Flinders UNIVERSITY
ADELAIDE · SOUTH AUSTRALIA

Introducing Universal Model-free Multivariable Adaptive Neural Network Controllers for MIMO Systems

by

ARASH MEHRAFROOZ

Thesis

Submitted to Flinders University

for the degree of

Doctor of Philosophy

College of Science and Engineering

November 2019

This page is intentionally left blank.

DECLARATION

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

As specified under clause 15 of Appendix E of the Flinders University Research Higher Degrees Policies and Procedures, I hereby request that access to this thesis be restricted for a period of 36 months from the acceptance of the award of the degree.

Arash Mehrafrooz

Date: 20/7/2019

ACKNOLWEDGMENT

Today, I submitted my Ph.D. thesis. It's been a fantastic and long journey. This work would not have been possible without the support of my supervisor, family and many great individuals in my life.

Firstly, I would like to express my sincere appreciation to my supervisor Prof. Fangpo He for her continuous support of this journey, for her motivation, patience, and immense knowledge. Her guidance and leadership helped me all the time of conducting this research, publishing papers and writing of this thesis.

Secondly, I would like to thank Australians for their generosity and hospitality that encouraged me to make this decision to become an Australian and call Australia home now. I would like to acknowledge Flinders University, Macquarie University, and Western Sydney University for supporting me in this study, work, and life during this intensive program.

Last but not the least, my deep and sincere gratitude to my family; my dear wife Banafsheh, my darling daughter Viola, my parents, and my parents-in-law for supporting me spiritually and unconditionally throughout my Ph.D. program, writing this thesis, and my life in general. Additionally, I would like to thank all my friends and colleagues in Adelaide and Sydney for cheering me up to enjoy my time in this journey.

Arash Mehrafrooz

20/07/2019

Table of Contents

List of Figures

List of Tables

ABSTRACT

In this thesis, novel universal model-free adaptive controllers based on neural networks are designed to control various types of model-free coupled Multiple-Input Multiple-Output (MIMO) systems. For each controller, a specific model-free adaptive learning algorithm, with accumulated gradients using the error back-propagation algorithm, is defined and developed to be automatically trained based on the history of inputs and outputs of the system. The system is considered as a 'black box' with no pre-knowledge of its internal structure. By online monitoring of the system inputs and system outputs, the controller is able to adjust itself to the new conditions such as changing the desired outputs, structural uncertainty of the system's model, and unwanted disturbances. In this study, the design of the proposed controller is developed step-by-step as follows: (i) Firstly, by designing an Adaptive Neural Network Controller (ANNC) for controlling model-free Single-Input Single-Output (SISO) systems (in Chapter 2); (ii) Secondly, by expanding the ANNC to a Multivariable Adaptive Neural Network Controller (MANNC) to apply to square coupled MIMO systems (in Chapter 3); (iii) Thirdly, by improving the MANNC to Multivariable Adaptive Neural Network Controller with two dynamic layers (MANNC2) to deal with the non-linear characteristic of square coupled MIMO systems (in Chapter 4); (iv) and finally, by modifying the MANNC2 to a universal Multivariable Adaptive Neural Network Controller for Non-Square MIMO systems (MANNCNS), a general framework for controlling non-linear model-free non-square coupled MIMO systems is provided as the final product of this thesis (in Chapter 5); The Lyapunov stability criteria are developed for each multivariable controller and are embedded in their learning algorithms to guarantee the stability of the closed loop control system during the entire time of the adaptive control process. Several meaningful experiments by computer simulations relevant to each controller are performed in this study. The simulation results demonstrate the proper performance of the introduced controllers such as set-point tracking, accumulated error reduction, high control speed, unwanted overshoot/undershoot reduction, disturbance rejection, and dead-time compensation. These are compared to the best recent counterparts in significant examples with properties such as time-variance, non-linearity, and hybrid structure. The usefulness of the controllers in a wide range of applications, makes them potential candidatures to be implemented in industrial control software packages for use in many practical applications.

CHAPTER 1: INTRODUCTION

Overview

It is often observed that the performance of an industrial controller deteriorates during its operational life and, as time goes by, the controller may not perform as good as it was initially intended. This is largely due to the fact that the parameters of the controller that was chosen based an original system's nominal features at the time of the design, may have changed over time as the dynamic characteristics of the given system may vary during the system operation due to reasons such as aging, disturbances, environmental changes, and other unexpected circumstances. This problem will become compound when the system to be controlled is highly-nonlinear, strongly-coupled, Multiple-Input Multiple-Output (MIMO) in nature. Such systems are frequently seen in industrial applications. Although an off-line routine inspection procedure can be employed to check and correct a controller's performance for a considered industry plant, this procedure can't be implemented in real time and is often time-consuming, expensive, and involving human errors.

Given that the reason for the sensitivity of a controller often comes from the model of the considered system based on which the controller was initially designed, the sensitivity issue of the controller may be addressed if the design of the controller is less dependent on the model of the system. If the fundamental design principle of a controller relies only on the current exhibited actual dynamics of the system, by tuning the parameters of the controller online based on the input/output history of the system, the changes in the underlying dynamic system will be automatically accounted for. A controller whose design is independent of a pre-defined analytical model of a given system is thus of particular preferable.

There are many efforts in designing less-model-dependent or model-independent controllers that have auto-tuning-parameter capabilities for specific applications. However, a universal control method that can deal with a wide range of nonlinear, time-varying (or model-changing), multi-input, and multi-output

plants commonly existed in industrial applications, is clearly missing. To fill this gap, in this thesis, by using the framework of neural networks, a step-by-step design process of a universal nonlinear MIMO controller is presented. Utilising the features provided by a neural network, the produced controller will process a self-learning capability that can tune its parameters online to adapt to changes in system dynamics. The controller will be model-independent, suitable for implementation in industrial plants, and potentially operatable in real time.

The development of the intended universal nonlinear MIMO controller will take a few stages, reflecting the gradual evolvement of the design complexities. Firstly, an Adaptive Neural Network Controller (ANNC) for nonlinear Single-Input Single-Output (SISO) systems is produced. This controller does not use the model of a system, and can adjust its parameters online to track a desired behaviour of the system using solely the input/output history of the system. Secondly, by expanding the framework of the ANNC, a Multivariable Adaptive Neural Network Controller (MANNC) applicable for nonlinear square MIMO systems is derived. Although this controller can potentially cope with a range of nonlinearities, its capability in dealing with highly nonlinear industrial plants will be strengthened if its one-dynamic-layer neural-network structure can be further evolved. Thus, thirdly, a Multivariable Adaptive Neural Networks Controller with two-dynamic layers (MANNC2) for nonlinear square MIMO systems is introduced. This controller is fundamentally more mature towards the goal of the universal model-independent nonlinear MIMO controller, and its structure is naturally generalizable to include almost all industrial plants where the numbers of system inputs and system outputs are random. By modifying the MANNC2, finally, the intended universal Multivariable Adaptive Neural Network Controller for Non-Square systems (MANNCNS) is formed. At each of the above development stages, the performance of the proposed nonlinear controller is adequately compared with that of its justifiable counterpart, and the effectiveness of the proposed controller is evidently approved via comparative simulation studies on chosen examples that can reflect typical nonlinear features of industrial plants.

In the following sections of this chapter, an overview of the whole project including the importance, the research steps, the research process, the literature review, and the motivation of this study, is provided. Control methods are categorised into two classes: model-based and model-free. The restrictions of model-based methods in industrial applications are discussed, and the benefits and importance of model-free methods are described. A brief review of controlling multivariable systems based on their various types is presented, which is followed by a presentation of neuro-adaptive methods and their inherent advantages in controlling systems with nonlinear characteristics. The reason for using neural networks as a design framework of this thesis is thus justified. As necessary background knowledge of the thesis, concepts of neural-network topology, learning rules, learning rates, activation functions, and number of layers, that are associated with a neural-network structure and useful for the materials presented in the next chapters, are explained. To substantiate the importance of developing the proposed universal nonlinear MIMO controller for industrial applications, the shortcomings of the existing approaches in the literature are discussed, which bring about the thesis outline describing the development procedure of the universal controller and the list of original contributions to be presented in this thesis.

The aforementioned contents are described according to the following outline of this chapter:

1.1 Model-based advanced control methods

Systems in control theory are identified as having internal dynamics, inputs that can be manipulated and outputs that can be measured. Feedback control of a dynamic system involves supplying suitable control inputs, according to the difference between observed behaviour and desired behaviour of the system, so that the observed behaviour of the system matches with a pre-defined desired behaviour. There are well-established methods for designing and analysing feedback control systems which have led to many successes in many industrial applications. These methods include classical methods for linear control, multivariable control, adaptive control, robust control, nonlinear control, optimal control, H-infinity control, and others. In many real industrial systems, the systems often possess unknown dynamics, modelling errors, and different types of disturbances, noises and uncertainties. There are many adaptive algorithms proposed to find the system's model in the literature. In these methods, a structure of the dynamical system is identified and only the unknown parameters are computed in an online manner to provide the controller with the plant dynamic model. As the complexity of the modern dynamic systems increases, the demand for more advanced control techniques such as adaptive control, predictive control and feedback linearization that can handle complex control problems currently limited by the available feedback control strategies increases.

The majority of the current advanced control methods are largely based on pre-identified models of the systems to be controlled [1]. In these methods, using an identified model of a system, a controller is designed to lead the outputs of the system (controlled variables) to their desired values (setpoints). By having the exact or estimated model of the system, it is possible to determine the system's outputs for any applied system's inputs. A main problem of these model-based control methods in industrial applications is that real industrial systems are often nonlinear in nature. Consequently, a predicted (typically linear) model of such a system would be dynamically different than the real system itself, and a controller designed for the predicted model of the system would have a limited

effective working range. In addition, due to having uncertainties in a dynamic system and unknown environmental changes, the procedure for finding the exact model could be expensive and time-consuming. Furthermore, if one parameter of the system for any reasons such as the plant's aging changes, in order to achieve the desired outcomes, the whole system's model may need to be reidentified, which can even lead to redesigning the multivariable controller. Although the adaptive control algorithms dealing with structural and parametric uncertainty have resolved the problem of the absence of an exact model for some applications, the assumption of known structure for the dynamic system is still a constraint in the design procedure. Indeed, real-time implementations of such a control system would pose both high-complexities in realizing the required performances and high-demands in meeting the required computational powers. The system modelling becomes even more challenging when the system is identified with time-varying characteristics in nature. Because time variation is a result of system parameters changing as a function of time, all industrial systems are time-varying in principle e.g. circuit parameters in electronic circuits aerodynamic coefficients of aircrafts, hydrodynamic terms in marine vessels, and mechanical parameters in machinery. Time variation also occurs as a result of linearizing a nonlinear system about a family of operating points and/or about a time-varying trajectory for developing control system. Due to these problems, it is not always possible to control an industrial system using its identified predicted model, if the system is either highly nonlinear or highly time-varying or both.

1.2 Model-free control techniques

Despite significant improvements in designing controllers using model-based advanced control techniques in control theory, they still suffer from the difficulty of having a proper and exact mathematical model of the process in some industrial applications. In order to deal with the problem of model identification, Machine Learning Control Methods (MLCM) are being used in a wide range of applications to predict the system's behaviour when an actual pattern of inputs and outputs is mysterious in large datasets [2-5]. Therefore, machine learning

techniques which do not require the model of the system - known as “model-free control methods” in control theory - have been introduced. In these methods, the controlled system is identified by the history of inputs and outputs rather than the exact or approximate model of the system. In model-free control methods, design of the controller depends on the inputs and outputs measurement data of the controlled system without using the system model explicitly regardless of linearity or non-linearity of the plant. However, the controller may be designed using implicit information from the dynamics or structure of the controlled system. Due to the fact that the data-based model-free adaptive control method does not require the plant model in the controller designing process, in these methods there are no theoretical assumptions, modelling process and dynamics of the controlled system. Model-free control methods use the information in massive amounts of process inputs and outputs data samples to design controllers, predictors and monitoring systems. These methods are attentive to improvement of the control system performance by optimisation methods where no information exists for a priori model or very limited such information is available. Therefore, model-free control methods are often associated with the available datasets for the system’s inputs and outputs.

In addition to dealing with the constraints of the model-based methods explained in the previous section, if during the control process, the modelling phase of a system can be skipped, the time required for the adjustment of the real-time controller will be significantly reduced, which will result in a faster and more precise control outcome [1, 6, 7]. Therefore, a general model-free controller which does not require knowledge of the structure is required to remove this constraint [8]. As a result, model-free controllers are gradually becoming more preferred than model-based ones for industrial applications.

1.3 Recently introduced model-free control methods

In this section, a literature review to model-free control methods is given, despite studies on this topic are found to be much more limited in the literature compared to the advanced model-based control methods. Recently, various model-free control methods have been introduced with different names in

publications. They are called by data-driven [1, 9-18], data-based [19-22], model-free [1, 6-8, 10, 11, 14, 22-30], unfalsified control [31-34], iterative learning control [27], iterative feedback tuning [35, 36] and virtual reference feedback tuning [9, 28, 37]. In this study, control methods which are independent of the system's model are called "Model-Free Control Methods".

Model-free advanced control methods play a significant role in control of engineering systems. Several applications can be found based on the new model-free control algorithms in the literature [25, 26, 38-41]. As the first model-free control method Ziegler and Nichols proposed a procedure for tuning PID controllers which was a graphical method based on the controlled system step responses, but it was not applicable to other control problems [42-44]. ILC method is proposed as an appropriate control method on a finite time interval with a data-based feature for repeating control tasks [13, 17]. In the past decade, UC method has been used for falsifying controllers that fail to complete a performance specification using online inputs and outputs pattern of a controlled system directly [45, 46]. VRFT and IFT are other data-based control methods which can be placed in the controller parameters tuning framework. VRFT is developed for an unknown discrete-time linear single-input single-output system without requiring identification of the model of the controlled system. In this method, the controller parameters are identified by applying a virtual reference signal under the assumption that the controller's structure is already recognised as a priori. Therefore, in this method the design of controller has been changed to an identification problem for the controller's parameters [9, 27, 37]. In IFT method, as a replacement for one-shot tuning of the controller's parameters, three experiments recursively occur in each iteration to compute the controller parameters for controlling an unknown discrete-time SISO linear system is also suggested for an unknown discrete-time SISO linear system [35, 36, 47, 48]. In the VRFT and IFT, the significant concerns are the stability and limitation of implementing the method on multivariable systems.

In the recently introduced method of MFAC which a model-free adaptive control method for nonlinear single inputs single outputs systems is developed, instead of identifying the nonlinear model of the controlled system, a new dynamic

linearization technique (DLT) is utilised to build a series of equivalent local dynamic linearization data models along the dynamic operation points of the system [49, 50]. This model-free method is limited to SISO systems, however in some studies this method has been generalised to a certain class of MIMO systems [11, 15, 16, 28, 29, 51].

In [24] the essentials of model-free control for single-input and single-output systems have been introduced by employing some old functional analysis and some elementary differential algebra. By using a recent online parameter identification approach, the estimation techniques become quite straightforward. The significance of iPIs and particularly of iPs is deduced from the presence of friction. The strange industrial ubiquity of classical PIDs and the great difficulty for tuning them in complex situations is deduced, using an elementary sampling, from their connections with iPIDs. In the introduced method the system has been estimated by:

$$y^{(v)} = F + \alpha u \tag{1}$$

Which $y^{(v)}$ is the derivative of order $v \geq 1$ of y. The integer v is selected by the researcher. $\alpha \in R$ is a non-physical constant parameter, F subsumes the poorly known parts of the plant as well as of the various possible disturbances. In this study the plant is not considered as a complete unknown model and the model is estimated via the above equation which is in contrast with some definitions about model-free control methods. Also, the discussed methods are not applicable to multi-input multi-output systems.

Because of the complexity and coupling difficulty of inputs, the majority of the model-free control methods developed for SISO systems cannot be directly applied to multiple-input multiple-output (MIMO) nonlinear systems. In order to decouple the inputs, generally a decoupling invertible matrix is required to be identified. However, it is not always possible to guarantee the non-singular property of the decoupling matrix. In [52, 53] decoupling control was studied for MIMO system, and the linearized mathematical model of the controlled MIMO system and all the state variables are assumed to be available, that are too restrictive to be obtained in practical applications. So far, few papers have

investigated the adaptive control problem in the general nonlinear discrete-time system, and they focused on some special classes of nonlinearities. Over the past few years, there has been a great interest in designing model-free controllers without the need of mathematical model of controlled system to control the MIMO system online and automatically by using the functions of biological processes. These techniques include "artificial neural networks" which are based on biological neuronal structures of interconnected nodes and "fuzzy logic control" which mimics linguistic and reasoning functions. Despite lots of efforts in this area, there is still a gap in developing a universal mode-free controller applicable to general form of industrial multivariable systems. In the following section a brief overview of different types of control of multivariable systems is introduced.

1.4 Control of multivariable systems

In real-life processes in industry, not only do we deal with single-input single-output (SISO) systems but also it is quite common to face multiple-input multiple-output (MIMO), multiple-input single-output (MISO) or single-input multiple-output (SIMO) systems in various applications. Currently, many SISO control methods have been expanded for controlling SIMO and Uncoupled Multi-Input Multi-Output (U-MIMO) systems by cascading multiple SISO controllers together. However, these methods are not applicable to Multi-Input Single-Output as well as Coupled Multi-Input Multi-Output (C-MIMO) systems, because due to cross coupling in the structure of these systems, one setpoint not only affects the output variable corresponding to this setpoint, but also causes a response in each of the process output variables. In some improved methods, ordinary single-loop classical control method with multiple decentralised PI or PID controllers at lower level are used for coupled MIMO control systems [54]. Nevertheless, these single loop controllers are not able to suppress the interactions of the coupled MIMO process, and therefore each input affects not only its corresponding output but also the other ones, therefore the stability and robustness of the control system may become worse. Many articles have dealt with this drawback of decentralized control. In addition, as the PID and PI

controllers are linear, the performance of these controllers in non-linear systems e.g. heating, ventilation and air conditioning (HVAC) systems is not reliable. In another approach, cross-coupled gains are defined for a multi-variable controller [55] which are largely based on the identified linear model of the MIMO system to be controlled e.g. in [56] which has been applied to a high-speed train model. In these methods, based on the linear identified model of a MIMO system, a multivariable controller is designed to lead the outputs of the system to their desired values. The main problem of these methods in industrial applications is that the exact model of the real MIMO system is often challenging to be identified. Accordingly, a multivariable controller designed for the predicted model of the MIMO system wouldn't work properly or would work in a limited range, because the predicted linear model of such a system would be dynamically different from the original system especially where the system has nonlinear characteristic.

In the recent two decades, adaptive control methods have been introduced to adjust parameters of a designed multivariable controller's automatically. An extensive variety of adaptive control systems have been used in various industrial applications whereas variables of MIMO processes are controlled in real-time using advanced control methods instead of their classical counterparts [57-59]. By emerging computer-based controllers online, industrial plants and processes have become more and more effectively automated. In fact, auto-tuned controllers with adaptive capabilities have become extremely in demand by industry, and their developments are increasing speedily.

With regard to the number of inputs and outputs, multivariable systems can be categorised into square MIMO systems and non-square MIMO systems. A system which contains an equal number of inputs and outputs is called a square multi-input multi-output system. However, a system with an unequal number of inputs and outputs which is very common in the industrial processes is called a non-square multi-input multi-output system. Non-Square MIMO systems may have either more inputs than outputs or more outputs than inputs, therefore they can be separated into two classes in system analysis, system control and system identification. Some typical industrial samples for non-square systems

are, shell standard control problem (a 2-input 3-output system), hot oil fractionator problem (a 2-input 4-output system), distillation column (a 2-input 3-output system) and DC motor control problem (a 3-input 2-output system).

Over the past few years, there has been an increasing interest in the development of control techniques for both square and non-square multivariable systems. Some of those techniques are defined specifically for controlling square multivariable systems and some of them are specified for non-square ones. However, a common approach towards the control of non-square systems is to obtain a square system matrix by squaring up or squaring down through adding or removing appropriate inputs (manipulated variables) or outputs (controlled variables). Nevertheless, as a limitation of this approach, adding unnecessary outputs to be measured can be costly, while deleting inputs leaves fewer variables to be manipulated to achieve the desired control target. Likewise, the amount of feedback information available to the system can be reduced by decreasing the number of measured outputs, and arbitrarily adding new manipulated inputs can incur an unnecessary cost.

In closed-loop control of multivariable systems, each output to be controlled (controlled variable), requires a desired output (a setpoint) as a reference. In industry, a number of variable parameters in the controller are to be adjusted some of which depend on model of the multivariable system. If there are cross couplings in the process, the system can be difficult or impossible to control by classical controllers, because each output is coupled with two or more process's inputs. There are some difficulties of controlling a multivariable process if these cross couplings are not counteracted by the multivariable controller. Some examples of the potential problems are as follows: (i) A change in one setpoint will cause a response not only in the output variable corresponding to the setpoint but also in each of the process output variables. (ii) Assuming that ordinary single loop classical control method with multiple PI or PID controllers is used, the controllers will observe a complicated dynamic system which consists of the multivariable process with all control loops. This can make it difficult to tune each of the controllers, and the stability and robustness of the control system may become worse. (iii) As the classical controllers such as PID and PI

controllers are linear, and symmetric, the performance of these controllers in non-linear systems such as HVAC (Heating, ventilation, and air conditioning) systems is variable. As a meaningful example, in temperature control, a common use case is active heating (via a heating element) instead of passive cooling (heating off, but no cooling). Therefore, overshoot can only be corrected slowly, and it cannot be forced downward. In this case, the PID controller should be tuned to be overdamped, to prevent or reduce overshoot, though this can reduce control performance via increasing the response settling time. Many of the current advanced control methods for controlling MIMO systems are largely applicable only in uncoupled multivariable systems. In some limited scenarios a cross-coupled gain is defined to apply the controllers in coupled MIMO systems [55] which are largely based on identified models of systems to be controlled e.g. [56]. Therefore, the model-free MIMO controllers are found limited in the literature compared to the model-based methods.

1.5 Neural Network Adaptive Feedback Control

Due to presence of nonlinearities in complex system dynamics, the linearized models are ineffective for controller design in many applications. Thus, nonlinear multivariable control has constantly been the topic of research over the past several decades. A wide variety of adaptive control systems have been employed by various industries where variables of plants or processes are controlled in real-time using advanced control methods versus their classical counterparts [57-59]. By developing computer-based controllers online, industrial plants and processes become more and more effectively automated. In fact, auto-tuned controllers with adaptive capabilities have become highly sought-after by industry, and their developments are growing rapidly. The online adjustment of the controller's parameters is the primary issue of adaptive control, as these parameters are time varying. Nevertheless, the majority of these control schemes are normally based on algorithms which require specific knowledge of the process dynamics. These also need assessment and evaluation of the outputs' variation versus the inputs' variation. The implementation of neural networks is a promising alternative for the control of multi-input multi-output nonlinear

dynamical systems. There has been an increasing amount of interest in recent years regarding adaptive control methodologies based on neural networks [60-66]. Using neural networks in feedback control systems was firstly introduced by Werbos in 1989 [67]. Later, neural network control has been studied by many researchers such as [68] and this field has been the mainstream of control theory as a branch of adaptive control systems with nonlinear adjustable parameters.

Originally, Artificial neural network (ANN) has been identified as being of potential interest to the area of control systems for their capability to approximate nonlinear mapping and their ability to learn and adapt. ANN has been introduced as a mathematical computational model, inspired by biological networks. It involves interconnected groups of artificial neurons. These neurons process information for computation. An ANN is an adaptive system with a structure that varies based on external and/or internal information during learning processes. More precisely, neural networks can be considered as statistical nonlinear data modelling tools. In the last few decades, widespread research has been conducted to develop the theory as well as the application of the neural networks. Neural networks have emerged to be a powerful computational tool for solving several problems in different research fields such as advanced control, medical imaging, pattern recognition and classification, speech recognition to name a few.

As a useful and widespread strategy, neural networks have been found as a powerful approximator in the group of advanced multivariable adaptive control methods. However most of researches that used neural network for adaptive control have been applied to single-input single-output (SISO) systems [30] and applications of neural network in multi-input multi-output (MIMO) control are limited in the literatures [69]. Adaptive neural networks can be utilised for different aspects of multivariable systems such as control, modelling and identification. Recently, using neurons with dynamic structure has being rapidly increased. By means of dynamic elements embedded in a static neuron structure, such as adaptive delays, embedded adaptive filter at the output of each neuron, the desired dynamics can be formed.

A neural network control system can work promisingly for controlling multivariable systems via taking the advantage multi-input multi-output structure of neural networks as well as parallel computation capability using high-speed and multi-task processors. Neural networks have been proven to be suitable for many important control applications because of their abilities in nonlinear approximation, adaptation and parallel hardware implementation. Adding adaptive features to the neural network schemes, adaptive neural networks have been seen in controlling multivariable systems successfully [70-76]. Depending on the nature of the application and the strength of the internal data patterns it is generally expected that a network to be trained appropriately. This applies to problems where the relationships may be quite dynamic or non-linear. ANNs provide an analytical alternative to conventional techniques which are often limited by strict assumptions such as normality, linearity and variable independence. Recently, the control community has applied neural networks for different application to provide better control solutions to old control problems. Some of the features that make ANN attractive in the area of control systems are listed below:

i. Due to the ability of performing arbitrary nonlinear mappings, ANN are considered as a cost-efficient tool to produce accurate forward and inverse models of nonlinear systems. This capability allows traditional control schemes to be extended to the control of nonlinear plants without the requirement of detailed information of the plant.
ii. The ability to produce arbitrary decision regions means that ANN have the capability to be applied to fault detection problems on existing operational conditions.
iii. As the ANN training can be done online or off-line, their applications could be designed for an online or off-line adaptive control scheme.
iv. Based on ANN parallel structure, high-speed and real-time implementations are feasible which is a critical feature in control systems application.
v. If the ANN is tuned online, it is considered a system with internal dynamic states. The learning algorithms make the neural networks

strictly passive in a certain novel strong sense called as 'state-strict passivity', so that by the power delivered to the system, the energy in the internal states is bounded. It makes the closed-loop control system using neural networks robust to bounded unknown disturbances.

vi. It is understood that neural networks offer a sophisticated extension of adaptive control methods to systems that are nonlinear with unknown parameters.

Today, majority of the neural network feedback controllers are implemented by computer software. Therefore, it requires the requirement of control algorithms in discrete-time domain. In order to design these controllers, the discrete-time dynamics below may be considered:

$$x(k+1) = f\big(x(k)\big) + g\big(x(k)\big)u(k) \tag{2}$$

which $f(.)$ and $g(.)$ are unknown. The learning algorithm for a discrete-time neural network controller for the i^{th} layer is of the form below:

$$W_i(k+1) = W_i(k) - \alpha_i \varphi_i(k) \tag{3}$$

where $\varphi_i(k)$ are the output functions of layer i and α_i are the design functions base on the learning method used.

Among several different Artificial Neural Networks models, three models are identified to be the most suitable ones for control systems applications: The Cerebellar Model Articulation Controller (CMAC), The Hopfield model, and the Multi-Layer Perceptrons (MLPs) [77]. Among these models, the Multi-Layer Perceptron (MLP) with error back-propagation (BP) learning algorithm is found to be useful for solving a wide range of real-world control systems problems [78-81] as a result of their suitable flexible topology, nonlinearity, feedback and feedforward capabilities. As MLP model is used in this study a brief introduction about it is given in the next sub-section.

1.5.1 Multi-Layers Perceptrons (MLPs)

MLPs are undoubtedly the most commonly employed neural networks structure in control systems theory. In this model neurons are grouped into different layers, as input layer, output layer or hidden layers depending on whether

respectively they receive, send, or do not have direct communication with the exterior. The topology of a multilayer perceptron with a single hidden layer is illustrated in Figure 1-1.

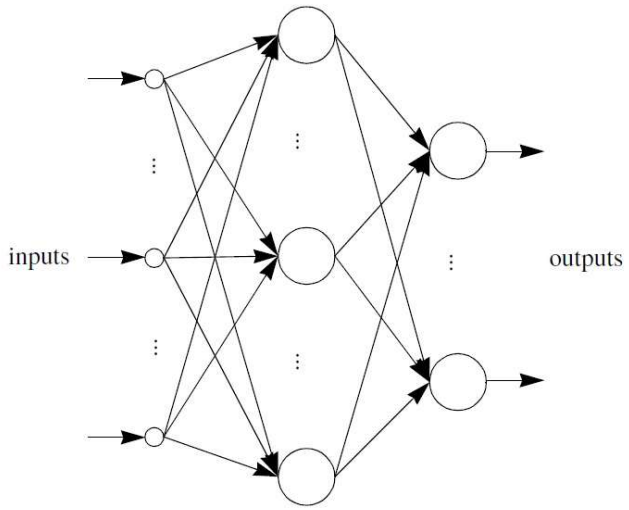


Figure 1-1. Multi-Layer Perceptron

The possibility of having feedback in the topology of MLPs and also having connections between the non-adjacent layers make this structure the main candidate to be used in control applications. In terms of the operating procedure of MLPs, data is fed at the input layer, where the neurons perform as input buffers. For each neuron in the layer immediately above, its net input is computed using weights and biases and is then passed through an output function (f) via equation (4) and (5):

$$net_i = \sum o_j w_{j,i} + b_i \tag{4}$$

$$o_i = f(net_i) \tag{5}$$

Where net_i, o_j, $w_{j,i}$, b_i and o_i are respectively i^{th} layer's input, j^{th} layer's output, weight between i^{th} and j^{th} layers, i^{th} layer's bias and i^{th} layer's output. This process is repeated through all the other layers until the output layer is reached. This operation is typically called the recall operation. With the introduction of the error back-propagation algorithm by Rumelhart and the PDP group [82] a generic learning rule for MLPs with hidden layers has been introduced.

1.5.2 Back-propagational Neural Networks

Neural network applications in closed-loop control are completely different from open-loop applications such as image processing and classification. Back-propagation is the basic multi-layer neural network tuning strategy in closed-loop control. In general, back-propagational neural networks have been designed for systems in which there is no relationship available between the inputs and output called “Black box” systems. In this case, after defining the general architecture of the network and learning algorithm and perhaps initially setting it with random numbers, the user only needs to feed in the inputs and watch the network to be trained and wait for the outputs. Back-propagation is a powerful learning method in neural networks yet on the other hand in some textbooks has been defined as a "you almost don't know what you're doing!" method [83]. However, it is possible for the user to monitor and sample the networks’ progress at regular time intervals in some software packages such as NevProp, bp, Mactivation and LAtlab Simulink.

Back-propagational neural networks are generally slower to train than other types of training techniques and typically require hundreds or thousands of epochs. If it is simulated on a serial machine training, it can take some time due to the required computation of nodes and connections’ function separately by machines CPU. Therefore, it can be problematic in large-scale neural networks with a large number of connections and weights. Nevertheless, the speed of recent computers has overcome this problem by using multi-processing and parallel computational techniques.

By using the procedure of error-based learning algorithms, back propagation can train multilayer artificial neural networks. Supervised algorithms work by using error signals generated by comparing external reference signals and the actual outputs. In order to improve the system performance, neural networks adjust their connection weights based on the error signals. In this method, it is always assumed that the desired outputs are known. By computing the errors of the output layer, back-propagation learns to find the errors in the hidden layers. Because of this optimal capability of back-propagating, this method is vastly

recommended for problems in which no relationship is defined between the neural networks' outputs and inputs. Additionally, due to its learning capabilities and flexibility, back-propagation is able to be implemented in a wide variety of applications such as controlling systems with unknown models successfully. This fact introduces back-propagation neural networks as a powerful candidate to design a model-free controller.

Back-propagation networks contain no less than three layers. There is always one input-layer, one output-layer and at least one hidden-layer in the back-propagation neural networks structure.

In order to train neural networks using the error back-propagation algorithm, inputs' patterns are presented at the input layer. Formerly, the neurons transfer the patterns activations to the hidden layer. The outputs of the hidden layer neurons can be obtained by using a bias, and also an activation function determined by the weights and the inputs. Afterwards, hidden layer outputs are used as inputs of the output neurons. Finally, the outputs of the neural networks are determined by the output layer's activations. By comparing the computed pattern and the input pattern, the error for each component of the output pattern is calculated to adjust the output weights of connections between the hidden layer and the output layer based on minimising the errors. For the connection weights between the input and hidden layers, a similar process in the output, is derived. For each pattern pair assigned for training the neural network, this procedure must be repeated. A cycle or an epoch means one pass through all the training patterns. This process which is referred as learning algorithm is then repeated as many cycles as required until the point at which the error is settled in a prescribed range. By applying the learning algorithm, the change ΔC in the cost function and the change $\Delta w_{j,i}$ in the weights are related by:

$$\Delta C \approx \frac{\partial C}{\partial w_{j,i}} \Delta w_{j,i} \tag{6}$$

This relationship shows a possible approach to compute $\frac{\partial C}{\partial w_{j,i}}$ by tracking the changes in the weights and the cost function. Also, if Δa_j^l is the change in activation of j^{th} neuron in the l^{th} layer, it can be given by:

$$\Delta a_j^l \approx \frac{\partial \Delta a_j^l}{\partial w_{j,i}} \Delta w_{j,i}^l \tag{7}$$

This change in the activation causes subsequent changes in activations in the next layer, therefore:

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \Delta a_j^l \tag{8}$$

Substituting (4) in (5), one has:

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial \Delta a_j^l}{\partial w_{j,i}} \Delta w_{j,i}^l \tag{9}$$

If the path passes through all the activations in all layers, we have:

$$\Delta C \approx \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \dots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{j,i}^l} \Delta w_{j,i}^l \tag{10}$$

The total change in C should be sum over all the possible paths between the weight and the final cost function as illustrated in Figure 1-2.

Hence:

$$\frac{\Delta C}{\Delta w_{j,i}^l} \approx \sum_{mnp\dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \dots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{j,i}^l} \tag{11}$$

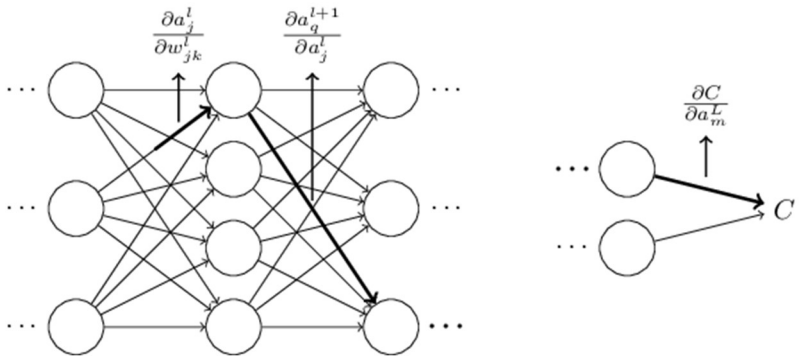


Figure 1-2. Back-propagation in MLPs

1.5.3 Importance of Learning Rate in Neural Networks

In a training cycle, the extent of weight adjustment, can be limited or expanded by changing learning rates. While a high learning rate may lead to instability in the network, on the contrary, the training time of the network can be increased when the learning rate is too low. Generally, in the beginning of weight adjustment process a high learning rate is utilised to speed up the learning

process until the it starts to reach plateau. In many cases, based on the problem, the learning rates is selected by trial-and-error methods. In reality, different values for learning rate are used for each weight during the learning process because there is no optimal value for all dimensions of a neural network, however some constraints (such as stability criteria in control problems) can restrict the learning rate in a limited range. In neural network-based control methods, it is important to use dynamic learning rate, to be able change it during the learning algorithm to keep the control system stable. In this study, tuning the learning rate is used to guarantee the stability of the controlled system at all time during the weight training process.

1.5.4 Importance of Activation Function in Neural Networks

There is a vast library of many activation functions to use in neural networks structure. As well as this, it is also possible to define a custom activation function for a specific application. Choosing a proper activation function plays an important role in the convergence speed of the learning algorithms and may strongly impact the performance and complexity of neural networks. In addition, it is also related to the application which the neural networks are used for. Despite, in order to define activation functions for neural network-based control methods it is typical to use the same functions used in classical control methods, but there is no restriction to define new activation functions for specific applications. This capability of customising activation functions in neural networks introduce it as a proper candidate to define innovative solutions in control theory.

1.5.5 Neural Network Control Topology

The most significant criteria in designing a successful neural network model to be able to converge properly is defining a suitable topology of the network. Designing a suitable neural network topology requires a comprehensive understanding of its application and it can be improved gradually by performing various experiments and trial-and-error methods. Selecting the elements of neural networks is not a very straight forward task and often, several aspects

must be considered including, but not limited to the type of the activation functions, the number of hidden layers, and the number of neurons in different layers.

As feedback control involves the measurement of output signals from a dynamic system and computing the difference between the measured values and desired values to make system's inputs, it is critical to guarantee the tracking performance and the internal stability of all variables at the same time. Failure to do so can cause serious complications in the closed-loop system, such as instability of system's responses that can result in system failure. In using neural networks for feedback control, the challenge is to choose an appropriate neural network topology for the controller, and to demonstrate how the neural networks weights can be adjusted using mathematically satisfactory methods so that performance and closed-loop stability are guaranteed as well.

1.5.6 Single Adjustable Layer vs Multiple Adjustable Layer Neural Network

A single adjustable layer neural network is normally considered as a linear classifier, and as such is ineffective at learning a sizable variety of tasks. Most notably, it is proved that single-layer neural networks could not learn to model non-linear functions even when as simple as the "XOR" function amongst other non-linearly separable classification problems [84-86], unless the function is assumed to be linear around its operation point. Therefore, neural networks with more than one adjustable layer for nonlinear modelling and classification are preferable to those with just one adjustable layer. However, more layers in the neural network structure leads to a greater number of weights to be adjusted and slower training and learning process. Therefore, number of layers in neural network topology in a controller must be selected wisely considering the type of the application and system of concern and also the required controller's parameters learning speed.

1.5.7 Best performance of Neural Networks and Why We Used It

In general, neural networks have their best performance in applications with the following conditions [83]:

(i) When it is required to discover regularities within a mysterious set of patterns and previous behaviour;

(ii) In circumstances where the number of variables, volume or range of the data is extremely large;

(iii) Whereas understanding of relationships between variables is ambiguous;

(iv) In problems which the conventional approaches cannot sufficiently describe the relationships between the variables.

The above-mentioned criteria lead us to use neural networks as a powerful and practical adaptive method in designing the previously-mentioned model-free controller for nonlinear multi-input multi-output systems because of the following reasons in respective order to the list above:

(i) In the proposed solution, we are looking for an adaptive self-learner controller which can be trained by having the history of previous behaviour of the system's inputs and outputs;

(ii) Our proposed control solution will be a generic solution for controlling coupled multivariable systems that can have several inputs and outputs which can lead to an extreme number of weights to be tuned;

(iii) We are going to use the input-output information of the system and restrict the controller's parameters to the manipulated inputs, the controlled outputs and the desired values.

(iv) Dynamic and non-linear systems are commonly used in industrial plants and for many reasons such as changing the process's parameters, aging and the structural uncertainty, the model of the systems is unknown and variable, and therefore the classical and conventional control methods could be ineffective.

1.6 Current approaches and shortcomings

By adding adaptive features to the neural networks schemes, adaptive neural networks have been seen in controlling single-input single-output (SISO) systems successfully [75, 87-89]. In these works, structure of a conventional proportional, integral and derivative (PID) controller have been used, and PID

coefficients have been tuned by neural network adaptive learning rules. For instance, in [87-89], an auto-tuned PID-like controller based on neural network principles is proposed and the neural network plays the role of estimating a suitable set of PID coefficients automatically. This method can be considered as a successful auto-tuning method which is suitable for linear systems or approximately linearized nonlinear systems.

In 2006, PID Neural Network (PIDNN) has been introduced in [90] as a combination of neural network and the PID control method. This method used the development of neural network theory for automatic control of systems. By adding the function of auto-tuning by neural networks, PID provided the memory and prediction capability in this technique.

A decoupling method for controlling MIMO plants using PIDNN has been studied in [91] which had a better performance compared to the previous PIDNN methods. However, since in this method the online learning algorithm for weights training uses the gradient descent method, it leads to the problem that the controller falls into the local minimum. As it is not possible to determine the position and the time of the local minimum, the control effect has the shortcoming of randomness. Afterwards, [92] proposed a solution based on an algorithm called PSO to optimize the initial weights of PIDNN. The online learning time of the PID neural network has been reduced significantly using this method and the local minimum problem has been resolved for a particular application. Nevertheless, in this technique the optimization strategy requires that the precise model of the controlled system be known. Therefore, this method cannot be considered as a model-free control method and inherits the problems of the model-based control methods introduced in section 1.1.

According to the literature review in using neural networks in controlling various systems, the following shortcoming have been observed:

- The majority of proposed adaptive methods have only been applicable in controlling SISO systems [93, 94] and in their best performance on MIMO systems several cascaded controllers have been used for non-coupled controlled systems. However, in many applications in industry controlling

the coupled MIMO systems are necessary as they cannot be decoupled by traditional methods. Therefore, lack of a universal solution to deal with inputs-outputs coupling in MIMO system was observed in the neural network adaptive control solutions in the literature.

- In some proposed solutions for controlling coupled MIMO systems, decoupling the MIMO systems using traditional MIMO control methods has been considered which needs the model of the system. Nevertheless, the model of the system in many applications is not achievable. Refer to section 1.2. for more information regarding the benefits of model-free control methods [95, 96].
- Some of the proposed methods are applied to specific linear applications and the performance of the controller as a general method on non-linear system has not been investigated [97-102].
- Stability and robustness of the proposed methods in the literature has not been discussed and checked thoroughly. In limited studies the possibility of falling the controlled system into a local minimum has been resolved by using conservative learning rates which disadvantage the training speed and also leads to lower quality performance of the controlled system [91].
- In the solutions based on neural networks, using only two previous samples of the output signals in all the proposed learning algorithm of PIDNN, instead of using the full history of the outputs has led to low accuracy and low reliability in the control process [94, 95, 102-105].
- The performance of the controllers on disturbance cancellation has not been tested in the proposed neural networks methods in [43, 44, 55, 63, 73, 75, 87-91, 93-95, 97-104, 106-115] except [105].
- There is no universal solution proposed for non-square MIMO systems. Most of the MIMO methods are only applied to specific square MIMO systems. However, it is very common to have non-square MIMO systems in industry as formerly explained in 1.4.
- In the majority of the similar closed loop adaptive control methods which are based on repeating a learning algorithm, the stopping point for

running the learning process has not yet been discussed thoroughly and an optimum number of iterations has not been identified.

1.7 Original contribution and proposed solution

1.7.1 The aim of the thesis

The aim of this study is a step by step development of a universal automatic real-time model-free controller for coupled square and non-square multi-input multi-output systems. The designed controller works independently from the model of the system and only requires the history of manipulated and controlled variables.

This universal controller practices only the real time input output measurement of the controlled system. In this method, there is no need for structural information and mathematical model of the controlled system, which indicates that the controller can be designed independently to the model of the system. The proposed controller and the controlled system have the following properties:

i. The controlled system is assumed to be an unknown coupled MIMO nonlinear system, but it is assumed to be known accurately in terms of the full history of inputs and outputs.
ii. The controller does not require any offline training process of the open loop system. The training process is performed during the control action in the closed loop system.
iii. The control method has been designed by only using the measurement of input and output data of the controlled system. The traditional unmodelled dynamics does not appear in this solution, therefore, they have very strong robustness when compared with the model-based control methods.
iv. The analysis approach of the proposed controller for the MIMO nonlinear system is also a novel model-free one, which is based on minimization of accumulated error during the training process.

v. Using the new learning method based on accumulated error minimisation, the possibility of falling into the local minimum has been significantly lowered. In addition, the stability of the systems has been continuously checked during the adaptive training process and the learning rates have been revised accordingly when necessary.

1.7.2 Original Contribution of the thesis

To the author's knowledge, the development of the learning algorithms, the development of the stability analysis, the use of the real-time stability criteria mixed with the weight training algorithms, the simulation studies in various MIMO applications and determination of the optimum number of trainings in chapter 2, 3, 4 and 5 are original. More specifically, the original contributions in each chapter are highlighted as follows:

- Chapter 2:
  - i) An online model-free dynamic neural network learning algorithm using the accumulated gradient in the error back-propagation algorithm is developed. The proposed method is able to control single-input single-output systems. The introduced method significantly improves the algorithms used in the existing counterpart [7, 75, 88, 89], and can provide a faster and more precise learning capacity to the system.
  - ii) Specific stability criteria have been defined for the proposed control method and have been checked in each training step to assure it can eliminate the potential instability during the training process.
- Chapter 3:
  - i) The developed version of the online dynamic neural network learning algorithm to be used for model-free control of square couple multi-input multi-output systems is novel. This method is applied to the proposed controller called MANNC. Different challenging applications are tested with the proposed controller in this section and compared with the latest solutions and counterparts.
  - ii) A specific Lyapunov stability analysis has been established for the aforementioned controller. The Lyapunov stability criteria and

continuous online observation of that during the learning process to regulate learning rates is novel.

- Chapter 4:
  i) A novel online dynamic neural network learning algorithm has been modified for two adjustable layers to improve the performance of the controller for non-linear applications. The proposed method is applied in the proposed controller called MANNC2.
  ii) A specific Lyapunov stability analysis has been developed considering two adjustable layers in the controller. The Lyapunov stability criteria embedded in the learning procedure during the neural network weight training process is novel.
- Chapter 5
  i) For the first time, an online universal model-free neural network controller has been designed to be applied to non-square non-linear couple multi-input multi-output systems. The neural network structure and learning algorithm are novel and original. The proposed method is applied to the proposed controller called MANNCNS.
  ii) A specific Lyapunov stability analysis has been developed and embedded in the designed controller. The Lyapunov criteria and real-time observation of that during the learning process is novel.
  iii) Capability of the controller to deal with external disturbances in MIMO systems is a novel investigation in controllers using neural networks.

1.7.3 The outcome of the thesis

Overall, the outcome of this thesis is a universal model-free framework for controlling non-square and square Multiple-Input Multiple-Output (MIMO) systems with nonlinear characteristics. This outcome is mostly demonstrated and concluded in chapter 5 and the previous chapters reveal the research steps that have been taken to gradually build the final universal controller. This controller has been constructed by 2-layer adjustable neural networks which are automatically trained by having the history of inputs and outputs of the system

and their desired values. Real-time and automatic control of this controller along with online real-time Lyapunov stability criteria checks develop a universal control method as a framework which can be implemented and programmed in industrial control software packages.

1.8 Thesis Outline

To illustrate the development of the proposed control method, the following process has been performed to design the methodology and generalise the suggested method step by step:

i) In chapter 2 a method is proposed for non-linear model-free single input single output systems. The learning algorithm for a basic neural network structure is designed to declare the proper functionality and performance of the method for model-free systems. It yields to propose an Adaptive Neural Network Controller called ANNC. Although the proposed ANNC in this chapter is considered in the context of SISO systems, it is developed in such a way that the proposed structure can establish a foundation upon which new adaptive neural network methods for MIMO coupled systems.

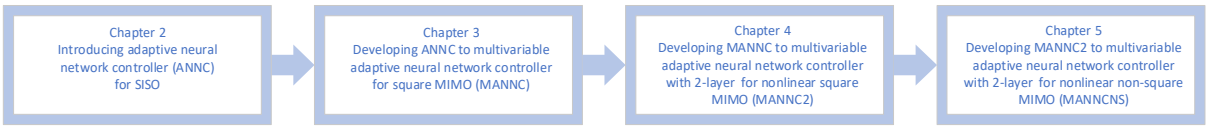
ii) In chapter 3, by having a fundamental structure of the proposed controller for SISO systems (ANNC), it is used as the basic element of a model-free multivariable adaptive neural network controller (MANNC) for coupled multi-input multi-output systems. MANNC contains one adjustable layer in its neural network structure and the learning algorithm has been developed based on accumulated error minimisation by considering the online stability criteria check during the training process.

iii) In chapter 4, the MANNC has been improved to two adjustable layers in its neural network structure to introduce MANNC2 which is applicable to square multi-input multi-output systems. By adding another dynamic layer to the neural network structure of the controller, it is improved to apply to square MIMO systems with more

non-linear characteristic. The learning algorithm and stability criteria for this novel controller structure has been developed accordingly.

iv) In chapter 5, by considering the improved results for MANNC2 compared with MANNC in chapter 4, the method with two adjustable layer neural networks has been extended to non-square multi-input multi-output systems. The method has been developed for both types of non-square MIMO systems either the number of inputs is more than the number of outputs or vice versa. The learning algorithm and stability criteria have been modified and applied to different types of simulation studies.

The above step by step procedure is illustrated in the diagram below:



Chapter 2 to 5 of this thesis are organised on the basis of the following peer reviewed publications:

Chapter 2:

- Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems

  This paper has been accepted and published in the proceeding of the IEEE International Conference on Information and Automation Wuyi Mountain, China, August 2018

Chapter 3:

- Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems

  This paper has been submitted to the journal of "IEEE Transactions on Automatic Control".

Chapter 4:

- Modification of Model-Free Multivariable Adaptive Neural Network Controller Using Two Dynamic Layers for Controlling Nonlinear Square

Chapter 1

MIMO Systems

This paper has been submitted to the journal of "IEEE Transactions on Control Systems Technology"

Chapter 5:

- Introducing a Universal Model-free Multivariable Adaptive Neural Network Controller for Non-Square MIMO Systems
  This paper has been submitted to the journal of "IEEE Transactions on Automation Science and Engineering"

References

[1] *Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.*

[2] *Wang, S., W. Chaovalitwongse, and R. Babuska, Machine Learning Algorithms in Bipedal Robot Control. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012. 42(5): p. 728-743.*

[3] *Xu, X., et al., Online Learning Control Using Adaptive Critic Designs With Sparse Kernel Machines. IEEE Transactions on Neural Networks and Learning Systems, 2013. 24(5): p. 762-775.*

[4] *Mozaffari-Kermani, M., et al., Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare. IEEE Journal of Biomedical and Health Informatics, 2015. 19(6): p. 1893-1905.*

[5] *Fadlullah, Z.M., et al., State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. IEEE Communications Surveys & Tutorials, 2017. 19(4): p. 2432-2455.*

[6] *Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.*

[7] *Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.*

[8] *Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.*

[9] *Previdi, F., et al., Data-driven control design for neuroprotheses: a virtual reference feedback tuning (VRFT) approach. IEEE Transactions on Control Systems Technology, 2004. 12(1): p. 176-182.*

[10] *Oomen, T., et al., Iterative Data-Driven${\cal H}_{\infty}$Norm Estimation of Multivariable Systems With Application to Robust Active Vibration Isolation. IEEE Transactions on Control Systems Technology, 2014. 22(6): p. 2247-2260.*

[11] *Radac, M.B., et al. Data-driven model-free control of twin rotor aerodynamic systems: Algorithms and experiments. in 2014 IEEE International Symposium on Intelligent Control (ISIC). 2014.*

[12] *Chi, R., et al., Enhanced Data-Driven Optimal Terminal ILC Using Current Iteration Control Knowledge. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2939-2948.*

[13] *Chi, R., et al., Enhanced Data-Driven Optimal Terminal ILC Using Current Iteration Control Knowledge. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2939-2948.*

[14] *Roman, R.C., et al. Data-driven optimal model-free control of twin rotor aerodynamic systems. in 2015 IEEE International Conference on Industrial Technology (ICIT). 2015.*

[15] *Roman, R.C., et al. Two data-driven control algorithms for a MIMO aerodynamic system with experimental validation. in 2015 19th International Conference on System Theory, Control and Computing (ICSTCC). 2015.*

[16] *Liu, S., Z. Hou, and J. Zheng. Attitude adjustment of quadrotor aircraft platform via a data-driven model free adaptive control cascaded with intelligent PID. in 2016 Chinese Control and Decision Conference (CCDC). 2016.*

[17] *Chi, R., et al., An Improved Data-Driven Point-to-Point ILC Using Additional On-Line Control Inputs With Experimental Verification. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017: p. 1-10.*

[18] *Pang, Z., et al., Data-Driven Control With Input Design-Based Data Dropout Compensation for Networked Nonlinear Systems. IEEE Transactions on Control Systems Technology, 2017. 25(2): p. 628-636.*

[19] *Aumi, S., et al., Data-Based Modeling and Control of Nylon-6, 6 Batch Polymerization. IEEE Transactions on Control Systems Technology, 2013. 21(1): p. 94-106.*

[20] *Pang, Z., et al., Data-Based Predictive Control for Networked Nonlinear Systems With Network-Induced Delay and Packet Dropout. IEEE Transactions on Industrial Electronics, 2016. 63(2): p. 1249-1257.*

[21] *Wang, D., et al., Data-Based Adaptive Critic Designs for Nonlinear Robust Optimal Control With Uncertain Dynamics. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016. 46(11): p. 1544-1555.*

[22] *Luo, B., et al., Policy Gradient Adaptive Dynamic Programming for Data-Based Optimal Control. IEEE Transactions on Cybernetics, 2017. 47(10): p. 3341-3354.*

[23] *Fliess, M. and C. Join, MODEL-FREE CONTROL AND INTELLIGENT PID CONTROLLERS: TOWARDS A POSSIBLE TRIVIALIZATION OF NONLINEAR CONTROL? IFAC Proceedings Volumes, 2009. 42(10): p. 1531-1550.*

[24] *Fliess, M. and C. Join, Model-free control. International Journal of Control, 2013. 86(12): p. 2228-2252.*

[25] *Lafont, F., et al., A model-free control strategy for an experimental greenhouse with an application to fault accommodation. Computers and Electronics in Agriculture, 2015. 110: p. 139-149.*

[26] *Madadi, E. and D. Söffker, Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control. 2015(57243): p. V001T03A004.*

[27] *Radac, M.B., R.E. Precup, and E.M. Petriu, Model-Free Primitive-Based Iterative Learning Control Approach to Trajectory Tracking of MIMO Systems With Experimental Validation. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2925-2938.*

[28] *Roman, R.C., M.B. Radac, and R.E. Precup, Multi-input–multi-output system experimental validation of model-free control and virtual reference feedback tuning techniques. IET Control Theory & Applications, 2016. 10(12): p. 1395-1403.*

[29] *Gao, B., et al. Model-free adaptive MIMO control algorithm application in polishing robot. in 2017 6th Data Driven Control and Learning Systems (DDCLS). 2017.*

[30] *Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.*

[31] *Safonov, M.G. and T. Tung-Ching, The unfalsified control concept and learning. IEEE Transactions on Automatic Control, 1997. 42(6): p. 843-847.*

[32] *Helvoort, J.v., B.d. Jager, and M. Steinbuch, Data-Driven Controller Unfalsification With Analytic Update Applied to a Motion System. IEEE Transactions on Control Systems Technology, 2008. 16(6): p. 1207-1217.*

[33] *Battistelli, G., et al., Stability of Unfalsified Adaptive Switching Control in Noisy Environments. IEEE Transactions on Automatic Control, 2010. 55(10): p. 2424-2429.*

[34] *Bianchi, F.D., et al., Fault-Tolerant Unfalsified Control for PEM Fuel Cell Systems. IEEE Transactions on Energy Conversion, 2015. 30(1): p. 307-315.*

[35] *Heertjes, M.F., B.V.d. Velden, and T. Oomen, Constrained Iterative Feedback Tuning for Robust Control of a Wafer Stage System. IEEE Transactions on Control Systems Technology, 2016. 24(1): p. 56-66.*

[36] Meng, W., et al., Robust Iterative Feedback Tuning Control of a Compliant Rehabilitation Robot for Repetitive Ankle Training. IEEE/ASME Transactions on Mechatronics, 2017. 22(1): p. 173-184.

[37] Campi, M.C. and S.M. Savaresi, Direct nonlinear control design: the virtual reference feedback tuning (VRFT) approach. IEEE Transactions on Automatic Control, 2006. 51(1): p. 14-27.

[38] Bu, X., et al., Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information. Asian Journal of Control, 2018. 20(2): p. 962-968.

[39] Khadraoui, S., et al., Adaptive Controller Design for Unknown Systems Using Measured Data. Asian Journal of Control, 2016. 18(4): p. 1453-1466.

[40] Xie, C.H. and G.H. Yang, Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults. International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.

[41] Cui, X., et al., Adaptive dynamic programming for tracking design of uncertain nonlinear systems with disturbances and input constraints. International Journal of Adaptive Control and Signal Processing, 2017. 31(11): p. 1567-1583.

[42] Bhatti, S.A., S.A. Malik, and A. Daraz. Comparison of P-I and I-P controller by using Ziegler-Nichols tuning method for speed control of DC motor. in 2016 International Conference on Intelligent Systems Engineering (ICISE). 2016.

[43] Azman, A.A., et al. Modeling and comparative study of PID Ziegler Nichols (ZN) and Cohen-Coon (CC) tuning method for Multi-tube aluminum sulphate water filter (MTAS). in 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS). 2017.

[44] Anto, E.K., J.A. Asumadu, and P.Y. Okyere. PID control for improving P&O-MPPT performance of a grid-connected solar PV system with Ziegler-Nichols tuning method. in 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA). 2016.

[45] Ding, X., et al. A Fuzzy Neutral Network Controller Based on Optimized Genetic Algorithm for UC Rolling Mill. in 2009 Second International Symposium on Knowledge Acquisition and Modeling. 2009.

[46] Liu, W. and L. Yu. Motion Controller Design of DR Camera's UC Arm. in 2013 Sixth International Symposium on Computational Intelligence and Design. 2013.

[47] Radac, M.B., et al., Application of IFT and SPSA to Servo System Control. IEEE Transactions on Neural Networks, 2011. 22(12): p. 2363-2375.

[48] Liu, Q., et al., Iteration Tuning of Disturbance Observer-Based Control System Satisfying Robustness Index for FOPTD Processes. IEEE Transactions on Control Systems Technology, 2017. 25(6): p. 1978-1988.

[49] Huang, Z., et al. Boost converter optimal control based on MFAC and FPSOA under model mismatch. in 2016 Chinese Control and Decision Conference (CCDC). 2016.

[50] Shangtai, J. and H. Mengxue. An improved full-form-dynamic-linearization based MFAC for a class of nonlinear systems with exogenous disturbance. in 2015 34th Chinese Control Conference (CCC). 2015.

[51] Zhang, X., Z. Hou, and S. Liu. Robust model free adaptive control for a class of nonlinear MIMO systems with measurement noise and data dropout. in 2017 11th Asian Control Conference (ASCC). 2017.

[52] Meihui, L., et al. Greenhouse multi-variables control by using feedback linearization decoupling method. in 2017 Chinese Automation Congress (CAC). 2017.

[53] Dache, C.R., et al. Linearized model of the variable flux induction motor drive. in 2016 International Conference and Exposition on Electrical and Power Engineering (EPE). 2016.

[54] Åström, K.J., K. Johansson, and Q.-G. Wang, Design of decoupled PID controllers for MIMO systems. Vol. 3. 2001. 2015-2020 vol.3.

[55] Shu, H. and H. Guo, Decoupling control of multivariable time-varying systems based on PID neural network, in 2004 5th Asian Control Conference (IEEE Cat. No.04EX904). 2004,

Department of Information and Control Engineering, Guangzhou University, China: Melbourne, Victoria, Australia, Australia.

[56] Zhang, K., et al., MIMO Evolution Model-Based Coupled Fault Estimation and Adaptive Control With High-Speed Train Applications. IEEE Transactions on Control Systems Technology, 2017. PP(99): p. 1-15.

[57] Astrom, K.J. and T. Hagglund, PID Controllers: Theory, Design and Tuning. 2 ed. 1995, USA: Instrument Society of America.

[58] Astrom, K.J. and B. Wittenmark, Adaptive Control Stability Convergence and Robustness. 1989, University of California: Prentice Hall Information and System Science Series.

[59] Farivar, F., et al., Hybrid Control of Flexible Manipulator. Journal of Applied Sciences, 2009. 9(4): p. 639-650.

