Peninsula Technikon

Department of Electrical Engineering

Faculty of Engineering

AN INVESTIGATION INTO USING FUZZY LOGIC TECHNIQUES TO CONTROL A REAL-WORLD APPLICATION

by

Quinton Jerome Bart

Submitted in fulfilment of the requirement for the
Masters Degree of Technology (Mtech): Electrical Engineering

Under the Supervision of

Professor Raynitchka Tzoneva

December 2002

DECLARATION

I, Quinton Jerome Bart, hereby declare that this dissertation represents my own work and the opinions contained therein are my own and not necessarily those of the Technikon. All references used have been correctly reported.

Signature:  ..... Name: Quinton Jerome Bart

SUMMARY

In this thesis fuzzy control is analyzed and applied to two complex processes. A process is deemed to be complex if it possesses characteristics that make it non-linear. Examples of such characteristics are backlash, hysteresis, saturation and dead zones, which are common in industrial processes. These characteristics do not allow for the easy implementation of controllers and often there would be a requirement to employ some non-linear form of control. Often with complex processes linearization techniques would be employed to enable the application of linear controllers. However, these controllers will only operate over a limited range and will have degradation in performance when subjected to unpredictable changes in parameters. Fuzzy controllers can handle non-linear characteristics in complex processes very well and also provides some transparency between the human machine interface.

Fuzzy control is based upon the fuzzy methodology that was introduced by Lotfi Zadeh (Zadeh, 1965) in his seminal paper on fuzzy sets. The first practical application (Mamdani and Assilian, 1975) paved the way for fuzzy control and although this alternate paradigm of control came up against much criticism it managed to capture the interest of many researchers. Although research into fuzzy control saw a slump in the late 70's and early 80's it rapidly progressed in the 90's with literally thousands of research papers being published.

In this study this alternate paradigm of control is investigated and applied to two processes. An initial study was done on the fundamental concepts of fuzzy logic and how this methodology can be applied to develop a controller that is fuzzy logic based. A

plethora of information exist on fuzzy theory however only the fundamental concepts are dealt with here. This is sufficient for the realization of a rudimentary controller.

A detailed investigation was also done on the development of the rules of the knowledge base of the controller with the emphasis on the development of a fuzzy controller that has characteristics akin to a proportional-integral (PI) controller. For real-time implementation a commercial software package known as fuzzyTECH was utilized to develop the software to realize a generic fuzzy PI controller. It was important to have a generic fuzzy PI controller as it was to be implemented on two separate processes as setpoint controllers with the only requirement being the adjustment of the fuzzy parameters for application on each individual process.

The first process investigated is a 4-stage cascaded tank system, which is actually a heat exchanger. Here the fuzzy controller was utilized in a decentralized approach to independently control the setpoint of the water levels in each tank. The second process was a simulation study of an activated sludge process of a sewage wastewater treatment plant and here the fuzzy controller was used to control the dissolved oxygen concentration in an aerobic reactor of the process. Emphasis was also placed upon the tuning of the fuzzy parameters of the fuzzy controller for dissolved oxygen control and a large part of the study dealt with tuning techniques of fuzzy parameters.

Finally, the results obtained for fuzzy setpoint control in both applications provided excellent results and this has paved the way for future applications. Future work will focus upon the application of this technology in the field of sewage wastewater treatment

and plans are currently underway for the implementation of the fuzzy controller on a pilot plant and then on a real sewage plant.

ACKNOWLEDGEMENTS

The author would like to thank his wife, Audrey Bart for her time, patience and help offered during the writing of this dissertation.

The author would also like to thank Prof. Raynitchka Tzoneva for her inspirational supervision. This study turned out to be a mammoth task as it was punctuated with many problems that caused discontinuity in the project. I am therefore extremely grateful for all the effort and help I received from Professor Tzoneva.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table 6.15: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 13 to day 14

NOMENCLATURE

μ	membership degree
$\wedge$	minimum, (fuzzy) conjunction, logical AND
$\vee$	maximum, (fuzzy) disjunction, logical OR
o	max-min composition
χ	characteristic function
$\cap$	(fuzzy) set intersection (conjunction)
$\cup$	(fuzzy) set union (disjunction)
*	product
R	fuzzy relation
hgt (A)	height of fuzzy set A
supp (A)	support of fuzzy set A
core (A)	core of fuzzy set A
X, Y	domains (universes) of variables x and y
ce (A)	cylindrical extension of fuzzy set A
K_P	proportional gain
K_I	integral gain
K_D	derivative gain
C	actuator rate
$e\ (t)$	error, difference between reference signal and process output
$y_{sp}(t)$	reference signal
$y(t)$	process output (measured) value
$\Delta e(t)$	first error difference
Δu	change in control output
NH_4^+	ammonium
NO_3^-	nitrate
N_2	nitrogen
$K_La(u)$	oxygen transfer function

ABBREVIATIONS

ASP	Activated Sludge Process
ASM No.1	Activated Sludge Model No. 1
WWTP	Waste Water Treatment Plant
IAWQ	International Association On Water Quality

CHAPTER ONE

THE PROBLEM FOR FUZZY LOGIC CONTROL OF COMPLEX PROCESSES

This chapter describes the reasons for embarking upon a study in the area of fuzzy control and looks at the different approaches that have been developed in the area of modeling and control of complex systems. This chapter also looks at the aim and objectives of this study, which are explained.

1.1 THE PROBLEM FOR USING FUZZY CONTROLLERS

Technology has enabled the automation of many systems, which before has been far too complex to automatically control. Coupled with this is the demand of very high performance requirements, which sometimes cannot be easily handled by conventional control techniques. Non-linearity and uncertainty is often a characteristic of most industrial processes due to uncertainty in the relationships between the inputs and outputs of a system, which can be influenced by external disturbances. Factors causing non-linearity in systems can be due to the high order of the system, the presence of long time delays between input stimulus and output response and also the complexity of mathematical models of the system.

1.1.2 Why Non-linear Control

1.1.2.1 Improvement of existing control systems

When linear controllers are designed it is assumed that the process of concern is linear. In essence most real world applications have linear characteristics only over a small range of operation. This implies that the designed linear controller will only perform satisfactorily over a limited range. When certain nonlinearities of the system come into

effect the controller will either perform poorly or become unstable. On the other hand a non-linear controller will be able to compensate for the non-linear characteristics in the system.

1.1.2.2 Analysis of Hard Non-linearities

In linear control it is also often assumed that the derived model of the process can be linearized. However certain characteristics such as backlash, hysteresis saturation and dead zones, which are common in industrial processes, will not allow for linear approximation. The effects that they produce cannot be derived from linear methods and therefore non-linear approaches have to be adopted to predict the performance of the system in the presence of these non-linearities.

1.1.2.3 Dealing with Uncertainties in Model Parameters

When linear controllers are designed the assumption is made that the parameters of the system model are fairly well known. However many control problems are related to uncertainties in the parameters. This can occur in processes where there are slow or abrupt variations in parameters over time. This variation in parameters can manifest itself as degradation in controller performance or instability. With the use of robust and adaptive controllers non-linear characteristics can be introduced into the control system to tolerate the uncertainties.

1.1.2.4 Design Simplicity

Non-linear control designs can be simpler and more intuitive such as that offered by the fuzzy control methodology. Often industrial plants that posses many uncertainties as mentioned in section 1.1.2.3 can be easily controlled by human operators who only posses a global understanding of the process. Very often a fuzzy controller can be easily deigned to mimic the control strategies of the operator. Hence the simplicity of the design without a cumbersome complex mathematical model.

1.2 FUZZY LOGIC AND CONTROL ENGINEERING

1.2.1 Brief History

Lotfi A.Zadeh initiated fuzzy theory in 1965 when he introduced the concept of fuzzy sets (Zadeh, 1965). After the introduction of fuzzy sets Zadeh proposed the concept of fuzzy algorithms in 1968 (Zadeh, 1968) and fuzzy decision making in 1970 (Bellman and Zadeh, 1970). However the paper, which established the foundation for control, was published in 1973 (Zadeh, 1973). In this paper he introduced the concept of linguistic variables and proposed the use of IF-THEN rules to formulate human knowledge.

It was then in 1975 when Mamdani and Assilian published their results in an application of a fuzzy controller to control a steam engine (Mamdani and Assilian, 1975).
The first industrial process to be controlled by a fuzzy controller was a cement kiln developed by Holmblad and Ostergaard in 1978. The late '70s and early '80s saw a slump in the area of fuzzy control, as it never received much support. It was the Japanese who took advantage of this technology and Japanese researchers developed numerous applications (Sugeno and Nishida, 1985;Hirota, Arai and Hachisu, 1989;Yamakawa, 1989). The success of these many applications surprised many researches in the United States and Europe and as a result there was a renewed interest in fuzzy logic and fuzzy control. Today there are literally thousands of researches worldwide that are researching the area of fuzzy control and related technologies.

1.2.2 Modeling

Any engineering discipline necessitates the development of a mathematical model. Models are important especially in this modern day and age where impressive computer simulations can be performed to enable engineers and scientists to analyze and better understand a system's behavior. In terms of control understanding the process dynamics constitutes the most important aspect in the control of the process. Through process

modeling and simulation with computers a process can be better understood before the development of any hardware (Franks, 1972; Holland & Liapsis, 1983; Morbidella, Sewida, Storti & Carras, 1982).

The traditional engineering approach to modeling is gaining a thorough understanding of the nature and behavior of the system and then developing a suitable mathematical model. This is termed the "white-box" approach. However to have a good understanding of the mechanisms involved in the process for the development of an adequate model can be limiting when there exists uncertainties in underlying phenomena and uncertainty about process parameters. Even if a fairly good model is derived it is the task of system identification techniques to estimate the parameters from measured data. This could however also prove to be inadequate as identification methods have developed to a fairly mature level for linear processes only and most real world processes have non-linear characteristics.

Another approach would be to approximate the system under study by some "black-box" structure that acts as a general function approximator. With the use of process data an appropriate structure of the approximator can be developed but it is not really related to the structure of the real system. A major drawback of this approach is that the parameters of the model do not have any physical significance

Now, the advantage of the fuzzy methodology introduces a different approach known as "gray box" modeling. This type of modeling is an attempt to combine both "white box" and "black box" techniques where well-understood parts of the system are modeled using physical knowledge and the vague and uncertain parts are determined in a "black box" manner using process data. The advantage here is that heuristic information such as knowledge and experience of engineers and operators can be incorporated into the model. The different types of modeling paradigms are depicted in Table 1 (Babuska, 1998).

Modeling approach	Source of information	Method of acquisition	Example	Deficiency
White box	Formal knowledge and data	Mathematical	Differential equations	Cannot use "soft" knowledge
Black box	Data	Optimization (learning)	Regression, neural network	Cannot at all use knowledge
Fuzzy	Various knowledge and data	Knowledge-based and learning	Rule based model	"Curse" of dimensionality.

Table 1.1: Different modeling paradigms

1.2.2.1 The Relevance of Fuzzy Modeling of Real World Applications

It is often the case that systems such as electro-mechanical systems can be easily mathematically modeled because physical laws govern them. The problem with most real world applications is that there does not exist enough data to adequately characterize a system mathematically. This is especially the case with systems in the area of biotechnology, ecology and sociology. Systems in these areas may be under the control of human operators and a significant portion of the data is in the form of human experience. Mathematical models fail to encompass this knowledge resulting in the omission of a very important factor that can contribute to the design of a more efficient controller. This knowledge however can be expressed in the form of natural language and with the implementation of fuzzy rule-based systems a knowledge base can be developed encompassing the knowledge of experts in the particular field of interest (Pedrycz, 1990; Yager & Filev, 1994)

1.2.2.2 Rule Based Fuzzy Models

Rule based fuzzy models are constitutes the classical type of fuzzy model. With this type of model variables are related to each other in terms of IF-THEN rules. There are fundamentally two types. The first being a linguistic model (Mamdani-type) that implies that both the antecedent and consequent of the IF-THEN rule are fuzzy propositions. This will be elaborated upon in chapter 2. The second type known as the Takagi-Sugeno (TS) model has a fuzzy proposition in the antecedent and a crisp function for the consequent. The different models can be defined as the following

- Linguistic fuzzy Model (Mamdani, 1977)

Rules of such a model are in the following format

If antecedent is A then consequent is B

Where, A and B are linguistic terms such as "small", "medium", "large". This type of modeling uses rules consisting of only linguistic terms

- Takagi-Sugeno model (Takagi & Sugeno, 1985)

Rules of such a model are in the following format

If antecedent is A then y=f (antecedent)

Where A is a linguistic term, y is a function of the antecedent.

