V_{g_t}(x_{t+1}) - V_{g_t}(x_t) + R(x_t, u_t) \\ &= 0 \end{aligned} \tag{3.24}$$

Let $V_t^* \triangleq V_{g_t}^*(x_t)$ be the optimal cost function, obviously $H(x_t, u_t, V_t^*) = 0$. Hence, the controller mapping (3.21) can be rewritten as

$$g_t^*(x_t) \triangleq \arg\min_{g_t} H(x_t, u_t, V_t^*) \tag{3.25}$$

In general, the optimal control policy g_t^* is obtained by solving HJB equations with V_t^*. While the system model is necessary to solve HJB equations and it is difficult or impossible to get HJB solution for nonlinear system. Hence, it is implementable in practical. To get the optimal control policy g_t of the proposed system, a model-free learning method, named Q-learning method, is introduced. It is important to introduce Q-function of Q-learning method, which is an action-state value function. For $\forall g_t(x_t) \in \mathscr{U}(\mathscr{X}_t)$, Q-function is

$$\begin{aligned} Q_{g_t}(x_t,\mu) &\triangleq R(x_t,\mu) + \sum_{t+1}^{T-1} R(x_t,u) \\ &= R(x_t,\mu) + Q_{g_t}(x_{t+1},u) \end{aligned} \tag{3.26}$$

where $(x_t,\mu) \in \mathscr{X}_t \times \mathscr{U}(\mathscr{X}_t)$. In addition, $Q_{g_t}(x_t,g_t(x_t)) = V_{g_t}(x_t)$. Hence, equation (3.26) can be rewritten as

$$Q_{g_t}(x_t,\mu) = R(x_t,\mu) + V_{g_t}(x_{t+1}) \tag{3.27}$$

Based on the optimal controller mapping (3.21) and (3.25), the optimal controller mapping can be rewritten as

$$g_t^*(x_t) = \arg\min_{g_t} V_{g_t}(x_t) = \arg\min_{g_t} Q_{g_t}(x_t,g_t(x_t)) \tag{3.28}$$

In general Q-learning algorithm,the Q-matrix starts from zero matrix. The policy is updated by policy evaluation and policy improvement. For policy evaluation, the updating equation is

$$Q^{\eta+1}(x_t,\mu) = R(x_t,\mu) + Q^{\eta}(x_{t+1},u^{\eta}) \tag{3.29}$$

where $Q^{\eta} = Q_{g_t{}^{\eta}}$.

For policy improvement component, g_t can be solved by

$$g_t^{\eta+1}(x_t) = \arg\min_{\mu} Q^{\eta+1}(x_t,\mu) \tag{3.30}$$

Let $\eta = \eta + 1$ and iterate these two learning processes.

3.4.2 Theoretical Analysis

The convergence is proved by demonstrating that the sequence $\{Q_t^{\eta}\}$ and $\{g_t^{\eta}\}$ generated by policy evaluation and policy improvement can converge to the optimal Q-value Q_t^* and the optimal control policy g_t^*, respectively.

Theorem 2. Let $g_t^{\eta+1}$ be given by policy improvement in algorithm, then

$$g_t^{\eta+1} = \arg\min_{\mu} H(x_t,\mu,V^{\eta}) \tag{3.31}$$

Proof. Based on equation (3.27) and (3.29),

$$Q^{\eta}(x_t,\mu) = R(x_t,\mu) + V^{\eta}(x_{t+1}) \tag{3.32}$$

where $V^{\eta}(x_{t+1}) = Q^{\eta}_{g_t^{\eta}}(x_{t+1}, g_t^{\eta}(x_{t+1}))$. Subtracting $V^{\eta}(x_t)$ on both side of equation (3.32),

$$\begin{aligned} Q^{\eta}(x_t,\mu) - V^{\eta}(x_t) &= R(x_t,\mu) + V^{\eta}(x_{t+1}) - V^{\eta}(x_t) \\ &= H(x_t,\mu,V^{\eta}) \end{aligned}$$

Let μ represents $g_t^{\eta+1}$,

$$\begin{aligned} \min_{\mu}(Q^{\eta}(x_t,\mu) - V^{\eta}(x_t)) &= \min_{\mu}(H(x_t,\mu,V^{\eta})) \\ \min_{\mu}(Q^{\eta}(x_t,\mu)) - V^{\eta}(x_t) &= \min_{\mu}(H(x_t,\mu,V^{\eta})) \\ \arg\min_{\mu}(Q^{\eta}(x_t,\mu)) &= \arg\min_{\mu}(H(x_t,\mu,V^{\eta})) \end{aligned}$$

Based on policy improvement (3.30) in algorithm,

$$g_t^{\eta+1} = \arg\min_{\mu}(Q^{\eta}(x_t,\mu)) = \arg\min_{\mu}(H(x_t,\mu,V^{\eta})) \tag{3.33}$$

This proof is completed. □

Theorem 2 shows the control policy $g_t^{\eta+1}$ in equation (3.30) is generated not only by minimizing Q-value function but also by minimizing the Hamilton equation.

Theorem 3. Let $g_t^0(x_t) \in \mathscr{U}(\mathscr{X})$. The sequence $\{g_t^{\eta}(x_t)\}$ is generated by policy improvement (3.30), then $g_t^{\eta}(x_t) \in \mathscr{U}(\mathscr{X})$ for $\forall \eta = 1,2,3,\cdots$.

Proof. Firstly, Theorem (3) holds for $\eta = 0$, $g_t^0(x_t) \in \mathscr{U}(\mathscr{X})$. Then, assume that Theorem (3) holds for $\eta = a$, that is $g_t^a(x_t) \in \mathscr{U}(\mathscr{X})$. The Hamilton equation of the cost function $V^a(x_t)$ associated with $g_t^a(x_t)$ is

$$H(x_t, u_t^a, V^a) = 0 \tag{3.34}$$

After that, we should prove the Theorem (3) holds for $\eta = a+1$.

Selecting the value function $V^a(x_t)$ as the Lyapunov function and the difference of $V^a(x_t)$ is

$$\begin{aligned} \nabla V &= V^a(x_{t+1}) - V^a(x_t) \\ &= V^a(x_{t+1}) - V^a(x_t) + R(x_t,u^{a+1}) - R(x_t,u^{a+1}) \\ &= H(x_t,u^{a+1},V^a) - R(x_t,u^{a+1}) \end{aligned} \tag{3.35}$$

Based on Theorem (2), u^{a+1} is admissible controller,

$$\begin{aligned} H(x_t,u^{a+1},V^a) &= \min_{\mu} H(x_t,\mu,V^a) \\ &\leq H(x_t,\mu,V^a) \\ &= 0 \end{aligned} \tag{3.36}$$

Combined with equation (3.35) and (3.36),

$$\nabla V \leq -R(x_t, u^{a+1}) \leq 0$$

It shows that the closed loop system with $x_{t+1} = Ax_t + Bu^{a+1} + v_t$ is asymptotically stable. Hence, $g_t^{a+1} \in \mathscr{U}(\mathscr{X})$ is admissible controller. In addition, Theorem (3) holds for $\eta = a+1$. This proof is completed. □

Theorem 4. $\forall (x_t, u_t) \in \mathscr{X}_t * \mathscr{U}(\mathscr{X}_t)$, the sequence $\{Q^\eta(x_t, \mu)\}$ and $\{g_t^\eta(x_t)\}$ are generated by policy evaluation (3.29) and policy improvement (3.30).

1. $Q^\eta(x_t, \mu) \geq Q^{\eta+1}(x_t, \mu) \geq Q^*(x_t, \mu)$.
2. $Q^\eta(x_t, \mu) \to Q^*(x_t, \mu)$ and $g_t^\eta(x_t) \to g_t^*(x_t)$, $\eta \to \infty$.

Proof. 1) Assume that $Q^0(x_t, \mu)$ is a Q-value function of a stable control policy $h(x_t)$, then

$$\begin{aligned} Q^0(x_t, \mu) &= Q_h(x_t, \mu) \\ &= R(x_t, \mu) + Q^0(x_{t+1}, h(x_{t+1})) \end{aligned} \tag{3.37}$$

Based on policy evaluation (3.29) and equation (3.37),

$$\begin{aligned} Q^1(x_t, \mu) &= R(x_t, \mu) + Q^0(x_{t+1}, h(x_{t+1})) \\ &= R(x_t, \mu) + \min_\mu Q^0(x_{t+1}, \mu) \\ &\leq R(x_t, \mu) + Q^0(x_{t+1}, \mu) \\ &= Q^0(x_t, \mu) \end{aligned} \tag{3.38}$$

Assume that $Q^\eta(x_t, \mu) \geq Q^{\eta+1}(x_t, \mu)$ and it holds for $\eta = 0$. Furthermore, assume that $Q^\eta(x_t, \mu) \geq Q^{\eta+1}(x_t, \mu)$ holds for $\eta = a-1$, so that $Q^{a-1}(x_t, \mu) \geq Q^a(x_t, \mu)$.

According to the policy improvement (3.28),

$$Q^a(x_t, u^a) = \min_\mu Q^a(x_t, \mu) \leq Q^a(x_t, \mu) \tag{3.39}$$

For $\forall x_t, \mu$,

$$\begin{aligned} Q^{a+1}(x_t, \mu) &= R(x_t, \mu) + Q^a(x_{t+1}, u^{a+1}) \\ &= R(x_t, \mu) + \min_\mu Q^a(x_{t+1}, \mu) \\ &\leq R(x_t, \mu) + Q^a(x_{t+1}, u^a) \\ &\leq R(x_t, \mu) + Q^{a-1}(x_{t+1}, u^a) \\ &= Q^a(x_t, \mu) \end{aligned} \tag{3.40}$$

Hence, $Q^{\eta}(x_t,\mu) \geq Q^{\eta+1}(x_t,\mu)$ holds for $\eta = a$. After that $Q^{\eta}(x_t,\mu) \geq Q^{*}(x_t,\mu)$ should be proved. Since $Q^{\eta-1}(x_t,\mu) \geq Q^{\eta}(x_t,\mu)$, $\forall (x_t,\mu) \in \mathscr{X} * \mathscr{U}$,

$$Q^{\eta-1}(x_{t+1},u^{\eta}) \geq Q^{\eta}(x_t,u^{\eta}) \tag{3.41}$$

Based on equation (3.41) and policy evaluation (3.29),

$$\begin{aligned} Q^{\eta}(x_t,\mu) &= R(x_t,\mu) + Q^{\eta-1}(x_{t+1},u^{\eta}) \\ &\geq R(x_t,\mu) + Q^{\eta}(x_{t+1},u^{\eta}) \\ &= R(x_t,\mu) + V^{\eta}(x_{t+1}) \\ &\geq R(x_t,\mu) + V^{*}(x_{t+1}) \\ &= Q^{*}(x_t,\mu) \end{aligned} \tag{3.42}$$

Hence, $Q^{\eta}(x_t,\mu) \geq Q^{\eta+1}(x_t,\mu) \geq Q^{*}(x_t,\mu)$. The proof about part 1) of Theorem (4) is completed.

2) According to part 1) of Theorem (4), $\{Q^{\eta}(x_t,\mu)\}$ is a non-increasing sequence and bounded with $Q^{*}(x_t,\mu)$. For a non-increasing sequence, there exists a limitation, which can be presented as $Q^{\infty}(x_t,\mu) \triangleq \lim_{\eta\to\infty} Q^{\eta}(x_t,\mu)$. The corresponding controller is $g_t^{\infty}(x_t) \triangleq \lim_{\eta\to\infty} Q^{\infty}(x_t,\mu)$. Moreover, $\{V^{\eta}(x_t)\}$ is also a non-increasing sequence and the boundary is $V^{*}(x_t)$. Let $V^{\infty}(x_t) \triangleq \lim_{\eta\to\infty} V^{\eta}(x_t)$.

Based on Theorem (2),

$$\begin{aligned} g_t^{\infty}(x_t) &= \arg\min_{\mu} Q^{\infty}(x_t,\mu) \\ &= \arg\min_{\mu} H(x_t,\mu,V^{\infty}) \end{aligned} \tag{3.43}$$

Because $g_t^{\infty}(x_t) \in \mathscr{U}(\mathscr{X})$, $H(x_t,\mu,V^{\infty}) = 0$ (based on equation (3.24)). It clearly shows that the value function V^{∞} satisfies the HJB equations (3.23). Because of the unique solution of HJB equations, $V^{\infty} = V^{*}$. Based on policy evaluation (3.29),

$$\begin{aligned} Q^{\infty}(x_t,\mu) &\triangleq \lim_{\eta\to\infty} Q^{\eta}(x_t,\mu) \\ &= \lim_{\eta\to\infty} R(x_t,\mu) + \lim_{\eta\to\infty} Q^{\eta-1}(x_{t+1},u^{\eta}) \\ &= R(x_t,\mu) + Q^{\infty}(x_t,u^{\infty}) \\ &= R(x_t,\mu) + V^{\infty}(x_{t+1}) \\ &= R(x_t,\mu) + V^{*}(x_{t+1}) \\ &= Q^{*}(x_t,\mu) \end{aligned} \tag{3.44}$$

Obviously, substitution of (3.44) into (3.43),

$$g_t^{\infty}(x_t) = \arg\min_{\mu} Q^{*}(x_t,\mu) = \arg\min_{\mu} H(x_t,\mu,V^{*}) = g_t^{*}(x_t)$$

The proof of part 2) is completed. □

Theorem (4) proves the convergence of proposed Q-learning algorithm. The Q-value function is a non-increasing sequence and has the boundary Q^*. Similarity, the control policy can also converge to g_t^*.

The following is how to fix the optimal control g_t^*. In general finite time horizon system, the optimal control policies can be obtained by dynamic programming with the backward induction. While it is impractical in application. Based on the optimality principle, the optimal control g_t^* can be expressed by Theorem 5

Theorem 5. With the fixed encoder, linear plant (3.1), memoryless noise channel (3.3), the controller mapping (3.4), which can minimize the system performance (3.5), can be obtained by the following recursive equation:

$$\begin{aligned} u_{t-1}^* &= \arg\min_{u_{t-1}} V_t \\ V_t &= \mathbb{E}\{(x_t^T D x_t + u_{t-1}^T E u_{t-1}) | j_{t-1}\} + \mathbb{E}\{V_{t+1}^* | j_{t-1}\} \end{aligned} \tag{3.45}$$

where $t = 1, 2, \cdots, T$. V_t is the cost-to-go that summarised the instantly cost from t to T, and it can be initialized as $V_{T+1} = 0$.

Proof. Based on equation (3.21), the optimal control policy is that minimizes the sum of future cost can be formulated as

$$u_t^* = \arg\min_{u_{t-1}} \mathbb{E}\{\sum_{\tau=t+1}^{T} (x_\tau^T D x_\tau + u_{\tau-1}^T E u_{\tau-1}) | j_t\} \tag{3.46}$$

let's start from the terminal time step $t = T$, u_{T-1}^* can be formulated as

$$u_{T-1}^* = \arg\min_{u_{T-1}} \mathbb{E}\{(x_T^T D x_T + u_{T-1}^T E u_{T-1}) | j_{T-1}\} \tag{3.47}$$

u_{T-1}^* can be rewritten as

$$\begin{aligned} u_{T-1}^* &= \arg\min_{u_{T-1}} V_T \\ V_T &= \mathbb{E}\{(x_T^T D x_T + u_{T-1}^T E u_{T-1}) | j_{T-1}\} + \mathbb{E}\{V_{T+1}^* | j_{T-1}\} \end{aligned} \tag{3.48}$$

Since the cost-to-go $V_{T+1} = 0$,

$$\begin{aligned} V_T =& \mathbb{E}\{(x_T^T D x_T + u_{T-1}^T E u_{T-1}) | j_{T-1}\} \\ =& \mathbb{E}\{(A x_{T-1} + B u_{T-1} + v_{T-1})^T D (A x_{T-1} + B u_{T-1} + v_{T-1}) + u_{T-1}^T E u_{T-1} | J_{T-1}\} \end{aligned} \tag{3.49}$$

The process noise v_{T-1} follows Gaussian distribution with mean 0. Hence, $\mathbb{E}\{v_{T-1} | j_{T-1}\} = 0$. In addition, the co-variance of process noise can be represented as $\Psi_{T-1} = Tr(v_{T-1} v_{T-1}^T)$, which is straightforward to expand to $\Psi_t = Tr(v_t v_t^T)$. Obviously, it is independent with other parameters of the system.

The cost-to-go V_T can be rewritten as

$$\begin{aligned} V_T =& \mathbb{E}\{x_{T-1}^T A^T DA x_{T-1} + 2u_{T_1}^T B^T DA x_{T-1} + u_{T-1}^T (E + B^T DB) u_{T-1} + \mathrm{Tr}(D\Psi_{T-1}) | j_{T-1}\} \\ =& \mathbb{E}\{x_{T-1}^T A^T DA x_{T-1} + 2u_{T_1}^T B^T DA x_{T-1} + u_{T-1}^T (E + B^T DB) u_{T-1} | j_{T-1}\} + \mathrm{Tr}(D\Psi_{T-1}) \end{aligned} \tag{3.50}$$

Applied $\frac{\partial V_T}{\partial u_{T-1}} = 0$, u_{T-1}^* is

$$\begin{aligned} u_{T-1}^* &= -(B^T DB + E)^{-1} B^T DA \mathbb{E}\{x_{T-1} | j_{T-1}\} \\ &= -(B^T DB + E)^{-1} B^T DA \hat{x}_{T-1} \end{aligned} \tag{3.51}$$

where $\hat{x}_{T-1} = \mathbb{E}\{x_{T-1} | j_{T-1}\}$ is the estimation of x_{T-1}. Then, substituting u_{T-1}^* to V_T, the optimal cost-to-go V_T^* is

$$\begin{aligned} V_T^* =& \mathrm{Tr}(D\Psi_{T-1}) + \mathbb{E}\{x_{T-1}^T A^T DA x_{T-1} - 2\hat{x}_{T-1}^T A^T DB (B^T DB + E)^{-1} B^T DA x_{T-1} \\ &+ \hat{x}_{T-1}^T A^T DB (B^T DB + E)^{-1} B^T DA \hat{x}_{T-1} | j_{T-1}\} \\ =& \mathrm{Tr}(D\Psi_{T-1}) + \mathbb{E}\{x_{T-1}^T (A^T DA - A^T DB (B^T DB + E)^{-1} B^T DA) x_{T-1} | j_{T-1}\} \\ &+ \mathbb{E}\{\tilde{x}_{T-1}^T A^T DB (B^T DB + E)^{-1} B^T DA \tilde{x}_{T-1} | j_{T-1}\} \end{aligned} \tag{3.52}$$

where $\tilde{x}_{T-1} = x_{T-1} - \hat{x}_{T-1} = x_{T-1} - \mathbb{E}\{x_t | j_t\}$.

Note that $(B^T DB + E)^{-1}$ can be calculated by pseudo-inverse. The optimal cost-to-go V_T can be rewritten as

$$\begin{aligned} V_T^* &= \mathbb{E}\{x_{T-1}^T I_1 x_{T-1} + \varpi_1 | j_{T-1}\} \\ I_1 &\triangleq A^T DA - \pi_1 \\ \pi_1 &\triangleq A^T DB (E + B^T DB)^{-1} B^T DA \\ \varpi_1 &\triangleq \mathrm{Tr}(D\Psi_{T-1}) + \mathbb{E}\{\tilde{x}_{T-1}^T \pi_1 \tilde{x}_{T-1} | j_{T-1}\} \end{aligned} \tag{3.53}$$

Similarly, we can obtain the optimal control u_{T-2}^* at time $t = T-1$ is

$$\begin{aligned} u_{T-2}^* &= \arg\min_{u_{T-2}} V_{T-1} \\ &= \arg\min_{u_{T-2}} \mathbb{E}\{x_{T-1}^T D x_{T-1} + u_{T-2}^T E u_{T-2} | j_{T-2}\} + \mathbb{E}\{V_T^* | j_{T-2}\} \end{aligned} \tag{3.54}$$

Hence, it can be expended to general case that u_{t-1}^* is

$$\begin{aligned} u_{t-1}^* &= \arg\min_{u_{t-1}} V_t \\ &= \arg\min_{u_{t-1}} \mathbb{E}\{x_t^T D x_t + u_{t-1}^T E u_{t-1} | j_{t-1}\} + \mathbb{E}\{V_{t+1}^* | j_{t-1}\} \end{aligned} \tag{3.55}$$

Note that the cost-to-go at $t = 0$ is $V_1 = \mathbb{E}\{J_{tot}\}$. This proof is completed. □

It is worth to point out that the optimal control in Theorem (5) can't be solved efficiently, because of huge computation. To given an efficiently analysis, a virtual help-system is built, which is

named open-loop encoder system (Tatikonda et al. [2004], Nair et al. [2007], Aoki [1967], Lei et al. [2011]).

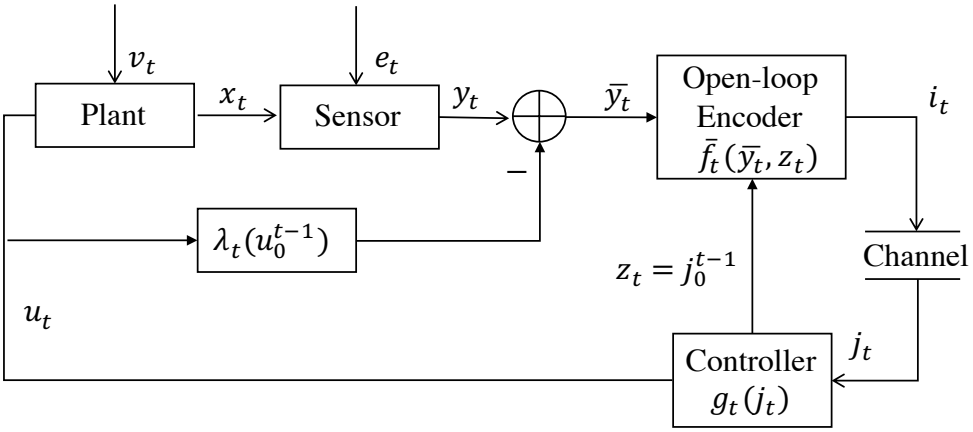

FIGURE 3.5: Closed-loop control system with open-loop encoder and side information from controller to encoder.

In open-loop encoder system, the input of the encoder is part of the measurement value y_t, where all effects of control commands are removed. Base on Fig 3.5,

$$\bar{y}_t = y_t - \lambda_t(u_0^{t-1}) \tag{3.56}$$

where u_0^{t-1} is the previous control command set, such as $u_0^{t-1} = [u_0, u_1, \cdots, u_{t-1}]$. $\lambda_t(u_0^{t-1})$ is the information of controller, which can be defined as

$$\begin{aligned}
\bar{y}_t &= y_t - \sum_{\tau=0}^{t-1} CA^{t-1-\tau} Bu_\tau \\
&= Cx_t + e_t - \sum_{\tau=0}^{t-1} CA^{t-1-\tau} Bu_\tau \\
&= C(Ax_{t-1} + Bu_{t-1} + v_{t-1}) + e_t - \sum_{\tau=0}^{t-1} CA^{t-1-\tau} Bu_\tau \\
&= CA^t x_0 + \sum_{\tau=0}^{t-1} CA^{t-1-\tau} v_\tau + e_t
\end{aligned} \tag{3.57}$$

It is clearly shown that $\bar{y}_t$ is only affected by x_0, $[v_0, v_1, \cdots, v_{t-1}]$, e_t, which is independent with control commands. In addition, side information from controller to encoder is given, where $z_t = j_0^{t-1} = [j_0, j_1, \cdots, j_{t-1}]$. The open-loop encoder mapping $\bar{f}_t$ is

$$i_t = \bar{f}_t(\bar{y}_t, z_t) \tag{3.58}$$

where $i_0 = \bar{f}_0(\bar{y}_0)$.

In the open-loop encoder system, the open-loop encoder mapping $\bar{f}_t$ and same controller mapping $u_t = g_t(j_t)$ with system in Fig. 3.1, $t = 1, 2, \cdots, T-1$, can be determined with same system

performance in Fig. 3.1. With same controller mappings and same design purpose, the open-loop encoder mapping $\bar{f}_t(\bar{y}_t, z_t)$ is same with the encoder mapping $f_t(y_t)$ in Fig. 3.1, which can be formulated as

$$f_t(y_t) = \bar{f}_t(\bar{y}_t, z_t) \tag{3.59}$$

The formula is established because the property of iterative design. With fixed controller, the information constructed by $\bar{y}_t, z_t$ equals to y_t. Hence, the open-loop encoder system is very helpful to analysis the proposed system optimization in this Chapter.