[60] Kostarigka, A.K. and G.A. Rovithakis, Adaptive Dynamic Output Feedback Neural Network Control of Uncertain MIMO Nonlinear Systems With Prescribed Performance. IEEE Transactions on Neural Networks and Learning Systems, 2012. 23(1): p. 138-149.

[61] Yang, H. and J. Liu, An adaptive RBF neural network control method for a class of nonlinear systems. IEEE/CAA Journal of Automatica Sinica, 2018. 5(2): p. 457-462.

[62] Li, N., et al., Adaptive Semi-Periodically Intermittent and Lag Synchronization Control of Neural Networks With Mixed Delays. IEEE Access, 2018. 6: p. 4742-4749.

[63] Wang, Q., et al., Multiple models and neural networks based adaptive PID decoupling control of mine main fan switchover system. IET Control Theory & Applications, 2018. 12(4): p. 446-455.

[64] Hayakawa, T., W.M. Haddad, and N. Hovakimyan, Neural Network Adaptive Control for a Class of Nonlinear Uncertain Dynamical Systems With Asymptotic Stability Guarantees. IEEE Transactions on Neural Networks, 2008. 19(1): p. 80-89.

[65] Hayakawa, T., et al., Neural network adaptive control for nonlinear nonnegative dynamical systems. IEEE Transactions on Neural Networks, 2005. 16(2): p. 399-413.

[66] Cao, C. and N. Hovakimyan, Novel <formula formulatype="inline"><tex>${cal L}_{1}$</tex></formula> Neural Network Adaptive Control Architecture With Guaranteed Transient Performance. IEEE Transactions on Neural Networks, 2007. 18(4): p. 1160-1171.

[67] Werbos, P.J. Neural networks for control and system identification. in Proceedings of the 28th IEEE Conference on Decision and Control. 1989.

[68] Al-Dulaimi, A., et al., A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. Computers in Industry, 2019. 108: p. 186-196.

[69] Alamdari, B.V., A. Fatehi, and A. Khaki-Sedigh. Neural network model-based predictive control for multivariable nonlinear systems. in 2010 IEEE International Conference on Control Applications. 2010.

[70] Wang, Y., et al., Adaptive decoupling switching control based on generalised predictive control. IET Control Theory & Applications, 2012. 6(12): p. 1828-1841.

[71] Peng, K., et al., A Frequency Domain Decoupling Method and Multivariable Controller Design for Turbofan Engines. IEEE Access, 2017. 5: p. 27757-27766.

[72] Nguyen, T.N., S. Su, and H.T. Nguyen, Neural Network Based Diagonal Decoupling Control of Powered Wheelchair Systems. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2014. 22(2): p. 371-378.

[73] Cong, S. and Y. Liang, PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems. IEEE Transactions on Industrial Electronics, 2009. 56(10): p. 3872-3879.

[74] Hwang, C.L. and C. Jan, Recurrent-Neural-Network-Based Multivariable Adaptive Control for a Class of Nonlinear Dynamic Systems With Time-Varying Delay. IEEE Transactions on Neural Networks and Learning Systems, 2016. 27(2): p. 388-401.

[75] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[76] Merabet, A., A.A. Tanvir, and K. Beddek, Speed control of sensorless induction generator by artificial neural network in wind energy conversion system. IET Renewable Power Generation, 2016. 10(10): p. 1597-1606.

[77] Ruano, A.E.d.B., APPLICATIONS OF NEURAL NETWORKS TO CONTROL SYSTEMS. 1992, UNIVERSITY OF WALES.

[78] Xiao, X., et al., Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting Android malware with system call sequences. IET Information Security, 2017. 11(1): p. 8-15.

[79] Singh, K.R. and S. Chaudhury, Efficient technique for rice grain classification using back-propagation neural network and wavelet decomposition. IET Computer Vision, 2016. 10(8): p. 780-787.

[80] Wu, K., et al., A Novel Approach to Subpixel Land-Cover Change Detection Based on a Supervised Back-Propagation Neural Network for Remotely Sensed Images With Different Resolutions. IEEE Geoscience and Remote Sensing Letters, 2017. 14(10): p. 1750-1754.

[81] Wei, P., C. Cheng, and T. Liu, A Photonic Transducer-Based Optical Current Sensor Using Back-Propagation Neural Network. IEEE Photonics Technology Letters, 2016. 28(14): p. 1513-1516.

[82] Rumelhart, et al., Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations. 1986.

[83] Strickland, J.S., Data Science and Analytics for Ordicary People. 2015: Lulu Inc.

[84] Ahmed, J. and E.M. Alkhalifa. Efficient single layer handwritten digit recognition through an optimizing algorithm. in Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on. 2002.

[85] Zhao, Y., B. Deng, and Z. Wang. Analysis and study of perceptron to solve XOR problem. in The 2nd International Workshop on Autonomous Decentralized System, 2002. 2002.

[86] Zhang, Y., X. Wang, and E.G. Friedman, Memristor-Based Circuit Design for Multilayer Neural Networks. IEEE Transactions on Circuits and Systems I: Regular Papers, 2018. 65(2): p. 677-686.

[87] Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.

[88] Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.

[89] Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.

[90] Shu, H.-l., PID Neural Network and Its Control System. Defense Industry Press Pub, 2006.

[91] Dingguo, L.S.W.Z.W.G.W., Pid neural network sliding-mode controller design for three level based vsc-hvdc converter of offshore wind power. Proceedings of the CSEE, 2012. 32: p. 20-28.

[92] Xiao Deng, Z.-l.L., Chao Yang, Mu-yi Hu, Study on the control of black liquor level based on improved PID neural network. Transactions of China Pulp and Paper, 2014. 29: p. 58-62.

[93] Yu, Y., Y. Huang, and B. Zeng. A PID neural network controller. in Proceedings of the International Joint Conference on Neural Networks, 2003. 2003.

[94] Changhua, L. and S. Hua. Design of electric loading system in flight simulator based on PIDNN. in 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC). 2011.

[95] Guo, A., J. Li, and J. Yang. PIDNN decoupling control of doubly fed hydro-generator system based on PSO method. in Proceedings of the 30th Chinese Control Conference. 2011.

[96] Han, K., et al. Tracking control of ball and plate system using a improved PSO on-line training PID neural network. in 2012 IEEE International Conference on Mechatronics and Automation. 2012.

[97] Aiwen, G., Y. Jiandong, and B. Haiyan. PID neural network decoupling control for doubly fed hydro-generator system. in 2008 7th World Congress on Intelligent Control and Automation. 2008.

[98] Lin, L. and X. Peng. A PID neural network control for permanent magnet synchronous motor servo system. in 2010 5th International Conference on Computer Science & Education. 2010.

[99] Elamvazuthi, I., et al. An intelligent control of Blood Pressure system using PID and Neural Network. in 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA). 2013.

[100] Teng, W.f., H.p. Pan, and J. Ren. Neural network PID decoupling control based on chaos particle swarm optimization. in Proceedings of the 33rd Chinese Control Conference. 2014.

[101] Tian, Z., et al. A PID Neural Network Control for Position Servo System with Gear Box at Variable Load. in 2016 IEEE Vehicle Power and Propulsion Conference (VPPC). 2016.

[102] Yong, Z., Z. Hai-bo, and L. Tian-qi. PIDNN decoupling control of boiler combustion system based on MCS. in 2017 29th Chinese Control And Decision Conference (CCDC). 2017.

[103] Weijie, D., et al. Active Power Filter based on PIDNN controller. in IEEE PES Innovative Smart Grid Technologies. 2012.

[104] Ping, H. An analytical design of GAPIDNN algorithm for AQM. in Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC). 2013.

[105] Ge, S.L., et al. Design of PIDNN adaptive disturbance rejection decoupling controller. in 2017 Chinese Automation Congress (CAC). 2017.

[106] Huailin, S., G. Xiucai, and S. Hua. PID neural networks in multivariable systems. in Proceedings of the IEEE Internatinal Symposium on Intelligent Control. 2002.

[107] Vlachos, C., D. Williams, and J.B. Gomm, Solution to the Shell standard control problem using genetically tuned PID controllers. Control Engineering Practice, 2002. 10(2): p. 151-163.

[108] Cong, S., G. Li, and B. Ji, A NOVEL PID-LIKE NEURAL NETWORK CONTROLLER. IFAC Proceedings Volumes, 2005. 38(1): p. 121-126.

[109] Yazdizadeh, A., et al., Adaptive Neuro-PID Controller Design with Application to Nonlinear Water Level in NEKA Power Plant. Journal of Applied Sciences, 2009. 9(19): p. 3513-3521.

[110] Yazdizadeh, A., et al. Adaptive PID controller design with application to nonlinear water level in NEKA Power Plant. in 2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP). 2010.

[111] Mehrafrooz, A., M.R. Sorkhkolaei, and A. Yazdizadeh. Simultaneous application of adaptive PID controller and smith dead-time predictor rule in nonlinear water level control in Neka power plant. in 2011 6th IEEE Conference on Industrial Electronics and Applications. 2011.

[112] Shu, H. and Y.k. Xu. Application of additional momentum in PID neural network. in 2014 4th IEEE International Conference on Information Science and Technology. 2014.

[113] Meng, L., et al. Design of an improved PID neural network controller based on particle swarm optimazation. in 2015 Chinese Automation Congress (CAC). 2015.

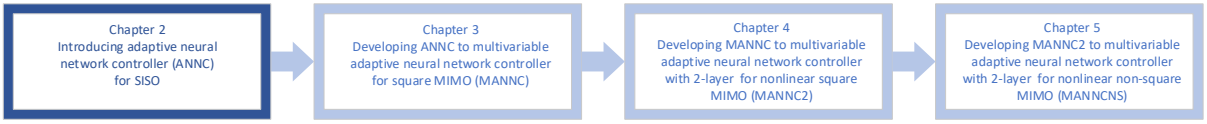
[114] Sento, A. and Y. Kitjaidure. Neural network controller based on PID using an extended Kalman filter algorithm for multi-variable non-linear control system. in 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI). 2016.

[115] Shan, W.j. and W. Tang. A neural network fractional order PID controller for FOLPD process. in 2016 35th Chinese Control Conference (CCC). 2016.

CHAPTER 2: INTRODUCING A MODEL-FREE ADAPTIVE NEURAL NETWORK AUTO-TUNED CONTROL METHOD FOR NONLINEAR SISO SYSTEMS

Aim

In this chapter, an introduction of a model-free Adaptive Neural Network Controller (ANNC) for SISO systems is presented in such a way that the proposed structure can establish a foundation upon which new adaptive neural network method for MIMO coupled systems in the following chapters as per diagram below:



Description

To achieve this aim, firstly the structure of a neural network-based adaptive controller including the overview of the used neurons is introduced. Then considering transfer function of the closed-loop system and stability analysis, the dynamic neural network learning method using accumulated gradient in the error back-propagation algorithm is described. For this purpose, a typical tank system is chosen as a sample representative of a non-linear SISO model-free system for which the designed controller is tested.

The above sections are described in a paper:

- Arash Mehrafrooz and Fangpo He, "Introducing a model-free adaptive neural network auto-tuned control method for nonlinear SISO systems" published in the proceeding of the IEEE International Conference on Information and Automation Wuyi Mountain, China, August 2018.

The outline of this paper is given below:

Results

The original contribution in this study is the development of a novel advanced control method including the neural network structure of the controller and associated learning algorithm. The obtained results clarify the capability of the chosen method to control model-free nonlinear SISO systems.

Conclusion

Overall, we show with the powerful learning abilities inherent in a neural network strategy, the proposed ANNC is capable of controlling model-free SISO systems with nonlinear properties to achieve the desirable control outcomes and has the ability of expanding this method for different types of multivariable systems in the following chapters.

Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems

Arash Mehrafrooz & Fangpo He
Advanced Control Systems Research Group
College of Science and Engineering, Flinders University
Adelaide, Australia
arash.mehrafrooz@flinders.edu.au & fangpo.he@flinders.edu.au

Abstract - In this study, a novel Adaptive Neural Networks Controller (ANNC) is proposed for controlling single-input single-output nonlinear systems. The proposed ANNC does not rely on an existing model of a system for its weights' training, and does make a full use of the history of the system input and output information for achieving a suitable control effect. The model of the system is used for checking the stability of the system after the calculation of the learning algorithm at each training step, and the controller weights are appropriately tuned to deliver a stable system during the entire training process. Using the accumulated gradient of the system error, the weights' adjustment convergence of the system can be observed and an optimal training number of the system can be selected. The effectiveness of the ANNC in controlling nonlinear industrial plants is demonstrated via simulation. The proposed control scheme provides a building block for the development of comparable schemes useful for more complicated systems involving multiple inputs and outputs.

Keywords - adaptive neural networks; model-free control; auto-tuning; error back-propagation; accumulated gradient; nonlinear systems; closed-loop stability

2.1 Introduction

Current advanced control methods are largely based on identified models of systems to be controlled [1]. In these methods, using an identified model of a system, a controller is designed to lead the output of the system to its desired value. A main problem of these methods in industrial applications is that real industrial systems are often highly nonlinear in nature. Consequently, a predicted (typically linear) model of such a system would be dynamically different than the real system itself, and a controller designed for the predicted

model of the system would have a limited effective working range. Furthermore, if a parameter of the system changes, the system may need to be remodelled, which may lead to the need to redesign the system controller in order to achieve the desired outcomes. Indeed, real-time implementations of such a control system would pose both high-complexities in realizing the required performances and high-demands in meeting the required computational powers. Due to these difficulties, it is not always feasible to control a real-life system using its identified model, if the system is either highly nonlinear or highly time-varying or both. As a result, model-free controllers are generally preferred for industrial applications. Clearly, if the modelling phase of a control system can be ignored, the time required for the adjustment of the real-time controller will be significantly shortened, which will result in a faster and more efficient control effect [1-3].

To adjust a controller's parameters automatically, adaptive control methods are required. A wide variety of adaptive control systems have been employed by various industries where variables of plants or processes are controlled in real-time using advanced control methods versus their classical counterparts [4-6]. By developing computer-based controllers online, industrial plants and processes become more and more effectively-automated. In fact, auto-tuned controllers with adaptive capabilities have become highly sought-after by industry, and their developments are growing rapidly.

As a versatile and universal method, Neural Network (NN) strategy has been found as a powerful approximator in the library of advanced control methods. An NN control system can work best by taking the advantage of the computational speed offered by modern computers and processors. Adding adaptive features to the NN schemes, adaptive NNs have been seen in controlling Single-Input Single-Output (SISO) systems successfully [7-10]. In these works, conventional PID controller structures are used, and PID coefficients are tuned by NN adaptive learning rules. For instance, in [7-9], an auto-tuned PID-like controller based on NN principles is proposed and the NN plays the role of estimating a suitable set of PID coefficients automatically. This method can be considered as an auto-tuning method which is suitable for linear systems or approximately-

linearized nonlinear systems. However, a few fundamental drawbacks are involved in this method. Firstly, the method depends on an existing model of a system in order to perform the auto-tuning of the system controller parameters, which means that the method does not offer the model-free characteristic favoured by industrial applications. Secondly, the method ignores the need to check the system stability criteria during the NN weights' training process, which means that the method may run the risk of producing an unstable closed-loop system in one or more steps of the auto-tuning process. Thirdly, the method relies on the use of conservative learning rules in order to keep a system in stable conditions, which means that the method is fundamentally limited in providing the utmost control performance with the fastest control speed. Fourthly, the method uses the current gradient of the system error in the NN learning algorithms and provides no information about the controller weights' adjustment convergences, which means that the method does not guarantee the adopted weights being the best possible adjustments to the controllers and nor does it provide a clear indication on what the optimum number of a training process is.

To overcome the above-mentioned deficiencies associated with the existing method, a new advanced adaptive NN control method is proposed in this paper. Firstly, while keeping the predicated model of a system (as the existing method does), the proposed method provides a model-free auto-tunning of the controller parameters in which the system is considered as a grey box and only the assessable input and output signals of the system are used. Secondly, the proposed method uses the predicated model of the system to check the stability criteria of the system during the controller weights' adjustment process, which safeguards the closed-loop stability of the overall system in each of the training steps. Thirdly, as the system dynamic stability is ensured, the proposed method uses a new online dynamic NN learning algorithm for the weights' training, which allows the closed-loop system to achieve its utmost control performance with a superior control speed. Fourthly, the proposed method uses the accumulated gradient of the system error in the NN learning algorithm, thus enabling the provision of the controller weights' adjustment convergence as well

as the identification of the suitable number of the training process.

The proposed method potentially possesses several key benefits over its existing counterpart. (i) By generating control actions independent to the model of a system, the proposed method is able to control nonlinear and/or time-varying systems (including servo systems) over a wide range of operating conditions. (ii) Since the system stability is checked in each training step, the proposed method can eliminate the potential unstable chances caused by, e.g., an inappropriate choice of the initial conditions of the controller parameters and/or an unwanted transient of the adopted adaptive learning algorithm. (iii) The new online dynamic NN learning algorithm used in the proposed method significantly improves the algorithms used in the existing counterpart [3, 8-10], and can provide a fast learning ability to the system. (iv) By applying accumulated gradient in the error back-propagation algorithm, the proposed method uses the history of the system's output together with the current weights to find the weights for the next step. In doing so, the proposed method can ensure the convergence of the weights' adjustments which is an important indicator for the implementation of the controller in practical applications.

To demonstrate the proposed method, the rest of the paper is organized as follows. In Section 2.2, the structure of the new NN-based adaptive controller is introduced. The transfer function of the closed-loop system used for the system stability analysis is presented in Section 2.3. In Section 2.4, the dynamic NN learning method using accumulated gradient in the error back-propagation algorithm is described. The system stability criteria using the proposed controller are investigated in Section 2.5. Section 2.6 contains the validation results in simulation where the proposed method is seen to control a highly-nonlinear SISO system effectively. Meaningful conclusions in relation to the design are drawn in Section 2.7.

2.2 Adaptive Neural Network Controller (ANNC)

A new structure of a dynamic neuron capable of creating dynamics for an adaptive controller is proposed in this paper. The neuron includes a number of inputs and outputs, and its general structure is illustrated in Figure 2-1. A

specific use of this neuron is identified by its activation function which defines the relationship between the outputs and the inputs, the number of inputs, the number of outputs, and the weight of each input that determines the influence of each input as compared with the other inputs.

If the neuron in Figure 2-1 is numbered as neuron number j, then the input equation of the neuron is described as:

$$net_j(k) = \sum_{i=1}^{n} w_{ji} x_i(k) \quad (1)$$

where n is the number of inputs, j is the neuron's unique number, k is the time sample number, net is the sum of weighted inputs, xi are the inputs, and w_{ji} are the weights to be trained. Expressing the activation function of the neuron by f, one has:

$$o_j(k) = f\left(net_j(k)\right) \quad (2)$$

where o_j is the neuron output. It is possible to use the neuron output as inputs to other parts of the neural networks. Therefore, one can have more than one output for each neuron. The introduced neuron is Multi-Input Multi-Output (MIMO) in its configuration, with its outputs being arranged to have the same value. Essentially, equation (2) describes the principal behaviour of the neuron and plays the key role in an NN structure. The proposed neuron can be readily used in different NN structures via definitions of the corresponding proper activation functions.

In the context of conventional PID control principles, the controller output is the sum of the proportional component, integral component, and derivative component of the input. This structure is used to construct the proposed controller, where distinguished Proportional-type (P-type), Integral-type (I-type), and Differential-type (D-type) neurons are presented. Each type of neurons is classified by its specific activation function which describes the behaviour of the particular functionality of the type of neurons.

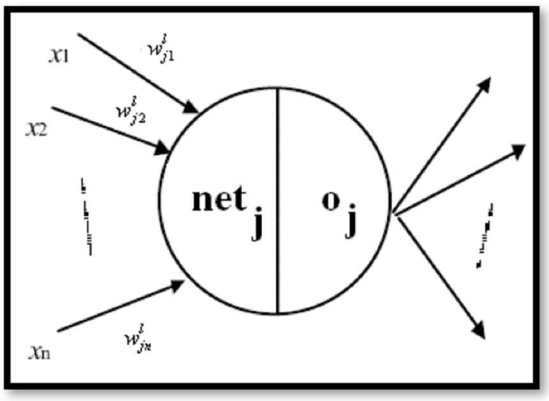


Figure 2-1. Schematic of a proposed neuron

2.2.1 P-type Neurons

The activation function of the P-type neurons is characterized in the continuous-time domain as:

$$o_j(t) = net_j(t) \tag{3}$$

and in the discrete-time domain as:

$$o_j(k) = net_j(k) \tag{4}$$

A neuron of this type only adds up the inputs by applying a weight to each of them. Based on Figure 2-1, the input of this neuron is described as:

$$net_j(k) = \sum_{i=1}^{n} w_{ji} x_i(k) \tag{5}$$

This neuron has a linear activation function and behaves as an adaptive weighted adder. In fact, a P-type neuron is considered as a "Unit Gain Block".

2.2.2 I-type Neurons

An I-type neuron acts as an integrator with its output being the integral of its weighted inputs. Its input-output relationship in the continuous and discrete forms are respectively described as:

$$o_j(t) = \int_0^t net_j(t')\, dt' \tag{6}$$

and

$$o_j(k) = \sum_{k'=1}^{k} net_j(k') \tag{7}$$

In the discrete form, it is possible to present the output as an addition of the

output in the previous sample together with the current input, expressed as:

$$o_j(k) = \sum_{k'=1}^{k} net_j(k') = \sum_{k'=1}^{k-1} net_j(k') + net_j(k) = o_j(k-1) + net_j(k) \quad (8)$$

2.2.3 D-type Neurons

A D-type neuron behaves as a derivative operator. The input-output relationship of the neuron in the continuous-time domain and the discrete-time domain are respectively presented as:

$$o_j(t) = \frac{d}{dt} net_j(t) \quad (9)$$

and

$$o_j(k) = net_j(k) - net_j(k-1) \quad (10)$$

The output of the neuron in the discrete form can be expressed as the difference between the current-sample input and the previous-sample input.

Upon definitions of the three basic-types of neurons as described above, these elements can be combined together to form dedicated controllers inspired by classic PI, PD, and PID controllers.

2.2.4 Proposed ANNC Structure

Considering the structure of a conventional PID controller, a new Adaptive Neural Network Controller (ANNC) structure is proposed in Figure 2-2. The ANNC contains six neurons and three layers. In the input layer, there are two P-type neurons which perform the distribution of the inputs in the constructed neural network. In the hidden layer, there are three neurons each being a P-type, an I-type, and a D-type, respectively; the P-neuron compares the desired output with the actual output; the I-neuron provides the necessary action to eliminate the steady-state error; the D-neuron predicts the future behavior of the error. In the output layer, there is a P-type neuron which performs the summation of the PID functionalities of the neuron. There are six weights ($\overline{w_{1,1}}$, $\overline{w_{1,2}}$, $\overline{w_{2,1}}$, $\overline{w_{2,2}}$, $\overline{w_{3,1}}$, and $\overline{w_{3,2}}$) in the hidden layer associated with the P-neuron, I-neuron, and D-neuron, respectively, and three weights (w_1, w_2, and w_3) in the output layer each dedicated to the output of the P-neuron, I-neuron, and D-

neuron, respectively. Applying the proposed ANNC structure to a SISO system, $G(s)$, the closed-loop system can be illustrated in Figure 2-3.

To demonstrate the capability of the proposed ANNC in replacing a conventional PID controller, the following values of the hidden-layer weights can be selected in order to generate the error for each of the hidden-layer neurons to be the difference between the system setpoint r and the system output y:

$\overline{w_{1,1}} = 1, \overline{w_{1,2}} = -1,$

$\overline{w_{2,1}} = 1, \overline{w_{2,2}} = -1,$

$\overline{w_{3,1}} = 1, \overline{w_{3,2}} = -1.$

Considering the output-layer weights as the parameters of a conventional PID controller, one would have: $w_1 = K_P$, $w_2 = K_I$, and $w_3 = K_D$, where K_P, K_I, and K_D denote respectively the proportional, integral, and derivative parameters of the conventional PID controller.

To evoke the adaptive feature of the proposed ANNC, however, the weights of the hidden layer are in fact deliberately left unlocked to +1 or -1 in order to allow the weights to be freely adjusted. The same arrangement is made to the output-layer weights as well, so that the overall structure of the proposed ANNC will be highly adjustable to suit the need of a real nonlinear and/or time-varying system setting. This specific capability of adjusting all nine weights simultaneously in real-time in the proposed ANNC is a significant advantage as compared to its conventional counterpart which has only three fixed (off-line tuned) parameters and could only ideally work for linear or linearized systems. In comparison with the existing adaptive NN-PID controllers (e.g., [8]) in which only three parameters can be adjusted online, the proposed ANNC offers a much higher level of flexibilities to allow a control system to achieve a better control outcome. Incorporating with a properly-selected auto-tuning method, a larger degree of freedom provided by the larger number of weights in the proposed ANNC structure would bring direct benefits to the overall control system performance in terms of fast-gaining the desired output of the system while keeping the system in its stable condition.

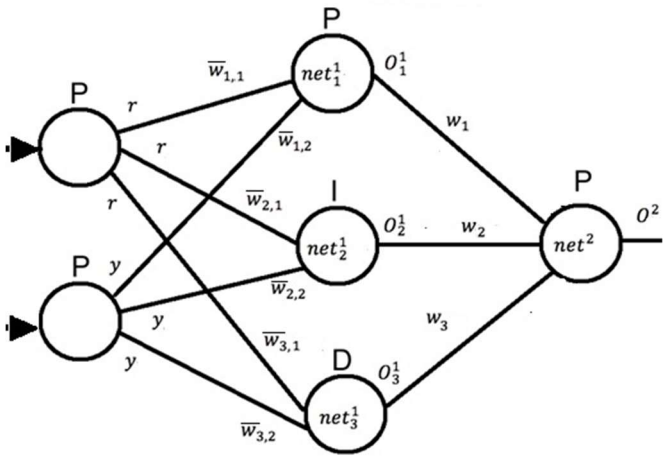


Figure 2-2. Proposed ANNC structure

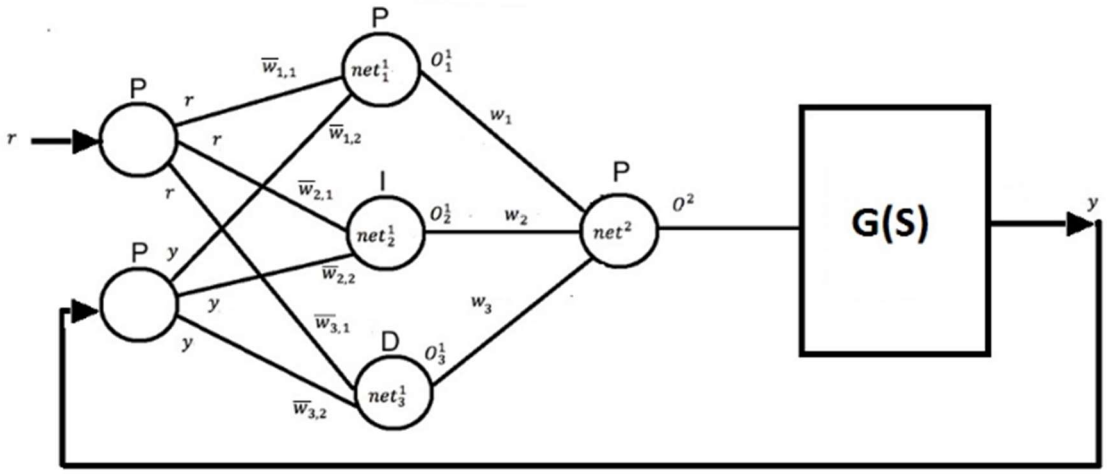


Figure 2-3. Applying Proposed ANNC to SISO system

2.3 Transfer function Derivation

The closed-loop transfer function of a SISO system using the proposed ANNC, as illustrated in Figure 2-3, is derived in this section in order to establish an analysis base upon which investigations of the closed-loop control system stability can be conducted in Section V of this paper. Let $G(s)$ be the estimated linear model of the SISO system. The output of the system can be expressed as:

$$Y(s) = G(s)O^2(s) \tag{11}$$

Since the output-layer neuron is a P-type neuron, one has:

$$O^2(s) = NET^2(s) \tag{12}$$

and

$$NET^2(s) = w_1\, O_1^1(s) + w_2\, O_2^1(s) + w_3\, O_3^1(s) \quad (13)$$

As the hidden layer has P-type, I-type, and D-type neurons, the following relationships are established:

$$O_1^1(s) = NET_1^1(s) \quad (14)$$

$$O_2^1(s) = s.NET_2^1(s) \quad (15)$$

$$O_3^1(s) = \frac{1}{s} NET_3^1(s) \quad (16)$$

Hence:

$$O_1^1(s) = \overline{w_{1,1}} R(s) + \overline{w_{1,2}} Y(s) \quad (17)$$

$$O_2^1(s) = s\left(\overline{w_{2,1}} R(s) + \overline{w_{2,2}} Y(s)\right) \quad (18)$$

$$O_3^1(s) = \frac{\left(\overline{w_{3,1}} R(s) + \overline{w_{3,2}} Y(s)\right)}{s} \quad (19)$$

Substituting (12)-(19) into (11), one obtains:

$$Y(s) = Y(s)G(s)\left(w_1\overline{w_{1,2}} + sw_2\overline{w_{2,2}} + \frac{w_3\overline{w_{3,2}}}{s}\right) +$$

$$R(s)G(s)\left(w_1\overline{w_{1,1}} + sw_2\overline{w_{2,1}} + \frac{w_3\overline{w_{3,1}}}{s}\right) \quad (20)$$

The closed-loop transfer function of the system then becomes:

$$\frac{Y(s)}{R(s)} = \frac{G(s)\left(w_1\overline{w_{1,1}} + sw_2\overline{w_{2,1}} + \frac{w_3\overline{w_{3,1}}}{s}\right)}{1 - G(s)\left(w_1\overline{w_{1,2}} + sw_2\overline{w_{2,2}} + \frac{w_3\overline{w_{3,2}}}{s}\right)} \quad (21)$$

2.4 Learning algorithm

To achieve a precise control effect for the SISO system, the ANNC weights are adjusted using the principle of the multi-step error back-propagation algorithm [11]. However, instead of using merely the current gradient of the system error as the literature does [8-10], the accumulated gradient of the system error is proposed to be used in this paper to achieve a better control result. The proposed method minimises the squared accumulated gradient of the system error, where the system error is taken as the difference between the desired system output $r(k)$ (i.e., the system setpoint) and the actual system output $y(k)$. Euclidean Norm J is defined for calculating the quadratic cost function of the system for the

system error. Power of two in this expression makes the error of each sample positive so that larger errors will be more significant. The cost function is thus defined as:

$$J(k) = \frac{1}{2}\big(r(k) - y(k)\big)^2 \tag{22}$$

Using the accumulated gradient of the system error, a learning algorithm must be designed to minimize the system error and to bring the system output as close as possible to the system setpoint. According to the principle of the error back-propagation learning algorithm, the output-layer weights must be adjusted so that in each step they move slightly in the opposite direction of the gradient of the cost function. This is to ensure that the cost function will be decreasing step by step. The weight x between the hidden layer and the output layer will therefore be adjusted based on the following learning rule:

$$w_x(h+1) = w_x(h) - \frac{\lambda}{m}\sum_{k=1}^{m}\frac{\partial J}{\partial w_x} \tag{23}$$

where $1 \leq x \leq 3$; h is the step number of the learning algorithm; $w_x(h)$ and $w_x(h+1)$ are the weights of the output layer in the current and next steps, respectively; λ decides how fast the cost is changing and particularly determines the weight adjustment learning speed; m is the required number of discrete samples in the system output and setpoint. By increasing m, the system output will be compared more accurately with the system setpoint. However, a large value of m may decrease the adjustment speed of the controller and become undesirable when the speed of the control system (often a critical requirement for a real-time industrial system) is of concern. Therefore, a reasonable value of m must be used to make a necessary trade-off between the desired accuracy and the required speed of the control system by considering the nature of the application of concern.

The gradient of the error subject to each weight is required to be calculated. Using partial derivatives, one has:

$$\frac{\partial J}{\partial w_x} = \frac{\partial J}{\partial y} \times \frac{\partial y}{\partial o^2} \times \frac{\partial o^2}{\partial net^2} \times \frac{\partial net^2}{\partial w_x} \tag{24}$$

where:

$$\frac{\partial J}{\partial y} = y(k) - r(k) \tag{25}$$

$$\frac{\partial y}{\partial o^2} \cong \frac{y(k)-y(k-1)}{o^2(k)-o^2(k-1)} \tag{26}$$

$$\frac{\partial o^2}{\partial net^2} = 1 \tag{27}$$

as the output neuron is a P-type neuron, and

$$\frac{\partial net^2}{\partial w_x} = o_x^1(k) \tag{28}$$

Substituting (25)-(28) into (24) yields:

$$\frac{\partial J}{\partial w_x} \cong \big(y(k) - r(k)\big) \times \frac{y(k)-y(k-1)}{o^2(k)-o^2(k-1)} \times o_x^1(k) \tag{29}$$

Defining $\gamma(k)$ as:

$$\gamma(k) = \big(y(k) - r(k)\big) \times \frac{y(k)-y(k-1)}{o^2(k)-o^2(k-1)} \tag{30}$$

the output-layer weight adjustment rule can then be derived as:

$$w_x(h+1) = w_x(h) - \frac{\lambda}{m} \sum_{k=1}^{m} [\gamma(k) \times o_x^1(k)] \tag{31}$$

As equation (31) is used for calculation of the weights in the next step, therefore equal sign has been used rather than approximately equal sign. Using this learning method, all weights are simultaneously tuned, and their values at the current step, together with the current input and current output, are used for the weights' training at the next step. This online dynamic learning feature of the proposed method makes it stand out from the current existing counterpart pool [7, 8]. It should be noted that the weights of the hidden layer are preserved to guarantee the stability of the system according the stability conditions defined in the following section. It should be note that the number of weight trainings depends on the specific properties of the application. The weight adjustment process can be stopped after reaching a specific error, settling time, and any other measurable specification of the response.

2.5 Stability Analysis for first order system

The stability of the closed-loop system using the proposed ANNC structure is analyzed in this section to determine the criteria by which the controller's free

weights are tuned to deliver the desired output performance while the closed-loop system is maintained in its stable region. For this purpose, the first-order Taylor series approximation for nonlinear system dynamics about an operating point is used. Generally, a first order linear model of the SISO system can be expressed as:

$$G(s) = \frac{\alpha}{S+\beta} \tag{32}$$

where $\frac{1}{\beta}$ is the time constant and $\frac{\alpha}{\beta}$ is the gain of the first order system. Substituting (32) into (21), the closed-loop transfer function of the first-order linear model of the system can be expressed as:

$$H(s) = \frac{\frac{\alpha}{S+\beta}\left(w_1\overline{w_{1,1}}+sw_2\overline{w_{2,1}}+\frac{w_3\overline{w_{3,1}}}{s}\right)}{1-\frac{\alpha}{S+\beta}(w_1\overline{w_{1,2}}+sw_2\overline{w_{2,2}}+\frac{w_3\overline{w_{3,2}}}{s})} \tag{33}$$

which leads to:

$$H(s) = \frac{\alpha w_2\overline{w_{2,1}}s^2+\alpha w_1\overline{w_{1,1}}s+\alpha w_3\overline{w_{3,1}}}{(1-\alpha w_2\overline{w_{2,2}})s^2+(\beta-\alpha w_1\overline{w_{1,2}})s-\alpha w_3\overline{w_{3,2}}} \tag{34}$$

The characteristic equation of system (34) is represented by:

$$\left(1-\alpha w_2\overline{w_{2,2}}\right)s^2+\left(\beta-\alpha w_1\overline{w_{1,2}}\right)s-\alpha w_3\overline{w_{3,2}}=0 \tag{35}$$

According to the Routh-Hurwitz stability criterion for a generic quadratic polynomial [12], the conditions for the stability of system (34) are derived as:

$$\left(1-\alpha w_2\overline{w_{2,2}}\right)\left(-\alpha w_3\overline{w_{3,2}}\right)>0 \tag{36}$$

and

$$(\beta-\alpha w_1\overline{w_{1,2}})(-\alpha w_3\overline{w_{3,2}})>0 \tag{37}$$

Conditions (36)-(37) form the closed-loop stability criteria for system (34). They are used when the initial values of the controller weights are selected and after each-step calculation of the learning algorithm. They ensure that the selected weights can indeed deliver a stable closed-loop system at each learning step, and that the overall system can remain stable constantly during the entire training process.

2.6 Simulation results

To investigate the performance of the ANNC proposed in Section 2.2 together with the dynamic NN learning algorithm developed in Section 2.4, simulation studies are conducted where the plant is chosen as a typical liquid tank system. The level of the liquid within the tank is to be controlled via controlling the input pump flow of the tank. The input pump injects fluid into the tank from the top (non-gravitational filling), and the injection speed can be varied by an input voltage applied to the pump. Such a system is very popular and often challenging in industrial applications, as the presented system is highly nonlinear in nature. The output tracking behavior of the system using the proposed control method is compared with that using a pre-designed fixed-parameter PID controller suggested by Matlab and that using an existing adaptive NN-PID controller suggested in [8].

Assume that the pump's output (the flow rate) is linearly related to the pump's input current. The changes in the fluid level inside the tank depend on the difference between the input flow to the tank and the output flow from the tank. The rate of change in the liquid volume inside the tank is therefore described as:

$$\frac{dvol}{dt} = b.c - a\sqrt{h} \tag{38}$$

where vol is the tank's liquid volume, b depends on the input flow rate linearly (i.e., a coefficient determined by the pump's characteristics), c is the control signal applied to the control valve with a magnitude of 4mA to 20mA, a is the characteristics coefficient of the tank's output pipe, and h is the height of the liquid in the tank. By having:

$$\frac{dvol}{dt} = A\frac{dh}{dt} \tag{39}$$

where A is the area of the tank's base, one has:

$$\frac{dh}{dt} = \frac{b.c}{A} - \frac{a\sqrt{h}}{A} \tag{40}$$

Clearly, the term $\sqrt{h}$ shown in (40) makes the liquid tank system highly nonlinear and difficult to control. Assuming the tank has a circular base of 10 meters in diameter and a height of 12 meters, and considering the pump has a

control valve at a rate of 50m^3/hr, the desired level of the liquid in the tank, h_0, is set to be 5 meters. Let the pump starts at time $t = t_0$. The nonlinear differential equation describing the relationship between the input c and output h of the system is then expressed as:

$$\frac{dh}{dt} = \frac{b.c}{A}\mathrm{u}(\mathrm{t} - t_0) - \left(\frac{a\sqrt{h}}{A}\right) \tag{41}$$

where u(t) is the unit step function.

Applying the proposed ANNC to the liquid tank system, the overall control system is illustrated in Figure 2-4. For comparison purposes, the parameters of a Matlab pre-designed PID controller are determined as $K_P = 2$, $K_I = 3$, and $K_D = 0.2$. Applying the existing adaptive NN-PID control method suggested in [8] (where the current gradient of the system error is used instead), the weights of the hidden layer are obtained as $\mathrm{w}_1 = 4.598$, $\mathrm{w}_2 = 4.901$, and $\mathrm{w}_3 = 4.303$. The performance of the ANNC controlled system is compared with that of the above specified fixed-parameter PID controlled system together with that of the above obtained existing adaptive NN-PID controlled system. The simulation results are given in Figure 2-5.

Figure 2-5 demonstrates the liquid level control effects of the three control schemes when the tank system is subjected to a step change in its setpoint. According to the simulation result of the fixed-parameter PID controlled system, when the input pump starts, the water level increases to 7.88m which represents a significant overshoot that may result in overflowing of the liquid from the tank.

To apply the ANNC to the liquid tank system, the initial values of the output-layer weights are set to be: $\mathrm{w}_1 = 1$, $\mathrm{w}_2 = 1$, and $\mathrm{w}_3 = 1$. The weights of the hidden layer for a reasonable comparison with the fixed-parameter PID controller are selected as follows to generate the error signal as the difference between the desired system output and the actual system output:

$\overline{w_{1,1}} = 1$, $\overline{w_{1,2}} = -1$,

$\overline{w_{2,1}} = 1$, $\overline{w_{2,2}} = -1$,

$\overline{w_{3,1}} = 1$, $\overline{w_{3,2}} = -1$.

After repeating the online and simultaneous learning algorithm on the output-layer weights for 100 times, the following values are achieved:

$w_1 = 5.848$,

$w_2 = 3.941$,

$w_3 = 5.622$.

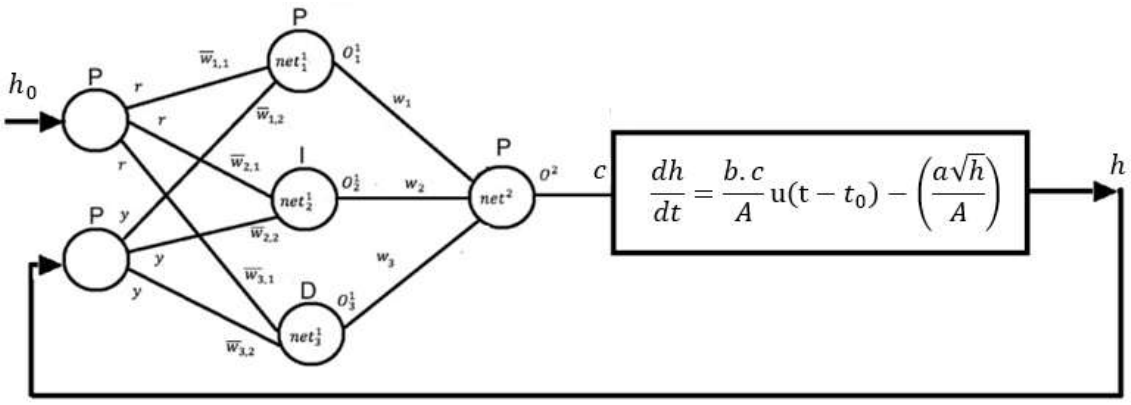


Figure 2-4. The ANNC controlled liquid tank system

While the weight learning algorithm of the ANNC was running, the closed-loop stability of the system was checked at each learning step according to the inequalities mentioned in (36)-(37). It is observed that the selected weights have kept the system stable at all time, and that the hidden-layer weights remain constant as their selected initial values. If the inequalities were not satisfied, it would be required to change the hidden-layer weights according to the mentioned stability criteria in (36)-(37).

As shown in Figure 2-5, compared to the ANNC controlled system, the classical fixed-parameter PID controlled system and the existing adaptive NN-PID controlled system have much larger system errors, particularly in the overshoot and undershoot time periods. By using the ANNC, however, the overshoot and undershoot can be reduced, respectively, from 57.6% and 35.2% to 28.1% and from 27.2% and 12.2% to 8.5%. In addition, the settling time of the liquid level can be reduced from 29 and 27 time-samples to 23 time-samples.

Figure 2-6 represents the accumulated error versus the iteration number of the ANNC weight learning algorithm. The error shown in this figure is the difference

between the actual system output (the real liquid-level) and the desired system output (the liquid-level of 5 meters). It is observed that, with the ANNC, the magnitude of the error decreases as the number of iterations in the weight learning algorithm increases. This is an important result which demonstrates the suitable performance of the proposed ANNC for the considered tank system. As shown in this figure, the error reduction rate slows down as the number of iterations reaches to 50 (the fraction point in Figure 2-6). Hence, the optimal training number of the learning algorithm is around 50 for this particular system, which indicates the trade-off between the control speed and the control performance of the ANNC controlled system.

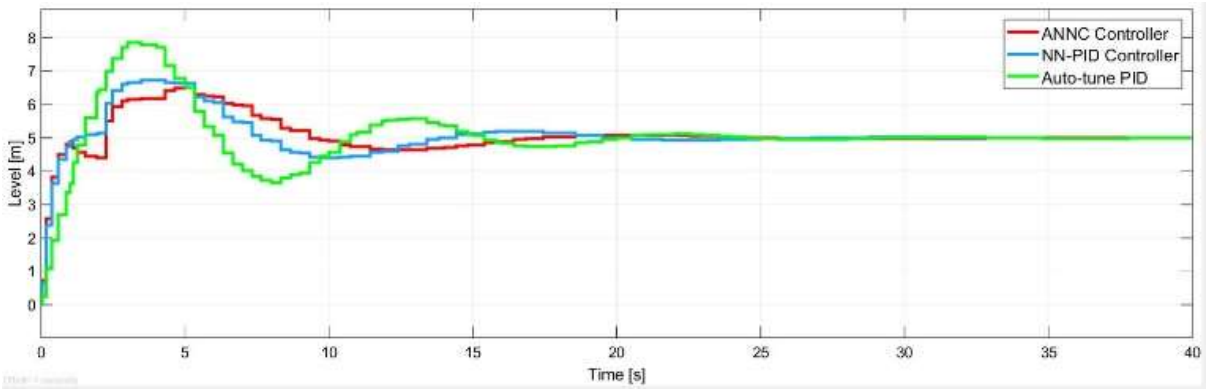


Figure 2-5. Tank level control performance comparison using the three schemes

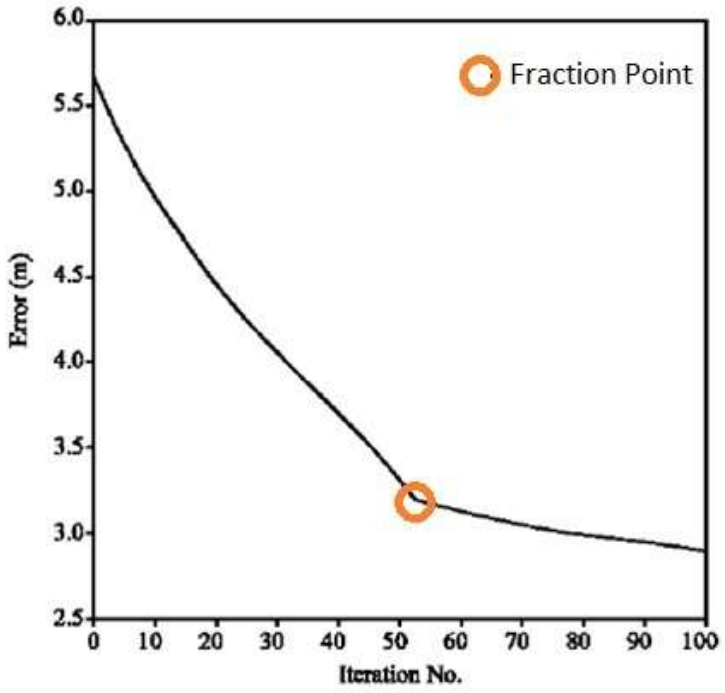


Figure 2-6. ANNC accumulated error vs. number of trainings

2.7 Conclusion

With the powerful learning abilities inherent in a neural network strategy, the proposed ANNC is capable of controlling SISO systems with nonlinear properties. The ANNC uses a new auto-tune dynamic online learning algorithm with accumulated error back-propagation for the proposed neural network structure, and effectively tunes its parameters to achieve the desirable control outcomes. By checking the stability of the system at each step of the controller parameters' update, the closed-loop system is guaranteed to remain stable during the entire controller training process. The effectiveness of the proposed ANNC is validated via simulation studies. When compared with the representatives of existing counterparts, the ANNC is seen to provide a better and faster system performance. By selecting an appropriate number of samples, the ANNC can be effectively used for several types of control systems in industrial applications, such as flow, level, and temperature control systems in water or wastewater plants.

Although the proposed ANNC currently discussed in this paper is in the context of SISO systems and PID control strategies, it is developed in such a way that the proposed structure can establish a foundation upon which new adaptive NN methods for MIMO coupled systems and/or other types of control laws can be readily derived. The aspects of those studies will be revealed in the authors' future papers.

References

[1] *Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.*

[2] *Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.*

[3] *Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.*

[4] *Astrom, K.J. and T. Hagglund, PID Controllers: Theory, Design and Tuning. 2 ed. 1995, USA: Instrument Society of America.*

[5] *Astrom, K.J. and B. Wittenmark, Adaptive Control Stability Convergence and Robustness. 1989, University of California: Prentice Hall Information and System Science Series.*

[6] *Farivar, F., et al., Hybrid Control of Flexible Manipulator. Journal of Applied Sciences, 2009. 9(4): p. 639-650.*

[7] *Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.*

[8] *Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.*

[9] *Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.*

[10] *Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.*

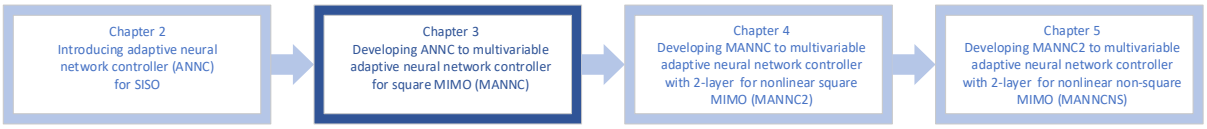
[11] *Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.*

[12] *Apte, Y.S., Routh-Hurwitz stability criterion and its equivalents. India, IEE-IERE Proceedings -, 1969. 7(4): p. 149-154.*

CHAPTER 3: INTRODUCING A NOVEL MODEL-FREE MULTIVARIABLE ADAPTIVE NEURAL NETWORK CONTROLLER (MANNC) FOR SQUARE MIMO SYSTEMS

Aim

In this chapter, using the foundation of an Adaptive Neural Network Controller (ANNC) introduced for controlling SISO systems in Chapter 2, a Multivariable Adaptive Neural Network Controller (MANNC) for controlling MIMO systems is represented in such a way that the proposed structure is able to control coupled square MIMO systems without any knowledge of the systems' models. The represented controller will be expandable to include two dynamic layers, the development of which will be discussed in Chapters 4 and 5 as per diagram below:



Description

To achieve this aim, the structure of a multivariable neural network adaptive controller for closed-loop model-free control systems is firstly built using the basic structure of the ANNC controller introduced in Chapter 2. It is noted that the structure of the MANNC is developed in such a way that can be extended into a greater number of adjustable layers with a different number of inputs and outputs, so that the proposed structure will be able to be further modified in the following chapters. Subsequently, a novel dynamic learning method for the neural network structure with one adjustable layer and using the error back-propagation algorithm is designed, which utilizes the real-time history of the actual system inputs and outputs rather than the model of the system. The learning algorithm for the neural network weights of the proposed MANNC is developed, and its associated Lyapunov stability analysis is conducted to ensure the global asymptotical stability of the resulting control system during the training process. In this chapter, to demonstrate that the proposed MANNC is

able to perform as a general controller for a wide range of square MIMO systems, three different cases of coupled nonlinear MIMO systems are chosen: a time-invariant drum boiler system, a sample nonlinear time-variant system, and a nonlinear hybrid two-tank system. Simulation studies for all three cases are carried out in which the designed controller is tested and compared with the existing methods. The above-mentioned contents are described in a paper:

- Arash Mehrafrooz and Fangpo He, "Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems", Submitted to journal of "IEEE Transactions on Automatic Control".

The outline of this paper is given below:

Results

To improve the ANNC (introduced in Chapter 2) that is only applicable to SISO systems and its stability is checked by the traditional Routh-Hurwitz stability

criterion, a novel advanced control method comprising a neural network controller (MANNC) and its associated learning algorithm together with its Lyapunov stability constraint is developed in this chapter for square MIMO systems. This development constitutes the original contribution of this chapter. The obtained simulation results justify that the proposed MANNC can achieve an appropriate set-point tracking of all outputs of a square MIMO system via the controller's powerful auto-tuning capability, while keeping the system stable during the entire turning process.

Conclusion

Overall, this chapter demonstrates that with the powerful learning abilities inherent in a neural network strategy, the proposed MANNC is capable of controlling model-free square coupled MIMO systems with nonlinear properties and achieving the desired control outcomes. The MANNC can be effectively used for a wide range of square MIMO control systems in industrial applications, especially for those systems whose outputs' overshoots are unwanted and for which a fast set-point tracking is desirable. The proposed MANNC will be further improved in the next chapter by changing its neural network structure and defining new learning algorithms to be more reliable for MIMO systems with nonlinearities.

Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems

[1]*Advanced Control Systems Research Group, College of Science and Engineering, Flinders University, Australia*

Arash Mehrafrooz[1, a], Fangpo He[1, b]

[a]<*arash.mehrafrooz@flinders.edu.au*>, [b]<*fangpo.he@flinders.edu.au*>

***Abstract* - In this study, a novel Multivariable Adaptive Neural Network Controller (MANNC) is developed for coupled model-free *n*-input *n*-output systems. The learning algorithm of the proposed controller does not rely on the model of a system, and uses only the history of the system inputs and outputs. The system is considered as a 'black box' with no pre-knowledge of its internal structure. By online monitoring and possessing the system inputs and outputs, the parameters of the controller are adjusted. Using the accumulated gradient of the system error along with the Lyapunov stability analysis, the weights' adjustment convergence of the controller can be observed and an optimal training number of the controller can be selected. The Lyapunov stability of the system is checked during the entire weight training process to enable the controller to handle any possible nonlinearities of the system. The effectiveness of the MANNC in controlling nonlinear square multiple-input multiple-output (MIMO) systems is demonstrated via three simulation studies covering the cases of a time-invariant nonlinear MIMO system, a time-variant nonlinear MIMO system, and a hybrid MIMO system, respectively. In each case, the performance of the MANNC is compared with that of a properly-selected existing counterpart. Simulation results demonstrate that the proposed MANNC is capable of controlling various types of square MIMO systems with much improved performance over its existing counterpart. The unique properties of the MANNC will make it a suitable candidate for many industrial applications.**

Key Words - square MIMO; model free control; neural network control; learning algorithm; Lyapunov Stability;

3.1 Introduction

Over the past few years, there has been a significant improvement in controlling Multiple-Input Multiple-Output (MIMO) systems using adaptive control methods

[1]. Many proposed adaptive controllers rely on model-based approaches [2, 3] where mathematical models of the respective dynamic systems must be identified either directly or indirectly in advance. For majority industry applications in practice, however, there exist major challenges in relation to MIMO systems' model identifications. For instance, a predicted model of an industrial plant can be dynamically different than the true plant itself, due largely to the plant's structural uncertainties, unmodelled nonlinearities, and time-varying natures [4-6]. In some cases, if a constraint of an actual system for any reason changes, in order to achieve the desired outcomes, the system's model may need to be re-identified resulting in a redesigning of the corresponding controller [7]. Even at the circumstances where an exact model of a MIMO system could be identified, the controller designed for the predicted model of the system may still be subjected to conditional variations both internal and external to the system [8]. Due to these practical problems associated with model-based approaches, many existing adaptive control schemes are seen to be impractical or limited in controlling real industrial MIMO plants. In contrast to model-based approaches, model-free approaches [9-12] can entirely omit the modelling phase of a system, thus significantly reducing the time required for the design and tuning of the system's real-time controller. This will result in a faster and more precise control outcome [13-15]. Due to this consideration, model-free controllers are becoming more preferable than their corresponding model-based counterparts, especially for industrial MIMO control applications where the modelling phase of a true plant can be time-consuming and inaccurate. Among the recently introduced model-free control methods [13, 15-19], the neural network technique stands out as a powerful and practical tool for controlling MIMO systems due mainly to its astonishing capabilities in dealing with large volumes of data, estimating ambiguous relationships between a system's inputs and outputs, and predicting future behaviours of a system.

Recently, by adding adaptive features to neural network schemes, adaptive neural networks have been seen in controlling MIMO systems successfully [20-26]. Many adaptive control methods based on neural networks have been introduced to Single-Input Single-Output (SISO) systems and further developed

for Uncoupled Multiple-Input Multiple-Output (U-MIMO) systems [27-30]. In [23], a neural-network controller and its associated learning rules are proposed which can be successfully applied to Single-Input Multi-Output (SIMO) plants. This controller is a combination of several SISO controllers cascaded together and, therefore, is unable to be further developed for coupled MIMO systems. Since in general control problems are more challenging if cross-couplings among the various inputs and outputs of a MIMO system exist, model-free control of coupled MIMO systems has become an active area of research with a growing number of publications [31-37]. Despite some improvements, however, neural network based controllers have not been extensively used in industrial model-free control systems due to the following apparent deficiencies [38]:

- During the weight training process of the neural networks, the controlled systems can become unstable;
- It is not always clear when to stop the weight training process;
- A long training time for the weights can be unsatisfactory for the speed of the control systems;
- The traditional activation functions employed in the neural networks may not be suitable for control purposes;
- The common error back-propagation learning algorithm uses only the last two consecutive samples of the outputs in discrete derivative functions, and does not comply with the requirement of a proper model-free approach in which a full history of inputs and outputs must be used in order to generate an effective control action.

In order to overcome the above-listed deficiencies, in this study, a novel model-free Multivariable Adaptive Neural Network Controller (MANNC) is proposed for controlling coupled MIMO systems in which:

- By using the constraint generated from the Lyapunov stability conditions at each step of the online weight training process, the overall control system stability can be guaranteed at all time – This eliminates the risk of the system been falling into its local minimum [39, 40] and prevents the

loss of the control speed due to conservative learning rate selections [15, 18, 25, 40, 41];

- By constantly observing the accumulated errors and comparing them with their desired values, the controller can decide to stop the learning algorithm and lock the neural network weights at an optimal point – This ensures the convergence of the controller weight adjustments and provides a clear optimal number for the weight training steps;
- By choosing proper initial learning rates and dynamically changing them during the learning process according to the system stability criteria, the weight training speed can be significantly increased – This forms a clear comparison with and improvement over the traditional static learning rates [15, 18, 25, 40, 41];
- By designing specific activation functions that utilize typical proportional, integral, and derivative operations in the neural network structure of the controller, the proposed controller is simple and straightforward in its configuration – This makes the controller a potential candidate suitable for replacing classical PID controllers in industrial applications;
- By applying accumulated gradients in the error back-propagation algorithm and using new partial derivative estimations, the proposed method fully uses the history of the system outputs together with the current weights to produce the outputs of the controller (the inputs of the system) for the next step – This new learning method significantly reduces the overshoot and settling time of the system by minimising the summation of errors of the system outputs in each step rather than using only the last two consecutive samples of the system outputs as its traditional counterparts do [39, 42-45], and allows the closed-loop system to achieve its best control performance with a minimum number of weight training steps.

It is anticipated that, being truly model-free, the proposed MANNC can generate adequate control actions over a wide range of operating conditions. Also, by using a new cross-coupling network structure, the proposed controller is expected to be able to control strongly-coupled MIMO systems effectively. It should be

mentioned that, according to the design to be presented in this paper, the proposed MANNC will only be applicable for square $(n \times n)$ MIMO systems. If an industrial MIMO system has different numbers of inputs and outputs, by squaring up (adding) or squaring down (removing) the inputs (i.e., the manipulated variables) or the outputs (i.e., the controlled variables), the given non-square MIMO system can be rewritten as a corresponding square MIMO system [46] to which the proposed MANNC will be applicable.

To reveal the proposed MANNC, the rest of this paper is organised as follows. In Section 3.2, the structure and matrix representation of the new neural network based adaptive controller are introduced. In Section 3.3, the new learning method of the neural network based on the error back-propagation algorithm is described. The closed-loop system stability is analysed in Section 3.4. In Section 3.5, the controlled system and the introduced method are specified for SISO systems. Section 3.6 demonstrates the validation results via simulation studies where the proposed method is seen to control three chosen nonlinear MIMO systems without the use of their respective models. Conclusions in relation to the design are drawn in Section 3.7.