This type of modeling combines both linguistic terms as well as mathematical expressions in the rule base. The linguistic terms form part of the antecedent and the mathematical expression is in the consequent.

These two types of models form the fundamental fuzzy models of which there are many variations.

1.2.2.3 Developing Fuzzy Models

As it is indicated in Table 1 fuzzy models are flexible in that sources of information for the development of fuzzy models can be both of a quantitative and qualitative nature. The qualitative data, which usually originates from human operators, can be regarded as information from "experts". Hence fuzzy models can be regarded as simple fuzzy expert systems (Zimmermann, 1987)

Historical data from processes or experimentation can also be used for building fuzzy models. This constitutes the quantitative data and models can be derived from methods based on fuzzy logic, neural networks and data analysis techniques. With integration of neural networks and fuzzy logic neuro-fuzzy modeling (Jang, 1993; Jang & Sun, 1993; Brown & Harris, 1994) was developed. With non-linear systems measured data would be used for purposes of identification. In this arena neural networks and fuzzy logic techniques are often used for this purpose. Also when input-output data is available fuzzy models can provide effective approximation of the input-output relationship with reasonable accuracy. This is coined the term general function approximation (Kosko, 1994; Wang, 1992; Zeng & Singh, 1995).

1.2.2.4 Structure and Fuzzy Parameters

The structure of the fuzzy model determines how flexible the model is in terms of its approximation of the process or system. This structure is dictated by the following (Babuska, 1998).

- Input and output variables.
  This includes both the selection of appropriate variables as inputs and outputs for the model as well as the number of input and output variables
- Rule Structure
  This is the choice between the Mamdani and Takagi-Sugeno type
- The type and number of fuzzy sets for each input and output linguistic variable

This choice has a great effect on the granularity of the control surface.

- Type of inference mechanism, connective operators, defuzzification method
  This choice is usually restricted by the type of model (Mamdani or Takagi-Sugeno) and on the low level relates to the necessary computations involved for output determination.

Once the structure is chosen, the fuzzy model still has to be fine-tuned and this is achieved by adjusting the parameters of the fuzzy controller. Parameter adjustment can be deemed as the most crucial aspect of fuzzy controller design and is a large topic of research. It is here where training algorithms from the area of neural networks can be employed to optimize the parameters. This is elaborated upon in chapter 6 where the tuning of fuzzy parameters will be investigated.

1.3 FUZZY CONTROL

Broadly speaking fuzzy control can be subdivided into two sections (Wang, 1997)

- Nonadaptive fuzzy control
- Adaptive fuzzy control

In nonadaptive fuzzy control the controller is static, which means that the structure and parameters of the fuzzy controller are fixed and does not change during real-time operation. In adaptive control the fuzzy controller is dynamic with changes to its structure and parameters during real time operation. In terms of complexity nonadaptive controllers are simpler than adaptive controllers but they require more knowledge of the process for controller development. The opposite is true for adaptive controllers as it will start with an initial fundamental controller and then adapt in the learning process.

1.3.1 Nonadaptive Fuzzy Control

With nonadaptive control a typical approach would be the trial-and-error method, which can be roughly summarized, in the following method (Wang, 1997)

- Step 1: Analyze the real system and choose state and control variables.
  The state variables should characterize the key features of the system and the control variables should be able to influence the states of the system. The state variables are the inputs to the fuzzy controller and the control variables are the outputs of the fuzzy controller. Essentially this step defines the domain in which the fuzzy controller is going to operate.

- Step 2: Derive fuzzy IF-THEN rules that relate the state variables with the control variables.
  The formulation of these rules can be achieved by means of two heuristic approaches. The most common approach involves an introspective verbalization of human expertise. Another approach includes an interrogation of experienced experts or operators using a carefully organized questionnaire. In these ways a prototype of fuzzy control rules can be obtained.

- Step 3: Combine these derived fuzzy IF-THEN rules into a fuzzy system and test the closed-loop system with this fuzzy system as the controller.
  That is, run the closed system with the fuzzy controller and if the performance is not satisfactory, fine-tune or redesign the fuzzy controller by trial and error and repeat the procedure until the performance is satisfactory.

1.3.2 Adaptive Fuzzy Control

The initial idea of utilizing fuzzy controllers is that they have the ability to handle large uncertainties in processes. Hence the basic idea of adaptive fuzzy control is to have a fuzzy controller that can maintain consistent performance in the presence of varying plant

parameters and structures. Figure 1.1 (Wang, 1997) shows the basic configuration of an adaptive fuzzy control system.

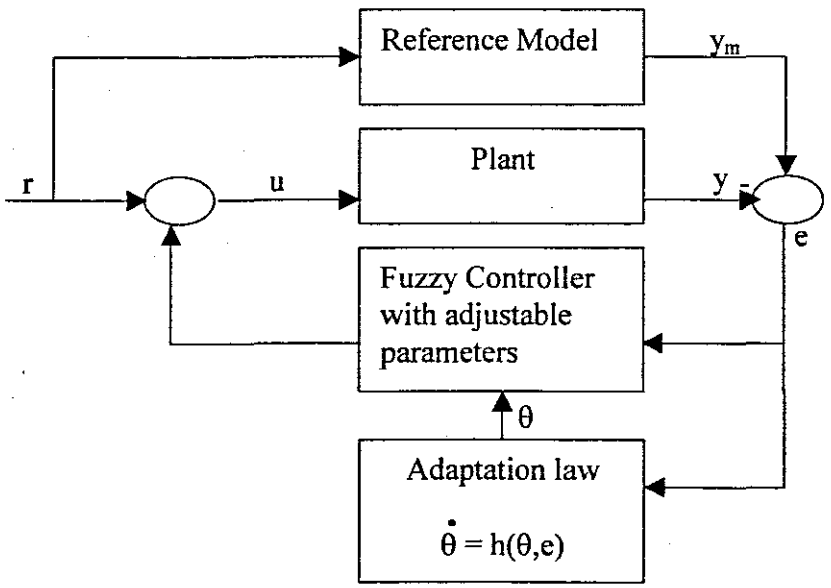


Figure 1.1: Basic configuration of adaptive fuzzy control systems

The reference model is used to specify the ideal response that the fuzzy control system should follow. The plant is assumed to contain unknown components. The fuzzy controller is constructed from fuzzy systems whose parameters θ are adjustable. The adaptation law adjusts the parameters θ online such that the plant output $y\ (t)$ tracks the reference model $y_m\ (t)$.

In chapter 6 a description of different adaptive techniques is given with emphasis on techniques borrowed from neural networks and genetic algorithms.

1.3.3 Types of Fuzzy Controllers

Proportional-Integral-Derivative (PID) controllers have dominated the process control industry for many years; however there has always been problems related to the practical implementation of these controllers to effectively control processes that exhibit non-linear characteristics. These PID controllers provide excellent control for linear systems, with simple design and low cost. Reports have shown (Yamamoto & Hashimoto, 1991) that the Japanese market alone have PID control loops in more than 90% of the industrial applications and it is believed to be similar elsewhere (Swallow, 1991).

Because PID controllers are effective in linear systems many variations of the PID controllers were developed to control non-linear processes. Amongst the techniques proposed were auto-tuning and adaptive PID controllers (Astrom et al., 1992) and also non-conventional PID controllers which employed fuzzy logic technology were developed (Ying et al., 1990; He et al., 1993; Malki et al., 1994). The fuzzy control methodology has been reported to be ideal for controlling non-linear systems as PID controllers are not known as being able to work effectively for these systems (Tang et al, 2001).

From a control point of view information and data from complex processes can be regarded as inexact and imprecise and the fuzzy logic methodology provides an efficient way of handling this data for purposes of reasoning. The effectiveness of this methodology is due to the fact that the uncertain data can be utilized in an exact algorithm for control. In essence the fuzzy controller is represented by a set of linguistic if-then rules that processes measured input values and produces exact output signals in a deterministic way.

The fundamental advantage of fuzzy controllers is that these controllers can control complex processes with non-linear characteristics very well. After all, on a very abstract level all controllers can be regarded as simply input-output mappings. Conventional controllers are expressed mathematically and fuzzy controllers make use of a rule base.

Because of the success of the implementation of fuzzy PID, PD and PI controllers work in this area is under continual development. In retrospect many successes have been reported. Linear fuzzy PID controllers were developed that gave precisely the same response as conventional PID controllers (Buckley and Ying, 1990). Also fuzzy PI controllers were developed yielding results comparative to conventional PI controllers (Ying, 1993). In a similar vein two types of improved fuzzy PI controllers were developed (Lee, 1993). Latest developments in the area of fuzzy PI controllers for various applications exhibiting non-linear characteristics have been progressing (Tang, K et al., 2001; Tang, W et al., 2001; Ranganathan et al, 2002)

This literature has shown that the work in the area of fuzzy controller development and enhancement is indeed an active area as in effect a work in progress. It is thus worthwhile to study the fuzzy control methodology, as previous implementations of fuzzy controllers for complex non-linear processes have proven effective and positive.

1.4 INDUSTRIAL APPLICATIONS

Many applications ranging from robotics (Bestaoui, 1989; Tomei, 1991; Tang and Chen, 1994; Tang, W. et al., 2001) to Chemical and Biological processes have been successful with the alternative fuzzy control methodology. Examples of this are the application of fuzzy control for heating control (Georgescu et al., 1993) and fuzzy control of steam turbines (Kiupel et al, 1994). A fuzzy PI type controller was developed to control the pH titration in a continuously stirred tank reactor (Qin et al., 1994). In wastewater treatment fuzzy control has been used for direct sludge flow (Yu et al, 1990) and for automatic startup of a high rate anaerobic reactor (Gomez et al, 1998)

1.5 OBJECTIVE OF DISSERTATION

The objective of this dissertation is to study the fuzzy control methodology, and to develop fuzzy control strategies, algorithms and tuning procedures for setpoint control of level in a cascaded tank system and for setpoint control of the dissolved oxygen concentration in the aerobic tanks of an activated sludge system. These two processes were studied because of their non-linear characteristics and also because of the two systems being composed of interconnected units. These objectives are as follows:

- Study of the fuzzy logic methodology for control
- Development of generic fuzzy controller for setpoint control
- Model development for fuzzy control of the levels in a cascaded tank system
- Software development for simulation and real time control of the levels in the cascaded tank system
- Study of the activated sludge process including software development for the simulation of the process.
- Implementation of fuzzy setpoint control of the dissolved oxygen concentration in an aerobic tank by simulation.

1.6 ROADMAP OF DISSERTATION

In conclusion a brief outline of the subsequent chapters is given in section 1.6.1. This provides a roadmap of what to expect in the forthcoming chapters and describes the layout of the dissertation.

1.6.1 Outline of Dissertation

Chapter two gives an introduction to fuzzy set theory and how this can be applied in the field of control engineering. This chapter provides the reader unfamiliar to the various concepts of fuzzy theory and fuzzy control with an overview and paves the way for understanding the development of the controller.

Chapter three describes the development of a generic fuzzy PI controller and shows how the fuzzy PI controller can be developed based upon its conventional counterpart. It also provides a step-by-step procedure for the realization of the fuzzy controller in software with the aid of a commercial package called FUZZYTECH, which is widely used for the development of code for the control of industrial applications.

Chapter four describes the first process, a cascaded tank system upon which the generic controller was implemented and also shows the results of the controller implementation. The results of the implementation on the actual process is shown and compared with those obtained from the simulation. The simulation of the process was done in the LABVIEW simulation environment.

Chapter five describes the second process, which is an Activated Sludge Process (ASP), used for the biological treatment of sewage wastewater. This is a process that exhibits highly non-linear characteristics and is ideal for the implementation of fuzzy control. There are many aspects that can be controlled in this process and in the study the control of the dissolved oxygen concentration is focused upon. This part of the study is purely a simulation study which forms part of a preliminary study for implementation on a pilot ASP process and then later on an actual wastewater treatment plant. This will form part of future work to be embarked upon.

