RESEARCH ON APPLICATION OF ARTIFICIAL NEURAL NETWORKS AND NETWORK PRUNING STRATEGIES TO TEMPERATURE CONTROL SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
SCIENCE AND ENGINEERING OF
GUNMA UNIVERSITY FOR THE
DEGREE OF ENGINEERING

Yuan Liu
Gunma University
August 2022

Abstract

Thermal process is usually viewed as a nonlinear, large lag and large inertia system. In general, it takes lots of time, patience and expense to achieve accurate and stable control of the thermal systems. Furthermore, for a multi-input multi-output, there exists strong mutual interference between different heating channels, the generated temperature differences will seriously affect the temperature control effect and the quality of the workpiece. All these factors make good accurate temperature control difficult, and traditional control methods are difficult to keep up with the increasing requirements of control performance. The optimized temperature control algorithm is required for effective temperature adjustment to overcome the nonlinearity, strong coupling, disturbances, etc. With the development of artificial intelligence technology in the industry, which has the ability to process amounts of data and complex patterns in them, the temperature control during industrial processes can also be moved from traditional control methods based on precise mathematical models or complex optimization algorithms to simpler real-time artificial intelligence-based control, to satisfy more stable and precise control performance, and then to realize higher quality and lower energy costs.

This thesis is aimed at improving the performance of temperature control in multi-input multi-output systems and effectively compress the pre-trained NN-based temperature model without the obvious control accuracy loss. The following solutions are mainly proposed:

(1) A multi-input multi-output temperature control system not only has the characteristics of nonlinear and large lag, but also has strong coupling and other uncertain factors. Commonly, multiple heaters are used to control the temperature of the controlled object. However, the output of each heater (heating channel) will affect the output of other channels. The temperature difference between different heating channels of the controlled object will negatively affect the quality of the products in industrial process. The design of the optimization control to such mutually interfering channel temperature of the controlled object is difficult and complicated. To address this problem, a multi-layer neural network control system is proposed for the multi-input multi-output system, which is driven by a reference model. It eliminates the temperature difference between

each pair of heating channels for achieving the uniform temperature of different heating channels, while improving the transient and steady-state control performances of coupling channels. The designed NN-based control system can receive real-time data and do self-tuning to give optimal control input to each channel without the need of precise system modeling and additional decoupling links. The effectiveness and reliability of the proposed control scheme are verified based on the simulation and experimental results.

(2) Deep network models are usually over-parameterized to provide stronger characterization and optimization capabilities. Even a network model that is not so big also needs a large amount of memory and hardware to perform computation, which results in slow inference speed and limits the deployment of models to embedded devices with small storage capacity and low computational power. Considering our pretrained NN control models, the network pruning technique is one of the effective model compression methods, which removes some unimportant parameters without affecting the initial model accuracy obviously. Therefore, inspired by the reconstruction error-based pruning method used in CNN models, which is based on minimizing the linear reconstruction error to optimize the network, a layer-wise iterative pruning method is adopted to prune our FNN-based and RNN-based temperature control model, by minimizing the nonlinear reconstruction error between the outputs of the pruned model and unpruned models. It is verified that the proposed method can eliminate a large number of redundant parameters of the well-trained FNN and RNN temperature control models without a significant loss of accuracy by experiments.

Contents

Chapter 1

Introduction

In order to ensure production efficiency, quality and safety in manufacturing and processing industries, temperature controllers play a crucial role in helping regulate and control the temperature of different industrial systems. A temperature controller which has characteristics of high speed, high performance and easy settings is always necessary to meet a wide range of application needs, especially in packaging machines, semiconductor production equipment, food processing, and many other areas[1, 2]. Commonly, the long-term poor control actions may eventually lead to quality deterioration, extensive downtime and extra costs in industries. It will be a huge loss both to the enterprise and to customers. Such as in the semiconductor industry, a state-of the-art production equipment may cost more than 1.5 billion. In some cases, any kind of downtime can even lead to losses of more than 100,000 dollars an hour. Therefore, the requirement for highly reliable temperature control performance cannot be emphasized enough. Furthermore, the geometry of integrated circuits becomes smaller and smaller with the expansion of the global market, most manufacturing firms including the semiconductor field are in urgent need of achieving more accurate and reliable temperature control[3–5].

Meanwhile, although most manufacturing processes require low cost automation systems and devices that can meet desired control targets, the establishment of systematic mechanism model and global mathematical model requires many experts and high-level researchers, and the cost is high, especially for different batches, different products and different cycles. It is hard to imagine modeling every batch, product, and cycle to improve product yield and quality, and not every system can be mathematically modeled[6, 7]. For many complex process systems, such as the research object of this thesis is the thermal process system with non-linearity and large dead time, due to the complexity of the system itself and various disturbances, it's hard to establish a global mathematical model of the system. Even if the partial model is obtained, it's difficult to ensure the accuracy, so model-based control methods are weak in solving practical

problems.

Actually, there are many factors will lead to temperature variations during the production process as Figure.1.1. As the processing speed increases, the precision and stability of the machine temperature are easily affected by the proportion of processing materials, the size and shape of the workpiece, etc. In most cases, experienced operators are required to repeatedly make appropriate adjustments to stabilize temperature changes and maintain production quality. Therefore, it is really difficult to achieve high speed production while also maintaining the quality. Correctly identifying and controlling the temperature of the processing machine is the key factor to ensure the smooth operation of the whole production process and prolong the service life of the machine.

As Industry 4.0 becomes the standard for various processing machines, equipment manufacturers are in urgent of updating their temperature controller for more "smart" features, making the production processes more efficient[8]. In the past few years, the digital intelligent controller has been widely used in various applications, along with the developed microcomputer processing and a variety of advanced algorithm arises for overcoming the control problems in complicated systems[9, 10].

Move temperature control during industrial process from traditional control methods based on precise mathematical models or complex optimization algorithm to simpler real-time artificial intelligence-based(AI-based) control to reach higher quality and reduce downtime, energy and costs.

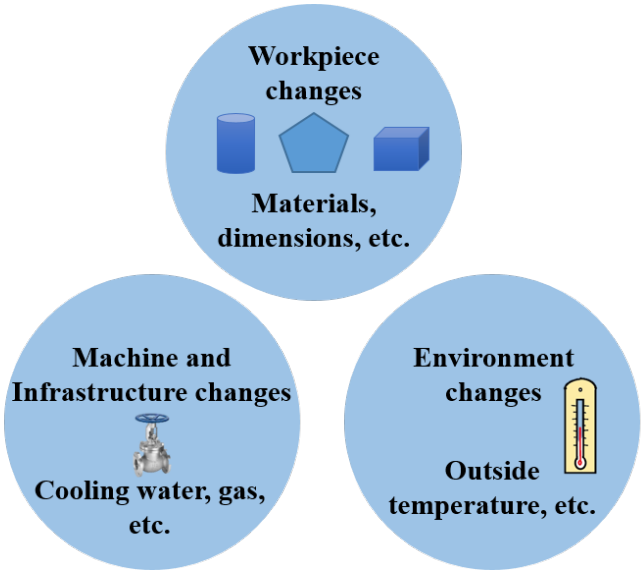


Figure 1.1: Various causes of temperature variations during production

1.1 Temperature Control Requirements

The ideal temperature control needs to consider meeting the following requirements. In practice, it is practically difficult to satisfy all of these conditions at once, and it is normal to control each content to the point where the control system can accept it.

(1) Rapid temperature increase: Reach the target temperature setting as soon as possible or within the set time, while reducing the overshoot;

(2) Disturbance suppression: Even if there is disturbance, it can immediately return to the set value with smaller temperature fluctuations.

(3) Temperature uniformity: Reduce the temperature differences of coupling points and ensure the surface temperature uniform.

Take the packing process as an example, once the products(disturbance) are placed on a packaging machine, the heater temperature will be decreased, thus resulting in heat sealing error and poor product quality. As shown in Figure.1.2, once the disturbance occurs, it will take a long time until the heater temperature is stabilized. The desired control performance refers to a much smaller drop in temperature by stabilizing the heater temperature immediately. However, the traditional control methods are difficult to achieve the ideal interference suppression effect[11].

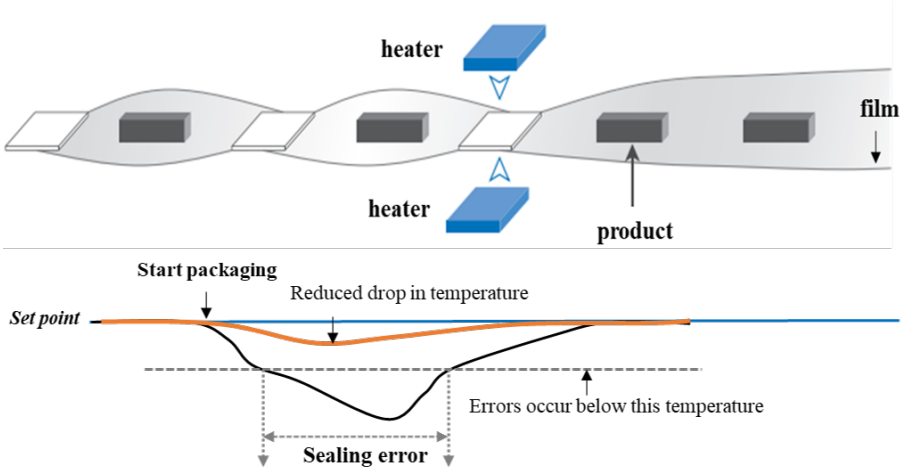


Figure 1.2: Disturbance effect in the industrial production process

1.1.1 Classical control strategies

Until now, PID(Proportional-Integral-Derivative) controllers are still the most widely used closed-loop feedback controller in the industrial process control. The advantages of PID control include simple structure, relatively good control performance, ease of use, etc. According to the error signal between the actual process output and the target value, they can adjust the control output automatically to maintain the output of process

variables (PV) at the target value. The common process variables include temperature, flow, pressure, etc. To ensure the quality of products and the lifespan of the equipment, any change in the setpoint or sudden disturbance should be handeled timely. A typical PID closed-loop control system can be illustrated as Figure 1.3.

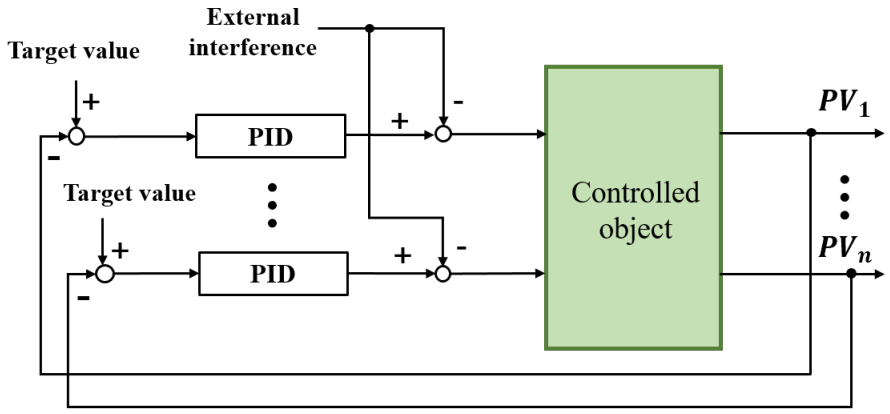


Figure 1.3: Block diagram of the closed-loop control system with PID controllers

In fact, although the concept of PID control is simple, it is often very difficulty to determine the controller parameters for achieving optimal control performance. In the actual production process, define the initial start-up PID settings and the corresponding adjustment to fluctuations for a standard PID controller always take a lot of time, and it is difficult to reach the optimal adjustment without the operator's rich experience. Instead of manually adjusting parameters of controllers, many parameter tuning methods have been proposed as: Ziegler-Nichols(ZN) tuning method, Cohen-Coon response curve method, Integral Square error(ISE)-based tuning method, etc[12, 13].

Today, most temperature process controllers are well equipped with auto-tuning functions, they automatically calculate PID parameters when there are changes in target value or a disturbance signal is added. As the industrial processes are becoming more and more complex, the controllers are required to meet the higher control performance demand under various working conditions. The main approaches of auto-tuning parameters for solving the nonlinear, imprecise uncertain problems occurred in practical industrial process control can be summarized as follows[14–17]:

(1) Adaptive PID control. It can not only adjust the parameters of the controller automatically, but also has strong adaptability to overcome the problem that process parameters and even model structure change caused by the noise, load disturbance or other factors.

(2) Fuzzy PID control. It is a kind of control method using fuzzy set theory. Especially in those large lag, time varying, nonlinear complex systems without relying on the accurate mathematical model of the system. It can achieve high performance of PID

control, but also has characteristics of simple structure, strong flexibility and robustness.

(3) Neural network-based PID control. It combines neural network which has the advantage of adaptive learning ability, strong real-time performance with the PID controller. Based on the designed network structure, it can achieve the flexible adjustment to the controller parameters and have obtained good dynamic performance in many studies.

(4) Predictive PID control. It uses nonparametric model predictive control algorithm. It has the advantages of simple structure, convenient parameter setting, and can predict the output signal of time-delay process.

Smith predictor control

To overcome the delay time of the controlled object, one of the common used strategies is Smith predictor control method[18]. The block diagram of the corresponding control architecture is as Figure 1.4.

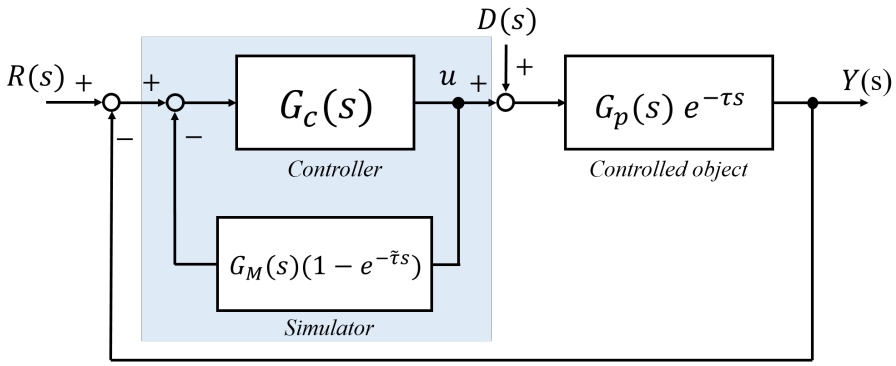


Figure 1.4: Block diagram of control system with smith predictor

Here, $G_p(s)$ represents the transfer function of the plant without delay term, which is stable and strict proper term and $e^{-\tau s}$ is prue delay term. $G_M(s)e^{-\tilde{\tau}s}$ is the internal model of the controlled object $G_p(s)e^{-\tau s}$, which is used for the prediction of the system behavior. Generally, the feedback controller G_c is PID or PI controller.

Assume that the predictor can match up with the actual model(without model error), which means $G_M(s) = G_p(s)$ and $\tilde{\tau} = \tau$. And there is no disturbance $D(s) = 0$, the transfer function of the smith predictor part is given as Equation 1.1. The closed-loop transfer function can be obtained as Equation 1.2.

$$G'(s) = \frac{G_c(s)}{1 + G_c(s)G_M(s)(1 - e^{-\tilde{\tau}s})} \tag{1.1}$$

$$\frac{Y(s)}{R(s)} = \frac{G'(s)G_p(s)e^{-\tau s}}{1 + G'(s)G_p(s)e^{-\tau s}} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}e^{-\tau s} \tag{1.2}$$

By introducing a pure delay term in reverse parallel with the controller, smith predictive control strategy makes the equivalent transfer function not include the pure delay term, so that the system without time delay can be easily controlled by conventional control methods.

Although it is useful to handle the problem of time-delay system theoretically, it has some defects in practical application: (1) It requires an accurate process model, and when the model changes, the quality of control will significantly deteriorate; (2) It is very sensitive to the change of parameters of the actual object. When the parameters change greatly, the closed-loop system will become unstable and even completely fail.

Gradient temperature control

In order to achieves the uniform temperature over a surface with multiple heaters by effective multi-point control, the gradient temperature method(GTC)[19] is one of effective strategies, which is a multi-loop PID control technology. The control block diagram is shown as Figure 1.5. It calculates the average temperature and the temperature differences between any two points by the mode converter. Meanwhile, the outputs of PID control are distributed by the precompensator to eliminate the effects of each control output to the other points. In this way, the interference between heaters of each point is reduced, thus the uniformity of the surface temperature is achieved.

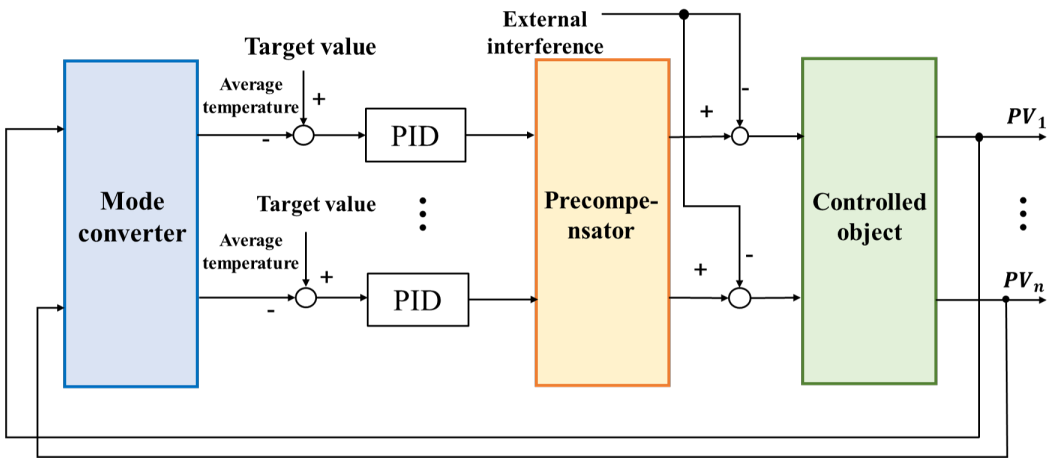


Figure 1.5: Control block diagram of gradient temperature control

This method is viewed as an uniform temperature control for multi-point control system, which not only increasing the dynamic performance of the thermal process, but also eliminating the interference between heaters. However, the design of multiple compensator and controllers is complex, including the structural mathematical analysis of the controlled object and it is still difficult to obtain the optimal temperature control effects automatically without the human intervention for the changes in the process.

1.1.2 Intelligent control strategies

With the rapid development of the computers and other modern science and technology, as well as the continuous expansion of production system scale form complex control systems, thus resulting in more complex control tasks and controller design. In addition, the requirements for the levels of automation become more extensive, considering the intelligent robot system, computer integrated manufacturing system and other complex systems, the classical and modern control theory and technology can not adapt to the complex system control. In past decades, intelligent control becomes more and more necessary in the production process[20, 21].

With the gradual development, intelligent control has been one of key advanced information and control technologies. It spans many disciplines, such as the information theory, artificial intelligence, neurophysiology, electrical engineering and computer science. If a system can continuously capture changes in the environment, obtain effective information to overcome the uncertainty, and a series of control behaviors can be effectively planned and performed, it is usually viewed as an intelligent control system. Actually, intelligent control can be viewed as is a kind of control technology using various artificial intelligence calculation methods. It represents a kind of advanced control mode that integrates intelligent information processing, feedback and decision making. In order to solve the problems that are difficult to be controlled well by the traditional methods, intelligent control has been one of best practicable technologies which reaches the advanced stage of this traditional control theory.

It is the advanced stage of the development of control theory, mainly used to solve the control problems of complex systems that are difficult to be solved by traditional methods[22, 23].

The main target of traditional control method is the controlled object, while that of intelligent control is the controller itself. Classical control is more based on the mathematical model analysis of the controlled object, but the intelligent control focuses on the establishment of intelligent controller model. The process involves the data acquisition, representation and storage, and the design of intelligent reasoning mode, etc.

On the other hand, the controlled object of intelligent control is quite different from the classical control, its characteristics are as: (1) No need to establish the mathematical model of the controlled object, which is especially suitable for solving problems with nonlinear object, time-varying object and complex uncertain control object; (2) Easy to process a large amount of information and stored data, and also including inference; (3) Own adaptive ability and good robustness; (4) Own the ability to learn and the ability can be increased gradually.

Based on the above, the differences between intelligent learning theory and traditional methods are simply compared as follows.

(1) Conventional PID control strategies: They can not deal with strong nonlinear, time-varying and existing disturbance system well, and does not have the learning function, does not have the adaptability to the system structure change.

(2) Intelligent control strategy: They focus on not the mathematical model analysis of the control object, but the establishment of intelligent controller model, including the acquisition, representation and storage of knowledge, the design of intelligent reasoning mode, etc. It can deal with strong nonlinear, time-varying and periodic systems well.

Intelligent control is quite different from the framework of traditional control theory which must be built based on the accurate mathematical/physical model in the traditional control. It is basically based on the actual control effect and does not depend on or completely depend on the mathematical model[24]. It enables to simulate the nonlinear characteristics of human thinking. Some intelligent control methods also have the ability of online identification, decision-making or overall self-optimization, as well as the function of hierarchical information processing and decision-making.

1.1.3 Artificial intelligence control

Artificial Intelligence As shown in Figure 1.6, artificial intelligence(AI), starts from studying and simulating different and complex human activities, and then gradually learns the rules of control and information of the process system, it is actually regarded as a system combining engineering control and information processing technologies. It is gradually developing towards achieving the same level of intelligence as the human brain[25]. It is a mixed disciplinary which is based on computer science, and developed from psychology, philosophy and other knowledge.

Machine Learning It is an AI sub field and consist of technique that can recognize date to trains programs, rather than explicitly writing a method to accomplish a specific task with specific instructions, and it can learn from past experiences. Here, training refers to provide the program with a lot of data and let the program automatically complete configuration and improving itself.

Deep Learning Deep learning(deep neural learning/deep neural network), is a sub field of machine learning, which is inspired by the structure and function of human brain and then form neural networks consist of large amounts of neurons[26]. These algorithms, which mimic the biological structure of human brain, are known as artificial neural networks (ANN). *"Deep"* is built by layers, meaning that the more layers a

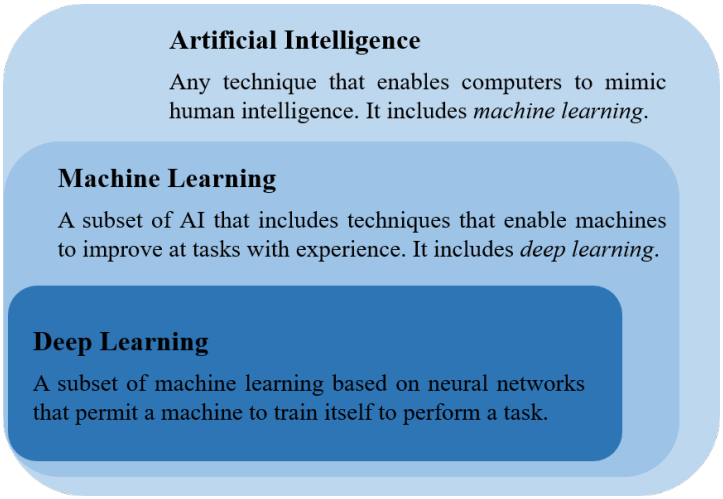


Figure 1.6: AI vs. ML vs. DL

network has, the deeper or complex it becomes.

As mentioned, deep learning is part of machine learning. However, machine learning is the process of giving data to a computer to learn features of a task by making repeated judgments. In other words, let the computer master the rules of the task, and then the computer automates the task[27]. Different from the general machine learning, the characteristics, classification and amount of data as learning materials are selected manually, while deep learning is done automatically by the computer, as shown in Figure.1.7.

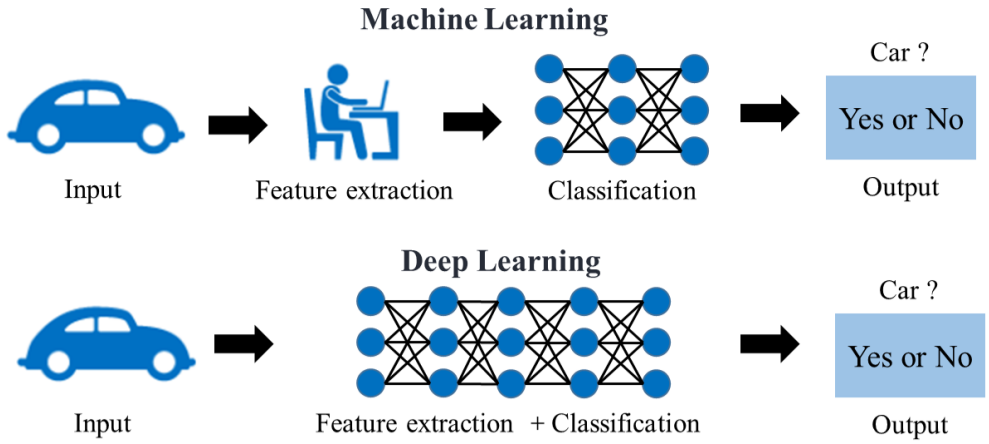


Figure 1.7: Comparison of machine learning and deep learning

In modern advanced manufacturing system, it is necessary to use incomplete or inaccurate data to solve the difficult or unpredictable situation. Artificial intelligence technology provides some effective solutions to solve this problem, such as follows[28–30]:

(1) The fuzzy neural network method adopted to model the complex dynamic environment of different manufacturing processes, and combine multi-sensor data fusion technologies to preprocess and realize an integrated process of the effective information.

(2) The expert system is adopted as a feedback mechanism, relying on apriori knowledge to modify the control mechanism or select better control modes and parameters.

(3) The fuzzy set decision is adopted to select more appropriate control actions. It can help in increasing the accuracy of the decisions in uncertain environment.

(4) The neural network is adopted for its strong learning and parallel information processing abilities. It has realized the online pattern recognition and processing.

Artificial intelligence-based(AI-based) controllers have special characteristics include the ability of processing non-traditional input data(visual information about the equipment or product state), the ability of processing and feedback to control in real time after data monitoring and the ability of continuous learning from huge amounts of raw data with different features. Today, all adjustments typically made by experts in the field can be automated adjusted by using artificial intelligence algorithms. In terms of the operation quality of the current artificial intelligence control system, the neural network control system plays a greater role in the industrial automation control system because of its advantages of fast data analysis and higher control accuracy.

In view of the above problems related to temperature control that cannot be solved by traditional control methods in the production process, it is a very important and valuable research direction to combine neural network learning and control theory methods to design controllers directly based on system input and output data.

1.2 Thesis Outline

Based on the above all, this thesis focus on how to improve the control performance of nonlinear, larger delay, strong coupling temperature systems by utilizing the neural network-based artificial intelligence technology, achieving optimal and automatic temperature control.

This thesis mainly consists of six chapters, and it is organized as follows. In Chapter 1, first introduce the background of temperature control, review the current application of classical control and intelligent control methods in temperature control and the existing problems and difficulties. And then it leads to the advantages of neural network in the application of temperature control and the research content of this thesis.

In Chapter 2, the design of the controlled objects and the establishment of experiment setup are introduced, including the single-input single-output and multi-input multi-output temperature control systems. The controlled systems are modelled by system identification and expressed in transfer functions, respectively. They are used for simulating and experimental verification of the designed temperature control methods in the following chapters.

From Chapter 3 to Chapter 4, the main work of this thesis is introduced in detail. Specifically, Chapter 3 introduces the basic knowledge involved in this chapter, mainly related to neural networks. And then propose a multi-layer FNN-based control method for multi-input multi-output uniform temperature, and an ideal reference model is introduced for guiding the optimization of the neural networks. The neural network self-learns and adjusts the output to achieve the decoupling effect of mutual interference. Simulation and experiments are performed and the results are quantitatively analyzed to demonstrate the efficiency and feasibility of the proposed control systems. In chapter 4, the over-parameterization problem of the neural network model is considered. Based on the pre-trained neural network model, an efficient network pruning method based on the optimization objective of layer-wise nonlinear reconstruction error is proposed to eliminate redundant parameters in the temperature control model based on FNN and RNN. The temperature control system based on RNN model is proposed to overcome the problem that the traditional FNN can not effectively utilize and extract useful information from time-series data, for further improving the performance of the temperature control. After detailed design, a reasonable and feasible control system structure is given. The effectiveness of the proposed RNN-based control method is shown through experiments.

Finally, Chapter 5 makes a summary of the results obtained in this thesis, and pointing out the inadequacies in studies and briefly describes the future research directions.

Chapter 2

Modeling Method for Thermal Process

Generally, the design and analysis of the control system requires the design of the controller according to the characteristics of the controlled object, in order to obtain the optimal control system that meets the requirements of performance indicators. One of the main purposes of analyzing and studying the dynamic system is to obtain the optimal setting parameters of the controller, but in the actual process, most of the control objects are relatively complex, and for the consideration of safety, economy and feasibility of experiments, it is necessary to carry out experimental research by replacing the actual system with a model (physical model or mathematical model)[31, 32]. Such as in the study of guided flight, aerospace and reactor control systems, or the high-temperature processing systems, without simulation experiments at first, it will not only affect the production, but also bring great danger to human life and health.

In this chapter, the background on the modeling method of the thermal process is detailed introduced and the system identification experiments are performed on the Single-Input Single-Output temperature control system and Multi-input Multi-output temperature control system, respectively. The model parameters obtained by system identification will be used to select the controller parameters in the comparison experiment in later chapters.

2.1 Modeling Method Based on System Identification

Commonly, there are three modeling methods for temperature control system research: (1) mechanism modeling (white box), (2) identification modeling (black box) and (3) gray box modeling. Mechanism modeling is based on the analysis of heat transfer and the establishment of temperature model through the energy balance. Using mechanism model can clearly simulate the nature of temperature change of the control system, this means that the users know all the details about how the system works. Hence, the

measurement of unknown parameters will waste lots of manpower and wealth. On the opposite, identification modeling which is based on measurement, deriving the mathematical models that describe system behavior from input-output data of the controlled system. A system model entirely derived from experimental data (input and output data) is called a black box model[33–35].

In the production process, many used modeling method is to regard the process as a black box, according to the input and output data of the process, through the method of system identification to establish a mathematical model. However, more commonly, mechanism modeling and data-driven identification modeling are combined for better model estimation. As some of the physical laws and model structure of our temperature control system are known, such as the step response characteristics of the temperature control system match the first-order plus dead-time mathematical model. The specific and simple structure can be obtained in advanced as above, the next is deriving these unknown numerical values of the model parameters from data. In this chapter, in order to describe the behavior of the controlled temperature system, system identification based on the time input and output signals is performed.

2.1.1 Dynamic characteristics of thermal process systems

Commonly, a thermal process can be simplified by a First Order Plus Dead Time (FOPDT) model[36] which is derived from the following differential Equation 2.1 and the step response curve of it is as shown in Figure 2.1. Dead time or time delay refers to a shift effect of the input variable $u(t)$ for the output of the dynamic system response.

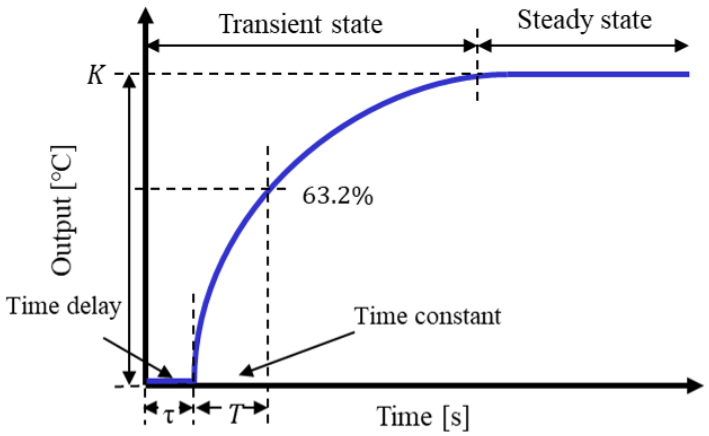


Figure 2.1: Step response of a FOPDT system

$$T\frac{\mathrm{d}y(t)}{\mathrm{d}t} = -y(t) + Ku(t-\tau) \tag{2.1}$$

where the unknown parameters are steady-state process gain (K), overall process time constant (T) and process dead time (τ), respectively. As the step response curve shown below, the process time constant (T) is the time used for the system output reaches 63.2% of the steady state and it reflects the response speed of the system response.

The transfer function in the Laplace domain can be written as Equation 2.2. The common process variables models are listed in Table 2.1. The difficulty of such control systems usually depends on the ratio of τ to T. The larger the ratio, the harder it is to control the system. Commonly, the ratio value τ/T in the temperature process is within the rang of 0∼0.5, and even in some special process systems with extremely slow response speed, the ratio may exceed 1.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{Ke^{-\tau s}}{Ts+1} \tag{2.2}$$

Table 2.1: Representative models of different process variables

Process	Transfer function	Gain	Time Constant T[s]	Dead Time τ[s]
Temperature	$\frac{K}{Ts+1}e^{-\tau s}$	0.1∼1.0	1∼3000 120∼12000	0∼30 0∼600
Pressure	$\frac{K}{Ts+1}e^{-\tau s}$	0.5∼5	60∼6000	0∼600
Flow	$\frac{K}{Ts+1}$	1∼4	6∼18	0∼12
Level	$\frac{K}{Ts}e^{-\tau s}$	-	120∼1200	0∼180

Time delay effect The pure delay term of the controlled object brings damage to the control performance of the control system, which reduces the stability and deteriorates the dynamic characteristics of the system[37]. Especially when the ratio of the pure delay time (τ) to the time constant (T) of the object is greater than 0.3 (called large delay process), it is difficult to obtain satisfactory control performance by conventional control methods. The magnitude of the time delay $e^{-\tau s}$ in the frequency domain can also be written in $e^{-jw\tau}$ as Equation 2.3, where the operator $s = jw$. If the magnitude and phase of a pure time delay in the frequency domain are plotted, the magnitude of it is independent of frequency which is just constant as Equation 2.4.

$$e^{-jw\tau} = cos(w\tau) - jsin(w\tau) \tag{2.3}$$

$$\begin{aligned} \left|e^{-jw\tau}\right| &= |cos(w\tau) - jsin(w\tau)| \\ &= \sqrt{cos^2(w\tau) + sin^2(w\tau)} = 1 \end{aligned} \tag{2.4}$$

The phase shift produced by a time delay decreases linearly with a slope of $-w\tau$ as given in Equation 2.5, where τ is the length of the time delay. The delay terms only change the timing of output. The magnitude is constant at 0dB, and the phase decreases exponentially. The delay term makes the phase margin decrease and when the phase margin becomes too small or negative.

$$\begin{aligned} \angle e^{-jw\tau} &= tan^{-1}(\frac{-sin(w\tau)}{cos(w\tau)}) \\ &= -tan^{-1}(tan(w\tau)) = -w\tau \end{aligned} \tag{2.5}$$

2.1.2 Overview of system identification

For model-based method control system, it is very important to build a good model that enables to represent the original system characteristics perfectly. The models describing the system behavior can be mainly divided into the following two types: parametric model and non-parametric model[38, 39].

