

**APPLICATION OF ARTIFICIAL NEURAL NETWORKS FOR
THE DESIGN OF A FEEDBACK CONTROLLER FOR A
CONTINUOUS – STIRRED TANK REACTOR**

BY

SULAIMAN MOSHOOD ADESOJI

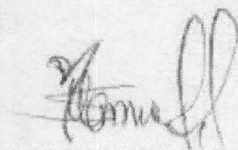
(M. ENG/SEET/2005/'06/1130)

**A THESIS SUBMITTED TO THE POSTGRADUATE SCHOOL
FOR THE DEGREE OF MASTER'S (M. ENG) IN CHEMICAL
ENGINEERING,
FEDERAL UNIVERSITY OF TECHNOLOGY,
MINNA.**

AUGUST, 2006.

DECLARATION

I SULAIMAN, MOSHOOD.ADESOJI (M.ENG/SEET/2005/1130), hereby declare that this research project, "Application of Artificial Neural Networks for the Design of Feedback Controller for a Continuous Stirred Tank Reactor" carried out under the supervision of Prof. K.R. Onifade for the award of Master's Degree of Engineering in the Department of Chemical Engineering, Federal University of Technology, Minna, has not been submitted elsewhere for any degree.



Sulaiman Moshood A.

15/08/08

Date

CERTIFICATION

The thesis titled:” Application of Artificial Neural Networks for the Design of a Feedback Controller for a Continuous Stirred Tank Reactor” by: SULAIMAN MOSHOOD ADESOJI (M.Eng/SEET/05/1130) meets the regulations governing the award of the degree of master’s degree of the Federal University of Technology, Minna and is approved for its contribution to scientific knowledge and literary presentation.



Supervisor
Prof. K.R. Onifade

28/08/08

Date

Head of Department
Dr. M. O. Edoga

Date

Dean SEET
Prof. J. O. Odigure

Date

Dean, Postgraduate School
Prof. S. L. Lamai.

Date

DEDICATION

This thesis is dedicated to Almighty Allah for His care, guidance and protection over the entire universe, then Prof. K. R. Onifade for his tolerance, support and invaluable help.

ACKNOWLEDGEMENTS

Many thanks to my supervisor, Prof. K.R Onifade for his invaluable help, guidance, support and tolerance couples with rapt attention and giving me sense of belongings throughout the course of this work. I did really appreciate of every bit of it, Sir.

My profound gratitude also goes to Engr. D.O Araromi of the Department of Chemical Engineering and Technology, Ladoke Akintola University of Technology, Ogbomoso who made available many books, journals and his computer which were of great help to me.

I am particularly grateful to all my lectures Prof. K. R Onifade, Prof. J.O. Odigure, Prof. F.Aberuagba, Dr. M.O. Edoga, Dr. F. Duncan Aloko, Engr. M.O. Olutoye, and Engr. Akpan who have in one way or the other contributed to the good foundation upon which I have been building.

Thanks are due to Engr. Agbajelola, Engr. (Mrs) Eterigo, Engr. Fatai, Engr. Kovo, Engr. Isah, Mr. E. Afolabi, Mallam Giwa Abdul-Wahab, Mr Menase and the entire staff of the Chemical Engineering Department, Federal University of Technology, Minna, for their morally support and kind gesture.

Thanks to my colleagues Mallam Buhari, Mrs Gbenga Adeola for their assistance. Thanks are also due Popoola Ayobami, Mal. Haske, Dim Paul, Adama Keneth, Babalola Rasheed, Momoh Adija ,Abdullahi Muh'd, and Ajakaye Nelson for their cooperation.

Finally, I appreciate the moral and financial support of my parents, uncle and brothers.

ABSTRACT

Despite the wide application of PID controller in process industries, there are still some limitations which militating against the use of PID controller especially when used alone. The need arises to get equivalent control scheme which will be self-tuning and still perform better function than PID controller.

ANN control approach using references model to tailor system output to a desired responses was developed. The ANN controller using model reference was able to track set point change and reject the uncertainties resulting from external disturbances.

The responses were somehow sluggish in the faces of external disturbances but give no oscillatory behaviors. For PID controller, the performance deteriorated for set point changes and under the influence of external disturbances. This reason for poor performance can be adduced because of high nonlinearity of the CSTR.

The controller has been able to take care of nonlinearly aspect of the system. ANN control scheme has better trajectory tracking ability than PID since the former is based on nonlinearity of the model, while the latter based on particular operating conditions. The control was able to adapt to system changes and operating condition change. There was no need for turning parameter in the control. The control was very adaptive and has self-tuning capability for any change in operating conditions and system parameters. The proposed method is somehow sluggish because the optimal control action is iterative in order to converge to an acceptable accuracy.

The result of the research work also show that the techniques in controlling the system offer a better control alternative to those formerly used and a host of other nonlinear control problems.

TABLE OF CONTENTS

Title Page	i
Abstract	ii
Declaration	iii
Certification	iv
Dedication	v
Acknowledgement	vi
Table of contents	vii
List of figures	xii
List of tables	xvi
Notations, Symbols and Abbreviations	xvii
CHAPTER ONE	1
1.0 INTRODUCTION	1
1.1 Background	1
1.2 Artificial Neural Networks (ANNs)	3
1.2.1 Motivation for an ANN approach to non-linear control	4
1.3 Aims and Objectives	7

CHAPTER TWO	8
2.0 LITERATURE REVIEW	8
2.1 Historical Background of Neural Networks Analysis	8
2.2 Neural Analysis in Control Systems	12
2.2.1 Control technology	13
2.3 Process Control System	14
2.4 Types of control	16
2.4.1 Feedback control	16
2.4.2 Feed forward control	21
2.5 Elements of a control system	22
2.6 Chemical Reactors	23
2.6.1 Batch reactor	24
2.6.2 Continuous stirred-tank reactor (CSTR)	24
2.6.3 Plug flow reactor (PFR)	25
2.6.4 State variables and state equations for the chemical process	25
2.7 Artificial Neural Nets (ANNs)	27
2.7.1 The Mathematical Representation of a Neuron	30
2.8 Learning	31
2.8.1 Human learning	32
2.8.2 Artificial neural networks learning	32
2.8.3 Types of Learning	34
2.9 Transfer Function	34
2.10 Learning Algorithm Using Back-Propagation	35

2.10.1 Selection and preparation of training data	35
2.10.2 The back-propagation algorithm - a mathematical approach	35
2.10.3 Modification of the neutron connection weights	38
2.11 Neural Network Structures	41
2.11.1 Single layer feedforward network	41
2.11.2 Multi-layered perceptrons	43
2.11.3 Recurrent networks	44
2.12 ANN Control Strategies	45
2.12.1 Supervised control	45
2.12.2 Adaptive neural control	46
2.12.3 Inverse models	47
2.12.4 Specialised inverse modelling	48
2.12.5 Direct inverse control	49
2.12.6 Internal model control	50
2.12.7 Model reference control	51
2.12.8 Model predictive control	52
2.12.9 Unsupervised control	53
CHAPTER THREE	54
3.0 RESEARCH METHODOLOGY	54
3.1 Process identification	54
3.2 Description of the Process	55
3.3 Model Equations	56
3.3.1 Reactor total continuity equation	56

3.3.2 Reactor components continuity equations	56
3.3.3 Energy balance equation for the reactor	57
3.3.4 Energy balance equation for the cooling jacket	58
3.4 Application of Neural Analysis in Control Systems	61
3.4.1 Description of model reference control	62
3.4.2 Using the model reference control approach	63
3.4.3 Steps to run the simulink model	64
3.4.4 Requirement of plant identification	70
CHAPTER FOUR	72
RESULTS AND DISCUSSION	72
4.1 Results	72
4.2 Discussion of Results	84
CHAPTER FIVE	84
5.0 CONCLUSION AND RECOMMENDATION	88
5.1 Conclusion	88
5.2 Recommendations	89
References	90
Appendix A	96
Appendix B	100
Appendix C	103
Appendix D	104

LIST OF FIGURES

Figure 1.1 The Components of a Neuron	4
Figure 1.2 Diagram of Neuron Model	5
Figure 1.3 Diagram Shows the Parallelism of Neural Networks	6
Figure 2.1 Block Diagram of a Closed Loop Control System	18
Figure 2.2 Characteristic of a Closed Loop Response to Step Change	21
Figure 2.3 Structure of Feed Forward Control Scheme	22
Figure 2.4 Block Diagram for Feed Forward and Feedback Control	22
Figure 2.5 Block Diagram Component (A) Dynamic Relationship (B) Comparison of Signals	23
Figure 2.6 A Simple Neuron	28
Figure 2.7 Artificial Neural Nets	30
Figure 2.8 Mathematical Representation of a Neuron	31
Figure 2.9 Back propagation Neural Networks Representation	36
Figure 2.10 Backpropagation Net	39
Figure 2.11 Diagram of the Perceptron Model	42
Figure 2.12 Diagram of a Multi-Layered Perceptron	44
Figure 2.13 Diagram of a Recurrent Neural Network	45
Figure 2.14 Supervised Learning Using an Existing Controller	46
Figure 2.15 Adaptive Neural Controls	47
Figure 2.16 Direct Inverse Modelling	48
Figure 2.17 Specialised Inverse Modelling	49
Figure 2.18 Direct Inverse Controls	50

Figure 2.19 The IMC Structure	50
Figure 2.20 The Model Reference Control Structure	52
Figure 2.21 The Model Predictive Control Structure	53
Figure 3.1 Continuous Stirred Tank Reactors for the Process	54
Figure 3.2: Simulink Model Connection Procedures for the Reactor	59
Figure 3.3: Model Reference Control	63
Figure 3.4 Model Reference Control Windows	66
Figure 3.5 Plant Identification Windows	67
Figure 3.6 Plant Responses for NN Model Reference Control	68
Figure 4.1 Concentration profile for Open Loop simulation of CSTR for reversible reaction	70
Figure 4.2 Temperature profile for open loop simulation of CSTR for reversible reaction	70
Figure 4.3 Concentration profile for Closed Loop simulation of CSTR for reversible reaction	71
Figure 4.4 Temperature profile for Closed Loop simulation of CSTR for reversible reaction	72
Figure 4.5 Neural Network Plant Responses after Training (Data C)	72
Figure 4.6 Validation Data Showing the Difference between Plant and Neural Network Outputs	73
Figure 4.7a Training Response Profile for the NN Controller Performance (Table 2, Data A)	73
Figure 4.7b Training Response Profile for the NN Controller Performance	

(Table 2, Data C)	74
Figure 4.8(a) Reference Model and Neural Network Output Signals (Table 2, Data A)	75
Figure 4.8(b) Reference Model and Neural Network Output Signals (Table 2, Data B)	75
Figure 4.8(c): Reference Model and Neural Network Output Signals (Table 2, Data C)	76
Figure 4.9: Closed loop response for the system under PID and ANN control for a Step down Input of 5%	76
Figure 4.10: Closed loop response for the system under PID and ANN control for a Step down Input of 10%	77
Figure 4.11 Closed loop response for the system under PID and ANN control for a Step up Input of 5%	77
Figure 4.12 Closed loop response for the system under PID and ANN control for a step up input of 10%	78
Figure 4.13 Closed loop responses for set point tracking for the system	78

LIST OF TABLES

Table 2.1 Effect of Each Controller on System Output Signal	20
Table 3.1 Nominal Operating Condition and Parameter Value for the Simulation of Reversible Exothermic Reaction in CSTR	61
Table 3.2 Parameter for Model Reference Training Windows	64
Table 3.3 Parameter for Plant Identification Windows	64
Table 4.1 Controller setting for PID Controller and their Performance	73
Table 4.2 Parameters for Training the Plant and Neural Network Controller	76
Table 4.4 Performance Response for NN and PID Controllers at Various Set-point Changes in the Concentration of the Inlet Flow	83

NOTATIONS, SYMBOLS AND ABBREVIATIONS

ANN	:	Artificial Neural Networks
BP	:	Back-propagation
NET	:	Neural Networks or Network
EBP	:	Error Back Propagation
RUN	:	Recurrent Neural Networks
LMS	:	Least Mean Square
CSTR	:	Continuous Stirred Tank Reactor
Σ	:	Summation
F	:	Transfer Function/ Activation function
P	:	Proportional
I	:	Integral
D	:	Derivative
PI	:	Proportional-plus-Integral
PD	:	Proportional-plus-derivative
PID	:	Proportional-plus-Integral-plus derivative
MADALINE	:	Multiple Adaptive Linear Elements
ADALINE	:	Adaptive Linear Elements
PFR	:	Plug Flow Reactor
ρ	:	Density of the material in the system
ρ_i	:	Density of the material in the i^{th} inlet system

- ρ_j : Density of the material in the j^{th} outlet system
- V : Total volume of the system
- F_i : Volumetric flow rate of the i^{th} inlet system
- F_j : Volumetric flow rate of the j^{th} output stream
- C_A : Molar concentration (moles/volume) of A in the system
- C_{Ai} : Molar concentration of A in the i^{th}
- C_{Aj} : Molar concentration of A in the j^{th} output
- r : Reaction rate per unit volume for component A in the system
- sgm : sigmoid
- BPA : Back Propagation Algorithm
- x_j : weighted input
- y_i : activity level of the j^{th} unit in the previous layer
- W_{ij} : weight of the connection between the i^{th} and the j^{th} unit.
- W_{ij} : weight of the connection from unit u_i to unit u_j .
- W : weight matrix
- d_j : desired output of the j^{th} unit.
- EA : Error derivative
- EI : Error changes as the total input received by an output unit is changed

- EW : Error changes as a weight on the connection into an output unit is changed
- I_1, I_2 : Inputs layers
- H_1, H_2 : Hidden layers
- O_1, O_2 : output layers
- MSE : Mean square error
- E : Error function
- δ_n^0 : Learning rate

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background

Despite the advent of many complicated control theories and techniques more than 94% of control loop based on PID controllers are still being used in various industrial processes. The PID controller is one of the easiest ways of control in chemical system reactor because of its simplicity in structure, robustness in operation and easy comprehension in principle. Nevertheless, the PID algorithm may not be effective in highly nonlinear and time varying chemical process. Nonlinearities may be intrinsic to the physics or the chemistry of the process or may arise through the close coupling of a number of simpler processes. In either case, complicated differential equations of the system dynamics pose a challenging problem in the sense of mathematical tractability. To improve the control performance, several scheme of self- tuning PID controller were proposed in the past. Wittenmark proposed the control structure with the PID algorithm calculated vial pole placement design. The self-tuning PI or PID algorithms were automatically derived from dynamic of the controlled processes.

The controller structure was oriented to have a PID structure. The control parameters were obtained using a parameter estimated scheme. Many other forms of PID can be found in literature. However, the limitation of the above stated self-tuning adaptive control techniques is that the control model with linear model is operated in the linear region. If some changes in the process or environment occur, it must be manually checked whether the model is adequate to represent the real model or not since the control design should be based on a reliable model.

Presently, neural networks constitute a very large research interest. They have required capability in solving complex mathematical problems since they have proven to approximate a continuous function accurately. Hence, it has been a subject of focus in the field of chemical process control and has been applied to system identification and controller design. All the above shows that the neural network can capture the characteristics of system patterns and performance approximation function for nonlinear system.

Besides, the state of the art in the area of neural networks in control systems has become increasingly challenging. The need to meet demanding control requirements in increasingly complex dynamical control systems under significant uncertainties makes neural networks very attractive, because of their ability to learn, to approximate functions, to classify patterns and because of their potential for massively interconnection hardware implementation. Neural networks do appear to be able to implement many functions essential to control systems with higher degree of reliability.

Also neural network technology has received much attention in the field of chemical process control, this is because of inherently non-linear nature of most of the processes and neural network have great capability in solving complex nonlinear mathematical problem (Junghui and Titen-chu, 2004).

Neural networks have shown great progress in identification of nonlinear system, which is due to increase in cheap computing power and certain powerful theoretical algorithms (Cybenko, 1989; Lippman, 1987; Rumelhart and Mecelland; 1986).

Application of ANNs to non-linear process control have attracted a rapidly growing interest in the recent time (Willis et al, 199, Hunt etal,1992) several approaches have been used to train ANN to model either the process behaviour or its inverse and subsequently used within conventional model based control schemes including model based predictive control, internal model control, adaptive control and feed forward control (Hunt and Sbarbaro,1992; Barto, Sutton and Anderson, 1983)

1.2 Artificial Neural Networks (ANNs)

An artificial neural network (ANN), also called a simulated neural network (SNN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is a system that is capable of changes its structure based on external or internal information that flows through the network.

A neural network operates as a parallel-distributed processor capable of acquiring knowledge from experimental data, and applying knowledge learned to unseen problems. Haykin outlines how neural networks mimic a very simplified version of the brain in two aspects;

- (i) Knowledge is acquired through a learning process i.e. the network can be trained.
- (ii) Interneuron connection strengths known as synaptic weights are used to store the knowledge.

Neurons are the processing elements of the network. The layout of the neurons in space and the interconnection between them determine the structure of the network. Common structures include layered feedforward, recurrent and radial bias networks. These can be modified to suit a particular application, or alternatively a completely new structure can be designed (Emuoyibofarhe, 2004).

A learning algorithm describes how the interconnection weights are adjusted to achieve the desired behaviour of the network. The science of artificial neural networks is based on the neuron. In order to understand the structure of artificial networks, the basic elements of the neuron should be understood. Neurons are the fundamental elements in the central nervous system.

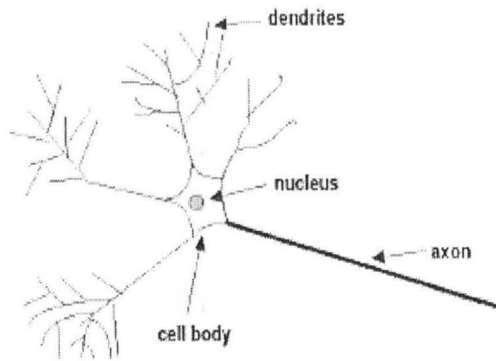


Figure 1.1 The components of a neuron.

A neuron is made up of 3 main parts -dendrites, cell body and axon. The dendrites receive signals coming from the neighbouring neurons. The dendrites send their signals to the body of the cell. The cell body contains the nucleus of the neuron. If the sum of the received signals is greater than a threshold value, the neuron fires by sending an electrical pulse along the axon to the next neuron. The following model is based on the components of the biological neuron

1.2.1 Motivation for an ANN approach to non-linear control

The main appeal of Artificial Neural Networks (ANNs) in control systems

Engineering is that they offer the potential of a generic approach to the modelling and control of linear and non-linear systems. The term "artificial neural network" originates from research which attempted to understand, and proposed simple models of, the operation of the human brain. Consequently, ANNs do possess characteristics which are common with the biological system. They consist of numerous simple processing elements (neurons) joined together by variable strength connections (synapses) to form a massively parallel and highly interconnected information processing system. This gives ANNs several characteristics which are appealing for the modelling and control of non-linear systems, such as the ability to:-

i) Neural networks are composed of elements operating in parallel. Parallel processing allows increased speed of calculation compared to slower sequential processing.

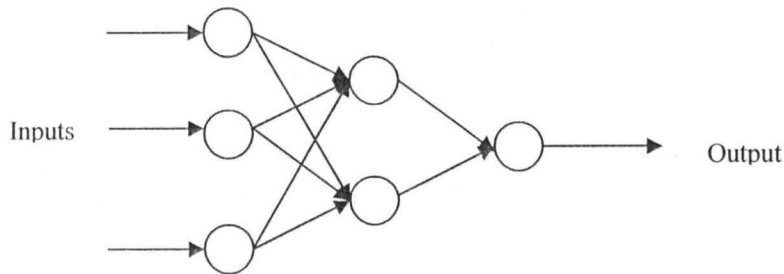


Figure.1.3: Diagram shows the parallelism of neural networks

ii) Artificial neural networks (ANN) have memory

The memory in neural networks corresponds to the weights in the neurons. Neural networks can be trained offline and then transferred into a process where adaptive learning takes place.

iii) Learn by example

Neural networks are trained using data records from the system under study, and hence they are parametric models.

(iv) Degrades gracefully due to their parallel nature

A fault in an ANN could be caused by an erroneous weighted connection, either because of incorrect identification or because of an open or short circuit in a hardware implementation. The parallel structure of a neural network affords it a high degree of tolerance to faults.

v) Attain relatively fast execution times

Once a network has been trained, it can attain fast execution times. In software implementation, because the processing elements perform relatively simple functions, this leads to fast computation compared to other non-linear models (e.g. Volterra series model).

vi) Ability to solve new kinds of Problems

Neural networks are effective at solving problems whose solution are difficult, and may not be possible to define, and since it has the ability to learn from experience (previous examples) when presented with a new but similar problem it can provide solution.

vii) Robustness

Neural networks tend to be more robust than their conventional counterparts because they have the ability to cope well with incomplete data.

viii) Flexibility and Ease of Maintenance

Neural based computers are very flexible in that they are able to adapt their behaviour to new and changing environments. On the other hand the serial conventional computing, are strictly algorithmic and require writing a new program for any modification. They are also easier to maintain to accommodate changes or modifications.

The main disadvantage of ANN is that they operate as black boxes. The rules of operation in neural networks are completely unknown. It is not possible to convert the neural structure into known model structures such as ARMAX, etc. Another disadvantage is the amount of time taken to train networks. It can take considerable time to train an ANN for certain functions.

1.3 Aims and Objectives

The aims of this work include;

- (i) To apply artificial neural network for the design of a feedback controller for a continuous stirred tank reactor using model reference control
- (ii) To simulate and compare the performance of a PID controller with artificial neural network controller.

Objectives

- (i) To formulate a mathematical model for the non-linear CSTR in time domain.
- (ii) To develop an overall model for the nonlinear CSTR using simulink software.
- (iii) To incorporate the overall CSTR model with a PID controller.
- (iv) To incorporate the overall CSTR model with an artificial neural networks controller.

CHAPTER TWO

LITERATURE REVIEW

2.1 Historical Background of Neural Networks Analysis

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts wrote a paper on how neurons might work. In order to describe how neurons in the brain might work, they modeled a simple neural network using electrical circuits.

In 1949, Donald Hebb wrote *The Organization of Behavior*, a work which pointed out the fact that neural pathways are strengthened each time they are used, a concept fundamentally essential to the ways in which humans learn. If two nerves fire at the same time, he argued, the connection between them is enhanced. As computers became more advanced in the 1950's, it was finally possible to simulate a hypothetical neural network. The first step towards this was made by Nathaniel from the IBM research laboratories. Unfortunately for him, the first attempt to do so failed.

In 1959, Bernard and Marcian of Stanford developed models called "ADALINE" and "MADALINE." In a typical display of Stanford's love for acronyms, the names come from their use of Multiple ADaptive LINear Elements. ADALINE was developed to recognize binary patterns so that if it was reading streaming bits from a phone line, it could predict the next bit. MADALINE was the first neural network applied to a real world problem, using an adaptive filter that eliminates echoes on phone lines. While the system is as ancient as air traffic control systems, like air traffic control systems, it is still in commercial use.

In 1962, Widrow and Hoff developed a learning procedure that examines the value before the

weight adjusts it (i.e. 0 or 1) according to the rule: $\text{Weight Change} = (\text{Pre-Weight line value}) * (\text{Error} / (\text{Number of Inputs}))$. It is based on the idea that while one active perceptron may have a big error, one can adjust the weight values to distribute it across the network, or at least to adjacent perceptrons. Applying this rule still results in an error if the line before the weight is 0, although this will eventually correct itself. If the error is conserved so that all of it is distributed to all of the weights then the error is eliminated.

Despite the later success of the neural network, traditional von Neumann architecture took over the computing scene, and neural research was left behind. Ironically, John von Neumann himself suggested the imitation of neural functions by using telegraph relays or vacuum tubes. In the same time period, a paper was written that suggested there could not be an extension from the single layered neural network to a multiple layered neural network. In addition, many people in the field were using a learning function that was fundamentally flawed because it was not differentiable across the entire line. As a result, research and funding went drastically down.

This was coupled with the fact that the early successes of some neural networks led to an exaggeration of the potential of neural networks, capability especially considering the practical technology at the time. Promises went unfulfilled, and at times greater philosophical questions led to fear. Writers pondered the effect that the so-called "thinking machines" would have on humans, ideas which are still around today. The idea of a computer which programs itself is very appealing. If Microsoft's Windows 2000 could reprogram itself, it might be able to correct the programming errors. Such ideas were appealing but very difficult to implement. In addition, von Neumann architecture was gaining in popularity. There were a few advances in the field, but for the most part research was few and far between. In 1972, Kohonen and Anderson developed a

similar network independently of one another. They both used matrix mathematics to describe their ideas but did not realize that what they were doing was creating an array of analog ADALINE circuits. The neurons are supposed to activate a set of outputs instead of just one.

The first multilayered network was developed in 1975, an unsupervised network. In 1982, interest in the field was renewed. John Hopfield of Caltech presented a paper to the National Academy of Sciences. His approach was to create more useful machines by using bidirectional lines. Previously, the connection between neurons was only one way.

<http://www.en.wikipedia.org/wiki/weighted-sum> .

That same year, Reilly and Cooper used a "Hybrid network" with multiple layers, each layer using a different problem-solving strategy. Also in 1982, there was a joint US-Japan conference on Cooperative/Competitive Neural Networks. Japan announced a new Fifth Generation effort on neural networks, and US papers expressed apprehension that the US could be left behind in the field. (Fifth generation computing involves artificial intelligence. First generation used switches and wires, second generation used the transistor, third generation used solid-state technology like integrated circuits and higher level programming languages and the fourth generation is code generators.) As a result, there was more funding and thus more research in the field.

In 1986, with multiple layered neural networks in the news, the problem was how to extend the Widrow-Hoff rule to multiple layers. Three independent groups of researchers, one of which included David Rumelhart, a former member of Stanford's psychology department, came up with similar ideas which are now called back propagation networks because it distributes pattern recognition errors throughout the network. Hybrid networks used just two layers, these back-

propagation networks use many. The result is that back-propagation networks are "slow learners," needing possibly thousands of iterations to learn.