Chapter six describes different tuning procedures for fuzzy control and describes a tuning method and algorithm for the fuzzy dissolved oxygen concentration setpoint controller.

Chapter seven is the concluding chapter that discusses the results of the fuzzy controller implementation and provides a summary of the work done as well as future work to be embarked upon.

CHAPTER TWO

BASIC CONCEPTS OF FUZZY LOGIC AND FUZZY CONTROL

This chapter provides an overview of the fuzzy logic technology employed in this dissertation as it will be helpful to explain the fundamental concepts of fuzzy logic, such as fuzzy sets, fuzzy operations and how it relates to the field of control engineering.

Control engineering is a highly mathematical discipline that has its foundations in mathematical concepts. Systems to be controlled are expressed in terms of a mathematical model derived from differential equations that describe the dynamics and internal characteristics of the process. These conventional techniques have been used for more than 50 years and achieved 90% of very satisfactory control.

However generally, complex processes that are non-linear or time variant, cannot be easily modeled mathematically. Also high dimensional mathematical models are not really useful, as many control theory concepts tend to be applicable to models of a low order. With high dimensional models there is a need for enormous computational requirements. These high dimensional models can also result in ill-conditioned problems and stiff numerical problems due to the interaction of slow and fast dynamics (Kokotovic *et al.,* 1986*)*. As a consequence these processes cannot be analyzed and controlled using conventional control methods and hence necessitates alternate methods of control. A typical example of a complex process is a cement kiln. One of the first commercial fuzzy controller applications was the control of such a cement kiln, which was developed, in the early 1980's by F.L. Smidth & Co in conjunction with researchers from Queen Mary College. In industry, a human operator that bases his controlling decisions on experience and familiarization with the plant performs the controlling function. His knowledge of the plant is purely of a global nature and not of the internal dynamics of the system. His controlling actions are related to the dynamics of the process output states. These operators perform their tasks quite effectively and this essentially encouraged the development of a controller that implements the same control strategy e.g. a controller that is fuzzy logic based.

2.1 FUZZY LOGIC OVERVIEW

Fuzzy logic emulates the way humans think, perceive and make decisions. This form of logic has its foundations in fuzzy set theory that was introduced by Lotfi A. Zadeh in his paper on fuzzy sets in 1965 (Zadeh, 1965). Fuzzy logic is a misnomer as it conjures up the idea that it is an attempt to fuzzify logic. It is however a rigorous methodology for dealing with elements of uncertainty and vagueness.

The introduction of fuzzy logic to the area of control was first introduced by Mamdani (Mamdani and Assilian, 1975) which was initially outlined by Zadeh (Zadeh, 1973), with the intention of duplicating the behavior of a human operator.
Also the discussion of fuzzy logic is limited to the calculus of fuzzy if/then rules as controllers can be built utilizing the basic fundamental theory of fuzzy logic. There is a plethora of literature on fuzzy logic with the bulk of it being theoretical. (Mendel M, 1995). Only a small percentage is really required to realize a controller. The following statement made by Zadeh has corroborated this. (Zadeh, 1992):

The calculus of fuzzy if/then rules is simple and close to intuition. Furthermore, it is largely self-contained and does not require an extensive familiarity with fuzzy logic.

This however, is not to say that new theoretical ideas have not contributed to new practical solutions.

2.1.1 Description of a Fuzzy Control Rule

Human thought is subjective and qualitative in nature. We group elements within certain limits and use these groupings to base our decisions upon. In effect we are unconsciously summarizing a continuous space into a finite number of groups to facilitate our thought processes. To illustrate this qualitative approach in terms of control consider the following example. The temperature of a room is to be controlled by the adjustment of fan speed. Under manual control a typical thought process would take the following form.

"If the room temperature is hot then the fan speed is fast."

In the statement above "hot" and "fast" will encompass a range of values on the numeric space of "temperature" and "speed" respectively. If a mathematical formula is to be used it would assume a general form as below.

$$S = K(\Delta T) \qquad (2.1)$$

Here S is the speed of the fan, K is a constant and delta T is the difference between the set temperature and the actual temperature. From this example one can appreciate the difference between the human approach to control, which is qualitative, and the mathematical formula, which is quantitative in nature.

Since the controller will be utilized on digital platforms, which are quantitative in nature, a technique has to be implemented to transform the qualitative expression into a quantitative form without losing the effect of a humanistic approach. Fuzzy set theory provides this transformation.

2.2 SET THEORY

Gaining an understanding of fuzzy concepts is best understood by comparing these ideas with classical logic concepts that nearly everyone learns in school. Classical logic is based on the writings of Aristotle that gave rise to two fundamental axioms.

(i) The Law of Contradiction.
(ii) The Law of the Excluded Middle.

The first axiom implies that an element X cannot be Y and not-Y at the same time. A light bulb cannot be on and off at the same time.
The second axiom implies that element X must either be Y or not-Y. A light bulb must either be on or off.

These axioms basically imply that there exist boundaries between characteristics that elements can possess and that there are no gray areas. In terms of classical set theory it stipulates that an element either belongs to a set 'Y'or it does not. This is where fuzzy set theory differs in that it allows elements to have varying degrees of membership in set 'Y'.

2.2.1 Crisp Sets

When considering crisp sets this concept is defined by what is known as a characteristic function, denoted here by χ. For example consider the following set.

$$Y= \{5,6,7,8,9\} \tag{2.2}$$

In this set the characteristic function of "5" would be "1". For the number "2" it would be "0". Therefore the characteristic function, denoted here by χ, can be defined as whether an element belongs to a set or not. For example consider a set "A", then

$$\chi_A = \begin{cases} 0 \text{ if } x \notin A \\ 1 \text{ if } x \in A \end{cases} \tag{2.3}$$

2.2.2 Fuzzy Sets

In fuzzy sets however this characteristic function will have values within the range [0,1]. If an element in a fuzzy set has a characteristic function value that is close to 1 it implies that the element has a high degree of membership in that particular set.
In fuzzy sets this characteristic function is more commonly known as a membership function and denoted by $\mu(x)$. Such fuzzy sets can then be represented as.:

$$F = \{1/a, 0.6/c, 0.2/f, 0.5/g\} \tag{2.4}$$

The notation above does not utilise the "/" as a divisor as in mathematics. It merely separates the element, on the right of the slash, and its membership function value on the left.

2.2.3 Comparison between Fuzzy Sets and Crisp Sets

To illustrate this idea consider a variable such as temperature with a range of values on the closed interval [0,100] degrees. If we consider temperatures ranging from 45 to 55 degrees it could be described in terms of classical set theory by Figure 2.1 and in terms of a fuzzy set by Figure 2.2

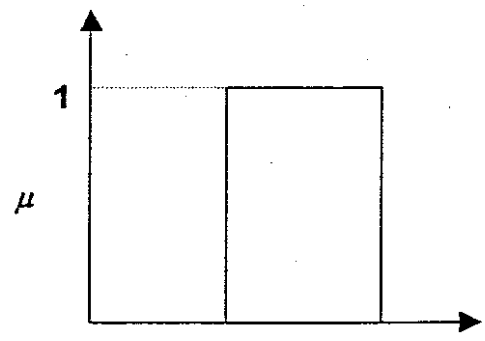


Figure.2.1: Crisp Set

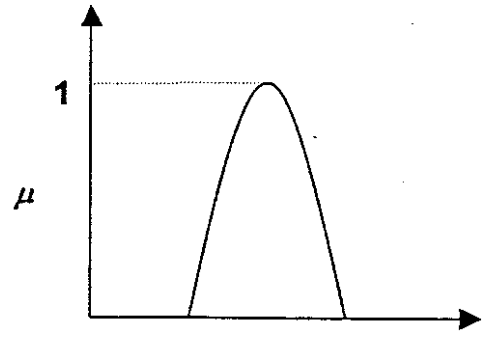


Figure.2.2: Fuzzy set

The definition of a fuzzy set was formally defined by Zadeh (Zadeh, 1965])as:
" A fuzzy subset 'A' of universe 'U' is characterized by a membership function UA:U→[0,1] which associates with each element 'y' of 'U' a number UA(y) in the interval [0,1] which represents the grade of membership UA(y) in the interval [0,1] which represents the grade of membership in 'A'."

2.3 FUZZY RELATIONS

In control these relations are very important because they describe interactions between variables. This is particularly important when one considers the if/then rule, because a fuzzy relation can describe these rules. If one considers fuzzy sets on separate universes they can be related by the cartesian product. On universes X and Y the relation R = X*Y. A relation can also be regarded as the realization of fuzzy sets to a multi-dimensional space. In terms of membership functions the relation R is defined as

$$\mu_R(x,y)=min(\mu_X(x),\mu_Y(y)) \quad (2.5)$$

In practice relations are the rule translation matrices of a fuzzy controller. The rule base is essentially comprised of a number of rules as the one described below.

"If room temperature is hot then fan speed is fast"

The terms "hot" and "fast" can be regarded as fuzzy sets on separate universes and their cartesian product forms the relation. See Figure 2.3 below. For each rule in the rule base a relation will be formed .These relations will be comprised of a matrix with entries defined by:

$$\mu_R(x,y)=min(\mu_X(x),\mu_Y(y)) \quad (2.6)$$

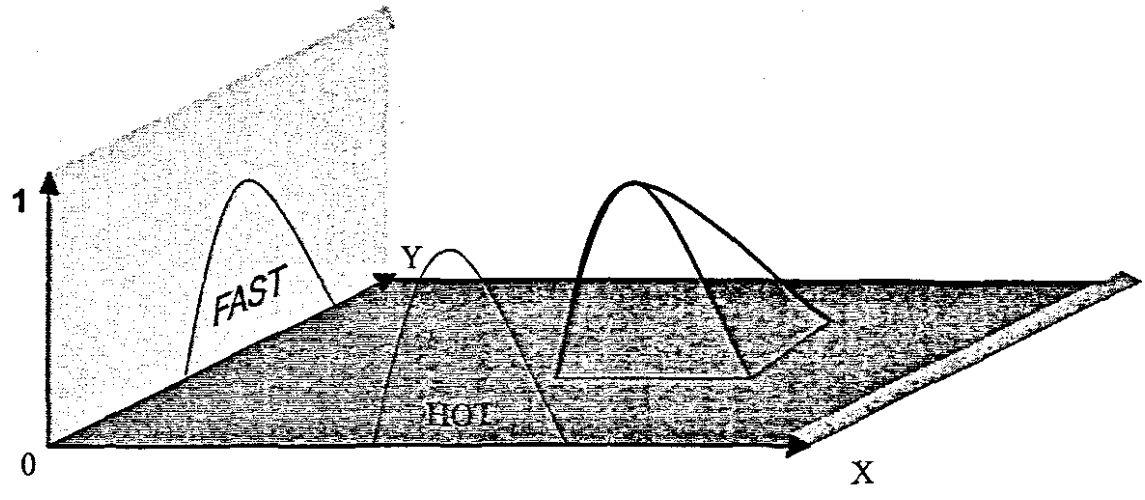


Figure 2.3: Relation "R" formed by the Cartesian product of "X" and "Y".

With respect to fuzzy sets a fuzzy relation is a fuzzy set with a membership function in two or more variables.

Driankov *et al* (1993, p.64) gives the following example to describe, "approximately equal".

When $U = \{1,2,3\}$ then the fuzzy binary relation can be described as follows.

$$1/(1,1)+1/(2,2)+1/(3,3)$$
$$+0.8/(1,2)+0.8/(2,3)+0.8/(2,1)+0.8/(3,2)$$

$$+0.3/(1,3)+0.3/(3,1)$$

The membership function μ_R of this relation can be described by

$$\mu_R(x,y) = \begin{cases} 1 & \text{when } x = y, \\ 0.8 & \text{when } |x-y| = 1 \\ 0.3 & \text{when } |x-y| = 2 \end{cases}$$

In matrix notation it can be represented in the following format.