3.2 Multivariable Adaptive Neural Networks Controller (MANNC)

3.2.1 Closed-loop Structure of MANNC

As previously proposed in [18] by the authors, the outputs of three types of neurons: P-type, I-type, and D-type, in the discrete form can be respectively expressed as:

$$o_P(k) = net_P(k) \tag{1}$$

$$o_I(k) = o_I(k-1) + net_I(k) \tag{2}$$

$$o_D(k) = net_D(k) - net_D(k-1) \tag{3}$$

where $o_X(k)$ and $net_X(k)$ represent the X-type neuron's output and input at the k^{th} sample time, respectively. In this study, for stability concerns, further constraints to the neurons' activation functions of (1)-(3) are applied as follows:

$$o_P(k) = \begin{cases} 1 & net_P(k) > 1 \\ net_P(k) & -1 \le net_P(k) \le 1 \\ -1 & net_P(k) < -1 \end{cases} \quad (4)$$

$$o_I(k) = \begin{cases} 1 & net_I(k) > 1 \\ o_I(k-1) + net_I(k) & -1 \le net_I(k) \le 1 \\ -1 & net_I(k) < -1 \end{cases} \quad (5)$$

$$o_D(k) = \begin{cases} 1 & net_D(k) > 1 \\ net_D(k) - net_D(k-1) & -1 \le net_D(k) \le 1 \\ -1 & net_D(k) < -1 \end{cases} \quad (6)$$

Considering the structure of the Adaptive Neural Network Controller (ANNC) proposed in [18], a new Multivariable Adaptive Neural Network Controller (MANNC) is proposed in Figure 3-1 as a closed-loop MIMO controller applicable to coupled square $(n \times n)$ multivariable systems.

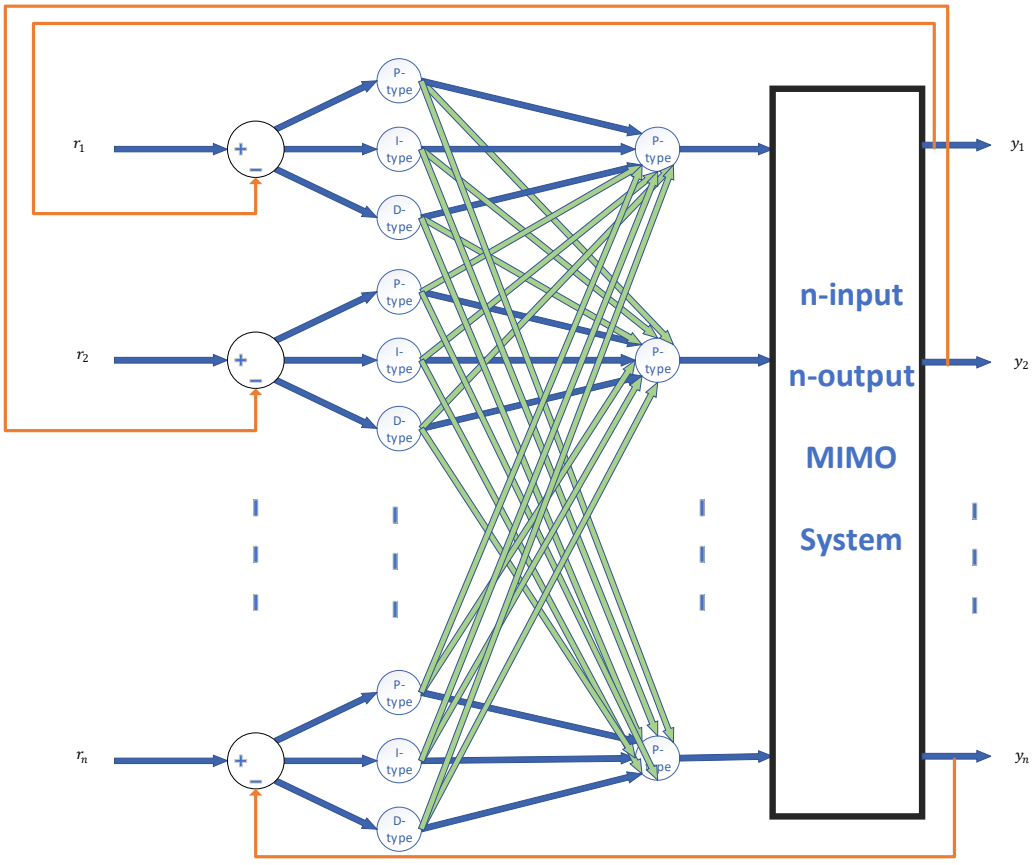


Figure 3-1. Structure of the proposed MANNC

Assuming that there are strong cross-coupling between the n inputs and n outputs of a MIMO system of concern, the proposed MANNC is designed using a $n - 3n - n$ neural network structure. In this structure, the error for each output

(i.e., the difference between each desired output (r_i) and its corresponding actual output (y_i) ($i = 1, 2, \dots, n$)) is generated, and the generated errors propagate to the two layers, the hidden layer and the output layer, of the MANNC network. In the hidden layer, there are $3n$ neurons including clusters of P-type, I-type, and D-type neurons that are repeated consecutively. In the output layer, there are n P-type neurons that form the outputs of the MANNC, i.e., the inputs of the $n \times n$ MIMO system. There are $3n^2$ weights in the output layer that are associated with the hidden-layer neurons and decide the impact of each neuron of the hidden layer on the generation of the inputs applied to the MIMO system.

3.2.2 Structure of Sub-MANNC (S-MANNC)

To be able to deal with the cross-couplings of an $n \times n$ MIMO system of concern, the proposed MANNC structure in Figure 3-1 is further decomposed into n-parallel sub-controllers each named as a Sub-MANNC (S-MANNC) and illustrated in Figure 3-2. The neural network associated with the l^{th} S-MANNC with input r_l and output y_l ($l = 1, 2, \dots, n$) is designed to have two layers (hidden layer and output layer) and four neurons; each neuron is able to be connected with the neurons of the other S-MANNCs in order to account for the cross-coupling effects of the underlying MIMO system. In the hidden layer of the l^{th} S-MANNC, there are three neurons each being a P-type, an I-type, and a D-type, respectively. The P-neuron amplifies the difference between the desired output and the actual output, the I-neuron provides the necessary action to eliminate the steady-state error, and the D-neuron predicts the future behaviour of the error. In the output layer of the l^{th} S-MANNC, there is a P-type neuron which performs the summation of the PID functionalities of the hidden-layer neurons. This neuron accumulates the outputs of the hidden layer and forms the control command applied to the l^{th} output of the $n \times n$ MIMO system.

Due to the fact the MANNC will be used for controlling coupled multivariable systems, the output neuron of each S-MANNC has 'n' number of inputs to be able to produce each control command by considering all the desired outputs and actual outputs. The inputs of the P-type, I-type, and D-type neurons (net^1_{3l-2}, net^1_{3l-1}, and net^1_{3l}) and the outputs of these neurons (O^1_{3l-2}, O^1_{3l-1}, and O^1_{3l}) are

related together by the activation functions of the neurons represented in (4)-(6). There are three weights ($w_{1,3l-2}$, $w_{1,3l-1}$, and $w_{1,3l}$) in the output layer of each S-MANNC, relating to the P-type, I-type, and D-type neurons in the hidden layer.

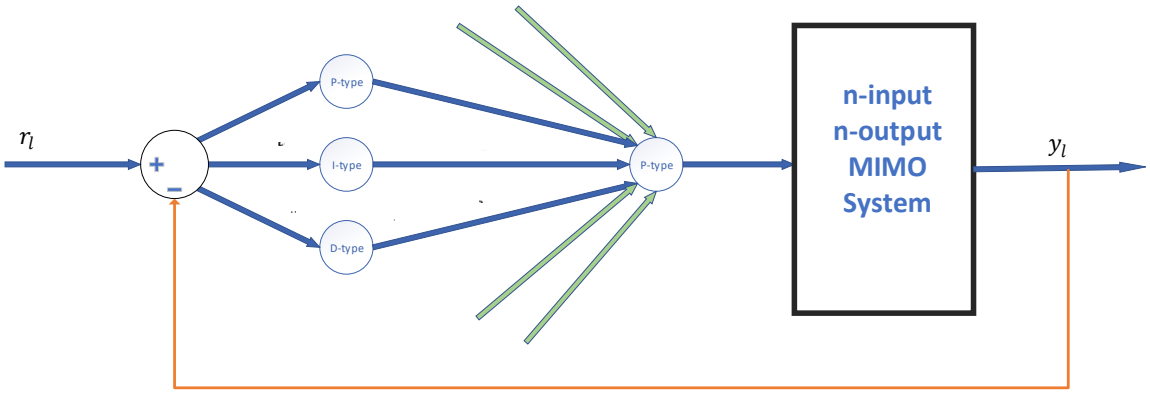


Figure 3-2. S-MANNC structure

3.2.3 Matrix representation

The matrix representation of a closed-loop square MIMO system using the proposed MANNC (Figure 3-1) and S-MANNC (Figure 3-2) is derived as follows.

Let O_l^2 and y_l be, respectively, the l^{th} input and the l^{th} output of the system where $1 \leq l \leq$ n, and let G_{ij} be the transfer function relating the j^{th}-component of the i^{th} output (y_i) to the j^{th} input (O_j^2) where $1 \leq i \leq$ n and $1 \leq j \leq$ n. The vectors and matrices associated with Figure 3-2 are named in Table 3-1 and are defined as:

$\boldsymbol{Y_{n\times1}}$	$\boldsymbol{R_{n\times1}}$	$\boldsymbol{O^2_{n\times1}}$	$\boldsymbol{G_{n\times n}}$
System's outputs	System's desired outputs	System's inputs	System's transfer matrix
$\boldsymbol{net^2_{n\times1}}$	$\boldsymbol{W_{n\times3n}}$	$\boldsymbol{O^1_{3n\times1}}$	$\boldsymbol{P_{3n\times3n}}$
Output layer's inputs	Neural Network Weights	Neurons' outputs	Activation functions
$\boldsymbol{net^1_{3n\times1}}$	$\boldsymbol{R^{(3)}_{3n\times1}}$	$\boldsymbol{Y^{(3)}_{3n\times1}}$	$\boldsymbol{I^{(3)}_{3n\times n}}$
Hidden layer's inputs	Triple desired outputs	Triple system's outputs	Triple Unit

Table 3-1 – Matrices defined for MANNC

$$Y = Y_{n\times 1} = [y_1 \quad y_2 \quad \cdots \quad y_l \quad \cdots \quad y_n]_{1\times n}{}^T \tag{7}$$

$$O^2 = O^2_{n\times 1} = \left[O^2_1 \quad O^2_2 \quad \cdots \quad O^2_l \quad \cdots \quad O^2_{n\,n}\right]_{1\times n}{}^T \tag{8}$$

$$G = G_{n\times n} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1l} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2l} & \cdots & G_{2n} \\ \vdots & \vdots & & & & \vdots \\ G_{l1} & G_{l2} & \cdots & G_{ll} & \cdots & G_{ln} \\ \vdots & \vdots & & \vdots & & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nl} & \cdots & G_{nn} \end{bmatrix}_{n\times n} \tag{9}$$

$$net^2 = net^2_{n\times 1} = [net^2_1 \quad net^2_2 \quad \cdots \quad net^2_l \quad \cdots \quad net^2_n]_{1\times n}{}^T \tag{10}$$

$$W = W_{n\times 3n} =$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,3l-2} & w_{1,3l-1} & w_{1,3l} & \cdots & w_{1,3n-2} & w_{1,3n-1} & w_{1,3n} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,3l-2} & w_{2,3l-1} & w_{2,3l} & \cdots & w_{2,3n-2} & w_{2,3n-1} & w_{2,3n} \\ \vdots & \vdots & & & & \vdots & \vdots & & & & \vdots \\ w_{l,1} & w_{l,2} & w_{l,3} & \cdots & w_{l,3l-2} & w_{l,3l-1} & w_{l,3l} & \cdots & w_{l,3n-2} & w_{l,3n-1} & w_{l,3n} \\ \vdots & \vdots & & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ w_{n,1} & w_{n,2} & w_{n,3} & \cdots & w_{n,3l-2} & w_{n,3l-1} & w_{n,3l} & \cdots & w_{n,3n-2} & w_{n,3n-1} & w_{n,3n} \end{bmatrix}_{n\times 3n} \tag{11}$$

$$O^1 = O^1_{3n\times 1} = [O^1_1 \quad O^1_2 \quad O^1_3 \quad \cdots \quad O^1_{3n}]_{1\times 3n}{}^T \tag{12}$$

$$P = P_{3n\times 3n} = \begin{bmatrix} 1 & 0 & 0 & & & & & \\ 0 & D^{-1} & 0 & \cdots & & \mathbf{0} & & \\ 0 & 0 & D & & & & & \\ & \vdots & & \ddots & & \vdots & & \\ & & & & 1 & 0 & 0 & \\ & \mathbf{0} & & \cdots & 0 & D^{-1} & 0 & \\ & & & & 0 & 0 & D & \end{bmatrix}_{3n\times 3n} \tag{13}$$

$$net^1 = net^1_{3n\times 1} = [net^1_1 \quad net^1_2 \quad net^1_3 \quad \cdots \quad net^1_{3n}]_{3n\times 1}{}^T \tag{14}$$

$$R^{(3)} = R^{(3)}_{3n\times 1} = [r_1 \quad r_1 \quad r_1 \quad r_2 \quad r_2 \quad r_2 \quad \cdots \quad y_n \quad y_n \quad y_n]_{1\times 3n}{}^T \tag{15}$$

$$Y^{(3)} = Y^{(3)}_{3n\times 1} = [y_1 \quad y_1 \quad y_1 \quad y_2 \quad y_2 \quad y_2 \quad \cdots \quad y_n \quad y_n \quad y_n]_{1\times 3n}{}^T \tag{16}$$

$$I^{(3)} = I^{(3)}_{3n\times n} = \begin{bmatrix} 1 & & \\ 1 & \cdots & 0 \\ 1 & & \\ \vdots & \ddots & \vdots \\ & & 1 \\ 0 & \cdots & 1 \\ & & 1 \end{bmatrix}_{3n\times n} \tag{17}$$

$$R = R_{n\times 1} = [r_1 \quad r_2 \quad r_3 \quad \cdots \quad r_n]_{1\times n}{}^T \tag{18}$$

Considering that the system at each calculation step can be identified as being linear or been linearized around an operation point, the relationship between the inputs and outputs of the system will be:

$$Y_{n\times 1} = G_{n\times n} O^2_{n\times 1} \tag{19}$$

Since net_l^2 is the l^{th} input of the P-type neuron in the output layer, one has:

$$O^2_{n\times 1} = net^2_{n\times 1} \tag{20}$$

where the relationship between the inputs and outputs in the output layer of the neural networks can be expressed as:

$$net^2_{n\times 1} = W^2_{n\times 3n} O^1_{3n\times 1} \tag{21}$$

As the hidden layer has clusters of P-type, I-type, and D-type neurons, the proportional, integral, and derivative operators (1, D^{-1}, and D) are considered respectively in the matrix form of the activation function. Thus, one has:

$$O^1_{3n\times 1} = P_{3n\times 3n} net^1_{3n\times 1} \tag{22}$$

where the inputs of the hidden layer are defined as the differences between the desired outputs and the actual outputs and are expressed as:

$$net^1_{3n\times 1} = R^3_{3n\times 1} - Y^3_{3n\times 1} \tag{23}$$

By having:

$$R^{(3)}_{3n\times 1} = I^{(3)}_{3n\times n} \times R_{n\times 1} \tag{24}$$

and:

$$Y^{(3)}_{3n\times 1} = I^{(3)}_{3n\times n} \times Y_{n\times 1} \tag{25}$$

and using (19)-(25), one obtains:

$$Y = GWP(R^{(3)} - Y^{(3)}) = GWP\left(I^{(3)}R - I^{(3)}Y\right) \tag{26}$$

Hence, the system's outputs can be derived as:

$$Y = GWPI^{(3)}R\left(I + GWPI^{(3)}\right)^{-1} \tag{27}$$

where: $\left|I + GWPI^{(3)}\right| \neq 0$.

Additionally, by having:

$$Y = GWPI^{(3)}(R - Y) \tag{28}$$

one has:

$$G^{-1}Y = WPI^{(3)}(R - Y) \tag{29}$$

where: $|G| \neq 0$.

Since $PI^{(3)}(R - Y)$ is a non-square matrix and P is symmetrical, from:

$$G^{-1}Y\left\{PI^{(3)}(R - Y)\right\}^T = WPI^{(3)}(R - Y)\left\{PI^{(3)}(R - Y)\right\}^T \tag{30}$$

one derives:

$$W = G^{-1}Y(R - Y)^T I^{(3)^T} P\left[PI^{(3)}(R - Y)(R - Y)^T I^{(3)^T} P\right]^{-1} \tag{31}$$

where W is expressed as a function of the desired outputs and the actual outputs, and: $\left|PI^{(3)}(R - Y)(R - Y)^T I^{(3)^T} P\right| \neq 0$ and $|W| \neq 0$.

The system can then be identified as:

$$G = Y(R - Y)^T I^{(3)^T} P\left[PI^{(3)}(R - Y)(R - Y)^T I^{(3)^T} P\right]^{-1} W^{-1} \tag{32}$$

where: $|W| \neq 0$.

It should be pointed out that although Equations (19)-(32) are derived under the assumption that the system is linear or can be linearized around an operating point, they can potentially be used for nonlinear systems where the nonlinearities of the systems can be approximated by piece-wise linear systems whose time-varying nature can account for the nonlinearities of the systems satisfactorily. It should also be pointed out that if any of the non-singularity conditions for matrices $(I + GWPI^{(3)})$, (G), $(PI^{(3)}(R - Y)(R - Y)^T I^{(3)^T} P)$, and (W) could not be met, the selected weights (matrix W) would not be acceptable and would be re-updated until all these matrices become non-singular.

3.3 Learning algorithm

To achieve a precise control effect for a square MIMO system, the neural network weights of the MANNC are adjusted using the principle of the multi-step error back-propagation algorithm described in [47]. The choice of this

algorithm is based on the considerations that the system is treated as a "black-box" and the controller will be designed based on a model-free approach. In this study, instead of using merely the current gradient of the system error as the literature does for SISO systems [25, 40, 41], an accumulated gradient of the system error for the last m samples is proposed to be used to achieve a more precise control performance. The proposed method minimises the sum of the square accumulated gradient of the error for each system output in each learning step, where the error is taken as the difference between the desired output $r_l(k)$ (i.e., the system set-point) and the actual output $y_l(k)$. Euclidean Norm E is defined for calculating the quadratic cost function of the system for the system error. Power of two in this expression makes the error of each output positive so that larger errors will be more significant than the smaller errors. The cost function for the l^{th} S-MANNC in Figure 3-2 is then defined as:

$$\boldsymbol{E_l(h) = \frac{1}{2}(\sum_{k=1}^{m}(r_l[k] - y_l(h)[k]))^2} \tag{33}$$

where $E_l(h)$ is the error of the l^{th} output in the h^{th} step number of the learning algorithm and m is the required number of discrete samples in the system output and the set-point. By increasing m, the system output will be compared more accurately with the set-point. However, a large value of m may decrease the adjustment speed of the controller and become undesirable when the speed of the control system (often a critical requirement for a real-time industrial system) is of concern. Therefore, a reasonable value of m must be used to make a necessary trade-off between the desired accuracy and the required speed of the control system. The total cost function of the system (J) which is the sum of n errors by considering all outputs is written as:

$$J(h) = \sum_{l=1}^{n} E_l(h) = \frac{1}{2}\sum_{l=1}^{n}(\sum_{k=1}^{m}(r_l[k] - y_l(h)[k]))^2 \tag{34}$$

Using the accumulated gradient of the system error, a learning algorithm must be designed to minimise the defined cost function and to bring the system outputs as close as possible to the desired outputs. According to the principle of the error back-propagation learning algorithm, the output layer's weights must be adjusted so that in each step they move slightly in the opposite direction of the gradient of the cost function. This is to ensure that the cost function will be

decreasing step by step. The weights of the output layer will therefore be adjusted based on the following learning rule:

$$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l \frac{\partial J(h)}{\partial w_{l,x}} \tag{35}$$

where $1 \leq x \leq 3l$ and $1 \leq l \leq$ n; h is the step number of the learning algorithm; $w_{l,x}(h)$ and $w_{l,x}(h+1)$ are the weights of the output layer in the current and the following steps, respectively; λ_l is the learning rate which decides how fast the cost is changing and, in particular, determines the weight adjustment speed. The gradient of the error subject to each weight is required to be calculated. Using partial derivatives, one has:

$$\frac{\partial J}{\partial w_{l,x}} = \frac{\partial J}{\partial E_l}\frac{\partial E_l}{\partial y_l}\frac{\partial y_l}{\partial O_l^2}\frac{\partial O_l^2}{\partial net_l^2}\frac{\partial net_l^2}{\partial w_{l,x}} \tag{36}$$

where:

$$\frac{\partial J}{\partial E_l} = 1 \tag{37}$$

$$\frac{\partial E_l}{\partial y_l} = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \tag{38}$$

$$\frac{\partial y}{\partial O_l^2} \cong \frac{\sum_{k=1}^{m}(y(h)[k]) - \sum_{k=1}^{m}(y(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{39}$$

$$\frac{\partial O_l^2}{\partial n_l^2} = 1 \tag{40}$$

Because the output neuron is a 'P-type' neuron, one writes:

$$\frac{\partial net_l^2}{\partial w_{l,x}} = \sum_{k=1}^{m}(O_x^1(h)[k]) \tag{41}$$

Substituting (37)-(41) into (36), one obtains:

$$\frac{\partial J}{\partial w_{l,x}} \cong 1 \times \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \frac{\sum_{k=1}^{m}(y(h)[k]) - \sum_{k=1}^{m}(y(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} O_x^1(h)[k] \tag{42}$$

Defining $\gamma_l(k)$ as:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \frac{\sum_{k=1}^{m}(y(h)[k]) - \sum_{k=1}^{m}(y(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{43}$$

the output layer weight adjustment rule can thus be derived as:

$$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l[\gamma_l(h) \sum_{k=1}^{m}(O_x^1(h)[k])] \tag{44}$$

The resulting weight adjustment algorithm is summarized in Table 3-2:

$1 \leq l \leq$ n $1 \leq x \leq 3$n	$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l[\gamma_l(h) \sum_{k=1}^{m} (O_x^1(h)[k])]$ $\gamma_l(h) = \sum_{k=1}^{m} (y_l(h)[k] - r_l(h)[k]) \frac{\sum_{k=1}^{m}(y(h)[k]) - \sum_{k=1}^{m}(y(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])}$

Table 3-2. Weights adjustment learning algorithm

Using the described learning method, all weights are simultaneously tuned, and their values at the current step, together with all the current inputs and all the current outputs, are used to create the outputs for the weights' training at the next step. This online dynamic learning feature of the proposed method (i.e., applying the accumulated gradient and the accumulated errors) makes it more suitable for MIMO systems than its existing counterparts that are mainly applicable for SISO systems [39, 40, 48]. The neural network structure presented for the MANNC is designed for a general n-input n-output system with cross-couplings among all its inputs and outputs. This structure can be simplified for systems with less cross-couplings to reduce the number of the associated weights and, consequently, to increase the learning speed.

3.4 Stability Analysis

It is well known that when a new control method is proposed, it is necessary to investigate the stability condition of the resultant closed-loop system in order to ensure the achievement of the desired control outcomes. For an unconstrained control system, the system stability can be defined using Bounded-Input Bounded-Output (BIBO) stability criteria. While the eigenvalue analysis concept based on the BIBO stability criteria can be used to investigate the stability condition of a linear system, it can't be applied to nonlinear systems. Instead, the Lyapunov stability analysis concept becomes a useful tool for nonlinear systems. As the proposed model-free MANNC method will inherently result in a nonlinear closed-loop control system, its closed-loop stability investigation will need to be carried out using the Lyapunov stability analysis concept.

Although, unlike a linear system, the stability of a nonlinear system may not need to be global as the system can have multiple equilibrium points and limit

cycles, global asymptotic stability is sought in this study for the proposed MANNC to ensure its satisfactory closed-loop performance over a wide range of operating points. This will make the controller more suitable for use in industrial applications. According to the Lyapunov global asymptotic stability theorem, for a defined function $V(x)$, if:

(i) $V(0) = 0$

(ii) For all $x \neq 0$, $V(x) > 0$ (i.e., V is positive definite)

(iii) For all $x \neq 0$, $\Delta V(x) < 0$

then every trajectory of $X(k+1) = X(k) + f(X(k))$ will converge to zero as $k \to \infty$ and the system will be globally asymptotically stable.

Consider a model-free MANNC control system whose Lyapunov function for each of its outputs is defined as:

$$V_l(h) = (E_l(h))^2 \tag{45}$$

where $E_l(h)$ is the cost function related to the h^{th} step of the learning algorithm. One can write:

$$\Delta V_l(h) = \left(E_l(h) + \Delta E_l(h)\right)^2 - (E_l(h))^2 = 2E_l(h)\Delta E_l(h) + \left(\Delta E_l(h)\right)^2 \tag{46}$$

and:

$$\Delta E_l(h) \cong \Delta w_{l,x}(h) \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}} \tag{47}$$

From (46), one has:

$$\Delta w_{l,x}(h) = -\frac{\lambda_l}{m} \sum_{k=1}^{m} \frac{\partial J}{\partial w_{l,x}} \tag{48}$$

where:

$$\frac{\partial J}{\partial w_{l,x}} = \frac{\partial J}{\partial E_l(h)} \frac{\partial E_l(h)}{\partial w_{l,x}} = \frac{\partial E_l(h)}{\partial w_{l,x}} \tag{49}$$

Substitute (49) into (47), respectively, one derives:

$$\Delta w_{l,x}(h) = -\frac{\lambda_l}{m} \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}} \tag{50}$$

and:

$$\Delta E_l(h) \cong -\frac{\lambda_l}{m}\left(\sum_{k=1}^{m}\frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^2 \tag{51}$$

Hence, (46) can be expressed as:

$$\Delta V(h) \cong -\frac{2\lambda_l E_l(h)}{m}\left(\sum_{k=1}^{m}\frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^2 + \frac{{\lambda_l}^2}{m^2}\left(\sum_{k=1}^{m}\frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^4 \tag{52}$$

Define:

$$H_l(h) = \sum_{k=1}^{m}\frac{\partial E_l(h)[k]}{\partial w_{l,x}} \cong \sum_{k=1}^{m}\frac{\Delta E_l(h)[k]}{\Delta w_{l,x}(h)} \tag{53}$$

The condition for $\Delta V(h) < 0$ will yield the following constraint on the selection of the learning rate λ_l defined in (44):

$$0 < \lambda_l < \frac{2m\ _l(h)}{H_l(h)^2} \tag{54}$$

If the above constraint is satisfied at each training step, the system will be globally asymptotically stable during the entire training process. This indicates that (54) must be checked in a real-time and simultaneous fashion at each training step during the operation of the weight adjustment algorithm. The flowchart of the whole real-time weight training process, including both the learning algorithm (Table 3-2) and the stability criteria check (54), is illustrated in Figure 3-3.

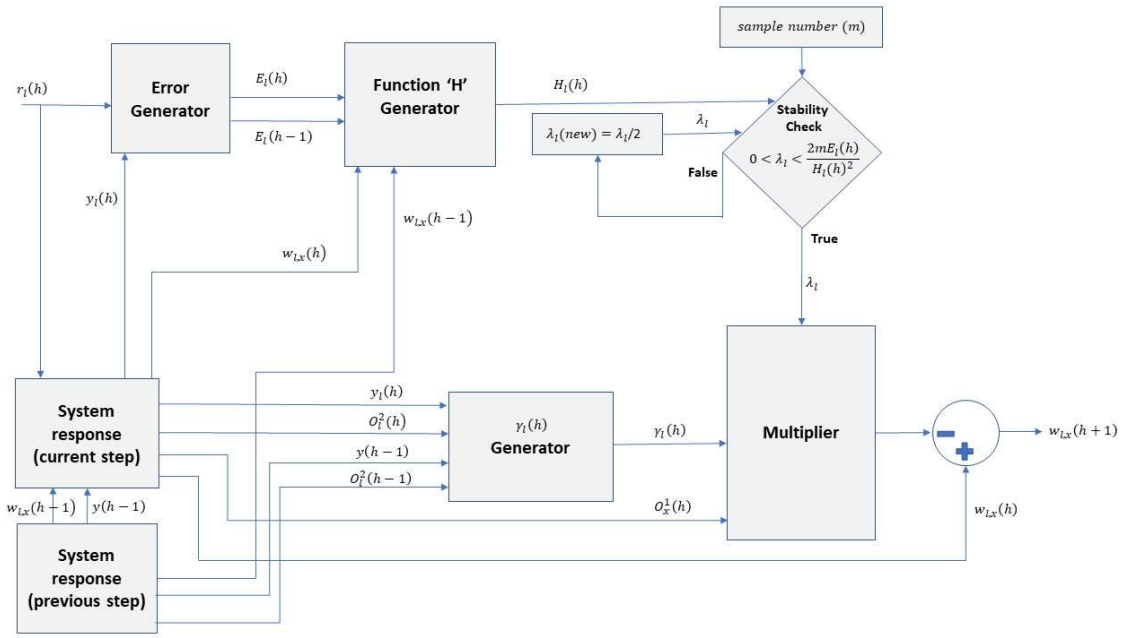


Figure 3-3. Flowchart of real-time simultaneous stability criteria check and weight adjustment algorithm for MANNC

3.5 Specifying MANNC to control SISO systems

The proposed MANNC can be used in a SISO environment where the number of inputs and outputs of the controller is chosen as one, i.e., $n = 1$. The resultant SISO closed-loop system is shown in Figure 3-4.

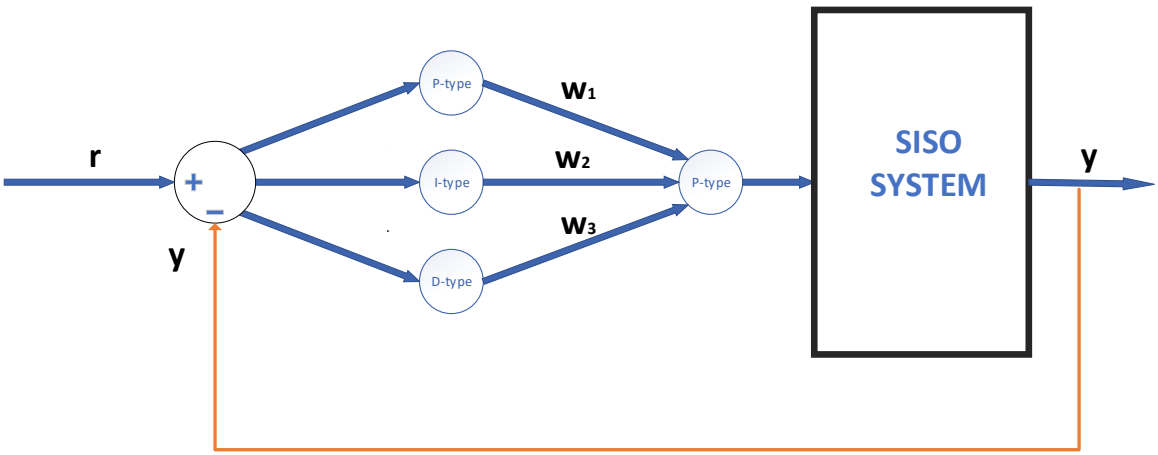


Figure 3-4. Applying MANNC to SISO system

Considering Figure 3-4 and assuming a linear and time-invariant estimation for the system's SISO transfer function, one has:

$$Y = G(s)O^2 = G(s)(w_1 O_1^1(s) + w_2 D^{-1} O_2^1(s) + w_3 D O_3^1(s)) \tag{55}$$

where:

$$O_1^1(s) = O_2^1(s) = O_3^1(s) = R(s) - Y(s) \tag{56}$$

Then,

$$Y(s) = G(s)(w_1 + w_2 s^{-1} + w_3 s)(R(s) - Y(s)) \tag{57}$$

and:

$$\frac{Y(s)}{R(s)} = \frac{G(s)(w_1 + w_2 s^{-1} + w_3 s)}{1 + G(s)(w_1 + w_2 s^{-1} + w_3 s)} \tag{58}$$

If w_1, w_2, and w_3 are taken as the coefficients of a classical proportional, integral, and derivative controller, i.e., K_P, K_I, and K_D, respectively, the resultant control system will perform similar to an auto-tune SISO PID control system whose closed-loop transfer function is expressed as:

$$T(s) = \frac{G(s)(K_P + K_I s^{-1} + K_D s)}{1 + G(s)(K_P + K_I s^{-1} + K_D s)} \tag{59}$$

The weight learning algorithm illustrated in Table 3-2 can be customized for SISO system applications and described in Table 3-3 where λ is the learning rate.

$$K_P(h+1) = K_P(h) - \lambda\gamma(h)\sum_{k=1}^{m}(O_1^1(h)[k])]$$

$$K_I(h+1) = K_I(h) - \lambda\gamma(h)\sum_{k=1}^{m}(O_2^1(h)[k])]$$

$$K_D(h+1) = K_D(h) - \lambda\gamma(h)\sum_{k=1}^{m}(O_3^1(h)[k])]$$

$$\gamma(h) = \sum_{k=1}^{m}(y(h)[k] - r(h)[k])\frac{\sum_{k=1}^{m}(y(h)[k]) - \sum_{k=1}^{m}(y(h-1)[k])}{\sum_{k=1}^{m}(O^2(h)[k]) - \sum_{k=1}^{m}(O^2(h-1)[k])}$$

Table 3-3. Specified weight learning algorithm for auto-tune classical PID controller

Following the Lyapunov stability analysis described in (45)-(52) and redefining (53) as:

$$H(h) = \sum_{k=1}^{m}\frac{\partial E(h)[k]}{\partial w_{l,x}} \cong \sum_{k=1}^{m}\frac{\Delta E(h)[k]}{\Delta w_x(h)} \tag{60}$$

the condition for $\Delta V(h) < 0$ yields:

$$0 < \lambda < \frac{2mE(h)}{H(h)^2} \tag{61}$$

3.6 Simulation results

Simulation studies using Matlab are carried out to evaluate the performances of the proposed MANNC in tracking set-points, reducing unwanted overshoots or undershoots, and securing the global stability of a closed-loop system during the system's entire control process. The structure of the MANNC proposed in Section 3.2, the dynamic neural network algorithm developed in Section 3.3, and the stability criteria checking condition discussed in Section 3.4 are used in three simulation cases, each presenting a type of square MIMO systems. They are: a time-invariant nonlinear system, a time-variant nonlinear system, and a hybrid system.

3.6.1 Case 1 – Application of MANNC on a time-invariant nonlinear square MIMO system

In this case, a drum-boiler plant (Figure 3-5) which is a generic nonlinear time-invariant coupled two-input two-output system with heat flow rate and mass flow rate as inputs and pressure and level as outputs, is chosen. The nonlinearities of the plant cause the system dynamic characteristics to vary with operating conditions. In addition, cross-couplings and parameter variations of the plant makes it a challenging case to control. For years, constructing a nonlinear controller directly from the original nonlinear model of the drum-boiler has been a general approach to use in order to improve the system performance in compensating the plant's nonlinearities. In this study, however, the proposed model-free MANNC method is used in which the nonlinear model of the drum-boiler is not required by the controller design process and only the inputs and outputs of the plant are used for the construction of the controller. The performances of the MANNC in improving the system set-point tracking time and reducing undesirable overshoots are compared with those of the best and foremost existing neural network method reported in the literature.

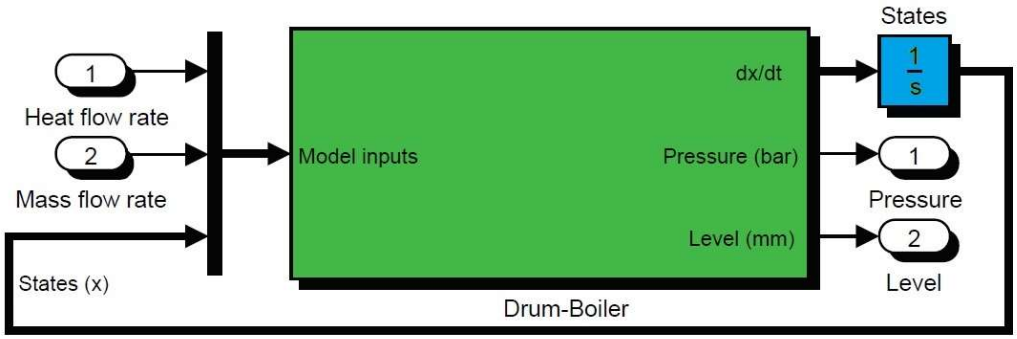


Figure 3-5. Two-input two-output drum-boiler system

The nonlinear state equations describing the relationships between the inputs and outputs of the drum-boiler system of Figure 3-5 are written as:

$$\dot{x_1}(t) = -x_1(t) + u_1(t) \tag{62}$$

$$y_1(t) = x_1(t) + 2x_1^3(t) \tag{63}$$

$$x_2(t) = y_1(t) + u_2(t) \tag{64}$$

$$\ddot{y_2}(t) + 2\dot{y_2}(t) + y_2(t) = \dot{x_2}(t) + 2x_2(t) \tag{65}$$

where "dot" denotes time derivative, $u_1(t)$ and $u_2(t)$ are respectively heat flow rate and mass flow rate as the system inputs, $y_1(t)$ and $y_2(t)$ are respectively pressure and level as the system outputs, and $x_1(t)$ and $x_2(t)$ are the state variables. The simulation block diagram of the given nonlinear system is presented in Figure 3-6 where box 'Fcn1' produces the highly nonlinear relationship between the inputs and outputs of the system. Due to the fact that Input 1 affects both Output 1 and Output 2, the system is also a cross-coupled MIMO system and, thus, will not be able to be controlled by multiple ANNC controllers introduced in [18], nor by any other SISO or non-coupled MIMO counterparts.

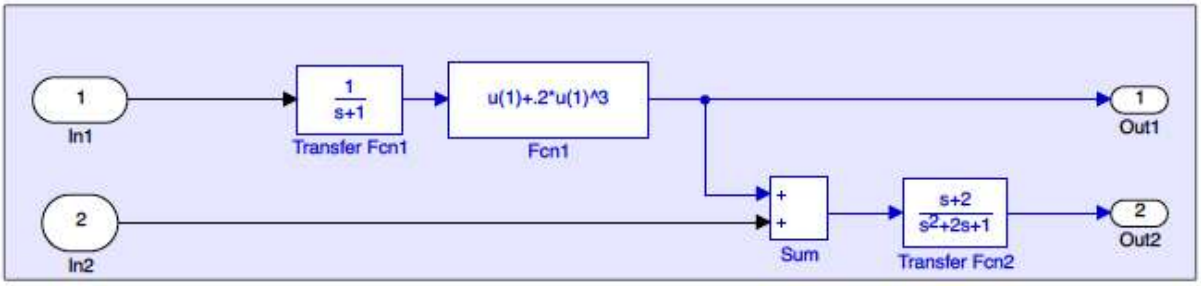


Figure 3-6. Two-input two-output nonlinear drum-boiler model

Figure 3-7 demonstrates the implementation of the proposed MANNC in the two-input two-output system. In this neural network, 12 weights in the hidden layer must be trained.

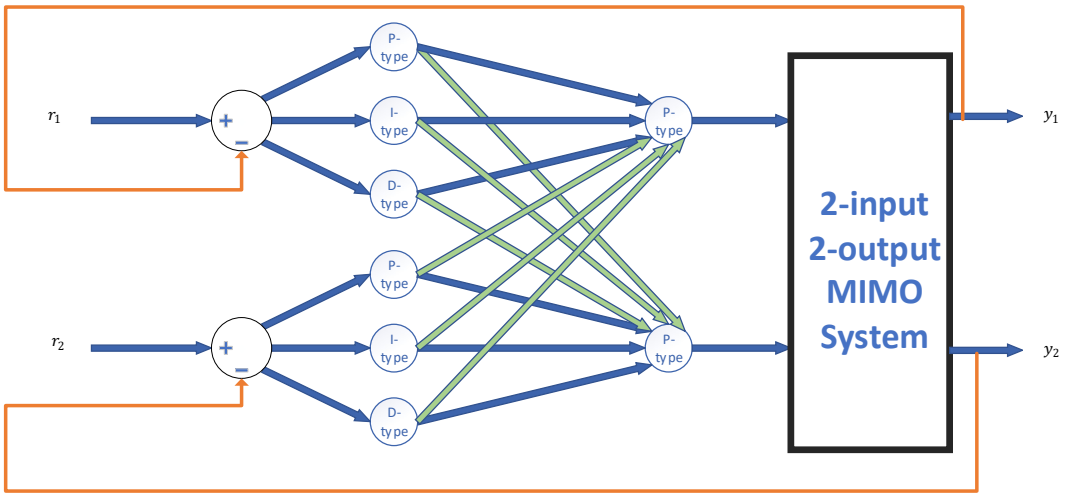


Figure 3-7. Two-input two-output system controlled by MANNC

The following desired outputs are selected for the drum-boiler system.

- $r_1(t) = 5.0u(t-1)$ where $u(t)$ is the standard unit step function.
- $r_2(t) = 0.2r(t-1)$ where $r(t)$ is the standard unit ramp function.

These desired outputs represent a scenario in which the pressure will tend to reach 5 bars and the level will increase following a smooth ramp function. The ramp function is chosen to be the set-point for Output 2 in order to make the MANNC control more challenging. The objective is to find suitable values for the weights of the output layer so as to force the outputs, y_1 and y_2, to follow the desired set-points, r_1 and r_2, respectively.

To apply the MANNC to the drum-boiler system, all initial weights are set to 1, the learning rates are set to 0.1 ($\lambda_1 = \lambda_2 = 0.1$), and the number of samples is set to 40 ($m = 40$). After repeating, simultaneously, the online learning algorithm (Table 3-2) and the Lyapunov stability criteria check (Figure 3-3) for 20 times, the final weight values of the output layer of the MANNC neural network are obtained and reported in Table 3-4.

$w_{l,x}$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$
$l = 1$	10.23	0.33	6.93	-2.23	-2.13	-2.10
$l = 2$	-1.35	3.59	1.36	3.22	3.24	1.94

Table 3-4. Final weight values of the output layer

During the running of the weight training algorithm, the control system remains stable all the time as λ_1 and λ_2 are kept unchanged. If the control system becomes unstable at a training step, the learning rates will be reduced and the learning algorithm will be continued with a lower training speed that is dictated by the new learning rates. Applying the final adjusted weights (Table 3-4) to the control system, Figure 3-8 shows that Output 1 tracks the desired step output properly with a zero-overshoot. This zero-overshoot effect is most desirable for many industrial control systems, as unwanted overshoots during set-point changes can bring devastated results to the industrial plants. For example, when filling a hazardous liquid tank in a water plant, overshoots can result in overflows if the set-point is close to the tank's height [49].

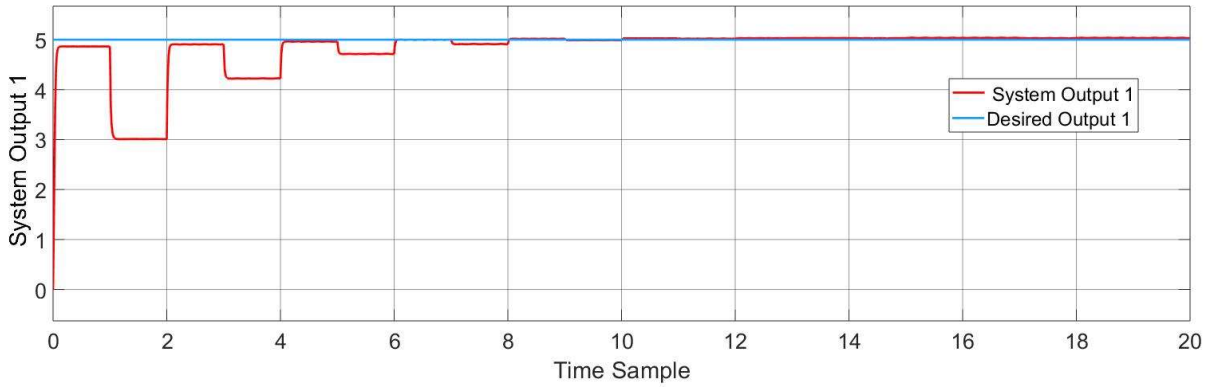


Figure 3-8. Case 1 - Output one and desired output one (MANNC)

Figure 3-9 demonstrates that Output 2 tracks the desired ramp output properly, and in less than 9 samples the output can reach the desired level with a less than 5% error.

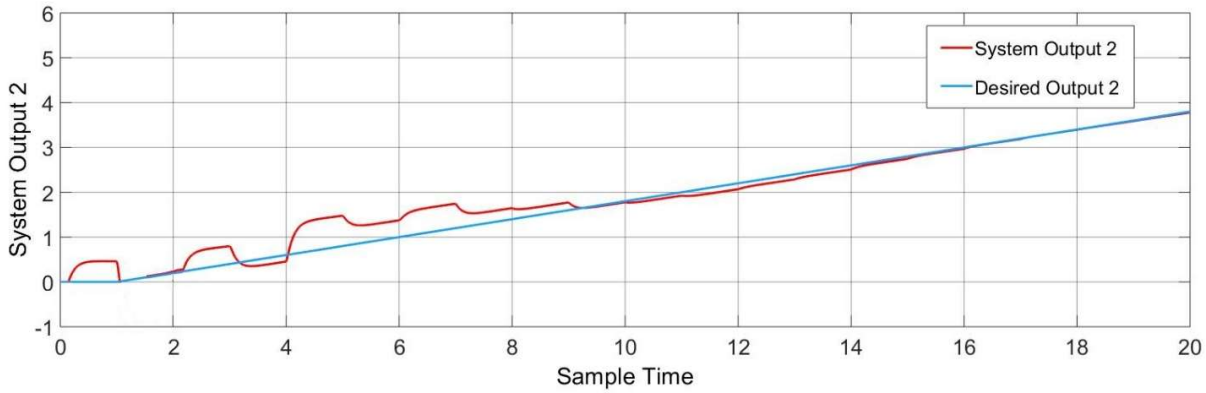


Figure 3-9. Case 1 - Output two and desired output two (MANNC)

To demonstrate the anticipated highly improved performance of the MANNC over its existing adaptive counterparts, a most recent neural network controller (named PIDNN) introduced in [39] is chosen for the drum-boiler system. The PIDNN uses a learning algorithm that is based on two consecutive time-samples. The weights of the PIDNN are set after running the learning algorithm for 20 times – the same condition upon which the weights of the MANNC are obtained. The comparison results are given in Figure 3-10 and Figure 3-11. It is seen that using the PIDNN method, Output 1 presents a 22% undesirable overshoot and Output 2 exhibits a slower set-point tracking with larger fluctuations.

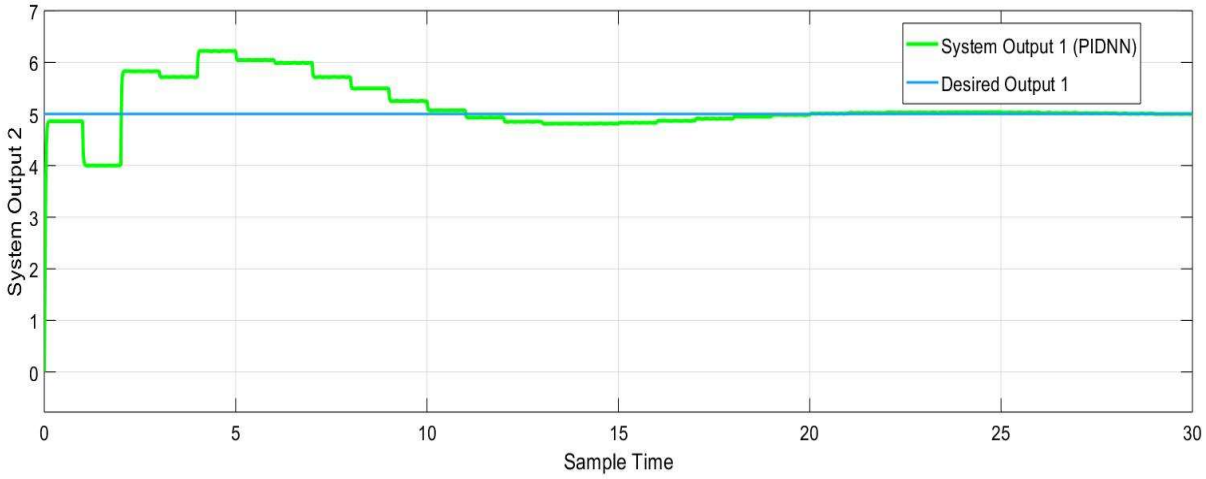


Figure 3-10. Case 1 - Output one and set-point one (PIDNN)

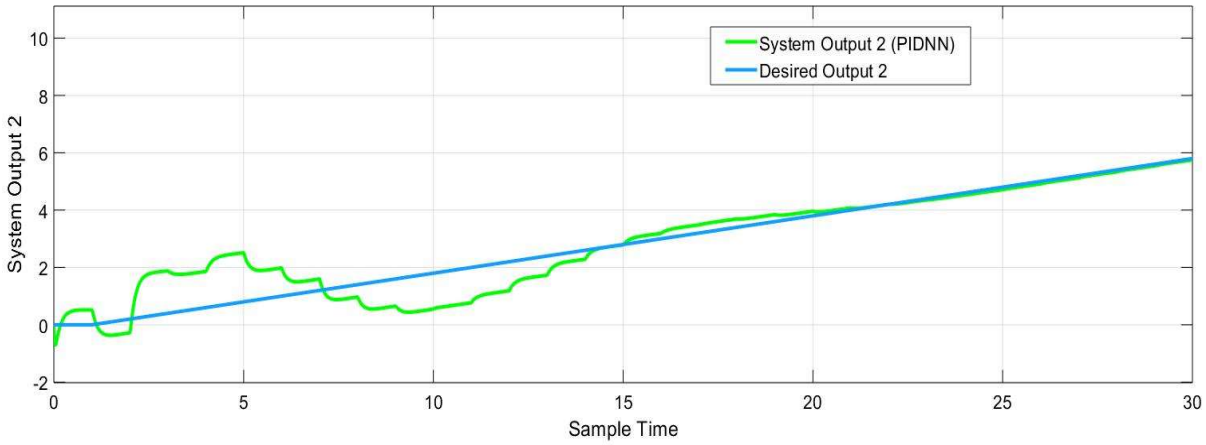


Figure 3-11. Case 1 - Output two and set-point two (PIDNN)

The comparison results between the MANNC and the PIDNN control effects with an equal number of trainings are given in Table 3-5. It is evident that, as compare to the PIDNN control system, the MANNC control system can significantly reduce the required training time (by 59%), achieve a zero-overshoot (100% reduction), and reach the 5% and 2% error bands in much shorter times (47% and 50% faster for Output 1, and 47% and 36% faster for Output 2). These results demonstrate a superior performance of the MANNC over its existing counterpart. In particular, Output 1 under MANNC control exhibits a deadbeat-like response with minimum rise-time, minimum settling-time, no overshoot, and no steady-state error, comparable with an optimal closed-loop response. This remarkable result is attributed to the MANNC strategy that uses the accumulated error and the response of the system in consecutive learning steps rather than in consecutive sample times.

Controller	Number of trainings	Time of training	Output 1 Overshoot	Output 1 maximum error less than 5%	Output 1 maximum error less than 2%	Output 2 maximum error less than 5%	Output 2 maximum error less than 2%
MANNC	20	1.48 s	0%	8 s.t.*	10 s.t.	9 s.t.	14 s.t.
PIDNN	20	3.67 s	22%	15 s.t.	20 s.t.	17 s.t.	22 s.t.
*s.t. stands for sample times.							

Table 3-5. Case 1 - MANNC vs. PIDNN

Figure 3-12 and Figure 3-13 represent, respectively, the accumulated error versus the iteration number of the MANNC weight learning algorithm for Output 1 and Output 2. The errors shown in these figures are the differences between the actual outputs and the desired outputs. It is observed that, with the MANNC, the magnitudes of the errors generally decrease as the number of iterations in the weight learning algorithm increases. This is an evidence for the convergence of the proposed learning algorithm in Section 3.3, and is viewed as a significant result that demonstrates the suitable performance of the proposed MANNC for the considered coupled two-input two-output nonlinear system. As shown in these figures, with 35 trainings, the errors are reasonably low and the values of the weights can be locked in at this point. By continuing the weight adjustment algorithm up to 50 times, the steady-state errors for both outputs become nearly zero. In general, choosing the optimal training number in this method depends on the particular application where the controller is employed, and implies a trade-off between the control speed and the control performance of the closed-loop system. It is possible to pre-define a desired accumulated error, so that when the actual error reaches to that value the training process stops and the weights become locked until the next change in the system happens (e.g. a change in the model of the system and/or a change in any of the set-points).

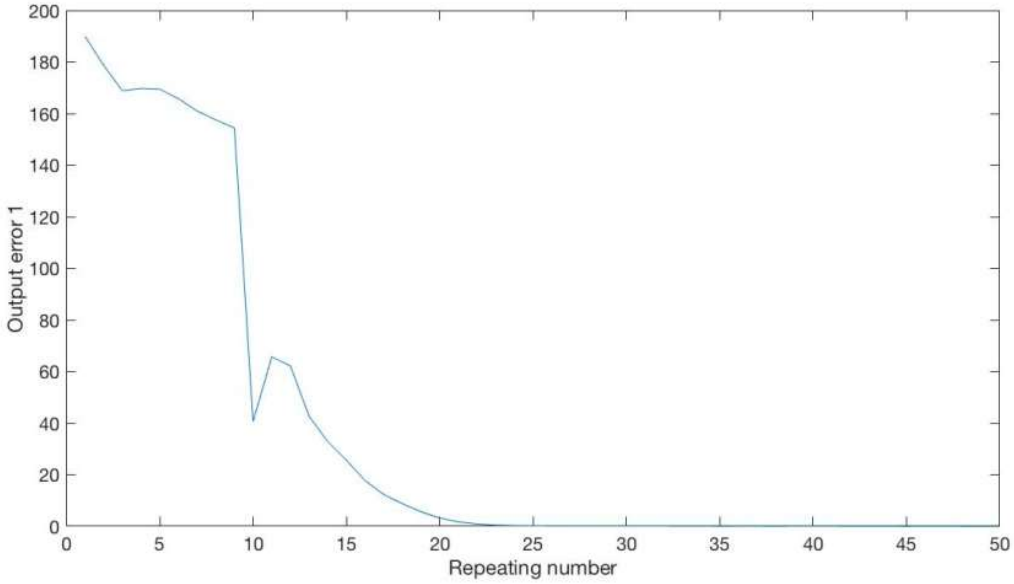


Figure 3-12. Case 1 - Error value for Output one versus the repeating number

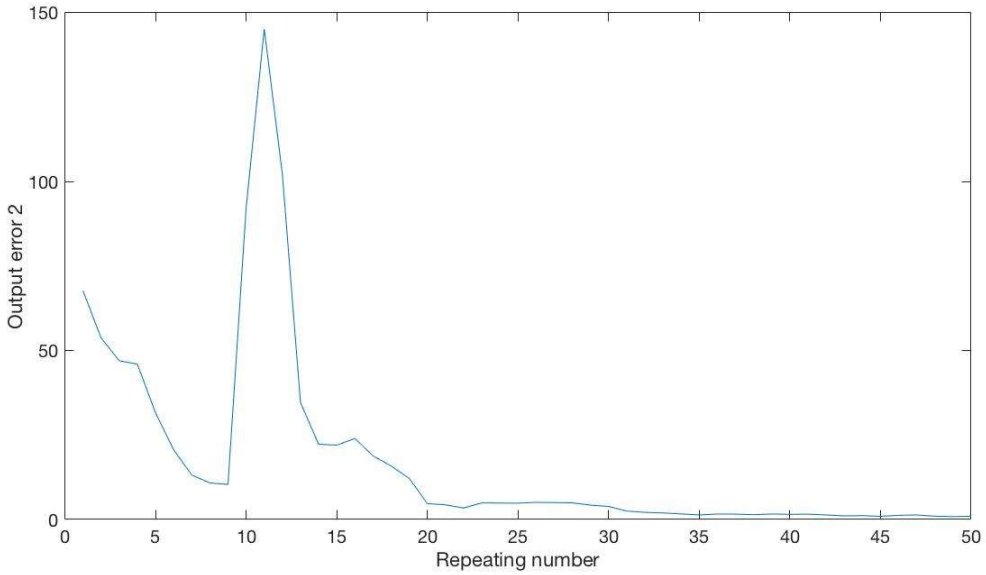


Figure 3-13. Case 1 - Error value for Output two versus the repeating number

3.6.2 Case 2 – Application of MANNC on a time-variant nonlinear MIMO system

In this case, a discrete-time highly-nonlinear time-variant two-input two-output coupled system is chosen to test the suitability and performance of the MANNC for time-variant MIMO systems. The chosen system is expressed as:

$$y_1(k+1) = \frac{1}{{y_1}^2(k)+1}(0.8y_1(k) + v_1(k-2) + 0.2v_2(k-3)) \quad (66)$$

$$y_2(k+1) = \frac{1}{{y_2}^2(k)+1}(0.9y_2(k) + 0.3v_1(k-3) + v_2(k-2)) \quad (67)$$

where $y_1(k)$ and $y_2(k)$ are the system outputs, and $v_1(k)$ and $v_2(k)$ are the system inputs. The desire outputs are selected as $r_1(k) = 0.6$ and $r_2(k) = 0$.

By applying, simultaneously, the online learning algorithm (Table 3-2) and the stability criteria check (Figure 3-3) for 50 times, the final weight values of the output layer of the MANNC neural network are obtained and reported in Table 3-6.

$w_{l,x}$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$
$l = 1$	3.34	2.43	4.73	-5.12	-8.13	-2.19
$l = 2$	-11.30	-4.44	-7.74	6.32	3.55	3.82

Table 3-6. Final weight values of the output layer

The simulation results of the final adjusted weights of the MANNC are shown in Figure 3-14 and Figure 3-15. In order to compare the MANNC results with those of a properly-selected existing counterpart, the PIDNN introduced in [39] is applied to the same system. The system outputs under the PIDNN control are shown in Figure 3-16 and Figure 3-17.