Now, neural networks are used in several applications, some of which will describe later in thesis. The fundamental idea behind the nature of neural networks is that if it works in nature, it must be able to work in computers. The future of neural networks, though, lies in the development of hardware. Much like the advanced chess-playing machines like Deep Blue, fast, efficient neural networks depend on hardware being specified for its eventual use. Research that concentrates on developing neural networks is relatively slow. Due to the limitations of processors, neural networks take weeks to learn. Some companies are trying to create what is called a "silicon compiler" to generate a specific type of integrated circuit that is optimized for the application of neural networks. Digital, analog, and optical chips are the different types of chips being developed. One might immediately discount analog signals as a thing of the past. However neurons in the brain actually work more like analog signals than digital signals. While digital signals have two distinct states (1 or 0, on or off), analog signals vary between minimum and maximum values. It may be awhile, though, before optical chips can be used in commercial applications.

2.2 Neural analysis in control systems

The ever-increasing technological demands of our modern society require innovative approaches to highly demanding control problems. Artificial neural networks with their massive parallelism and learning capabilities offer the promise of better solutions, at least to some problems. By now, the control community has heard of neural networks and wonders if these networks can be used

to provide better control solutions to old problems or perhaps solutions to control problems that have withstood our best efforts (Paul, 1996).

Neural networks have the potential for very complicated behavior. They consist of many interconnected simple nonlinear systems, which are typically modeled by sigmoid functions. The massive interconnections of the rather simple neurons, which make up the human brain, provided the original motivation for the neural network models. The terms artificial neural networks and connectionist models are typically used to distinguish them from the biological networks of neurons of living organisms. Interest in neural networks has made a comeback in this decade after a period of relative inactivity following the shortcomings of early neural networks (the single-layer perceptron), which were publicized in the late 1960s. The renewed interest was due, in part, to powerful new neural models, the multilayer perceptron and the feedback model of Hopfield, and to learning methods such as back propagation; but, it was also due to advances in hardware that have brought within reach the realization of neural networks with very large numbers of nodes.

In a neural network, the simple nonlinear elements called nodes or neurons are interconnected, and the strengths of the interconnections are denoted by parameters called weights. These weights are adjusted, depending on the task at hand, to improve performance. They can be assigned new values in two ways: either determined via some prescribed off-line algorithm remaining fixed during operation or adjusted via a learning process. Learning is accomplished by, first, adjusting these weights step by step (typically to minimize some objective function) and, then, storing these best values as the actual strengths of the interconnections. The interconnections and their strength provide the memory, which is necessary in a learning process.

ne ability to learn is one of the main advantages that make the neural networks so attractive. They also have the capability of performing massive parallel processing, which are in contrast to the von Neumann machines the conventional digital computers in which the instructions are executed sequentially. Neural networks can also provide, in principle, significant fault tolerance, since damage to a few links need not significantly impair the overall performance. The benefits are most dramatic when a large number of nodes are used and are implemented in hardware. The hardware implementation of neural networks is currently a very active research area; optic and more conventional means of implementation of these large networks have been suggested.

Neural networks are characterized by their network topology that is, by the number of interconnections, the node characteristics that are classified by the type of nonlinear elements used, and the kind of learning rules implemented.

2.1 Control technology

The use of neural networks in control systems can be seen as a natural step in the evolution of control methodology to meet new challenges. Looking back, the evolution in the control area has been fueled by three major needs: the need to deal with increasingly complex systems, the need to accomplish increasingly demanding design requirements, and the need to attain these requirements with less precise advanced knowledge of the plant and its environment that is, the need to control under increased uncertainty. Today, the need to control, in a better way, increasingly complex dynamical systems under significant uncertainty has led to a re-evaluation of the conventional control methods, and it has made the need for new methods quite apparent. It has also led to a more general concept of control, one that includes higher-level decision making, planning, and learning, which are capabilities necessary when higher degrees of system

autonomy are desirable (Paul, 1996). In view of this, it is not surprising that the control community is seriously and actively searching for ideas to deal effectively with the increasingly challenging control problems of our modern society. Need is the mother of invention, and this has been true in control since the times of Ktesibios and his water clock with its feedback mechanism in the third century B.C. (Mayr, 1970), the earliest feedback device on record. So the use of the neural networks in control is rather a natural step in its evolution. Neural networks appear to offer new promising directions toward better understanding and perhaps even solving some of our most difficult control problems. History, of course, has made clear that neural networks will be accepted and used if they solve problems that have been previously impossible or very difficult to solve (Lippmann, 1987).

2.2.2 Application of neural analysis in control systems

Neural networks have been applied very successfully in the identification and control of dynamic systems. The universal approximation capabilities of the multilayer perceptron make it a popular choice for modelling nonlinear systems and for implementing general-purpose nonlinear controllers. This chapter presents brief descriptions of Model Reference Control (MRC) architectures and it is used to simulate our system.

There are typically two steps involved when using neural networks for control:

- a. System Identification
- b. Control Design

In the system identification stage, a neural network model of the plant that we want to control is developed. In the control design stage, we use the neural network plant model to design (or train) the controller (MATLAB 7, Control System Toolbox, User's Guide). Depending on the architecture, a number of control design methodologies can be used. In this research work model

reference control (MRC) has been used. Here, the controller is a neural network that is trained to control a plant so that it follows a reference model. The neural network plant model is used to assist in the controller training.

2.3 Process control system

Control systems are every where around us and within us. Many complex control systems are included among the functions of the human body. An elaborate control system centered in the hypothalamus of the brain maintains body temperature at $35^{\circ}\text{C} - 37^{\circ}\text{C}$, in spite of changes in physical activity and external ambience (Murril, 1996).

A control system is therefore any group of components that maintains some desired result in a process. A process is any combination of materials and equipment that produces a desirable result through changes in energy, physical properties or chemical properties (Bateson, 1993). Examples of processes are a food processing plant, a petroleum refinery, an electrical power plant, a textile making plant, a plastic making plant etc. Process control involves the control of variables in a manufacturing process. It involves combinations of any materials and equipment that modifies a product, making it more useful and more valuable (Emuoyibofarhe, 2004). The most common controlled variables in a process are temperature, pressure, flow rate, and level. Other includes color, conductivity, PH, hardness, viscosity, density and composition. Many process control system are used to maintain constant processing conditions and, hence, are regulator systems.

The process control usually incorporates a device or group of devices that automatically controls a mechanism, a source power or other variables. The system automatically compares the controlled output of a system to the controlling input. The difference between the output and input is called the error signal e , which regulates the output to a desired value.

Process control may be either open or closed loop, but closed loop systems are more common. The process of sending the error signal back for comparison with the input is called Feedback and the whole process of the input, output, error signal and feedback is called a closed loop. Process control systems are not limited to the field of chemical engineering but also electrical engineering and mechanical engineering. This paper work focuses on chemical process control.

Research on chemical process control came up about 1930 (Grebe et al, 1933), discussed some difficult PH control problems and showed the advantage of using controllers with derivative action. Ivanoff, (1934), introduced the concept of potential deviation and potential correlation as an aid in quantitative evaluation of control system. The field has continued to attract researchers and many attractive research results continue to appear in publications (Emuoyibofarhe, 2004).

2.4 Types of control

Control system are classified in various ways

- i. Open and closed loops depending on whether or not feedback is used.
- ii. Regulator and servo up systems depending on whether the set point is constant or changing.
- iii. Process control, Servo mechanism, Sequential control and Numerical control depending on the types of application.
- iv. Analog and Digital depending on the nature of output signal.

2.4.1 Feed back control

Essential to most automatic control mechanism is is an interdisciplinary branch of engineering and mathematics, which deals with the behavior of dynamical systems, which enables a designer to endow a machine, reactor or system with the capacity for automatic control. A feedback loop

is a mechanical, pneumatic, or electronic device that senses or measures a physical, quantity such as position, temperature, size or speed etc. compares it with a pre-established standard, and takes whatever pre programmed action that is necessary to maintain the measured quantity within the limits of the acceptable level. In a feedback control loop, the controlled variable is compared to the set point, with difference, deviation or error e acted upon by the controller to move in such a way as to reduce the error. This action is specifically negative feedback, in that an increase in deviation moves so as to decrease the deviation (Mayr, 1970). (Positive feedback would cause the deviation to expand rather than diminish and therefore does not regulate). The action of the controller is selectable to allow use, form example, appropriate sign for gains.

(b) Block diagrams

Block diagram is a shorthand pictorial representation of the relationship between input and output of a system. This representation is commonly called the block diagram. Control systems are made up of various combinations of the following basic blocks.

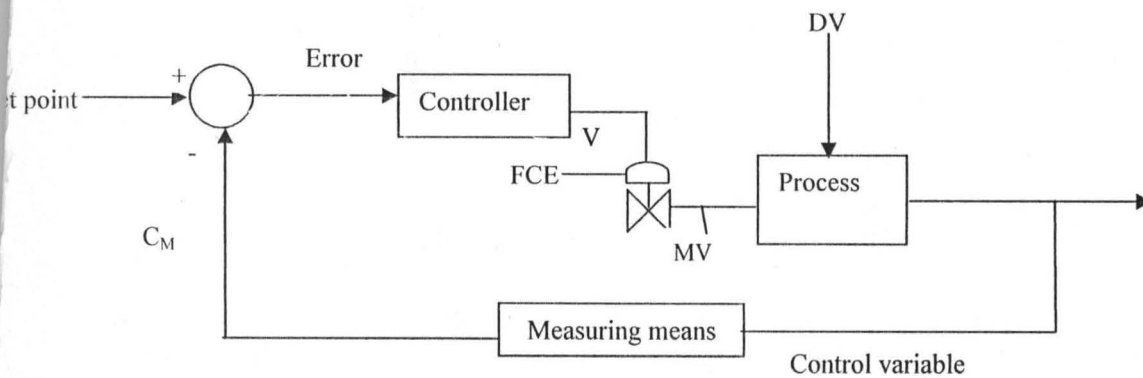


Figure 2.1 Block Diagram of a Closed Loop Control System

Where

V = Controller Output

DV = Disturbance Variable

Cm = Measured Value of Controlled variable.

FCE = Final control element

MV = Manipulated Variable

Set point (SP): the desired value of the controlled variable.

Measured variable: the output of the measuring means / element.

Error: the difference between the set point and the measured value of CV

Controller Output: control action intended to drive the measured of CV toward the set point value.

Manipulated Variable: the variable regulated by the FCE to achieve the desired value of the CV.

Disturbance Variable: the process input variable that affect the controlled variable but are not controlled by the control system.

Controlled variable: the process output variable which is to be controlled.

Sensor: a device that measure the process variable

Transmitter: transmitter is the interphase between the process and its control system. It incorporates a transducer that converts the sensor signal (liquid flow rate, pressure difference etc) into an equivalent electrical or air control signal.

(c) Types of feedback controllers

(i) Proportional Control Mode: Proportional control produces a change in controller output proportional to the error signal.

(ii) Integral Control Mode: The integral control mode changes the output of the controller by an amount proportional to the integral of the error signal.

(iii) Derivative Control Mode: Derivative control changes the output of the controller proportional to the rate of change of the error signal. This change may be caused by the variation in the measured variable, the set-point or both

(iv) Proportional-plus-Integral control: Proportional-plus-integral control is used on process with large load changes when the proportional mode alone is not capable of reducing the offset to an acceptable level. The integral mode provides a reset action which eliminates the proportional offset.

(v) Proportional-plus-Derivative Control: The derivative control mode is sometimes used with the proportional mode to reduce the tendency for oscillation and allow a higher proportional gain setting. The proportional mode provides a change in the controller output which is proportional to the rate of change of error signal.

(vi) Proportional – integral – derivative control (PID): This is usually referred to as three-mode controller. It is used on process with sudden, large load changes when one or two mode control is not capable of keeping the error within acceptable limits. The derivative mode produces an anticipatory action which reduces the maximum error produced by sudden load changes (Paul W. Austin, 1996).

The integral mode provides a reset action which eliminates the amount of offset error. The controller compares a measured value from a process (typically an industrial process) with a reference setpoint value. The difference (or "error" signal) is then used to calculate a new value for a manipulatable input to the process that brings the process' measured value back to its desired setpoint. The three gain factor associated with the Proportional, integral and derivative actions at an acceptable degree of error reduction simultaneously with acceptable dynamic response.

(d) Manual tuning

If the system must remain online, one tuning method is to first set the I and D values to zero. Increase the P until the output of the loop oscillates, and then the P should be left set to be approximately half of that value for a "quarter amplitude decay" type response. Then increase D until any offset is correct in sufficient time for the process. However, too much D will cause instability. Finally, increase I , if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much I will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an "over-damped" closed-loop system is required, which will require a P setting significantly less than half that of the P setting causing oscillation.

Table 2.1 Effect of Each Controller on System Output Signal

Closed Loop Response	Rise time	Overshoot	Settling time	Steady State Error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Increase	Increase	Increase	Eliminate
K_D	Small change	Decrease	Decrease	Small change

Source: Instrument Engineers' Handbook: Process Control by Liptak, Bela, 1995

(e) Ziegler–Nichols method

Another tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols. As in the method above, the I and D gains are first set to zero. The "P" gain is increased until it reaches the "critical gain" K_c at which the output of the loop starts to oscillate. K_c and the oscillation period P_c are used to set the gains as shown:

Zeigler-Nichols Method			
Control Type	K_p	K_i	K_d
P	$0.5 K_c$	-	-
PI	$0.45 K_c$	$1.2 K_p/P_c$	-
PID	$0.6 K_c$	$2 K_p/P_c$	$K_p P_c/8$

Source: Loop Tuning Fundamentals". Control Engineering by Van, Doren, Vance J.

(f) Response in feedback control systems

Feedback control systems are used for various purposes and must meet certain performance requirements. These requirements not only affect such things as speed of response and accuracy, but also the manner in which the system responds in carrying out its control function. All systems contain certain errors. The problem is to keep them within allowable limits.

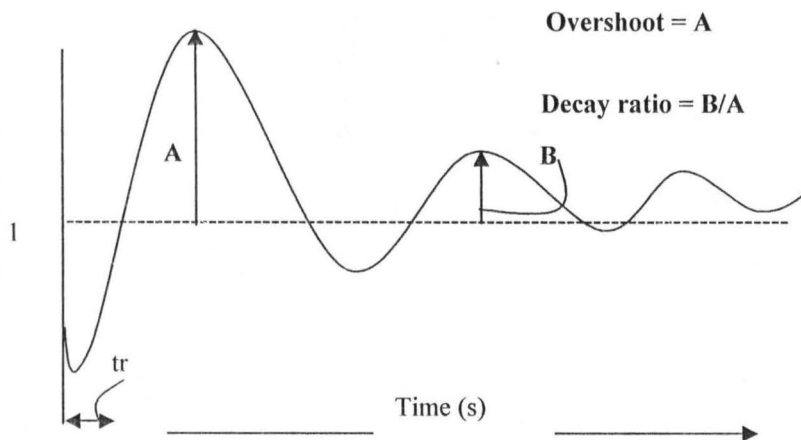


Figure 2.2 Characteristic of a Closed Loop Response to Step Change.

(g) Step response analysis

The following features are evaluating to know the performance of a closed – loop system

- i.) **Rise time:** the time taken the output to first reach 90% of its final value which is always required to be small

- ii.) **Setting time:** the time after which the output remains within $\pm 5\%$ of its final value.
- iii.) **Overshoot:** the peak value divided by the final value, which should be 20% or less
- iv.) **Decay ratio:** the ration of the second and first peaks, which should be 0.3 or less.
- v.) **Steady-state Offset:** the difference between the final value and the desired final value.

The rise time and setting time are measures of the speed of the response, whereas the overshoot, decay ratio and steady state error are related to quality of a response.

2.4.2 Feed forward control

Most feedback systems act post facto (after the fact) that is the effect of the disturbance has been felt by the process. Unlike the feedback systems, a feedforward system uses measurements of disturbance variables to position the manipulated variable in such a way as to minimize any resulting deviation. The disturbance variable could be either measured loads or the set point. The feedforward gain must be set precisely to offset the deviation of the controlled variables from the set point.

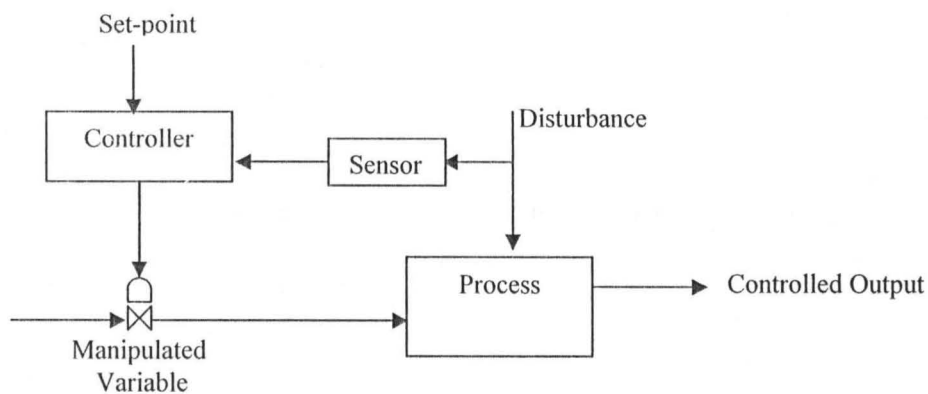


Fig. 2.3 Structure of feedforward control scheme (Fundamental of Process Control Theory, United State, 1997)

Feed forward control is usually combined with feedback control to eliminate any offset resulting from inaccurate measurements and calculations and unmeasured load components.

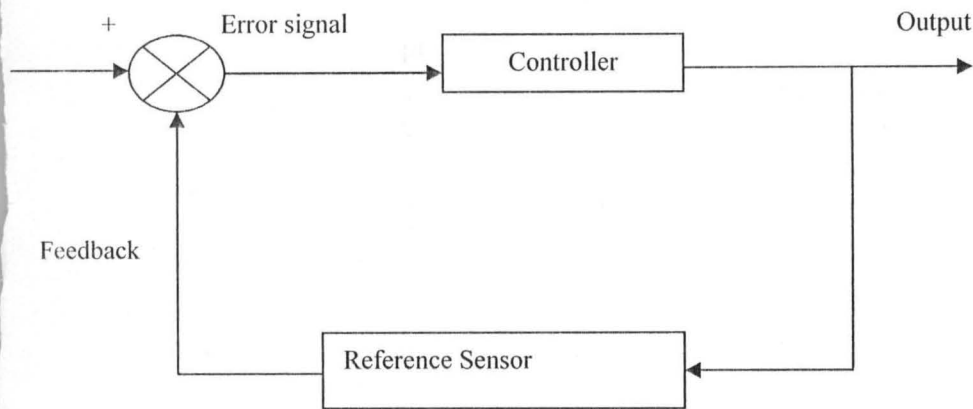


Figure 2.4 Block diagram for feedback control (Chemical Engineering Handbook, Perry and Green, 1997)

2.5 Elements of a control system

a) Variables

In attempting a design a control system that will satisfy the control needs for a chemical process, one must be able to identify and classify the variables associated with the chemical process.

The variables (flow rate, temperature, concentrations etc) associated with a chemical process are divided into two

- i) Input variables
- ii) Output variables.

The input variables can be further classified into two categories

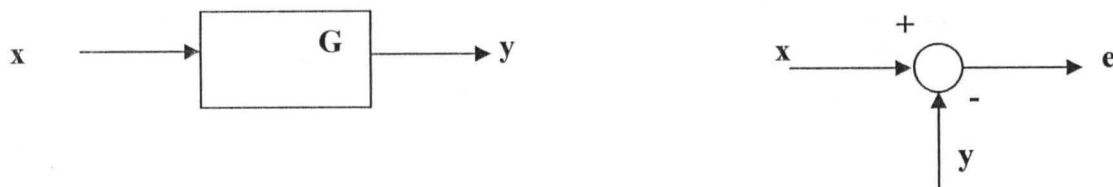
- i) Manipulated (or Adjustable) variables
- ii) Disturbances

The output variables are also classified into two

- i) Measured output variables.
- ii) Unmeasured output variables

b) Block Diagrams

For control problems, it is helpful to use a block diagram to show the functional relationship between input and output (Paul, 1996).



(a) $y = Gx$

$y = f(x, t)$

(b) $e = x - y$

Figure 2.5 Block diagram component (a) Dynamic relationship (b) comparison of signals

2.6 Chemical Reactors

Chemical reactors are vessels that are designed for a chemical reaction is taking place. It is an apparatus for holding substances that are undergoing a chemical reaction. The design of a chemical reactor deals with multiple aspects of chemical engineering. Vessels may be tanks (usually enclosed to keep contaminant out of the reaction vessel) or tubular (a pipe). Both types can be used as continuous reactor or batch reactors <http://www.en.wikipedia.org/wiki/reactors>. Chemical vessels must satisfy several requirements imposed by it designers and the general technical, economical and social conditions in presence of ever – changing external influence (disturbances). Among such requirements are the following;

- i) Production specifications
- ii) Operational constraints
- iii) Safety
- iv) Environmental regulations

v) Economics

The requirements listed above dictate the need for continuous monitoring of a chemical vessel and external intervention (control) to guarantee the satisfaction of the operational objectives.

Types of Reactors

The three main basic models are:

- Batch Reactor model (batch)
- Continuous Stirred-Tank Reactor model (CSTR)
- Plug Flow Reactor model (PFR).

2.6.1 Batch reactor

A batch reactor is used in chemical processes for small scale operation, for testing new processes that have not been fully developed, for the manufacture of expensive products and for processes that are difficult to convert to continuous operations. It is a reactor in which all the reactants are loaded at once, the reaction is allowed to proceed for a given time whereupon the mixture of unreacted material together with the products is withdrawn. Agitation serves to mix separate feeds initially and to enhance heat transfer. Batch reactors are popular in practice because of their flexibility with respect to reaction time and to the kinds and quantities of reactions that they process.

2.6.2 Continuous stirred-tank reactor (CSTR)

The continuous stirred-tank reactor or CSTR is that in which one or more fluid reagents are introduced into a tank reactor equipped with an impeller while the reactor effluent is removed. The impeller stirs the reagents to ensure proper mixing. Simply dividing the volume of the tank by the average volumetric flow rate through the tank gives the residence time, or the average amount of time a discrete quantity of reagent spends inside the tank.

2.6.3 Plug flow reactor (PFR)

A Plug flow reactor is a chemical reactor where the fluid passes through in a coherent manner, so that in the ideal case the residence time is the same for all fluid elements. It is the one in which one or more fluid reagents are pumped through a pipe or tube. The chemical reaction proceeds as the reagents travel through the PFR. In this type of reactor, the reaction rate is a gradient; at the inlet to the PFR the rate is very high, but as the concentrations of the reagents decrease and the concentration of the product(s) increases the reaction rate decreases.

2.6.4 State variables and state equations for the chemical process

In order to characterize a processing system (tank, batch reactor, continuous stirred tank reactor, etc) and its behaviour, the following are needed:

- i) a set of functional dependent quantities whose values will describe the natural state of a given system.
- ii) a set of equations, in the variables above will describe how the natural state of the given system changes with time.

For most of the processing systems of interest to a chemical engineer there are only three such fundamental quantities: mass, energy, and momentum. Quite often, though, the fundamental dependent variables cannot be measured directly and conveniently. In such cases other variables, which can be measured conveniently, are selected and when grouped appropriately they determine the value of the fundamental variables.

These characterized variables are called state variables and their values define the state of a processing system. The equations that relate the state variable (dependent variables) to the

various independent variables are derived from application of the conservation principles of the fundamental quantities and are called state equation.

The principle of conservation of a quantity S states that

$$\frac{\left(\begin{array}{l} \text{Accumulation of } S \\ \text{within a system} \end{array} \right)}{\text{time period}} = \frac{\left(\begin{array}{l} \text{flow of } S \text{ in} \\ \text{the system} \end{array} \right)}{\text{time period}} - \frac{\left(\begin{array}{l} \text{flow of } S \\ \text{out of the system} \end{array} \right)}{\text{time period}} + \frac{\left(\begin{array}{l} \text{Amount of } S \\ \text{generated within system} \end{array} \right)}{\text{time period}} - \frac{\left(\begin{array}{l} \text{amount of } S \text{ consumed} \\ \text{within a system} \end{array} \right)}{\text{time period}}$$

The quantity S can be any of the following fundamental quantities:

Total mass

Mass of individual component

Total energy

The balance equations for these quantities are given as

Total mass balance

$$\frac{d(\rho V)}{dt} = \sum_{i:\text{inlet}} \rho_i F_i - \sum_{j:\text{outlet}} \rho_j F_j \dots\dots\dots 2.1$$

Mass balance on component A

$$\frac{d(n_A)}{dt} = \frac{d(C_A V)}{dt} = \sum_{i:\text{inlet}} C_{Ai} F_i - \sum_{j:\text{outlet}} C_{Aj} F_j \pm rV \dots\dots\dots 2.2$$

Where the variables in the above equations are:

ρ = Density of the material in the system

ρ_i = Density of the material in the i^{th} inlet system

- ρ_j = Density of the material in the j^{th} outlet system
 V = Total volume of the system
 F_i = Volumetric flow rate of the i^{th} inlet system
 F_j = Volumetric flow rate of the j^{th} output stream
 C_A = Molar concentration (moles/volume) of A in the system
 C_{Ai} = Molar concentration of A in the i^{th}
 C_{Aj} = Molar concentration of A in the j^{th} output
 r = Reaction rate per unit volume for component A in the system

2.7 Artificial neural nets (ANNs)

The fundamental processing element of a neural network is a neuron. This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result. Figure 2.6 shows the relationship of these four parts.

traditional computing. This is achieved by simulating the four basic functions of natural neurons.

Figure 2.6 shows a fundamental representation of an artificial neuron.

In short, an artificial neural network is a method of solving problems through artificial intelligence, by building a system of circuits to simulate the human brain, includes behaviour (i.e. learning, walking and making discoveries). Through technical computational models inspired by simulating the neural structure of intelligent organisms, ANN's can be taught to acquire knowledge through experience. <http://www.answers.com/topic/artificialintelligence>. Referring to natural brains and their connections, ANN's are characterized by the meeting of a large amount of Artificial Neurons (AN), interlinked by a great number of connections (synapses) that process the information in a parallel way. The fact that the information is processed in a parallel way provides larger reliability and readiness because the information is shared only one time with all of the neurons of the following layer. Therefore, if there is a possible flaw in a neuron in the net then it does not cause the whole process to lose all the information since the information is already present in other neurons. This makes it possible to make a total recovery, or at least a partial recovery. Just like the human brain, ANN's store knowledge through experience. Therefore the more the ANN used the more the ANN learns.