- Parametric models characterize the dynamic characteristics of a system with a finite number of parameters.
- Non-parametric models can not use a finite number of parameters to describe the system and are usually expressed as response curves or discrete values.

Therefore, the parametric model identification method estimates the unknown parameters in the given model structure (transfer function, state equation or difference equation) by minimizing the criterion function which calculates the error between the estimation model and actual models. The process of estimating parameters is in the way of numerical searching, and finally it can obtain the mathematical models of a dynamic system, such as a transfer function based on the measured data to describe the behavior of a linear system. And there are many well-established models have been successfully used in linear or nonlinear system identification. On the opposite, the non-parametric method is modeling the system directly with an impulse response (correlation analysis) or frequency response (spectral analysis). It doesn't rely on given parameterized models to estimate the system. A typical technique is that the neural network models the dynamic system based on the black box identification method.

In order to implement our temperature control systems in the following chapters, the transfer function which can represent the thermal dynamics of our controlled objects is derived. Here, the parametric system identification method is considered to fit a transfer function based on the measured input-output data.

2.1.3 Basic process description

System identification is a data-driven method, the first step is to set up an experiment or a test to collect that data from the real system. The general process of mathematical simulation which is generally called computer simulation can be summarized as follows:

Step 1: Determine the system identification conditions;

Step 2: Collect the input and output signals from the control system in time or frequency domain and preprocessing them;

Step 3: Select a right model structure;

Step 4: For these adjustable unknown parameters of the model, apply an estimation method to get them;

Step 5: Analyze and evaluate the estimated model to ensure the model is accurate enough to meet your control system requirements.

It begins with prior information about the control system to be identified, then modifying some choices as needed and working through the various sub problems described above in turn, this process is performed iteratively until the model validation meets the criteria. In addition, the selected estimation model needs to reproduce measured signals while the structure is required as simple as possible. Based on the above, the general process of modeling dynamic systems can be summarized in Figure 2.2.

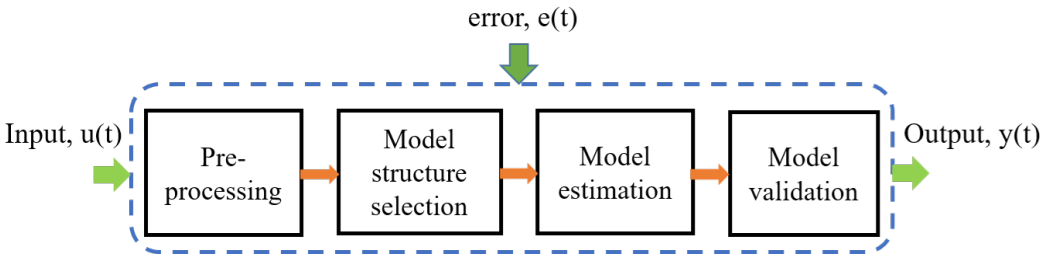


Figure 2.2: Flow chart of building models of dynamic systems

2.1.4 Parameter estimation method based on ARX model

(1) ARX Model Structure

One of parametric models for black box modeling is Auto-Regressive with eXogenous(ARX) model[40], which parameters can be estimated using the least-squares method. Commonly, the difference equation of the process be written as the following n-th order function:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_n u(t-n) \qquad (2.6)$$

The ARX polynomial model in Figure 2.3 can be described as the following Equation 2.7, including the model input $u(t)$, the model output $y(t)$ at time t, the added white noise $w(t)$ and n_i is the i-th input delay.

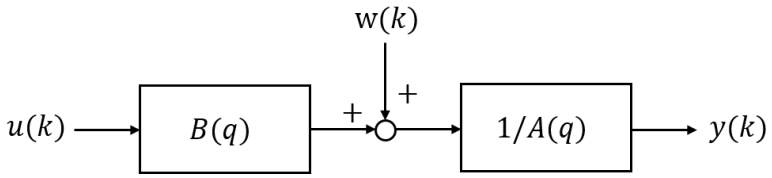


Figure 2.3: ARX model structure

$$A(q)y(t) = B(q)u(t - n_i) + w(t) \tag{2.7}$$

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \\ B(q) &= b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b} \end{aligned} \tag{2.8}$$

Both variables $A(q)$ and $B(q)$ are polynomials by the time-shift factor q^{-1} ($q^{-1}u(k) = u(k-1)$), which can be defined by the following equations, respectively. The variables a_i and b_j are the estimated model parameters, where $i = 1, \cdots, n_a$ and $j = 1, \cdots, n_b$, respectively. The n_a is the number of poles, representing the order of the observed state. n_b is the number of zeros, representing the model order of the control signal.

Rewrite the Equation.2.6 with error as:

$$\begin{aligned} y(t) &= -a_1 y(t-1) - \cdots - a_n y(t-n) \\ &\quad + b_1 u(t-1) + \cdots + b_n u(t-n) + w(t) \\ &= \boldsymbol{\varphi}(t)\boldsymbol{\theta} + w(t) \end{aligned} \tag{2.9}$$

where the parameter vector $\boldsymbol{\theta}$ of the current model and data vector $\boldsymbol{\varphi}(t)$ are defined as:

$$\boldsymbol{\theta} = [a_1, \cdots, a_{n_a}, b1, \cdots, b_{n_b}]^T \in \mathbb{R}^{(n_a+n_b)} \tag{2.10}$$

$$\boldsymbol{\varphi}(t) = [-y(k-1), \cdots, -y(t-n_a), u(t-1), \cdots, u(t-n_b)]^T \in \mathbb{R}^{(n_a+n_b)} \tag{2.11}$$

Then, the output $y(t)$ can be written as:

$$y(t) = \boldsymbol{\varphi}^T(t)\theta + w(t) \tag{2.12}$$

The transfer function of the process and noise model are respectively given by:

$$G(q,\theta) = \frac{B(q)}{A(a)}, H(q,\theta) = \frac{1}{A(q)} \tag{2.13}$$

(2) Parameter Estimation

For the ARX model, least squares(LS) algorithm which is a special case of the prediction error method (PEM) is used to estimate the model parameters[41]. It is the most efficient polynomial estimation method because this method solves linear regression equations in analytic form. Moreover, the solution is unique. The general process of parameter estimation is as Figure 2.4.

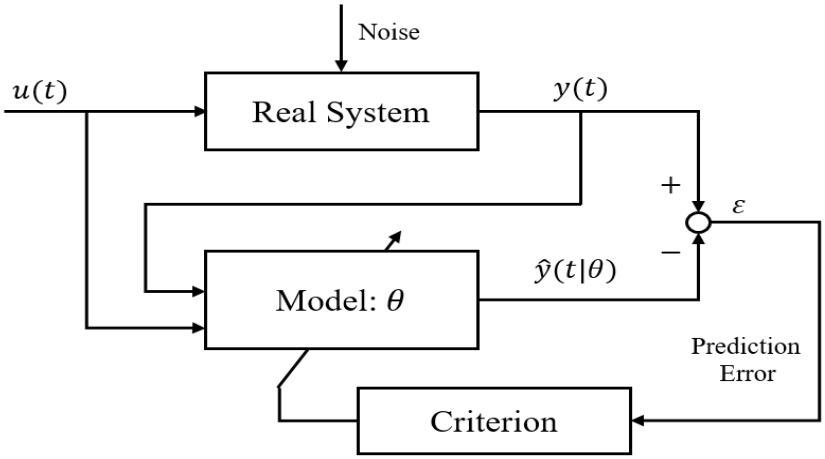


Figure 2.4: System identification based on the prediction error method

As mentioned above, the prediction output $\hat{y}$ which uses all the past information and established models with parameter vector $\boldsymbol{\theta}$ is described as:

$$\begin{aligned} \hat{y}(k|\boldsymbol{\theta}) &= [1 - A(q)]y(k) + B(q)u(k) \\ &= \boldsymbol{\theta}^T \boldsymbol{\varphi}(k) \end{aligned} \tag{2.14}$$

And then the error ε between the model output $\hat{y}(k)$ and the actual measured output $y(k)$ is calculated as Equation 2.15.

$$\varepsilon(k, \boldsymbol{\theta}) = y(k) - \hat{y}(k|\boldsymbol{\theta}) = y(k) - \boldsymbol{\theta}^T \boldsymbol{\varphi}(k) \tag{2.15}$$

The prediction error method estimates the parameter vector of the current model by optimizing the loss function J_N as:

$$J_N(\boldsymbol{\theta}) = \frac{1}{N} \sum_{k=1}^{N} l(k, \boldsymbol{\theta}, \varepsilon(k, \boldsymbol{\theta})) \tag{2.16}$$

$$\hat{\boldsymbol{\theta}}(N) = \arg\min_{\boldsymbol{\theta}} J_N(\boldsymbol{\theta}) \tag{2.17}$$

where $\boldsymbol{\theta}$ is unknown parameter, $l(k, \boldsymbol{\theta}, \varepsilon(k, \theta))$ is prediction error

$$\begin{aligned} J_N(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k,\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N}\{y(k)-\hat{y}(k|\boldsymbol{\theta})\}^2 \\ &= \boldsymbol{\theta}^T\boldsymbol{R}(N)\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\boldsymbol{f}(N) + c(N) \end{aligned} \tag{2.18}$$

and $\boldsymbol{R}(N)$, $\boldsymbol{f}(N)$ and $c(N)$ are given by:

$$\begin{aligned} \boldsymbol{R}(N) &= \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\varphi}(k)\boldsymbol{\varphi}(k)^T(k) \\ \boldsymbol{f}(N) &= \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\varphi}(k)y(k) \\ c(N) &= \frac{1}{N}\sum_{k=1}^{N}y^2(k) \end{aligned} \tag{2.19}$$

Let the differential of the cost function J at θ equal 0 as Equation 2.20., then least squares estimation of unknown parameters based on N input and output data can be derived by:

$$\frac{d}{d\boldsymbol{\theta}}J_N(\boldsymbol{\theta}) = 2\boldsymbol{R}(N)\boldsymbol{\theta}(N) - 2\boldsymbol{f}(N) = 0 \tag{2.20}$$

$$\hat{\boldsymbol{\theta}}(N) = \boldsymbol{R}(N)^{-1}\boldsymbol{f}(N) \tag{2.21}$$

(3) Model Validation

In order to evaluate the accuracy of the identified model, error between the actual output $y(k)$ and the estimated model output $\hat{y}(k)$, the percentage of fit (Fit(%)) as the following expression, where $\overline{y}(k)$ is the mean of the actual output. The higher the fit value, the higher the model identification accuracy[42].

$$Fit(\%) = (1 - \frac{\sqrt{\sum_{k=1}^{N}[\hat{y}(k)-y(k)]^2}}{\sqrt{\sum_{k=1}^{N}[y(k)-\overline{y}(k)]^2}}) \times 100\% \tag{2.22}$$

$$\overline{y}(k) = \frac{1}{N}\sum_{k=1}^{N}y(k) \tag{2.23}$$

2.2 System Identification Experiments

In terms of temperature control, the delay time of the process control system can reach 10 to 100 seconds. If the experiment time is too short, sometimes the system

cannot reach the stable state, so in order to obtain the data of the process from the initial input excitation to the steady state, it is usually necessary to set significant experiment time. Ensure the measured data capture the complete dynamics of the control system, the time of the open-loop identification experiment needs to be set much longer than the settling time of the system. In addition, apply appropriate sampling time or frequency resolution to capture data is also important for an accurate estimation model.

In this part, specific experimental setup of our temperature control system and experiment process of system identification are described.

2.2.1 Experimental setup

The overall platform for identifying the Single-Input Single-Output(SISO) system and Multiple-Input Multiple-Output(MIMO) system used in the following chapters is setup and detailed explanation is given as follows.

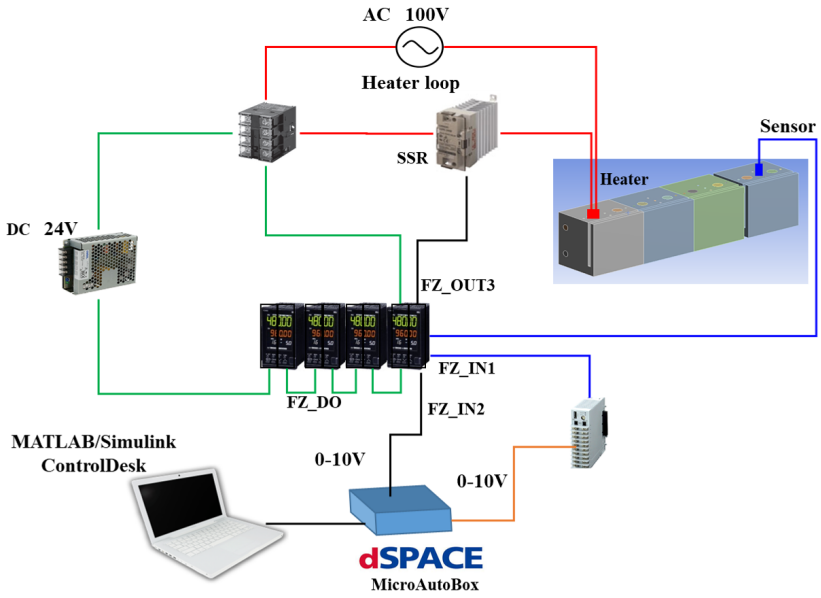


Figure 2.5: Experimental setup of the temperature control system

As shown in Figure 2.5, the digital temperature controllers(RKC, FZ400) are used to control and monitor the temperature of the controlled object, multiple sensors are connected between controller and controlled object, and then send its feedback to the controller. The process heaters start working when get power from the Solid-State Relay(SSR) and the AC power source. Here, the SSR is switching device/power regulator between the heaters and the controller for rapidly controlling the real-time temperature. In particular, the controller can send the control signal to the primary terminals of the SSR for turning on and turning off control. The whole control model is developed and

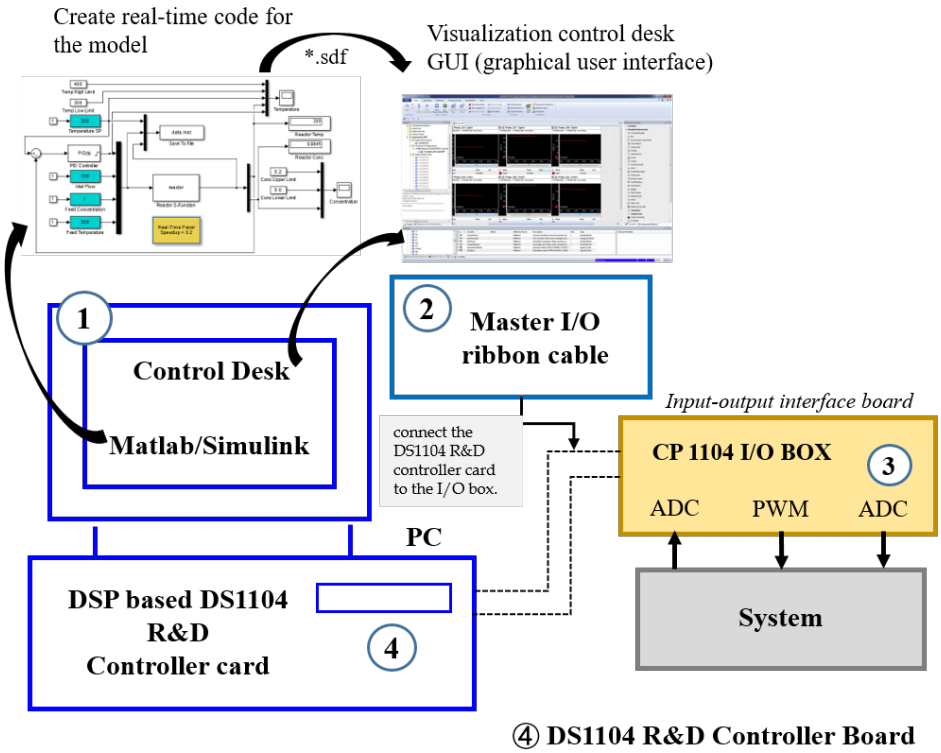


Figure 2.6: Setup of the DSP-based experimental system

built in the MATLAB/Simulink and using the dSPACE DS1104 R&D digital signal processing controller board to link to the hardware[43]. The real-time testing of a designed control algorithm in Simulink is executed by the control board. The DSP-based DS1104 board is commonly installed in the personal computer with a PCI slot and it automatically complies the real-time model and generates the controller code for the hardware. As shown in Figure 2.6, the installed model of the control system in the Matlab/Simulink software will be compiled and the real-time codes are created which are included in a specific (.sdf) file, then the control desk can access and monitor the real-time variables which include the inputs and outputs of the implemented control system. The data will be transferred by the ribbon cable which connects the DS1104 controller card and the CP1104 I/O box. The digital signals will be sent from the DS1104 to the CP 1104.

The detailed information of used equipment are listed in Table 2.2 as follow. The simplified block diagram of the temperature control system is given in Figure 2.7. The applied process and temperature digital controllers(FZ400), which has features including: (1) measurement accuracy: $\pm 0.1\%$ of displayed value; (2) sampling time: 0.05 sec; (3) 11-segment 5 digit LCD display. The physical drawings of the experimental platform and modules for process modeling and control system design are shown in the Figures 2.8 and 2.9.

Table 2.2: Experimental equipment information of our temperature control system.

Products	Information
Controlled Object	Aluminum block: 120 × 60 × 50 (mm)
PC	CPU: i5-8400; RAM: 16.0 GB; GPU: On-Board
DSP	dSPACE: DS1104 R&D Controller Board
AD-DA Converter	dSPACE: Panels for Single-Board-Hardware
Heater	Watlow: Firerod Cartridge Heater, type G2A56 (150 Watt)
Thermocouple	RKC: Type K, class 2 (−200 °C–900 °C)
Solid State Relay (SSR)	Omron: G3PE-245BL, DC12-24
Temperature Controller	RKC: FZ400

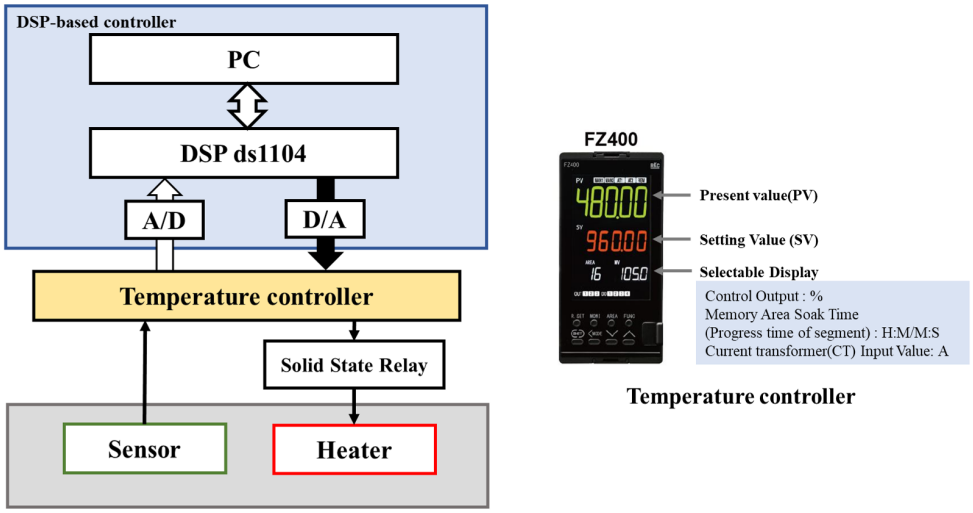


Figure 2.7: Simplified control flow chart

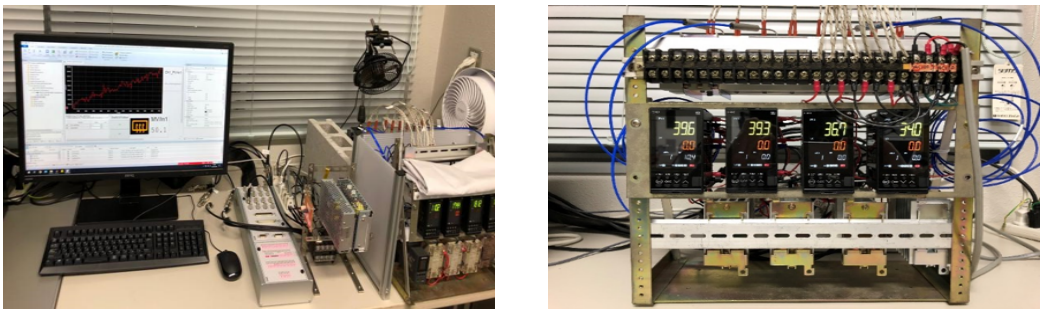


Figure 2.8: Experimental platform of temperature control systems

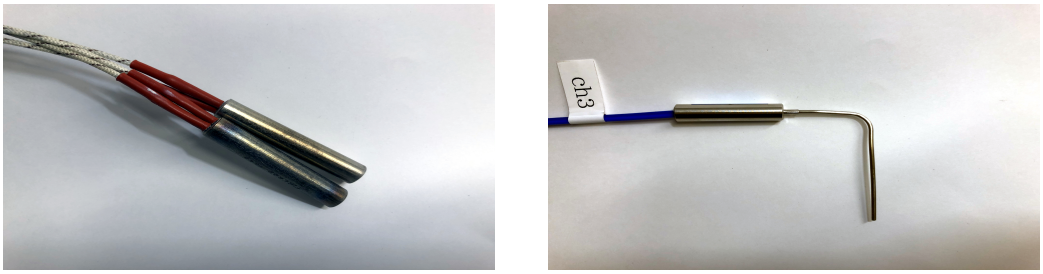


Figure 2.9: Images of the heater (left) and thermocouple (right)

2.2.2 Single-Input Single-Output control system identification

The controlled object for our Single-Input Single-Output (SISO) control system is built as Figure 2.10. There are two aluminum blocks connected tightly on the same plane by nuts and each block is 60×60×50(mm) in size. Define the left block as Channel1(ch1), and the right block as Channel2(ch2), respectively. Each block has two heaters in the hole of 30[mm] and one sensor for detecting the block temperature output, which are placed closer to the inner center, respectively. The detailed experimental conditions are given in Table 2.3. The ambient temperature during the performed identification experiments was 22°C, the sampling time is 0.1s and the experimental time is set as 80,100s for capturing enough information of the controlled system.

Table 2.3: Experimental condition settings

Sampling time	0.1 s
Experimental time	80,100 s
Input signal	40% PWM duty of step signal (4V)
Output signal	temperature of sensor : 400°C/10V
Ambient temperature(initial)	22°C

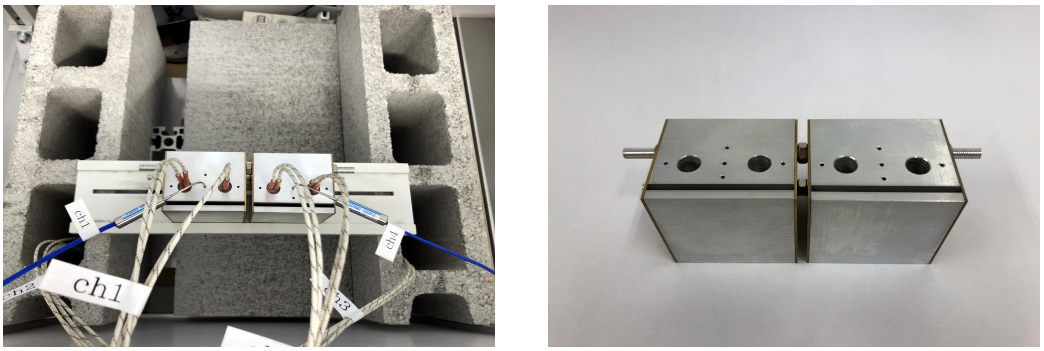


Figure 2.10: Experimental setup for the SISO controlled object

For getting the transfer functions of each channel of the temperature system. In simulation, the system identification procedure and identification condition by m-file in MATLAB software for control system model identification are given as follows:

1. Selection of input and output signals
2. Power spectrum verification of measured signals from actual systems
3. Check correlation of input and output signals
4. System Identification: the range of selection of degree of ARX model in minimizing prediction error including: numerator oreder, denominator order and delay time
5. Confirm pole, zero of the identification model (higher order model)
6. Model order reduction

7. Compare time response or frequency response between identified model and reduced dimension model.

2.2.3 Identification results of SISO system

As introduced above, the measured output time-domain response data is preprocessed for system identification in the open-loop step response test, such as removing the initial value from the output data shown in Figure 2.11. For a sequence of time samples, Figure 2.12 illustrates the power spectrum (PS) estimation of the measured signals.

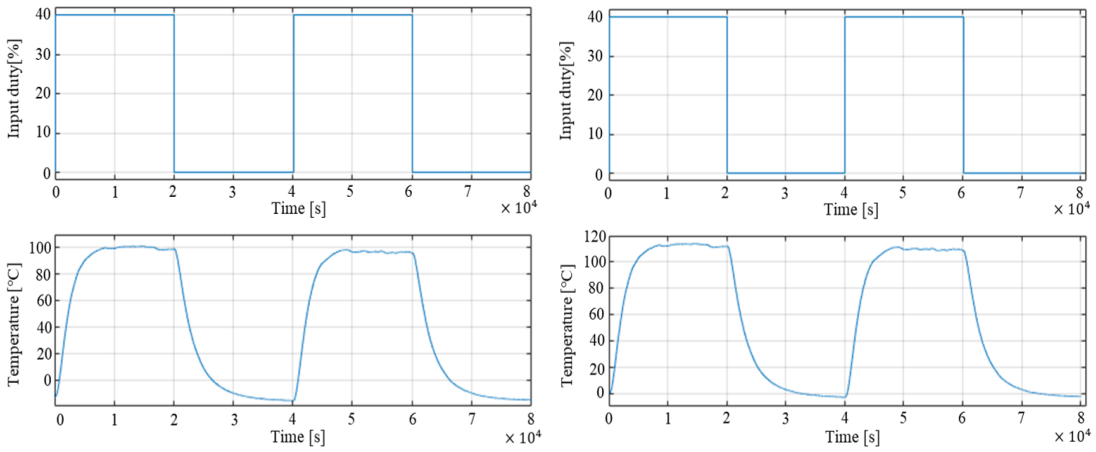


Figure 2.11: Measured input and output data of step response (left) and after removing the offset (right)

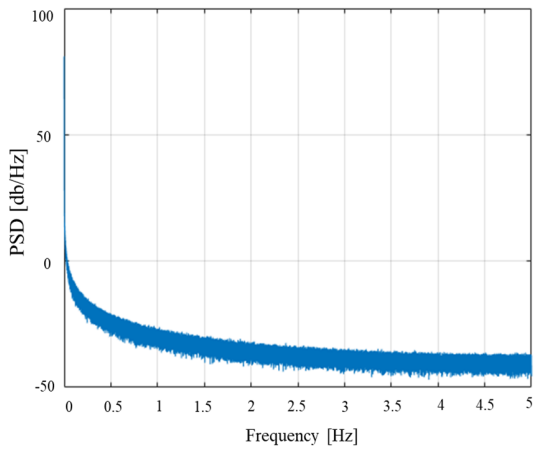


Figure 2.12: Power spectral density

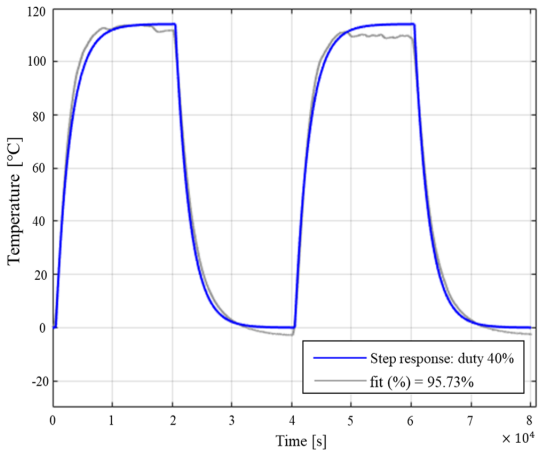


Figure 2.13: Fit ratio of identified model

After the higher order ARX model is derived, reduce the dimension of the model into the objective transfer function in the first-order form. Model identification process includes fitting parameters in a dynamic continuous or discrete form of the FOPDT model, and the unknown parameters for this system include the time constant (T), gain (K), and

dead time (L), introduced in previous section 2.1.1. According to the high fitting ratio of the identified result as Figure 2.13, the process is finally derived into the following transfer function:

$$P(s) = \frac{2.854}{2395s + 1} e^{-444.7s} \tag{2.24}$$

As shown in Figure 2.13, the percentage of fit which indicates the error between the actual output $y(k)$ and the estimated model output $\hat{y}(k)$ is equal to 95.73%. The stability and performance of the closed-loop system are illustrated by observing the open-loop behavior of the system in Figure 2.14, which is the Nyquist diagram of the identified system model. The point -1+j0 is not encircled and the time delay term which will add the negative phase makes the curve rotate in the direction of the origin.

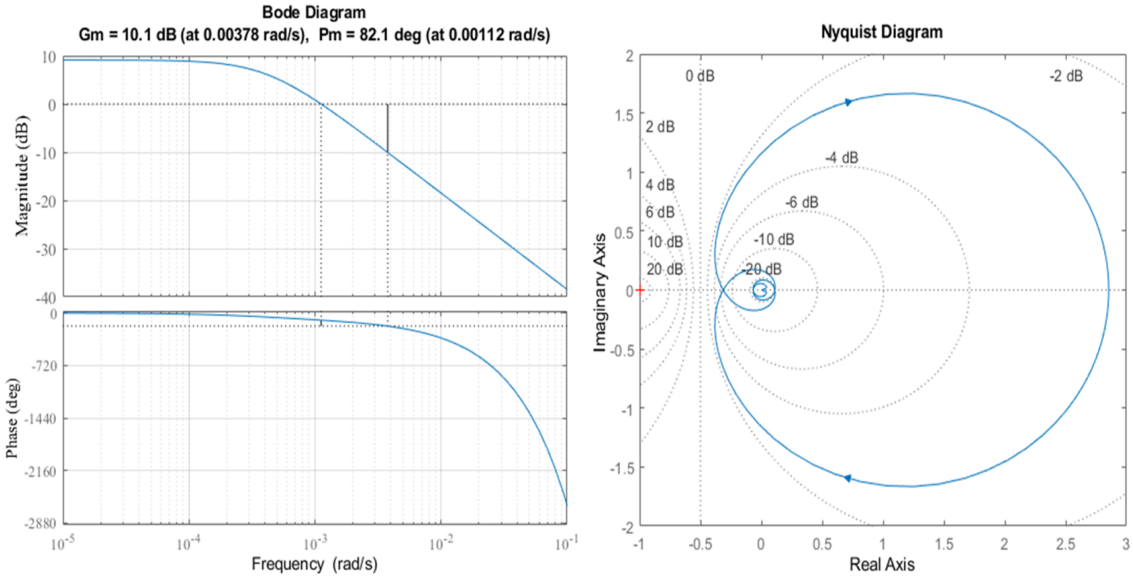


Figure 2.14: Bode and nyquist plots of the identified system model with time-dealy

2.2.4 Multi-Input Multi-Output control system identification

The Multi-Input Multi-Output Control (MIMO) System is built as Figures 2.15. In order to verify the control performance of our proposed NN-based MIMO control system in the following chapter, the controlled object is simplified and constructed by a two-input two-output model. The controlled object consists of four aluminum blocks which are tightly connected on the same plane by nuts and each is 60×60×50(mm) size. And each block is separated in the same distance. Define the left two blocks into a whole and marked by Channel1(ch1), the right two blocks are regarded as a whole and marked by Channel2(ch2). Each block has two heaters in the hole of 30[mm] and one sensor for detecting the block temperature output, which are placed closer to the inner center, respectively. Considering the coupling effect in such a two-input two-output control system, which can be described as Figure 2.16. During the system identification experiments, the ambient temperature (initial temperature) of the identification experiments was 28°C. A step signal (20% PWM duty cycle) is sent to Channel1(ch1) and Channel2(ch2) heaters across the solid state relay, respectively.