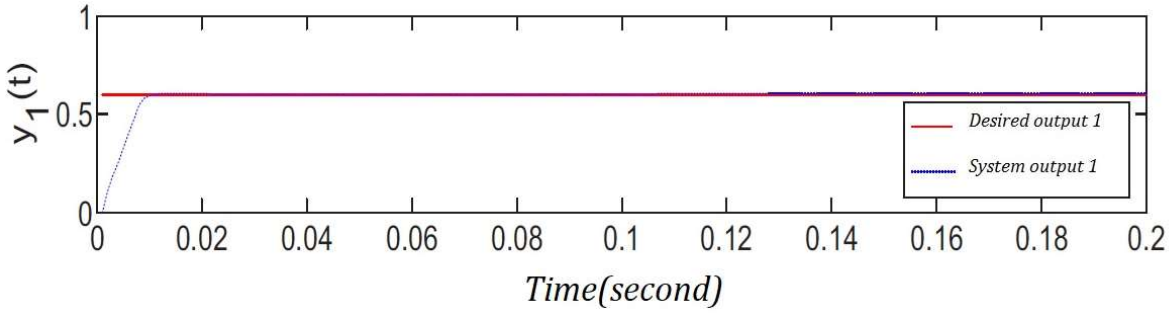


Figure 3-14. Case 2 - Output one and desired output one by MANNC

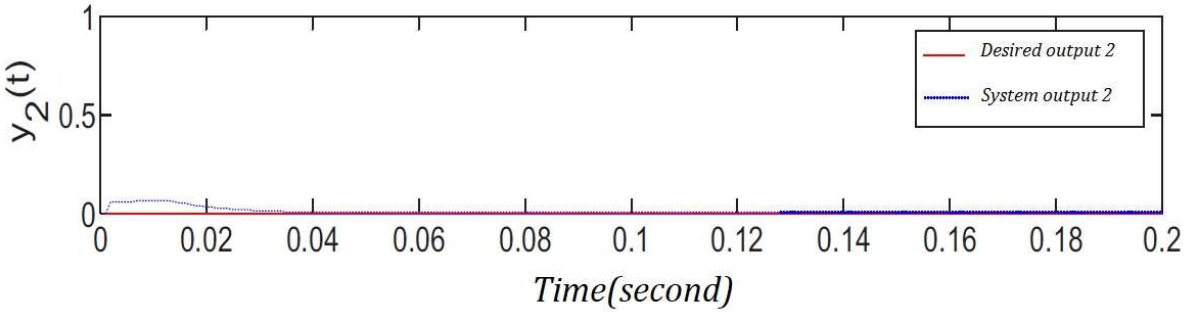


Figure 3-15. Case 2 - Output two and desired output two by MANNC

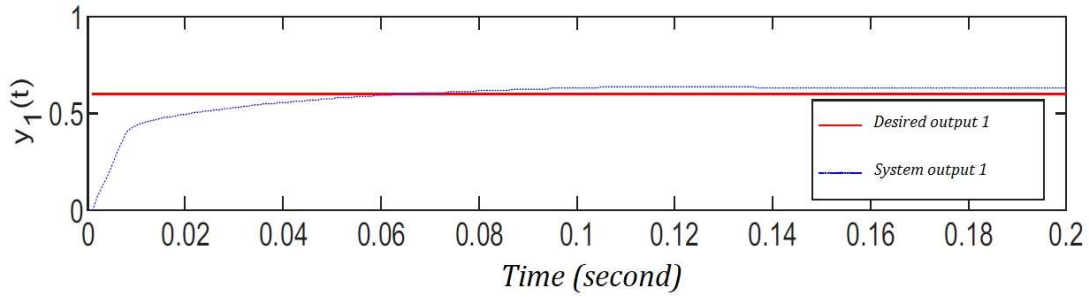


Figure 3-16. Case 2 - Output one and desired output one by PIDNN

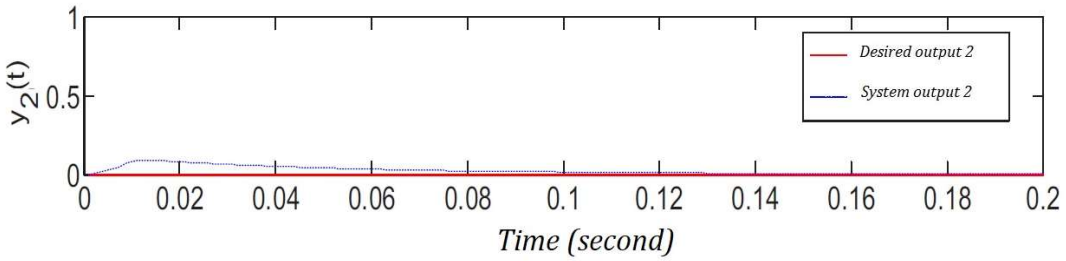


Figure 3-17. Case 2 - Output two and desired output two by PIDNN

It is demonstrated that both outputs of the MANNC control system track the desired outputs faster than those of the PIDNN control system. In addition, a zero-overshoot in Output 1 using the MANNC is achieved, which is a significant result as this is not achievable by using the PIDNN. Decreasing the accumulated error by increasing the number of iterations, as shown in Figure 3-18, demonstrates the convergence of the proposed learning algorithm for the MANNC evidently. Table 3-7 presents the performance comparison between the MANNC controller and the PIDNN controller with equal number of trainings. As can be seen, the MANNC control results outperform the PIDNN control results in all aspects.

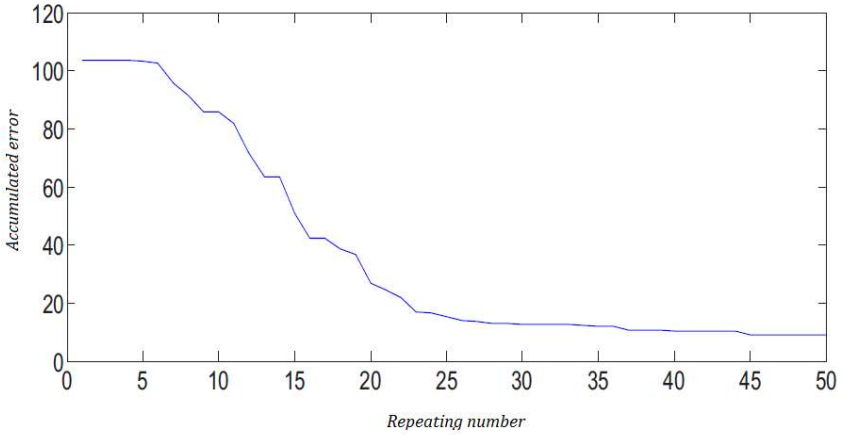


Figure 3-18. Case 2 - Accumulated error vs repeating number for MANNC

Controller	Number of trainings	Time of training	Output 1 Overshoot	Output 1 maximum error less than 5%	Output 2 maximum error less than 5%
MANNC	50	2.39 s	0%	0.01 s	0.05 s
PIDNN	50	3.99 s	22%	0.03 s	0.08 s

Table 3-7. Case 2 - MANNC vs. PIDNN

3.6.3 Case 3 – Application of MANNC on a hybrid system

Since the MANNC has been designed as a universal controller for black-box square MIMO systems, in this case, the performance of this controller is tested on a hybrid system. By definition, a hybrid system is a dynamic system that switches between continuous states and, thus, involves both continuous and discrete behaviours. Due to sudden changes in system dynamics at the time of switching between two states, conventional control methods are usually unsuccessful for hybrid systems.

In this study, a two-tank plant shown in Figure 3-19 is selected to test the control performance of the MANNC.

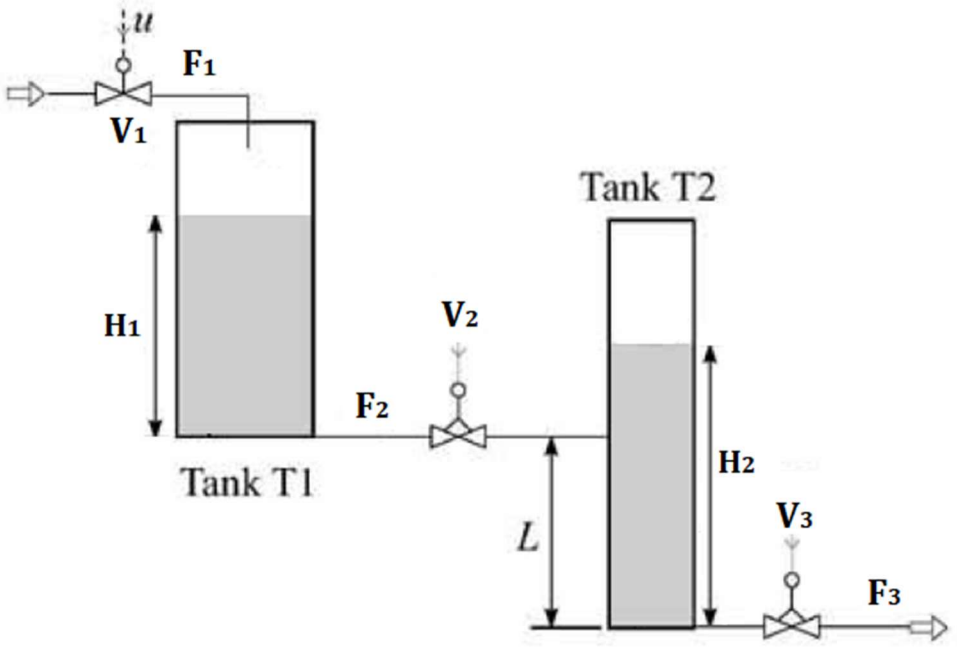


Figure 3-19. Two-tank hybrid system

The plant consists of two tanks, where tank T1 is filled by flow F_1 through a fully open valve V_1. The liquid is transferred from tank T1 into tank T2 via a connecting pipe. Valve V_2 is an on-off valve which is either fully open or fully closed to adjust flow F_2 discretely. Similarly, flow F_3 is adjusted by another on-off valve V_3.

When the behaviour of the given plant is modelled, it must be considered that the liquid levels, H_1 and H_2, respectively for both tanks will change separately when H_2 crosses level L. At $H_2 = H_1 + L$ the direction of flow through the inter-connecting pipe is reversed. Hence, for F_2, two dynamics must be modelled as:

$$F_2 = \begin{cases} k_1.V_2.\sqrt{(H_1 - H_2 + L)} & if \quad H_2 > L \\ k_1.V_2.\sqrt{(H_1)} & if \quad H_2 \leq L \end{cases} \tag{68}$$

and:

$$\dot{H}_1 = (F_1 - F_2)/k_3 \tag{69}$$

$$\dot{H}_2 = (F_2 - k_2.V_3.\sqrt{H_2})/k_4 \tag{70}$$

where k_1, k_2, k_3, and k_4 are constants depending on characteristic coefficients of the pipes and the cross-section areas of the tanks, and V_2 and V_3 represent the Boolean values in which "1" and "0" signify, respectively, a fully-open valve and a fully-closed valve.

It is evident that the given plant is a nonlinear hybrid system with two inputs (V_2 and V_3) and two outputs (H_1 and H_2). The control problem is defined as:

Using the on-off control valves V_2 and V_3, the liquid heights (H_1 and H_2) in tanks should be derived from an initial state of $H_0 = (0.01,0.01)$ to a target area of R_T with the following conditions:

$$H \in [0,10] \times [0,8] \tag{71}$$

$$H_\mathrm{f} \in R_T = [6,10] \times [6,8] \tag{72}$$

where H_f represents the final desired heights located in the target area (R_T). In addition, the forbidden areas, R_{F1} and R_{F2}, are considered as:

$$H \notin R_{F1} = [0,5] \times [4,8] \tag{73}$$

$$H \notin R_{F2} = [4,10] \times [0,3] \tag{74}$$

The desired liquid heights that lead the actual liquid heights to the target area are defined as:

$$R_{H1} = 0.08r(t) \tag{75}$$

$$R_{H2} = \begin{cases} R_{H1} & if\ R_{H1} < 3.5 \\ 3.5 & if\ 3.5 \le R_{H1} < 5.5 \\ 1.8R_{H1} - 6.4 & if\ 5.5 \le R_{H1} \le 8 \end{cases} \tag{76}$$

where $r(t)$ is a standard ramp function. The final desired heights for both tanks in this simulation are selected as eight meters i.e. $H_f = [8,8]$.

After repeating, simultaneously, the online learning algorithm (Table 3-2) and the stability criteria check (Figure 3-3) for 20 times, the state trajectories of the MANNC control system for the liquid heights are achieved and are shown in Figure 3-20 where the forbidden areas and the desired track are also illustrated.

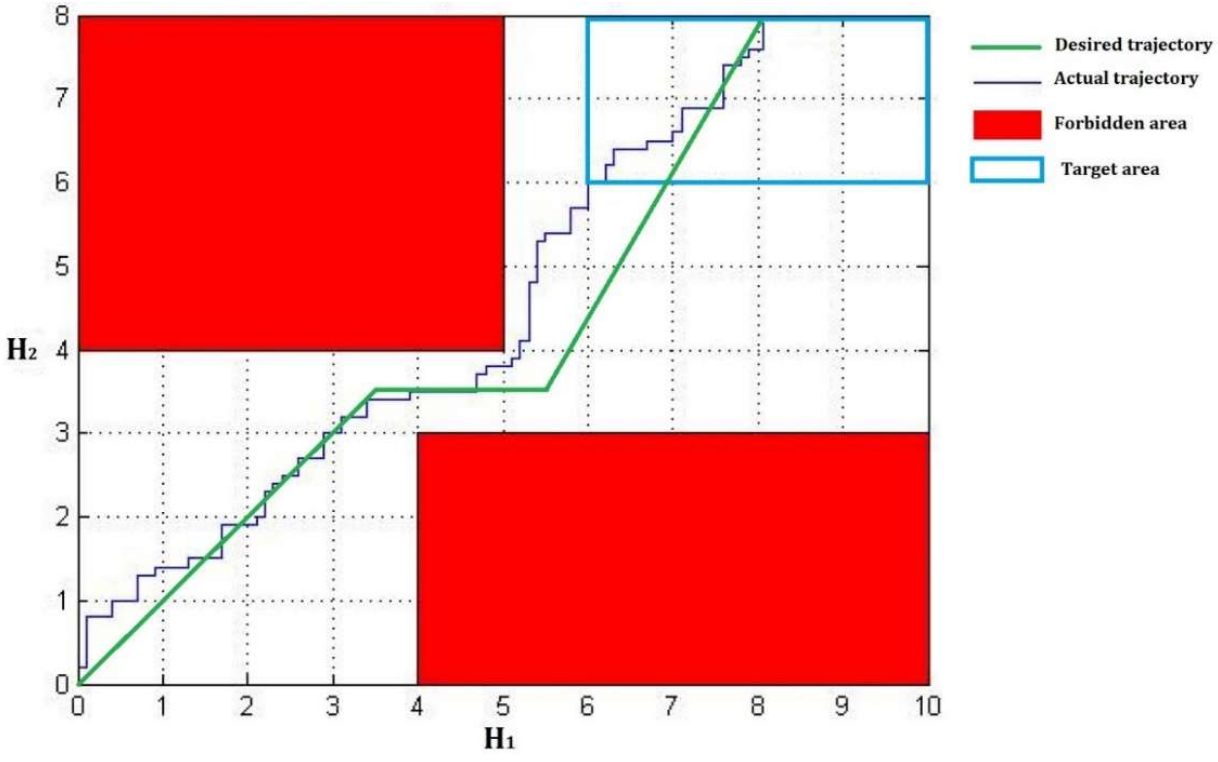


Figure 3-20. Actual and desired state trajectories

It is clear that by using the MANNC in this complex nonlinear hybrid control problem, the liquid height trajectory can pass the narrow zone between the forbidden areas successfully and eventually reach the target area. This is

achieved due to the powerful and fast set-point tracking property of the proposed method that can cope with changes in the system dynamics. In addition, the liquid heights of the two tanks versus time during this process are shown in Figure 3-21.

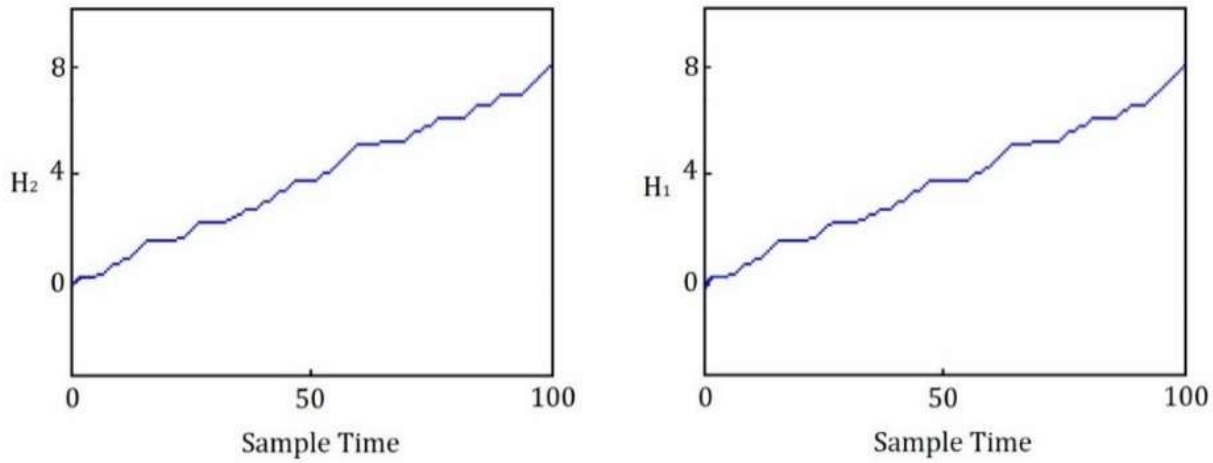


Figure 3-21. Height of liquid in tank T1 and tank T2

The above results are achieved using the final weight values of the output layer of the MANNC neural network, shown in Table 3-8.

$w_{l,x}$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$
$l = 1$	3.1	1.21	2.43	-1.68	-4.34	-5.2
$l = 2$	-0.13	-1.58	4.28	0.88	2.06	3.41

Table 3-8. Final weight values of the output layer

Since the final desired liquid heights can be selected as any points in the target area, simulation studies are carried out for several points in this area and the accumulated errors are computed in a range of 285 to 310, as shown in Figure 3-22. It is clear that, regardless of the location of the selected target point in the target area, the accumulated error is restricted and the convergence is evidently achieved. This justifies the suitability and adequacy of the proposed MANNC in controlling the highly complex hybrid system.

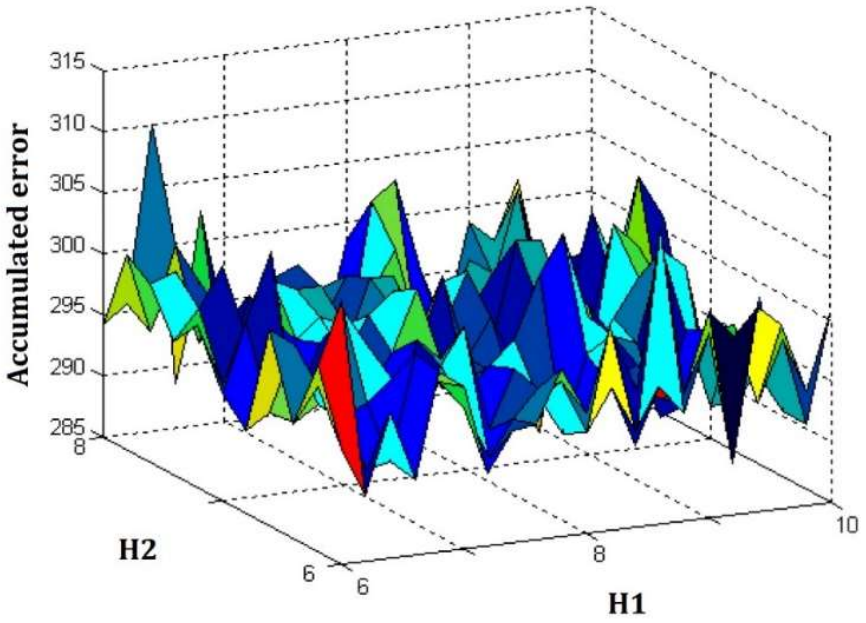


Figure 3-22. Accumulated error for the target area

3.7 Conclusion

By using a powerful learning algorithm designed for its neural network structure and making use of the parallel computational strength offered by its neural network structure, the proposed model-free MANNC is capable of controlling nonlinear square MIMO systems with significant cross-couplings satisfactorily within a short period of time. The MANNC uses a new auto-tune dynamic online learning algorithm with accumulated error back-propagation for the proposed neural network structure, and effectively tunes its weights to achieve the desirable control outcomes. The learning algorithm is integrated with the Lyapunov stability criteria while running and applied to the control system. The effectiveness of the proposed MANNC is validated via simulation studies on typical time-invariant and time-variant square MIMO systems. When compared with the best representative of existing counterparts (i.e., the PIDNN) for applications to both time-invariant and time-variant systems, the MANNC in the time domain is seen to provide less overshoot, less settling time, less accumulated errors, and faster set-point tracking. By selecting an appropriate number of samples, the MANNC can be effectively used for several types of square MIMO control systems in industrial applications, especially when

overshoots in outputs are undesirable and a fast set-point tracking is critical. The simulation results for controlling a complex and challenging MIMO hybrid system demonstrate the superior performance of the MANNC when it deals with significant changes in the system dynamics. As a future study, adding more layers to the neural network structure of the MANNC can be considered in order to further improve the control performance especially for highly nonlinear plants.

References

[1] *Zhou, Q., et al., Approximation-Based Adaptive Tracking Control for MIMO Nonlinear Systems With Input Saturation. IEEE Transactions on Cybernetics, 2015. 45(10): p. 2119-2128.*

[2] *Patino, H.D. and D. Liu, Neural network-based model reference adaptive control system. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2000. 30(1): p. 198-204.*

[3] *Liu, M., Delayed Standard Neural Network Models for Control Systems. IEEE Transactions on Neural Networks, 2007. 18(5): p. 1376-1391.*

[4] *Meng, D., et al., Data-Driven Control for Relative Degree Systems via Iterative Learning. IEEE Transactions on Neural Networks, 2011. 22(12): p. 2213-2225.*

[5] *Oomen, T., et al., Iterative Data-Driven${\cal H}_{\infty}$Norm Estimation of Multivariable Systems With Application to Robust Active Vibration Isolation. IEEE Transactions on Control Systems Technology, 2014. 22(6): p. 2247-2260.*

[6] *Zhang, M. and M. Gan, Data-Driven Adaptive Optimal Control for Linear Systems With Structured Time-Varying Uncertainty. IEEE Access, 2019. 7: p. 9215-9224.*

[7] *Hay, M., et al., Resisting structural re-identification in anonymized social networks. Proc. VLDB Endow., 2008. 1(1): p. 102-114.*

[8] *Donald, C., K.A. Charles, and K.J. Ronald, Control Systems. 2005, McGraw-Hill Education: New York.*

[9] *Fliess, M. and C. Join, Model-free control. International Journal of Control, 2013. 86(12): p. 2228-2252.*

[10] *Lafont, F., et al., A model-free control strategy for an experimental greenhouse with an application to fault accommodation. Computers and Electronics in Agriculture, 2015. 110: p. 139-149.*

[11] *Madadi, E. and D. Söffker, Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control. 2015(57243): p. V001T03A004.*

[12] *Radac, M.B., R.E. Precup, and E.M. Petriu, Model-Free Primitive-Based Iterative Learning Control Approach to Trajectory Tracking of MIMO Systems With Experimental Validation. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2925-2938.*

[13] *Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.*

[14] *Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.*

[15] *Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.*

[16] *Gao, B., et al. Model-free adaptive MIMO control algorithm application in polishing robot. in 2017 6th Data Driven Control and Learning Systems (DDCLS). 2017.*

[17] *Luo, B., et al., Policy Gradient Adaptive Dynamic Programming for Data-Based Optimal Control. IEEE Transactions on Cybernetics, 2017. 47(10): p. 3341-3354.*

[18] *Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.*

[19] *Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.*

[20] Wang, Y., et al., Adaptive decoupling switching control based on generalised predictive control. IET Control Theory & Applications, 2012. 6(12): p. 1828-1841.

[21] Peng, K., et al., A Frequency Domain Decoupling Method and Multivariable Controller Design for Turbofan Engines. IEEE Access, 2017. 5: p. 27757-27766.

[22] Nguyen, T.N., S. Su, and H.T. Nguyen, Neural Network Based Diagonal Decoupling Control of Powered Wheelchair Systems. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2014. 22(2): p. 371-378.

[23] Cong, S. and Y. Liang, PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems. IEEE Transactions on Industrial Electronics, 2009. 56(10): p. 3872-3879.

[24] Hwang, C.L. and C. Jan, Recurrent-Neural-Network-Based Multivariable Adaptive Control for a Class of Nonlinear Dynamic Systems With Time-Varying Delay. IEEE Transactions on Neural Networks and Learning Systems, 2016. 27(2): p. 388-401.

[25] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[26] Merabet, A., A.A. Tanvir, and K. Beddek, Speed control of sensorless induction generator by artificial neural network in wind energy conversion system. IET Renewable Power Generation, 2016. 10(10): p. 1597-1606.

[27] Yang, B. and A.J. Calise, Adaptive Control of a Class of Nonaffine Systems Using Neural Networks. IEEE Transactions on Neural Networks, 2007. 18(4): p. 1149-1159.

[28] Tee, K.P., S.S. Ge, and F.E.H. Tay, Adaptive Neural Network Control for Helicopters in Vertical Flight. IEEE Transactions on Control Systems Technology, 2008. 16(4): p. 753-762.

[29] Park, J., S. Kim, and C. Moon, Adaptive Neural Control for Strict-Feedback Nonlinear Systems Without Backstepping. IEEE Transactions on Neural Networks, 2009. 20(7): p. 1204-1209.

[30] Liu, Y., et al., Adaptive Neural Output Feedback Controller Design With Reduced-Order Observer for a Class of Uncertain Nonlinear SISO Systems. IEEE Transactions on Neural Networks, 2011. 22(8): p. 1328-1334.

[31] Jin, Z., S.S. Ge, and L. Tong Heng, Output feedback control of a class of discrete MIMO nonlinear systems with triangular form inputs. IEEE Transactions on Neural Networks, 2005. 16(6): p. 1491-1503.

[32] Zhang, T. and S.S. Ge, Adaptive Neural Network Tracking Control of MIMO Nonlinear Systems With Unknown Dead Zones and Control Directions. IEEE Transactions on Neural Networks, 2009. 20(3): p. 483-497.

[33] Chen, M., S.S. Ge, and B.V.E. How, Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities. IEEE Transactions on Neural Networks, 2010. 21(5): p. 796-812.

[34] Yang, Q., Z. Yang, and Y. Sun, Universal Neural Network Control of MIMO Uncertain Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2012. 23(7): p. 1163-1169.

[35] Chen, Z., et al., Adaptive Neural Control of MIMO Nonlinear Systems With a Block-Triangular Pure-Feedback Control Structure. IEEE Transactions on Neural Networks and Learning Systems, 2014. 25(11): p. 2017-2029.

[36] Meng, W., Q. Yang, and Y. Sun, Adaptive Neural Control of Nonlinear MIMO Systems With Time-Varying Output Constraints. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(5): p. 1074-1085.

[37] Yang, Q., S. Jagannathan, and Y. Sun, Robust Integral of Neural Network and Error Sign Control of MIMO Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(12): p. 3278-3286.

[38] Ronald, J., NEURAL NETWORK APPLICATIONS, in Electronic Engine Control Technologies. 2004, SAE. p. 661-661.

[39] Yong, Z., Z. Hai-bo, and L. Tian-qi. PIDNN decoupling control of boiler combustion system based on MCS. in 2017 29th Chinese Control And Decision Conference (CCDC). 2017.

[40] Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.

[41] Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.

[42] Shu, H. and Y. Xu. Application of additional momentum in PID neural network. in 2014 4th IEEE International Conference on Information Science and Technology. 2014.

[43] Meng, L., et al. Design of an improved PID neural network controller based on particle swarm optimazation. in 2015 Chinese Automation Congress (CAC). 2015.

[44] Teng, W., H. Pan, and J. Ren. Neural network PID decoupling control based on chaos particle swarm optimization. in Proceedings of the 33rd Chinese Control Conference. 2014.

[45] Tian, Z., et al. A PID Neural Network Control for Position Servo System with Gear Box at Variable Load. in 2016 IEEE Vehicle Power and Propulsion Conference (VPPC). 2016.

[46] Bahri, N., et al. Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems. in 2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14). 2014.

[47] Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.

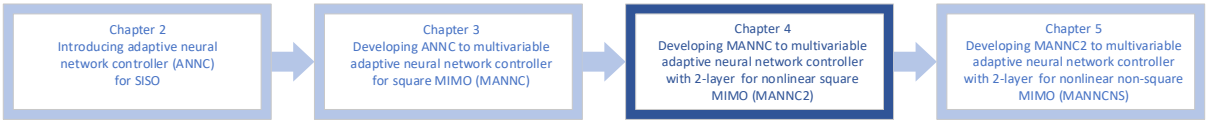
[48] Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.

[49] Phillips, S.F. and D.E. Seborg. Conditions that Guarantee No Overshoot for Linear Systems. in 1987 American Control Conference. 1987.

CHAPTER 4: MODIFICATION of MODEL-FREE MULTIVARIABLE ADAPTIVE NEURAL NETWORK CONTROLLER USING TWO DYNAMIC LAYERS FOR CONTROLLING NONLINEAR SQUARE MIMO SYSTEMS

Aim

In this chapter, a Multivariable Adaptive Neural Network Controller with two dynamic layers (MANNC2) for application in square MIMO systems is introduced. This controller is designed in such a way that the new proposed structure is able to improve the control performance of the Multivariable Adaptive Neural Network Controller (MANNC) introduced in Chapter 3 in controlling coupled square MIMO systems with highly nonlinear characteristics. The MANNC2 presented in this chapter will be able to be re-structured and further developed for controlling black-box non-square MIMO systems in Chapter 5 as per diagram below:



Description

To achieve this aim, firstly, by considering the basic structure of the Adaptive Neural Network Controller (ANNC) introduced in Chapter 2 and using the neural network structure of the MANNC proposed in Chapter 3, a modified multivariable neural network structure for controlling highly nonlinear square MIMO systems is represented. By acknowledging the fact that using two adjustable layers in neural networks can potentially improve the performance of a controller for nonlinear systems, the new modified neural network structure is designed with two dynamic layers. Subsequently, modified Lyapunov stability criteria together with a set of dynamic learning methods are developed for the new neural network structure. The modified learning algorithms for both layers and their associated stability-criterion checks are executed online simultaneously in order to guarantee the real-time performance of the controller

and to ensure the corresponding global asymptotical stability of the resulting control system during the neural network weight training process at all time. To demonstrate the improved performance of the MANNC2 versus that of the MANNC introduced in Chapter 3 for nonlinear systems, a time-invariant drum boiler and a twin-tank level-control system are chosen for which the new MANNC2 is tested and compared with the results of the MANNC. The above-mentioned contents are described in a paper:

- Arash Mehrafrooz and Fangpo He, "Modification of Model-Free Multivariable Adaptive Neural Network Controller Using Two Dynamic Layers for Controlling Nonlinear Square MIMO Systems" Submitted to Journal of "IEEE Transactions on Control Systems Technology"

The outline of the paper is given below:

Results

The original contribution of this study is the development of an advanced control method including the new neural network structure of the controller and its

associated learning algorithm as applied to coupled square multivariable systems. The obtained simulation results clarify the significantly improved set-point tracking and substantial error reduction with the controller's powerful auto-tuning capability compared to the MANNC introduced in Chapter 3 for non-linear square MIMO systems while the system remains stable during the entire training process.

Conclusion

Overall, this chapter demonstrates that using the powerful learning abilities inherent in a neural network approach, the proposed MANNC2 is capable of controlling model-free coupled MIMO systems with highly nonlinear properties to achieve more reliable control outcomes as compared to the MANNC. Although the current structure of the MANNC2 is only designed for square MIMO systems, it potentially can be further developed into a universal controller for non-square MIMO systems with nonlinear characteristics, as demonstrated in the next chapter.

Modification of Model-Free Multivariable Adaptive Neural Network Controller Using Two Dynamic Layers for Controlling Nonlinear Square MIMO Systems

Arash Mehrafrooz[1, a] and Fangpo He[1, b]

[1]Advanced Control Systems Research Group, College of Science and Engineering, Flinders University, Australia

[a]<arash.mehrafrooz@flinders.edu.au>, [b]<fangpo.he@flinders.edu.au>

Abstract. **In this paper, a novel model-free Multivariable Adaptive Neural Network Controller with two-dynamic layers (MANNC2) is introduced for controlling black-box coupled square Multi-Input Multi-Output (MIMO) systems with highly nonlinear characteristics. Specific learning algorithms with accumulated gradients in the error back-propagation algorithm are developed for the output and hidden layers of the MANNC2, and the Lyapunov stability criteria for ensuring the real-time global asymptotic stability of the ensuing control system are established. The performance of the two-layer MANNC2 is compared with its single-layer predecessor, the MANNC introduced in [1], in two highly nonlinear square MIMO cases. The simulation results approve the superiority of the MANNC2 over its counterpart. The usefulness of the MANNC2 in dealing with a wider range of nonlinear characteristics makes it a potential candidature for use in many practical industrial applications.**

Key Words – square MIMO; coupled MIMO; multi-layer neural networks; accumulated error back-propagation; Nonlinear control; Lyapunov stability;

4.1 Introduction

Current advanced control techniques (such as adaptive control, robust control, sliding-mode control, and back-stepping control) that are used for controlling nonlinear Multiple-Input Multiple-Output (MIMO) systems are largely developed using model-based approaches [2]. Although model-based approaches can function satisfactorily in many industrial applications, they suffer from several limitations in real-time operations. For instance, a model-based controller may be difficult to cope with modelling errors or structural uncertainties. It may have a limited effective working range, and produce

uncertain parameters that may need to be properly identified in a timely fashion during online operations [3, 4]. In addition, model-based nonlinear MIMO controllers are usually limited to specific classes of nonlinear systems or applications, and cannot be generalised as universal or generic control strategies. Considering these limitations associated with model-based approaches, model-free approaches [5] that can avoid the complications involved in system modelling and identification have been adopted in engineering control systems in recent years. In a model-free approach, a massive amount of input/output data samples of a system of concern is used for controller design and performance monitoring, without the need for a system model. This is particularly useful when no sufficient information about the system is available for producing a priori model of the system. For improving the system dynamic performance, a model-free approach can afford to use a more advanced control strategy that would otherwise rely heavily on the accuracy of an existing model of the system and thus be vulnerable for any changes in the system if a model-based approach were to be employed. Several model-free control system designs can be found in the recent literature [6-11], in which the effectiveness of the model-free approaches in industrial applications is demonstrated.

From the literature, neural network techniques have been seen as a popular and powerful model-free approach that can be employed for controlling poorly-known nonlinear dynamics [12, 13]. Using their highly adequate approximation capabilities catered for real-time operations, various neural network control schemes have been applied to industrial plants where a required modelling process would have been challenging [14-16]. In [1], a model-free Multivariable Adaptive Neural Network Controller (MANNC) was introduced, in which a neural network of one layer of adjustable weights is used for controlling black-box coupled square $(n \times n)$ MIMO systems. Although the MANNC has been successfully tested using real industrial examples with different types of MIMO characteristics, and its superior performance has been evident shown as compared to its most competitive existing counterpart, its potential in dealing with highly nonlinear systems is still limited by its inherent nature of being a single-dynamic layer controller. Typically, a single-dynamic layer neural

network is in theory considered as a linear classifier, and as such may not be very effective in learning highly nonlinear relationships such as the "XOR" function among other nonlinearly separable classification problems [17-19]. Though the MANNC has not been purposely designed for working with "XOR"-like nonlinear systems and the time-varying nature of the MANNC can indeed account for nonlinearities to a certain degree, the capacity of the controller can still be significantly improved if the number of its dynamic layer is increased. As supported by the literature, to maximize the control outcomes for systems with highly nonlinear characteristics and to develop a more universal controller useful for a wider range of nonlinear systems, neural networks with more than one-dynamic layer are preferred. For this reason, in this study, the former single-layer MANNC is further developed to include an additional layer and to form a new Multivariable Adaptive Neural Network Controller with two-dynamic layers (MANNC2). Unlike the two-layer Adaptive Neural Network Controller (ANNC) introduced in [13] for Single-Input Single-Output (SISO) systems where the hidden layer of the controller is static with no associated learning algorithm, the newly proposed MANNC2 will be for coupled square MIMO systems with both layers of the controller being dynamic and with their associated learning algorithms. Furthermore, unlike many existing neural-network controllers where the critical closed-loop system stability issue has either been ignored [20-25] or been bypassed (by using conservative learning rates that will reduce the instability risk, but will sacrifice the training speed and lead to a low quality control performance [26]), the stability issue of the MANNC2 method will be investigated in this study to bring up the necessary conditions upon which the resultant closed-loop system stability is guaranteed in real-time operations. It is anticipated that the proposed MANNC2 will outperform the MANNC and further advance the performances presented by other existing counterparts.

The proposed MANNC2 will be designed to possess the following features that can distinguish itself from the previous MANNC and the other existing counterparts:

(i) A new two-dynamic layer neural-network structure with cross-coupled

connections in the output layer will be used to construct the MANNC2 – This structure will enable the MANNC2 to control coupled square MIMO systems with highly nonlinear characteristics.

(ii) Specific learning algorithms with accumulated gradients in the error back-propagation algorithm will be developed for the output and hidden layers of the MANNC2, together with the development of new partial derivative estimations for the MANNC2 – This development will give the MANNC2 a much-improved set-point tracking ability over those of the MANNC and the other existing counterparts ([25], [12, 27-29]).

(iii) The closed-loop system stability of the MANNC2 will be investigated using the Lyapunov stability conditions to yield the constraints for the weight training of both dynamic layers – This investigation will result in a simultaneous weight-training and stability-checking process of the MANNC2 to be executed in real time in order to guarantee its closed-loop system stability at all time.

(iv) Dynamic learning rates for the learning algorithms of the MANNC2 will be used to allow the controller to automatically adjust itself to avoid its neural networks falling into the local minimum – This arrangement will permit a continuous weight training process of the MANNC2 to deliver improved output responses with minimum errors, less overshoots and undershoots, and faster settling times.

(v) The accumulated errors versus the numbers of iterations of the learning algorithms of the MANNC2 will be automatically monitored constantly to produce an optimal training-step number for each specific application based on required control specifications – This optimal number will allow the MANNC2 to determine its ideal control speed whilst maintaining its best control performance.

It should be pointed out that although the MANNC2 is designed for square MIMO systems as many multivariable systems in industrial applications are in square forms [30], it can be applied to a non-square MIMO case where the given system can be converted into a square form by adding or removing its inputs or

outputs. Non-square multivariable industrial plants can often be reformed into desirable square MIMO forms with which the proposed MANNC2 can be applied.

To demonstrate the proposed MANNC2, the rest of this paper is organised as follows. In Section 4.2, the structure and matrix representation of the new neural- network-based adaptive controller is introduced. In Section 4.3, the learning method of the neural networks using the error back-propagation algorithm is described. The closed-loop system stability criteria are developed in Section 4.4. Section 4.5 demonstrates the simulation results where the proposed method is applied to two highly nonlinear MIMO cases and, in each case, the performance of the MANNC2 is compared with that of the former MANNC to validate the anticipated improvement quantitatively. Meaningful conclusions in relation to the MANNC2 design are given in Section 4.6.

4.2 Multivariable Adaptive Neural Network Controller (MANNC2)

4.2.1 Closed-loop Structure of MANNC2

For the proposed P-type, I-type, and D-type neurons in [1], the outputs of these neurons in discrete form can be expressed respectively as:

$$o_P(k) = \begin{cases} 1 & net_P(k) > 1 \\ net_P(k) & -1 \le net_P(k) \le 1 \\ -1 & net_P(k) < -1 \end{cases} \quad (1)$$

$$o_I(k) = \begin{cases} 1 & net_I(k) > 1 \\ o_I(k-1) + net_I(k) & -1 \le net_I(k) \le 1 \\ -1 & net_I(k) < -1 \end{cases} \quad (2)$$

$$o_D(k) = \begin{cases} 1 & net_D(k) > 1 \\ net_D(k) - net_D(k-1) & -1 \le net_D(k) \le 1 \\ -1 & net_D(k) < -1 \end{cases} \quad (3)$$

where $o_X(k)$ and $net_X(k)$ are the X-type neuron's output and the X-type neuron's sum of inputs at the k^{th} sampling time, respectively, as illustrated in Figure 4-1.

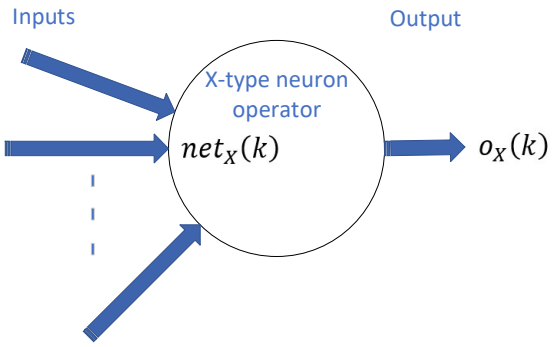


Figure 4-1.General structure of a X-type neuron

Considering the structure of the ANNC proposed in [13] and the MANNC proposed in [1], a new Multivariable Adaptive Neural Network Controller with two- dynamic layers (MANNC2), illustrated in Figure 4-2 is proposed as a closed-loop MIMO controller that is applicable to coupled square MIMO nonlinear systems. In the structure of the MANNC2, in contrast to the MANNC in [1], the errors of the outputs are not generated in the neural network structure. The sum of the weighted desired outputs and the weighted actual outputs are produced in the first dynamic layer of the controller and, consequently, propagated to the second dynamic layer of the controller.

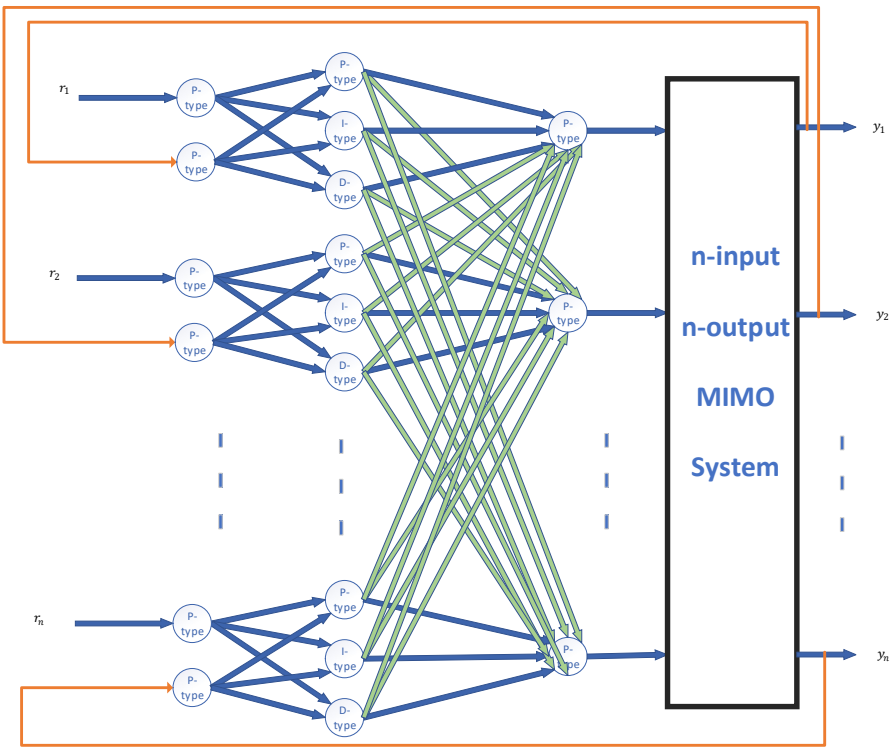


Figure 4-2. MANNC2 control system structure

Structurally, the MANNC2 contains three layers: the input layer, the hidden layer, and the output layer; only the latter two layers are dynamic layers. There are $2n$ P-type neurons within the input layer. In the hidden layer, there are $3n$ neurons including P-type, I-type, and D-type neurons in clusters of three, respectively. In the output layer, there are n P-type neurons to generate the outputs of the MANNC2. The outputs of the output-layer neurons enter into the multivariable system as inputs. There are $6n$ adjustable weights in the hidden layer which are multiplied by the input layer's outputs to make the hidden layer's inputs. In addition, there are $3n^2$ adjustable weights in the output layer associated with the hidden-layer neurons, and they determine the impact of each neuron in the hidden layer and generate the multivariable system's inputs.

4.2.2 Structure of Sub-MANNC2 (S-MANNC2)

Using the concept of the sub-controllers introduced in [1] for the MANNC, the proposed MANNC2's structure illustrated in Figure 4-2 is further decomposed into n-parallel sub-controllers each named as a Sub-MANNC2 (S-MANNC2) as shown in Figure 4-3. The functionality of the sub-controllers is to enable the neural-network controller to account for the cross-couplings of an $n \times n$ MIMO system of concern. For the l^{th} S-MANNC2 with input r_l (which is also the desired output) and (actual) output y_l ($l = 1, 2, \dots, n$) as shown in Figure 4-3, there are in total three layers and six neurons. The first layer is called the input layer and contains two 'P-type' neurons. The desired output (r_l) and the actual output (y_l) propagate via those 'P-type' neurons to the second layer named the hidden layer. Within the hidden layer, there are three neurons each being a P-type, an I-type, and a D-type, respectively. The P-neuron is responsible for producing the total sum of the weighted desired outputs and the weighted actual outputs, the I-neuron provides the necessary action to eliminate the steady-state error, and the D-neuron predicts the future behaviour of the error. The third layer is named as the output layer that accumulates the outputs of the hidden layer, generates the control commands, and applies the control commands to the $n \times n$ MIMO system. Since the MANNC2 is used for controlling coupled multivariable systems, the output neuron of each S-MANNC2 contains 'n' number of inputs in order to be

able to generate each control command respective to all desired outputs and actual outputs. The inputs of the P-type, I-type, and D-type (named net^1_{3l-2}, net^1_{3l-1}, and net^1_{3l}, respectively) and the outputs of these neurons (named O^1_{3l-2}, O^1_{3l-1}, and O^1_{3l}, respectively) are related together by the activation functions of the neurons represented in (1)-(3). In the hidden layer of the S-MANNC2, there are in total six weights ($\bar{w}_{1,2l-1}$, $\bar{w}_{2,2l-1}$, $\bar{w}_{3,2l-1}$, $\bar{w}_{1,2l}$, $\bar{w}_{2,2l}$, and $\bar{w}_{3,2l}$) that are associated with the neurons of the input layer and hidden layer. Likewise, in the output layer of the S-MANNC2, there are in total three weights ($w_{1,3l-2}$, $w_{1,3l-1}$, and $w_{1,3l}$) that are associated with the P-type, I-type, and D-type neurons in the hidden layer as well as the P-type neuron in the output layer.

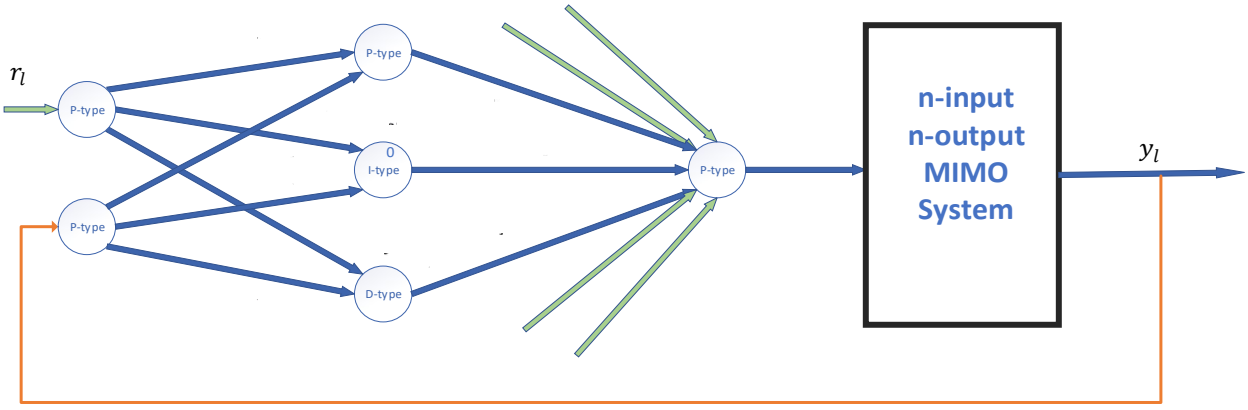


Figure 4-3. S-MANNC2 structure

4.2.3 Matrix representation

The matrix representation of a closed-loop square MIMO control system using the proposed MANNC2 (Figure 4-2) and S-MANNC2 (Figure 4-3) is derived as follows.

Let O^2_l and y_l be, respectively, the l^{th} input and the l^{th} output of the system where $1 \leq l \leq n$, and let G_{ij} be the transfer function relating the j^{th}-component of the i^{th} output (y_i) to the j^{th} input (O^2_j) where $1 \leq i \leq n$ and $1 \leq j \leq n$. The

vectors and matrices associated with Figure 4-2 and Figure 4-3 are defined as follows.

$$Y = Y_{n\times 1} = [y_1 \quad y_2 \quad \cdots \quad y_l \quad \cdots \quad y_n]_{1\times n}{}^T \tag{4}$$

$$O^2 = O^2_{n\times 1} = [O^2_1 \quad O^2_2 \quad \cdots \quad O^2_l \quad \cdots \quad O^2_n]_{1\times n}{}^T \tag{5}$$

$$G = G_{n\times n} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1l} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2l} & \cdots & G_{2n} \\ \vdots & \vdots & & & & \vdots \\ G_{l1} & G_{l2} & \cdots & G_{ll} & \cdots & G_{ln} \\ \vdots & \vdots & & \vdots & & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nl} & \cdots & G_{nn} \end{bmatrix}_{n\times n} \tag{6}$$

$$net^2 = net^2_{n\times 1} = [net^2_1 \quad net^2_2 \quad \cdots \quad net^2_l \quad \cdots \quad net^2_n]_{1\times n}{}^T \tag{7}$$

$$W = W_{n\times 3n} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,3l-2} & w_{1,3l-1} & w_{1,3l} & \cdots & w_{1,3n-2} & w_{1,3n-1} & w_{1,3n} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,3l-2} & w_{2,3l-1} & w_{2,3l} & \cdots & w_{2,3n-2} & w_{2,3n-1} & w_{2,3n} \\ \vdots & \vdots & & & & \vdots & \vdots & & & & \vdots \\ w_{l,1} & w_{l,2} & w_{l,3} & \cdots & w_{l,3l-2} & w_{l,3l-1} & w_{l,3l} & \cdots & w_{l,3n-2} & w_{l,3n-1} & w_{l,3n} \\ \vdots & \vdots & & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ w_{n,1} & w_{n,2} & w_{n,3} & \cdots & w_{n,3l-2} & w_{n,3l-1} & w_{n,3l} & \cdots & w_{n,3n-2} & w_{n,3n-1} & w_{n,3n} \end{bmatrix}_{n\times 3n} \tag{8}$$

$$O^1 = O^1_{3n\times 1} = [O^1_1 \quad O^1_2 \quad O^1_3 \quad \cdots \quad O^1_{3n}]_{1\times 3n}{}^T \tag{9}$$

$$P = P_{3n\times 3n} = \begin{bmatrix} 1 & 0 & 0 & & & \\ 0 & D^{-1} & 0 & \cdots & \mathbf{0} & \\ 0 & 0 & D & & & \\ & \vdots & & \ddots & \vdots & \\ & & & 1 & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & D^{-1} & 0 \\ & & & & 0 & 0 & D \end{bmatrix}_{3n\times 3n} \tag{10}$$

$$net^1 = net^1_{3n\times 1} = [net^1_1 \quad net^1_2 \quad net^1_3 \quad \cdots \quad net^1_{3n}]_{3n\times 1}{}^T \tag{11}$$

$$\overline{W}^r = \overline{W}^r_{3n\times 3n} = \begin{bmatrix} \overline{w_{1,1}} & 0 & 0 & & & \\ 0 & \overline{w_{2,1}} & 0 & \cdots & \mathbf{0} & \\ 0 & 0 & \overline{w_{3,1}} & & & \\ & \vdots & & \ddots & \vdots & \\ & & & \overline{w_{1,2n-1}} & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & \overline{w_{2,2n-1}} & 0 \\ & & & & 0 & 0 & \overline{w_{3,2n-1}} \end{bmatrix}_{3n\times 3n} \tag{12}$$

$$\bar{W}^{y}=\bar{W}^{y}_{3n\times 3n}=\begin{bmatrix} \overline{w_{1,2}} & 0 & 0 & & & & \\ 0 & \overline{w_{2,2}} & 0 & \cdots & & \mathbf{0} & \\ 0 & 0 & \overline{w_{3,2}} & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & \overline{w_{1,2n}} & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & \overline{w_{2,2n}} & 0 \\ & & & & 0 & 0 & \overline{w_{3,2n}} \end{bmatrix}_{3n\times 3n} \tag{13}$$

$$R^{(3)}=R^{(3)}_{3n\times 1}=[r_1 \quad r_1 \quad r_1 \quad r_2 \quad r_2 \quad r_2 \quad \cdots \quad r_n \quad r_n \quad r_n]_{1\times 3n}{}^{T} \tag{14}$$

$$Y^{(3)}=Y^{(3)}_{3n\times 1}=[y_1 \quad y_1 \quad y_1 \quad y_2 \quad y_2 \quad y_2 \quad \cdots \quad y_n \quad y_n \quad y_n]_{1\times 3n}{}^{T} \tag{15}$$

$$I^{(3)}=I^{(3)}_{3n\times n}=\begin{bmatrix} 1 & & \\ 1 & \cdots & 0 \\ 1 & & \\ \vdots & \ddots & \vdots \\ & & 1 \\ 0 & \cdots & 1 \\ & & 1 \end{bmatrix}_{3n\times n} \tag{16}$$

$$R=R_{n\times 1}=[r_1 \quad r_2 \quad r_3 \quad \cdots \quad r_n]_{1\times n}{}^{T} \tag{17}$$

Assuming that the nonlinear system of concern is linearized around an operation point, the relationship between the inputs and outputs of the system will be:

$$Y_{n\times 1}=G_{n\times n}O^{2}_{n\times 1} \tag{18}$$

Since net^{2}_{l} is the l^{th} input of the P-type neuron in the output layer, one has:

$$O^{2}_{n\times 1}=net^{2}_{n\times 1} \tag{19}$$

where the relationship between the inputs and outputs in the output layer of the neural networks can be expressed as:

$$net^{2}_{n\times 1}=W^{2}_{n\times 3n}O^{1}_{3n\times 1} \tag{20}$$

As the hidden layer holds P-type, I-type, and D-type neurons, the proportional, integral, and derivative operators (1, D^{-1}, and D) are considered respectively in the matrix form of the activation function. Hence, one has:

$$O^{1}_{3n\times 1}=P_{3n\times 3n}net^{1}_{3n\times 1} \tag{21}$$

where the inputs of the hidden-layer neurons are the total sum of the weighted desired outputs and the actual outputs, and are described as:

$$net^{1}_{3n\times 1}=\bar{W}^{r}_{3n\times 3n}R^{3}_{3n\times 1}+\bar{W}^{y}_{3n\times 3n}Y^{3}_{3n\times 1} \tag{22}$$

By having:

$$R_{3n\times1}^{(3)} = I_{3n\times n}^{(3)} \times R_{n\times1} \tag{23}$$

and:

$$Y_{3n\times1}^{(3)} = I_{3n\times n}^{(3)} \times Y_{n\times1} \tag{24}$$

and using (18)-(22), one obtains:

$$Y = GO^2 = G.net^2 = GWO^1 = GWP.net^1 = GWP(\bar{W}^r R^3 + \bar{W}^y Y^3) \tag{25}$$

Substituting (23) and (24) into (25), one has:

$$Y = \mathrm{GWP}(\bar{W}^r I^{(3)} R + \bar{W}^y I^{(3)} Y) \tag{26}$$

Hence, the system output can be derived as:

$$Y = \left(I - GWP\bar{W}^y I^{(3)}\right)^{-1} GWP\bar{W}^r I^{(3)} R \tag{27}$$

where: $\left|I - GWP\bar{W}^y I^{(3)}\right| \neq 0$.

It should be pointed out that although Equations (18)-(27) are derived under the assumption that the system is linear or can be linearized around an operating point, they can potentially be used for nonlinear systems where the nonlinearities of the systems can be approximated by piece-wise linear systems whose time-varying nature can account for the nonlinearities of the systems satisfactorily. It should also be pointed out that if the non-singularity condition for $\left(I - GWP\bar{W}^y I^{(3)}\right)$ could not be met, the selected weights (matrices W, $\bar{W}^y$, and $\bar{W}^r$) would not be acceptable and would be re-updated until $\left(I - GWP\bar{W}^y I^{(3)}\right)$ becomes non-singular.

4.3 Learning algorithm

To achieve a precise control effect for a square MIMO system, the MANNC2 weights are adjusted using the principle of the multi-step error back-propagation algorithm described in [31]. However, instead of using the current gradient of the system error as the literature does for SISO systems in neural-network-based controllers [27-29], an accumulated gradient of the system error that utilises the full history of the system outputs and the desired outputs is used to achieve a

more accurate control result. The proposed method minimises the sum of the squared accumulated gradient of the error for each system output, where the error is taken as the difference between the desired output $r_l(k)$ (i.e., the system set-point) and the actual output $y_l(k)$. Euclidean Norm (E) is defined for calculating the quadratic cost function of the system for the system error. Power of two in this expression makes the error of each sample positive so that larger errors become weightier than the smaller errors. Thus, the cost function for the l^{th} S-MANNC2 (Figure 4-3) is defined as:

$$E_l(h) = \frac{1}{2}\left(\sum_{k=1}^{m}(r_l[k] - y_l(h)[k])\right)^2 \tag{28}$$

where $E_l(h)$ is the error of the l^{th} output at the h^{th} step of the learning algorithm, and m is the required number of discrete samples of the actual output and desired output. By increasing m, the system output will be compared more accurately with the system set-point. However, a large value of m may slow down the controller's learning process, which is undesirable especially when the speed of the control action is critical in many demanding real-time industrial operations. Therefore, a reasonable value of m must be chosen to make a necessary trade-off between the desired control accuracy and the essential speed of the control system.

The total cost function of the system (J) which is the sum of all errors described in (28) is expressed as:

$$J(h) = \sum_{l=1}^{n} E_l(h) = \frac{1}{2}\sum_{l=1}^{n}\left(\sum_{k=1}^{m}(r_l[k] - y_l(h)[k])\right)^2 \tag{29}$$

where n is the number of inputs and outputs of the square MIMO system. This expression will be used to derive the learning algorithm for weights of both dynamic layers of the neural networks of the MANNC2.

Using the accumulated gradient of the system error, a learning algorithm is to be developed to minimise the defined cost function and to bring the system actual outputs as close as possible to the system desired outputs. There are two-dynamic layers of weights in the neural network structure of the MANNC2 that are required to be trained: the output layer and the hidden layer, for which two learning algorithms are derived in Subsections 1.3.1 and 1.3.2, respectively.

