Reinforcement-learning based control for nonlinear systems

Shi, Qian

Awarding institution:
King's College London

A thesis submitted for the degree of
Doctor of Philosophy

Reinforcement-Learning based Control for Nonlinear Systems

Author : Qian Shi
Student Number : 1637690
First Supervisor : Dr. Hak-Keung Lam
Second Supervisor : Dr. Hongbin Liu

30/12/2021

Ph.D. in Robotics
Department of Engineering
Faculty of Natural & Mathematical Sciences
King's College London

Acknowledgment

I would like to express my sincere gratitude to the people who have helped me with the completion of the PhD thesis.

Firstly, please allow me to express the greatest appreciation to my supervisor Dr. Hak-Keung Lam, who guided me through my whole PhD study with professional experience and suggestion. Many thanks for his patient help with the revision of this thesis and the solid support in my research works carried out in the past four years, without which the thesis could not be completed on my own. His outstanding research ability, strong enthusiasm and disciplined attitude in the academia give me great inspiration of which I wish to pursue in my future career.

I would also like to express my thanks to China Scholarship Council for financially support my PhD study and King's College London for providing the advanced research conditions. Besides, many thanks to my colleagues and my friends for providing me great support through the PhD life.

Finally, I would like to say a huge thank you to my family for the understanding and support. Especially, I would like to express my special appreciation to my mother, Shuli Xue, for everything she has done for me and accompanying me through such a long way.

Abstract

In control theory, the development and analysis of control systems usually have high dependency on the mathematical model of the dynamic system. Variety of conventional control strategies have been proposed based on the linearization of the dynamic models. However, nonlinear systems suffer from obtaining precise mathematical models considering the complex properties of the systems. The inaccurate and complex models significantly increase the difficulty in proper linearization, which leads to the limited application of the above mentioned control techniques. Besides, the control performance can be significantly affected by the variation of system dynamic, parameter uncertainty, measurement error and external disturbance, etc. The above mentioned issues remain the development of control strategies for nonlinear systems a challenging issue. Nevertheless, reinforcement learning (RL) algorithms have less requirement on the knowledge of the dynamic model, of which control policy can be iteratively optimized in the interaction process with environment by learning from either online or offline experience. Therefore, RL algorithms reveal high potential of being an alternative solution to proposing optimal control policy for nonlinear systems.

Considering the limitations revealed in conventional control strategies and the advantages of RL algorithms shown in nonlinear system control, this thesis explores the possibility of developing control systems based on RL algorithms. The main focuses of this research stay on the design and development of optimal control strategies for nonlinear systems by infusing conventional control techniques with RL algorithms, aiming at enhancing the system performance both in transient response and robustness from control perspective and improving the learning efficiency and stability in the training process from RL algorithms perspective. Several hybrid models are proposed in this thesis which apply conventional PID controller or FLS as part of RL learning mechanism. The performance of the control system after training are compared and analysed on nonlinear platforms in simulation environments, where the advantage of the hybrid models are shown both in transient properties and robustness. The main contributions of the thesis can be summarized as three main sections, which are presented as following:

1) The first work is presented in Chapter 3. In this work, an innovative structure of adaptive PID controller based on Q-learning algorithm (Q-PID controller) is

proposed, which aims to provide a RL based training scheme for multiple PID controllers in order to improve the transient performance and the adaption of PID controller in complex environments. An adaptive learning rate scheme is applied as an acceleration method in the learning process. The proposed controller is tested on the inverted pendulum system in simulation environment with two comparisons, which are conventional PID controller and the controller simply using Q-learning algorithm, respectively. The simulation results indicate the benefit of proposed Q-PID controller over other two opponents in both generality and stability.

2) The second work is presented in Chapter 4, where an adaptive neuro-fuzzy PID controller, implemented as an actor, based on twin delayed deep deterministic policy gradient (TD3) algorithm is developed, which addresses the training challenge caused by the increased number of parameters in the system. Another linear PID controller based on TD3 algorithm is also provided for comparison purpose. The input values are infused with fuzzy information and a specially designed neuro-fuzzy PID controller is proposed as actor approximator in the RL algorithm. An inverted pendulum simulation environment is provided as to test the performance of the controller after optimization, which shows advantages over the comparison in both generalization and robustness tests.

3) In Chapter 5, the developed approach inherits the research idea presented in Chapter 4 by applying FLS as actor approximator in an off-policy actor-critic algorithm. The proposed method reduces the complexity in optimizing FLS based on RL algorithm especially for IT2 cases as well as enhances the robustness and control performance of the controller after optimization. The type of FLS is extended from type-1 to IT2 with a more flexible learning architecture, of which parameters in both antecedent and consequent are adjustable in the optimization process. The details of the update rules for the actor approximator in the proposed RL algorithm are provided. Two other types of controllers are provided as actor function approximators for comparison purposes which are type-1 fuzzy PD (T1-FPD) controller and neuro-PD controller, respectively. The update rules of T1-FPD controller are also provided as a special case of IT2-FPD. The performance of the proposed controllers are tested on the inverted pendulum system to compare the properties in transient response and robustness. The advantage of the IT2-FPD controller as function approximator over other two comparisons in these tests are verified.

Contents

Author's Publications

[1] **Shi, Q.**, Lam, H.K., Xiao, B. and Tsai, S.H., 2018. Adaptive PID controller based on Q-learning algorithm. *CAAI Transactions on Intelligence Technology*, 3(4), pp. 235-244.

[2] **Shi, Q.**, Lam, H.K., Xuan, C.B. and Chen, M., 2020. Adaptive neuro-fuzzy PID controller based on twin delayed deep deterministic policy gradient algorithm. *Neurocomputing*, 402, pp. 183-194.

[3] **Shi, Q.**, Lam, H.K., Xuan, C., 2021, Off-policy actor-critic algorithm with IT2-FPD as actor approximator. *IEEE Transactions on Fuzzy Systems* (under review).

[4] Yu, Y., **Shi, Q.** and Lam, H.K., 2018, December. Fuzzy sliding mode control of a continuum manipulator. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 2057-2062). IEEE.

[5] Chen, M., Lam, H.K., **Shi, Q.** and Xiao, B., 2019. Reinforcement Learning-based Control of Nonlinear Systems using Lyapunov Stability Concept and Fuzzy Reward Scheme. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(10), pp. 2059-2063.

[6] Xuan, C.B., Lam, H.K., **Shi, Q.** and Chen, M., 2021. Continuous Interval Type-2 Fuzzy Q-learning Algorithm for Trajectory Tracking Tasks for Vehicles. *International Journal of Robust and Nonlinear Control* (under review).

[7] Wen, S.H., Wang, T., Chen, J. H., Lam, H.K. and **Shi, Q.** Research on Obstacle Avoidance of Robotic Manipulators Based on DDPG and Transfer Learning. *Journal of Intelligent and Robotic Systems* (under review).

List of Figures

List of Tables

Notations and Acronyms

AC	Actor-critic
DDPG	Deep deterministic policy gradient
DL	Deep learning
DRL	Deep reinforcement learning
FCNN	Fully connected neural network
FLS	Fuzzy logic system
FOU	Footprint of uncertainty
IT2	Interval type-2
IT2-FPD	Interval type-2 fuzzy proportional differential
LMF	Lower membership function
MDP	Markov decision process
MF	Membership function
NN	Neural network
PID	Proportional-integral-differential
RL	Reinforcement learning
T1	Type-1
TD	Temporal difference
TD3	Twin delayed deep deterministic policy gradient
T-S	Takagi-Sugeno
UMF	Upper membership function

Chapter 1

Introduction

1.1 Background

In the past several decades, control theory has been widely applied to various fields including domestic appliance, traffic control, robotics, energy system [1], medical area and chemical processing [2], etc. With the increased level of requirements on the industrial applications from various aspects such as precision, accuracy, robustness, dynamic response and the raising complexity of the environment, developing control schemes with optimal performance becomes a crucial issue in research area. In the traditional designing process of control systems, the quantitative mathematical model is usually required as to approximate the dynamic characteristics of the environment, based on which the control scheme is developed after linearization or simplification process [3]. Therefore, the accuracy of the dynamic model has significant effect on the performance of the control system. However, in real physical scenarios, most intricate environments are associated with nonlinear properties, which leads the approximating and processing mathematical models with high accuracy difficult to be guaranteed. Additionally, the performance of developed controllers has a high dependency on the values of parameters [4]. When controllers are exposed to adverse situations with unexpected disturbances and damages, requirements of frequent adjustments are inevitable, which is found to be demanding and time consuming [5]. Thus, advanced approaches are required to release the difficulty in developing control strategies for systems associated with high nonlinearity and address the issue of adjusting parameters of developed controllers to optimum values under various operational conditions.

Contrary to the conventional approaches for the development of control systems, RL is a branch of machine learning algorithms apart from supervised learning and unsupervised learning, of which idea was originally inspired by the pattern of human behavior proposed in behaviorist psychology. The learning procedure of RL algorithm imitates the paradigm of human being in the trial-and-error pattern, where the agent progressively optimizes decision policy by interacting with environment

based on the only information from reward signal [6]. Thus, RL algorithm could generate optimal policy only by interacting with the environment and receiving reward signals without knowing the underlying dynamic models. Especially when RL concept is infused with deep learning techniques, deep reinforcement learning (DRL) algorithms achieve significant successes in solving challenging tasks such as playing videos games Atari [7] and Go [8] with massive state and action space and achieving scores surpassing human level, which shows the powerful learning ability of DRL algorithms. The model-free property of RL algorithms in generating control strategy for complex systems avoids formulating accurate mathematical models as traditional control schemes do [9] while the remarkable advantage in finding optimal control strategies could provide conventional controllers quick response and adaption to unexpected changes occurred in the environment [10], which indicates a promising direction of employing RL algorithms with conventional control techniques as to address the difficulty in developing control systems with high nonlinearity and parameter adjustment.

Therefore, the research direction proposed in this thesis focuses on the development of RL-based control systems for nonlinear systems. Specifically, the control systems this research topic mainly covers are the proportional-integral-derivative (PID) controller, which is widely used in industrial applications because of the high robustness and simple structure, as well as the FLS, which has high tolerance to the noise and powerful ability for nonlinear approximation.

1.2 Research Challenges

The RL based control system proposed in this thesis can be specified into two main directions, which are the optimization of PID controller based on Q-learning algorithm and the development of FLS (type-1 and interval type-2) based on TD3 algorithm. Therefore, the research challenges are introduced separately according different research directions.

PID controllers have been widely applied in various occasions because of the simplicity in structure and robust performance. However, PID controller has high dependency on the accuracy of parameters while the adjustment of parameter is found to be time-consuming. This leads to unsatisfactory performance when PID controller is exposed to challenging environment with high uncertainties and disturbances. RL algorithm is found be an efficient strategy in finding proper combination of gains for PID controllers. Nevertheless, training with RL algorithms always require high computational ability, which can be challenging for many application environments supporting PID controllers. Therefore, proposing an efficient RL based tuning method as well as keeping low computational cost for multiple PID controllers remains an open issue.

FLS is regarded as an universal function approximator which embeds expert experience and knowledge in fuzzy rule format and has been widely applied in broad areas. With the development of DRL techniques in recent years, applying RL algorithms as a tuning method reveals high potential in optimization of FLS. In traditional approaches proposed for the development of RL based FLS, the number of parameters usually determines the dimensionality of action space in RL algorithms. However, when FLS is applied to complicated environments, the increased number of fuzzy rules and membership functions would lead to the inevitable high-dimensional action space in RL algorithm, which brings converging difficulty in RL training process.

1.3 Research Motivations

This thesis aims to propose advanced approaches for the development of RL algorithms based control systems which address the design difficulty for nonlinear systems and implement parameter optimization with improved learning efficiency. Based on the research challenges stated above, the research gaps related to different categories of controllers are presented as follow.

1) In the development of RL based PID controllers, the tuning strategy for multiple PID controllers which keeps the efficiency in training procedure as well as remains low computational cost are less addressed.

2) In the development of RL based fuzzy control system, the training efficiency of RL algorithms still significantly affected by the complexity of the fuzzy system. Advanced training structure needs to be proposed to avoid the learning difficulty in RL algorithms caused by the increased number of parameters in FLS.

3) In the development of RL based interval type-2 FLS, the type-reduction procedure and extra parameters related to the FOU in IT2 FMs lead to the difficulty in RL training procedure on both parameter updating and computational cost aspects. Further researches need to be done regarding to provide solutions to the aforementioned issues.

Correspondingly, the objectives can be specified as follows.

1) Develop an adaptive PID controller based on Q-learning algorithm (named as Q-PID controller), aiming to achieve the tuning of parameters of PID controllers through the training procedure of Q-learning algorithm with improved efficiency. Besides, improve the transient response performance of the Q-PID controller compared with the conventional controller.

2) Propose a training method for type-1 fuzzy logic control systems based on actor-critic structure, which releases the training difficulty caused by the rising number of parameters as well as enhances the control performance of type-1 controller.

3) Extend the actor-critic based optimization method from type-1 to interval type-2 FLS for occasions with high uncertainties. Provide theoretical derivation of update rule regarding to actor approximator where a neuro-fuzzy network replaces the traditional NN. Besides, explore the controlling performance and robustness of the IT2 controller after training.

1.4 Contributions of the Thesis

The contributions of the thesis are summarized as follows:

1. An adaptive PID controller based on Q-learning algorithm is proposed and tested on the cart-pole platform in simulation environment, compared with conventional PID controllers and the controller only using Q-learning algorithm. The comparing experiment indicates the benefits of the adaptive PID controller in generality and stability. The result has been published in " Adaptive PID controller based on Q-learning algorithm," *CAAI Transactions on Intelligence Technology*, 3, no. 4 (2018): 235-244.

2. An adaptive neuro-fuzzy PID controller, implemented as an actor, based on TD3 algorithm is proposed. A linear PID controller based on TD3 algorithm is provided for comparison purpose. The generalization and robustness tests were conducted on the inverted pendulum system to show the advantages of the proposed method. The results have been published in "Adaptive neuro-fuzzy PID controller based on twin delayed deep deterministic policy gradient algorithm," *Neurocomputing*, 402 (2020): 183-194.

3. An innovative actor-critic algorithm with IT2-FPD controller as actor approximator is proposed. The updating rules for this new actor in AC algorithm are derived with both the parameters in the antecedent and the consequent of the IT2-FPD controller covered. The performance of the proposed controllers are tested on the inverted pendulum system to compare the properties in transient response and robustness. The advantage of the IT2-FPD controller as function approximator over other two comparisons in these tests are verified. The results have been submitted to *IEEE Transactions on Fuzzy Systems* as a regular paper "Off-policy Actor-critic Algorithm with Interval Type-2 Fuzzy PD Controller as Actor Approximator".

1.5 Organisation of the Works

This thesis presents the research topic regarding to the development of RL based control system for nonlinear systems. Chapter 2 introduces the preliminary knowledge and literature review related to this research while the main works are divided into three main chapters presented in Chapter 3 to Chapter 5, where various approaches embedding RL techniques with control systems have been proposed. In Chapter 3, the control system which applies Q-learning algorithm as an optimization scheme to PID controllers are developed. This system is designed for occasions that require adaptive performance of linear PID controllers but not capable of high computational cost. Inspired by the success of Q-learning based PID controller proposed in Chapter 3. The following two chapters extend the RL based control system from PID controller to fuzzy PID controllers. The latter can be regarded as a nonlinear combination of several local linear PID controllers, which has priority in approximating systems with high nonlinearities. However, the increased number of parameters in fuzzy PID controllers leads to the learning difficulty in RL training process. Therefore, DDPG and its variant TD3 algorithm are utilized instead of Q-learning for better training efficiency. Chapter 4 explores the combination of TD3 algorithm with type-1 fuzzy logic system with an innovative neuro type FLS as network approximator. While Chapter 5 further extends this successful training structure from type-1 to interval type-2 fuzzy controller, which is more robust to the external noise and parameter uncertainties in more challenging environments. Followed by three main chapters, Chapter 6 summarizes the whole research work and discusses the potential research directions in the future work. The details of each chapter are shown as follows.

- In Chapter 2, the basic concepts of PID controller, fuzzy control systems, RL algorithms, experience replay technique and inverted pendulum platform will be introduced.

- In Chapter 3, a novel adaptive PID based on Q-learning algorithm is demonstrated. The construction of the controller and the training procedure are explored while the performance of the controller after training is tested on the inverted pendulum.

- In Chapter 3, TD3 algorithm with an adaptive neuro-fuzzy PID controller is provided. A specially designed fuzzy PID controller in NN formation acts as the actor role in the TD3 algorithm. The revised TD3 algorithm could efficiently find optimal coefficient parameters for FLS. As to test the performance of fuzzy controller after training, the invented pendulum system is chosen to test the proposed method with a neuro-PID controller provided for comparison.

- In Chapter 5, an innovative off-policy AC algorithm with interval type-2 fuzzy PD (IT2-FPD) controller as actor approximator is proposed. Similarly to the approach proposed in Chapter 4, the IT2-FPD controller in neuro-fuzzy network structure instead of the traditional fully-connected neural network (FCNN) as actor approximator. The update rules of the proposed algorithm are provided for parameters in both antecedent and consequent of IT2 controller which achieves optimization in more general cases. The fuzzy controller with improved learning efficiency while the trained controller achieved stable and robust controlling performance. Two other types of controllers are provided as actor function approximators for comparison purposes which are type-1 fuzzy PD (T1-FPD) controller and neuro-PD controller, respectively.

- In Chapter 6, the final conclusions of the thesis are given and the potential research directions for future work are discussed.

Chapter 2

Literature Review

2.1 Literature review

In this section, the previous research works related to developing RL-based control systems are introduced. Considering this research direction is an intersection between two areas, the fundamental methods related to each individual field will be introduced separately in subsection 2.1.1, which are the methodologies of FLSs and the highlighted techniques applied in RL algorithms, as to provide better support for the discussion of the junction area. While in subsection 2.1.2 the works utilizing RL methods for control system development relating to the research topic proposed in this thesis are analysed.

2.1.1 Literature review of fundamental methods

Overview of fuzzy logic system

The theory of "fuzzy set" was originally proposed in 1965 by L. A. Zadeh [11]. Based on this concept, the first fuzzy logic control system was proposed in [12] in 1996 and has made significant development in recent years. The FLS synthesis expert knowledge or human experience through linguistic fuzzy rules defined in IF-THEN format and operates reasoning process in human-like pattern. This model-free approach releases the difficulty of conventional controllers have in solving ill-defined or nonlinear systems [13]. Therefore, fuzzy logic control techniques have been extensively applied to wide range of fields including control theory [14,15], data mining [16–18], assessment [19], modelling [20–22], classification [23–25], forecasting [26, 27], etc.

1) **Mamdani and T-S fuzzy model**

   Mamdani fuzzy model [28–30] is the first type of FLS proposed in early 1970s, where the rules are formalized heuristically without accessing mathematical model of the systems. This model-free method is straight forward and is successfully

applied in various complex occasions with high nonlinearity, such as the control of steam engine [31], DC-DC power converters [32], economics [33], risk assessment [34], etc. However, because of the lack of theory support, the appropriate decision for fuzzy rules of Mamdani system is challenging and time consuming. Furthermore, the performance and stability of control system are not guaranteed [13]. As to address these limitations, the model-based Takagi-Sugeno (T-S) fuzzy systems [35] are proposed. Different from Mamdani type fuzzy models, the consequent of T-S fuzzy rule is the function of antecedents variables, which provides high support in constructing mathematical model and theoretical analysis. When bridging the plant presented by a T-S fuzzy model with state-feedback fuzzy controller [36], fuzzy-model-based (FMB) control systems is constructed. Several research works regarding stability analysis [37–39] have been carried out based on the FMB control systems and its extension polynomial fuzzy-model-based control systems [40].

2) **Type-1 and type-2 fuzzy model**

There are several circumstances where uncertainties occur in FLS. Since FLS is based on the human experience, the linguistic descriptions can be ambiguous and subjective depending on the personal experience and judgement from different people. Besides, the noise of data and measurement also lead to vagueness of FLS [41]. Type-1 fuzzy model becomes limited in handling these models considering the lack of descriptive variables for uncertainty in type-1 membership function (MF). The type-2 fuzzy set [42] embedding the uncertainties in membership function was proposed in 1975. A type-2 [41] MF is based in three-dimensional domain and is represented in two styles, which are vertical-slice representation and wavy-slice representation, respectively. Though the secondary MF of type-2 fuzzy model brings more freedom for designing uncertainty, the precise decision of secondary MF is still a challenging issue, while the secondary memberships bring extra computational burden [43]. IT2 FLS can be regarded as a special case of general type-2 fuzzy set, where only the footprint of uncertainty (FOU) remains with the secondary membership reduces to a constant number. Thus, the IT2 FLS keeps the characteristics of modelling uncertainty while minimises the intense computation cost, which leads to the wide utilization of IT2 FLS in most of the real applications with the request of modelling uncertainty [41].

3) **Neuro-fuzzy system**

Neural Network (NN) has powerful ability in function approximation and feature representation [44] but has less tolerance to the noisy or incomplete data [45, 46]. Nevertheless, as stated above, fuzzy systems show advantage in handling system uncertainty and noise therefore can achieve enhanced learning performance when infused with NN architecture. The hybrid of FLS with NN is referred to

as neuro-fuzzy systems [46], which can be sorted into two main categories. The first category neuro-fuzzy systems embeds fuzzy information with either input or output variables without modifying the structure of NN, such as research [47–52], where the fuzzy information provides better feature representation ability which boosts the learning efficiency or enhances the classification of the input information. While the second type of neuro-fuzzy system infuses the NN structure with fuzzy logic models [53–57], which inherits the characteristics of FLS with enhanced robustness to the NN. The successful applications of neuro-fuzzy systems include traffic control [54,58–60], text processing [47,61,62], image processing [54,63,64], etc.

The methodology proposed in this thesis refers to the T-S fuzzy model. Both RL based type-1 and type-2 FLS in NN format are explored.

Overview of reinforcement learning algorithms

With the massive development of Deep Learning (DL) technique in recent years, applying DL to RL field brings extra flexibility and complexity which scales up the original RL problem definition to high-dimensional problems. This modern field is named as deep reinforcement learning (DRL) which brings significant breakouts to a broad scale of areas including robotics [65–71], video games [7, 8, 72, 73], natural language processing [74–76], computer vision [77–80], autonomous driving [81–84], finance [85–87], navigation [9, 88], etc.

There are two main categories of RL algorithms which are model-based and model-free algorithms. In model-based algorithms [89–94], a transition model is approximated by collecting samples from the interaction process, which further generates samples without directly interacting with the real environment. Therefore, model-based RL algorithms are sample efficient, especially for occasions where the interactions with environment are rare and expensive. However the model-based approaches suffer from model errors in approximation while the policy optimization has high dependency on the accuracy of the approximated model. This leads to the difficulty in successful training of model-based methods. Contrarily, model-free algorithms learn control policies without access to the underlying model of the environment. Due to the generality characteristics, most of the state-of-the-art RL algorithms belong to the model-free branch. The rest of this thesis mainly focuses on the model-free methods and the RL methods mentioned below are regarded as model-free RL algorithms in default. There are three branches of model-free RL algorithms which are value-based algorithms, policy-based algorithms and actor-critic algorithms with details introduced as follows.

1) **Value-based algorithms**

In value-based algorithms, an individual value function is approximated for the evaluation of the accumulated discounted future rewards. The state value or the state action value indicates the performance of the agent and guides the policy generation. Temporal difference (TD) learning [95] method was proposed in 1988 as the fundamental method of calculating value functions. As extensions of TD learning, two classic value-based RL algorithms were proposed, which are State-Action-Reward-State-Action (SARSA) [96] as an on-policy version and Q-learning [97] algorithm as an off-policy version. Deep Q-Network (DQN) [7] adopts DL technique with Q-learning algorithm, which applies NN as value function approximator. Taking advantage of the significant function approximation ability of NN while utilizing experience replay and target NN methods, the agent outperforms human-level performance in a range of Atari 2600 video games with compelling stability. Inspired by the success of DQN algorithm, several variants have been proposed to further improve the performance, such as Double DQN (D-DQN) [98], dueling DQN [99], RAINBOW [100] and Normalized Advantage Function (NAF) [93] algorithm, etc. Therefore, Q-learning algorithm and DQN with its variants becomes one of important value-based algorithm among the state-of-the-art RL algorithms.

2) **Policy-based algorithms**

Different from value-based algorithms, the policy-based algorithms directly formalize a parameterized policy function for policy generation instead of relying on extra value function approximation. The parameters of policy function are updated in the iteration process aiming to maximize the expected over return value through either gradient-free or gradient-based methods [44]. Gradient-free updating approach [101, 102] shows advantage in arbitrary and nondifferentiable parameter space but exposes limitations in cases where either population or parameter are intense. Contrarily gradient-based methods are data efficient for large scale parameters and are preferred for most DRL algorithms. One of the famous policy-based algorithms REINFORCE [103] was proposed in 1992, in which algorithm the gradient is estimated with the log probability of sampled action weighted by the return value [44]. Natural policy gradient (NPG) [104] addresses the instability problem in the policy gradient methods [105] but has high requirement on the computational ability and is only limited to linear function with small tasks. Trust region policy optimization (TRPO) [106] and Proximal Policy Optimization (PPO) [107] further extend the idea of NPG method to DRL area. TRPO introduces the trust region concept that constrains the policies to the small field where the true approximation still holds, which leads to the performance of the agent increases monotonically with improved robustness. PPO simplifies the algorithm of TRPO using only first-order optimization and clipped surrogate objective function to avoid large update steps [105]. [108] also pro-

posed Generalized Advantage Estimation (GAE) based on TRPO with several improved baselines reducing the variance. The combination of GAE and TRPO becomes one of the state-of-the-art algorithms in DRL field.

3) **Actor-critic algorithms**

Remark 2.1 *There are argues regarding to the category of actor-critic (AC) algorithms, where AC algorithm is regarded as a special case of policy-based algorithms since critic approximator is considered as baseline for variance reduction. However based on the compelling performance and massive contributions made to this specific category, in this paper, the AC algorithm is discussed as an individual section apart from value-based and policy-based algorithms in RL algorithms.*

Actor-critic methods take advantage of the merits from both the value-based and policy-based algorithms, which evaluate two parameterized functions for state evaluation and policy generation separately. The value function refers to the "critic" part in the algorithm name, of which value estimates the quality of current state and provides guidance for the policy function optimization. This separate value function reduces the approximation variance and improves the learning efficiency [109]. The policy function refers to the "actor" section in the name, which develops the control policy for the agent based on the feedback given by the value function. Benefit from the research work of Deterministic Policy Gradient (DPG) which extends policy gradient theorems from stochastic policies to deterministic policies, Deep Deterministic Policy Gradient (DDPG) [110] provides a DRL algorithm with continuous output space. In DDPG algorithm the experience replay method are applied to break the correlations among experience samples and separate target networks technique are utilized as implemented in the deep Q-network (DQN) [111] algorithm to stabilize the learning process. Twin delayed deep deterministic policy gradient (TD3) [112] algorithm was proposed in 2018 to further address the overestimation issue and release the hyper-parameter sensitivity exposed in DDPG algorithm. Research develops Advantage Actor-critic (A2C) which uses advantage value function for critic evaluation instead of state value function to reduce the high variance of conventional AC algorithm. Soft Actor-critic (SAC) involves the concept of maximum energy in the objective function aiming to maximize both the expected return value and the entropy.

The RL algorithms applied in this thesis includes the Q-learning from value-based algorithms and TD3 algorithm from actor-critic algorithms.

2.1.2 Literature review of existing works

In this section, the research works related to the overlapping area between RL and traditional control fields are discussed. The methodologies proposed in the previous researches regarding to the development of RL based control systems can be divided into two main directions, one of which purely utilizes RL algorithms to replace the traditional controller to generate control policy while the other infuses RL together with conventional control systems to improve control performance in nonlinear systems. The methodologies applying RL as individual control strategies are presented in the first subsection. The research works related to the combination of RL and conventional controllers are further categorised according to different types of controllers covered in this thesis, where the RL based PID controller is introduced in second subsection and RL based FLS is discussed in the last subsection. The limitations of RL based control systems are summarized in the end of this section, which lead to the research objectives proposed in the following section.

1) **Controllers replaced with RL algorithm**

In this category of RL based control systems, RL algorithms totally replace the conventional controllers to generated control strategy to solve the required tasks. In [113], five different RL algorithms (two value-based algorithms and three policy-based algorithms) were tested to replace a double-loop PID controller to control a ball screw feed drive. All five algorithms indicate the superiority in controlling performance compared to the conventional PID controller, which indicates the potential of RL being an alternative control method compared with conventional PID controller. Research [114] and [115] both utilized Q-learning algorithm for the generation of control strategies, which are applied for liquid level control of a non-linear conical tank and robot balancing, accordingly. Both researches reach satisfactory performance in finishing required tasks. Besides the control system shows priority in convenient implementation and high training efficiency because of the simplicity of the Q-learning algorithm. However, since the output domain of Q-learning algorithm is located in discontinuous space, the output are discrete values, which leads to unsatisfactory performance in the stable state, where small fluctuations can be obviously observed. This can be quite undesirable for many real-world applications where smooth and stable response is needed. [116] applied DDPG algorithm directly to generate control policy based on the data collected by the sensors to control low-level autonomous underwater vehicles and showed successful application in real experiment while [117] applied DPG method for the depth control of an unmanned aerial vehicle.

Though the researches with RL algorithms replacing conventional controllers achieve ideal performance as stated above, most of the control tasks are less challenging, where the tracking or stabilizing requirement in the nonlinear sce-

narios are restricted to only one or two degree-of-freedom (DOF). Furthermore, the success of the performance has high dependency on the proper choose of state representation, output and reward of the RL algorithms, which is time assuming from the designers. This situation can be more challenging when transferring the aforementioned approaches to scenarios with increased complexity and dimensionality [4].

2) **RL algorithm based PID controller**

Consider the limitations addressed above, another considerable approach is to combine the RL algorithm with the conventional controllers. In this section, the approaches embedding RL with PID controller are analyzed. In 1997, the discrete action reinforcement learning automata (DARLA) [118] was proposed, based on which two extension versions [119] and [120] are developed to tune the coefficients of one single PID controller. The PID controller after optimization were tested on an automatic voltage regulator system. Similar to DARLA, continuous action reinforcement learning automata (CARLA) [118] for PID controller was applied as an online tuning method in the research done by [121] and [122]. The approaches were also tested via an engine idle-speed control system and the ball and beam system in research [122]. The controller optimized by DARLA and CARLA methods are compared with the conventional PID controllers tuned following the Ziegler-Nichols methods in the simulation environment. Both DARLA and CARLA methods successfully find optimal coefficients for the PID controllers with satisfactory learning efficiency. The controllers tuned by both mechanisms showed the superior performance compared with conventional PID controller. However, DARLA method is found to be parameter sensitive especially in the initialization process while both methods have high dependency on the range of gains values truly works for the controller, of which information relies on the prior knowledge of the designers and not always available for all occasions. Besides, the integral and interpolation operation in the algorithm brings the extra cost of computation [119]. [123] implements actor-critic method for parameter tuning of PID controller in wind turbine control while a similar approach was implemented in [5] as to achieve tracking control for special linear systems. In both approaches, a radial basis function is utilized to approximate the critic and actor functions simultaneously. Both approaches indicated better adaption and robustness in the control performances compared with the conventional PID controllers.

Additionally, Q-learning algorithm is also widely used in tuning parameters for PID controllers. [124] adopted Q-learning algorithm to generate additional force for correcting the output of a pre-tuned PID controller, where the weights of the PID controller remain fixed during the whole training process. [4] tuned the PID controller using an incremental Q-PID algorithm by dynamically dividing the actions into more specific areas to obtain higher controlling accuracy. [125]

tuned two sets of PID controllers to control the angular and linear speed of a soccer robot and achieved good performance in speed response and stability. In these cases, either Q-table or NN is used as the Q-value approximators with Q-learning algorithm to improve the properties of controllers in transient response or robustness. There are also different techniques have been proposed to optimize the performance of Q-learning based PID controllers. Research [126] designs a hybrid model which infuses PID controller with Q-learning algorithm aiming to handle continuous vehicle control problems, where a quadratic Q-function is applied to guide the actor approximating towards optimal policy. While research [127] utilizes fuzzy method to reduce the input information as to accelerate the Q-learning procedure.

In the researches developed for the RL based PID controllers, the RL algorithm acts as an alternative strategy for tuning PID controller. Compared with other classical tuning algorithms for PID controller, such as Ziegler-Nichols method, ant colony optimization algorithm [128,129] and genetic algorithm [130], RL algorithms require less iterative procedures and is more computational friendly [113], among which the simplicity of structure of Q-learning algorithm brings extra implementation convenience in several industrial application scenarios [119]. Nevertheless, the training of multiple PID controllers based on Q-learning algorithm is not fully addressed from the aspects of training efficiency, computational cost and control performance of the controller after optimization.

3) **RL algorithm based fuzzy controller**

In this subsection, different approaches which infuse RL algorithms with FLSs are discussed. Research [130] implements Q-learning algorithm for fuzzy PD/PI controllers optimization, of which performance is compared with the conventional PID controller. The Q-learning based controller has quicker response speed and increased robustness. However, Q-learning algorithm shows limited capability in optimizing fuzzy logic control systems, since FLS has multiple fuzzy rules involved in, each rule is associated with several coefficients in both antecedent and consequent part. This leads to the obvious increase in the number of parameters. Besides, the outputs of Q-learning algorithm are limited to discrete action domain, the proper discretization for numerous parameters of FLS becomes an challenging issue, which also makes the "the curse of dimensionality" an inevitable problem.

Different from Q-learning algorithm, AC algorithm utilizes two universal approximators for value and policy evaluation separately, where both state and action space could be continuous, which alleviates the difficulty in discretization problem caused by Q-learning algorithm. There have been several researches proposed for AC based fuzzy controllers. In the following contents, the similar

works regarding to the actor-critic based fuzzy or fuzzy PID controller are analysed. [131] applies DDPG algorithm to adjust the coefficients of an single input type-2 fuzzy PI controller. In this approach, a primary fuzzy PI controller has already existed with regulated operating condition, of which PI coefficients of controller are provided as a baseline for the reference of DDPG algorithm. The DDPG algorithm is added as an online learning algorithm, which enhances the performance of the primary controller under challenging environment. Similarly, the works [132] and [133] have similar control structure but different application scenarios. In these cases, the system is stabilized with an IT2 controller and the DDPG algorithm generates extra control signal which is added to the output of IT2 fuzzy controller for compensation of the load variation and external disturbances. However, in the above researches the DDPG algorithm applied in this research generates the coefficient values as output based on the various input value of the fuzzy controller, where the dimensions of the action space are defined by the number of gains that required to be optimized. Based on the results of the research, this strategy could work properly when the number of tuning parameters are small, or when another primary controller exists in the system, which releases the training difficulty required from RL algorithms. Nevertheless, when the number of tuning parameters rises or the stability of the system is not guaranteed, the capability of AC algorithm as an optimizing scheme to optimize the coefficients in the fuzzy controllers remains an open issue.

Another alternative approach is to replace actor NN directly with FLS in neuro-fuzzy formation. [134] applies this idea by replacing both critic and actor approximator with two type-1 T-S FLS. The parameters in the antecedent part of the FLS are updated in the RL training process while the parameters in the consequent remain fixed. The proposed method shows improved control performance without relying on additional pre-tuned controller to keep the system stable before the application of RL algorithm, which indicates the priority of the proposed training structure. However, in some cases with high uncertainties and disturbances, the ability of type-1 FLS becomes limited. While as previously stated, IT2 fuzzy controller, as a special case of type-2 fuzzy controller, with better computational efficiency while remaining the high robustness of general type-2 fuzzy controller, shows high potentials in handling environments with noisy inputs and parameter uncertainties. Several research works applied IT2 fuzzy controller as an actor approximator in the AC algorithm as to take advantage of these properties. [135] developed a RL based type-2 controller for drone flight game. In this method, though acceptable control performance has been achieved, the research reveals the training results are strongly affected by parameters in initialization which can only be decided based on the trial-and-error approach. Therefore, the reporducibility of the proposed method remains an open issue. Another work

was introduced in [136], where both critic and actor NNs are replaced by the IT2 T-S fuzzy NN. Both the parameters in antecedent and consequent are updated online with the number of fuzzy rules adjusted using fuzzy clustering method, which enhances the robustness of the controller. Though IT2 MFs improve the robustness of the controller in the research, having IT2-FNN as both critic and actor approximator can be inevitably computational heavy. Besides, the samples in online updating method are only used once before being abandoned. These factors lead the learning efficiency in the proposed method not fully addressed.

By applying FLS as the actor approximator in the AC algorithm, the parameters required to be optimized in the controllers equals to the parameters of the actor approximator of AC algorithms. Therefore, increasing the number of parameters in the actor approximator does not increase the action dimensions, which avoids the learning difficulty caused by the expansion of action space [134]. Furthermore, FLS as function approximator is more tolerant to the approximation error generated by critic and therefore enhances the robustness of the learning process [137]. However, in the researches developed so far, the training difficulty caused by the increase of parameters still not fully addressed while the dynamic performance and robustness of the controller after optimization can be further improved. Besides, in most of the researches, only partial parameters of FLS are optimized in the RL training procedure while a training scheme with comprehensive coefficients associated remains unaddressed.