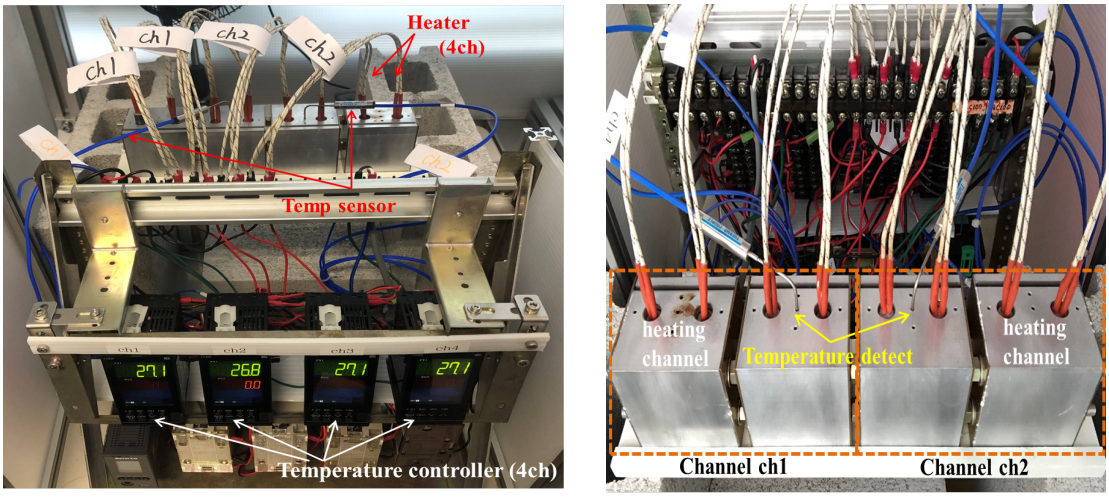


Figure 2.15: Overall view of equipment(left) and the controlled blocks(right)

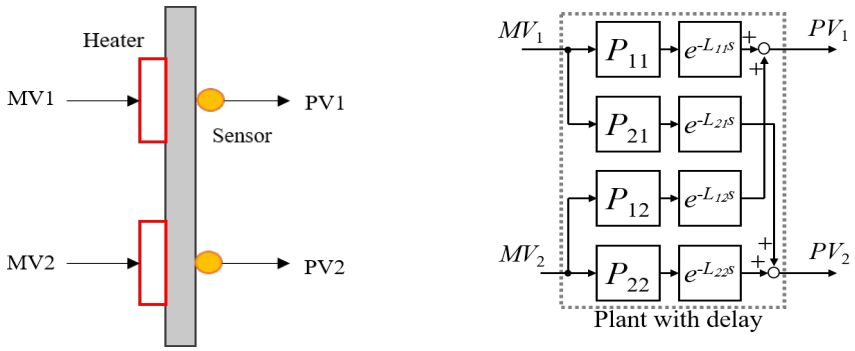


Figure 2.16: Coupling effects in the 2I2O temperature system

As the same in the SISO system identification, the control system models of two-input two-output controlled object are identified from the input and output data obtained from the open-loop step response tests, respectively. Similarly, the system transfer functions of each blocks are derived by estimating the ARX model based on the least-squares criterion in MATLAB. Set the objective structure of the control model is first-order plus dead-time form, the high order models are reduced to the first-order transfer function. For each block, the following FOPTD transfer function as Equation 2.25 can be obtained. In the obtained 2×2 matrix, P_{11} and P_{22} represent the self-interaction of each heating channel, respectively. The off-diagonal terms P_{12} and P_{21} indicate the mutual coupling effect between the heating channels of the controlled object, respectively.

$$
\begin{aligned}
G_p(s) = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} &= \begin{bmatrix} \frac{K_{11}}{P_{11}s+1}e^{-L_{11}s} & \frac{K_{12}}{P_{12}s+1}e^{-L_{12}s} \\ \frac{K_{21}}{P_{21}s+1}e^{-L_{21}s} & \frac{K_{22}}{P_{22}s+1}e^{-L_{22}s} \end{bmatrix} \\
&= \begin{bmatrix} \frac{2.7502}{2482.4s+1}e^{-431s} & \frac{1.4614}{3085.1s+1}e^{-1042s} \\ \frac{1.7352}{3195.9s+1}e^{-973s} & \frac{2.3937}{2588.6s+1}e^{-464s} \end{bmatrix}
\end{aligned}
\tag{2.25}
$$

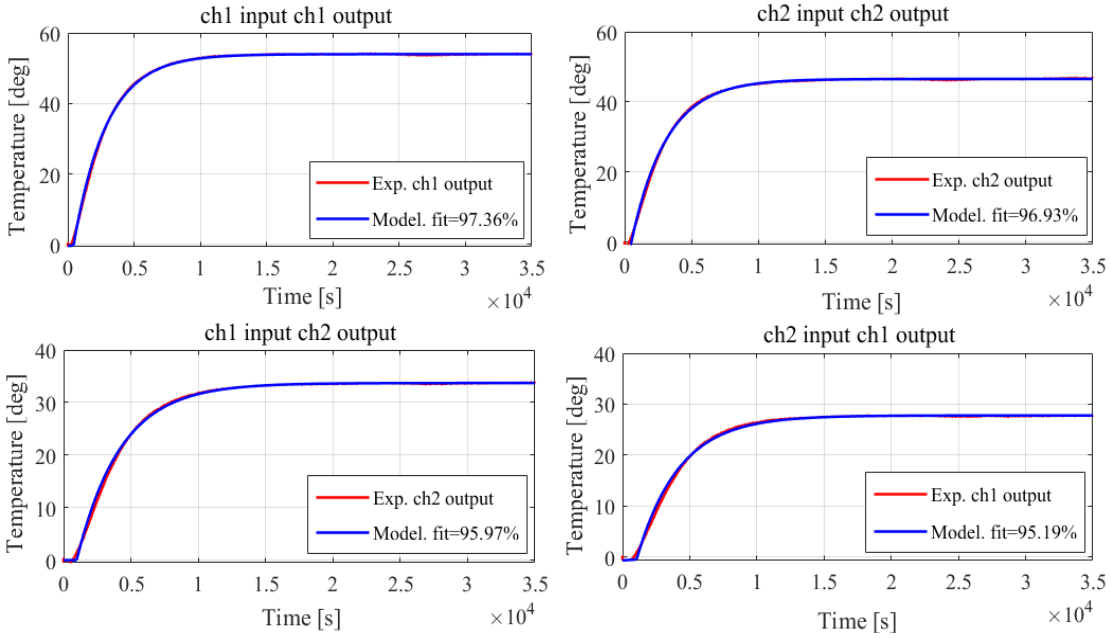


Figure 2.17: System identification results.

From the fitting ratio of of each block with coupling terms in the controlled system in Figure 2.17, the estimation accuracy of each system model compared with the corresponding estimated model is over 95%. The bode plots and nyquist plots of the elements of main diagonal in the identified coupling system model (P_{11} and P_{22}) are given in

Figure 2.18 and 2.19, respectively.

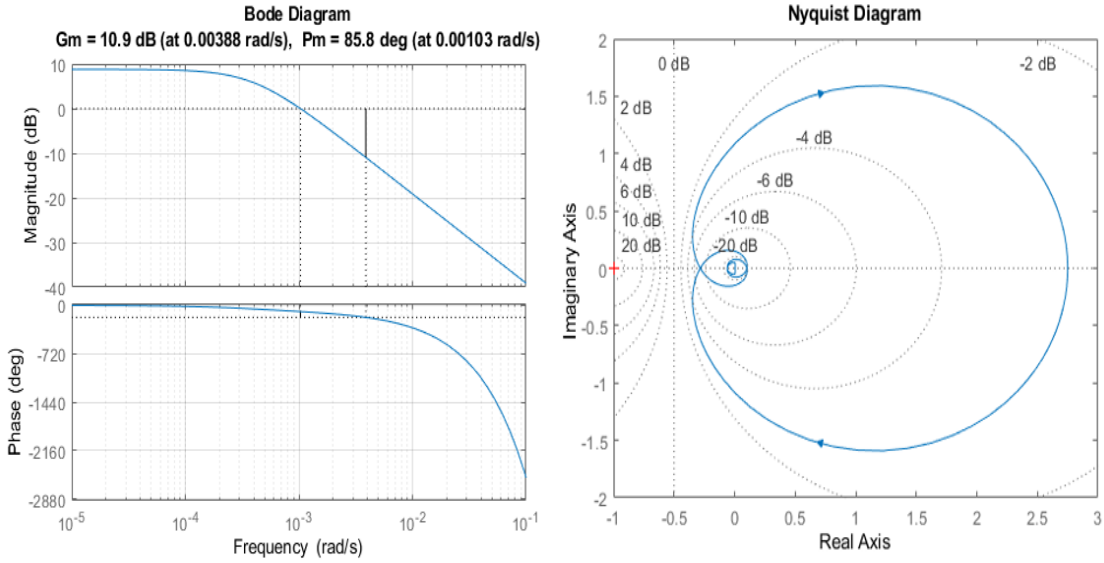


Figure 2.18: Bode plot (left) and Nquist plot (right) of identified MIMO (P_{11})

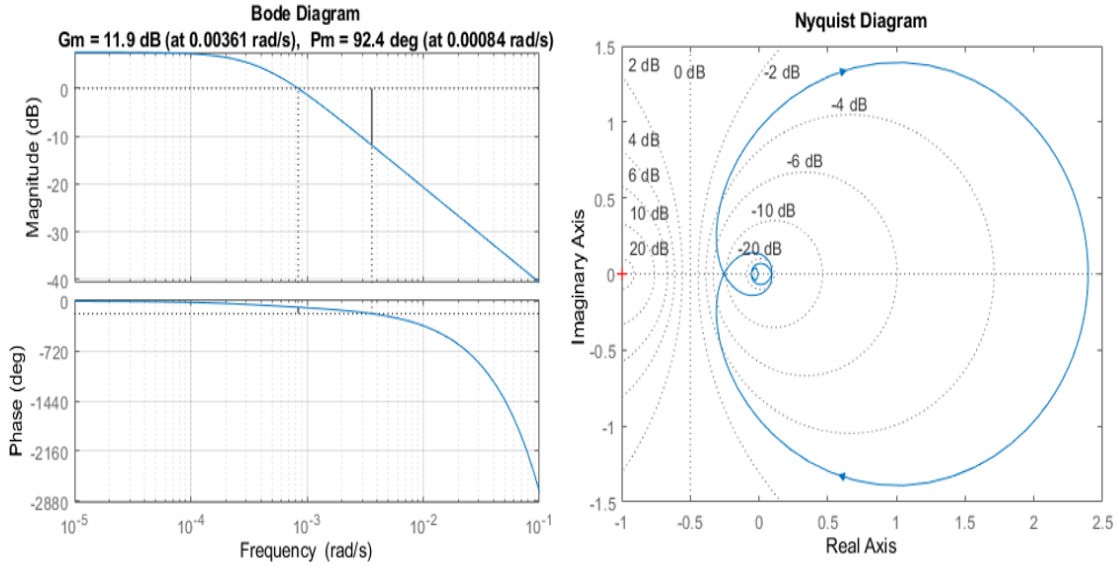


Figure 2.19: Bode plot (left) and Nquist plot (right) of identified MIMO (P_{22})

2.2.5 Approximation of time delay transfer function

The exponential term $e^{-s\tau}$ for time delay in the transfer function can also be expressed in the power series expansion as Equation 2.26. One of common applied rational approximation methods of the time delay term is $Padé$ approximation[44], which has no restriction on the order of both numerator m and denominator n. It is suitable for weak condition of physical realizability and the denominator order n$\leq$10.

$$e^{-s\tau} \approx 1 - s\tau + \frac{1}{2}(s\tau)^2 - \frac{1}{3!}(s\tau)^3 + \cdots \tag{2.26}$$

The corresponding first order and second order of $Padé$ approximations of the exponential terms are:

$$e^{-s\tau} \approx \frac{1 - \frac{s\tau}{2}}{1 + \frac{s\tau}{2}} \tag{2.27}$$

$$e^{-s\tau} \approx \frac{1 - \frac{s\tau}{2} + \frac{\tau^2}{12}s^2}{1 + \frac{s\tau}{2} + \frac{\tau^2}{12}s^2} \tag{2.28}$$

Figure 2.20 shows the step response of a pure time delay of 444.7s, which is expressed by $Padé$ approximations of 1st, 2nd and 3rd order, respectively. Here, the time and frequency responses of the true delay time is compared with different order approximation of it. Although they have an obvious deviation from the exact response for the early time, as the order increases it has less overshoot and faster convergence.

$$e^{x} \approx \frac{1}{(\frac{x}{n} + 1)^n} \tag{2.29}$$

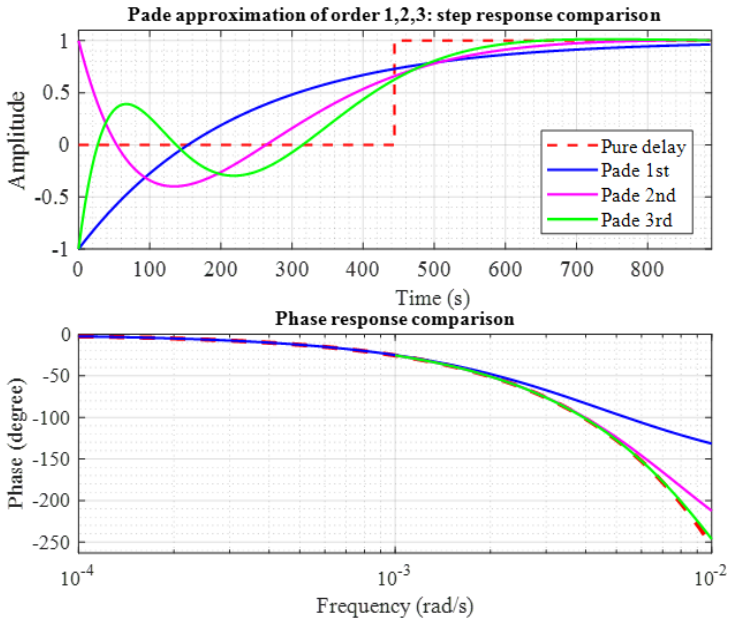


Figure 2.20: $Padé$ approximation of different orders

The amplitude and phase approximation of $Padé$ approximation is more accurate as the order increases, the non-minimum phase is also introduced and shows up as a reverse shock in the step response curve as shown. In order to ensure the accuracy and reduce the order of the delay term, here the 2-order exponential approximation is used to describe the system delay as Equation 2.29.

For a better observation of the approximation results, Figure 2.21 illustrates step response results of the control system with exact delay, $Padé$ approximation and exponential approximation in time domain.

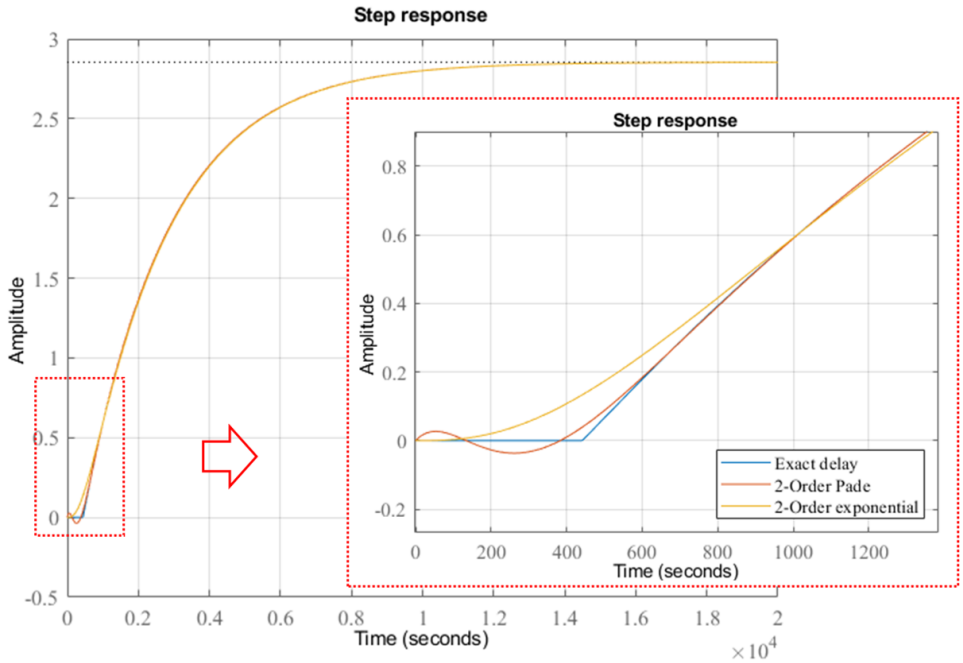


Figure 2.21: Comparison of approximate results for pure delay links in time domain

The first-order plus dead-time transfer function of the identified model in the SISO system can be written by Equation 2.30. The frequency response of the identified model is also plotted in Figure 2.22 and compared with the approximation model.

$$P(s) \approx \frac{2.854}{2395.4s+1} * \frac{1}{(\frac{444.7s}{2}+1)^2} \tag{2.30}$$

In the same way, the transfer function of time delay for the identified P_{11} and P_{22} in the MIMO system can be written as follows:

$$P_{11}(s) \approx \frac{2.7502}{2482.4s+1} * \frac{1}{(\frac{431s}{2}+1)^2} \tag{2.31}$$

$$P_{22}(s) \approx \frac{2.3937}{2588.6s+1} * \frac{1}{(\frac{464s}{2}+1)^2} \tag{2.32}$$

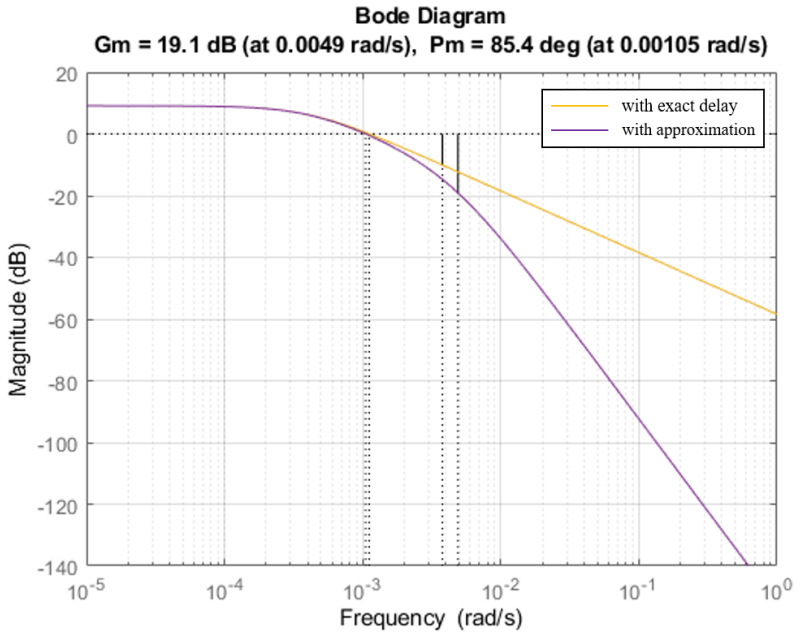


Figure 2.22: Comparison of approximate results in frequency domain

2.3 Classical Control Strategies

2.3.1 Integral-proportional derivative(I-PD) controller

For the heating system with frequent changes in the reference signal, how to reduce the overshoot and steady-state error is important tasks for designing a good control algorithm. Consider a general formula of a two-degree-of-freedom PID controller in the parallel form as[45]:

$$u = P(br - y) + I\frac{1}{s}(r - y) + D\frac{1}{1 + \eta\frac{1}{s}}(cr - y) \tag{2.33}$$

where, r, y and u are the reference signal, system output and the controller output, respectively. And P, I and D correspond to the proportional gain, integral gain and derivative gain, respectively. Here, the low-pass filter factor η in the derivative term is usually used to inhibit the effect of high frequency noise. In addition, b and c are defined as the reference value weights affecting the proportional and derivative terms, respectively.

To reduce the impact of frequent changes in the reference signal r on the controller output u, the components b and c are equal to 0 for avoiding a large spike caused by the derivative and proportional terms. This variant of the PID controller is called is integral-proportional derivative (I-PD) controller, given by Equation2.34.

$$u = -Py + I\frac{1}{s}(r - y) + D\frac{1}{1 + \eta\frac{1}{s}}y \tag{2.34}$$

The block diagram of a I-PD controller-based control system is shown as Figure 2.23. The controller parameters contains proportional gain K_p, integral time constant T_i and derivative time constant T_d.

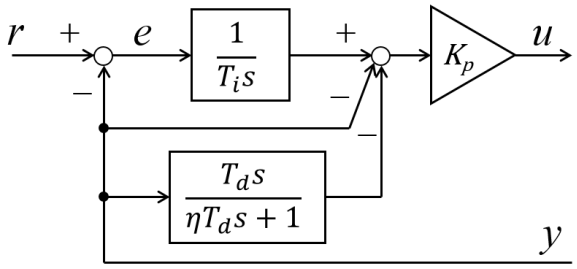


Figure 2.23: Structure of a parallel I-PD controller

The unknown parameters of the controller are determined by Ziegler–Nichols proposed tuning rules in 1942, which is also based on the information obtained from the step response test[46]. Here, the parameters are adjusted for an ideal controller as rules given in Table 2.4, where the variable $R = K/T$ is obtained by the following FOPDT transfer function:

$$P(s) = \frac{K}{Ts + 1} e^{Ls} \tag{2.35}$$

Table 2.4: ZN controller tuning rules based on step response

Controller	Proportional Gain K_p	Integration Time T_i	Derivative Time T_D
P	1/RL	-	-
PI	0.9/RL	3.33L	-
PID	1.2/RL	2L	0.5L

Here, consider our identified and approximate model in the SISO and MIMO control system as Equations 2.30 and 2.33, the corresponding parameters of the controller are calculated in Table 2.5.

Table 2.5: Determined I-PD controller parameters

Controlled Object	Proportional Gain K_p	Integration Time T_i	Derivative Time T_D
SISO	2.264	889.48	222.37
MIMO (P_{11})	2.525	861.51	215.38
MIMO (P_{22})	2.797	927.98	231.99

2.3.2 Anti-windup compensator

When the system reaches saturation, due to the existing of integration, the system is continuously stacking errors and thus affects the speed of desaturation. In most PID control systems, without the anti-windup, the controller will operate in a nonlinear region which output is outside the saturation limit for input signal[47]. It means that even though there is increasing control signal, it doesn't have any effect on the system output. And when the reference signal changes frequently, there is an obvious delay before the controller output recover to the actuator's range with a steady-state value.

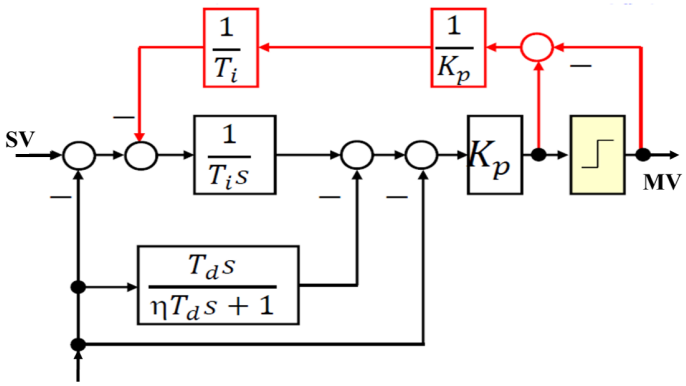


Figure 2.24: I-PD controller block with back-calculation based on back-calculation

In order to ensure the controller output returns to the linear region quickly and improve the control performance, a commonly used strategy is anti-windup based on back-calculation as Figure 2.24, where SV and MV are the setpoint value and the manipulated variable value, respectively. The saturation limit of control output is also set as 100%. It adds a feedback loop to reduce the effect of the integrator when the I-PD controller output reaches the saturation and start to operate in the nonlinear region.

Figure 2.25 illustrates the controller output results of simulating the I-PD control model without and with anti-windup activated on the closed-loop. With the anti-windup feedback loop, the control signal can quickly return to the linear region and with smaller vibrations.

To better visualize the effect of anti-windup compensation in control system, Figure 2.26 illustrates the measured temperature output PV(process variable) with and without the anti-windup feedback loop. It is obvious that there is a big overshoot and long delay responding to the changes in the reference signal without the anti-windup loop. In contrast, the saturated input with anti-windup effectively improves the reference tracking performance.

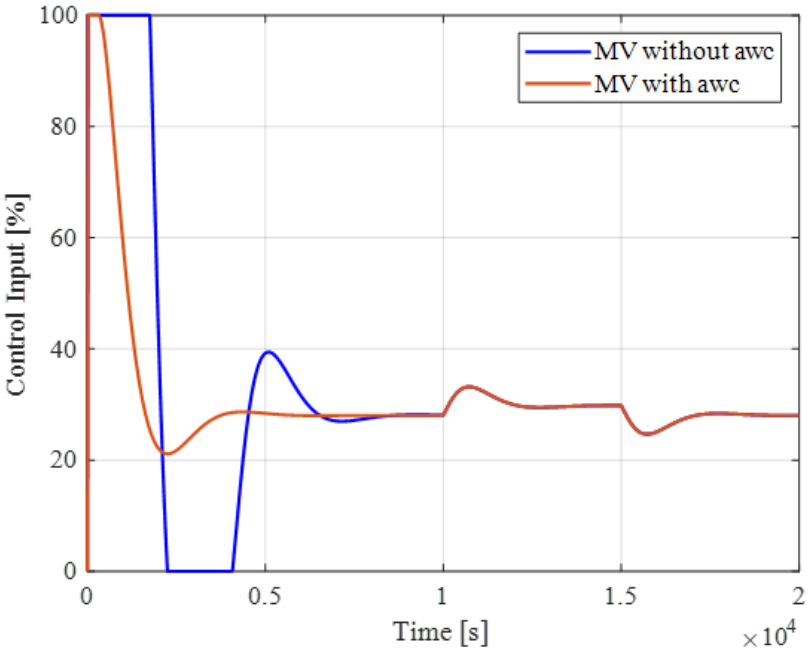


Figure 2.25: Comparison of saturated controller output MV with and without the back-calculation feedback loop

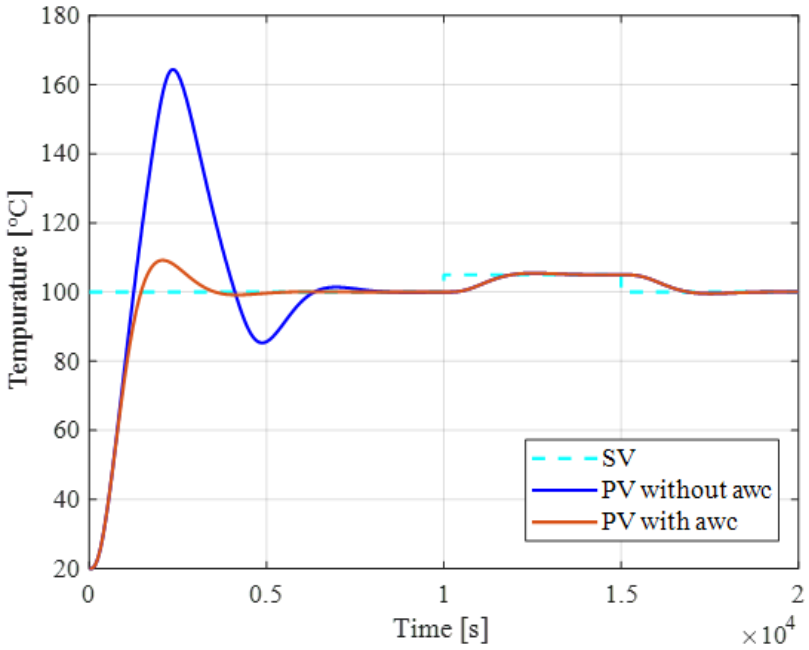


Figure 2.26: Measured process output(PV) with and without anti-windup compensator

2.3.3 Feedforward compensation-based I-PD controller

The main task of control algorithm design is to ensure the reference tracking performance and anti-interference ability of the system. Whereas the pure feedback controller (single-degree) has the inherent limitations, so that cannot simultaneously meet the requirements of tracking reference and disturbance rejection. In most cases, it is difficult

to find the best compromise between them.

Therefore, after the feedback loop is adjusted for the disturbance suppression control, a feedforward control loop [48] is added as Figure 2.27 to improve the tracking of the target value, where k_f is a feedforward gain.

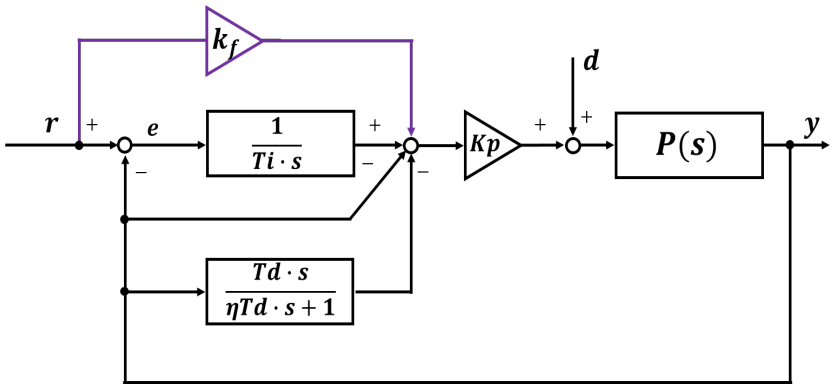


Figure 2.27: Structure of a feedforward compensation-based I-PD controller

Generally, the feedforward compensation term need to calculate the corresponding control input according to the target output, which requires the reverse model of the controlled object. The reverse models sometimes are physically hard to be realized, such as system with dead time, or zero of the controlled object in the right half plane. Hence, instead of the calculating the transfer function of inverse system, using a static feedforward compensator (k_f) that is equal to a constant gain for compensate the steady-state error. This structure is simple and easily implemented. Here, define the k_f=1 is fast response mode and k_f=0 is slow response mode.

The experiment time is set as 15,000s and the sampling time is 0.5s. The initial value is zero. Perform reference value tracking tests in both response modes. The multiple cycles of time response for reference tracking based on the identified SISO system model, which containing the temperature increasing (100°C→105°C in 5000s) and decreasing (105°C→100°C in 5000s) periods in 10,000s, are shown in Figure 2.28 and one complete cycle is enlarged as Figure 2.29. The actual control output MV with different feedforward gains are compared in Figure 2.30.

From the above simulation results, compared with the continuous time response curve without the feedforward loop (k_f), the fast mode control results in approximately 40% overshoot although it responses faster than the slow mode with about 8% overshoot. Obviously, the feedforward compensation gain is helpful to improve the system response speed in reference tracking.

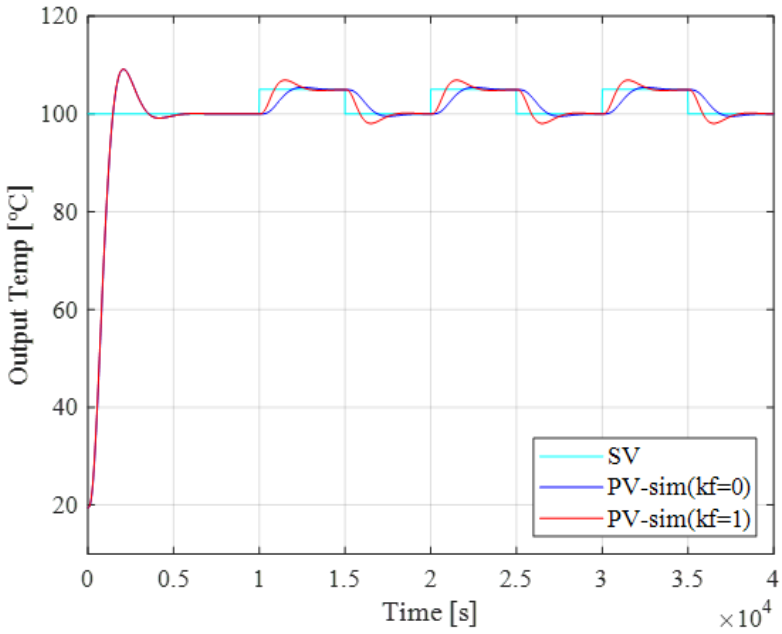


Figure 2.28: Comparison of time response curves of reference tracking for multiple cycles in two modes (fast: k_f=1 and slow: k_f=0)

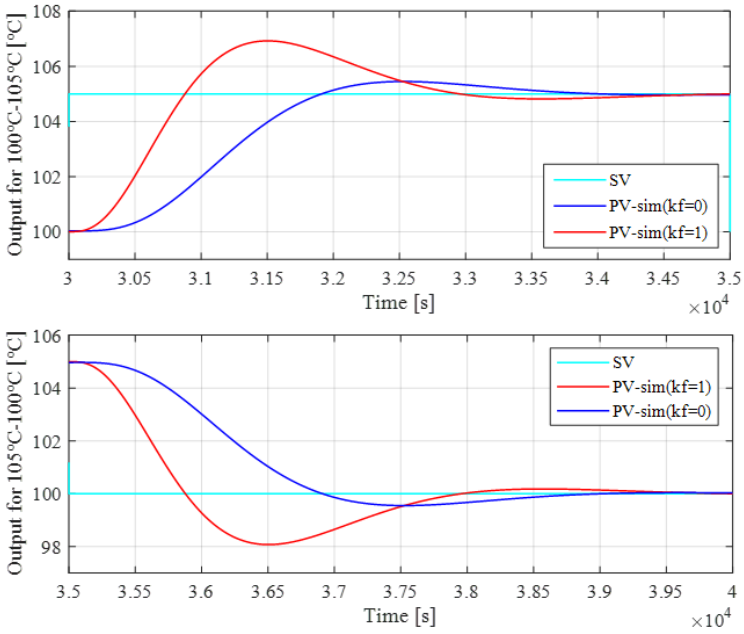


Figure 2.29: One cycle of reference value tracking response (containing two periods from 100°C to 105°C and from 105°C to 100°C)

Experimental Results

The designed control structure with two response modes (fast: k_f=1 and slow: k_f=0) are tested in the actual system for verifying the tracking performance and accuracy of the identified control model. The block diagram of the control system designed in MATLAB/Simulink is given in Figure 2.31.