Theorem 6. With the optimized solution of the proposed system $\{f_t^*(y_t), g_t^*\}$ from time $t = 0$ to $T-1$, the open-loop encoder system can also be specified by the proposed optimization solution, named $\bar{f}_t(\bar{y}_t, z_t) = f_t^*(y_t)$.

Proof. Based on proofs by contradiction, we assume that $\{f_t^*(y_t), g_t^*\}, t = 0, 1, \cdots, T-1$ is not a solution of open-loop encoder system. Hence, there are other encoder controller mappings that can minimize the system performance. That is to say $\exists\{\bar{f}_t(\bar{y}_t, z_t), g_t^*\}, t = 0, 1, \cdots, T-1$,

$$\min_{\{\bar{f}_t(\bar{y}_t, z_t), g_t^*\}} \mathbb{E}\{J_{tot}\} \leq \min_{\{f_t^*(y_t), g_t^*\}} \mathbb{E}\{J_{tot}\} \tag{3.60}$$

However, it is given that $\{f_t^*(y_t), g_t^*\}$ from time $t = 0$ to $T-1$ is the optimal solutions, and the minimal system cost $\mathbb{E}\{J_{tot}\}$ can be obtained. Since the proposed system have same system performance criteria $\mathbb{E}\{J_{tot}\}$, the assumption is false. Obviously,

$$\bar{f}_t(\bar{y}_t, z_t) = f_t^*(y_t) \tag{3.61}$$

The proof is completed. □

Similarly, if $\{\bar{f}_t^*(\bar{y}_t, z_t), g_t^*(j_t)\}, t = 0, 1, \cdots, T-1$ is the optimal solution of open-loop encoder system, the optimal solution of proposed system is the same controller $g_t^*(j_t), t = 0, 1, \cdots, T-1$ and $f_t^*(y_t), t = 0, 1, \cdots, T-1$. Based on Theorem 6, $\bar{f}_t^*(\bar{y}_t, z_t) = f_t(y_t)$.

Theorem 6 indicates that the solution of proposed system can be specified by the open-loop encoder system and vice versa. It is relatively easy to analysis the open-loop encoder system. Furthermore, the output of the encoder i_t, with fixed controller, is independent with the control commands.

$$i_t = \bar{f}_t(\bar{y}_t, j_0, j_1, \cdots, j_{t-1})$$

where $\bar{y}_t$ is independent with u_t and j_t is only determined by i_t.

Theorem 7. The open-loop encoder system is considered in Fig. 3.5. Assuming the open-loop encoder $\bar{f}_t(\bar{y}_t, z_t), t = 0, 1, \cdots, T-1$ is fixed. Given the linear plant (3.1), the memoryless channel (3.3), the controller $u_t = g_t(j_t)$ that can minimize the system performance (3.5) is given by

$$u_t = k_t \hat{x}_t \tag{3.62}$$

where $\hat{x}_t = \mathbb{E}\{x_t|j_t\}$.

$$\begin{aligned} k_t &= -(E+B^T(D+I_{T-t-1})B)^{-1}B^T(D+I_{T-t-1})A \\ I_{T-t-1} &\triangleq A^T(D+I_{T-t-2})A-\pi_{T-t-1} \\ \pi_{T-t-1} &\triangleq A^T(D+I_{T-t-2})B(E+B^T(D+I_{T-t-2})B)^{-1}B^T(D+I_{T-t-2})A \end{aligned} \tag{3.63}$$

where the initial condition $I_1 = A^TDA - A^TDB(E+B^TDB)^{-1}B^TDA$.

The optimal cost-to-go V_{t+1} is

$$\begin{aligned} V_{t+1}^* &= \mathbb{E}\{x_t^T I_{T-t}x_t + \varpi_{T-t}|j_t\} \\ I_{T-t} &\triangleq A^T(D+I_{T-t-1})A-\pi_{T-t} \\ \pi_{T-t} &\triangleq A^T(D+I_{T-t-1})B(E+B^T(D+I_{T-t-1})B)^{-1}B^T(D+I_{T-t-1})A \\ \varpi_{T-t} &\triangleq \varpi_{T-t-1} + \mathrm{Tr}\{(D+I_{T-t-1})\Psi_t\} + \mathbb{E}\{\tilde{x}_t^T\pi_{T-t}\tilde{x}_t|j_t\} \end{aligned} \tag{3.64}$$

Proof. With the fixed open-loop encoder mappings $\bar{f}_t(\bar{y}_t, z_t)$, $t = 0,1,\cdots,T-1$, it can be deduced that the state error $\tilde{x}_t$ is independent with the control commands u_t, $t = 0,1,\cdots,T-1$, which can be derived by

$$\begin{aligned} \tilde{x}_t &= x_t - \hat{x}_t \\ &= A^t x_0 + \sum_{\tau=0}^{t-1} A^{t-1-\tau}v_\tau - \mathbb{E}\{A^t x_0 + \sum_{\tau=0}^{t-1} A^{t-1-\tau}v_\tau|j_t\} \end{aligned} \tag{3.65}$$

It is clearly shown that $\tilde{x}_t$ is only affected by $\{x_0, v_\tau, j_t\}$. In addition, j_t is determined by i_t, i_t can be obtained by $\bar{f}_t(\bar{y}_t, z_t)$, which has no relation with u_t. Hence, the conclusion is that the estimation error $\tilde{x}_t$ with fixed open-loop encoder mapping is independent with u_t. This process can considerably simplify the optimal controller derivation.

Based on the proof of Theorem 5, we have already given u_{T-1}. Obviously, $\tilde{x}_{T-1}$ is independent with u_{T-2}, which means $\frac{\partial \varpi_1}{\partial u_{T-2}} = 0$. Based on equation (3.53) and (3.54) in Theorem 5,

$$V_{T-1} = \mathbb{E}\{x_{T-1}^T D x_{T-1} + u_{T-2}^T E u_{T-2}|j_{T-2}\} + \mathbb{E}\{V_T^*|j_{T-2}\} \tag{3.66}$$

The optimal controller $u_{T-2}^* = \arg\min_{u_{T-2}} V_{T-1}$ can be obtained by $\frac{\partial V_{T-1}}{\partial u_{T-2}} = 0$,

$$\begin{aligned} u_{T-2}^* &= k_{T-2}\hat{x}_{T-2} \\ k_{T-2} &= -(E+B^T(D+I_1)B)^{-1}B^T(D+I_1)A \end{aligned} \tag{3.67}$$

Substituting the optimal control u_{T-2}^* to equation (3.66), V_{T-1}^* is

$$\begin{aligned} V_{T-1}^* &= \mathbb{E}\{x_{T-2}^T I_2 x_{T-2} + \varpi_2 | j_{T-2}\} \\ I_2 &\triangleq A^T(D+I_1)A - \pi_2 \\ \pi_2 &\triangleq A^T(D+I_1)B(E+B^T(D+I_1)B)^{-1}B^T(D+I_1)A \\ \varpi_2 &\triangleq \varpi_1 + \mathrm{Tr}\{(D+I_1)\Psi_{T-2}\} + \mathbb{E}\{\tilde{x}_{T-2}^T \pi_2 \tilde{x}_{T-2} | j_{T-2}\} \end{aligned} \tag{3.68}$$

where I_1, π_1, ϖ_1 has already been given in Theorem 5.

For general case, let's proof u_t^*. First of all, assuming u_{t+1}^* satisfied. Furthermore, u_{T-1}^* and u_{T-2}^* have already been proven. Based on Theorem 5,

$$u_t^* = \arg\min_{u_t} \mathbb{E}\{x_{t+1}^T D x_{T+1} + u_t^T E u_t + V_{t+2}^* | j_t\} \tag{3.69}$$

Based on equation (3.68), the optimal cost-to-go $\mathbb{E}\{V_{t+2}^* | j_{t+1}\}$ can be written as

$$\begin{aligned} V_{t+2}^* &= \mathbb{E}\{x_{t+1}^T I_{T-t-1} x_{t+1} + \varpi_{T-t-1} | j_{t+1}\} \\ I_{T-t-1} &\triangleq A^T(D+I_{T-t-2})A - \pi_{T-t-1} \\ \pi_{T-t-1} &\triangleq A^T(D+I_{T-t-2})B(E+B^T(D+I_{T-t-2})B)^{-1}B^T(D+I_{T-t-2})A \\ \varpi_{T-t-1} &\triangleq \varpi_{T-t-2} + \mathrm{Tr}\{(D+I_{T-t-2})\Psi_{t+1}\} + \mathbb{E}\{\tilde{x}_{t+1}^T \pi_{T-t-1} \tilde{x}_{t+1} | j_{t+1}\} \end{aligned} \tag{3.70}$$

It has been proven that the $\mathbb{E}\{\tilde{x}_{t+1}^T \pi_{T-t-1} \tilde{x}_{t+1} | j_{t+1}\}$ is not a function of u_t. Hence, u_t can be expressed as

$$u_t^* = \arg\min_{u_t} \mathbb{E}\{x_{t+1}^T D x_{t+1} + u_t^T E u_t + x_{t+1}^T I_{T-t-1} x_{t+1} | j_t\} \tag{3.71}$$

Based on the optimality principle, the optimal control u_t^* is

$$\begin{aligned} u_t &= k_t \hat{x}_t \\ k_t &= -(E+B^T(D+I_{T-t-1})B)^{-1}B^T(D+I_{T-t-1})A \end{aligned} \tag{3.72}$$

Substituting u_t^* to V_{t+1}, the optimal cost-to-go V_{t+1}^* is

$$\begin{aligned} V_{t+1}^* &= \mathbb{E}\{x_t^T I_{T-t} x_t + \varpi_{T-t} | j_t\} \\ I_{T-t} &\triangleq A^T(D+I_{T-t-1})A - \pi_{T-t} \\ \pi_{T-t} &\triangleq A^T(D+I_{T-t-1})B(E+B^T(D+I_{T-t-1})B)^{-1}B^T(D+I_{T-t-1})A \\ \varpi_{T-t} &\triangleq \varpi_{T-t-1} + \mathrm{Tr}\{(D+I_{T-t-1})\Psi_t\} + \mathbb{E}\{\tilde{x}_t^T \pi_{T-t} \tilde{x}_t | j_t\} \end{aligned} \tag{3.73}$$

The proof is completed. □

The optimal cost-to-go V_1 at $t=0$ equals to the system performance $\mathbb{E}\{J_{tot}\}$. Based on Theorem 7,

$$\begin{aligned}\mathbb{E}\{J_{tot}\} &= V_1 \\ &= x_0^T I_T x_0 + \sum_{t=0}^{T-1} \mathrm{Tr}\{(D+I_{T-t})\Psi_t\} + \sum_{t=0}^{T-1} \mathbb{E}\{\tilde{x}_t^T \pi_{T-t} \tilde{x}_t | j_t\}\end{aligned} \tag{3.74}$$

It is worth to point out that the solution of equation (3.74) can be calculated, which can be the criteria of the encoder and controller design by Q-learning. Furthermore, it is clearly from equation (3.73) that i_t only have an influence on the term $\mathbb{E}\{\tilde{x}_t^T \pi_{T-t} \tilde{x}_t | j_t\}$ in the cost-to-go V_{t+1}. Hence, the conclusion is that the open-loop encoder mapping $\bar{f}_t$ can be obtained by

$$i_t = \arg\min_i \mathbb{E}\{\tilde{x}_t^T \pi_{T-t} \tilde{x}_t | j_t\} \tag{3.75}$$

where $u_t = k_t \hat{x}_t$ is the fixed controller mapping.

3.4.3 Q-Learning Design Process

For linear quadratic regulator (LQR) problem of system (3.1) with cost function (3.5), the optimal Q-function can be given by

$$\begin{aligned}Q^*(\hat{x}_t, u) &= \begin{bmatrix}\hat{x}_t \\ u\end{bmatrix}^T G \begin{bmatrix}\hat{x}_t \\ u\end{bmatrix} \\ &= \begin{bmatrix}\hat{x}_t \\ u\end{bmatrix}^T \begin{bmatrix}G_{xx} & G_{xu} \\ G_{ux} & G_{uu}\end{bmatrix} \begin{bmatrix}\hat{x}_t \\ u\end{bmatrix}\end{aligned} \tag{3.76}$$

where Q^* represents the optimal value, $\hat{x}_t$ is the estimate of x_t obtained by decoding j_t to $\hat{x}_t$. $G \leq 0$ can be viewed as a kernel matrix.

Applying $\frac{\partial Q(\hat{x}_t, u)}{\partial u_t} = 0$ to equation (3.76) yields

$$u_t = -G_{uu}^{-1} G_{ux} \hat{x}_t \tag{3.77}$$

In standard LQR problem, the kernel matrix G can be calculated by system parameters. However, if the kernel matrix G can be estimated online without knowing system parameters, the optimal control policy can obtained straightforward. Specifically, Q-value function can be rewritten as

$$Q(\hat{x}_t, u_t) = W^T \phi(\hat{x}_t, u_t) \tag{3.78}$$

where W is a vector and it collects the upper triangle or lower triangle of symmetric matrix G. Hence, W has the $(n+m)*(n+m+1)/2$ elements with $\hat{x}_t \in \mathbb{R}^n$, $u \in \mathbb{R}^m$. ϕ is the dense basis set and composed by quadratic terms of $\hat{x}_t$ and u_t. Similarly, there are $(n+m)*(n+m+1)/2$

independent elements in the dense basis. The basis function matrix is

$$\begin{bmatrix} \hat{x}_1^2 & \hat{x}_1\hat{x}_2 & \cdots & \hat{x}_1u_1 & \cdots & \hat{x}_1u_m \\ \hat{x}_2\hat{x}_1 & \hat{x}_2^2 & \cdots & \hat{x}_2u_1 & \cdots & \hat{x}_2u_m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{x}_n\hat{x}_1 & \cdots & \hat{x}_n^2 & \hat{x}_nu_1 & \cdots & \hat{x}_nu_m \\ u_1\hat{x}_1 & u_1\hat{x}_2 & \cdots & u_1^2 & \cdots & u_1u_m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u_m\hat{x}_1 & u_m\hat{x}_2 & \cdots & \hat{x}_nu_m & \cdots & u_m^2 \end{bmatrix} \tag{3.79}$$

Letting the basis function is

$$\begin{aligned} \phi(\hat{x}_t,u_t) = [\hat{x}_1^2 \quad \hat{x}_1\hat{x}_2 \quad \cdots \quad \hat{x}_1u_1 \quad \cdots \quad \hat{x}_1u_m \\ \hat{x}_2^2 \quad \cdots \quad \hat{x}_2u_1 \quad \cdots \quad u_m^2]' \end{aligned} \tag{3.80}$$

and weight sequence W is

$$\begin{aligned} W_\eta = [G_{1,1}^\eta \quad G_{1,2}^\eta \quad G_{1,3}^\eta \quad \cdots \quad G_{1,n+m}^\eta \quad G_{2,2}^\eta \\ \cdots \quad G_{2,n+m}^\eta \quad \cdots \quad G_{n+m,n+m}^\eta]' \end{aligned} \tag{3.81}$$

where $G_{1,1}^\eta$ is the element $(1,1)$ of matrix G with on η^{th} iteration.

Based on policy evaluation (3.29) , the updating equation for W is

$$W_{(\eta+1,t)}^T\phi(\hat{x}_t,u_t,t) = r + W_{(\eta,t)}^T\phi(\hat{x}_{t+1},u_{t+1},t) \tag{3.82}$$

where $r = x_t^TD_tx_t + u_t^TE_tu_t$ is the reward, η is the iterative number; $W_{(\eta,t)}$ represents the weight vector in the η^{th} iteration. Note that $u_t = k_t * \hat{x}_t$ and $u_{t+1} = k_t * \hat{x}_{t+1}$.

Obviously, the updating equation (3.82) is a recursive equation, where $W_{(\eta+1,t)}$ is the only unknown parameter in the η^{th} iteration at time t. Recursive least squares (RLS) method can be used to obtain $W_{(\eta+1,t)}$, and then the Riccati matrix G can be derived from the weight sequence W. Once the Riccati matrix is available, the optimized control policy can be obtained simply.

The controller design process with memoryless, memory encoder and predictive encoder are same and is given by in Fig. 3.6. There are three loops in this design process, the dense basis vector is defined as a three dimensional matrix, namely, $\phi(\hat{x}_t,u_t,t)$. This loop is for data collecting with $t = t+1$. Dense basis vector can be calculated for all time step. The middle loop $l = l+1$, $W_{(\eta+1,t)}$ is updated by RLS method, which can be regarded as online learning process.

In finite time horizon case, the optimal control policy from $t = 0$ to $T-1$ is optimized separately and repeat to run the system. Hence, the weight sequence W updates independently at each time t with the iteration. Finally, the optimized control policy $k_0,k_1,\cdots,k_{T-1}$ can be got and each of

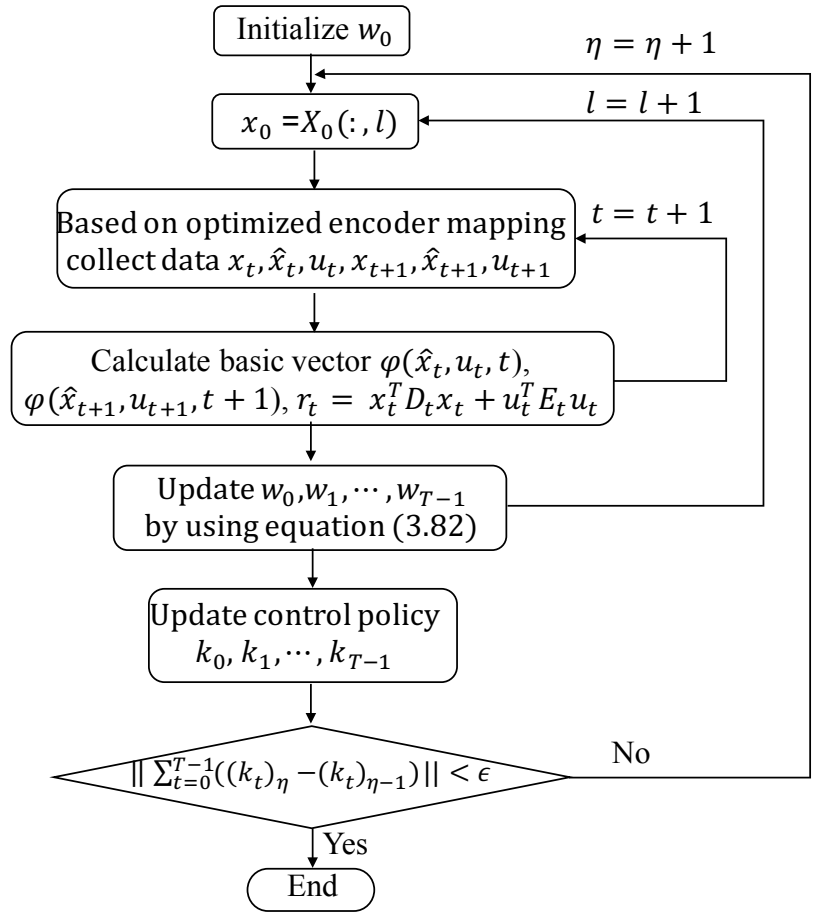

FIGURE 3.6: The controller design process by Q-learning method

them are close to the optimal controller calculated with open-loop encoder system (Theorem 7). Compared with Zhao et al. [2015, 2014] using time-dependent basis function, the design complexity of the proposed controller optimization decreases a lot. Because time-dependent factor is very difficult to determine, trail-error method is utilized in Zhao et al. [2015, 2014].

3.5 Iterative Encoder-Controller Design

Based on the results presented in Section 3.3 and Section 3.4, an iterative training algorithm is introduced in this section to obtain a pair of optimized encoder and controller that can minimize the expected overall cost. Both memoryless and memory encoder design are utilized online and model-free learning method to optimize, while the method used in predictive encoder design is offline and model-based.

These three encoder design methods are similar with the only different about reward calculation. The iterative design process is also similar and is given by Fig 3.7. First of all, all the encoder mappings f_0, f_1, $\cdots$, f_{T-1} are optimized with fixed controller mappings g_0, g_1, $\cdots$, g_{T-1}. Then, the controller mapping is calculated based on these optimized encoder mappings.

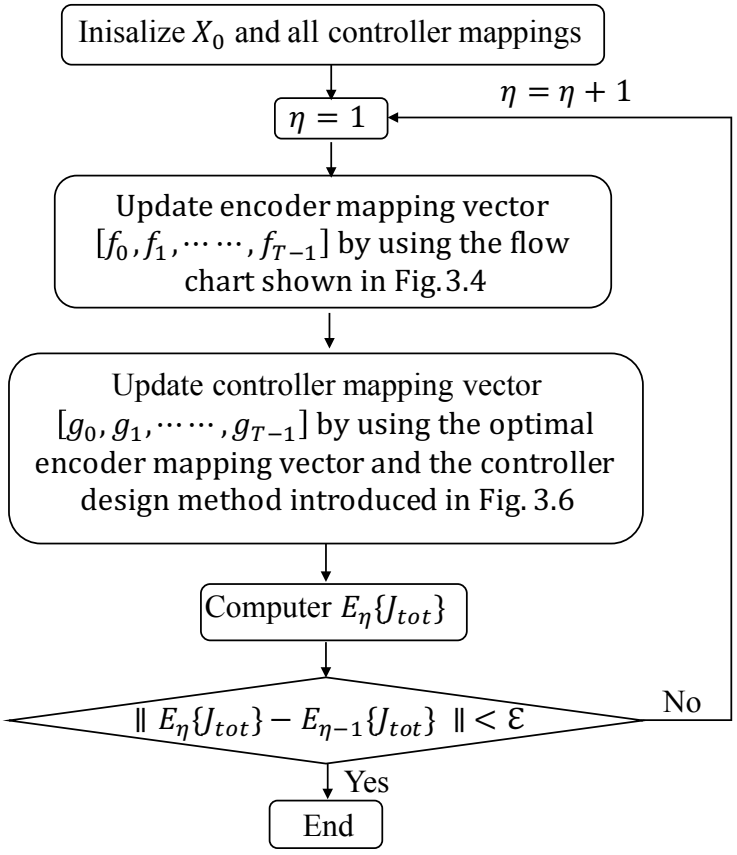

FIGURE 3.7: The flow chart of iterative encoder and controller design by Q-learning

Finally, the expected overall cost are calculated by the optimized encoder mappings and controller mappings. Note that the encoder mapping f_t of predictive encoder should be determined with future information, which mean the encoder mapping f_{t+1}, f_{t+2}, $\cdots$, f_{T-1} need to be fixed in advanced.

The expected overall cost is $\mathbb{E}_\eta\{J_{tot} = \sum_{t=0}^{T-1} x_{t+1}^T V_t x_{t+1} + u_t^T R_t u_t\}$, which can be calculated by Montecarlo method with the optimized encoder-controller mappings. The algorithm is terminated when the error between $\mathbb{E}_\eta\{J_{tot}\}$ and $\mathbb{E}_{\eta-1}\{J_{tot}\}$ is small enough, that is there is no improvement in the system performance.

3.6 Convergence Analysis

For open-loop encoder system, the iterative design is convergence, because the controller is always optimal with any open-loop encoder mapping $\bar{f}_t$. More specifically, the optimal controller mapping can be obtained in open-loop control system, the system performance should be improved or stable with an updated encoder. So that the design of open-loop encoder system

trends to converge. It has been proven that the open-loop encoder system with side-information is equivalent to the proposed system. Hence, the design by Q-learning can converge.

The optimal controller can converge to the optimal controller obtained in open-loop encoder system and the minimized system performance by Q-learning technique are very close to the minimal system cost calculated by open-loop encoder system. While it is the common convergence analysis about vector quantization or related problems based on iterative design, where the system is divided and one component is update, others are fixed Gersho and Gray [2012], Farvardin [1990], Sabin and Gray [1986]. However, only local optimal result can be reached. It is difficult to say global optimal Sabin and Gray [1986].