To summarize, the researches stated above reveal the advantage of RL in improving the control performance of traditional control systems. Comparing the two main directions of developing control systems with RL techniques, replacing conventional controllers directly with RL algorithms are restricted to simple tasks or less challenging environments while infusing RL with controllers is more capable of dealing with complex control problems under sophisticated scenarios. However, there are also limitations existed in the systems embedding RL with conventional controllers. For RL based PID controller, Q-learning algorithm shows advanced potentials in tuning PID controllers but less attention was paid to the automatic tuning issue for PID controllers with multiple loops. For RL based FLS, AC structure reveals advantage in finding optimized set of coefficients of FLS, nevertheless, the difficulty caused by the increased number of gains of FLS under complex operating conditions are not fully addressed. Besides, there is still no comprehensive structure proposed for optimizing FLS through AC algorithm with full components covered. Furthermore, for both PID controller and FLS, the dynamic response and robustness of controllers can be further improved. Based on these open issues existing in the RL based control systems, the aims and objectives of the thesis are defined in the following subsection.

2.2 Preliminaries

This section introduces the preliminary knowledge related to the research objectives. Three main directions will be covered, which are FLS in Section 2.2.1, concepts of RL algorithms in Section 2.2.2 and inverted pendulum platform in Section 2.2.3. Regarding to the FLS theory, the details of T-S fuzzy model, IT2 FLS, fuzzy PID controller and neuro-fuzzy system are introduced. While in the RL section, the core elements of RL problems such as mathematical definition of Markov Decision Process (MDP), value functions and policy functions as well as classic algorithm including Q-learning algorithm, AC algorithm, TD3 algorithm and Prioritized Experience Replay (PER) technique are discussed in the section afterwards. Finally, the inverted pendulum system used as experiment platform is introduced.

2.2.1 Fuzzy logic system

The FLS is formalized with four sections, which are fuzzify, rule base, inference engine, and output processor. In the fuzzify section, the fuzzy set is associated with various membership functions mapping the input variable to a continuous truth value ranging from 0 to 1, which is referred as the grade of membership function. The rule base stores all expert knowledge described as linguistic fuzzy rules in IF-THEN format. In each rule, the IF section is named as "antecedent" while the THEN part is denoted as "consequent". The reference engine generates fuzzy outputs by implementing fuzzy operators through all the fuzzy rules. The output processing procedure interprets the fuzzy variables input crisp value, which is an accurate and specific output that can be used as control or reference signal. To be noticed, this procedure differs from type-1 to type-2 FLS, where type-2 FLS has an extra type-reduction procedure to reduce type-2 fuzzy sets to type-1 ones before conducting defuzified process compared with type-1 FLS. The detailed difference between these two types of FLS can be found in Fig. 2.1 and Fig. 2.2.

T-S fuzzy model

The structure of a T1 FLS [138] is shown in Fig. 2.1, which is a revised version referenced from [139].

The consequent of a typical T-S fuzzy model is a linear function of input variables in the antecedent which can be defined as:
Rule i: IF x_1 is M_1^i AND x_2 is M_2^i AND...AND x_ψ is M_ψ^i
THEN $y_i = a_0^i + a_1^i x_1 + a_2^i x_2 + ... + a_\psi^i x_\psi$,

where $i = (1, ..., P)$, P is the number of fuzzy rules; M_j^i represents the type-1 fuzzy set (T1 FS) of the j^{th} linguistic variable in the i^{th} rule; $j = (1, ..., \psi)$ and ψ is the number of linguistic variables in the antecedent; y_i is the output of i^{th} rule; a_j^i is the

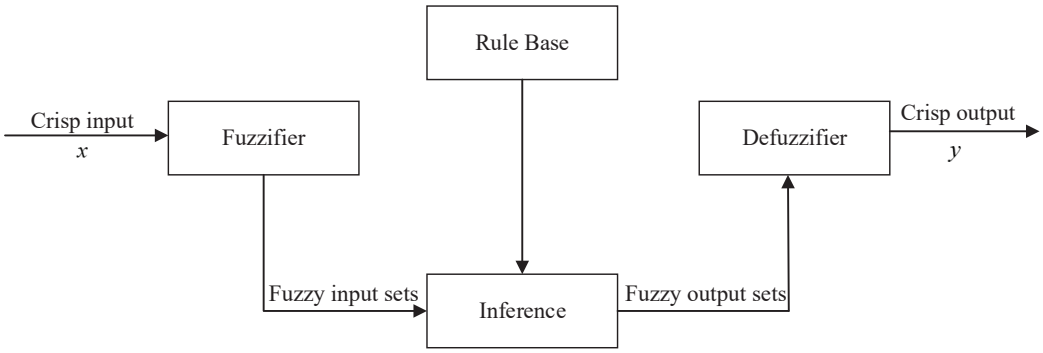

Figure 2.1: Structure of T1 FLS

coefficient of the j^{th} term of the polynomial in the consequent. The firing strength of i^{th} rule is

$$f^i(\boldsymbol{x}) \equiv \prod_{j=1}^{\psi} \mu_{M_j^i}(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{x} = [x_1, x_2, ...x_\psi]$, $\mu_{M_j^i}(\boldsymbol{x})$ is the degree of membership M_j^i. The output of T1 FLS is defined as

$$y = \frac{\sum_{i=1}^{\psi} f^i y_i}{\sum_{i=1}^{\psi} f^i}. \tag{2.2}$$

Interval type-2 fuzzy logic system

Type-2 FLS is defined as at least one T2 FS is applied in the FLSs. The structure of a type-2 FLS is demonstrated in Fig. 2.2, which is a revised version from [41]. A general T2 FS is sketched in 3-D domain and an IT2 FS is regarded as a special case of T2 FS with uniform weighting on the secondary grade. Fig. 2.3 presents a triangular IT2 MF for illustration purpose, the upper bound marked in blue color is referred as upper membership function (UMF) and and lower bound marked in red is named as lower membership function (LMF), respectively. The yellow shading area between the LMF and UMF composes the footprint of uncertainty (FOU).

The rule of a typical IT2 T-S fuzzy model is defined as:
Rule i: IF x_1 is $\widetilde{M_1^i}$ AND x_2 is $\widetilde{M_2^i}$ AND...AND x_ψ is $\widetilde{M_\psi^i}$ THEN $y_i = a_0^i + a_1^i x_1 + a_2^i x_2 + ... + a_\psi^i x_\psi$,

where $i = (1, ..., P)$, P is the number of rules; $\widetilde{M_j^i}$ represents the interval type-2 fuzzy membership function (IT2 MF) of the j^{th} linguistic variable in the i^{th} rule; $j = (1, ..., \psi)$ and ψ is the number of linguistic variables in the antecedent; y_i is the output of i^{th} rule; a_j^i is the coefficient of the j^{th} term of the polynomial in the consequent. The firing strength interval of i^{th} rule is

$$F^i(\boldsymbol{x}) \equiv [\underline{f}^i(\boldsymbol{x}), \overline{f}^i(\boldsymbol{x})], \tag{2.3}$$

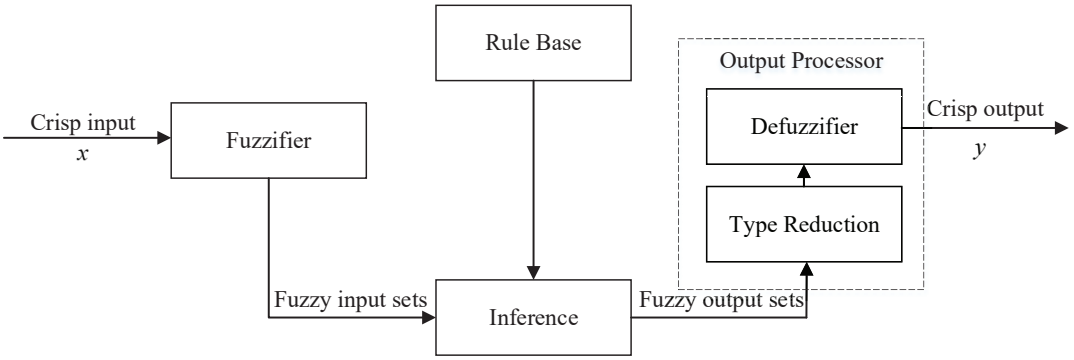

Figure 2.2: Structure of an T2 FLS

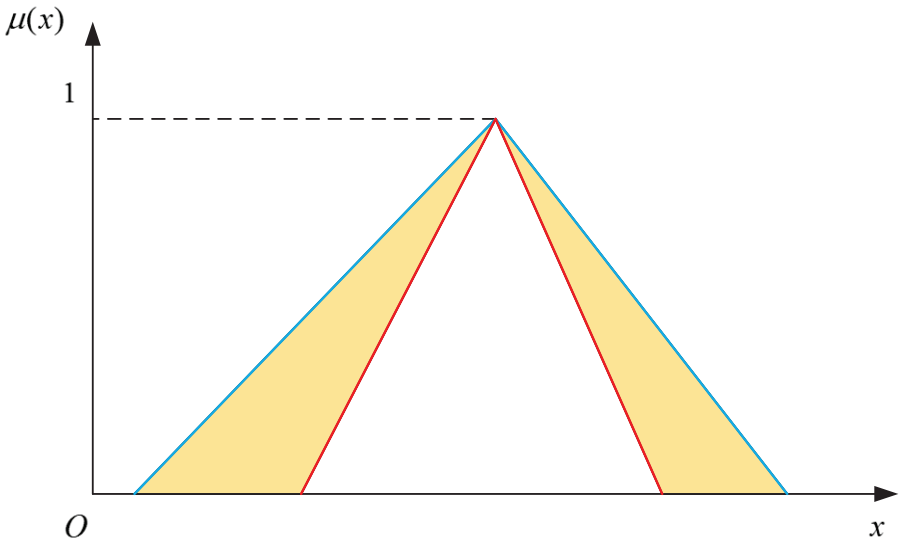

Figure 2.3: Structure of a triangular IT2 FS

$$\underline{f^i}(\boldsymbol{x}) \equiv \prod_{j=1}^{\psi} \underline{\mu}_{\widetilde{M}_j^i}(\boldsymbol{x}), \tag{2.4}$$

$$\overline{f^i}(\boldsymbol{x}) \equiv \prod_{j=1}^{\psi} \overline{\mu}_{\widetilde{M}_j^i}(\boldsymbol{x}), \tag{2.5}$$

$\underline{\mu}_{\widetilde{M}_j^i}(\boldsymbol{x})$ and $\overline{\mu}_{\widetilde{M}_j^i}(\boldsymbol{x})$ denote the degree of membership from the LMF and UMF of IT2 MF $\widetilde{M}_j^i$, respectively.

In this paper, the Begian-Melek-Mendel (BMM) [140, 141] method is chosen as the type reduction method, which has no requirement for sorting y_i $(i = 1, 2, ..., P)$ and was proven to be computational efficient in [142]. The output of IT2 FLS with BMM type reduction method is

$$y = \alpha \frac{\sum_{i=1}^{\psi} \underline{f}^i y_i}{\sum_{i=1}^{\psi} \underline{f}^i} + \beta \frac{\sum_{i=1}^{\psi} \overline{f}^i y_i}{\sum_{i=1}^{\psi} \overline{f}^i}, \tag{2.6}$$

where α and β are weighted coefficients.

Fuzzy PID controller

PID controller

The structure of a linear PID controller is shown in Fig. 2.4, which is consisted of three components namely proportional term, integral term and derivative term. Equation (2.7) provides the output of the PID controller in discrete time form.

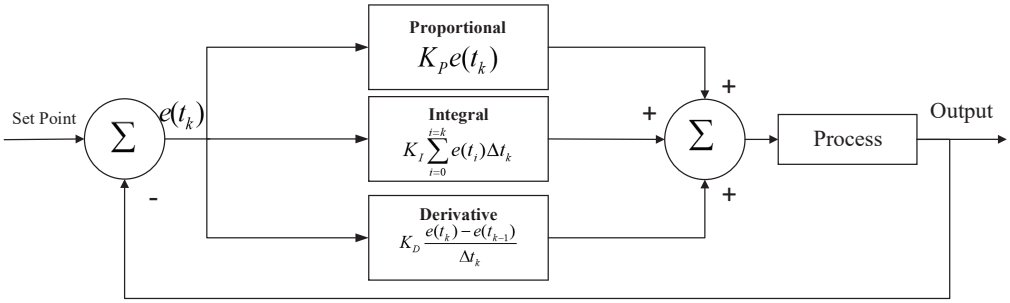

Figure 2.4: Structure of PID controller

$$u(t_k) = K_P e(t_k) + K_I \sum_{i=0}^{k} e(t_i)\Delta t_k + K_D \frac{e(t_k) - e(t_{k-1})}{\Delta t_k}, \tag{2.7}$$

where t_k is the k^{th} time step; $u(t_k)$ is the output in k^{th} time step; $e(t_k)$ is the error in k^{th} time step; Δt_k is the interval of sampling time in simulation; K_P, K_I, K_D are the proportional, integral and derivative gain, respectively.

Type-1 fuzzy PD controller

The fuzzy PID controller in discrete time domain can be expressed in the following format:

Rule i: IF $e(t_k)$ is M_1^i AND $de(t_k)$ is M_2^i AND $se(t_k)$ is M_3^i THEN $u(\boldsymbol{x}(t_k)) = u_i(t_k)$,

where $de(t_k)$ is the change of error in the k^{th} time step, $de(t_k) = e(t_k) - e(t_{k-1})$; $s_e(t_k)$ is the accumulated error until k^{th} time step, $se(t_k) = \sum_{m=0}^{k} e(t_m)$; $u_i(t_k)$ is the output of i^{th} PID controller. The final output of the fuzzy PID controller can be expressed as

$$u(\boldsymbol{x}(t_k)) = \sum_{i=1}^{p} w_i(\boldsymbol{x}(t_k)) u_i(t_k), \tag{2.8}$$

where $p \in \mathcal{N}$ is the total number of fuzzy rules.

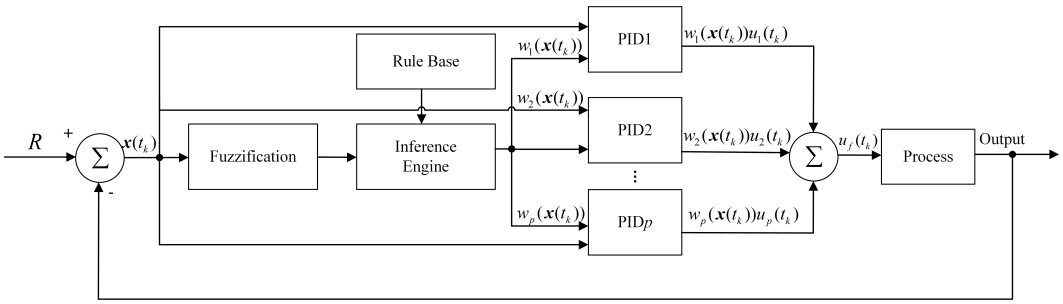

Figure 2.5: Structure of fuzzy PID controller

Interval type-2 PD controller

A typical IT2-FPD controller has similar structure to IT2 T-S model presented in Section 2.2.1. However, the consequent part of each rule in the IT2-FPD controller is expressed in the mathematical formalism of a PD controller instead of a polynomial expression, which can be expressed as follows. For i^{th} rule,
Rule i: IF $e(t_k)$ is $\widetilde{M_1^i}$ AND $\Delta e(t_k)$ is $\widetilde{M_2^i}$ THEN $y_i(t_k) = k_p^i e(t_k) + k_d^i \Delta e(t_k)$,

where $e(t_k)$ and $\Delta e(t_k)$ are the error and the change of error in k^{th} time step; k_p^i and k_d^i are the gains of PD controller in the i^{th} fuzzy rule, accordingly. The output of the IT2-FPD is defined as

$$u(t_k) = \alpha \frac{\sum_{i=1}^{P} \underline{f}^i(\boldsymbol{x}(t_k)) y_i(t_k)}{\sum_{m=1}^{P} \underline{f}^m(\boldsymbol{x}(t_k))} + \beta \frac{\sum_{i=1}^{P} \overline{f}^i(\boldsymbol{x}(t_k)) y_i(t_k)}{\sum_{m=1}^{P} \overline{f}^m(\boldsymbol{x}(t_k))}. \tag{2.9}$$

Neuro-fuzzy system

The neuro-fuzzy system presented in this thesis references the similar structure proposed in [143]. The fuzzy system is expressed by a feedforward multilayer perceptron, each layer of which achieves one step of the fuzzy operations. Fig. 2.6 is an example of a neuro-fuzzy network structure with two inputs x_1 and x_2. In the MF layer, the input values are fuzzified to the degree of membership function. The number of neurons in the MF layer is determined by the summarized number of fuzzy MFs associated with all input variables in the antecedent part. In the example shown in the Fig. 2.6, x_1 is connected with j neurons which indicates there are number of j MFs of x_1, the same applies to the x_2, where x_2 has q fuzzy MFs. Therefore, the total number of neurons in MF layer is $j + q$. In the Rule layer, each neuron is connected to different neurons from the previous MF layer based on the definition of fuzzy rules and the total number of neurons equals to the number of fuzzy rules in the rule base. The Norm layer conduct the normalization operation in the fuzzy inference system. Finally, the crisp output is generated through Output layer, which is the normalized weighted summary of outputs from all fuzzy rules.

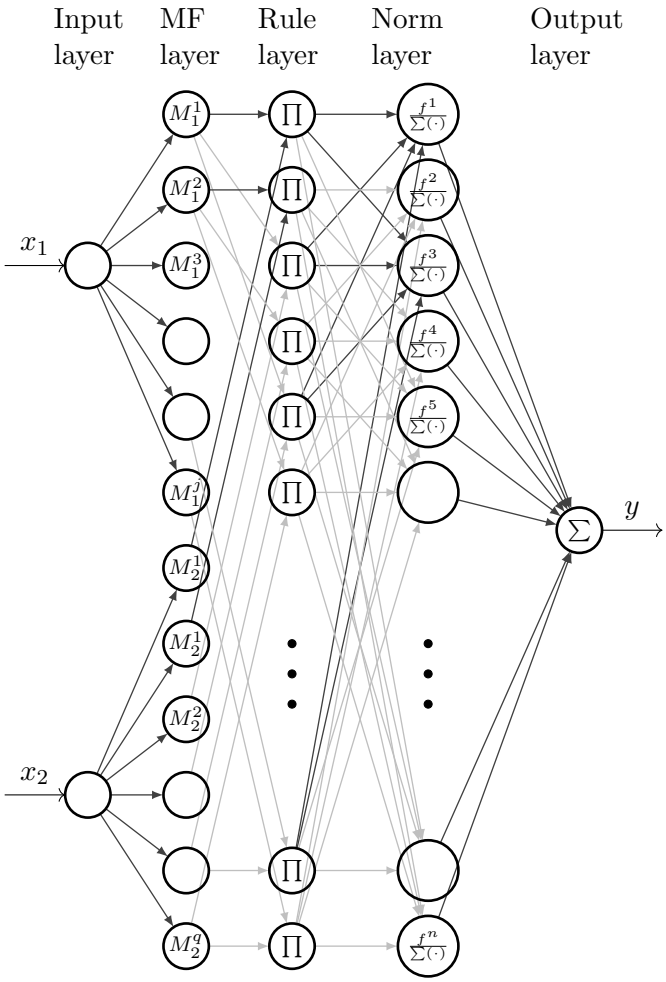

Figure 2.6: Structure of a neuro-fuzzy network

2.2.2 Reinforcement learning algorithms

In the following section, the basic elements of RL algorithms are introduced in Section 2.2.2. The details of Q-learning algorithm is provided in Section 2.2.2. The structure of AC algorithm is presented in Section 11 while the procedures of TD3 algorithm are discussed in Section 11.

Core elements in RL problems

Markov Decision Process

The RL problem is defined based on MDP with format $(\mathcal{S}, \mathcal{A}, \gamma, P, r)$, where $\mathcal{S}$ denotes the state space of the environment, $\mathcal{A}$ presents the action space, γ denotes the discount factor, P denotes the state-transition probability of the environment which is defined as $P = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a)\}$ and r_t denotes the reward function. Fig. 2.7 [95] presents the typical agent-environment interaction process in RL problems. In this process, agent observes current state $s_t \in \mathcal{S}$ and take action $a_t \in \mathcal{A}(s_t)$ according to policy function $\pi(\cdot|s_t)$. The environment transits to next state s_{t+1} according to the dynamics of the environment which follows the transition

probability P. Reward r_t is generated based on the taken action and the new state s_{t+1}. The discounted accumulated return value G is defined as Eq. 2.10, which calculates the expectation value of the discounted accumulated future rewards.

$$G = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \tag{2.10}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of random variables under policy π, $\gamma \in [0, 1]$ is discounted factor determines how the future reward is taken into consideration to current evaluation. The specific property of MDP is the state in current time step encodes enough information for the agent to make decision despite of the previous states it has been visited.

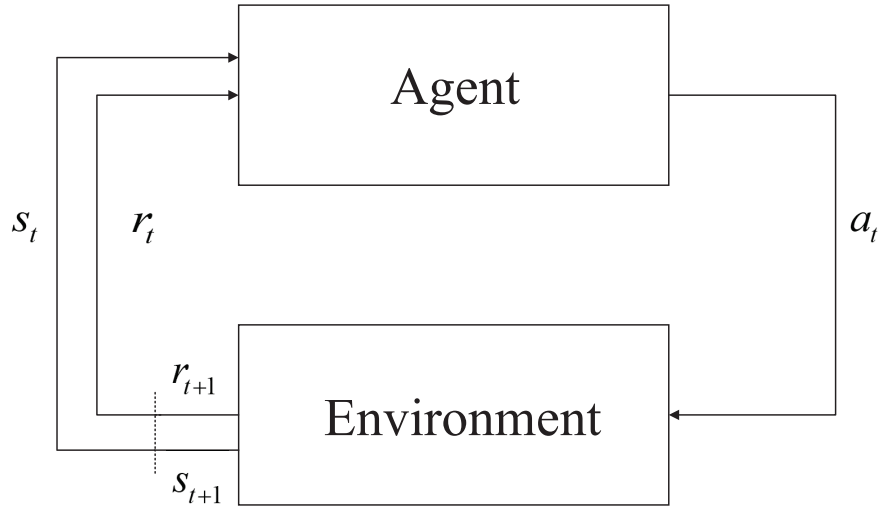

Figure 2.7: Illustration of agent-environment interaction in RL

Value functions

There are two main categories of functions for agent evaluating the current situation, which are the function of states $V^\pi(s)$ and function of state-action pairs $Q^\pi(s, a)$. State-value function $V^\pi(s)$ is the expected return value when starting from state s and following policy π afterwards, which is defined as

$$V^\pi(s_t) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi \left[\sum_{t=t_k}^{\infty} \gamma^{t-t_k} r_t \middle| s_t = s \right]. \tag{2.11}$$

The optimal state-value function is defined as

$$V^*(s) = \max_\pi V^\pi(s). \tag{2.12}$$

Action-value function $Q_\pi(s_t, a_t)$ is defined as the expectation value of accumulated discount return value when in state s_t taking action a_t and following the policy π.

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a] = \mathbb{E}_{\pi}\left[\sum_{t=t_k}^{\infty} \gamma^{t-t_k} r_t \middle| s_t = s, a_t = a\right]. \tag{2.13}$$

where γ is discounted factor determines how the future reward is taken into consideration to current evaluation.

The optimal action-value function is defined as

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a). \tag{2.14}$$

The aim of the RL is to find the optimal policy π^* which maximize the return value R.

Policy functions

The parameterized policy function π is a mapping which projects state s to the distribution of probabilities for all potential actions $a \in \mathcal{A}$ [109], which gives

$$a \sim \pi(\cdot|s) \tag{2.15}$$

for stochastic policy and

$$a = \pi(s) \tag{2.16}$$

for deterministic policy.

Q-learning algorithm

The Q-learning Algorithm is an off-policy value-based learning algorithm of RL which was firstly proposed in 1992. This algorithm has been one of the important branches in RL algorithms. In this algorithm, each state-action pair is evaluated by Q-function. All the Q-values associate with the state-action pair are stored in the Q-table. The agent chooses action based on ϵ-greedy method defined in Eq. 2.17.

$$a_{t+1} = \begin{cases} \text{random } a \in \mathcal{A} & \text{if } \xi < \epsilon \\ \arg\max\limits_{a} Q(s_{t+1}, a) & \text{otherwise,} \end{cases} \tag{2.17}$$

where ξ is random value generated between range $[0, 1]$ and $\epsilon \in [0, 1]$ is a preset variable that determines the degree of exploration in policy. The Q function is updated according to the Bellman equation which is shown in Eq. 2.18.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)], \tag{2.18}$$

where α denotes the learning rate and γ presents the discount factor. The general procedures of the Q-learning algorithm is presented in Algorithm 1, which is a revised version based on [6].

Algorithm 1: Q-learning Algorithm

```
Initialize Q(s, a), ∀a ∈ A, ∀s ∈ S, arbitrarily;
For every episode;
repeat
    Initialize state s_t;
    repeat
        Perform action a_t following ε-greedy method;
        Observe s_{t+1} and r_{t+1};
        Update Q values according to Eq. (2.18);
        s_t ← s_{t+1}
    until s_t reaches terminal state s_T;
until episode ends;
```

Actor-critic algorithm

AC algorithm uses two individual functions to approximate the policy function value function individually. The value function approximating state or state-action value is referred as "*critic*" while the policy function is denoted as "*actor*". The AC algorithm implements the bootstrapping technique to accelerate the training speed. The TD error is calculated for updating the target critic value. Additionally, the TD error value guides the direction in policy optimization. The positive TD error indicates the enhancement of the chosen action while the negative value means to weaken the same action in the future policy. The general paradigm of the AC algorithm is illustrated in Fig. 2.8, the original version of which is from [109].

Twin delayed deep deterministic policy gradient algorithm

TD3 algorithm is an improved off-policy RL algorithm based on DDPG algorithm proposed in 2015 [110]. This approach is applied in continuous domain for both input and output variables. Though significant achievements have been made by DDPG algorithm, this algorithm shows limitation in overestimation of the approximated value and sensitivity to the hyper-parameters. To solve the instability caused by the aforementioned issues, the improvements made by TD3 algorithm proposed are as follows.

1) The double critic networks structure was implemented by referencing the Double-DQN algorithm to reduce the overestimation bias in function approximation.

2) Target policy smoothing strategy is applied, where another target policy target network is set up with clipped noise added.

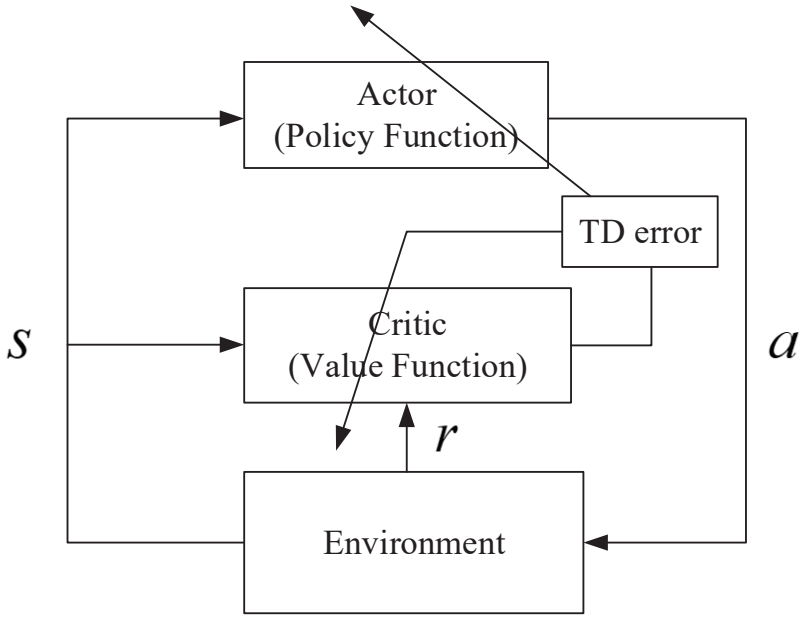

Figure 2.8: Illustration of actor-critic algorithm

3) Delayed policy updating is implemented to reduce the accumulated error before conducting policy optimization.

The detailed procedures of TD3 are shown in Algorithm 2. The Q-value is approximated by two separate critic networks denoted as $Q_1(s, a|\theta_1)$ and $Q_2(s, a|\theta_2)$, of which corresponding target networks denoted as Q'_1, Q'_2. The control policy is generated by deterministic policy function $\pi(s|\phi)$ with a Gaussian noise added to the output to encourage the exploration. Target actor network π' is initialized to increase the stability of the algorithm. Samples are stored in replay buffer $\mathcal{R}$ and in each iterative update process, a mini batch with K samples are chosen for update. The minimum value between two target critic networks are chosen for the critic update as to reduce the over estimation issue in the DDPG algorithm. Target actor network is added with clipped noise and updated d steps. The parameters in both target critic and actor networks are softly updated.

Prioritized experience replay

Experience replay is one of the widely used acceleration techniques for off-policy RL algorithms. The samples are stored in replay memory where all the samples can be repeatedly used more than once. This mechanism breaks the correlations among samples in sequence and increases the sample efficiency [144]. In regular experience replay methodology the experiences are sampled with uniform possibility, however the samples are not equally valued from the training perspective because of redundant and task-relevant issue. Therefore, as to further improve the learning efficiency, the prioritized experience replay (PER) [145] was proposed in 2015 to

Algorithm 2: TD3 Algorithm

Initialize critic networks $Q_1(s, a|\theta_1)$ and $Q_2(s, a|\theta_2)$, actor network $\pi(s|\phi)$;
Initialize target critic networks Q'_1, Q'_2 with $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$ and target actor network π' with $\phi' \leftarrow \phi$;
Initialize replay buffer $\mathcal{R}$;
For every episode:
Initialize state s;
while $t = 1 : T$ **do**
    Select action $a \sim \pi(s|\phi) + \mathcal{N}(0, \sigma)$;
    Observe next state s' and reward r;
    Store transition tuple (s, a, r, s') in $\mathcal{R}$;
    Sample mini-batch of K transitions;
    Get $a'_i \sim \pi'(s'_i|\phi') + \epsilon$,where $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$;
    Set $y_i = r_i + \gamma \min\{Q'_{1i}(s'_i, a'_i|\theta'_1), Q'_{2i}(s'_i, a'_i|\theta'_2)\}$;
    Update critic network Q_m by minimizing loss
    $L_m = \frac{1}{K}\sum_{i=1}^{K}(y_i - Q_m(s_i, a_i|\theta_m))^2$, $m = 1, 2$;
    while t mod $d = 0$ **do**
        Update actor network π following gradient
        $\nabla_\phi = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q_1(s, a|\theta_1)|_{s=s_i, a=\pi(s_i|\phi)} \nabla_\phi \pi(s_i|\phi)$;
        Update target networks:
        $\theta'_m \leftarrow \tau\theta_m + (1 - \tau)\theta'_m$, $m = 1, 2$
        $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$;
    end
    $s \leftarrow s'$;
end

replay experience based on prioritization. In this approach, the possibility of samples being replayed is proportional to the associated priority, which is defined as

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \tag{2.19}$$

where $P(i)$ and p_i represent the probability and the priority of the i^{th} sample. k is the total number of samples in the replay buffer. $\alpha \in [0, 1]$ denotes the level of prioritization having effect on the probability calculation, $\alpha = 0$ brings PER back to ordinary experience replay without prioritizing.

There are two criteria used to evaluate the priority of the samples. The priority is determined by TD error

$$p_i = |\delta_i| + \epsilon, \tag{2.20}$$

where δ represents the TD error, ϵ is a small constant number which avoids p_i from zero value, or rank based

$$p_i = \frac{1}{L(i)}, \tag{2.21}$$

where $L(i)$ is the ranking number of the i^{th} sample in the replay buffer determined by the $|\delta|$ value. As to address the loss of diversity lead by the prioritized replay scheme, the importance sampling (IS) trick is introduced, which adds a corrective coefficient w_i to the probability. The w_i is defined as

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta, \tag{2.22}$$

where $\beta \in [0, 1]$ determines the degree for compensation.

2.2.3 Inverted pendulum system

As shown in Fig. 2.9, the inverted pendulum system consists of a cart connected with a pole through a hinge. The cart can only move in x direction while the pole rotates in $x - y$ plane. A horizontal force $u(t)$ is applied to the cart in the same direction as the movement of the cart to balance the pole. The stable equilibrium is when pole stays up right and the cart goes back to origin. Taking the state of the inverted pendulum system as $\boldsymbol{x_{cp}}(t) = [x_1(t), x_2(t), x_3(t), x_4(t)]$, the dynamic model of the inverted pendulum system is given as follows [143],

$$\dot{x}_1(t) = x_2(t), \tag{2.23}$$

$$\dot{x}_2(t) = \frac{\begin{pmatrix} -F_1(M+m)x_2(t) - m^2l^2x_2(t)^2\sin x_1(t) \\ \cos x_1(t) + F_0mlx_4(t)\cos x_1(t) + (M+m)mgl \\ \sin x_1(t) - ml\cos x_1(t)u(t) \end{pmatrix}}{(M+m)(J+ml^2) - m^2l^2(\cos x_1(t))^2}, \tag{2.24}$$

$$\dot{x}_3(t) = x_4(t), \tag{2.25}$$

$$\dot{x}_4(t) = \frac{\begin{pmatrix} F_1mlx_2(t)\cos x_1(t) + (J+ml^2)mlx_2(t)^2 \\ \sin x_1(t) - F_0((J+ml^2)x_4(t) - m^2gl^2 \\ \sin x_1(t)\cos x_1(t) + (J+ml^2)u(t) \end{pmatrix}}{(M+m)(J+ml^2) - m^2l^2(\cos x_1(t))^2}, \tag{2.26}$$

where $x_1(t)$ is the displacement of angle (rad); $x_2(t)$ is the angular velocity (rad/sec); $x_3(t)$ is the position of the cart (m); $x_4(t)$ is the linear velocity of the cart (m/s); $u(t)$ is the force applied to the cart (N); M is the mass of the cart (kg); m is the mass of the pendulum (kg); l is the half length of the pendulum (m); J is the moment of inertia of the pendulum, $J = \frac{1}{3}ml^2$ kgm^2; F_0 is the friction factor of the cart (N/m/s); F_1 is the friction factor of the pendulum (N/rad/s); g is the gravity acceleration, $g = 9.8$m/s^2.

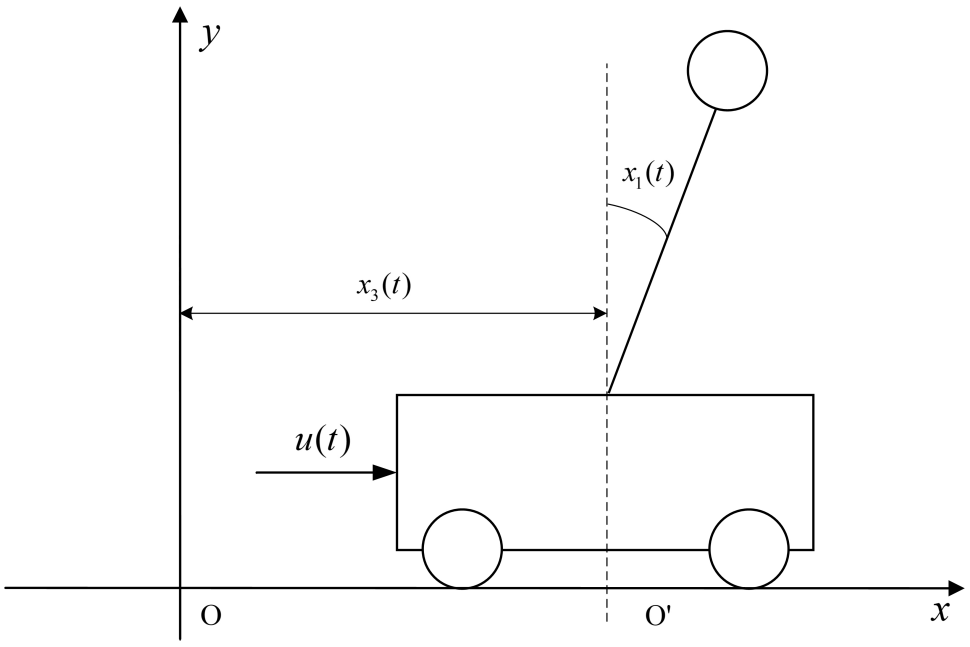

Figure 2.9: Inverted pendulum system

Chapter 3

Adaptive PID Controller based on Q-learning Algorithm

Regarding to development of RL based PID controller, training with RL algorithms always require high computational ability, which can be challenging for many application environments supporting PID controllers. Therefore, proposing an efficient RL based tuning method as well as keeping low computational cost for multiple PID controllers remains an open issue. Based on the challenges stated above, in this chapter, an adaptive PID controller based on Q-learning algorithm (Q-PID controller) is proposed to balance the inverted pendulum system in simulation environment. This controller was trained using Q-learning algorithm, where the Q-tables after training are utilized to adjust the parameters of linear PID controllers based on the various states of the system during the control process. The adaptive PID controller based on Q-learning algorithm was trained from a set of fixed initial positions and was able to balance the system starting from a wide range of random initial positions different from the ones used in the training session. The performance of the proposed controller is compared with the conventional PID controller and the controller only applies Q-learning algorithm on the inverted pendulum platform, which indicates the advantage of the adaptive PID controller based on Q-learning algorithm both in the generality and transient response. The structure of rest of the chapter is organized as follows. The methodology and technical details of Q-PID controller are described in Section 3.1 and the simulation results of three different controllers are illustrated and compared in Section 3.2. Finally conclusions are drawn in Section 3.3.