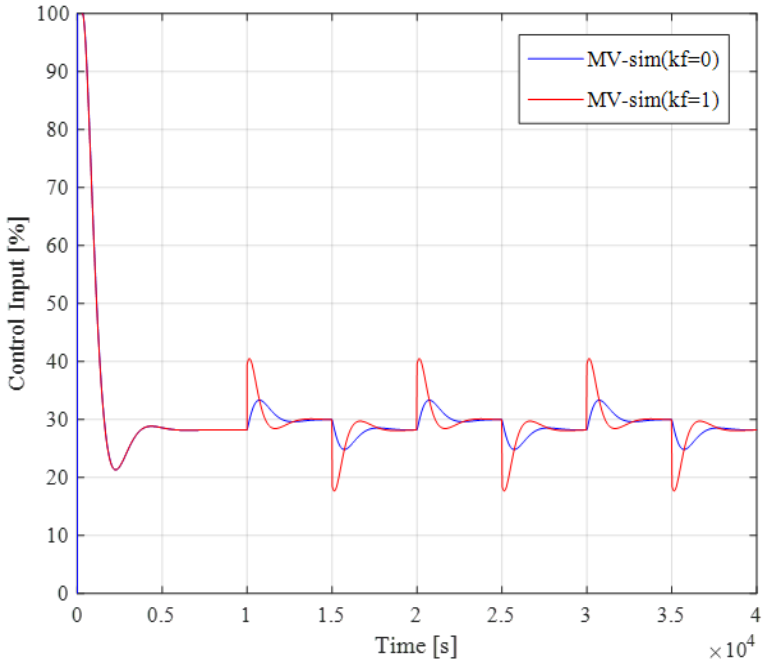


Figure 2.30: Control input in one cycle of reference value tracking response

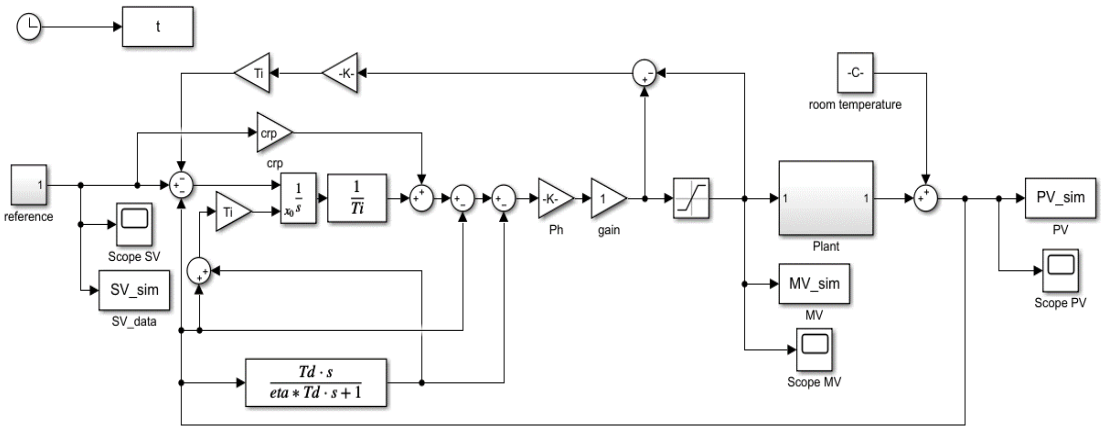


Figure 2.31: Control system model built in MATLAB/Simulink software

The experiment time is set as 40,000s and the sampling time is 0.5s. After the temperature of the controlled object goes up from the initial temperature to 100°C and the system output reaches steady state in 10,000s, a step signal of ±5 degree is added to the baseline value 100°C. The controlled object is heated and the output temperature increases from 100°C to 105°C in 5000s, and then decreases from 105°C to 100°C in 5000s, respectively. One complete cycle contains one increasing change and one decreasing change in the continuous time response.

The following figures show the comparison results of different response modes based on the identified model in the actual experimental system, respectively. The temperature changes up and down multiple times in the whole time response as shown in Figure 2.32.

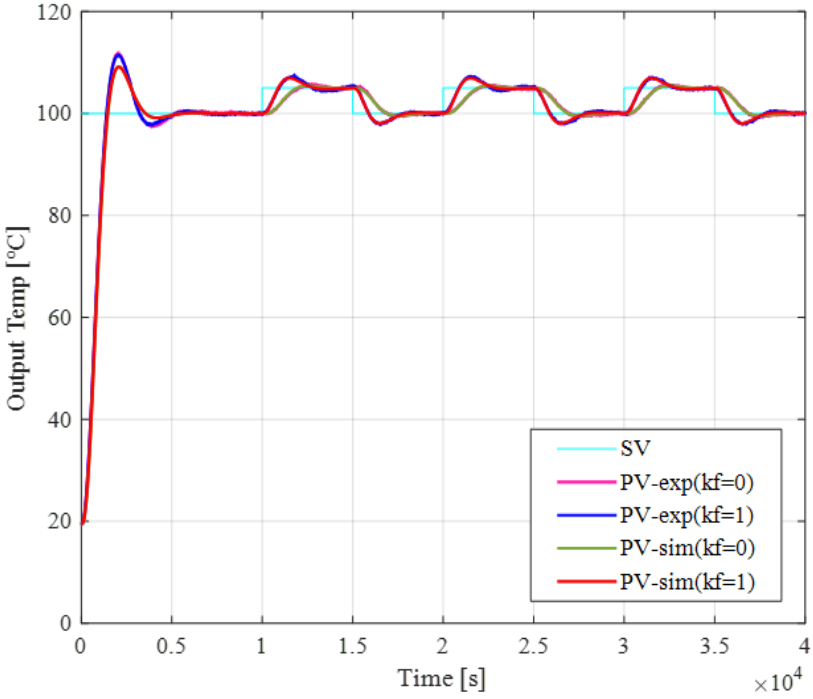


Figure 2.32: Comparison of time response curves of reference tracking for multiple cycles in two modes (fast: k_f=1 and slow: k_f=0)

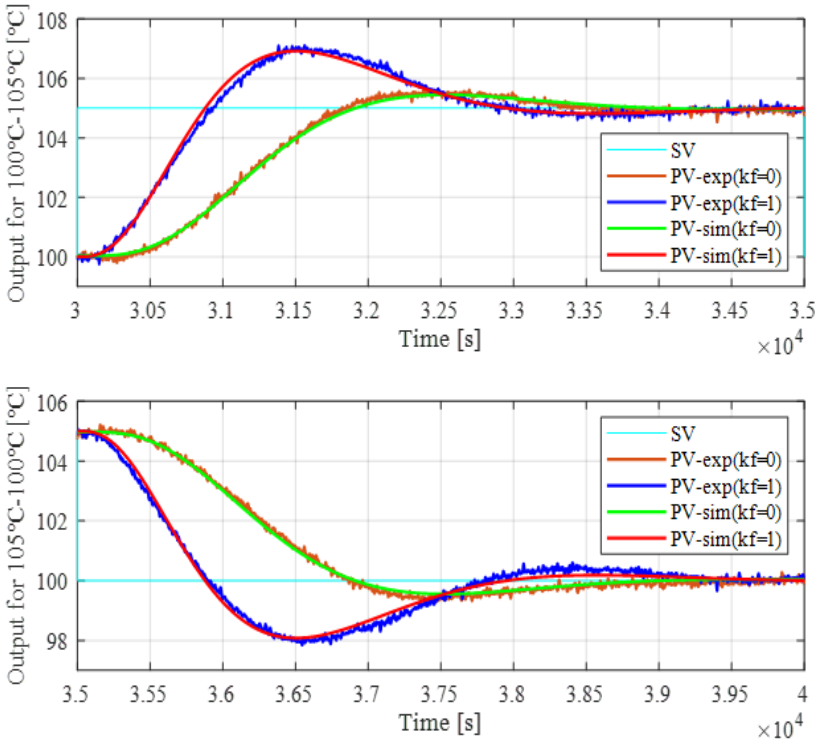


Figure 2.33: One complete cycle of reference value tracking response (containing two periods from 100°C to 105°C and from 105°C to 100°C)

From the results of the enlarged one cycle response in Figure 2.33, the experimental and simulation results are consistent. The accuracy of the identified model is confirmed. On the other hand, by adding the feedforward gains, the system output can follow the constantly changing setpoint at different speeds without steady-state error. However, although the response speed of system with the gain k_f=1 is accelerated compared to that with the gain k_f=0, the overshoot of the system is also increased and thus the system stability deteriorates. It is apparent that although the operation of PID controller is simple, but the quality of regulation is general, and the parameters need to be constantly corrected in application for satisfying the desired dynamic and steady-state response requirements.

2.4 Conclusion

In this chapter, the modeling method based on the system identification method are applied for obtaining the transfer functions which describes the controlled system behavior. Based on the identified models, classical control strategies commonly used in industrial process control are also discussed. To solve the problem in classical control theory that is hard to meet the requirements in reference tracking performance and disturbance rejection, especially in large-delay and strong coupling temperature control systems, the I-PD control method based on the identified models will be used in the following chapters, for comparing with the proposed NN-based learning control methods. The aim is to deal with the conflict between response speed, overshoot, disturbance suppression capacity in temperature controlling, and to get satisfied static and dynamic response performances.

Chapter 3

Reference-model-based Neural Network Control Method for MIMO Temperature Control System

The temperature control requirements for most industrial processing are very strict, and thus the precise and automatic control performances are gradually to be realized in practical production. There are many factors affecting the control accuracy including: (1) a time delay (thermal lag) of the temperature change of the controlled object; (2) the relative location of the sensors and heaters in the load; (3) the response speed and applicability to different applications of controllers; (4) capacity and excessive heat losses of the heaters, etc. Commonly, for an operating thermal system, one or several heaters and sensors are placed in the workload, the temperature difference(gradient) always exists at all times. Physically, it is observed that the measured temperature decreases gradually from the position near the heat source to the edge of the entire system. The relative location of measurement and heater elements are required to be as close as possible to the controlled points. Moreover, the appropriate thermal insulation measures are required to reduce the heat loss from the process systems.

In addition, due to the asymmetry of the structure of the products, the interference of each control output of heaters placed over a surface, and disturbance errors caused by different processing equipment in the high speed machining process, the inconsistent distribution of temperature becomes more likely to happen and brings huge damages to the production. Such as in many applications, a multi-level furnace and wafer surfaces, uniform temperature of the surface is a very important factor affecting the product quality[49, 50]. In order to reduce energy loss, the time required to control and adjust the temperature deviation, and the interference between heaters placed over the surface,

precise and effective control algorithm is the key to the higher product quality and production efficiency in the thermal processing.

In recent years, the application of neural networks has become a new research focus in the control of time-delay systems, especially for a large lag, strong coupling and nonlinear temperature system, many studies have proved the effectiveness of it. Because of its nonlinear approximation, self-learning and self-organizing ability, neural network control is easier to realize adaptive control of time-delay systems without accurate identification of controlled objects. And the neural network controller has the characteristics of simple structure, strong adaptability and real-time control. For an equipment with multi-modules, Zhang et al. developed a control system which utilized the advantages of PID controller and fully connected neural network, which can adjust the internal temperature of the controlled object without object modeling[51]. Lee C et al. designed separate PID-NN controllers to control a nonlinear and complex fan cooling system for improving the transient-state temperature response[52]. The focus of past studies is almost utilizing the learning properties of neural networks to realize the optimal parameters adjustment of the PID controller. Actually, the neural networks can act as a controller directly in feedback control to replace a traditional PID controller and improve the control performance effectively[53, 54].

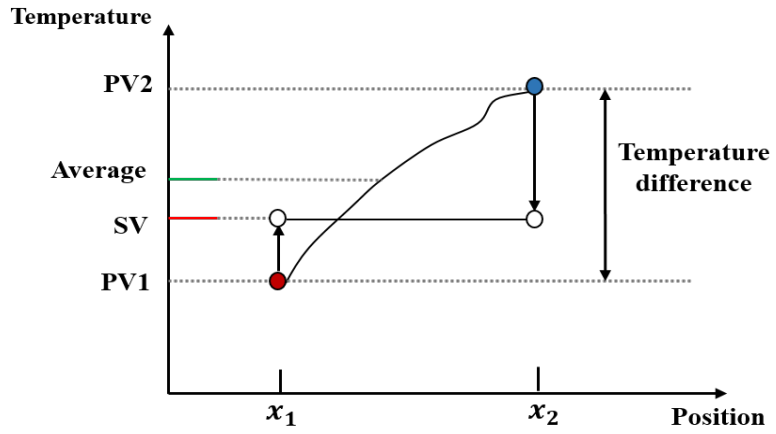


Figure 3.1: Temperature difference between a pair of points on the panel

In this chapter, in order to achieve a uniform temperature of different points in one plane and reduce the temperature difference between output of every pair of points as shown in Figure 3.1, while improving the dynamic and steady performances, a reference-model-based multi-layer neural network control system is proposed for the multi-input multi-output(MIMO) system. The basic concepts of applied neural networks are introduced at first, and then describes the proposed reference model-based MIMO control system composition and principle in detail. Finally, to verify the effectiveness and reliability of the proposed control scheme, the simulation and experiments are carried out.

3.1 Data-driven Control Method

A large amount of data is generated and stored in industrial processes at all times, containing all useful information about process operation and equipment state. Therefore, in the case that accurate process model cannot be obtained easily, data-driven control methods can be adopted to directly design the controller by analyzing and learning the offline or online process data.

Model-free Adaptive Control From the perspective of data model, model-free adaptive control mainly refers to the design, analysis and operation of the control system in adaptive control without mechanism model and parameter analytic model of the controlled system. However, as a data-driven control method, model-free adaptive control needs to effectively organize and sort out input and output data of the system, mine out effective information and calculate control instructions according to the information[55, 56]. Therefore, consider the complexity of thermal processes and the exact process model is not easy to be obtained, a neural network-based data-driven control method is presented to solve nonlinear, uncertain problems in our control system.

3.2 Review of Artificial Neural Networks

Artificial Neural Network (ANN) models refer to a series of mathematical models inspired by biology and neuroscience[57]. They are formed on the basis of mimicking the information transmission patterns of neurons in the human brain, but they are applied in machine. In the field of artificial intelligence, ANN is often called Neural networks (NN) and is one of performing frameworks in machine learning. They are widely used for processing large amount of data samples to model these unclear relationships between different patterns. Many successful applications of neural networks in different fields including: regression estimation of temperature and stock prices, classifications of images and handwritten digits, etc.

A fully connected multi-layer neural networks are built up of many layers with lots of nodes/neurons connected to each other. The dimension of each layer can be different as required in applications. Figure 3.2 shows a typical multi-layer(two-layer) fully-connected neural network structure with different numbers of neurons i, j, k in input layer, hidden layer and output layer, respectively. It consists of the input signal x=$(x_1, \ldots, x_i)^T$ and the output signal y=$(y_1, \ldots, y_k)^T$. Weight parameters $W_{in} \in \mathbb{R}^{j \times i}$ and $W_{out} \in \mathbb{R}^{k \times j}$ are the connections between the input layer and the hidden layer and the connections between the hidden layer and the output layer.

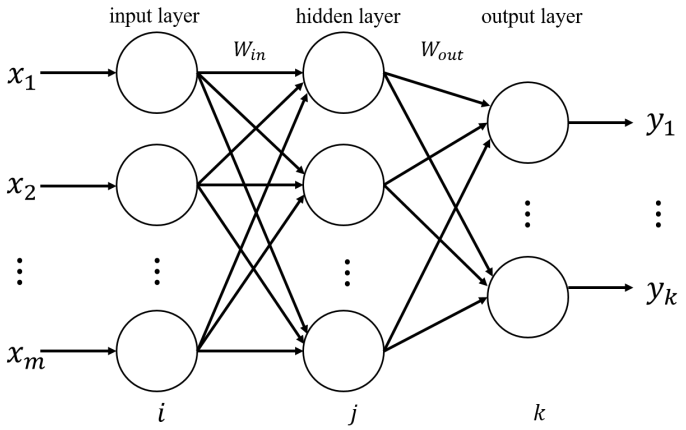


Figure 3.2: A typical multilayer forward neural network

In a neural network, each layer of neurons learns a more abstract representation of the values of the previous layer. Commonly, a neural network model with many hidden layers (>2) is called deep neural network (DNN)[58, 59]. As the number of layers of the network increases, the latter can represent a deeper information abstraction of the former. Such as in the classification problem, more abstract features are extracted to distinguish things, so as to obtain better ability to distinguish and classify. Different from machine learning, it can use a large amount of labeled data for training the model and directly learns features from data without manual feature extraction.

3.2.1 Types of learning modes

Supervised learning It develops models for prediction based on labeled input and output data, which means there are corresponding output variables to the given input variables. It's usually used to solve classification (put things into categories) and regression (predict continuous, specific values) problems. It is characterized by a clear goal, so that results can be measured.

Unsupervised learning It commonly refers to one technique of machine learning for pattern recognition and clustering of data[60]. The input data feed into it are unlabeled, which means there are not corresponding output variables to the given input variables. It's characterized by training without a clear purpose, and you can't know in advance what the outcome will be.

3.2.2 Forward propagation computation

Feedforward neural network obtains the final output $\mathbf{a}^{(L)}$ of the network through layer by layer information transmission as Equation 3.1. The entire network can be viewed as

a composite function $f(\mathbf{x}; \mathbf{W}, \mathbf{b})$, taking the vector $\mathbf{x}$ as the input $\mathbf{a}^{(0)}$ of the first layer and the output $\mathbf{a}^{(L)}$ of layer L as the output of the whole function, where $\mathbf{a}^{(0)} = \mathbf{x}$. A typical neuronal structure is as Figure 3.3.

$$\begin{aligned} \mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \cdots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} \\ = f(\mathbf{x}; \mathbf{W}, \mathbf{b}) \end{aligned} \tag{3.1}$$

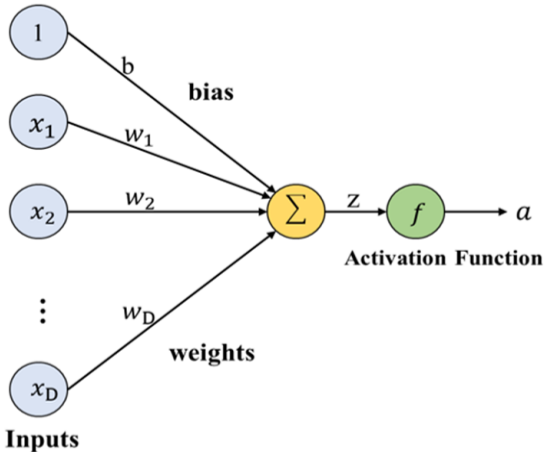


Figure 3.3: Typical neuronal structure

Suppose a neuron receives D inputs: x_1, x_2,..., x_D, let vector $\mathbf{x}$ = [x_1; x_2;...; x_D] represents this set of inputs, and use net input z $\in \mathbb{R}$ to represent the weighted sum of input signals $\mathbf{x}$ obtained by a neuron. $\mathbf{W}$ = [w_1; w_2;...; w_D]$\in \mathbb{R}^D$ is D dimensional weight vector and the bias b $\in \mathbb{R}$.

$$z = \sum_{d=1}^{D} w_d x_d + b = \mathbf{W}^\top \mathbf{x} + b \tag{3.2}$$

The following table gives the common notations describing the feedforward neural networks[61].

Table 3.1: Signs of a feedforward neural network

Sign	Definition
L	number of the network layer
M_l	number of neurons in layer l
$f_l(\cdot)$	activation function of layer l
$\mathbf{W}^l \in \mathbb{R}^{M_l \times M_{l-1}}$	Weight matrix for layer $l-1$ to layer l
$\mathbf{b}^l \in \mathbb{R}^{M_l}$	bias for layer $l-1$ to layer l
$\mathbf{z}^l \in \mathbb{R}^{M_l}$	input of neurons in layer l (net activation)
$\mathbf{a}^l \in \mathbb{R}^{M_l}$	output of neurons in layer l (activation)

Activation function

Activation function is very important for neurons in each layer to enhance the representation ability and learning ability of the network, the activation function should have the following properties[62]:

(1) They are usually continuous and differentiable (allowing non-differentiable at a few points) nonlinear functions, because the network parameters can be learned directly using numerical optimization methods.

(2) The activation function and its derivative function should be as simple as possible to improve the efficiency of network calculation.

(3) The range of the derivative function of the activation function should be within a suitable interval, neither too large nor too small, otherwise it will affect the efficiency and stability of training.

Several widely used activation functions are characterized by:

a) Sigmoid function: It is a S-shaped function and can smooth and bound its total input. It always gives a real-valued output and have nice derivatives which make learning easy.

$$y = \frac{1}{1 + e^{-z}} \tag{3.3}$$

b) Tanh (hyperbolic tangent) function: It can be regarded as the enlarged and shifted sigmoid function, whose range is $(-1, 1)$ and defined as follows.

$$tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{3.4}$$

c) Rectified linear unit(ReLU): It is also called rectifier function[63], which can make all the negative input values zero and keep the positive inputs the same. It has been the most commonly used function in deep neural networks. It is actually a ramp function and defined as:

$$ReLU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.5}$$

The derivative of the function is given in Equation:

$$f^{'}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.6}$$

For most NN models with ReLU activation function, the current widely used weight initialization method is called *He* initialization. It produces random numbers with a Gaussian probability distribution. And it is usually along with the mean value of zero

and standard deviation value of $\sqrt{(2/n^l)}$, where the variable n indicates the number of neurons at layer l.

3.2.3 Error backpropagation and gradient descent

Since the deep learning network is characterized by layer depth and layer nesting, when calculating the gradient of the objective function of the deep network, it is necessary to calculate and update parameters in reverse propagation mode from deep to shallow. So the back propagation method is the concrete implementation of gradient descent method in deep network. It is a fast algorithm for computing partial derivative for gradient descent, which speeds up the process of updating parameters in networks[64, 65].

Gradient descent method It is an iterative optimization algorithm for finding the local minimum of the given loss function J containing parameters to be adjusted as shown in Figure 3.4. Two steps are performed by iteration: firstly, compute the derivative of the loss function for the starting point, namely gradient. Then, take the next step in the opposite to the calculated gradient from last point. A tuning parameter η called *learning rate* is applied in the optimization process to decide the length of each updating step.

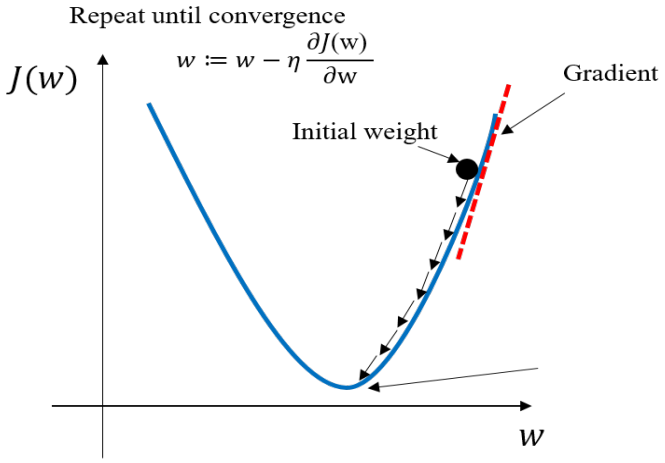


Figure 3.4: Gradient descent calculation for updating model parameters

Compared with the gradient descent method which uses all or batch data samples to perform an update iteration, Stochastic Gradient Descent(SGD)[66] only randomly selects one data each time to calculate the gradient and update the gradient. This has the advantage of being fast to update the parameters and can be used for online streaming. The new data at each time step can be directly input to the algorithm for gradient update.

A schematic diagram of the error back propagation method is shown in Figure 3.5. Meanwhile, the most commonly used loss function is mean squared error (MSE) for regression as Equation 3.7.

$$E = \frac{1}{2}\sum_{k}(t_k - y_k)^2 \tag{3.7}$$

It is an evaluation function to solve the optimization problem by the gradient descent method. In this way, weights and thresholds are modified repeatedly to minimize the error function, as the same as the loss function above.

The backpropagation algorithm calculates that the error term of a neuron at layer l, which is the sum of the weight of the error terms of all neurons at layer $l+1$ connected to this neuron. Then, multiply by the gradient of the neuron's activation function as Equation 3.8.

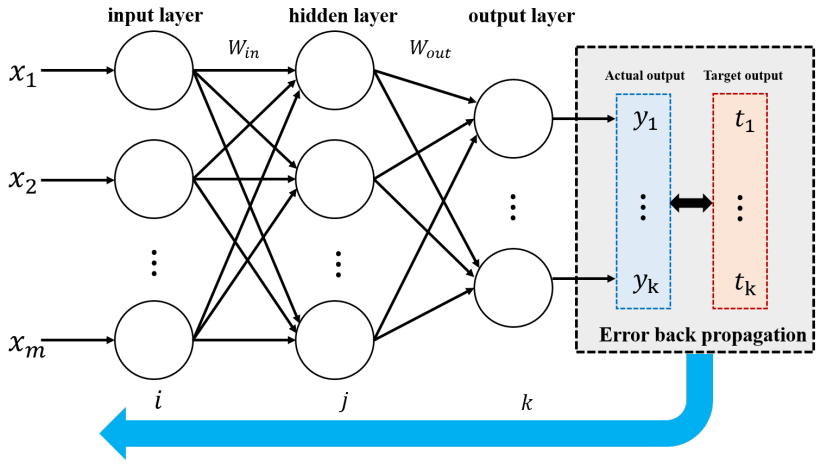


Figure 3.5: Backpropagation algorithm flow

$$\begin{aligned}\delta_k^{out} &= \frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k} \\ &= f'_k(z_k)\frac{\partial E}{\partial y_k} = f'_k(z_k)(y_k - t_k)\end{aligned} \tag{3.8}$$

The gradients of loss with respect to the hidden layer nodes δ_j^{hidden} and the front layer δ_i^{hidden} can be derived by:

$$\delta_j^{hidden} = f'_j(z_j)\sum_k \delta_k^{out} w_{jk} \tag{3.9}$$

$$\delta_i^{hidden} = f'_i(z_i)\sum_j \delta_j^{hidden} w_{ji} \tag{3.10}$$

The gradients for weights can be obtained from the following equations.

$$\frac{\partial E}{\partial w_{kj}} = \delta_k^{out} y_j \tag{3.11}$$

$$\frac{\partial E}{\partial w_{ji}} = \delta_j^{hidden} y_i \tag{3.12}$$

Therefore, the training process of feedforward neural network using error backpropagation algorithm can be summarized as the following three steps:

(1) Feedforward calculation of the net input $\mathbf{z}^{(l)}$ and activation values $\mathbf{a}^{(l)}$ of each layer until the last layer, as Equation 3.2 above;

(2) The error term $\delta^{(l)}$ of each layer is calculated by back propagation as follows.

$$\delta^{(l)} = f'_l(\mathbf{z}^{(l)}) \odot (W^{(l+1)})^\top \delta^{(l+1)} \tag{3.13}$$

(3) Calculate the partial derivatives of parameters of each layer and update the parameters, including weights and biases at each layer.

3.3 Proposed Multi-layer NN-based MIMO Temperature Control System

This section describes the composition and control principle of the proposed reference-model-based NN control method in the multi-input multi-output (MIMO) temperature control system. In this thesis, the MIMO system is simplified into a two-input two-output (2I2O) model as introduced in previous chapter. The block diagram of the proposed control system is given in Figure 3.6 and the details are described in the subsequent sections.

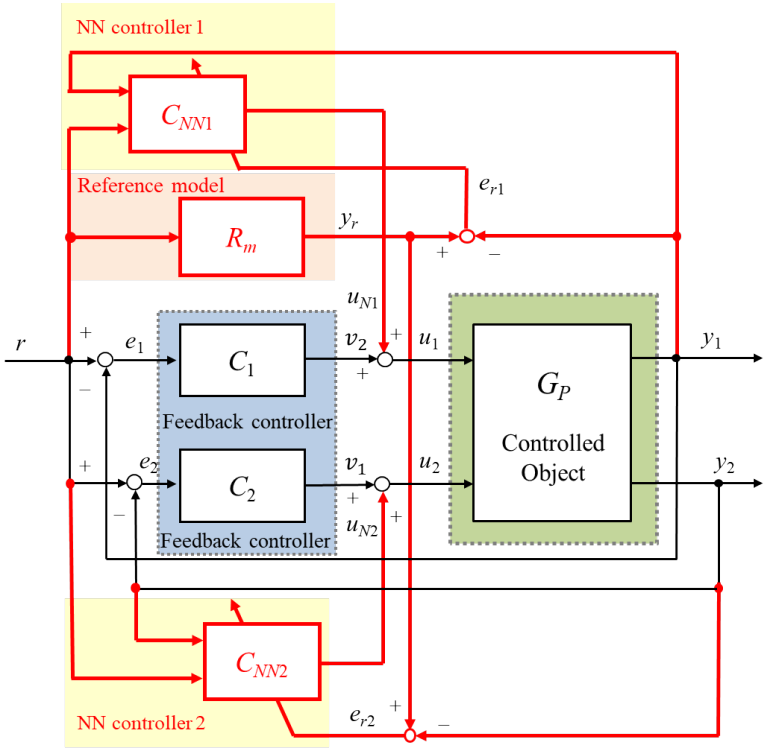


Figure 3.6: Block diagram of the proposed Multi-layer NN-based control system

As shown above, the NN-based MIMO temperature control system is mainly composed of the 2I2O controlled object G_p, two I-PD controllers C_1 and C_2 in each closed loop, and two customized neural network controllers C_{NN1} and C_{NN2} for adjusting each control input, where u_{N1} and u_{N2} are controller outputs, respectively. In particular, r indicates the input reference signal and y_1 and y_2 are the actual responses of two channels in the system, respectively. The inputs to each NN controller consist of the reference input and the actual output of each heating point/channel. Each control input of coupling channels is the sum of the output of feedback controller and the output of NN controller (v_1 and u_{N1}, v_2 and u_{N2}), respectively.

In order to eliminate the effects of time-delay, coupling interference between different heating channels and improve the response performance of all channels, a reference model R_m is specifically pre-designed for providing the reference output to each channel response. To make full use of the self-adapting learning and adjustment abilities of neural network controller, the reference model R_m is established with the maximum time-delay among the heating channels, each channel output can follow the reference model output by training the NN to minimize the error between the ideal output and the actual output, respectively.

The NN controller learns the control law by processing real-time input and output data, then adjusting the control signals to each controlled channel. Through the NN controller self-learning, each controlled channel can completely track the output of the designed reference model, so as to achieve decoupling results effectively and improve the dynamic response performance.

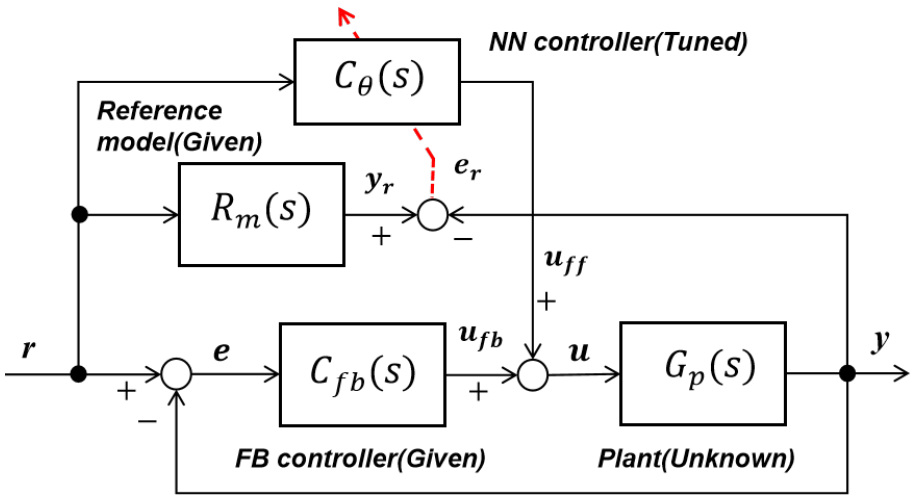


Figure 3.7: Proposed RM-based FEL control system

Such a two-degree-of-freedom control system can be simplified as Figure 3.7, it can be viewed as a variant of feedback error learning (FEL) control [67]. Each signal is

defined as follows:

$$\begin{aligned} &y = G_p(s)u, u = u_{fb} + u_{ff}, \\ &u_{fb} = C_{fb}(s)e, u_{ff} = C_\theta(s)r, \\ &e_r = y_r - y, y_r = R_m(s)r. \end{aligned} \tag{3.14}$$

In this reference-model-based feedback error learning control scheme, the controlled object can be stabilized by the designed feedback controller firstly. Then the feedforward controller tunes the control input by online feedback error learning. Here, the NN controller learns the adjustable parameters θ to reduce the error between the reference signal r and plant output y to be zero, i. e., $e_r(t) \to 0$ and $e(t) \to 0$ ($t \to \infty$). Therefore, the feedforward control signal u_{ff} needs to satisfy the following equation:

$$\begin{aligned} &y = G_p(s)u = R_m(s)r \\ &u_{ff} = \frac{R_m(s)}{G_p(s)}r \end{aligned} \tag{3.15}$$

However, the accurate parameters of the transfer function $G_p(s)$ can not be obtained in most cases, the exact feedforward controller is hard to design. Therefore, an adjustable feedforward controller is essential in the FEL control system. In many studies, it is proved that the output error can converge to zero for any given reference signal under a certain strictly positive real condition[68, 69]. The following sections detail the implementation of the proposed control system.

3.3.1 Multi-layer fully connected network

In our proposed control system, a fully-connected neural network controller with a 2-10-10-1 architecture for each channel control is as shown in Figure 3.8. Each neural network controller input NN_{in} consists of a reference input signal SV, the actual system output PV, and involved weight $\mathbb{W}$ and bias θ parameters are adjustable parameters during training each NN controller.

$$\mathbb{W}_1 = \begin{bmatrix} W_{1,1} & W_{1,2} \\ \vdots & \vdots \\ W_{10,1} & W_{10,2} \end{bmatrix}, \mathbb{W}_2 = \begin{bmatrix} W_{1,1} & \cdots & W_{1,10} \\ \vdots & \ddots & \vdots \\ W_{10,1} & \cdots & W_{10,10} \end{bmatrix}, \mathbb{W}_3 = \begin{bmatrix} W_1 \\ \vdots \\ W_{10} \end{bmatrix} \tag{3.16}$$

$$\theta_1 = \begin{bmatrix} \theta_{1,1} \\ \vdots \\ \theta_{1,10} \end{bmatrix}, \theta_2 = \begin{bmatrix} \theta_{2,1} \\ \vdots \\ \theta_{2,10} \end{bmatrix} \tag{3.17}$$

The number of the layers and neurons in each layer of the network are hyper-parameters,

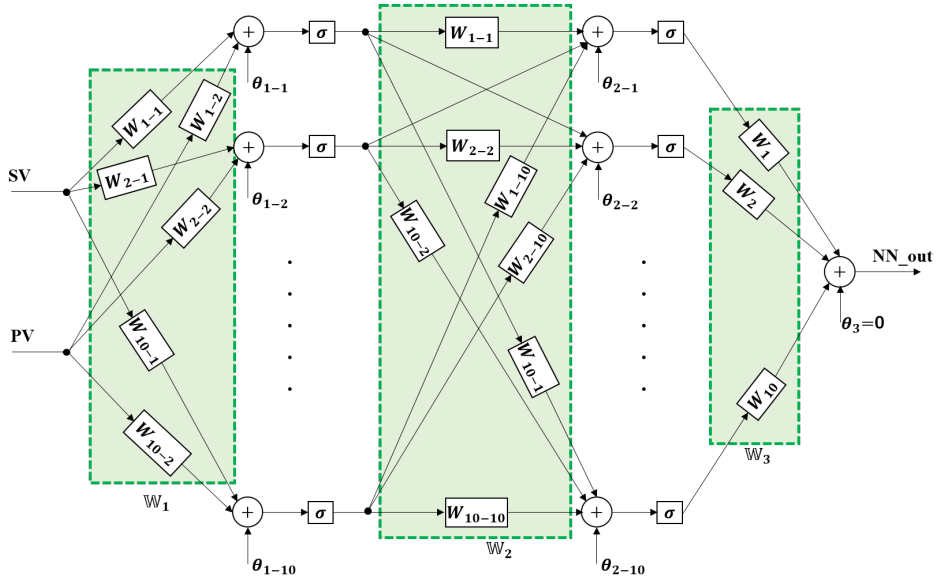


Figure 3.8: The multi-layer neural network with a specific structure

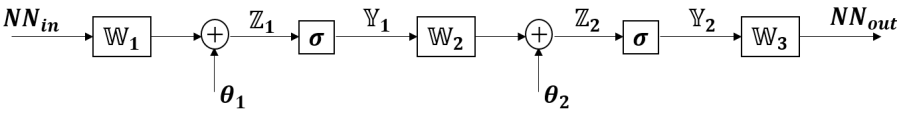


Figure 3.9: Forward propagation of a multi-layer neural network

which are usually obtained by manually adjusting parameters based on experience. Too large model will slow down the inference speed and lead to over-fitting problem, while too small network will lead to under-fitting problem. Although there is still not enough theoretical guidance to determine the optimal settings of the network model, some tips will take effect like: in most cases, one or two layers has quite enough information processing ability and add the number of layers tends to obtain a higher performance than adding more neurons in one layer. The optimal number of hidden layer neurons should be obtained by continually experimenting and comparing. It is recommended to start with a small number such as one to five layers, and reduce the number of layers and neurons if the network is over-fitting. In addition, dropout, regularization and other methods to reduce over-fitting can also be considered in practice.