3.7 Numerical Results

The system parameters for simulation are $A = \begin{bmatrix} 0.99 & 0.3 \\ 0.8 & -0.7 \end{bmatrix}$, $B = I_2$ and $C = I_2$. The weighting matrices in J_T are $E = 0.1I_2$ and $D = 4I_2$. The initial state x_0, process noisee_t and measurement noise e_t follow Gaussian distribution with mean 0 and variance $\begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\begin{bmatrix} 1e-5 \\ 1e-5 \end{bmatrix}$, $\begin{bmatrix} 1e-5 \\ 1e-5 \end{bmatrix}$, respectively.

Because of low rate noisy channel, the transmission rate used for simulation is $\rho = 2$. The number of binary codeword is $2^\rho = 4$, which are $\begin{bmatrix} 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 \end{bmatrix}$. For second order system, there are $2^{\rho n} = 16$ encoder indexes and the corresponding binary codeword matrix b is

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Hence, there are 16 encoder regions $S_i(y_t)$ for all set of y_t and each region has its corresponding binary codeword which are the column of matrix b.

For all simulation, the sample size L is 3000 and the algorithm stop accuracy is set as $\varepsilon = 10^{-5}$. The size of $Q(y_t, i_t, t)$ is $L * (2^{\rho n}) * T$, named $3000 * 16 * T$, T is the finite time horizon.

3.7.1 Encoder Design

The optimized encoder mappings by memoryless encoder design, memory encoder design and predictive encoder design are similar. The analysis of memoryless encoder design is given and the other two are omitted. Since the encoder is specified by the encoder regions.

Fig. 3.8 shows the optimized encoder mappings from $t = 1$ to $t = 6$. Each region of each figure corresponds to one column of binary codeword matrix b, such as $b(:,1) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$. It

is clearly to show that the boundary of each region is similar from $t = 3$ to $t = 6$. Since y_t is two dimensional vector $y_t = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$, these two axises are y_1 and y_2. For horizon axis y_1 and vertical axis y_2 means two elements of y_t.

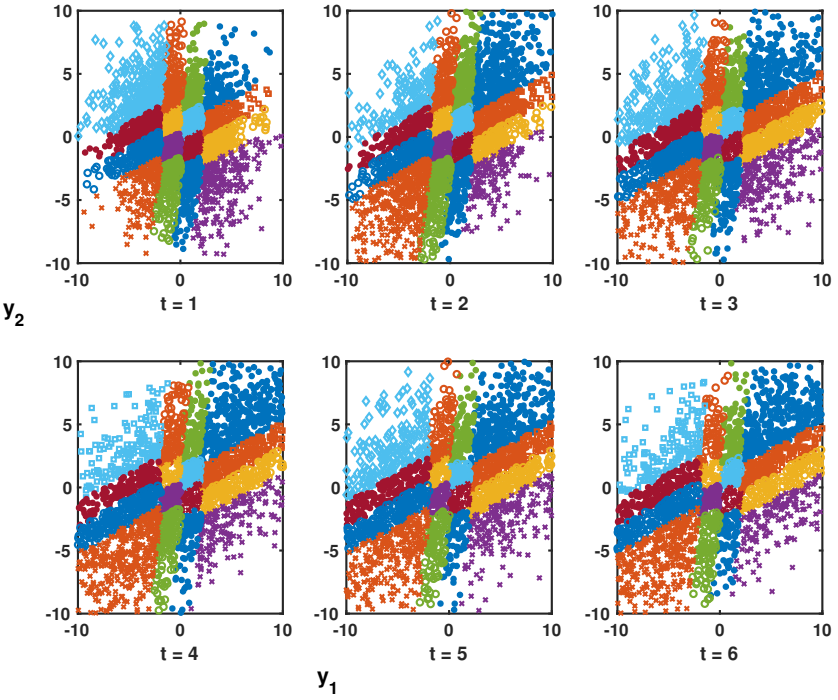

FIGURE 3.8: The optimized encoder regions with the crossover probability $p = 0.02$ over time interval from $t = 1$ to $t = 6$.

Fig. 3.9 shows the encoder regions of open-loop encoder system, which is the optimized encoder regions with the crossover probability $p = 0.02$ at $t = 1$. It is clearly shown that this encoder region bounds are quite similar with Fig. 3.8. In open-loop encoder system, the encoder mappings can be viewed as optimal result, since the controller mapping is already optimal. Hence, the achieved open-loop encoder mapping can be as a criterion for encoder optimization component by Q-learning.

Fig. 3.10 illustrates the evolution of Q-value with randomly choosing one state-action pair (y_t, i_t) from each encoder region at $t = 1$. Because of 16 encoder regions, the evolution of 16 state-action pairs (y_t, i_t) are plotted in this figure. The horizon axis is episode η and the vertical axis is Q-value. For each state-action pair, the Q-value tends to stable and converges to a fixed value. Other pairs have a similar convergence.

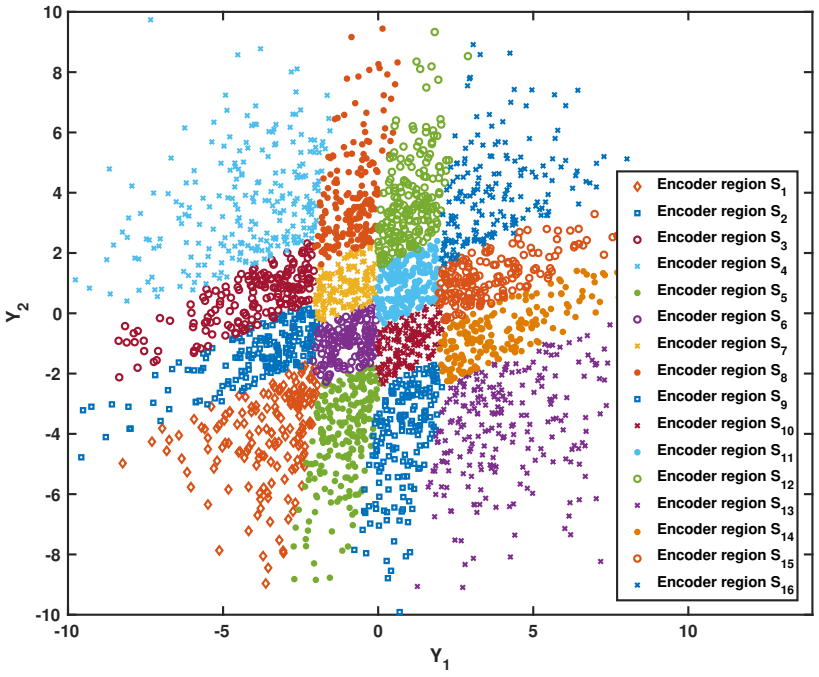

FIGURE 3.9: The optimized encoder regions of open-loop encoder system with the crossover probability $p = 0.02$ at $t = 1$.

3.7.2 Controller Design

In controller design, the basis function is

$$\phi(\hat{x},u) = [\hat{x}_1^2 \quad \hat{x}_1 * \hat{x}_2 \quad \hat{x}_1 * u_1 \quad \hat{x}_1 * u_2 \quad \hat{x}_2^2 \quad \hat{x}_2 * u_1 \quad \hat{x}_2 * u_2 \quad u_1^2 \quad u_1 * u_2 \quad u_2^2]$$

The optimal controller design is conducted based on the obtained optimal encoder mapping. The optimized control gain can be obtained by Q-learning method. In this simulation, K is a $2*2$ matrix, which are

$$K_t = \begin{bmatrix} K_t(1,1) & K_t(1,2) \\ K_t(2,1) & K_t(2,2) \end{bmatrix}$$

In Fig. 3.11, there are three figures, which represent the evolution of control gain with different system time t. The optimized control gain obtained by Q-learning are close to the result $K_t = \begin{bmatrix} -0.9666 & -0.2931 \\ -0.7806 & 0.1832 \end{bmatrix}$ calculated with open-loop encoder system.

Fig. 3.12 illustrates the weight sequence W. Based on the system utilized in the simulation, the size of weight sequence W is $10*1$ at each time step, which is

$$W_t = [W_t(1) \quad W_t(2) \quad W_t(3) \quad W_t(4) \quad W_t(5) \quad W_t(6) \quad W_t(7) \quad W_t(8) \quad W_t(9) \quad W_t(10)]'$$

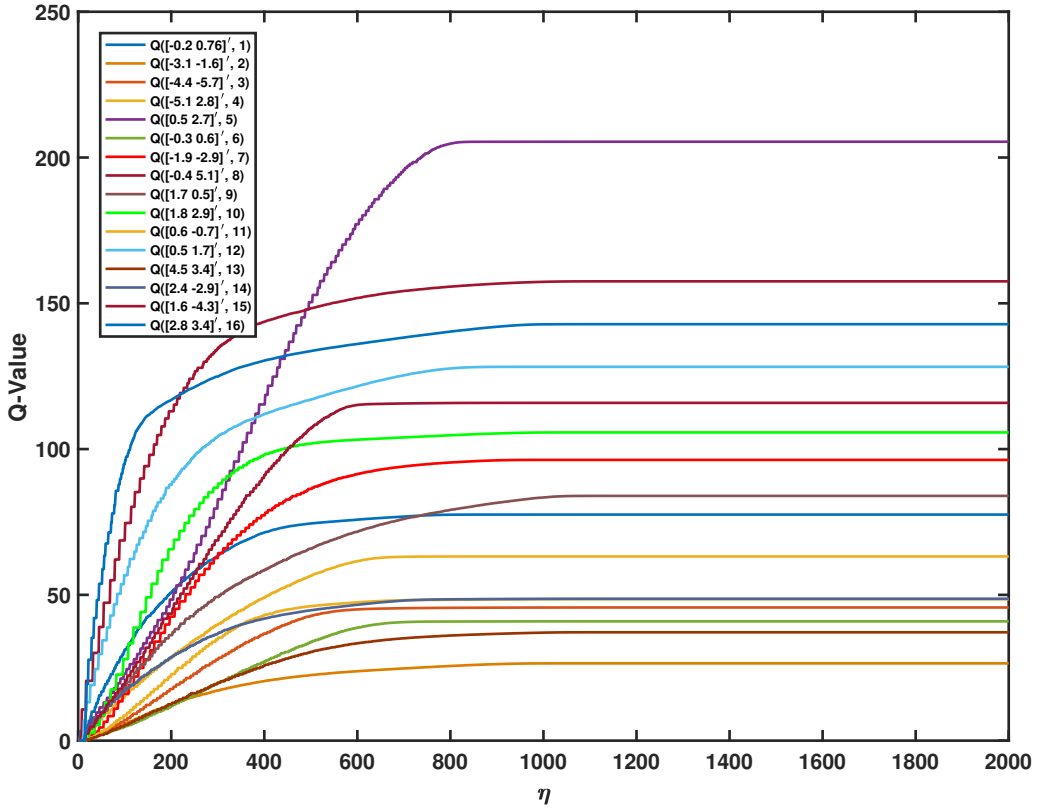

FIGURE 3.10: Randomly choose one state action pair (y_t, i_t) from 16 encoder regions and plot its Q-value evolutions at $t = 1$ with the crossover probability $p = 0.02$(memoryless encoder).

Fig. 3.12 plots 10 elements evolution with system iteration. These three figures mean that the finite time horizon $T = 3$. From Fig. 3.11, the elements of weight sequence can converge to the solution of open-loop encoder system.

3.7.3 Iterative Design

For iterative design component, the evolution of the expected overall cost with different cross over probability p for various encoder policies are shown in Fig. 3.13. It is assumed that the crossover probability $0 \leq p \leq 0.5$. If $p > 0.5$, then the receiver can swap the output (interpret 1 when it sees 0, and vice versa) and obtain an equivalent channel with crossover probability $1 - p \leq 0.5$.

The expected overall cost $\mathbb{E}(J_{tot})$ is calculated based on the optimized encoder and controller mappings after the iteration loop, and the comparison results are depicted in Fig 3.13. In this figure, the horizon axis is crossover probability p of communication channel, and the vertical axis is the expected overall cost $\mathbb{E}\{J_{tot}\}$. The system performance has an rising trend with

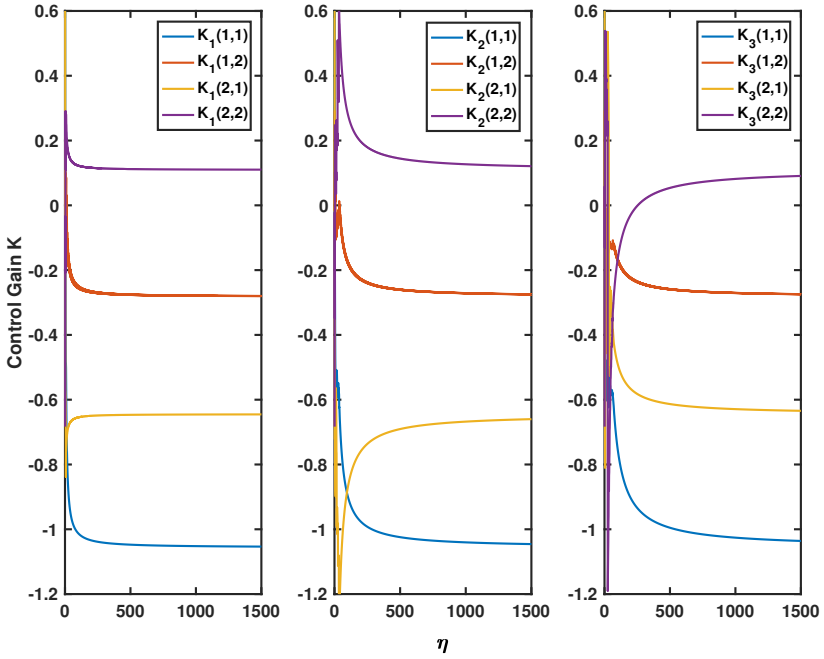

FIGURE 3.11: The evolution of control gain with crossover probability $p = 0.02$

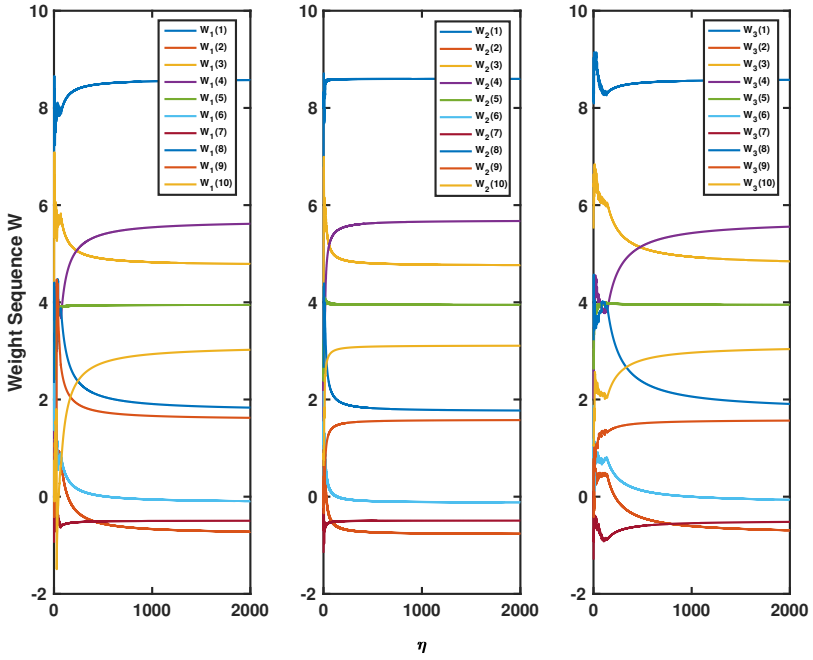

FIGURE 3.12: The evolution of weight sequence W with crossover probability $p = 0.02$

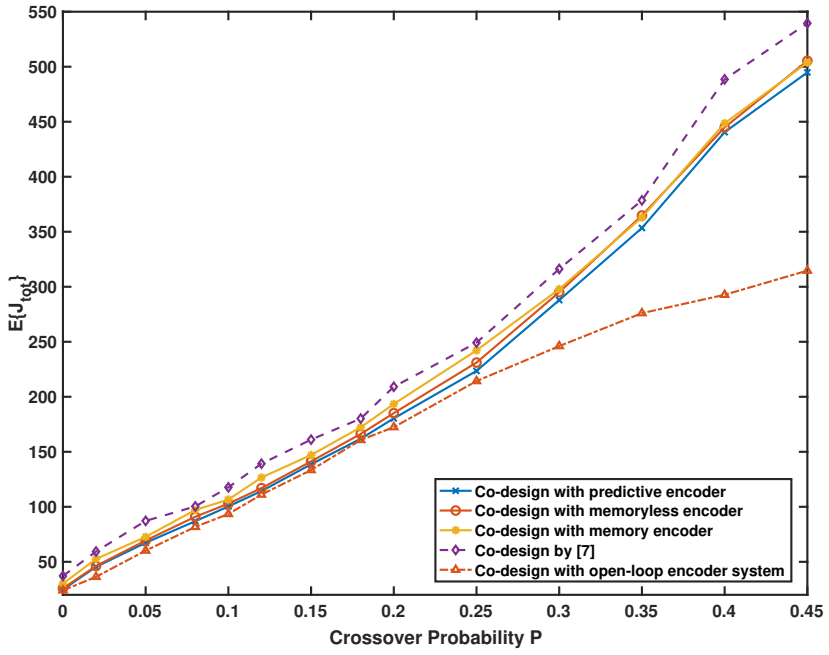

FIGURE 3.13: Performance Comparison

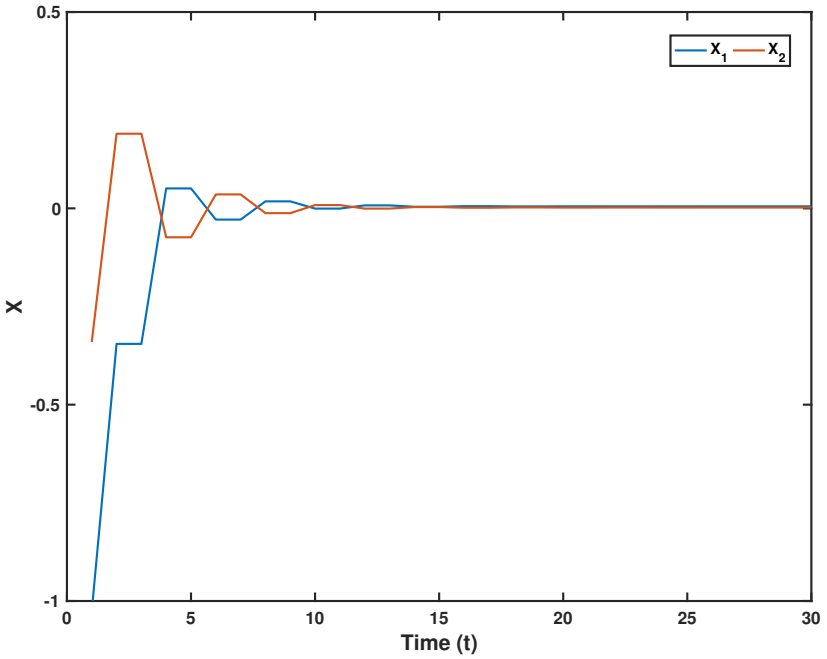

FIGURE 3.14: The closed loop trajectories of system state $x(t)$ obtained by Q-learning method with memoryless encoder strategy.

the crossover probability increasing. Obviously, the system performance of the proposed co-design (solid line) is better than that of Lei et al. [2011](dash line). However, the orange dash line with triangle in the figure represents the overall expected cost of open-loop encoder system, which is calculated by equation (3.74). Undoubtedly, this is the best result. Because the crossover probability has a very little impaction about these terms of equation (3.74), $x_0^T I_T x_0$ and $\sum_{t=0}^{T-1} \mathrm{Tr}\{(D+I_{T-t})\Psi_t\}$. Furthermore, the $\sum_{t=0}^{T-1} \mathbb{E}\{\tilde{x}_t^T \pi_{T-t} \tilde{x}_t | j_t\}$ which determines the encoder optimization, is also little affected by crossover probability, because π_{T-t} is determined by system parameters. Hence, model-based method has advantages when the crossover probability increases large.

While the system performance obtained by these three encoder strategies are very close. In particular, the expected overall cost with predictive encoder is better than others, since model information is utilized. To sum up, memoryless encoder design method is the better than two others, since model-free and computation reduction.

Fig. 3.14 shows the closed-loop trajectories of system state and control signal with memoryless encoder strategy. It can be seen that both the system states finally converge to zero.

3.8 Conclusions

In this chapter, the co-design problem of encoder and controller for feedback control systems over BSC has been considered. An iterative design based on Q-learning has been proposed to obtain a pair of encoder and controller that can optimize a finite-horizon cost function. Three encoder strategies have been considered in this chapter, and theoretical analysis and simulation results have shown the merits of the proposed method.

Chapter 4

Co-Design of Encoder and Controller for Feedback Control Systems Over Gaussian Channels Using Q-Learning

This chapter illustrates encoder and controller design problem with infinite communication channel outputs, which is the main difference with Chapter 3. The challenge is that the trained encoder-controller can no longer be implemented as a simple look-up table. Since the infinite-channel outputs mainly affect the controller design, three types of controller are considered. Hence, soft-information based controller design, hard-information based controller design and the combination of soft-hard information based controller design are studied.

4.1 Introduction

In Chapter 3, co-design encoder and controller over BSC is introduced, where the channels have finite input and output. Finite input and output means the alphabet is finite. There are some infinite channels output in practical, which can be regarded as continuous channel output. Hence, we investigate how to design encoder and controller system with infinite channel output. More specifically, we study a problem about the optimization of encoder and controller mappings affected by infinite channel output. The system model is same with Chapter 3.

Gaussian channel is a typical representative, which has the continuous channel output. In communication theory, Gaussian channel is a very common channel model, such as adding additive Gaussian noise on digital signals. Furthermore, it can provide structural and functional insights of the solution, by means of instructive and relatively simple calculation.

The rest of this chapter is organized as follows. Section 4.2 describes the extended system model, where the Gaussian channel is utilized in the system to generate the infinite output channels. Then, three controller design methods are given in Section 4.3, which are soft-controller

design, hard-controller design and combined soft-hard controller design. In this chapter, soft channel output is viewed as a real-valued channel output. While a hard channel output to an integer-valued channel output, which is similar to BSC. After that, there is a brief introduction about encoder design, since encoder design is similar as Chapter 3. Section 4.4 presents the iterative design of encoder and controller. Finally, simulations are given to demonstrate the system performance with three controller strategies.

4.2 Preliminaries

In this section, we first introduce a general feedback control system with encoder and controller over Gaussian channels in Section 4.2.1, which is an extended version of the control system described in Chapter 3. The main difference is infinite-output channels considered in this chapter. It is worth to point out that infinite-output channels can be viewed as a special case of finite-output channels where the output alphabet is infinite, or even uncountable. Other system components are same with the system introduced in Chapter 3 where more descriptions can be found. In Section 4.2.2, the design problem is formulated. The design purpose is to find the optimal encoder and controller policies so as to minimize the expected system performance.

4.2.1 System Model

In the most general case, we consider a control system with a communication channel as depicted in Fig 4.1. Consider a multi-variable linear plant (same as the linear plant in Fig 3.1). The system state-space equation is

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + v_t \\ y_t &= Cx_t + e_t \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ are the state, the control and the measurement, respectively. The matrix (A,B) is controllable and (A,C) is observable. v_t and e_t are process noise and measurement noise, which are independent and identically distributed. Furthermore, the initial state x_0, v_t and e_t are assumed as Gaussian distribution with mean zero.

The encoder is a mapping from the various measurement values y_t to a discrete set of symbols. The index of each symbol is represented by i_t, $i_t \in \mathscr{I} = \{1,2,\cdots,I\}, I \in \mathbb{N}$, where I is a integer index. It has already given that I is determined by the transmission rate ρ as $\rho = \log_2 I$. The encoder mapping f_t is expressed as

$$i_t = f_t(y_t) \tag{4.2}$$

These discrete symbols are transmitted through channels, and then picked up by the controller. Due to the noise or other factors, the picked symbol may not be the same as the original one, and this is described by the following channel mapping function.