3.1 Adaptive Q-PID controller

3.1.1 Structure of Q-PID controller

The adaptive PID controller based on Q-learning algorithm proposed in this paper was designed to balance the inverted pendulum system. The architecture of the controller is shown in Fig. 3.1. In the higher level, there are several Q-tables optimized through Q-learning algorithm, each of which is associated with one of the parameters of the linear PID controllers in the lower level. The output of linear PID controllers is utilized as the direct signal to control the inverted pendulum platform.

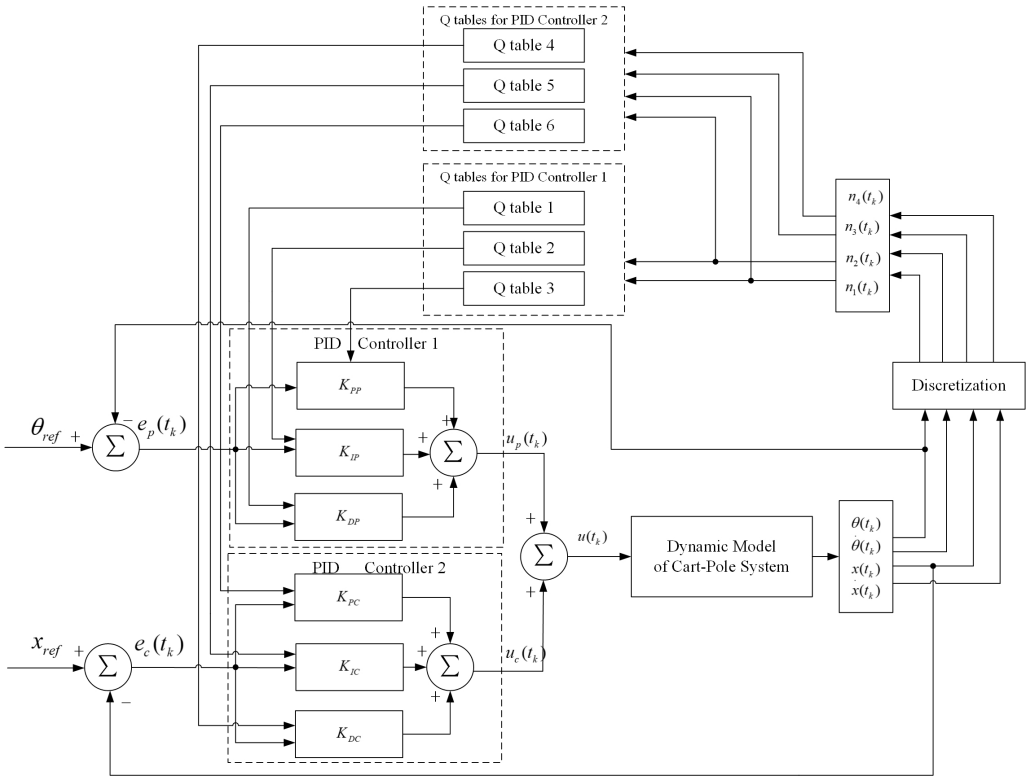

Figure 3.1: The architecture of control system with Q-PID controller and inverted pendulum system

As shown in Fig. 3.1, the position of pendulum and the cart are controlled by two separate controllers, which are PID controller 1 and PID controller 2. At time step t_k, PID controller 1 takes $\theta_{ref} = 0$ as reference, the error of the pendulum angle $e_p(t_k)$ is defined as

$$e_p(t_k) = \theta_{ref} - \theta(t_k) \tag{3.1}$$

and output of PID controller 1 $u_p(t_k)$ is defined as

$$\begin{aligned} u_p(t_k) =& K_{PP} e_p(t_k) + K_{IP} \sum_{i=0}^{k} e_p(t_i) \Delta t_k \\ &+ K_{DP} \frac{e_p(t_k) - e_p(t_{k-1})}{\Delta t_k}, \end{aligned} \tag{3.2}$$

where K_{PP}, K_{IP} and K_{DP} are the proportional, integral and derivative gain of PID controller 1, respectively, Δt_k is the time interval where $\Delta t_k = t_k - t_{k-1}$.

PID controller 2 takes $x_{ref} = 0$ as reference, the error of the cart position $e_c(t_k)$ is defined as

$$e_c(t_k) = x_{ref} - x(t_k) \tag{3.3}$$

and outputs $u_c(t_k)$ is defined as

$$\begin{aligned} u_c(t_k) =& K_{PC} e_c(t_k) + K_{IC} \sum_{i=0}^{k} e_c(t_i) \Delta t_k \\ &+ K_{DC} \frac{e_c(t_k) - e_c(t_{k-1})}{\Delta t_k}, \end{aligned} \tag{3.4}$$

where K_{PC}, K_{IC} and K_{DC} are the proportional, integral and derivative gain of PID controller 2.

The control signal from two PID controllers, $u_p(t_k)$ and $u_c(t_k)$, are summed up as force $u(t_k)$, which is applied to balance the inverted pendulum system. After obtaining the states of the system $(\theta(t_k), \dot{\theta}(t_k), x(t_k), \dot{x}(t_k))$ from the simulation environment of inverted pendulum system, the continuous state vector is discretized into four discrete variables $(n_1(t_k), n_2(t_k), n_3(t_k), n_4(t_k))$, which are taken as the presentation of state $s_{t_k} \in \mathcal{S}$ in the Q-learning algorithm.

There are six learned Q tables in the high level, where each Q table is associated with one specific gain of PID controllers. Q table 1 to Q table 3 are associated with the derivative, integral and proportional gain of the PID controller 1 controlling the pendulum. Similarly, Q table 4 to Q table 6 are associated with the derivative, integral and proportional gain of the PID controller 2 which controls the cart, accordingly. To be noted, Q table 1 to Q table 3 only take $n_1(t_k)$ and $n_2(t_k)$ as state vector while Q table 4 to Q table 6 take whole state variables $n_1(t_k)$, $n_2(t_k)$, $n_3(t_k)$ and $n_4(t_k)$ into consideration. Each learned Q table generates the optimal gain value for the corresponding controller according to various state.

3.1.2 Adaptive learning rate

As to improve the efficiency of convergence, the adaptive learning rate algorithm named Delta-Bar-Delta [146] was implemented. The algorithm is defined as

$$\Delta\alpha_{t_k} = \begin{cases} \kappa, & \text{if } \overline{\delta}_{t_{k-1}} \delta_{t_k} > 0 \\ -\phi\alpha_{t_k}, & \text{if } \overline{\delta}_{t_{k-1}} \delta_{t_k} < 0 \\ 0, & \text{if } \overline{\delta}_{t_{k-1}} \delta_{t_k} = 0 \end{cases} \tag{3.5}$$

where $\Delta\alpha_{t_k}$ is the increment of the learning rate in the k^{th} time step t_k; κ is a positive constant value to increase the learning rate; ϕ is a positive constant value denoting the discounting factor; δ_{t_k} is the TD error in time step t_k, $\delta_{t_k} = r_{t_{k+1}} +$

$\gamma \max_a Q(s_{t_{k+1}}, a) - Q(s_{t_k}, a_{t_k})$; $\overline{\delta}_{t_k} = (1-\varphi)\delta_{t_k} + \varphi\overline{\delta}_{t_{k-1}}$. The learning rate in time step t_{k+1} is updated as

$$\alpha_{t_{k+1}} = \alpha_{t_k} + \Delta\alpha_{t_k}. \tag{3.6}$$

By applying this rule, the learning rate will be updated by comparing the current TD error with the accumulated TD error from the previous steps. When the learning rate becomes too large, the increment of the learning rate changes sign and reduces the learning rate. On the other hand, if the learning rate is too small, the learning rate keeps changing in the previous trend and speeds up the convergence.

The adaptive learning rate scheme are applied in the training process for all the Q tables but with different parameter settings. Q tables related to PID controller 1 shares the same set of parameters while the other Q tables share another set of parameters. The setting of parameters related to the Delta-Bar-Delta algorithm are defined in Table 3.1, where α_0 is the initial learning rate set in the Q-learning algorithm, κ is a positive constant value to increase the learning rate, ϕ is the positive constant value denoting the discounting factor, φ is the weighting factor used in $\overline{\delta}_{t_k} = (1-\varphi)\delta_{t_k} + \varphi\overline{\delta}_{t_{k-1}}$ for learning rate updating.

Table 3.1: Parameter setting in adaptive learning rate scheme

	Q tables of PID controller 1	Q tables of PID controller 2
α_0	0.015	0.3
κ	$0.1\alpha_0$	$0.002\alpha_0$
ϕ	0.5	0.5
φ	0.5	0.5

3.1.3 Pseudo code of training procedure

One of the crucial procedures in designing the proposed method is the training process of the Q tables, which maps the current states to actions (different values of gains in this situation). The whole training process is shown in pseudo code form in Algorithm 3 with adaptive learning rate method Delta-Bar-Delta implemented as the acceleration technique. In the beginning of the training procedure, the Q-values in each Q-tables are initialized with 0 values, while the learning rate α_1 for Q-tables related to PID 1, learning rate α_2 for Q-tables related to PID 2 and the exploration rate ϵ for policy generation are initialized to the preset values. In each episode, the state of the system is set to certain initial position and the episode will continue step by step until reaching the maximum time step or getting intermediate termination when the state is outside the legal range. In each time step, the continuous state s_{t_k} are discretized to n_{t_k}, based on which the action for each Q-table is chosen following the ϵ-greedy scheme. As previously stated, the actions from Q-tables are the gain values for two PID controllers, when given the combination of gains, two controllers generate control signal $u(t_k)$ to balance the inverted pendulum system.

After observing the new discretized state $n_{t_{k+1}}$, two reward signals r_p and r_c are received while two learning rate α_1 and α_2 are decided base on Delta-Bar-Delta scheme for the two sets of Q-tables, which are the set from Q_1 to Q_3 and the set from Q_4 to Q_6. According to the variables collected above, all the Q-tables are optimized following the update rule defined in Eq. 2.18. Until this point, the optimization process is finished for one time step and whole procedure continuously repeated in the following time step.

Algorithm 3: The Training process of adaptive PID controller based on Q-learning algorithm

Initialize $Q_i(s,a) = 0$, $\forall a \in \mathcal{A}, \forall s \in \mathcal{S}$, $i = 1, 2, ..., 6$;
Initialize learning rates α_1 and α_2;
Initialize exploration rate ϵ;
while *episode* $<$ *maxepisode* **do**
    $k = 0$;
    Initialize $s_{t_k}(\theta(t_k), \dot{\theta}(t_k), x(t_k), \dot{x}(t_k))$;
    Decay ϵ (when *episode* > 0.6*maxepisode* ϵ=0);
    for $k = 1; k \leq maxtime; k++$ **do**
        Discretization of state s_{t_k}, obtain $n_{t_k}(n_1(t_k), n_2(t_k), n_3(t_k), n_4(t_k))$;
        for $i = 1; i \leq 3; i++$ **do**
            According to $n_1(t_k), n_2(t_k)$, choose action a_i following ϵ-greedy policy;
        end
        for $i = 3; i \leq 6; i++$ **do**
            According to $n_1(t), n_2(t_k), n_3(t_k), n_4(t_k)$, choose action a_i following ϵ-greedy policy;
        end
        Obtain force $u(t_k)$ according to Eq. 3.1 to 3.4;
        Observe new state $s_{t_{k+1}}(\theta(t_{k+1}), \dot{\theta}(t_{k+1}), x(t_{k+1}), \dot{x}(t_{k+1}))$;
        Receive reward r_p for $Q_1(s,a), Q_2(s,a)$ *and* $Q_3(s,a)$;
        Receive reward r_c for $Q_4(s,a), Q_5(s,a)$ *and* $Q_6(s,a)$;
        Discretization of state $s_{t_{k+1}}$, obtain $n_1(t_{k+1}), n_2(t_{k+1}), n_3(t_{k+1}), n_4(t_{k+1})$;
        Update learning rate α_1 for $Q_1(s,a), Q_2(s,a)$ *and* $Q_3(s,a)$;
        Update learning rate α_2 for $Q_4(s,a), Q_5(s,a)$ *and* $Q_6(s,a)$;
        Update $Q_1(s,a), Q_2(s,a)$ *and* $Q_3(s,a)$ using r_p and α_1;
        Update $Q_4(s,a), Q_5(s,a)$ *and* $Q_6(s,a)$ using r_c and α_2;
        $s_{t_k} \leftarrow s_{t_{k+1}}$
    end
end

The following content specified some details in the training process.

Discretization

Each continuous variable is divided into several buckets, the values within the same bucket are regarded as the same discrete state. The rule used in buckets division is

Table 3.2: Values set for discretization

Values	X_{min}	X_{max}
$\theta(t_k)$	$-\frac{45}{180}\pi$	$\frac{45}{180}\pi$
$\dot{\theta}(t_k)$	-15	15
$x(t_k)$	-3	3
$\dot{x}(t_k)$	-15	15

defined as

$$n = \begin{cases} 1 & \text{if } x_{con} < X_{min} \\ 10 & \text{if } x_{con} > X_{max} \\ \lfloor \frac{x_{con}}{X_{max} - X_{min}} \times N \rfloor + 1 & \text{if } X_{min} \leq x_{con} \leq X_{max}, \end{cases} \tag{3.7}$$

where $\lfloor x \rfloor = max\{n \in \mathbb{Z} | n \leq x\}$; n denotes the discrete variable; x_{con} denotes the continuous variable; X_{min} and X_{max} are the lower and upper bound of x_{con}, respectively; N denotes the number of buckets each variable is divided into, $N = 10$ in this situation. The number of the buckets is decided depending on the simulation performance.

The values set for discretization are shown in Table 3.2.

$\epsilon -$ greedy Method

The generation of actions for all six Q tables generate follow the $\epsilon -$ greedy method, which sis defined by

$$A = \begin{cases} \text{random action} & \text{if } \xi < \epsilon \\ \arg\max_a Q(s, a) & \text{otherwise,} \end{cases} \tag{3.8}$$

where ξ is a randomly generated number in each time step subject to normal distribution, $0 \leq \xi \leq 1$.

As to speed up the convergence speed and stablize the tranning process, the value of ϵ gradually decays with the increment of training episodes and is finally set to zero. The details are defined as

$$\epsilon(eps) = \begin{cases} \frac{1}{1+e^{eps}} + 0.001 & eps < 0.6 maxepisode \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

where *eps* represents the number of current episode; *maxepisode* is the maximum number of episodes.

Reward Scheme

The reward schemes are different according to different PID controllers. The Q tables related to PID controller 1 share the same reward scheme while the other Q tables share another reward scheme. The reward scheme for the Q tables associated with the gains of PID controller 1 are defined as

$$r_p = \begin{cases} 1 & \text{if } |\theta(t_{k+1})| < |\theta(t_k)| \text{ and } |\theta(t_{k+1})| > \theta_{lim} \\ 2 & \text{if } |\theta(t_{k+1})| \leq \theta_{lim} \\ 0 & \text{otherwise,} \end{cases} \tag{3.10}$$

where θ_{lim} is the threshold set for the angle of the pendulum, $\theta_{lim} = 0.01$ in this situation.

The reward scheme for the Q tables associated with the gains of PID controller 2 are defined as

$$R_c = \begin{cases} 1 \text{ if } |\theta(t_{k+1})| < |\theta(t_k)| \text{ and } |x(t_{k+1})| < |x(t_k)| \text{ and } |x(t_{k+1})| > x_{lim} \\ 2 \text{ if } |x(t_{k+1})| \leq x_{lim} \\ 0 \text{ otherwise,} \end{cases} \tag{3.11}$$

where x_{lim} is the threshold set for position of the cart, $x_{lim} = 0.1$ in this situation.

3.2 Simulation Results

In this section, the simulation tests and results of implementing adaptive PID controller based on Q-learning algorithm (named as QPID controller in the following content for the simplicity of expression) are demonstrated. The structure of this section is organised as follows. Section 3.2.1 provides the setting of the parameters in the simulation environment, which includes parameter setting of the inverted pendulum system and hyper-parameters of different categories of controllers in the training process. Section 3.2.2 provides the details of the training process. Section 3.2.3 describes the simulation tests set for controllers after training and Section 3.2.4 compares the controlling performance of QPID controller against Q controller and PID controller in the simulation tests.

3.2.1 Setting of Parameters

Parameter settings for environment

The parameters set for the inverted pendulum system reference research [143] and are illustrated in Table 3.3.

Table 3.3: Parameters of inverted pendulum system

Variables	Values
M	1.3282kg
m	0.22kg
l	0.304m
F_0	22.915N/m/s
F_1	0.007056N/rad/s
g	9.8m/s^2

Parameter of Q-learning training process

The initialized parameters of QPID controller are listed in Table 3.4. The episode will be interrupted if one of the following condition is satisfied, $|\theta(t)| > 0.5\pi$ or $|x(t)| > 10.0$m. These are considered as non-ideal circumstances in real physical world and appropriate determinations could help improve the training efficiency in the simulation experiments.

Table 3.4: Parameters set for training process

Variables	Values
$\theta(t_0)$	$\pm\frac{45}{180}\pi$
$\dot{\theta}(t_0)$	0
$x(t_0)$	$\pm$0.1m
$\dot{x}(t_0)$	0
N	50
K_{PP}	$[-3000, 0]$
K_{IP}	$[-300, 0]$
K_{DP}	$[-3000, 0]$
K_{PC}	$[-2000, 0]$
K_{IC}	$[-5, 0]$
K_{DC}	$[0, 2000]$
Maximum episodes	6000
Maximum timesteps	8000
Δt	10ms
γ	0.99

Parameters of Q Controller

The Q controller takes four variables of the inverted pendulum system $(\theta(t), \dot{\theta}(t), x(t), \dot{x}(t))$ as input and generates single control signal which is directly applied to balance the system. $\theta(t)$ is divided into 6 buckets (with $-\frac{21}{180}\pi, -\frac{6}{180}\pi, 0, \frac{21}{180}\pi, \frac{6}{180}\pi 0$ as dividing points); $\dot{\theta}(t)$ is divided into 3 buckets (with $\pm\frac{6}{180}\pi$ as dividing points); $x(t)$ is divided into 3 buckets (with $\pm$2.4 as dividing points); $\dot{x}(t)$ is divided into 3 buckets (with $\pm$0.5 as dividing points). The continuous output range is mapped into 10 discrete action buckets which evenly distribute among $[-100N, +100N]$. The reward scheme is defined as,

$$r = \begin{cases} 1 & \text{if } |\theta(t)| < \frac{6}{180}\pi \text{ or } |x(t)| < 0.1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

The relevant parameters set for Q controller is given in Table 3.5.

Table 3.5: Parameters of Q controller

Variables	Values
$\theta(t_0)$	$\frac{10}{180}\pi$
$\dot{\theta}(t_0)$	0
$x(t_0)$	0
$\dot{x}(t_0)$	0
Maximum episodes	30000
Maximum time step	2000
Δt	10ms
γ	0.99
α	0.01
ϵ	$min(1, \frac{500}{episode})$

Parameters for PID Controllers

The PID controllers for comparison purpose are consisted of two linear PID controllers, which control the position of the pole and the position of the cart separately. The gains of two linear PID controllers were obtained by manually tuned according to Ziegler-Nichols method [147] and are shown in Table 3.6.

Table 3.6: Parameters of PID controllers

Gains	PID controller 1	PID controller 2
K_P	-270	-80
K_I	-10	-0.5
K_D	-1500	2000

3.2.2 Training results

Training Process

The QPID controller was trained from four fixed initial position points ($\theta(t_0) = \pm\frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = \pm 0.05\text{m}, \dot{x}(t_0) = 0$) with random noise $\Delta\theta \in [-0.04\text{rad}, 0]$ added to each fixed point as disturbances. The learned Q tables were implemented to generate the optimal gains of the PID controllers.

Similarly, the Q controller was trained from four fixed initial position points ($\theta(t_0) = \pm\frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = \pm 0.05\text{m}, \dot{x}(t_0) = 0$) with random noise $\Delta\theta \in [-0.04\text{rad}, 0]$ added to each fixed point as disturbances. To be noticed, Q controller is trained with smaller initial θ values $\pm\frac{10}{180}\pi$ instead of $\pm\frac{45}{180}\pi$. This is because

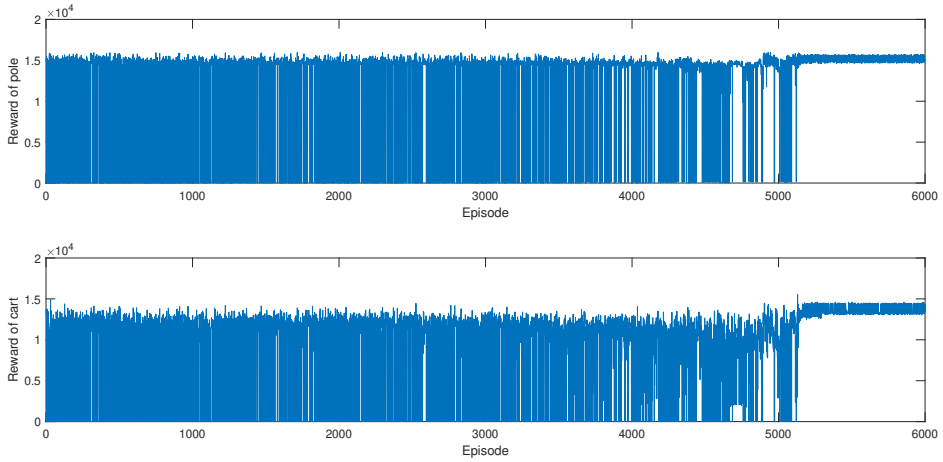

Figure 3.2: Learning curve of QPID controller

the Q controller shows limitation in balancing the inverted pendulum system back to reference point with larger initial angle values during the training process. Any initial θ value larger than $\pm\frac{10}{180}\pi$ would lead to the failure of the task and the diverge of training process. Therefore, only $\theta(t_0) = \pm\frac{10}{180}\pi$ are chosen.

Learning Curves

The learning performance of adaptive PID controller based on Q-learning algorithm is indicated by the learning curve, which plots the accumulated reward obtained with the increment of the episode. The aim of comparing learning curves is to present the learning efficiency of various categories of controllers in the training process. The convergence of the learning curve indicate the success of Q-learning algorithm in finding proper combination of gains for controllers and the maximum time step used to reach the convergence indicate the learning efficiency in the training process. The learning curve of QPID controller is presented in Fig. 3.2 and the learning curve of the Q controller is shown in Fig. 3.3, both of which show the success in convergence.

The learning curve of QPID controller shows converging trend around 4000 episodes after the exploration rate reduces to zero and stays stable after 5000 episodes, which indicates the success of finding optimal solutions for PID controllers through Q-learning algorithm. Nevertheless, though the Q controller also gradually converges after 4000 episodes, the frequency that drop occurs is significant high around 5000 episodes. These drops progressively decrease with the increase of episodes but still not fully disappear until the end of the training process.

3.2.3 Simulation Test

As to test the performance of the Q-PID controller after training, random initial position test is conducted in inverted pendulum system. In the random initial position test, the QPID controller after training was tested to balance the inverted pendulum

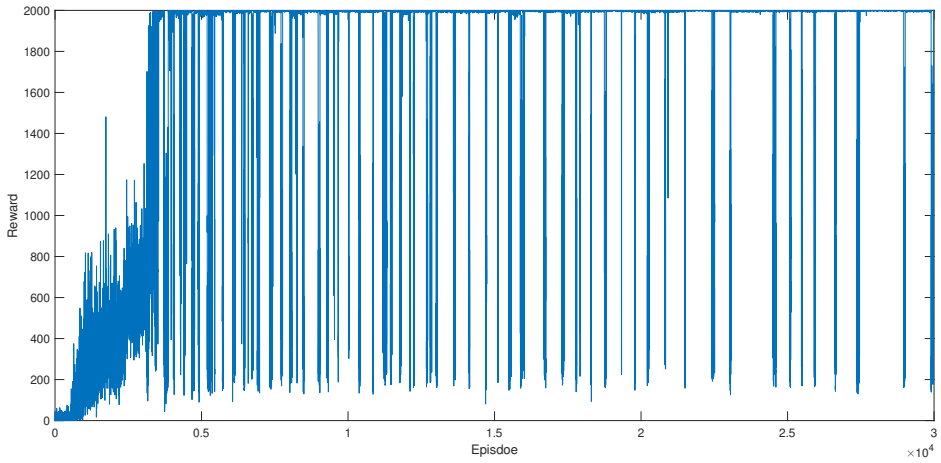

Figure 3.3: Learning curve of Q Controller

system from a series of fixed initial positions, which are the combinations of $\theta(t_0) \in \left\{\pm\frac{45}{180}\pi, \pm\frac{30}{180}\pi, \pm\frac{15}{180}\pi, 0\right\}, \dot{\theta}(t_0) = 0, x(t_0) \in \{\pm 0.05\text{m}, \pm 0.02\text{m}, 0\text{m}\}, \dot{x}(t_0) = 0$, back to reference point $\theta = 0, \dot{\theta} = 0, x = 0, \dot{x} = 0$. Additionally, as for comparison purpose, two other types of controllers are applied to the system to compare the controlling performances. The first type of controller employs the tabular Q-learning algorithm (named as Q controller in the following content for the simplicity of expression) and the second one is the conventional PID controller.

3.2.4 Comparison of controlling performance

After the training process, the Q tables with optimized Q values are utilized to provide gains of PID controllers aiming to balance the inverted pendulum system from different initial position points. The transient response curves of the inverted pendulum under control of QPID controller is compared with Q controller and PID controller accordingly. In each comparison, the values of variables $\theta, \dot{\theta}, x, \dot{x}$ are drawn separately against the change of time.

(1) Comparison of QPID controller with Q controller

Considering the limit initial θ_0 values for QPID controller and Q controller found in the training procedure are different, where the limit initial position value for QPID controller is $\theta(t_0) = \frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$ while for the Q controller is $\theta(t_0) = \frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$. Therefore, the minimum limit initial position is chosen as testing point to compare the performance of QPID controller against Q controller. Fig. 3.4 compares the response curve of the system controlled by Q controller and QPID controller from initial position $\theta(t_0) = \frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$ and the forces generated by two controllers are shown in Fig. 3.5. The key characteristics of the response curves are summarized in Table 3.7. In the transient response test, the controller is expected to balance both the positions of pole and cart in the inverted pendulum system

back to and stabilize at the equilibrium point. However, as shown in Fig. 3.4, when comparing the position of the pendulum controlled by two controllers, though Q controller is able to reach the reference point but to fails to stabilize the pendulum at this position, where significant fluctuations with 0.026rad amplitude are noticed until the end of the episode. While QPID controller successfully stabilize the pendulum to the reference point after 69 time steps. The fluctuations can also be noticed from the control signals generated by two controllers shown in Fig. 3.5. Additionally, when comparing the position of the cart, QPID controller drives the cart back to origin point after 772 time steps without steady state error, while the cart position controlled by Q controller keeps drifting around the reference point until the end of the episode, which leads to an absolute steady state error 0.1351m. In conclusion, QPID controller successfully balances both the pendulum and the cart back to equilibrium point and stabilize the system at the reference point, however, Q controller reveals limitations in stabilizing the system where the position of the pendulum fluctuates around the reference point and the cart position fails to back to equilibrium point until the end of the episode, which indicates the advantage of QPID controller against Q controller in this simulation environment.

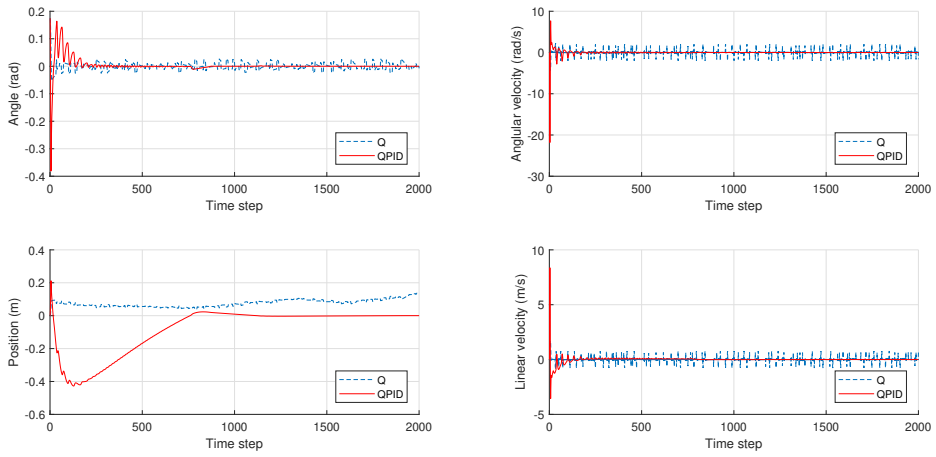

Figure 3.4: Response curves of Q controller and QPID controller (initial position $\theta(t_0) = \frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$)

As for better visualize the adaption of the gains of QPID controller during the whole controlling process, Fig. 3.6 plots the values of all six parameters of two PID controllers in each time step, where the red crosses denote the chosen values of the gains of PID controllers at t_k time step and the blue lines indicate the changing trend. Several switching points appear in various responding stage, showing different set of gains are chosen according to the change of environment states, which indicates the adaptive properties of proposed QPID controller.

(2) Comparison of QPID controller with PID controller

Initial position $\theta(t_0) = \frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0.05\text{m}, \dot{x}(t_0) = 0$ is chosen as the testing position to compare the performance of QPID controller against PID

Table 3.7: Key characteristics of response curves (Q controller and QPID controller)

Characteristics	Q controller	QPID controller
Overshoot (θ)	0.0479 (rad)	0.3763 (rad)
Undershoot (θ)	0.1316 (rad)	0.1574 (rad)
Rise time (θ)	6 (time step)	4 (time step)
Settling time (θ)	4 (time step)	69 (time step)
Steady error(θ)	0.026 (rad)	0 (rad)
Overshoot (x)	0.0930 (m)	0.2121 (m)
Undershoot (x)	0.2443 (m)	0.4264 (m)
Rise time (x)	76 (time step)	86 (time step)
Settling time (x)	–	772 (time step)
Steady error (x)	0.1351 (m)	0 (m)

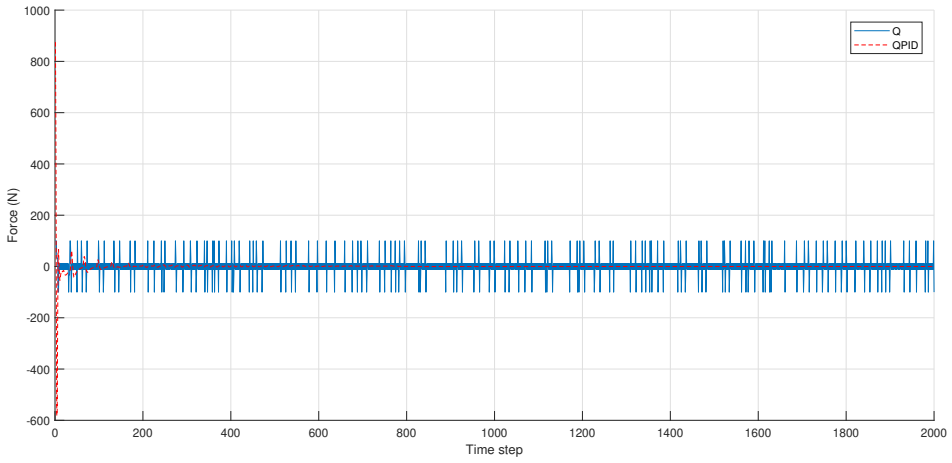

Figure 3.5: Forces generated by Q controller and QPID controller (initial position $\theta(t_0) = \frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$)

controller. Fig. 3.7 shows the response curves of inverted pendulum system controlled by QPID controller and PID controller and the control signals generated by QPID controller and PID controller are shown in Fig. 3.8. The key characteristics of the response curves are illustrated in Table 3.8. In the transient response test, the controllers are expected to balance the inverted pendulum system from initial position back to equilibrium point $\theta = 0, \dot{\theta} = 0, x = 0, \dot{x} = 0$ as quick as possible while keeping the overshoot in the transient response to the minimum. It could be noticed that QPID controller and PID controller shows advantage in the transient response from different aspects. Regarding to the position of the pendulum, QPID uses less time steps to drive the pendulum back to the reference point but has larger overshoot compared with PID controller, while regarding to the position of the cart, the performance of two controllers are totally reverse. The preference of choosing controller could vary based on the specific requirements in different tasks.

For better illustration purpose, Fig. 3.9 demonstrates the adaption of the gains of QPID controller during whole controlling process.

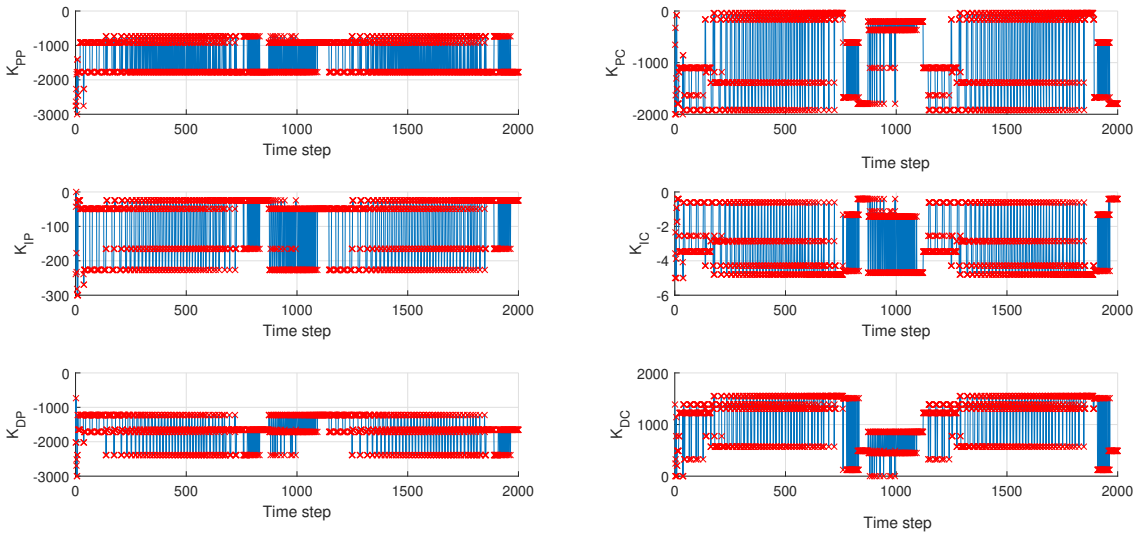

Figure 3.6: Adaption of the gains of QPID controllers (initial position $\theta(t_0) = \frac{10}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0, \dot{x}(t_0) = 0$)

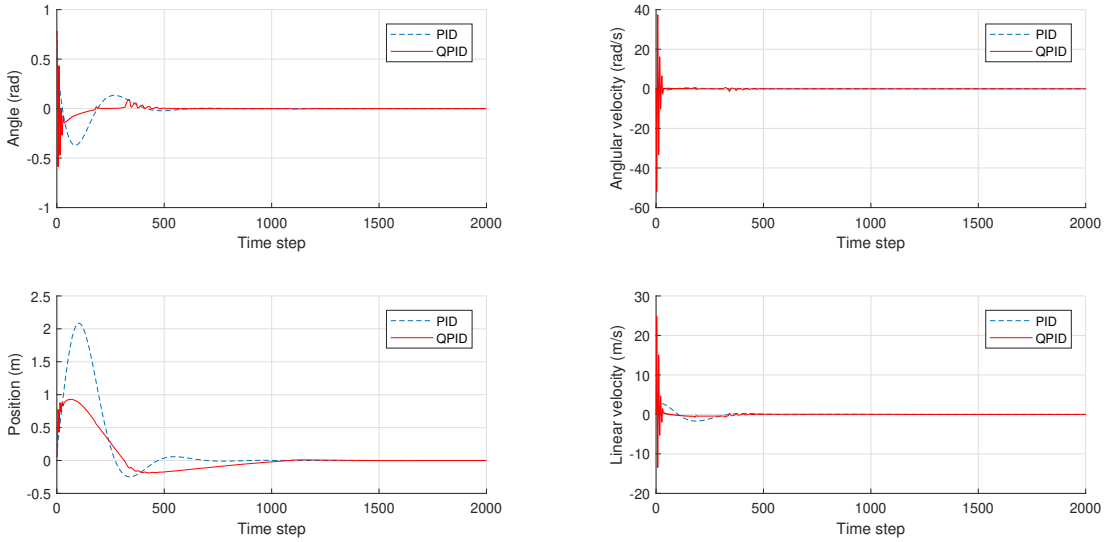

Figure 3.7: Response curves of PID controller and QPID controller (initial position $\theta(t_0) = \frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0.05m, \dot{x}(t_0) = 0$)

3.3 Conclusion

In this research, an adaptive PID controller based on Q-learning algorithm was proposed. In this approach, a set of linear PID controllers are set in lower level for system balance while a group of Q-tables are associated with the PID controllers in higher level which generate the proper combination of gain values optimized through Q-learning algorithm. The adaptive learning rate scheme Delta-Bar-Delta is applied to accelerate the training procedure. Both the QPID controller and the comparisons are tested on the 4th order inverted pendulum system in simulation environment.