Forward propagation calculation The forward propagation calculation of the multi-layer neural network is as Figure 3.9, where the parameter variables are described as Equations 3.16 and 3.17. The forward calculation of neural network propagating information by iterating the following formula.

$$\begin{aligned}
\mathbb{Y}_1 &= \sigma_1(\mathbb{W}_1 * NN_{in} + \theta_1) \\
\mathbb{Y}_2 &= \sigma_2(\mathbb{W}_2 * \mathbb{Y}_1 + \theta_2) \\
NN_{out} &= \mathbb{W}_2 * \mathbb{Y}_2
\end{aligned} \tag{3.18}$$

Adjust parameters by BP algorithm The error term δ of each layer is calculated and update weight $\mathbb{W}$ and bias θ parameters by BP algorithm as introduced above. Here, involved weight parameters $\mathbb{W}_3$, $\mathbb{W}_2$ and $\mathbb{W}_1$ are adjusted as the following equations.

$$\begin{aligned}
\delta_3 &= -(RM_{out} - PV) \\
\mathbb{W}_3 &= \mathbb{W}_3 - \alpha * \delta_3 * \mathbb{Y}_2
\end{aligned} \tag{3.19}$$

$$\begin{aligned}
\delta_2 &= \nabla\mathbb{Y}_2 * \mathbb{W}_3 * \delta_3 \\
\mathbb{W}_2 &= \mathbb{W}_2 - \alpha * \delta_2 * \mathbb{Y}_1
\end{aligned} \tag{3.20}$$

$$\begin{aligned}
\delta_1 &= \nabla\mathbb{Y}_1 * \mathbb{W}_2 * \delta_2 \\
\mathbb{W}_2 &= \mathbb{W}_1 - \alpha * \delta_1 * NN_{in}
\end{aligned} \tag{3.21}$$

Similarly, the biases θ_2 and θ_1 of each hidden layer are adjusted as Equation 3.22, where the learning rate α and β decide the learning step of adjusting the weight and bias values based on the gradient descent method during training the neural network controller. The least squares cost function calculates the total squared error between the target output (reference model output RM_{out}) and actual output (temperature output of each channel) to find a best set of parameters $\mathbb{W}$ and θ which can minimize the error.

$$\begin{aligned}
\theta_2 &= \theta_2 - \beta * \delta_2 \\
\theta_1 &= \theta_1 - \beta * \delta_1
\end{aligned} \tag{3.22}$$

It's also important to note that there is no way to ensure that the network to converge certainly. There are many factors will affect the results, such as the initial parameters, the optimization algorithm, the learning rate, etc. Considering these facts, there are some techniques that help our network to converge easier and faster, for instance, the stochastic gradient descent learning is easier to converge than batch learning and can realize online learning.

3.3.2 Reference model design

For the flexible control, the output of the system does not directly track the given input, but tracks the output trajectory of the pre-designed reference model response. This adaptive control system will adjust parameters according to the working state of the ideal

model. Commonly, the reference model is designed based on the characteristics of the controlled object, which makes the model more referential and improves the performance of the system.

Therefore, the design of the reference model (RM) used in our MIMO control system is as Equation 3.23, which takes the same transfer function form (FOPDT).

$$R_m(s) = \frac{1}{T \cdot P_{RM} \cdot s + 1} * \frac{1}{(\frac{\tau s}{2} + 1)^2}. \tag{3.23}$$

Where the time constant T is the minimum value between the identified diagonal terms P_{11} and P_{22}, and the dead time τ is the maximum value between them. The identified models of different heating channels/points mainly provide the design format for the reference model. It has desired response speed by multiplying a constant gain $P_{RM}(<1)$ to time constant, and ensures the heating channel model with smaller time delay can be consistent with the slow model that refers to the channel with largest time delay in the total heating channels of the controlled object.

Based on the identified system models P_{11} and P_{22}, the transfer function of the reference model R_m is as Equation 3.24. Here, the introduced factor is 0.01. The step responses of our defined reference models with and without the gain P_{RM} are compared with the responses of system models P_{11} and P_{22} as Figure 3.10 below.

$$R_m(s) = \frac{1}{2482.4 * 0.01s + 1} * \frac{1}{(\frac{464s}{2} + 1)^2} \tag{3.24}$$

The characteristics of the step response for the reference model can be derived by the syntax "*stepinfo*" in MATLAB, including the rise time, settling time and overshoot, etc[70]. By default, the rise time refers to the time it takes for the output of the dynamic system rises from 10% to the 90% of the final target value at the first time. The settling time is the time of the system takes from the initial value to the final steady-state value within 2% or 5% error. The overshoot is the percentage amount of the response in which the peak of the fluctuation exceeds the stable value.

In Table 3.2, the open-loop step response characteristics of the designed reference model with the model without additional gain P_{RM} are compared. As the figure shows, the pre-defined model based on the controlled object characteristics has ideal transient and steady-state response performances for the step input by adding the gain to the time constant of which is smaller than 1, it will be used to train our neural network in the following sections.

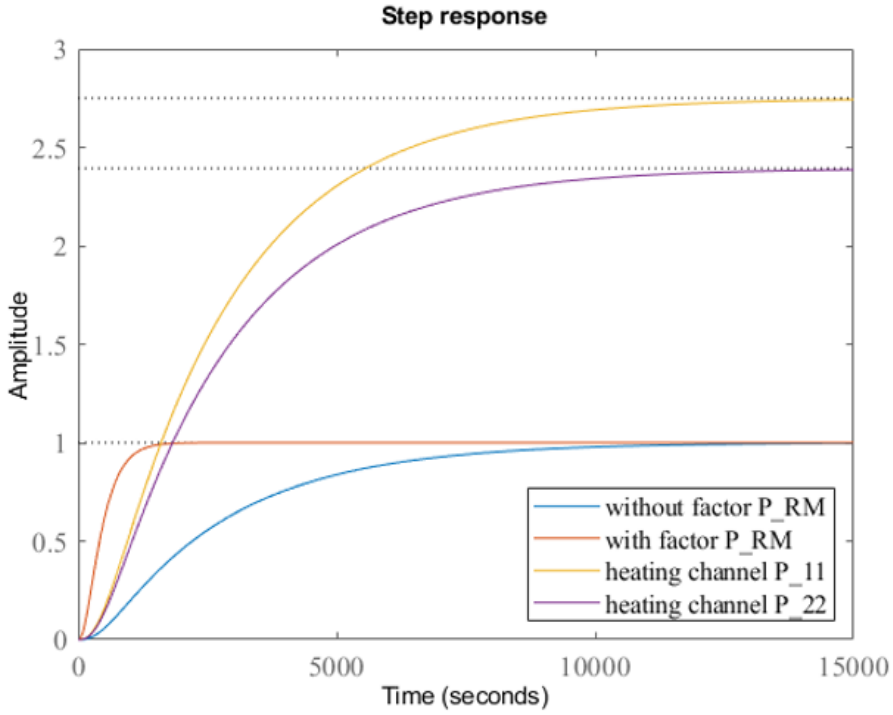


Figure 3.10: Comparison of step responses of designed reference model

Table 3.2: Step response characteristics for the designed reference model

Dynamic System	Rise Time [s]	Settling Time [s]	Overshoot [%]
With gain P_{RM}	781	1378	0
Without gain P_{RM}	5,530	10,198	0

3.4 Experiment

From the transfer functions of the identified system models P_{11} and P_{22}, the unknown parameters of each feedback controller (C_1 and C_2) are determined by the Ziegler–Nichols rules given above. The results of involved parameters in C_1 are K_{p1} = 2.5146, T_{i1} = 861.5 and T_{d1} = 215.375, respectively. The results of involved parameters in C_2 are K_{p2} = 2.7968, T_{i2} = 927.98 and T_{d2} = 231.995, respectively.

The backpropagation training uses stochastic gradient decent to update parameters for each layer in the neural network controller and initialize the training hyperparameters with $\alpha = 1 \times 10^{-10}$ and $\beta = 1 \times 10^{-5}$. These values are usually chosen by trial and error to observe the convergence of loss for better model performance. The ReLU (rectified linear unit) is used for improving the training speed and avoiding gradient vanishing problem as the increasing number of layers in the deep networks. Its derivation is easy to be obtained when computing the error gradient by back propagation.

The simulation process consists of two stages to verify the performance of the designed control system for reference signal tracking and the variation of temperature deviation. The first stage is training the neural network controller until minimizing the tracking error to find a set of model parameters. Ensure enough time for the training process, the reference signal 100°C is provided and the length of time is 15,000. The sampling time is 0.5s. At this stage, the temperature of each channel is controlled from the initial temperature to the reference 100°C. After the system response reaches the steady-state value, an amplitude with value ± 5 is added to the reference input periodically.

Therefore, at the second stage, the controlled temperature output goes up (100°C $\rightarrow$ 105°C) and down (105°C $\rightarrow$ 100°C) in a regular cycle, the full tracking time of one cycle is 10,000s containing temperature up and down stages, respectively. For better observation of the response performance of tracking changed reference signals, the offset value is 100°C, and then verify the effectiveness of the trained NN controller in our coupling, time-delay temperature control system by computing the characteristics of the controlled system.

3.4.1 Simulation results

Follow the simulation settings, Figure 3.11 illustrates the results of temperature response for changed target inputs from 100 °C to 105 °C and from 105 °C to 100 °C, respectively. And these results based on the proposed control method are quantitatively compared with the control performance of the conventional I-PD control method as discussed above, which is divided into two control modes with feedforward gain K_f equals to 0 (slow) and 1 (fast), respectively.

For the target value tracking in both rising and falling directions, the proposed NN-based control system ensures each channel output can follow the reference model output after training each NN model effectively and quickly as the curves shown in Figure 3.11(b). In the multiple cycles for the changed reference tracking, the first complete rise and fall temperature response are nearly identical with the last cycle. It indicates that each neural network controller quickly completes the parameter adjustment in the training stage, and the well-trained NN model effectively learns the control law, so that it can control each channel to track the ideal reference model output rapidly. The response characteristics describing the temperature rise tracking performance of each channel are computed and listed in Table 3.3, which are compared with the I-PD (fast mode) and I-PD (slow mode) control responses.

For each channel temperature output, the rise time it takes for the proposed NN-based control is 753s and 788s, respectively, compared with the I-PD (K_f=0) control

Table 3.3: Comparison of simulation results for time response (100 °C to 105 °C).

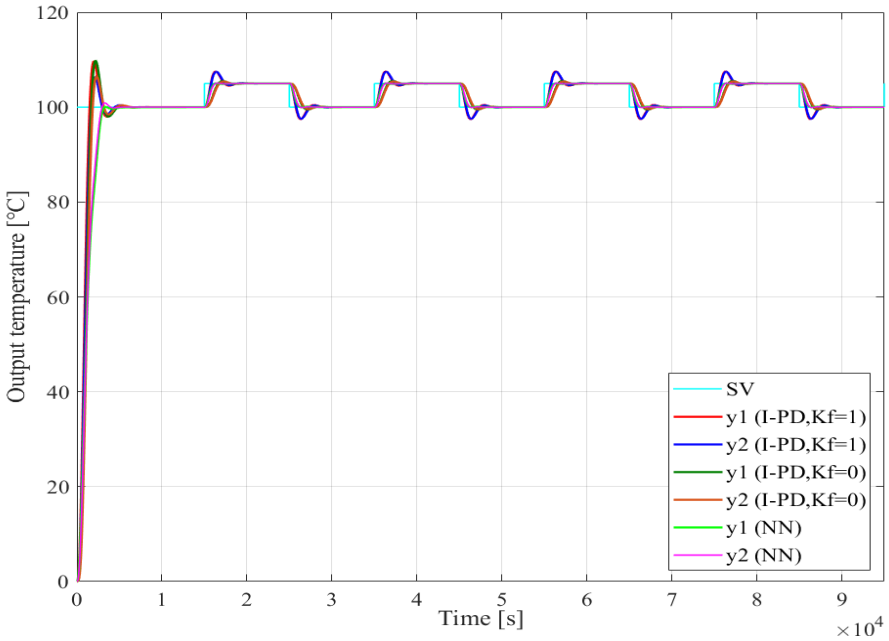
Characteristics	**Ref**	**CH1 (K_f=1)**	**CH2 (K_f=1)**	**CH1 (K_f=0)**	**CH2 (K_f=0)**
Rise Time [s]	781(100%)	459(58.8%)	482(57.9%)	931(119%)	990(126.8%)
Settling Time [s]	1378(100%)	3663(265.8%)	3554(257.9%)	3634(263.7%)	3144(228.2%)
Overshoot [%]	0	48.6	47.5	10.1	8.0
Characteristics	**Ref**	**CH1 (NN)**	**CH2 (NN)**		
Rise Time [s]	781(100%)	753(96.4%)	788(100.1%)	-	-
Settling Time [s]	1378(100%)	2122(153%)	2178(158.1%)	-	-
Overshoot [%]	0	0	0	-	-

that takes 931s and 990s, respectively. More than 20% of the time is shorten for each channel response, respectively. Although the time takes in I-PD (K_f=1) is 459s (CH1) and 482s (CH2), respectively, which is smaller than the NN-based control, there are also obvious overshoots of 48.6% (CH1) and 47.5% (CH2) over the final steady-state value, respectively. Compared with the rise time of the ref-model output, the rise time of NN-based system outputs,

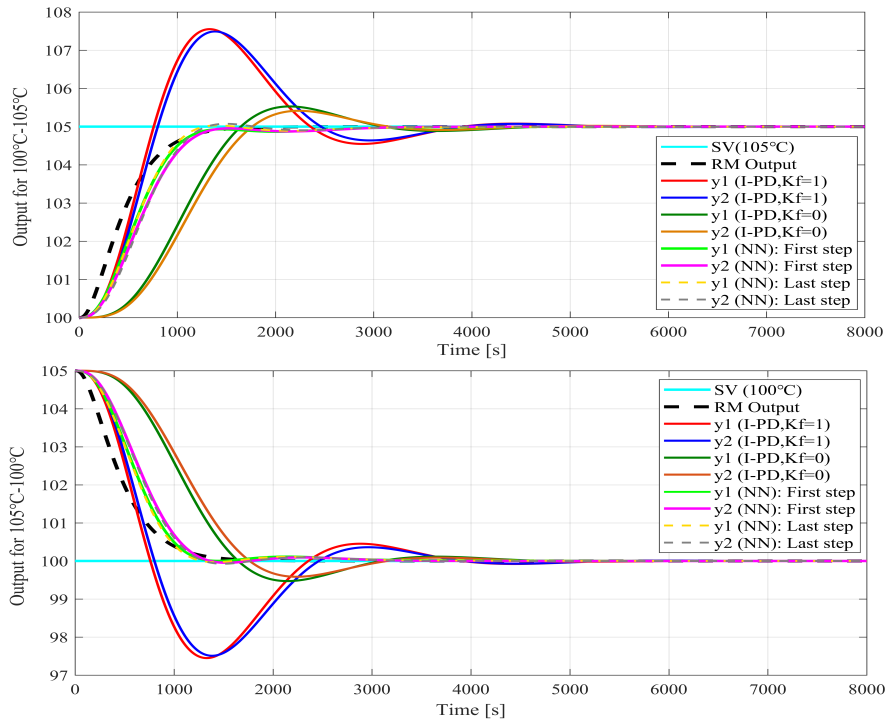
The settling time (2%) it takes for the proposed NN-based control is 2,122s (CH1) and 2,178s (CH2), respectively, which is only 57.9% and 61.7% of the results compared with I-PD (K_f=1) control taking 3,663s (CH1) and 3,554s (CH2), respectively. The similar results can be obtained when comparing the settling time between NN-based and I-PD (K_f=0), which is only 58.4% of the time for the CH1 response and 69.3% for the CH2 response. The overshoots of each output in I-PD (K_f=0) are about 10% (CH1) and 8% (CH2) of the steady-state value based on the offset, respectively. These results indicate that the proposed NN-based multi-point control method cannot only improve the transient response speed but also almost has no overshoot during the reference tracking from the initial value to the steady-state value in the rising direction.

In the temperature falling direction, the response characteristic indictors of each channel control output are computed and listed in Table 3.4, which are also compared with the I-PD (K_f=0) and I-PD (K_f=0) control responses.

Compare the results of each channel response output, the rise time it takes for the proposed NN-based control is 2141s and 2065s, respectively, which is only 58% (CH1) and 65.8% (CH2) of the corresponding heating channel output of I-PD (K_f=0), respectively. Although the I-PD (K_f=1) control response takes shorter rise time, there are significant overshoots in both heating channels, which are over 45% of the final steady-state value based on the offset, respectively. The settling time it takes for the proposed NN-based control, which is 2,141s (CH1) and 2,065s (CH2), respectively, it reduces about 41.6%



(a) Full cycles of time response for the control system.



(b) One complete cycle of the reference tracking.

Figure 3.11: Comparison of simulation results for reference tracking performances. (a) Time response of the whole control cycles. (b) Temperature output response in the rising direction (100 °C → 105 °C) and falling direction (105 °C → 100 °C), respectively.

and 41.9% by comparing with I-PD (K_f=1) control taking 3,669s (CH1) and 3,552s (CH2), respectively. The similar results can be obtained when comparing the settling time between NN-based and I-PD (K_f = 0), which is shorten by about 42% and 34.2% for each channel response output. And the overshoots of each output in I-PD (K_f=0)

Table 3.4: Comparison of simulation results for time response (105 °C to 100 °C).

Characteristics	**Ref**	**CH1 (K_f=1)**	**CH2 (K_f=1)**	**CH1 (K_f=0)**	**CH2 (K_f=0)**
Rise Time [s]	781(100%)	459(58.8%)	482(61.7%)	931(119.2%)	990(126.8%)
Settling Time [s]	1378(100%)	3669(266.3%)	3552(257.8%)	3689(267.7%)	3136(227.6%)
Overshoot [%]	0	48.6	47.5	10.1	7.9
Characteristics	**Ref**	**CH1 (NN)**	**CH2 (NN)**		
Rise Time [s]	781(100%)	767(98.2%)	772(98.8%)	-	-
Settling Time [s]	1378(100%)	2141(155.3%)	2065(149.9%)	-	-
Overshoot [%]	0	0	0	-	-

are about 10.1% (CH1) and 7.9% (CH2) of the steady-state value based on the offset, respectively. From the above results, the proposed control method cannot only improve the transient response speed but also almost has no overshoot during the reference tracking from the initial value to the steady-state value in both rising and falling directions of the temperature changing curves.

For verifying different reference signal tracking performances, as shown in Figure 3.12 and 3.13, the proposed NN-based controller regulates the controlled MIMO system output by tracking the same reference output to overcome the coupling effects, short the dynamic response time and achieve better transient and steady performances.

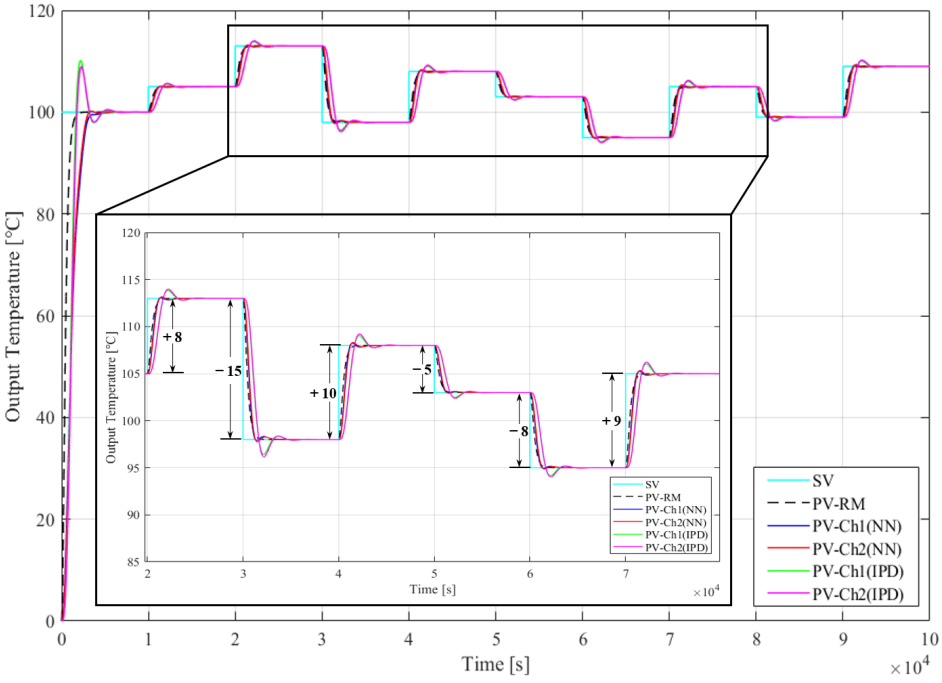


Figure 3.12: Compare the tracking responses of different reference signals(from 100°C)

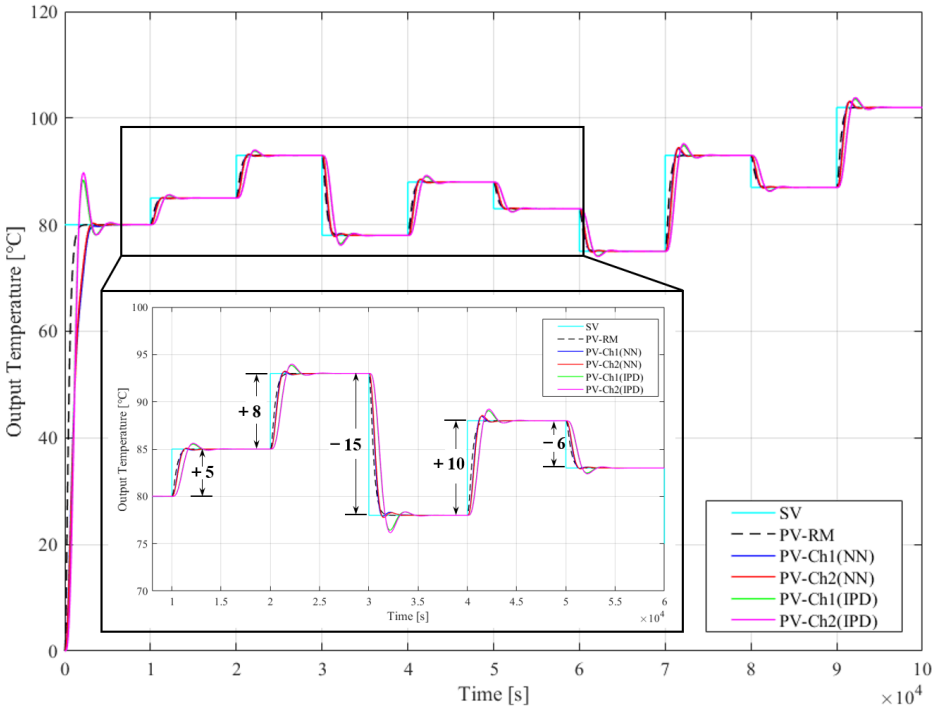


Figure 3.13: Compare the tracking responses of different reference signals(from 80°C)

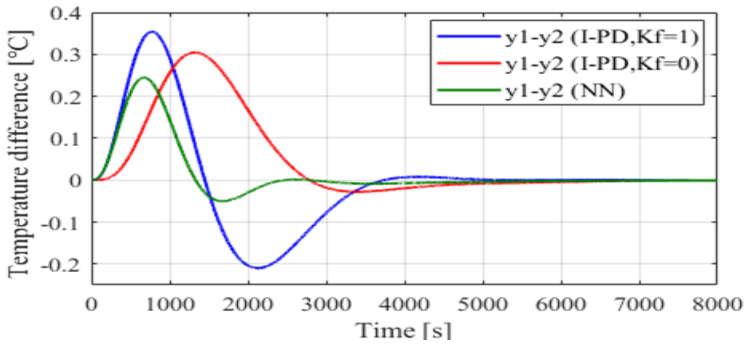
Temperature difference

The following figures illustrate the temperature difference during the reference tracking in both rising and falling directions.

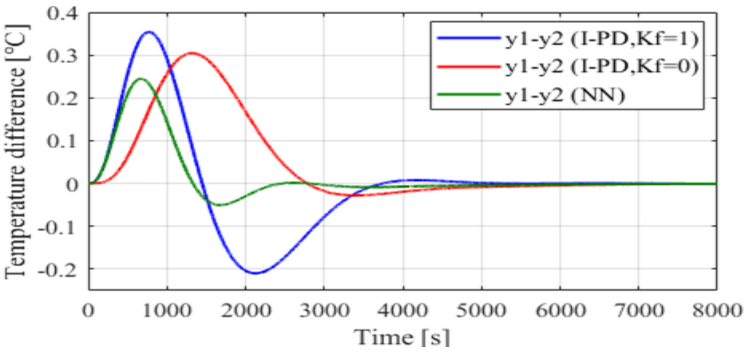
Compute the peak values which represent the maximum difference between channel temperature outputs, the results are 0.23°C (4.6% of $|y_{final} - y_{initial}|$) of the NN-based control, 0.35°C (7% of $|y_{final} - y_{initial}|$) of the I-PD ($K_f$ = 1) control, and 0.31°C (6.2% of $|y_{final} - y_{initial}|$) of the I-PD ($K_f$ = 0) control, respectively. The NN-based control reduced by 2.4% and 1.6% of the maximum differences in temperature, compared to the I-PD controller with the feedforward gain K_f equals to 1 and 0 in the feedforward loop, respectively. The time it takes for the fluctuation of temperature difference from the peak values to the steady-state value also reflects the control system performance for the changed input signals. The NN-based control system takes about 2,400s to control the difference in temperature to zero, I-PD (K_f = 1) takes 5,300s and I-PD (K_f = 0) takes 5,800s at the same condition, respectively. By comparison, the NN-based control shortens the time by over 50% of that in the other two control modes.

Mutual coupling interference rejection

To analyze the mutual coupling interference between heating channels and how to be rejected in different control systems, plot the reference tracking response curve of both



(a) Temperature difference changing curves(100 °C→105 °C).



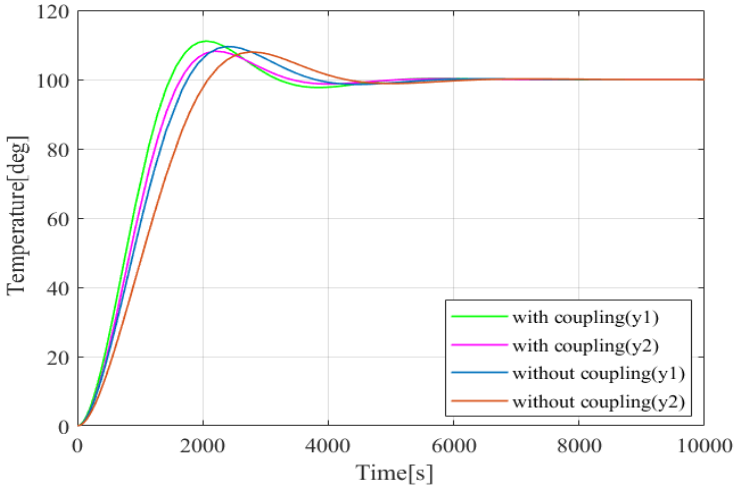
(b) Temperature difference changing curves(105 °C→100 °C).

Figure 3.14: Simulation results comparison for temperature differences between CH1 and CH2. (a) Rising period. (b) Falling period.

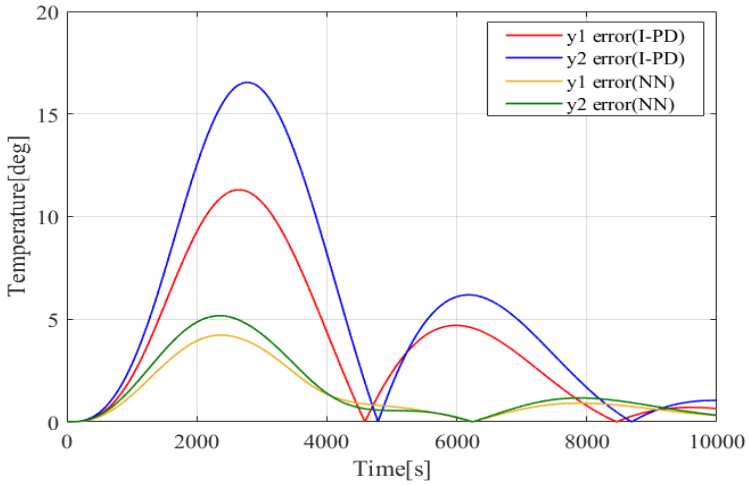
I-PD control systems with coupling terms (P_{12} and P_{21}) and without, respectively.

Figure 3.15(a) is comparison result of output time responses with and without coupling terms, Figure 3.15(b) is the curve of absolute difference between the outputs of coupling channels. To compare the results of tracking the target input 100°C at the same conditions as Figure 3.15(a), the systems with coupling has overshoots about 11.2% (CH1) and 8.4% (CH2) of the reference in each channel output, respectively. The system without couplings has overshoots about 9.4% (CH1) and 7.8% (CH2) of the reference in each channel output, respectively. Because of the existence of the interference between different channels, the response performance of each channel output is obviously changed.

Furthermore, compare the absolute values of the temperature difference between two channels in control systems with coupling terms as Figure 3.15(b). It's obvious that there are obvious tracking differences in both channel outputs of the I-PD control systems with coupling terms. On the contrary, the proposed NN-based decreased the maximum values to 11.4°C (CH1) and 16.6°C (CH2), respectively. The simulation results indicate that the proposed method can effectively reduce the mutual coupling inference without designing additional decoupling links.



(a) Time response comparison.



(b) Absolute temperature difference.

Figure 3.15: Mutual coupling interference in the control system.

Comparison of disturbance rejection performance

For testing the disturbance rejection performance in the rising and falling directions, similarly, a reference input of 100°C is set at first. After the control system output is controlled to stay the steady-state value in 10,000s, then the target value is set as 105°C lasting for 10,000s. In this stage, add a pulse disturbance signal with an amplitude of -20 and pulse width of 100s to the control system in 5,000s after the output keeps at the steady-state value of 105. In the negative direction for the temperature output changes from 105 to 100 which lasts for 10,000s, a same pulse disturbance signal with an amplitude of 20 is added in 5,000s after the output keeps at the steady-state value of 100. The system output goes up to 105 from the initial value and then down to 100 in one cycle for 20,000s.

To verify the disturbance rejection performance of the NN-based control systems, plot the reference tracking curves under the same simulation conditions as Figure 3.16(a).

The dynamic characteristics of the NN-based MIMO control systems are calculated in the Table 3.5 and Table 3.6 in both directions, respectively, where the I-PD control results are as a baseline for comparison. By comparing the temperature drop and overshoot after applying the disturbance signal, the NN-based control system shows the better performance in suppressing the disturbance.