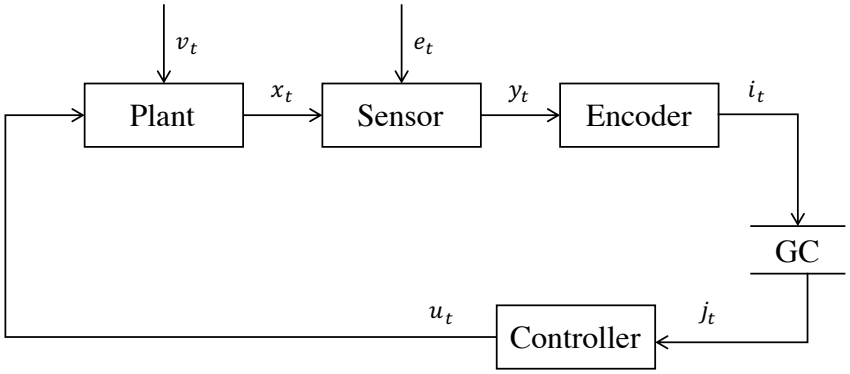

FIGURE 4.1: A general feedback control system over Gaussian channels (GC).

The discrete index i_t is then mapped into codewords b_t with ρ bits.

$$[b_t^{[\rho]}(i_t), b_t^{[\rho-1]}(i_t), \cdots, b_t^{[\kappa]}(i_t), \cdots, b_t^{[1]}(i_t)] \tag{4.3}$$

where $b_t^{[\kappa]}(i_t), \kappa \in \{1, 2, \cdots, \rho\}$, represents one codeword.The subscript t represents that the codeword vector b is time-varying. The superscript κ in the square bracket indicates that the bit is the κ^{th} element of the vector b_t. In the round bracket, it describes that the codeword b_t depends on i_t.

The codewords b_t are sent over the channel to the real-valued vector h_t, defined as

$$h_t = G_t(b_t) \tag{4.4}$$

where G_t is a random memoryless mapping. Since h_t is obtained from b_t, this mapping is independent with past events. The main difference with Chapter 3 is the real-valued channel output of Gaussian channel, h_t is a real-valued vector with $h_t \in \mathbb{R}^{\rho}$.

After Gaussian channel, a real-valued output vector h_t is produced. The κ^{th} element of h_t is defined as

$$h_t^{[\kappa]} = b_t^{[\kappa]} + w_t^{[\kappa]} \tag{4.5}$$

The additive noise $w_t^{[\kappa]}$ is independent and identically distributed, with zero mean Gaussian. The variance is time-invariant σ_w^2. Obviously, the channel has finite channel input and infinite channel output.

After these processes, the channel output h_t are coded to $\hat{h}_t$, which is a decoder process. The codewords $\hat{h}_t$ is mapped into a so-called received index $j_t \in \mathscr{J} = \{1, 2, \cdots, J\}, J \in \mathbb{N}$, which has the same finite alphabet with i_t. The channel is characterized by the additive noise, and output of channel is affected by Gaussian channel variance σ_w^2.

The Gaussian channel is the most important continuous alphabet channel, modelling a wide range of communication channels. In general, a real-valued channel outputs are defined as a soft channel output [Knagenhjelm and Agrell, 1996, Skoglund, 1999] and an integer-valued channel

outputs are referred to as a hard channel output. Hence, the soft controller refers to the controller designed with real-valued channel outputs and obviously the hard controller utilizes the integer-valued channel outputs. In addition, a combined soft-hard channel outputs are also considered by the controller design, which means hard information is utilized to decode real-valued h_t to codeword $\hat{h}_t$ and the Hadamard matrix decodes $\hat{h}_t$ to j_t.

The controller picks up the channel output j_t, and then generates control signal u_t, namely

$$u_t = g_t(j_t) \tag{4.6}$$

Note that the control command u_t is completely determined by the channel outputs.

4.2.2 Problem Statement

The design objective is to find a pair of encoder mapping $f_t^\star$and controller mapping $g_t^\star$ that can optimize the following cost function,

$$\mathbb{E}\{J_{tot}\} \triangleq \mathbb{E}\{\sum_{t=0}^{T-1}(x_{t+1}^T D x_{t+1} + u_t^T E u_t)\} \tag{4.7}$$

where matrix D and E are weight sequence of state and control and are symmetric and positive definite.

4.3 Controller Design

In the proposed system, the controller can only utilize the information after wireless communication channel, in which the information loss happens. The information loss process is invertible and affects the system performance. How to design the controller to minimize the system expected cost is considered in this section. What's more, low transmission rate is considered to overcome time delay. The objective of this section is to find controller mappings with optimized encoder such that expected overall linear quadratic cost can be minimized. More specifically, the optimized controller mappings $[g_0^*, g_1^*, g_2^*, \cdots, g_{T-1}^*]$ can be obtained by the Q-learning method.

Theoretically, the optimal encoder-controller of open-loop encoder system in Chapter 3 has no constraints on channel input and output. Hence, the optimal controller of open-loop encoder system in Chapter 3 can be utilized in finite-input infinite-output channels.

Theorem 8. In open-loop encoder system, the encoder component is fixed with $\bar{f}_0, \bar{f}_1, \cdots, \bar{f}_{T-1}$. Given the linear plant (4.1) and the Gaussian channel (4.4), the controller $u_t = g_t(j_t)$ that can

minimizes the system performance (4.7) is given by

$$\begin{aligned} u_t &= k_t \hat{x}_t \\ k_t &= -(E + B^T(D + I_{T-t-1})B)^{-1} B^T (D + I_{T-t-1})A \\ I_{T-t} &\triangleq A^T (D + I_{T-t-1})A - \pi_{T-t} \\ \pi_{T-t} &\triangleq A^T (D + I_{T-t-1})B(E + B^T(D + I_{T-t-1})B)^{-1} B^T (D + I_{T-t-1})A \end{aligned} \tag{4.8}$$

where k_t can be calculated by Theorem 7. The initial condition $I_1 = A^T DA - A^T DB(E + B^T DB)^{-1} B^T DA$.

The proof is same as Theorem 7 in Chapter 3. However, it is very difficult to calculate the control policy k_t in Theorem 8 in practice. There are several reasons. First of all, there is no analysis solution of $\mathbb{E}\{x_t | j_t\}$. Secondly, curse dimensionality is unavoidable. Last but not least, there is side-information in open-loop encoder system, which means the information from controller get back to the encoder. While it is unrealistic to transfer real-valued channel outputs to encoder. Because it is a certain local optimal solution, it can be a reference of the proposed design.

4.3.1 Controller Design by Q-Learning

Generally, Hamilton-Jacobi-Bellman (HJB) equations, such as Riccati equation for linear systems, are essential to optimal control design. However, it is difficult to obtain a solution of HJB equation when there is information loss between the encoder and the controller. Based on the techniques presented in Lewis and Vrabie [2009], Lewis et al. [2012b], Bradtke et al. [1994], the controller in the proposed system can be optimized without solving the HJB equation.

The nonlinear Q-value function can be written by dense basis function [Lewis and Vrabie, 2009, Lewis et al., 2012b, Bradtke et al., 1994],

$$Q(\hat{x}_t, u_t) = W^T \phi(\hat{x}_t, u_t) \tag{4.9}$$

where $\hat{x}_t$ is the estimate of x_t obtained by decoding j_t to $\hat{x}_t$ and $\hat{x}_t = \mathbb{E}\{x_t | j_t\}$. The main different of soft controller design, hard controller design and combined soft-hard controller design is how to calculate the estimate value $\hat{x}_t$.

It is stressed that the decoding process depends on the used encoder, and therefore encoder needs to be fixed first. Based on the introduction of Chapter 3, ϕ is the dense basis set and composed by quadratic terms of $\hat{x}_t$ and u_t. The weighting vector W is generated by the elements of Riccati matrix, which is symmetric and has the size $(n+m) \times (n+m)$. Similarly, there are $\frac{(n+m)*(n+m+1)}{2}$ independent elements in the dense basis $\phi(\hat{x}_t, u_t) = \begin{bmatrix} \hat{x}_t^2 & \hat{x}_t u_t & u_t^2 \end{bmatrix}^T$.

Based on the Q-learning updating principle, the updating equation for control policy is

$$W_{\eta+1}^T \phi(\hat{x}_t, u_t) = r_t + W_\eta^T \phi(\hat{x}_{t+1}, u_{t+1}) \tag{4.10}$$

where $r_t = \hat{x}_t^T D\hat{x}_t + u_t^T E u_t$ is the reward; η is the iterative number; W_η represents the weight vector in the η^{th} iteration. Note that u_t and u_{t+1} can be estimated by control gain k_t, that is $u_t = k_t\hat{x}_t$ and $u_{t+1} = k_t\hat{x}_{t+1}$.

Obviously, the updating equation (4.10) is a recursive equation, where $W_{\eta+1}$ is the only unknown parameter in the η^{th} iteration. Least square method or recursive least square method can be used to calculate $W_{\eta+1}$, and then the Riccati matrix can be derived from W_η. Once the Riccati matrix is available, the optimized controller policy can be obtained immediately. The controller design process is similar as Fig. 3.6 in Chapter 3.

In the remainder of this section, we propose three estimator-based controllers according to different decoder strategies of Gaussian channel, which are soft controller design, hard controller design and combined soft-hard controller design.

4.3.2 Soft-Information Based Controller Design

Infinite channel output is impossible to be decoded like a look-up table which is utilized in finite-output channel. The estimate value $\hat{x}_t = \mathbb{E}\{x_t|h_t\}$, which means the controller can only use the information h_t (see Fig. 4.2). In Fig. 4.2, all elements of i_t are coded to codeword b_t. w_t is the adding Gaussian noise. h_t is the real-valued vector and then sent to controller.

How to decode the real-value to integer value is the key point.Hadamard-based decoding techniques is utilized to transform the real-valued channel output to source symbol and has been shown that the real-value can be mapped into integer values [Knagenhjelm and Agrell, 1996, Skoglund, 1999, Bao and Skoglund, 2010].

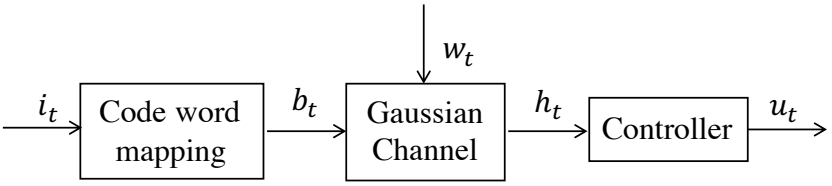

FIGURE 4.2: Soft information based controller

Baes on Skoglund [1999], the following decibels the soft controller design with Hadamard matrix decoding. First of all, we formulate $\mathbb{E}\{x_t|h_0^t\}$, h_0^t is a fixed sequence of channel outputs, which is very useful in the controller design. Like dynamic programming structure, we rewrite $\mathbb{E}\{x_t|h_0^t\}$ to a recursive formulation according to the system model and Bayes' rule.

$$\begin{aligned}\mathbb{E}\{x_t|h_0^t\} &= \sum_{c=0}^{I-1} P(i_t=c|h_0^t)\mathbb{E}\{x_t|i_t=c,h_0^{t-1}\} \\ &= \frac{\sum_{c=0}^{I-1} P(i_t=c|h_0^{t-1})p(h_t|i_t=c,h_0^{t-1})\mathbb{E}\{x_t|i_t=c,h_0^{t-1}\}}{\sum_{d=0}^{I-1} P(i_t=d|h_0^{t-1})p(h_t|i_t=d,h_0^{t-1})} \\ &= \frac{\sum_{c=0}^{I-1} p(h_t|i_t=c)P(i_t=c|h_0^{t-1})\mathbb{E}\{x_t|i_t=c,h_0^{t-1}\}}{\sum_{d=0}^{I-1} p(h_t|i_t=d)P(i_t=d|h_0^{t-1})}\end{aligned} \tag{4.11}$$

where P is probability mass function and p is the probability density function. $p(h_t|i_t)$ can be computed since h_t follows on Gaussian distribution. h_0^t is a fixed sequence of channel output from $t=0$ to t.

Because we emphasis that the channel is memoryless, h_t is independent with h_0^{t-1}. In equation (4.11), the calculation of $\mathbb{E}\{x_t|i_t,h_0^{t-1}\}$, $P(i_t=c|h_0^{t-1})$ is very difficult and even need huge space for storage. While $p(h_t|i_t=c)$ is easy to compute since h_t is adding an additive noise on i_t, which is a Gaussian pdf. The following describes the calculation of $p(x_t|h_0^t)$ with given $p(x_{t-1}|h_0^{t-1})$.

According to equation (4.11), $P(i_t|h_0^{t-1})$ can be obtained by the encoder mapping and $p(x_t|h_0^{t-1})$, which means $p(x_{t-1}|h_0^{t-1})$ is utilized. $p(x_{t-1}|h_0^{t-1})$ can be written as

$$p(x_t|h_0^{t-1}) = p(Ax_{t-1}+Bu_{t-1}+v_{t-1}|h_0^{t-1}) \tag{4.12}$$

where this computation is straightforward,, since u_{t-1} can be got by r_0^{t-1}. v_{t-1} is process noise, which follows Gaussian distribution and independent with others. Therefore, $\mathbb{E}\{x_t|i_t=c,h_0^{t-1}\}$ can be calculated based on $p(x_t|i_t,h_0^{t-1})$, which can be formulated as

$$p(x_t|i_t,h_0^{t-1}) = \frac{p(x_t|h_0^{t-1})P(i_t=c|x_t,r_0^{t-1})}{\int_{x_t} p(x_t|h_0^{t-1})P(i_t=c|x_t,r_0^{t-1})dx_t} \tag{4.13}$$

where all elements can be calculated in this equation based on the previous analysis.

However, it is very difficult to calculate $p(x_t|h_0^{t-1})$ with high dimension, to some extent impossible. Furthermore, the storage problem is still here because of curse dimensionality. Obviously, this is not a good method to solve the proposed problem. While Knagenhjelm and Agrell [1996], Skoglund [1999] have shown that the estimation value $\mathbb{E}\{x_t|h_0^t\}$ is reconstructed by Hadamard matrix. The idea is that the encoder process can be imitated based on the known information. The reversion process can be utilized as decoding process, which can decode the real-valued vector to discrete integer values.

To solve the problem, we first need to obtain the encoding matrix. The matrix $\bar{C}_t(h_0^{t-1})$ is constructed which carries a priori information about h_t. It can be viewed as the c^{th} column of $\bar{C}_t(h_0^{t-1})$ and defined as

$$\bar{c}_t(c,h_0^{t-1}) = \mathbb{E}\{x_t|i_t=c,h_0^{t-1}\}, c\in\mathscr{I} \tag{4.14}$$

The matrix $\bar{C}_t(h_0^{t-1})$ also can decomposed as two matrices, which is

$$\bar{C}_t(h_0^{t-1}) = \bar{T}_t(h_0^{t-1})H_t(h_0^{t-1}) \tag{4.15}$$

where $H_t(h_0^{t-1})$ is Hadamard matrix, and the c^{th} column of $H_t(h_0^{t-1})$, named $\bar{h}_t(c,h_0^{t-1})$ $c \in \mathscr{I}$, is composed by the codeword

$$\left[b_t^{[\rho]}(i_t = c, h_0^{t-1}), b_t^{[\rho-1]}(i_t = c, h_0^{t-1}), \cdots, b_t^{[1]}(i_t = c, h_0^{t-1})\right]$$

where b_t is known and can be obtained by i_t.

The column of $\bar{h}_t(c,h_0^{t-1})$ cab be expressed as math equation.

$$\bar{h}_t(c,h_0^{t-1}) = \begin{bmatrix} 1 \\ b_t^{[\rho]}(i_t = c, h_0^{t-1}) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ b_t^{[\rho-1]}(i_t = c, h_0^{t-1}) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ b_t^{[1]}(i_t = c, h_0^{t-1}) \end{bmatrix}$$

where the symbol $\otimes$ is the Kronecker product. If $\bar{C}_t(h_0^{t-1})$ and $H_t(h_0^{t-1})$ are known, it is straightforward to get the matrix $\bar{T}_t(h_0^{t-1})$ by equation 4.15. Because $\bar{T}_t(h_0^{t-1})$ is the relation between channel input i_t and the priori information of h_t, named $\bar{C}_t(h_0^{t-1})$, it can be regarded as the encoding matrix. Obviously, $\bar{T}_t(h_0^{t-1})$ is unique for each h_0^{t-1}.

Since the encoder process is same as decoder process in this paper, $\bar{T}_t(h_0^{t-1})$ can be utilized as the decoding matrix, which is a reverse of channel encoding process. Hence, equation 4.11 can be rewritten as

$$\begin{aligned} \mathbb{E}\{x_t|h_0^t\} &= \hat{x}_t \\ &= \bar{T}_t(h_0^{t-1}) \frac{\sum_{c=0}^{I-1} p(h_t|i_t = c)P(i_t = c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})}{\sum_{d=0}^{I-1} p(h_t|i_t = d)P(i_t = d|h_0^{t-1})} \\ &= \bar{T}_t(h_0^{t-1})\hat{h}_t(h_0^t) \end{aligned} \tag{4.16}$$

Obviously, only $\hat{h}_t(h_0^t)$ contains h_t and encoding matrix $\bar{T}_t(h_0^{t-1})$ is non-changeable with different h_t. Following Theorem 1 in Knagenhjelm and Agrell [1996], Skoglund [1999], Bao and Skoglund [2010], the lemma is

Lemma 1. $\hat{h}_t(h_0^t)$ can be calculated by

$$\hat{h}_t(h_0^t) = [(m_{\bar{h}}(h_0^{t-1}))^T \hat{P}_t(h_0^t)]^{-1} R_{\bar{h}_t\bar{h}_t}(h_0^{t-1})\hat{P}_t(h_0^t) \tag{4.17}$$

where $m'_{\bar{h}}(h_0^{t-1})$, $\hat{P}_t(h_0^t)$ and $R_{\bar{h}_t\bar{h}_t}(h_0^{t-1})$ is

$$\begin{aligned} R_{\bar{h}_t\bar{h}_t}(h_0^{t-1}) &\triangleq \sum_{c=0}^{I-1} P(i_t = c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T \\ m'_{\bar{h}}(h_0^{t-1}) &\triangleq \sum_{c=0}^{I-1} P(i_t = c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1}) \\ \hat{P}_t(h_0^t) &\triangleq \mathbb{E}\{\bar{h}_t(c,h_0^{t-1})|h_t, P(i_t = c|h_0^{t-1}) = \frac{1}{I}, \forall c\} \end{aligned} \tag{4.18}$$

Proof.

$$\hat{h}_t(h_0^t) = \frac{\sum_{c=0}^{I-1} p(h_t|i_t=c)P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})}{\sum_{d=0}^{I-1} p(h_t|i_t=d)P(i_t=d|h_0^{t-1})} \tag{4.19}$$

It can also be written as

$$\hat{h}_t(h_0^t) = \frac{\sum_{c=0}^{I-1} P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T \frac{1}{I}\sum_{a=0}^{I-1}\bar{h}_t(a,h_0^{t-1})p(h_t|i_t=a)}{\sum_{d=0}^{I-1} P(i_t=d|h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T \frac{1}{I}\sum_{b=0}^{I-1}\bar{h}_t(b,h_0^{t-1})p(h_t|i_t=b)} \tag{4.20}$$

For the numerator of equation (4.18),

$$\begin{aligned}
&\sum_{c=0}^{I-1} P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T \frac{1}{I}\sum_{a=0}^{I-1}\bar{h}_t(a,h_0^{t-1})p(h_t|i_t=a)\\
=&\sum_{c=0}^{I-1}\sum_{a=0}^{I-1}\frac{1}{I}(P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1}))((\bar{h}_t(c,h_0^{t-1}))^T\bar{h}_t(a,h_0^{t-1}))p(h_t|i_t=a)\\
=&\sum_{c=0}^{I-1}\sum_{a=0}^{I-1}\frac{1}{I}(P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1}))I\delta_{c,a}p(h_t|i_t=a)\\
=&\sum_{c=0}^{I-1}(P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1}))p(h_t|i_t=c)
\end{aligned} \tag{4.21}$$

where $\delta_{c,a}$ is delta function and can be defined as

$$\delta_{c,a} \triangleq \begin{cases} 1, & c=a \\ 0, & c\neq a \end{cases} \tag{4.22}$$

Here, we have used the special property of the Hadamard matrix, which has been given by Skoglund [1999],

$$(\bar{h}_t(c,h_0^{t-1}))^T\bar{h}_t(a,h_0^{t-1}) = \begin{cases} I, & c=a \\ 0, & c\neq a \end{cases} \tag{4.23}$$

For denominator of equation (4.20),

$$\begin{aligned}
&\sum_{d=0}^{I-1} P(i_t=d|h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T \frac{1}{I}\sum_{b=0}^{I-1}\bar{h}_t(b,h_0^{t-1})p(h_t|i_t=b)\\
=&\sum_{d=0}^{I-1}\sum_{b=0}^{I-1}\frac{1}{I}P(i_t=d|h_0^{t-1})((\bar{h}_t(c,h_0^{t-1}))^T\bar{h}_t(b,h_0^{t-1}))p(h_t|i_t=b)\\
=&\sum_{d=0}^{I-1}\sum_{b=0}^{I-1}\frac{1}{I}P(i_t=d|h_0^{t-1})I\delta_{d,b}p(h_t|i_t=b)\\
=&\sum_{d=0}^{I-1}P(i_t=d|h_0^{t-1})p(h_t|i_t=d)
\end{aligned} \tag{4.24}$$

Combine both numerator and denominator together to equation (4.20) and yields to

$$\begin{aligned}\hat{h}_t(h_0^t) =& \frac{\sum_{c=0}^{I-1} P(i_t=c|h_0^{t-1})\bar{h}_t(c,h_0^{t-1})(\bar{h}_t(c,h_0^{t-1}))^T}{\sum_{d=0}^{I-1} P(i_t=d|h_0^{t-1})(\bar{h}_t(d,h_0^{t-1}))^T} \times \frac{\frac{1}{I}\sum_{a=0}^{I-1}\bar{h}_t(a,h_0^{t-1})p(h_t|i_t=a)}{\frac{1}{I}\sum_{a=0}^{I-1}p(h_t|i_t=a)} \\ &\times \frac{\frac{1}{I}\sum_{b=0}^{I-1}\bar{h}_t(b,h_0^{t-1})p(h_t|i_t=b)}{\frac{1}{I}\sum_{a=0}^{I-1}p(h_t|i_t=a)}\end{aligned} \tag{4.25}$$

Now we can easily identify the equation (4.17) and (4.18). The proof is completed. □

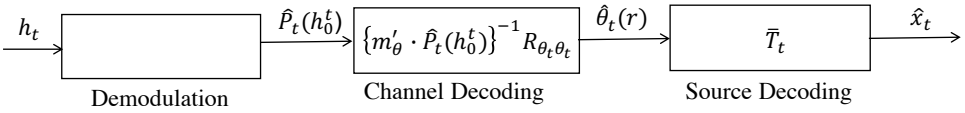

FIGURE 4.3: Separation of the soft controller design decoding procedure

The decoding process shows clearly in Fig. 4.3. It is easy to calculate $R_{\bar{h}_t\bar{h}_t}(h_0^{t-1})$ and $m'_{\bar{h}}(h_0^{t-1})$. $\hat{P}_t(h_0^t)$ is affected by r_t and is a posteriori expectation of $\bar{h}_t(h_0^{t-1})$. Because the probability of $P(i_t=c|h_0^{t-1})=\frac{1}{I}, \forall c \in \mathscr{I}$ are equal for each i_t, the codeword $b_t^{[\kappa]}(h_0^{t-1}), \kappa \in \{1,2,\cdots,\rho\}$ are independent.