		Y		
		1	2	3
X	1	1	0.8	0.3
	2	0.8	1	0.8
	3	0.3	0.8	1

Table 2.1: Table showing the matrix of the relation "approximately equal"

2.4 MATHEMATICAL OPERATIONS ON FUZZY SETS

2.4.1 Intersection

Fuzzy mathematics is important in the development of the fuzzy controller and a short description of the essential concepts will be briefly explained. Intersection of two fuzzy sets is characterised by the equation (2.7) and depicted in Figure 2.4.

$$\mu_{A\cap B} = \min(\mu_A(x),\mu_B(x)) \tag{2.7}$$

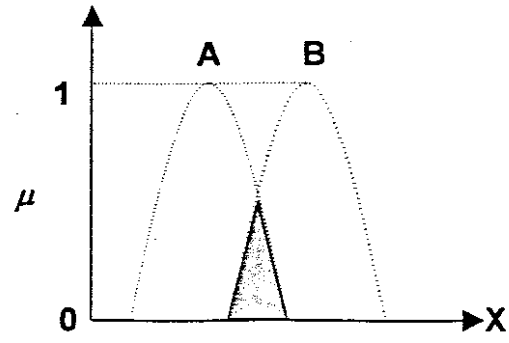


Figure 2.4: Intersection of two fuzzy sets

2.4.2 Union

Union of two fuzzy sets is characterized by the equation (2.8) and depicted in Figure.2.5.

$$\mu_{A \cup B} = \max(\mu_A(x), \mu_B(x)) \tag{2.8}$$

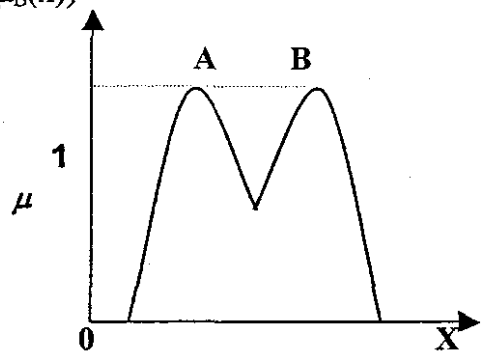


Figure: 2.5: Union of two fuzzy sets

2.4.3 Height

Height, is defined as the largest value for a membership function of a fuzzy set. Hence for fuzzy set A on a universe of discourse X:

$$\text{hgt}(A) = \sup_{x \in X} \mu_A(x) \tag{2.9}$$

2.4.4 Projection.

From Figure 2.3 , consider universe of discourse X and Y .

Projection is the process whereby,for example, a fuzzy set on a particular universe of discourse can be derived when the profile of the relation is projected onto that universe and expressed by equation (2.10).

$$\text{Proj R on Y} = \int_{Y} \sup_{x} \mu_R(x,y)/y \tag{2.10}$$

As an example consider the relation R below.

	y1	*y2*	*y3*	*y4*
x1	0.4	1	0.2	0.6
x2	0.2	0.8	0	0.1
x3	0.9	1	0.5	0.8

Table 2.2: Table of the matrix showing the membership values of a particular relation formed by universes X and Y in Figure (2.3)

For projection onto X the following values will be found by finding the maximum value in each row. Hence for each x value the following membership degrees will be:

- $x1 = 1$
- $x2 = 0.8$
- $x3 = 1$

As a result the fuzzy set derived from this procedure is:

$$\text{Proj } R \text{ on } X = 1/x1 + 0.8/x2 + 1/x3 \tag{2.11}$$

Also the projection onto Y can be found by extracting the maximum values from the columns. This will yield the following result.

$$\text{Proj } R \text{ on } Y = 0.9/y1 + 1/y2 + 0.5/y3 + 0.8/y4 \tag{2.12}$$

2.4.5 Cylindrical Extension

This operation can be viewed as the opposite procedure to projection. In this instance fuzzy sets are extended to fuzzy relations. For a binary relation the following definition applies where A is a fuzzy set on universe of discourse Y and there exist a cartesian product space of X x Y.

$$ce(A) = \int_{X \times Y} \mu_A(y)/(x,y) \tag{2.13}$$

By way of an example consider the fuzzy sets obtained in by equation (2.11) and (2.12). When the fuzzy set obtained by equation (2.11) is extended onto X x Y , the following is obtained.

$ce(A) =$

	y1	*y2*	*y3*	*y4*
x1	1	1	1	1
x2	0.8	0.8	0.8	0.8
x3	1	1	1	1

Table 2.3: Matrix showing the extension of fuzzy set A.

By the same token for equation 2.12 the following is obtained.

$ce(B) =$

	y1	*y2*	*y3*	*y4*
x1	0.9	1	0.5	0.8
x2	0.9	1	0.5	0.8
x3	0.9	1	0.5	0.8

Table 2.4: Matrix showing the extension of fuzzy set B.

2.4.6 Composition

Now, by combining the concepts of both projection and cylindrical extension the operation of composition is obtained. Composition enables a fuzzy set on one universe to be inferred given a fuzzy relation and a fuzzy set on another universe. This fuzzy set is obtained by taking the intersection of the extension, from the first universe, and the fuzzy relation and projecting the result onto the second universe. Composition is annotated with the symbol o.

Hence given a set A on a universe X and a fuzzy relation R in the space $X \times Y$, a set B can be inferred by composition. This can be defined as:

$$B = A \circ R = \text{proj } (ce\ (A) \cap R)\ on\ Y \qquad (2.14)$$

Also the membership function of B is as follows. Here the intersection is performed using the maximum operation and projection with the minimum operation.

$$\mu_B(y)=\max_x [\min(\mu_A(x), \mu_R(x,y)] \quad (2.15)$$

Alternatively the membership function can be defined as follows. Here the intersection is performed using the maximum operation and projection with the product operation.

$$\mu_B(y)=\max_x \mu_A(x). \mu_R(x,y) \quad (2.16)$$

2.5 FUZZY RULES

Up till now it has been shown how important fuzzy relations are in modeling of the actual rule. The next interesting aspect of the rule is how the connective "and " and the implication "if .....then "is modeled.

2.5.1 Fuzzy Connectives

Previously in section 2.4.1 a formal definition for the intersection of two fuzzy sets was stated. Here the min operator which is used for determining the intersection of fuzzy sets was introduced and it is this operation that is used for modeling the "and" connective in a fuzzy rule. Also for the "or" operation the max operator which is used to determine the union of fuzzy sets is used. When connectives operate on fuzzy sets that exist on the same universe the result will be a fuzzy set on the same universe. However when they operate on fuzzy sets that exist on different universes the result will be a fuzzy relation that exists in the product space of the two universes.

2.5.2 Fuzzy Implications

There exists a number of models for purposes of implication with the most popular implication being that defined by the minimum operation. Bear in mind that a fuzzy rule can be modeled using a relation. Consider the following rule.

If a is A then c is C.

Here the rule can be modeled by a relation R . A represents the fuzzy set that is the antecedent portion of the rule and C represents the fuzzy set that is the consequent of the rule. "a"and "c"represents the fuzzy variables which are essentially the universes of discourse. Hence implication can be defined as equation 2.6 above.

2.5.3 Compositional Rule of Inference

The next concept of importance is the compositional rule of inference.This relates to the relations generated and enables the output fuzzy set to be determined when the input and a relation is given. See figure 2.6. Formally stated the compositional ruleof inference is defined as follows:

" If R is a fuzzy relation from U to V, and x is a fuzzy subset of U then the fuzzy subset y of V which is induced by x is given by the composition of R and x ;that is

$$y = x \circ R"$$

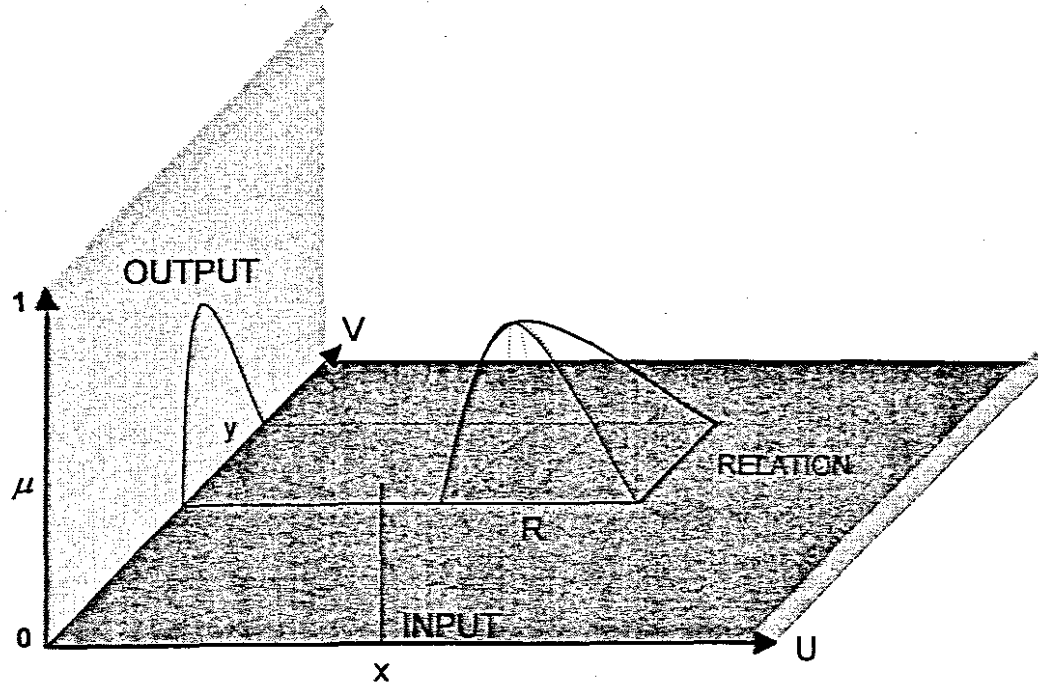


Figure 2.6: Diagram depicting the compositional rule of inference

In practice 'x' is normally a single value as depicted in the Figure (2.6) and this value will interact with more than one relation and the output fuzzy set chosen is defined by

$$\mu_Y(y) = max[min(\mu_X(x), \mu_R(x,y))] \tag{2.17}$$

Note that for explanatory purposes Figure (2.6) depicts one relation only.

Basically it implies that when a rule has been modeled by a relation R , an output fuzzy set B can be inferred from an input set X by implementing composition

2.6 SUMMARY OF CHAPTER

This chapter introduced the concept of fuzzy logic and drew the distinction between crisp logic and fuzzy logic. Certain fundamental fuzzy logic concepts were introduced and it was shown how based upon the concepts of fuzzy logic the rules of the rule base could be evaluated. This chapter has explained the fundamental mathematical computations involved in fuzzy logic and how it is used in rule evaluation. In chapter three the emphasis will be on the formulation of the fuzzy controller

CHAPTER THREE

FUZZY CONTROL AND FUZZY CONTROLLER SYSNTHESIS

The most obvious question would be to ask "What is fuzzy control?' Definitions are wide and varied but this form of control can be better understood by way of comparison with classical control. First and foremost there should be an understanding of the purpose of control. The purpose being to influence the behavior of a system by changing an input or inputs to that system according to certain methodologies to accomplish some desired goal.

Classical control and its associated techniques would make use of a mathematical model of the system to be controlled to define the relationship that will transform the measured state into the desired state. With classical control the most common example of a control model is the (proportional-integral-derivative) PID controller. Here the output of a system is measured and compared with the desired output of the system. The comparison is in the form of a difference value between the two and termed the error value. The error value is then taken and the output of the controller action is determined by the following equation, which represents a PID controller.

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{3.1}$$

$$\frac{du(t)}{dt} = K_P \cdot \frac{de(t)}{dt} + K_I \cdot e(t) + K_D \frac{d^2 e(t)}{dt^2} = \frac{dC}{dt} \tag{3.2}$$

where K_P, KI, KD are constants, $e(t)$ is the error term and C the actuator rate.

The major disadvantage of this type of control system implementation is that there is an assumption that the system being controlled is linear or that its behavior is such that is monotonic. Hence the issue here is that of complexity in terms of modeling of mathematically complex systems.

The fuzzy controller replaces the mathematical model above with one that is linguistically based. Essentially what this means is that the controller has a linguistic interpretation of the system being controlled. It still however processes precise or exact input data and produces exact output data. The advantage that fuzzy controllers have over their classical counterpart is that it is ideal for applications that exhibit highly non-linear characteristics. Although fuzzy control performs admirably in situations where there is a high degree of non-linearity it does not imply that fuzzy control is superior when automating control tasks (Jager, 1995).