4.3.1 Output Layer Weight Learning Algorithm

A learning algorithm of the output layer is defined to train the output-layer weights and to link with the learning algorithm of the hidden layer (to be represented in the following subsection) together in order to run simultaneously during a control action. According to the principle of the error back-propagation learning algorithm, the output-layer weights must be adjusted so that in each step they slightly move in the opposite direction of the gradient of the cost function respective to the weights of the output layer. This is to guarantee that the cost function will be decreasing gradually while the learning algorithm is running. Therefore, the weights between the hidden layer and the output layer will be adjusted based on the following learning rule:

$$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l \frac{\partial J(h)}{\partial w_{l,x}} \tag{30}$$

where $3l-2 \leq x \leq 3l$; $1 \leq l \leq n$; h is the step number of the learning algorithm; $w_{l,x}(h)$ and $w_{l,x}(h+1)$ are the weights of the output layer at the current and following steps, respectively; λ_l is the output-layer learning rate that decides how fast the cost is changing and principally determines the weight-training speed of the output layer. Using partial derivatives to calculate the gradient of the error subject to each weight, one has:

$$\frac{\partial J}{\partial w_{l,x}} = \frac{\partial J}{\partial E_l}\frac{\partial E_l}{\partial y_l}\frac{\partial y_l}{\partial O_l^2}\frac{\partial O_l^2}{\partial net_l^2}\frac{\partial net_l^2}{\partial w_{l,x}} \tag{31}$$

where:

$$\frac{\partial J}{\partial E_l} = 1 \tag{32}$$

$$\frac{\partial E_l}{\partial y_l} = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \tag{33}$$

By a reasonable approximation, one has:

$$\frac{\partial y_l}{\partial O_l^2} \cong \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{34}$$

Due to having P-type neuron in the output layer, one writes:

$$\frac{\partial O_l^2}{\partial net_l^2} = 1 \tag{35}$$

and:

$$\frac{\partial net_l^2}{\partial w_{l,x}} = \sum_{k=1}^{m}(O_x^1(h)[k]) \tag{36}$$

Substituting (32)-(36) into (31), one obtains:

$$\frac{\partial J}{\partial w_{l,x}} \cong 1 \times \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \times 1 \times O_x^1(h)[k] \tag{37}$$

Defining $\gamma_l(k)$ as:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{38}$$

the output-layer weight-adjustment rule is derived as:

$$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l[\gamma_l(h) \times \sum_{k=1}^{m}(O_x^1(h)[k])] \tag{39}$$

4.3.2 Hidden Layer Weight Learning Algorithm

Similar to the weight training algorithm performed for the output layer, according to the principle of the error back-propagation learning algorithm, the hidden-layer weights (i.e, the weights between the input layer and the hidden layer) must be adjusted so that in each step they move slightly in the opposite direction of the gradient of the cost function respective to the weights of the hidden layer. This will cause the cost function to decrease gradually during the learning process. The hidden-layer weights are therefore adjusted based on the following learning rule:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \frac{\partial J(h)}{\partial \overline{w_{a,b}}} \tag{40}$$

where $1 \leq a \leq 3$; $2l-1 \leq b \leq 2l$; $1 \leq l \leq n$; h is the step number of the learning algorithm; $\overline{w_{a,b}}(h)$ and $\overline{w_{a,b}}(h+1)$ are the weights of the hidden layer at the current and following steps, respectively; μ_l is the hidden-layer learning rate that decides how fast the cost is changing and principally determines the weight-training speed of the hidden layer. The gradient of the error with respect to each weight is required to be calculated. Using partial derivatives, one has:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \frac{\partial J}{\partial E_l}\frac{\partial E_l}{\partial y_l}\frac{\partial y_l}{\partial O_l^2}\frac{\partial O_l^2}{\partial net_l^2}\frac{\partial net_l^2}{\partial O_x^1}\frac{\partial O_x^1}{\partial net_x^1}\frac{\partial ne\ ^1_x}{\partial \overline{w_{a,b}}} \tag{41}$$

Since $x = 3l - 3 + a$, x must be chosen adequately based on a so that a proper partial derivative path within the hidden layer can be selected. Considering the partial derivative calculations of the output layer, one has:

$$\frac{\partial J}{\partial E_l} \times \frac{\partial E_l}{\partial y_l} \times \frac{\partial y_l}{\partial O_l^2} \times \frac{\partial O_l^2}{\partial net_l^2} = \sum_{k=1}^{m} (y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m} (y_l(h)[k]) - \sum_{k=1}^{m} (y_l(h-1)[k])}{\sum_{k=1}^{m} (O_l^2(h)[k]) - \sum_{k=1}^{m} (O_l^2(h-1)[k])} \quad (42)$$

Define $\gamma_l(h)$ as:

$$\gamma_l(h) = \sum_{k=1}^{m} (y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m} (y_l(h)[k]) - \sum_{k=1}^{m} (y_l(h-1)[k])}{\sum_{k=1}^{m} (O_l^2(h)[k]) - \sum_{k=1}^{m} (O_l^2(h-1)[k])} \quad (43)$$

Considering the output layer, one has:

$$\frac{\partial ne_l^2}{\partial O_x^1} = w_{l,x}(h) \quad (44)$$

where $w_{l,x}(h)$ is derived from the output-layer weight-adjustment algorithm given in Subsection 4.3.1. It is seen that the weight adjustment processes for both the hidden layer and the output layer must run step by step and one after another alternately.

As the neurons in the hidden layers are from various types of P-type, I-type, and D-type, consecutively, the calculation of term $\frac{\partial O_x^1}{\partial net_x^1}$ in (41) depends on the type of the selected neuron. Because the inputs of the hidden layer are the desired output (r_l) or the actual output of the system (y_l), term $\frac{\partial net_x^1}{\partial w_{a,b}}$ in (41) is also relevant to the selection of the actual output or the desired output. Hence, the gradient of the cost function should be expressed in six different types of expressions depending on a and b as listed below:

- **Type 1) $a = 1$ and $b = 2l - 1$**

In this type, the cost function's partial derivative path has a P-type neuron in the hidden layer, therefore:

$$\frac{\partial O_x^1}{\partial net_x^1} = 1 \quad (45)$$

The input of the hidden layer is the desired output (r_l), which gives:

$$\frac{\partial net_x^1}{\partial w_{a,b}} = r_l \quad (46)$$

Thus, one has:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \gamma_l(h) \times w_{l,3l-2}(h) \times r_l(h) \tag{47}$$

Substituting (41)-(47) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \gamma_l(h) w_{l,3l-2}(h) r_l(h)[m] \tag{48}$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{49}$$

- **Type 2) $a = 1$ and $b = 2l$**

In this type, the cost function's partial derivative path has a P-type neuron in the hidden layer, therefore:

$$\frac{\partial O_x^1}{\partial net_x^1} = 1 \tag{50}$$

The input of the hidden layer is the desired output (y_l), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = y_l(h) \tag{51}$$

Thus, one has:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \gamma_l(h) \times w_{l,3l-2}(h) \times y_l(h) \tag{52}$$

Substituting (41)-(44) and (50)-(52) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \gamma_l(h) w_{l,3l-2}(h) y_l(h)[m] \tag{53}$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \tag{54}$$

- **Type 3) $a = 2$ and $b = 2l - 1$**

In this type, the cost function's partial derivative path has an I-type neuron in the hidden layer, therefore:

$$O_x^1(k) = \sum_{k'=1}^{k} net_x^1(k') = \sum_{k'=1}^{k-1} net_x^1(k') + net_x^1(k) = O_x^1(k-1) + net_x^1(k) \tag{55}$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k) - O_x^1(k-1)}{net_x^1(k) - net_x^1(k-1)} = \frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)} \quad (56)$$

The input of the hidden layer is the desired output (r_l), which gives:

$$\frac{\partial net_x^1}{\partial w_{a,b}} = r_l \quad (57)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial w_{a,b}} = \gamma_l(h) \times w_{l,3l-1}(h) \times sign[\frac{net_x^1[m]}{net_x^1[m] - net_x^1[m-1]}] \times r_l(h)[m] \quad (58)$$

Substituting (41)-(44) and (55)-(57) into (40), the learning rule is derived as:

$$w_{a,b}(h+1) = w_{a,b}(h) - \mu_l \gamma_l(h) w_{l,3l-1}(h) sign[\frac{net_{3l-1}^1(h)[m]}{net_{3l-1}^1(h)[m] - net_{3l-1}^1(h)[m-1]}] r_l(h)[m] \quad (59)$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \quad (60)$$

If $[\frac{net_{3l-1}^1(h)[m]}{net_{3l-1}^1(h)[m] - net_{3l-1}^1(h)[m-1]}]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 4) $a = 2$ and $b = 2l$**

In this type, the cost function's partial derivative path has an I-type neuron in the hidden layer, therefore:

$$O_x^1(k) = \sum_{k'=1}^{k} net_x^1(k') = \sum_{k'=1}^{k-1} net_x^1(k') + net_x^1(k) = O_x^1(k-1) + net_x^1(k) \quad (61)$$

and:

$$\frac{\partial O_x^1}{\partial ne\ _x^1} = \frac{O_x^1(k) - O_x^1(k-1)}{net_x^1(k) - ne\ _x^1(k-1)} = \frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)} \quad (62)$$

The input of the hidden layer is the desired output (y_l), which gives:

$$\frac{\partial ne\ _x^1}{\partial w_{a,b}} = y_l(h) \quad (63)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)}{net_x^1(k)-net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \gamma_l(h) \times w_{l,3l-1}(h) \times sign[\frac{net_{3l-1}^1[m]}{net_{3l-1}^1[m]-net_{3l-1}^1[m-1]}] \times y_l(h)[m] \quad (64)$$

Substituting (41)-(44) and (61)-(63) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \gamma_l(h) w_{l,3l-1}(h) sign\left[\frac{net_{3l-1}^1[m]}{net_{3l-1}^1[m]-net_{3l-1}^1[m-1]}\right] y_l(h)[m] \quad (65)$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \quad (66)$$

If $[\frac{net_{3l-1}^1(h)[m]}{net_{3l-1}^1(h)[m]-net_{3l-1}^1(h)[m-1]}]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 5) $a = 3$ and $b = 2l - 1$**

In this type, the cost function's partial derivative path has a D-type neuron in the hidden layer, therefore:

$$O_x^1(k) = net_x^1(k) - net_x^1(k-1) \quad (67)$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k)-O_x^1(k-1)}{net_x^1(k)-net_x^1(k-1)} = \frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)} \quad (68)$$

The input of the hidden layer is the desired output (r_l), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = r_l \quad (69)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)-2net_x^1(k-1)+n\ \ _x^1(k-2)}{net_x^1(k)-net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \gamma_l(h) \times w_{l,3l}(h) \times sign[\frac{net_{3l}^1[m]-2net_{3l}^1[m-1]+net_{3l}^1[m-2]}{net_{3l}^1[m]-net_{3l}^1[m-1]}] \times r_l(h)[m] \quad (70)$$

Substituting (41)-(44) and (68)-(70) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \gamma_l(h) w_{l,3l}(h) sign[\frac{net_{3l}^1[m]-2net_{3l}^1[m-1]+net_{3l}^1[m-2]}{net_{3l}^1[m]-net_{3l}^1[m-1]}] r_l(h)[m] \quad (71)$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k])-\sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k])-\sum_{k=1}^{m}(O_l^2(h-1)[k])} \quad (72)$$

If $[\frac{net_{3l}^1[m]-2net_{3l}^1[m-1]+net_{3l}^1[m-2]}{net_{3l}^1[m]-net_{3l}^1[m-1]}]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 6) $a = 3$ and $b = 2l$**

In this type, the cost function's partial derivative path has a D-type neuron in the hidden layer, therefore:

$$O_x^1(k) = net_x^1(k) - net_x^1(k-1) \quad (73)$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k)-O_x^1(k-1)}{net_x^1(k)-ne\ _x^1(k-1)} = \frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)} \quad (74)$$

The input of the hidden layer is the desired output (y_l), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = y_l(h) \quad (75)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \gamma_l(h) \times w_{l,3l}(h) \times sign[\frac{net_{3l}^1[m]-2net_{3l}^1[m-1]+net_{3l}^1[m-2]}{net_{3l}^1[m]-net_{3l}^1[m-1]}] \times y_l(h)[m] \quad (76)$$

Substituting (41)-(44) and (74)-(76) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l \gamma_l(h) w_{l,3l}(h) sign[\frac{net_{3l}^1[m]-2net_{3l}^1[m-1]+net_{3l}^1[m-2]}{net_{3l}^1[m]-net_{3l}^1[m-1]}] y_l(h)[m] \quad (77)$$

where:

$$\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])} \quad (78)$$

If $[\frac{net_{3l}^1[m] - 2net_{3l}^1[m-1] + ne\ _{3l}^1[m-2]}{net_{3l}^1[m] - net_{3l}^1[m-1]}]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

The weight learning algorithms for both the output layer and the hidden layer of the neural networks of the MANNC2 are summarised in Table 4-1. From the summary shown in Table 4-1, it is noted that since the weights in the output layer are used by the hidden-layer learning algorithm at each step, the neural networks of the both layers must be trained simultaneously (i.e., one after another alternately). The fact that the calculations of the output-layer weights and the hidden-layer weights must be linked together indicates that the weights of the different layers of the neural networks of the MANNC2 are dependent on each other. This dependence reflects the inherent capacity of the MANNC2 in dealing with strong cross-couplings of the MIMO system of concern. It is also noted that the overall weight training procedure must initially start from the output layer, because at each learning step the calculation of the hidden-layer weights needs the values of the output-layer weights from the previous learning step.

Output layer	
$3l-2 \le x \le 3l$ $1 \le l \le n$	$w_{l,x}(h+1) = w_{l,x}(h) - \lambda_l[\gamma_l(h) \times \sum_{k=1}^{m}(O_x^1(h)[k])]$ $\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])}$
Hidden layer	
$a = 1$ $b = 2l-1$	$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l\gamma_l(h)w_{l,3l-2}(h)r_l(h)[m]$
$a = 1$ $b = 2l$	$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h) - \mu_l\gamma_l(h)w_{l,3l-2}(h)y_l(h)[m]$
$a = 2$ $b = 2l-1$	$\overline{w_{a,b}}(h+1)$ $= \overline{w_{a,b}}(h) - \mu_l\gamma_l(h)w_{l,3l-1}(h)sign[\frac{net_{3l-1}^1(h)[m]}{net_{3l-1}^1(h)[m] - net_{3l-1}^1(h)[m-1]}]r_l(h)[m]$
$a = 2$ $b = 2l$	$\overline{w_{a,b}}(h+1) = \overline{w_{a,b}}(h)$ $- \mu_l\gamma_l(h)w_{l,3l-1}(h)sign[\frac{net_{3l-1}^1[m]}{net_{3l-1}^1[m] - net_{3l-1}^1[m-1]}]y_l(h)[m]$
$a = 3$ $b = 2l-1$	$\overline{w_{a,b}}(h+1)$ $= \overline{w_{a,b}}(h) - \mu_l\gamma_l(h)w_{l,3l}(h)sign[\frac{net_{3l}^1[m] - 2net_{3l}^1[m-1] + net_{3l}^1[m-2]}{net_{3l}^1[m] - net_{3l}^1[m-1]}]r_l(h)[m]$
$a = 3$ $b = 2l$	$\overline{w_{a,b}}(h+1)$ $= \overline{w_{a,b}}(h) - \mu_l\gamma_l(h)w_{l,3l}(h)sign[\frac{net_{3l}^1[m] - 2net_{3l}^1[m-1] + net_{3l}^1[m-2]}{net_{3l}^1[m] - net_{3l}^1[m-1]}]y_l(h)[m]$ where, $1 \le l \le n$ $\gamma_l(h) = \sum_{k=1}^{m}(y_l(h)[k] - r_l(h)[k]) \times \frac{\sum_{k=1}^{m}(y_l(h)[k]) - \sum_{k=1}^{m}(y_l(h-1)[k])}{\sum_{k=1}^{m}(O_l^2(h)[k]) - \sum_{k=1}^{m}(O_l^2(h-1)[k])}$ • If the $sign[\ldots]$ is unknown due to zero denominator, the associated weight will remain unchanged for the next step.

Table 4-1. Weight adjustment learning algorithms

4.4 Stability Analysis

When a new control method is proposed, in order to achieve the desired control outcomes, the stability criteria of the resultant closed-loop system must be derived and any ensuing time-dependent conditions or constraints must be

checked online during the entire dynamic process of the closed-loop control. For an unconstrained control system, the system stability is satisfied if all the output responses are bounded for all bounded inputs. Because the eigenvalue analysis concept applies only to linear systems with model-based approaches, it cannot be used in nonlinear systems with model-free approaches. Therefore, for the proposed MANNC2, the Lyapunov stability analysis concept that is suitable for nonlinear systems, relies only on a system's inputs and outputs, and does not need to use the model of the system, must be used. Although, unlike linear systems, the stability of a nonlinear system does not need to be always global as the system can have multiple equilibrium points and limit cycles, the global asymptotic stability of a nonlinear MIMO system under the MACCN2 control is to be sought in this study in order to guarantee that the ensuing closed-loop system will never be locked in its local minimum during the entire dynamic control process.

As summarised in Table 4-1, there are two sets of free parameters, λ_l and μ_l ($1 \leq l \leq n$), for the output-layer and hidden-layer weight-learning algorithms of the MANNC2, respectively. These parameters are denoted as the learning rates of the respective dynamic layers of the MANNC2. They determine the weight-training speed of the respective layers and can dynamically change during the controller learning process. A set of constraints for these learning rates will be developed when the closed-loop system stability issue of the MANNC2 method is to be examined.

According to the Lyapunov global asymptotic stability theorem, for a defined function $V(x)$, if:

(i) $V(0) = 0$

(ii) For all $x \neq 0$, $V(x) > 0$ (i.e. V is positive definite)

(iii) For all $x \neq 0$, $\Delta V(x) < 0$

then every trajectory of $X(k+1) = X(k) + f(X(k))$ will converge to zero as $k \to \infty$ and the system will be globally asymptotically stable. Let the Lyapunov function

for each output of a nonlinear MIMO system under the MANNC2 control be the total cost function of that output, i.e.:

$$V_l(h) = (E_l(h))^2 \tag{79}$$

where $E_l(h)$ is the cost function related to the h^{th} step of the learning algorithm. One writes:

$$\Delta V_l(h) = (E_l(h) + \Delta E_l(h))^2 - (E_l(h))^2 = 2E_l(h)\Delta E_l(h) + (\Delta E_l(h))^2 \tag{80}$$

and:

$$\Delta E_l(h) \cong \Delta w_{l,x}(h) \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}} \tag{81}$$

For the output layer of the MANNC2, considering the Lyapunov function $V_l(h)$ in (79) and the expression $\Delta V_l(h)$ in (80), from the learning rule expressed in (30) for the output-layer weights, one has:

$$\Delta w_{l,x}(h) = -\frac{\lambda_l}{m} \sum_{k=1}^{m} \frac{\partial J}{\partial w_{l,x}} \tag{82}$$

and:

$$\frac{\partial J}{\partial w_{l,x}} = \frac{\partial J}{\partial E_l(h)} \frac{\partial E_l(h)}{\partial w_{l,x}} = \frac{\partial E_l(h)}{\partial w_{l,x}} \tag{83}$$

Substituting (83) into (82), one obtains:

$$\Delta w_{l,x}(h) = -\frac{\lambda_l}{m} \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}} \tag{84}$$

and:

$$\Delta E_l(h) \cong -\frac{\lambda_l}{m} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^2 \tag{85}$$

Therefore, (80) can be expressed as:

$$\Delta V(h) \cong -\frac{2\lambda_l E_l(h)}{m} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^2 + \frac{{\lambda_l}^2}{m^2} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right)^4 \tag{86}$$

Define:

$$H_l(h) = \min \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial w_{l,x}}\right) \cong \min_{3l-2 \le x \le 3l} \left(\sum_{k=1}^{m} \frac{\Delta E_l(h)[k]}{\Delta w_{l,x}(h)}\right) \tag{87}$$

The condition for $\Delta V(h) < 0$ yields the following constraint on the selection of learning rate λ_l defined in (30):

$$0 < \lambda_l < \frac{2mE_l(h)}{H_l(h)^2} \tag{88}$$

The above constraint for learning rate λ_l must be satisfied as a necessary condition of the MANNC2 control system at the h^{th} step. This constraint must be met at each training step in order for the MANNC2 to maintain its closed-loop system's global asymptotical stability during the entire learning process.

For the hidden layer of the MANNC2, considering the same Lyapunov function $V_l(h)$ in (79) and the same expression $\Delta V_l(h)$ in (80), one has the following estimation based on the hidden-layer weights described in (77):

$$\Delta E_l(h) \cong \Delta\overline{w_{a,b}}(h) \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}} \tag{89}$$

Thus, (40) can be expressed as:

$$\Delta\overline{w_{a,b}}(h) = -\frac{\mu_l}{m} \sum_{k=1}^{m} \frac{\partial J}{\partial \overline{w_{a,b}}} \tag{90}$$

where:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \frac{\partial J}{\partial E_l(h)} \frac{\partial E_l(h)}{\partial \overline{w_{a,b}}} = \frac{\partial E_l(h)}{\partial \overline{w_{a,b}}} \tag{91}$$

Substituting (91) into (90), one obtains:

$$\Delta\overline{w_{a,b}}(h) = -\frac{\mu_l}{m} \sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}} \tag{92}$$

Also, substituting (92) into (89), one writes:

$$\Delta E_l(h) \cong -\frac{\mu_l}{m} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}}\right)^2 \tag{93}$$

Hence, $\Delta V_l(h)$ can be expressed as:

$$\Delta V(h) \cong -\frac{2\mu_l E_l(h)}{m} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}}\right)^2 + \frac{{\mu_l}^2}{m^2} \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}}\right)^4 \tag{94}$$

Define:

$$\overline{H_l}(h) = \min \left(\sum_{k=1}^{m} \frac{\partial E_l(h)[k]}{\partial \overline{w_{a,b}}}\right) \cong \min_{2l-1 \le b \le 2l \,\&\, 1 \le a \le 3} \left(\sum_{k=1}^{m} \frac{\Delta E_l(h)[k]}{\Delta \overline{w_{a,b}}(h)}\right) \tag{95}$$

The condition for $\Delta V(h) < 0$ yields the following constraint on the selection of learning rate μ_l defined in (40):

$$0 < \mu_l < \frac{2mE_l(h)}{\overline{H_l}(h)^2} \tag{96}$$

The above constraint for learning rate μ_l must be satisfied as another necessary condition of the MANNC2 control system at the h^{th} step. This constraint must also be met at each training step in order for the MANNC2 to maintain its closed-loop system's global asymptotical stability during the entire learning process.

Collectively, conditions (92) and (100) must both be satisfied at each training step in order to guarantee the global asymptotical stability of the MANNC2 control system at the next training step. Simultaneous satisfaction of these two conditions is then considered as the sufficient condition for the global asymptotical stability of the MANNC2 control system, as summarised in Table 4-2.

Output layer constraint	$0 < \lambda_l < \frac{2mE_l(h)}{H_l(h)^2}$ where: $H_l(h) = \min_{3l-2 \le x \le 3l} \left(\sum_{k=1}^{m} \frac{\Delta E_l(h)[k]}{\Delta w_{l,x}(h)}\right)$
Hidden layer constraint	$0 < \mu_l < \frac{2mE_l(h)}{\overline{H_l}(h)^2}$ where: $\overline{H_l}(h) = \min_{2l-1 \le b \le 2l \,\&\, 1 \le a \le 3} \left(\sum_{k=1}^{m} \frac{\Delta E_l(h)[k]}{\Delta \overline{w_{a,b}}(h)}\right)$

Table 4-2. Stability criteria for hidden layer and output layer

It should be pointed out that it is necessary to check the stability conditions at each learning step, due to the fact that these conditions depend on the real-time values of the cost function ($E_l(h)$), the change of cost function ($\Delta E_l(h)$), and the change of weights ($\Delta w_{l,x}(h)$) at each training step. Both λ_l and μ_l can dynamically change to keep the closed-loop control system stable while the weights are trained by the learning algorithms. To avoid changing λ_l and μ_l dynamically, a small conservative constant value (e.g. 0.01 or less) can be selected for these parameters during the entire weight learning process. However, this arrangement would lead to a slower weight-adjustment rate and would require a greater number of training steps in order to achieve satisfactory results. In addition, it is noted from (92) and (100) that, by choosing a larger sampling number (m), the possibility of an instable system will become lower as there will

be a larger range for λ_l and μ_l. However, a larger sampling number (m) will lead to a slower weight-adjustment rate and, thus, a slower control speed of the resultant closed-loop system.

The stability criteria presented in Table 4-2 will be checked in a real-time manner together with the computation of the weight adjustment algorithms presented in Table 4-1 at each training step. This online simultaneous procedure is illustrated in Figure 4-4 as a flowchart. It can be seen from the flowchart that, when the stability criteria (92) and (100) are not met, the values of the learning rates are halved in order to enable the continuation of the learning process. It is acknowledged that while halving the learning rates may lead to a better control performance in terms of the error reductions, this arrangement may prolong the weight adjustment process and reduce the control speed of the system as a result.

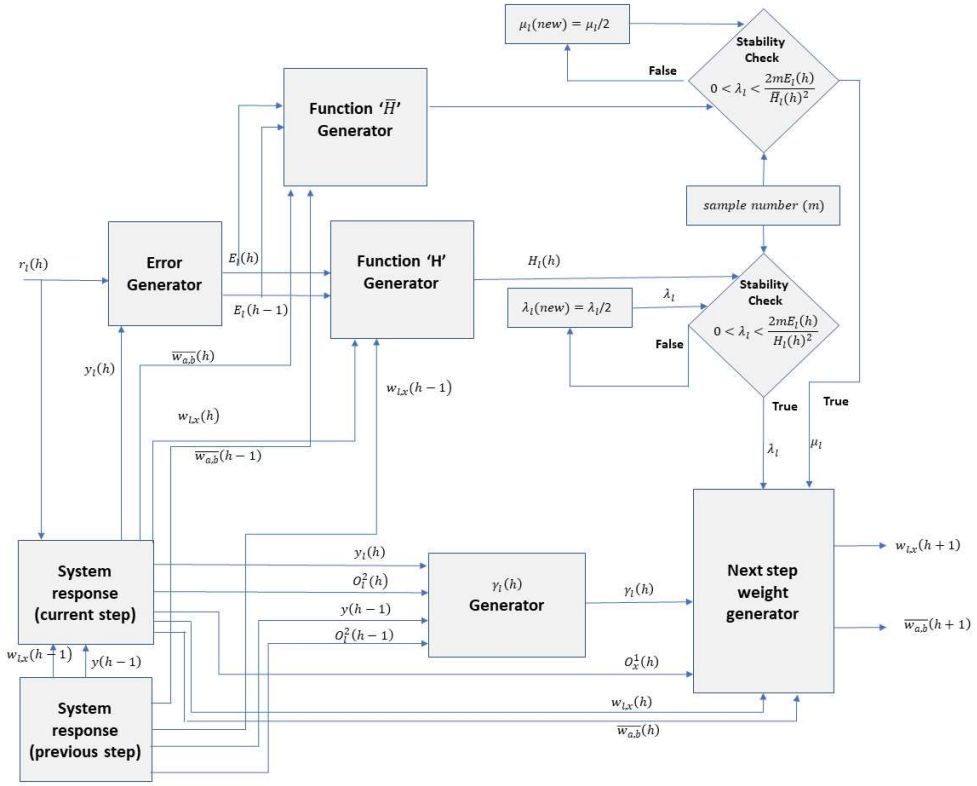


Figure 4-4. Real-time simultaneous stability criteria check and weight adjustment algorithm

4.5 Simulation results

Simulation studies using Matlab are carried out to evaluate the performances of the proposed MANNC2 in tracking set-points, reducing unwanted overshoots and undershoots, reducing settling times, and securing the global stability of a closed-loop system during the system's entire control process. The MANNC2 structure proposed in Section 4.2, the dynamic neural network algorithm developed in Section 4.3, and the stability conditions derived in Section 4.4 are used in the simulations. Two highly nonlinear MIMO cases are chosen for this study. In each case, the performances of the MANNC2 are compared with those of the MANNC introduced in [1]. As the performance superiority of the MANNC over its most competitive existing counterpart [25] has already been established in [1], the comparison of the MANNC2 with other existing counterparts, apart from the MANNC, is omitted here for brevity.

4.5.1 Case 1 – Application of MANNC2 on drum boiler MIMO system

In this case, an industrial drum-boiler plant that is a highly non-linear coupled two-input two-output system is selected. As illustrated in Figure 4-5, the system inputs are: heat flow rate and mass flow rate, and the system outputs are: pressure and level.

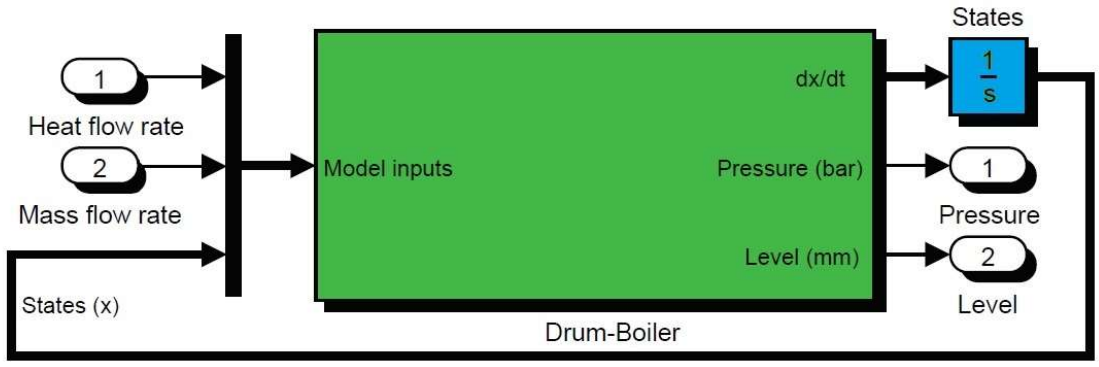


Figure 4-5. Two-input two-output drum-boiler plant

The state equations that describe the nonlinear relationships between the system inputs, the system outputs, and the state variables of the drum-boiler plant are:

$$\dot{x_1}(t) = -x_1(t) + u_1(t) \tag{97}$$

$$y_1(t) = x_1(t) + 2x_1^3(t) \tag{98}$$

$$x_2(t) = y_1(t) + u_2(t) \tag{99}$$

$$\ddot{y_2}(t) + 2\dot{y_2}(t) + y_2(t) = \dot{x_2}(t) + 2x_2(t) \tag{100}$$

where "dot" denotes the time derivative, $u_1(t)$ and $u_2(t)$ are the system inputs (heat flow rate and mass flow rate), $y_1(t)$ and $y_2(t)$ are the system outputs (pressure and level), and $x_1(t)$ and $x_2(t)$ are the state variables. Figure 4-6 represents the functional block diagram of the aforementioned nonlinear system, in which block "Fcn1" marks the highly nonlinear relationship between the system inputs and the system outputs. Because Input 1 affects both Output 1 and Output 2, the system is considered as a cross-coupled MIMO system that cannot be feasibly controlled by multiple cascaded SISO controllers such as the ANNC [13], nor by any other existing SISO or non-coupled MIMO counterparts.

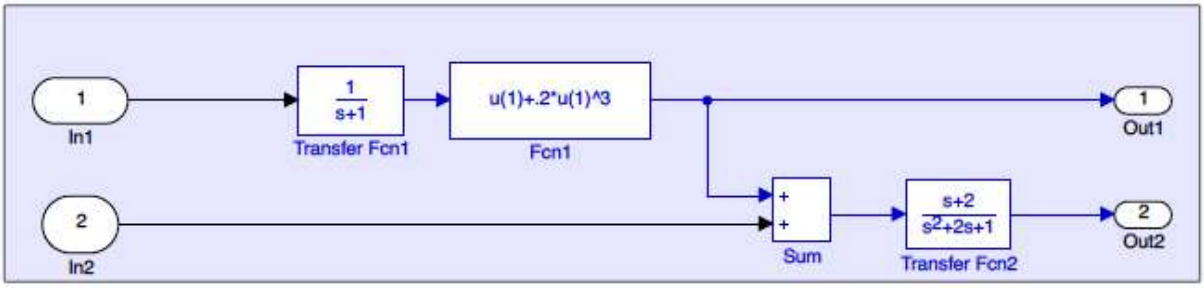


Figure 4-6. Functional block diagram of drum-boiler plant

Figure 4-7 demonstrates the implementation of the MANNC2 for the two-input two-output drum-boiler plant. All blue arrows in the hidden layer and the output layer represent the dynamic weights. The green links between the hidden layer and the output layer are added into the structure to enable the controller to handle any cross-couplings between the inputs and outputs of the system.

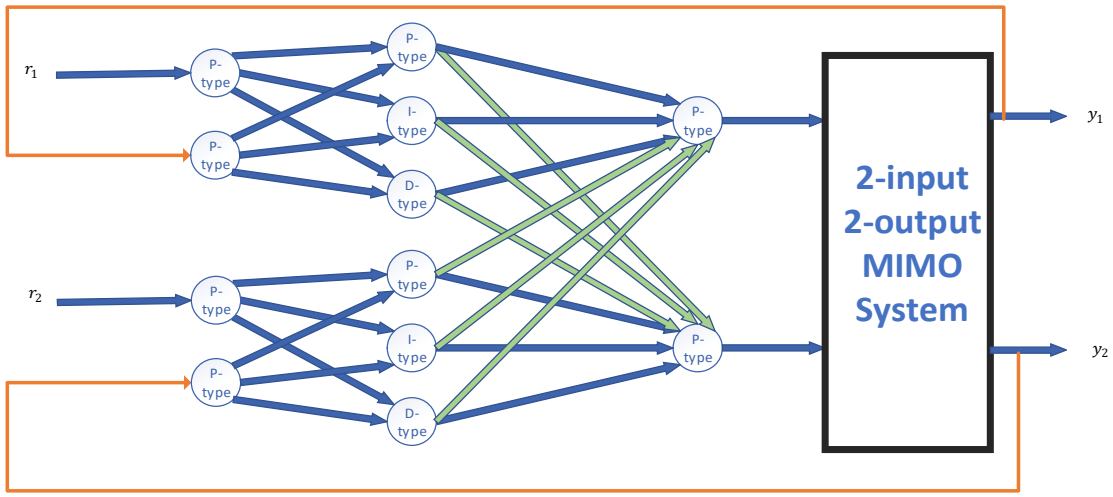


Figure 4-7. MANNC2 controlled drum-boiler plant

The following desired outputs are selected for the drum-boiler plant.

- $r_1(t) = 5u(t-1)$ where $u(t)$ is the standard unit step function.
- $r_2(t) = 0.2r(t-1)$ where $r(t)$ is the standard unit ramp function.

The control objective is for y_1 (Output 1) and y_2 (Output 2) to track, respectively, the set-points r_1 and r_2, with smaller overshoots and undershoots, shorter settling times, and minimum errors, and with a set of suitable and converged dynamic weights of controller.

To apply the MANNC2 to the underlying drum-boiler plant, the initial values of the weights are set to 1, the initial learning rates are set to 0.1 ($\lambda_1 = \lambda_2 = \mu_1 = \mu_2 = 0.1$), and the number of samples used in the learning algorithms is set to 40 ($m = 40$). After running, online and simultaneously, the learning algorithms presented in Table 4-1 and the stability criteria presented in Table 4-2 for 15 training steps, the converged values of the dynamic weights of both the output and hidden layers of the MANNC2 are achieved, as summarised in Table 4-3 and Table 4-4, respectively. These converged weights approve the convergences of the respective learning algorithms of the MANNC2 and, thus, the output convergences of the MANNC2 control system. During the simulation, it is observed that all learning rates of the MANNC2 are automatically reduced to 0.05 in the middle of the weight training process (at the 11th training step) to keep the control system stable for the next step. This designed-capability of the

MANNC2 enables the controller to work dynamically with highly nonlinear characteristics of the system.

$w_{l,x}$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$
$l = 1$	9.26	2.21	5.58	-2.32	-2.30	-2.53
$l = 2$	-1.43	1.58	1.42	4.13	4.01	0.41

Table 4-3. Converged MANNC2 output-layer weights' values

$\overline{w_{a,b}}$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
$a = 1$	0.780	-1.015	0.992	-1.000
$a = 2$	0.949	-1.014	1.004	-1.042
$a = 3$	0.993	-1.002	0.998	-1.002

Table 4-4. Converged MANNC2 hidden-layer weights' values

Figure 4-8 demonstrates that Output 1 with the MANNC2 can track the desired step output $r_1(t)$ satisfactorily with a tiny negligible overshoot of less than 2%. This can be considered as a perfect result for industrial applications. Overshoots during set-point changes are undesirable for many industrial control systems, especially when a set-point is ultimately close to the higher boundary of the output range (e.g. the risk of liquid overflow when filling a tank). Furthermore, a less than 8% of the undershoot of Output 1 with the MANNC2 is also shown Figure 4-8. This is a remarkable result, as such a low undershoot arising from switching or changing the set-point could not be achieved by conventional industrial controllers [32]. In Figure 4-9, Output 2 with the MANNC2 is seen to be able to track the desired ramp output $r_2(t)$ in less than 8 training steps. A satisfactory set-point tracking ability of the MANNC2 control system when dealing with a highly nonlinear plant is then evident.

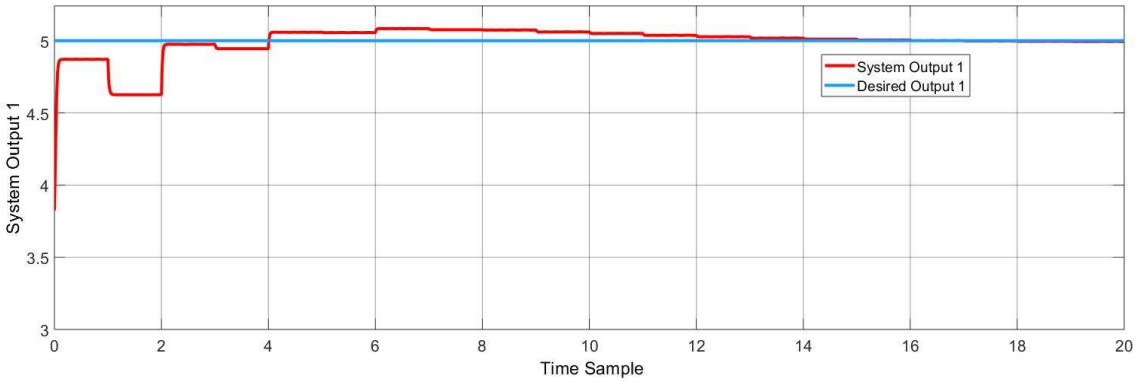


Figure 4-8. Desired Output 1 and Actual Output 1 with MANNC2

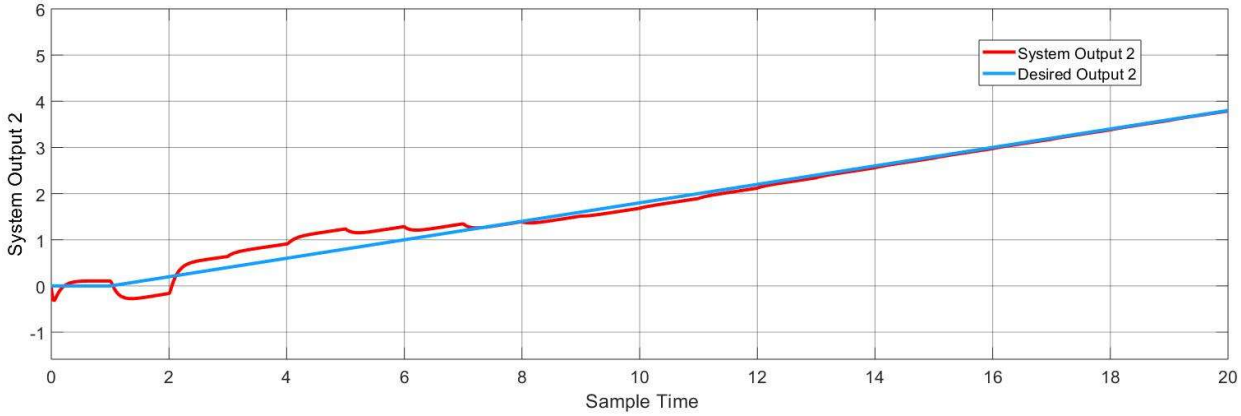


Figure 4-9. Desired Output 2 and Actual Output 2 with MANNC2

In order to have a meaningful comparison between the proposed MANNC and its predecessor, the MANNC introduced in [1], the results of both controllers for the same nonlinear MIMO system with the same desired outputs are presented in Figure 4-10 and Figure 4-11. Faster set-point tracking, lower undershoot, shorter settling time, and smaller accumulated error for both outputs of the MANNC2 control system, as compared to those of the MANNC control system, are evidently shown in these simulations.

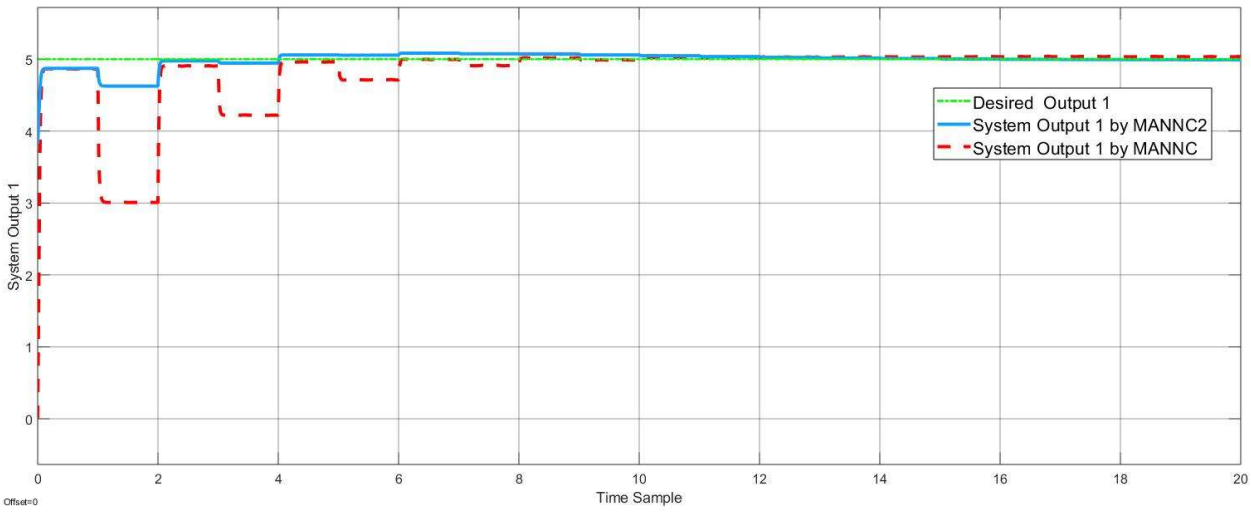


Figure 4-10. Desired Output 1 and Actual Output 1 with MANNC2 and MANNC

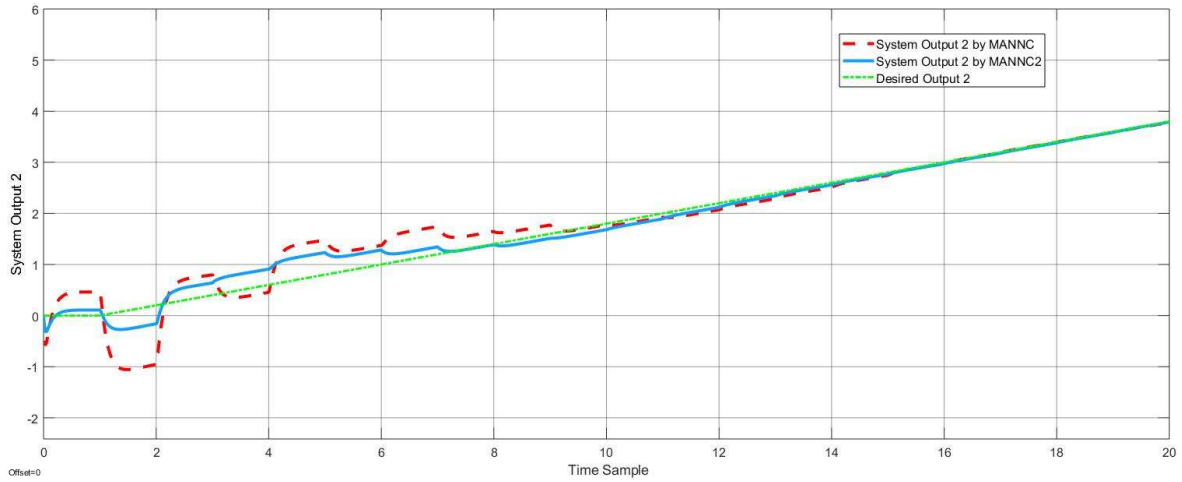


Figure 4-11. Desired Output 2 and Actual Output 2 with MANNC2 and MANNC

To effectively demonstrate the overall improved performance of the MANNC2 over that of the MANNC, the accumulated error versus the iteration number of the weight learning algorithm for each of the system outputs of the MANNC2 and the MANNC is simulated. The error is taken as the difference between the desired system output and the actual system output. A continuous reduction in the magnitude of the accumulated error while the iteration number of the respective weight learning algorithm is increasing would indicate the smooth convergence of the respective weight learning algorithm and, thus, the superiority of the respective control system. For a predefined level of the accumulated error, the smaller the required iteration number, the better the anticipated control system performance. Therefore, by comparing the continuous reductions in accumulated errors and the iteration numbers associated with the MANNC2 and MANNC, respectively, evidence as to which controller would be more effective could be established.

Figure 4-12 and Figure 4-13 represent the accumulated error versus the iteration number of the MANNC2 weight learning algorithm for Output 1 and Output 2 of the MANNC2 control system, respectively. Likewise, Figure 4-14 and Figure 4-15 represent the accumulated error versus the iteration number of the MANNC weight learning algorithm for Output 1 and Output 2 of the MANNC control system, respectively. From these four figures, it can be seen that

while the accumulated errors of the two outputs of the MANNC2 are in general decreased continuously as the iteration number of the MANNC2 weight learning algorithm increases, the accumulated error of Output 2 of the MANNC presents a huge increase during the increment of the iteration number of its weight learning algorithm. This fact demonstrates a weaker learning capability of the MANNC when facing a highly nonlinear system and, thus, a worse control effect of the MANNC control system. The MANNC2 is thus seen to be a better choice for drum-boiler plant. Furthermore, from the results shown in these four figures, for a pre-selected common allowable level of the accumulated error, say 2%, the required iteration number associated with the MANNC2 (15) is seen to be much smaller than that associated with the MANNC (35), as indicated in Table 4-5

	MANNC2	MANNC
Output 1	15	20
Output 2	11	32

Table 4-5. Iteration number for less than 2% accumulated error with MANNC and MANNC2

This fact further proves that the MANNC2 will provide a faster and more satisfactory control effect than the MANNC. The superiority of the MANNC2 over its predecessor, the MANNC, is thus established.

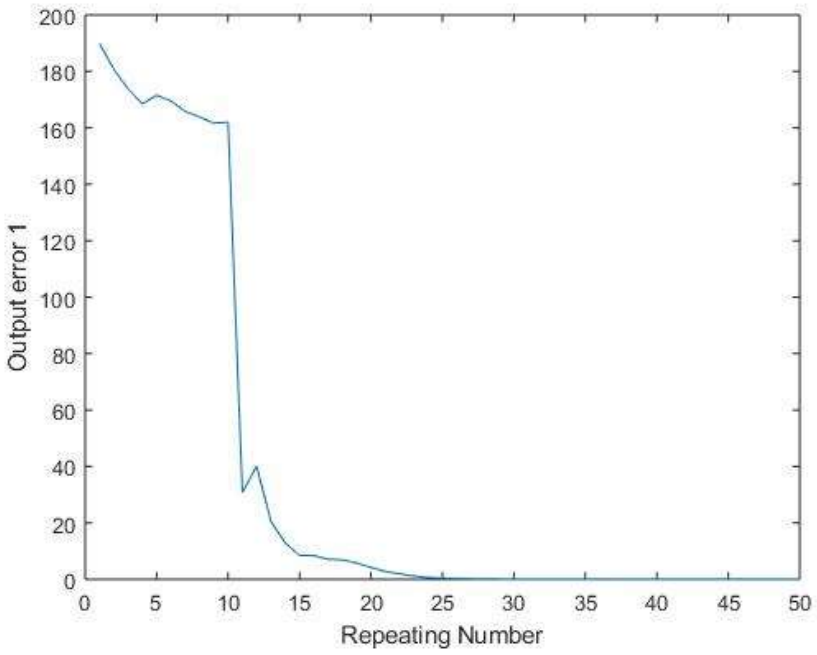


Figure 4-12. Accumulated error versus iteration number for Output 1 with MANNC2

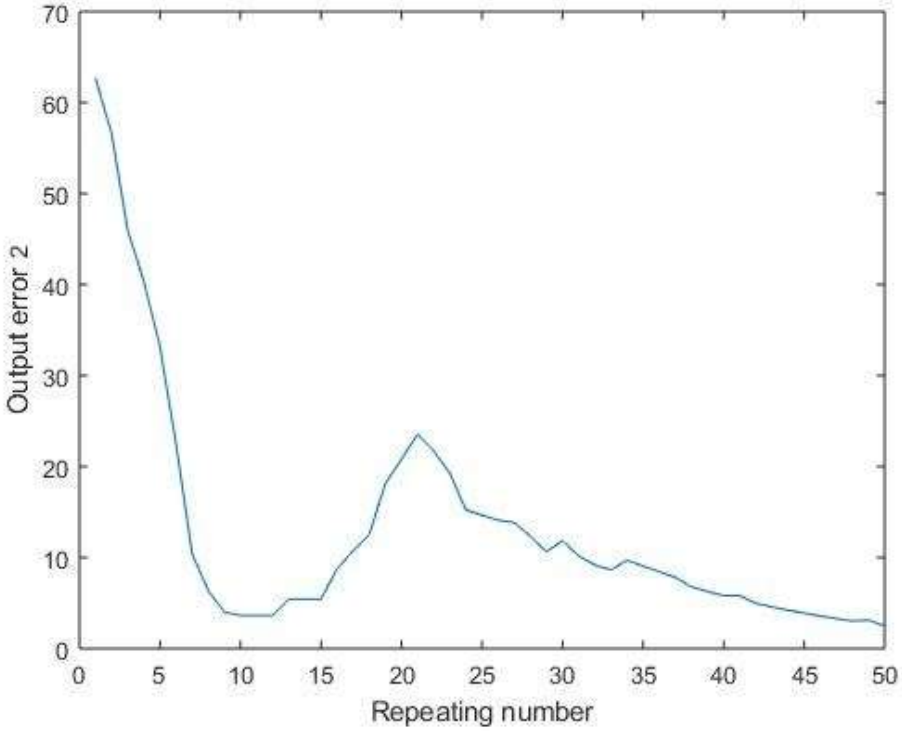


Figure 4-13. Accumulated error versus iteration number for Output 2 with MANNC2

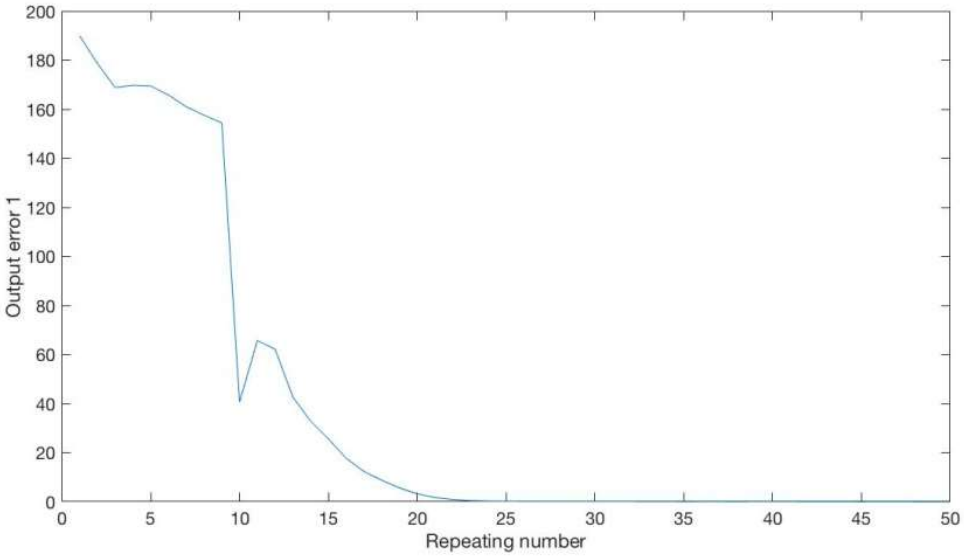


Figure 4-14. Accumulated error versus iteration number for Output 1 with MANNC

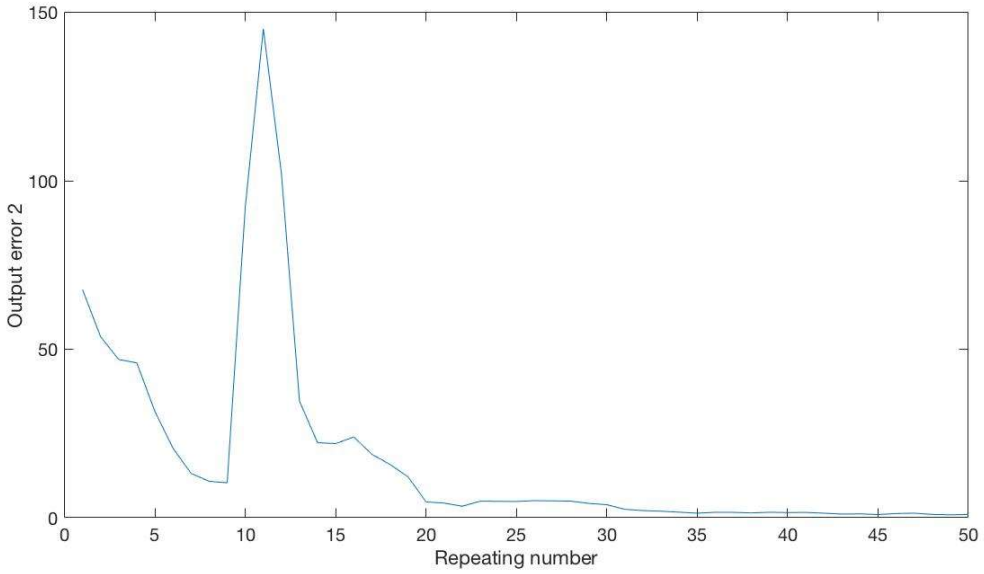


Figure 4-15. Accumulated error versus iteration number for Output 2 with MANNC

From Figure 4-12 and Figure 4-13, it is seen that an optimal training number of the learning algorithms can be automatically chosen by the MANNC2 control system by comparing a predefined allowable error with the system's actual error, so that the training process can be stopped at the 15th step and the weights become locked at their 15th-step values for the rest of the control process until a new change either in the model of the system or in any of the set-points of the system occurs. By doing so, the resultant closed-loop control system will be anticipated to have a less computation load (thus a less power consumption) but a slightly worsen control performance, as compared to a non-stop training case. Choosing an optimal training number depends on the specific application that the controller is designed for, and requires a trade-off between the computational demand and the control performance of the resultant closed-loop system of concern.

It should be point out that as the MANNC2 involves two-dynamic layers whereas the MANNC involves only one-dynamic layer, the time required by each training step of the MANNC2 will need to be physically longer than that of the MANNC (as illustrated in the example of the next case). Although this is a recognisable fact, it may become a constraint for systems with critical real-time requirements.

4.5.2 Case 2 – Application of MANNC2 on twin-tank level-control system

In this case, a twin-tank level-control system that is a complex and significant nonlinear coupled two-input two-output system is selected. The same system has been adopted as a typical example for evaluating the performances of many novel advanced control methods in the literature [33, 34]. The given physical system consists of two tanks, two pumps, two discharging valves, and one inter-connected valve between the two tanks. Each tank has its own pump for inflow of liquid, and the two tanks are connected by the inter-connected valve at their common base level. The system inputs are: the flow rate of Pump 1 (Q_{i1}) and the flow rate of Pump 2 (Q_{i2}), and the system outputs are: the liquid height of the left tank (H_1) and the liquid height of the right tank (H_2), as illustrated in Figure 4-16.

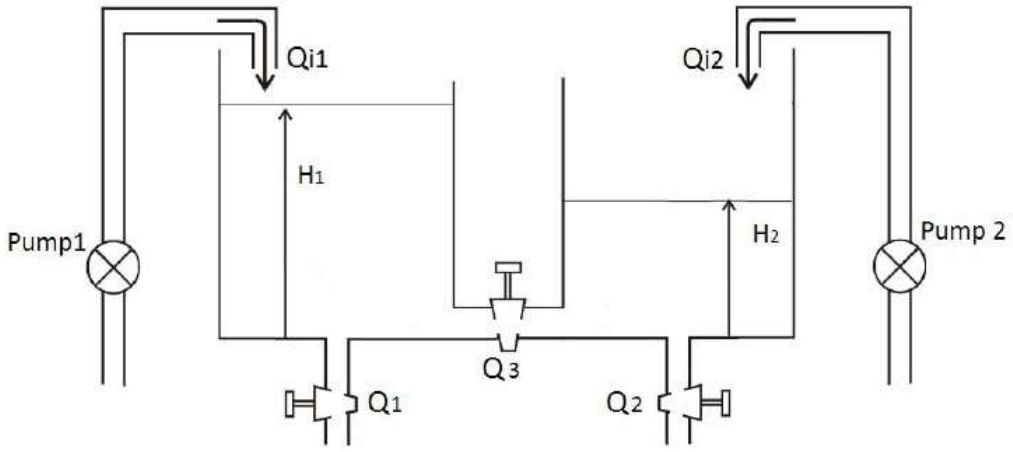


Figure 4-16. Twin-tank system

From Figure 4-16, according to the Bernoulli's equation for non-viscous and incompressible fluid, one writes:

$$Q_1 = s_1.a_0.\sqrt{2gH_1} \tag{101}$$

$$Q_2 = s_2.a_0.\sqrt{2gH_2} \tag{102}$$

$$Q_3 = s_3.a_1.\sqrt{2g(H_1 - H_2)} \tag{103}$$

where Q_1, Q_2, and Q_3 are the flow rates associated with the respective three valves; s_1, s_2, and s_3 are the cross-sectional areas of the channels that are connected to the three valves (Q_1, Q_2, and Q_3), respectively; a_0 is the discharge coefficient of Channel 1 (with Q_1) and Channel 2 (with Q_2); a_1 is the discharge

coefficient of Channel 3 (with Q_3); g is the acceleration due to gravity. By defining $\alpha_1 = s_1.a_0.\sqrt{2g}$, $\alpha_2 = s_2.a_0.\sqrt{2g}$, and $\alpha_3 = s_3.a_1.\sqrt{2g}$ as constant values, (105)-(107) are simplified as:

$$Q_1 = \alpha_1\sqrt{H_1} \tag{104}$$

$$Q_2 = \alpha_2\sqrt{H_2} \tag{105}$$

$$Q_3 = \alpha_3\sqrt{H_1 - H_2} \tag{106}$$

Thus, the nonlinear equations describing the dynamics of the twin-tank system are written as:

$$A_1\frac{dH_1}{dt} = Q_{i1} - Q_1 - Q_3 = Q_{i1} - \alpha_1\sqrt{H_1} - \alpha_3\sqrt{H_1 - H_2} \tag{107}$$

$$A_2\frac{dH_2}{dt} = Q_{i2} - Q_2 + Q_3 = Q_{i2} - \alpha_2\sqrt{H_2} + \alpha_3\sqrt{H_1 - H_2} \tag{108}$$

where A_1 and A_2 are the two tanks' cross-sectional areas, respectively. (111)-(112) show clearly the high-nonlinearity and coupling between the outputs of the given MIMO system. The selections of the parameters of the twin-tank system are given in Table 4-6.

It is assumed that the initial liquid heights of both tanks are at one meter and the desired liquid heights of both tanks are at seven meters. The control objective in this case is to achieve the desired heights in tanks by controlling the pumps' flow. In this experiment, according to the limited heights of the tanks in a real industrial application, large overshoots in liquid heights are counted as highly undesirable, and controllers capable of achieving satisfactory overshoot reduction and set-point tracking are considered as highly valuable.

Symbol	Description	Value
A_1, A_2	Tank cross-sectional area	45 m^2
s_1, s_2, s_3	Channels cross-sectional area	0.02 m^2
a_0,	Discharge coefficient of Channel 1 and Channel 2	0.5
a_1	Discharge coefficient of Channel 3	0.25
g	Acceleration due to gravity	9.81 m/s^2

Table 4-6. Parameters of given twin-tank system

To demonstrate a meaningful comparison between the proposed MANNC2 and the previous MANNC, both controllers are applied to the same nonlinear two-input two-output system, and the outputs of their respective resultant control systems are represented in Figure 4-17 and Figure 4-18, respectively. The initial learning rates of both controllers are set to 0.1 (i.e., $\lambda_1 = \lambda_2 = 0.1$ for MANNC and $\lambda_1 = \lambda_2 = \mu_1 = \mu_2 = 0.1$ for MANNC2), and the number of samples used in the learning algorithms of both controllers is set to 40 (i.e., $m = 40$). The learning algorithms of both controllers are shut down at the 35th training step, and the weights of the respective controllers are locked at their respective the 35th-step's values for the rest of the control processes. As can be seen in Figure 4-17 and Figure 4-18, faster set-point tracking, reduced undesirable overshoot, shorter settling time, and lower accumulated error can be achieved by the MANNC2 control system, as compared with those achieved by the MANNC control system. The superiority of the MANN2 over its counterpart, the MANNC, is again evidently confirmed in this case study. The detailed results of the application of the MANNC2 and the MANNC to the twin-tank system are represented in Table 4-7.