Table 3.8: Key characteristics of response curves (PID controller and QPID controller)

Characteristics	PID	QPID
Overshoot (θ)	0.3527 (rad)	0.5866 (rad)
Undershoot (θ)	0.1316 (rad)	0.4306 (rad)
Rise time (θ)	44 (time step)	4 (time step)
Settling time (θ)	320 (time step)	64 (time step)
Steady error (θ)	0 (rad)	0 (rad)
Overshoot (x)	2.075 (m)	0.9279 (m)
Undershoot (x)	0.2443 (m)	0.1807 (m)
Rise time (x)	76 (time step)	13 (time step)
Settling time (x)	426 (time step)	772 (time step)
Steady error (x)	0 (m)	0 (m)

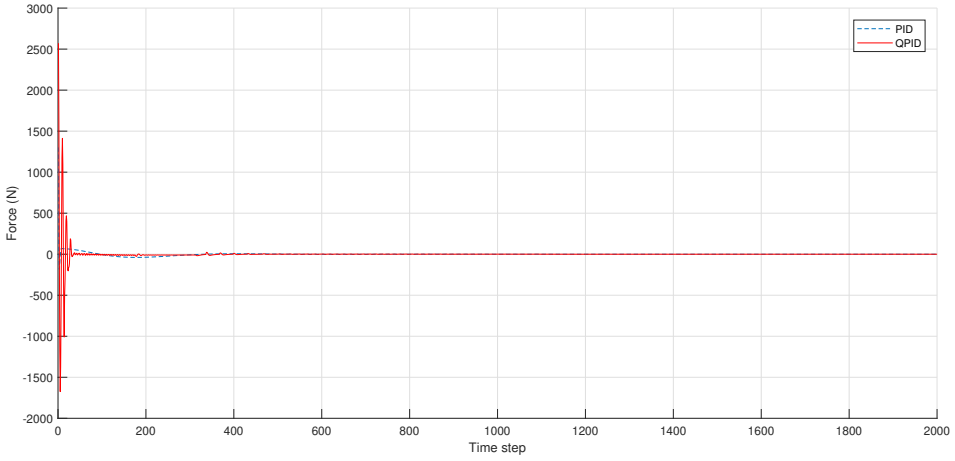

Figure 3.8: Forces generated by PID controller and QPID controller (initial position $\theta(t_0) = \frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0.05m, \dot{x}(t_0) = 0$)

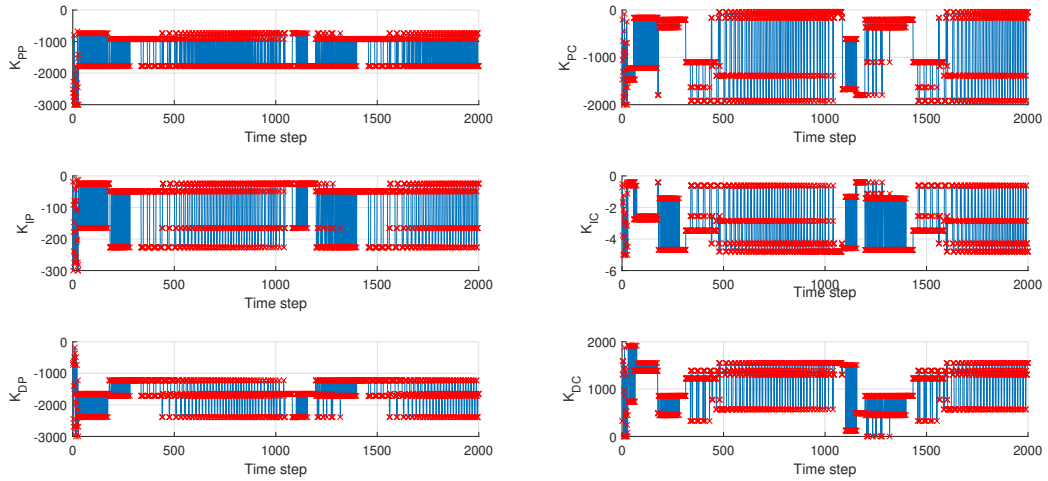

Figure 3.9: Adaption of the gains of QPID controllers (initial position $\theta(t_0) = \frac{45}{180}\pi, \dot{\theta}(t_0) = 0, x(t_0) = 0.05m, \dot{x}(t_0) = 0$)

According to the experimental results, QPID controller designed in this research was trained in four fixed positions and was able to balance the system starting from various different initial positions, which shows the advantage of QPID controller in adapting changes made to systems. Two other types of controllers are proposed for comparison purpose, which are conventional PID controller and the Q controller. The transient performances of the inverted pendulum system controlled by different categories of controllers from fixed initial position point are compared and analyzed, where the QPID controller shows advantage in balancing the system with quicker response speed and less steady-state error.

Chapter 4

Adaptive Neuro-fuzzy PID Controller based on Twin Delayed Deep Deterministic Policy Gradient Algorithm

Regarding to the development of RL based FLS, the increased number of fuzzy rules and membership functions always lead to the inevitable high-dimensional action space in RL algorithm, which brings converging difficulty in RL training process and unsatisfactory performance of FLS after optimization. As to address the issue mentioned above, this chapter presents an adaptive type-1 neuro-fuzzy PID controller based on TD3 algorithm for nonlinear systems. In this approach, the observation of the environment is embedded with information of a multiple input single output FLS and have a specially designed fuzzy PID controller in NN formation acting as the actor approximator in the TD3 algorithm, which achieves automatic tuning of gains in fuzzy PID controller. From the control perspective, the controller combines the merits of both FLS and PID controller and utilizes RL algorithm for optimizing parameters. From the RL point of view, infusing the structure of fuzzy PID controller in the actor network helps reduce the learning difficulty in the training process. The proposed method was tested on the inverted pendulum system in simulation environment with comparison of a linear PID controller, which demonstrates the robustness and generalization of the proposed approach. The remainder of the chapter is organised as follows. Section 4.1 presents the details of the proposed approach; Section 4.2 shows the simulation results of the proposed methodology and the conclusions are drawn in Section 4.3.

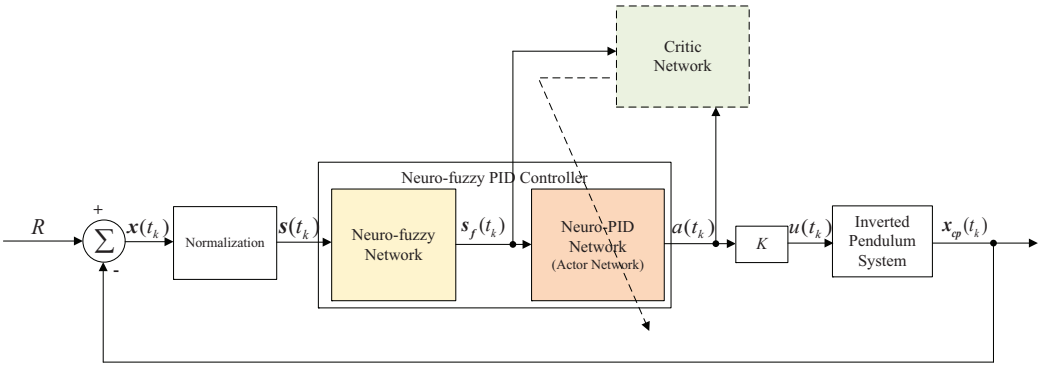

Figure 4.1: Structure of control system with type-1 neuro-fuzzy PID controller and inverted pendulum system

4.1 Adaptive neuro-fuzzy PID controller

In this section, the details of the adaptive neuro-fuzzy PID network based on TD3 algorithm are provided. Fig. 4.1 presents the structure of whole system with type-1 neuro-fuzzy PID controller and inverted pendulum system. The state of the environment $x(t_k)$ is normalized before fed into neuro-fuzzy PID controller. The neuro-fuzzy PID network is consisted of two sections which are neuro-fuzzy network marked as yellow block on the left-hand side and neuro-PID network marked as orange block on the right-hand side. When $\boldsymbol{s}(t_k)$ is input to neuro-fuzzy network, the information of FLS is associated with the normalized state, which generates fuzzy state $\boldsymbol{s_f}(t_k)$ as the output of the neuro-fuzzy network. The neuro-PID network takes the fuzzy state $\boldsymbol{s_f}(t_k)$ as input and generates action $a(t_k) \in [0, 1]$, which is then multiplied with scale factor K to generate control signal $u(t_k)$ for balancing the system. To be noticed, only the neuro-PID network works as the actor network in TD3 while the parameters of neuro-fuzzy network keep fixed during the training process. Therefore, state $s_f(t_k)$ instead of $s(t_k)$ and action $a(t_k)$ are taken as the input of critic network, which approximates the value function of state-action pairs and affects the action generated by actor network. The detailed structure of these two networks in Neuro-fuzzy PID controller shown in Fig. 4.1 are depicted in Fig. 4.2 and will be further discussed in Section 4.1.1 while the training procedure is described in Section 4.1.2.

4.1.1 Structure of Neuro-fuzzy PID network

The neuro-fuzzy PID network presents the combination of FLS with PID controller in the NN formation. The neuro-fuzzy PID network proposed in this section is consisted of two main section namely neuro-fuzzy network and neuro-PID network. In the neuro-fuzzy network, the parameters are preset and remain fixed during the update process. The operations of fuzzy reasoning process are achieved by passing input state through multiple partially connected layers and output state

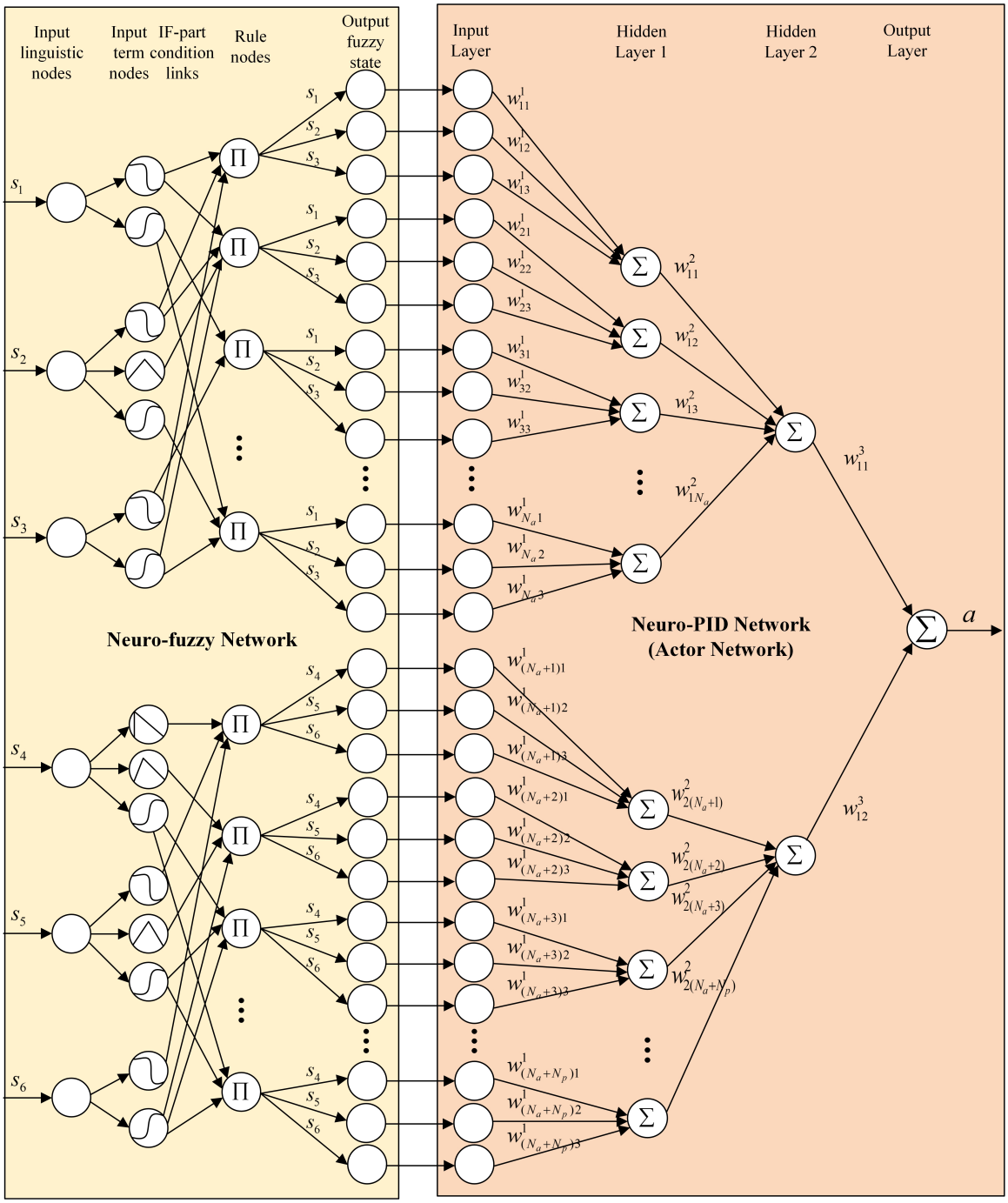

Figure 4.2: Structure of neuro-fuzzy PID network

vector embedding with fuzzy information. The neuro-PID network takes the fuzzy state $\boldsymbol{s_f}$ as input and generate action a, which is the weighted summation of PID outputs from all fuzzy rules. The details of each layer are introduced as follows.

Neuro-fuzzy network

In the input layer, there are six variables that are chosen to present the state of the inverted pendulum system, which can be expressed as $\boldsymbol{s}(t_k) = \boldsymbol{x}(t_k)/\boldsymbol{X}$, where $\boldsymbol{x}(t_k) = [e_\theta(t_k), de_\theta(t_k), se_\theta(t_k), e_x(t_k), de_x(t_k), se_x(t_k)]$. $e_\theta(t_k) = R_\theta - x_1(t_k)$ is the error of the angle, which is the difference between the position of pole and the reference position, $R_\theta = 0$ in this case; $de_\theta(t_k) = e_\theta(t_k) - e_\theta(t_{k-1})$ is the change of angle error in the k^{th} time step; $se_\theta(t_k) = \sum_{i=0}^{k} e_\theta(t_i)$ is the accumulated angle error until time step t_k. Similarly, $e_x(t_k) = R_x - x_3(t_k)$ is the position error, which is the difference between the position cart and the reference point, $R_x = 0$ in this case; $de_x(t_k) = e_x(t_k) - e_x(t_{k-1})$ is the change of position error in the k^{th} time step; $se_x(t_k) = \sum_{i=0}^{k} e_x(t_i)$ is the accumulated position error until time step t_k; $\boldsymbol{X} = [X_1, X_2, \cdots, X_6]$ is a constant normalization vector which guarantees all state variables are within $[-1, 1]$. In the rest of the chapter, the variable of time t_k will be omitted for the simplicity of expression. As depicted in Fig. 4.2, the inputs of neuro-fuzzy network are separated as two sets of variables, which are variables related to the angle of the pole $\boldsymbol{s_a} = [s_1, s_2, s_3]$ and ones related to the position of the cart $\boldsymbol{s_p} = [s_4, s_5, s_6]$. In the input linguistic layer, the input term nodes are partially connected with the nodes from previous input linguistic layer, The number of input terms connected to a single input linguistic is determined by the number of fuzzy terms certain linguistic variable has in the FLS. s_1 has two fuzzy terms SMALL and LARGE; s_2 has three fuzzy terms SMALL, MEDIUM and LARGE; s_3 has two fuzzy terms SMALL and LARGE. The antecedent membership functions of $\boldsymbol{s}_a$ are shown in Fig. 4.3, where Z-shaped and S-shaped membership functions are used for the fuzzy terms of state s_1; Z-shaped, S-shaped and triangular membership functions are chosen for fuzzy terms of state s_2; Z-shaped and S-shaped membership functions are chosen for the fuzzy terms of state s_3. Similarly, the fuzzification of state vector related to the cart $\boldsymbol{s_p}$ are as follows, s_4 has three fuzzy terms SMALL, MEDIUM and LARGE; s_5 has two fuzzy terms SMALL and LARGE; s_6 has three fuzzy terms SMALL, MEDIUM and LARGE. The membership functions for $\boldsymbol{s_p}$ are shown in Fig. 4.4, where triangular and S-shaped membership functions are used for the fuzzy terms of state s_4; Z-shaped and S-shaped membership functions are chosen for fuzzy terms of state s_5; Z-shaped, triangular and S-shaped membership functions are chosen for the fuzzy terms of state s_6.

In the following layer, the number of the rule nodes indicates the number of fuzzy rules in the FLS while the connections between rule nodes and input term nodes are referred as the condition links in the IF part. For the case shown in Fig. 4.2, the

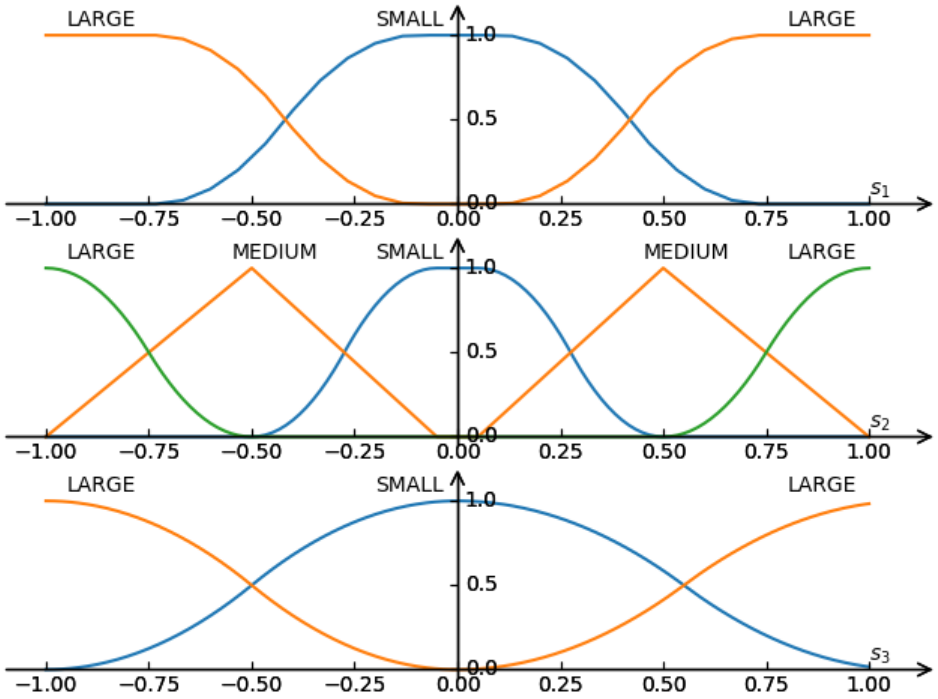

Figure 4.3: Membership functions of $\boldsymbol{s}_a$

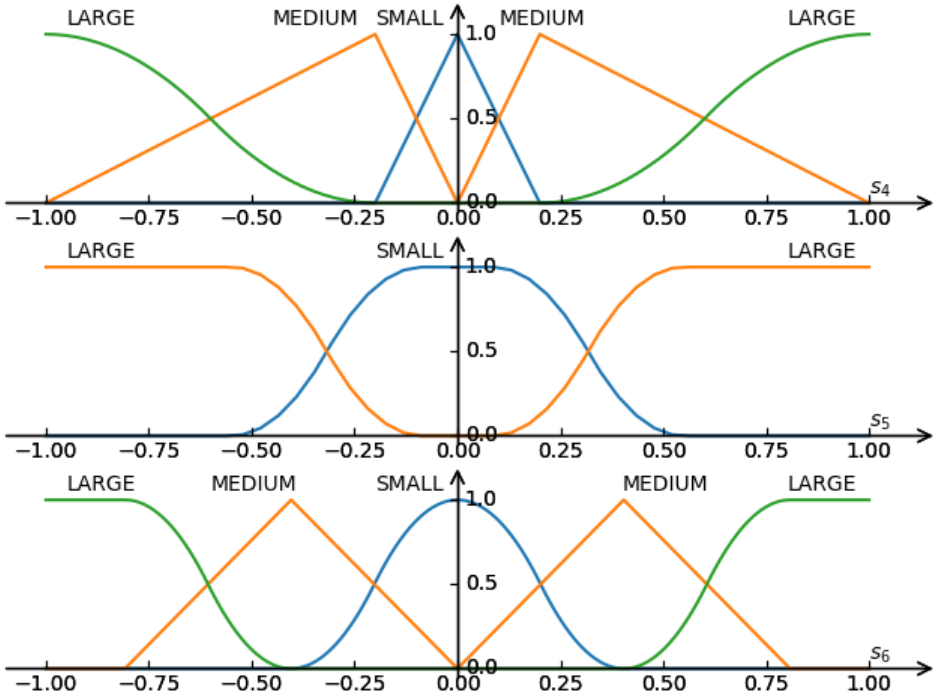

Figure 4.4: Membership functions of $\boldsymbol{s}_p$

number of fuzzy rules related to $\boldsymbol{s_a}$ is $N_a = 2 \times 3 \times 2 = 12$ and the number of rules related to $\boldsymbol{s_p}$ is $N_p = 3 \times 2 \times 3 = 18$. Therefore, there are $N_a + N_p = 12 + 18 = 30$ nodes in the Rule nodes layer. Each rule node generates the normalized weight of the corresponding rule. Weights related to $\boldsymbol{s_a}$ are expressed in the form

$$w_{1i}(\boldsymbol{s_a}) = \frac{\prod_{k=1}^{3} \mu_{M_k^i}(\boldsymbol{s_a})}{\sum_{n=1}^{N_a}(\prod_{k=1}^{3} \mu_{M_k^n}(\boldsymbol{s_a}))}, \tag{4.1}$$

where $i = 1, 2, \cdots, N_a$, N_a is the number of fuzzy rules related to $\boldsymbol{s_a}$, $N_a = 12$ in this case.

The normalized weights for rules related $\boldsymbol{s_p}$ are in the form

$$w_{2j}(\boldsymbol{s_p}) = \frac{\prod_{k=1}^{3} \mu_{M_k^j}(\boldsymbol{s_p})}{\sum_{n=N_a+1}^{N_a+N_p}(\prod_{k=1}^{3} \mu_{M_k^n}(\boldsymbol{s_p}))}, \tag{4.2}$$

where $j = N_a + 1, N_a + 2, \cdots, N_a + N_p$, N_p is the number of fuzzy rules related to $\boldsymbol{s_p}$, $N_p = 18$ in this case. The fuzzy state is in the form $\boldsymbol{s_f} = [\boldsymbol{s_{fa}}, \boldsymbol{s_{fp}}]$, where $\boldsymbol{s_{fa}} = [w_{11}s_1, w_{11}s_2, w_{11}s_3, \cdots, w_{1N_a}s_1, w_{1N_a}s_2, w_{1N_a}s_3]$ and $\boldsymbol{s_{fp}} = [w_{2(N_a+1)}s_4$, $w_{2(N_a+1)}s_5, w_{2(N_a+1)}s_6, ..., w_{2(N_a+N_p)}s_4, w_{2(N_a+N_p)}s_5, w_{2(N_a+N_p)}s_6]$. The fuzzy output $\boldsymbol{s_f}$ is obtained by multiplying the output value of each rule node with the corresponding state. The number of nodes in the output layer is $3(N_a+N_p) = 3\times 30 = 90$ in this case.

Neuro-PID Network

Neuro-PID network takes the output of neuro-fuzzy network, $\boldsymbol{s_f}$, as the input variables, which embeds information of states with the fuzzy information of FLS. Inside the neuro-PID network, each fuzzy rule is associated with an individual linear PID controllers and the final output is the weighted summary of PID controllers in all fuzzy rules. Referring to Fig. 4.2, the outputs of Hidden Layer 1 are expressed as

$$g_i^1 = \begin{cases} \sum_{j=1}^{3} w_{ij}^1 w_{1i}(\boldsymbol{s_a}) s_j, i = 1, 2, \cdots, N_a \\ \sum_{j=1}^{3} w_{ij}^1 w_{2i}(\boldsymbol{s_p}) s_{j+3}, i = N_a+1, N_a+2, ..., N_a+N_p. \end{cases} \tag{4.3}$$

The outputs of the Hidden Layer 2 are

$$g_i^2 = \begin{cases} \sum_{j=1}^{N_a} w_{1j}^2 g_j^1, i = 1 \\ \sum_{l=N_a+1}^{N_a+N_p} w_{2l}^2 g_l^1, i = 2. \end{cases} \tag{4.4}$$

The final output of the neuro-PID network is

$$a = \text{clip}(\sum_{i=1}^{2} w_{1i}^3 g_i^2, -0.8, 0.8), \tag{4.5}$$

where $\text{clip}(a, a_l, a_u)$ is defined as

$$\text{clip}(a, a_l, a_u) = \begin{cases} a_l, \text{if } a \leq a_l \\ a, \text{if } a_l < a < a_u \\ a_u, \text{if } a \geq a_u. \end{cases} \tag{4.6}$$

The value of 0.8 is chosen based on the experiment test results as to stabilize the learning process. The force that applied to the inverted pendulum system is

$$u = Ka, \tag{4.7}$$

where $K = 1500$. According to Eq. 4.3, Eq. 4.4 and Eq. 4.5, the force then can be rewritten as

$$\begin{aligned}
u =& K(w_{11}^3 g_1^2 + w_{12}^3 g_2^2) \\
=& K(w_{11}^3 \sum_{j=1}^{N_a} w_{1j}^2 g_j^1 + w_{12}^3 \sum_{l=N_a+1}^{N_a+N_p} w_{2l}^2 g_l^1) \\
=& K(w_{11}^3 \sum_{j=1}^{N_a} w_{1j}^2 \sum_{k=1}^{3} w_{jk}^1 w_{1j}(\boldsymbol{s_a}) s_k \\
&+ w_{12}^3 \sum_{l=N_a+1}^{N_a+N_p} w_{2l}^2 \sum_{q=1}^{3} w_{lq}^1 w_{2l}(\boldsymbol{s_p}) s_{q+3}) \\
=& \sum_{j=1}^{N_a} w_{1j}(\boldsymbol{s_a})(K w_{11}^3 w_{1j}^2 w_{j1}^1 s_1 + K w_{11}^3 w_{1j}^2 w_{j2}^1 s_2 \\
&+ K w_{11}^3 w_{1j}^2 w_{j3}^1 s_3) + \sum_{l=N_a+1}^{N_a+N_p} w_{2l}(\boldsymbol{s_p})(K w_{12}^3 w_{2l}^2 w_{l1}^1 s_4 \\
&+ K w_{12}^3 w_{2l}^2 w_{l2}^1 s_5 + K w_{12}^3 w_{2l}^2 w_{l3}^1 s_5)) \\
=& \sum_{j=1}^{N_a} w_{1j}(\boldsymbol{s_a})(K_{Pj} s_1 + K_{Dj} s_2 + K_{Ij} s_3) \\
&+ \sum_{l=N_a+1}^{N_a+N_p} w_{2l}(\boldsymbol{s_p})(K_{Pl} s_4 + K_{Dl} s_5 + K_{Il} s_5))
\end{aligned} \tag{4.8}$$

where $K_{Pj} = K w_{11}^3 w_{1j}^2 w_{j1}^1$, $K_{Dj} = K w_{11}^3 w_{1j}^2 w_{j2}^1$, $K_{Ij} = K w_{11}^3 w_{1j}^2 w_{j3}^1$, $K_{Pl} = K w_{12}^3 w_{2l}^2 w_{l1}^1$, $K_{Dl} = K w_{12}^3 w_{2l}^2 w_{l2}^1$, $K_{Il} = K w_{12}^3 w_{2l}^2 w_{l3}^1$. Therefore, Eq. 4.8 can be presented as

$$u = \sum_{j=1}^{N_a} w_{1j}(\boldsymbol{s_a}) u_{aj} + \sum_{w=1}^{N_p} w_{2(w+N_a)}(\boldsymbol{s_p}) u_{pw}, \tag{4.9}$$

where

$$u_{aj} = K_{Pj} s_1 + K_{Dj} s_2 + K_{Ij} s_3, \tag{4.10}$$

$$u_{pw} = K_{P(w+N_a)} s_4 + K_{D(w+N_a)} s_5 + K_{I(w+N_a)} s_6. \tag{4.11}$$

Accordingly, the rule base of $\boldsymbol{s_a}$ is shown in Table 4.1 and Table 4.2 while the rule

Table 4.1: Rule-base table of s_2 and s_3 if s_1 is SMALL

s_2 \ s_3	SMALL	LARGE
SMALL	u_{a1}	u_{a2}
MEDIUM	u_{a9}	u_{a10}
LARGE	u_{a3}	u_{a4}

Table 4.2: Rule-base table of s_2 and s_3 if s_1 LARGE

s_2 \ s_3	SMALL	LARGE
SMALL	u_{a5}	u_{a6}
MEDIUM	u_{a11}	u_{a12}
LARGE	u_{a7}	u_{a8}

Table 4.3: Rule-base table of s_4 and s_6 if s_5 is SMALL

s_4 \ s_6	SMALL	MEDIUM	LARGE
SMALL	u_{p1}	u_{p13}	u_{p2}
MEDIUM	u_{p9}	u_{p17}	u_{p10}
LARGE	u_{p5}	u_{p15}	u_{p6}

base of $\boldsymbol{s_p}$ is shown in Table 4.3 and Table 4.4. The consequent membership function of each rule is chosen as singleton membership function. The outputs shown in the rule bases from Table 4.1 to Table 4.4 corresponding to the variables in Eq. 4.10 and Eq. 4.11, which are $u_{aj}, j = 1, 2, ..., N_a$ and $u_{pw}, w = 1, 2, ..., N_p$, respectively. These variables stand for the outputs u_i in the concept of fuzzy PID controller expressed in Eq. 2.8, which are not shown in Fig. 4.2.

Therefore, the neuro-fuzzy network and neuro-PID network combine together to formulate adaptive neuro-fuzzy PID controller. In the RL training procedure, the weights of neuro-fuzzy network keep fixed while the weights of the neuro-PID network are updated.

Remark 4.1 *The reason to keep the weights of neuro-fuzzy network fixed while updating the weights of neuro-PID network in the training process is for the consideration of updating parameters in the actor network. As presented in TD3 algorithm, updating actor network follows the gradient* $\nabla = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q_1 \nabla_{s_f} a$. *If the weights in both neuro-fuzzy network and neuro-PID network are adjustable, the gradient for updating actor network will be* $\nabla' = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q_1 \nabla_{s_f} a \nabla_s s_f$. *However, as shown in Fig. 4.2, the calculation of gradient* $\nabla_s s_f$ *related to neuro-fuzzy network can be challenging considering the discontinuity existing among the fuzzy membership functions Therefore, only the weights of neuro-PID networks are updated while the weights in neuro-fuzzy network remain fixed.*

Table 4.4: Rule-base table of s_4 and s_6 if s_5 is LARGE

s_4 \ s_6	SMALL	MEDIUM	LARGE
SMALL	u_{p3}	u_{p14}	u_{p4}
MEDIUM	u_{p12}	u_{p18}	u_{p12}
LARGE	u_{p7}	u_{p16}	u_{p8}

The reward function in the training process is designed as follows.

$$r = \begin{cases} 0.4r_1, \text{if } |s_4| \geq 0.3 \\ 0.4 + 0.4r_2, \text{if } 0.05 \leq |s_4| < 0.3 \\ 0.8 + 0.2r_3, \text{if } |s_4| < 0.05, \end{cases} \tag{4.12}$$

where

$$r_1 = 0.4 \times (1 - \frac{|s_4| - 0.3}{0.7}) + 0.5\ r_{s_5} + 0.1 \times (1 - |s_1|), \tag{4.13}$$

r_{s_5} is a reward regarding to the error and the change of error of the position, which is expressed as

$$r_{s_5} = \begin{cases} 1, \text{if } s_4 s_5 < 0 \\ 1 - \frac{|s_5|}{0.5}, \text{otherwise;} \end{cases} \tag{4.14}$$

$$\begin{aligned} r_2 =& 0.4 \times (1 - \frac{|s_4|}{0.2}) + 0.2 \times (1 - \frac{|s_5|}{0.3}) + 0.1 \times (1 - \frac{|s_2|}{0.3}) \\ &+ 0.2 \times (1 - \frac{\sum_{i=0}^{t} |s_4(i)|}{100}) + 0.1 \times (1 - \frac{\sum_{i=0}^{t} |s_1(i)|}{200}); \end{aligned} \tag{4.15}$$

$$\begin{aligned} r_3 =& 0.5 \times (1 - \frac{|s_4|}{0.05}) + 0.3 \times (1 - \frac{\sum_{i=0}^{t} |s_4(i)|}{50}) \\ &+ 0.2 \times (1 - \frac{\sum_{i=0}^{t} |s_1(i)|}{100}). \end{aligned} \tag{4.16}$$

The reward has three levels which are based on the absolute normalized error of the cart position $|s_4|$. When $|s_4| \geq 0.3$, the cart is considered far away from the origin which is undesirable. The reward in this situation is decided by r_1 shown in Eq. 4.13. r_1 has three terms with the maximum reward restricted to 0.4. The first term is inversely proportional to the absolute error of the cart position, the closer the cart is to the origin, the higher reward can be achieved. The second term is affected by the relationship between the error and the change of error for the cart position, of which details are defined in Eq. 4.14. When $|s_4|$ is large, the primary goal is set to push the cart back to origin, in which case $s_4 s_5 < 0$, where r_{s_5} is set as 1. Otherwise, it indicates that the error is changing in the undesired direction, where the larger the absolute change of error of the cart position is, the smaller reward r_{s_5} has. The third term is related to the normalized absolute error of the pole, the closer the pole

to the up straight position, the higher reward it has. When $0.05 \leq |s_4| < 0.3$, it is considered the inverted pendulum system is approaching the balanced position, which receives 0.4 as a base value while having an extra reward r_2 shown in Eq. 4.15, in this level, reward r changes between 0.4 and 0.8. Reward r_2 has five terms. The first term indicates the cart position, where the smaller absolute error has higher reward. The second term indicates the change error of the cart position, considering the inverted pendulum system is approaching the balanced point, the change of error is expected to be small. Similarly, the third term expects the change error of the pole to be small value as well. The fourth and fifth term are related to the accumulated absolute error of the cart and pole position, which aims to reduce the fluctuations of the cart and the pole. When $|s_4| < 0.05$, it is considered that the system is getting to a relative stable situation, where reward has 0.8 base value with an extra value r_3. In this level, the reward changes between 0.8 and 1.0. r_3 has three terms, which have similar meanings described in r_2. It expects the cart continuously going back to origin while minimizing the fluctuations of the cart and the pole during the whole process.

4.1.2 Training procedure

The training procedure of the TD3 algorithm with neuro-fuzzy PID controller as actor approximator is shown in Algorithm 4. In the beginning of the training procedure, critic networks Q_1 and Q_2 and the corresponding target networks Q'_1 Q'_2 are initialized. The same applies to the actor network, neuro-fuzzy PID network π_f, and corresponding target actor network $\pi_f{}'$. In each training episode, the state s is fed into neuro-fuzzy network which outputs fuzzy state s_f. Based on the fuzzy state s_f, action a is generated following the deterministic policy π_f, where Gaussian noise $\mathcal{N}$ is added to encourage the exploration. After taking action a, new state and reward are observed. This new transition tuple (s, a, r, s') is stored in replay buffer R. As to update parameters in current time step, a mini-batch of experiences with K transition tuples are sampled and embedded with fuzzy information through neuro-fuzzy network. The corresponding actions of the mini-batch are generated based on the target actor network $\pi_f{}'$ with clipped Gaussian noise. The target value used for updating critic networks is estimated using the reward value summarized with the discounted minimum value between two target Q-values ${Q_1}'$ and ${Q_1}'$ for each sample in the mini-batch. The critic networks are updated following the TD-error calculated based on the previous target value. However, the update of the actor network and corresponding target network will only be implemented every d steps as to stabilize the training process. The update repeats until the end of the episode or being terminated when state vector is out of the preset range values.

Algorithm 4: TD3 Algorithm with Neuro-fuzzy PID as Actor

Initialize critic networks $Q_1(s, a|\theta_1)$ and $Q_2(s, a|\theta_2)$, neuro-fuzzy PID actor network $\pi_f(s|\phi_f)$;
Initialize target critic networks Q'_1, Q'_2 with $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$ and target actor network π'_f with $\phi'_f \leftarrow \phi_f$;
Initialize replay buffer $\mathcal{R}$;
For every episode:
Initialize state s;
repeat
    Observe fuzzy state s_f;
    Select action $a \sim \pi_f(s_f|\phi_f) + \mathcal{N}(0, \sigma)$;
    Observe next state s' and reward r;
    Store transition tuple (s, a, r, s') in $\mathcal{R}$;
    Sample mini-batch of K transitions;
    Get fuzzy states of mini-batch s_{fi} and s'_{fi} $(i = 1, 2, ..., K)$;
    Get $a'_i \sim \pi'_f(s'_{fi}|\phi') + \epsilon$,where $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$
    Set $y_i = r_i + \gamma \min\{Q'_{1i}(s'_{fi}, a'_i|\theta'_1), Q'_{2i}(s'_{fi}, a'_i|\theta'_2)\}$
    Update critic network Q_m by minimizing loss
    $L_m = \frac{1}{K}\sum_{i=1}^{K}(y_i - Q_m(s_{fi}, a_i|\theta_m))^2$, $m = 1, 2$;
    Every d steps:
        Update actor network π_f following gradient
        $\nabla_{\phi_f} = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q_1(s, a|\theta_1)|_{s=s_{fi}, a=\pi_f(s_{fi})} \nabla_{\phi_f} \pi_f(s_{fi}|\phi_f)$
        Update target networks:
        $\theta'_m \leftarrow \tau\theta_m + (1 - \tau)\theta'_m$
        $\phi'_f \leftarrow \tau\phi_f + (1 - \tau)\phi'_f$
    $s \leftarrow s'$
until *s reaches terminal state s_T*;

4.2 Simulation results

Overview what will be test in this section. Several tests have been implemented to test the performance of the neuro-fuzzy PID controller after training. Besides, as to test the generalization of the proposed controller, a random initial position test from a large operating scope ([−60deg, +60deg]) will be implemented. Furthermore, the trained controller will be tested with different mass of the pole or the cart as well as a low-frequency external disturbance in the beginning period of controlling process as to demonstrate the robustness of the proposed method. The transient response curves are expected to have less oscillations, overshoot, undershoot and steady state error. The obove tests are compared with a linear PID controller training with TD3 algorithm. The details of comparison controller are provided in Section 5.2.1. These requirements can be achieved by adopting neuro-fuzzy PID controller as actor NN and designing more comprehensive reward scheme.