$\hat{P}_t(h_0^t)$ can be written as

$$\begin{aligned}\hat{P}_t(h_0^t) &\triangleq \mathbb{E}\{\bar{h}_t(c,h_0^{t-1})|h_t, P(i_t=c|h_0^{t-1})=\frac{1}{I}, \forall c\} \\ &= \frac{\frac{1}{I}\sum_{d=0}^{I-1}\bar{h}_t(d,h_0^{t-1})p(h_t|i_t=d))}{\frac{1}{I}\sum_{b=0}^{I-1}p(h_t|i_t=b)}\end{aligned}$$

Since $\hat{P}_t(h_0^t)$ is the only item affected by h_t, and all the codeword bits $b_t^{[\kappa]}(h_0^{t-1},h_t)$ are independent, together with the assumption of memoryless channel and encoder, the decoding process is similar with encoding process. Similarity, $\hat{P}_t(h_0^t)$ can be calculated as

$$\hat{P}_t(h_0^t) = \begin{bmatrix} 1 \\ \hat{b}_t^{[\rho]}(h_0^{t-1},h_t^{[\rho]}) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ \hat{b}_t^{[1]}(h_0^{t-1},h_t^{[1]}) \end{bmatrix}$$

where $\hat{b}_t^{[\kappa]}$ is the estimate value of b_t^{κ}. If the codeword utilized in this paper is 1, -1. $\hat{b}_t^{[\kappa]}$ can be defined as

$$\hat{b}_t^{[\kappa]}(h_0^{t-1},h_t^{[\kappa]}) = \mathbb{E}\{b_t^{[\kappa]}|h_0^{t-1},h_t^{[\kappa]},P(b_t^{[\kappa]}=-1)=\mathscr{P}(b_t^{[\kappa]}=1)=\frac{1}{2}\} \tag{4.26}$$

where -1 and 1 represents the code of channel, which is similar 0 and 1 in BSC. Note that the code can be used any label to replace, which doesn't affect the solution.

Since the current codeword transmission process is independent with all past, $\hat{b}_t^{[\kappa]}(h_0^{t-1}, h_t^{[\kappa]}) = \hat{b}_t^{[\kappa]}(h_t^{[\kappa]})$. It is straightforward to calculate $\hat{b}_t^{[\kappa]}(h_t^{[\kappa]})$,

$$\begin{aligned}\hat{b}_t^{[\kappa]}(h_t^{[\kappa]}) &= \mathbb{E}\{b_t^{[\kappa]} | h_0^{t-1}, h_t^{[\kappa]}, P(b_t^{[\kappa]} = -1) = P(b_t^{[\kappa]} = 1) = \frac{1}{2}\} \\ &= \frac{e^{\frac{h_t^{[\kappa]}}{\sigma_w^2}} - e^{-\frac{h_t^{[\kappa]}}{\sigma_w^2}}}{e^{\frac{h_t^{[\kappa]}}{\sigma_w^2}} + e^{-\frac{h_t^{[\kappa]}}{\sigma_w^2}}} \\ &= \tanh(\frac{h_t^{[\kappa]}}{\sigma_w^2})\end{aligned}$$

where time-invariant σ_w^2 is the variance of Gaussian channel.

That's the process of how to use Hadamard framework to obtain the estimate value $\hat{x}_t = \mathbb{E}\{x_t | h_t\}$. While it is quiet complex to calculate and need huge storage space, especially in high dimensional system or high transmission rate ρ. The following section introduce to hard controller design, which is used very common in practical.

4.3.3 Hard-Information Based Controller Design

In hard controller design, the formulation of the estimate value is $\hat{x}_t = \mathbb{E}\{x_t | j_t\}$ (see Fig. 4.4). Compared with soft controller decoder process via Hadamard matrix, hard controller design is utilized binary decision making to decode the real-valued channel output to codeword, which is quiet simple and easy to use. In particular, hard controller design is much more implementable. What's more, the design process is same with controller design with binary symmetric channel. While the drawback is that only partial information of channel output is considered and may deteriorate the system performance.

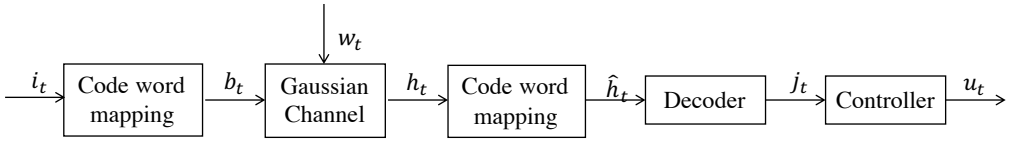

FIGURE 4.4: Hard controller $u_t = k_t \mathbb{E}\{x_t | j_t\}$

The integer-valued channel output r_t are coded to $\hat{r}_t$. The code decision $\hat{r}_t$ can be obtained by the real-valued channel. r_t can be viewed as a normal distribution with mean 1 or -1, variance σ_w^2, because b_t is composed with 1 or -1. The probability of $h_t^{[k]}$, $p(h_t^{[k]})$, can be calculated straightforward based on probability density function. The value of P is a channel error probability, which affects the information loss. The principle is

$$\hat{h_t}^{[\kappa]} = \begin{cases} b_t^{[\kappa]}, & P(h_t^{[\kappa]}) \geq \tau \\ -b_t^{[\kappa]}, & P(h_t^{[\kappa]}) < \tau \end{cases} \tag{4.27}$$

where $\tau \in [0,1]$. The code $\hat{h}_t$ is then mapped in to the received index $j_t \in \mathscr{J}$, which is same as encoder process.

4.3.4 Combined Soft-Hard Controller Design

Based on the previous two sections illustration, it is quite complex even impossible to compute $\mathbb{E}\{x_t|j_0^t\}$ in soft controller design. For hard controller design, only partial information of channel output is utilized which may deteriorated the system performance with channel error increasing. Combined these two designs together may avoid these design imperfection, which means hard information is utilized to get $\hat{h}_t$ and the Hadamard matrix decodes $\hat{h}_t$ to j_t. The estimation value of $\hat{x_t}$ is $\mathbb{E}\{x_t|h_t, j_0^{t-1}\}$ (see Fig. 4.5). The following shows how to use the Hadamard matrix to design.

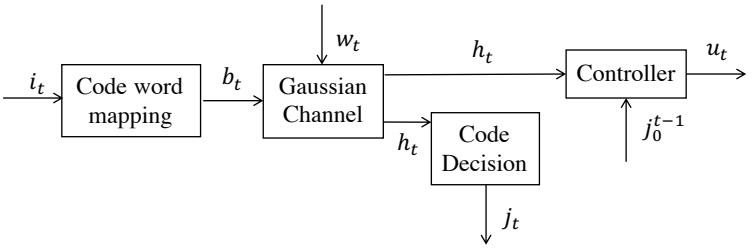

FIGURE 4.5: Combined soft and hard controller $u_t = k_t \mathbb{E}\{x_t|h_t, j_0^{t-1}\}$

The controller mapping is

$$\begin{aligned} u_t &= k_t * \mathbb{E}\{x_t|h_t, j_0^{t-1}\} \\ &= k_t * \hat{T}(j_0^{t-1})\mathbb{E}\{\bar{h}_t(j_0^{t-1})|h_t\} \end{aligned} \tag{4.28}$$

where k_t is control gain, the encoding matrix $\hat{T}(j_0^{t-1})$ can be obtained by $\hat{C}(j_0^{t-1}) = \hat{T}(j_0^{t-1})H_t(j_0^{t-1})$. The column of matrix $\hat{C}(j_0^{t-1})$ can be computed by $\mathbb{E}\{x_t|i_t, j_0^{t-1}\}$. In addition, the column of Hadamard matrix $H_t(j_0^{t-1})$ is specified by encoder bit $b_t(i_t, j_0^{t-1})$.

Based on Hadamard framework, $\mathbb{E}\{\bar{h}_t(j_0^{t-1})|h_t\}$ is defined as

$$\mathbb{E}\{\bar{h}_t(j_0^{t-1})|h_t\} = [m'_{\bar{h}_t}(j_0^{t-1})\hat{P}_t(h_t, j_0^{t-1})]^{-1} R_{\bar{h}_t\bar{h}_t}(j_0^{t-1})\hat{P}_t(h_t, j_0^{t-1})$$

where

$$\begin{aligned} R_{\bar{h}_t\bar{h}_t}(j_0^{t-1}) &\triangleq \sum_{c=0}^{I-1} P(i_t = c|j_0^{t-1})\bar{h}_t(c, j_0^{t-1})(\bar{h}_t(c, j_0^{t-1}))^T \\ (m_{\bar{h}}(j_0^{t-1}))^T &\triangleq \sum_{c=0}^{I-1} P(i_t = c|j_0^{t-1})\bar{h}_t(c, j_0^{t-1}) \\ \hat{P}_t(h_0^t) &\triangleq \mathbb{E}\{\bar{h}_t(c, j_0^{t-1})|h_t, P(i_t = c|j_0^{t-1}) = \frac{1}{I}, \forall c \in \mathscr{I}\} \\ &= \begin{bmatrix} 1 \\ \hat{b}_t^{[\rho]}(j_0^{t-1}, h_t^{[\rho]}) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ \hat{b}_t^{[1]}(j_0^{t-1}, h_t^{[1]}) \end{bmatrix} \end{aligned}$$

Note that $\hat{b}_t^{[\kappa]}(j_0^{t-1}, r_t^{[\rho]}) = \tanh \frac{j_t}{\sigma_w^2}, \kappa \in \{1, 2, \cdots, \rho\}$.

4.4 Iterative Encoder-Controller Design

For the controller component, the mainly design methods are described in Section 3.3, in which three controller strategies are given and utilized in the iterative encoder-controller design. While encoder design method is already discussed in Chapter 3, and memoryless encoder is considered in this chapter. While it is impractical to apply soft information to design encoder, because of the complexity limitation and bandwidth requirement. Obviously, there are three design processes with three controller strategies.

The iterative training algorithm is introduced in this section to obtain a pair of optimized encoder and controller that can minimize the expected overall cost. The encoder is optimized based on hard information, and then the controller is optimized with soft information, hard information, combined hard-soft information, respectively. For each time iterative learning process, the optimized encoder and controller are obtained and updated and utilized for next time iteration. The flow chart of iterative design shows as

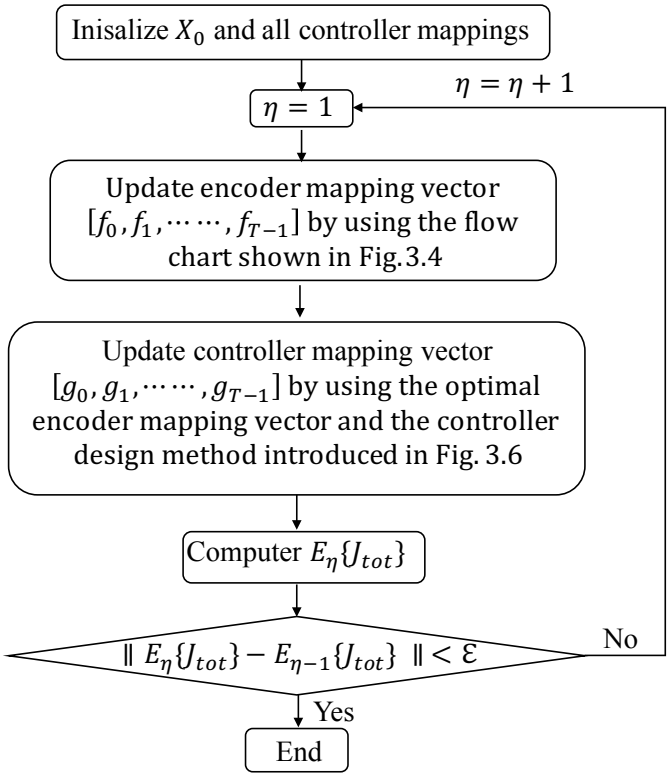

FIGURE 4.6: The flow chart of iterative memoryless/memory encoder and controller design by Q-learning (model-free and online learning method)

First of all, all the encoder mappings f_0^{T-1}, named f_0, f_1, $\cdots$, f_{T-1}, over the time interval from $t=0$ to $T-1$ are optimized with fixed controller mappings g_0^{T-1} (g_0, g_1, $\cdots$, g_{T-1}). Then, the controller mapping is calculated based on the previous optimized encoder mappings. Finally, the expected overall cost are calculated based the optimized encoder mappings f_0^{T-1} and controller mappings g_0^{T-1}.

The expected overall cost is $\mathbb{E}_n\{J_{tot} = \sum_{t=0}^{T-1} x_{t+1}^T D x_{t+1} + u_t^T E u_t\}$. It can be easily calculated with the optimized encoder-controller mappings. The algorithm is terminated when the error between $\mathbb{E}_\eta\{J_{tot}\}$ and $\mathbb{E}_{\eta-1}\{J_{tot}\}$ is small enough, which means there is no improvement in the system performance.

4.5 Numerical Results

The system parameters for simulation are $A = \begin{bmatrix} 0.99 & 0.3 \\ 0.8 & -0.7 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $C = I_2$. The time horizon is $T = 3$ and the weighting matrices in objective function are $E = 0.1I_1$ and $D = 4I_2$.

The initial state x_0, process noise v_t and measurement noise e_t follow Gaussian distribution with mean 0 and variance $\begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\begin{bmatrix} 1e-5 \\ 1e-5 \end{bmatrix}$, $\begin{bmatrix} 1e-5 \\ 1e-5 \end{bmatrix}$, respectively.

Because of low transmission rate used during system design, $\rho = 2$ is utilized in this simulation, and the number of binary codeword is $2^\rho = 4$, which are $\begin{bmatrix} -1 \\ -1 \end{bmatrix}$, $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$, and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Consequently, the number of encoder regions for the second order system is 16. The corresponding binary codeword matrix is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

For all simulation, the sample size L is 3000 and the algorithm stop accuracy is set as $\varepsilon = 10^{-5}$. The size of $Q(y_t, i_t, t)$ is $L * (2^{\rho n}) * T$, named $3000 * 16 * T$, T is the finite time horizon.

The following results are control gain obtained by these three design methods, and then the system performances obtained with optimized encoder and controller are given.

4.5.1 Controller Design

In this section, we mainly introduce numerical results of three controller design methods and all results are based on the optimized encoder. First of all, The basis function matrix is

$$\begin{bmatrix} \hat{x}_1^2 & \hat{x}_1\hat{x}_2 & \cdots & \hat{x}_1u_1 & \cdots & \hat{x}_1u_m \\ \hat{x}_2\hat{x}_1 & \hat{x}_2^2 & \cdots & \hat{x}_2u_1 & \cdots & \hat{x}_2u_m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{x}_n\hat{x}_1 & \cdots & \hat{x}_n^2 & \hat{x}_nu_1 & \cdots & \hat{x}_nu_m \\ u_1\hat{x}_1 & u_1\hat{x}_2 & \cdots & u_1^2 & \cdots & u_1u_m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u_m\hat{x}_1 & u_m\hat{x}_2 & \cdots & \hat{x}_nu_m & \cdots & u_m^2 \end{bmatrix} \tag{4.29}$$

Letting the basis function is

$$\begin{aligned} \phi(\hat{x}_t, u_t) = [\hat{x}_1^2 \quad \hat{x}_1\hat{x}_2 \quad \cdots \quad \hat{x}_1u_1 \quad \cdots \quad \hat{x}_1u_m \\ \hat{x}_2^2 \quad \cdots \quad \hat{x}_2u_1 \quad \cdots \quad u_m^2]' \end{aligned} \tag{4.30}$$

and weight sequence W is

$$\begin{aligned} W_\eta = [G_{1,1}^\eta \quad G_{1,2}^\eta \quad G_{1,3}^\eta \quad \cdots \quad G_{1,n+m}^\eta \quad G_{2,2}^\eta \\ \cdots \quad G_{2,n+m}^\eta \quad \cdots \quad G_{n+m,n+m}^\eta]' \end{aligned} \tag{4.31}$$

where $G_{1,1}^\eta$ is the element $(1,1)$ of matrix G with on η^{th} iteration.

Thereafter, the impact of soft and hard information on the overall system performance is investigated. For soft controller design, the decoder process is finished with Hadamard framework, in which the encoding matrix should be calculated and play an very important role. While the optimized control gains can be got and are close to the result $K = [-0.8692 \quad 0.2929]$ obtained by open-loop encoder system in Theorem 8.

The following Fig. 4.7, 4.8, 4.9 are the evolution of control gain with $\sigma_w^2 = 1$ for soft controller design, hard controller design and combined soft-hard controller design, respectively. The horizon axis is iteration learning time η and the vertical axis is control gain k. There are three figures in each figure, which represent the control gain evolution at the system time step $t = 1$, $t = 2$ and $t = 3$. Note that the system time step is different with iteration learning time η. While $k_t = [k_t(1) \quad k_t(2)]$ means the control gain at system time t.

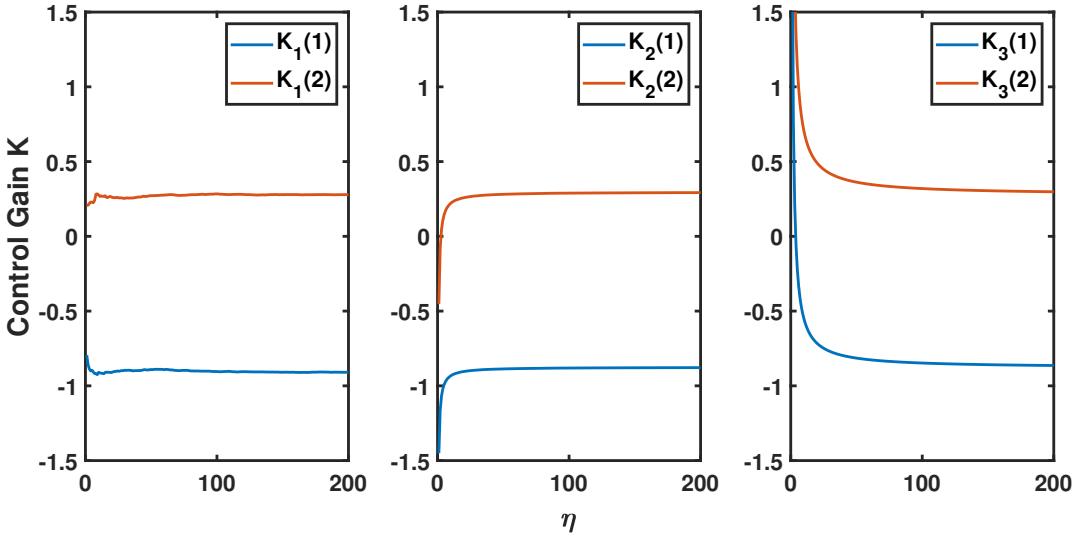

FIGURE 4.7: The evolution of control gain for soft controller design at different system times with $\sigma_w^2 = 1$

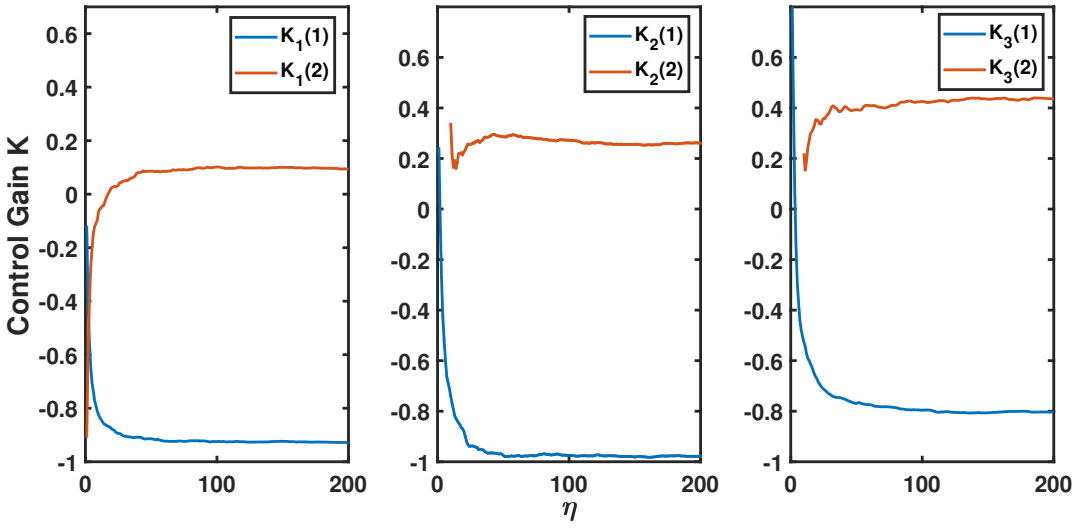

FIGURE 4.8: The evolution of control gain for hard controller design at different system times with $\sigma_w^2 = 1$

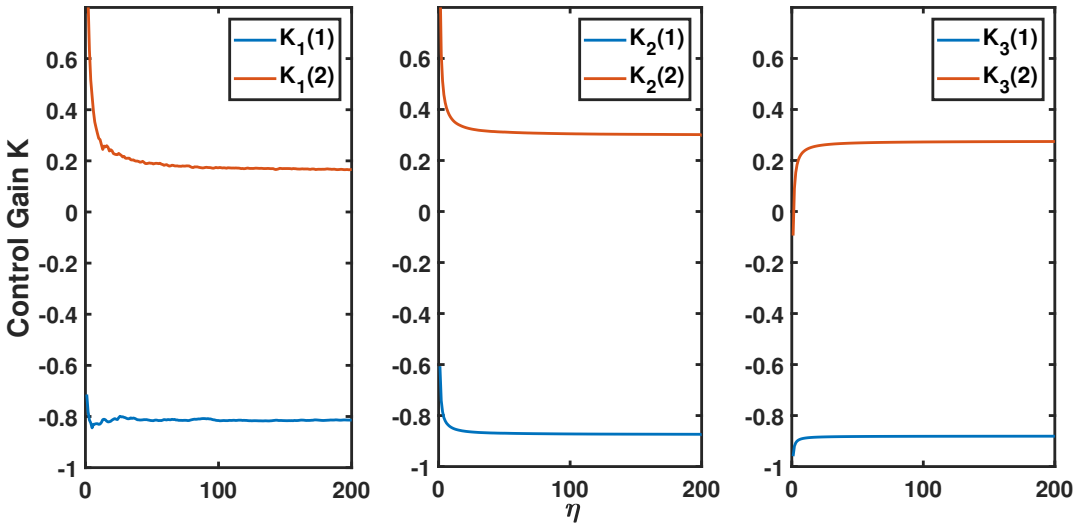

FIGURE 4.9: The evolution of control gain for combined soft and hard controller design at different system times with $\sigma_w^2 = 1$

4.5.2 Iterative Design

This section presents the simulation results about system performance of iterative encoder-controller design with three controller strategies. The encoder here we utilized is memoryless encoder and time varying. Note that three controller strategies are obtained with same encoder strategies. Fig. 4.10 and Fig. 4.11 describe the system performance evolution with different channel noise σ_w^2 with for two different R_t matrix values. Because the matrix R_t constraints the system control input, which has an affect on system performance. The system performance is larger with $R_t = 4$ than $R_t = 1$. The expected overall cost $\mathbb{E}(J_{tot})$ is calculated based on the optimal encoder and controller mappings after the iteration loop, and the comparison results are depicted in the following two figures.

The encoder and controller utilized for comparison are listed as,

1. Co-design with soft controller $u_t = K_t * \mathbb{E}\{x_t|h_t\}$.
2. Co-design with hard controller $u_t = K_t * \mathbb{E}\{x_t|j_t\}$.
3. Co-design with combined soft-hard controller $u_t = K_t * \mathbb{E}\{x_t|h_t, j_0^{t-1}\}$.

The expected overall cost of co-design with combined soft-hard controller is lower than the two others at lower channel error variance σ_w^2, because this method combines the advantages of soft controller and hard controller design. While the information changes a lot with large channel error variance, the system performance is not good as soft controller design and hard controller design. The difference of soft controller design and hard controller design is very small with low-level channel variance. On the other hand, the expected overall cost of soft controller design

is lower since the channel output r_t is directly decodes to $\mathbb{E}\{x_t|r_t\}$, because the Hadamard matrix of decoding process is same with encoding process and the accuracy is higher than look-up table decoding.