3.1 FUZZY CONTROLLER DESIGN

There is no formal design procedure for fuzzy controllers however there are certain tasks to be performed when synthesizing any fuzzy controller. The designer has to ensure that the following steps are done.

(1) Determination of the input and output linguistic variables.
(2) The amount of fuzzy sets for each linguistic term.
(3) The choice of membership functions
(4) Rule base structure
(5) Pre and post processing of data. This will include the fuzzification and defuzzification of data.
(6) Tuning of the controller.

3.1.1 Selection Of Input and Output Variables

In this dissertation the emphasis is placed upon control mechanism for setpoint tracking. In this instance feedback control will be employed and normally this will involve determination of the error signal, which is the difference between the set-point and measured value,

where, $$e(t) = y_{sp}(t) - y(t) \tag{3.3}$$

Here the control objective would be to maintain an error signal that is as small as possible. As a second objective the change of the error rate could also be computed and used as a second input to the fuzzy controller where,

$$\Delta e(t) = e(t) - e(t\text{-}1) \tag{3.4}$$

Using these variables as inputs fuzzy controllers that behave as PI or PD controllers can be synthesized. In Section 3.2.4 this concept will be further investigated when determining the structure of the rule base. Also the amount of input variables to the controller is restricted to two as fuzzy controllers with many inputs suffer from dimensionality problems. The amount of rules in the rule base grows exponentially with the addition of input variables.

3.1.2 Type and Amount of Membership Functions

There are many different types of membership functions to choose from. From a computational point of view it is always convenient to make a selection that requires minimal use of processing power and that can be efficiently manipulated in real time. Amongst the more common membership functions the triangular type has been selected as the most viable one.

If a plot of the control surface of the fuzzy controller is analyzed it can easily be seen that the rules and the membership functions play an important role in the shaping of the control surface. The rippled surface depends upon the rules and type of membership functions used. Increasing the amount of membership functions will increase the ripple and the amplitude of the ripple will decrease. The amount of membership functions will also determine the amount of rules in the rule-base. It was decided upon to sub-divide the linguistic variables into 5 membership functions each. This will, together with two input variables generate a rule base consisting of 25 rules. This was decided upon as the shape of the control surface can still be manipulated through the process of tuning which is covered in chapter 5. In essence each input variable will have its crisp numerical values

translated from a numerical universe into a domain that is linguistically defined as depicted by the figure 3.1 where

NB is negative big, NS is negative small, Z is zero, PS is positive small and PB is positive big.

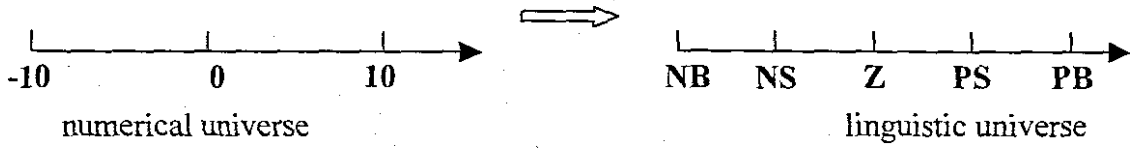


Figure.3.1: Translation from a numerical to a linguistic universe

In this study each variable is subdivided into five linguistic terms as indicated in Table 3.1 where, for the input variable "error", "Above" is furthest above the setpoint value, "CLAbove" is above the setpoint value but closer to it, "Ok" is in a region (positive or negative) that is very close to the setpoint value, "CLBelow" is below the setpoint value but close to it, and "Below" is furthest below the setpoint value.

For the input variable "error_change", "Incfast" specifies that the rate of change of error is increasing fast, "Incslow" specifies that the rate of change of error is increasing slowly, "Constant" defines a region where the rate of change of error remains fairly stable, "Decslow specifies that the rate of change of error is decreasing slowly, "Decfast" implies that the rate of change of error is decreasing fast.

For the output variable " Actuator_change", "Openfast" specifies that the signal to valve should open it fairly large, "Openslow" specifies that the signal to valve should open it in the medium range, "Constant" specifies that the signal effect a change in the valve so that it it midway open, "Closeslow" specifies that the signal should set the valve so that it is closed beyond the midway region, "Closefast" specifies that the signal should set the valve so that the valve aperture is fairly small.

error	error_change	Actuator_change
Above	Incfast	Openfast
CLAbove	Incslow	Openslow
Ok	Constant	Nochange
CLBelow	Decslow	Closeslow
Below	Decfast	Closefast

Table 3.1: Linguistic terms for each variable

The linguistic terms shown above are descriptive names that can be equated to the following previously defined terms as indicted in the table below. They can be regarded as alternate names, which are more descriptive in terms of describing the position of the process output with respect to the setpoint value.

error		error change		actuator change	
Above	NB	Incfast	NB	Openfast	NB
CLAbove	NS	Incslow	NS	Openslow	NS
Ok	Z	Constant	Z	Nochange	Z
CLBelow	PS	Decslow	PS	Closeslow	PS
Below	PB	Decfast	PB	Closefast	PB

Table 3.2: Linguistic terms equated to its sign and magnitude

3.1.3 Rule Base Structure

The structure of the rule-base is very much dependent upon the type of a fuzzy controller to be implemented. The type refers to the control action required by the controller. In this study the type of control action required is of the PI type. Consider the discrete-time version of a PI controller.

$$\Delta u(k) = K_P.\Delta e(k) + K_I.e(k) \equiv \Delta C \tag{3.5}$$

K_P and K_I are the coefficients for the proportional and integral coefficients respectively. Δu , Δe and ΔC represents change in control output, change in error and actuator change respectively.

As mentioned in Chapter 2 the rules are in the following if-then format.

If (state of system) then (control signal)

Where, (state of system) is represented by the value of the error and error change and (control signal) is assigned to the actuator change.

To have a rule-base that is comprised of rules that are PI characteristic the antecedents that describe the state of the system should be 'error' and 'error change' and the consequent that describes the control signal should be ' actuator change'.

Before proceeding with the rule base development consider table 3.2 above.

From the table it can be seen that the sign of each linguistic variable can either be positive or negative. The state of the process can be easily tracked by observing the effect all possible combinations each controller input has on the output. This is summarized in table 3.3 bearing equation (3.3) in mind.

error	error change	Effect on output
negative	negative	Above setpoint and moving away from it.
Negative	positive	Above setpoint and moving towards it.
Positive	negative	Below setpoint and moving towards it.
Positive	positive	Below setpoint and moving away from it.

Table 3.3: Position of the process output determined by sign of the input variables

Based upon table 3.3 and the linguistic terms as defined in Table 3.2 a set of rules can be easily defined. In the following section it will be shown how a list of rules for a PI fuzzy controller were formulated.

3.1.4 Rules for PI-type Controller

To assist with the formulation of the rules the output will be sub-divided into different regions.

(a) Steady state region

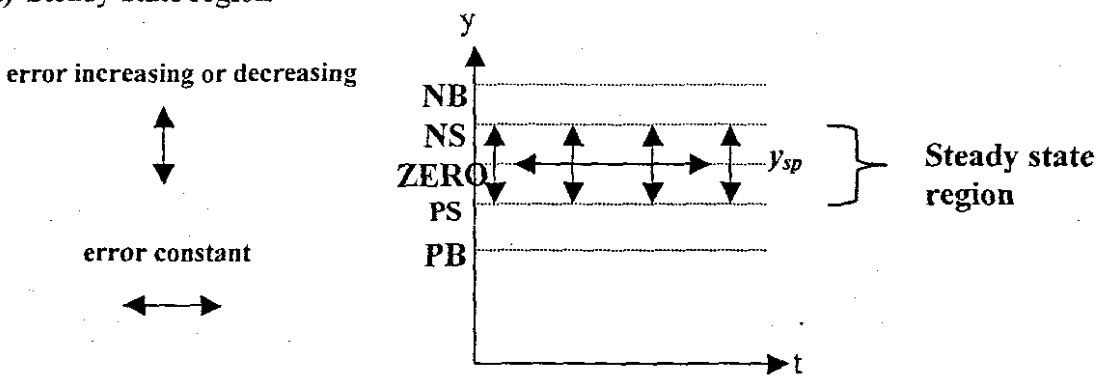


Figure.3.2: Steady state region

If one considers the region close to the setpoint (± 5 % from the setpoint) as indicated by Figure. 3.2 and the linguistic terms that are small in magnitude (NS,ZERO,PS) this area can be regarded as the steady state region. Here the input variables can be either positive, zero or negative. Hence the amount of change to be added to the previous output of the controller should also be small or zero and of the right sign to correct the deviations from the setpoint. Thus the possible rules for this region can take the following format as indicated in Table 3.4.

	IF		THEN
	Error	**Error change**	**Actuator change**
1	Negative Small	Negative Small	Positive Small
2	Negative Small	Zero	Positive Small
3	Negative Small	Positive Small	Zero
4	Zero	Negative Small	Positive Small
5	Zero	Zero	Zero
6	Zero	Positive Small	Negative Small
7	Positive Small	Negative Small	Zero
8	Positive Small	Zero	Negative Small
9	Positive Small	Positive Small	Negative Small

Table 3.4: Table showing possible rules for the steady state region

(b) Region significantly above the setpoint.

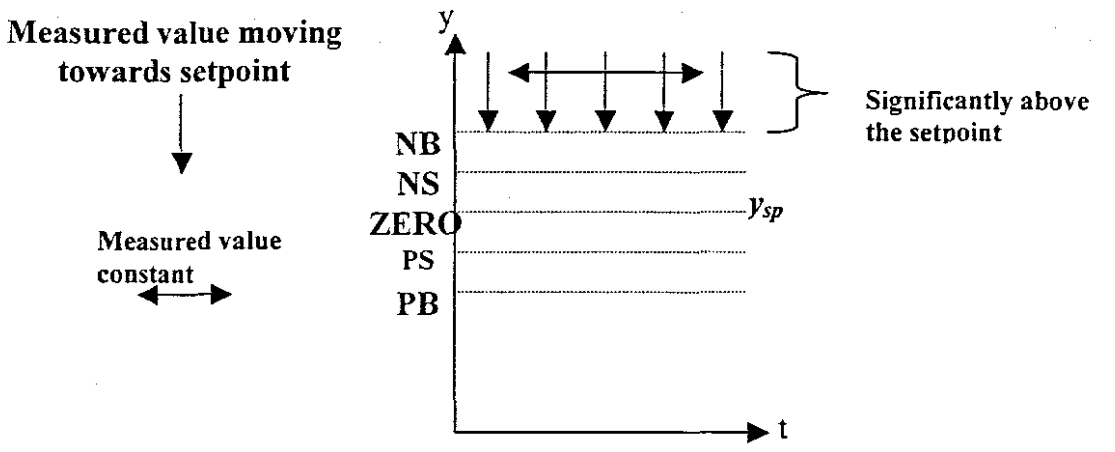


Figure. 3.3: Region significantly above setpoint

The next region to consider is when the measured value is significantly above the desired value and the error change is either constant or causing the process output to be moving towards the setpoint. Here it is considered where error is NB and the error change falls within the ZERO,PS or PB terms. Hence the amount of change to be added to the output

must either speed up or slow down the approach to the setpoint. Table 3.5 is possible control rules for this region.