Healthy ANN's divide into layers, where layer patterns represent the net. These layers extract the characteristics, and work as partial intermediaries that process the information in parts before exiting from the ANNs. As can be observed in figure 2.7, the treatment of information in RNN's happens in the following way: Each signal that arrives at the neuron multiplies the signal according to the number of synapses with signals (the value of the synapses is substituted during the process of training of the net). The resultant signals are added to obtain only one entrance

value in order to reach the threshold value. The information produces an exit and it is spread in the net. If this threshold is not reached the information is not considered relevant and is blocked.

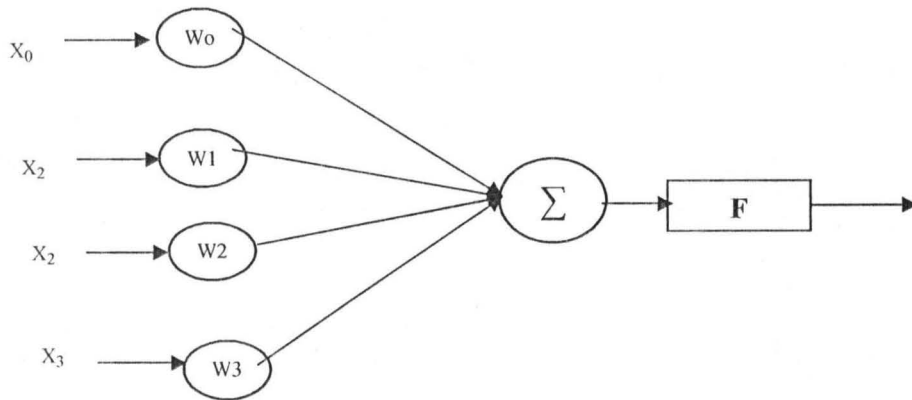


Figure 2.7 Artificial Neural Nets

2.7.1 The mathematical representation of a neuron

A first-order mathematical model for a neuron could be that shown in figure 2.8.

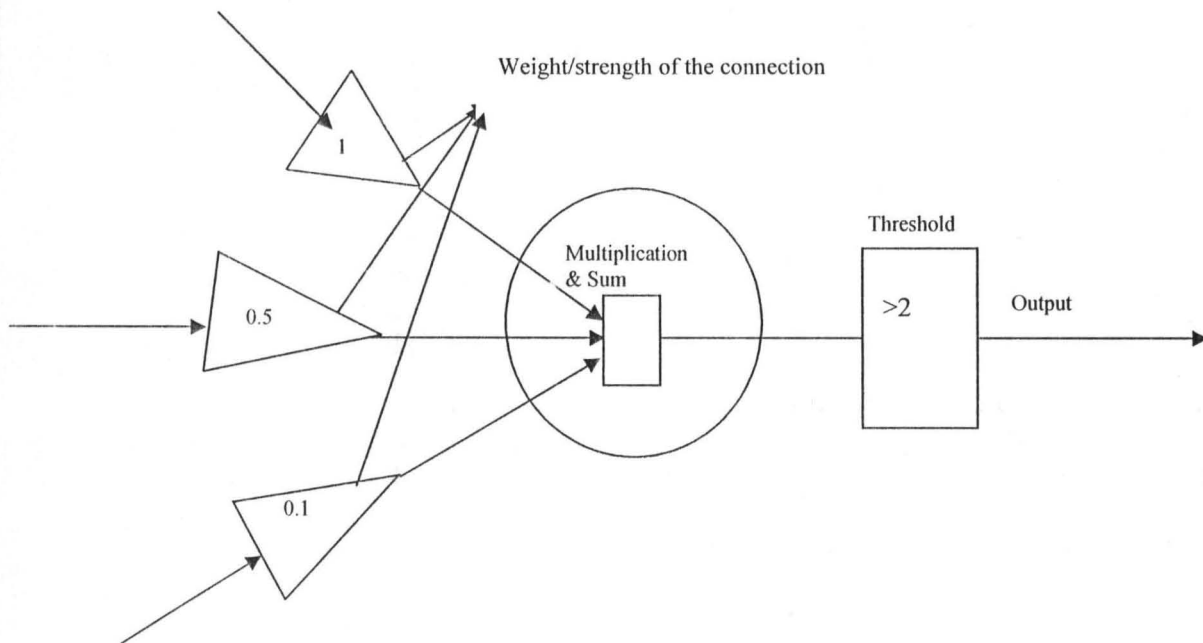


Figure 2.8 Mathematical Representation of a neuron

Incoming connections are represented by input lines with an associated weight. The neuron itself only performs accumulation and thresholding for incoming pulses from its inputs.

When a pulse comes from a connection, it is first multiplied by a number called the weight of the connection which assigns a certain importance to the connection (identical to the largeness of a biological dendrite), and then accumulates the overall result, passing the value through a threshold which emits a pulse when a certain value is reached. The output of the threshold stage is in turn connected to the inputs of several other neurons, which forms a complete network.

In practice, real input and output numbers are used to model the transactions between the different neurons. Instead of a biological threshold function, we use a mathematical function such as the sigmoid function $[1/(1+e^{-x})]$, arctangent, arcsine, etc. These functions should be smooth and continuous (i.e. you should not use a piecewise linear or step function) with lower and upper limit. They should also be differentiable.

2.8. Learning

Learning is the area of cognitive science that deals with the ability to segment the world into classes of equivalents. The mechanism by which intelligent system groups physically distinct objects into classes are among the most fundamental aspects of cognition. Without these mechanisms, every instant of each type of object, event or situation would appear new every time it is encountered.

2.8.1 Human learning

Learning is the neural process for which the experience modifies the behavior and is centrally regulated. Learning can be of an associative type or a non-associative type. In other words, it can

have direct relationship with incentives (as in associative learning); or not, depending on the relationship between incentives and answers (such as acclimatizing and sensitivity in non-associative learning). Conditioning is a reflex answer to an incentive that first produced little or no answer, and then produces an answer after repeated association with the incentive. Where there is no relationship with other incentives then the brain learns by searching through its “database” for similarities among the incentives that are available until they find some reasons (acclimatization, sensitivity) or learn that there is no relationship with other incentives in the brain. Every time a signal is received, this stimulates learning of their properties and informs the “database”.

2.8.2 Artificial neural networks learning

The process of learning in neural nets is accomplished when there are several significant modifications in the “synapses” of the “neurons”. Those changes happen in agreement with the activation of the neuron. The process of learning can be categorised into two general paradigms: associative mapping and regularity detection.

1. Associative mapping in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:

2. Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, that is, to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.

3. Hetero-association: is related to two recall mechanisms:

- **Nearest-neighbour** recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and
- **Interpolative** recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented.

4. Regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

2.8.3 Types of learning

Neural networks have 3 main modes of operation – supervised, reinforced and unsupervised learning (Hunt and Sbarbaro, 1992)

(i) Supervised learning is that which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. Supervised learning compares the output from the neural network with a set of targets; the error signal is used to update the weights in the neural network. The aim is to determine a set of weights which minimises the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

(ii) Reinforced learning is similar to supervised learning however there are no targets given, the algorithm is given a grade of the ANN performance.

(iii) Unsupervised learning uses no external teacher and is based upon only local information. It is also referred to as self-organisation, in the sense that it self-organises data presented to the

network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning.

2.9 Transfer function

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This activation function typically falls into one of three categories:

- linear (or ramp)
- threshold
- sigmoid

For **linear units**, the output activity is proportional to the total weighted output.

For **threshold units**, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For **sigmoid units**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations (Pham, 1995)

2.10 Learning algorithm using back propagation

Clearly, the uses of an efficient algorithm that will perfectly modify the different connection weights to minimize the errors at the output is very important. This is called optimization. The

famous LMS algorithm was developed to solve similar problems, however the neural network is a more generic system and requires a more complex algorithm to adjust the many network parameters.

One algorithm which has hugely contributed to neural network fame is the back-propagation algorithm. The principle advantages of back-propagation are simplicity and reasonable speed; although there are several modifications which can make it work faster (Daniel, 2003)

The training algorithm for a BPN consists of the following steps:

2.10.1 Selection and preparation of training data

The best training procedure is to compile a wide range of examples (for more complex problems, more examples are required) which exhibit all the different characteristics you are interested in. It is important to select examples which do not have major dominant features which are of no interest to you, but are common to your input data anyway (Daniel, 2003)

2.10.2 The back-propagation Algorithm - a mathematical approach

Units are connected to one another. Connections correspond to the edges of the underlying directed graph. There is a real number associated with each connection, which is called the weight of the connection. We denote by W_{ij} the weight of the connection from unit u_i to unit u_j . It is then convenient to represent the pattern of connectivity in the network by a weight matrix W whose elements are the weights W_{ij} . Two types of connection are usually distinguished: excitatory and inhibitory. A positive weight represents an excitatory connection whereas a

negative weight represents an inhibitory connection. The pattern of connectivity characterises the architecture of the network.

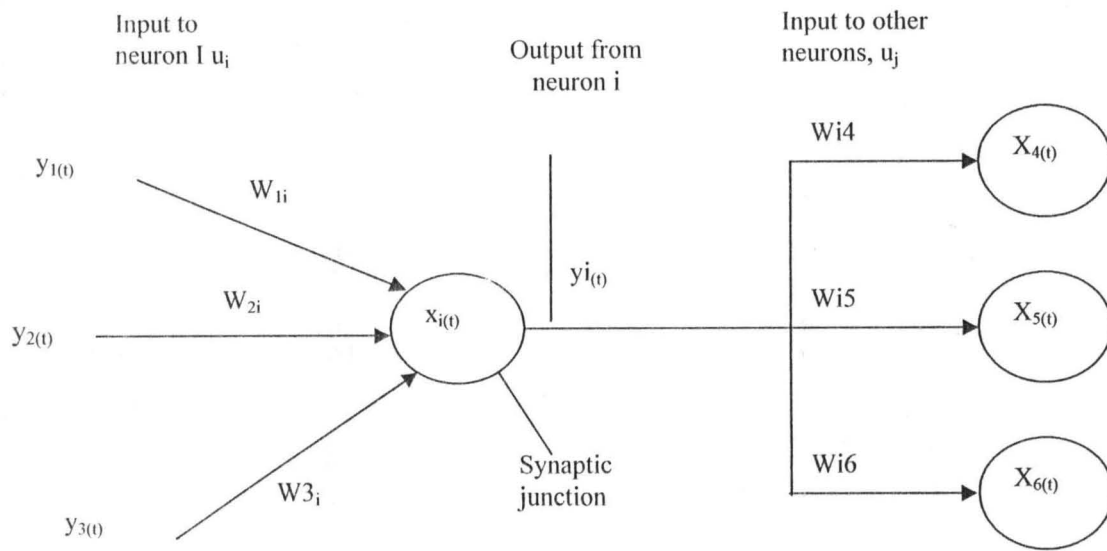


Figure 2.9 Back propagation Neural Networks Representation

A unit in the output layer determines its activity by following a two step procedure.

- a) First, it computes the total weighted input x_j , using the formula:

$$x_j = \sum_i y_i W_{ij} \dots \dots \dots 2.3$$

Where y_i is the activity level of the j th unit in the previous layer and W_{ij} is the weight of the connection between the i th and the j th unit.

- b) Next, the unit calculates the activity y_j using some function of the total weighted input.

Typically we use the sigmoid function:

$$y_j = \frac{1}{1 + e^{-x_j}} \dots \dots \dots 2.4$$

Once the activities of all output units have been determined, the network computes the error E, which is defined by the expression:

$$E = \frac{1}{2} \sum_i (y_i - d_i)^2 \dots\dots\dots 2.5$$

where y_j is the activity level of the j th unit in the top layer and d_j is the desired output of the j th unit.

The back-propagation algorithm consists of four steps:

1. Compute how fast the error changes as the activity of an output unit is changed. This error derivative (EA) is the difference between the actual and the desired activity.

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j \dots\dots\dots 2.6$$

2. Compute how fast the error changes as the total input received by an output unit is changed. This quantity (EI) is the answer from step 1 multiplied by the rate at which the output of a unit changes as its total input is changed.

$$EI_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_j} = EA_j y_j (1 - y_j) \dots\dots\dots 2.7$$

3. Compute how fast the error changes as a weight on the connection into an output unit is changed. This quantity (EW) is the answer from step 2 multiplied by the activity level of the unit from which the connection emanates.

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial W_{ij}} = EI_j y_j \dots\dots\dots 2.8$$

4. Compute how fast the error changes as the activity of a unit in the previous layer is changed. This crucial step allows back propagation to be applied to multilayer networks. When the activity of a unit in the previous layer changes, it affects the activities of all the output units to which it is connected. So to compute the overall effect on the error, we add together all these separate effects on output units. But each effect is simple to calculate. It is the answer in step 2 multiplied by the weight on the connection to that output unit.

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_j EI_j W_{ij} \dots \dots \dots 2.9$$

By using steps 2 and 4, we can convert the EAs of one layer of units into EAs for the previous layer. This procedure can be repeated to get the EAs for as many previous layers as desired. Once we know the EA of a unit, we can use steps 2 and 3 to compute the EWs on its incoming connections.

2.10.3 Modification of the neuron connection weights

Back propagation algorithm can be best understood by considering a very simple arrangement shown below.

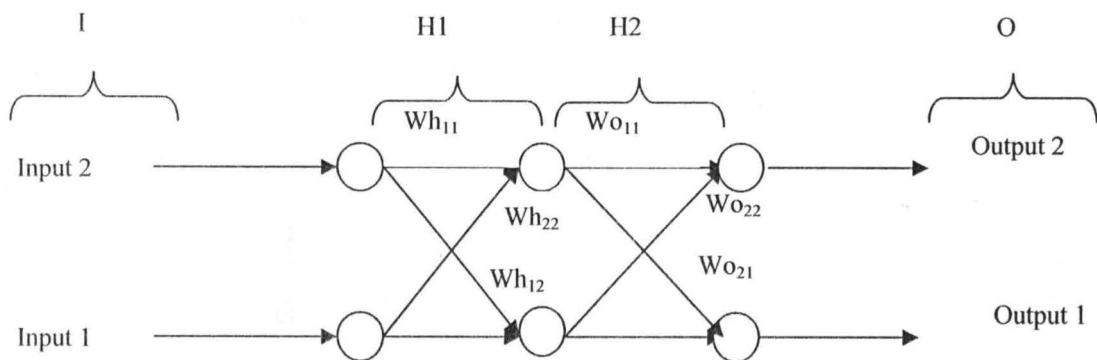


Figure 2.10. Back propagation Net (2.2.2)

Designating $(I_1 I_2)$, $(H_1 H_2)$ and $(O_1 O_2)$, as the inputs, hidden layer outputs and output – layer outputs respectively the output of Hidden Node 1 and 2 are given by

$$H_1 = \text{sgm} \left(\sum_{i=1}^2 I_i W_{i1}^h \right) \dots\dots\dots 30$$

$$H_2 = \text{sgm} \left(\sum_{i=1}^2 I_i W_{i2}^h \right) \dots\dots\dots 31$$

$$\text{Where } \text{sgm}(x) = \frac{1}{1 + e^{-x}} \dots\dots\dots 32$$

The output – layer outputs are given by

$$O_1 = \text{sgm} \left(\sum_{m=1}^2 H_m W_{m1}^o \right) \dots\dots\dots 33$$

$$\text{And } O_2 = \text{sgm} \left(\sum_{m=1}^2 H_m W_{m2}^o \right) \dots\dots\dots 34$$

Putting equations (1) and (2) in (4) and (5) respectively gives

$$O_1 = \sum_{m=1}^2 \text{sgm} \left(\sum_{l=1}^2 I_l W_{lm}^h \right) W_{m1}^o \dots\dots\dots 35$$

$$O_2 = \sum_{m=1}^2 \text{sgm} \left(\sum_{l=1}^2 I_l W_{lm}^h \right) W_{m2}^o \dots\dots\dots 36$$

Now we can calculate the output given a particular set of input. This allows us to calculate the mean square error (MSE) between the actual output and the desired output for the given input in the above figure. This is simply the average of the squares of the difference between what we want. The precise mean square error function is of importance, we do not need to divide by the number of outputs, and minimization algorithm will still find the correct minimum. Thus, the error function can be written as

$$E = \sum_{n=1}^2 (D_n - O_n)^2 \dots\dots\dots 37$$

Substituting (6) and (7) in (8) gives

$$E = \sum_{n=1}^2 \left(D_n - \text{sgm} \left(\sum_{m=1}^2 \text{sgm} \left(\sum_{i=1}^2 I_i W_{im}^h \right) W_{mn}^o \right) \right)^2 \dots\dots\dots 38$$

Substituting D_K is the desired output

Since the fact that the derivative of the sigma function can be expressed in terms of the function itself, the gradient is calculated as

$$\frac{d(\text{sgm})}{dx} = \frac{d\left(\frac{1}{1+1^{-x}}\right)}{dx} = \frac{1^{-x}}{(1+1^{-x})^2} = (1 - \text{sgm}(x))\text{sgm}(x) \dots 39$$

$$\frac{\partial O_n}{\partial W_{mn}^o} = \frac{\partial}{\partial W_{mn}^o} \left(\sum_{k=1}^2 W_{kn}^o H_k \right) = H_m \dots\dots\dots 50$$

Note $S^o = \sum_{k=1}^2 W_{mn}^o$

Now, the gradient of the error function can be calculated as

$$\begin{aligned} \frac{\partial E}{\partial W_{mn}^o} &= \frac{\partial}{\partial W_{mn}^o} \sum_{n=1}^2 (D_n - O_n)^2 \\ &= -2 (D_n - O_n) \frac{\partial}{\partial S^o} \text{sgm}(S^o) \frac{\partial S^o}{\partial W_{mn}^o} \\ &= -2 (D_n - O_n) \left((1 - \text{sgm}(S^o)) \text{sgm}(S^o) \right) H_m \dots\dots\dots 51 \end{aligned}$$

Where $\delta_n^o = -2 (D_n - O_n) \left((1 - \text{sgm}(S^o)) \text{sgm}(S^o) \right)$

The new values of the network weights are calculated by multiplying the negative gradient with a step size parameter (called the learning rate) and adding the resultant vector to the vector weights attached to the current layer.

However, if the error at the output layer will be affected by the weight at the middle layer a new gradient is derived, but the output weight are treated as constant rather than the hidden – layer.

This gives

$$\frac{\partial E}{\partial W_{in}^h} = \left((1 - \text{sgm}(S^h)) \text{sgm}(S^h) \right) \sum_{n=1}^2 \delta^o W_{mn}^o I_i \dots\dots\dots 52$$

The middle weights are updated using the same procedure as for the output layer, and the output layer weights are updated as well. This is a complete training cycle for one piece of training data.

It should be noted that the input layer is really only a buffer to hold the input vector. Therefore, it has no weights which need to be modified. For more than one hidden layer, the update procedure is quite similar.

2.11 Neural network structures

There are 3 main types of ANN structures -single layer feedforward network, multi-layer Feed forward network and recurrent networks.

2.11.1 Single layer feed forward network

The most common type of single layer feed forward network is the perceptron. Other types of single layer networks are based on the perceptron model. The details of the perceptron are shown below (Figure 2.8).

Inputs to the perceptron are individually weighted and then summed. The perceptron computes the output as a function F of the sum. The activation function, F is needed to introduce nonlinearities into the network. This makes multi-layer networks powerful in representing nonlinear functions.

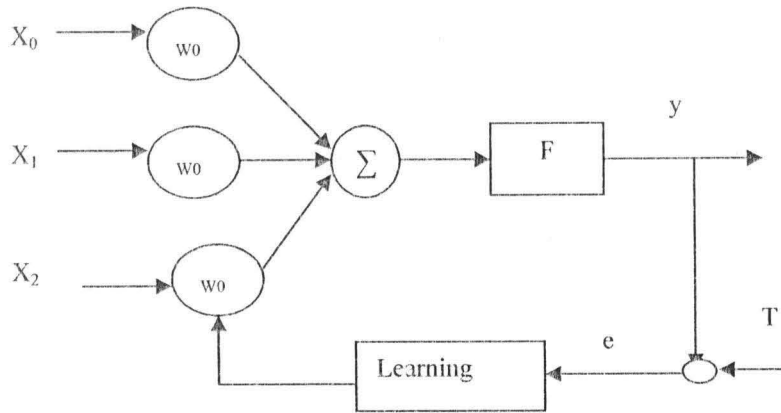


Figure 2.11: Diagram of the perceptron model

The output from the perceptron is

$$y[k] = f(w^T[k] * x[k]) \dots\dots\dots 53$$

The weights are dynamically updated using the back propagation algorithm. The difference between the target output and the actual output (error) is calculated.

$$e[k] = T[k] - y[k] \dots\dots\dots 54$$

The errors are back propagated through the layers and the weight changes are made. The formula for adjusting the weights is

$$w[k+1] = w[k] + \mu * e[k] * x[k] \dots\dots\dots 55$$

Once the weights are adjusted, the feed-forward process is repeated. The weights are adapted until the error between the target and actual output is low. The approximation of the function improves as the error decreases. Each connection branch is described by a weight representing the strength of connection between two linked nodes. The so called learning or training process is the procedure to adjust the weights. Single-layer feedforward networks are useful when the data to be trained is linearly separable. If the data we are trying to model is not linearly separable or the function has complex mappings, the simple perceptron will have trouble trying to model the function adequately (Tim, 2003).

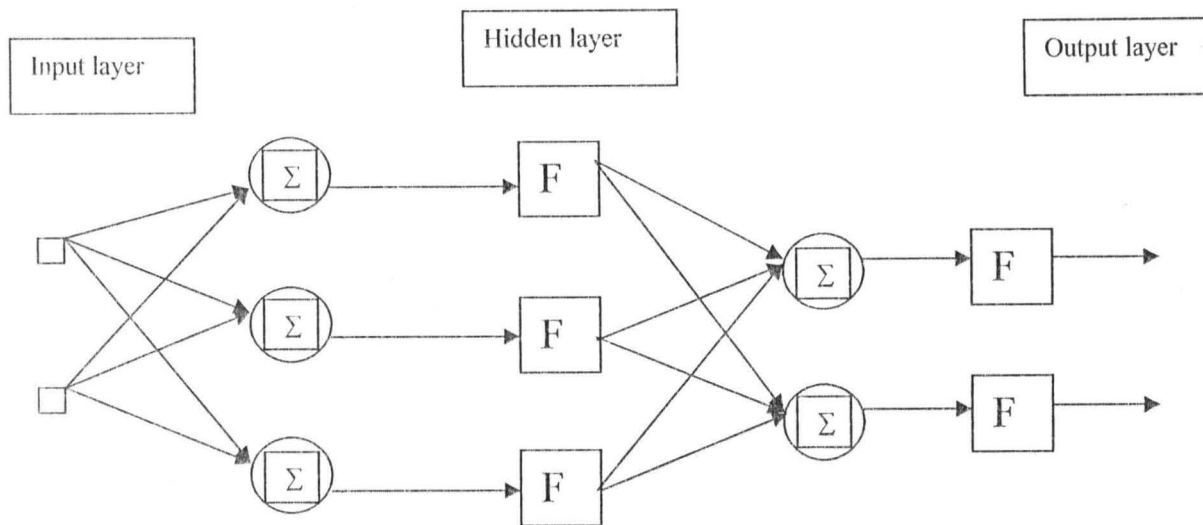


Figure 2.12 Diagram of a multi-layered Perceptrons

2.11.3 Recurrent networks

The second type of multi-layer networks is recurrent (Figure 2.11). Recurrent networks have at least one feedback loop. This means an output of a layer feeds back to any preceding layer. This gives the network partial memory due to the fact that the hidden layer receives data at time t but also at time $t-1$. This makes recurrent networks powerful in approximating functions depending on time. The Simulink model for the nonlinear continuous stirred tank reactor shows that there are many feedback loops. This means the next state of the model depends on previous states (Narendra and Parthasarathy, 1990). It is expected that to accurately model this type of dynamic system, a recurrent neural network with feedback loops will perform better than a static feedforward network.

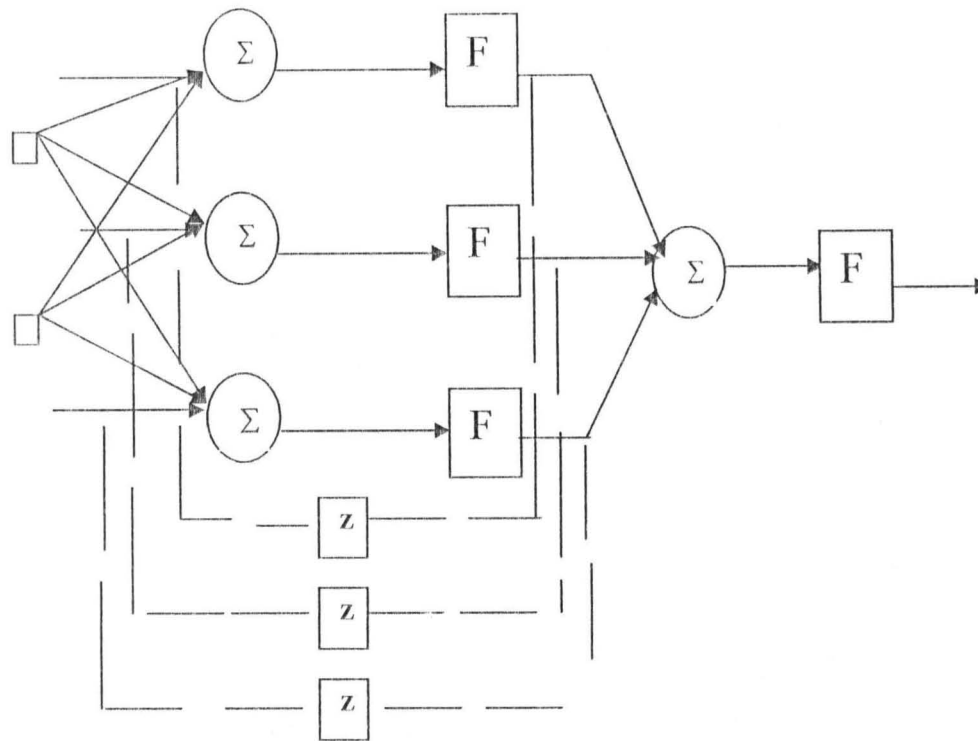


Figure 2.13 Diagram of a Recurrent Neural Network

2.12 ANN control strategies

These are a number of control strategies which are based on some type of process model.

Most of these control designs use a forward and/or an inverse linear parametric model (Narendra and Parthasarathy, 1990).

These linear model control strategies are well established and often benefit from the ability to incorporate robustness more directly in the controller design. It has been proposed that many of these model based control strategies could employ neural network models and could, thus, benefit from the nonlinear approximation properties of ANNs. The below reviews the main types of ANN based control structures.