4.2.1 Comparison controller

As for comparison, a linear PID controller tuned with TD3 algorithm is designed. Similar to the proposed approach, the actor network is replaced with a neuro-PID controller of which structure presented in Fig. 5.3. The output of the NN is

$$a = \sum_{i=1}^{6} s_i w_i^1. \tag{4.17}$$

The force applied to the inverted pendulum system is

$$u = Ka, \tag{4.18}$$

K is the scale factor, $K = 1500$.

Remark 4.2 *The linear PID controller can be regarded as a special case of neuro-fuzzy PID controller where the number of fuzzy rule is reduced to 1.*

4.2.2 Parameter settings

During the training process, both the neuro-fuzzy PID controller and linear PID controller are trained with the same set of parameters except for the learning rate of critic and actor network, which are chosen differently to achieve the best learning performance of the controller itself. The settings of parameters are shown in Table 4.5. For the neuro-fuzzy PID controller, the learning rates of the actor and critic network are 2.2×10^{-6} and 6.5×10^{-5}, respectively, while for the linear PID controller, the learning rates of the actor and critic network are 8×10^{-6} and 5×10^{-5},

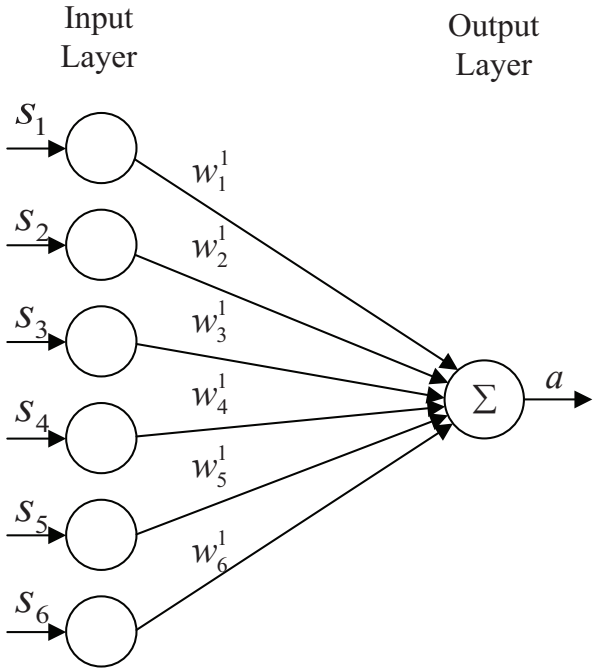

Figure 4.5: Structure of neuro-PID controller

respectively. The critic networks for both neuro-fuzzy PID controller and linear PID controller have 150 neurons in the first hidden layer, 100 neurons in the second hidden layer and single neuron in the output layer. Besides, the parameters of inverted pendulum system are given in Table 4.6 [143].

Table 4.5: Parameters Setting in TD3 Algorithm

Variable	Value
max episodes	3000
steps per episode	1000
d	3
K	64
buffer size	10^6
τ	0.001
γ	0.99

Table 4.6: Parameters Setting for the Inverted Pendulum System

Variable	Value
M	1.3282kg
m	0.22kg
l	0.304m
F_0	22.915N
F_1	0.007056N

4.2.3 Training results

The whole training process includes 3000 episodes. In each episode, the episode will be terminated when it reaches the maximum time step or the system exceeds the limited position range. The learning curves of neuro-fuzzy PID controller and linear PID controller are shown in Fig. 4.6, which present the accumulated reward with the increment of the training episode. The drops in the learning curves are associated with the sudden terminations of the episodes, which indicate the failure of tasks. In Fig. 4.6, though the learning curve of linear PID controller shows converging trend earlier than neuro-fuzzy PID controller, neuro-fuzzy PID controller manages to reach the maximum accumulated reward faster with less drops than the linear PID controller in the late training stage, which shows the advantage of neuro-fuzzy PID controller in learning efficiency and stability during the training process.

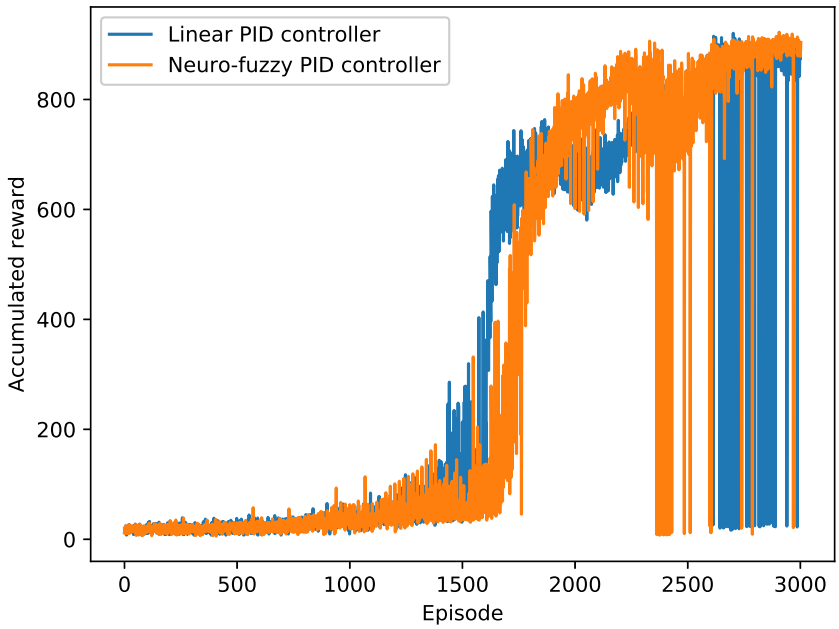

Figure 4.6: Comparison of learning curves

Comparison of controlling performance

The transient response curves of inverted pendulum system under control by neuro-fuzzy PID controller and linear PID controller from initial position $\theta = \frac{60}{180}\pi,\ \dot{\theta} = 0,\ x = 0,\ \dot{x} = 0$ are shown in Fig. 4.7 and the forces two controllers are shown in Fig. 4.8.In the transient response test, the controllers are expected to balance the inverted pendulum system from initial position back to equilibrium point $\theta = 0, \dot{\theta} = 0, x = 0, \dot{x} = 0$ as quick as possible while keeping the overshoot in the transient response to the minimum. Only when both the position of the pendulum and the position of the cart are driven back to the reference point can the system be regarded

as stable.

Comparing the response curves shown in Fig. 4.7, neuro-fuzzy PID controller and linear PID controller spend almost the same time steps to balance the position of the pendulum back to the reference point $\theta = 0$, however, neuro-fuzzy PID controller drives the position of the cart back to the reference point $x = 0$ quicker than the linear PID controller. Thus, the system controlled by neuro-fuzzy PID controller is regarded as stabilized faster than the linear PID controller. Furthermore, comparing the overshoot in the response curves of system, neuro-fuzzy PID controller have smaller overshoot in both response curves relating to the position of the pendulum and the position of the cart. Therefore, in the transient response test, neuro-fuzzy PID controller could balance the system back to equilibrium point with shorter settling time and smaller overshoot, which indicates its advantage compared with linear PID controller.

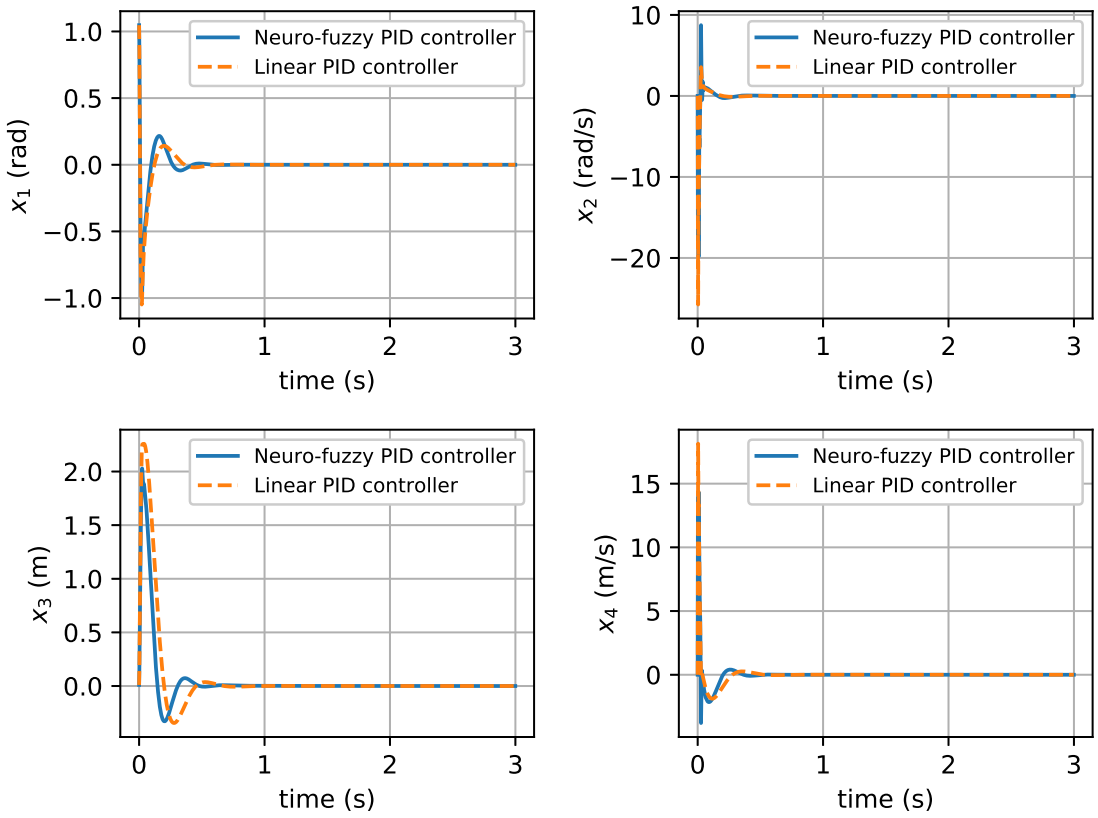

Figure 4.7: Comparison of transient response curves (Initial position: $\theta = \frac{60}{180}\pi,\ \dot{\theta} = 0,\ x = 0,\ \dot{x} = 0$)

Random initial positions test

In the random initial positions test, the angle of the pole is sampled from the range of $[-60\text{deg}, 60\text{deg}]$ with a step size of 15deg while $\dot{\theta} = 0$, $x = 0.01$ and $\dot{x} = 0$. Three categories of phase portraits are chosen to illustrate the response curves of inverted pendulum system, which are shown from Fig. 4.9 to Fig. 4.11. In each

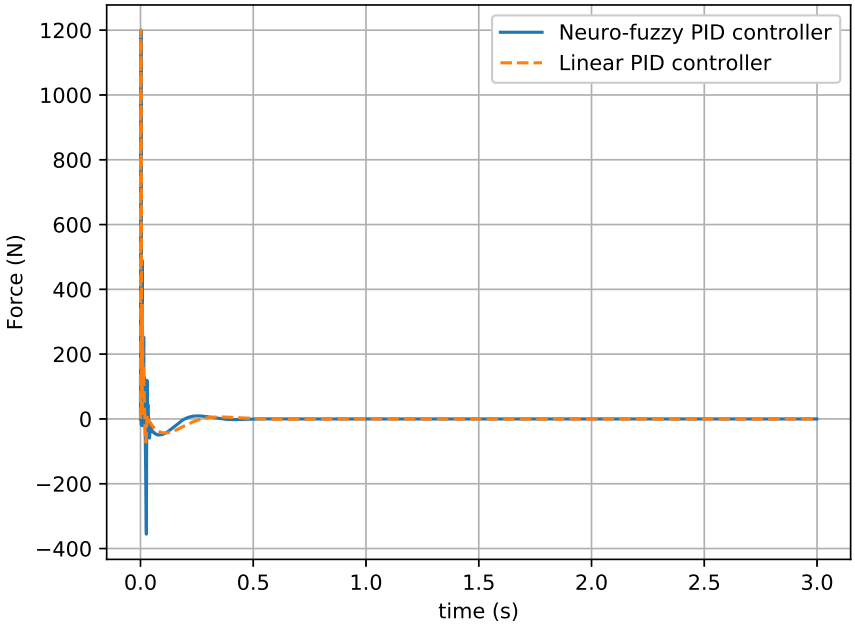

Figure 4.8: Comparison of forces (Initial position: $\theta = \frac{60}{180}\pi,\ \dot{\theta} = 0,\ x = 0,\ \dot{x} = 0$)

phase portrait, the red circles indicate different initial positions and the solid lines present the changing trend in the whole response process. Fig. 4.9 presents how the pendulum position changes with the change of cart position; Fig. 4.10 illustrate how the angular velocity varies with the change of pendulum position; Similarly, Fig. 4.11 shows how the linear velocity varies with the change of cart position. For each initial position, if the corresponding lines in all three phase portraits converge to the origin point, it indicates the system starting from certain initial position could finally stabilize at the position where both the position and the velocity of cart and pendulum are zero, which indicates the system is balanced to the equilibrium point. It could be noticed that all the lines in three phase portraits starting from different red points all lead towards in the direction of reducing the state error and finally converge to the origin point. Therefore, the neuro-fuzzy PID controller is regarded as successfully stabilize the inverted pendulum at equilibrium point from all initial positions.

Robustness test

Two different experiments are designed as to test the robustness of the proposed controller. In the first test, both controllers are tested by changing the mass of the pole or the cart with the fixed initial condition, where $\theta = 60$deg, $x = 0.01$m with the velocity of cart and pole are zero. The mass of the pole changes with $\pm 90\%$ and the mass of the cart changes with $\pm 50\%$. The neuro-fuzzy PID controller is able to successfully balance the system in all situations while neuro-linear PID controller fails when the mass of the pole is reduced greater than 88%. In the

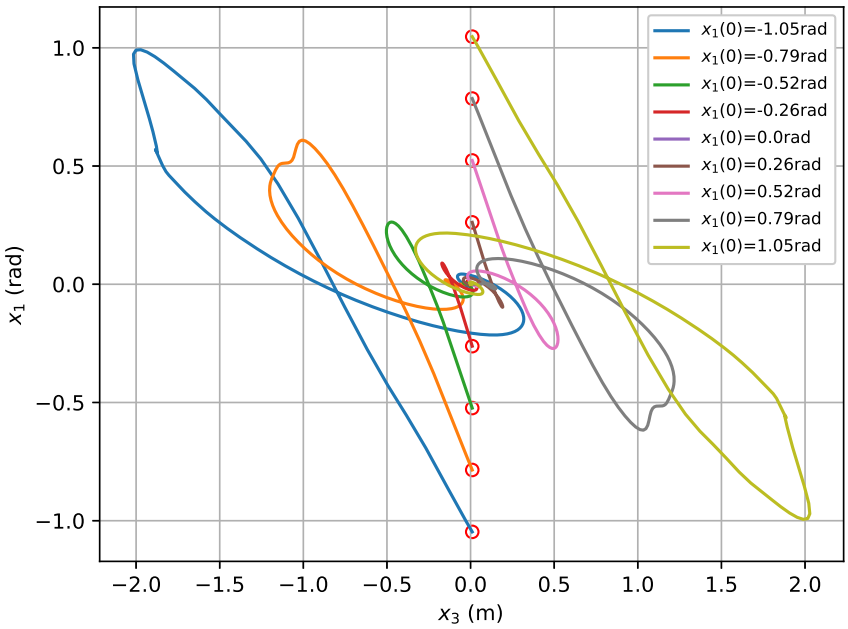

Figure 4.9: Phase portrait x_3 and x_1. The curves in different colours represent the phase flows. The red circles indicate the initial states of the trajectories.

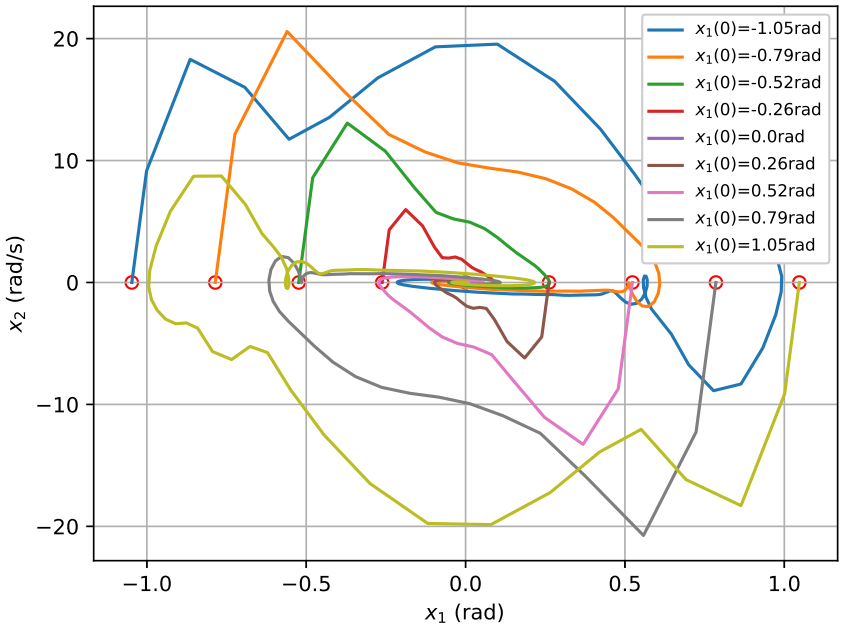

Figure 4.10: Phase portrait of x_1 and x_2 from different initial positions. The curves in different colours represent the phase flows. The red circles indicate the initial states of the trajectories.

second test, a low-frequency external force is added to control force u as to imitate the disturbance in real environment, like bumping into obstacles or the disturbance of the wind. The disturbance force is shown in Fig. 4.12, where the disturbance

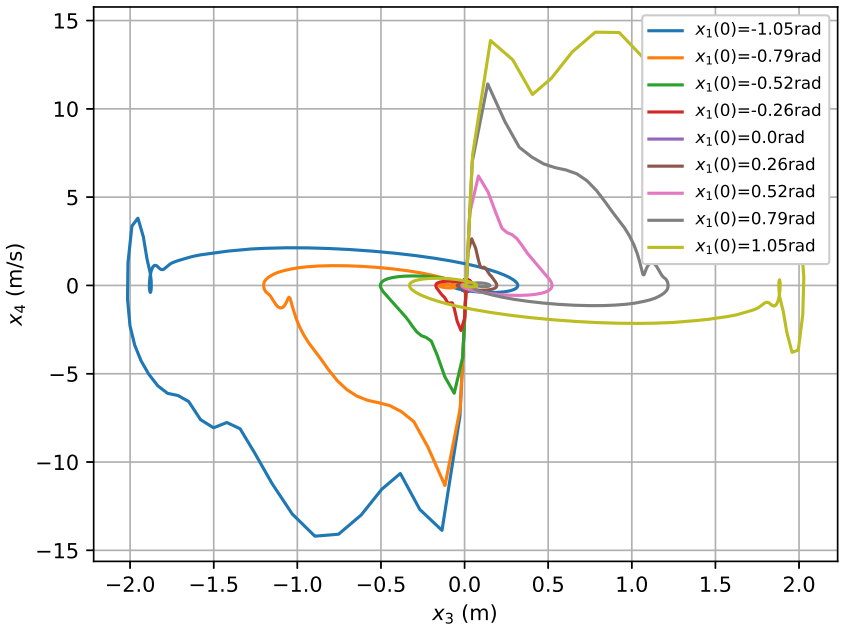

Figure 4.11: Phase portrait of x_3 and x_4 from different initial positions. The curves in different colours represent the phase flows. The red circles indicate the initial states of the trajectories.

is mainly added in the beginning period of controlling process as to increase the difficulty of balancing the inverted pendulum system. When comparing the response curves of both controllers shown in Fig. 4.13, it can be seen that the neuro-fuzzy PID controller is able to balance the system back to origin while linear PID controller fails in the beginning of the control process (the break point is marked with red cross in Fig. 4.13). According to results of robustness tests, the proposed neuro-fuzzy PID controller shows obvious priority of robustness both in tolerating the change of system parameters and defending against external disturbance.

4.3 Conclusion

In this research, an adaptive neuro-fuzzy PID controller based on TD3 algorithm is proposed. The specially designed neuro-fuzzy PID controller embeds fuzzy system information in the state variables and acts as the actor approximator in the TD3 algorithm, which automatically tune the parameters for multiple PID controllers. The proposed controller is tested on the benchmark inverted pendulum problem with a neuro-linear PID controller provided as comparison. The performance of two controllers are investigated over the transient response test and robustness tests. According to the simulation results, the generalization and the robustness of tolerating the change of system parameters and external disturbance of the proposed controller is proved.

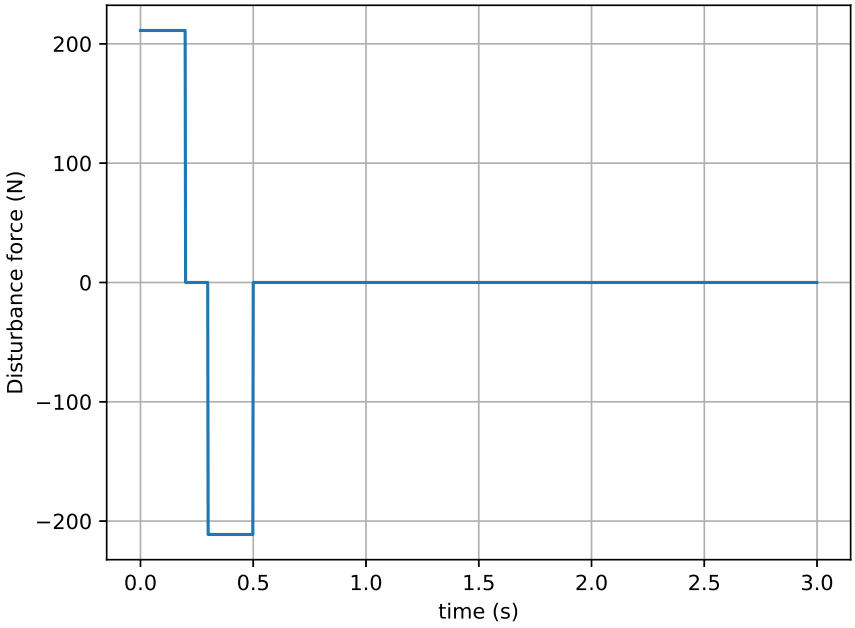

Figure 4.12: Disturbance force

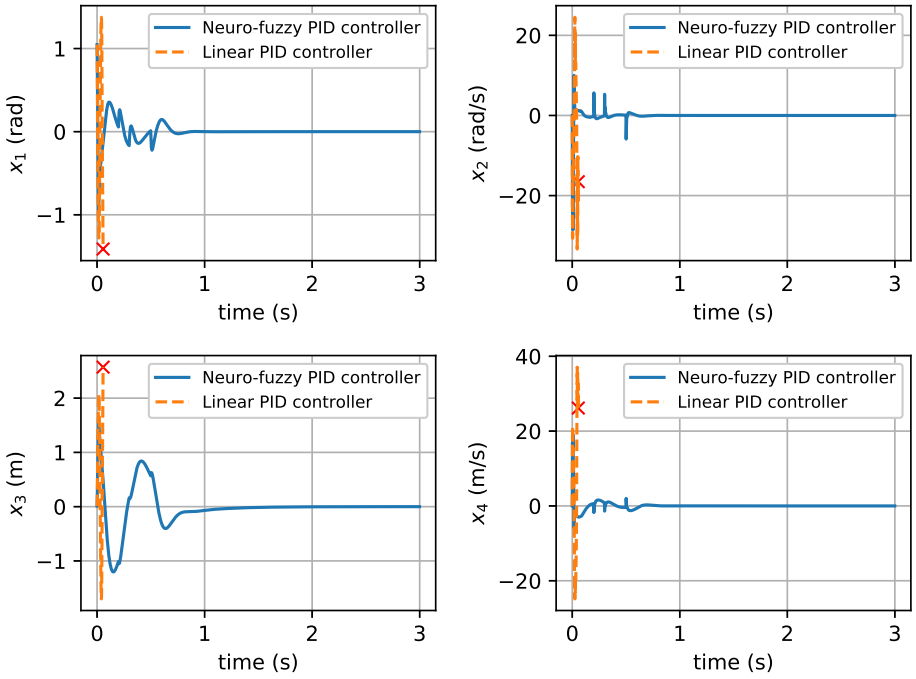

Figure 4.13: Comparison of transient response curves with disturbance

Chapter 5

Off-policy Actor-critic Algorithm with Interval Type-2 Fuzzy PD Controller as Actor Approximator

Compared with type-1 FLS, interval type-2 FLS is more capable of handling complex environments with uncertainties and robust to external disturbances. However, the extra parameters related to the footprint of uncertainty (FOU) in IT2 FMs increase the complexity of the FLS, which brings additional computational burden to the training process of RL based IT2 FLS. Furthermore, the discontinuous type-reduction process in the IT2 FLS lead to high difficulty in calculating the derivation of updating rules in RL algorithms. As a result, in the development of RL based IT2 FLS, proposing comprehensive training structure covering parameters both in antecedent and consequent part adjustable, as well as guarantee efficient training procedure are still not fully addressed.

Regarding to the challenges stated above, in this chapter, an innovative actor-critic algorithm with IT2-FPD as actor approximator is proposed. In this approach, the actor is an IT2-fuzzy PD controller which can be represented in neuro-fuzzy network structure while the the critic remains a FCNN. The parameters in both the antecedent and consequents of the IT2-FPD are automatically optimized, including the shape, position, FOU of the MFs, and the gains of PD controller in each fuzzy rule. The derivation of the update rules with this new approxmiator is provided. For comparison purposes, two other types of controllers are provided as actor approximator in the TD3 algorithm, which is a type-1 fuzzy PD (T1-FPD) controller and a neuro-PD controller, respectively. Similar to the IT2-FPD, the parameters of T1-FPD controller in both the antecedent and the consequent are adjustable. PER technique is applied in RL training process as an acceleration method. The controllers after training are tested on an 2nd order inverted pendulum platform in simulation environments. The code is implemented in Python language, where the toolkit for constructing IT2 FLSs provided by [148] was referenced. The results

verify the feasibility of the proposed controller with better controlling performance and robustness compared with T1-FPD and neuro-PD controller.

5.1 Actor-critic algorithm with IT2-FPD controller as actor approximator

In this section, the details of the AC algorithm with IT2-FPD as function approximator are introduced. The structure of the whole training system can be found in Section 5.1.1 with details of IT2-FPD actor approximator addressed. Section 5.1.2 provides the derivation of update rules in this innovative algorithm and Section 5.1.3 presents the whole RL training procedures in pseudo code.

5.1.1 Structure of actor-critic training system

Fig. 5.1 presents the structure of actor-critic algorithm with IT2-FPD controller as an actor approximator. In the algorithm, both critic networks and target critic networks are FCNNs. Two critic networks approximate the values of value-function and the minimum value of two target critic networks is taken as the target value. Both actor and target actor networks are IT2-FPD controllers. In the k^{th} time step, actor (IT2-FPD controller) observes normalized state $s(t_k)$ and generates action $a(t_k)$. By scaling action with factor K_F, control signal $F(t_{k+1})$ is applied to the environment which obtains new state $s(t_{k+1})$ and reward $r(t_k)$. New sample $[s(t_k), a(t_k), r(r_k), s(t_{k+1})]$ is added to the replay buffer. A mini-batch samples are generated in each time step for updating critic and actor networks and their target networks accordingly.

Structure of IT2-FPD as actor approximator

A typical IT2-FPD controller has similar structure to IT2 T-S model presented in Section 2.2.1. However, IT2-FPD controller has output of PD controller in the consequent of each rule instead of a polynomial expression, which can be expressed as follows. For i^{th} rule,

$$\text{Rule } i\text{: IF } e(t_k) \text{ is } \widetilde{M_1^i} \text{ AND } \Delta e(t_k) \text{ is } \widetilde{M_2^i} \text{ THEN } y_i(t_k) = k_p^i e(t_k) + k_d^i \Delta e(t_k),$$

where $e(t_k)$ and $\Delta e(t_k)$ are the error and the change of error in k^{th} time step; k_p^i and k_d^i are the gains of PD controller in the i^{th} fuzzy rule, accordingly. The output of the IT2-FPD is defined as

$$u(t_k) = \alpha \frac{\sum_{i=1}^{P} \underline{f}^i(\boldsymbol{x}(t_k)) y_i(t_k)}{\sum_{m=1}^{P} \underline{f}^m(\boldsymbol{x}(t_k))} + \beta \frac{\sum_{i=1}^{P} \overline{f}^i(\boldsymbol{x}(t_k)) y_i(t_k)}{\sum_{m=1}^{P} \overline{f}^m(\boldsymbol{x}(t_k))}. \tag{5.1}$$

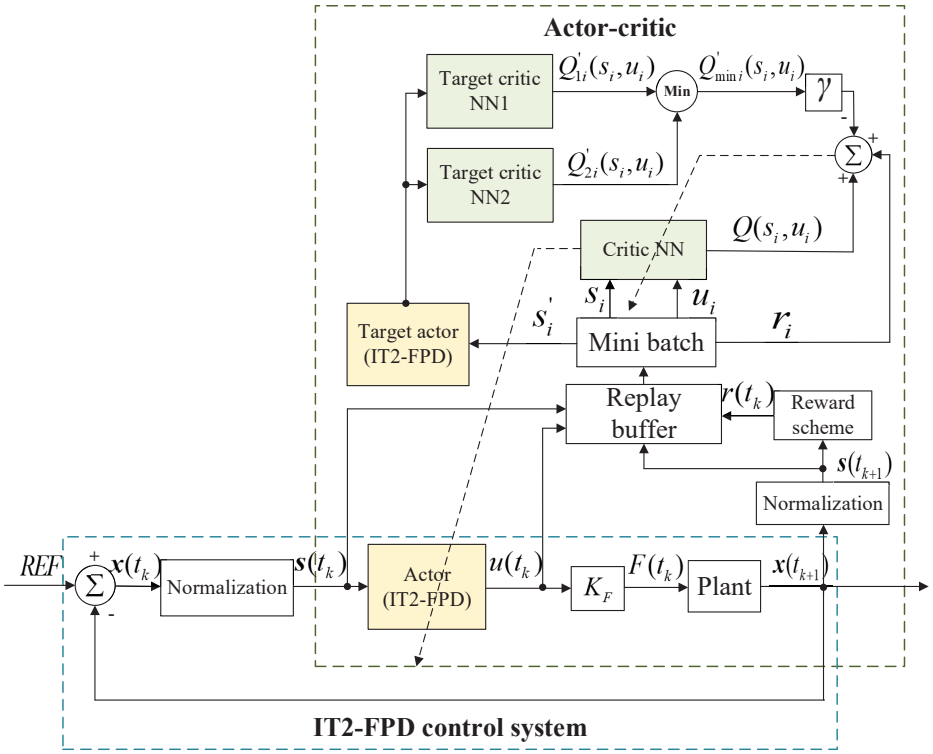

Figure 5.1: Structure of whole training system

In the proposed AC algorithm, both the IT2-FPD controllers in actor and target actor NN are in neuro-fuzzy network structure which is presented in Fig. 5.2. The antecedent of the IT2-FPD refers to the layers before PD layer, while the layers afterwards represent the consequent part.

In this research, the LMF and UMF of IT2-FS are chosen as Gaussian functions with uncertain standard deviation value as follows,

$$\mu_{\widetilde{M_j^i}}(x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^i}{\sigma_j^i}\right)^2\right], \sigma_j^i \in [\underline{\sigma_j^i}, \overline{\sigma_j^i}] \tag{5.2}$$

where $i = (1, 2, ...P)$, $j = (1, 2)$. In the rest of the paper, $y_i(t_k), \underline{f}^i(\boldsymbol{x}(t_k)), \overline{f}^i(\boldsymbol{x}(t_k))$, $\mu_{\widetilde{M_j^i}}(x_j)$ will be written as $y_i, \underline{f}^i, \overline{f}^i, \mu_{\widetilde{M_j^i}}$ for simplicity. The derivation of the update rules introduced in the following section is also based on the cases where IT2-FS are Gaussian functions.

Remark 5.1 *The proposed IT2-FPD demonstrate certain level of explainability [149] of the control policy. The local i^{th} linear PD control policy interprets the control action to be linearly related to the error $e(t_k)$ and the change of the error $\Delta e(t_k)$. The linear relationship, i.e., the PD gains k_p^i and k_d^i, are learnt through the AC learning algorithm. The global control policy is to combine the local PD control policies thought the fuzzy rules. Thanks to the fuzzy rules, the overall control policy can be explained in an IF-THEH format which are human understandable. Through AC learning algorithm, the boundary of individual rule (characterized by membership*

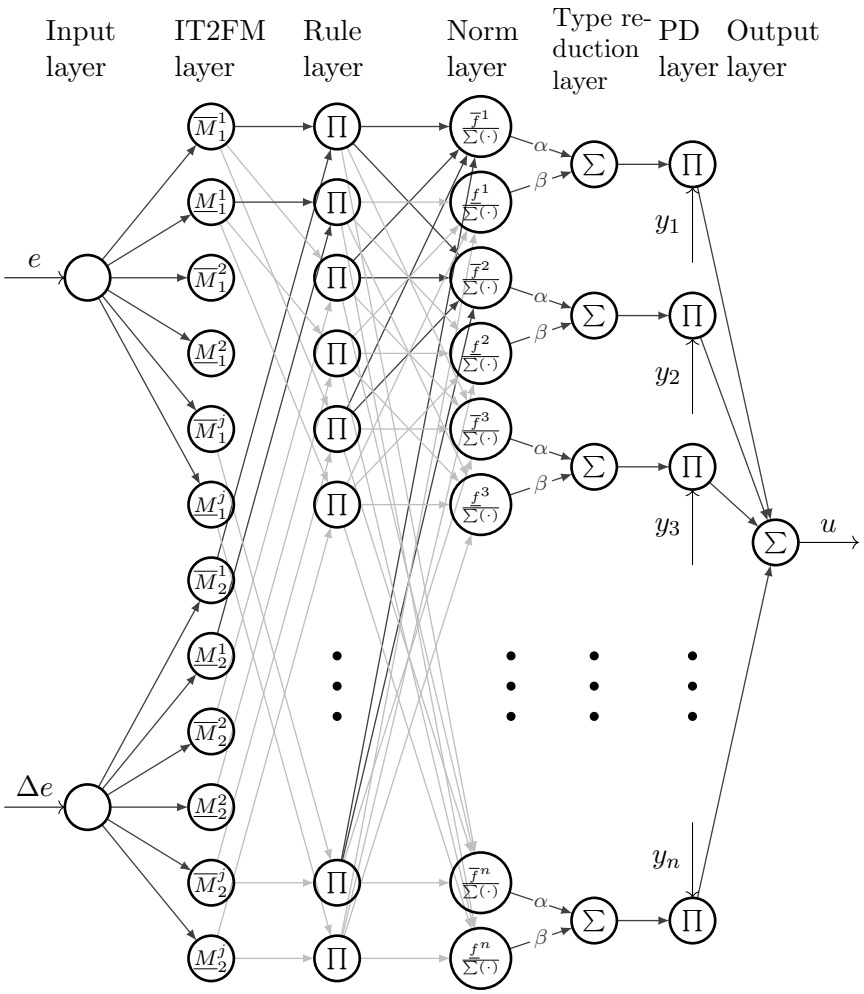

Figure 5.2: IT2-FPD controller in neuro-fuzzy network structure

functions) is learnt.

5.1.2 Derivation of update rules

Considering the specific structure of the actor approximator proposed in this paper, the calculation of gradients for the actor is different from conventional actor with FCNN structure. Therefore, the rest of this section will provide the details of the update rules for fuzzy PD actor approximator.

The parameters in the IT2-FPD actor approximator can be categorized into two sections, which are the parameters in the antecedent and the ones in the consequent. The parameters in the antecedent can be found in the IT2FM layer as shown in Fig. 5.2 while the parameters in the consequent refers to the weights in the PD layer. The goal of the actor is to maximize the Q-value approximated by the critic. If let ϕ denote a general parameter in the actor network, then the update rule of the ϕ can be expressed as

$$\phi(t+1) = \phi(t) + \eta \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \phi}, \tag{5.3}$$

where η is the learning rate of actor. $\frac{\partial Q}{\partial u}$ can be obtained in the update of critic and the expression of $\frac{\partial u}{\partial \phi}$ will be discussed as follows according to the parameters in different situations.