	IF		THEN
	Error	**Error change**	**Actuator change**
1	Negative Big	Zero	Positive Big
2	Negative Big	Positive Small	Positive Small
3	Negative Big	Positive Big	Zero

Table 3.5: Table showing possible rules for the region significantly above the setpoint

(c) Region close below the setpoint and process output rising AND above the setpoint and level rising

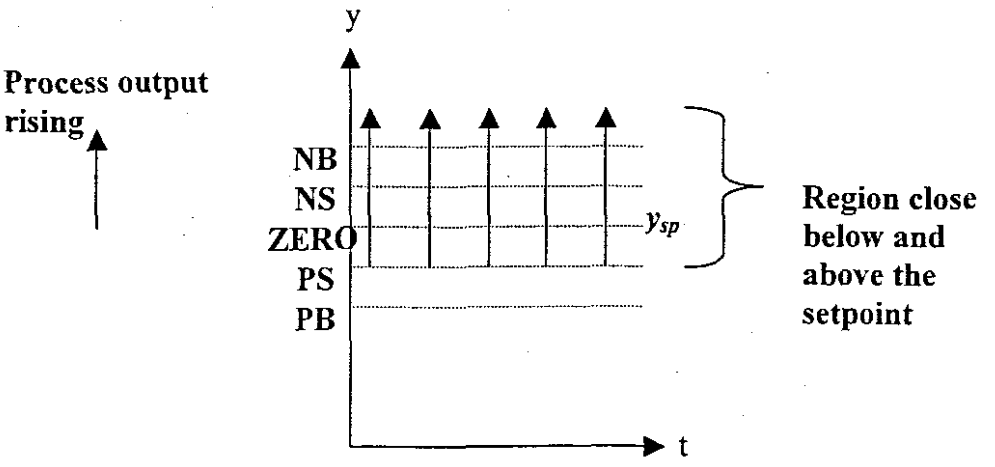


Figure.3.4: Region close and above the setpoint

Now consider when the process output is close below the setpoint and rising as well as when it is above the setpoint and rising. Here the error change will be NB and NS and the linguistic terms for error will be NB,NS,ZERO,PS. Because the process output is tending to move above the setpoint the change to be added to the output should reverse the

direction in which the output is moving. Therefore possible rules will be as indicated in table 3.6.

	IF		THEN
	Error	**Error change**	**Actuator change**
1	Negative Big	Negative Big	Positive Big
2	Negative Big	Negative Small	Positive Big
3	Negative Small	Negative Big	Positive Big
4	Zero	Negative Big	Positive Big
5	Positive Small	Negative Big	Positive Small

Table 3.6: Table showing possible rules for the region close below and above the setpoint with the process output rising

(d) Region close above the setpoint and process output decreasing AND below the setpoint and process output decreasing

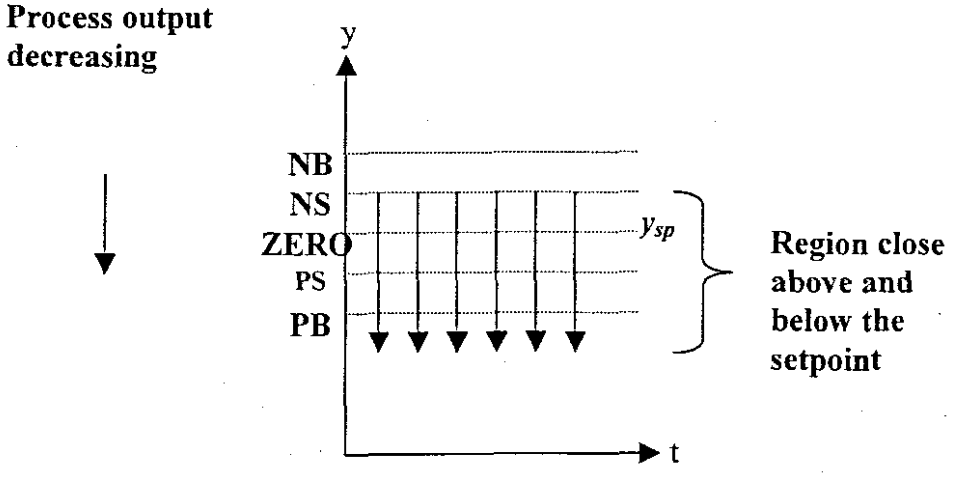


Figure.3.5: Region close above and below setpoint with decreasing process output

The next region of interest has a behavior that is opposite to the one above. Here the process output is close above the setpoint and decreasing or below the setpoint and decreasing.

The error change could be PS or PB and the error could be NS,ZERO,PS or PB. In this region the process output is falling below the setpoint value and requires an addition of a change value that will steer it towards the setpoint. Hence possible rules could take on the following format indicated in table 3.7.

	IF		THEN
	Error	**Error change**	**Actuator change**
1	Negative Small	Positive Big	Negative Small
2	Zero	Positive Big	Negative Big
3	Positive Small	Positive Big	Negative Big
4	Positive Big	Positive Big	Negative Big
5	Positive Big	Positive Small	Negative Big

Table 3.7: Table showing possible rules for the region close below and above the setpoint with the process output decreasing

(e)Region significantly below the setpoint.

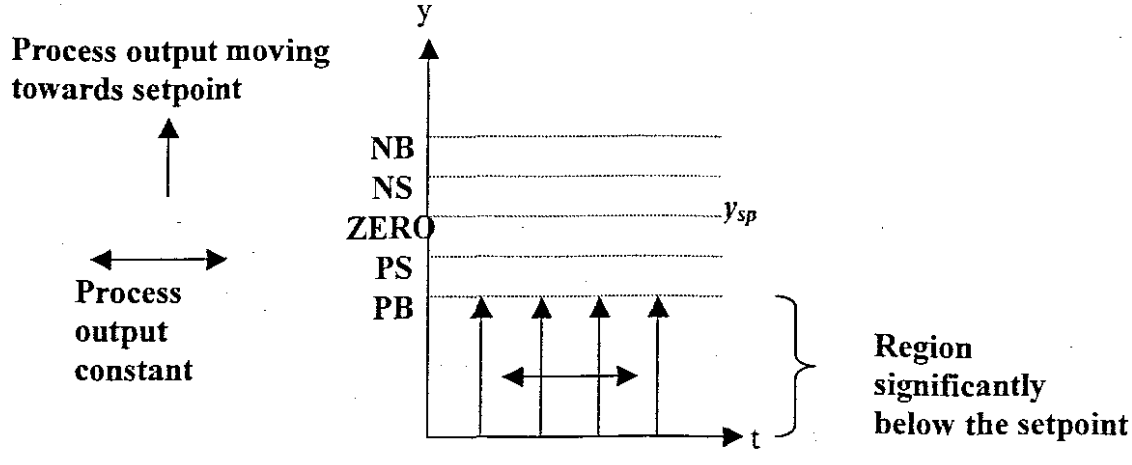


Figure.3.6: Region significantly below the setpoint

The last region to consider is when the process output is significantly below the setpoint and the error change is either constant or causing the process output to be moving towards the setpoint. Here we consider where error is PB and the error change falls within the ZERO,NS or NB terms. Hence the amount of change to be added to the output must either speed up or slow down the approach to the setpoint. Table 3.8 indicates possible are control rules for this region.

	IF		THEN
	Error	**Error change**	**Actuator change**
1	Positive Big	Negative Big	Zero
2	Positive Big	Negative Small	Negative Small
3	Positive Big	Zero	Negative Big

Table 3.8 Table showing possible rules for the significantly below the setpoint with the process output rising or constant

The rules obtained can now be expressed in a shortened tabular form as depicted in table 3.9

ERROR \ ERROR CHANGE	NB	NS	ZERO	PS	PB
NB	PB	PB	PB	PS	Z
NS	PB	PS	PS	Z	NS
ZERO	PB	PS	Z	NS	NB
PS	PS	Z	NS	NS	NB
PB	Z	NS	NB	NB	NB

ACTUATOR CHANGE

Table 3.9: Fuzzy PI rule base in tabular form

3.2. FUZZY COMPUTATIONS

Fuzzy computations can be broken down into three major areas. This is fuzzification, inferencing and defuzzification. To give an idea of the procedure computation of a single rule in a rule base will be explained.

3.2.1 Computation of a Single Fuzzy Rule

Sections 2.2 to 2.5 of chapter 2 have provided the basic theory to realize the implementation of a fuzzy rule. Here a single rule will be stated and the various operations and concepts defined in chapter 2 will be used to show how it can be modeled using fuzzy logic. As an example consider the following crisp values as inputs. Assume that the values for error are –0.4 and for error change –0.8 indicated in figure 3.7 and figure 3.8 respectively.

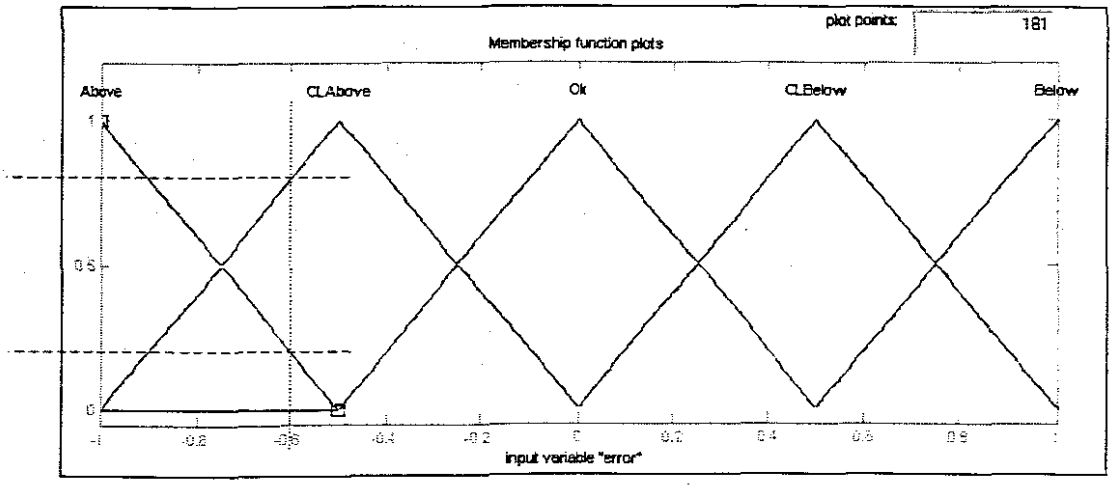


Figure 3.7: Membership functions for input linguistic variable 'error'

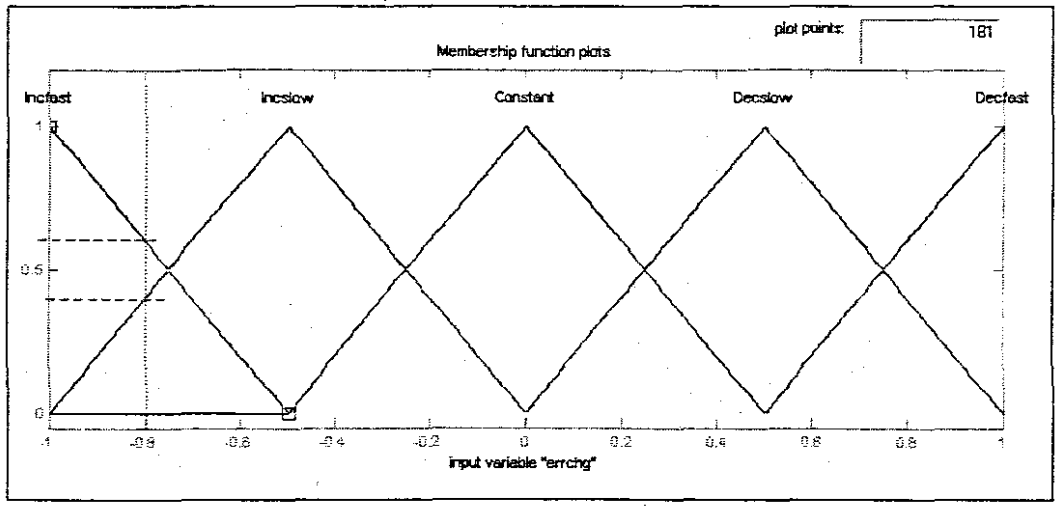


Figure 3.8: Membership functions for input linguistic variable 'error change'

Now based upon the input conditions consider the following rule.

If error is CL Above and error change is Incfast then actuator change is Openfast.

premise consequent

Now for the rule above and all the rules contained in the rule base the truth value for the premise of each rule has to be computed, and applied to the consequent portion of each rule. The truth value is determined by taking the minimum value of the membership functions generated by the fuzzified inputs. Now considering the rule above the membership functions of the fuzzified inputs are as follows. In the fuzzy set labeled 'CLAbove' the degree of membership for –0.4 is approximately 0.75. For the fuzzy set 'Incfast' the value of -0.8 is approximately 0.55. To determine the strength of the premise the min of the two membership functions is determined, in this instance 0.55 and taken to be the premise of the rule. This value is then used to modify the consequent relation. In this case it would be the output fuzzy set labeled 'Openfast'. The modification of the output fuzzy set can be determined by one of two methods. It can be done by applying the 'min' or 'product' operator. When using the 'min' operator for implication the output fuzzy set is clipped Figure 3.9 by the value as specified by the truth value

previously determined. When using the product operator the output fuzzy set is scaled Figure 3.10 by the value as determined by the premise.