Table 3.5: Disturbance rejection performance for 100°C-105°C

Characteristics	Ch1(IPD)	Ch2(IPD)	Ch1(NN)	Ch2(NN)
Settling Time T_s [s]	4235(100%)	4144(100%)	2788(65.8%)	2715(65.5%)
Overshoot [°C]	1.6(100%)	1.5(100%)	1.3(81.2%)	1.2(80%)
Drop [°C]	2.1(100%)	2.2(100%)	1.9(90.4%)	1.8(81.8%)

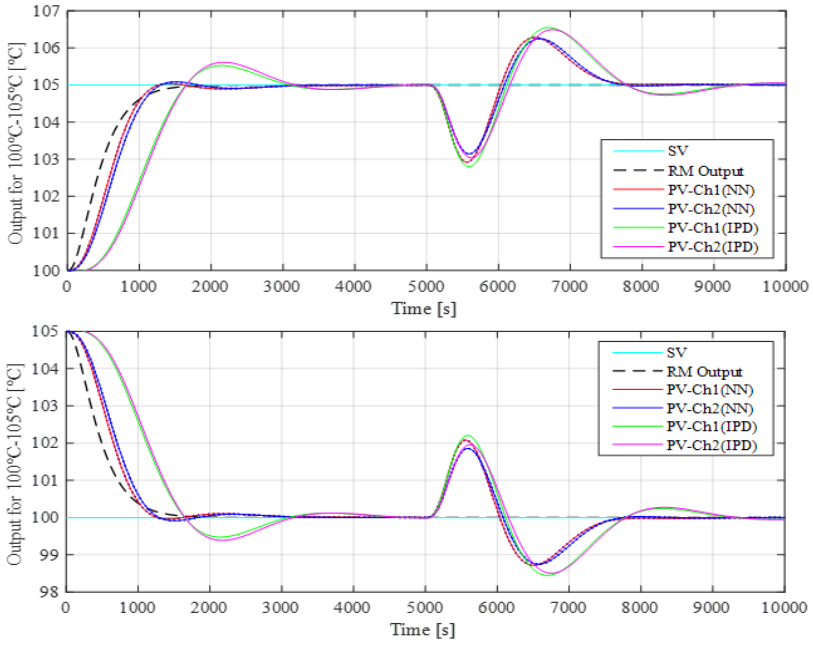
Table 3.6: Disturbance rejection performance for 105°C-100°C

Characteristics	Ch1(IPD)	Ch2(IPD)	Ch1(NN)	Ch2(NN)
Settling Time T_s [s]	4221(100%)	4156(100%)	2795(66.2%)	2788(67.1%)
Overshoot [°C]	2.2(100%)	1.9(100%)	2.0(90.9%)	1.8(94.7%)
Drop [°C]	1.6(100%)	1.5(100%)	1.3(81.3%)	1.2(80%)

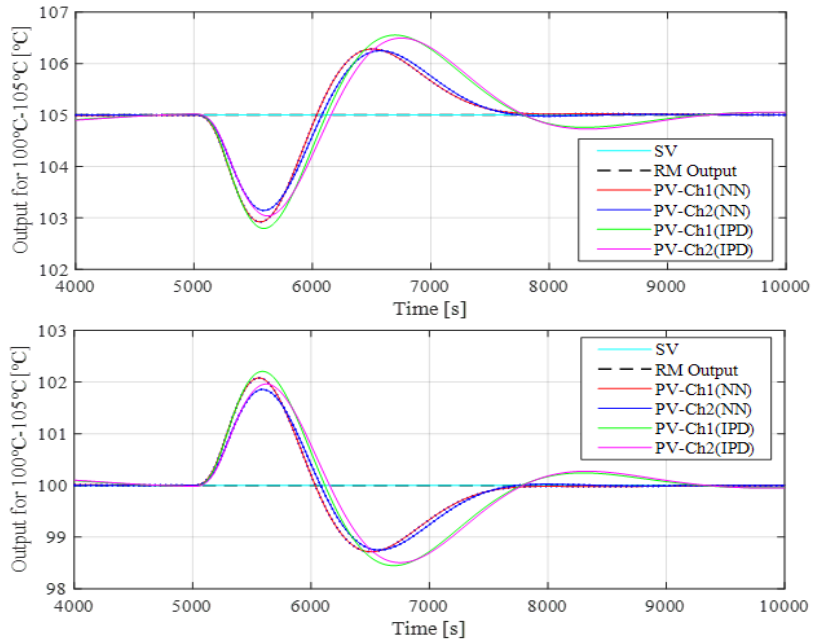
3.4.2 Experimental results: reference tracking

In the simulation experiment, there is no power devices and drive circuits such as the SSR(Solid state relay). In practice, however, the semiconductor manufacturing equipment is affected by such delays, nonlinearity and noise, so the actual experiments are carried out simultaneously. To validate the validity and reliability of the designed control systems, perform the experiments on the implemented experimental platform as introduced in the previous chapter. The detailed experimental conditions are given below.

The ambient temperature was 28°C and the sampling time is 0.1s. Similar to the simulation settings, the reference input is set to 100°C in 10,000s, and the NN controllers finish the model parameter training in this length of time. In each sampling time, the training data for the NN controller include the input signal contains the reference r and the actual channel outputs (y_1 and y_1), the teaching signals contain the error values (e_{r1} and e_{r2}) between the actual outputs and the reference model output, and the real-time control outputs (y_1 and y_2), respectively. Then an amplitude of $\pm$5°C input signal is added in one cycle, the channel outputs follow the given reference changing which contains the rising stage of 8,000s and the falling stage of 8,000s. It means that after the output temperature reaches the offset value 100°C, both channel outputs are controlled



(a) Temperature output in one cycle

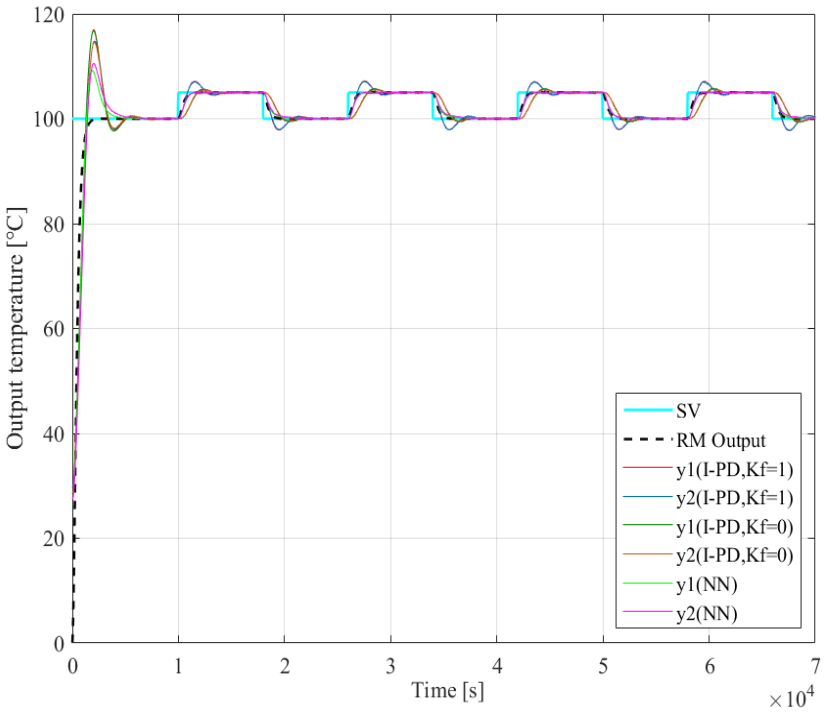


(b) Local enlarged image of disturbance response

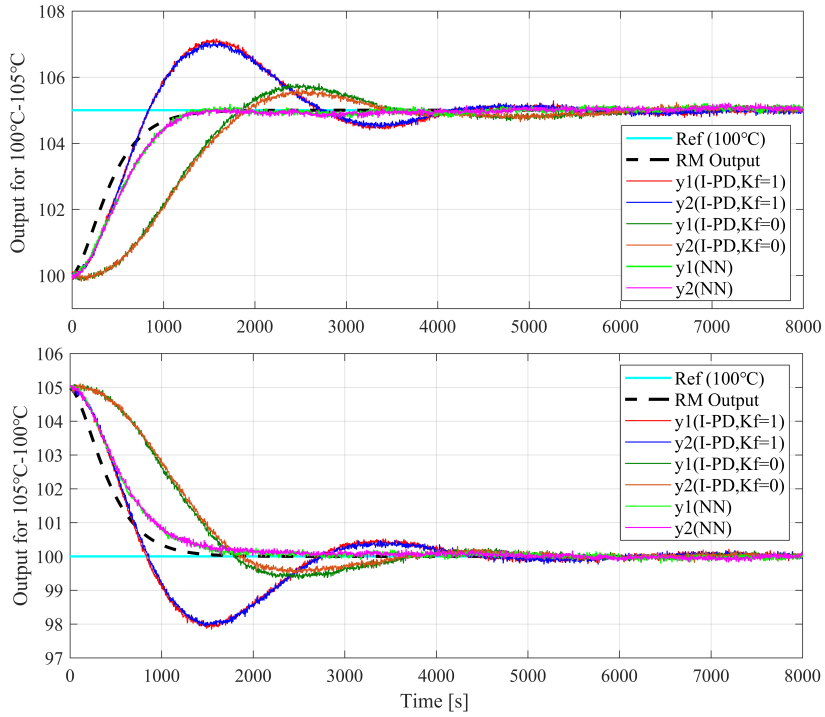
Figure 3.16: Disturbance response of NN-based MIMO control system in time domain

varied within the ranges of [100, 105] and [105, 100] periodically. The ascent and descent phases are viewed as the testing phases of the control system. For verifying the effectiveness and learning efficiency of the NN controller, compare the measured ascent and descent temperature outputs with the experimental results for the conventional I-PD control systems in the same conditions.

Figure 3.17(a) illustrates the reference tracking results for changed input signals in



(a) Full cycles of time response for the control system.



(b) One complete cycle of the reference tracking.

Figure 3.17: Comparison of experimental results for reference tracking performances. (**a**) Time response of the whole control cycles. (**b**) Temperature output response in the rising direction (100 °C → 105 °C) and falling direction (105 °C → 100 °C), respectively.

multiple cycles, and (b) shows enlarged comparison results for both channel outputs in

ascent and descent direction. In both directions of the output temperature changes, the I-PD control with the feedforward gain 1 response rapidly, but is along with obvious large overshoots. The I-PD control with the feedforward gain 0 gives flat response, which is slower than the NN-based control without the overshoot. The proposed NN-based control system follows the reference model response rapidly and stably as desired.

The measured time series data of each channel output are preprocessed to suppress the noise by a second-order Butterworth digital low-pass filter, which has a cut-off frequency of 0.05 Hz. The sampling rate of the filter is 10Hz. In MATLAB, the final steady-state values are set as 105 and 100 for the corresponding stages manually to compute the response characteristic values, respectively, for processing the data with noise that cannot reflect the true final value. Define the threshold percentage of the settling time is 5%, which represents the system response converges to the steady-state value within the acceptable error percentage. To compare the rise time, settling time and overshoots of different control systems in the response [$y_{initial}$, y_{final}], where the initial offset is 100 in the rising direction which corresponds to y_{final}=105, and 105 in the falling direction which corresponds to y_{final}=100, respectively.

To better view and compare the tracking performances between the proposed NN-based control system and the traditional control systems, the corresponding evaluation indexes of dynamic characteristics are computed as listed in Table 3.7.

Table 3.7: Comparison of results for reference tracking in rising direction.

Characteristics	**Ref**	**CH1 (K_f=1)**	**CH2 (K_f=1)**	**CH1 (K_f=0)**	**CH2 (K_f=0)**
Rise Time [s]	781(100%)	535(68.5%s)	558(71.4%)	947(121.3%)	973(124.6%)
Settling Time [s]	1378(100%)	3824(277.5%)	3719(269.9%)	3414(247.8%)	3218(233.5%)
Overshoot [%]	0	2.01(40.1%)	1.92(38.4%)	0.69(13.9%)	0.53(10.5%)
Characteristics	**Ref**	**CH1 (NN)**	**CH2 (NN)**		
Rise Time [s]	781(100%)	812(104%)	821(105.1%)	-	-
Settling Time [s]	1378(100%)	2217(161.8%)	2205(160%)	-	-
Overshoot [%]	0	0.06(1.3%)	0.08(1.7%)	-	-

Compare to the reference model output, the numerical experiment results show that the controlled coupling channel outputs follow the same model output as much as possible, which bring quicker and smooth tracking curve than the traditional control, though there are still errors between them. Form the comparison results between the NN-based control and the traditional control, the improved percentage values in the rise time for the proposed NN-based control are 14.6% and 18.3% and in the settling time reach 38.6% and 37%, respectively, when compare with I-PD with gain K_f=0. And compare with

the I-PD control with gain K_f=1, the NN-based control has an improvement of over 40% in settling time and the overshoot is reduced to 0.06(1.3%) and 0.08(1.7%), respectively. The overshoots of the traditional control systems in fast control mode are about 40.1 and 38.4, in the slow control mode are 13.9 and 10.5 of the reference response ($|y_{final} - y_{init}|$) expressed as a percentage, respectively. From the ascent response characteristics, the NN-based control system provides better response characteristics not only in the rise time and settling time, but also the overshoots are reduced obviously.

Table 3.8: Comparison results for reference tracking in falling direction.

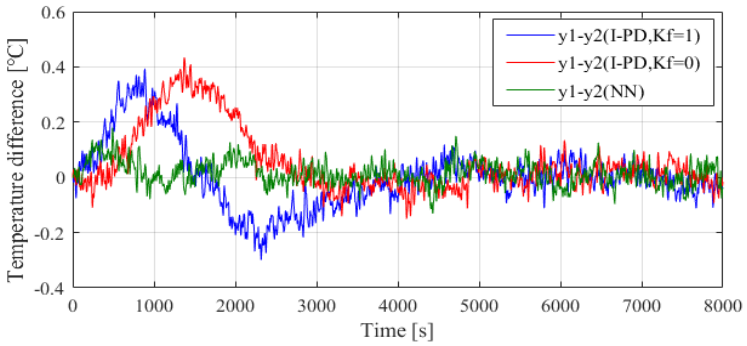
Characteristics	**Ref**	**CH1 (K_f=1)**	**CH2 (K_f=1)**	**CH1 (K_f=0)**	**CH2 (K_f=0)**
Rise Time [s]	781(100%)	519(66.5%)	536(103.3%)	1027(131.5%)	1093(139.9%)
Settling Time [s]	1378(100%)	3863(280.3%)	3870(280.8%)	3493(253.5%)	3325(241.3%)
Overshoot [%]	0	2.14(42.7%)	2.07(41.4%)	0.62(12.3 %)	0.48(9.7%)
Characteristics	**Ref**	**CH1 (NN)**	**CH2 (NN)**		
Rise Time [s]	781(100%)	877(112.3%)	894(114.5%)	-	-
Settling Time [s]	1378(100%)	2146(155.7%)	2093(152.9%)	-	-
Overshoot [%]	0	0.11(2.1%)	0.12(2.4%)	-	-

Similarly, the numerical experiment results for the falling direction are compared between the NN-based control and I-PD control in percentage and given in Table 3.8. The NN-based control system provides better response characteristics not only in the rise time and settling time are shortened, but also the overshoot occurs in the reference tracking are decreased to 2.1% and 2.4% of the reference response, respectively. Obviously, the reference tracking of both channels in the proposed system has faster speed and smaller vibration compared to the results with the traditional methods. Compare to the reference model output, the numerical experiment results show that the coupling channel outputs follow the same model output as much as possible, which bring quicker and smooth tracking curve than the traditional control, though there are still errors between them.

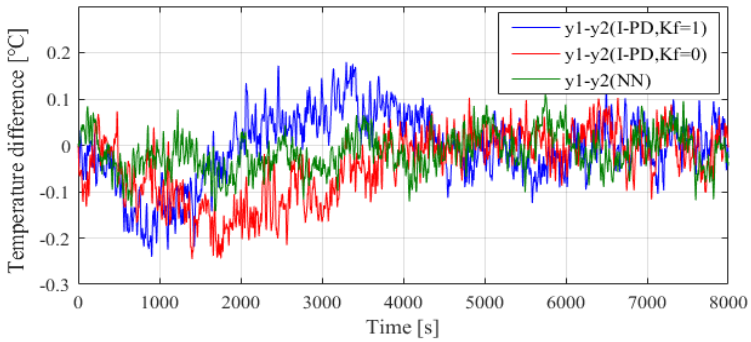
3.4.3 Experimental results: temperature difference

Figure 3.18(a) and (b) respectively show the temperature difference variation curves of each channel output (between Ch1 and Ch2) during the reference tracking, including the increasing and decreasing stages.

In the rising direction of temperature output, compute the response values of the temperature variations in the proposed control system and in the traditional control systems with fast and slow response modes, respectively. The absolute peak value of the



(a) Temperature difference changing curves in the rising stage.



(b) Temperature difference changing curves in the falling stage.

Figure 3.18: Temperature difference in the reference tracking stages.

temperature difference is 0.15°C(3% of the reference) in the NN-based control system, 0.39°C(7.8%) in the I-PD (fast mode) control and 0.43°C(8.6%) in the I-PD (slow mode) control system, respectively. The corresponding time when the difference decreases to the steady-state value zero is calculated, and the results are 2,300s (NN-based), 3,900s (I-PD fast mode) and 3,600s (I-PD slow mode), respectively. By comparison of the shorten percentage in the time, the NN-based control reaches 41 and 36 percentage reduction based on the time two modes of I-PD control takes, respectively. Similar to tracking the reference in the rising direction, the peak value of the absolute temperature difference in the falling direction is 0.14°C (2.8% of the reference) in the NN-based control system, 0.24°C (4.8%) in the I-PD (fast mode) control and 0.25°C (5%) in the I-PD (slow mode) control system, respectively. The corresponding time when the difference decreases to the steady-state value zero is 3,300s (NN-based), 4,400s (I-PD fast mode) and 4,100s (I-PD slow mode), respectively. The accuracy values during the changed reference tracking stages are within about $\pm$0.1°C.

From the experimental results in response characteristics and temperature difference variation, the proposed method makes the controlled channel outputs follow the reference model output to obtaining quick and stable response results, and the temperature

differences can be rapidly reduced to zero for achieving the surface temperature uniformity. The proposed NN-based multi-channel temperature control method can eliminate the mutual interference and improve the tracking performance of the temperature control system by comparing the experimental results of transient response and steady response characteristics.

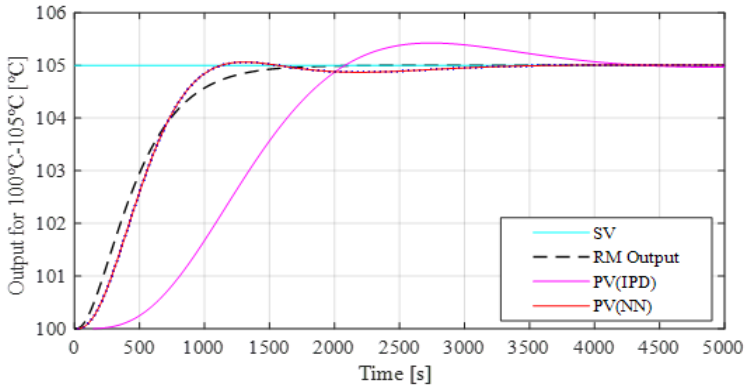
3.4.4 Application and verification to various plants

As introduced above, time delays exist in different engineering systems. If a process system with time delay L and the inertial time constant T, the ratio of L/T can indicate the difficulty in controlling the system with time delay. Because of the delay in the system, the stability of the system will be reduced. The longer the delay, the more unstable the system becomes. It is generally considered that the system is a large time-delay system if the ratio of pure delay time to the system time constant is larger than 0.5. Here, in order to verify whether the proposed control strategy based on neural network can adapt to the changes of object parameters, the following controlled objects with different ratios of L/T in Table3.9 are test by the same control situation. The default controlled plant ① in Table3.9 is the SISO aluminum block introduced in the previous section. The I-PD parameters corresponding to different controlled objects are also computed. For different plants with delay time, the initial learning time of the NN controller needs to consider the length of the delay time to ensure enough sample data which can cover the open-loop system response characteristics for the input signal as much as possible. For example, the plant with delay-time L=1198 in Table as below, the settling time is about 11286s, the learning time of NN controller should be larger than it, such as 12000s which containing 24000 training data when sampling time is 0.5s.

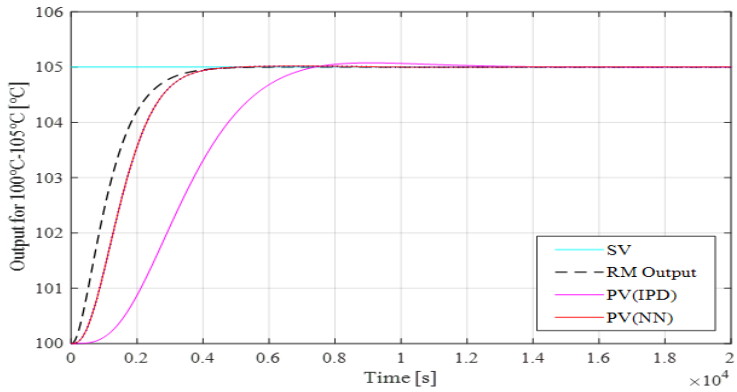
The simulation results of the same reference tracking (amplitude of 5 degree) are shown in Figure3.19, where only the positive direction results are plotted and compared, respectively. The corresponding dynamic response characteristics of the listed plants are calculated and compared with the traditional I-PD method, as given in Table3.10 below. From the results for the plants with different ratios of L/T, the proposed NN-based method can control the time-delay plants to track the reference model output, thus shortening the transient response time and quickly reaching the steady-state.

Table 3.9: Parameter setting of I-PD controller for plants with different L/T ratios

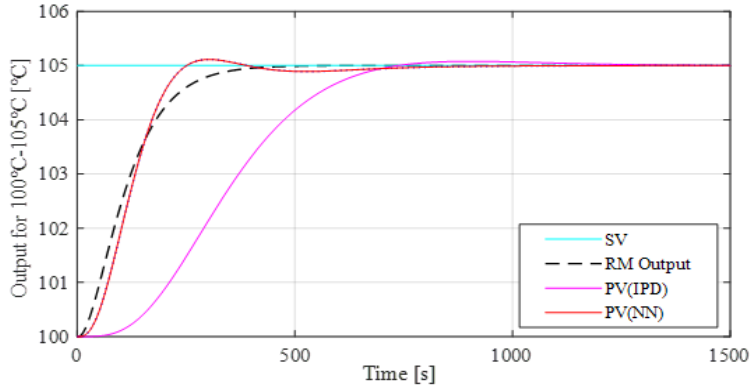
No	L/T	Parameters	Kp	Ti	Td
①	0.2	K=2.9, T=2395, L=479	2.07	958	239.5
②	0.5	K=2.9, T=2395, L=1198	0.86	2395	598.8
③	0.5	K=2.9, T=239, L=120	0.86	240	60
④	0.8	K=2.9, T=2395, L=1912	0.54	3832	958
⑤	0.8	K=2.9, T=239, L=192	0.52	384	96
⑥	2.0	K=2.9, T=239, L=478	0.21	956	239



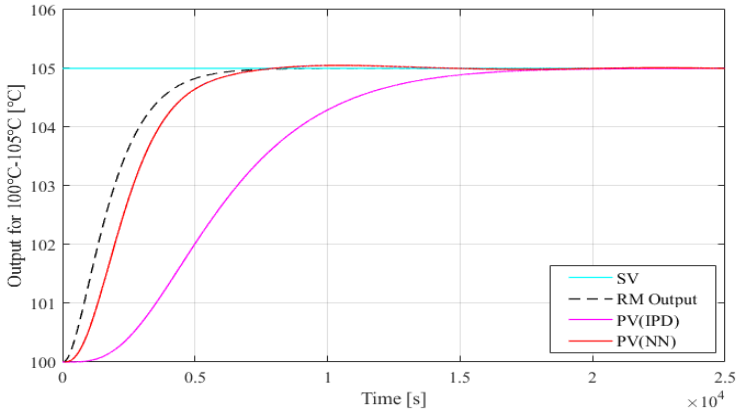
(a) Reference tracking results of plant ①.



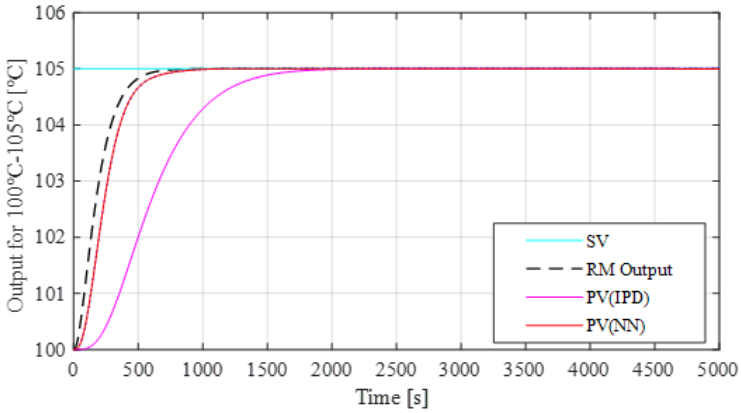
(b) Reference tracking results of plant ②.



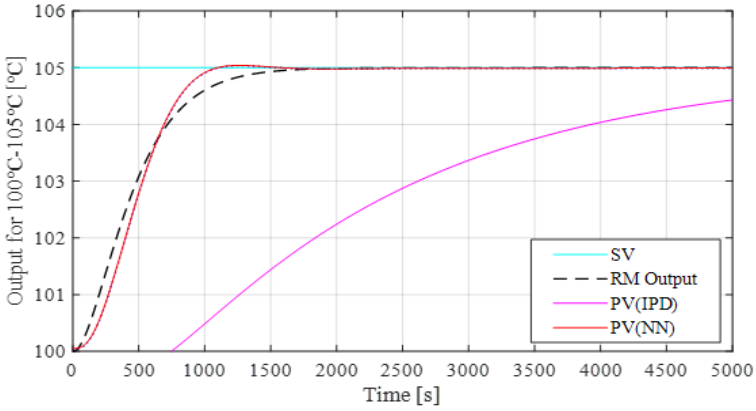
(c) Reference tracking results of plant ③.



(d) Reference tracking results of plant ④.



(e) Reference tracking results of plant ⑤.



(f) Reference tracking results of plant ⑥.

Figure 3.19: Reference tracking performances of varying plant parameters.

Table 3.10: Reference tracking characteristics of plants with different L/T ratios

I-PD	①	②	③	④	⑤	⑥
Rise Time [s]	1204(100%)	3952(100%)	393(100%)	8349(100%)	837(100%)	4494(100%)
Settling Time [s]	3876(100%)	6770(100%)	675(100%)	15231(100%)	1529(100%)	NaN
Overshoot[%]	8.4(100%)	1.5(100%)	1.5(100%)	0.4(100%)	0.1(100%)	0
NN	①	②	③	④	⑤	⑥
Rise Time [s]	652(54.2%)	2164(54.7%)	147(37.4%)	3594(43%)	350(41.8%)	655(100%)
Settling Time [s]	2590(66.8%)	3785(55.9%)	579(85.7%)	6471(42.5%)	667(43.6%)	2329
Overshoot[%]	1.2(14.2%)	0.32(21.3%)	1.8(120%)	0.7(175%)	0.1(100%)	0

3.5 Conclusion

In this chapter, to realize the uniformly distributed temperature on the coupling channels of the controlled object, a multi-layer neural network controller-based multi-channel temperature control system is designed, which controls the system output to follow the desired output response by introducing a pre-designed reference model. To train the NN controller by minimizing the error between the provided ideal output and the actual output of each channel, the NN controller of each channel can quickly get the optimal result to control each channel output to be consistent with the ideal output.

By quantitative comparison of the simulation and experimental results in both temperature rising and falling directions, it was found that the proposed NN-based control method can effectively improve the transient response and steady-state response, such as the settling time of each channel was reduced by over 40% compared to the conventional method and the overshoot of each channel was decreased below 2% of the reference in the changed temperature tracking response. And for temperature differences caused by the mutual interference, the NN-based control reduced the differences between channel outputs quickly and smoothly, the time spent is reduced by 41 and 36 percentage compared to the fast mode and slow mode control in the I-PD control, respectively. Simulation and experimental analysis results show the NN controller in the proposed method can rapidly adjust its control output to the corresponding channels without the need of the controlled system modeling and the coupling channels can reach the uniformity output without design complex coupling links. The proposed control method obtained good tracking performance in the strong coupling, large time-delay multi-input multi-output temperature control system.

Chapter 4

Efficient Model Compression Method for Temperature Control System

The success of neural networks is due in large part to the larger and larger network architectures and more and more neurons. Although this allows neural networks to perform better in many tasks, it also puts more demands on computer hardware, including more computing power and storage space. With the development of the Internet of Things, the need for enterprises to deploy deep learning models on embedded devices is also growing rapidly, but the computing power of embedded devices is limited, and the deployed model is required as small as possible to save costs[71]. Consider our NN-based temperature control system, even if the network model is not so big, it also needs plenty of memory and hardware for doing lots of floating point operations and thus affects the trained network to be further deployed in the embedded devices with limited memory space and low computational power. On the other hand, many studies have proved that there are lots of redundant connection in the network model, such an over-parameterized model can be compressed into a much smaller size by various model compression techniques[72, 73].

Model compression refers to that use the datasets to simplify the trained model, and then obtain a lightweight and accurate network[74]. The compressed network has a smaller structure and fewer parameters, which can effectively reduce the cost of computing and storage, and is easier to be deployed in the resource constrained hardware environment. Methods can be generally divided into two types: the first is to modify the model structure to reduce the model storage size, which will lead to sparse weight matrices and an irregular model structure.The second does not produce the irregular model structure, but it reduces the number of parameters in the original models to reduce model storage. Based on the two types, this chapter focuses on how to effectively reduce the number of parameters in our NN-based temperature control model while guaranteeing

the control performance. Details are introduced in the following sections.

4.1 Related Background

In common with many other machine learning models, deep neural network can be divided into two stages: training and inferencing stages. In the training stage is to train the connections in the model according to the data (for the neural network, it is mainly the weight in the network). In the inferencing stage, the new data are fed into the trained model and the results are calculated. A model is over-parameterized means that in the training stage, a large number of parameters participate in the training of the neural network, so as to capture the tiny information in the data. However, once the training is completed, in the inferencing stage, many studies have demonstrated that the trained network actually does not need so many parameters to achieve the same performance. In other words, there are a large number of redundant connections in a trained network, and sometimes only a very small portion (below 10%) of them are involved in the inferencing stage, ensuring the trained model to achieve similar performance to the original network.

The final speed of the model is not only related to the amount of computation, but also to factors such as memory bandwidth, optimization, CPU pipelining, Cache, and so on. Commonly, the calculation complexity of neural network models can be roughly measured by floating-point operations(FLOPs) or multiply–accumulate operations (MACs) (1MACs $\approx$ 2FLOPs). These operations which include addition, subtraction, multiplication, division, etc., are treated as a single FLOP for calculation purposes and computing in NN focuses on matrix multiplication and dot products. For a fully connected layer with the number of input units I and the output units O, the connecting weights W are stored in a matrix of $I \times O$. Therefore, the calculation amount is $I \times O$ MACs and $(2I - 1) \times O$ FLOPs. For the nonlinear activation function ReLU ($y = max(x, 0)$) with the number of output J, the calculation amount is J FLOPs which performs a signle operation on the CPU. Too many layers and neurons will bring over billion multiply-add calculations and keep the system busy.

Therefore, for those devices with limited memory and low computing power, and do not support large model online computing, it is possible to compress and accelerate the model to save parameter storage space and reduce the calculation time without losing the model accuracy[75].

4.1.1 Objectives of model compression

The objectives of model compression can be divided into two types: 1) Reduce runtime memory/video memory consumption; 2) Reducing the memory consumption of the model on disk (without reducing the runtime memory) makes the model smaller and easier to transfer between devices. The model with smaller storage has many advantages. For example, small storage space enables many mobile machine translation applications to use smaller offline download packages to improve user experience, reduce bandwidth and improve parallelism. The experiments in this chapter mainly aim at the second purpose, which is to compress the model storage space.

4.1.2 Methods of model compression

The method of compression and acceleration of a deep network can be divided into four categories as:

(1) Quantization: It refers to that use a finite number of values to represent all connection parameters. For example, use the clustering method to represent similar connections by a single value. Binary quantization is one of typical methods, which only using 1 and 0 to represent all parameters in the network. But it commonly needs to be realized under specific hardware conditions[76].

(2) Knowledge distillation: It uses complex but high-performance networks to guide training of small networks. If the small network can achieve similar performance to the large network, it is equivalent to realize the network compression indirectly. However, the selection of the small network structure is still an unsolved problem[77].

(3) Low-rank factorization: It reduces the size of the model by decomposing the large matrix into small matrix, such as singular value decomposition (SVD) method, full-rank decomposition method, etc[78].

(4) Network pruning: Its main idea is to reduce model size by reducing redundancy in model parameter. Initially, pruning was used to improve the generalization performance of neural networks. For a specific task, it is difficult to determine the number of hidden layer neurons, so it is usually necessary to train a large network first, and then to improve the generalization performance of the neural network by pruning to avoid over-fitting caused by the large network. In some studies, experiment results show that the pruned model can obtain higher model accuracy[79].

In recent years, network pruning methods have made great achievements in computer vision tasks based on convolutional neural networks(CNN). Some research work can even drastically reduce the number of the parameters in the model without losing

the performance of the model. Compared with the previous three technologies for compressing the networks, pruning is simple and effective, so it becomes one of the important research directions of neural network compression.

4.1.3 Network pruning

Specifically, network pruning can be divided into:

(1) Structured pruning: It commonly directly removes unimportant neurons, channels or filters in the networks. Therefore, the pruned model can achieve obvious inference acceleration and storage advantages under existing hardware conditions. It is usually along with a great accuracy loss of the compressed model.

(2) Unstructured Pruning: The basic unit of it is a single weight, which usually brings smaller accuracy loss after pruning, but will eventually produce a sparse weight matrix, which requires extra support from specific hardware and the computing library to achieve inference acceleration and storage advantages. Weight pruning belongs to the unstructured pruning, which directly sets unimportant weight connections to 0, so it can reduce the number of connection parameters to be stored in the network[80].

Actually, network pruning problem can be abstracted into an optimization problem, that is, to minimize the number of remaining parameters in networks under the constraint of performance loss, as shown in the following formula:

$$min \left\| \mathbf{W} \right\|_0, s.t. f(\mathbf{W}) \leq \delta \tag{4.1}$$

where $\mathbf{W}$ indicates the network weight parameters to be optimized. $\left\| \cdot \right\|_0$ represents L0 norm which calculates the total number of nonzero elements in $\mathbf{W}$ matrices. $f(\cdot)$ is the performance/accuracy loss of a pruned neural network. Obviously, because of the existence of L0 norm, it is a combinatorial optimization and NP-hard problem[81].

4.1.4 Pruning algorithm

Commonly, pruning algorithms can be divided into one-shot pruning and iterative pruning: The former first evaluates the importance of parameters like neurons or weights in the well-trained model. After that, parameters below a threshold determined by the target pruning/sparsity ratio are directly cut off, and then the pruned model is fine-tuned. However, if the target sparsity ratio is high, this direct pruning way will seriously endanger the model accuracy.