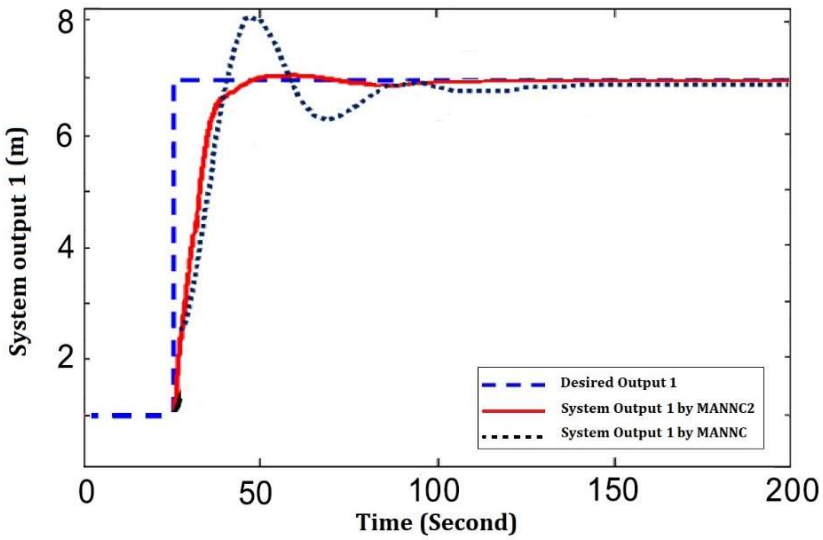


Figure 4-17. Liquid height of left tank controlled by MANNC2 and MANNC

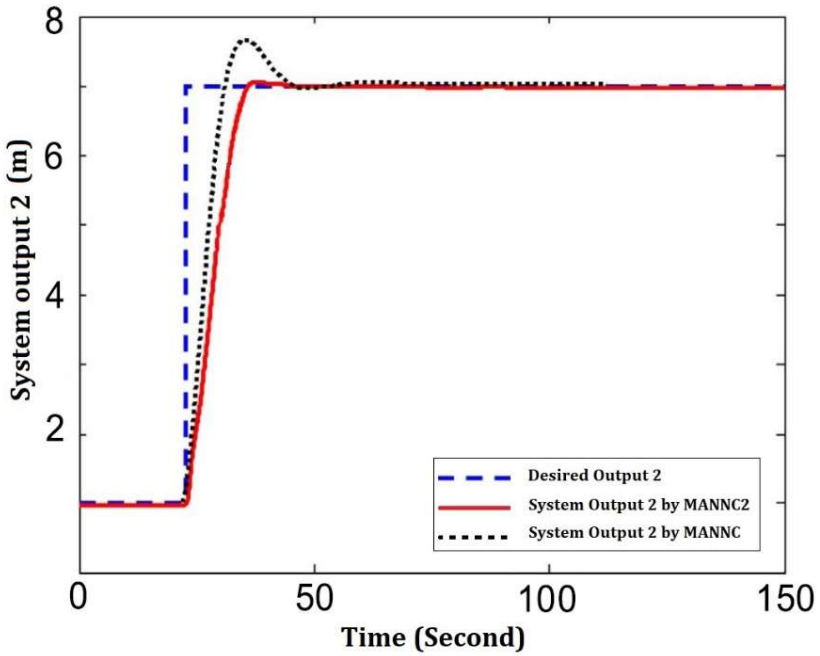


Figure 4-18. Liquid height of right tank controlled by MANNC2 and MANNC

Controller	Number of trainings	Time of training	Output 1 Overshoot	Output 2 Overshoot	Output 1 settling time	Output 2 settling time
MANNC	35	1.65 s	14.1%	11.7%	137 s	47 s
MANNC2	35	3.9 s	1.9%	0.7%	64s	47 s

Table 4-7. Comparison of MANNC2 and MANNC controlled twin-tank system

From Table 4-7, it is evident that although the overall required time of training for the MANNC2 is more than that of the MANNC, the overshoot of both outputs of the MANNC2 as well as the settling time of Output 1 of the MANNC2 are significantly less than those of the MANNC. The reason for a longer weight training time of the MANNC2 is justified, as the controller has two-dynamic layers in its structure and, thus, a greater number of weights to be trained as compared to the MANNC. This fact demonstrates that training an extra layer can lead to a better system performance but with the expense of spending longer time to compute the control action at each training step and, thus, requiring a higher computation load. Hence, choosing between the MANNC2 and the MANNC depends on the expected system performance and the required duration of the each-step's control action for a particular application.

The weight training time versus the variation of the number of samples between $5 \leq m \leq 40$ in both the MANNC2 and the MANNC is further investigated, and the comparative result is shown in Figure 4-19. As expected, due to the two-dynamic-layer structure, the weight training time of the MANNC2 is longer than that of the MANNC. In addition, it is observed that, unlike the MANNC, the required training time of the MANNC2 significantly depends on the number of samples (m) and increases with the increase in m exponentially. Therefore, when the MANNC2 is used, the number of samples must be selected carefully to avoid unwanted excessive training time. Nevertheless, to reduce the total training time of the MANNC2, the number of training steps of the MANNC2 can be optimally selected to be lesser than that of the MANNC, for achieving a similar accumulated error. For instance, in Case 1 presented in Subsection 4.5.1, a number of 15 can be selected as the optimal training number of the MANNC2 versus the number 35 that must be selected as the optimal training number for the MANNC in order to achieve a similar level of acceptable accumulated error. The overall advantage of having an adjustable hidden layer in addition to an adjustable output layer in the MANNC2 structure is clearly verified via both of the case studies.

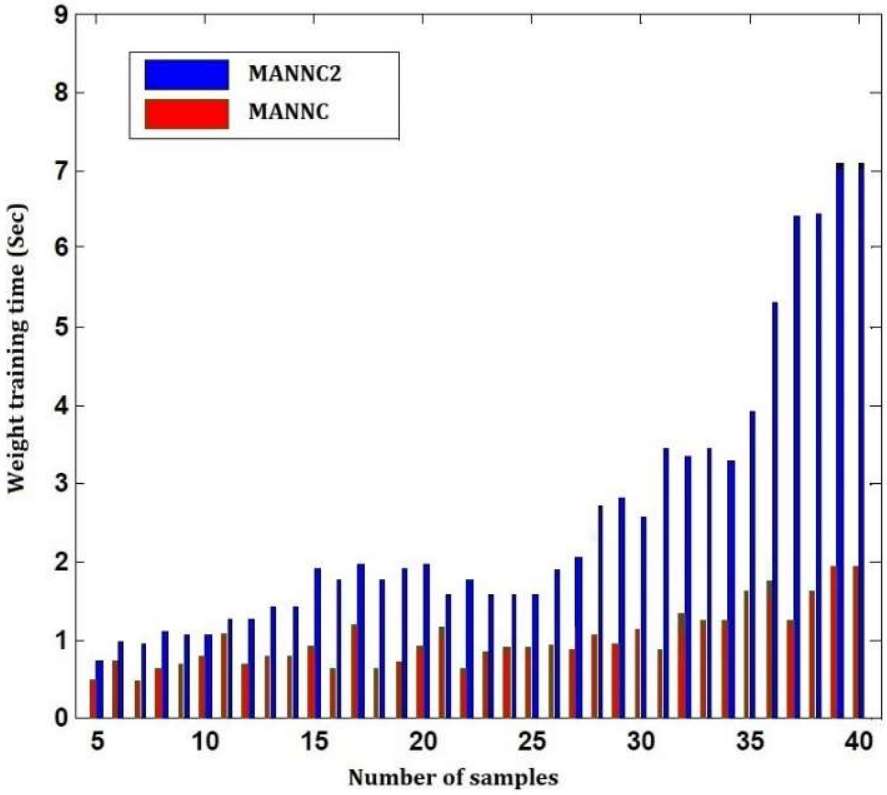


Figure 4-19. Weight training time vs number of samples taken for MANNC2 and MANNC

4.6 Conclusion

By the use of the powerful learning abilities inherent in a neural network strategy, the proposed MANNC2 can effectively control highly-nonlinear coupled square MIMO systems. Using a two-dynamic-layer structure in which each layer has its own auto-tune learning algorithm that is equipped with a dynamic accumulated-error back-propagation capability, the MANNC2 can successfully tune its weights online to achieve the desirable control outcomes while keeping the global stability of the resultant closed-loop system during the entire control process. The effectiveness of the proposed MANNC2 is validated via simulation studies. When compared to the MANNC introduced in [1], the MANNC2 is seen to provide a better control performance especially for systems with more complex nonlinear characteristics. By selecting an appropriate number of samples, the MANNC2 can be effectively used for a wide range of nonlinear square MIMO control systems found in industrial applications where minimum overshoot and undershoot in the system outputs are particularly desirable. Although the total training time required by the MANNC2 is more dependent on the number of

samples taken in the learning algorithms, by selecting a proper optimal training number, the required training steps of the MANNC2 can be much less than that of the MANNC, thus reducing the effective total training time of the MANNC2 significantly. Overall, it is demonstrated that the MANNC2 makes a more suitable candidate for industrial applications than its predecessor, the MANNC, as well as its relevant competitors in the literature.

References

[1] Arash Mehrafrooz, F.H., *Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems.* IEEE Transactions on Automatic Control (Submitted), 2019.

[2] Chen, M., S.S. Ge, and B.V.E. How, *Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities.* IEEE Transactions on Neural Networks, 2010. 21(5): p. 796-812.

[3] Dobrowiecki, T.P. and J. Schoukens, *Linear Approximation of Weakly Nonlinear MIMO Systems.* IEEE Transactions on Instrumentation and Measurement, 2007. 56(3): p. 887-894.

[4] Zhao, S., X. Gao, and C. Che. *Robust adaptive feedback linearization control for a class of MIMO uncertain nonlinear systems*. in *The 27th Chinese Control and Decision Conference (2015 CCDC)*. 2015.

[5] Fliess, M. and C. Join, *Model-free control.* International Journal of Control, 2013. 86(12): p. 2228-2252.

[6] Bu, X., et al., *Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information.* Asian Journal of Control, 2018. 20(2): p. 962-968.

[7] Madadi, E. and D. Söffker, *Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control.* 2015(57243): p. V001T03A004.

[8] Khadraoui, S., et al., *Adaptive Controller Design for Unknown Systems Using Measured Data.* Asian Journal of Control, 2016. 18(4): p. 1453-1466.

[9] Xie, C.H. and G.H. Yang, *Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults.* International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.

[10] Cui, X., et al., *Adaptive dynamic programming for tracking design of uncertain nonlinear systems with disturbances and input constraints.* International Journal of Adaptive Control and Signal Processing, 2017. 31(11): p. 1567-1583.

[11] Lafont, F., et al., *A model-free control strategy for an experimental greenhouse with an application to fault accommodation.* Computers and Electronics in Agriculture, 2015. 110: p. 139-149.

[12] Wang, Z., L. Liu, and H. Zhang, *Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.

[13] Mehrafrooz, A., He, F. *Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems*. in *IEEE ICIA 2018*. 2018. Wuyi Mountain, Fujian, China.

[14] Ge, S.S., C. Yang, and T.H. Lee, *Adaptive Predictive Control Using Neural Network for a Class of Pure-Feedback Systems in Discrete Time.* IEEE Transactions on Neural Networks, 2008. 19(9): p. 1599-1614.

[15] Park, J., S. Kim, and C. Moon, *Adaptive Neural Control for Strict-Feedback Nonlinear Systems Without Backstepping.* IEEE Transactions on Neural Networks, 2009. 20(7): p. 1204-1209.

[16] Nodland, D., H. Zargarzadeh, and S. Jagannathan, *Neural Network-Based Optimal Adaptive Output Feedback Control of a Helicopter UAV.* IEEE Transactions on Neural Networks and Learning Systems, 2013. 24(7): p. 1061-1073.

[17] Ahmed, J. and E.M. Alkhalifa. *Efficient single layer handwritten digit recognition through an optimizing algorithm*. in *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*. 2002.

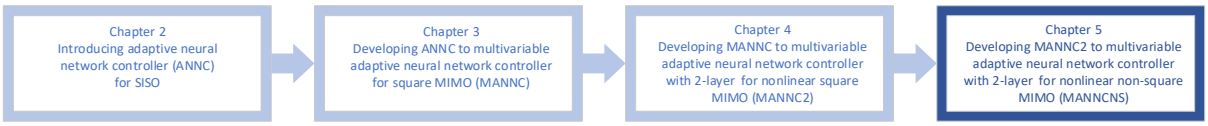
[18] Zhao, Y., B. Deng, and Z. Wang. *Analysis and study of perceptron to solve XOR problem*. in *The 2nd International Workshop on Autonomous Decentralized System, 2002.* 2002.

[19] Zhang, Y., X. Wang, and E.G. Friedman, *Memristor-Based Circuit Design for Multilayer Neural Networks.* IEEE Transactions on Circuits and Systems I: Regular Papers, 2018. 65(2): p. 677-686.
[20] Huailin, S., G. Xiucai, and S. Hua. *PID neural networks in multivariable systems*. in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*. 2002.
[21] Yu, Y., Y. Huang, and B. Zeng. *A PID neural network controller*. in *Proceedings of the International Joint Conference on Neural Networks, 2003.* 2003.
[22] Lin, L. and X. Peng. *A PID neural network control for permanent magnet synchronous motor servo system*. in *2010 5th International Conference on Computer Science & Education*. 2010.
[23] Changhua, L. and S. Hua. *Design of electric loading system in flight simulator based on PIDNN*. in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. 2011.
[24] Ge, S.L., et al. *Design of PIDNN adaptive disturbance rejection decoupling controller*. in *2017 Chinese Automation Congress (CAC)*. 2017.
[25] Yong, Z., Z. Hai-bo, and L. Tian-qi. *PIDNN decoupling control of boiler combustion system based on MCS*. in *2017 29th Chinese Control And Decision Conference (CCDC)*. 2017.
[26] Dingguo, L.S.W.Z.W.G.W., *Pid neural network sliding-mode controller design for three level based vsc-hvdc converter of offshore wind power.* Proceedings of the CSEE, 2012. 32: p. 20-28.
[27] Hernandez-Alvarado, R., et al. *Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles*. in *OCEANS 2016 MTS/IEEE Monterey*. 2016.
[28] Jing, X. and L. Cheng, *An Optimal PID Control Algorithm for Training Feedforward Neural Networks.* IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.
[29] Scott, G.M., J.W. Shavlik, and W.H. Ray, *Refining PID Controllers Using Neural Networks.* Neural Computation, 1992. 4(5): p. 746-757.
[30] Bahri, N., et al. *Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems*. in *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*. 2014.
[31] Saerens, M. and A. Soquet, *Neural controller based on back-propagation algorithm.* IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.
[32] Phillips, S.F. and D.E. Seborg. *Conditions that Guarantee No Overshoot for Linear Systems*. in *1987 American Control Conference*. 1987.
[33] Hou, Z., S. Liu, and T. Tian, *Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems.* IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.
[34] Ribeiro, J.M.S., et al. *Comparison of PID controller tuning methods: analytical/classical techniques versus optimization algorithms*. in *2017 18th International Carpathian Control Conference (ICCC)*. 2017.

CHAPTER 5: INTRODUCING UNIVERSAL MODEL-FREE MULTIVARIABLE ADAPTIVE NEURAL NETWORK CONTROLLER FOR NON-SQUARE MULTIVARIABLE SYSTEMS

Aim

In this chapter, an introduction of a Multivariable Adaptive Neural Networks Controller for Non-Square MIMO systems (MANNCNS) is represented. Using the controllers designed in previous chapters namely ANNC, MANNC, and MANNC2, this controller is designed in such a way to represent the ultimate product of this thesis as a universal controller to provide a framework to control MIMO coupled systems with unknown models, highly non-linear characteristics and any different number of inputs and outputs as per diagram below:



Description

To achieve this aim, firstly, the neural network structure of the multivariable adaptive controller and matrix representation of the closed loop control system for a general non-square MIMO system is represented. The neural network topology of the MANNCNS is designed via the modification of the square neural network structure of the MANNC2 introduced in Chapter 4. Subsequently, a fully modified learning algorithm for a 2-layer neural network structure and the Lyapunov stability criteria for the non-square structure of the controlled system are developed. The Lyapunov stability criteria are being checked simultaneously with running the learning algorithm to guarantee that the neural network control system does not fall in its local minimum. In order to validate the performance of the MANNCNS, this controller is applied to a non-square MIMO case of a distillation column system (with three manipulated variables and two controlled variables) and also a non-square MIMO case of DC motor (with two manipulated variables and three controlled variables). Subsequently, the

designed controller is tested and compared with the best existing counterparts. The above-mentioned contents are described in a paper:

- Arash Mehrafrooz and Fangpo He, "Introducing a Universal Model-free Multivariable Adaptive Neural Network Controller for Non-Square MIMO Systems" submitted to "IEEE Transactions on Automation Science and Engineering".

Results

In this chapter, the MANNC2 introduced in the Chapter 4 - which was applicable on "square" nonlinear model-free MIMO systems - is modified to a universal controller for "non-square" non-linear model-free MIMO systems. The original contribution in this study is the development of a novel advanced control method including the new neural network structure of the controller and associated learning algorithm to be applied to general non-square coupled multivariable systems with unknown models. The obtained results clarify the appropriate set-point tracking, error reduction, and unexpected disturbance

cancelation of the controller with its powerful auto-tuning capability in industrial applications.

Conclusion

Overall, this chapter shows that with the powerful learning abilities inherent in a neural network strategy, the proposed MANNCNS is capable of controlling model-free coupled non-square MIMO systems with non-linear properties to achieve the desirable control outcomes. The MANNCNS can be effectively used as a general novel solution for industrial applications as a practical framework. Future proposed studies will be outlined in Chapter 6 of this thesis.

Chapter 5

Introducing a Universal Model-free Multivariable Adaptive Neural Network Controller for Non-Square MIMO Systems

Arash Mehrafrooz[1, a] ***and*** *Fangpo He*[1, b]

[1]*Advanced Control Systems Research Group, College of Science and Engineering, Flinders University, Australia*

[a]<*arash.mehrafrooz@flinders.edu.au*>, [b]<*fangpo.he@flinders.edu.au*>

Abstract **- In this paper, a universal model-free Multivariable Adaptive Neural Network Controller for Non-Square systems (MANNCNS) is introduced for controlling black-box nonlinear non-square couple Multiple-Input Multiple-Output (MIMO) systems. The controller follows a model-free adaptive learning algorithm to train weights in two dynamic layers of its neural networks automatically. By having the history of the system's inputs and outputs, the controller is able to adjust itself with the new conditions such as changes in the desired outputs, structural uncertainties in the model of the system, and unwanted disturbances. Two adjustable layers in the neural network structure have been considered to deal with nonlinear behavior of the non-square multivariable system of concern. The Lyapunov stability criteria which are defined based on the data from input signals and output signals can guarantee stability of the closed loop control system during the entire learning process. The simulation results demonstrate the proper control performance as well as powerful disturbance rejection of the controller for a model of distillation column with three manipulated variables (inputs) and two controlled variables (outputs). Additionally, the performance of the designed controller is checked for MIMO systems when the number of outputs is more than the number of inputs via applying to a highly non-linear DC motor model. To demonstrate the highly improved performance of MANNCNS, the simulation results are compared to the best recent neuro-fuzzy counterpart, namely RHONN.**

Key Words – auto tuning, disturbance rejection, learning algorithm, Lyapunov stability, neural network control, non-square MIMO.

5.1 Introduction

Control of non-square MIMO systems that is frequently required in many industrial applications, has always been a challenging problem in the field of multivariable control systems due to asymmetrical structure of their system's

model [1, 2]. The control problem becomes more challenging where strong cross-couplings exist between inputs and outputs of the non-square MIMO system, as interactions among control loops cause controlled variables (system's outputs) to be affected by multiple manipulated variables (system's inputs). In the literature, there are many control problems defined for non-square MIMO industrial applications, to name a few: mixing-tank process with 2 inputs and 3 outputs (e.g. [3]), distillation column with 3 inputs and 2 outputs (e.g. [2]), Shell control problem with 7 inputs and 5 outputs (e.g. [4]), air path scheme of a turbo charged diesel engine with 3 input and 2 outputs (e.g. [5]), crude distillation unit with 5 inputs and 4 outputs (e.g. [6]), and hot oil fractionators with 4 inputs and 2 outputs (e.g. [7]). This diversity of applications with different number of inputs and number of outputs demonstrates the importance of having a universal control solution with dimensional flexibility to apply to coupled non-square MIMO systems.

In model-based control solutions for non-square MIMO systems, there are many drawbacks with regard to the difficulty of inverting non-square system model's matrices. To cope with this difficulty, many of the modern strategies for controlling non-square MIMO systems are designed based on square control methods. This is to take the advantage of system's invertible matrix for simplicity in mathematical computations. For this purpose, in model-based approaches, adding / removing the appropriate number of rows or columns to / from the system's matrices is being used. By doing so, the non-square system's model is converted to a square one via squaring up or squaring down the original non-square system's model. Once the non-square MIMO system becomes square, then a control solution can be developed for the new square MIMO system's model. [8] has compared square and non-square structures of different case studies and found the square system was much more sensitive to modelling errors. Squaring up/down method is also used in some model-free approaches by adding/removing inputs or outputs to achieve a symmetrical input-output structure for the system of concern. Despite the fact that this approach has been successful for many applications, it has practical shortcomings in industrial applications as follows: (i) Adding unnecessary outputs to be measured can be

costly. (ii) Deleting inputs leaves fewer variables to be set to achieve the desired control outcome. (iii) The amount of feedback information available to the controller can be reduced by decreasing the number of measured outputs. (iv) adding new manipulated inputs can incur an unnecessary cost. Considering the mentioned problems in squaring up or squaring down non-square MIMO systems, it has been recently demonstrated that there are many benefits in synthesizing a controller for the original non-square system over using the square models in many industrial applications.

In model-based control methods that use the non-square model of the MIMO systems, the procedure for finding the exact model of the system is a time-consuming and costly process due to unmodelled nonlinearities, structural uncertainties, and time-varying natures. Also, application of a non-square model-based control methods is limited to particular classes of non-square MIMO systems. As a meaningful example, in [5] a static decoupling method based on the structure of internal model control is proposed that used a Smith compensator for non-square multivariable systems. The simulation results demonstrated that although this method can achieve a reasonable control performance, this type of static decoupling only guarantees complete decoupling for low frequency responses, and does not address the high frequency responses happening in many industrial applications [9].

Due to the above-mentioned deficiencies related to model-based methods, over the past few years, model-free control methods play an important role in the control of non-square MIMO industrial systems [10, 11]. MFC method which was firstly proposed for controlling single-input single-output (SISO) systems in [12], was modified to control a non-square MIMO dynamic system via model approximation of nonlinear dynamic system in [13]. MFC method for non-square MIMO was applied to a linear time-invariant system and the optimal control equation included a term that was derived from the solution of a linear quadratic regulator (LQR) problem. In the mentioned method, optimal control problem was solved off-line. Although, this off-line control method is applicable for model-free non-real time (NRT) applications, it did not address the situations that an online and real-time control action is required.

Nowadays, neural networks have been proven to be suitable for implementation of important machine learning algorithms for model-free control approaches [14]. By using parallel structure and adding adaptive features to neural network scheme, adaptive neural networks have been seen in real-time multivariable control systems. The multi-layer neural networks have been used in controlling coupled nonlinear MIMO systems for their abilities in nonlinear approximation. Due to the capacity of having non-square neural network structures, non-square neural networks can be chosen to be used in controlling non-square MIMO systems. In [15], a direct adaptive control scheme is introduced for regulation of nonlinear unknown MIMO plants. The control method is based on a new Neuro-Fuzzy Dynamical Systems method, that used the concept of Fuzzy Dynamical Systems (FDS) operating in conjunction with High Order Neural Network Functions (HONNF). The unknown system was expressed by a F-RHONN introduced in [16] and contains a number of unknown constant-value parameters known as synaptic weights. The control signal is constructed to be valid for both square and non-square systems. This method is used to compare and validate the performance of MANNCNS in simulation case in this paper.

Considering the above-mentioned deficiencies associated with the existing model-based methods, off-line model-free methods and neural network-based methods for controlling non-square MIMO systems, it is noted that there is a need to develop a general online model-free control method for black-box non-square MIMO systems. In this study, by using the parallel computation capability and the superior flexibility in the topology of neural networks, the neural network controller introduced in [17] is modified to a Multivariable Adaptive Neural Network Controller for Non-Square MIMO systems called MANNCNS. The proposed controller potentially possesses several key benefits over MANNC and MANNC2 introduced in [17, 18] and the existing counterparts as follows:

- By knowing the fact that in many applications, it is not suitable to convert a non-square multivariable system to a square multivariable one by adding or removing extra inputs or outputs, MANNCNS provides a

general model-free control framework for controlling original non-square MIMO systems with various number of inputs and number of outputs.

- By using flexible and cross-coupled properties and non-square structure of multi-layer neural networks, MANNCNS provides a universal controller solution that is capable of controlling non-square coupled MIMO systems with unknown models and therefore can cope with nonlinearities, structural uncertainties, and environmental disturbances.
- The new online dynamic neural network learning algorithm used in the proposed method significantly modifies the previous learning methods for square MIMO systems introduces by the authors in [17, 18] to be applied to non-square MIMO systems.
- By applying an accumulated gradient in the error back-propagation algorithm, the proposed method uses the history of the system's outputs together with the current weights to find the weights for the next step and significantly improves current model-free methods for controlling a non-square MIMO in a real-time manner.
- New stability conditions are established for the non-square closed loop control system, and yield to constraints to be checked during the weight learning process to guarantee the stability of the control system at entire time of the control process.
- The proposed method enables the provision of the weights' adjustment convergence which is an important indicator for the implementation of the controller in industrial applications to identify the optimal number of weight trainings automatically during the learning process.

To reveal the proposed MANNCNS, the rest of the paper is organized as follows: In section 5.2 the structure and matrix representation of the new neural network-based adaptive controller is demonstrated. In section 5.3, the learning algorithm of the neural networks for two layers using error back-propagation algorithm is illustrated. The controlled system's stability criteria are investigated in section 5.4. Section 5.5 contains the simulation results where the proposed method is applied to two case-studies of non-square MIMO system to validate the performance of the controller on both types of non-square MIMO

systems and compared with its utmost counterpart. The paper's conclusion is given in Section 5.6.

5.2 Multivariable Adaptive Neural Networks Controller for Non-Square Multivariable Systems (MANNCNS)

5.2.1 Closed-loop Structure of MANNCNS

According to the proposed structure of P-type, I-type, and D-type neurons introduced by the authors in [17, 18], the output of the neurons in the discrete form can be expressed respectively as:

$$o_P(k) = \begin{cases} 1 & net_P(k) > 1 \\ net_P(k) & -1 \le net_P(k) \le 1 \\ -1 & net_P(k) < -1 \end{cases} \quad (1)$$

$$o_I(k) = \begin{cases} 1 & net_I(k) > 1 \\ o_I(k-1) + net_I(k) & -1 \le net_I(k) \le 1 \\ -1 & net_I(k) < -1 \end{cases} \quad (2)$$

$$o_D(k) = \begin{cases} 1 & net_D(k) > 1 \\ net_D(k) - net_D(k-1) & -1 \le net_D(k) \le 1 \\ -1 & net_D(k) < -1 \end{cases} \quad (3)$$

where $o_X(k)$ and $net_X(k)$ are the X-type neuron's output and the X-type neuron's sum of inputs at the k^{th} sampling time, respectively. Considering the structure of ANNC, MANNC, and MANNC2 ,respectively, proposed in [19], [18], and [17], a new Multivariable Adaptive Neural Networks Controller for Non-Square MIMO systems (MANNCNS) illustrated in Figure 5-1 is developed as a closed-loop controller to be applied to coupled non-square ($p \times q$) MIMO systems, where p and q are the number of the system inputs and number of the system outputs, respectively. In this structure, the sum of the weighted desired outputs and weighted actual outputs are produced in the first dynamic layer of the controller and, subsequently, propagated to the second dynamic layer of the controller. Structurally, the MANNCNS contains three layers called the input layer, the hidden layer, and the output layer; There are $2q$ P-type neurons within the input layer. In the hidden layer, there are $3q$ neurons including P-type, I-type, and D-type neurons in clusters of three, respectively. In the output layer, there are p P-type neurons to supply the outputs of the MANNCNS. The outputs of the output

layer enter to the multivariable system as inputs. There are $6q$ adjustable weights in the hidden layer which are multiplied by the input layer's outputs to make the hidden-layer inputs. Additionally, there are $3pq$ adjustable weights in the output layer associated with the hidden-layer neurons. Hidden-layer weights determine the impact of each neuron in the hidden layer to generate the multivariable system's inputs.

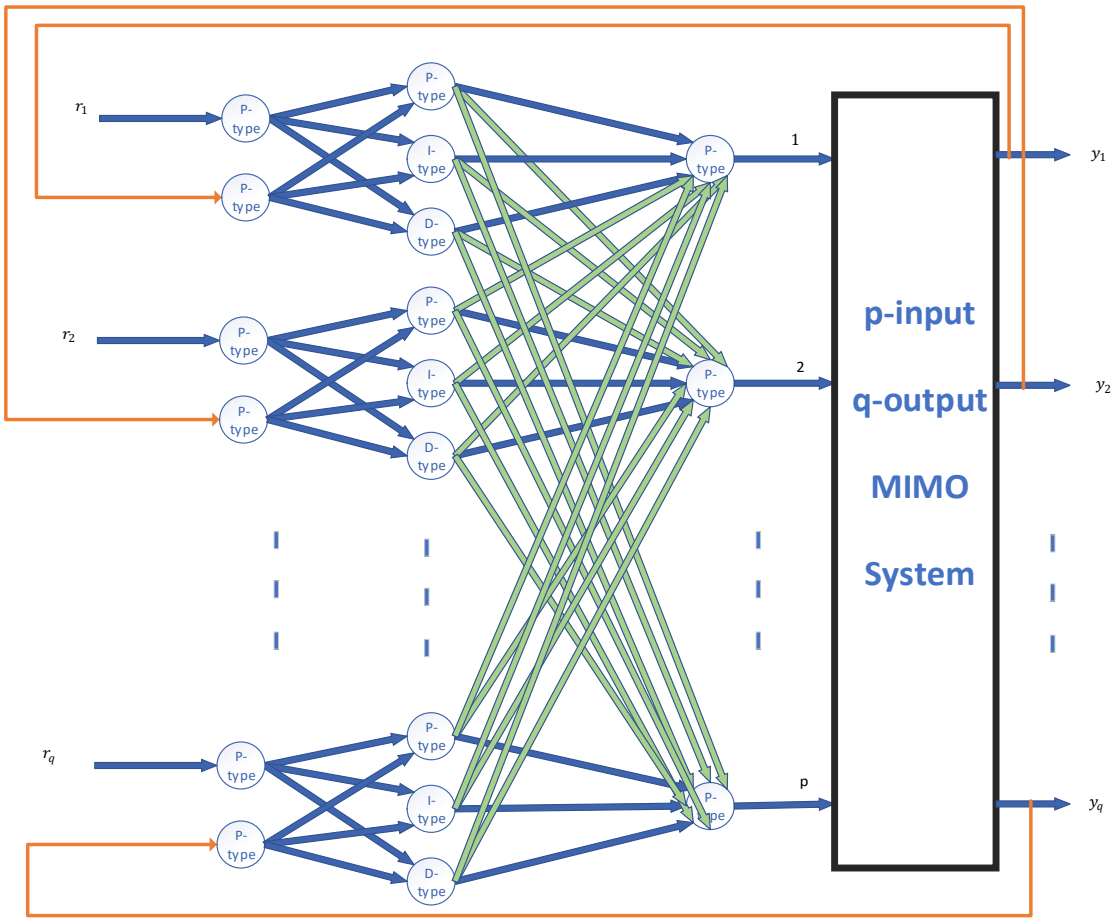


Figure 5-1. MANNCNS control system structure

5.2.2 Structure of Sub-MANNCNS (S-MANNCNS)

Using the concept of the sub-controllers introduced in [17] for the MANNC2, the proposed MANNCNS's structure illustrated in Figure 5-1 is further decomposed into pq sub-controllers each named as a Sub-MANNCNS (S-MANNCNS) as shown in Figure 5-2. The functionality of the sub-controllers is to enable the neural-network controller to account for the cross-couplings of an $p \times q$ non-square MIMO system of concern.

For the S-MANNCNS associated with the desired output r_f and actual output y_f ($f = 1, 2, \ldots, q$), and the h^{th} system's input ($h = 1, 2, \ldots, p$) as shown in Figure 5-2, there are in total three layers and six neurons. The first layer is called the input layer and contains two 'P-type' neurons. The desired output (r_f) and the system's output (y_f) propagate via those P-type neurons to the second layer named hidden layer. Within the hidden layer, there are three neurons each being a P-type, an I-type, and a D-type, respectively. The P-neuron is responsible for producing the total sum of the weighted desired outputs and the weighted actual outputs, the I-neuron provides the necessary action to eliminate the steady-state error, and the D-neuron predicts the future behaviour of the error. The third layer is named as the output layer that accumulates outputs of the hidden layer, generates the control commands respective to all desired outputs and actual outputs, and applies the control commands to the non-square MIMO system as inputs. The inputs of the P-type, I-type, and D-type (named net^1_{3f-2}, net^1_{3f-1}, and net^1_{3f}, respectively) and the outputs of these neurons (named O^1_{3f-2}, O^1_{3f-1}, and O^1_{3f}, respectively) are related together by the activation functions of the neurons represented in (1)-(3). In the hidden layer of S-MANNCNS, there are total six weights ($\overline{w}_{1,2f-1}$, $\overline{w}_{2,2f-1}$, $\overline{w}_{3,2f-1}$, $\overline{w}_{1,2f}$, $\overline{w}_{2,2f}$, and $\overline{w}_{3,2f}$) associated with the input layer and hidden layer. Likewise, in the output layer of the S-MANNCNS there are three weights ($w_{h,3f-2}$, $w_{h,3h-1}$, and $w_{h,3h}$) that connect the P-type, I-type, and D-type neurons in the hidden layer to the P-type neuron in the output layer.

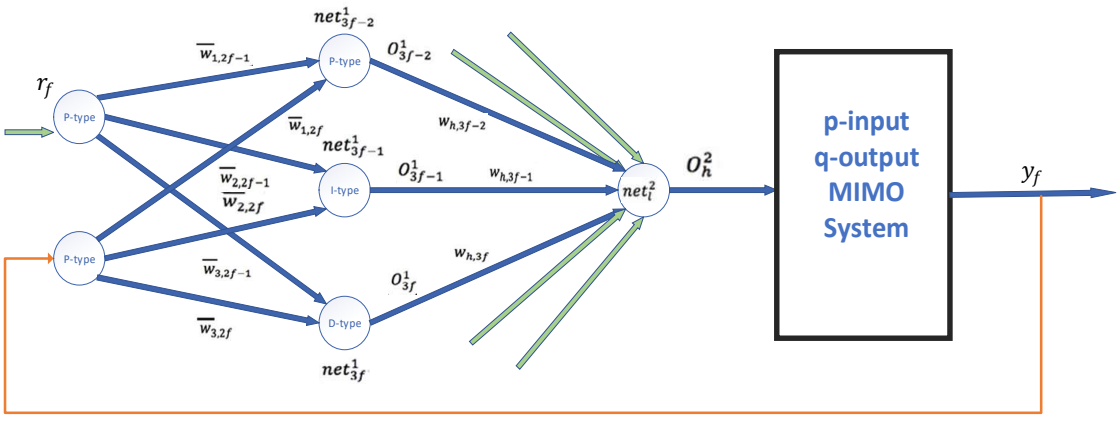


Figure 5-2. S-MANNCNS structure

5.2.3 Matrix representation

The matrix representation of a closed-loop non-square MIMO system using the proposed MANNCNS (Figure 5-1) and S-MANNCNS (Figure 5-2), is derived as follows. Let O_h^2 and y_f be, respectively, the h^{th} input and f^{th} output of the system, where, $1 \leq h \leq$ p and $1 \leq f \leq$ q, and let G_{ij} be the transfer function relating the system input O_j^2 to the system output y_i, where $1 \leq j \leq$ p and $1 \leq i \leq$ q. The vectors and matrices associated with Figure 5-1 and Figure 5-2 (named $Y_{q\times1}$, $O_{p\times1}^2$, $G_{q\times p}$, $net_{p\times1}^2$, $W_{p\times3q}$, $O_{3q\times1}^1$, $P_{3q\times3q}$, $net_{3q\times1}^1$, $\bar{W}_{3q\times3q}^r$, $\bar{W}_{3q\times3q}^y$, $R_{3q\times1}^{(3)}$, $Y_{3q\times1}^{(3)}$, $I_{3q\times q}^{(3)}$, and $R_{q\times1}$) are, respectively, defined as follows.

$$Y = Y_{q\times1} = [y_1 \quad y_2 \quad \cdots \quad y_l \quad \cdots \quad y_q]_{1\times q}^{\ T} \tag{4}$$

$$O^2 = O_{p\times1}^2 = \left[O_1^2 \quad O_2^2 \quad \cdots \quad O_l^2 \quad \cdots \quad O_p^2\right]_{1\times p}^{\ T} \tag{5}$$

$$G = G_{q\times p} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1h} & \cdots & G_{1p} \\ G_{21} & G_{22} & \cdots & G_{2h} & \cdots & G_{2p} \\ \vdots & \vdots & & & & \vdots \\ G_{f1} & G_{f2} & \cdots & G_{fh} & \cdots & G_{fp} \\ \vdots & \vdots & & \vdots & & \vdots \\ G_{q1} & G_{q2} & \cdots & G_{qh} & \cdots & G_{qp} \end{bmatrix}_{q\times p} \tag{6}$$

$$net^2 = net_{p\times1}^2 = \left[net_1^2 \quad net_2^2 \quad \cdots \quad net_l^2 \quad \cdots \quad net_p^2\right]_{1\times p}^{\ T} \tag{7}$$

$$W = W_{p\times3q} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,3f-2} & w_{1,3f-1} & w_{1,3f} & \cdots & w_{1,3q-2} & w_{1,3q-1} & w_{1,3q} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,3f-2} & w_{2,3f-1} & w_{2,3f} & \cdots & w_{2,3q-2} & w_{2,3q-1} & w_{2,3q} \\ \vdots & \vdots & & & & \vdots & \vdots & & & & \vdots \\ w_{h,1} & w_{h,2} & w_{h,3} & \cdots & w_{h,3f-2} & w_{h,3f-1} & w_{h,3f} & \cdots & w_{h,3q-2} & w_{h,3q-1} & w_{h,3q} \\ \vdots & \vdots & & & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ w_{p,1} & w_{p,2} & w_{p,3} & \cdots & w_{p,3f-2} & w_{p,3f-1} & w_{p,3f} & \cdots & w_{p,3q-2} & w_{p,3q-1} & w_{p,3q} \end{bmatrix}_{p\times3q} \tag{8}$$

$$O^1 = O_{3q\times1}^1 = \left[O_1^1 \quad O_2^1 \quad O_3^1 \quad \cdots \quad O_{3q}^1\right]_{1\times3q}^{\ T} \tag{9}$$

$$P = P_{3q\times 3q} = \begin{bmatrix} 1 & 0 & 0 & & & & \\ 0 & D^{-1} & 0 & \cdots & & \mathbf{0} & \\ 0 & 0 & D & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & 1 & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & D^{-1} & 0 \\ & & & & 0 & 0 & D \end{bmatrix}_{3q\times 3q} \tag{10}$$

$$net^1 = net^1_{3q\times 1} = \begin{bmatrix} net^1_1 & net^1_2 & net^1_3 & \cdots & net^1_{3q} \end{bmatrix}^T_{3q\times 1} \tag{11}$$

$$\bar{W}^r = \bar{W}^r_{3q\times 3q} = \begin{bmatrix} \overline{w_{1,1}} & 0 & 0 & & & & \\ 0 & \overline{w_{2,1}} & 0 & \cdots & & \mathbf{0} & \\ 0 & 0 & \overline{w_{3,1}} & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & \overline{w_{1,2q-1}} & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & \overline{w_{2,2q-1}} & 0 \\ & & & & 0 & 0 & \overline{w_{3,2q-1}} \end{bmatrix}_{3q\times 3q} \tag{12}$$

$$\bar{W}^y = \bar{W}^y_{3q\times 3q} = \begin{bmatrix} \overline{w_{1,2}} & 0 & 0 & & & & \\ 0 & \overline{w_{2,2}} & 0 & \cdots & & \mathbf{0} & \\ 0 & 0 & \overline{w_{3,2}} & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & \overline{w_{1,2q}} & 0 & 0 \\ & \mathbf{0} & & \cdots & 0 & \overline{w_{2,2q}} & 0 \\ & & & & 0 & 0 & \overline{w_{3,2q}} \end{bmatrix}_{3q\times 3q} \tag{13}$$

$$R^{(3)} = R^{(3)}_{3q\times 1} = \begin{bmatrix} r_1 & r_1 & r_1 & r_2 & r_2 & r_2 & \cdots & r_q & r_q & r_q \end{bmatrix}^T_{1\times 3q} \tag{14}$$

$$Y^{(3)} = Y^{(3)}_{3q\times 1} = \begin{bmatrix} y_1 & y_1 & y_1 & y_2 & y_2 & y_2 & \cdots & y_q & y_q & y_q \end{bmatrix}^T_{1\times 3q} \tag{15}$$

$$I^{(3)} = I^{(3)}_{3q\times q} = \begin{bmatrix} 1 & & \\ 1 & \cdots & 0 \\ 1 & & \\ \vdots & \ddots & \vdots \\ & & 1 \\ 0 & \cdots & 1 \\ & & 1 \end{bmatrix}_{3q\times q} \tag{16}$$

$$R = R_{q\times 1} = \begin{bmatrix} r_1 & r_2 & r_3 & \cdots & r_q \end{bmatrix}^T_{1\times q} \tag{17}$$

Assuming that the nonlinear system of concern is linearized around an operation point, the relationship between the inputs and outputs of the system will be:

$$Y_{q\times 1} = G_{q\times p} O^2_{p\times 1} \tag{18}$$

Since net^2_h is the h^{th} input of the P-type neuron in the output layer, one has:

$$O^2_{p\times 1} = net^2_{p\times 1} \tag{19}$$

where the relationship between the inputs and outputs in the output layer of the neural networks can be expressed as:

$$net^2_{p\times 1} = W^2_{p\times 3q} O^1_{3q\times 1} \tag{20}$$

As the hidden layer has P-type, I-type, and D-type neurons, the proportional, integral, and derivative operators (1, D^{-1}, and D) are considered respectively in the matrix form of the activation function as:

$$O^1_{3q\times 1} = P_{3q\times 3q} net^1_{3q\times 1} \tag{21}$$

where inputs of the hidden-layer neurons are the total sum of the weighted desired outputs and the actual outputs. Hence, $net^1_{3q\times 1}$ can be described as:

$$net^1_{3q\times 1} = \bar{W}^r_{3q\times 3q} R^3_{3q\times 1} + \bar{W}^y_{3q\times 3q} Y^3_{3q\times 1} \tag{22}$$

By having:

$$R^{(3)}_{3q\times 1} = I^{(3)}_{3q\times q} \times R_{q\times 1} \tag{23}$$

and:

$$Y^{(3)}_{3q\times 1} = I^{(3)}_{3q\times q} \times Y_{q\times 1} \tag{24}$$

and using equations (18)-(22), one obtains:

$$Y = GO^2 = G.net^2 = GWO^1 = GWP.net^1 = GWP(\bar{W}^r R^3 + \bar{W}^y Y^3) \tag{25}$$

Substitute (23) and (24) into (25), one has:

$$Y = \mathrm{GWP}(\bar{W}^r I^{(3)} R + \bar{W}^y I^{(3)} Y) \tag{26}$$

Hence, the system output can be derived as:

$$Y = \left(I - GWP\bar{W}^y I^{(3)}\right)^{-1} GWP\bar{W}^r I^{(3)} R \tag{27}$$

where: $\left|I - GWP\bar{W}^y I^{(3)}\right| \neq 0$.

It should be pointed out that although Equations (18)-(27) are derived under the assumption that the system can be linearized around an operating point, they can potentially be used for nonlinear systems where the nonlinearities of the systems can be approximated by piece-wise linear systems whose time-varying

nature can account for the nonlinearities of the systems satisfactorily. It should also be pointed out that if the non-singularity condition for matrix $\left(I - GWP\bar{W}^{y}I^{(3)}\right)$ could not be met, the selected weights (matrices W, $\bar{W}^{y}$, and $\bar{W}^{r}$) would not be acceptable and would be re-updated until $\left(I - GWP\bar{W}^{y}I^{(3)}\right)$ becomes non-singular.

5.3 Learning algorithm

In order to achieve a precise control effect for a non-square MIMO system, the neural network weights of the MANNCNS are adjusted using the principle of the multi-step error back-propagation algorithm described in [20] . However, instead of using merely the current gradient of the system error as the literature does for SISO systems in [21-23], an accumulated gradient of the system error that utilises the full history of the system outputs and the desired outputs to achieve a more accurate control outcome. The proposed method minimises the sum of the square accumulated gradient of the error for each system output, where the error is taken as the difference between the desired output $r_f(k)$ (i.e., the system set-point) and the actual output $y_f(k)$. Euclidean Norm (E) is defined for computing the quadratic cost function of the system for the system error. The cost function for the 'l^{th}' S-MANNCNS (Figure 5-2) is thus defined as:

$$E_f(v) = \frac{1}{2}\left(\sum_{k=1}^{m}(r_f[k] - y_f(v)[k])\right)^2 \tag{28}$$

where $E_f(v)$ is the error of the f^{th} output at the v^{th} step of the learning algorithm and m is the required number of discrete samples of the actual output and desired output. It should be noted that power of two in this expression makes the error of each sample positive so that, larger errors become weightier than the smaller errors. By increasing m, the system output will be compared more accurately with the system set-point. However, a large value of m may slow down the controller 's learning process which is undesirable when the speed of the control action is critical in many demanding real-time industrial operations. Therefore, a reasonable value for the number of samples (m) must be used to compromise between the desired control accuracy and the essential speed of the

control action. The total cost function of the system (J) which is the sum of all errors described in (28) is written as:

$$J(v) = \sum_{f=1}^{q} E_f(v) = \frac{1}{2}\sum_{f=1}^{q}\left(\sum_{k=1}^{m}(r_f[k] - y_f(v)[k])\right)^2 \quad (29)$$

where q is the number outputs of the non-square MIMO system. This expression will be used to derive the learning algorithm for weights of both dynamic layers of the neural networks of the MANNCNS.

Using the accumulated gradient of the system error, a learning algorithm is to be developed to minimise the defined cost function and to bring the system actual outputs as close as possible to the system desired outputs. There are two-dynamic layers of weights in the neural network structure of the MANNCNS that are required to be trained: the output layer and the hidden layer, for which two learning algorithms are derived in Subsections 1.3.1 and 1.3.2, respectively.

5.3.1 Output-layer Weights Learning Algorithm

A learning algorithm of the output layer is defined to train the output-layer weights and to link with the learning algorithm of the hidden layer (to be represented in the following subsection) together in order to run simultaneously during a control action. According to the principle of the error back-propagation learning algorithm, the output-layer weights must be adjusted so that in each step they slightly move in the opposite direction of the gradient of the cost function respective to the weights of the output layer. This is to guarantee that the cost function will be decreasing gradually while the learning algorithm is running. Therefore, the weights between the hidden layer and the output layer will be adjusted based on the following learning rule:

$$w_{h,x}(v+1) = w_{h,x}(v) - \lambda_h \frac{\partial J(v)}{\partial w_{h,x}} \quad (30)$$

where $1 \le h \le p$, $3f-2 \le x \le 3f$ and $1 \le f \le q$, and v is the step number of the learning algorithm; $w_{h,x}(v)$ and $w_{h,x}(v+1)$ are weights of the output layer in the current and following steps, respectively; λ_h determines weight training rate and speed that decides how fast the cost is changing and principally determines

the weight-training speed of the output layer. Using partial derivatives to calculate the gradient of the error subject to each weight, one has

$$\frac{\partial J}{\partial w_{h,x}} = \frac{\partial J}{\partial E_f}\frac{\partial E_f}{\partial y_f}\frac{\partial y_f}{\partial O_h^2}\frac{\partial O_h^2}{\partial net_h^2}\frac{\partial net_h^2}{\partial w_{h,x}} \tag{31}$$

where:

$$\frac{\partial J}{\partial E_f} = 1 \tag{32}$$

$$\frac{\partial E_f}{\partial y_f} = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \tag{33}$$

$$\frac{\partial y_f}{\partial O_h^2} \cong \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \tag{34}$$

Due to having P-type neuron in the output layer, one writes:

$$\frac{\partial O_h^2}{\partial net_h^2} = 1 \tag{35}$$

and:

$$\frac{\partial net_h^2}{\partial w_{h,x}} = \sum_{k=1}^{m}(O_x^1(v)[k]) \tag{36}$$

Substituting (32)-(36) into (31), one obtains:

$$\frac{\partial J}{\partial w_{h,x}} \cong 1 \times \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \times 1 \times O_x^1(v)[k] \tag{37}$$

Defining $\gamma_l(k)$ as:

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \tag{38}$$

the output-layer weight adjustment rule thus be derived as:

$$w_{h,x}(v+1) = w_{h,x}(v) - \lambda_h[\gamma_{fh}(v) \times \sum_{k=1}^{m}(O_x^1(v)[k])] \tag{39}$$

5.3.2 Hidden-layer Weights Learning Algorithm

Corresponding to the output-layer weight adjustment in the previous section, according to the principle of the error back-propagation learning algorithm, the hidden-layer weights (i.e, the weights between the input layer and the hidden layer) must be adjusted so that in each step they move slightly in the opposite direction of the gradient of the cost function respective to the weights in this

layer. This is to guarantee that the cost function will be decreasing gradually during the learning process. The hidden-layer weights are therefore adjusted based on the following learning rule:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \mu_f \frac{\partial J(v)}{\partial \overline{w_{a,b}}} \tag{40}$$

where $a = 1, 2, 3$, $b = 2f - 1, 2f$ and $f = 1, 2, \ldots, q$; v is the step number of the learning algorithm; $\overline{w_{a,b}}(v)$ and $\overline{w_{a,b}}(v+1)$ are the weights of the hidden layer at the current and following steps, respectively; μ_l is the hidden-layer learning rate that decides how fast the cost is changing and principally determines the weight-training speed of the hidden layer. The gradient of the error with respect to each weight is required to be calculated. Using partial derivatives, one has:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q} \sum_{h=1}^{p} \left(\frac{\partial J}{\partial E_f} \times \frac{\partial E_f}{\partial y_f} \times \frac{\partial y_f}{\partial O_h^2} \times \frac{\partial O_h^2}{\partial net_h^2} \times \frac{\partial net_h^2}{\partial O_x^1} \times \frac{\partial O_x^1}{\partial net_x^1} \times \frac{\partial net_x^1}{\partial \overline{w_{a,b}}} \right) \tag{41}$$

Since $x = 3f - 3 + a$, x must be chosen adequately based on a so that a proper partial derivative path within the hidden layer can be selected. Considering the partial derivative calculations of the output layer, one has:

$$\frac{\partial J}{\partial E_f} \times \frac{\partial E_f}{\partial y_f} \times \frac{\partial y_f}{\partial O_h^2} \times \frac{\partial O_h^2}{\partial net_h^2} = \sum_{k=1}^{m} (y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m} (y_f(v)[k]) - \sum_{k=1}^{m} (y_f(v-1)[k])}{\sum_{k=1}^{m} (O_h^2(v)[k]) - \sum_{k=1}^{m} (O_h^2(v-1)[k])} \tag{42}$$

Define $\gamma_{fh}(h)$ as:

$$\gamma_{fh}(h) = \sum_{k=1}^{m} (y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m} (y_f(v)[k]) - \sum_{k=1}^{m} (y_f(v-1)[k])}{\sum_{k=1}^{m} (O_h^2(v)[k]) - \sum_{k=1}^{m} (O_h^2(v-1)[k])} \tag{43}$$

Considering the output layer, one has:

$$\frac{\partial net_h^2}{\partial O_x^1} = w_{h,x}(h) \tag{44}$$

where $w_{h,x}(v)$ are derived from the output-layer weight adjustment algorithm given in Subsection 1.3.1. It is seen that the weight adjustment processes for both the hidden layer and the output layer must run step by step and one after another alternately.

As the neurons in the hidden layers are from various types of P-type, I-type and D-type, consecutively, the calculation of term $\frac{\partial O_x^1}{\partial net_x^1}$ in (41) depends on the type of

the selected neuron in the hidden layer. Because the inputs of the hidden layer are the desired output (r_f) or the actual output of the system (y_f), term $\frac{\partial net_x^1}{\partial w_{a,b}}$ in (41) is also relevant to the selection of the actual output or the desired output. Hence, the gradient of the cost function should be expressed in six different types of expressions depending on a and b as listed below:

- **Type 1)** $\boldsymbol{a = 1}$ **and** $\mathbf{b} = \boldsymbol{2f - 1}$

In this type, the cost function's partial derivative path has a P-type neuron in the hidden layer, therefore:

$$\frac{\partial O_x^1}{\partial net_x^1} = 1 \tag{45}$$

The input of the hidden layer is the desired output (r_f), which gives:

$$\frac{\partial net_x^1}{\partial w_{a,b}} = r_f \tag{46}$$

Thus, one has:

$$\frac{\partial J}{\partial w_{a,b}} = \sum_{f=1}^{q} \sum_{h=1}^{p} (\gamma_{fh}(h) \times w_{h,3f-2}(h) \times r_f(h)) \tag{47}$$

Substituting (41)-(47) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q} \sum_{h=1}^{p} (\mu_f \gamma_{fh}(v) w_{h,3f-2}(v) r_f(v)[m]) \tag{48}$$

where,

$$\gamma_{fh}(v) = \sum_{k=1}^{m} (y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m} (y_f(v)[k]) - \sum_{k=1}^{m} (y_f(v-1)[k])}{\sum_{k=1}^{m} (O_h^2(v)[k]) - \sum_{k=1}^{m} (O_h^2(v-1)[k])} \tag{49}$$

- **Type 2)** $\boldsymbol{a = 1}$ **and** $\mathbf{b} = \boldsymbol{2f}$

In this type, the cost function's partial derivative path has a P-type neuron in the hidden layer, therefore:

$$\frac{\partial O_x^1}{\partial net_x^1} = 1 \tag{50}$$

The input of the hidden layer is the actual output (y_f), which gives:

$$\frac{\partial net_x^1}{\partial w_{a,b}} = y_f(h) \tag{51}$$

Thus, one has:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q}\sum_{h=1}^{p}(\gamma_{fh}(h) \times w_{h,3f-2}(v) \times y_f(v)) \quad (52)$$

Substituting (41)-(44) and (50)-(52) into (40) the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}\{\mu_f \gamma_{fh}(v) w_{h,3f-2}(v) y_f(v)[m]\} \quad (53)$$

where,

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \quad (54)$$

- **Type 3) $a = 2$ and $b = 2f - 1$**

In this type, the cost function's partial derivative path has an I-type neuron in the hidden layer, therefore:

$$O_x^1(k) = \sum_{k'=1}^{k} net_x^1(k') = \sum_{k'=1}^{k-1} net_x^1(k') + net_x^1(k) = O_x^1(k-1) + net_x^1(k) \quad (55)$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k) - O_x^1(k-1)}{net_x^1(k) - ne\ {}_x^1(k-1)} = \frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)} \quad (56)$$

The input of the hidden layer is the desired output (r_f), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = r_f \quad (57)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q}\sum_{h=1}^{p}(\gamma_{fh}(v) \times w_{h,3f-1}(v) \times sign\left[\frac{net_x^1[m]}{net_x^1[m] - net_x^1[m-1]}\right] \times r_f(v)[m]) \quad (58)$$

Substituting (41)-(44) and (56)-(58) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) -$$

$$\sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f-1}(v) sign\left[\frac{net_{3f-1}^1(v)[m]}{net_{3f-1}^1(v)[m] - net_{3f-1}^1(v)[m-1]}\right] r_f(v)[m]) \quad (59)$$

where,

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \quad (60)$$

If $\left[\frac{net_{3f-1}^{1}(v)[m]}{net_{3f-1}^{1}(v)[m] - net_{3f-1}^{1}(v)[m-1]}\right]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 4) $a = 2$ and $b = 2f$**

In this type, the cost function's partial derivative path has an I-type neuron in the hidden layer, therefore:

$$O_x^1(k) = \sum_{k'=1}^{k} net_x^1(k') = \sum_{k'=1}^{k-1} net_x^1(k') + net_x^1(k) = O_x^1(k-1) + net_x^1(k) \quad (61)$$

and:

$$\frac{\partial O_x^1}{\partial n \quad _x^1} = \frac{O_x^1(k) - O_x^1(k-1)}{net_x^1(k) - net_x^1(k-1)} = \frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)} \quad (62)$$

The input of the hidden layer is the actual output (y_f), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = y_f(v) \quad (63)$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)}{net_x^1(k) - net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q}\sum_{h=1}^{p}(\gamma_{fh}(v) \times w_{h,3f-1}(v) \times sign\left[\frac{net_{3f-1}^{1}[m]}{net_{3f-1}^{1}[m] - net_{3f-1}^{1}[m-1]}\right] \times y_f(v)[m])$$

(64)

Substituting (41)-(44) and (62)-(64) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) -$$

$$\sum_{f=1}^{q}\sum_{h=1}^{p}\left\{\mu_f \gamma_{fh}(v) w_{h,3f-1}(v) sign\left[\frac{net_{3f-1}^{1}[m]}{net_{3f-1}^{1}[m] - net_{3f-1}^{1}[m-1]}\right] y_f(v)[m]\right\} \quad (65)$$

Where:

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])} \quad (66)$$

If $[\frac{net_{3f-1}^{1}[m]}{net_{3f-1}^{1}[m]-net_{3f-1}^{1}[m-1]}]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 5) $a = 3$ and $b = 2f - 1$**

In this type, the cost function's partial derivative path has a D-type neuron in the hidden layer, therefore:

$$O_x^1(k) = net_x^1(k) - net_x^1(k-1) \tag{67}$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k)-O_x^1(k-1)}{net_x^1(k)-net_x^1(k-1)} = \frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)} \tag{68}$$

The input of the hidden layer is the desired output (r_f), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = r_f \tag{69}$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)}$, if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q}\sum_{h=1}^{p}(\gamma_{fh}(v) \times w_{h,3f}(v) \times sign\left[\frac{net_{3f}^{1}[m]-2net_{3f}^{1}[m-1]+net_{3f}^{1}[m-2]}{net_{3f}^{1}[m]-net_{3f}^{1}[m-1]}\right] \times r_f(v)[m]) \tag{70}$$

Substituting (41)-(44) and (68)-(70) into (40), the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) -$$
$$\sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f\gamma_{fh}(v)w_{h,3f}(v)sign\left[\frac{net_{3f}^{1}[m]-2net_{3f}^{1}[m-1]+net_{3f}^{1}[m-2]}{net_{3f}^{1}[m]-net_{3f}^{1}[m-1]}\right] \times r_f(v)[m])$$

(71)

Where:

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k])-\sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k])-\sum_{k=1}^{m}(O_h^2(v-1)[k])} \tag{72}$$

If $\left[\frac{net_{3f}^{1}[m]-2net_{3f}^{1}[m-1]+net_{3f}^{1}[m-2]}{net_{3f}^{1}[m]-net_{3f}^{1}[m-1]}\right]$ is undefined due to zero denominator in any step of the learning algorithm, the associated weight will remain unchanged until the next step.