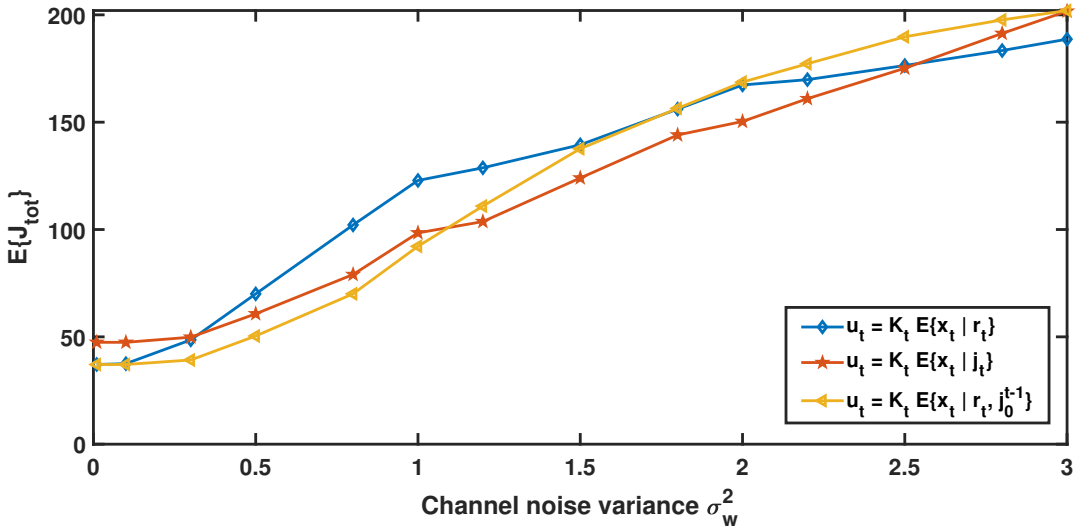

FIGURE 4.10: Performance Comparison with $E_t = 1$

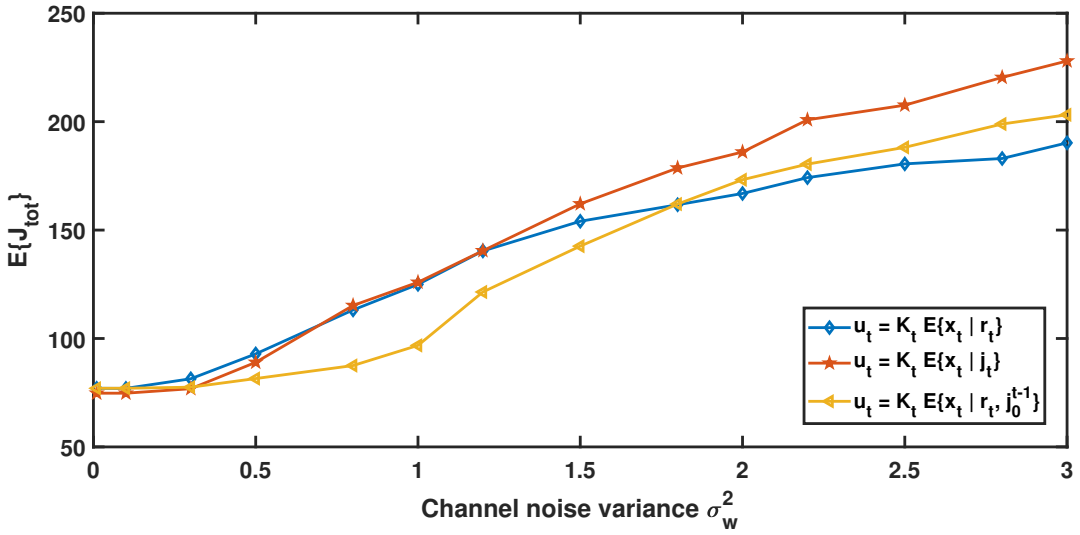

FIGURE 4.11: Performance Comparison with $E_t = 4$

4.6 Conclusions

In this chapter, the co-design problem of encoder and controller for feedback control systems over Gaussian channel has been considered. An iterative design based on Q-learning has been

proposed to obtain a pair of encoder and controller that can optimize a finite-horizon cost function. The main difference with Chapter 3 is the infinite channel outputs. Therefore, how the generalization to infinite-output channels affected the optimization of the encoder and controller theoretically and practically is studied in this chapter. From a practical point of view, the impact appeared not only in the training stage, but also in how to implement the trained encoder and controller pair. The challenge is that the trained encoder and controller can no longer be implemented as a simple look-up table. To get more insight to the optimal controller, the Hadamard-based soft controller is introduced, where fully exploited the channel outputs can be used over controller design process. However, we can not implement the soft-information-based controller in practice due to the complexity and memory demands. Based on this difficulty, a combined encoder and controller which exploited both the hard and soft information of the channel outputs were proposed. Simulations showed that the proposed scheme has good performance compared to the controllers which only used hard-information or ignored the information carried in the memory.

Chapter 5

Distributed Encoder and Controller Design for Feedback Control Systems Over BSC Using Q-Learning

Both Chapter 3 and Chapter 4 discuss the co-design encoder and controller by iterative design method, which means the encoder is optimized with fixed controller and the controller is optimized with fixed encoder. However, only local optimal solutions can be got by iterative design. Therefore, distributed encoder and controller design is proposed. Both encoder and controller learn independently with their own local information, while both of the agents can be optimized simultaneously. Obviously, the system performance is better than iterative design. However, global optimal can't be guaranteed. Distributed Q-learning (QD-learning) can be applied into complex networked control systems with multi-encoders or multi-controllers design.

5.1 Introduction

Reinforcement learning has been discussed in the literature review. A single agent learns what to do and find a policy (a mapping from state to action) that minimize or optimize the discounted infinite horizon in a stochastic environment. Since the framework of multiple agents (Figure 5.1), which is the same idea of the single agent, are needed in reality, there are many agents choosing actions over the environment [Sycara, 1998, Weiss, 1999, Durfee, 2001, Vlassis, 2003]. The biggest difference is that each agent has an influence on the environment, and all actions can have different effects depending on the communication between the agents. The objective of multi-agent reinforcement learning is to find the optimal stationary control policy to maximise or minimise the averaged one stage cost function.

The existence of multiple operating agents makes it possible to solve inherently distributed problems, but also allows one to decompose large problems, which are too complex or too expensive to be solved by a single agent, into smaller subproblems.

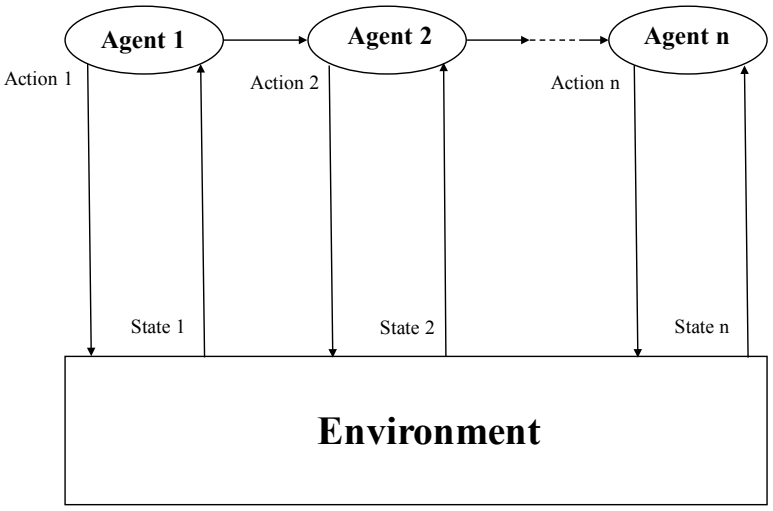

FIGURE 5.1: The multi-agent learning framework

Multi-agent reinforcement learning developed considerably in recent years, mainly because of the parallel computation and experience sharing which can help agents with similar tasks to learn faster and better. For instance,

- Agents can exchange information by communication, and skilled agents may serve as teachers for the learner, or the learner may watch and imitate the skilled agents.
- If one or more agents fail in a multi-agent system, the remaining agents could take over some of their tasks. This implies that multi-agent reinforcement learning is inherently robust.
- Most multi-agent systems also allow easy insertions of new agents into the system, leading to a high degree of scalability.

However, there are some challenges of multi-agent reinforcement learning

- the curse of dimensionality
  The curse of dimensionality is the exponential growth of the discrete state-action space with the number of state and action variables (dimensions).
- the goal of multi-agent reinforcement learning
  A suitable goal for multi-agent reinforcement learning in the general stochastic case is a difficult challenge, as the agents share information with each other, which means it is difficult to reach the optimal asymptotically.

There are three types of multi-agent reinforcement learning, which are competitive dynamic stochastic games [Littman], fully cooperative formulation [Shoham et al., 2003] and the combination of these two. In this chapter, only fully cooperative formulation will be considered.

- Nonlinear systems and online trade-off
  Non-stationarity of the multi-agent learning problem arises since all the agents in the system are learning simultaneously. Each agent is, therefore, facing with a moving-target learning problem: the best policy changes as the other agents policies change.

In multi-agent networked systems, the classical reinforcement learning can be used to set the problem within the location probability and a statistic environment. It requires the centralised controller to receive the data of one stage costs from all the agent continuously[Melo and Veloso, 2011]. Since the one-stage cost can only be generated on local process, all agents should forward their one-stage cost to the centralised controller, which is the main case of the unfeasible systems which increases the computation capacity.

To solve these problem, QD-learning has been introduced by Kar et al. [2013]. It is a fully distributed method, where all agents learn autonomous from local process and communication over a sparse possibly time-varying to minimise the network averaged infinite horizon discounted cost to find the optimal stationary control policy.

This chapter mainly discuss distributed reinforcement learning, which consists of multi-agents (actions) and the corresponding environment (state). Assumed that the environment is dynamic and uncertain, the actions of agents and the resulting states which follows on MDPs, have an influence on the statistical distribution of the random one-stage cost. The objective of the agent is to find an stationary control policy to minimise the networked-averaged infinite horizon discounted cost.

QD-learning tries to optimise global cost (the network averaged one stage cost) which is not directly observed by each agent. More specifically, at a given time instant, each agent is aware of only its local instantaneous parameters but the network averaged. Whilst as a fully cooperative formulation, QD-learning assumes that the global one-stage cost (network-average of the local instantaneous one-stage cost) is available for the agents at all times.

Compared to distributed Q-learning from existing literature review, the fully distributed setting in the network is considered, where the agents are only aware of local parameters and sense the local cost through their neighbourhood via communication over the control networks. However, QD-learning, as a fully cooperative formulation, does not emphasise the two issues, which are partial state observation and decentralised actuation. More specifically, we consider that each agent can observe the global state perfectly and the agents are set up with the local decentralised actuation.

5.2 Distributed Q-Learning

We describe QD-learning in this section, which is a distributed scheme for multi-agent Q-learning. The QD-learning method is similar to the Q-learning method, both of them are based on state-action trajectories. In general, the state-action trajectories are sample paths of stochastic processes $\{x_t\}$ and $\{u_t\}$ taking values in X and U, respectively. We randomly generate the action u_t and state x_t for the corresponding agents. Furthermore, QD-learning method also need to use the local one-stage cost $c_n(x_t,u_t)$ for each agent n. The goal of QD-learning is to ensure that each agent eventually learns the value function V^* based on the stochastic processes $\{x_t\}$ and $\{u_t\}$, and the one-stage cost processes $c_n(x_t,u_t)$.

This section is the main component of this chapter, which concerns the assumption about the constrains of QD-learning, the theorem of QD-learning and the algorithm of QD-learning.

5.2.1 Assumption

Formalising the distributed agent learning requires characterising the locally accessible agent information over time for decision-making.

5.2.1.1 Measurability and Moments

Assuming that there exists a complete probability space $(\Omega,\mathscr{F},\mathbb{P})$ with a filtration $\{\mathscr{F}_t\}$ ($\mathscr{F}_t$ denotes the global network information at each time instant t), such that the state and control processes,$\{x_t\}$ and $\{u_t\}$, respectively, are adapted to $\mathscr{F}_t$. The conditional probability law governing the controlled transitions of $\{x_t\}$ satisfies

$$\mathbb{P}(x_{t+1}=j|\mathscr{F}_t)=p^{u_t}_{x_{t,j}} \tag{5.1}$$

For each agent n

$$\mathbb{E}[c_n(x_t,u_t)|\mathscr{F}_t]=\mathbb{E}[c_n(x_t,u_t)|x_t,u_t] \tag{5.2}$$

which means $\mathbb{E}[c_n(i,u)]$ on the event $\{x_t=i,u_t=u\}$, such as both of the current state-action pair, the one-stage random costs are independent from $\mathscr{F}_t$.

The random cost $c_n(x_t,u_t)$ belongs to $\mathscr{F}_{t+1}$ for each t. In addition, the equation 5.1 and 5.2 use the state-action trajectories online and the corresponding costs are used to learn the value function, which satisfy the controlled Markov transitions in accordance with the MDP. Then $\mathscr{F}_t$ consists of all the state action pairs and the one stage cost function before time t. The function is

$$\mathscr{F}_t=\sigma(\{x_s,u_s\}_{s\leq t},\{c_n(x_s,u_s)\}_{n\in N,s<t}) \tag{5.3}$$

where σ represents the smallest $\sigma-algebra$, that is, combines all the sets.

$\mathscr{F}_n(t)$ is the local information available at each agent n at each time t. The local information $\mathscr{F}_n(t)$ at agent n that the agent n senses the one-stage cost and the messages or information it obtained from its neighbours over time, such as the instantaneous state and action. The local information $\mathscr{F}_n(t)$ at agent n at time t is

$$\mathscr{F}_n(t) = \sigma(\{x_s, u_s, \{m_{n,l}(s)\}_{l\in\Omega_n(s)}\}_{s\leq t}, \{c_n(x_s,u_s)\}_{n\in N, s<t}) \tag{5.4}$$

However, the difference between local information $\mathscr{F}_n(t)$ and global information $\mathscr{F}_t$ is the reward information. Local information consists of only the locally sensed cost, which consists of n elements, whereas the global information involves the sum-total network reward information from all agents at all times, which averaged all the agents one-stage cost.

$$\mathscr{F}_t = \bigvee_{n=1}^{N} \mathscr{F}_n(t) \tag{5.5}$$

where $\bigvee$ denotes the joint of $\sigma - algebras$.

Furthermore, assumed $m_{n,l}(t)$ ($m_{n,l}(t) \in \mathscr{F}_n(t)$) can be the message that agent n obtained from other agents l, $l \in \Omega_n(t)$ at time t, where $\Omega_n(t)$ denotes the communication over the neighbourhood of agent n on the time-varying (possibly stochastic) environment at time t.

The one-stage costs $c_n(i,u)$ is considered as a super-quadratic moments, there exists a constant ε_1 (choose randomly and sufficient small), while $\varepsilon_1 > 0$, such that

$$\mathbb{E}[c_n^{2+\varepsilon_1}(i,u)] < \infty, \quad \forall n,i,u \tag{5.6}$$

where i represents one sample of x_t.

5.2.1.2 Link Failures

Link failure, which can be regarded as spatially dependent, is a common failure mode in network systems. In fact, it is possible to have all these instantiations to be disconnected. Here, it only requires that the graph stays connected on average. This weak connectivity requirement enables us to capture a broad class of asynchronous communication models. Assumed that the inter-agent communication is noise-free and un-quantised.

5.2.1.3 Independence

The purpose of QD-learning is to learn the real value function V^* from communications among agents. There is a sequence of random times $\{T_{i,u}(k)\}$, in which $T_{i,u}(k)$ denotes the $(k+1)^{th}$

sampling instant of the state-action pair (i,u), such that

$$T_{i,u}(k) = \inf\{t \geq 0 | \sum_{s=0}^{t} \Lambda_{(x_t,u_t)=(i,u)} = k+1\} \tag{5.7}$$

Assuming that the random time $T_{i,u}(k)$ is a stopping time for each k and pair (i,u). Thus, for each k, the state-action pairs (x_t,u_t) , $T_{i,u}(k)$, qualifies as a stopping time following with the local filtrations $\mathscr{F}_n(t)$.

5.2.1.4 Finite Stopping Time

For each state-action pair (i,u) and each $k \geq 0$, the stopping time $T_{i,u}(k)$ is finite, which equals to $\mathbb{P}(T_{i,u}(k) < \infty) = 1$. The value function can only be updated until the current state action pair was chosen again assuming that this condition is also required by centralised Q-learning in application or simulation.

5.2.2 Updating Equation of QD-Learning

In QD-learning, each network agent n maintains a value sequence $\{Q_t^n\}$ (Q matrices) with the components $Q_{i,u}^n(t)$ for each possible state-action pair (i,u). The distributed equation likes

$$\begin{aligned} Q_{i,u}^n(t+1) = Q_{i,u}^n(t) - \beta_{i,u}(t) \sum_{l \in \Omega_n(t)} (Q_{i,u}^n(t) - Q_{i,u}^l(t)) + \\ + \alpha_{i,u}(t) \left(c_n(x_t,u_t) + \gamma \min_{u \in U} Q_{x_{t+1},u}^n(t) - Q_{i,u}^n(t) \right) \end{aligned} \tag{5.8}$$

where the weight sequence $\{\beta_{i,u}(t)\}$ and $\{\alpha_{i,u}(t)\}$ belong to $\mathscr{F}_n(t)$, which adapted the stochastic process for each pair (i,u):

$$\beta_{i,u}(t) = \begin{cases} \frac{b}{(k+1)^{\tau_2}} & \text{if } t = T_{i,u}(k) \text{ for some } k \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\alpha_{i,u}(t) = \begin{cases} \frac{a}{(k+1)^{\tau_1}} & \text{if } t = T_{i,u}(k) \text{ for some } k \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

where a and b are positive constants. The constants τ_1, τ_2 satisfies $\tau_1 \in (1/2, 1]$ and $\tau_2 \in (0, \tau_1 - 1/(2+\varepsilon_1))$, which need to be sufficiently small.

For each agent n, the random $T_{i,u}(k)$ is stopping times, which is a local information $\mathscr{F}_n(t)$. In addition, Q_t^l is the exchange information of agent n communicating with its neighbourhood. Letting

$$m_{n,l}(t) = Q_t^l, \quad l \in \Omega_n(t) \tag{5.11}$$

Based on equation 5.8, we can know that $Q^n_{i,u}(t)$ is updated at the instant time t for each agent n if and only if the current state-action (x_t, u_t) was chosen again, otherwise it stays constant.

The problem is Q-value updating. In previous work, $Q^n_{i,u}(t)$ will be updated at the instant time t for each agent n, if and only if the current state-action (x_t, u_t) is sampled again, which means the updating only happens when the current state-action pair equals to the next state-action pair $(s_t, a_t) = (s_{t+1}, a_{t+1})$. Otherwise it stays constant. Then the convergence rate is very slow.

However, based on the basic idea of Q-learning, Q-value should be updated on every iteration. More specifically, every time the state-action pair is sampled, the corresponding weight sequence $\alpha_{i,u}(t)$ and $\beta_{i,u}(t)$ will be updated, then Q-value would be updated using the corresponding weight sequence, there is no updating condition (current state-action pair equal to next state-action pair $((s_t, a_t) = (s_{t+1}, a_{t+1}))$.

Furthermore, each agent n owns the value process $\{V^n_t\}$, which converges to the optimal value function V^*. The i^{th} component of $\{V^n_t\}$, $V^n_i(t)$ defined as

$$V^n_i(t) = \min_{u \in U} Q^n_{i,u}(t), \quad i = 1, 2, 3, \cdots, M \tag{5.12}$$

For each agent n, $\{Q^n_t\}$ is adapted to the local filtration $\{\mathscr{F}_n(t)\}$. The update rule in equation 5.9 and 5.10 is the form of consensus + innovation, in which consensus represents all the agent can reach convergence at the same time stage. A local innovation potential is local sensing of the instantaneous cost.

5.2.2.1 The Algorithm of QD-Learning

In this section, we mainly discuss the algorithm of QD-learning based on the assumption and the theorem. In the initialisation part, we choose the initial Q value arbitrary, then use the episode as the repeat time step, the initialisation s and a should be randomly chosen in this step. There is another loop for each agent which embedded on the episode loop. For each agent, the next state and the reward should be observed. After that, the next action should be chosen based on the next state which correspond to the minimum Q value plus $\varepsilon - greedy$ to explore to guarantee that the global optimal can be obtained. Finally, the update part uses the equation 5.8. One should be aware of that the weight sequence can only be updated until the current state action pair was chosen again.

Then, the algorithm of QD-learning is:

Algorithm 5 QD-learning

```
Initialise Q(s,a) arbitrary
for t = 1 : Episode (Each time step) do
  Initial s and a
  for n = 1 : N (Each agent) do
    Observe one-stage cost function
    while 1 do
      Observe the next state s'
      Choose a' form s' (e.g. using ε − greedy to explore)
      if (s', a') equal to (s,a) then
        k = k + 1 update the weight sequence α^n_{i,u} and β^n_{i,u}
      end if
      Q^n_{i,u}(t) ← Q^n_{i,u}(t) − β_{i,u}(t) Σ_{l∈Ω_n(t)} (Q^n_{i,u}(t) − Q^l_{i,u}(t)) + α_{i,u}(t)(c_n(x_t,u_t) +
      γ min_{u∈U} Q^n_{x_{t+1},u}(t) − Q^n_{i,u}(t))
    end while
  end for
end for
```

This procedure of this algorithm is similar to single agent Q-learning, while the difference is that there are n agents in QD-learning, each agent should be repeated on one time step. Meanwhile in the updating part, there is a communication component. This algorithm will be terminated until there is no updating on Q-value.

5.2.2.2 Restriction of Convergence

The parameter of weight sequence $\tau_1 and \tau_2$ has been given. However, the stochastic sequence $\{\alpha_{i,u}(t)\}$ and $\{\beta_{i,u}(t)\}$ should satisfy the conditions, like

$$\sum_{t\leq 0} \alpha_{i,u}(t) \to \infty \qquad \sum_{t\leq 0} \beta_{i,u}(t) \to \infty \tag{5.13}$$

which can guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. This condition is a standard requirement in stochastic approximation type algorithms to drive the updates to the desired limit from arbitrary initial conditions. There is a further condition that the innovation weight sequences are square summable, such as

$$\sum_{t\leq 0} \alpha^2_{i,u}(t) < \infty \tag{5.14}$$

For each state-action pair, the consensus potential dominates the innovation potential eventually. Whereas the asymptotic domination of the consensus potential over the local innovations ensures the right information mixing, as shown below, leading to optimal convergence.

$$\frac{\beta_{i,u}(t)}{\alpha_{i,u}(t)} \to \infty, \qquad t \to \infty \tag{5.15}$$

In terms of the constant value a and b, it should be arbitrary positive constants. We further assume that the constants are small enough, such that, for each time instant t and state-action pair (i,u), the matrix $I_N - \beta_{i,u}(t)L_t - \alpha_{i,u}(t)I_N$ is non-negative. The largest eigenvalue of the Laplacian L_t, at an instant t, is upper-bounded by N (the number of network agents). The above condition is ensured by requiring a and b to satisfy :

$$a + Nb \leq 1 \tag{5.16}$$

However, this condition is not necessary, which can considerably reduce the analytical overhead.

5.3 Examples about QD-Learning

5.3.1 Parameters Setting

The network has 40 agents ($N = 40$) with binary-valued state and action spaces, while in this simulation, we consider that the basis of the state-action space $\mathscr{X} \times U$ is 4, that is, we have two states $\mathscr{X} = \{x_1, x_2\}$ and two actions $U = \{u_1, u_2\}$. Hence, there are 8 controlled transition parameters $p^u_{i,j}$, where $i, j \in \mathscr{X}$ and $u \in U$. The transition probability were chosen independently at random in the interval $[0,1]$, which means the other 4 transition probability are fixed since we just have two choices based on one state.

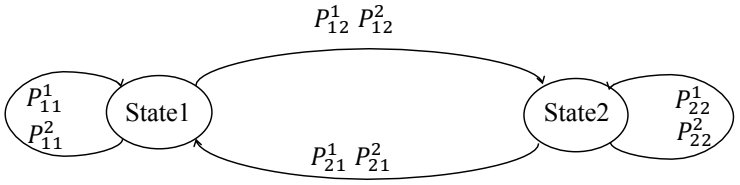

FIGURE 5.2: State-action framework

Figure 5.2 consists of two states (state 1 and state 2) and two actions (action 1 and action 2).