The latter is similar to the former, but the parameters are gradually pruned from a low sparsity ratio toward the target sparsity by iterative optimization. The basic iterative

pruning framework proposed by Han[82] as the following figure:

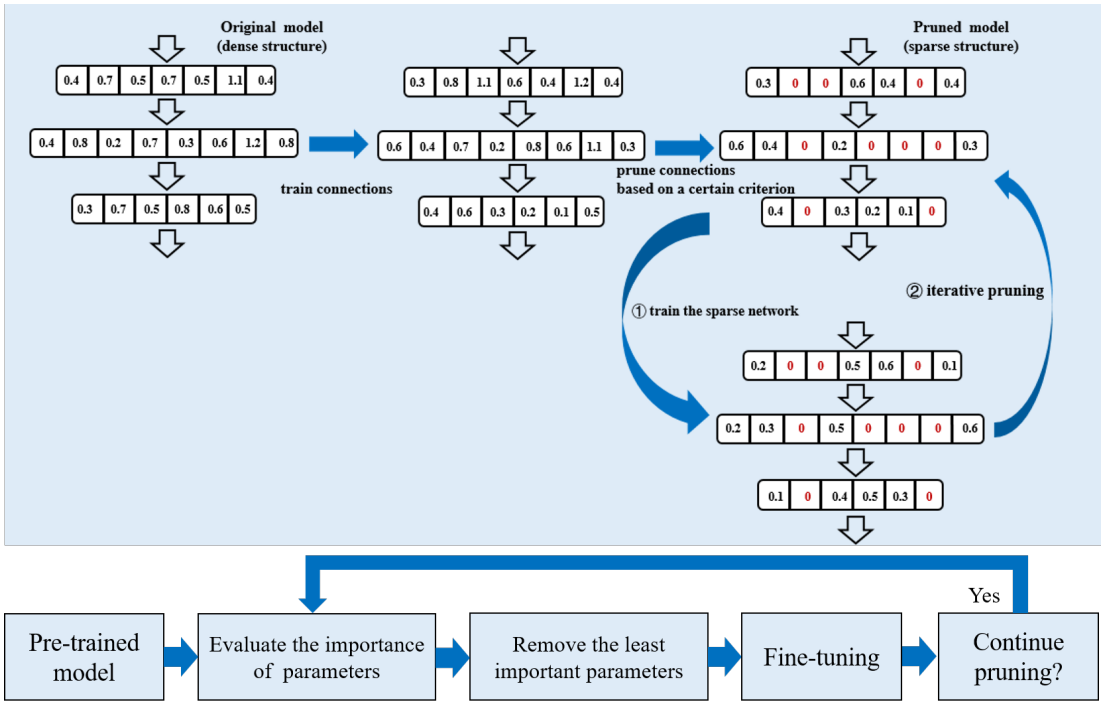


Figure 4.1: Flow chart of a typical iterative pruning process

It can be summarized as the following steps: First, delete unimportant weight parameters according to the absolute value of weight or other evaluation indicators, and then retrain the sparse network to recover the model accuracy. The above two steps are iterated throughout the process until the sparsity and accuracy loss of the target network meet the requirements. Here, fine-tuning is just a retraining process of the network with the retained parameters. A pruned model is generally accompanied by varying degrees of accuracy loss, so it is necessary to take measures to recover it. The model obtained in iterative pruning way tends to be more compact and smaller, but its training cost is higher than that of one-shot method.

One problem needs to be solved in the process is how to evaluate the importance of a connection. That is, how to effectively remove the model parameter and minimize the loss of accuracy. One is based on the magnitude of parameter[83], and the other is based on the loss function. Because the output of a feature input is weighted by multiplying the input and weight parameters, the smaller the magnitude of the weight, the smaller the contribution to the output. Therefore, pruning methods based on the magnitude of the parameter is straightforward, such as calculate the L1/L2 norm of weights. Another is based on the optimization objective of the network, Optimal Brain Damage (OBD)[84] method is one of representative technologies proposed in the 1990s. It builds a local model (Taylor series) of error function to predict how the disturbance of the parameter vector affect the optimization objective function. It uses the second derivative of the loss

function w.r.t the weight (Hessian matrix for weight vector) to measure the importance of the weight in the network. Because this method requires calculating the Hessian matrix or its approximation, it is time-consuming. Similar to the weight pruning, there are also some classical pruning methods for neurons or channels. The first is also based on the importance, to evaluate the effectiveness of a neuron, and then to make the model structure itself sparse under some constrains for pruning operations[85, 86]. The second is to measure the sensitivity of the neuron based on its influence on the optimization objectives. The third is to use reconstruction error to guide pruning operation, which can measure the effect of a neuron on the output of the network indirectly[87, 88].

In addition, the sparsity ratio of the network which is the proportion of zero value parameters in the network can be divided into the pre-defined type which specifies the target pruning rate manually and is required to decide the sparsity of each layer parameters ahead. Another is the automatic type which removes the network parameters based on the global information of the layers and the final sparsity is not defined in the beginning[80]. In order to reduce the accuracy loss of our pre-trained control model as much as possible, the automatic selection type is adopted to remove the redundant connections in our control model.

4.2 Pruning in NN-based temperature control model

In this section, the whole framework of the performed layer-wise pruning method in the a pre-trained NN-based temperature control model is introduced. Inspired by the linear reconstruction error-based filter pruning method used in convolutional neural network, the optimization objective of our model pruning is modified to nonlinear reconstruction error and the pruning rate of the layer connections is decided according to the effect of each connection on the global accuracy loss. This method helps effectively remove the redundant weights or neurons of our NN-based temperature control models layer by layer, and as far as possible to ensure the original control performance of the models.

4.2.1 Layer-wise pruning algorithm

Previous researches have proved the validity of layer-wise pruning method[89]. In the layer-wise pruning process, the objective function(loss function) of each layer for optimization has an important influence on the performance of the layer pruning. If the loss function calculates the error between the output of the pruned model that has not been activated and the output of the unpruned model that has not been activated, it

means that the linear reconstruction error (LRE) is taken as the optimization objective in layer-wise pruning. On the opposite, calculate the error between the outputs that has been activated, it represents nonlinear reconstruction error(NRE) as the optimization objective.

For current popular network models, the widely used activation function is ReLU, which outputs zeros for all negative inputs. Even though the absolute value of the difference between them are big, it will not affect the output of the activation function. Furthermore, the output has no effect on the loss calculation. Therefore, calculate the NRE after pruning each layer and take it as the optimization objective to retrain the network will be reasonable and effective. Therefore, the NRE-based pruning is applied in our experiments to solve the sparse optimization problem as Equation 4.2, where $E(\cdot)$ represents the NRE loss function, δ is the number of parameters to be cut off.

$$\min_{W \in \mathbb{R}^d} E(W), s.t. \|W\|_0 \leq \delta \tag{4.2}$$

To prune a multi-layer neural networks as Figure 4.2.1, the first three layers of the network is separated from the original model, containing the weights $W_{ji}^{(l)}$ between input and hidden, and weights $W_{kj}^{(l+1)}$ between output and hidden. The number of neuron at each layer is i, j and k, respectively. z and a are inactive input and activated output, respectively. The output of the current three-layer network is y_{l+1}. The layer-wise pruning based on the NRE is performed on such a network to minimize the error between the output y_{l+1} and the output $\hat{y}_{l+1}$ after pruning.

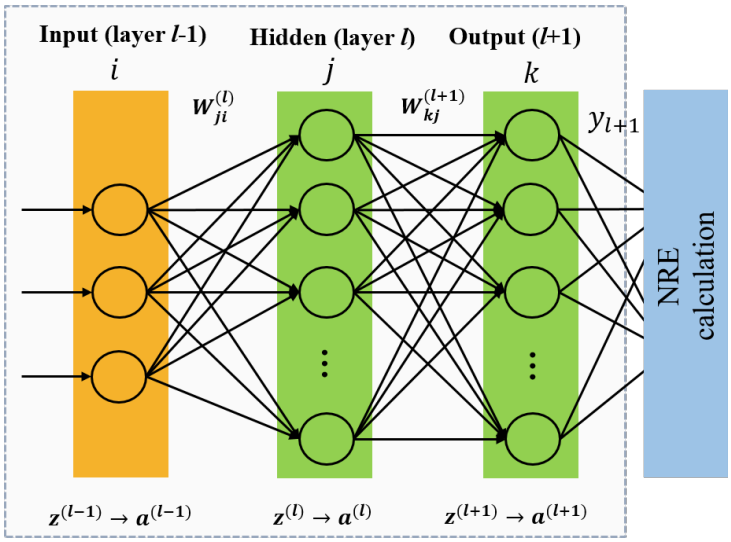


Figure 4.2: NRE-based layer-wise pruning

The importance of each neuron j in the hidden layer are determined by Equation 4.3, where all the connections of the neuron j in hidden layer l to be removed are considered.

The importance of each neuron k in the second hidden layer $l+1$ is also calculated in this way.

$$\delta_j^{(l)} \approx \sum_j (W_{ji}^{(l)}) * \sum_j (W_{kj}^{(l+1)}) \tag{4.3}$$

Commonly, a binary mask $M^{(l)} \in \{0,1\}^{W^l}$ is used to prune weights at each layer in the networks, which has the same size as the corresponding weight matrix. In the neuron pruning, all of the incoming(a whole row) and outgoing(a whole column) weight connections of one pruned neuron will be removed as Figure 4.2.1. Here, the increased percentage of the model accuracy loss E is recorded after pruning each neuron in order for setting the limit value of the number k of each layer neuron to be removed.

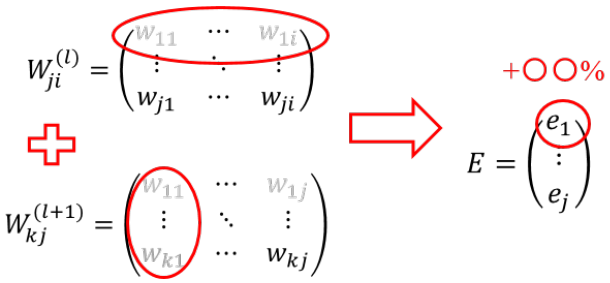


Figure 4.3: Prune the haidden layer neurons in the pre-trained NN model

For our pre-trained neural network model, define the weight matrix between each layer as $W^{(l)}$, where $l \in [xh, hh, ho]$ and xh: input-hidden, hh: hidden-hidden, ho: hidden-output. The symbol $W_{i,j}^{(l)}$ represents the weight connection between the $l-1$ layer with i neurons and l layer with j neurons. By performing $M^{(l)} \odot W^{(l)}$, the parameters to be removed will be equal to zero and the remaining parameters are unchanged, thus the masked weight matrices are updated as 4.4, where $\hat{W}^{(l)}$ is th masked weight values in layer l and $\odot$ denotes the element-wise product operation. The threshold k indicates how many neurons to be cut off can be obtained according to the calculated accuracy loss E after removing each neuron in order. Specifically, rank the importance values $\delta_j^{(l)}$ of the hidden neurons j in layer l at first, then the accuracy loss after pruning each neuron in ascending order of importance is recorded. Within the limit value of the accuracy loss, the maximum number k of the neurons in the current hidden layer can be removed is computed. Then the incoming and outgoing weights of the less important k neurons will be set as zero as Equation 4.4.

$$\hat{W}^{(l)} = \begin{cases} 0, & remove \\ W_{i,j}^{(l)}, & retain \end{cases} \tag{4.4}$$

4.2.2 Optimization process

The optimization objective of the current three-layer network work is equal to the least square loss ε_{l+1} after weights $W^{(l)}$ and $W^{(l+1)}$ are pruned can be written as Equation 4.5. It calculates the nonlinear reconstruction error of the output layer in the separated network as shown in Figure 4.4, where the left image is the current three-layer network to be pruned. Here, $\hat{y}_{l+1}$ represents the output of pruned network model, y_{l+1} is the and original model, and $\|\cdot\|_2$ indicates the L2-norm calculation.

$$E_{l+1} = \frac{1}{2}\|y_{l+1} - \hat{y}_{l+1}\|_2^2 \tag{4.5}$$

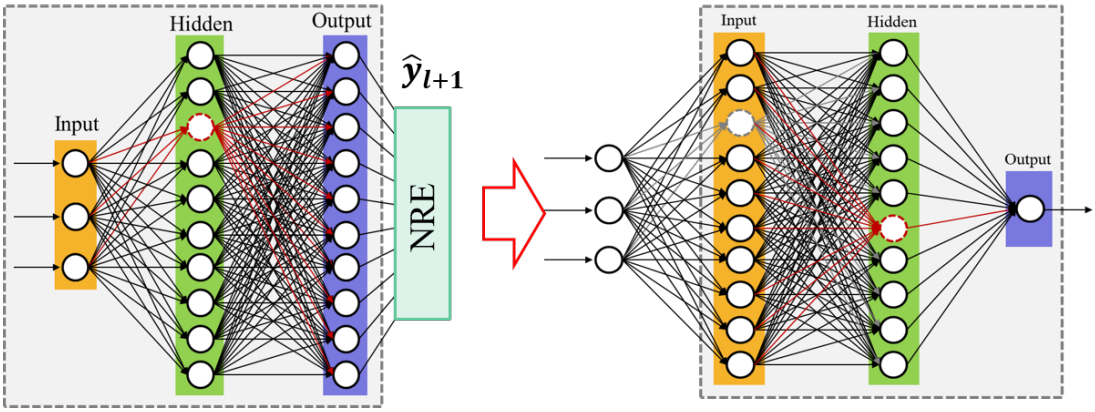


Figure 4.4: Neuron pruning in each separated three-layer network

From the formula, the optimization objective of the current three-layer network is minimizing the loss E_{l+1} and satisfying the constrain of pruning number k at the same time, which means that a fixed number of weight parameters always need to be removed during the optimization process. And consider the nonlinear activation function used in the network, the optimization of the nonlinear reconstruction loss is NP-hard. Therefore, the parameters to be removed/preserved are determined at first under the constrain and then minimize the loss by the back propagation learning in each iteration loop. And this process is repeated until the model reaches the convergence or the maximum number of iterations.

During the feedforward propagation, after the pruning masks $M^{(l-1)}$ and $M^{(l)}$ are firstly decided, the output of hidden layer $a^{(l)}$ and the output $\hat{y}_{l+1}$ of pruned network in the current three-layer network are calculated as Equation 4.6 and 4.7, respectively, where $g(\cdot)$ and $f(\cdot)$ are the activation function of hidden layer and output layer, respectively. The used activation function in our NN model is ReLU as introduces above.

$$a^{(l)} = g(W^{(l)}(a^{(l-1)} \odot M^{(l-1)})) \tag{4.6}$$

$$\hat{y}_{l+1} = f(M^{(l)} \odot W^{(l+1)} \cdot a^{(l)}) \tag{4.7}$$

During the error back propagation learning, considering the masked weight, the standard weight adjustment formula based on gradient descent can be modified to the formula 4.8, where η is learning rate. The weight $W_{j,i}^{(l)}$ is also updated in this way.

$$W_{k,j}^{(l+1)} := W_{k,j}^{(l+1)} + \eta \frac{\partial \varepsilon_{l+1}}{\partial (M_{k,j}^{(l)} \odot W_{k,j}^{(l+1)})} \tag{4.8}$$

Specifically, for our pre-trained FNN model, the network with weight parameters W^{xh} and W^{hh} is separated first, and then rank the importance scores of the hidden layer neurons in descending order. According to the calculated accuracy loss after removing the neurons in the order of importance, the maximum number k is determined within a pre-set limit 2% of accuracy loss. By multiplying the mask matrices, those weight parameters connected to the neurons will be removed, which are set as zero. Then in one iteration loop, feedforward propagation and error back propagation are performed in order.

(1) In the feedforward path, the output of the pruned network which is also the second hidden output of the original model will be calculated by Equation 4.6 and 4.7. According to the Equation 4.5, the NRE loss function of the network can be rewritten as Equation 4.9.

$$\varepsilon_h = \frac{1}{2} \| y_h - \hat{y}_h \|_2^2 \tag{4.9}$$

(2) In the feedback path, the loss ε_h is back propagated to adjust the weight W^{hh} between the hidden and hidden layer, and weight W^{xh} between the input and hidden layer, respectively. The gradient descent algorithm is applied to train the current pruned model. For the RNN model, the backpropagation is based on the time. Assume that the RNN model is at t+1 step, the loss ε across all the time steps is back propagated to adjust the weight W^{hh} between the hidden and hidden layer, and weight W^{xh} between the input and hidden layer, respectively, written as follows:

$$\frac{\partial \varepsilon(t+1)}{\partial W^{hh}} = \sum_{k=1}^{t+1} \frac{\partial \varepsilon(t+1)}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_k} \frac{\partial h_t}{\partial W^{hh}} \tag{4.10}$$

$$\frac{\partial \varepsilon(t+1)}{\partial W^{xh}} = \sum_{k=1}^{t+1} \frac{\partial \varepsilon(t+1)}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_k} \frac{\partial h_t}{\partial W^{xh}} \tag{4.11}$$

In each loop iteration, the pruning masks are updated first under the constrain condition(the pruning number k), then the three-layer network containing W^{hh} and W^{xh} completes one feedforward and one backpropagation calculation. After such iterative

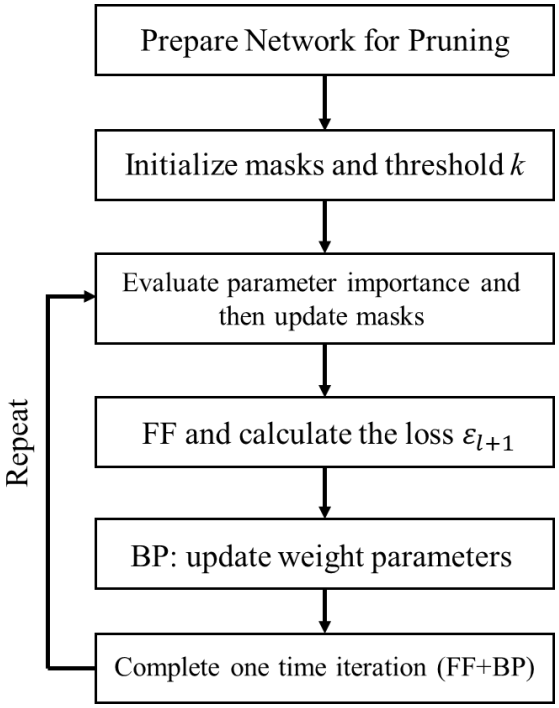


Figure 4.5: Flow chart of layer-wise iterative pruning algorithm

optimization process, the final optimized weight parameters are preserved for subsequent operations on the loss at the next three-layer network, which is as the right image in Figure 4.4. For our NN model, the next three-layer network refers to the network containing weights W^{hh} (the first hidden layer $\rightarrow$ the second hidden layer) and W^{ho}(the second hidden layer $\rightarrow$ the output layer). The same iterative pruning and optimization operations are performed to minimize the least square loss of the output layer. All the pruned model parameters will be fine tuned. The pruning algorithm process for each separated three-layer network can be illustrated as Figure 4.5.

4.3 Experiment

In the pruning experiments, a pre-trained feedforward neural network(FNN) and a recurrent neural network(RNN) are prepared first for verifying the results of the layer-wise pruning method. In our temperature control system, the executed FNN model is 3-10-10-1 structure and the RNN model is 3-10-1 structure, respectively. Both models are based on our previous proposed reference-model-based control method. The RNN controller with memory cells is very effective for sequential data and it can mine the temporal information in data. Therefore, it has been used to further improve the control performance in our temperature control system[90]. The improved input signals of the

two models consist of the target value, the actual model output and the error signal e between the ideal output y_{ref} and actual output y through the proportional plus derivative actions as Figure 4.7, which can provide real-time output deviation changes for more effective NN learning. The input signal through the proportional-derivative(PD) controller $G(s)$, which is given in Equation 4.12. The differential part is added with a low-pass filter, so that part of the feedback signal is processed and then to reduce the system interference, the filter coefficient γ = 0.5. The controller parameters involved are determined by Ziegler–Nichols rules as introduced. The additional input signal can make the NN controller better learn the control law to adjust its control output to the controlled system, especially when the target value signal changes or the interference signal is added.

$$G(s) = K_p + \frac{T_d s}{\gamma T_d s + 1} \tag{4.12}$$

Based on the identified transfer function of the controlled object as Equation4.13, the parameters of the feedback controller $C(s)$ can be derived as K_p = 2.264, T_d = 889.48 and η = 1/6. The reference model has the same time delay as the controlled object to ensure that the reference model generates the output at the same time as the controlled object. The error signal is the teaching signal or called the optimization objective of the NN controller, which guides the learning period of the NN controller as introduced in the MIMO control system design. The learning rate α of the weight optimization is1.0e-9 and the learning rate β of the bias optimization is 1.0e-6, respectively. In RNN controller, Adam(Adaptive moment estimation) is used to replace the stochastic gradient descent to accelerate the speed of convergence by adding the momentum and adaptive learning rate[91]. The hyperparameters of it are η = 0.001, ρ_1 = 0.9, ρ_1 = 0.999 and ϵ = 10^{-8}, respectively. The timestep of RNN is set as 10.

$$P(s) = \frac{2.854}{2395s + 1} e^{-444.7s} \tag{4.13}$$

$$R(s) = \frac{1}{0.01 \times 2395s + 1} \frac{1}{((444.7/2)s + 1)^2} \tag{4.14}$$

(1) FNN-based control model pruning

The control performance of the unpruned model is as shown in Figure 4.8, which illustrates the reference tracking responses of the system output in the up and down directions, respectively. The sum of squared error of each learning cycle is as shown in Figure 4.9, where the result in the positive reference tracking direction(SV: 100→105)

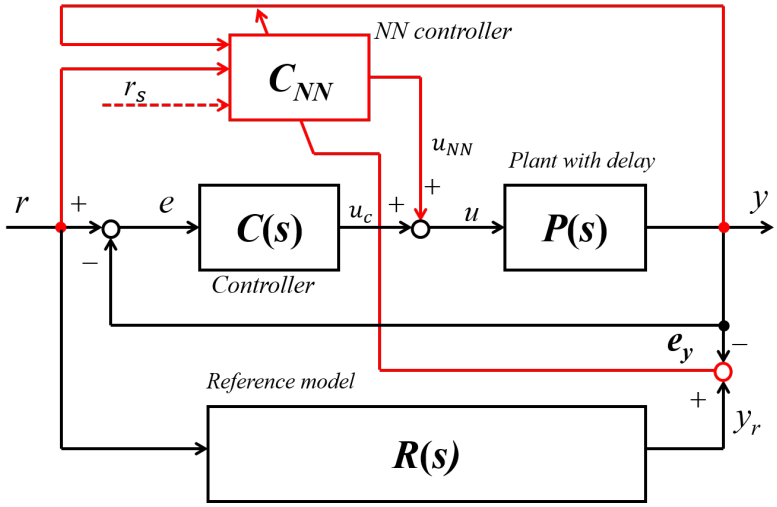


Figure 4.6: Architecture of the FNN-based Temperature Control System

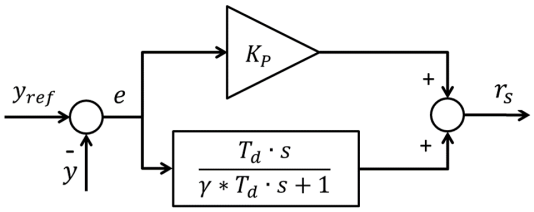


Figure 4.7: Proportional-derivative control of the input signal

is about 170 and in the negative reference tracking direction(SV: 105→100) is 267, respectively. It reflects the ability of the control system output to track the output of the ideal reference model with the gradual learning of the NN controller. As introduced in the previous chapter, the pre-trained NN controller adjusts its parameters by optimizing the error function which calculates the error between the actual output and ideal output. The dynamic characteristic indicators of the controlled system with the well-trained NN model parameters are calculated in Table 4.1 and will be used to compare with the results of the control system with the pruned NN model.

Table 4.1: Reference tracking characteristics comparison

SV: 105	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
RM	748(100%)	1321(100%)	0.00
I-PD	1101(147%)	3586(271%)	8.96(100%)
Unpruned NN	610(81%)	2343(177%)	0.63(0.07%)
SV: 100	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
RM	748(100%)	1321(100%)	0.00
I-PD	1101(147%)	3586(271%)	8.96(100%)
Unpruned NN	599(80%)	2289(173%)	2.42(27%)

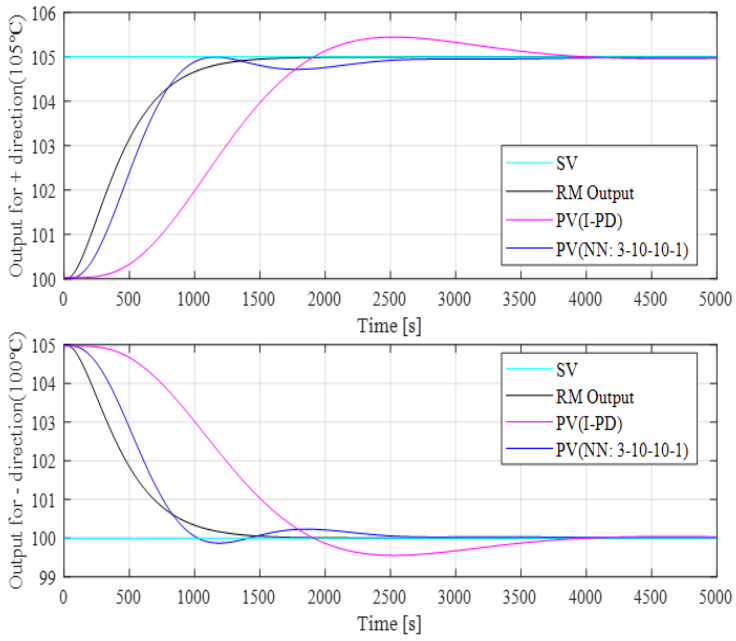


Figure 4.8: Reference tracking response of the control system with the pre-trained NN

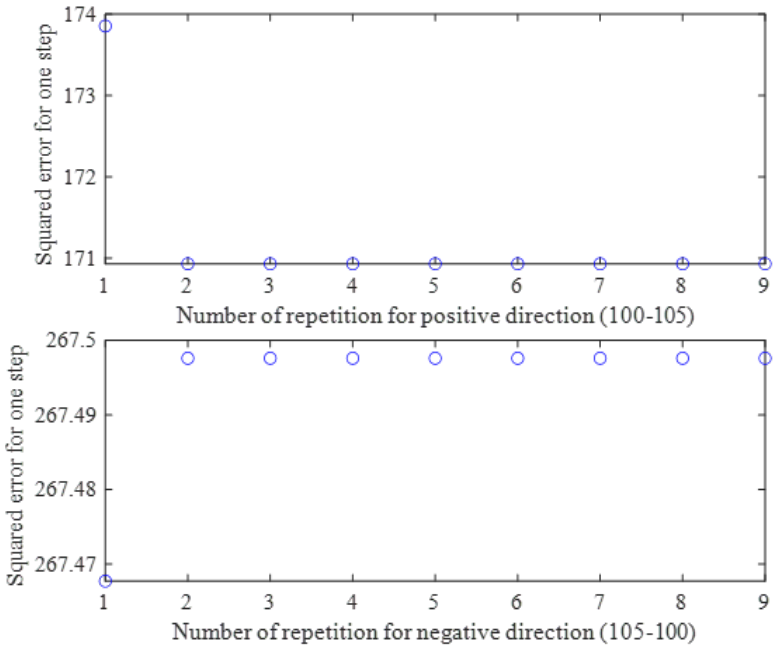


Figure 4.9: Sum of squared error of each learning cycle

Then, the first three-layer of the pre-trained model is separated to perform the neuron pruning as described above. The importance of each neuron in the first hidden layer is calculated as Equation 4.3 and the neurons are removed from the least important to the most important. The increments of the accuracy losses after removing each neuron are recorded. The maximum number of the neurons that can be pruned in the first hidden layer is constrained within the admissible accuracy range, which is set as 2% in the

experiment. The calculated threshold value in the first layer is two. The two least important neuron will be removed by the binary masks which set the corresponding row and column parameters in the weight matrices to be zero. And then retrain the model by minimizing the output error between the pruned model and the original model. The changing curve of the sum of output deviation during the retraining process is plotted as shown in Figure 4.10. With the repeated iterative optimization in the positive and negative directions, the loss of the output layer is gradually reduced to zero. The pruned weights are put into the original model for inference calculation. The sum of squared loss of one positive direction response and one negative direction response is 170 and 267, respectively. There is almost no change in both of directions compared to the control system with the unpruned model.

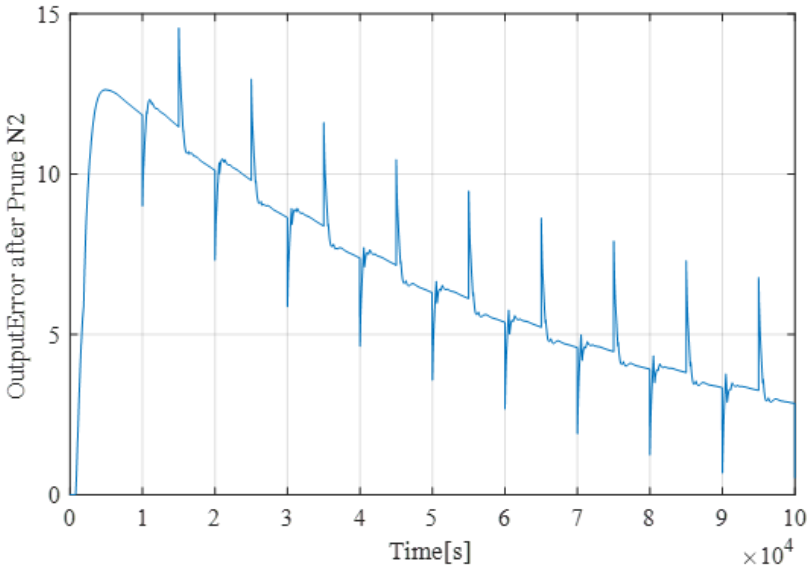


Figure 4.10: Error changing curve in the output layer

The trained parameters and the masked used to prune the connection weights are preserved to perform pruning operations in the next three-layer as shown in Figure 4.4. The importance of each neuron in the next hidden layer is also calculated first, then the maximum number of the neurons that can be pruned within the admissible accuracy range is computed, here the result is five. The five least important neuron will be removed by the binary masks which set the corresponding row and column parameters in the weight matrices to be zero. And then retrain the current network by minimizing the output error as the same steps as before. The importance of each neuron will be recalculated first in each iteration loop to ensure the neurons to be removed are always the least important.

Finally, the initial network is pruned from the structure of 3-10-10-1 into the structure of 3-8-5-1. The ratio of the number of removed parameters to the total number of the original network parameters is 50.7%. For a large full matric of size $m \times n$, it accounts

for $m \times n \times 8$ bytes of memory because the storage class is double. The full matrices of the unpruned NN model requires 1120 bytes of memory and the pruned model only requires 552 bytes which only 49.2% of the former. The FLOPs of the pruned model reduced about 52.1% compared with unpruned model.

The reference tracking response of the system with the pruned model structure is compared with the original control model, which are plotted as shown in Figure 4.11. The dynamic characteristic indicators are calculated and compared in Table 4.2. In Figure 4.12, the sum squared error of the system with the pruned NN model in the positive reference tracking direction(100→105) is 165.7 and that in the positive reference tracking direction(100→105) is 260.7, respectively. From the results, even more than half of the parameters of the original model are removed, the pruned model can still retain almost the same control performance as the original control system.

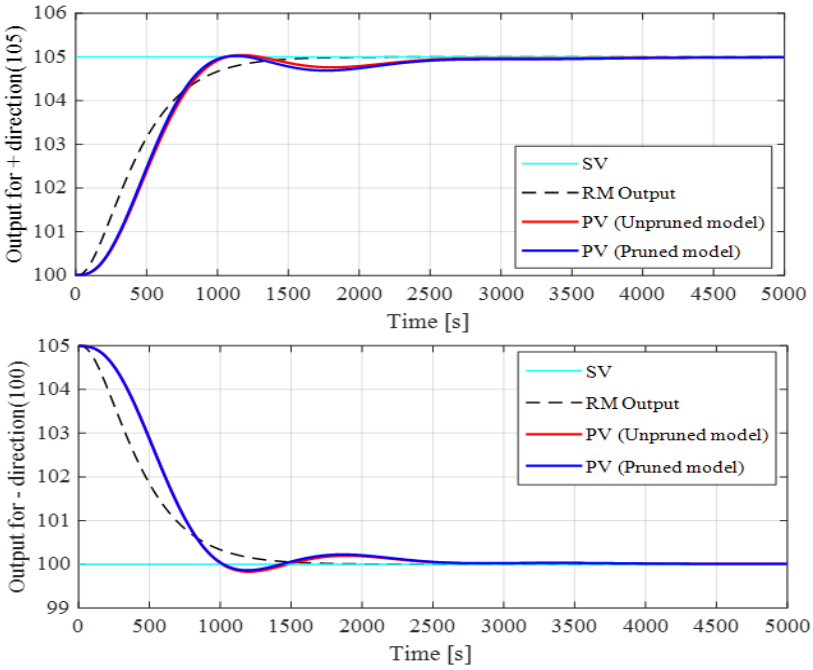


Figure 4.11: Comparison of the control performances between the unpruned and pruned models

Table 4.2: Reference tracking characteristics comparison

SV: 105	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
Unpruned NN	610(100%)	2343(100%)	0.63(100%)
Pruned NN	613(100.4%)	2394(102%)	0.13(20.6%)
SV: 100	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
Unpruned NN	599(100%)	2289(100%)	2.42(100%)
Pruned NN	601(100.3%)	2336(102%)	2.59(107%)

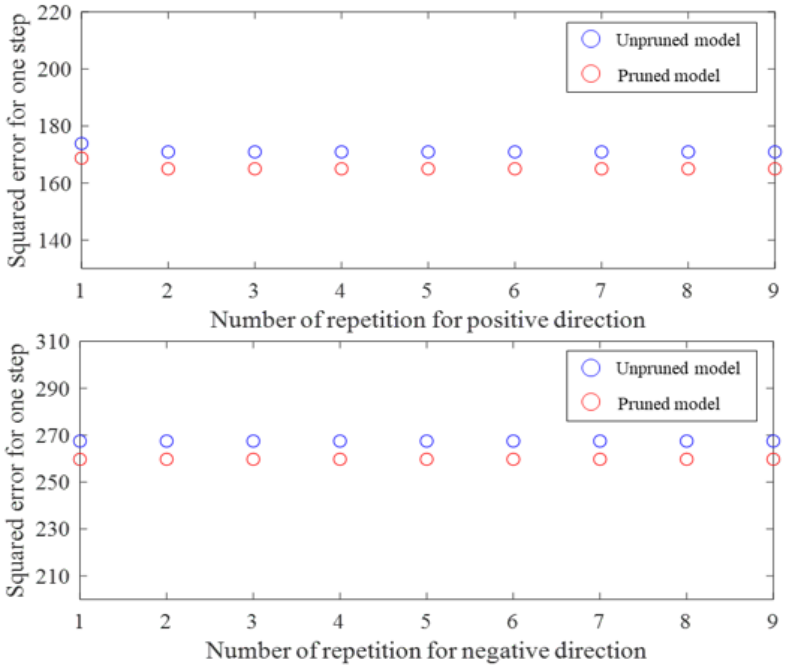


Figure 4.12: Comparison of the sum of squared error between the unpruned and pruned models

(b) RNN-based control model pruning

In RNN model pruning, because the weight in the hidden layer is shared at each time step. The whole network is most sensitive to the reduction of hidden layer neurons compared to other layers. In order to avoid the significant accuracy loss by directly eliminating the neurons, fine-grained weight pruning is more appropriate. The control performance of the unpruned model is shown as Figure 4.13, which is also compared with the I-PD control results. The dynamic characteristics of the pruned model are calculated in the Table 4.3 and compared with the results of the unpruned control model, which are expressed as a percentage. From the results of tracking responses to different set points, the well-trained RNN model can fully track the reference model output with better transient and steady-state performances.