2.12.1 Supervised control:

Supervised control involves a mechanism of providing the network with the desired output either by manually “grading” the network’s performance or by providing the desired outputs with the inputs. It is then possible to teach a neural network the correct actions by using an existing

controller or human feedback. This type of control is called supervised learning. Most traditional controllers (feedback linearisation, rule-based control) are based on an operating point. This means that the controller can operate correctly if the plant/process operates around a certain point. These controllers will fail if there is any sort of uncertainty or change in the unknown plant. The advantage of neuro-control is that if an uncertainty in the plant occurs the ANN is capable of adapting its parameters and maintains controlling the plant when other robust controllers would fail. In supervised control, a teacher provides correct actions for the neural network to learn as shown in Figure 2.14. In offline training the targets are provided by an existing controller, the neural network adjusts its weights until the output from the ANN is similar to the controller (Hagan and Demuth, 1996).

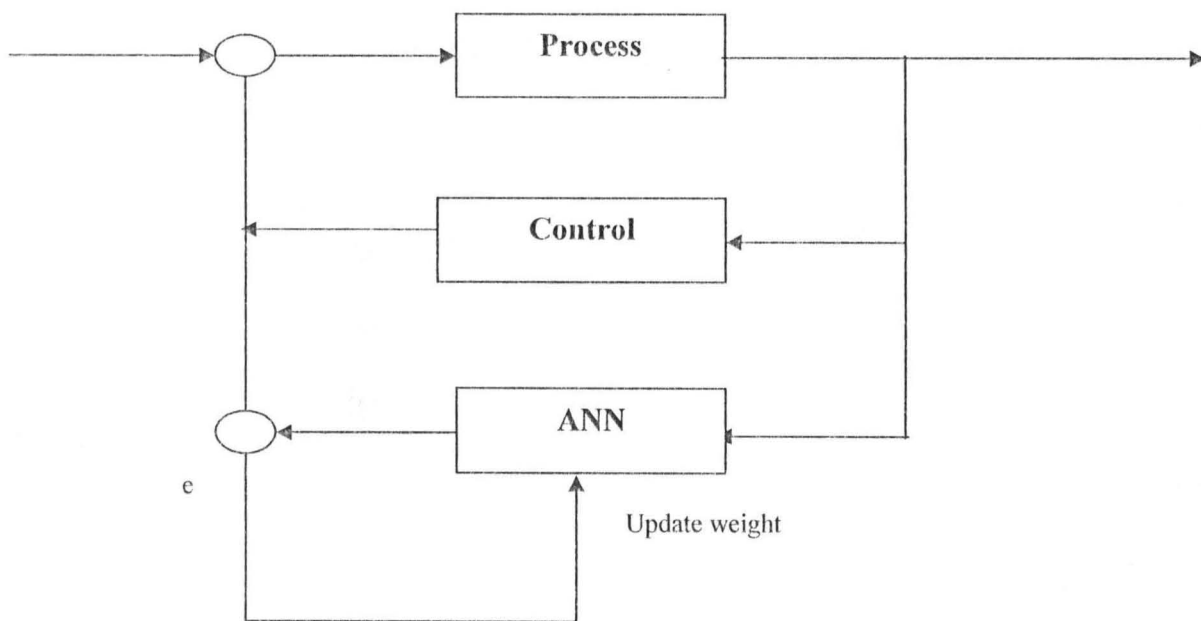


Figure 2.14 Supervised learning using an existing controller

When the neural network is trained, it is placed in the feedback loop. Because the ANN is trained using the existing controller targets, it should be able to control the process.

2.12.2 Adaptive neural control

Adaptive neural control is an ANN which controls the process similar to the existing controller. The real advantage of neuro-control is the ability to be adaptive online. (Figure 2.13) An error signal (desired signal – real output signal) is calculated and used to adjust the weights online (Tim, 2003).

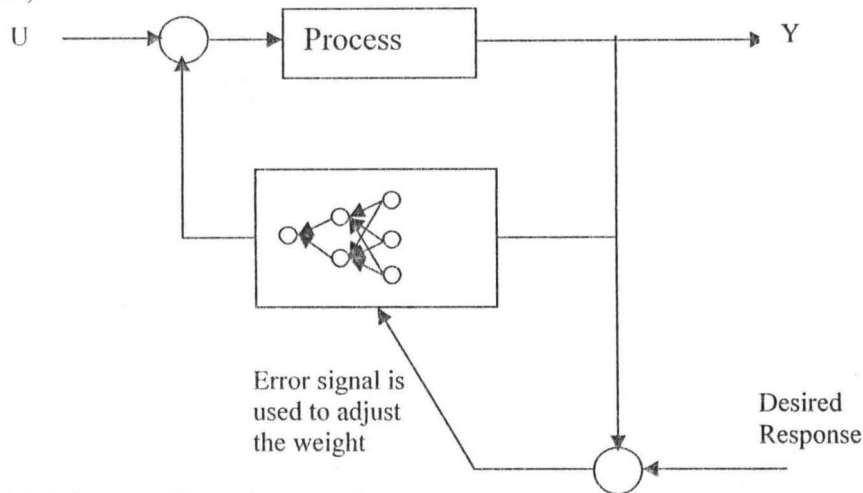


Figure 2.15 Adaptive Neural Control

If a large disturbance/uncertainty occurs in the process- the large error signal is fed back into the ANN and this adjusts the weights so that system remains stable.

2.12.3 Inverse models

Inverse process models play a central role in some model based control structures. The ANN models considered so far are called forward models since the direction of information flow through the model is from process input to process output. Conversely, the direction of information flow through an inverse process model is opposite to that of the process, and thus, an inverse model predict the manipulated variable. An inverse ANN model can be developed using a direct or indirect training structure. Conceptually the simplest approach is direct inverse learning which is shown in Figure 2.14 (Tim, 2003). The process manipulated variable is applied to the process and the process output is used, together with lagged process I/O values as the

ANN inputs. The ANN output is then compared with the process input to produce an error signal which is used to train the neural network. Intuitively, this approach will force the ANN to represent the process inverse. This is a disadvantaged and cannot be goal directed (Jordan and Rumelhart, 1991).

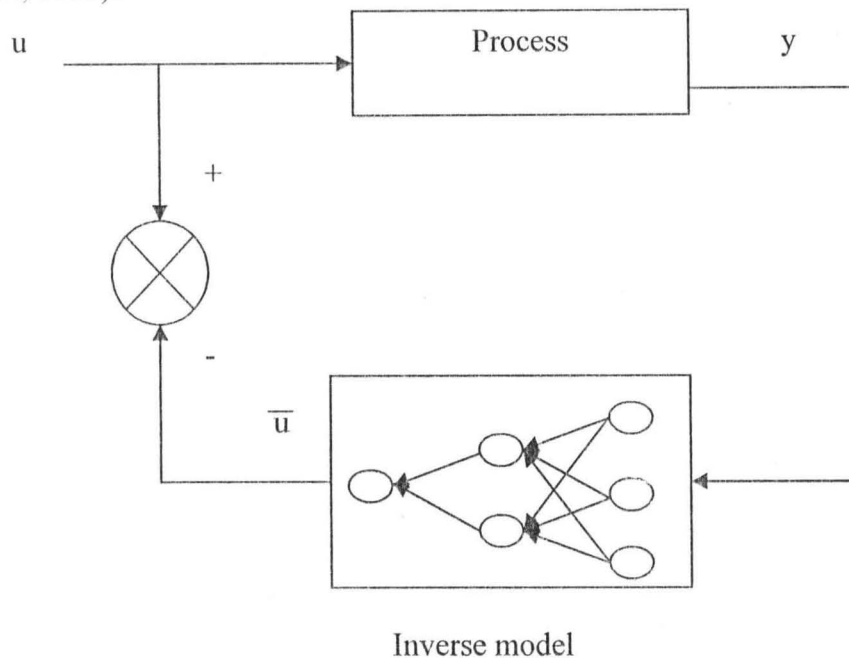


Figure 2.16 Direct inverse modelling

The root of the problem with direct inverse learning is that the training of the inverse model attempts to minimise the manipulated variable error and this does not correspond to the control objective which is to minimise the process output error.

2.12.4 Specialised inverse modelling

The specialised inverse modelling method overcomes the above by using the process output error (Figure 2.15), to generate the error signal. This error signal is passed back through the forward model to give the manipulated variable error which is used to train the inverse model. For online learning, the forward model output error forces the inverse model into different regions of the process input space which correspond to the operating region spanned by the set-point, r . Hence

the specialised inverse modelling method is goal directed (Psaltis et al, 1988). Using the forward model rather than the process to generate the error signal can be advantageous for a noisy process or when use of the real process is not viable, (Anderson, 1989 and Economou and Morari, 1986).

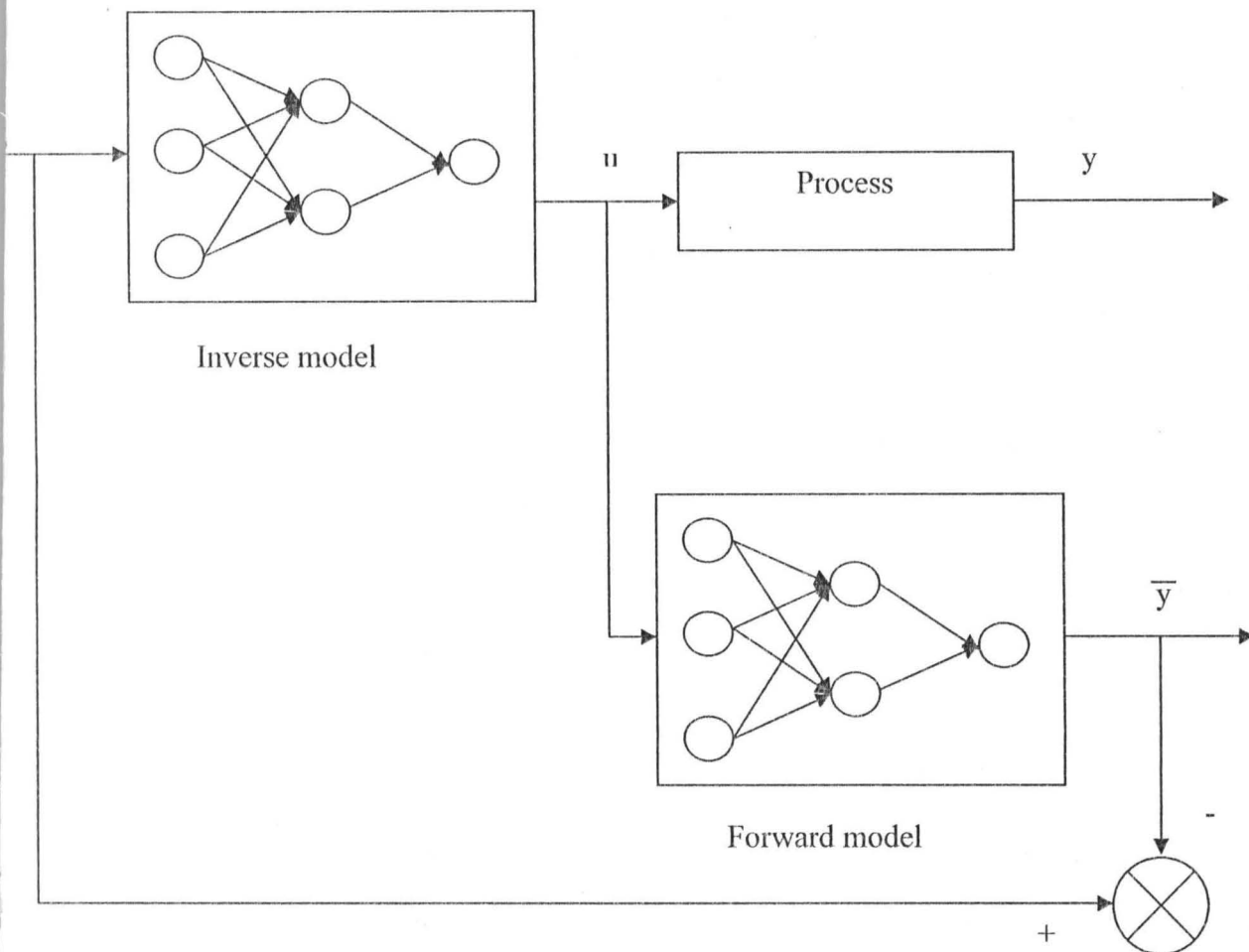


Figure 2.17 Specialised inverse modelling

2.12.5 Direct inverse control

The simplest use in an inverse process model for control is to place it in front of the process so that the composite system results in the identity mapping from set point to process output (Figure 2.16). This approach is called Direct Inverse Control and has been implemented mainly in robotic applications. However, the absence of feedback in the Direct Inverse Control structure

results in a lack of robustness for the practical case of an imperfect inverse model (Hunt and Sbarbaro, 1992).

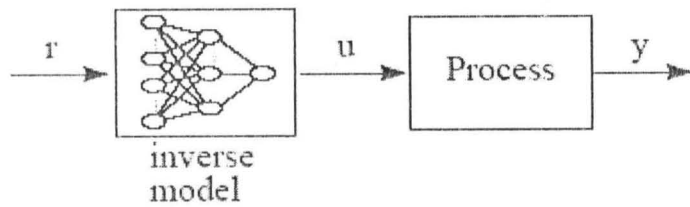


Figure 2.18 Direct Inverse Controls

2.12.6 Internal model control

Linear Internal Model Control (IMC) (Garcia and Morari, 1982) has been extensively studied and many robustness and stability results have been proven. For open-loop stable systems some of these results can be extended to non-linear IMC (Sean, 1999) although perfect forward and inverse models are generally assumed. The use of neural networks in the IMC structure has been proposed by several workers [Bhatt and Mcavoy, 1992 and Hunt and Sbarbaro, 1992) and has been implemented for the control of simulated processes.

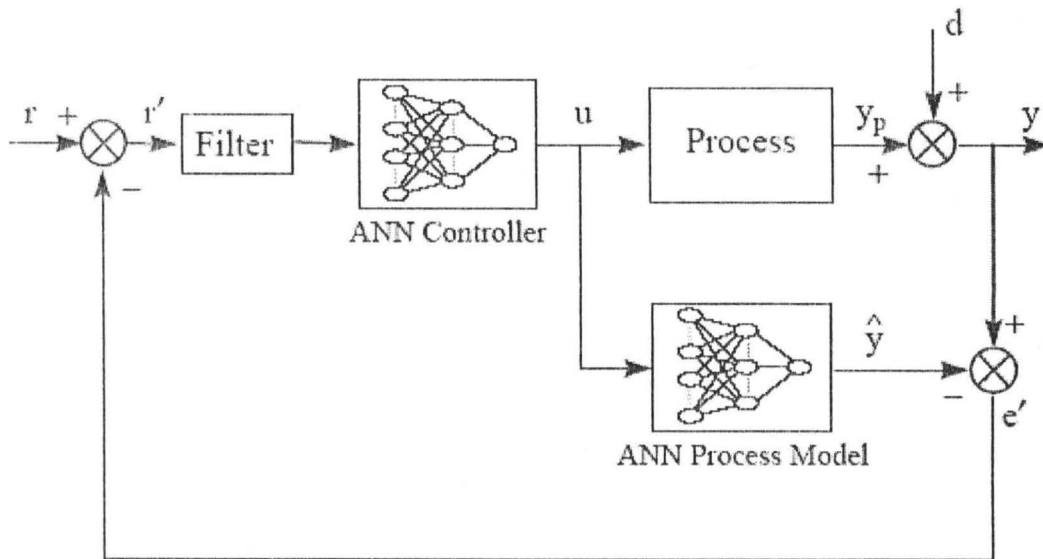


Figure 2.19 The Internal Model Control Structure

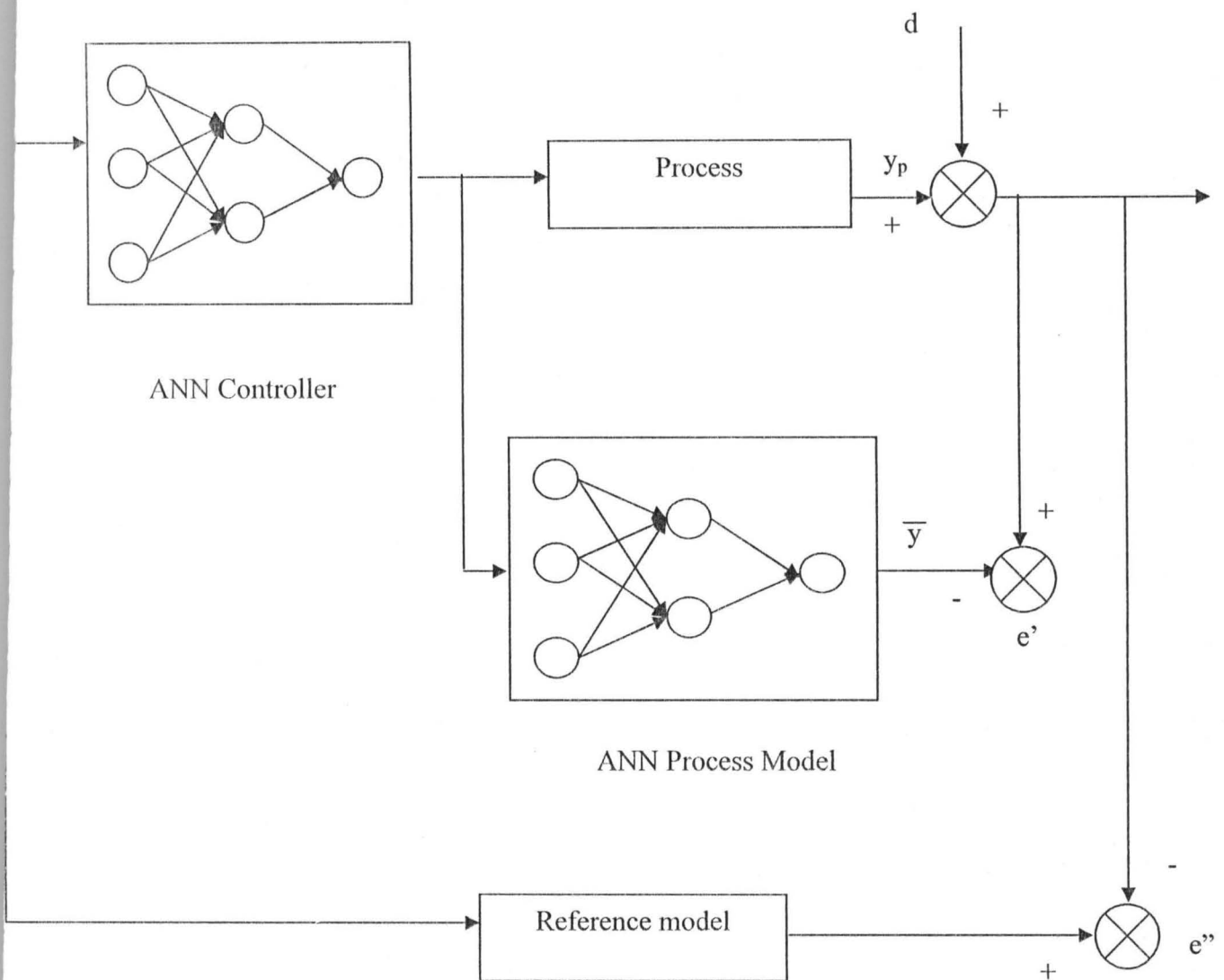


Figure 2.20 The Model Reference Control Structure

The performance of model reference control is highly dependent on the choice of reference model.

2.7.8 Model predictive control

It has been postulated that Linear Model Predictive Control (LMPC) is the most important control technology for the process industries since the PID controller (Vandoren, (1997). Figure

2.19 shows a neural network MPC scheme which, in contrast to the other reviewed control strategies, does not incorporate an inverse process model. The non-linear optimiser in ANN-MPC is used to select the manipulated variable that minimises a cost function, which is quadratic in the set-point/process output error. To do so, the non-linear optimiser uses the ANN process model to predict the possible future responses of the process to different possible future manipulated variable sequences and the current measured disturbances (Tim, 2003 and Garcia and Morari, 1982). By using the ANN model to predict multi-step ahead, the control scheme can anticipate the process trajectory and compensate for measured disturbances before their impact on the process output is detected. In common with IMC, the process/model mismatch, e' , is used for feedback purposes and the filter adds robustness to the control system.

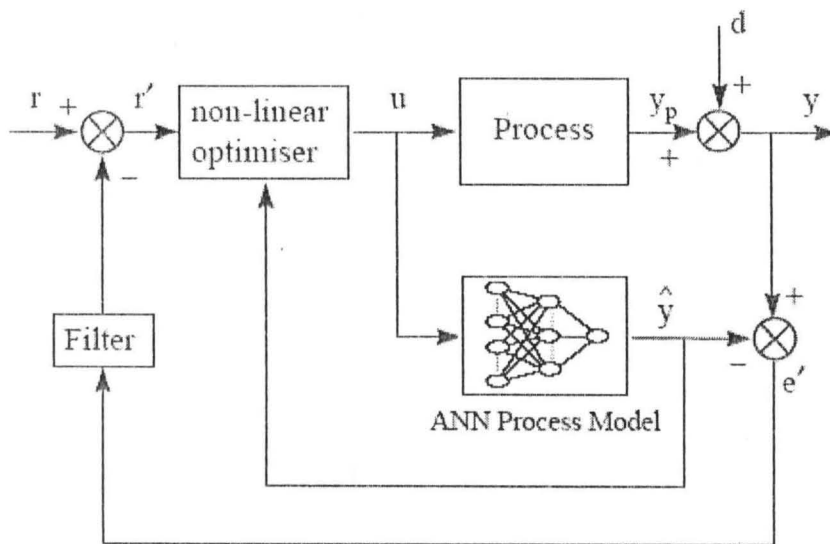


Figure 2.21 The Model Predictive Control Structure

2.7.9 Unsupervised control

The previous neural control methods are all trained using a priori knowledge such as an explicit teacher providing correct actions. In unsupervised learning set-up, no existing

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

3.1 Description of the Process

The reversible, exothermic reaction is suitable to test PID controller performance due to its non-linearity, incomplete conversion and lack of stability. A reversible, exothermic reaction shown below was carried out in a single perfectly mixed continuous stirred tank reactor; Figure 3.1 shows the inputs and output of a system. The continuous stirred tank reactor system is a non-linear process and to adequately model it, non-linear methods using neural networks analysis can be used. The mathematical model in this case is the black box, it describes the relationship between the input and output signals. The reaction is described by equation 3.1. It is a first order reversible reaction and has a heat of reaction ΔH . The heat of reaction was removed by an incorporated cooling jacket surrounding the reactor. Cooling water is added to the jacket at the rate F and the inlet temperature $T_{c,in}$.

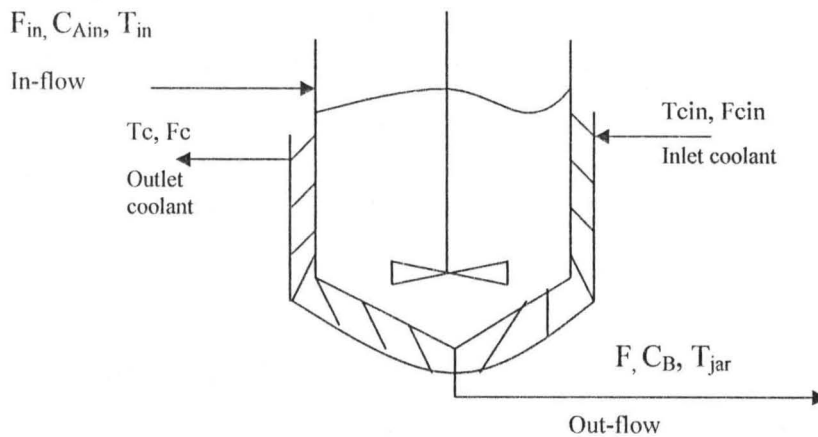


Figure 3.1 Continuous Stirred Tank Reactors with Cooling Jacket

The dynamic equations of the system were developed. The model was then simulated using Simulink approach and a basic controller was used.

The first things considered before the model can be developed in Simulink were the dynamic equations that describe the continuous stirred tank reactor. These equations were derived using the total continuity equation, that is, the material, components and energy balance equations.

The basic assumptions considered for the easy derivation of our dynamic equations are given below:

- (i) Constant densities
- (ii) Constant flow rate
- (iii) Perfectly mixed jacket water
- (iv) Constant water volume in the jacket
- (v) The feed is free of product B
- (vi) Constant physical and thermo-chemical properties
- (vii) The mass of the metal wall is negligible, the thermal inertia need not be considered.

3.2 Model Equations

The equations that describing the system are:

3.2.1 Reactor total continuity equation.

$$\left(\begin{array}{l} \text{Time rate of} \\ \text{change of mass} \\ \text{inside the system} \end{array} \right) = \left(\begin{array}{l} \text{Mass flow} \\ \text{in} \end{array} \right) - \left(\begin{array}{l} \text{Massflow} \\ \text{out} \end{array} \right) \dots\dots\dots 3.2$$

That is, $\frac{\partial(V\lambda)}{\partial t} = F_{in} \rho - F \rho \dots\dots\dots 3.3$

controllers can be imitated and the ANN doesn't have a target to compare to its output. The ANN must try different states and determine which state produces a good output. Learning from experience during periods of no performance feedback is difficult (Barto, Sutton et al al, 1983).

ANNs are used in many applications where the input-output relationship is not known or is too complex to model. They are used in pattern recognition, speech recognition, and many other areas. They are also used in control systems where the system dynamics are unknown or too complex to model.