Update rules for consequent parameters in IT2-FPD

Let ϕ_c^i denote the parameter in consequent of i^{th} rule of IT2-FPD controller, the gradient of output u regarding to parameter ϕ_c^i can be expressed as

$$\frac{\partial u}{\partial \phi_c^i} = \frac{\partial u}{\partial y_i} \frac{\partial y_i}{\partial \phi_c^i}. \tag{5.4}$$

According to Eq. 5.1

$$\frac{\partial u}{\partial y^i} = \alpha \frac{\underline{f}^i}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f}^i}{\sum_{m=1}^{P} \overline{f}^m}, \tag{5.5}$$

if $\phi_c^i = k_p^i$

$$\frac{\partial y_i}{\partial k_p^i} = e(t_k), \tag{5.6}$$

if $\phi_c^i = k_d^i$

$$\frac{\partial y_i}{\partial k_d^i} = \Delta e(t_k). \tag{5.7}$$

According to eqs. (5.4) to (5.6), the gradient of k_p^i is

$$\frac{\partial u}{\partial k_p^i} = (\alpha \frac{\underline{f}^i}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f}^i}{\sum_{m=1}^{P} \overline{f}^m}) e(t_k), \tag{5.8}$$

therefore, the update rule of k_p^i is

$$k_p^i(t+1) = k_p^i(t) + \eta \frac{\partial Q}{\partial u} (\alpha \frac{\underline{f}^i}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f}^i}{\sum_{m=1}^{P} \overline{f}^m}) e(t_k). \tag{5.9}$$

According to eqs. (5.4), (5.5) and (5.7), the gradient of k_d^i is

$$\frac{\partial u}{\partial k_d^i} = (\alpha \frac{\underline{f^i}}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f^i}}{\sum_{m=1}^{P} \overline{f}^m}) \Delta e(t_k), \tag{5.10}$$

and the update rule of k_d^i is

$$k_d^i(t+1) = k_d^i(t) + \eta \frac{\partial Q}{\partial u} (\alpha \frac{\underline{f^i}}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f^i}}{\sum_{m=1}^{P} \overline{f}^m}) \Delta e(t_k). \tag{5.11}$$

Update rules of antecedent parameters in IT2-FPD

The parameters in the antecedent required optimization are $m_j^i, \underline{\sigma}_j^i, \overline{\sigma}_j^i$. As to guarantee LMF is always less than UMF for all input values, $\underline{\sigma}$ and $\overline{\sigma}$ are presented by σ and κ as

$$\underline{\sigma}_j^i = \sigma_j^i - |\kappa| \sigma_j^i \tag{5.12}$$

and

$$\overline{\sigma}_j^i = \sigma_j^i + |\kappa| \sigma_j^i. \tag{5.13}$$

Therefore, the contents below provide the derivation of update rules regarding to variable m_j^i, σ and κ.

According to the general update rule equation provided in Eq. 5.3, the unknown part is $\frac{\partial u}{\partial \phi_a^i}$, where ϕ_a^i represents a general parameter of the i^{th} rule in the antecedent. When $\phi_a^i = m_j^i$

$$\frac{\partial u}{\partial \phi_a^i} = \frac{\partial u}{\partial m_j^i} = \frac{\partial u}{\partial \underline{f}^i} \frac{\partial \underline{f}^i}{\partial m_j^i} + \frac{\partial u}{\partial \overline{f}_i} \frac{\partial \overline{f^i}}{\partial m_j^i}. \tag{5.14}$$

According to Eq. 5.1,

$$\frac{\partial u}{\partial \underline{f}^i} = \alpha \frac{y_i - u}{\sum_{m=1}^{P} \underline{f}^m}, \tag{5.15}$$

$$\frac{\partial u}{\partial \overline{f}^i} = \beta \frac{y_i - u}{\sum_{m=1}^{P} \overline{f}^m}. \tag{5.16}$$

Eq. 2.4, Eq. 2.5 and Eq. 5.2 provide

$$\begin{aligned}\frac{\partial \underline{f^i}}{\partial m_j^i} &= \frac{\partial \underline{f^i}}{\partial \underline{\mu}_{\widetilde{M}_j^i}} \frac{\partial \underline{\mu}_{\widetilde{M}_j^i}}{\partial m_j^i} \\ &= \prod_{k=1,k\neq j}^{\psi} \underline{\mu}_{\widetilde{M}_k^i} \left[\underline{\mu}_{\widetilde{M}_j^i} \frac{\partial \left(-\frac{1}{2} \left(\frac{x_j - m_j^i}{\underline{\sigma}_j^i} \right)^2 \right)}{\partial m_j^i} \right] \\ &= \underline{f_i} \frac{(x_j - m_j^i)}{\underline{\sigma}_j^{i^2}}.\end{aligned} \tag{5.17}$$

Similarly,

$$\begin{aligned}\frac{\partial \overline{f^i}}{\partial m_j^i} &= \frac{\partial \overline{f^i}}{\partial \overline{\mu}_{\widetilde{M}_j^i}} \frac{\partial \overline{\mu}_{\widetilde{M}_j^i}}{\partial m_j^i} \\ &= \overline{f^i} \frac{(x_j - m_j^i)}{\overline{\sigma}_j^{i^2}}.\end{aligned} \tag{5.18}$$

Then Eq. 5.14 can be expressed as

$$\begin{aligned}\frac{\partial u}{\partial m_j^i} = \alpha &\frac{\underline{f^i}(y_i - u)(x_j - m_j^i)}{\underline{\sigma}_j^{i^2} \sum_{k=1}^{P} \underline{f}^k} \\ &+ \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)}{\overline{\sigma}_j^{i^2} \sum_{k=1}^{P} \overline{f}^k}.\end{aligned} \tag{5.19}$$

Therefore the update rule of m_j^i is

$$\begin{aligned}m_j^i(t+1) &= m_j^i(t) + \eta \frac{\partial Q}{\partial u} \frac{\partial u}{\partial m_j^i} \\ &= m_j^i(t) + \eta \frac{\partial Q}{\partial u} (\alpha \frac{\underline{f^i}(y_i - u)(x_j - m_j^i)}{\underline{\sigma}_j^{i^2} \sum_{k=1}^{P} \underline{f}^k} \\ &+ \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)}{\overline{\sigma}_j^{i^2} \sum_{k=1}^{P} \overline{f}^k}).\end{aligned} \tag{5.20}$$

When $\phi_a^i = \sigma_j^i$, the expression of $\frac{\partial u}{\partial \sigma_j^i}$ can be expressed as

$$\frac{\partial u}{\partial \sigma_j^i} = \frac{\partial u}{\partial \underline{f^i}} \frac{\partial \underline{f^i}}{\partial \underline{\sigma}_j^i} \frac{\partial \underline{\sigma}_j^i}{\partial \sigma_j^i} + \frac{\partial u}{\partial \overline{f^i}} \frac{\partial \overline{f}^i}{\partial \overline{\sigma}_j^i} \frac{\partial \overline{\sigma}_j^i}{\partial \sigma_j^i}, \tag{5.21}$$

where

$$\begin{aligned}\frac{\partial \underline{f}^i}{\partial \underline{\sigma}_j^i} &= \frac{\partial \underline{f}^i}{\partial \underline{\mu}_{\widetilde{M}_j^i}} \frac{\partial \underline{\mu}_{\widetilde{M}_j^i}}{\partial \underline{\sigma}_j^i} \\ &= \prod_{k=1,k\neq j}^{\psi} \underline{\mu}_{\widetilde{M}_k^i} \left[\underline{\mu}_{\widetilde{M}_j^i} \frac{\partial\left(-\frac{1}{2}\left(\frac{x_j - m_j^i}{\underline{\sigma}_j^i}\right)^2\right)}{\partial \underline{\sigma}_j^i} \right] \\ &= \underline{f}^i \frac{(x_j - m_j^i)^2}{(\underline{\sigma}_j^i)^3},\end{aligned} \tag{5.22}$$

while

$$\frac{\partial \overline{f}^i}{\partial \overline{\sigma}_j^i} = \overline{f^i} \frac{(x_j - m_j^i)^2}{(\overline{\sigma}_j^i)^3}. \tag{5.23}$$

Therefore,

$$\frac{\partial u}{\partial \underline{f}^i} \frac{\partial \underline{f}^i}{\partial \underline{\sigma}_j^i} \frac{\partial \underline{\sigma}_j^i}{\partial \sigma_j^i} = \alpha \frac{\underline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\underline{\sigma}_j^i)^3 \sum_{m=1}^{P} \underline{f}^m}. \tag{5.24}$$

Similarly,

$$\frac{\partial u}{\partial \overline{f}^i} \frac{\partial \overline{f}^i}{\partial \overline{\sigma}_j^i} \frac{\partial \overline{\sigma}_j^i}{\partial \sigma_j^i} = \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\overline{\sigma}_j^i)^3 \sum_{m=1}^{P} \overline{f}^m}. \tag{5.25}$$

The gradient of u regarding to σ_j^i is

$$\frac{\partial u}{\partial \sigma_j^i} = \alpha \frac{\underline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\underline{\sigma}_j^i)^3 \sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\overline{\sigma}_j^i)^3 \sum_{m=1}^{P} \overline{f}^m}, \tag{5.26}$$

which provides the update rule of σ_j^i

$$\begin{aligned}\sigma_j^i(t+1) &= \sigma_j^i(t) + \eta \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \sigma_j^i} \\ &= \sigma_j^i(t) + \eta \frac{\partial Q}{\partial u} (\alpha \frac{\underline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\underline{\sigma}_j^i)^3 \sum_{m=1}^{P} \underline{f}^m} \\ &+ \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\overline{\sigma}_j^i)^3 \sum_{m=1}^{P} \overline{f}^m}).\end{aligned} \tag{5.27}$$

When $\phi_a^i = |\kappa|$, the expression of $\frac{\partial u}{\partial |\kappa|}$ is

$$\frac{\partial u}{\partial |\kappa|} = \frac{\partial u}{\partial \underline{f}_i} \frac{\partial \underline{f}_i}{\partial \underline{\sigma}_j^i} \frac{\partial \underline{\sigma}_j^i}{\partial |\kappa|} + \frac{\partial u}{\partial \overline{f}_i} \frac{\partial \overline{f}_i}{\partial \overline{\sigma}_j^i} \frac{\partial \overline{\sigma}_j^i}{\partial |\kappa|} \tag{5.28}$$

According to Eq. 5.12, Eq. 5.13, Eq.5.22 and Eq. 5.23, the above equation can be

rewritten as

$$\begin{aligned}\frac{\partial u}{\partial|\kappa|} =& \frac{\partial u}{\partial \underline{f_i}}\frac{\partial \underline{f}_i}{\partial \underline{\sigma}_j^i}\frac{\partial \underline{\sigma}_j^i}{\partial|\kappa|} + \frac{\partial u}{\partial \overline{f_i}}\frac{\partial \overline{f}_i}{\partial \overline{\sigma}_j^i}\frac{\partial \overline{\sigma}_j^i}{\partial|\kappa|} \\ =& -\alpha\frac{\sigma_j^i \underline{f_i}(y_i-u)(x_j-m_j^i)^2}{(\underline{\sigma}_j^i)^3\sum_{m=1}^{P}\underline{f}^m} \\ &+\beta\frac{\sigma_j^i \overline{f_i}(y_i-u)(x_j-m_j^i)^2}{(\overline{\sigma}_j^i)^3\sum_{m=1}^{P}\overline{f}^m}.\end{aligned} \tag{5.29}$$

As a result, the update rule for $|\kappa|$ is

$$\begin{aligned}|\kappa|(t+1) = |\kappa|(t) + \eta(&-\alpha\frac{\sigma_j^i \underline{f_i}(y_i-u)(x_j-m_j^i)^2}{(\underline{\sigma}_j^i)^3\sum_{m=1}^{P}\underline{f}^m} \\ &+\beta\frac{\sigma_j^i \overline{f_i}(y_i-u)(x_j-m_j^i)^2}{(\overline{\sigma}_j^i)^3\sum_{m=1}^{P}\overline{f}^m}).\end{aligned} \tag{5.30}$$

Update rules of parameters in T1-FPD

Remark 5.2 *The update rule of T1-FPD controller can be regarded as a special case when the LMF equals UMF in the IT2-FPD controller. In the previous researches, though the optimization of T1 fuzzy controller based on RL training progress has been discussed, no comprehensive update rules are presented for T1 fuzzy PD controller with both parameters in the antecedent and consequent adjustable. Hence, this subsection provides the update rules of T1-FPD controller in AC training algorithm as a supplementary.*

Let ψ denote a general parameter in the T1-FPD actor approximator. The update rule of the ψ is expressed as

$$\psi(t+1) = \psi(t) + \eta\frac{\partial Q}{\partial u}\frac{\partial u}{\partial \psi}, \tag{5.31}$$

where η is the learning rate of actor. $\frac{\partial Q}{\partial u}$ is provided in the update procedure of critic, the rest of this section will focus on derivation of $\frac{\partial u}{\partial \psi}$ for different parameters. As stated above, the T1-FPD is a special case when LMF equals UMF which gives,

$$\begin{aligned}&\underline{f}^i = \overline{f}^i = f^i, \\ &\underline{\sigma} = \overline{\sigma} = \sigma, \\ &\kappa = 0,\end{aligned} \tag{5.32}$$

where f_i, σ represents the firing strength of i^{th} fuzzy rule and the standard deviation of Gaussian MF in T1-FPD controller. The update rules of parameters in IT2-FPD

controller can be reduced as follows,

$$\begin{aligned} k_p^i(t+1) &= k_p^i(t) + \eta(\alpha \frac{\underline{f}^i}{\sum_{m=1}^{P} \underline{f}^m} + \beta \frac{\overline{f}^i}{\sum_{m=1}^{P} \overline{f}^m}) e(t_k) \\ &= k_p^i(t) + \eta(\alpha+\beta) \frac{f^i}{\sum_{m=1}^{P} f^m} e(t_k) \\ &= k_p^i(t) + \eta \frac{f^i}{\sum_{m=1}^{P} f^m} e(t_k). \end{aligned} \tag{5.33}$$

Similarly, the update rule for k_d^i can be reduced as

$$k_d^i(t+1) = k_d^i(t) + \eta \frac{f^i}{\sum_{m=1}^{P} f^m} \Delta e(t_k). \tag{5.34}$$

In the antecedent of T1-FPD controller, only two parameters needs optimization, which are m_j^i, σ_j^i. The update rule of m_j^i is

$$\begin{aligned} m_j^i(t+1) &= m_j^i(t) + \eta(\alpha \frac{\underline{f_i}(y_i - u)(x_j - m_j^i)}{\underline{\sigma}_j^{i^2} \sum_{k=1}^{P} \underline{f}^k} \\ &\quad + \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)}{\overline{\sigma}_j^{i^2} \sum_{k=1}^{P} \overline{f}^k}) \\ &= m_j^i(t) + \eta((\alpha+\beta) \frac{f^i(y_i - u)(x_j - m_j^i)}{\sigma_j^{i^2} \sum_{k=1}^{P} f^k}) \\ &= m_j^i(t) + \eta(\frac{f_i(y_i - u)(x_j - m_j^i)}{\sigma_j^{i^2} \sum_{k=1}^{P} f^k}). \end{aligned} \tag{5.35}$$

The update rule of σ_j^i can be expressed as

$$\begin{aligned} \sigma_j^i(t+1) &= \sigma_j^i(t) + \eta(\alpha \frac{\underline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\underline{\sigma}_j^i)^3 \sum_{m=1}^{P} \underline{f}^m} \\ &\quad + \beta \frac{\overline{f^i}(y_i - u)(x_j - m_j^i)^2}{(\overline{\sigma}_j^i)^3 \sum_{m=1}^{P} \overline{f}^m}) \\ &= \sigma_j^i(t) + \eta((\alpha+\beta) \frac{f^i(y_i - u)(x_j - m_j^i)^2}{(\sigma_j^i)^3 \sum_{m=1}^{P} f^m}) \\ &= \sigma_j^i(t) + \eta(\frac{f^i(y_i - u)(x_j - m_j^i)^2}{(\sigma_j^i)^3 \sum_{m=1}^{P} f^m}). \end{aligned} \tag{5.36}$$

5.1.3 Pseudo code of training procedure

The training procedure of the proposed algorithm is shown in Algorithm 5. The algorithm proposed in this chapter references the similar structure applied in TD3 algorithm with PER technique added to increase the efficiency of the training process. The main difference focuses on the actor approximator. Instead of using FCNN

for actor function approximation, an IT2-FPD controller in neural format is utilized in the proposed algorithm to generate the action signal, where the update rules for actor in NN formalism no longer apply. The derivation of updating rules for the IT2-FPD controller are specified in Section 5.1.1. Accordingly, the consequent parameters of IT2-FPD actor follows the update rules provided in Eq. 5.9 and Eq. 5.11, while update antecedent parameters according to Eq. 5.20, Eq. 5.27 and Eq. 5.29.

Algorithm 5: Novel TD3 algorithm with IT2-FPD as actor approximator

Initialize critic networks $Q_1(s,a|\theta_1)$ and $Q_2(s,a|\theta_2)$, IT2-FPD actor network $\pi_f(s|\phi_f)$;
Initialize target critic networks Q'_1, Q'_2 with $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$ and target actor network π'_f with $\phi'_f \leftarrow \phi_f$;
Initialize replay buffer $\mathcal{R}$, mini-batch K, exponents λ and ρ;
For every episode:
Initialize state s;
while $t = 1 : T$ **do**
    Select action $a \sim \pi_f(s|\phi_f) + \mathcal{N}(0,\sigma)$;
    Observe next state s' and reward r;
    Store transition tuple (s,a,r,s',p^*) in $\mathcal{R}$, where p^* is the maximum priority in replay buffer;
    Sample mini-batch of K transitions with probability $P(i) \sim \frac{p_i^\lambda}{\sum_{j=1}^{K} p_j^\lambda}$;
    Calculate importance sampling weight (ISW)
    $w_i = \frac{(KP(i))^{-\rho}}{w^*}$, $w^* = \max_j w_j$ $(j = 1, 2, ..., K)$;
    Get $a'_i \sim \pi'_f(s'_i|\phi') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0,\sigma), -c, c)$;
    Set $y_i = r_i + \gamma \min\{Q'_{1i}(s'_i, a'_i|\theta'_1), Q'_{2i}(s'_i, a'_i|\theta'_2)\}$;
    Calculate TD-error $\delta_{im} = y_i - Q_m(s_i, a_i|\theta_m)$, $m = 1, 2$;
    Update critic network Q_m by minimizing loss
    $L_m = \frac{1}{K}\sum_{i=1}^{K} w_i \delta_{im}^2$, $m = 1, 2$;
    Update priority of selected samples in replay buffer $p_i = |\delta_i| + \epsilon_p$ $(i = 1, 2, ..., K)$;
    while $t \bmod d = 0$ **do**
        Update consequent parameters following Eq. 5.9 and Eq. 5.11;
        Update antecedent parameters following Eq. 5.20, Eq. 5.27 and Eq. 5.29;
        Update target networks:
        $\theta_m \leftarrow \tau\theta_m + (1-\tau)\theta'_m$, $m = 1, 2$
        $\phi'_f \leftarrow \tau\phi_f + (1-\tau)\phi'_f$
    end
    $s \leftarrow s'$;
end

5.2 Simulation results

In this section, the training of IT2-FPD controller with the proposed novel actor-critic algorithm is tested on the 2^{nd} order inverted pendulum platform in simulation environment. Two comparison controllers are provided, which are type-1 fuzzy PD controller (T1-FPD) and neuro-PD controller, respectively. As to compare the performance of the trained controllers, four tests which are comparison of transient response, random initial position test, robustness test and noise test are conducted on the same inverted pendulum system. In different tests, all the controllers aim to drive the inverted pendulum back to equilibrium point ($\theta = 0, \dot{\theta} = 0$) with less settling time, overshooting and oscillations. The sum of absolute error (SAE) is chosen as the measurement metric for comparing the oscillations during transient response. In the transient response test, the response properties are compared under control of three different controllers starting from variant initial positions. In the robustness test, different system parameters are tested as to explore the tolerance of trained controllers under variant circumstances, of which results are presented with phase portraits. Finally in the noise test, a clipped Gaussian noise is added to the state vector and the transient response curves are compared among three controllers as to assess the performance of different controllers against external disturbance.

5.2.1 Comparison Controllers

Neuro-PD Controller

The structure of neuro-PD controller is presented in Fig. 5.3. The output of the neuro-PD controller can be expressed as

$$a = k_p e + k_d \Delta e. \tag{5.37}$$

The control force applied to the inverted pendulum system is

$$u = K_F a, \tag{5.38}$$

where K_F is the scale factor, i.e., $K_F = 400$ in this simulation is chosen in this simulation according to the trial-and-error method.

Remark 5.3 *The neuro-PD controller can be regarded as a special case of neuro-fuzzy PD controller where the number of fuzzy rule is reduced to 1.*

For comparison purposes, the neuro-PD controller is trained with same parameters as set for IT2-FPD controller except for different learning rates.

T1-FPD controller

The fuzzy rule of T1-FPD controller is defined as

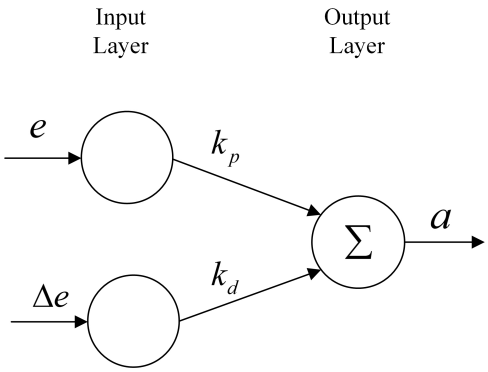

Figure 5.3: Structure of neuro-PD controller as actor network

Rule i: IF $e(t_k)$ is M_1^i AND $\Delta e(t_k)$ is M_2^i THEN $y_i(t_k) = k_p^i e(t_k) + k_d^i \Delta e(t_k)$, where M_1^i and M_2^i are type-1 fuzzy MFs in the i^{th} fuzzy rule.

The output of the T1-FPD control is defined as

$$u(t_k) = \sum_{i=1}^{P} f^i(\boldsymbol{x}(t_k)) y_i(t_k). \tag{5.39}$$

Gaussian function with standard deviation value σ is chosen as the T1 MFs which are defined as

$$\mu_{M_j^i}(x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^i}{\sigma_j^i}\right)^2\right], \tag{5.40}$$

where $i = (1, 2, ...P)$, $j = (1, 2)$. Similarly, in the rest of the paper, $y_i(t_k)$, $f^i(\boldsymbol{x}(t_k))$ and $\mu_{M_j^i}(x_j)$ will be written as y_i, f^i and $\mu_{M_j^i}$, respectively, for simplicity.

The initial parameter settings are similar to the ones set in IT2-FPD controller. The m value of Gaussian function in T1-FPD is chosen as the same value in the corresponding IT2 MF in IT2-FPD controller. The σ value of the Gaussian function in T1-FPD uses the average value of $\underline{\sigma}$ and $\overline{\sigma}$ initialised in IT2-FPD controller, which is $\sigma = \frac{(\underline{\sigma}+\overline{\sigma})}{2}$. The details of initial parameter settings are shown in Tab. 5.1. The initialized T1 Gaussian functions are shown in Fig. 5.4 and Fig. 5.5.

Table 5.1: T1 MFs of $e(t_k)$ and $\Delta e(t_k)$ in initialization

MFs	m	σ
ES	0.0	0.15
EM	0.5	0.15
EB	1.0	0.15
DES	0.0	0.15
DEM	0.5	0.15
DEB	1.0	0.15

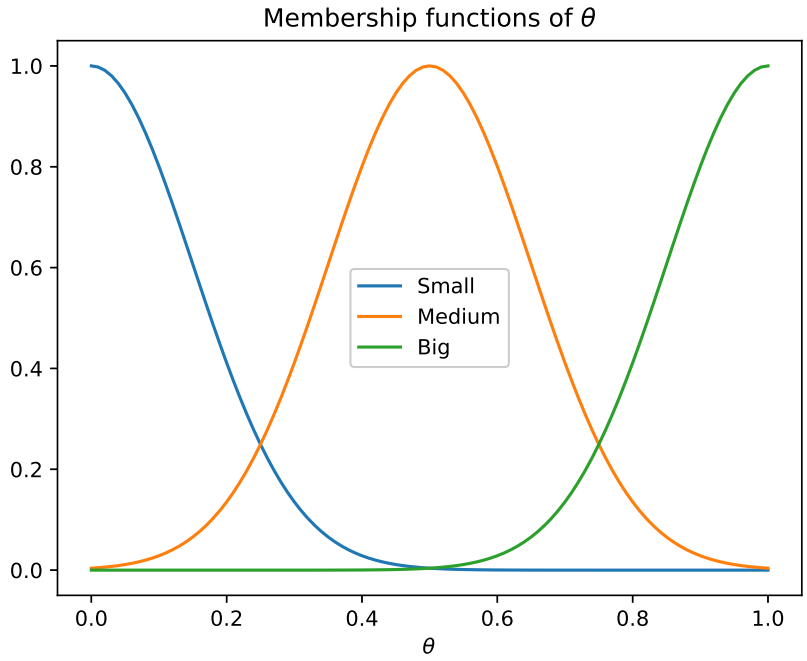

Figure 5.4: T1 MFs of $e(t_k)$ in initialization

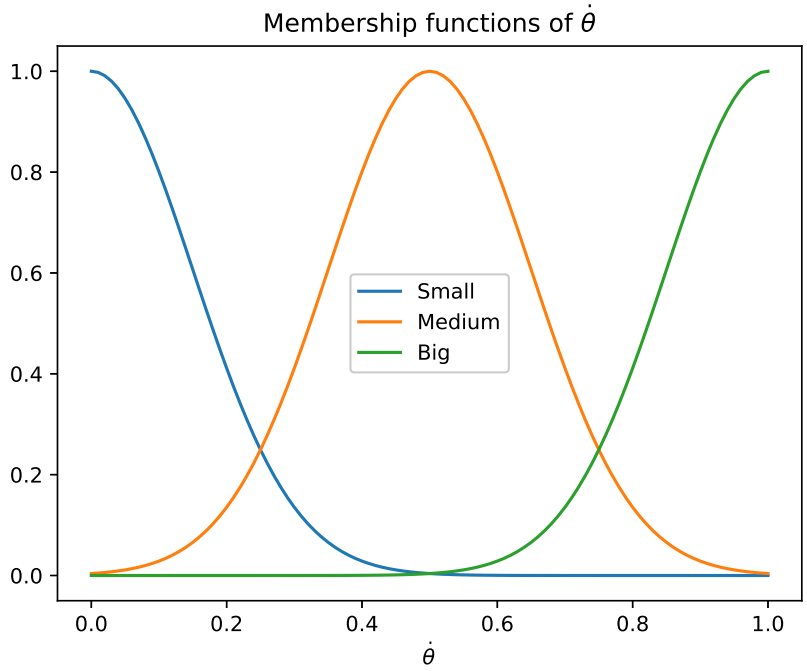

Figure 5.5: T1 MFs of $\Delta e(t_k)$ in initialization

5.2.2 Parameter settings

RL parameters

The critic networks in three AC algorithms are fully connected NNs with size $2 \times 200 \times 100 \times 1$. The actor NN and target actor NN are IT2-FPD controllers, which are defined as follows,

Rule 1: IF $e(t_k)$ is EB AND $\Delta e(t_k)$ is DEB THEN $y_1(t_k) = k_p^1 e(t_k) + k_d^1 \Delta e(t_k)$,
Rule 2: IF $e(t_k)$ is EB AND $\Delta e(t_k)$ is DEM THEN $y_2(t_k) = k_p^2 e(t_k) + k_d^2 \Delta e(t_k)$,
Rule 3: IF $e(t_k)$ is EB AND $\Delta e(t_k)$ is DES THEN $y_3(t_k) = k_p^3 e(t_k) + k_d^3 \Delta e(t_k)$,
Rule 4: IF $e(t_k)$ is EM AND $\Delta e(t_k)$ is DEB THEN $y_4(t_k) = k_p^4 e(t_k) + k_d^4 \Delta e(t_k)$,
Rule 5: IF $e(t_k)$ is EM AND $\Delta e(t_k)$ is DEM THEN $y_5(t_k) = k_p^5 e(t_k) + k_d^5 \Delta e(t_k)$,
Rule 6: IF $e(t_k)$ is EM AND $\Delta e(t_k)$ is DES THEN $y_6(t_k) = k_p^6 e(t_k) + k_d^6 \Delta e(t_k)$,
Rule 7: IF $e(t_k)$ is ES AND $\Delta e(t_k)$ is DEB THEN $y_7(t_k) = k_p^7 e(t_k) + k_d^7 \Delta e(t_k)$,
Rule 8: IF $e(t_k)$ is ES AND $\Delta e(t_k)$ is DEM THEN $y_8(t_k) = k_p^8 e(t_k) + k_d^8 \Delta e(t_k)$,
Rule 9: IF $e(t_k)$ is ES AND $\Delta e(t_k)$ is DES THEN $y_9(t_k) = k_p^9 e(t_k) + k_d^9 \Delta e(t_k)$,

where the initial k_p^i and k_d^i are random values normally sampled from $[0, 100]$ ($i = 1, 2, ..., 9$).

The initial parameters of IT2 MFs are shown in Tab. 5.2 with shapes shown in Fig. 5.6 and Fig. 5.7 for better illustration. The parameters for RL learning process are shown in Tab. 5.3.

Table 5.2: Initial parameter settings of IT2 MFs

MFs	m	σ	$\Delta\sigma$
ES	0.0	0.15	0.07
EM	0.5	0.15	0.07
EB	1.0	0.15	0.07
DES	0.0	0.15	0.07
DEM	0.5	0.15	0.07
DEB	1.0	0.15	0.07

Inverted pendulum parameters

The parameters of the 2^{nd} order inverted pendulum system are shown in Tab. 5.4. In the rest of tests, the objective of different controllers is to balance the inverted

Table 5.3: Parameters in RL training process

Variable	Value
actor lr	1.9×10^{-4}
critic lr	0.4×10^{-4}
max episodes	2000
steps per episode	500
d	3
K	16
buffer size	10^6
τ	10^{-3}
γ	0.99
α	0.3
β	0.7

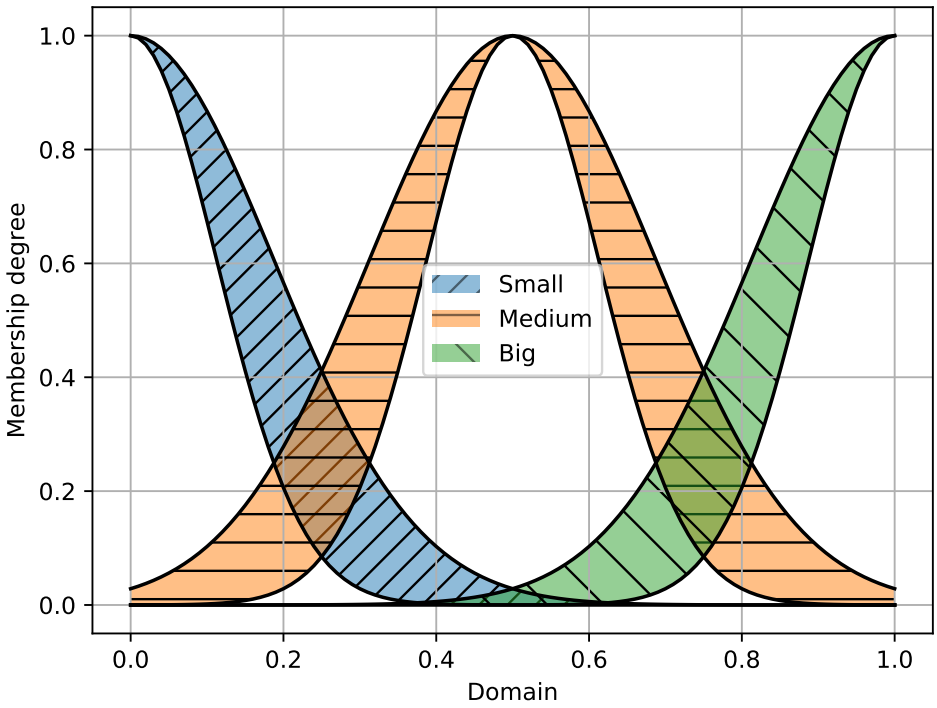

Figure 5.6: IT2 MFs of $e(t_k)$ in initialization

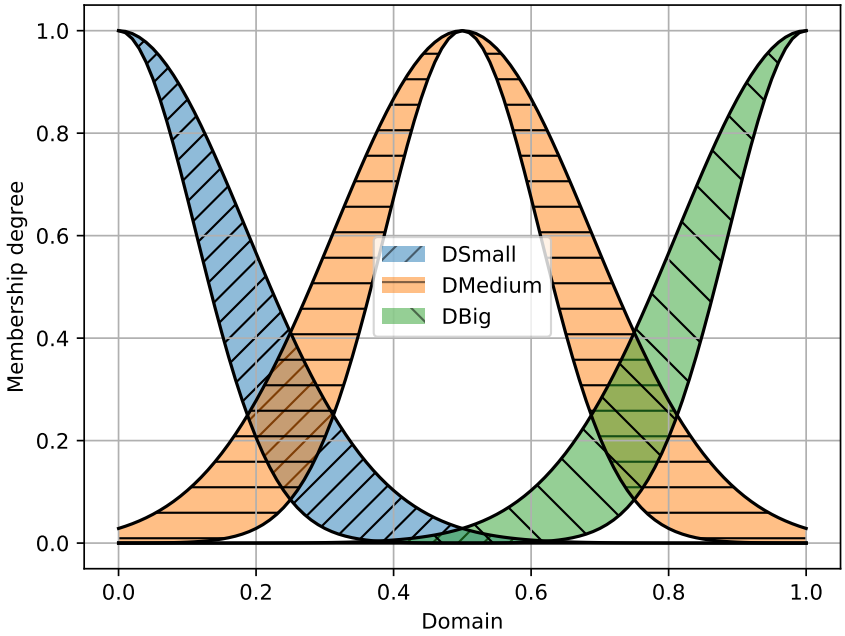

Figure 5.7: IT2 MFs of $\Delta e(t_k)$ in initialization

pendulum system back to equilibrium point ($\theta = 0, \dot{\theta} = 0$) with shorter settling time and smaller magnitude of oscillations.

Reward scheme

In each time step, the agent receives a positive reward when the episode is not terminated. The positive reward is smaller than 1 and depends on the observed state and chosen action. As defined in Eq.5.41, the value of the positive reward

Table 5.4: Parameters settings of inverted pendulum system

Variable	Value
m	2.0kg
M	8.0kg
l	0.5m

is affected by the absolute value of error e, the change of error Δe and action a. The reward is larger when system tends to decrease the error and velocity with less control signal. When the system stays at equilibrium point without control forces, the reward reaches the maximum value 1. Therefore, from the transient response view, as to maximise the accumulated long-term reward, the controller is encouraged to balance the system with shorter settling time and smaller magnitude of oscillations. If the task fails before the episode ends, the agent get a punishment -5 and the Boolean variable is set as $fail = 1$. The definition can be find as follows

$$r = \begin{cases} 0.8(1 - |e|) + 0.1(1 - |\Delta e|) + 0.1(1 - |a|), \text{if } fail = 0 \\ -5, \text{if } fail = 1, \end{cases} \tag{5.41}$$

where e and Δe are the expression of $e(t_k)$ and $\Delta e(t_k)$ for simplicity. The Boolean variable $fail$ indicates the failure of the task, which is defined in Eq. 5.42. When $fail = 1$, the episode is terminated before reaching the maximum time steps.

$$fail = \begin{cases} 1, \text{if } |\theta| > \frac{60}{180}\pi \\ 0, \text{otherwise} \end{cases} \tag{5.42}$$

5.2.3 Training results

All three controllers are trained with random initial positions, where θ is uniformly sampled from the range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$, $\dot{\theta} = 0$. Each episode has maximum 500 time steps with 0.01 time step. If θ exceeds the preset range, the episode is terminated and the *fail* flag is set as 1. The total training process has 2000 episodes. The learning curves of three controllers are shown in Fig. 5.8. The IT2-FPD controller converges after 1500 episodes. T1-FPD controller approaches to the maximum accumulated reward in the end of the training process (around 1900 episode) while the neuro-PD controller only stays around the maximum reward value for several episodes and diverge afterwards. Since IT2-FPD and T1-FPD achieved convergent results, the trained parameters at 2000 episode are chosen as the data used for the rest of the tests. However, the training results of neuro-PD controller diverge at 2000 episode. If the trained parameters at 2000 episode are used, neuro-PD will fail to balance the inverted pendulum system. In the following comparison tests, it is to compare the properties of each controller with the best performance after training. Therefore,

instead of choosing the diverged results of neuro-PD controller at 2000 steps, the parameters at the 1400-th training episode are chosen for the following experiments.

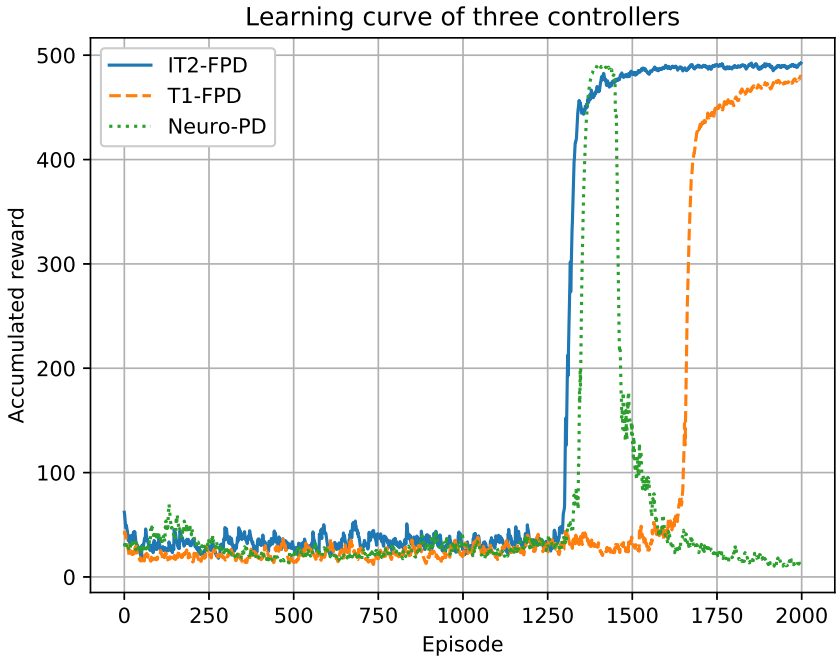

Figure 5.8: Learning curve of three controllers as actor approximator within 2000 episodes (the blue solid line indicates the IT2-FPD controller, the orange dashed line indicates the T1-FPD controller and the green dotted line indicates the neuro-PD controller)

The IT2 MFs of the antecedent in the IT2-FPD controller are shown in Fig. 5.9 and Fig. 5.10 with parameters presented in Tab. 5.5. The gains of IT2-FPD controller in the consequent after training are shown in Tab. 5.6.