- **Type 6) $a = 3$ and $b = 2f$**

In this type, the cost function's partial derivative path has a D-type neuron in the hidden layer, therefore:

$$O_x^1(k) = net_x^1(k) - net_x^1(k-1) \tag{73}$$

and:

$$\frac{\partial O_x^1}{\partial net_x^1} = \frac{O_x^1(k)-O_x^1(k-1)}{net_x^1(k)-net_x^1(k-1)} = \frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)} \tag{74}$$

The input of the hidden layer is the actual output (y_f), which gives:

$$\frac{\partial net_x^1}{\partial \overline{w_{a,b}}} = y_f(h) \tag{75}$$

Thus, because of the possibility of significant changes in $\frac{net_x^1(k)-2net_x^1(k-1)+net_x^1(k-2)}{net_x^1(k)-net_x^1(k-1)}$ if $net_x^1(k) - net_x^1(k-1) \ll 1$, the sign of this term will be used to determine the direction of the gradient as:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \sum_{f=1}^{q}\sum_{h=1}^{p}(\gamma_{fh}(v) \times w_{h,3f}(v) \times sign\left[\frac{net_{3f}^{1}[m]-2net_{3f}^{1}[m-1]+n\ \ {}_{3f}^{1}[m-2]}{net_{3f}^{1}[m]-net_{3f}^{1}[m-1]}\right] \times y_f(v)[m]) \tag{76}$$

Substituting (41)-(44) and (74)-(76) into (40) the learning rule is derived as:

$$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) -$$
$$\sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f\gamma_{fh}(v)w_{h,3f}(v)sign\left[\frac{net_{3f}^{1}[m]-2net_{3f}^{1}[m-1]+net_{3f}^{1}[m-2]}{net_{3f}^{1}[m]-net_{3f}^{1}[m-1]}\right] \times y_f(v)[m])$$

(77)

where:

$$\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k])-\sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k])-\sum_{k=1}^{m}(O_h^2(v-1)[k])} \tag{78}$$

The weight learning algorithms for both the output layer and the hidden layer of the neural networks of the MANNCNS are summarised in Table 5-1.

Output layer	
$1 \le f \le \mathrm{q}$ $1 \le h \le \mathrm{p}$ $3\mathrm{f}-2 \le x \le 3\mathrm{f}$	$w_{h,x}(v+1) = w_{h,x}(v) - \lambda_h[\gamma_{fh}(v) \times \sum_{k=1}^{m}(O_x^1(v)[k])]$ $\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])}$
Hidden layer	
$\boldsymbol{a = 1}$ $\mathbf{b = 2f-1}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f-2}(v) r_f(v)[m])$
$\boldsymbol{a = 1}$ $\mathbf{b = 2f}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}\{\mu_f \gamma_{fh}(v) w_{h,3f-2}(v) y_f(v)[m]\}$
$\boldsymbol{a = 2}$ $\mathbf{b = 2f-1}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f-1}(v) sign\left[\frac{net_{3f-1}^1(v)[m]}{net_{3f-1}^1(v)[m] - net_{3f-1}^1(v)[m-1]}\right] r_f(v)[m])$
$\boldsymbol{a = 2}$ $\mathbf{b = 2f}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f-1}(v) sign\left[\frac{net_{3f-1}^1[m]}{net_{3f-1}^1[m] - net_{3f-1}^1[m-1]}\right] y_f(v)[m])$
$\boldsymbol{a = 3}$ $\mathbf{b = 2f-1}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f}(v) sign\left[\frac{net_{3f}^1[m] - 2net_{3f}^1[m-1] + net_{3f}^1[m-2]}{net_{3f}^1[m] - net_{3f}^1[m-1]}\right] \times r_f(v)[m])$
$\boldsymbol{a = 3}$ $\mathbf{b = 2f}$	$\overline{w_{a,b}}(v+1) = \overline{w_{a,b}}(v) - \sum_{f=1}^{q}\sum_{h=1}^{p}(\mu_f \gamma_{fh}(v) w_{h,3f}(v) sign\left[\frac{net_{3f}^1[m] - 2net_{3f}^1[m-1] + net_{3f}^1[m-2]}{net_{3f}^1[m] - net_{3f}^1[m-1]}\right] \times y_f(v)[m])$ where, $\gamma_{fh}(v) = \sum_{k=1}^{m}(y_f(v)[k] - r_f(v)[k]) \times \frac{\sum_{k=1}^{m}(y_f(v)[k]) - \sum_{k=1}^{m}(y_f(v-1)[k])}{\sum_{k=1}^{m}(O_h^2(v)[k]) - \sum_{k=1}^{m}(O_h^2(v-1)[k])}$ $1 \le f \le \mathrm{q}$ and $1 \le h \le \mathrm{p}$ • If the $sign[\dots]$ is unknown due to zero denominator, the associated weight will remain unchanged until the next step.

Table 5-1. Weights adjustment learning algorithms

From the summary shown in Table 5-1, it is noted that since the weights in the output layer are used by the hidden-layer learning algorithm at each step, the neural networks of the both layers must be trained simultaneously (i.e., one after another alternately). The fact that the calculations of the output-layer weights and the hidden-layer weights must be linked together indicates that the weights of the different layers of the neural networks of the MANNCNS are dependent

on each other. This dependence reflects the inherent capacity of the MANNCNS in dealing with strong cross-couplings of the MIMO system of concern. It is also noted that the overall weight training procedure must initially start from the output layer, because at each learning step the calculation of the hidden-layer weights needs the values of the output-layer weights from the previous learning step.

5.4 Stability Analysis

It is acknowledged that a feedback control system must be stable as a precondition for a satisfactory control method. When a new control method is proposed, in order to achieve the desired control outcomes, the stability criteria of the resultant closed-loop system must be derived, and any ensuing time-dependent conditions or constraints must be checked online during the entire dynamic process of the closed-loop control. For an unconstrained control system, the system stability is satisfied if all the output responses are bounded for all bounded inputs. Because the eigenvalue analysis concept applies only to linear systems with model-based approaches, it cannot be used in nonlinear systems with model-free approaches. Therefore, for the proposed MANNCNS, the Lyapunov stability analysis concept that is suitable for nonlinear systems, relies only on a system's inputs and outputs, and does not need to use the model of the system, must be used. Although, unlike linear systems, the stability of a nonlinear system does not need to be always global as the system can have multiple equilibrium points and limit cycles, the global asymptotic stability of a nonlinear MIMO system under the MANNCNS control is to be sought in this study in order to guarantee that the ensuing closed-loop system will never be locked in its local minimum during the entire dynamic control process.

As summarised in Table 5-1, there are two sets of free parameters λ_h $(1 \leq h \leq$ p) and μ_f $(1 \leq f \leq$ q) for the output-layer and hidden-layer weight-learning algorithms of the MANNCNS, respectively. These parameters are denoted as the learning rates of the respective dynamic layers of the MANNCNS. They determine the weight-training speed of the respective layers and can dynamically change during the controller learning process. A set of constraints

for these learning rates will be developed when the closed-loop system stability issue of the MANNCNS method is to be examined.

According to the Lyapunov global asymptotic stability theorem, for a defined function $V(x)$, if:

(i) $V(0) = 0$

(ii) For all $x \neq 0$, $V(x) > 0$ (V is positive definite)

(iii) For all $x \neq 0$, $\Delta V(x) < 0$

then every trajectory of $X(k+1) = X(k) + f(X(k))$ will converge to zero as $k \to \infty$ and the system will be globally asymptotically stable. Let the Lyapunov function for each output of a nonlinear MIMO system under the MANNCNS control be the total cost function of that output, i.e.:

$$V_f(v) = (E_f(v))^2 \tag{79}$$

where $E_f(v)$ is the cost function related to v^{th} step of the learning algorithm. One writes:

$$\Delta V_f(v) = (E_f(v) + \Delta E_f(v))^2 - (E_f(v))^2 = 2E_f(v)\Delta E_f(v) + (\Delta E_f(v))^2 \tag{80}$$

and:

$$\Delta E_f(v) \cong \Delta w_{h,x}(h) \sum_{k=1}^{m} \frac{\partial E_f(v)[k]}{\partial w_{h,x}} \tag{81}$$

For the output layer of the MANNCNS, considering the Lyapunov function $V_f(h)$ in (79) and the expression $\Delta V_f(h)$ in (80), from the learning rule expressed in (30) for the output-layer weights, one has:

$$\Delta w_{h,x}(v) = -\frac{\lambda_h}{m} \sum_{k=1}^{m} \frac{\partial J}{\partial w_{h,x}} \tag{82}$$

and:

$$\frac{\partial J}{\partial w_{h,x}} = \frac{\partial J}{\partial E_f(v)} \frac{\partial E_f(v)}{\partial w_{h,x}} = \frac{\partial E_f(v)}{\partial w_{h,x}} \tag{83}$$

Substituting (83) into (82), one obtains:

$$\Delta w_{h,x}(v) = -\frac{\lambda_h}{m}\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial w_{h,x}} \tag{84}$$

and:

$$\Delta E_f(v) \cong -\frac{\lambda_h}{m}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial w_{h,x}}\right)^2 \tag{85}$$

therefore (80) can be expressed as:

$$\Delta V(v) \cong -\frac{2\lambda_h E_f(v)}{m}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial w_{h,x}}\right)^2 + \frac{{\lambda_h}^2}{m^2}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial w_{h,x}}\right)^4 \tag{86}$$

Define:

$$H_h(v) = min\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial w_{h,x}} \cong \min_{3\mathrm{f}-2 \le x \le 3\mathrm{f}} \sum_{k=1}^{m}\frac{\Delta E_f(v)[k]}{\Delta w_{h,x}(v)} \tag{87}$$

The condition for $\Delta V(v) < 0$ yields the following constraint on the selection of learning rate λ_h defined in (39):

$$0 < \lambda_h < \frac{2mE_f(v)}{H_h(v)^2} \tag{88}$$

The above constraint for learning rate λ_h must be satisfied as a necessary condition of the MANNCNS control system at the v^{th} step. This constraint must be met at each training step in order for the MANNCNS to maintain its closed-loop system's global asymptotical stability during the entire learning process.

For the hidden layer of the MANNCNS, considering the same Lyapunov function $V_f(h)$ in (79) and the same expression $\Delta V_f(h)$ in (80), one has the following estimation based on the hidden-layer weights described in (40):

$$\Delta E_f(v) \cong \Delta\overline{w_{a,b}}(h)\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial \overline{w_{a,b}}} \tag{89}$$

and:

$$\Delta\overline{w_{a,b}}(v) = -\frac{\mu_f}{m}\sum_{k=1}^{m}\frac{\partial J}{\partial \overline{w_{a,b}}} \tag{90}$$

where:

$$\frac{\partial J}{\partial \overline{w_{a,b}}} = \frac{\partial J}{\partial E_f(v)}\frac{\partial E_f(v)}{\partial \overline{w_{a,b}}} = \frac{\partial E_f(v)}{\partial \overline{w_{a,b}}} \tag{91}$$

Substituting (91) into (90), one obtains:

$$\Delta\overline{w_{a,b}}(v) = -\frac{\mu_f}{m}\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial\overline{w_{a,b}}} \tag{92}$$

Also, substituting (92) into (89):

$$\Delta E_f(v) \cong -\frac{\mu_f}{m}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial\overline{w_{a,b}}}\right)^2 \tag{93}$$

Hence, $\Delta V_f(v)$ can be expressed as:

$$\Delta V(v) \cong -\frac{2\mu_f E_f(v)}{m}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial\overline{w_{a,b}}}\right)^2 + \frac{{\mu_f}^2}{m^2}\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial\overline{w_{a,b}}}\right)^4 \tag{94}$$

Define:

$$\overline{H_f}(h) = \min\left(\sum_{k=1}^{m}\frac{\partial E_f(v)[k]}{\partial\overline{w_{a,b}}}\right) \cong \min_{2f-1 \le b \le 2f \,\&\, 1 \le a \le 3}\left(\sum_{k=1}^{m}\frac{\Delta E_f(v)[k]}{\Delta\overline{w_{a,b}}(v)}\right) \tag{95}$$

condition of $\Delta V(h) < 0$ yields to the following constraint on the selection of learning rate μ_f defined in (77):

$$0 < \mu_f < \frac{2mE_f(v)}{\overline{H_f}(v)^2} \tag{96}$$

The above constraint for learning rate μ_f must be satisfied as another necessary condition of the MANNCNS control system at the v^{th} step. This constraint must also be met at each training step in order for the MANNCNS to maintain its closed-loop system's global asymptotical stability during the entire learning process.

Collectively, conditions (96) and (88) must both be satisfied at each training step in order to guarantee the global asymptotical stability of the MANNNS control system at the next training step. Simultaneous satisfaction of these two conditions is then considered as the sufficient condition for the global asymptotical stability of the MANNNS control system, as summarised in Table 5-2.

Output layer constraint	$0 < \lambda_h < \frac{2mE_f(v)}{H_h(v)^2}$ where: $H_h(v) \cong \min_{3f-2 \le x \le 3f} \left(\sum_{k=1}^{m} \frac{\Delta E_f(v)[k]}{\Delta w_{h,x}(v)}\right)$
Hidden layer constraint	$0 < \mu_f < \frac{2mE_f(v)}{\overline{H_f}(v)^2}$ where: $\overline{H_f}(h) \cong \min_{2f-1 \le b \le 2f \,\&\, 1 \le a \le 3} \left(\sum_{k=1}^{m} \frac{\Delta E_f(v)[k]}{\Delta \overline{w_{a,b}}(v)}\right)$

Table 5-2. Stability criteria for hidden layer and output layer

It should be pointed out that it is necessary to check the stability conditions at each learning step, due to the fact that these conditions depend on the real-time values of the cost function ($E_f(v)$), the change of cost function ($\Delta E_f(v)$), and the change of weights ($\Delta w_{h,x}(v)$ and $\Delta\overline{w_{a,b}}(v)$) at each training step. Both λ_h and μ_f can dynamically change to keep the closed-loop control system stable while the weights are trained by the learning algorithms. To avoid changing λ_h and μ_f dynamically, a small conservative constant value (e.g. 0.01 or less) can be selected for these parameters during the entire weight learning process. However, this arrangement would lead to a slower weight-adjustment rate and would require a greater number of training steps in order to achieve satisfactory results. In addition, it is noted from (88) and (96) that, by choosing a larger sampling number (m), the possibility of an instable system will become lower as there will be a larger range for λ_h and μ_f. However, a larger sampling number (m) will lead to a slower weight-adjustment rate and, thus, a slower control speed of the resultant closed-loop system.

The stability criteria presented in Table 5-2 will be checked in a real-time manner together with the computation of the weight adjustment algorithms presented in Table 5-1 at each training step. This online simultaneous procedure is illustrated in Figure 5-3 as a block diagram.

It should be pointed out that, when the stability criteria (88) and (96) are not met, the values of the learning rates are halved in order to enable the continuation of the learning process. It is acknowledged that while halving the learning rates may lead to a better control performance in terms of the error

reductions, this arrangement may prolong the weight adjustment process and reduce the control speed of the system as a result.

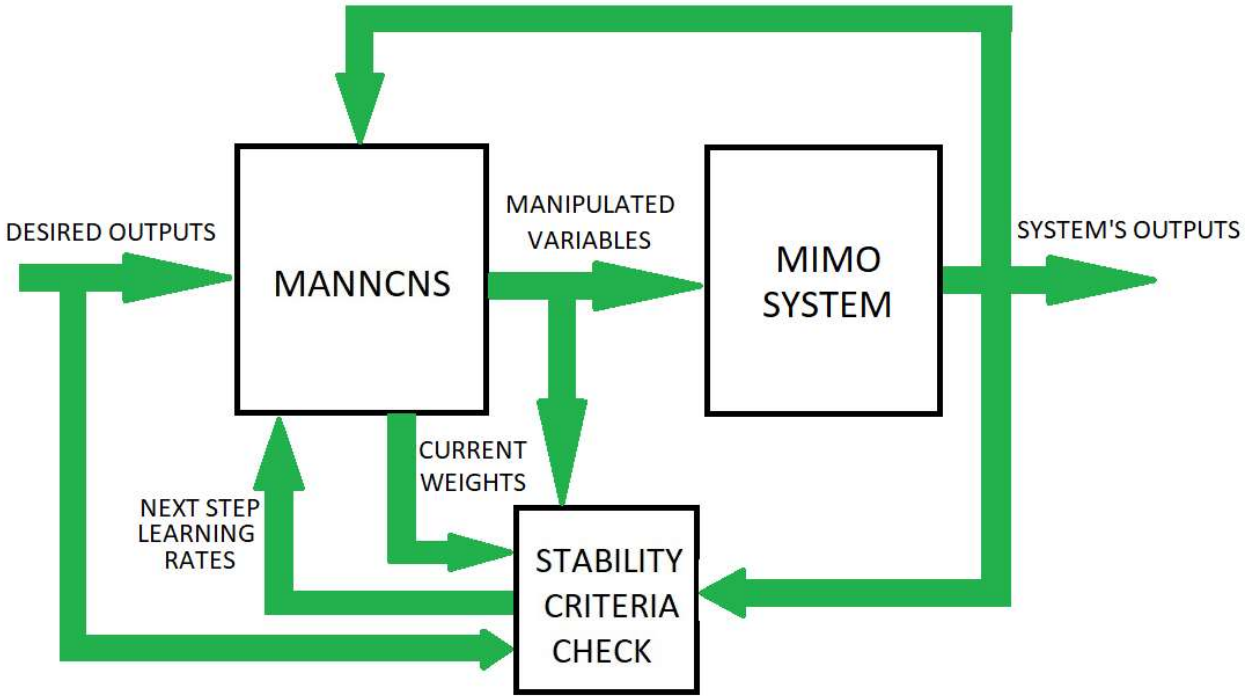


Figure 5-3. Block diagram of the controller and stability criteria checking process

5.5 Simulation results

Simulation studies using Matlab are carried out to assess the performances of the proposed MANNCNS in controlling both cases of non-square systems (i.e. number of inputs is more than number of outputs or vice versa), particularly in set-points tracking, continuous error reduction, weight re-adjustment when set-point changes, and rejecting unwanted disturbances, while the global stability of the closed-loop control system is secured during the system's entire control process. The MANNNS structure proposed in Section 5.2, the dynamic neural network algorithm developed in Section 5.3, and the stability conditions derived in Section 5.4 are used in the simulations.

5.5.1 Case 1 – Application of MANNCNS on a 3-input 2-output system

In this case, a typical industrial distillation column as shown in Figure 5-4 is considered to test the performance of the proposed control method via a computer simulation study.

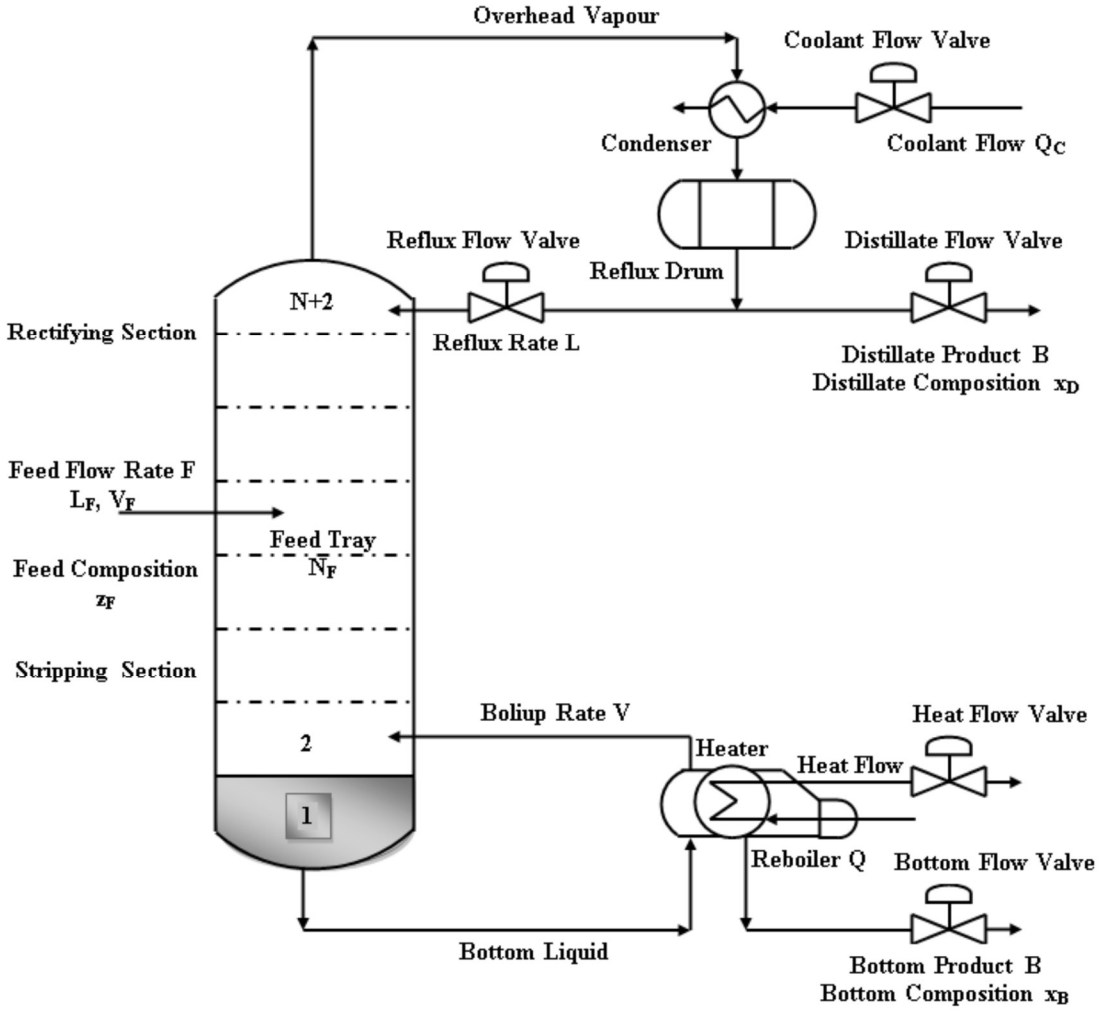


Figure 5-4. Plant Diagram of a distillation column [2]

Distillation is used to purify the end products, chemical and fuel industries as a separation technology. A range of components constitute the distillation columns, and each of these either moves the heat energy or augments mass transfer. Normally, a distillation column is comprised of a vertical column, wherein trays or plates are employed to improve the separation of the components; a reboiler positioned at the base of the column, which supplies the heat needed for vaporization; a condenser situated at the top of the column, which cools and condenses the vapor; lastly, a reflux drum which saves the vapor so condensed, whereby it can be recycled to repeat the process all over again [2]. To analyse the comparative study on the above control technique, a coupled distillation column with 3 inputs and 2 outputs is considered that is modelled by Levein and Morari in [24] with a linear estimation of the transfer function matrix with deadtime.

$$G(s)=\begin{bmatrix} \frac{0.052\ e^{-8s}}{19.8s+1} & \frac{-0.03(1-15.8s)}{108s^2+63s+} & \frac{0.012(1-47s)}{181s^2+29s+} \\ \frac{0.0725}{890s^2+64s+1} & \frac{-0.0029(1-560\ )}{293s^2+51s+1} & \frac{0.0078}{42.3s+1} \end{bmatrix} \quad (97)$$

The functional block diagram of the closed loop MANNCNS control system for the non-square MIMO distillation column plant in represented in Figure 5-5. Mole fraction of ethanol in distillate (y_1) and mole fraction of water in bottoms (y_2) are the controlled variables (system outputs) and the distillate flow rate (MV_1), steam flow rate (MV_2), and product fraction from the side column (MV_3) are the manipulated variables (system inputs). According to the mentioned transfer function in (97), and due to the fact that all system inputs affect both systems outputs, the system is a cross-coupled non-square MIMO and cannot be feasibly controlled by multiple cascaded ANNC controllers introduced in [19] by the authors. Also, MANNC and MANNC2 introduced in [17, 18] by the authors, are not able to control this system due to their restrictions of controlling square MIMO systems. Due to having e^{-8s} in member G_{11} of the transfer function matrix, the system experiences a delay between input one and output one. Therefore, the ability of MANNCNS on controlling systems with deadtime is tested in this simulation. It should be pointed out that, although the transfer function matrix of the system is known in (97), it is not used for the controller design and the plant is treated as a black-box system by utilising only the history of the input signals and output signals, but the transfer function matrix is used for generating the system outputs in the computer simulation. Hence, the MANNCNS performs as a fully model-free controller without attainment of any information from the system model.

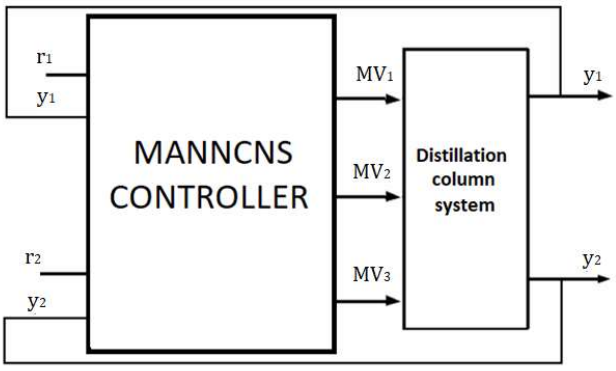


Figure 5-5. Block diagram of distillation column system and MANNCNS controller

Neural network structure of a MANNCNS for controlling the 3-input 2-output system is demonstrated in Figure 5-6. In this structure, there are totally, 12 dynamical weights in the hidden layer and 18 dynamical weights in the output layer of the MANNCNS to be adjusted. All blue, green, and red arrows in the hidden layer and the output layer represent the dynamical weights. The green arrows and red arrows are added into the structure to enable the controller to handle cross-couplings between the three inputs and two outputs of the plant. The implementation of online and real-time stability criteria check presented in Table 5-2 in represented in Figure 5-7.

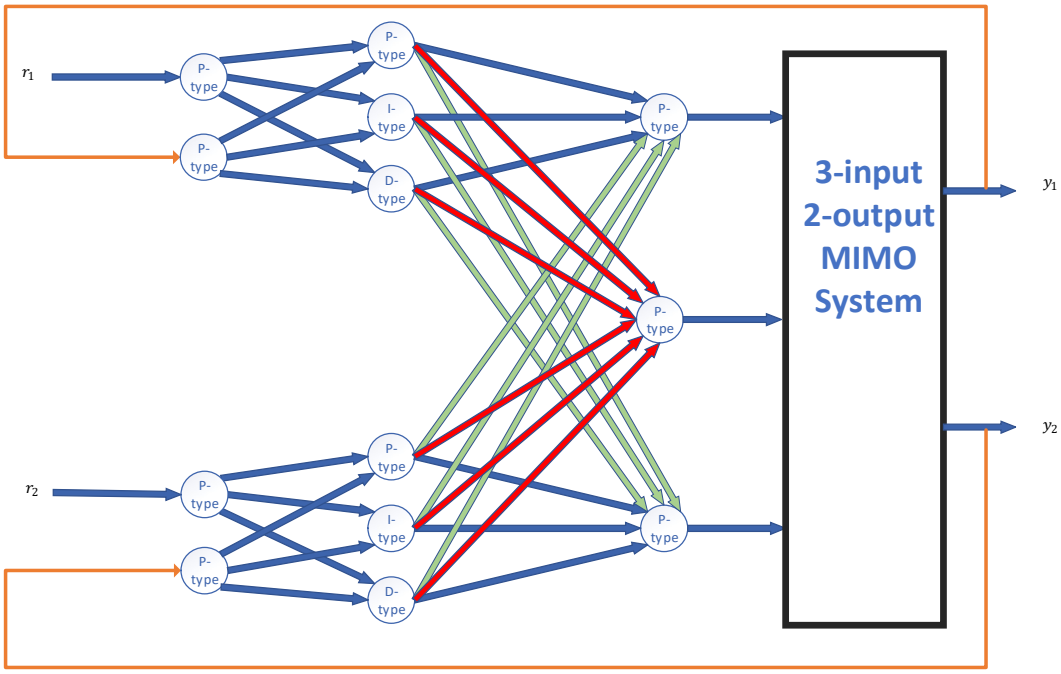


Figure 5-6. MANNCNS controlled 3-input 2-output distillation column plant

The following desired outputs are selected for the distillation column plant.

- $r_1(t) = 2u(t-1) - u(t-500)$
- $r_2(t) = 2u(t-1) - u(t-500)$; which, $u(t)$ is the standard unit step function.

The control objective is for y_1 (Output 1) and y_2 (Output 2) to track, respectively, the set-points r_1 and r_2, with minimum errors, and a set of suitable and converged dynamic weights of the controller.

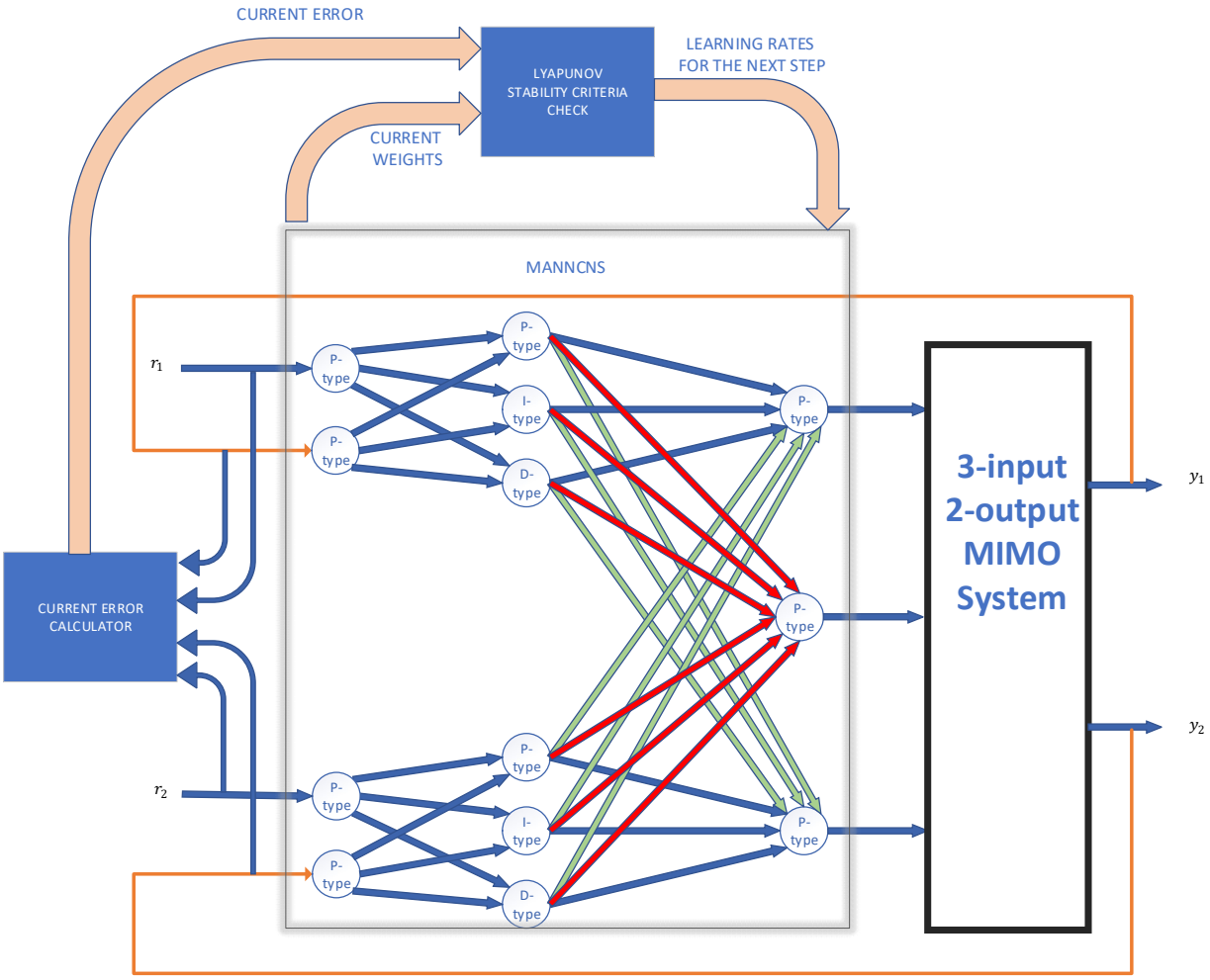


Figure 5-7. Adding stability criteria check to MANNCNS controlled distillation column plant

Due to set-points changes at 500th sample time this simulation will demonstrate the capability of the automatic set-point tracking of the MANNCNS. In order to apply the MANNCNS to the non-square multivariable system, all the initial weights in the output layer are set to one. In the hidden layer, the initial weights associated with the desired outputs are set to positive one and the ones associated with the systems outputs are set to negative one. In order to find the optimal number of weights training it is required to monitor the accumulated errors of the system outputs that are the absolute values of the accumulated difference between the actual outputs and the desired outputs. Figure 5-8 and Figure 5-9 represent, respectively, the accumulated error versus the iteration number of the MANNCNS weight learning algorithm for output 1 and output 2. After running, online learning algorithms presented in Table 5-1 it is observed that, convergence and the suitable performance of the proposed MANNCNS for

the considered coupled 3-input 2-output non-square multivariable system happens after the 27th running of the training algorithm and errors don't change dramatically after that point. By running the training algorithm for subsequent 8 times, the weights after the 35th number of iterations are locked in their values at this point (i.e. optimal point). It should be pointed out that choosing the optimal training number depends on the particular application that the controller is used for. It should indicate a trade-off between the control speed and control performance of the controlled system. It is possible to define a desired error so that when the actual error has reached that value the training process stops, and the weights become frozen until the next change in the system e.g. change in the model of the system or change in any of the set-points.

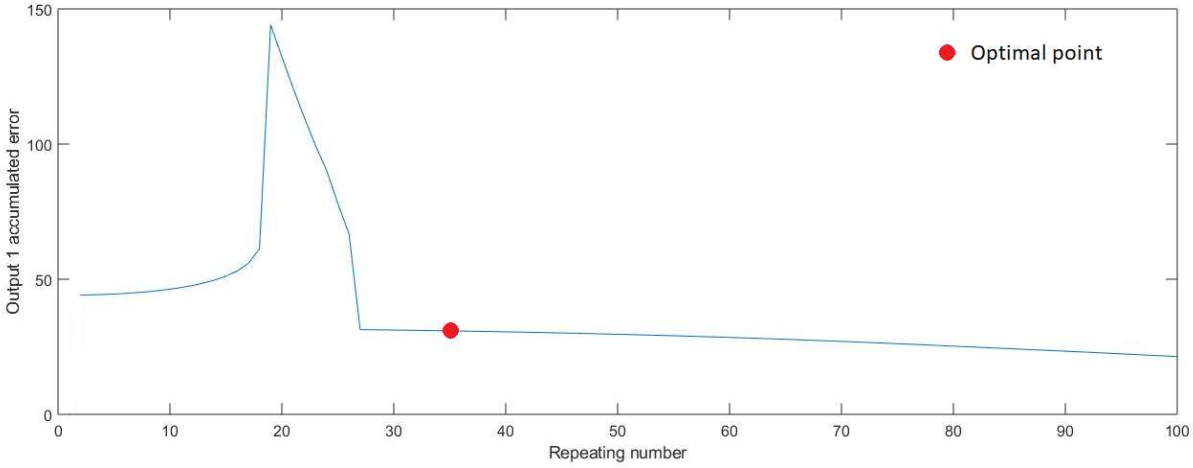


Figure 5-8. Error value for output 1 versus the repeating number

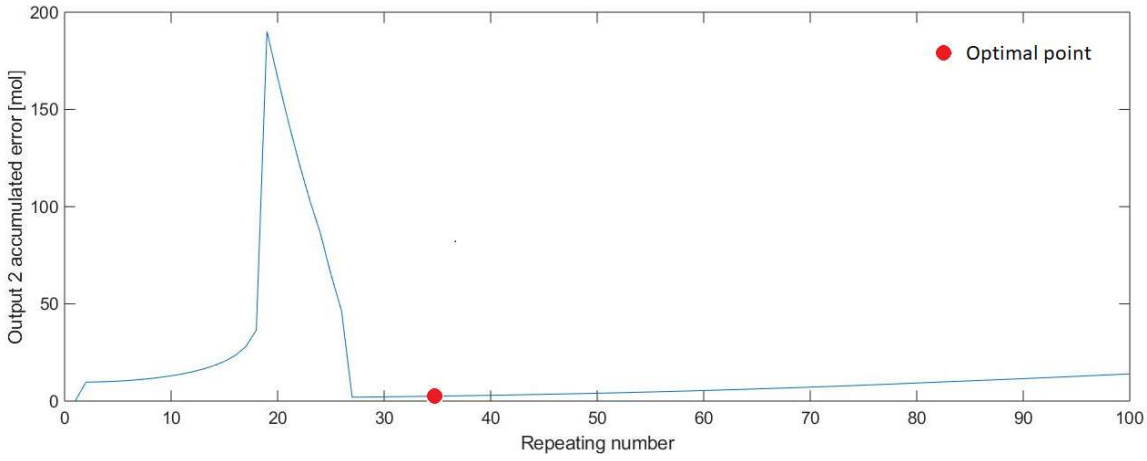


Figure 5-9. Error value for output 2 versus the repeating number

After running, online and simultaneously, the learning algorithms presented in Table 5-1 and the stability criteria presented in Table 5-2 for 35 training steps, the converged values of the dynamic weights of both the output and hidden layers of the MANNC2 are achieved, as summarised in Table 5-3 and Table 5-4.

$w_{h,x}$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$
$h = 1$	1.28	6.55	1.01	1.61	14.66	1.01
$h = 2$	1.53	5.48	1.04	1.40	12.03	1.03
$h = 3$	1.34	4.34	1.02	1.36	9.87	1.02

Table 5-3. Converged MANNCNS output-layer weights' values

$\overline{w_{a,b}}$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
$a = 1$	1.04	-0.96	1.27	-0.93
$a = 2$	0.56	-1.44	0.75	-1.25
$a = 3$	1.21	-0.79	0.95	-1.05

Table 5-4. Converged MANNCNS hidden-layer weights' values

To compare the variation of the adjusted weights, the output-layer weights, and the hidden-layer weights are illustrated as surface diagrams in Figure 5-10 and Figure 5-11, respectively.

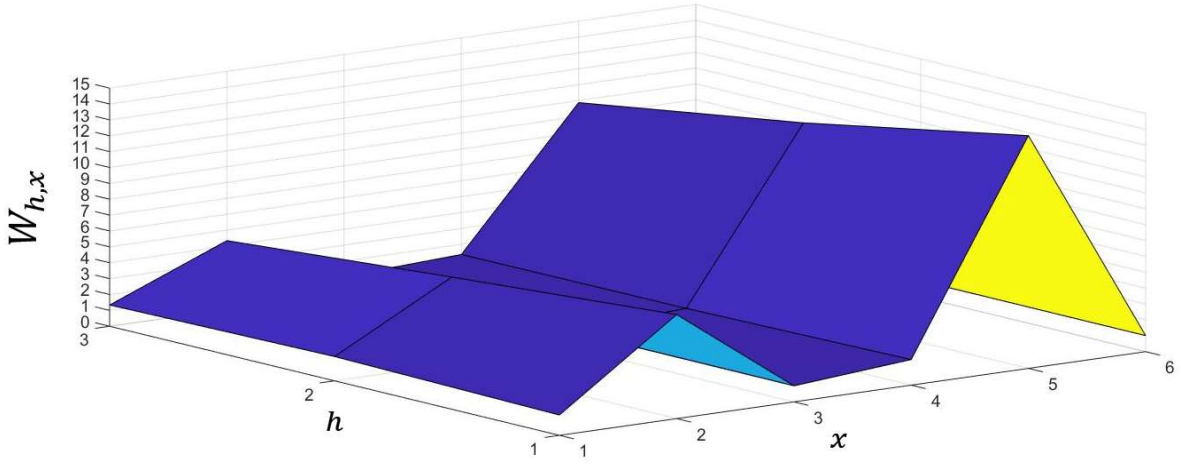


Figure 5-10. Output-layer weights diagram

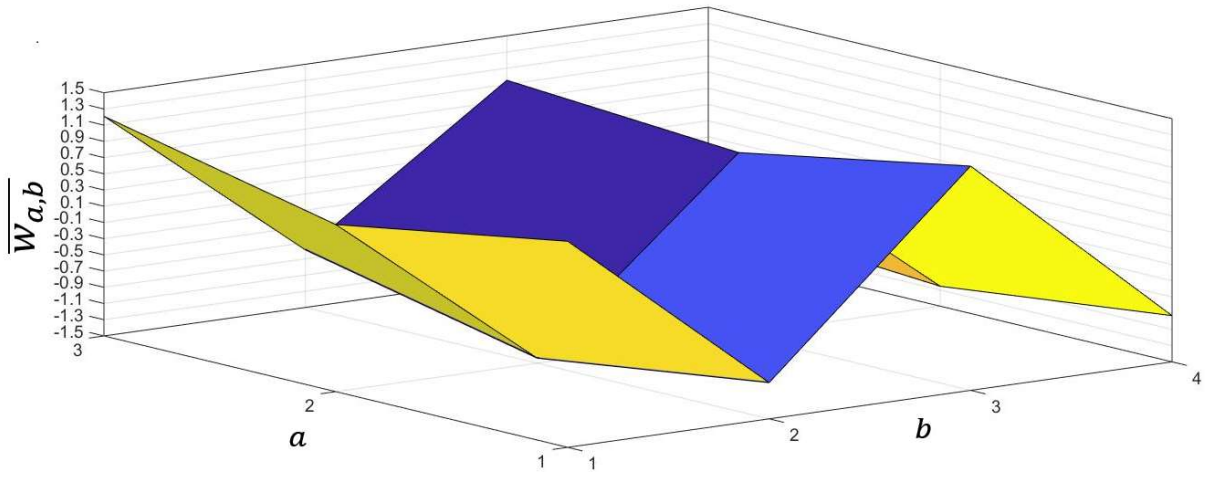


Figure 5-11. Hidden-layer weights diagram

Figure 5-12 represents the variation trend of all 12 weights in the hidden layer during the 35 times weight training process.

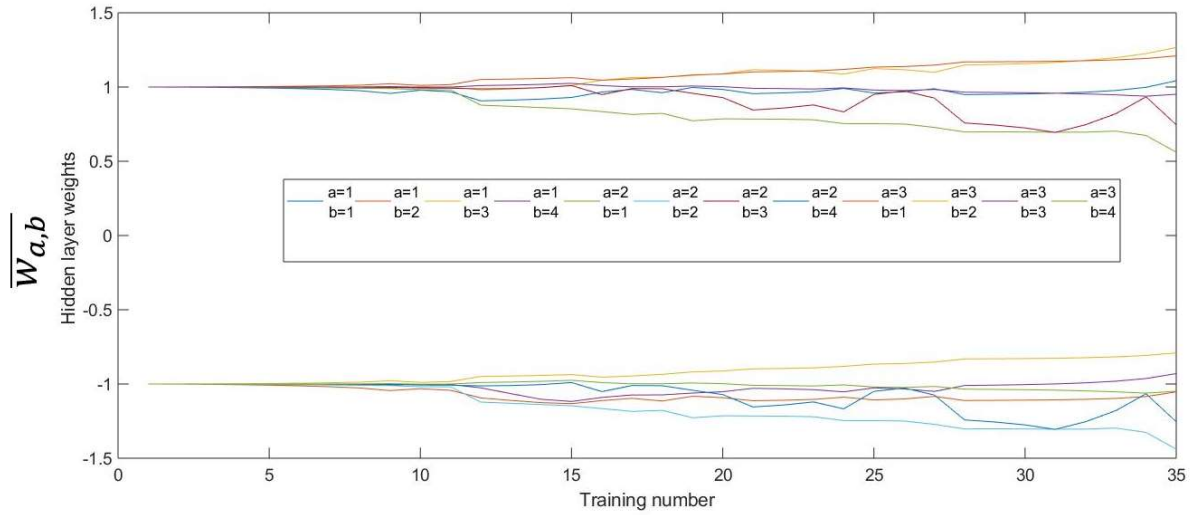


Figure 5-12. Hidden-layer weight adjustment during training process

By running the weight training algorithms and real-time applying updated weights to the output layer and the hidden layer for the controlled system during 35 times of iterations, Figure 5-13 and Figure 5-14 demonstrate the Output 1 and Output 2 of the non-square MIMO controlled system which both track the desired outputs properly with a reasonable settling time (200 sampling time) for the distillation column system. As control of the distillation column is a slow process the achieved settling time is considered as a remarkable result.

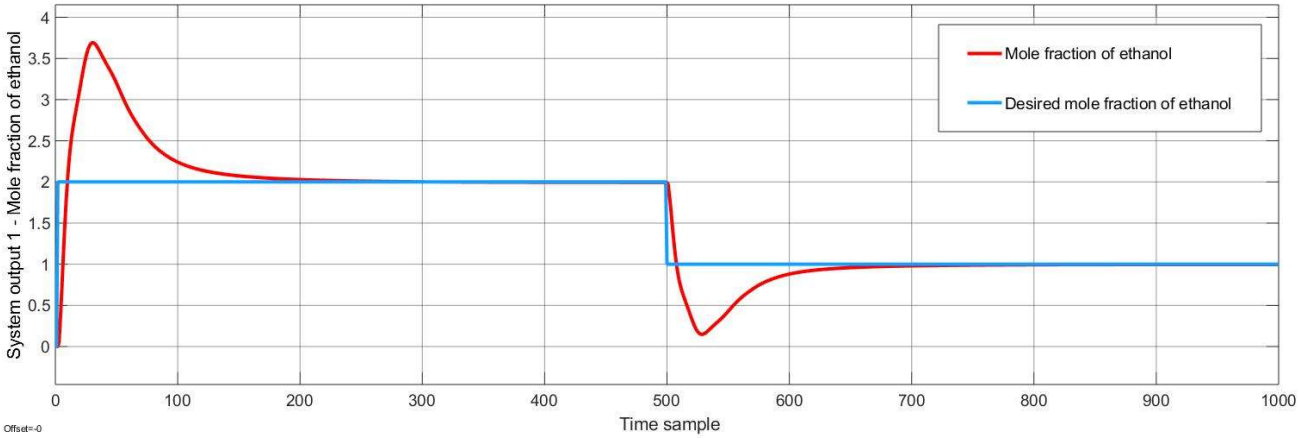


Figure 5-13. Desired output 1 and actual output 1 with MANNCNS

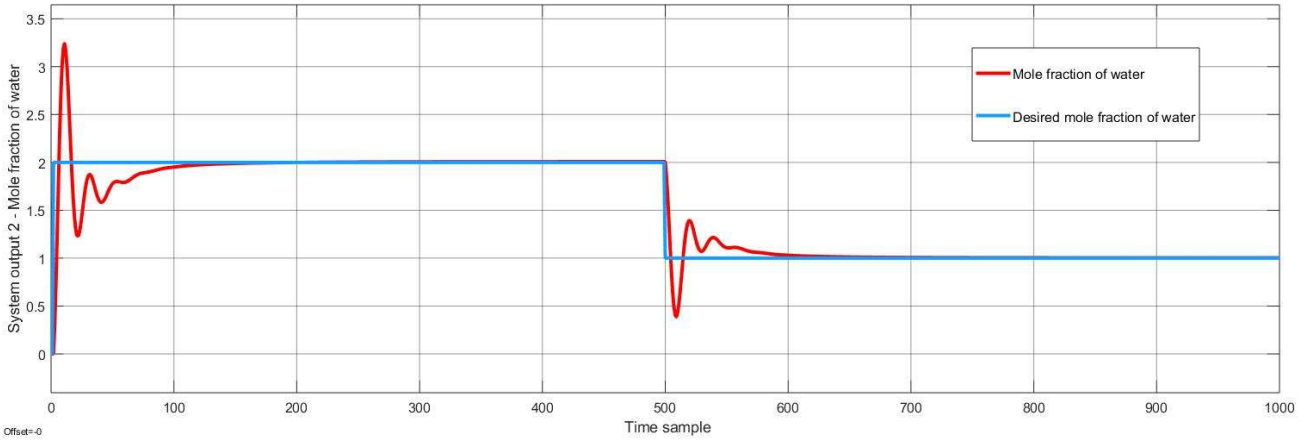


Figure 5-14. Desired output 2 and actual output 2 with MANNCNS

- **Robustness Against Disturbance**

To demonstrate the ability of MANNCNS in terms of disturbance cancelation, three disturbance signals at sample time 80 are applied to all the inputs of the multivariable systems, that are added with the manipulated system's variables with signals below, as Figure 5-15:

$d_1(t) = d_2(t) = d_3(t) = 0.2u(t - 80)$

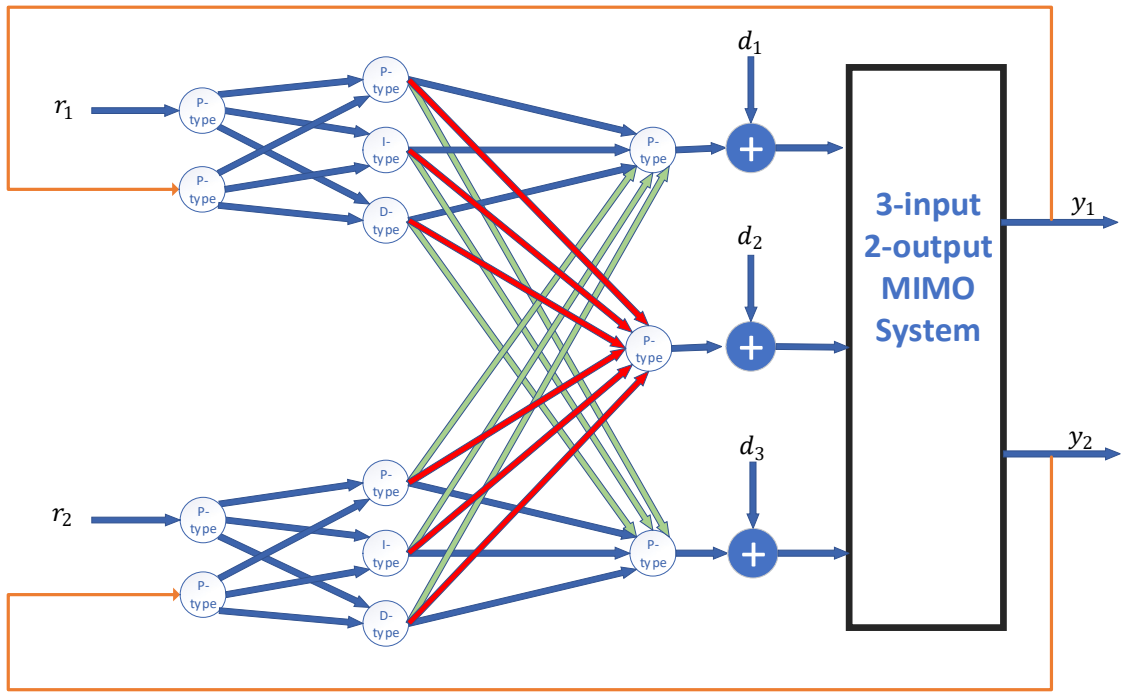


Figure 5-15. Applying disturbance to the system

The simulation result for the mole fraction of water shows the controller is able to re-train itself by the learning algorithm, and the system's outputs follow the desired outputs in a short time frame (60 sample time) after applying the disturbance signals according to Figure 5-16. The final weights after the disturbance cancellation are modified automatically and re-converged according to Table 5-5 and Table 5-6.

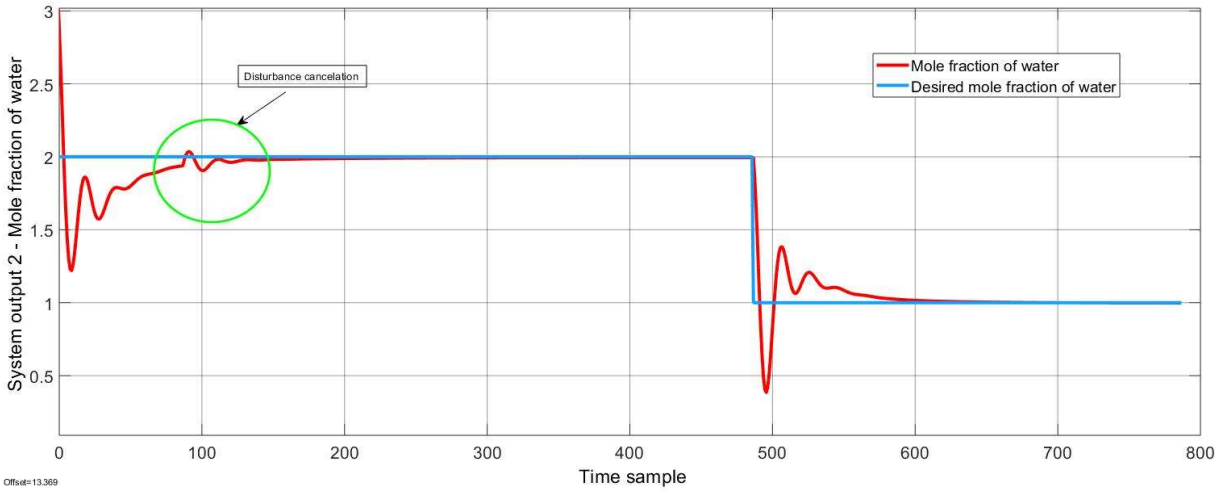


Figure 5-16. Output 2 and desired output 2 after applying disturnace signal

$w_{h,x}$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$
$h=1$	1.33	6.55	0.99	1.63	12.66	1.01
$h=2$	1.56	5.45	1.02	1.42	10.14	1.05
$h=3$	1.25	3.34	1.01	1.35	7.43	1.01

Table 5-5. Output-layer modified weight values after disturbance cancellation

$\overline{w_{a,b}}$	$b=1$	$b=2$	$b=3$	$b=4$
$a=1$	1.03	-0.97	1.17	-0.91
$a=2$	0.55	-1.35	0.69	-1.30
$a=3$	1.22	-0.88	0.92	-1.03

Table 5-6. Hidden-layer modified weight values after disturbance cancellation

5.5.2 Case 2 – Application of MANNCNS on a 2-input 3-output system

In this case, simulation studies are conducted on a highly non-linear coupled 2-input 3-output multivariable system. In this simulation, the ability of the proposed control method is tested to regulate a DC Motor with non-linear behaviour. The simulation results compare the performance of MANNCNS with a direct adaptive neuro-fuzzy control method called RHONN introduced in [15, 16] as the utmost counterpart. The control objective is the speed regulation of a 1KW DC motor designated by the following non-linear equations:

$$T_a \frac{dI_a}{dt} = -I_a - \varphi\omega + V_a \tag{98}$$

$$T_m \frac{d\omega}{dt} = \omega I_a - K_o \omega \tag{99}$$

$$T_f \frac{d\varphi}{dt} = I_f + V_f \tag{100}$$

$$\omega = \frac{aI_f}{1+bI_f} \tag{101}$$

where I_a, ω, and φ respectively are armature current, angular speed and stator flux of the DC motor which are the controlled system outputs. The system's manipulated variables are V_a and V_f which should be generated by the controller. The following constant values in Table 5-7 have been considered for the DC motor's parameters:

Parameter	Constant value
$1/T_a$	148.88 sec^{-1}
$1/T_m$	42.91 sec^{-1}
K_o/T_a	0.0129 sec^{-1}
T_f	31.88 sec
a	2.6
b	1.6

Table 5-7. DC motor parameters

The regulation problem of the DC motor is interpreted as finding a state feedback, in which could force the angular velocity and the armature current I_a to go to zero or very close to zero, while the modelling errors exists, and the magnetic flux varies slowly [25]. The functional block diagram of MANNCNS controller applied to the multivariable non-square DC motor system is shown in Figure 5-17.

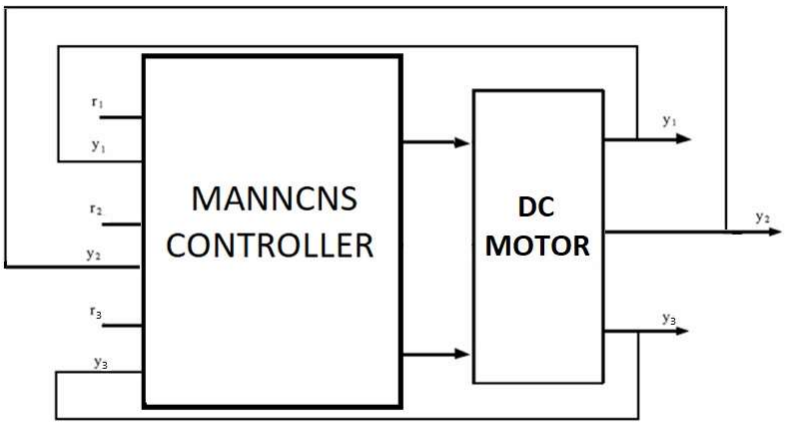


Figure 5-17. Block diagram of DC motor system and MANNCNS controller

Neural network structure of a MANNCNS for controlling a 2-input 3-output system is demonstrated in Figure 5-18. The green and blue connection in the hidden layer are used to enable the controller to handle cross-coupling between the inputs and outputs of the system. In this structure, there are totally 18 weights in the hidden layer and 18 weights in the output layer of the MANNCNS to be adjusted.

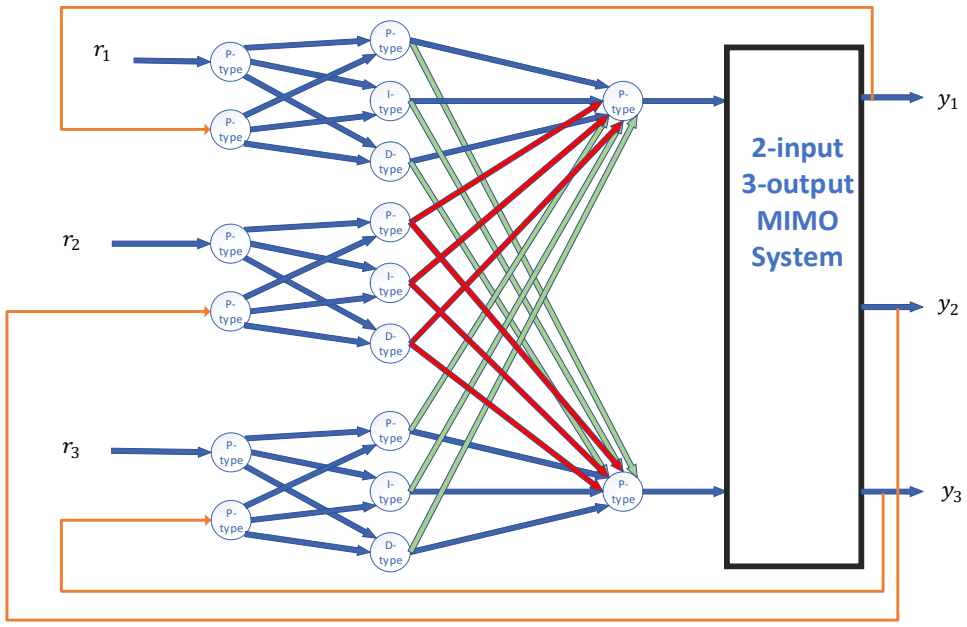


Figure 5-18. MANNCNS controlled 2-input 3-output DC motor system

The Lyapunov stability criteria illustrated in Table 5-2 have been applied to the controlled system to check and adjust the learning rates during the weight training algorithm demonstrated in Figure 5-19.