3.2.2 Reactor components continuity equations

i) For Component A

$$\left(\begin{array}{l} \text{time rate of change of } A \\ \text{within the system} \end{array} \right) = \left(\begin{array}{l} \text{moles of } A \\ \text{into the system} \end{array} \right) - \left(\begin{array}{l} \text{Moles of } A \\ \text{out of the system} \end{array} \right) \pm \left(\begin{array}{l} \text{Amount of} \\ \text{A disappeared /} \\ \text{generated} \end{array} \right) \dots 3.4$$

$$V \frac{dC_A}{dt} = F_{in} C_{Ain} - FC_A + k_2 C_B V - k_1 C_A V \dots 3.5$$

$$\frac{dC_A}{dt} = \frac{F_{in}}{V} C_{Ain} - F/V C_A + k_2 C_B - k_1 C_A \dots 3.6$$

Consider constant flow rate, F

$$\frac{dC_A}{dt} = \frac{F}{V} (C_{Ain} - C_A) + k_2 C_B - k_1 C_A \dots 3.7$$

The effect of temperature on the reaction rate k is usually found to be exponential:

$$k = k_0 \exp\left(-\frac{E_i}{RT}\right) \dots 3.8$$

where k_0 is an Arrhenius factor.

$$K_1 = k_{10} e^{-\frac{E_1}{RT}} \text{ and } K_2 = k_{20} e^{-\frac{E_2}{RT}} \dots 3.9$$

$$\therefore \frac{dC_A}{dt} = \frac{F}{V} (C_{Ain} - C_A) + k_{20} e^{-\frac{E_2}{RT}} C_B - k_{10} e^{-\frac{E_1}{RT}} C_A \dots 3.10$$

ii) For component B

$$\left(\begin{array}{l} \text{time rate of} \\ \text{change of } B \text{ within the system} \end{array} \right) = \left(\begin{array}{l} \text{Moles of } B \\ \text{into the system} \end{array} \right) - \left(\begin{array}{l} \text{Moles of } B \\ \text{out of the system} \end{array} \right) \pm \left(\begin{array}{l} \text{Amount of } B \\ \text{generated /} \\ \text{disappeared} \end{array} \right) \dots 3.11$$

$$\frac{VdC_B}{dt} = F_{in}C_{Bo} - FC_B - K_2C_BV + K_1C_AV \dots\dots\dots 3.12$$

$$\frac{dC_B}{dt} = \frac{F}{V}(C_{Bo} - C_B) + K_1C_A - K_2C_B \dots\dots\dots 3.13$$

$$\frac{dC_B}{dt} = K_1C_A - K_2C_B - \frac{F}{V}C_B \dots\dots\dots 3.14$$

$$K(T) = K_0 e^{\frac{-E}{RT}}$$

$$\frac{dC_B}{dt} = \frac{F}{V}(C_{Bo} - C_B) + K_{10}e^{\frac{-E_1}{RT}}C_A - K_{20}e^{\frac{-E_2}{RT}}C_B \dots\dots\dots 3.15$$

3.2.3 Energy balance equation for the reactor

$$\left(\begin{array}{l} \text{Rate of energy} \\ \text{into the system} \end{array} \right) - \left(\begin{array}{l} \text{Rate of Energy} \\ \text{out of the system} \end{array} \right) + \left(\begin{array}{l} \text{rate at which} \\ \text{heat is added} \\ \text{to the system} \end{array} \right) - \left(\begin{array}{l} \text{Rate at which} \\ \text{heat is removed} \\ \text{by the coolant} \end{array} \right) = \left(\begin{array}{l} \text{Rate of} \\ \text{change of} \\ \text{energy within} \\ \text{the system} \end{array} \right) \dots\dots\dots 3.1$$

$$\rho C_p F (T_{in} - T) + \Delta Hr V - Q = \rho V C_p \frac{dT}{dt} \dots\dots\dots 3.17$$

Dividing through by $\rho V C_p$

$$\frac{dT}{dt} = \frac{F}{V}(T_{in} - T) + \frac{\Delta Hr}{\rho C_p} - \frac{Q}{\rho V C_p} \dots\dots\dots 3.18$$

Where $\tau_A = -K_1C_A + K_2C_B$ and $Q = UA(T - T_{jar})$

$$\frac{dT}{dt} = \frac{F}{V}(T_{in} - T) - \frac{\Delta Hr}{\rho C_p}(-K_1C_A - K_2C_B) \left[\frac{UA}{\rho V C_p}(T - T_{jar}) \right] \dots\dots\dots 3.19$$

$$\frac{dT}{dt} = \frac{F}{V}(T_{in} - T) + \alpha K_1C_B - \alpha K_2C_B - \beta(T - T_{jar}) \dots\dots\dots 3.20$$

Where $\alpha = \frac{\Delta Hr}{\rho C_p}$ and $\beta = \frac{UA}{\rho V C_p}$

Using Arrhenius equation

$$\frac{dT}{dt} = \frac{F}{V}(T_{in} - T) + \alpha k_{10}e^{\frac{-E_1}{RT}}C_A - k_{20}e^{\frac{-E_2}{RT}}C_B - \beta(T - T_{jar}) \dots\dots\dots 3.21$$

3.2.4 Energy balance equation for the cooling jacket

$$\left(\begin{array}{l} \text{Time rate of change} \\ \text{of energy within} \\ \text{the jacket} \end{array} \right) = \left(\begin{array}{l} \text{Flow of heat} \\ \text{into the jacket} \end{array} \right) - \left(\begin{array}{l} \text{Flow of heat} \\ \text{out of the} \\ \text{jacket} \end{array} \right) - \left(\begin{array}{l} \text{heat removed} \\ \text{by the jacket} \\ \text{water} \end{array} \right) \dots 3.22$$

$$\rho_c V C_p \frac{dT_c}{dt} = C \rho_c F_c P_c (T_c - T_{jar}) + UA(T - T_{jar}) \dots 3.23$$

$$\frac{dT_c}{dt} = \frac{F_c}{V_c} (T_c - T_{jar}) + \frac{UA}{(\rho V C_p)_c} (T - T_{jar}) \dots 3.24$$

The four differential equations that described the system are

$$\frac{dC_A}{dt} = \frac{F_{in}}{V} (C_{Ain} - C_A) + k_{20} e^{\frac{-E_2}{RT}} C_B - k_{10} e^{\frac{-E_1}{RT}} C_A \dots (a)$$

$$\frac{dC_B}{dt} = \frac{F}{V} (C_{B0} - C_B) + k_{10} e^{\frac{-E_1}{RT}} C_A - k_{20} e^{\frac{-E_2}{RT}} C_B \dots (b)$$

$$\frac{dT}{dt} = \frac{F}{V} (T_{in} - T) + \alpha k_{10} e^{\frac{-E_1}{RT}} C_A - k_{20} e^{\frac{-E_2}{RT}} C_B - \beta (T - T_{jar}) \dots (c)$$

$$\frac{dT_c}{dt} = \frac{F_c}{V_c} (T_c - T_{jar}) + \frac{UA}{(\rho V C_p)_c} (T - T_{jar}) \dots (d)$$

At this stage, a set of nonlinear equations describing the continuous stirred tank reactor (CSTR) have been developed. The next stage is constructing a Simulink model of the CSTR system. (Simulink blocks Toolbox Library, MATLAB 7, and appendix B figure 1).

Steps

To begin modelling, start simulink, open a new model, an empty simulink window opens and save the model as freshplant.mdl. (Appendix B, figure 1)

1. In the Simulink Library Browser window, double-click simulink, and then double-click Sources. Click-and-drag any desired block; Constant, Product, Add,

Subtract, Divide, Uniform Random Number, Clock, Ramp, Random Number, Sine Wave, or Step to the simulink window.

2. Double-click Commonly Used Blocks. Click-and-dragging any desired block; Gain, In, Out, Product, Scope, Integrator, Sum or Terminator to the simulink window.
3. Double-click Sinks. Click-and-drag the next desired block; Display, Scope, To Workspace, XY Graph, Transfer Fcn, Derivative, Math Fcn, or Fcn to the Simulink window.
4. Double-click Math Operations. Click and drag, the next required block; Add, Derivative, Terminator, Math Fcn or Fcn.
5. Double-click control system. Click-and-drag Model Reference Control blocks and X (2Y) Graph.
6. Connect the constant block (F) to the Gain block (F/V) and the output to a product block. At the same time the constant block (C_A in) and integrator (C_A) were connected to Add block. The output of it to the same product block.
7. The function block (Fcn) together with a branch from C_A line was linked to another product block. The output was then linked to Add block. Similarly, another function block together with a branch from C_B line was linked to the same Add block.
8. The outputs from Product1 and Subtract blocks were linked to Add2 block. The output of this was linked to an Integrator, which upon integration gives C_A . The line is also link to X (2Y) Graph to study its output response or to a Display block where its exact value can be read. The link lines were produced by clicking and dragging from one block to the next. The simulink model module for the first dynamic equation was modelled, and is as shown below (Figure 3.2). In the same

way, the other dynamic equations were modeled and masked together as a reactor. These are shown in the appendix. After modeling, each block was double-clicked and dialog box opens, the required parameters were entered accordingly into the text box. Having created the Simulink model for the CSTR, the simulation of the model was run to see its response.

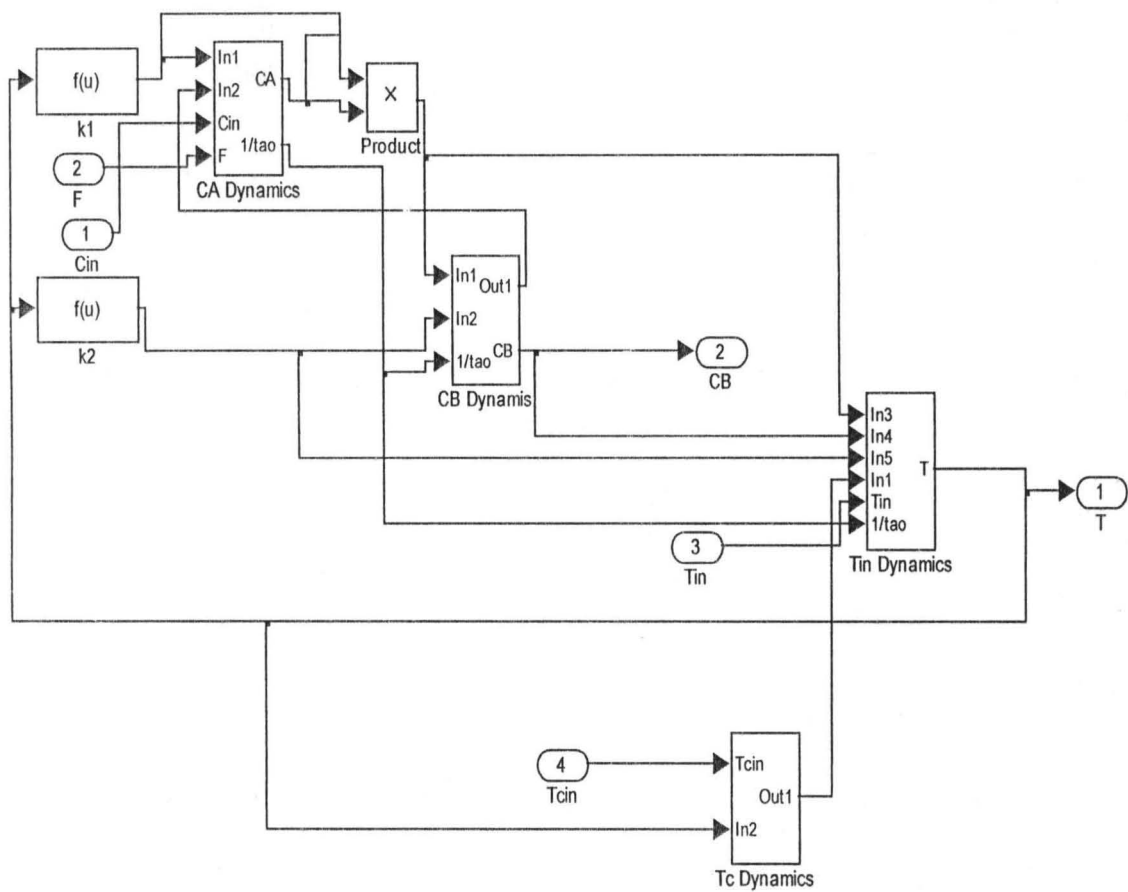


Figure 3.2: Simulink Model Connection Procedures for the Reactor

Having created the simulink model for the CSTR, I ran the simulation model to see its response. Similar procedure was adopted for designing the controllers. The diagram shown in the Appendix B is the non-linear CSTR Simulink model for the dynamic

equations describing the reversible reaction given above. The model was constructed using constants, integrators, sums, products, gains function blocks etc. The method though complicated is the Simulink representation of the non-linear state equation (Tim, 2003). The model is large so it was encapsulated in subsystem blocks shown in Appendix B, Figure 1.

The model was set up using a mask which makes it possible to change the value of any of the parameter used at will for different simulation.

3.4 Case Study

For the reaction described in figure 3.1 variables to be controlled are product concentration and temperature of the reacting mass. The nominal operating conditions and parameters for the above problem are shown in Table 3.1, and are used for the simulation of the model developed.

Table 3.1: Nominal Operating Condition and Parameter Value for the Simulation of Reversible Exothermic Reaction in CSTR.

Notation	Description	Value and units
ρ, ρ_c	Density of solution	1kg/L
V	Reactor volume	100L
C_p, C_{pc}	Heat capacity of solution	4184J/kgK
ΔH_r	Heat of reaction	-20920J/mol
E_1	Activation energy(forward reaction)	-41840J/mol
E_2	Activation energy(backward reaction)	-62760J/mol
UA	Heat transfer coefficient	418400J/min.K
F_{in}	Feed flow rate	1.6 L/min
C_{Ain}	Concentration of A in feed	1 mol/L
C_B	Concentration of B in product	0.0
T_{in}	Feed temperature	427K
T_{cin}	Coolant inlet temperature	300K
k_{10}	Reaction rate coefficient(forward)	$5 \cdot 10^3$ L/min
k_{20}	Reaction rate coefficient(backward)	$1 \cdot 10^6$ L/min
R	Ideal gas constant	8.314J/mol.K

Sources: William L. Luyben, 1999, "Design and Control of Gas-Phase Reactor/Recycle Processes with Reversible Exothermic Reactions" Revised edition-pg 1664.

The next stage was the incorporation of PID controller into the CSTR to improve conversion and stabilize the system. The PID chosen for this work is given as;

$$U(t) = K_p c(t) + K_i \int e(t) dt + K_d \frac{d}{dt} e(t)$$

Where U = output value r = desired value
 e = tracking error y = actual output

The variable $e(t)$ represents the tracking error, the difference between the desired value (r) and the actual output (y). This error signal will be used by PID controller. PID will take appropriate action according to the law and pass the signal (u) to the plant to adjust the appropriate manipulated variable. The simulink representation of the PID controller incorporated to the CSTR reactor shown in Figure 3.3. Other subsystems such as concentrations dynamic and temperatures dynamic are shown in appendix B Figures 4, 5, 6, and 7.

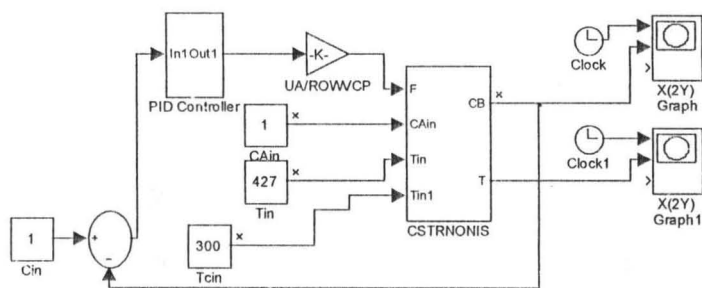


Figure 3.3: Simulink Block Diagram for the CSTR Incorporated with PID Controller

3.5 Application of Neural Analysis in Control Systems

This chapter presents brief descriptions of model reference control (MRC) architectures, which are used to simulate our system.

There are typically two steps involved when using neural networks for control:

- a. System Identification
- b. Control Design

In the system identification stage, a neural network model of the plant that we want to control is developed. In the control design stage, we use the neural network plant model

to design (or train) the controller (MATLAB 7, Control System Toolbox, User's Guide). Depending on the architecture, a number of control design methodologies can be used. In this research work model reference control (MRC) has been used. Here, the controller is a neural network that is trained to control a plant so that it follows a reference model. The neural network plant model is used to assist in the controller training.

3.6 Model Reference Control

An illustration of model reference control is presented in Figure 3.3. In the figure the network has two inputs, one of the inputs is difference between plant output and model reference output and second input is difference between model output and reference signal. Both plant model and reference model are used to train the network. The resulting ANN model will serve as controller for the system. Figure 3.3 explains the model reference control system. ANN controller uses these to adjust its weights until the output of the plant looks similar to model reference output trajectory

ANN controller will have two input, error signal from reference model output, and plant output, the second error signal comes from difference between reference signal and plant output. The procedures involved in training the network are generation and validation of training data sets, pre-processing of data set and training and validation of network. To have good representation of the model, two data sets were generated from the system to train the network, one data set for validation and another one testing. Uniform random input signals, which span the upper and lower limit of operating range were used to excite the system. This was done to enable network learn the nonlinear nature of the system. Before incorporating the network into the control scheme the networks were trained offline using the Gauss-Newton based Levenberg Marquardt algorithm (Levenberg 1944 and Marquardt 1963). The essence is to let the network learn the functional nonlinearities to a certain degree of accuracy before implementing the controller, and thus can give faster online adaptation as needed. In this study, data sets for the training were obtained by carrying out simulation on the open loop of the system, which in turn were used to train the network.

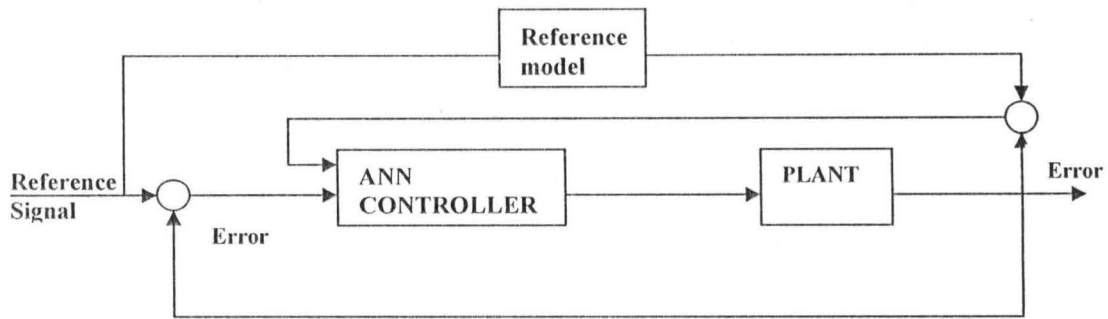


Figure 3.4 Model Reference Control

The network considered was multilayer perceptron with a single hidden layer. The activation function used in study is nonlinear sigmoid function in hidden layer and the linear function in the output layer. Neural network with large enough number of hidden neurons, which have continuous and differentiable transfer functions can approximate any continuous function over a closed interval (Cybenko, 1989). The numbers of nodes are initially fixed at small numbers, the number was increased in order to have a proper trained network. Satisfactory networks models were obtained when the sum of squared errors of the validation data set was satisfactorily small.

The first step in the design is plant identification. Here data for the identification were generated by simulating CSTR model. The data generated was used to train the ANN. The type of training is supervised learning described in section 2.8.3. The network has 20 neurons in the hidden layers. The activation functions in the hidden layer are tan-sigmoid and the output layer in a linear function.

3.5.2 Using the model reference control

The model reference control, Figure 3.4 was created with the Neural Network Toolbox using model reference controller, uniform random input, and graph blocks to demonstrate the model reference control action. The objective of the simulation is to show the effective performance of neural network controller in the presence of step changes to a process reactor.

Tables 3.2 and 3.3 show the parameters used for the model reference training windows and plant identification windows at optimal condition.

Table 3.2 Parameter for Model Reference Training Windows

Size of hidden layers	20
No. Delayed Reference inputs	2
No. Delayed Controller Outputs	1
No. Delayed Plant Outputs	2
Max. Reference. Value	1
Min. Reference. Value	0
Controller Training Samples	8000
Max. Interval Value	20
Min. Interval Value	5
Reactor with PID Cont roller	Plantref1(Given Name)
Controller Training Epochs	5
Segments	5
Uniform Random Input Set	0-1

Table 3.3 Parameter for Plant Identification Windows

Size of hidden layers	15
No. Delayed Plant inputs	2
No. Delayed Plant Outputs	2
Sampling interval (s)	0.2
Max. Plant Input	25.05
Min. Plant Input	2.36
Max. Plant Output	1.5
Min. Plant Output	0
Training Samples	10,000
Max. Interval Value (s)	20
Min. Interval Value (s)	5
Plant Name	Freshplant
Training Epochs	300
Training Function	Trainlm

3.4.3 Steps to run the simulink model.

1. Start MATLAB.
2. Run the demo model by clicking on the file, then opening in the MATLAB command windows. This creates the work file window.
3. Double click on the saved model, named mrc1. This command opens the saved model, mrc1 where further operations were carried out. (Appendix B, Figure 8).

4. Double-click the Model Reference Control block and gradually enter the required parameters as shown in the figure 3.3.
5. Browse and fit in the CSTR model, and the training epochs are selected in this window just before generating data. (Figure 3.5).

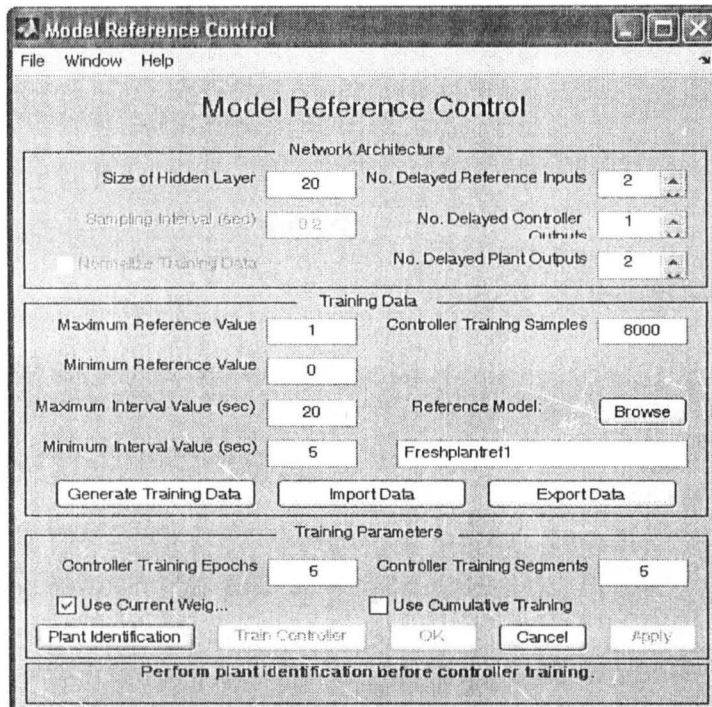


Figure 3.5 Model Reference Control Data Input Windows

6. Click Plant Identification, which opens the Plant Identification windows. You can then train the plant model. The neural network plant model must be developed before the controller is used. The controller is a neural network that is trained to control a plant so that it follows a reference model. The neural network plant model is used to assist in the controller training. The plant model neural network has one hidden layer, as shown below. The size of the layer, the sampling interval, the number of delayed inputs and delayed outputs, the maximum and minimum plant inputs and outputs, the maximum and

minimum interval values, the continuous stirred tank reactor model (named Fresh plant), the training epochs and the training function are all set for data generation and training.

7. Select any of the training function to train the neural network plant model. Trainlm is used for this training. (See Appendix, Figure 3.4 Plant Identification Windows)

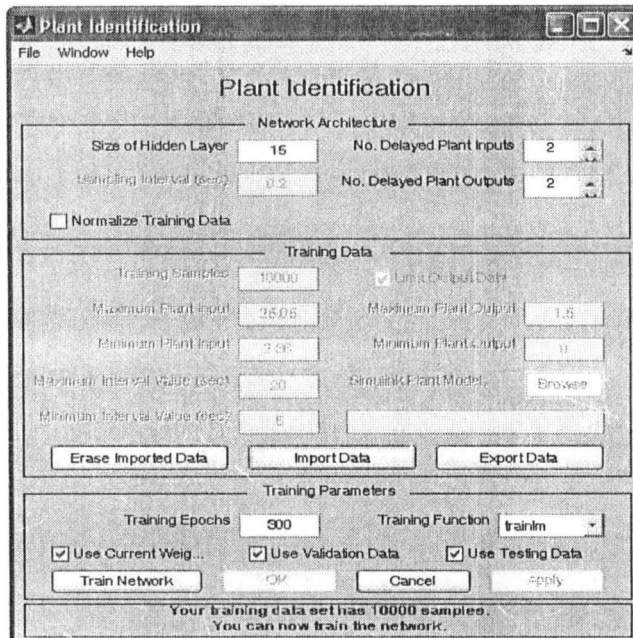


Figure 3.6 Plant Identification Windows

8. Click on generate training data button. The program generates training data by applying a series of uniform random inputs to the simulink plant model. The potential training data is then displayed which is as shown in appendix B Figure 9.

9. Select Accept Data, and then select Train Network from the Plant Identification windows. Plant model training begins. The training proceeds according to the selected training algorithm (trainlm in this case). The response of the resulting model is displayed, as shown in appendix B Figures 10. Training, validation and testing data windows are also displayed in appendix D Figures 3 and 4. The network can then be trained with the same data set by selecting Train Network again. The data set can be erased and a new one

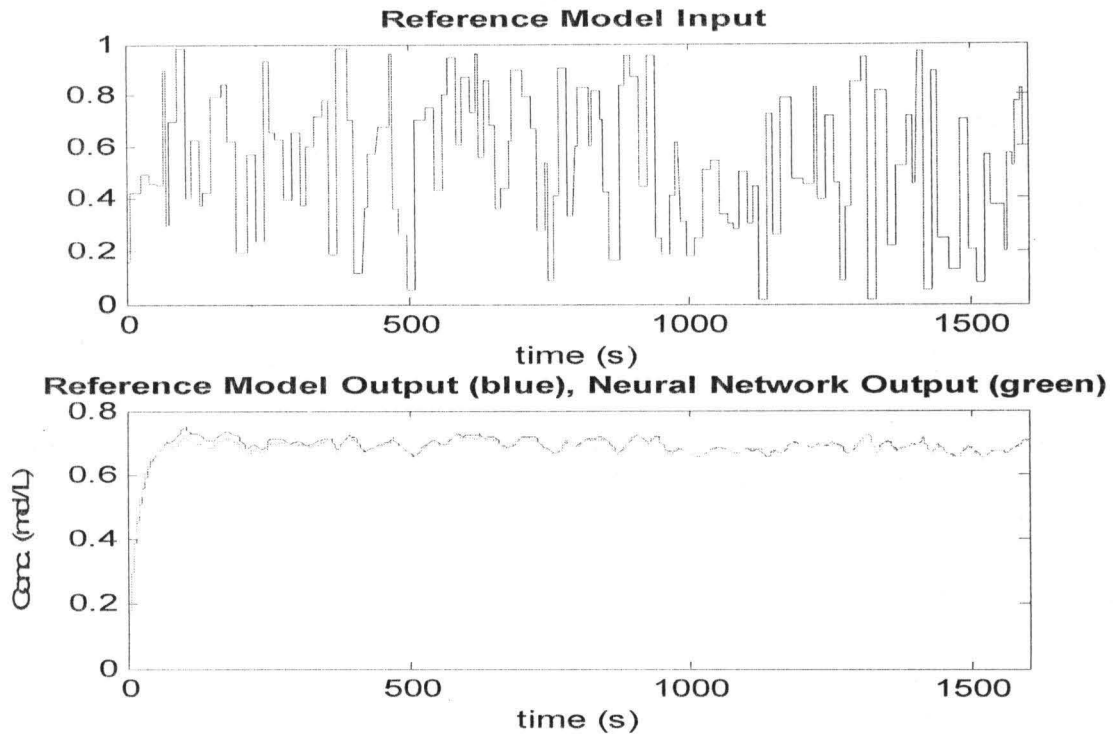


Figure 3.5 Plant Responses for NN Model Reference Control

12. Go back to the Model Reference Control windows, if the performance of the controller is not accurate, then you can select Train Controller again, which continues the controller training with the same data set. If you would like to use a new data set to continue training, select Generate Data or Import Data before you select Train Controller. It might also be necessary to retrain the plant model. If the plant model is not accurate, it can affect the controller training. For this demonstration, the controller should be accurate enough, and then select OK.