Table 5.5: IT2 MFs of $e(t_k)$ and $\Delta e(t_k)$ after training

MFs	m	σ	$\Delta\sigma$
ES	-0.0377	0.1331	0.0678
EM	0.5125	0.1091	0.0543
EB	1.0028	0.2500	0.1297
DES	-0.0748	0.0352	0.0177
DEM	0.3658	0.3552	0.1804
DEB	0.9980	0.1566	0.0784

The parameters of T1 MFs after training are shown in Tab. 5.7. Fig. 5.11 and Fig. 5.12 present the shape of T1 MFs after training. The gains of each rule in T1-FPD controller are shown in Tab. 5.8.

The gains of neuro PD controller at 1400 training episode are

$$k_p = -283.8092,$$

Table 5.6: K_p and K_d of IT2-FPD controller after training

	k_p	k_d
Rule1	-70.8901	-39.7840
Rule2	-632.5352	-121.4476
Rule3	-278.3510	-105.5429
Rule4	-55.2187	-43.9395
Rule5	-518.1646	12.6886
Rule6	-458.1053	-119.0944
Rule7	-98.0051	-73.9781
Rule8	-537.6866	-478.1020
Rule9	-688.8746	-117.5255

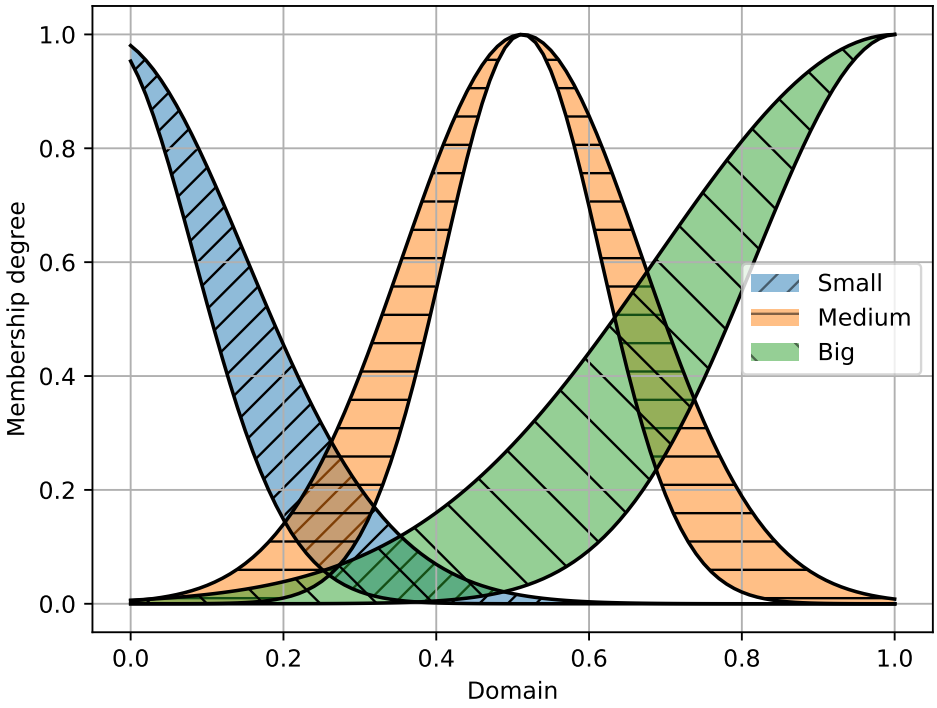

Figure 5.9: IT2 MFs of $e(t_k)$ after training

Table 5.7: T1 MFs of $e(t_k)$ and $\Delta e(t_k)$ after training

MFs	m	σ
ES	-0.0002	0.1676
EM	0.3328	0.8809
EB	1.2991	-1.0000
DES	-0.0018	0.3054
DEM	0.8038	-1.0000
DEB	1.0002	0.1366

$$k_d = -244.3432.$$

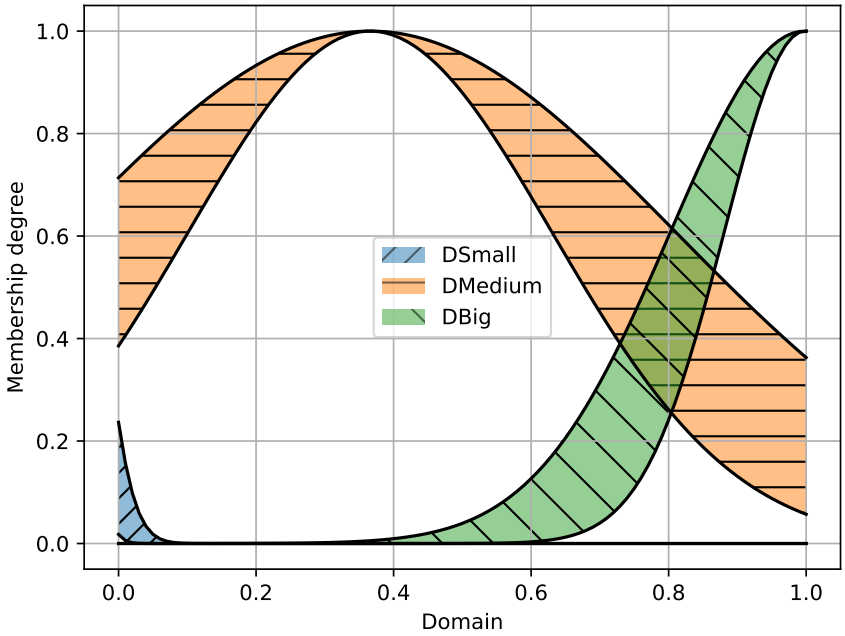

Figure 5.10: IT2 MFs of $\Delta e(t_k)$ after training

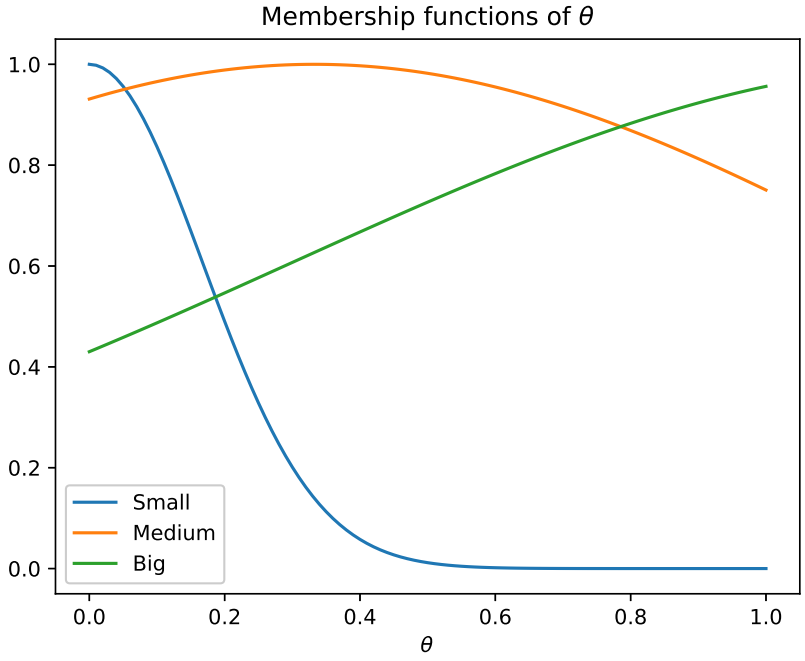

Figure 5.11: T1 MFs of $e(t_k)$ after training

5.2.4 Comparison of controlling performance

Comparison of transient response

In this test, the transient responses of three controllers starting from different initial positions are compared. The value of θ is sampled from range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$ with $\frac{10}{180}\pi$ step size, $\dot{\theta} = 0$. Tab. 5.9 illustrates the key properties in transient responses of inverted pendulum system under control of three controllers starting from different

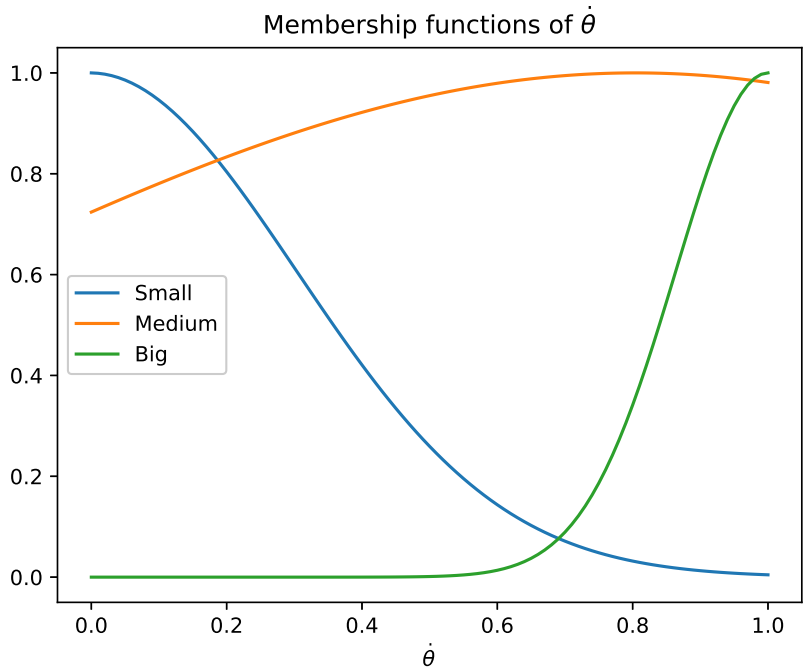

Figure 5.12: T1 MFs of $\Delta e(t_k)$ after training

Table 5.8: K_p and K_d of T1-FPD after training

	k_p	k_d
Rule1	69.2863	38.9836
Rule2	−794.1579	−278.8366
Rule3	−480.4393	−60.9055
Rule4	54.9232	43.7170
Rule5	−840.4654	−381.3812
Rule6	−607.4979	−169.1668
Rule7	98.0776	73.7952
Rule8	−57.6732	−153.7810
Rule9	−107.5450	−137.1337

initial positions. The key properties include rise time, settling time, overshoot, undershoot, maximum force, minimum force and steady-state error. For more intuitive presentation, $x = [\theta, \dot{\theta}] = [\frac{60}{180}\pi, 0]$ is chosen as one illustration point which is considered as the most challenging position within the sampling range. The transient response curves of three controllers are presented in Fig. 5.13 and the forces applied to the inverted pendulum are shown in Fig. 5.14. Fig. 5.15 compares the SAE of three controllers. All three controllers can balance the inverted pendulum systems without steady-state errors. T1-FPD controller has the largest overshoot and undershoot among three controllers as well as the longest settling time. Neuro-PD controller has relatively smaller oscillations but still takes relatively longer period to drive back to stable point. IT2-FPD controller shows the best transient performance with least oscillations and shortest settling time.

Table 5.9: Comparison of transient properties

Properties \ θ_0		$\frac{-60}{180}\pi$	$\frac{-50}{180}\pi$	$\frac{-40}{180}\pi$	$\frac{-30}{180}\pi$	$\frac{-20}{180}\pi$	$\frac{-10}{180}\pi$	$\frac{0}{180}\pi$	$\frac{10}{180}\pi$	$\frac{20}{180}\pi$	$\frac{30}{180}\pi$	$\frac{40}{180}\pi$	$\frac{50}{180}\pi$	$\frac{60}{180}\pi$
Rise time (s)	IT2-FPD	0.23	0.22	0.22	0.24	0.28	0.28	0.00	0.28	0.28	0.24	0.22	0.22	0.23
	T1-FPD	0.21	0.19	0.19	0.19	0.20	0.23	0.00	0.23	0.20	0.19	0.19	0.19	0.21
	neuro-PD	0.72	0.59	0.52	0.49	0.47	0.46	0.00	0.46	0.47	0.49	0.52	0.59	0.72
Settling time (s)	IT2-FPD	0.91	0.87	0.83	0.80	0.75	0.35	0.00	0.35	0.75	0.80	0.83	0.87	0.91
	T1-FPD	2.39	2.34	2.28	1.86	1.81	1.31	0.00	1.31	1.81	1.86	2.28	2.34	2.39
	neuro-PD	2.03	1.72	1.58	1.48	1.37	1.10	0.00	1.10	1.37	1.48	1.58	1.72	2.03
Overshoot	IT2-FPD	0.20	0.18	0.15	0.11	0.09	0.09	0.00	0.09	0.09	0.11	0.15	0.18	0.20
	T1-FPD	0.41	0.44	0.46	0.46	0.44	0.41	0.00	0.41	0.44	0.46	0.46	0.44	0.41
	neuro-PD	0.09	0.10	0.11	0.12	0.12	0.13	0.00	0.13	0.12	0.12	0.11	0.10	0.09
Undershoot	IT2-FPD	0.02	0.02	0.02	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.02	0.02	0.02
	T1-FPD	0.18	0.19	0.20	0.19	0.18	0.16	0.00	0.16	0.18	0.19	0.20	0.19	0.18
	neuro-PD	0.01	0.01	0.01	0.01	0.02	0.02	0.00	0.02	0.02	0.01	0.01	0.01	0.01
Max force (N)	IT2-FPD	133.50	103.91	69.14	36.44	19.06	9.38	0.00	61.72	111.98	167.21	231.78	294.43	371.27
	T1-FPD	170.87	146.44	115.33	81.07	48.07	21.44	0.00	51.69	121.46	207.38	291.70	368.37	400.00
	neuro-PD	19.77	18.80	16.58	13.31	9.28	4.75	0.00	31.53	63.07	94.60	126.14	157.67	189.21
Min force (N)	IT2-FPD	−371.27	−294.43	−231.78	−167.21	−111.98	−61.72	0.00	−9.38	−19.06	−36.44	−69.14	−103.91	−133.50
	T1-FPD	−400.00	−368.37	−291.70	-207.38	−121.46	−51.69	0.00	−21.44	−48.07	−81.07	−115.33	−146.44	−170.87
	neuro-PD	−189.21	-157.67	−126.14	−94.60	−63.07	−31.53	0.00	−4.75	−9.28	-13.31	−16.58	−18.80	−19.77
Steady state error	IT2_FPD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	T1_FPD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Neuro_PD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

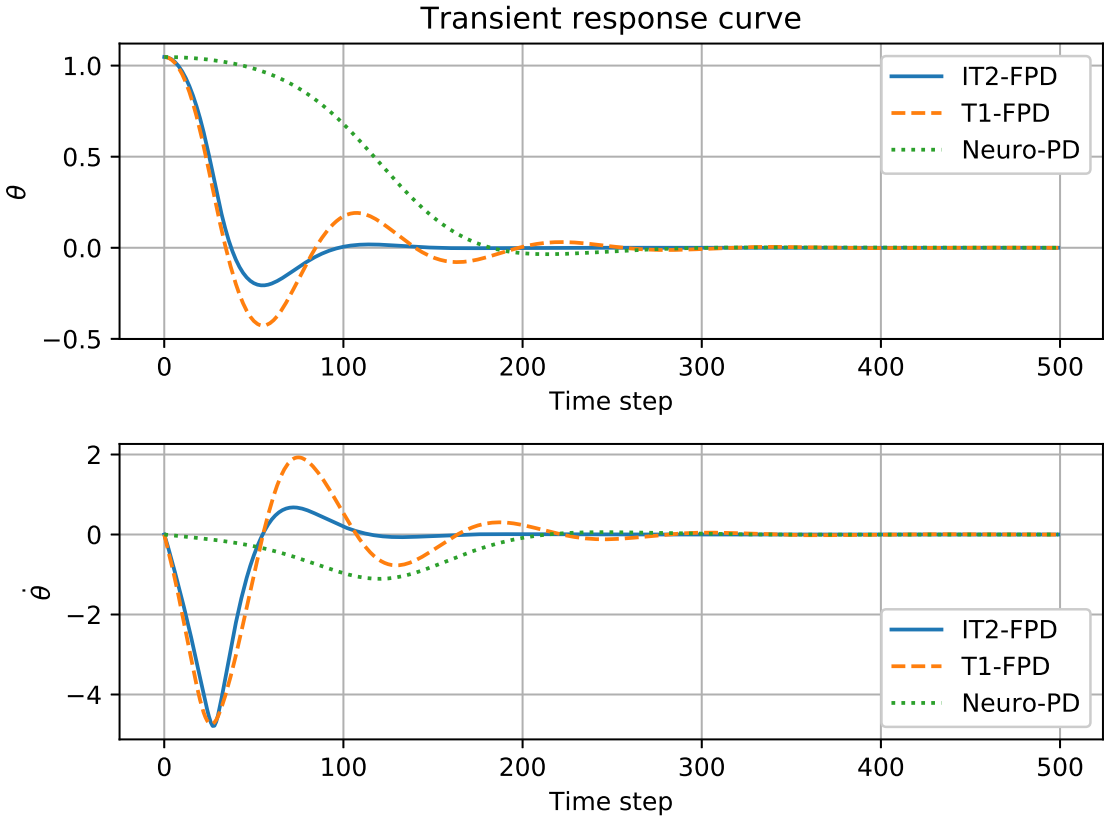

Figure 5.13: Comparison of transient response curves of three controllers after training with initial position: $\theta = \frac{60}{180}\pi$, $\dot{\theta} = 0$. (The upper figure presents the change of θ against time step while the lower figure presents the change of $\dot{\theta}$ against time step. In each figure, the blue solid line indicates the IT2-FPD controller, the orange dashed line indicates the T1-FPD controller and the green dotted line indicates the neuro-PD controller)

Random initial position test

As to verify the robustness of the controllers after training, the random initial position test is conducted by choosing different initial positions, where θ is sampled

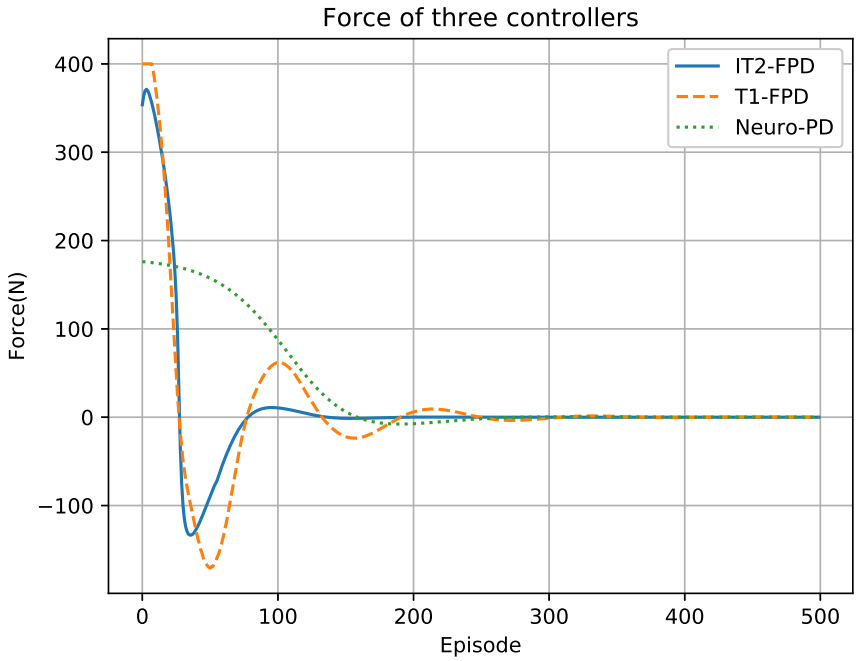

Figure 5.14: Compare of control signals of three controllers after training (initial position: $\theta = \frac{60}{180}\pi$, $\dot{\theta} = 0$). The blue solid line indicates the IT2-FPD controller, the orange dashed line indicates the T1-FPD controller and the green dotted line indicates the neuro-PD controller

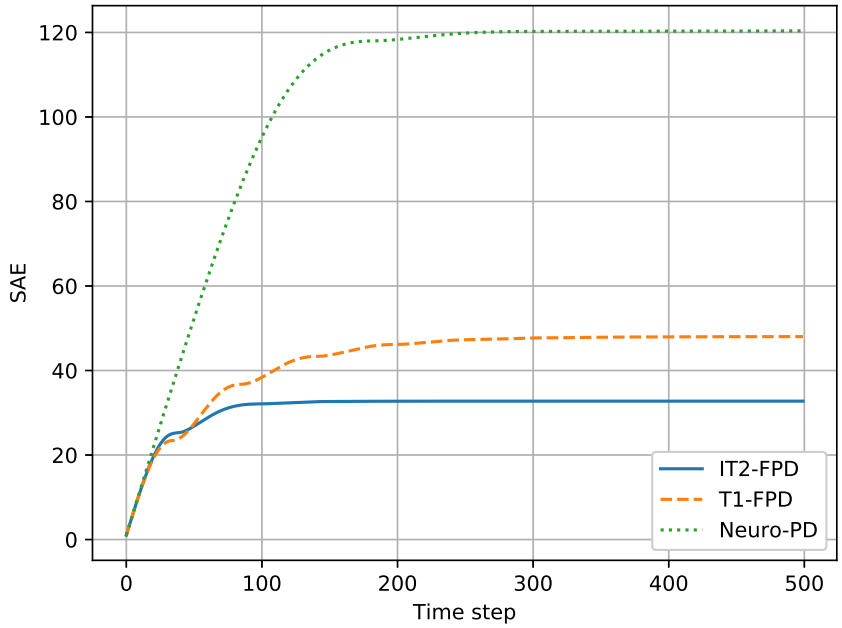

Figure 5.15: Compare of SAE of three controllers after training (initial position: $\theta = \frac{60}{180}\pi$, $\dot{\theta} = 0$). The blue solid line indicates the IT2-FPD controller, the orange dashed line indicates the T1-FPD controller and the green dotted line indicates the neuro-PD controller

from range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$ with $\frac{10}{180}\pi$ step size, $\dot{\theta} = 0$ and plotting phase portraits to check the number of initial positions each control is able to balance. The phase portrait of three controllers are presented in Fig. 5.16, Fig. 5.17 and Fig. 5.18, correspondingly. All three controllers are able to balance the system with initial angle position variants from $-\frac{60}{180}\pi$ to $\frac{60}{180}\pi$.

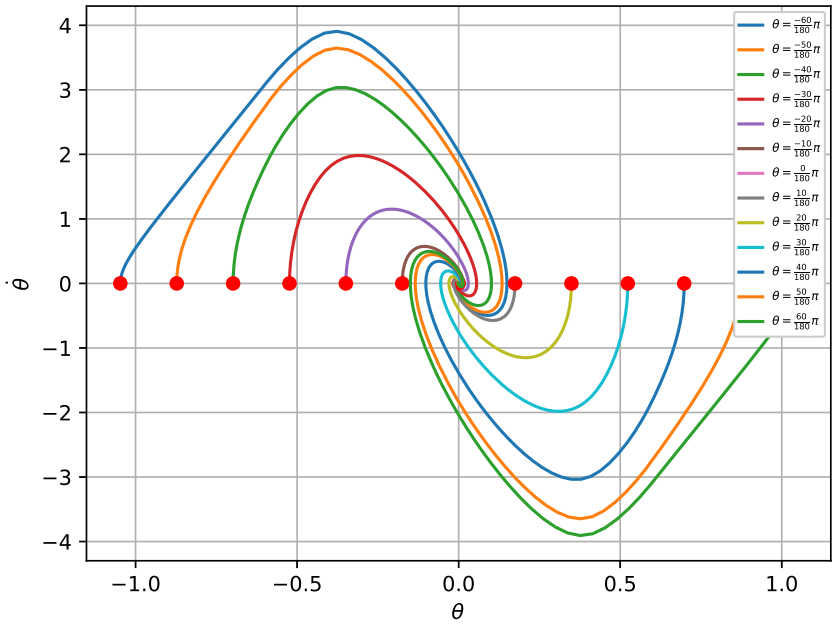

Figure 5.16: Phase portrait of IT2-FPD controller with different θ (θ is sampled from range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$ with $\frac{10}{180}\pi$ step size, $\dot{\theta} = 0$)

Robustness test

In real world applications, the system is not in ideal environments, the parameters of the system can variant because of the measurement error, external disturbance, etc. The controller is required to tolerance the parameters changes as to provide robust controlling performance. In the following test, different values of parameter m and l are tested under control of three different controllers trained by the approached AC algorithm to prove the robustness.

1) *Change in mass m*

In this test, the mass m is reduced or increased with 10% variation of m_0, where m_0 is the nominal mass of the inverted pendulum system. The limit mass value is confirmed when the controller starts to fail balancing the system back to equilibrium point ($\theta = 0, \dot{\theta} = 0$). Fig. 5.19, Fig. 5.20 and Fig. 5.21 present the phase portraits of inverted pendulum system under control of three controllers. It could be noticed that IT2-FPD controller is able to balance the system within

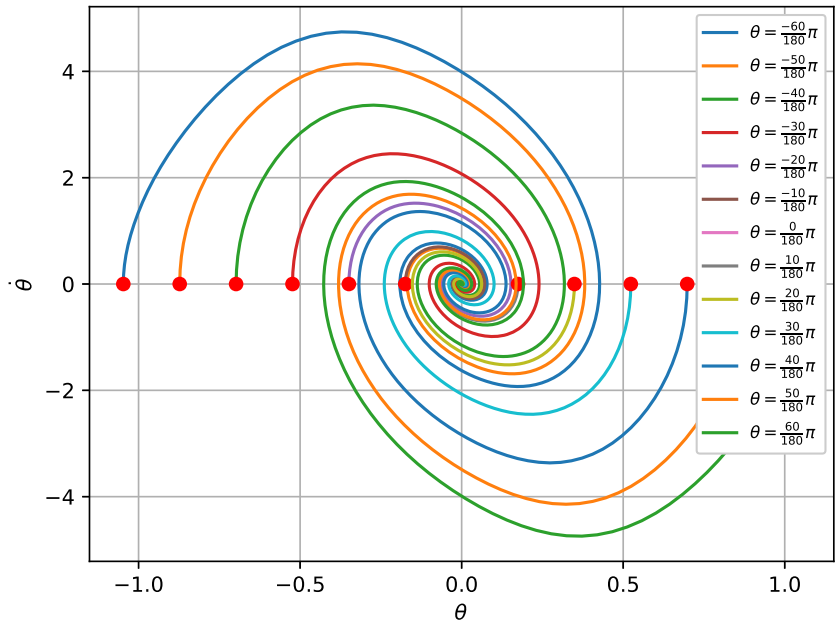

Figure 5.17: Phase portrait of T1-PD controller with different θ (θ is sampled from range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$ with $\frac{10}{180}\pi$ step size, $\dot{\theta} = 0$)

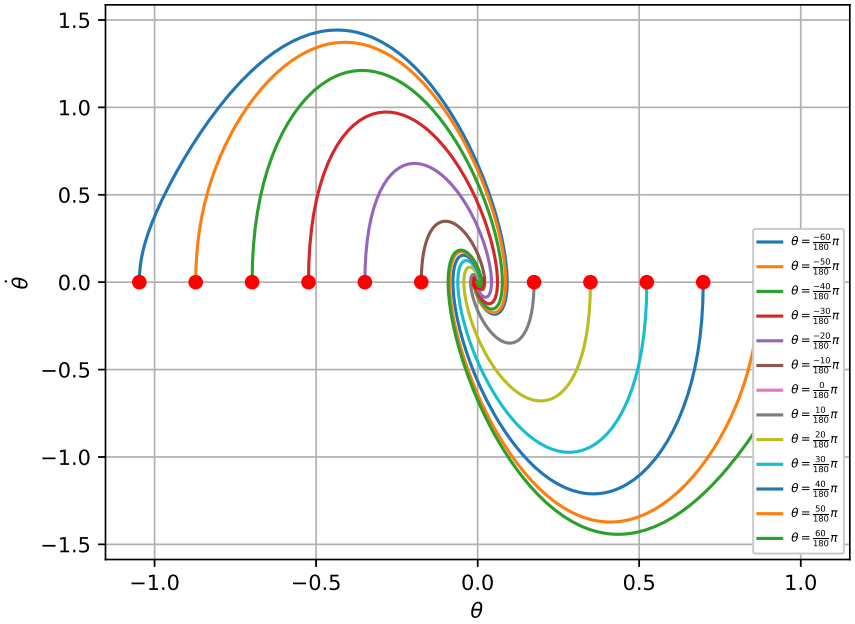

Figure 5.18: Phase portrait of neuro-PD controller with different θ (θ is sampled from range $[-\frac{60}{180}\pi, \frac{60}{180}\pi]$ with $\frac{10}{180}\pi$ step size, $\dot{\theta} = 0$)

range $m \in [0.1m_0, 1.1m_0]$, T1-FPD controller is within range $m \in [0.1m_0, 2.1m_0]$ and neuro-PD controller with $m \in [0.1m_0, 1.1m_0]$.

2) *Change in length of pole l*

Similar to the test with the robustness test in mass m, l is reduced or increased

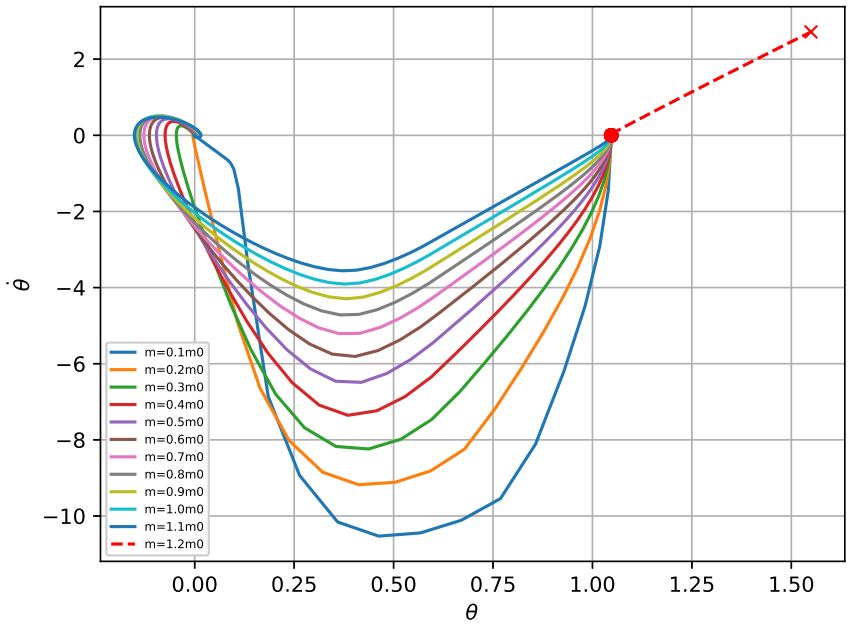

Figure 5.19: Phase portrait of IT2-FPD controller with different m values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max m values while the solid lines show the successful cases.

with 10% variation of l_0, where l_0 is the nominal length of the pole in inverted pendulum system. The limit minimum mass value is confirmed as the controller starts to fail balancing the system. However in the simulation, it is found that the controller has high tolerance with the large value of l. The controller is able to balance the system with large length value but it takes long period to drive back to stable state, which is not ideal in real-world circumstances. Therefore, in this paper, we chose the error band as $\pm 2\%$, with the increasement of l, the controller failing to drive the system to error band within 500 time steps is regarded as the upper bound of the l. The phase portraits are presented in Fig. 5.22, Fig. 5.23 and Fig. 5.24. The trained IT2-FPD controller is able to balance the system with l changing within $[0.1l_0, 4.5l_0]$, T1-FPD controller is able to balance the system within range $[0.1l_0, 2.0l_0]$ and linear PD controller is $[0.1l_0, 2.4l_0]$. The data of robustness test of three controllers are summarized in Table 5.10.

To conclude, IT2-FPD controller has better learning efficiency in the RL training process as well as better transient performance after training. In the robustness test, all three controllers are able to balance the inverted pendulum system from rang $\theta \in [\frac{-60}{180}\pi, \frac{60}{180}\pi]$. T1-FPD has better tolerance with the change of m value while IT2-FPD is more robust against the change of l.

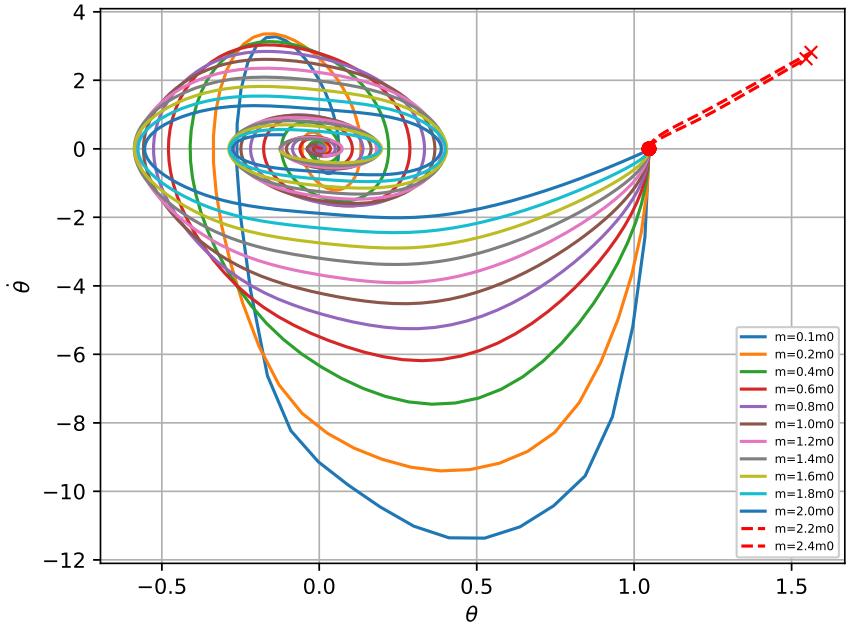

Figure 5.20: Phase portrait of T1-FPD controller with different m values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max m values while the solid lines show the successful cases (the system is tested with m varies with step of $0.1m_0$, the successful cases are not fully to avoid dense data presentation, the successful cases within range $l \in [0.1m_0, 2.4m_0]$ are only presented every $0.2m_0$, the cases omitted between each $0.2m_0$ are taken as successful cases for default).

Table 5.10: Comparison of three controllers in robustness test

	Neuro-PD	T1-FPD	IT2-FPD
θ	$[\frac{-60}{180}\pi, \frac{60}{180}\pi]$	$[\frac{-60}{180}\pi, \frac{60}{180}\pi]$	$[\frac{-60}{180}\pi, \frac{60}{180}\pi]$
m	[0.1,1.1]	[0.1,2.1]	[0.1,1.1]
l	[0.1,2.4]	[0.1,2.0]	[0.1,4.5]

Noise Test

The noise test is implemented to investigate the performance of proposed IT2-FPD controller in balancing the inverted pendulum system compared with other two types of controllers in noisy environment. In this test, a clipped Gaussian noise $\boldsymbol{n}$ of with amplitude gradually decreased is added to the state vector, which is defined as $\boldsymbol{s_n} = \boldsymbol{s} + \lambda \boldsymbol{n}$. The noise is chosen as $\boldsymbol{n} = [n_\theta, n_{\dot{\theta}}]$, where $n_\theta \sim \text{clip}(\mathcal{N}(0, 0.1), -0.1, 0.1)$, $n_{\dot{\theta}} \sim \text{clip}(\mathcal{N}(0, 0.01), -0.01, 0.01)$; the decay factor $\lambda = \max(1 - \frac{t_k}{250}, 0)$, where $0 < k < 500$ is the index of the time step. The initial position of the training episode is set as $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The response curves of three controllers are shown in Figure 5.25. According to comparison results, the neuro-PID fails to balance the inverted pendulum and the episode is terminated at time step 66. Though T1-FPD controller shows the trending of converge, obvious oscillations are observed until

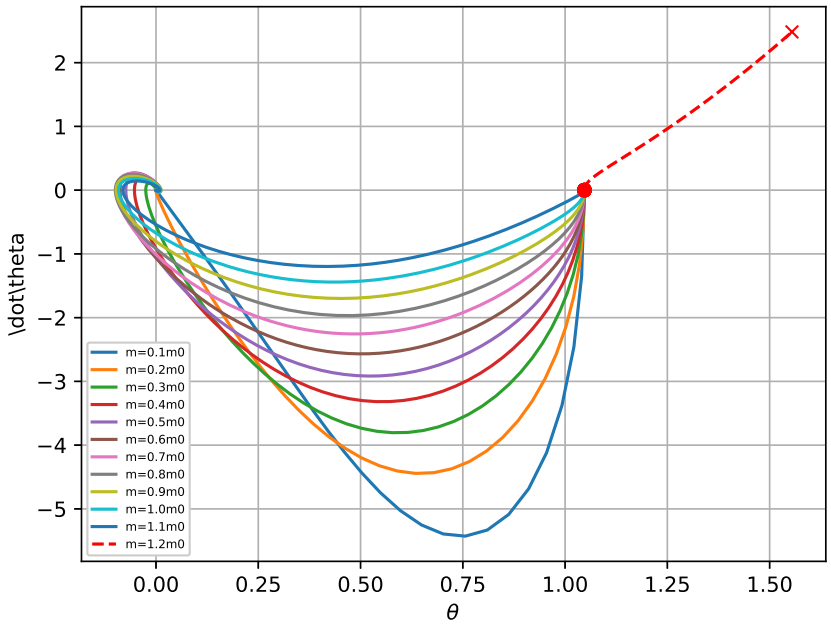

Figure 5.21: Phase portrait of neuro-PD controller with different m values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max m values while the solid lines show the successful cases.

the end of episode. IT2-FPD controller has relevant stable transient response when state noise exists and rapidly converges to steady state once the noise disappears. Therefore, the noise test indicates the advantage of IT2-FPD controller in robustness against the noise disturbances.