- P^1_{11} and $P^2_{11} \rightarrow$ the transition probability from state 1 to state 1 based on the action 1 and action 2, respectively;
- P^1_{12} and $P^2_{12} \rightarrow$ the transition probability from state 1 to state 2 based on the action 1 and action 2;
- P^1_{21} and $P^2_{21} \rightarrow$ the transition probability from state 2 to state 1 based on the action 1 and action 2;
- P^1_{22} and $P^2_{22} \rightarrow$ the transition probability from state 2 to state 2 based on the action 1 and action 2;

Furthermore, the state-action trajectory was generated independently and uniformly sampling control actions from U over time. Whereas, the state trajectory $\{x_t\}$ instantiation was generated by sampling independently x from the probability distribution $p_{x_t}^{u_t}$ of the past at each time t.

In terms of the cost function $c_n(i,u)$, it is assumed that it consists of two parts which are the one-stage cost part (expected one-stage cost value) and the noise part. As for the noisy part, it follows a Gaussian Distribution with variance 40, and another random sample of follows the uniform distribution on $[0,400]$, which is generated independently for each agent n and each state-action pair (i,u).

The discounting factor γ is 0.7. The inter-communication only happens between the 2-nearest neighbour, with the link-failure probability 0.5. In addition, we also assume that in the network, the $N = 40$ agents are placed on a circle symmetrically (see Figure 5.3), so that the agent can communicate with its neighbourhood on one side.

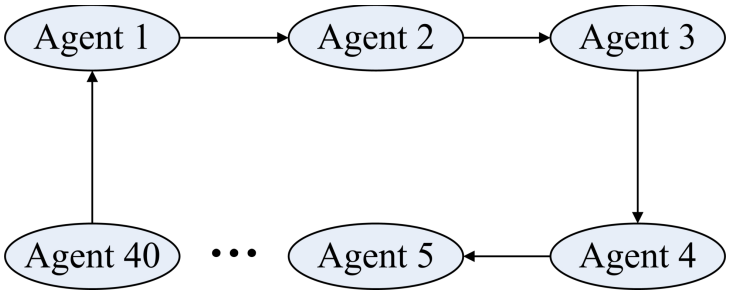

FIGURE 5.3: The set of agent network

The purpose of this simulation is to show the convergence of QD-learning. The centralised Q-learning is also simulated, which is similar to single agent Q-learning. The centralise Q-function $Q_{i,u}^n(t)$ is

$$Q_{i,u}^c(t+1) = Q_{i,u}^c(t) + \alpha_{i,u}(t)(\frac{1}{N}\sum_{n=1}^{N} c_n(x_t,u_t) + \gamma \min_{u\in U} Q_{x_{t+1},u}^c(t) - Q_{i,u}^c(t)) \tag{5.17}$$

where the exponent τ_2 which is the consensus weight sequence $\{\beta_{i,u}(t)\}$ (in QD-learning) was set to 0.2, while the innovation weight sequence $\{\alpha_{i,u}(t)\}$ (for both distributed Q-learning and centralised Q-learning) exponent τ_1 was set to 1.

Here we use the algorithm 5 to solve it. At the beginning, the one stage cost function follows Gaussian distribution, with a uniform distribution disturb. The expected part of one stage cost function should be generated. In addition, the transition probability which used to choose next state, is generated randomly. The purpose is to fix the onstage cost function and the transition probability once it is generated. It should be noticed that the weight sequence can only be updated until the current state action pair is chosen again.

5.3.2 Simulations

There are some simulation results, the figure plotted here is about the Q-factors (Q value) of the four state action pairs, which are $(1,1),(1,2),(2,1),(2,2)$, changing with episodes, respectively. The purpose is to show the convergence rate of each agent on each episode and the comparison of the centralised Q-learning algorithm and QD-learning method.

Figure 5.4 illustrates that the convergence rate of distributed Q-learning is reasonably close to centralised Q-learning.

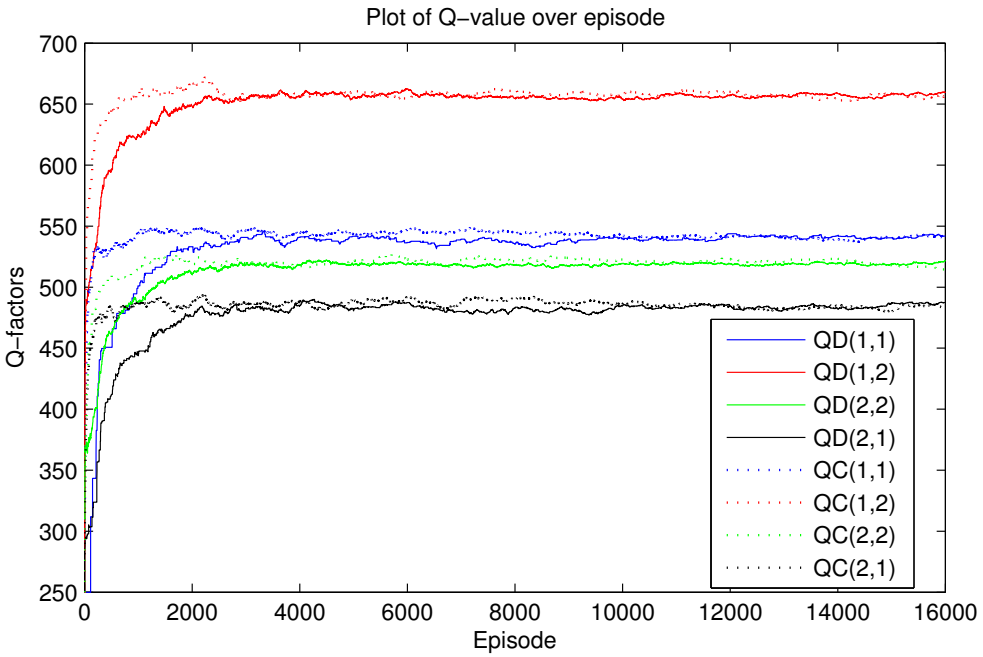

FIGURE 5.4: Centralised Q-learning (dotted lines) and distributed Q-learning (solid lines)

Figure 5.5 describes that the evolution of the Q-factors at 10 randomly (uniformly) selected network agents (for the distributed QD), verifying that they reach consensus on each state-action pair (i,u).

Figure 5.6 and 5.7 show that the learning-rate $\alpha_{i,u}(t)$ is changed, the convergence rate is different. In Figure 5.6, the convergence rate changes when the parameter of learning rate a changed ($a = 0.6, 0.3$ respectively). Figure 5.7 shows more result about the changing of convergence rate, in which $a = 0.25, 0.5$ and 0.75, respectively.

Figure 5.8 shows the consensus rate $\beta_{i,u}(t)$ has no influence in the convergence rate. More specifically, in Figure 5.8, the parameter of consensus rate is changed, which are $b = 0.001, 0.01$ and 0.2, respectively, while the convergence rate remains the same.

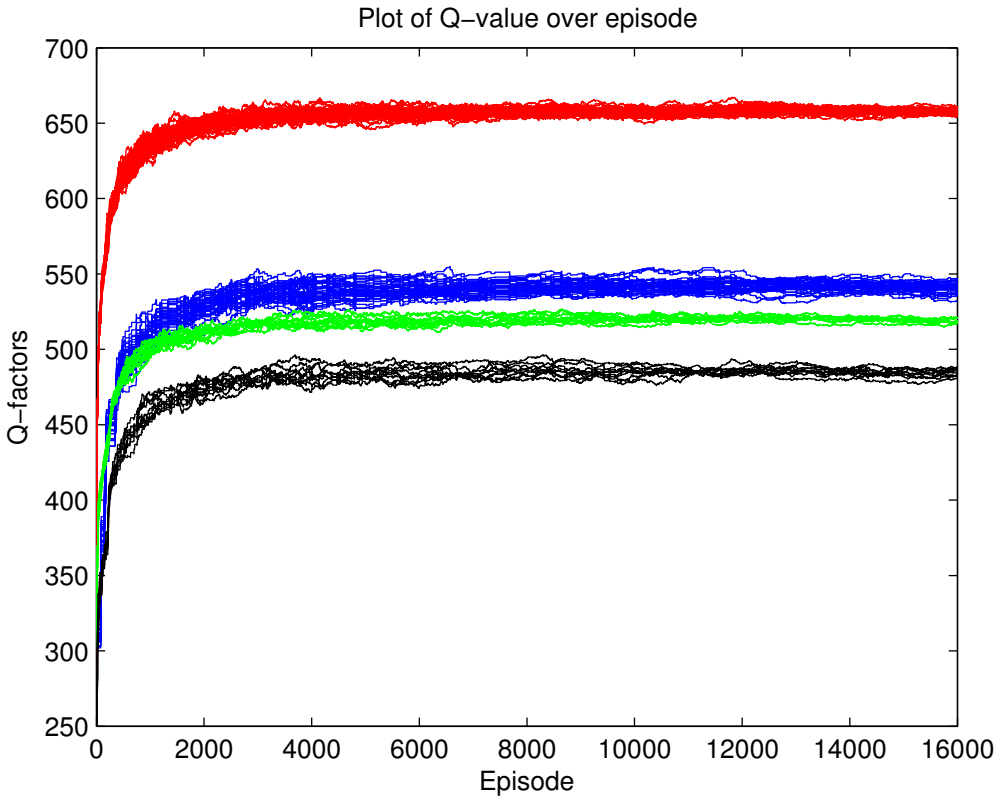

FIGURE 5.5: Consensus among distributed Q-factors

$\beta_{i,u}(t)$ is the consensus rate, which describes level of consistency of different agents in collaboration. It affects the convergence rate of each agent, which means the convergence rate of each agent becomes more consistent with the increase of $\beta_{i,u}(t)$.

In Figure 5.8, the red and blue lines represent the convergence of Q-factor for all the agents in one state-action pair (1,1). The $\beta_{i,u}(t)$ of the red line is larger than the blue line. The figure shows that the convergence rates of the agents represented by the red lines are less diverse than those of blue lines.

Figure 5.9 presents the trend of one agent's variance over episode. It shows that if the parameter of consensus rate b increases, the fluctuation will decrease.

5.4 Distributed Encoder-Controller Design

Our study of quantized control systems is motivated by the challenges of control over wireless networked systems, given limited communication resources. Clearly, besides the problem of optimizing the performance for each individual plant, another major challenge is to coordinate all distributed control nodes to provide a satisfactory overall performance. In a networked system, it can happen that several plants are communicating simultaneously which may give congestion

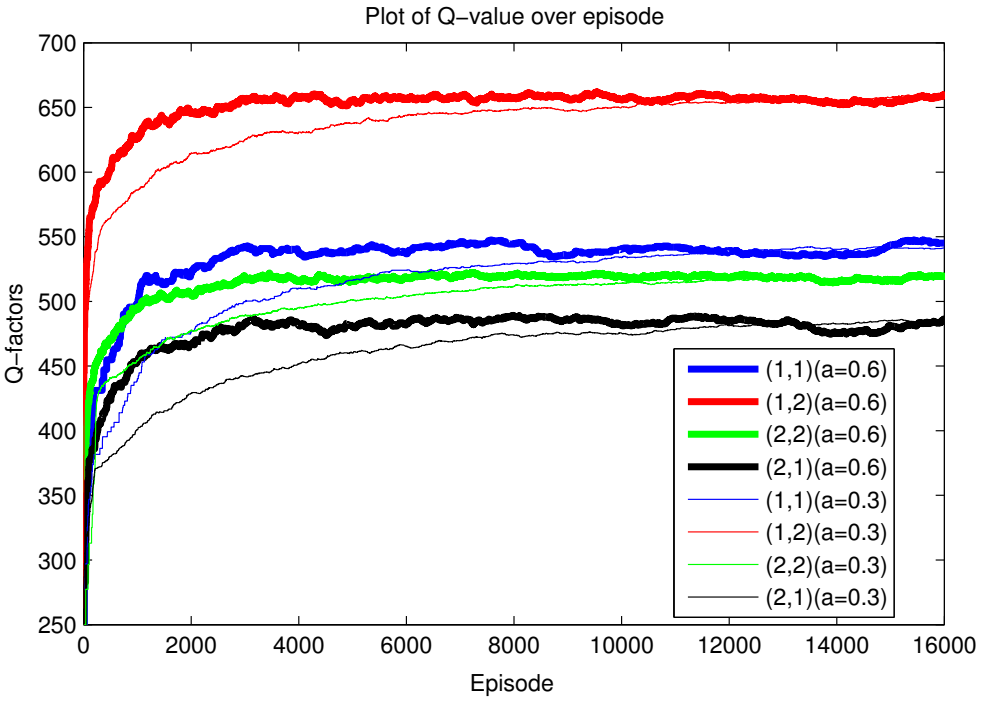

FIGURE 5.6: Changing the learning-rate $\alpha_{i,u}(t)$

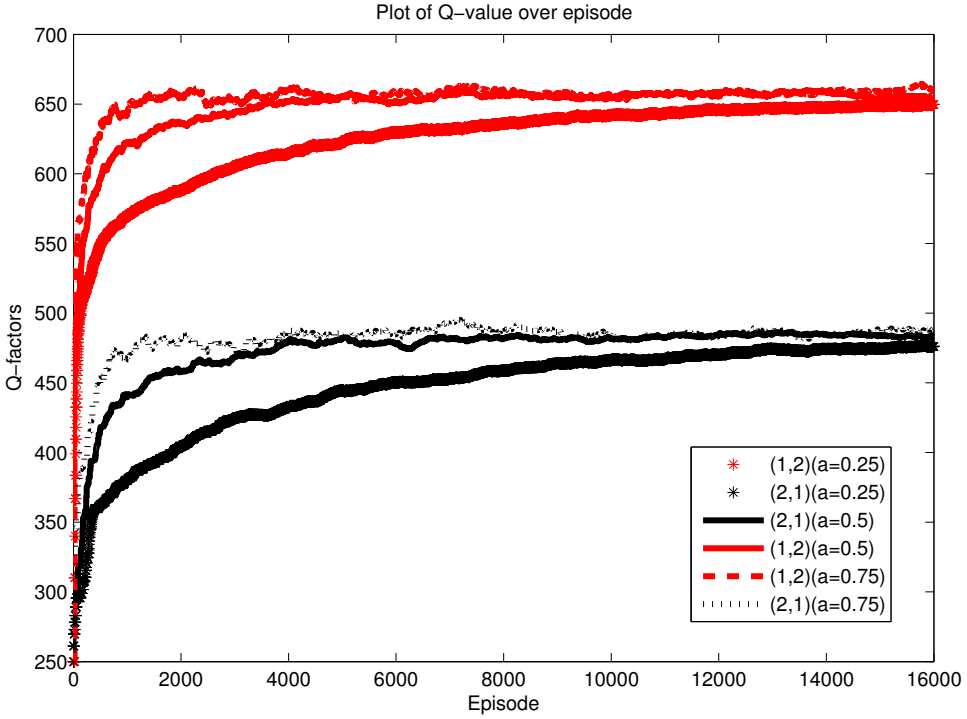

FIGURE 5.7: Changing the learning-rate $\alpha_{i,u}(t)$

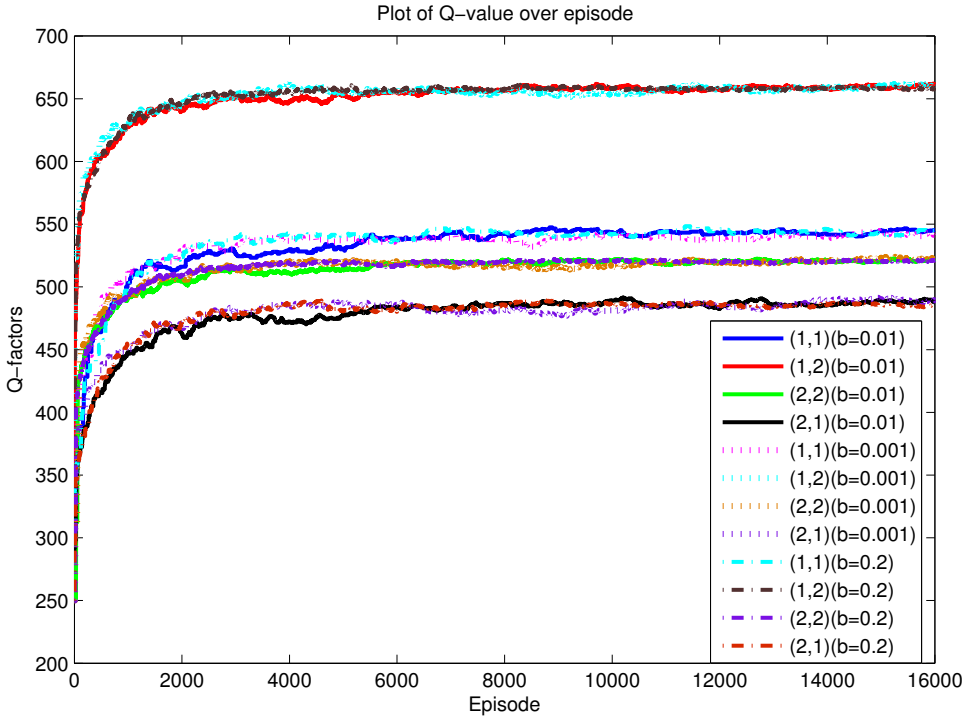

FIGURE 5.8: Changing the consensus-rate $\beta_{i,u}(t)$

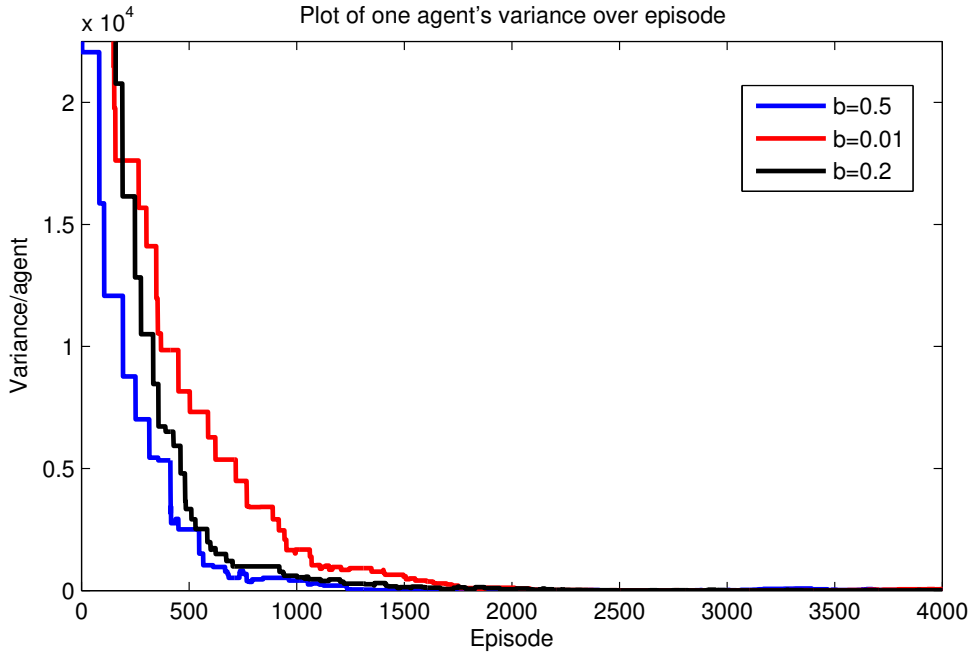

FIGURE 5.9: The variance of one agent over episode

and interference. How to design encoder and controller pairs that the system performance can be minimized, so that the system performance is better than the design by iterative method.

My research is on network control system design. The main purpose is to minimize the system performance which can be interpreted as minimizing the state variance for all time instances with a power constraint on the control input. Traditional control optimization methods are model-based and implementable in complex network control systems (nonlinearity and non-convexity). Adaptive dynamic programming, combining dynamic programming, reinforcement learning and functional approximation, makes use of data that is measured along the system trajectories to optimize the controller. However, encoder strategies in network control system have a significant effect on system performance. Hence, Q-learning, a reinforcement learning method, is introduced to find encoder strategies. Iterative design method updates each iterate by optimizing the encoder with fixed controller and optimizing the controller with fixed encoder alternatingly. However, only local optimal result can be guaranteed. To compensate for this, distributed learning method is proposed to optimize encoder and controller simultaneously, with achieving a better performance.

In distributed design, encoder and controller are two agents in the network control system. Each agent only accepts local information and utilizes the local information to make decisions. While the design objective is minimizing the system performance, which is $\mathbb{E}\{\sum_{t=0}^{T-1} x_{t+1}^T V_t x_{t+1} + u_t^T R_t u_t\}$, matrix V_t and R_t are symmetric and positive definite. The reward of these two agents are same, which is $x_{t+1}^T V_t x_{t+1} + u_t^T R_t u_t$. Fig. 5.10 shows the design process. Both encoder and controller update their Q-value matrix separately and utilize same reward. In distributed design, Q-value matrix of encoder is updated for each time running l. while the Q-value of controller is not updated for each l, because least square method is used to update control gain k_t. More specifically, control gain is updated over a time slot.

In Fig. 5.10, η is the episode, l is each time step of one episode and t is system time step. Based on the optimized encoder and controller, the expected overall cost can be calculated straightforwardly.

5.5 Numerical Results

The system parameters for simulation are $A = \begin{pmatrix} 0.99 & 0.3 \\ 0.8 & -0.7 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $C = I_2$. The time horizon is $T = 3$ and the weighting matrices in J_T are $E = 0.1I_1$ and $D = 4I_2$. The initial state x_0, process noisee_t and measurement noise e_t follow Gaussian distribution with mean 0 and variance $\begin{pmatrix} 10 \\ 10 \end{pmatrix}$, $\begin{pmatrix} 1e-5 \\ 1e-5 \end{pmatrix}$, $\begin{pmatrix} 1e-5 \\ 1e-5 \end{pmatrix}$, respectively.

The transmission rate used for simulation is $\rho = 2$, and the number of binary codeword is $2^\rho = 4$, which are $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Consequently, the number of encoder regions for the second order system 16.

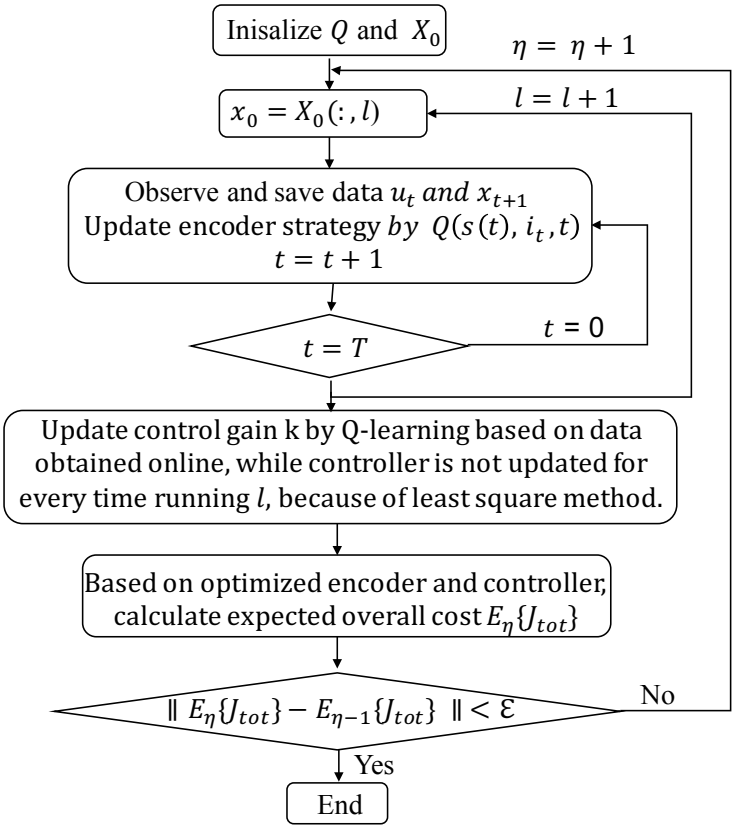

FIGURE 5.10: The flowchart of distributed encoder and controller design

Fig. 5.11 shows the encoder region evolution for one specific code $i_t = 4$ from time $t = 1$ to $t = 10$. These 10 figures have same horizon axis, named $x_t(1)$ and same vertical axis $x_t(2)$. All the points in the figure correspond to the same $i_t = 4$ and its corresponding binary code is $[0 \quad 0 \quad 1 \quad 1]$. It is clearly that the shape of encoder regions from $t = 6$ to $t = 10$ are quite similar, and it can be regarded as convergence.