13. Return to the simulink model and start the simulation by selecting the start command from the simulation menu. As the simulation runs, the plant output and the reference signal are displayed as show in Figure3.5.

CHAPTER FOUR

4.0 Results and Discussion

4.1 Results

The simulation results of the open loop Continuous Stirred Tank Reactor (CSTR), closed loop CSTR and Neural Networks CSTR are presented in the following figures.

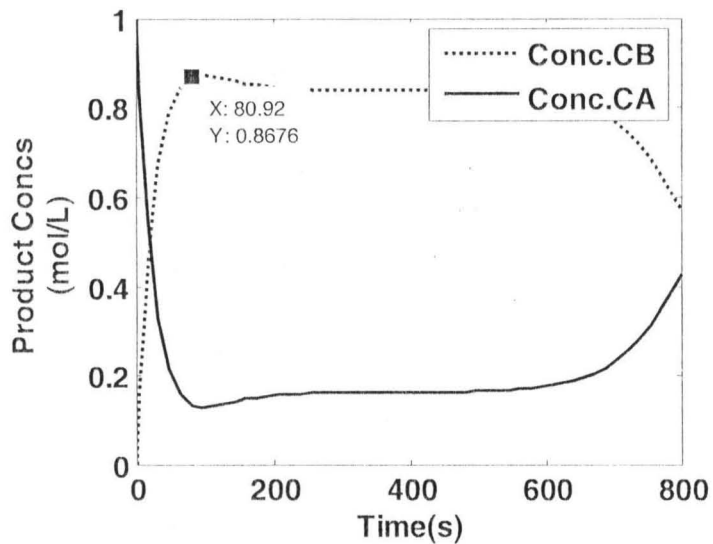


Figure 4.1: Concentration profile for Open Loop simulation of CSTR for reversible reaction

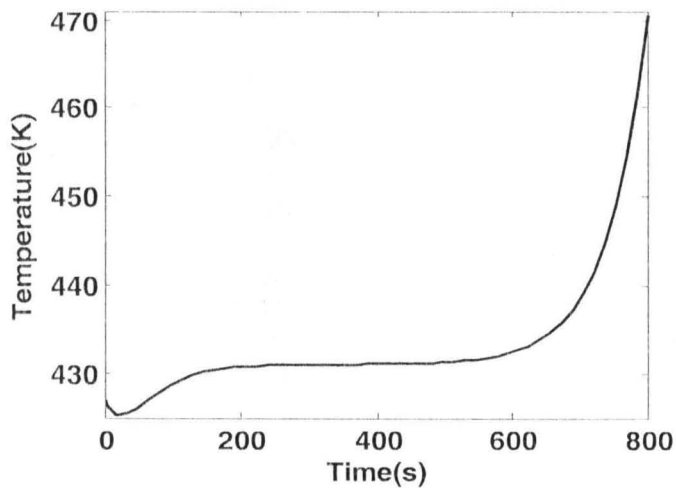


Figure 4.2: Temperature profile for open loop simulation of CSTR for reversible reaction

Table 4.1 Controller setting for PID Controller and their Performance

PID Controller	K_p	K_i	K_d	Rise time	Overshoot	Settling Time	Final Conc.
Setting 1	2.5	0.5	7	33.4	0.08	135	1.00
Setting 2	3.15	0.47	8	30.2	0.41	130	1.00

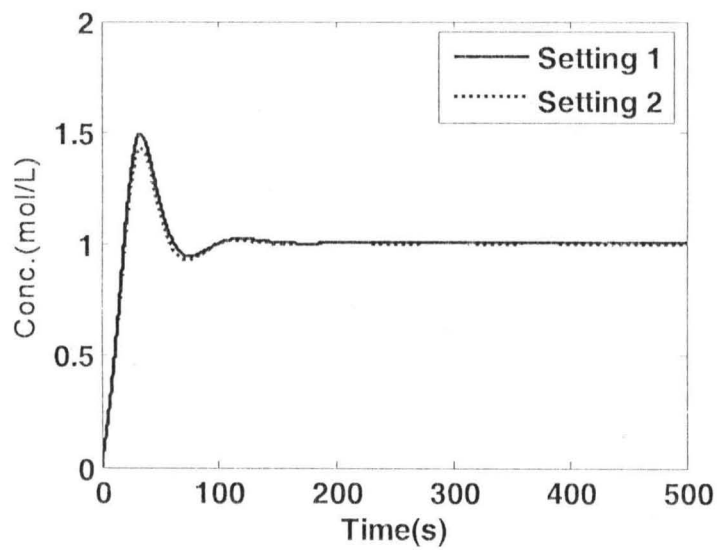


Figure 4.3: Concentration profile for Closed Loop simulation of CSTR for reversible reaction

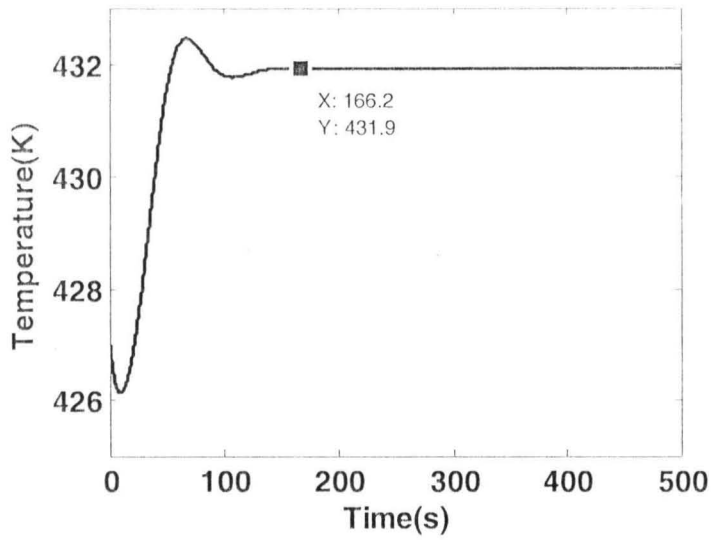


Figure 4.4: Temperature profile for Closed Loop simulation of CSTR for reversible reaction

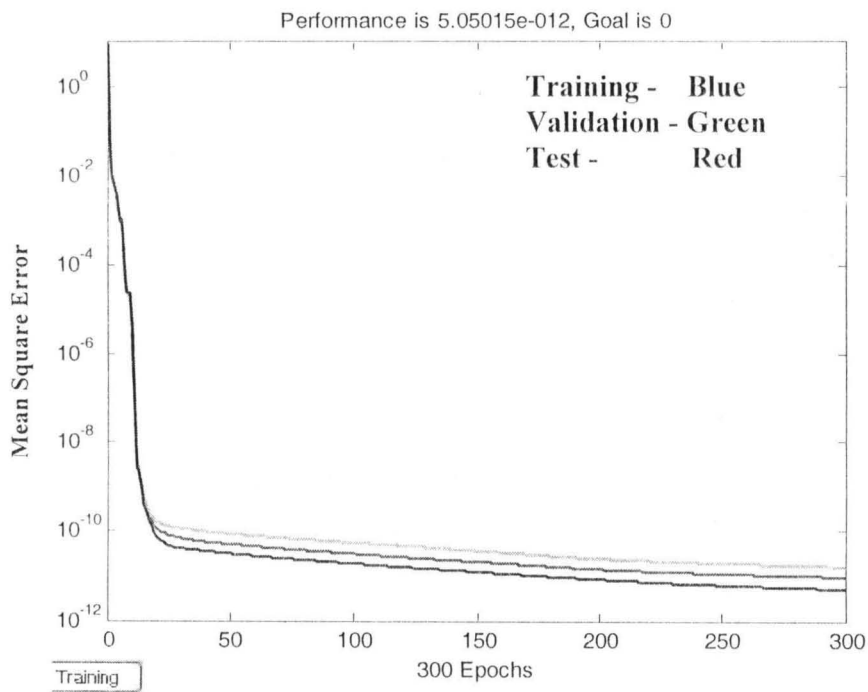


Figure 4.5: Neural Network Plant Responses after Training. (Using Data C on Table 4.2)

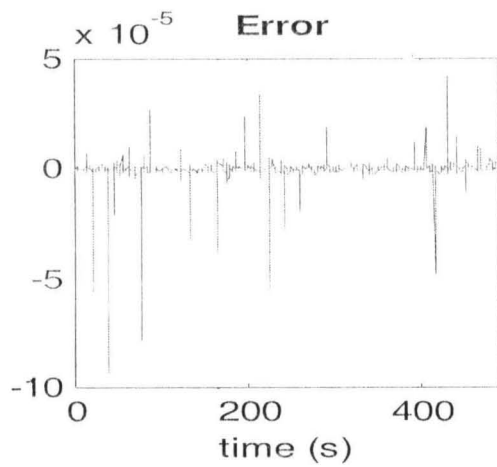


Figure 4.6: Validation Data Showing the Difference between Plant and Neural Network Outputs

Table 4.2 Parameters for Training the Plant and Neural Network Controller

Controller Training	MRC Hidden Layers	Plant Hidden Layers	Training Samples for Controller/Plant	Plant Training Epochs	Controller Training Epochs/Segments	MSE
Data A	10	8	3000/4000	300	5 of 5	1.891e-03
Data B	12	12	5000/7000	300	5 of 5	6.137e-05
Data C	20	15	8000/10000	300	5 of 5	6.298e-06

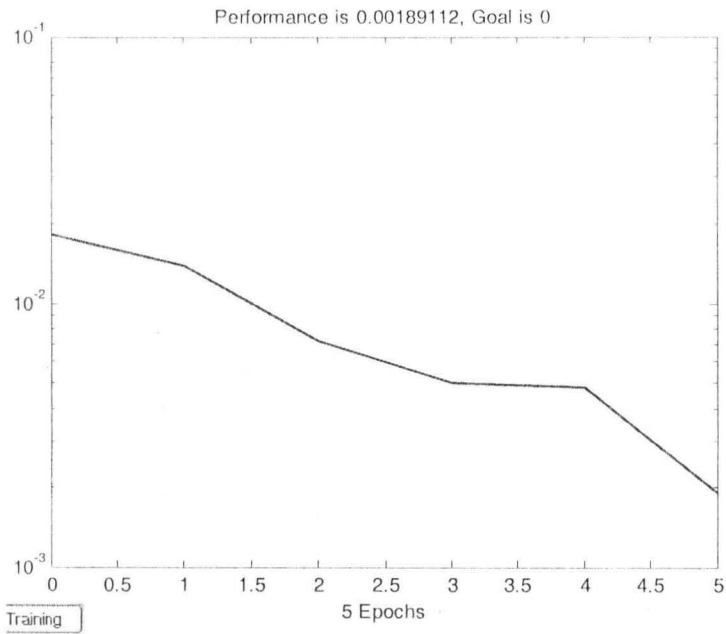


Figure 4.7(a): Training Response Profile for the NN Controller Performance
(Table 4.2, Data A)

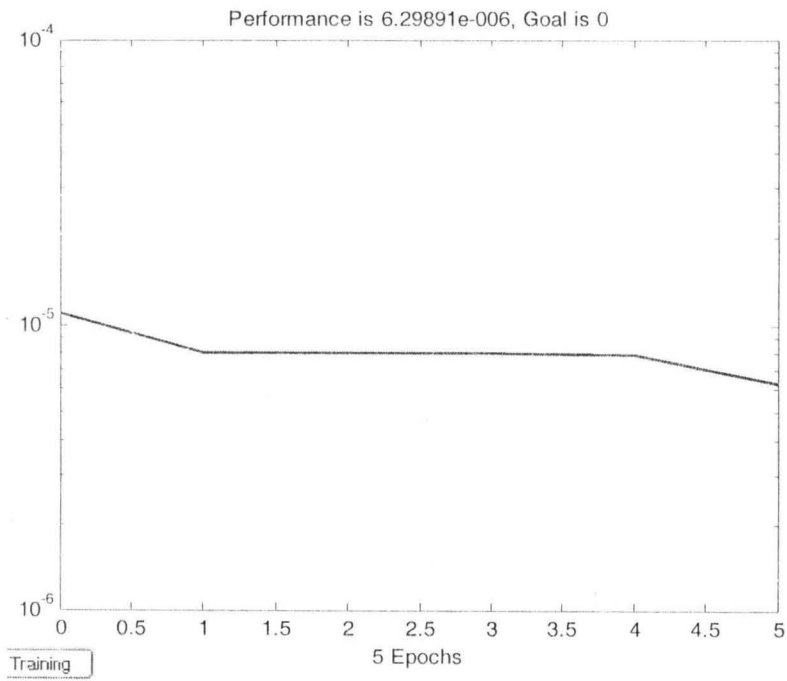


Figure 4.7(b): Training Response Profile for the NN Controller Performance (Table 4.2,
Data C)

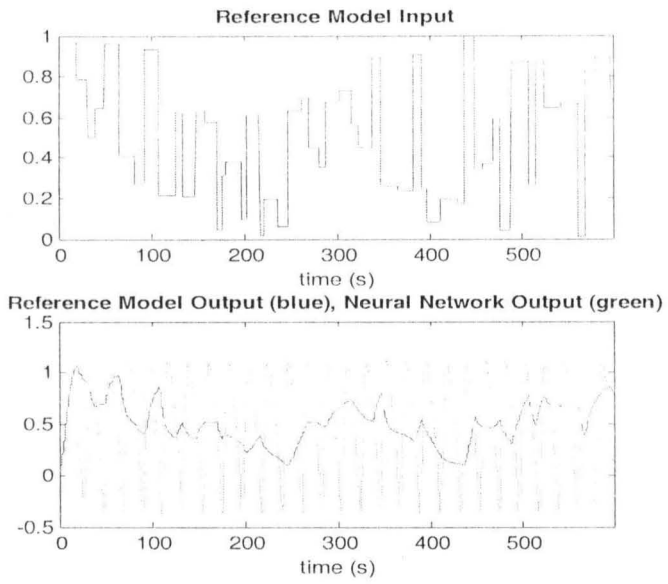


Figure 4.8(a): Reference Model and Neural Network Output Signals (Table 4.2, Data A)

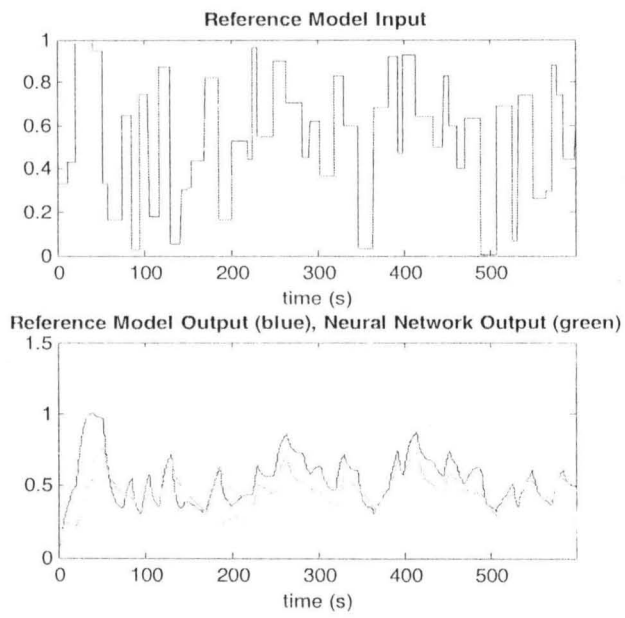


Figure 4.8(b): Reference Model and Neural Network Output Signals (Table 4.2, Data B)

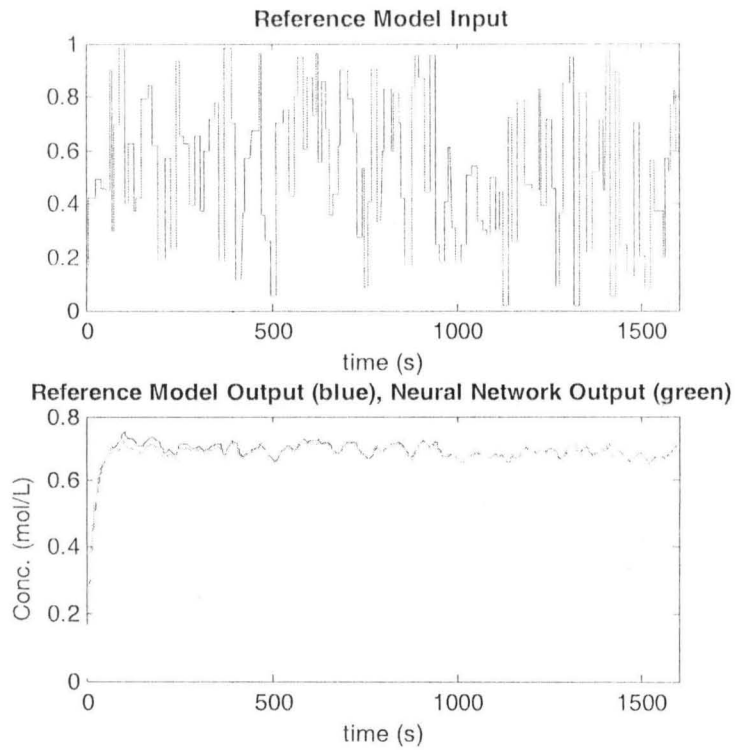


Figure 4.8(c): Reference Model and Neural Network Output Signals (Table 4.2, Data C)

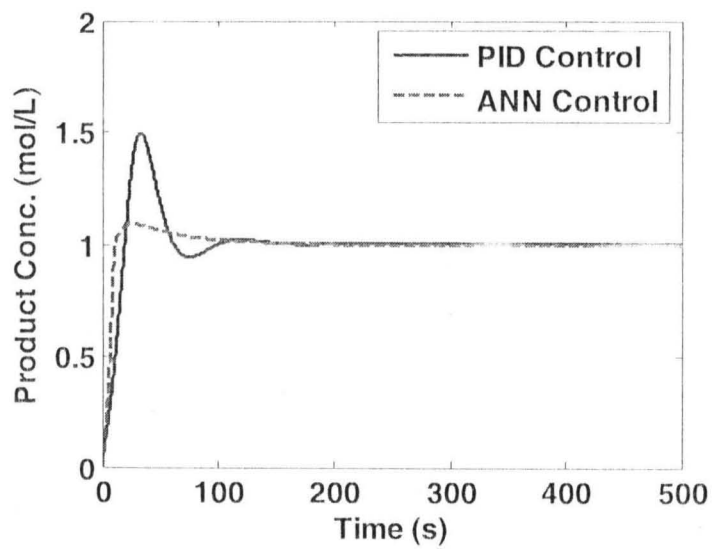


Figure 4.9: Closed loop response for the system under PID and ANN control for a Step down Input of 5%

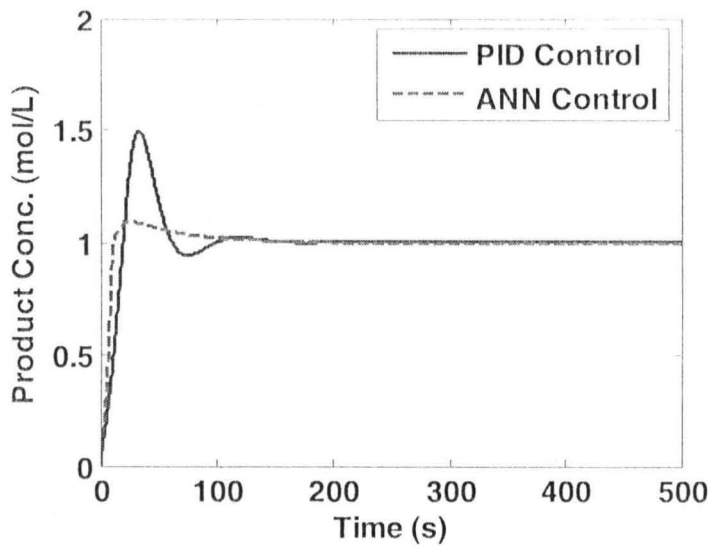


Figure 4.10: Closed loop response for the system under PID and ANN control for a Step down Input of 10%.

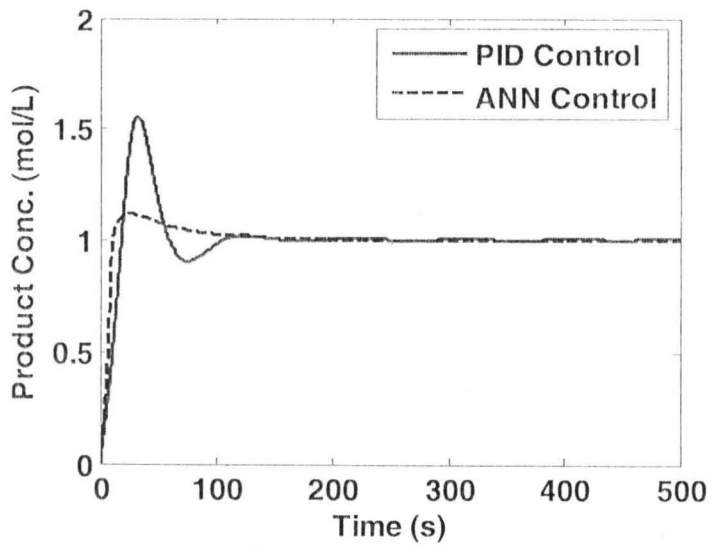


Figure 4.11: Closed loop response for the system under PID and ANN control for a Step up Input of 5%.

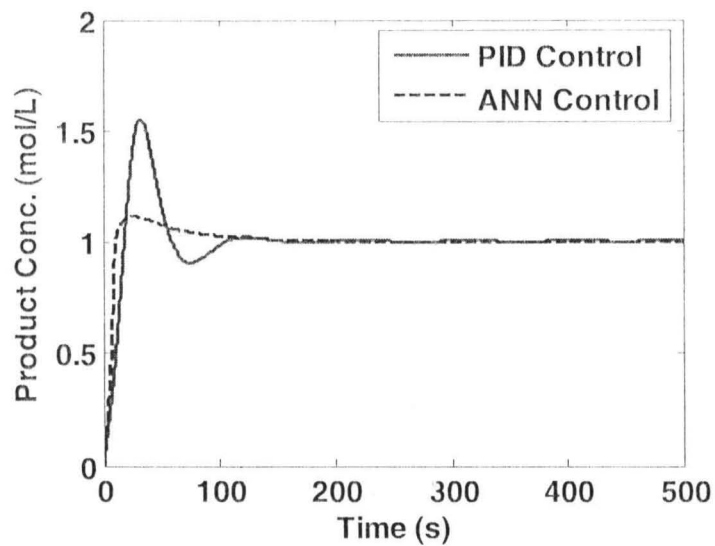


Figure 4.12: Closed loop response for the system under PID and ANN control for a step up input of 10%.

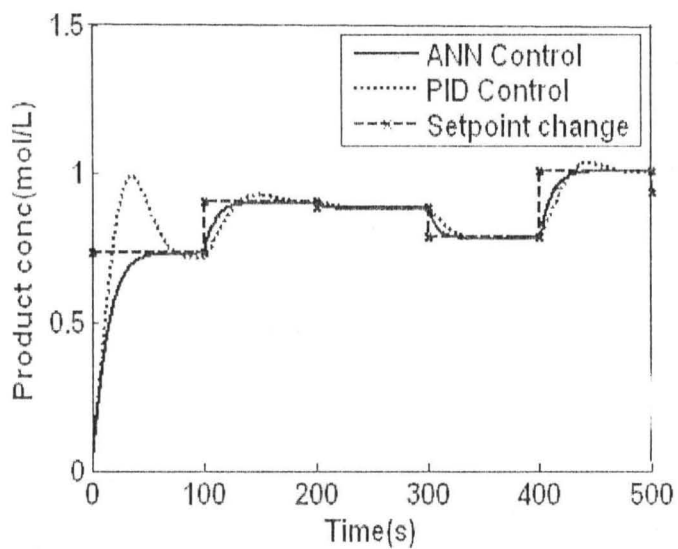


Figure 4.13: Closed loop responses for set point tracking for the system under PID and ANN control

4.2 Discussion of Results

Figure 4.1 shows the open loop concentration profile the simulation of CSTR for reversible reaction. At the beginning, there is an increase in concentration of product B until a maximum conversion of 86.7% at 80.9 seconds, and the curve later bends down to the minimum. The concentration of reactant, A decreases to 0.133mol/L at the same time. This concentration of reactant, A started to rise up after a little time. The reason for these responses is because the reaction we have considered is reversible. Initially, the reaction exhibits forward reaction and later reverses.

Figure 4.2 shows the temperature profile for the open loop simulation for the reactor system. The temperature rises as the reactor process reaction triggers off. This testifies to the instability of the CSTR system when no control method is employed.

In order to bring the reaction to complete conversion and to prevent backward reaction, the flow rate of reactant was used as manipulated variable in the control design and with the incorporation of a PID controller, we have been able to achieve total conversion by suppressing the reversible reaction. This is shown in Figure 4.3. The PID controller setting that gave the best performance was found to be $K_c=3.15$, $K_i = 0.47$ and $K_d = 8$. This is shown in Table 4.1. The response is fast and was able to bring the system output to a target value at 126 seconds. The rise time and settling time are 26 seconds and 96.4 seconds respectively. The response is not without an overshooting of 41.2% which is not only very high but also not welcome in PID control. The objective has been met as long as there is no disturbance or set point change.