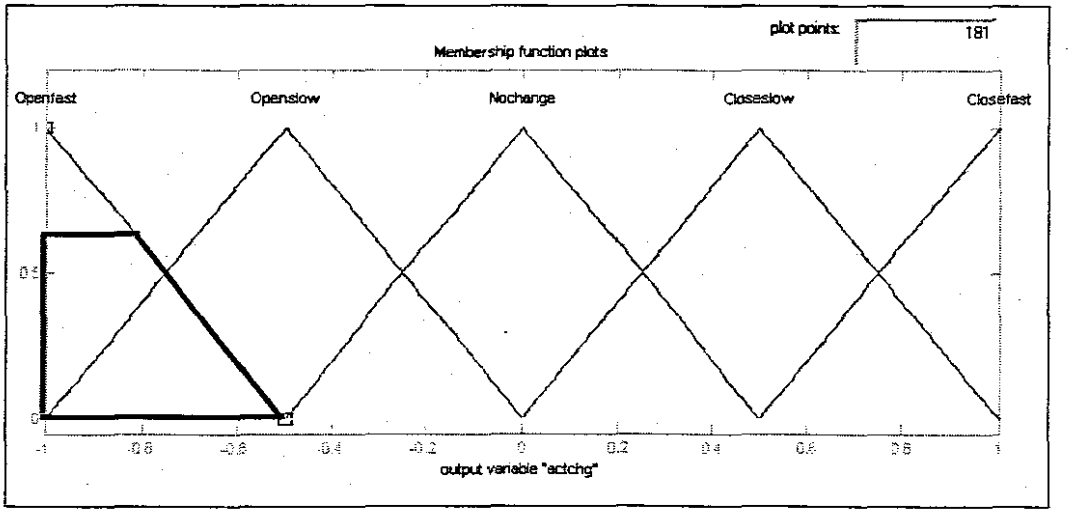


Figure 3.9: Output fuzzy set clipped using 'min' operator.

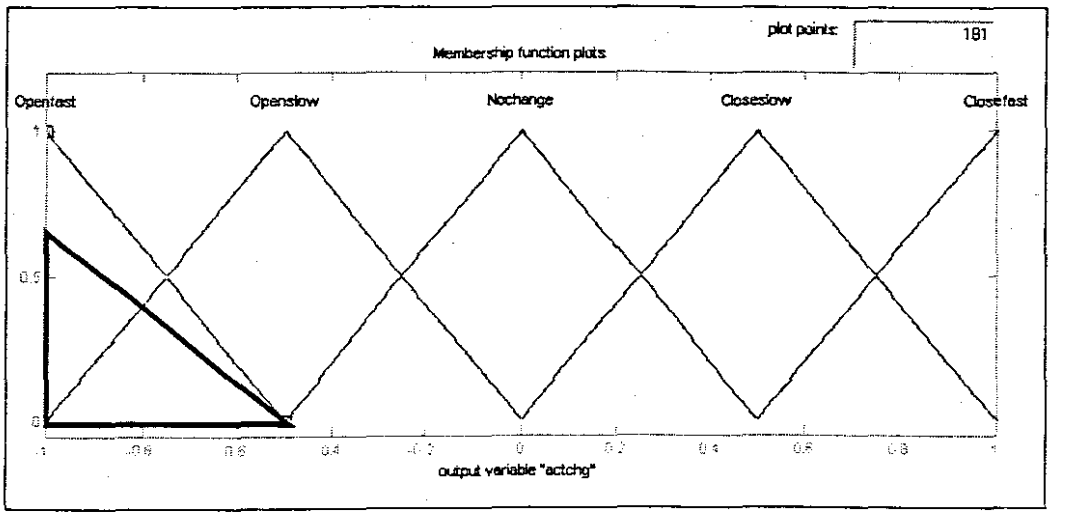


Figure 3.10: Output fuzzy set created using 'product' operator.

The obvious question would now be the choice of method for determining the output. It was stated by Kosko (1992,p.312) that product implication 'preserves more information' than the 'min' implication method. However when one considers simple membership functions this point becomes largely irrelevant because simple membership functions contain little information to start off with in the first place.

The abovementioned procedure is applied for each rule followed by a composition procedure where all the fuzzy subsets as determined by the premises of the rules are combined. Here the 'max' or 'sum' operator is employed. When using the 'max' operator the combined output fuzzy set is constructed by taking the pointwise maximum over all of the fuzzy subsets assigned to the variable by the inference rule (Fuzzy logic 'or'). In 'sum' composition, the combined output fuzzy subset is constructed by taking the pointwise sum over all of the fuzzy subsets assigned to the output variable by the inference rule.

3.2.2 Rule base completeness

The control strategy is determined by the rule base, which essentially is made up from the individual rules. It is thus important that the rule base has a rule for every possible situation and hence the term completeness has been coined for rule bases that possess this quality. To ensure completeness in a rulebase the following should be observed. For the fuzzy sets of the input linguistic variables there should exist at least one rule for each possible combination of the input sets. This will ensure that at least one rule will always be fired.

3.2.3 Defuzzification

Defuzzification involves finding a crisp value that bests represents the information contained in the output fuzzy sets. This is required, as it must be transformed into a more usable form for purposes of control. The centroidal defuzzificaton also known as the center of gravity technique finds the balance point of the previously determined fuzzy region in the output by calculating the weighted mean of the fuzzy region. Arithmetically this can be stated as follows:

$$\Re \leftarrow \frac{\sum_{i=0}^{n} d_i \mu_F(d_i)}{\sum_{i=0}^{n} \mu_F(d_i)} \tag{3.6}$$

Here the fuzzy solution region is called '*F*', where *d* is the *ith* domain value and $\mu(d)$ is the truth membership value for the domain point. Hence the centroid defuzzification method finds a point representing center of gravity of the output fuzzy solution set.

3.3 FUZZY LOGIC SOFTWARE FOR CONTROLLER DESIGN

Fuzzy logic development software has evolved extensively since the inception of fuzzy control, with the market being flooded with various software packages to assist with the development of fuzzy controllers. In this theses the controllers is developed with the assistance of "fuzzyTECH®", a software package developed by the INFORM SOFTWARE CORPORATION. When developing with this package the designer's steps are mostly graphically supported during the initial design stages of the controller. When developing a controller, the fuzzy logic computations are automatically generated by the software. The user however has to make a careful choice of computations that he would like to have embedded within the system. Once the graphical design has been completed the "C" code is generated. This code however only contains the fuzzy computations and it is still required by the user to integrate the generated code with the pre and post processing code for use with the various data acquisition cards. The initial development is far from completion as the final controller is only derived once tuning is completed.

Below the surface this software makes use of "Fuzzy Technology Language" (FTL), which is a standard that is supported by numerous software and hardware manufacturers. It is also supported by more control hardware manufacturers than any other description language for the design of fuzzy logic systems. FTL has an object-oriented structure and supports various design tools and hardware platforms with a common interface.

An FTL file is basically composed of two elements: objects and slots. All objects within an FTL file consist of an object name and a body region that is enclosed in "{}". Object bodies can also have other objects and bodies defined within them. In an FTL, file slots can be identified by the "=" signs. To the left of the "=" sign is the slot name and to the

right the slot entry. Entries can take the form of a value, string or keyword. As with any programming language the FTL language has certain dedicated and reserved keywords.

3.4 DESIGN METHODOLOGY USING "fuzzyTECH®"

When designing a system there are three basic steps.

(1) Linguistic variable specification.

(2) Defining of inference structure

(3) Fuzzy rule formulation.

It was previously mentioned that the Linguistic variables are " error", "error change " and "actuator change" for the controller input and output variables respectively. ". The first step in the design is to develop a block diagram of the fuzzy system as depicted figure 3.11. using the "project editor of "fuzzyTECH®" Here both the linguistic variables and the inference structure is defined. Definition of the inference structure is given by connecting the input and output variables to the rule base as depicted in figure 3.11.

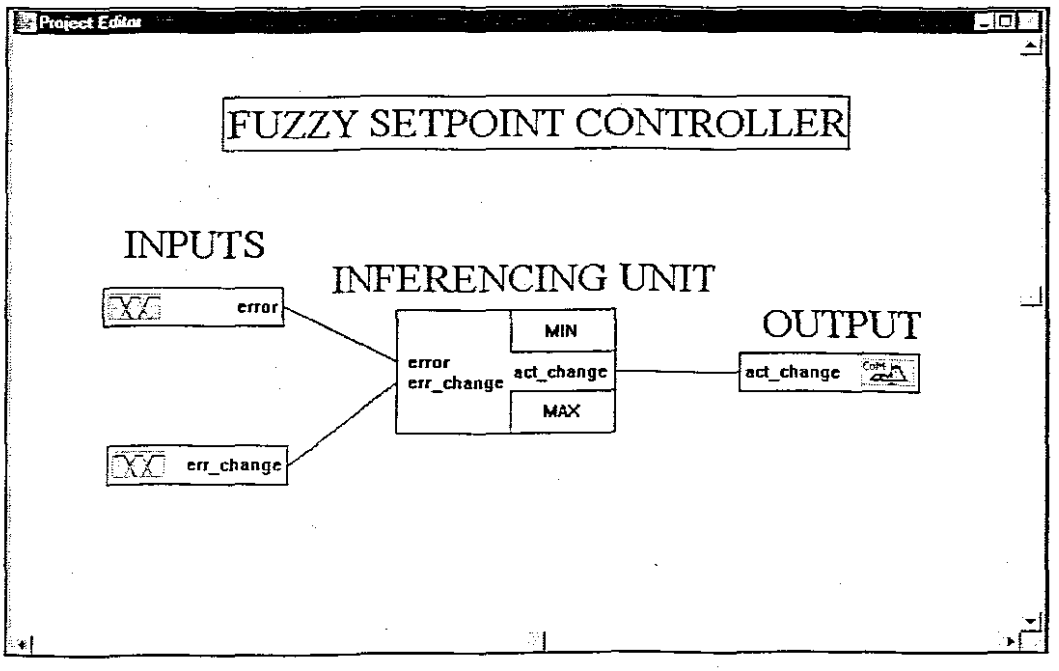


Figure. 3.11: Project editor indicating the block diagram of the fuzzy controller

Following the design procedure as mentioned in section 3.4 the linguistic variables are now further defined in the "LINGUISTIC VARIABLE EDITOR". From table 3.2 each

variable has five linguistic terms and has been defined below as depicted in figure 3.12,figure 3.13 and figure 3.14.