Table 4.3: Reference tracking characteristics comparison

SV: 105	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
RM	748(100%)	1321(100%)	0.00
I-PD	1101(147%)	3586(271%)	8.96(100%)
Unpruned NN	598(80%)	1086(82%)	0.68(7.6%)
SV: 100	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]
RM	748(100%)	1321(100%)	0.00
I-PD	1101(147%)	3586(271%)	8.96(100%)
Unpruned NN	597(80%)	1087(82%)	0.68(7.6%)

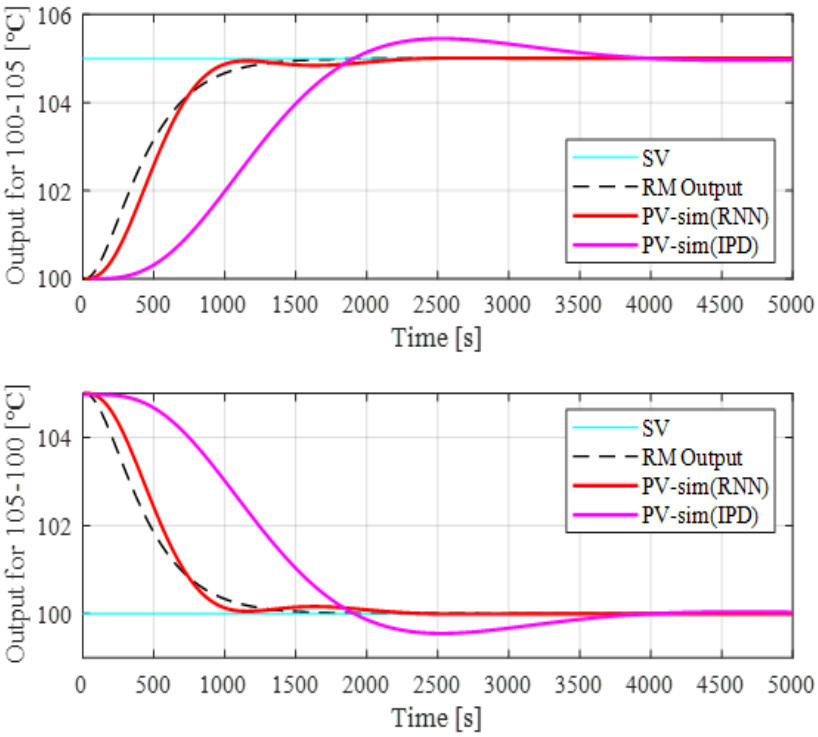


Figure 4.13: Reference tracking response of RNN-based control model

Then the layer-wise weight pruning is performed on the extracted network with input and hidden layer weights as introduced above. The pruning threshold is determined by calculating the increment percentage of the accuracy loss after deleting each connection weight. The acceptable accuracy loss of the network is set as 2% to ensure that the pruned parameters will not cause irreparable damage to the control performances. The optimization objective is the error between the nonlinear activation outputs, which is the same as that in FNN pruning. The pruned parameters of the input-hidden and hidden-hidden layers are preserved for the next layer pruning, as well as the corresponding masks used for keeping the pruned parameters zero. With the pruned weight matrices, the pruning threshold of the hidden-output layer parameters are also determined based on the increment percentage of the accuracy loss after deleting each connection. After pruning all the layers, the model with sparse weight matrices is retrained to recover the lost accuracy as much as possible.

The ratio of the number of removed parameters to the total number of the original network parameters is 68.5%. The corresponding pruned percentage of each layer weight is 11.4%, 56.4% and 0.7%, respectively. Since Matlab internally adopts the compressed sparse column (CSC) format to store the nonzero values in the sparse matrices for saving CPU memory and improving the computational efficiency. Compare with the full matric storage, it saves only those nonzero elements and their indices to effectively reduce the

required data storage. In addition, the sparse matrices will not do low-level calculations like zero-addition $(x + 0)$ to reduce the time required for performing many addition and product operations in NN model. In such sparse form, the storage of the pruned weight matrices is 928 bytes and compared with the unpruned model which is 1120 bytes(100%), the pruned model can roughly save about 17% of memory.

Obviously, there are most redundant parameters in the hidden layer, which can be removed without affecting the control accuracy. The reference tracking response of the system with the pruned model structure is compared with the original control model, which are plotted as shown in Figure 4.14.The dynamic characteristics of the pruned model are calculated in the Table 4.4 and compared with the results of the unpruned control model, which are expressed as a percentage. As shown in Figure 4.15, the sum of square errors in temperature rising and falling directions is about 205, which is almost the same as that in the unpruned control system. The pruned model can still realize the ideal control performance as the original model, even if more than half of the parameters in the original model are pruned.

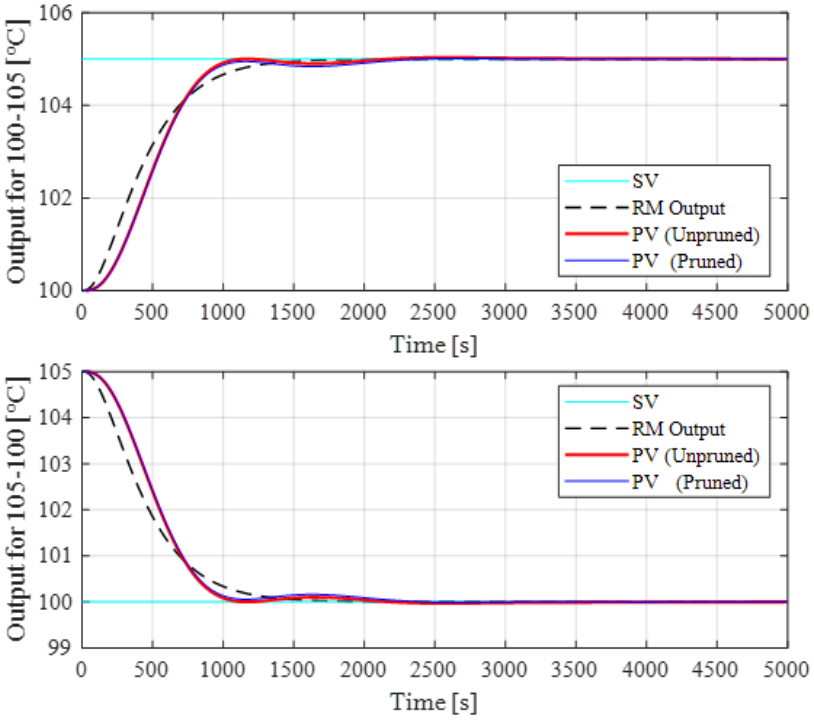


Figure 4.14: Comparison of the control performances between the unpruned and pruned models

Table 4.4: Reference tracking characteristics comparison

SV: 105	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]	Sum of SE
Unpruned NN	598(100%)	1086(100%)	0.68(100%)	205(100%)
Pruned NN	599(100.2%)	1758(162%)	0.40(59%)	208(101%)
SV: 100	Rise Time T_r [s]	Settling Time T_s [s]	Overshoot [%]	Sum of SE
Unpruned NN	597(100%)	1087(100%)	0.68(100%)	205(100%)
Pruned NN	599(100.3%)	1672(154%)	0.38(56%)	208(101%)

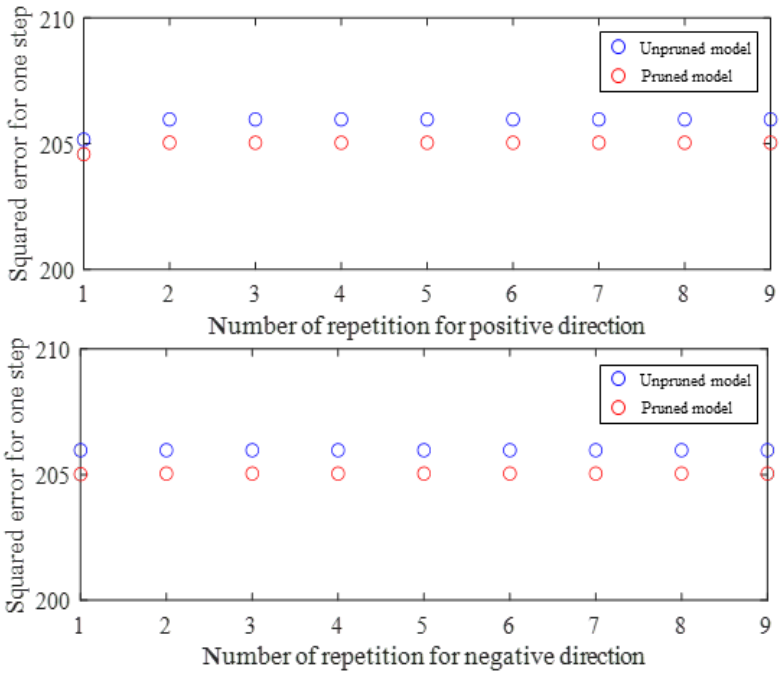


Figure 4.15: Comparison of the sum of squared error between the unpruned and pruned models

4.4 Conclusion

This chapter mainly discusses how to effectively reduce the number of parameter in our pre-trained control models by pruning technique for saving model storage space. Inspired by the linear reconstruction error-based pruning method used in CNN models, a nonlinear reconstruction error-guided layer-wise pruning method is proposed for iterative pruning the hidden layer neurons (weight connections) in our pre-trained FNN temperature control model and the weight parameters in the RNN model, respectively. The objective loss function is the error between the nonlinear activation outputs, which guides the pruned model to retrain the remaining parameters in each iteration. Experiments performed on our pre-trained FNN and RNN temperature control model proved the method can effectively delete about 50% and 68.5% connection parameters of the original model without obvious accuracy loss, respectively.

Chapter 5

Conclusion and Future Work

In this thesis, firstly, the current state of temperature control and related methods are discussed. Focusing on the problems existing in temperature control, neural network-based temperature control methods are proposed for improving the control performance and achieving precise and stable temperature control in different control systems. And based on the well-trained neural network temperature control model, we proposed an effective model compression method to remove redundant parameters in the model to save memory storage. In this chapter, we will briefly conclude our work in this thesis and discuss the deficiencies that still exist in the proposed control methods, and require more future work to solve them.

Chapter 3 first discussed the temperature control difficulties in multi-input multi-output control systems and how to solve the coupling effects between different heating channels in the past studies. Different from conventional control methods relying on precise mathematical and physical models, extra decoupling links or complex constrained optimization, we proposed a multi-layer FNN-based temperature control system and introduced an ideal model to provide the tracking target to different channels. The actual output of the controlled object is compared with the output of the reference model to provide a training signal (error signal) to the neural network controller, and then the network parameters are constantly adjusted according to the error signal. At each time step, by minimizing the optimization objective of an error function, the NN controller of each channel quickly obtains the optimal control signal to control each channel output to be consistent with the ideal model output. Furthermore, to ensure the whole control stability during the training period of the NN controller, a conventional I-PD controller in the feedback loop is employed. Based on experiments verification, the proposed NN-based multi-channel temperature control method can eliminate the mutual interference and improve the tracking performance of the temperature control system by comparing the experimental results of transient response and steady response characteristics.

Chapter 4 first discussed the problem of over-parameterization in network models. A network model with amounts of parameters will result in slow inference speed and limit the deployment of models to embedded devices with limited resources and low computational power. Therefore, the network pruning technique is adopted to compress our NN-based control models by removing unimportant parameters in the model without causing a significant loss of model accuracy. Here, inspired by the reconstruction error-based pruning method used in CNN models, which is based on minimizing the linear reconstruction error to optimize the network, a layer-wise pruning method based on the reconstruction error of the nonlinear activation outputs is proposed to prune our NN-based temperature control model. It removes the redundant connections based on the global information of the model loss and minimizes the nonlinear reconstruction error between the layer outputs of the pruned model and unpruned models. It's verified by experiments that the proposed method can eliminate a large number of redundant parameters of the well-trained NN models without significant loss of accuracy.

There are also some problems to be solved in our research, including (1) In our MIMO control system, two NN controllers are adopted to control a two-input and two-output controlled object which is simple and effective. However, if it is further extended to the systems with more coupling points, multiple NN controllers are needed and this will lead to the control system redundancy. And lots of parameters are needed to be trained in each NN model. Hence, it is necessary to further simplify the design of the control system structure, such as put multiple correlational tasks into one NN controller for simultaneous optimization learning. Moreover, the designed reference model is simply based on the controlled object, it can be further improved by adding other control loop for flexible changes. (2) In the study of model compression, layer-wise pruning is performed to effectively remove the redundant parameters in our pre-trained control models. However, pruning is only one of the model compression methods, and it does not conflict with other methods, such as knowledge meeting, quantization, it is worth further studying in combination with other methods to save the storage space of the pruned model. In addition, more work needs to be done on how to improve the efficiency of determining the pruning threshold while ensuring the minimum loss of control precision.

Acknowledgments

I would like to express my gratitude to all those who helped me a lot during the writing of this thesis and my PhD study in Gunma University.

My deepest gratitude goes first and foremost to my supervisor, Prof. Seiji Hashimoto, for his constant guidance, inspiration and encouragement. He provided support a lot and walked me through all the stages of my PhD study in Japan. From the topic selection to the determination of the outline of this thesis, modification, format adjustment and other links, he has spent much time and always given me careful guidance. Without his consistent and patient instruction, this thesis could not have reached its present form. His insightful feedbacks pushed me to sharpen my thinking and brought my work to a better level. No matter what problems I have in study or life, I can ask him for help and every time no matter how busy he was, he always took time to talk with me, and then discussed solutions together. He is always disciplined, brainy and funny, who becomes a role model and a great motivation throughout my study life and future work.

I would especially like to express my appreciation to Prof. Wei Jiang of Yangzhou University for recommending Gunma University to me so that I could have the opportunity to study here. He has provided care and help a lot, both directly and indirectly in the past years. I would also like to acknowledge all members of my PhD committee, Prof. Haruo Kobayashi, Prof. Nobuaki Nakazawa, Associate Prof. Toshiki Takahashi, and Specially Appointed Prof. Yoichi Shiraishi for their many helpful advice and suggestions in general. Their valuable comments and suggestions helped me to improve this thesis gradually. I would like to thank the members from RKC Instrument Inc.(Japan) Katsutoshi Izaki, Takeshi Kihara and Ryota Ikeda, for their technical support and financial assistance in this research project. I would also like to thank my research participants, Shinya Kobori, Kazutaka Ida, Yuta Nishizawa, Yuqi Jiang, as well as Song Xu who offered advice and guidance throughout this project. I am fortunate to have been a part of the team, we studied together, discussed and solved problems. We kept learning how to learn well, self-study, and basic skills in doing our research. We spent happy and unforgettable time together in the lab and I learnt a lot from them for their direct and indirect inspirations and advice during the whole project.

Very special thanks to all my other lab mates and friends, Kan Ni, Yu Cao, Ji Ya, Ting Yang and many others who I am really grateful for but cannot list all their names. The opportunity of studying and working with them, getting to know each other is one of the most valuable parts of my years at Gunma University. It is their kind help and support that have made my study and life in Japan a wonderful time, and for always being helpful and encouraging when my PhD study life is hard. I have been very blessed with the friendship of these amazing people. Looking back at my past study life in Japan, I have many wonderful and fulfilling memories, and these experiences transcend all the minor academic attainments of the past years and made me a better person gradually.

A special thank is owed to the student affairs division staff: Fukuda Emi and Maruoka Emi. They always help handle any problems in our daily life especially for international students during the COVID-19 pandemic, and organized many international exchange activities, which helped reduce many of the stresses in school life over the past years.

One of the best and unexpected harvest during the past five years is finding my best friend, my lover and soul-mate. The past few years have not been easy, my life also has been negatively affected both academically and personally. I really appreciate Mr.Qu sticking by me and always has faith in me when I was upset and helped me build my confidence. At the most difficult time in my life, losing my uncle who took the role of father during the previous 20 years of my life, it was hard to evaluate what his support meant to me. He has always been my supporter and has loved me unconditionally during my good times and bad. We grow up together under the care and support of each other.

Finally, I really appreciate my parents not only gave my life, but also raised me with lots of love and affection. They taught me to stick to my goals in life and fight for them with patience and courage, also instilling the belief in me that I could overcome any challenges with courage and determination. Owe to their love and encouragement that turn me from an innocent little girl into a more confident and optimistic person. They have given me so much, I love them.

Bibliography

[1] John W Dolan. The importance of temperature. *LC GC NORTH AMERICA*, 20(6):524–531, 2002.

[2] M Kato, Y Nara, M Kohno, T Sato, D Fukuda, and M Takahashi. Importance of temperature control during permeability test for measuring hydraulic constants of rock. In *ISRM International Symposium-EUROCK 2016*. OnePetro, 2016.

[3] He Qixin, Liu Huifang, Li Bin, Zheng Chuantao, and Wang Yiding. Multi-channel semiconductor laser temperature control system. *Acta Optica Sinica*, 37(11):1114002, 2018.

[4] F Patrick McCluskey, Richard Grzybowski, and Thomas Podlesak. *High temperature electronics*. CRC press, 2018.

[5] Sangyoung Park, Soohee Han, and Naehyuck Chang. Control-theoretic dynamic thermal management of automotive electronics control units. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(2):102–108, 2011.

[6] SK Jha, S Karthika, and TK Radhakrishnan. Modelling and control of crystallization process. *Resource-Efficient Technologies*, 3(1):94–100, 2017.

[7] MCR Davies, ET Bowman, and DJ White. Physical modelling of natural hazards. *Physical modelling in geotechnics, ICPMG*, 2010:3–22, 2010.

[8] Alejandro Germán Frank, Lucas Santos Dalenogare, and Néstor Fabián Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210:15–26, 2019.

[9] Emad Maher Natsheh, Abdel Razzak Natsheh, and Alhussein Albarbar. Intelligent controller for managing power flow within standalone hybrid power systems. *IET Science, Measurement & Technology*, 7(4):191–200, 2013.

[10] Mohammad Jafari, Alireza Mohammad Shahri, and Seyyed Hamid Elyas. Optimal tuning of brain emotional learning based intelligent controller using clonal selection algorithm. In *ICCKE 2013*, pages 30–34. IEEE, 2013.

[11] Ling Shen, Jianjun He, Chunhua Yang, Weihua Gui, and Honglei Xu. Temperature uniformity control of large-scale vertical quench furnaces for aluminum alloy thermal treatment. *IEEE Transactions on Control Systems Technology*, 24(1):24–39,

2015.

[12] Vishakha Vijay Patel. Ziegler-nichols tuning method. *Resonance*, 25(10):1385–1397, 2020.

[13] Najidah Hambali, Afandi Masngut, Abdul Aziz Ishak, and Zuriati Janin. Process controllability for flow control system using ziegler-nichols (zn), cohen-coon (cc) and chien-hrones-reswick (chr) tuning methods. In *2014 IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pages 1–6. IEEE, 2014.

[14] Xue-Song Wang, Yu-Hu Cheng, and Sun Wei. A proposal of adaptive pid controller based on reinforcement learning. *Journal of China University of Mining and Technology*, 17(1):40–44, 2007.

[15] Endrowednes Kuantama, Tiberiu Vesselenyi, Simona Dzitac, and Radu Tarca. Pid and fuzzy-pid control model for quadcopter attitude with disturbance parameter. *International journal of computers communications & control*, 12(4):519–532, 2017.

[16] Rosmin Jacob and Senthil Murugan. Implementation of neural network based pid controller. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 2769–2771. IEEE, 2016.

[17] Kenny Uren and George van Schoor. *Predictive PID control of non-minimum phase systems*. INTECH Open Access Publisher, 2011.

[18] Juan Garrido, Francisco Vazquez, and Fernando Morilla. Smith predictor with inverted decoupling for stable tito processes with time delays. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8. IEEE, 2014.

[19] Ikuo Nanno, Masahito Tanaka, Nobutomo Matsunaga, and Shigeyasu Kawaji. On performance of the gradient temperature control method for uniform heating. *IEEJ Transactions on Electronics, Information and Systems*, 124(8):1606–1612, 2004.

[20] Kevin M Passino. Intelligent control: an overview of techniques, 2001.

[21] Michael E Fisher, Arindam Ghosh, and Adel M Sharaf. Intelligent control strategies for permanent magnet dc motor drives. In *Proceedings of International Conference on Power Electronics, Drives and Energy Systems for Industrial Growth*, volume 1, pages 360–366. IEEE, 1996.

[22] Mo Jamshidi. Tools for intelligent control: fuzzy controllers, neural networks and genetic algorithms. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1809):1781–1808, 2003.

[23] Spyros G Tzafestas. *Methods and applications of intelligent control*, volume 16. Springer Science & Business Media, 2012.

[24] SN Vassilyev, A Yu Kelina, Yu I Kudinov, and FF Pashchenko. Intelligent control systems. *Procedia Computer Science*, 103:623–628, 2017.

[25] Deyi Li and Yi Du. *Artificial intelligence with uncertainty*. CRC press, 2017.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[27] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[28] Nima Amjady. Day-ahead price forecasting of electricity markets by a new fuzzy neural network. *IEEE Transactions on power systems*, 21(2):887–896, 2006.

[29] Yong Yu, Chi-Leung Hui, and Tsan-Ming Choi. An empirical study of intelligent expert systems on forecasting of fashion color trend. *Expert Systems with Applications*, 39(4):4383–4389, 2012.

[30] H-J Zimmermann. Fuzzy set theory. *Wiley interdisciplinary reviews: computational statistics*, 2(3):317–332, 2010.

[31] Dean C Karnopp, Donald L Margolis, and Ronald C Rosenberg. *System dynamics: modeling, simulation, and control of mechatronic systems*. John Wiley & Sons, 2012.

[32] Lino Guzzella and Christopher Onder. *Introduction to modeling and control of internal combustion engine systems*. Springer Science & Business Media, 2009.

[33] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[34] Martha A Zaidan, Darren Wraith, Brandon E Boor, and Tareq Hussein. Bayesian proxy modelling for estimating black carbon concentrations using white-box and black-box models. *Applied sciences*, 9(22):4976, 2019.

[35] Sathish Kumar Shanmugam, Yuvaraj Duraisamy, Meenakumari Ramachandran, and Senthilkumar Arumugam. Mathematical modeling of first order process with dead time using various tuning methods for industrial applications. *Mathematical Models in Engineering*, 5(1):1–10, 2019.

[36] B Wayne Bequette. *Process control: modeling, design, and simulation*. Prentice Hall Professional, 2003.

[37] Saffet Ayasun and Ayetül Gelen. Stability analysis of a generator excitation control system with time delays. *Electrical Engineering*, 91(6):347–355, 2010.

[38] Rolf Isermann and Marco Münchhof. *Identification of dynamic systems: an introduction with applications*, volume 85. Springer, 2011.

[39] Toshihiro Wakita, Koji Ozawa, Chiyomi Miyajima, and Kazuya Takeda. Parametric versus non-parametric models of driving behavior signals for driver identification.

In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 739–747. Springer, 2005.

[40] Sofia Rachad, Benayad Nsiri, and Bahloul Bensassi. System identification of inventory system using arx and armax models. 2015.

[41] RACHAD Sofia, FOURAIJI Hicham, and BENSASSI Bahloul. Modeling a production system by parametric identification approach. 2014 2nd World Conference on Complex Systems, WCCS 2014, 2015.

[42] Mohamed Naji Muftah, Wong Liang Xuan, and Ahmad Athif Mohd Faudzi. Arx, armax, box-jenkins, output-error, and hammerstein models for modeling intelligent pneumatic actuator (ipa) system. *Journal of Integrated and Advanced Engineering (JIAE)*, 1(2):81–88, 2021.

[43] Yang Shu, Hui Li, and Qian Wu. Expansion application of dspace for hils. In *2008 IEEE International Symposium on Industrial Electronics*, pages 2231–2235. IEEE, 2008.

[44] Annie Cuyt. *Padé approximants for operators: theory and applications*, volume 1065. Springer, 2006.

[45] Sudipta Chakraborty, Sandip Ghosh, and Asim Kumar Naskar. I–pd controller for integrating plus time-delay processes. *IET Control Theory & Applications*, 11(17):3137–3145, 2017.

[46] Brian R Copeland. The design of pid controllers using ziegler nichols tuning. *Internet: http://educypedia. karadimov. info/library/Ziegler_Nichols. pdf*, 2008.

[47] Lucian R da Silva, Rodolfo CC Flesch, and Julio E Normey-Rico. Analysis of anti-windup techniques in pid control of processes with measurement noise. *IFAC-PapersOnLine*, 51(4):948–953, 2018.

[48] Masanori Yukitomo, Takashi Shigemasa, Yasushi Baba, and Fumia Kojima. A two degrees of freedom pid control system, its features and applications. In *2004 5th Asian Control Conference (IEEE Cat. No. 04EX904)*, volume 1, pages 456–459. IEEE, 2004.

[49] Nobutomo Matsunaga, Shigeyasu Kawaji, Masahito Tanaka, and Ikuo Nanno. A novel approach of thermal process control for uniform temperature. *IFAC Proceedings Volumes*, 38(1):111–116, 2005.

[50] Takumi Tandou, Seiji Kubo, Nobuyuki Negishi, Masaru Izawa, et al. Improving the etching performance of high-aspect-ratio contacts by wafer temperature control: Uniform temperature design and etching rate enhancement. *Precision Engineering*, 44:87–92, 2016.

[51] Zhen Zhang, Cheng Ma, and Rong Zhu. Self-tuning fully-connected pid neural network system for distributed temperature sensing and control of instrument with

multi-modules. *Sensors*, 16(10):1709, 2016.

[52] Chengming Lee and Rongshun Chen. Optimal self-tuning pid controller based on low power consumption for a server fan cooling system. *Sensors*, 15(5):11685–11700, 2015.

[53] Carolina B Carvalho, Esdras P Carvalho, and Mauro ASS Ravagnani. Implementation of a neural network mpc for heat exchanger network temperature control. *Brazilian Journal of Chemical Engineering*, 37(4):729–744, 2020.

[54] Chiraz Ben Jabeur and Hassene Seddik. Design and implementation of pd-nn controller optimized neural networks for a quad-rotor. In *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 1–7. IEEE, 2021.

[55] Giacomo Baggio, Danielle S Bassett, and Fabio Pasqualetti. Data-driven control of complex networks. *Nature communications*, 12(1):1–13, 2021.

[56] Pauline Kergus, Simone Formentin, Charles Poussot-Vassal, and Fabrice Demourant. Data-driven control design in the loewner framework: Dealing with stability and noise. In *2018 European Control Conference (ECC)*, pages 1704–1709. IEEE, 2018.

[57] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[58] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[59] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[60] Michael W Berry, Azlinah Mohamed, and Bee Wah Yap. *Supervised and unsupervised learning for data science*. Springer, 2019.

[61] M Talaat, MA Farahat, Noura Mansour, and AY Hatata. Load forecasting based on grasshopper optimization and a multilayer feed-forward neural network using regressive approach. *Energy*, 196:117087, 2020.

[62] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[63] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[64] Mirza Cilimkovic. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15(1), 2015.

[65] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[66] David Newton, Farzad Yousefian, and Raghu Pasupathy. Stochastic gradient descent: recent trends. *Recent Advances in Optimization and Modeling of Contemporary Problems*, pages 193–220, 2018.

[67] Eiichi Muramatsu and Keiji Watanabe. Feedback error learning control without recourse to positive realness. *IEEE Transactions on Automatic Control*, 49(10):1762–1770, 2004.

[68] Xinyou Han, Wataru Imahayashi, and Kenji Sugimoto. Strictly positive real condition establishment in feedback error learning control. In *2019 12th Asian Control Conference (ASCC)*, pages 438–443, 2019.

[69] Kenji Sugimoto, Xinyou Han, and Wataru Imahayashi. Stability of mimo feedback error learning control under a strictly positive real condition. *IFAC-PapersOnLine*, 51(33):168–174, 2018.

[70] Piotr Oziablo, Dorota Mozyrska, and Małgorzata Wyrwas. A digital pid controller based on grünwald-letnikov fractional-, variable-order operator. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 460–465. IEEE, 2019.

[71] Halit Apaydin, Hajar Feizi, Mohammad Taghi Sattari, Muslume Sevba Colak, Shahaboddin Shamshirband, and Kwok-Wing Chau. Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting. *Water*, 12(5):1500, 2020.

[72] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[73] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460*, 2021.

[74] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[75] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[76] Miguel A Carreira-Perpinán and Yerlan Idelbayev. Model compression as constrained optimization, with application to neural nets. part ii: Quantization. *arXiv preprint arXiv:1707.04319*, 2017.

[77] Bharat Bhusan Sau and Vineeth N Balasubramanian. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*, 2016.

[78] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[79] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[80] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[81] Mingbin Feng, John E Mitchell, Jong-Shi Pang, Xin Shen, and Andreas Wächter. Complementarity formulations of l0-norm optimization problems. *Industrial Engineering and Management Sciences. Technical Report. Northwestern University, Evanston, IL, USA*, 5, 2013.

[82] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J Dally. Dsd: regularizing deep neural networks with dense-sparse-dense training flow. 2016.

[83] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[84] Chao Liu, Zhiyong Zhang, and Dong Wang. Pruning deep neural networks by optimal brain damage. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[85] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[86] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[87] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[88] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2525–2538, 2018.

[89] Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. Optimization based layer-wise magnitude-based pruning for dnn compression. In *IJCAI*, pages 2383–2389, 2018.

[90] Kazutaka Ida Shinya Kobori, Seiji Hashimoto. Development of temperature control

system based on reference model using machine learning. *IEE-Japan Industry Applications Society Conference*, (Y-77), 2019.

[91] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Publication Papers

Journal publications

1. **Y. Liu**, S. Xu, S. Hashimoto and T. Kawaguchi. A Reference-Model-Based Neural Network Control Method for Multi-Input Multi-Output Temperature Control System. Processes, Vol.8, No.11, 1365, Oct. 2020. (SCI, IF=2.847)

2. **Y. Liu**, T. Kawaguchi, S. Xu and S. Hashimoto. Recurrent Neural Network-Based Temperature Control System Weight Pruning Based on Nonlinear Reconstruction Error. Processes, Vol.10, No.1, 44, Dec. 2021. (SCI, IF=2.847)

International conference papers

1. **Y. Liu**, S. Xu, S. Kobori, S. Hashimoto and T. Kawaguchi. Time-Delay Temperature Control System Design based on Recurrent Neural Network. 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), pp.820-825, Victoria, Canada, online, May 2021.

2. **Y. Liu**, S. Xu, S. Hashimoto and T. Kawaguchi. Design and Verification of Reference-model-based Neural Network Control for Multi-input Multi-output Temperature Control System. Proceeding of International Conference on Technology and Social Science 2020 (ICTSS2020), IPS10-04 Invited paper, Gunma, Japan, Dec. 2020.

3. **Y. Liu**, K. Yoshida, S. Hashimoto, K. Izaki, T. Kihara and R. Ikeda, Slow Response Mode-Based Multi-point Temperature Control, Proc. of ICTSS2017, A031, Gunma, Japan, May 2017.

4. S. Hashimoto, **Y. Liu**, K. Yoshida, K. Izaki, H. Okada and R. Ikeda, Multi-point Temperature Control Method Based on Slow Response Mode. Proc. of IEEE AIM2017, Munich, Germany, July 2017.

5. S. Hashimoto, **Y. Liu**, K. Yoshida, K. Izaki, T. Kihara, R. Ikeda and W. Jiang. A Novel Multi-point Temperature Control Method Based on Slow Response Mode.

Proc. of IEEE IECON2017, SS37, pp.5623-5627, Beijing, China, Nov. 2017.

6. Y. Jiang, **L. Yuan**, S. Xu, K. Yoshida, S. Hashimoto, K. Izaki, T. Kihara, R.Ikeda and W. Jiang, Disturbance Rejection Evaluation Slow-Mode Based Control for MIMO Delay System, Electrical and Medical Intelligent System 2017, Gunma, Japan, Nov. 2017.

Domestic conference papers

1. **Y. Liu**, S. Hashimoto, T. Kawaguchi and S. Xu. Neuron Pruning-based Model Compression Method for NN-based Temperature Control System. 2022年電気学会産業応用部門大会, Sep. 2022.

2. **Y. Liu**, S. Xu, T. Kawaguchi and S. Hashimoto. Efficient Model Compression Method for Artificial Neural Network-based Time-delay Temperature Control Systems. Proceedings of "TOCHIGI GUNMA" Sub-branch Meeting, Tokyo Branch, IEE Japan, online, March 2022.

3. 橋本 誠司, 劉 媛, 小堀 伸哉, 川口 貴弘, 徐 松. 学習理論に基づくむだ時間システムの一制御法. 2021年電気学会産業応用部門大会講演論文集，2-S9-2，2021.