Figure 4.4 shows the temperature profile for the CSTR incorporated with a PID controller. Initially, there was an inverse response which can be adduced to the heat gained from the surrounding before reaction occurs. The temperature rises immediately and returns to a steady state of 431.9 K at minimum period of 172.7 seconds.

Despite the fact that the desire output has been obtained by the used of PID controller, the unwelcome high overshoot and settling time called for the used of intelligence control which will eliminate these problems and others that may be encountered by PID controller in the presence of disturbance as well as set point changes for non-linear system.

Figure 4.5 is the performance curve during the network training for the continuous stirred tank reactor. At the beginning of the training, the performance profile shows that the training, validation and testing errors were high. As the number of epochs increases the mean squared error (MSE) decreases. As the curves start to converge, very little learning is taking place. This shows that the training, validation and testing errors decrease as the network learns and hence an indication of accurate parameters selection.

Figure 4.6 is the validation data for NN model reference control. It shows the error differences between the plant output and the neural network output. It was used to ascertain that the error difference between the plant output and NN output is within an acceptable limit before the user could proceed to generate data for the controller. The testing and validation data responses are shown in the appendix D, Figures 3 and 4.

Figure 4.7(a) is the performance curve for the training of the network controller for error minimization. At the start of the training, the error between the network output and the CSTR model output is high. As the number of training iteration or epoch increases, the mean square error (MSE) decreases. The steepness of the curve shows that training samples and the number of hidden neurons need to be increased until possible accurate

fitting is achieved. It is an indication that further minimization of error is possible. The convergence of the performance curve displayed in Figure 4.7(b) shows that little or no training was taking place and an indication that the error minimization is within acceptable limit.

Table 4.2 shows the performance results for the various data sets used during the controller training. The results show that increasing the size of hidden layers and the training samples decreased the mean square error (MSE) between the model and the process outputs.

Figure 4.8c shows that the response of the resulting closed loop system after the whole training exercise was completed. The upper part (reference model input) is the random reference input that was used for the training. The lower part is the response of the reference model and the response of the closed loop plant. The plant response followed the reference model. This is an indication that the correct amount of training samples and size of hidden layers or neurons were used.

Figure 4.8a shows that the data set for the plant identification and model reference control results in oscillatory behaviour, while the plant response does not follow the reference model perfectly in Figure 4.8b. The two responses were due to less number of hidden layers or neurons as well as insufficient number of training samples. This is called under-fitting or under-parametisation.

Figures 4.9 and 4.10 are the simulation response of the NN controller to step down changes of 5% and 10% in the feed concentration. The responses show that both the controller met the set target. The response reveals that the NN controller was able to return the system output to the desired concentration at a minimum rise time of 10 and 10.7 seconds for 5% and 10% respectively with no oscillation and negligible overshoot.

In the case of PID controller, the rise time, settling time and the overshoot in both cases were increased. The increase in overshoot and settling time are not welcome in control system because it renders PID controller unsuitable for controlling many non-linear reversible CSTR reaction.

Figures 4.11 and 4.12 show the responses for both ANN and PID controller to a step up changes of 5% and 10% in the feed concentration. While ANN controller is able to return the system output to the desired steady state in the presence of set-point change at a minimum period of 10.7 seconds with neither oscillation nor overshoot, the PID controller failed. This may be difficult to tackle in the case of PID controller because its performance is only good in the operating region and most especially for linear system. Outside the region the controller performance will deteriorate. The above shows that the ANN's are capable of identifying complex non linear systems both within and outside the operating regions.

Figure 4.13 shows the closed loop response of the system in the presence of external disturbance. The system was disturbed by introducing 10% change in reactant temperature. The ANN controller was fast to arrest the disturbance but there is occurrence of overshoot. It counteracts disturbance and return the system to original condition on time. The PID control gives serious oscillation and it did not settled throughout the simulation period.

CHAPTER FIVE

5.0 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

From the results obtained in our simulation we can see that the ANN controller using model reference was able to track set point change and reject the uncertainties resulting from external disturbances. The responses were somehow sluggish in the faces of external disturbances but give no oscillatory behaviors. For PID controller, the performance deteriorated for set point changes and under the influence of external disturbances. This reason for poor performance can be adduced because of high nonlinearity of the CSTR.

ANN control approach using references model to tailor system output to a desired responses was developed. The controller has been able to take care of nonlinearly aspect of the system. ANN control scheme has better trajectory tracking ability than PID since the former is based on nonlinearity of the model, while the latter based on particular operating conditions. The control was able to adapt to system changes and operating condition change. There was no need for turning parameter in the control. The control was very adaptive and has self-turning capability for any change in operating conditions and system parameters. During the online implementation, the networks ware continuously adapted online, the controller did not require an integral action to obtain zero offset in the system output. The proposed method is somehow sluggish because the optimal control action is iterative in order to converge to an acceptable accuracy.

The result of the research work also show that the techniques in controlling the system offer a better control alternative to those formerly used.

5.2 Recommendations

The neural network techniques offer promising approach to truly intelligent system, which can provide optimal solution to many non-linear control problems. When dealing with non-linear system identification, we need to be sure that the system inputs and outputs cover the operating range for which the controller will be applied. For this application, we typically collect training data by applying random inputs which consist of a series of pulses of random amplitude and duration. The duration and amplitude must be chosen carefully to produce accurate identification.

REFERENCES

1. Anderson C.W., 1989, "Learning to control an inverted pendulum using neural networks", IEEE Controls Systems Magazine, Vol.9 pp 31-37.
2. Antsaklis P. J., K. M. Passino, and S. J. Wang, (1989), "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues", J. Intell. Robotic Syst., vol. 1, pp. 315-342.
3. Åström, K.J. and Wittenmark, B. (1989), "Adaptive Control", 2nd. Edition, Addison-Wesley.
4. Astrom K.J. and Haggund T., 1995, "PID Controllers Theory, Design and Tuning", Instrument Society of America, New York.
5. Barto, Sutton and Anderson, (1983), "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems", IEEE Tran on Systems, Man and Cybernetics, Vol.13, pp 834-846.
6. B. Bavarian, Guest Editor of Special Section on Neural Networks for Systems and Control, IEEE Contr. Sysr. Mag., vol. 8, no. 2, pp. 3-31, Apr. 1988.
7. Bhatt, N.V. and Mcavoy, T.J. (1992), "Determining Model Structure for Neural Models by Network Stripping", Computers Chem. Eng., Vol. 16, No. 4, Pp. 271-281.
8. Billings, S.A., Jamaluddin, H.B. And Chen, S. (1992), "Properties of Neural Networks with Applications to Modelling Non-Linear Dynamical Systems", Int. J.Control, Vol. 55, No. 1, pp. 193-224.

9. Campa, Fravolini, Napolitano- "A library of Adaptive neural networks for control purposes." The Simulink library can be downloaded from the Mathworks file exchange website in the ANN section.
<http://www.mathworks.com/matlabcentral/fileexchange/>
10. Chen J. T. and Huang C., "Applying Neural Networks to Online Updated PID Controllers for Nonlinear Process Control, Journal of Process Control, Vol. 14, pp 211 – 230, 2004
11. Cybenko G., 1989, "Approximation by superposition of a Sigmoidal Function, Mathematics of Control, Signals and Systems", Vol 2, No. 4, pp 303-314.
12. Daniel Franklin, 2003, http://www.ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English
13. Doherty, S.K., Gomm, J.B. and Williams, D., 1995, "Neural Network Identification and Predictive Control of an In-Line Ph Process", 4th I.Chem.E, Conf. on Advances in Process Control 4, New York, pp 57-64.
14. Doherty, S.K., Gomm, J.B. and Williams, D., 1997, "Experiment Design Considerations for Non-Linear Identification Using Neural Networks", Computers and Chemical Engineering, Vol. 21, No. 3, Pp. 327-346.
15. Economou, G. and Morari, M., 1986, "Internal Model Control -5. Extension to Nonlinear Systems", Ind. Eng. Chem. Process Des. Dev., Vol. 25, pp 403-411.
16. Emuoyibofarhe O.J, 2004, "A Computation Method for the Solutions of Optimal Control Problems Using Neural Networks"

27. Magnus Norgaard, Neural Network Design Toolkit,
[Http://Www.lau.Dtu.Dk/Research/Control/Nnlib/Manual.Pdf](http://www.lau.dtu.dk/research/control/nlib/manual.pdf)
28. Mcavoy, T.J., Hsu E. and Lowenthal S., 1972," Dynamics of PH in Controlled Stirred Tank Reactor, Ind. Eng. Chem. Process Des, Dev. Vol. 11, Pp. 68-70.
29. Nahas, E.P., Henson, M.A. And Seborg, D.E., 1992," Nonlinear Internal Model Control Strategy for Neural Network Models", Computers Chem. Engng., Vol. 16, pp. 1039-1057.
30. Narendra K.S. and Parthasarathy K., 1990, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. Neural Networks, Vol. 1, No. 1, pp 4-27.
31. Nechyba and Xu, 1994, "Neural Network Approach to Control System Identification with Variable Activation Functions", Robotics Institute, Carnegie-Mellon University
32. Mayr O., 1970, "The Origins of Feedback Control", Cambridge, MA:

MIT Press.
33. Paul W. Murrill, 1996, "Fundamental of Process Control Theory "United State of America, pp 50-60, pp 4-18.
34. Pham, D.T. and Liu, X., 1995, "Neural Networks for Identification, Prediction and Control", Springer-Verlag.

35. Psaltis, D, Sideris, A. And Yamamura, A. (1988) "A Multilayered Neural Network Controller", IEEE Control Systems Magazine, Vol. 8, Pp. 17-21.
36. Haykin S., 1999, Neural Networks – A Comprehensive Foundation, IEEE Press, Macmillan, New York.
37. Saerens M., Soquet A., 1991, "Neural Controller based on back-propagation algorithm", IEEE Proceedings –F, Vol. 138, No.1, pp 55-62, 81.
38. Sean Kevin Doherty, 1999, "Control of PH in Chemical Processes Using Artificial Neural Networks", Liverpool John Moore's University, p 94, pp 140-158.
39. Tim Callinan, 2003, "Artificial Neural Network Identification and Control of the Inverted Pendulum", pp 18-22, p58.
40. Vandoren, V.J., 1997," Multivariable Controllers Enter the Mainstream", Control Engineering, Vol. 44, No. 4, Pp. 107-112.
41. Widrow, B. and Smith,F., 1964, "Pattern-Recognising Control Systems," Computer and Information Sciences (COINS) Symp., Proc., Washington DC, Spartan, pp 288-317,
42. Xue.Z.Wang, 1999, "Data Mining and Knowledge Discovery for Process Monitoring and Control", Springer-Verlag, London Ltd, pp 85-99.
43. Zhang Y., Peng P.Y. And Jiang Z.P., 2000," Stable Neural Controller Design for Unknown Nonlinear Systems Using Backstepping", IEEE Trans., Neural Networks, Vol. 11, No. 6, pp 1347–1359.
44. <http://www.mathworks.com/matlabcentral/fileexchange/>
45. <http://www.matworks.com/acess/helpdesk/toolbox/nnet/control1.html>
46. <http://www//en.wikipedia.org/wiki/weighted-sum>

47. http://www.en.wikipedia.org/wiki/neural_network
48. www.cse.stanford.edu/class/sophomore_college/projects.oo.neural_networks
49. <http://www.answers.com/topic/artificialintelligence>
50. <http://www.en.wikipedia.org/wiki/reactors>
51. http://en.wikipedia.org/wiki/PID_controller#Loop_tuning

Each icon in the main Simulink window can be double clicked to bring up the corresponding block library. Blocks in each library can then be dragged into a model window to build a model.

Source Blocks are used to generate signals. Double-click on the **Sources** icon in the main Simulink window to bring up the Sources window.

Constant: The Constant Source Block simply generates a constant signal. The constant output value is displayed in the middle of the block, with a default value of 1. This can be change by double-clicking on the block in your model window to bring up the required box.

Step: The Step Source Block generates a step function. The initial and final values can be specified, as well as the step time.

Clock: The Clock Source Block generates a signal equal to the current time in the simulation. The clock's output reflects the times at which the other signals outputs occur.

Digital Clock: The Digital Clock Source Block generates a strictly periodic time signal at a specified sampling interval.

From Workspace: The From Workspace Source Block is identical to the From File Source Block except the values are taken from a variable (or expression) in the MATLAB Workspace.

Random Number: The Random Number Source Block generates a sequence of random numbers generated with the specified random number seed.

Sink Blocks are used to display or output signals.

Scope: The Scope Sink Block was described earlier. It is used to display a signal as a function of time.

XY Graph: The XY Graph Sink Block plots one signal against another. It is useful for phase-plane plots, etc.

Display: The Display Sink Block is a digital readout of a signal at the current simulation time.

To File: The To File Sink Block saves a signal to a .mat file in the same way that the From File Source Block reads from a file.

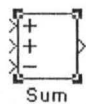
To Workspace: The To Workspace Sink Block stores a signal in a specified workspace variable. Unlike the To File Sink Block, the file is not saved in the variable, and must be stored separately.

Linear Blocks are elements of linear continuous-time dynamic systems. These are shown in the Simulink window below.

Gain: This is a scalar or vector gain. The specified gain multiplies the input. The output is either a scalar or vector signal following normal vector-scalar multiplication rules.

Sum: The Sum Block adds (or subtracts) two (or more) signals and outputs their sum (or difference). The two inputs must either all be scalars, or all be vectors of the same dimension. The output is the same dimension as the inputs.

Example



Integrator: The output of the Integrator is the integral of the input. An initial condition can be specified, as well as saturation limits. This block is very useful for modeling systems.

Transfer Function

Numerator and denominator polynomials can be specified to create a standard SISO LTI system transfer function.

Derivative: The output is equal to the derivative of the input.

Dot Product: The output is equal to the dot product of two vector signals.

Product: The output is equal to the product of the inputs. The number of inputs can be specified.

Mux, Demux

The Mux (Multiplexer) block is used to combine two or more scalar signals into a single vector signal. Similarly, a Demux (Demultiplexer) block breaks a vector signal into scalar signal components. The number of vector components must be specified in each case.

Appendix B

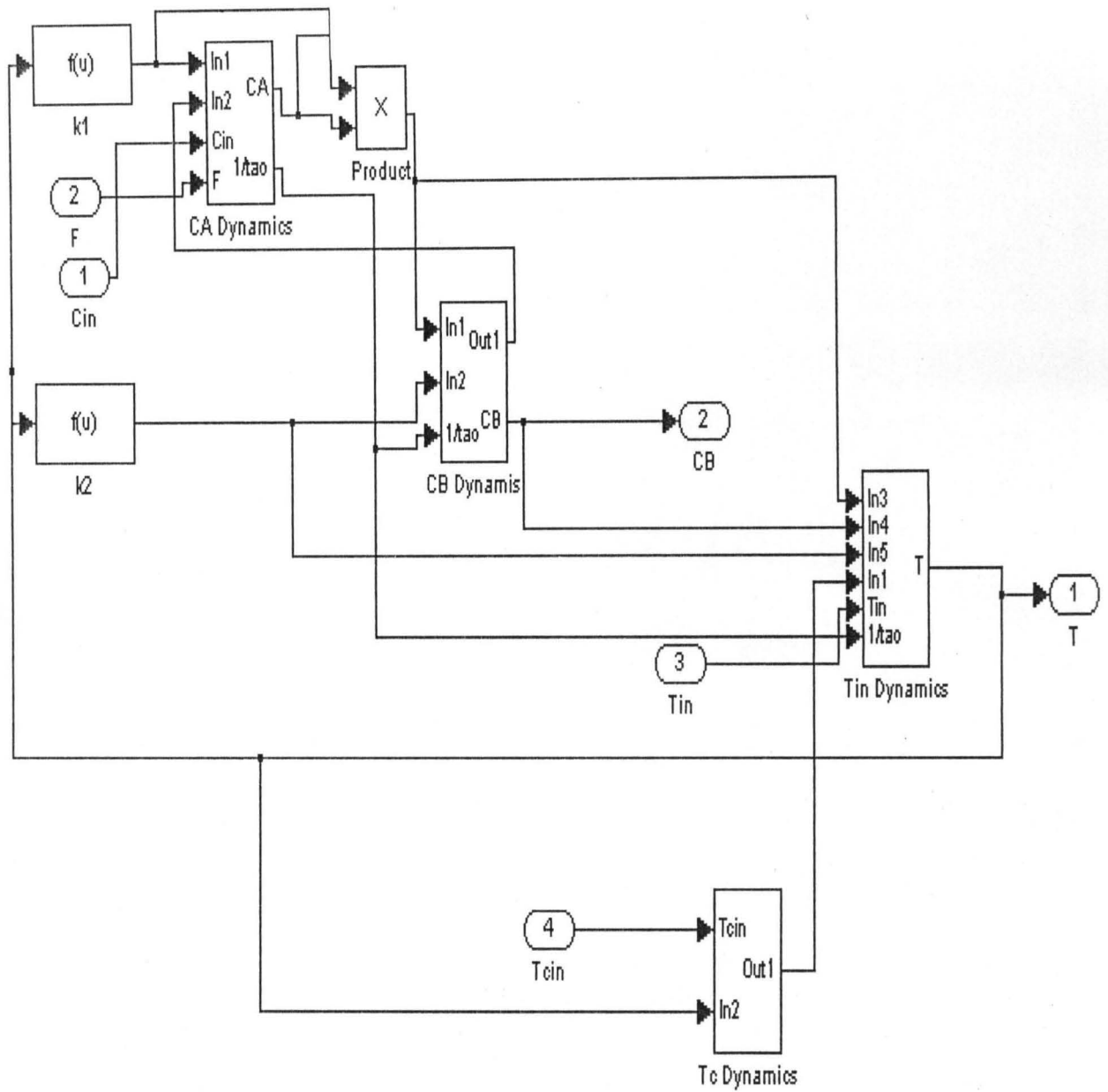


Figure 2: Simulink Block Diagram for the Model Equations

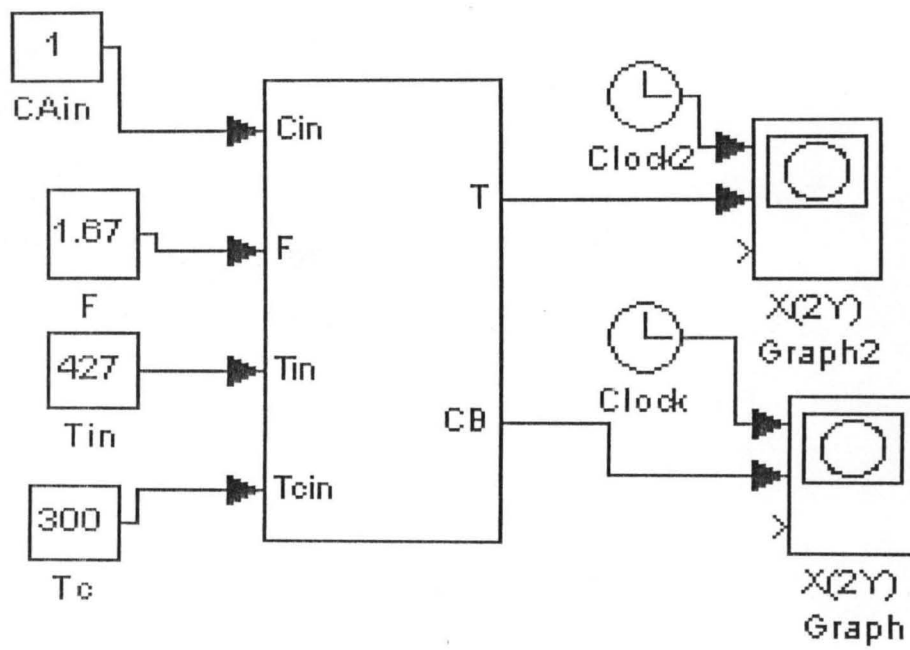


Figure 3: Simulink Subsystem Block Diagram for the CSTR

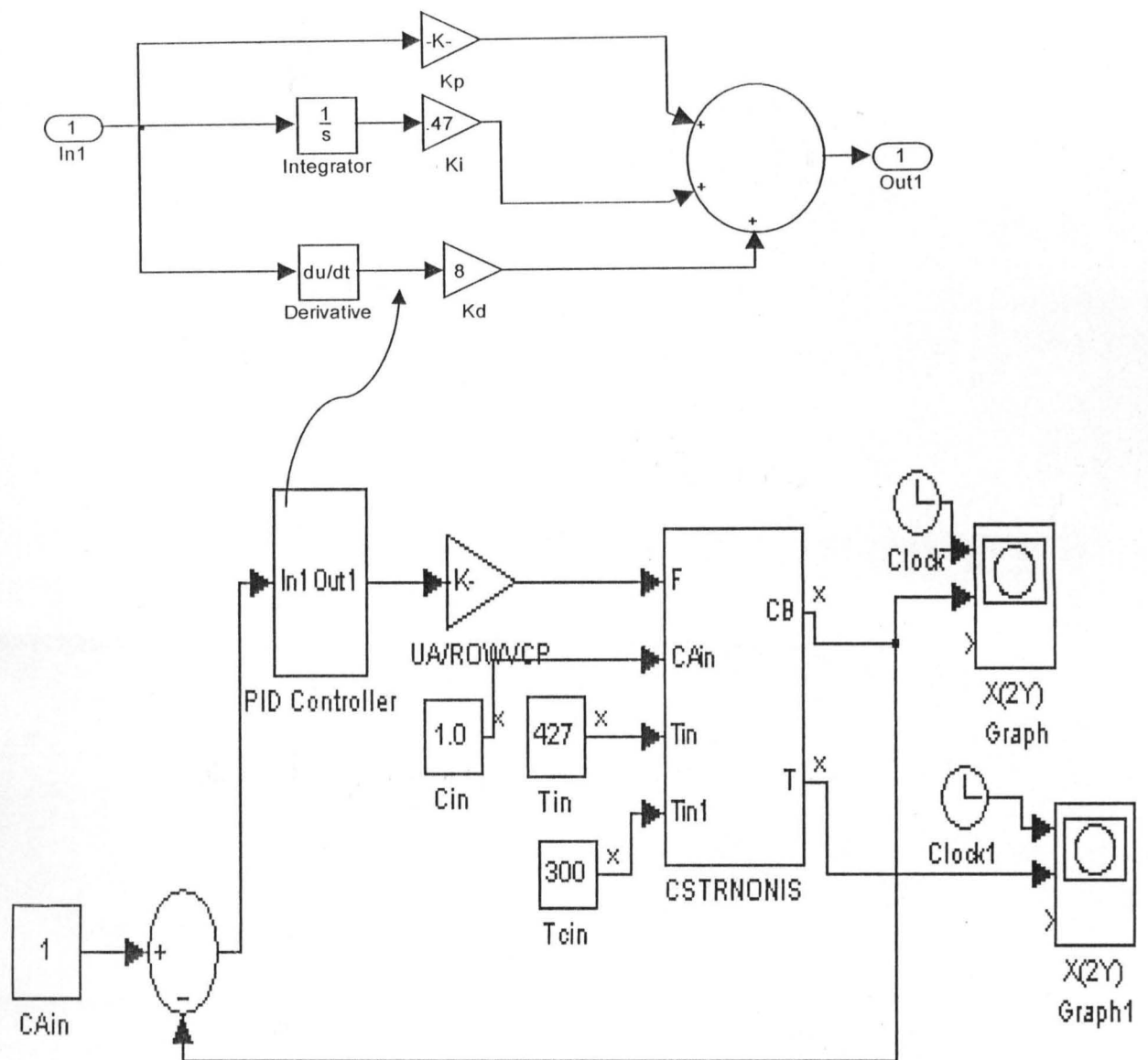


Figure 4: Simulink Block Diagram for the CSTR Incorporated with PID Controller.