Fig. 5.12 illustrates the evolution of Q-value with 16 encoder regions at $t = 0$. The horizon axis is episode η and the vertical axis is Q-value. We randomly choose one state-action pair (y_t, i_t) of each encoder region. For each state-action pair, the Q-value evolution converges to a fixed value. Other pairs have a similar convergence.

The expected overall cost $\mathbb{E}(J_{tot})$ is calculated based on the optimized encoder and controller mappings, and the comparison results are depicted in Fig 5.13.

Obviously, the system performance of distributed design is better than iterative design with low crossover probability. However, the indeterminacy of encoder and controller increase simultaneously with high crossover probability, the system performance is deteriorated, and even worse than iterative design. Because encoder is optimized with fixed controller in iterative, which can decrease the uncertainty.

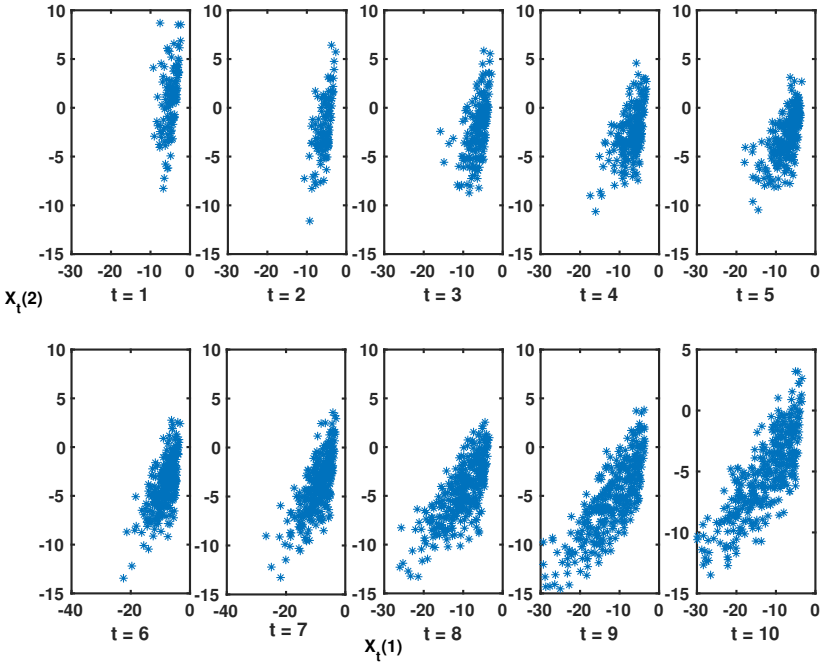

FIGURE 5.11: The evolution of encoder region with $i_t = 4$ over time interval from $t = 1$ to $t = 10$ and its corresponding binary code is $[0 \quad 0 \quad 1 \quad 1]$.

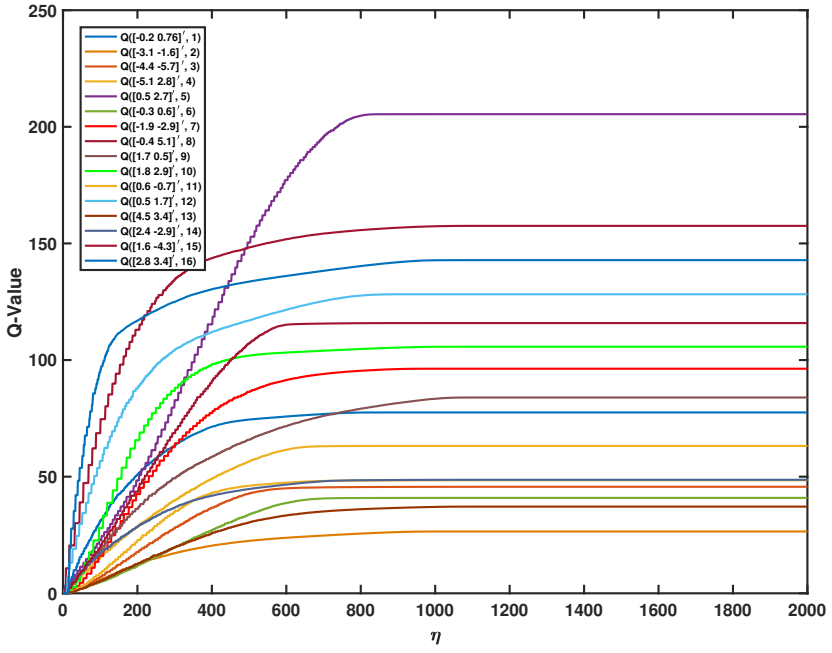

FIGURE 5.12: Randomly choose one state action pair (y_t, i_t) from 16 encoder regions and plot its Q-value evolutions at $t = 0$.

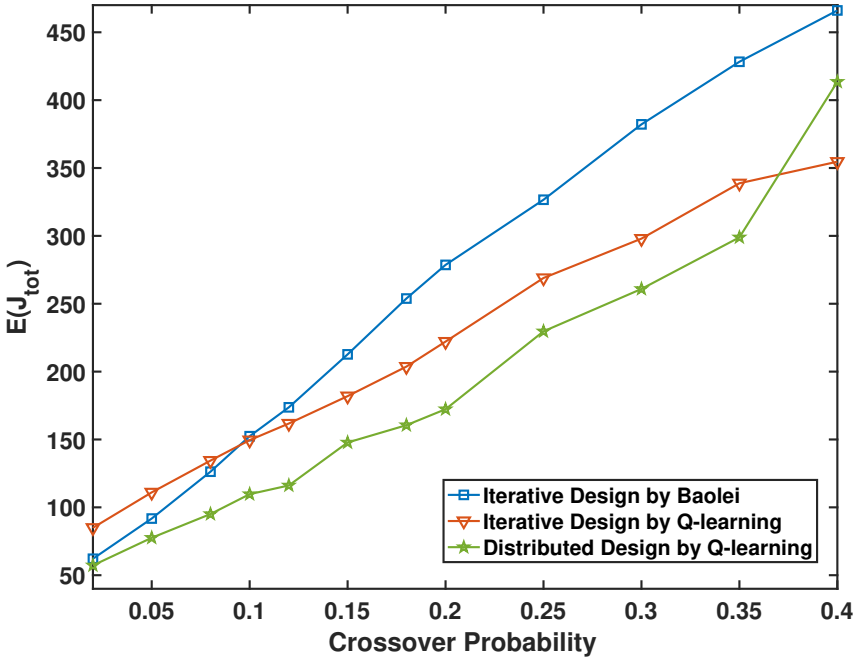

FIGURE 5.13: Performance Comparison

5.6 Conclusions

This Chapter illustrates a distributed multi-agent Q-learning learning setup in a networked environment, where the multi-agent are cooperative and non-competitive with the global objective. More specifically, compare with a centralised solution methodology requires each network agent to forward its instantaneous one-stage cost to a remote centralised controller at all times. All the agents try to learn and evaluate the optimal stationary control strategy to minimises the network-average in finite horizon discounted one-stage costs, that is, the network agents engage in network processing (learning) by mean of local communication and computation.

The result of distributed Q-learning shows that all the agents can achieve optimal learning performance asymptotically, which means the network agents reach consensus on the desired value function and the corresponding optimal control strategy, under minimal connectivity assumptions on the underlying communication graph. We have given an example on section 5.3, which proves that the convergence rate of the proposed distributed QD-learning is reasonably close to the centralised Q-learning. However, it is on low dimensional environment. It was argued that the convergence rate of the distributed Q-learning should asymptotically approach that of the centralised in more general scenarios (higher dimensional setups) due to the asymptotic domination of the consensus potential over the innovations, which is an important future direction. It may consist of analytically characterising the convergence rate of QD-learning under further assumptions on the state-action generation, for instance, by imposing specific statistical structure on the simulated state-action pairs.

After that, we formulated a distributed design problem to optimist the encoder and controller asymptotically. The based system is closed-loop control of a linear plant with a low-rate feedback link over a memoryless noisy channel. Compared with iterative design method, fixed encoder or controller is no need in distributed design method, which means encoder and controller learn online and update its corresponding policy. Based on the simulation results, the system performance of distributed design method is better than iterative design method. While this is only two agents, the system can be expand to multi-sensors, which is the trend of complex network control system.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis, we have discussed some fundamental knowledge about optimal control of networked systems using reinforcement learning. Many studies move to feedback control system by using measurement feedback over wireless communication channels. Because of the importance and popularity of industrial wireless networking, research on designing closed-loop systems for control using measurement feedback over imperfect communication channels has received increasing attention. It can be claimed, however, that the research area is still in its infancy and it evolves rapidly.

As explained in the introduction chapter, considerable efforts have been devoted to various stability issues for quantized control systems. This thesis, on the other hand, focuses mainly on the optimization of the overall system performance. In particular, a stochastic control problem was formulated and several aspects of the design and analysis of encoder-controllers for control over low-rate noisy channels were studied. To deal with one of the most fundamental problems in control systems, namely how to make the best use of the feedback information for the future evolution, the encoder and controller are required to perform efficient estimation and control using a few bits per sensor measurement. Because of the complex relation to all past and future events, optimal estimation and control are difficult tasks.

In the main component, we studied the problem of optimizing the encoder-controller jointly, by using an iterative training approach. The basic principle is to alternate between the optimization of the encoder mappings and the controller mappings. Q-learning technique is utilized over the design process, which means the encoder and controller mapping are produced by Q-learning method. It is worth to point out that Q-learning method is model free, only data measured along the system trajectories are utilized. It is a big improvement of co-design encoder-controller. Compared with classic dynamic programming method, the design process in this thesis is online, which means the updating process of encoder and controller design follows on system running.

Furthermore, multi-input and multi-out system is selected to show the results, while most of the current research works about encoder and controller design can only be applied into scalar case.

For the co-design with BSC, three encoder strategies are given, which are memoryless encoder design, memory encoder design and predictive encoder design. The previous two can use online learning method, while system model is necessary into predictive encoder design. By simulation experience, the system performance generated by predictive encoder design is a little bit better than the other two. For the controller design component, we first show that the control policy g_t and Q-value Q_t could converge to g_t^* and Q_t^*, respectively. What's more, the optimal control policy which is obtained by a virtual system, named open-loop encoder system, is given to help theoretical analysis.

For the co-design with Gaussian channel, the main challenge is that the channel output is real-valued vector. It is difficult or even impossible to get solutions by using method with BSC design, because the trained encoder-controller can no longer be implemented as a simple look-up table. Hadamard matrix is utilized to help to imitate the encoder process, invert this process and then the real-valued vector channel output can be decoded to discrete symbols. This process is soft controller design. While it is a bit complex to apply into practical, hard controller is proposed. Hard controller design is same with controller design with BSC. Only partial information is utilized for system optimization. Hence, a combination of soft-hard controller is proposed. Three controller design process are given in this chapter, which are soft-information based controller, hard-information based controller and combined soft-hard information based controller.

In the previous two chapter, iteration encoder and controller design is applied in both co-design with BSC and Gaussian channel, while only local optimal results are obtained. Distributed encoder and controller design is proposed to realize global optimal, while we can't guarantee global optimal solution. Clearly, the system performance of distributed design is better than iterative design. In addition, QD-learning is introduced and simulations are given. All agents learn independently and have communication with others. It means only partial information is used to estimate. By the example, distributed Q-learning can converge to centralised Q-learning. This is a big progress of multi-agent system, which can be applied in the system with multi-sensors, multi-encoders and multi-controllers. Finally, simulation results are given for system with BSC. Comparing results from simulations, distributed encoder and controller design is better than iteration co-design.

6.2 Future Research

The upcoming research includes the following:

1. Finish Distributed Encoder-Controller Design
   In Chapter 5, only distributed encoder and controller with BSC is given, that's not enough.

So that different system model will be given. Also, the theoretical analysis is necessary and can make sure the system performance of distributed design is better than iterative design.

Furthermore, it is interesting to explore complexity networked control system with multi-sensors, multi-encoders and multi-controllers. QD-learning is given already and can be applied into multi-encoders optimization.

2. High-Order System Design
   In high order system, the biggest challenge is how to design encoder to overcome the curse of dimensional. The shunting idea is proposed, which means that the encoder can be splite into shunting encoder. Each splitting encoder component can be viewed as one agent. All agents learns independently by their own information and communication is allowed among agents. Obviously, QD-learning method complies with the design requirement.

3. Application
   After all these simulation test, it is better to find an area to apply the research works proposed in this thesis. For example, wireless communication and control is very important for UAV(unmanned aerial vehicle). How to improving system performance with simple devices, such as low storage capacity sensors and low transmission power channel, is very important to ensure long lifetime. Distributed design can be used into complexity environment. Also, we can utilize our research on man-machine design.

Bibliography

P. Antsaklis and J. Baillieul. Guest editorial special issue on networked control systems. *IEEE Transactions on Automatic Control*, 49(9):1421–1423, 2004.

P. Antsaklis and J. Baillieul. Special issue on technology of networked control systems. *Proceedings of the IEEE*, 95(1):5–8, 2007.

M. Aoki. *Optimization of stochastic systems: topics in discrete-time systems*, volume 32. Academic Press, 1967.

B. Azimi-Sadjadi. Stability of networked control systems in the presence of packet losses. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 676–681. IEEE, 2003.

L. Bao and M. Skoglund. Encoder-controller design for control over the binary-input gaussian channel. In *Spread Spectrum Techniques and Applications (ISITA), 2010 IEEE 11th International Symposium on*, pages 23–28. IEEE, 2010.

L. Bao, A. Shirazinia, and M. Skoglund. Iterative encoder-controller design based on approximate dynamic programming. In *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, pages 1–5. IEEE, 2010.

M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5): 834–846, 1983.

R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

R. Bellman and R. E. Kalaba. *Dynamic programming and modern control theory*, volume 81. Citeseer, 1965.

R. E. Bellman and R. E. Kalaba. *Selected papers on mathematical trends in control theory*. Dover Publications, 1964.

D. P. Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

S. J. Bradtke, B. E. Ydstie, and A. G. Barto. Adaptive linear quadratic control using policy iteration. In *American Control Conference, 1994*, volume 3, pages 3475–3479. IEEE, 1994.

J. H. Braslavsky, R. H. Middleton, and J. S. Freudenberg. Feedback stabilization over signal-to-noise ratio constrained channels. *IEEE Transactions on Automatic Control*, 52(8):1391–1403, 2007.

W. L. Brogan. *Modern control theory*. Pearson education india, 1991.

Z. Bubnicki and Z. Bubnicki. *Modern control theory*, volume 2005925392. Springer, 2005.

G. Cena and A. Valenzano. Achieving round-robin access in controller area networks. *IEEE Transactions on Industrial Electronics*, 49(6):1202–1213, 2002.

M. B. Cloosterman, N. Van de Wouw, W. Heemels, and H. Nijmeijer. Stability of networked control systems with uncertain time-varying delays. *IEEE Transactions on Automatic Control*, 54(7):1575–1580, 2009.

T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

L. G. Crespo and J.-Q. Sun. Stochastic optimal control via bellman's principle. *Automatica*, 39 (12):2109–2114, 2003.

P. Dayan and B. W. Balleine. Reward, motivation, and reinforcement learning. *Neuron*, 36(2): 285–298, 2002.

D. F. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE transactions on automatic control*, 35(8):916–924, 1990.

J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

E. H. Durfee. Distributed problem solving and planning. In *Multi-agent systems and applications*, pages 118–149. Springer, 2001.

A. El Gamal and Y.-H. Kim. *Network information theory*. Cambridge university press, 2011.

F. Fagnani and S. Zampieri. Stability analysis and synthesis for scalar linear systems with a quantized feedback. *IEEE Transactions on automatic control*, 48(9):1569–1584, 2003.

N. Farvardin. A study of vector quantization for noisy channels. *IEEE Transactions on Information Theory*, 36(4):799–809, 1990.

J. Freudenberg and R. Middleton. Stabilization and performance over a gaussian communication channel for a plant with time delay. In *2009 American Control Conference*, pages 2148–2153. IEEE, 2009.

J. S. Freudenberg, R. H. Middleton, and V. Solo. Stabilization and disturbance attenuation over a gaussian communication channel. *IEEE Transactions on Automatic Control*, 55(3):795–799, 2010.

M. Fu and L. Xie. Finite-level quantized feedback control for linear systems. *IEEE Transactions on Automatic Control*, 54(5):1165–1170, 2009.

R. G. Gallager. *Information theory and reliable communication*, volume 588. Springer, 1968.

A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

A. L. Glasser. The evolution of a source code control system. *ACM SIGSOFT Software Engineering Notes*, 3(5):122–125, 1978.

R. M. Gray. *Source coding theory*, volume 83. Springer Science & Business Media, 2012.

J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

S. Kar, J. M. Moura, and H. V. Poor. Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+ innovations. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, 2013.

P. Knagenhjelm and E. Agrell. The hadamard transform-a tool for index assignment. *IEEE Transactions on Information Theory*, 42(4):1139–1151, 1996.

S. M. Kuo and D. R. Morgan. *Active noise control systems*, volume 4. Wiley, New York, 1996.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

E. B. Lee and L. Markus. Foundations of optimal control theory. Technical report, Minnesota Univ Minneapolis Center For Control Sciences, 1967.

B. Lei, M. Skoglund, and K. H. Johansson. Iterative encoder-controller design for feedback control over noisy channels. *IEEE Transactions on Automatic Control*, 56(2):265–278, 2011.

F. Lewis, S. Jagannathan, and A. Yesildirak. *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.

F. L. Lewis and D. Liu. *Reinforcement learning and approximate dynamic programming for feedback control*, volume 17. John Wiley & Sons, 2013.

F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE circuits and systems magazine*, 9(3), 2009.

F. L. Lewis, D. Vrabie, and V. L. Syrmos. *Optimal control*. John Wiley & Sons, 2012a.

F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, 32(6):76–105, 2012b.

Y. Li, E. Tuncel, and J. Chen. Optimal tracking over an additive white noise feedback channel. In *2009 7th Asian Control Conference*, pages 501–506. IEEE, 2009.

Z.-H. Li and M. Krstić. Optimal design of adaptive tracking controllers for non-linear systems. *Automatica*, 33(8):1459–1473, 1997.

C.-T. Lin and C. S. G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on computers*, 40(12):1320–1336, 1991.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 94.

D. Liu and Q. Wei. Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3): 621–634, 2013.

B. Luo, D. Liu, and T. Huang. Q-learning for optimal control of continuous-time systems. *arXiv preprint arXiv:1410.2954*, 2014.

D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

M. Mazo and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Transactions on Automatic Control*, 56(10):2456–2461, 2011.

F. S. Melo and M. Veloso. Decentralized mdps with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

R. H. Middleton, A. J. Rojas, J. S. Freudenberg, and J. H. Braslavsky. Feedback stabilization over a first order moving average gaussian noise channel. *IEEE Transactions on automatic control*, 54(1):163–167, 2009.

G. E. Monahan. State of the arta survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

L. A. Montestruque and P. Antsaklis. Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*, 49(9):1562–1572, 2004.

G. N. Nair and R. J. Evans. Stabilization with data-rate-limited feedback: Tightest attainable bounds. *Systems & Control Letters*, 41(1):49–56, 2000.

G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans. Feedback control under data rate constraints: An overview. *Proceedings of the IEEE*, 95(1):108–137, 2007.

Y. Niu, T. Jia, X. Wang, and F. Yang. Output-feedback control design for ncss subject to quantization and dropout. *Information Sciences*, 179(21):3804–3813, 2009.

M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam. The wireless control network: A new approach for control over networks. *IEEE Transactions on Automatic Control*, 56(10): 2305–2318, 2011.

P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson. Wireless network design for control systems: A survey. *IEEE Communications Surveys & Tutorials*, 20(2):978–1013, 2018.

N. J. Ploplys, P. A. Kawka, and A. G. Alleyne. Closed-loop control over wireless networks. *IEEE control systems*, 24(3):58–71, 2004.

W. K. Pratt, J. Kane, and H. C. Andrews. Hadamard transform image coding. *Proceedings of the IEEE*, 57(1):58–68, 1969.

M. Rabi, C. Ramesh, and K. H. Johansson. Separated design of encoder and controller for networked linear quadratic optimal control. *SIAM Journal on Control and Optimization*, 54 (2):662–689, 2016.

M. J. Rochkind. The source code control system. *IEEE transactions on Software Engineering*, (4):364–370, 1975.

W. Ryan and S. Lin. *Channel codes: classical and modern*. Cambridge university press, 2009.

M. Sabin and R. Gray. Global convergence and empirical consistency of the generalized lloyd algorithm. *IEEE Transactions on information theory*, 32(2):148–155, 1986.

A. Sahoo, H. Xu, and S. Jagannathan. Neural network-based event-triggered state feedback control of nonlinear continuous-time systems. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):497–509, 2016.

S. Shakkottai, R. Srikant, et al. Network optimization and control. *Foundations and Trends® in Networking*, 2(3):271–379, 2008.

A. Shirazinia, S. Chatterjee, and M. Skoglund. Joint source-channel vector quantization for compressed sensing. *IEEE Transactions on Signal Processing*, 62(14):3667–3681, 2014.

A. Shirazinia, A. A. Zaidi, L. Bao, and M. Skoglund. Dynamic source–channel coding for estimation and control over binary symmetric channels. *IET Control Theory & Applications*, 9(9):1444–1454, 2015.

Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: a critical survey. *Web manuscript*, 2003.

Z. Shu and R. H. Middleton. Stabilization over power-constrained parallel gaussian channels. *IEEE Transactions on Automatic Control*, 56(7):1718–1724, 2011.

M. Skoglund. Soft decoding for vector quantization over noisy channels with memory. *IEEE Transactions on Information Theory*, 45(4):1293–1307, 1999.

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

D. A. Sofge and D. A. White. Neural network based process optimization and control. In *29th IEEE Conference on Decision and Control*, pages 3270–3276. IEEE, 1990.

N. Sprague and D. Ballard. Multiple-goal reinforcement learning with modular sarsa (0). 2003.

R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

K. P. Sycara. Multi-agent systems. *AI magazine*, 19(2):79, 1998.

A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

S. Tatikonda and S. Mitter. Control over noisy channels. *IEEE transactions on Automatic Control*, 49(7):1196–1201, 2004.

S. Tatikonda, A. Sahai, and S. Mitter. Stochastic linear control over a communication channel. *IEEE transactions on Automatic Control*, 49(9):1549–1561, 2004.

N. Vlassis. A concise introduction to multi-agent systems and distributed AI. 2003.

G. C. Walsh, H. Ye, and L. G. Bushnell. Stability analysis of networked control systems. *IEEE transactions on control systems technology*, 10(3):438–446, 2002.

F. Wang, B. Chen, C. Lin, J. Zhang, and X. Meng. Adaptive neural network finite-time output feedback control of quantized nonlinear systems. *IEEE Transactions on Cybernetics*, 48(6): 1839–1848, 2018.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

Q. Wei, D. Liu, and H. Lin. Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. *IEEE Transactions on cybernetics*, 46(3):840–853, 2015.

G. Weiss. *Multi-agent systems: a Modern approach to distributed artificial intelligence*. MIT press, 1999.

T. C. Yang. Networked control system: a brief survey. *IEE Proceedings-Control Theory and Applications*, 153(4):403–412, 2006.

A. A. Zaidi, T. J. Oechtering, S. Yüksel, and M. Skoglund. Stabilization of linear systems over gaussian networks. *IEEE Transactions on Automatic Control*, 59(9):2369–2384, 2014.

L. Zhang, M. Z. Chen, C. Li, and Z. Shu. Event-triggered control over noisy feedback channels. *IFAC Proceedings Volumes*, 47(3):10493–10498, 2014.

W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE control systems magazine*, 21(1):84–99, 2001.

Q. Zhao, H. Xu, and S. Jagannathan. Near optimal output feedback control of nonlinear discrete-time systems based on reinforcement neural network learning. *IEEE/CAA Journal of Automatica Sinica*, 1(4):372–384, 2014.

Q. Zhao, H. Xu, and J. Sarangapani. Finite-horizon near optimal adaptive control of uncertain linear discrete-time systems. *Optimal Control Applications and Methods*, 36(6):853–872, 2015.

K. M. Zuberi and K. G. Shin. Design and implementation of efficient message scheduling for controller area network. *IEEE transactions on computers*, 49(2):182–188, 2000.