5.3 Conclusion

In summary, in this paper, an innovative AC algorithm with IT2-FPD controller as actor approximator is proposed. The update rules for the proposed AC algorithm is derived and the details of training procedures are discussed. Two other controllers are applied as actor approximator for comparison purpose, which are neuro-PD controller and T1-FPD controller. The update rule of T1-FPD controller is introduced as a special case of IT2-FPD controller when the UMF equals LMF. There are four experiments carried out to assess the performance of the controllers after training, which are the comparison of transient response, random initial position test, robustness test and noise test. All the tests are based on the inverted pendulum platform. In the transient test, different categories of controllers are required to balance the inverted pendulum system starting from initial position $\theta = \frac{60}{180}\pi$, $\dot{\theta} = 0$. The transient response curves is expected to have less oscillations and settling time therefore to achieve less accumulated error. IT2-FPD controller has relative small oscillations

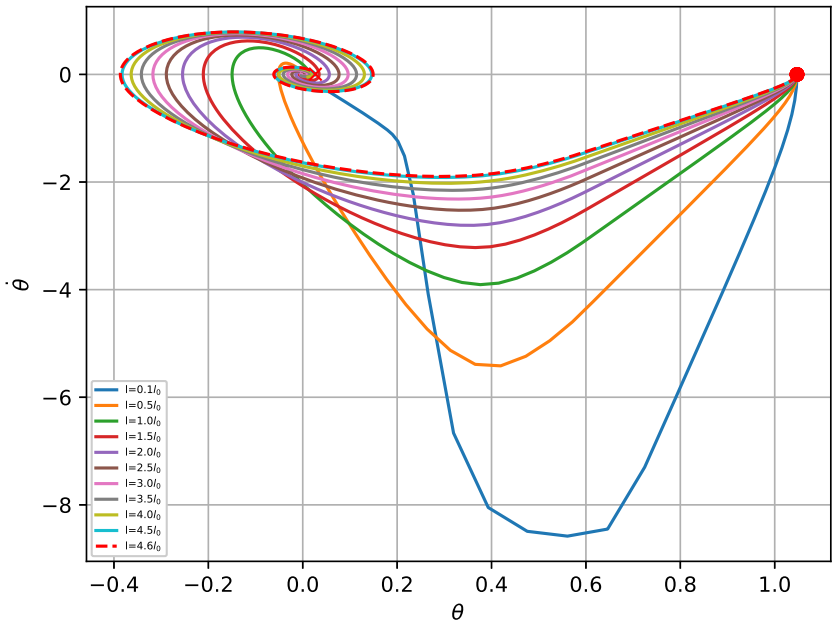

Figure 5.22: Phase plot of IT2-FPD controller with different l values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max l values while the solid lines show the successful cases (l varies with step of $0.1l_0$, the successful cases are not fully to avoid dense data presentation, the successful cases within range $l \in [0.1l_0, 4.5l_0]$ are only presented every $0.5l_0$, the cases omitted between each $0.5l_0$ are taken as successful cases for default).

with the fastest converge speed in stabilizing the inverted pendulum to steady state, of which accumulated error is the smallest. In the random initial position test, IT2-FPD controller successfully solve all the tasks from various initial positions. For robustness test, the mass value m and the length l in the system are adjusted with the fixed step size as to test the limit parameter value under control by different types of controller. The wider range of parameter values indicates the better robustness of the corresponding controllers. According to the data presented in Table 5.10, IT2 FPD controller shows advantage in against the change of l with the range $[0.1l_0, 4.5l_0]$ while T1 FPD is more robust to the change of m. Furthermore, in the noise test, when random Gaussian noise is added to the state vector, IT2-FPD shows the most stable transient response with the disturbance of noise and has the fastest speed in converging to stable state after the noise disappears. Additionally, from the training perspective, IT2-FPD controller uses least time to converge and stabilized to the desired accumulated reward value, which indicates the advantage in training efficiency. Therefore, the simulation experiment results indicate the superiority of IT2 FPD compared with neuro-PD controller and T1-FPD controller.

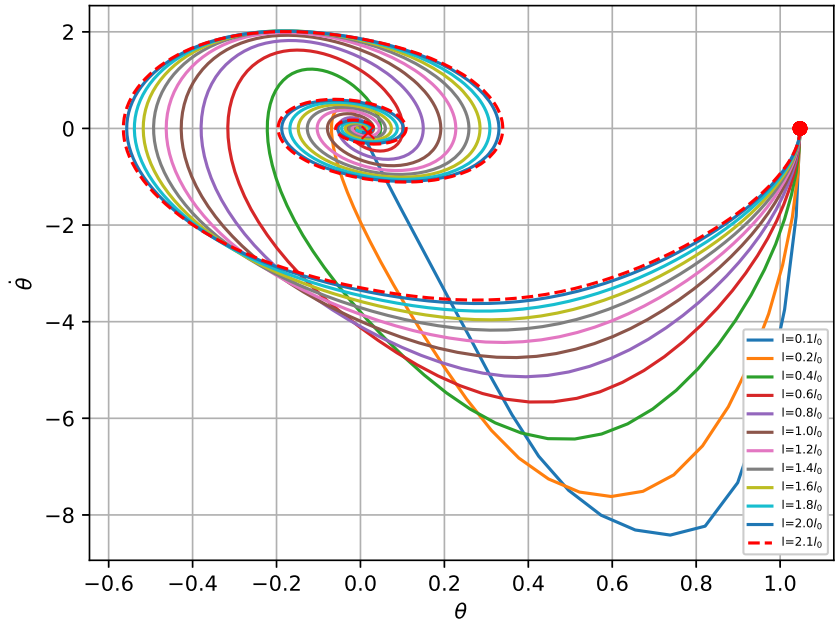

Figure 5.23: Phase plot of T1-FPD controller with different l values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max l values while the solid lines show the successful cases (Cases with certain l values are omitted in the figure for illustration purpose).

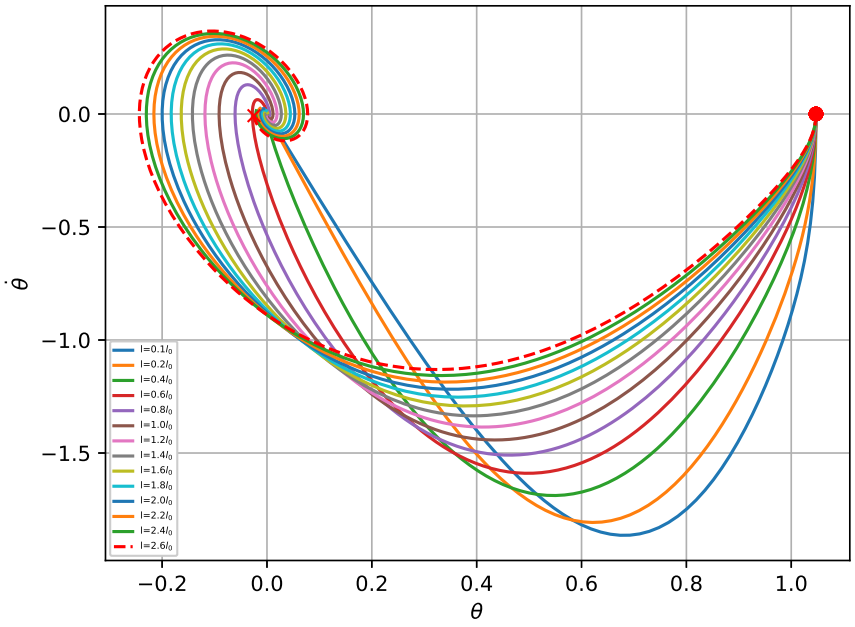

Figure 5.24: Phase plot of neuro-PD controller with different l values in robustness test. The initial position for robustness test is $\theta = \frac{60}{180}\pi, \dot{\theta} = 0$. The red dashed lines indicate the failure cases with limit min/max m values while the solid lines show the successful cases. The red dashed lines indicate the failure cases with limit min/max l values while the solid lines show the successful cases (l varies with step of $0.1l_0$, the successful cases are not fully to avoid dense data presentation, the successful cases within range $l \in [0.1l_0, 4.5l_0]$ are only presented every $0.2l_0$, the cases omitted between each $0.5l_0$ are taken as successful cases for default).

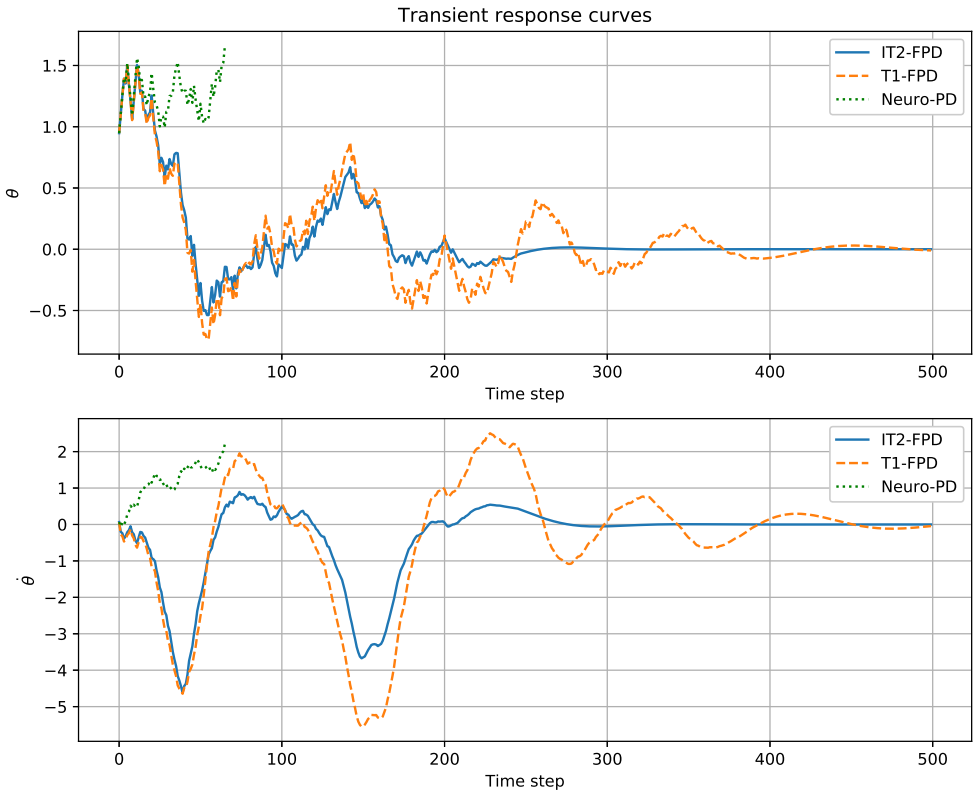

Figure 5.25: Comparison of transient response curves of three controllers with state noise (initial position: $\theta = \frac{60}{180}\pi$, $\dot{\theta} = 0$. The upper figure presents the change of θ against time step while the lower figure presents the change of $\dot{\theta}$ against time step. In each figure, the blue solid line indicates the IT2-FPD controller, the orange dashed line indicates the T1-FPD controller and the green dotted line indicates the neuro-PD controller)

Chapter 6

Conclusions and Future Work Plans

6.1 Conclusions

In this thesis, as to follow the aim of developing control system based on RL algorithms defined in the Section 1.1, three innovative control strategies have been proposed covering PID controller and FLSs. The proposed methods release the design difficulties existed in traditional methods for systems with high nonlinearity and uncertainty as well as improve the efficiency in parameter optimization. The proposed controllers reveal advantages in transient response properties and robustness. These research results are separately summarized and published as three journal papers, of which details are presented as follows.

In Chapter 3, an adaptive PID controller based on Q-learning algorithm is proposed. A set of linear PID controllers are utilized in the lower level for systems control, of which parameters are decided by a group of Q-tables in the upper level. The optimization is achieved through Q-learning algorithm with adaptive learning rate technique adopted for acceleration. The training results are tested on the 4th order inverted pendulum with two comparisons, linear PID controller and controller solely depends on Q-learning algorithm, which indicates the advantage of proposed QPID controller. The proposed approach provides an efficient optimizing method for multiple PID controllers. The adaptive PID controller based on Q-learning algorithm indicates enhanced dynamic performance and robustness, which achieves the first objective set in the thesis.

In Chapter 4, an off-policy actor-critic algorithm with type-1 neuro-fuzzy PID controller as actor approximator is proposed. The specially designed type-1 neuro-fuzzy PID network is composed of two sections, where the neuro-fuzzy network with a fixed set of parameters embeds the fuzzy information to the input variables and the neuro-PID networks generated control signals based on various fuzzy inputs. The latter network performs as the actor approximator in the TD3 algorithm of which

coefficients are updated following the update rules derived in the RL algorithm. The proposed controller is compared with linear PID controller on the inverted pendulum platform, which shows the priority of the proposed controller in transient response and robustness. This method releases the optimization difficulty caused by the increase of tuning parameters existed in the FLS and further improves the controlling performance of type-1 fuzzy controller, which fulfills the second objective defined in the thesis.

The research of Chapter 5 extends the method which utilizes neuro-fuzzy network as actor approximator in off-policy actor-critic algorithm from type-1 to IT2 cases. Theoretical derivation of updating rules is provided while the controller after optimization is compared with type-1 fuzzy controller and linear controller, which shows advanced performance in transient response, robustness against noisy measurement and parameter uncertainties. Therefore, the proposed methodology provides a comprehensive updating strategy for IT2 FLS based on AC optimizing method, which is not fully addressed in the previous researches as well as enhance the transient performance and robustness in the control system. These achievements reach the expectation of the third objective in the thesis.

Based on the researches stated above, the objectives raised in this thesis can be regarded as all achieved and the control system based on RL algorithms is successfully developed. However, there are still limitations exposed in the research works proposed in this thesis which can be further improved.

- Regarding to the Q-learning based PID controller introduced in Chapter 3, though proper set of gains of PID controllers could be found using Q-learning algorithm as optimization approach, the QPID controllers still haven't reached the optimal performance. This can be caused by the discretization in Q-tables or the design of reward function, which can be improved in the following research. Besides, in the simulation environments, only cases with two PID controllers are tested, more general cases can be further explored in the future research.

- Regarding to the adaptive type-1 fuzzy PID controller developed based on TD3 algorithm proposed in Chapter 4, this type-1 FLS in neuro format is consisted of two main sections, which are neuro-fuzzy network and neuro-PID network. Though the first section infuses the expert knowledge in fuzzy rule formation to accelerate the training process, the parameters in the first section remain fixed during the training process and only the parameters in the section part, neuro-PID networks, are adjustable through the RL algorithm. This structure limits the performance of the RL based FLS in certain extent and can be further improved to make parameters in both antecedent and consequent part adjustable.

- Regarding to the adaptive IT2 fuzzy PD controller developed based on actor-critic reinforcement learning algorithm proposed in Chapter 5, one of main contributions of this research work is the flexible RL based training structure, which provides adjustment for the position and the shape of IT2 FMs as well as the PD gains of the fuzzy PD controller, with the corresponding updating rules provided. Nevertheless, this structure are restricted to the cases where the IT2 FMs are differentiable (IT2 Gaussian function in this thesis). Further research can be done to provide more universal structure which could be applied to IT2 fuzzy controllers despite of the type of IT2 MFs.

6.2 Future work plans

As stated above, though considerable results have been achieved based on the contents proposed in this thesis, there are still issues that can be further improved in the field of control systems based on RL theory.

1) Further improvements can be done based on the research work in Chapter 5 where the IT2 FLS acts as the actor approximator in an actor-critic algorithm. More generic optimization schemes could be explored, such as achieving dynamic organization of fuzzy rule number. The current research of IT2 FLS based on TD3 algorithm tends to be an exploration test to find the potential performance this RL-based fuzzy logic system could achieve but is not verified in any specific scenarios. This system could be tested on more various tasks and transferred to physical platforms as well. Besides, the research could put more focus on the training efficiency of RL-based FLS system, such as computational cost, converging speed and training stability etc.

2) Investigate the potential of applying asynchronous advantage actor-critic (A3C) [150] structure with FLS in solving problems in complex environments with high nonlinearity and uncertainties. The A3C structure provides parallel actors running on different threads, which increases the exploration as well as increases the stability of the training process. While the FLS is a system nonlinearly consisted of several local linear controllers defined in various fuzzy rules. This multi-actors pattern of A3C can be referenced in optimizing FLS, where the controller in each fuzzy rule can be regarded as a local worker in the A3C training scheme, therefore helps boost the training efficiency and increases the stability in the training process.

3) Deep reinforcement learning algorithm can be further extended to more general adaptive control cases. Classical adaptive control requires complete dynamic model of the system, which leads to limitations in providing satisfactory solutions to dynamics with uncertainties and changes in dynamics. DRL techniques can

be applied to the such system which already has a nominal adaptive controller, where the developed nominal controller guarantees the basic performance of the system while DRL could be trained online coping with noises and disturbances occured in the environment with more efficient learning speed. Furthermore, the Lyapunov theory can be infused in the lost function as guidance to provide the RL-based adaptive control system with stability guarantee.

Bibliography

[1] M. Glavic, "(Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives," *Annual Reviews in Control*, vol. 48, pp. 22–35, 2019.

[2] M. Nikolaou and P. Misra, "Linear control of nonlinear processes: recent developments and future directions," *Computers & Chemical Engineering*, vol. 27, no. 8-9, pp. 1043–1059, 2003.

[3] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Pearson Prentice Hall, 2008.

[4] I. Carlucho, M. De Paula, S. A. Villar, and G. G. Acosta, "Incremental Q-learning strategy for adaptive PID control of mobile robots," *Expert Systems with Applications*, vol. 80, pp. 183–199, 2017.

[5] M. F. Miranda and K. G. Vamvoudakis, "Online optimal auto-tuning of PID controllers for tracking in a special class of linear systems," in *American Control Conference (ACC), 2016*, pp. 5443–5448, IEEE, 2016.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. MIT press Cambridge, 1998.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] M. Shadi and M. Sargolzaei, "Application of reinforcement learning to improve control performance of plant," in *2008 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pp. 79–82, IEEE, 2008.

[10] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[11] L. A. Zadeh, "Information and control," *Fuzzy sets*, vol. 8, no. 3, pp. 338–353, 1965.

[12] L. A. Zadeh, "Fuzzy sets," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pp. 394–432, World Scientific, 1996.

[13] H. K. Lam, "A review on stability analysis of continuous-time fuzzy-model-based control systems: From membership-function-independent to membership-function-dependent analysis," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 390–408, 2018.

[14] H. K. Lam and F. H. F. Leung, *Stability Analysis of Fuzzy-model-based Control Systems*, vol. 264. Springer, 2011.

[15] H. K. Lam, "Polynomial fuzzy model-based control systems," *Stability Analysis and Control Synthesis Using Membership Function-Dependent Techniques. Cham: Springer-Verlag*, vol. 307, 2016.

[16] K. Hirota and W. Pedrycz, "Fuzzy computing for data mining," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1575–1600, 1999.

[17] W. Pedrycz, "Fuzzy set technology in knowledge discovery," *Fuzzy Sets and Systems*, vol. 98, no. 3, pp. 279–290, 1998.

[18] R. R. Yager, "Database discovery using fuzzy sets," *International Journal of Intelligent Systems*, vol. 11, no. 9, pp. 691–712, 1996.

[19] A. Naji and M. Ramdani, "Toward a better self-regulation: degree of certainty through fuzzy logic in a formative assessment," *AI & society*, vol. 31, no. 2, pp. 259–264, 2016.

[20] T. Sheehan and M. Gough, "A platform-independent fuzzy logic modeling framework for environmental decision support," *Ecological Informatics*, vol. 34, pp. 92–101, 2016.

[21] J. C. Bezdek, "Fuzzy models-what are they and why?," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 1–6, 1993.

[22] D. Filev and R. Yager, "Essentials of fuzzy modeling and control," *Sigart Bulletin*, vol. 6, no. 4, p. 22, 1994.

[23] A. T. Nguyen, T. Laurain, R. Palhares, J. Lauber, C. Sentouh, and J. C. Popieul, "LMI-based control synthesis of constrained Takagi-Sugeno fuzzy systems subject to L2 or L_∞ disturbances," *Neurocomputing*, vol. 207, pp. 793–804, 2016.

[24] U. Ekong, H. K. Lam, B. Xiao, G. Ouyang, H. Liu, K. Y. Chan, and S. H. Ling, "Classification of epilepsy seizure phase using interval type-2 fuzzy support vector machines," *Neurocomputing*, vol. 199, pp. 66–76, 2016.

[25] M. Korytkowski, L. Rutkowski, and R. Scherer, "Fast image classification by boosting fuzzy classifiers," *Information Sciences*, vol. 327, pp. 175–182, 2016.

[26] F. Mekanik, M. Imteaz, and A. Talei, "Seasonal rainfall forecasting by adaptive network-based fuzzy inference system (ANFIS) using large scale climate signals," *Climate Dynamics*, vol. 46, no. 9-10, pp. 3097–3111, 2016.

[27] F. Ye, L. Zhang, D. Zhang, H. Fujita, and Z. Gong, "A novel forecasting method based on multi-order fuzzy time series and technical analysis," *Information Sciences*, vol. 367, pp. 41–57, 2016.

[28] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of the Institution of Electrical Engineers*, vol. 121, pp. 1585–1588, IET, 1974.

[29] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[30] E. H. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *International Journal of Man-Machine Studies*, vol. 8, no. 6, pp. 669–678, 1976.

[31] G. Feng, S. Cao, N. Rees, and C. Cheng, "Analysis and design of model based fuzzy control systems," in *Proceedings of 6th International Fuzzy Systems Conference*, vol. 2, pp. 901–906, IEEE, 1997.

[32] F. H. F. Leung, L. K. Wong, P. K. S. Tam, and H. K. Lam, "Realization of analog fuzzy logic control for PWM boost converters," *Journal of Circuits, Systems, and Computers*, vol. 8, no. 03, pp. 411–419, 1998.

[33] G. Stojić, "Using fuzzy logic for evaluating the level of countries'(regions') economic development," *Panoeconomicus*, vol. 59, no. 3, pp. 293–310, 2012.

[34] R. M. Darbra, E. Eljarrat, and D. Barceló, "How to measure uncertainties in environmental risk assessment," *TrAC Trends in Analytical Chemistry*, vol. 27, no. 4, pp. 377–385, 2008.

[35] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 116–132, 1985.

[36] K. Tanaka and M. Sugeno, "Stability analysis and design of fuzzy control systems," *Fuzzy Sets and Systems*, vol. 45, no. 2, pp. 135–156, 1992.

[37] K. Tanaka, T. Ikeda, and H. O. Wang, "Robust stabilization of a class of uncertain nonlinear systems via fuzzy control: Quadratic stabilizability, H^∞ control theory, and linear matrix inequalities," *IEEE Transactions on Fuzzy systems*, vol. 4, no. 1, pp. 1–13, 1996.

[38] H. K. Lam, "Output-feedback sampled-data polynomial controller for nonlinear systems," *Automatica*, vol. 47, no. 11, pp. 2457–2461, 2011.

[39] H. K. Lam, "Polynomial fuzzy-model-based control systems: stability analysis via piecewise-linear membership functions," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 3, pp. 588–593, 2011.

[40] K. Tanaka, H. Yoshida, H. Ohtake, and H. O. Wang, "A sum-of-squares approach to modeling and control of nonlinear dynamical systems with polynomial fuzzy systems," *IEEE Transactions on Fuzzy systems*, vol. 17, no. 4, pp. 911–922, 2008.

[41] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 20–29, 2007.

[42] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[43] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, 2000.

[44] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[45] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, and F. Wu, "A highly parallel framework for hevc coding unit partitioning tree decision on many-core processors," *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573–576, 2014.

[46] R. Das, S. Sen, and U. Maulik, "A survey on fuzzy deep neural networks," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–25, 2020.

[47] Y. Wang, Z. Wu, and J. Zhang, "Damaged fingerprint classification by deep learning with fuzzy feature points," in *2016 9th International Congress on*

Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 280–285, IEEE, 2016.

[48] X. Zhang and Z. Xu, "Extension of topsis to multiple criteria decision making with pythagorean fuzzy sets," *International Journal of Intelligent Systems*, vol. 29, no. 12, pp. 1061–1078, 2014.

[49] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets and Systems*, vol. 90, no. 2, pp. 111–127, 1997.

[50] P. R. Tabrizi, A. Mansoor, J. J. Cerrolaza, J. Jago, and M. G. Linguraru, "Automatic kidney segmentation in 3d pediatric ultrasound images using deep neural networks and weighted fuzzy active shape model," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 1170–1173, IEEE, 2018.

[51] S. Zhou, Q. Chen, and X. Wang, "Fuzzy deep belief networks for semi-supervised sentiment classification," *Neurocomputing*, vol. 131, pp. 312–322, 2014.

[52] H. A. Chopade and M. Narvekar, "Hybrid auto text summarization using deep neural network and fuzzy logic system," in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, pp. 52–56, IEEE, 2017.

[53] K. F. Leung, F. H. F. Leung, H. K. Lam, and S. H. Ling, "Application of a modified neural fuzzy network and an improved genetic algorithm to speech recognition," *Neural Computing and Applications*, vol. 16, no. 4, pp. 419–431, 2007.

[54] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 1006–1012, 2016.

[55] S. Rajurkar and N. K. Verma, "Developing deep fuzzy network with Takagi Sugeno fuzzy inference system," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, IEEE, 2017.

[56] C. P. Chen, C. Y. Zhang, L. Chen, and M. Gan, "Fuzzy restricted boltzmann machine for the enhancement of deep learning," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2163–2173, 2015.

[57] S. Park, S. J. Lee, E. Weiss, and Y. Motai, "Intra- and inter-fractional variation prediction of lung tumors using fuzzy deep learning," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 4, pp. 1–12, 2016.

[58] G. Pan, L. Fu, and L. Thakali, "Development of a global road safety performance function using deep neural networks," *International Journal of Transportation Science and Technology*, vol. 6, no. 3, pp. 159–173, 2017.

[59] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, IEEE, 2017.

[60] W. Chen, J. An, R. Li, L. Fu, G. Xie, M. Z. A. Bhuiyan, and K. Li, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial–temporal data features," *Future Generation Computer Systems*, vol. 89, pp. 78–88, 2018.

[61] N. S. Shirwandkar and S. Kulkarni, "Extractive text summarization using deep learning," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–5, IEEE, 2018.

[62] R. Zhang, F. Shen, and J. Zhao, "A model with fuzzy granulation and deep belief networks for exchange rate forecasting," in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 366–373, IEEE, 2014.

[63] S. Riaz, A. Arshad, and L. Jiao, "A semi-supervised CNN with fuzzy rough C-mean for image classification," *IEEE Access*, vol. 7, pp. 49641–49652, 2019.

[64] B. Rahmat, E. Joelianto, I. K. E. Purnama, and M. Hery, "Vehicle license plate image segmentation system using cellular neural network optimized by adaptive fuzzy and neuro-fuzzy algorithms," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 11, no. 12, pp. 383–400, 2016.

[65] A. Y. Ng, H. J. Kim, M. I. Jordan, S. Sastry, and S. Ballianda, "Autonomous helicopter flight via reinforcement learning.," in *NIPS*, vol. 16, Citeseer, 2003.

[66] G. Reddy, J. Wong Ng, A. Celani, T. J. Sejnowski, and M. Vergassola, "Glider soaring via reinforcement learning in the field," *Nature*, vol. 562, no. 7726, pp. 236–239, 2018.

[67] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[68] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, *et al.*, "Vector-based navigation using grid-like representations in artificial agents," *Nature*, vol. 557, no. 7705, pp. 429–433, 2018.

[69] C. Liu and M. Tomizuka, "Algorithmic safety measures for intelligent industrial co-robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3095–3102, IEEE, 2016.

[70] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proc. of the IEEE International Conference on International Conference on Robotics and Automation*, vol. 2, pp. 1615–1620, IEEE, 2001.

[71] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396, IEEE, 2017.

[72] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[73] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[74] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng, "Towards end-to-end reinforcement learning of dialogue agents for information access," *arXiv preprint arXiv:1609.00777*, 2016.

[75] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv preprint arXiv:1606.01541*, 2016.

[76] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.

[77] F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu, "3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5678–5687, 2017.

[78] M. Devrim Kaba, M. Gokhan Uzunbas, and S. Nam Lim, "A reinforcement learning approach to the view planning problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6933–6941, 2017.

[79] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[80] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. Torr, "Playing doom with slam-augmented deep reinforcement learning," *arXiv preprint arXiv:1612.00380*, 2016.

[81] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[82] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.

[83] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2765–2771, IEEE, 2019.

[84] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 336–342, IEEE, 2015.

[85] G. Theocharous, P. S. Thomas, and M. Ghavamzadeh, "Personalized ad recommendation systems for life-time value optimization with guarantees," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[86] N. Jiang and L. Li, "Doubly robust off-policy value evaluation for reinforcement learning," in *International Conference on Machine Learning*, pp. 652–661, PMLR, 2016.

[87] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall, "Concurrent reinforcement learning from customer interactions," in *International Conference on Machine Learning*, pp. 924–932, PMLR, 2013.

[88] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3357–3364, IEEE, 2017.

[89] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine Learning Proceedings 1990*, pp. 216–224, Elsevier, 1990.

[90] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[91] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*, pp. 617–629, PMLR, 2018.

[92] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.

[93] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *International Conference on Machine Learning*, pp. 2829–2838, PMLR, 2016.

[94] J. Oh, S. Singh, and H. Lee, "Value prediction network," *arXiv preprint arXiv:1707.03497*, 2017.

[95] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[96] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, vol. 37. Citeseer, 1994.

[97] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[98] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[99] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1995–2003, PMLR, 2016.

[100] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

[101] F. Gomez and J. Schmidhuber, "Evolving modular fast-weight networks for control," in *International Conference on Artificial Neural Networks*, pp. 383–389, Springer, 2005.

[102] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "Evolving large-scale neural networks for vision-based reinforcement learning," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 1061–1068, 2013.

[103] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[104] S. M. Kakade, "A natural policy gradient," *Advances in Neural Information Processing Systems*, vol. 14, 2001.

[105] A. Lonza, *Reinforcement Learning Algorithms with Python: Learn, Understand, and Develop Smart Algorithms for Addressing AI Challenges*. Packt Publishing Ltd, 2019.

[106] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, PMLR, 2015.

[107] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[108] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[109] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[110] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[111] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[112] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[113] B. Fernandez Gauna, I. Ansoategui, I. Etxeberria Agiriano, and M. Graña, "Reinforcement learning of ball screw feed drive controllers," *Engineering Applications of Artificial Intelligence*, vol. 30, pp. 107–117, 2014.

[114] P. Ramanathan, K. K. Mangla, and S. Satpathy, "Smart controller for conical tank system using reinforcement learning algorithm," *Measurement*, vol. 116, pp. 422–428, 2018.

[115] M. Rahman, S. M. H. Rashid, and M. M. Hossain, "Implementation of Q learning and deep Q network for controlling a self balancing robot model," *Robotics and Biomimetics*, pp. 4–9, 2018.

[116] I. Carlucho, M. D. Paula, S. Wang, Y. Petillot, and G. G. Acosta, "Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, 2018.

[117] H. Wu, S. Song, K. You, and C. Wu, "Depth control of model-free AUVs via reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[118] M. N. Howell, G. P. Frost, T. J. Gordon, and Q. H. Wu, "Continuous action reinforcement learning applied to vehicle suspension control," *Mechatronics*, vol. 7, no. 3, pp. 263–276, 1997.

[119] S. M. A. Mohammadi, A. A. Gharaveisi, M. Mashinchi, and S. M. R. Rafiei, "New evolutionary methods for optimal design of PID controllers for AVR system," in *2009 IEEE Bucharest PowerTech*, pp. 1–8, June 2009.

[120] F. M. Pour and A. A. Gharaveisi, "Opposition-based discrete action reinforcement learning automata algorithm case study: Optimal design of a PID controller," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 21, no. 6, pp. 1603–1614, 2013.

[121] M. N. Howell and M. C. Best, "On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata," *Control Engineering Practice*, vol. 8, no. 2, pp. 147–154, 2000.

[122] S. Mohammadi, A. Gharaveisi, M. Mashinchi, and V. N. SM, "Development of a novel reinforcement learning automata method for optimum design of proportional integral derivative controller for nonlinear systems," in *Proceedings of the World Congress on Engineering*, vol. 3, 2008.

[123] M. Sedighizadeh and A. Rezazadeh, "Adaptive PID controller based on reinforcement learning for wind turbine control," in *Proceedings of World Academy of Science, Engineering and Technology*, vol. 27, pp. 257–262, 2008.

[124] A. Younesi and H. Shayeghi, "Q-learning based supervisory PID controller for damping frequency oscillations in a hybrid mini/micro-grid," *Iranian Journal of Electrical and Electronic Engineering*, vol. 15, no. 1, pp. 126–141, 2019.

[125] A. el Hakim, H. Hindersah, and E. Rijanto, "Application of reinforcement learning on self-tuning PID controller for soccer robot multi-agent system," in *2013 Joint International Conference on Rural Information Communication Technology and Electric-Vehicle Technology (rICT ICeV-T)*, pp. 1–6, Nov 2013.

[126] P. Wang, H. Li, and C. Chan, "Quadratic Q-network for learning continuous control for autonomous vehicles," *ArXiv*, vol. abs/1912.00074, 2019.

[127] A. D. Pambudi, T. Agustinah, and R. Effendi, "Reinforcement point and fuzzy input design of fuzzy q-learning for mobile robot navigation system," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, pp. 186–191, 2019.

[128] H. B. Duan, D. B. Wang, and X. F. Yu, "Novel approach to nonlinear PID parameter optimization using ant colony optimization algorithm," *Journal of Bionic Engineering*, vol. 3, no. 2, pp. 73–78, 2006.

[129] H. A. Varol and Z. Bingul, "A new PID tuning technique using ant algorithm," in *Proceedings of the 2004 American Control Conference*, vol. 3, pp. 2154–2159 vol.3, June 2004.

[130] H. Boubertakh, M. Tadjine, P. Y. Glorennec, and S. Labiod, "Tuning fuzzy PD and PI controllers using reinforcement learning," *ISA Transactions*, vol. 49, no. 4, pp. 543–551, 2010.

[131] M. Gheisarnejad, J. Boudjadar, and M. H. Khooban, "A new adaptive type-II fuzzy-based deep reinforcement learning control: Fuel cell air-feed sensors control," *IEEE Sensors Journal*, vol. 19, no. 20, pp. 9081–9089, 2019.

[132] M. Gheisarnejad, J. Boudjadar, and M. Khooban, "A new adaptive type-II fuzzy-based deep reinforcement learning control: Fuel cell air-feed sensors control," *IEEE Sensors Journal*, vol. 19, no. 20, pp. 9081–9089, 2019.

[133] M. H. Khooban and M. Gheisarnejad, "A novel deep reinforcement learning controller based type-II fuzzy system: Frequency regulation in microgrids," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[134] S. N. Givigi, H. M. Schwartz, and X. Lu, "A reinforcement learning adaptive fuzzy controller for differential games," *Journal of Intelligent and Robotic Systems*, vol. 59, no. 1, pp. 3–30, 2010.

[135] E. Camci and E. Kayacan, "Game of drones: UAV pursuit-evasion game with type-2 fuzzy logic controllers tuned by reinforcement learning," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 618–625, IEEE, 2016.

[136] A. A. Khater, A. M. El Nagar, M. El Bardini, and N. M. El Rabaie, "Online learning of an interval type-2 TSK fuzzy logic controller for nonlinear systems," *Journal of the Franklin Institute*, vol. 356, no. 16, pp. 9254–9285, 2019.

[137] X. Dai, C. K. Li, and A. B. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 285–293, 2005.

[138] T. Zhao, Y. Chen, S. Dian, R. Guo, and S. Li, "General type-2 fuzzy gain scheduling PID controller with application to power-line inspection robots," *International Journal of Fuzzy Systems*, vol. 22, no. 1, pp. 181–200, 2020.

[139] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[140] M. B. Begian, W. W. Melek, and J. M. Mendel, "Stability analysis of type-2 fuzzy systems," in *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pp. 947–953, IEEE, 2008.

[141] C. Li, J. Yi, and T. Wang, "Stability analysis of SIRMs based type-2 fuzzy logic control systems," in *International Conference on Fuzzy Systems*, pp. 1–7, IEEE, 2010.

[142] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2012.

[143] H. K. Lam and H. F. H. Leung, "Fuzzy controller with stability and performance rules for nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, no. 2, pp. 147–163, 2007.

[144] L. J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[145] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[146] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[147] K. J. Åström and T. Hägglund, *PID controllers: Theory, Design, and Tuning*, vol. 2. Instrument society of America Research Triangle Park, NC, 1995.

[148] A. A. Haghrah and S. Ghaemi, "PyIT2FLS: A new python toolkit for interval type 2 fuzzy logic systems," *arXiv preprint arXiv:1909.10051*, 2019.

[149] H. Hagras, "Toward human-understandable, explainable AI," *Computer*, vol. 51, no. 9, pp. 28–36, 2018.

[150] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, PMLR, 2016.