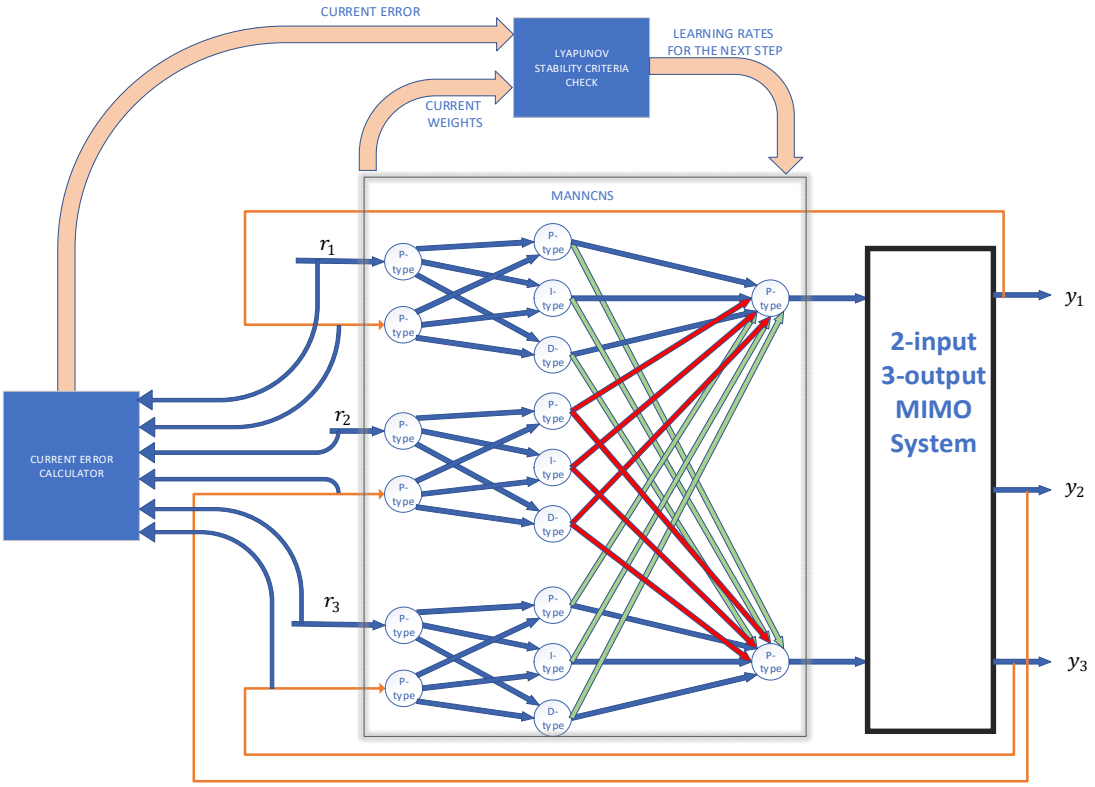


Figure 5-19. Adding stability criteria check to the 2-input 3-output controlled system

By choosing the number of samples as $m = 100$ and also the learning rates in Table 5-1 as $\lambda_h = 0.2$ and $\lambda_f = 0.2$, respectively, for $h = 1, 2, \ldots, \mathrm{p}$ and $f = 1, 2, , \mathrm{q}$, after 8 milliseconds the system becomes marginally stable. Therefore, all of the learning rates have been reduced to 0.1 to avoid the control system falling in the local minimum. The following response in Figure 5-20 for the motor speed has been obtained for MANNCNS after 60 times iteration of the learning algorithm. In this graph the response related to an adaptive control method counterpart called RHONN which is a neuro-fuzzy solution is also shown. It is observed that by using MANNCNS, the motor speed converges to zero faster when compared to the RHONN adaptive algorithm. Also, by using MANNCNS unlike the RHONN method, there is no rise or peak in the motor speed during the control process which is a significant result. In addition, the accumulated error of the motor speed is calculated for up to 200 trainings. The error versus the number of iterations is illustrated in Figure 5-21. In this graph it is evident that after 60 times training, the accumulated error is improved significantly. Hence, the optimal number of trainings in this case study is set to 60 and the weights are locked at this point.

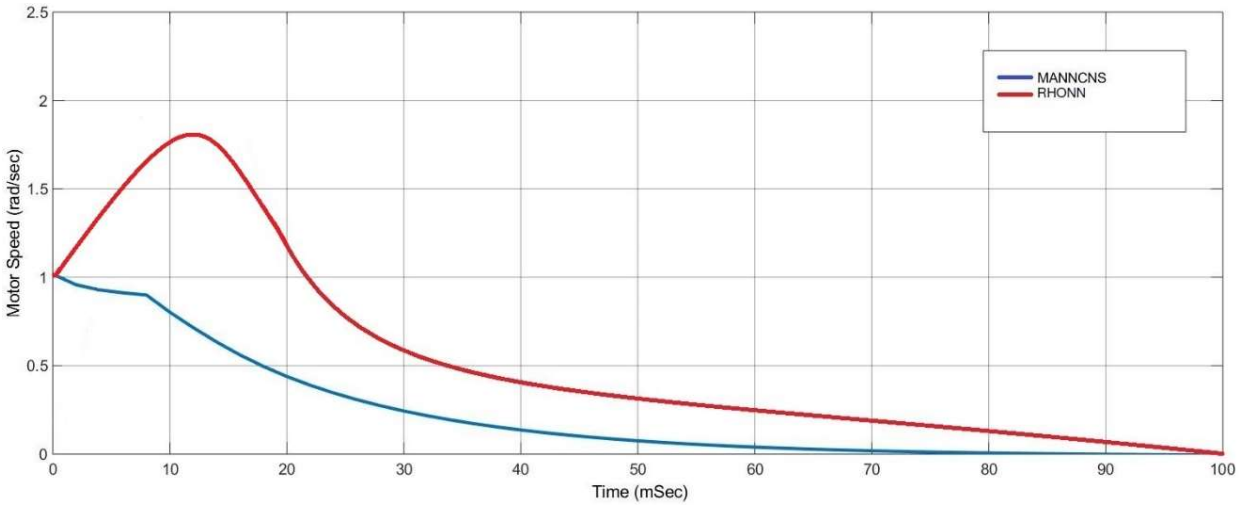


Figure 5-20. Motor speed by MANNCNS and RHONN

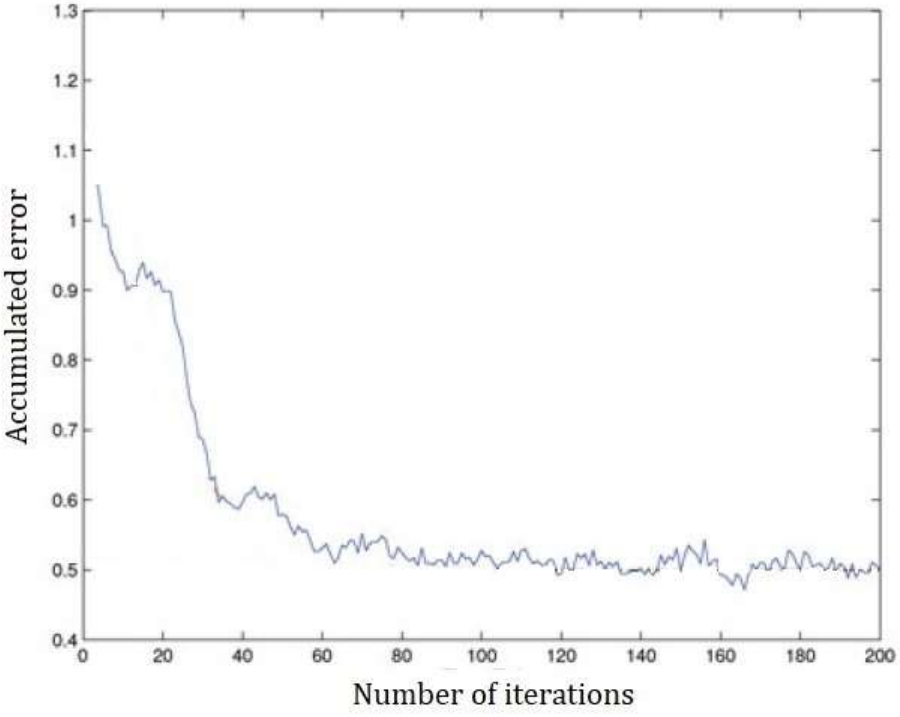


Figure 5-21. Motor speed accumulated error versus the number of iterations

5.6 Conclusion

By the use of flexible topology of neural networks in providing non-square Input-Output structures and powerful self-learning abilities inherent in a neural network strategy, this paper demonstrates that the proposed MANNCNS is capable of controlling model-free non-square coupled MIMO. In this design, the MANNCNS contains two layers with adjustable weights in its neural network structure and uses a modified auto-tune dynamic online learning algorithm with accumulated error back-propagation for the proposed neural network structure, and effectively tunes its weights in the output layer as well as the hidden layer simultaneously to achieve the desirable control outcomes. The effectiveness of the proposed MANNCNS is validated via simulation studies for a famous industrial application of distillation column known as a non-square MIMO system when the number of inputs is more than the number of outputs. In this example, the capability of the proposed controller for dealing with the deadtime and also affected by unwanted disturbances is tested and the results demonstrate that the controller can adapt itself appropriately in these circumstances and the outputs follow their desired values in case of having

unwanted situations. Likewise, MANNCNS is tested for speed regulation of a DC motor which is identified as a 2-input 3-output MIMO system (number of outputs is more than the number of inputs) with highly non-linear characteristics. The controller's response is compared with its best counterpart that used a neuro-fuzzy algorithm. The simulation results demonstrate that the armature current response is improved significantly by using MANNCNS. Overall, it is demonstrated that the MANNCNS is a suitable candidate for many industrial applications.

References

[1] *Kalpana, D., T. Thyagarajan, and N. Gokulraj, Modeling and control of non-square MIMO system using relay feedback. ISA Transactions, 2015. 59: p. 408-417.*

[2] *S. Bhat Vinayambika, S.S.P., and I. Thirunavukkarasu, A Comparative Study on Control Techniques of Non-square Matrix Distillation Column. International Science Press, 2015: p. 1129-1136.*

[3] *Reeves, D.E. and Y. Arkun, Interaction measures for nonsquare decentralized control structures. AIChE Journal, 1989. 35(4): p. 603-613.*

[4] *Vlachos, C., D. Williams, and J.B. Gomm, Solution to the Shell standard control problem using genetically tuned PID controllers. Control Engineering Practice, 2002. 10(2): p. 151-163.*

[5] *Tan, J.L.a.N.C.a.X.Y.a.S., Modified Internal Model Control for Non-square Systems Based on Smith Delay Compensator Control. Sensors & Transducers, 2014. 165(2): p. 96-101.*

[6] *Muske, K., et al., Crude unit product quality control. Computers & Chemical Engineering, 1991. 15(9): p. 629-638.*

[7] *Harmon Ray, W., Process Dynamics, Modeling, and Control / B.A. Ogunnaike, W.H. Ray. 2018.*

[8] *Morari, M., Grimm, W., Ogelsby, M. J., & Prosser, I. D., Design of resilient processing plants. VIII: Design of energy measurement systems for unstable plants - new insights. Chemical Engineering Science, 1985. 40(187).*

[9] *Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.*

[10] *Bu, X., et al., Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information. Asian Journal of Control, 2018. 20(2): p. 962-968.*

[11] *Xie, C.H. and G.H. Yang, Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults. International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.*

[12] *Fliess, M. and C. Join, MODEL-FREE CONTROL AND INTELLIGENT PID CONTROLLERS: TOWARDS A POSSIBLE TRIVIALIZATION OF NONLINEAR CONTROL? IFAC Proceedings Volumes, 2009. 42(10): p. 1531-1550.*

[13] *Roman, R.C., et al. Data-driven optimal model-free control of twin rotor aerodynamic systems. in 2015 IEEE International Conference on Industrial Technology (ICIT). 2015.*

[14] *Al-Dulaimi, A., et al., A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. Computers in Industry, 2019. 108: p. 186-196.*

[15] *Ansari, A.Q., N.K. Gupta, and Ekata. Adaptive neurofuzzy system for tuberculosis. in 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing. 2012.*

[16] *Theodoridis, D., M. Christodoulou, and Y. Boutalis. Direct adaptive control of unknown multi-variable nonlinear systems with robustness analysis using a new neuro-fuzzy representation and a novel approach of parameter hopping. in 2009 17th Mediterranean Conference on Control and Automation. 2009.*

[17] *Arash Mehrafrooz, F.H., Modification of Model-Free Multivariable Adaptive Neural Network Controller Using Two Dynamic Layers for Controlling Nonlinear Square MIMO Systems. IEEE Transactions on Control Systems Technology, 2019.*

[18] *Arash Mehrafrooz, F.H., Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems. IEEE Transactions on Automatic Control (Submitted), 2019.*

[19] *Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.*

[20] *Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.*

[21] Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.

[22] Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.

[23] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[24] Chidambaram, P.G.a.M., Multivariable Controller Tuning for Non-square Systems with RHP Zeros by Genetic Algorithm," Chemical and Biochemical Engineering. 2010. 24: p. 17-22.

[25] Farnaz, Z., et al., DC Motor Torque Control Using State Estimation. Vol. 44. 2016.

CHAPTER 6: CONCLUSION AND SUGGESTIONS FOR FUTURE WORKS

In this thesis, a step by step design process of a universal controller based on neural networks is described. The aim is defined as designing a controller for square and non-square multi-input multi-output systems which is able to handle non-linearity, coupling, and time variance in the system. Therefore, the control method is designed in a way that the learning algorithms and stability analysis are independent from the model of the system and work as a model-free contro method.

To achieve the above-mentioned purpose, firstly in chapter 2, with the powerful learning abilities inherent in a neural network strategy, an Adaptive Neural Network Controller (ANNC) is introduced which is capable of controlling single-input single-output (SISO) systems with nonlinear properties. The ANNC uses a new auto-tune dynamic online learning algorithm with accumulated error back-propagation for the proposed neural network structure, and effectively tunes its parameters to achieve the desirable control outcomes. By checking the stability of the system in each step of the controller parameters' update, the closed-loop system is guaranteed to remain stable during the entire controller training process. The effectiveness of the proposed ANNC is validated via simulation studies. When compared with the representatives of existing counterparts, the ANNC is seen to provide a better and faster system performance. By selecting an appropriate number of samples, the ANNC can be effectively used for several types of control systems in industrial applications, such as flow, level, and temperature control systems in water or wastewater plants. Although the proposed ANNC currently discussed in chapter 2 is in the context of SISO systems and PID control strategies, it is developed in such a way that the proposed structure can establish a foundation upon which new adaptive neural network methods for MIMO coupled systems and/or other types of control laws can be readily derived. The aspects of those studies are revealed in chapter 3 to 5 when the controller is generalized to apply to multi-input multi-output systems.

In chapter 3, the proposed Multivariable Adaptive Neural Networks Controller (MANNC), using the fundamental structure proposed in chapter 2, is capable of controlling coupled square MIMO systems. The MANNC uses a new auto-tune dynamic online learning algorithm with accumulated error back-propagation for the proposed neural network structure, and effectively tunes its weights to achieve the desirable control outcomes. The effectiveness of the proposed MANNC is validated via three significant simulation studies i.e. a drum boiler non-linear time invariant system, an example of a nonlinear time-variant system and a nonlinear 2-tank hybrid system. When compared with the representatives of existing counterparts, the MANNC is seen to provide smaller overshoots and faster system performance with better and more reliable setpoint tracking. The stability of the system and controller is analysed by Lyapunov global asymptotic stability theorem and it leads to stability criteria for the controller parameters and must be checked online while the controller's weights are getting trained. MANNC is generally suitable for square coupled multivariable systems, however when it comes to non-linear system control, because of having only one adjustable layer in its neural networks structure, it does not show the best performance as it has for linear systems or systems with less non-linear characteristics. Therefore, this matter leads us to develop the controller by adding another layer to the neural network structure of that in chapter 4.

In chapter 4, the proposed Multivariable Adaptive Neural Networks Controller with two adjustable layers (MANNC2) is introduced which is capable of controlling square coupled MIMO systems with higher non-linear characteristics. When compared with the MANNC, the MANNC2 is seen to provide much smaller undershoots, faster system performance, and better setpoint tracking which results in a significant reduction in errors. The stability of the system and controller is analysed by Lyapunov global asymptotic stability theorem and leads to new stability criteria for the controller parameters in both layers which must be checked online while the controller's weights are getting trained. In case of possibility of instability in the system, the learning rates can be updated automatically to prevent the controlled system from falling in local minimum. Having two adjustable layers in the neural networks structure of the controller

improves the results in the simulation studies. The limitation of the MANNC2 is that this controller is only applicable to square multi-input multi-output systems, however, in the industry it is highly common to deal with non-square multivariable systems. Hence, the concept of generalizing and modifying the MANNC2 controller to apply to non-square multi-input multi-output systems is introduced as the topic of Chapter 5.

In chapter 5, the final product of this study is presented. A Multivariable Adaptive Neural Networks Controller for Non-Square systems (MANNCNS) is introduced along with its automatic training algorithm and online real-time Lyapunov stability criteria check. This controller has two adjustable layers in its neural network structure, and it is applicable to non-square multivariable systems with high non-linear characteristics. The effectiveness of the proposed MANNCNS is validated via simulation studies for a well-known industrial application of distillation column known as a non-square MIMO system, of which the number of inputs is more than the number of outputs. In this example, the capability of the proposed controller has been tested on the systems with deadtime and also how it can be affected by unwanted disturbances. The results show the controller can adapt itself appropriately and the outputs follow their desired values in case of having unwanted disturbance. Similarly, MANNCNS has been tested for speed regulation of a DC motor which is identified as a 2-input 3-output MIMO system (the number of outputs is more than the number of inputs) with non-linear characteristics. The controller's responses have been compared with ones of the controller's best counterpart which used a neuro-fuzzy algorithm and it is demonstrated that the armature current response is significantly improved by using MANNCNS.

In this study, various applications in simulation studies are selected in order to cover an extensive range of significant problems in control theory. All in all, in several simulations in previous chapters, it is demonstrated that the proposed universal controller has the capacity to control a wide range of multi-input multi-output systems. From the given simulation studies, it is evident that the controller can cope with various system's properties such as time-variance and

time in-variance, non-linearity, continuous and discrete dynamics, hybrid systems, square and non-square systems.

The following studies are proposed with regards to the future research related to this project:

- *Implementation of the introduced control methods on programmable logic controllers (PLCs) and distributed control systems (DCSs) by converting the programs form Matlab to industrial control software packages*; the controller can be programmed as a function block diagram by the defined different features such as number of inputs and number of outputs, coupled or uncoupled inputs and outputs, square or non-square systems, learning rates, number of sample time, and number of iterations. Matlab recently introduced Simulink PLC Coder™ which produces hardware-independent Ladder Diagrams (LD) and IEC 61131-3 Structured Text (ST) from Simulink models, Stateflow charts, and MATLAB Simulink functions. The ladder diagrams and structured text are produced in PLCopen XML and other file formats supported by Integrated Development Environments (IDEs) including 3S-Smart Software Solutions CODESYS, Rockwell Automation Studio 5000, Siemens TIA Portal, and Omron Sysmac Studio. As a result, it is possible to compile and deploy the designed controllers to numerous programmable logic controller (PLC) and programmable automation controller (PAC) devices. Additionally, Simulink PLC Coder generates test benches that can be used to verify the Structured Text and Ladder Diagrams using PLC and PAC IDEs and simulation tools. It can also provide code generation reports with static code metrics and bidirectional traceability between the model and code. Despite conversion of Matlab code to a PLC/DCS program and also its implementation on an industrial control system could be theoretically performed on different PLC/DCS platforms from various vendors, it should be noted that for this purpose, proper hardware considerations must be taken into account, e.g. utilised CPUs should be able to handle high volume of parallel computations in MIMO neural network structure of the controller.

- *Increasing number of adjustable layers in the neural networks structure to observe whether it can improve the performances of the system by monitoring the control speed and setpoint tracking;* in the current project it is found that two adjustable layers had a better control performance compared to only one adjustable layer especially for controlling systems with highly nonlinear characteristics. Therefore, it could be further investigated that whether having three layers or more in the neural networks can improve the performance further. However, it is known that increasing the number of layers would exponentially increase the number of weights in the neural network topology to be trained. As a drawback, the control action would last for a longer time that should be considered in future studies especially in applications that real-time control process is essential. In addition, adding more hidden layers to the neural network structure of the controllers would make the stability analysis more complicated. Checking the stability criteria during the control action in an online manner could be a significant obstacle that has to be resolved in this study.

- *Speeding up the adaptive training algorithms using recently developed techniques in machine learning;* some techniques in neural networks can be tested such as "deep networks with stochastic depth", "progressively freezing layers" and "omitting weights associated with the non-cross-coupling inputs and outputs". This could be useful for the applications where the time of control is more critical, and it is not possible to reduce the number of trainings, because the error has not reached to its desired value. In addition to this, using dynamic learning rates during running of the learning algorithm can be tested. In this approach, reasonably large learning rates can be chosen at the beginning to speed up the learning algorithm process and gradually decrease them to lower the risk of instability of the controlled system.

- *Study on using different activation functions in the neural network structure and test the performance and compare between different applications;* by considering the flexibility of neural networks in using various types of activation functions as well as defining new ones, it is recommended to try other activation functions in the neural network structure of the introduced controller. The proposed new activation functions can be achieved by defining new limitations for the current activation functions, double differentiation, and double integration.

- *Modify the proposed controllers to apply to different industrial applications;* it must be pointed out that the controllers are designed as a general framework for neural network control of black-box systems, thus the controllers' structures can be modified to control specific applications of square and non-square MIMO systems which are difficult to be controlled by the other advanced control methods.

Bibliography

References for Chapter 1

[1] *Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.*

[2] *Wang, S., W. Chaovalitwongse, and R. Babuska, Machine Learning Algorithms in Bipedal Robot Control. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012. 42(5): p. 728-743.*

[3] *Xu, X., et al., Online Learning Control Using Adaptive Critic Designs With Sparse Kernel Machines. IEEE Transactions on Neural Networks and Learning Systems, 2013. 24(5): p. 762-775.*

[4] *Mozaffari-Kermani, M., et al., Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare. IEEE Journal of Biomedical and Health Informatics, 2015. 19(6): p. 1893-1905.*

[5] *Fadlullah, Z.M., et al., State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. IEEE Communications Surveys & Tutorials, 2017. 19(4): p. 2432-2455.*

[6] *Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.*

[7] *Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.*

[8] *Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.*

[9] *Previdi, F., et al., Data-driven control design for neuroprotheses: a virtual reference feedback tuning (VRFT) approach. IEEE Transactions on Control Systems Technology, 2004. 12(1): p. 176-182.*

[10] *Oomen, T., et al., Iterative Data-Driven${\cal H}_{\infty}$Norm Estimation of Multivariable Systems With Application to Robust Active Vibration Isolation. IEEE Transactions on Control Systems Technology, 2014. 22(6): p. 2247-2260.*

[11] *Radac, M.B., et al. Data-driven model-free control of twin rotor aerodynamic systems: Algorithms and experiments. in 2014 IEEE International Symposium on Intelligent Control (ISIC). 2014.*

[12] *Chi, R., et al., Enhanced Data-Driven Optimal Terminal ILC Using Current Iteration Control Knowledge. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2939-2948.*

[13] *Chi, R., et al., Enhanced Data-Driven Optimal Terminal ILC Using Current Iteration Control Knowledge. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2939-2948.*

[14] *Roman, R.C., et al. Data-driven optimal model-free control of twin rotor aerodynamic systems. in 2015 IEEE International Conference on Industrial Technology (ICIT). 2015.*

[15] *Roman, R.C., et al. Two data-driven control algorithms for a MIMO aerodynamic system with experimental validation. in 2015 19th International Conference on System Theory, Control and Computing (ICSTCC). 2015.*

[16] *Liu, S., Z. Hou, and J. Zheng. Attitude adjustment of quadrotor aircraft platform via a data-driven model free adaptive control cascaded with intelligent PID. in 2016 Chinese Control and Decision Conference (CCDC). 2016.*

[17] *Chi, R., et al., An Improved Data-Driven Point-to-Point ILC Using Additional On-Line Control Inputs With Experimental Verification. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017: p. 1-10.*

[18] *Pang, Z., et al., Data-Driven Control With Input Design-Based Data Dropout Compensation for Networked Nonlinear Systems. IEEE Transactions on Control Systems Technology, 2017. 25(2): p. 628-636.*

[19] *Aumi, S., et al., Data-Based Modeling and Control of Nylon-6, 6 Batch Polymerization. IEEE Transactions on Control Systems Technology, 2013. 21(1): p. 94-106.*

[20] *Pang, Z., et al., Data-Based Predictive Control for Networked Nonlinear Systems With Network-Induced Delay and Packet Dropout. IEEE Transactions on Industrial Electronics, 2016. 63(2): p. 1249-1257.*

[21] *Wang, D., et al., Data-Based Adaptive Critic Designs for Nonlinear Robust Optimal Control With Uncertain Dynamics. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016. 46(11): p. 1544-1555.*

[22] *Luo, B., et al., Policy Gradient Adaptive Dynamic Programming for Data-Based Optimal Control. IEEE Transactions on Cybernetics, 2017. 47(10): p. 3341-3354.*

[23] *Fliess, M. and C. Join, MODEL-FREE CONTROL AND INTELLIGENT PID CONTROLLERS: TOWARDS A POSSIBLE TRIVIALIZATION OF NONLINEAR CONTROL? IFAC Proceedings Volumes, 2009. 42(10): p. 1531-1550.*

[24] *Fliess, M. and C. Join, Model-free control. International Journal of Control, 2013. 86(12): p. 2228-2252.*

[25] *Lafont, F., et al., A model-free control strategy for an experimental greenhouse with an application to fault accommodation. Computers and Electronics in Agriculture, 2015. 110: p. 139-149.*

[26] *Madadi, E. and D. Söffker, Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control. 2015(57243): p. V001T03A004.*

[27] *Radac, M.B., R.E. Precup, and E.M. Petriu, Model-Free Primitive-Based Iterative Learning Control Approach to Trajectory Tracking of MIMO Systems With Experimental Validation. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2925-2938.*

[28] *Roman, R.C., M.B. Radac, and R.E. Precup, Multi-input–multi-output system experimental validation of model-free control and virtual reference feedback tuning techniques. IET Control Theory & Applications, 2016. 10(12): p. 1395-1403.*

[29] *Gao, B., et al. Model-free adaptive MIMO control algorithm application in polishing robot. in 2017 6th Data Driven Control and Learning Systems (DDCLS). 2017.*

[30] *Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.*

[31] *Safonov, M.G. and T. Tung-Ching, The unfalsified control concept and learning. IEEE Transactions on Automatic Control, 1997. 42(6): p. 843-847.*

[32] *Helvoort, J.v., B.d. Jager, and M. Steinbuch, Data-Driven Controller Unfalsification With Analytic Update Applied to a Motion System. IEEE Transactions on Control Systems Technology, 2008. 16(6): p. 1207-1217.*

[33] *Battistelli, G., et al., Stability of Unfalsified Adaptive Switching Control in Noisy Environments. IEEE Transactions on Automatic Control, 2010. 55(10): p. 2424-2429.*

[34] *Bianchi, F.D., et al., Fault-Tolerant Unfalsified Control for PEM Fuel Cell Systems. IEEE Transactions on Energy Conversion, 2015. 30(1): p. 307-315.*

[35] Heertjes, M.F., B.V.d. Velden, and T. Oomen, Constrained Iterative Feedback Tuning for Robust Control of a Wafer Stage System. IEEE Transactions on Control Systems Technology, 2016. 24(1): p. 56-66.

[36] Meng, W., et al., Robust Iterative Feedback Tuning Control of a Compliant Rehabilitation Robot for Repetitive Ankle Training. IEEE/ASME Transactions on Mechatronics, 2017. 22(1): p. 173-184.

[37] Campi, M.C. and S.M. Savaresi, Direct nonlinear control design: the virtual reference feedback tuning (VRFT) approach. IEEE Transactions on Automatic Control, 2006. 51(1): p. 14-27.

[38] Bu, X., et al., Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information. Asian Journal of Control, 2018. 20(2): p. 962-968.

[39] Khadraoui, S., et al., Adaptive Controller Design for Unknown Systems Using Measured Data. Asian Journal of Control, 2016. 18(4): p. 1453-1466.

[40] Xie, C.H. and G.H. Yang, Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults. International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.

[41] Cui, X., et al., Adaptive dynamic programming for tracking design of uncertain nonlinear systems with disturbances and input constraints. International Journal of Adaptive Control and Signal Processing, 2017. 31(11): p. 1567-1583.

[42] Bhatti, S.A., S.A. Malik, and A. Daraz. Comparison of P-I and I-P controller by using Ziegler-Nichols tuning method for speed control of DC motor. in 2016 International Conference on Intelligent Systems Engineering (ICISE). 2016.

[43] Azman, A.A., et al. Modeling and comparative study of PID Ziegler Nichols (ZN) and Cohen-Coon (CC) tuning method for Multi-tube aluminum sulphate water filter (MTAS). in 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS). 2017.

[44] Anto, E.K., J.A. Asumadu, and P.Y. Okyere. PID control for improving P&O-MPPT performance of a grid-connected solar PV system with Ziegler-Nichols tuning method. in 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA). 2016.

[45] Ding, X., et al. A Fuzzy Neutral Network Controller Based on Optimized Genetic Algorithm for UC Rolling Mill. in 2009 Second International Symposium on Knowledge Acquisition and Modeling. 2009.

[46] Liu, W. and L. Yu. Motion Controller Design of DR Camera's UC Arm. in 2013 Sixth International Symposium on Computational Intelligence and Design. 2013.

[47] Radac, M.B., et al., Application of IFT and SPSA to Servo System Control. IEEE Transactions on Neural Networks, 2011. 22(12): p. 2363-2375.

[48] Liu, Q., et al., Iteration Tuning of Disturbance Observer-Based Control System Satisfying Robustness Index for FOPTD Processes. IEEE Transactions on Control Systems Technology, 2017. 25(6): p. 1978-1988.

[49] Huang, Z., et al. Boost converter optimal control based on MFAC and FPSOA under model mismatch. in 2016 Chinese Control and Decision Conference (CCDC). 2016.

[50] Shangtai, J. and H. Mengxue. An improved full-form-dynamic-linearization based MFAC for a class of nonlinear systems with exogenous disturbance. in 2015 34th Chinese Control Conference (CCC). 2015.

[51] Zhang, X., Z. Hou, and S. Liu. Robust model free adaptive control for a class of nonlinear MIMO systems with measurement noise and data dropout. in 2017 11th Asian Control Conference (ASCC). 2017.

[52] Meihui, L., et al. Greenhouse multi-variables control by using feedback linearization decoupling method. in 2017 Chinese Automation Congress (CAC). 2017.

[53] Dache, C.R., et al. Linearized model of the variable flux induction motor drive. in 2016 International Conference and Exposition on Electrical and Power Engineering (EPE). 2016.

[54] *Åström, K.J., K. Johansson, and Q.-G. Wang, Design of decoupled PID controllers for MIMO systems. Vol. 3. 2001. 2015-2020 vol.3.*

[55] *Shu, H. and H. Guo, Decoupling control of multivariable time-varying systems based on PID neural network, in 2004 5th Asian Control Conference (IEEE Cat. No.04EX904). 2004, Department of Information and Control Engineering, Guangzhou University, China: Melbourne, Victoria, Australia, Australia.*

[56] *Zhang, K., et al., MIMO Evolution Model-Based Coupled Fault Estimation and Adaptive Control With High-Speed Train Applications. IEEE Transactions on Control Systems Technology, 2017. PP(99): p. 1-15.*

[57] *Astrom, K.J. and T. Hagglund, PID Controllers: Theory, Design and Tuning. 2 ed. 1995, USA: Instrument Society of America.*

[58] *Astrom, K.J. and B. Wittenmark, Adaptive Control Stability Convergence and Robustness. 1989, University of California: Prentice Hall Information and System Science Series.*

[59] *Farivar, F., et al., Hybrid Control of Flexible Manipulator. Journal of Applied Sciences, 2009. 9(4): p. 639-650.*

[60] *Kostarigka, A.K. and G.A. Rovithakis, Adaptive Dynamic Output Feedback Neural Network Control of Uncertain MIMO Nonlinear Systems With Prescribed Performance. IEEE Transactions on Neural Networks and Learning Systems, 2012. 23(1): p. 138-149.*

[61] *Yang, H. and J. Liu, An adaptive RBF neural network control method for a class of nonlinear systems. IEEE/CAA Journal of Automatica Sinica, 2018. 5(2): p. 457-462.*

[62] *Li, N., et al., Adaptive Semi-Periodically Intermittent and Lag Synchronization Control of Neural Networks With Mixed Delays. IEEE Access, 2018. 6: p. 4742-4749.*

[63] *Wang, Q., et al., Multiple models and neural networks based adaptive PID decoupling control of mine main fan switchover system. IET Control Theory & Applications, 2018. 12(4): p. 446-455.*

[64] *Hayakawa, T., W.M. Haddad, and N. Hovakimyan, Neural Network Adaptive Control for a Class of Nonlinear Uncertain Dynamical Systems With Asymptotic Stability Guarantees. IEEE Transactions on Neural Networks, 2008. 19(1): p. 80-89.*

[65] *Hayakawa, T., et al., Neural network adaptive control for nonlinear nonnegative dynamical systems. IEEE Transactions on Neural Networks, 2005. 16(2): p. 399-413.*

[66] *Cao, C. and N. Hovakimyan, Novel <formula formulatype="inline"><tex>${cal L}_{1}$</tex></formula> Neural Network Adaptive Control Architecture With Guaranteed Transient Performance. IEEE Transactions on Neural Networks, 2007. 18(4): p. 1160-1171.*

[67] *Werbos, P.J. Neural networks for control and system identification. in Proceedings of the 28th IEEE Conference on Decision and Control. 1989.*

[68] *Al-Dulaimi, A., et al., A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. Computers in Industry, 2019. 108: p. 186-196.*

[69] *Alamdari, B.V., A. Fatehi, and A. Khaki-Sedigh. Neural network model-based predictive control for multivariable nonlinear systems. in 2010 IEEE International Conference on Control Applications. 2010.*

[70] *Wang, Y., et al., Adaptive decoupling switching control based on generalised predictive control. IET Control Theory & Applications, 2012. 6(12): p. 1828-1841.*

[71] *Peng, K., et al., A Frequency Domain Decoupling Method and Multivariable Controller Design for Turbofan Engines. IEEE Access, 2017. 5: p. 27757-27766.*

[72] *Nguyen, T.N., S. Su, and H.T. Nguyen, Neural Network Based Diagonal Decoupling Control of Powered Wheelchair Systems. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2014. 22(2): p. 371-378.*

[73] *Cong, S. and Y. Liang, PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems. IEEE Transactions on Industrial Electronics, 2009. 56(10): p. 3872-3879.*

[74] Hwang, C.L. and C. Jan, Recurrent-Neural-Network-Based Multivariable Adaptive Control for a Class of Nonlinear Dynamic Systems With Time-Varying Delay. IEEE Transactions on Neural Networks and Learning Systems, 2016. 27(2): p. 388-401.

[75] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[76] Merabet, A., A.A. Tanvir, and K. Beddek, Speed control of sensorless induction generator by artificial neural network in wind energy conversion system. IET Renewable Power Generation, 2016. 10(10): p. 1597-1606.

[77] Ruano, A.E.d.B., APPLICATIONS OF NEURAL NETWORKS TO CONTROL SYSTEMS. 1992, UNIVERSITY OF WALES.

[78] Xiao, X., et al., Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting Android malware with system call sequences. IET Information Security, 2017. 11(1): p. 8-15.

[79] Singh, K.R. and S. Chaudhury, Efficient technique for rice grain classification using back-propagation neural network and wavelet decomposition. IET Computer Vision, 2016. 10(8): p. 780-787.

[80] Wu, K., et al., A Novel Approach to Subpixel Land-Cover Change Detection Based on a Supervised Back-Propagation Neural Network for Remotely Sensed Images With Different Resolutions. IEEE Geoscience and Remote Sensing Letters, 2017. 14(10): p. 1750-1754.

[81] Wei, P., C. Cheng, and T. Liu, A Photonic Transducer-Based Optical Current Sensor Using Back-Propagation Neural Network. IEEE Photonics Technology Letters, 2016. 28(14): p. 1513-1516.

[82] Rumelhart, et al., Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations. 1986.

[83] Strickland, J.S., Data Science and Analytics for Ordicary People. 2015: Lulu Inc.

[84] Ahmed, J. and E.M. Alkhalifa. Efficient single layer handwritten digit recognition through an optimizing algorithm. in Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on. 2002.

[85] Zhao, Y., B. Deng, and Z. Wang. Analysis and study of perceptron to solve XOR problem. in The 2nd International Workshop on Autonomous Decentralized System, 2002. 2002.

[86] Zhang, Y., X. Wang, and E.G. Friedman, Memristor-Based Circuit Design for Multilayer Neural Networks. IEEE Transactions on Circuits and Systems I: Regular Papers, 2018. 65(2): p. 677-686.

[87] Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.

[88] Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.

[89] Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.

[90] Shu, H.-l., PID Neural Network and Its Control System. Defense Industry Press Pub, 2006.

[91] Dingguo, L.S.W.Z.W.G.W., Pid neural network sliding-mode controller design for three level based vsc-hvdc converter of offshore wind power. Proceedings of the CSEE, 2012. 32: p. 20-28.

[92] Xiao Deng, Z.-l.L., Chao Yang, Mu-yi Hu, Study on the control of black liquor level based on improved PID neural network. Transactions of China Pulp and Paper, 2014. 29: p. 58-62.

[93] Yu, Y., Y. Huang, and B. Zeng. A PID neural network controller. in Proceedings of the International Joint Conference on Neural Networks, 2003. 2003.

[94] Changhua, L. and S. Hua. Design of electric loading system in flight simulator based on PIDNN. in 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC). 2011.

[95] Guo, A., J. Li, and J. Yang. PIDNN decoupling control of doubly fed hydro-generator system based on PSO method. in Proceedings of the 30th Chinese Control Conference. 2011.

[96] Han, K., et al. Tracking control of ball and plate system using a improved PSO on-line training PID neural network. in 2012 IEEE International Conference on Mechatronics and Automation. 2012.

[97] Aiwen, G., Y. Jiandong, and B. Haiyan. PID neural network decoupling control for doubly fed hydro-generator system. in 2008 7th World Congress on Intelligent Control and Automation. 2008.

[98] Lin, L. and X. Peng. A PID neural network control for permanent magnet synchronous motor servo system. in 2010 5th International Conference on Computer Science & Education. 2010.

[99] Elamvazuthi, I., et al. An intelligent control of Blood Pressure system using PID and Neural Network. in 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA). 2013.

[100] Teng, W.f., H.p. Pan, and J. Ren. Neural network PID decoupling control based on chaos particle swarm optimization. in Proceedings of the 33rd Chinese Control Conference. 2014.

[101] Tian, Z., et al. A PID Neural Network Control for Position Servo System with Gear Box at Variable Load. in 2016 IEEE Vehicle Power and Propulsion Conference (VPPC). 2016.

[102] Yong, Z., Z. Hai-bo, and L. Tian-qi. PIDNN decoupling control of boiler combustion system based on MCS. in 2017 29th Chinese Control And Decision Conference (CCDC). 2017.

[103] Weijie, D., et al. Active Power Filter based on PIDNN controller. in IEEE PES Innovative Smart Grid Technologies. 2012.

[104] Ping, H. An analytical design of GAPIDNN algorithm for AQM. in Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC). 2013.

[105] Ge, S.L., et al. Design of PIDNN adaptive disturbance rejection decoupling controller. in 2017 Chinese Automation Congress (CAC). 2017.

[106] Huailin, S., G. Xiucai, and S. Hua. PID neural networks in multivariable systems. in Proceedings of the IEEE Internatinal Symposium on Intelligent Control. 2002.

[107] Vlachos, C., D. Williams, and J.B. Gomm, Solution to the Shell standard control problem using genetically tuned PID controllers. Control Engineering Practice, 2002. 10(2): p. 151-163.

[108] Cong, S., G. Li, and B. Ji, A NOVEL PID-LIKE NEURAL NETWORK CONTROLLER. IFAC Proceedings Volumes, 2005. 38(1): p. 121-126.

[109] Yazdizadeh, A., et al., Adaptive Neuro-PID Controller Design with Application to Nonlinear Water Level in NEKA Power Plant. Journal of Applied Sciences, 2009. 9(19): p. 3513-3521.

[110] Yazdizadeh, A., et al. Adaptive PID controller design with application to nonlinear water level in NEKA Power Plant. in 2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP). 2010.

[111] Mehrafrooz, A., M.R. Sorkhkolaei, and A. Yazdizadeh. Simultaneous application of adaptive PID controller and smith dead-time predictor rule in nonlinear water level control in Neka power plant. in 2011 6th IEEE Conference on Industrial Electronics and Applications. 2011.

[112] Shu, H. and Y.k. Xu. Application of additional momentum in PID neural network. in 2014 4th IEEE International Conference on Information Science and Technology. 2014.

[113] Meng, L., et al. Design of an improved PID neural network controller based on particle swarm optimazation. in 2015 Chinese Automation Congress (CAC). 2015.

[114] Sento, A. and Y. Kitjaidure. Neural network controller based on PID using an extended Kalman filter algorithm for multi-variable non-linear control system. in 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI). 2016.

[115] Shan, W.j. and W. Tang. A neural network fractional order PID controller for FOLPD process. in 2016 35th Chinese Control Conference (CCC). 2016.

References for Chapter 2

[1] *Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.*

[2] *Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.*

[3] *Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.*

[4] *Astrom, K.J. and T. Hagglund, PID Controllers: Theory, Design and Tuning. 2 ed. 1995, USA: Instrument Society of America.*

[5] *Astrom, K.J. and B. Wittenmark, Adaptive Control Stability Convergence and Robustness. 1989, University of California: Prentice Hall Information and System Science Series.*

[6] *Farivar, F., et al., Hybrid Control of Flexible Manipulator. Journal of Applied Sciences, 2009. 9(4): p. 639-650.*

[7] *Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.*

[8] *Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.*

[9] *Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.*

[10] *Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.*

[11] *Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.*

[12] *Apte, Y.S., Routh-Hurwitz stability criterion and its equivalents. India, IEE-IERE Proceedings -, 1969. 7(4): p. 149-154.*

References for Chapter 3

[1] *Zhou, Q., et al., Approximation-Based Adaptive Tracking Control for MIMO Nonlinear Systems With Input Saturation. IEEE Transactions on Cybernetics, 2015. 45(10): p. 2119-2128.*

[2] *Patino, H.D. and D. Liu, Neural network-based model reference adaptive control system. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2000. 30(1): p. 198-204.*

[3] *Liu, M., Delayed Standard Neural Network Models for Control Systems. IEEE Transactions on Neural Networks, 2007. 18(5): p. 1376-1391.*

[4] *Meng, D., et al., Data-Driven Control for Relative Degree Systems via Iterative Learning. IEEE Transactions on Neural Networks, 2011. 22(12): p. 2213-2225.*

[5] *Oomen, T., et al., Iterative Data-Driven${\cal H}_{\infty}$Norm Estimation of Multivariable Systems With Application to Robust Active Vibration Isolation. IEEE Transactions on Control Systems Technology, 2014. 22(6): p. 2247-2260.*

[6] *Zhang, M. and M. Gan, Data-Driven Adaptive Optimal Control for Linear Systems With Structured Time-Varying Uncertainty. IEEE Access, 2019. 7: p. 9215-9224.*

[7] *Hay, M., et al., Resisting structural re-identification in anonymized social networks. Proc. VLDB Endow., 2008. 1(1): p. 102-114.*

[8] *Donald, C., K.A. Charles, and K.J. Ronald, Control Systems. 2005, McGraw-Hill Education: New York.*

[9] Fliess, M. and C. Join, Model-free control. International Journal of Control, 2013. 86(12): p. 2228-2252.

[10] Lafont, F., et al., A model-free control strategy for an experimental greenhouse with an application to fault accommodation. Computers and Electronics in Agriculture, 2015. 110: p. 139-149.

[11] Madadi, E. and D. Söffker, Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control. 2015(57243): p. V001T03A004.

[12] Radac, M.B., R.E. Precup, and E.M. Petriu, Model-Free Primitive-Based Iterative Learning Control Approach to Trajectory Tracking of MIMO Systems With Experimental Validation. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(11): p. 2925-2938.

[13] Hou, Z., S. Liu, and T. Tian, Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.

[14] Lu, W., P. Zhu, and S. Ferrari, A Hybrid-Adaptive Dynamic Programming Approach for the Model-Free Control of Nonlinear Switched Systems. IEEE Transactions on Automatic Control, 2016. 61(10): p. 3203-3208.

[15] Wang, Z., L. Liu, and H. Zhang, Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.

[16] Gao, B., et al. Model-free adaptive MIMO control algorithm application in polishing robot. in 2017 6th Data Driven Control and Learning Systems (DDCLS). 2017.

[17] Luo, B., et al., Policy Gradient Adaptive Dynamic Programming for Data-Based Optimal Control. IEEE Transactions on Cybernetics, 2017. 47(10): p. 3341-3354.

[18] Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.

[19] Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.

[20] Wang, Y., et al., Adaptive decoupling switching control based on generalised predictive control. IET Control Theory & Applications, 2012. 6(12): p. 1828-1841.

[21] Peng, K., et al., A Frequency Domain Decoupling Method and Multivariable Controller Design for Turbofan Engines. IEEE Access, 2017. 5: p. 27757-27766.

[22] Nguyen, T.N., S. Su, and H.T. Nguyen, Neural Network Based Diagonal Decoupling Control of Powered Wheelchair Systems. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2014. 22(2): p. 371-378.

[23] Cong, S. and Y. Liang, PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems. IEEE Transactions on Industrial Electronics, 2009. 56(10): p. 3872-3879.

[24] Hwang, C.L. and C. Jan, Recurrent-Neural-Network-Based Multivariable Adaptive Control for a Class of Nonlinear Dynamic Systems With Time-Varying Delay. IEEE Transactions on Neural Networks and Learning Systems, 2016. 27(2): p. 388-401.

[25] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[26] Merabet, A., A.A. Tanvir, and K. Beddek, Speed control of sensorless induction generator by artificial neural network in wind energy conversion system. IET Renewable Power Generation, 2016. 10(10): p. 1597-1606.

[27] Yang, B. and A.J. Calise, Adaptive Control of a Class of Nonaffine Systems Using Neural Networks. IEEE Transactions on Neural Networks, 2007. 18(4): p. 1149-1159.

[28] Tee, K.P., S.S. Ge, and F.E.H. Tay, Adaptive Neural Network Control for Helicopters in Vertical Flight. IEEE Transactions on Control Systems Technology, 2008. 16(4): p. 753-762.

[29] *Park, J., S. Kim, and C. Moon, Adaptive Neural Control for Strict-Feedback Nonlinear Systems Without Backstepping. IEEE Transactions on Neural Networks, 2009. 20(7): p. 1204-1209.*

[30] *Liu, Y., et al., Adaptive Neural Output Feedback Controller Design With Reduced-Order Observer for a Class of Uncertain Nonlinear SISO Systems. IEEE Transactions on Neural Networks, 2011. 22(8): p. 1328-1334.*

[31] *Jin, Z., S.S. Ge, and L. Tong Heng, Output feedback control of a class of discrete MIMO nonlinear systems with triangular form inputs. IEEE Transactions on Neural Networks, 2005. 16(6): p. 1491-1503.*

[32] *Zhang, T. and S.S. Ge, Adaptive Neural Network Tracking Control of MIMO Nonlinear Systems With Unknown Dead Zones and Control Directions. IEEE Transactions on Neural Networks, 2009. 20(3): p. 483-497.*

[33] *Chen, M., S.S. Ge, and B.V.E. How, Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities. IEEE Transactions on Neural Networks, 2010. 21(5): p. 796-812.*

[34] *Yang, Q., Z. Yang, and Y. Sun, Universal Neural Network Control of MIMO Uncertain Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2012. 23(7): p. 1163-1169.*

[35] *Chen, Z., et al., Adaptive Neural Control of MIMO Nonlinear Systems With a Block-Triangular Pure-Feedback Control Structure. IEEE Transactions on Neural Networks and Learning Systems, 2014. 25(11): p. 2017-2029.*

[36] *Meng, W., Q. Yang, and Y. Sun, Adaptive Neural Control of Nonlinear MIMO Systems With Time-Varying Output Constraints. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(5): p. 1074-1085.*

[37] *Yang, Q., S. Jagannathan, and Y. Sun, Robust Integral of Neural Network and Error Sign Control of MIMO Nonlinear Systems. IEEE Transactions on Neural Networks and Learning Systems, 2015. 26(12): p. 3278-3286.*

[38] *Ronald, J., NEURAL NETWORK APPLICATIONS, in Electronic Engine Control Technologies. 2004, SAE. p. 661-661.*

[39] *Yong, Z., Z. Hai-bo, and L. Tian-qi. PIDNN decoupling control of boiler combustion system based on MCS. in 2017 29th Chinese Control And Decision Conference (CCDC). 2017.*

[40] *Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.*

[41] *Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.*

[42] *Shu, H. and Y. Xu. Application of additional momentum in PID neural network. in 2014 4th IEEE International Conference on Information Science and Technology. 2014.*

[43] *Meng, L., et al. Design of an improved PID neural network controller based on particle swarm optimazation. in 2015 Chinese Automation Congress (CAC). 2015.*

[44] *Teng, W., H. Pan, and J. Ren. Neural network PID decoupling control based on chaos particle swarm optimization. in Proceedings of the 33rd Chinese Control Conference. 2014.*

[45] *Tian, Z., et al. A PID Neural Network Control for Position Servo System with Gear Box at Variable Load. in 2016 IEEE Vehicle Power and Propulsion Conference (VPPC). 2016.*

[46] *Bahri, N., et al. Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems. in 2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14). 2014.*

[47] *Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.*

[48] *Merayo, N., et al., PID controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2017. 9(5): p. 433-445.*

[49] Phillips, S.F. and D.E. Seborg. Conditions that Guarantee No Overshoot for Linear Systems. in 1987 American Control Conference. 1987.

References for Chapter 4

[1] Arash Mehrafrooz, F.H., *Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems.* IEEE Transactions on Automatic Control (Submitted), 2019.

[2] Chen, M., S.S. Ge, and B.V.E. How, *Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities.* IEEE Transactions on Neural Networks, 2010. 21(5): p. 796-812.

[3] Dobrowiecki, T.P. and J. Schoukens, *Linear Approximation of Weakly Nonlinear MIMO Systems.* IEEE Transactions on Instrumentation and Measurement, 2007. 56(3): p. 887-894.

[4] Zhao, S., X. Gao, and C. Che. *Robust adaptive feedback linearization control for a class of MIMO uncertain nonlinear systems*. in *The 27th Chinese Control and Decision Conference (2015 CCDC)*. 2015.

[5] Fliess, M. and C. Join, *Model-free control.* International Journal of Control, 2013. 86(12): p. 2228-2252.

[6] Bu, X., et al., *Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information.* Asian Journal of Control, 2018. 20(2): p. 962-968.

[7] Madadi, E. and D. Söffker, *Model-Free Approaches Applied to the Control of Nonlinear Systems: A Brief Survey With Special Attention to Intelligent PID Iterative Learning Control.* 2015(57243): p. V001T03A004.

[8] Khadraoui, S., et al., *Adaptive Controller Design for Unknown Systems Using Measured Data.* Asian Journal of Control, 2016. 18(4): p. 1453-1466.

[9] Xie, C.H. and G.H. Yang, *Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults.* International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.

[10] Cui, X., et al., *Adaptive dynamic programming for tracking design of uncertain nonlinear systems with disturbances and input constraints.* International Journal of Adaptive Control and Signal Processing, 2017. 31(11): p. 1567-1583.

[11] Lafont, F., et al., *A model-free control strategy for an experimental greenhouse with an application to fault accommodation.* Computers and Electronics in Agriculture, 2015. 110: p. 139-149.

[12] Wang, Z., L. Liu, and H. Zhang, *Neural Network-Based Model-Free Adaptive Fault-Tolerant Control for Discrete-Time Nonlinear Systems With Sensor Fault.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. 47(8): p. 2351-2362.

[13] Mehrafrooz, A., He, F. *Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems*. in *IEEE ICIA 2018*. 2018. Wuyi Mountain, Fujian, China.

[14] Ge, S.S., C. Yang, and T.H. Lee, *Adaptive Predictive Control Using Neural Network for a Class of Pure-Feedback Systems in Discrete Time.* IEEE Transactions on Neural Networks, 2008. 19(9): p. 1599-1614.

[15] Park, J., S. Kim, and C. Moon, *Adaptive Neural Control for Strict-Feedback Nonlinear Systems Without Backstepping.* IEEE Transactions on Neural Networks, 2009. 20(7): p. 1204-1209.

[16] Nodland, D., H. Zargarzadeh, and S. Jagannathan, *Neural Network-Based Optimal Adaptive Output Feedback Control of a Helicopter UAV.* IEEE Transactions on Neural Networks and Learning Systems, 2013. 24(7): p. 1061-1073.

[17] Ahmed, J. and E.M. Alkhalifa. *Efficient single layer handwritten digit recognition through an optimizing algorithm*. in *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*. 2002.

[18] Zhao, Y., B. Deng, and Z. Wang. *Analysis and study of perceptron to solve XOR problem*. in *The 2nd International Workshop on Autonomous Decentralized System, 2002.* 2002.
[19] Zhang, Y., X. Wang, and E.G. Friedman, *Memristor-Based Circuit Design for Multilayer Neural Networks.* IEEE Transactions on Circuits and Systems I: Regular Papers, 2018. 65(2): p. 677-686.
[20] Huailin, S., G. Xiucai, and S. Hua. *PID neural networks in multivariable systems*. in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*. 2002.
[21] Yu, Y., Y. Huang, and B. Zeng. *A PID neural network controller*. in *Proceedings of the International Joint Conference on Neural Networks, 2003.* 2003.
[22] Lin, L. and X. Peng. *A PID neural network control for permanent magnet synchronous motor servo system*. in *2010 5th International Conference on Computer Science & Education*. 2010.
[23] Changhua, L. and S. Hua. *Design of electric loading system in flight simulator based on PIDNN*. in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. 2011.
[24] Ge, S.L., et al. *Design of PIDNN adaptive disturbance rejection decoupling controller*. in *2017 Chinese Automation Congress (CAC)*. 2017.
[25] Yong, Z., Z. Hai-bo, and L. Tian-qi. *PIDNN decoupling control of boiler combustion system based on MCS*. in *2017 29th Chinese Control And Decision Conference (CCDC)*. 2017.
[26] Dingguo, L.S.W.Z.W.G.W., *Pid neural network sliding-mode controller design for three level based vsc-hvdc converter of offshore wind power.* Proceedings of the CSEE, 2012. 32: p. 20-28.
[27] Hernandez-Alvarado, R., et al. *Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles*. in *OCEANS 2016 MTS/IEEE Monterey*. 2016.
[28] Jing, X. and L. Cheng, *An Optimal PID Control Algorithm for Training Feedforward Neural Networks.* IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.
[29] Scott, G.M., J.W. Shavlik, and W.H. Ray, *Refining PID Controllers Using Neural Networks.* Neural Computation, 1992. 4(5): p. 746-757.
[30] Bahri, N., et al. *Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems*. in *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*. 2014.
[31] Saerens, M. and A. Soquet, *Neural controller based on back-propagation algorithm.* IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.
[32] Phillips, S.F. and D.E. Seborg. *Conditions that Guarantee No Overshoot for Linear Systems*. in *1987 American Control Conference*. 1987.
[33] Hou, Z., S. Liu, and T. Tian, *Lazy-Learning-Based Data-Driven Model-Free Adaptive Predictive Control for a Class of Discrete-Time Nonlinear Systems.* IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(8): p. 1914-1928.
[34] Ribeiro, J.M.S., et al. *Comparison of PID controller tuning methods: analytical/classical techniques versus optimization algorithms*. in *2017 18th International Carpathian Control Conference (ICCC)*. 2017.

References for Chapter 5

[1] Kalpana, D., T. Thyagarajan, and N. Gokulraj, Modeling and control of non-square MIMO system using relay feedback. ISA Transactions, 2015. 59: p. 408-417.
[2] S. Bhat Vinayambika, S.S.P., and I. Thirunavukkarasu, A Comparative Study on Control Techniques of Non-square Matrix Distillation Column. International Science Press, 2015: p. 1129-1136.
[3] Reeves, D.E. and Y. Arkun, Interaction measures for nonsquare decentralized control structures. AIChE Journal, 1989. 35(4): p. 603-613.

[4] Vlachos, C., D. Williams, and J.B. Gomm, Solution to the Shell standard control problem using genetically tuned PID controllers. Control Engineering Practice, 2002. 10(2): p. 151-163.

[5] Tan, J.L.a.N.C.a.X.Y.a.S., Modified Internal Model Control for Non-square Systems Based on Smith Delay Compensator Control. Sensors & Transducers, 2014. 165(2): p. 96-101.

[6] Muske, K., et al., Crude unit product quality control. Computers & Chemical Engineering, 1991. 15(9): p. 629-638.

[7] Harmon Ray, W., Process Dynamics, Modeling, and Control / B.A. Ogunnaike, W.H. Ray. 2018.

[8] Morari, M., Grimm, W., Ogelsby, M. J., & Prosser, I. D., Design of resilient processing plants. VIII: Design of energy measurement systems for unstable plants - new insights. Chemical Engineering Science, 1985. 40(187).

[9] Safaei, A. and M.N. Mahyuddin, Adaptive Model-Free Control Based on an Ultra-Local Model With Model-Free Parameter Estimations for a Generic SISO System. IEEE Access, 2018. 6: p. 4266-4275.

[10] Bu, X., et al., Model Free Adaptive Control for a Class of Nonlinear Systems Using Quantized Information. Asian Journal of Control, 2018. 20(2): p. 962-968.

[11] Xie, C.H. and G.H. Yang, Model-free fault-tolerant control approach for uncertain state-constrained linear systems with actuator faults. International Journal of Adaptive Control and Signal Processing, 2017. 31(2): p. 223-239.

[12] Fliess, M. and C. Join, MODEL-FREE CONTROL AND INTELLIGENT PID CONTROLLERS: TOWARDS A POSSIBLE TRIVIALIZATION OF NONLINEAR CONTROL? IFAC Proceedings Volumes, 2009. 42(10): p. 1531-1550.

[13] Roman, R.C., et al. Data-driven optimal model-free control of twin rotor aerodynamic systems. in 2015 IEEE International Conference on Industrial Technology (ICIT). 2015.

[14] Al-Dulaimi, A., et al., A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. Computers in Industry, 2019. 108: p. 186-196.

[15] Ansari, A.Q., N.K. Gupta, and Ekata. Adaptive neurofuzzy system for tuberculosis. in 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing. 2012.

[16] Theodoridis, D., M. Christodoulou, and Y. Boutalis. Direct adaptive control of unknown multi-variable nonlinear systems with robustness analysis using a new neuro-fuzzy representation and a novel approach of parameter hopping. in 2009 17th Mediterranean Conference on Control and Automation. 2009.

[17] Arash Mehrafrooz, F.H., Modification of Model-Free Multivariable Adaptive Neural Network Controller Using Two Dynamic Layers for Controlling Nonlinear Square MIMO Systems. IEEE Transactions on Control Systems Technology, 2019.

[18] Arash Mehrafrooz, F.H., Introducing a Novel Model-free Multivariable Adaptive Neural Network Controller (MANNC) for Square MIMO Systems. IEEE Transactions on Automatic Control (Submitted), 2019.

[19] Mehrafrooz, A., He, F. Introducing a Model-free Adaptive Neural Network Auto-tuned Control Method for Nonlinear SISO Systems. in IEEE ICIA 2018. 2018. Wuyi Mountain, Fujian, China.

[20] Saerens, M. and A. Soquet, Neural controller based on back-propagation algorithm. IEE Proceedings F - Radar and Signal Processing, 1991. 138(1): p. 55-62.

[21] Hernandez-Alvarado, R., et al. Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles. in OCEANS 2016 MTS/IEEE Monterey. 2016.

[22] Jing, X. and L. Cheng, An Optimal PID Control Algorithm for Training Feedforward Neural Networks. IEEE Transactions on Industrial Electronics, 2013. 60(6): p. 2273-2283.

[23] Scott, G.M., J.W. Shavlik, and W.H. Ray, Refining PID Controllers Using Neural Networks. Neural Computation, 1992. 4(5): p. 746-757.

[24] Chidambaram, P.G.a.M., Multivariable Controller Tuning for Non-square Systems with RHP Zeros by Genetic Algorithm," Chemical and Biochemical Engineering. 2010. 24: p. 17-22.

[25] Farnaz, Z., et al., DC Motor Torque Control Using State Estimation. Vol. 44. 2016.