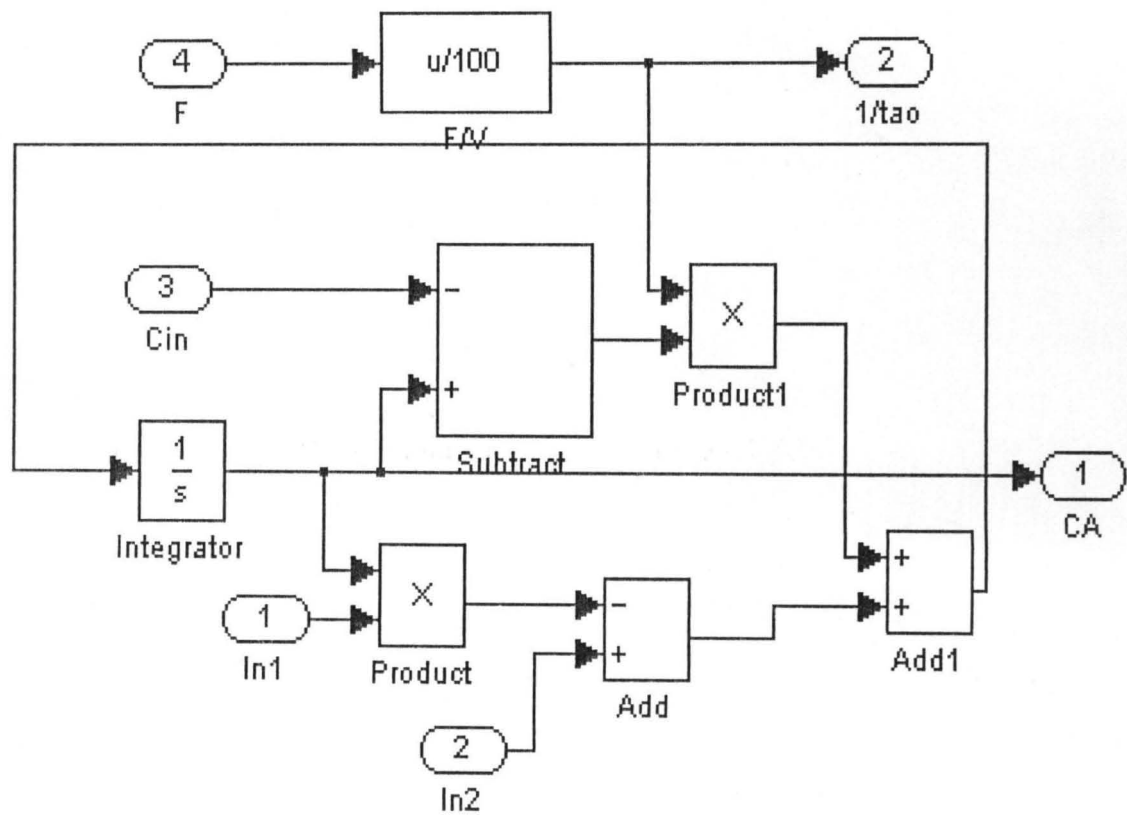


Figure 5: Simulink Block Diagram for Concentration Dynamic, C_A

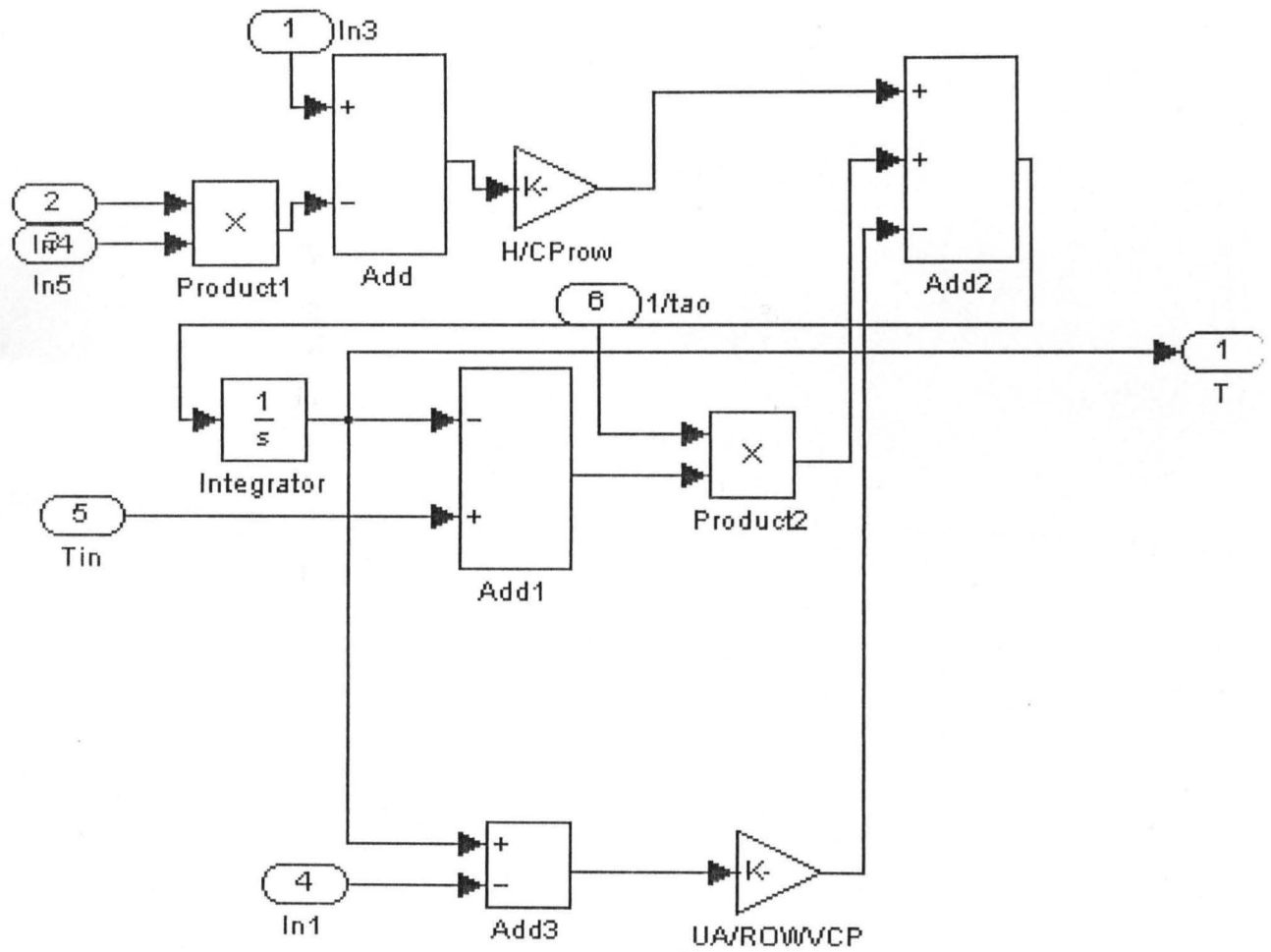


Figure 7: Simulink Block Diagram for Temperature Dynamics, T_{in} .

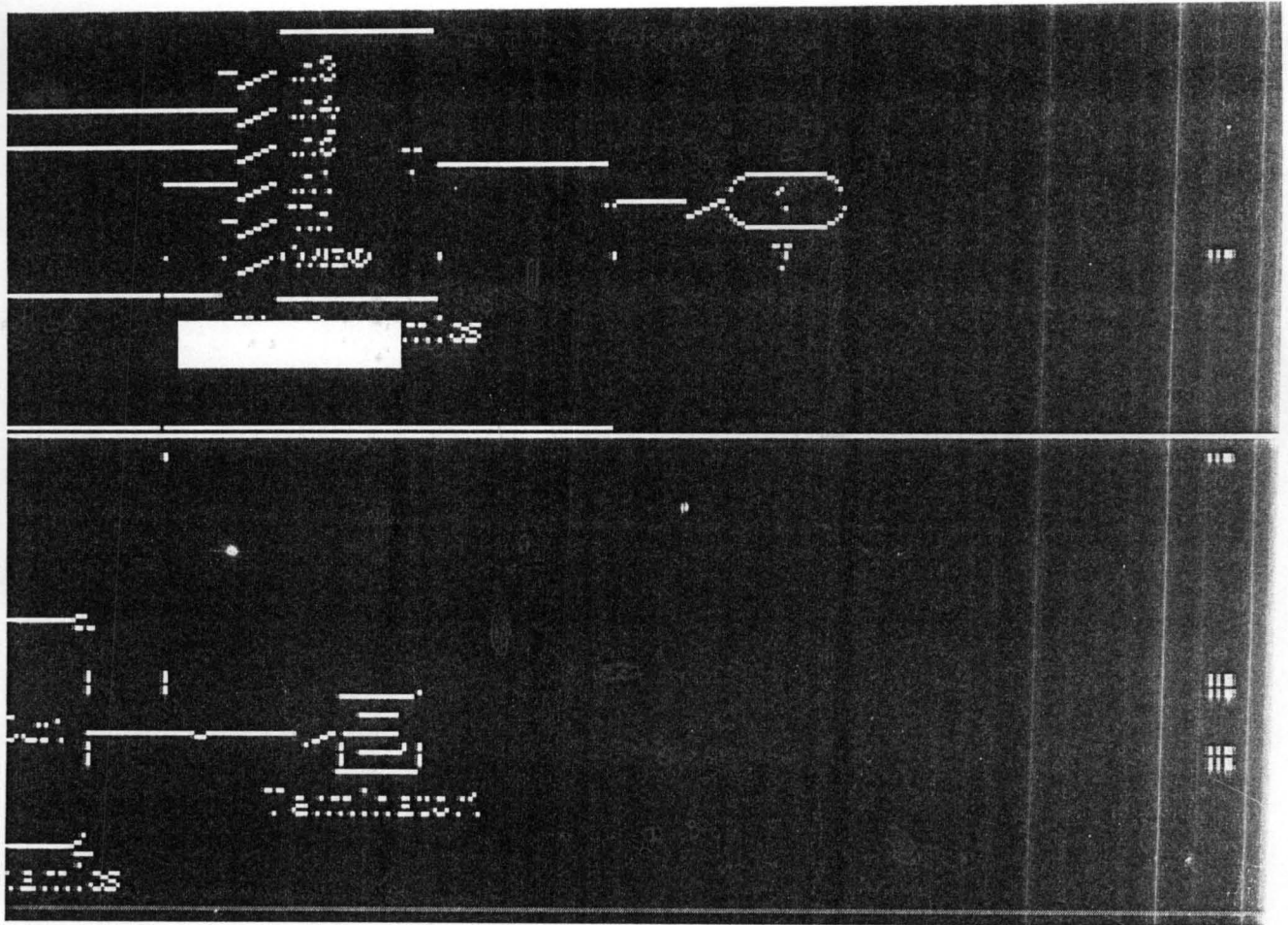


Figure 8: Simulink Block Diagram for Cooling Jacket Temperature Dynamics, T_c .

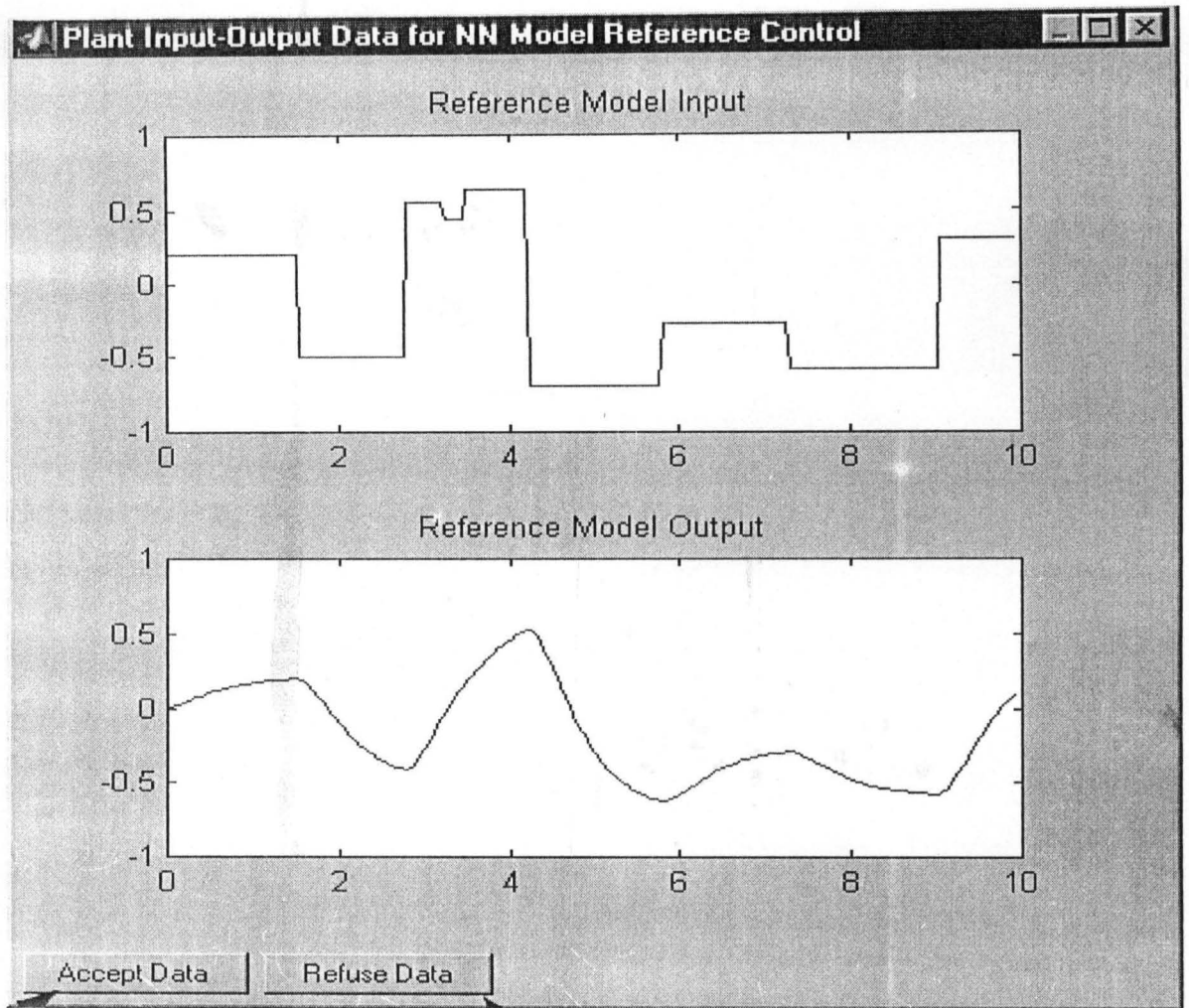


Figure 9: Plant Input-Output Data

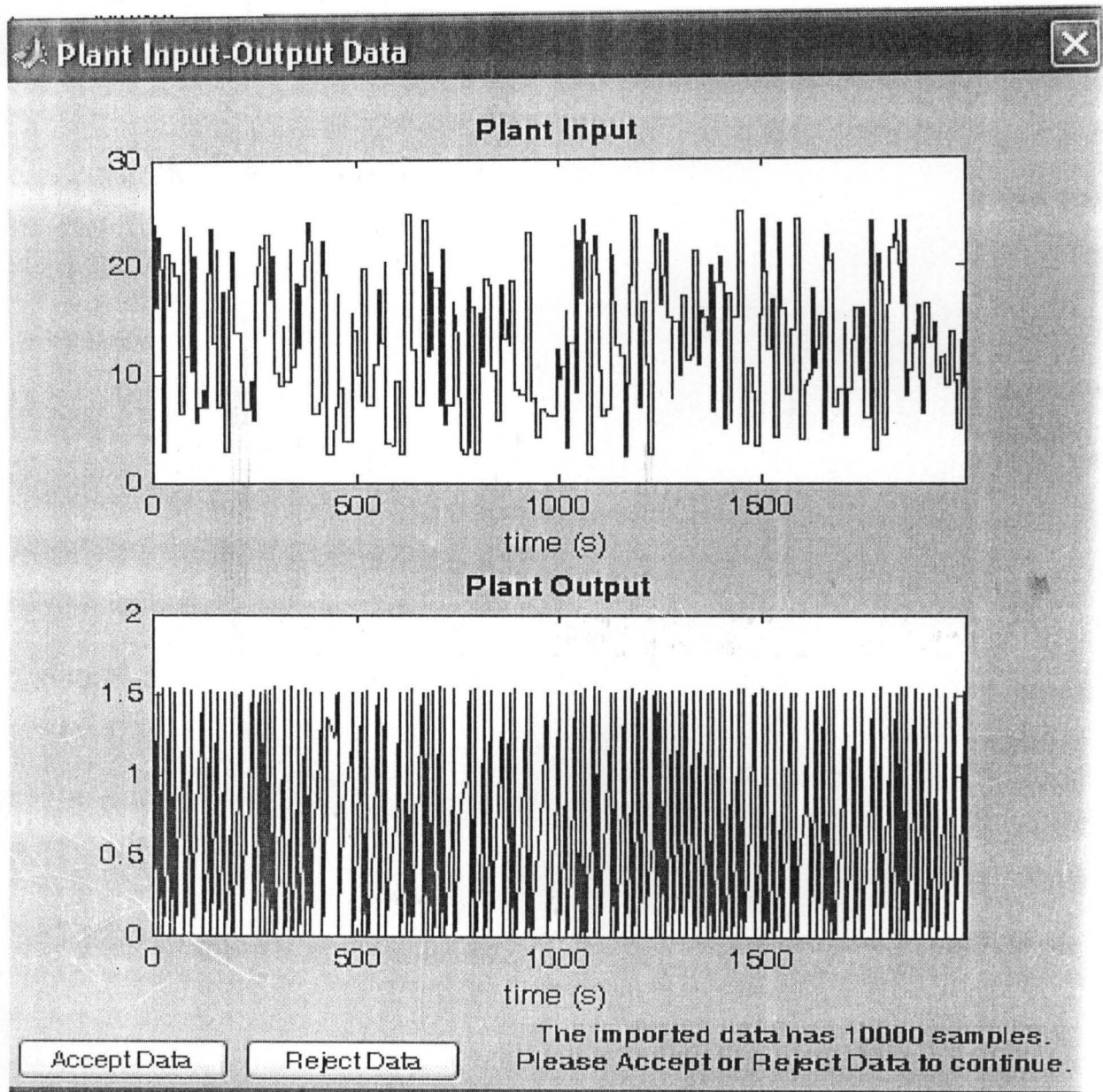


Figure 3.10 Plant input output data for NN Model Reference Control

Figure 10: Plant input output data for NN Model Reference Control

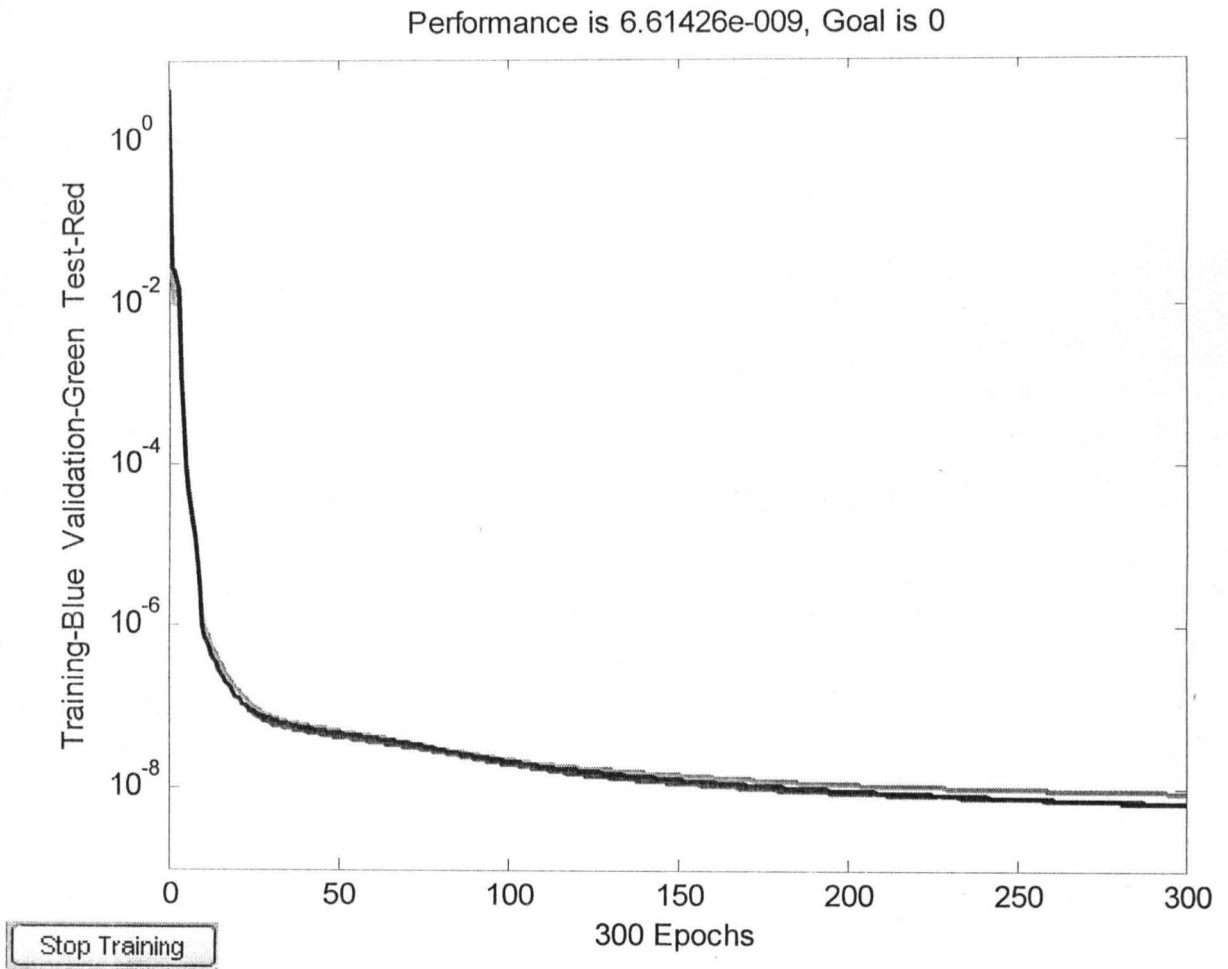


Figure 12: Neural Network Plant Responses after Training (Data B)

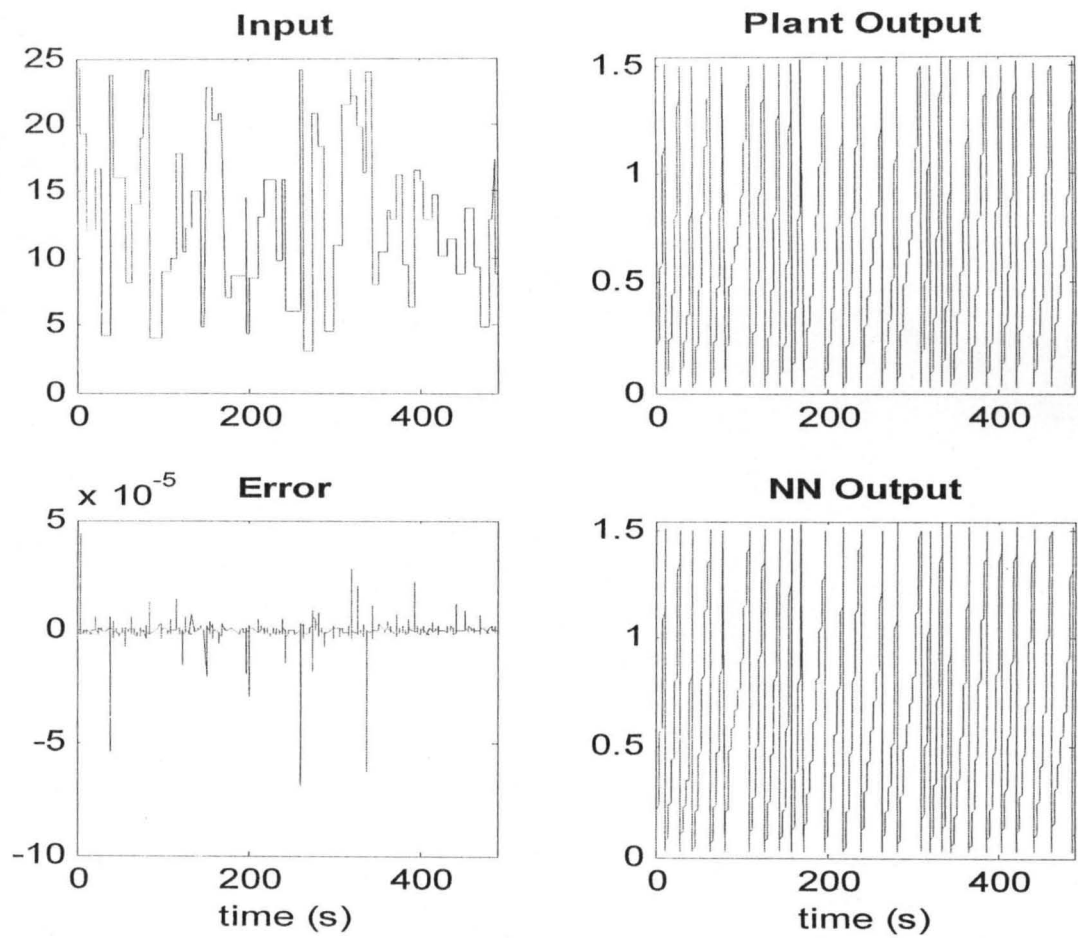


Figure 13: Testing data for NN Model Reference Control

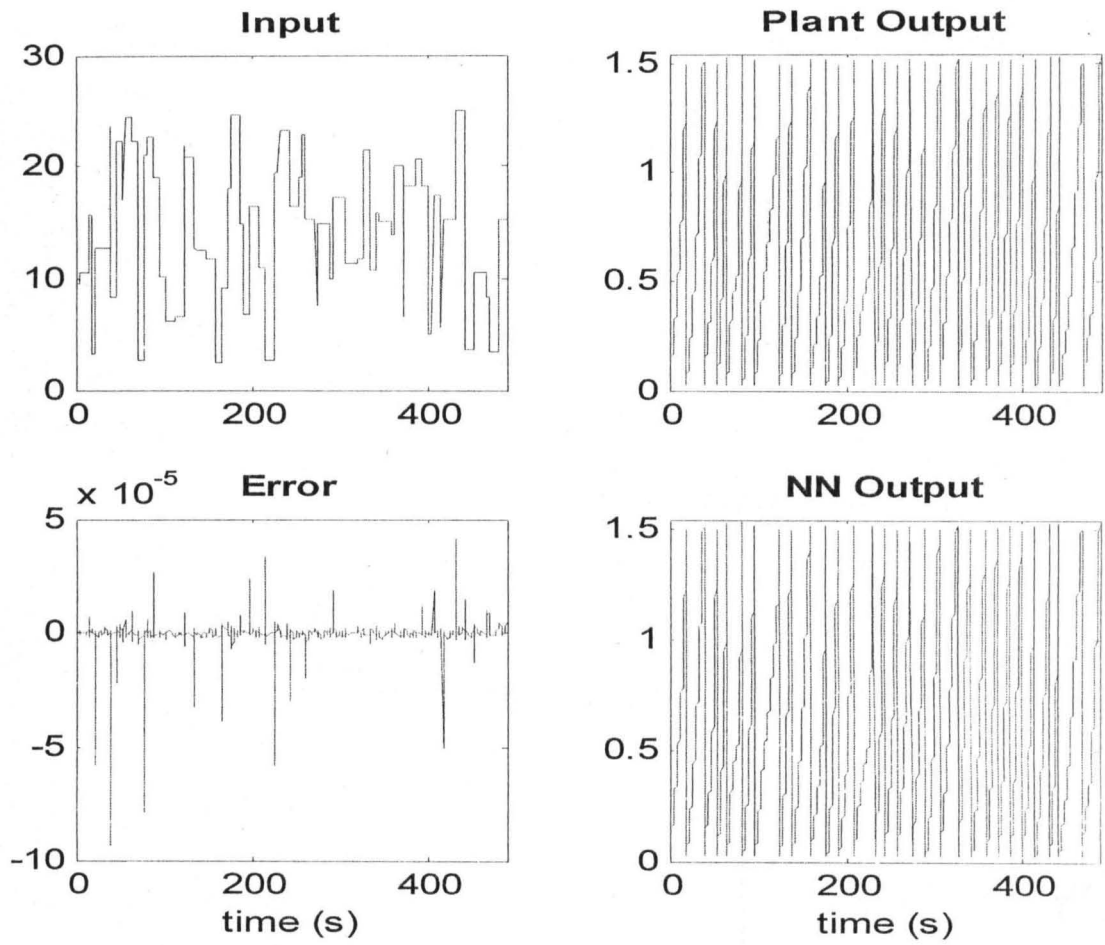


Figure 14: Validation data for NN Model Reference Control