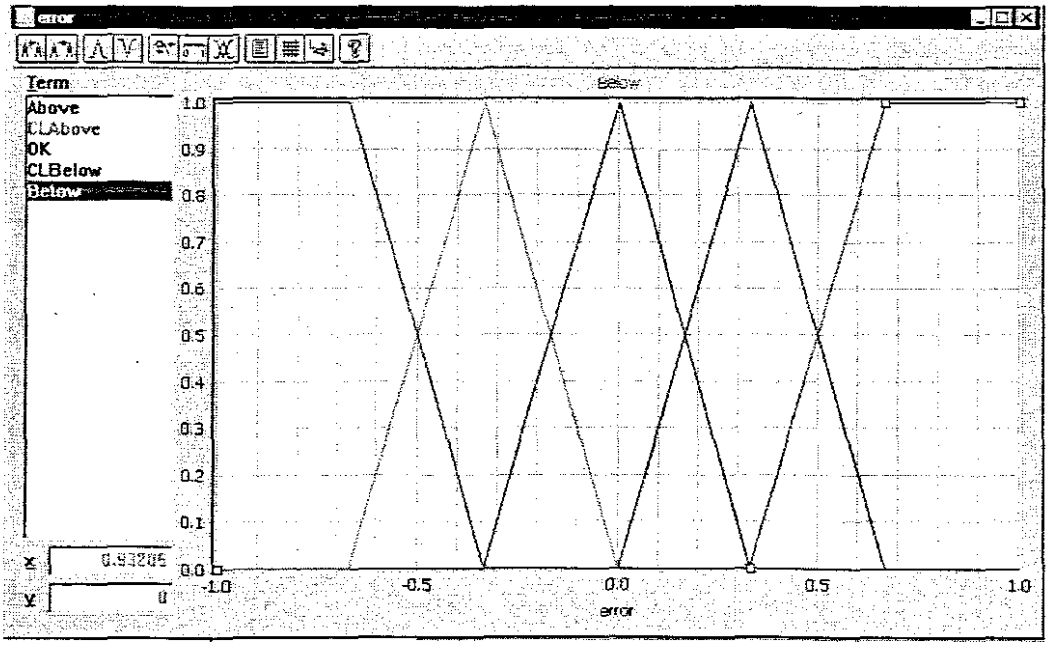


Figure 3.12: Linguistic variable editor depicting the linguistic variable " error"

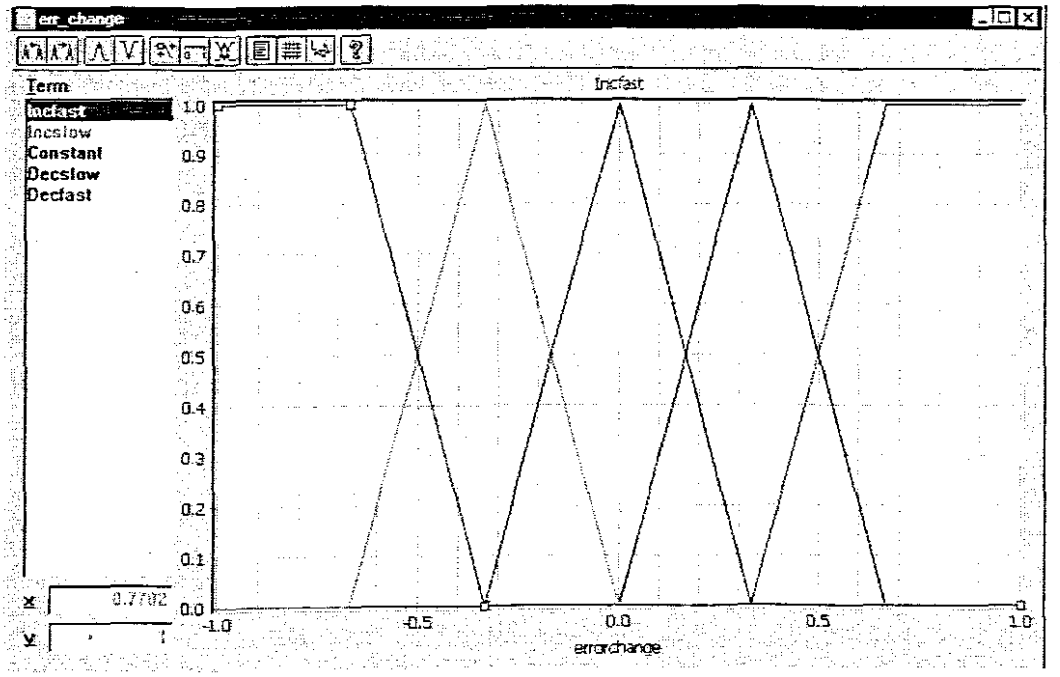


Figure 3.13: Linguistic variable editor depicting the linguistic variable "error change"

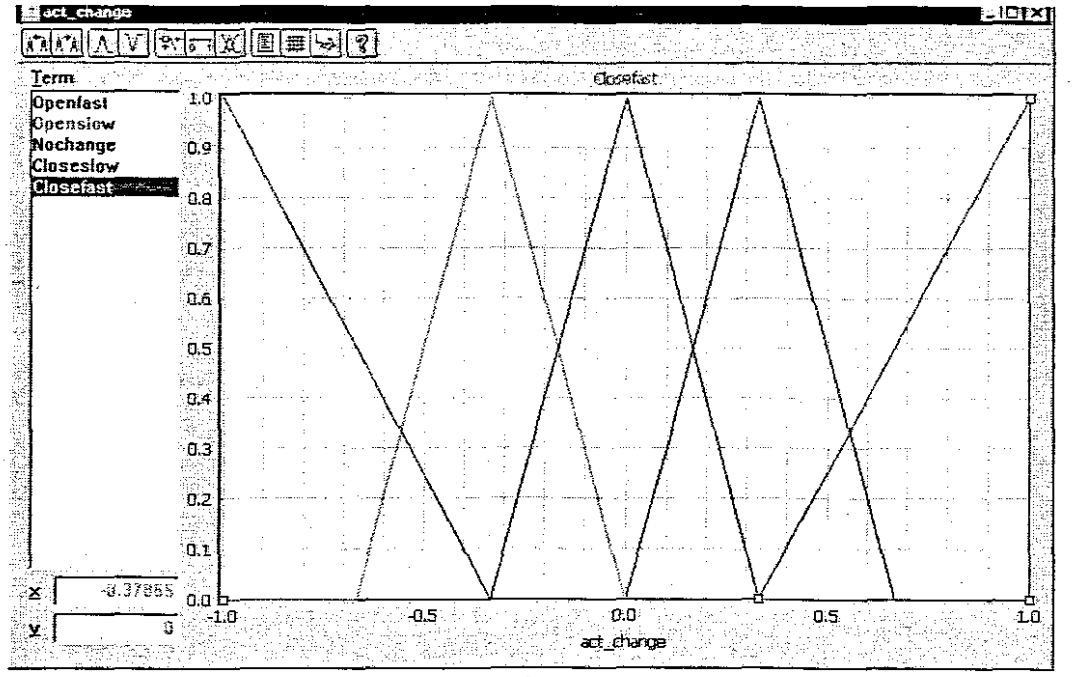
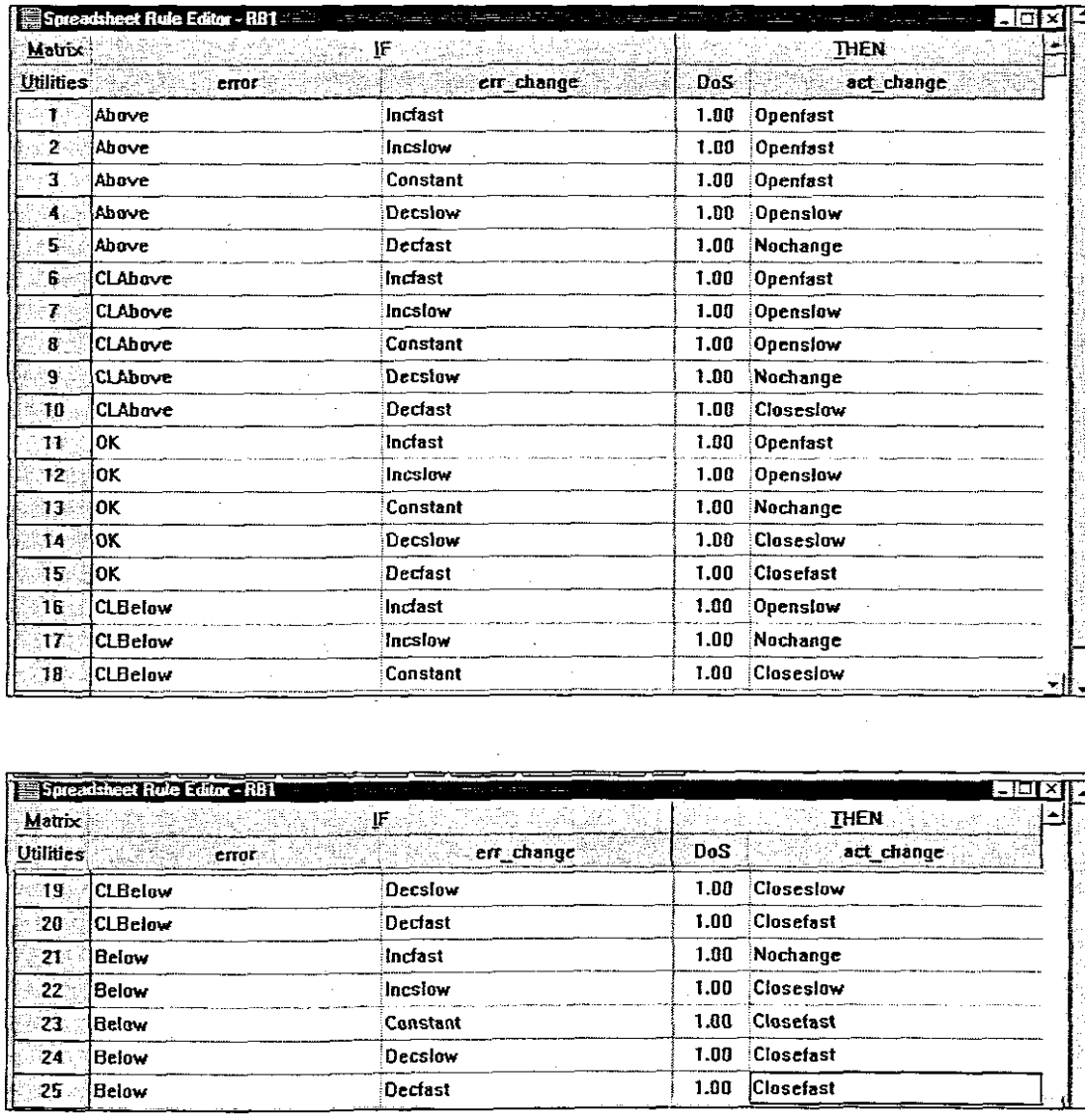


Figure 3.14: Linguistic variable editor depicting the linguistic variable "actuator_change"

The last phase of design is the definition of the rules. During the development of the project in the project editor the rule base is automatically formulated with all possible rules. In this instance it formulated 125 rules. However, only 25 rules are utilized, the rest regarded as being redundant. The necessary rules are obtained from table 3.9 above. After rule editing the following rule base was obtained where DoS is the degree of support.



Spreadsheet Rule Editor - RB1

Matrix	IF		THEN	
Utilities	error	err_change	DoS	act_change
1	Above	Incfast	1.00	Openfast
2	Above	Incslow	1.00	Openfast
3	Above	Constant	1.00	Openfast
4	Above	Decslow	1.00	Openslow
5	Above	Decfast	1.00	Nochange
6	CLAbove	Incfast	1.00	Openfast
7	CLAbove	Incslow	1.00	Openslow
8	CLAbove	Constant	1.00	Openslow
9	CLAbove	Decslow	1.00	Nochange
10	CLAbove	Decfast	1.00	Closeslow
11	OK	Incfast	1.00	Openfast
12	OK	Incslow	1.00	Openslow
13	OK	Constant	1.00	Nochange
14	OK	Decslow	1.00	Closeslow
15	OK	Decfast	1.00	Closefast
16	CLBelow	Incfast	1.00	Openslow
17	CLBelow	Incslow	1.00	Nochange
18	CLBelow	Constant	1.00	Closeslow

Spreadsheet Rule Editor - RB1

Matrix	IF		THEN	
Utilities	error	err_change	DoS	act_change
19	CLBelow	Decslow	1.00	Closeslow
20	CLBelow	Decfast	1.00	Closefast
21	Below	Incfast	1.00	Nochange
22	Below	Incslow	1.00	Closeslow
23	Below	Constant	1.00	Closefast
24	Below	Decslow	1.00	Closefast
25	Below	Decfast	1.00	Closefast

Figure 3.15: Listing of rule base of the fuzzy PI controller

3.4.1 Fuzzy Control Algorithm Generation Using "fuzzyTECH®" for realtime implementation

The preceding section described the design of the fuzzy controller within the "fuzzyTECH®" environment. The next procedure is to develop the actual software code to realize the real-time implementation of the fuzzy controller.

Below follows the procedure that should be performed to ensure that the fuzzy logic software is correctly setup before useful "C" code can be generated for the fuzzy controller.

(1) In the software directory of the "fuzzyTECH®" software find a sub-directory called "LIB". This folder contains files of "C" code, which has to be integrated with the project designed in the Fuzzy Tech environment. These files are indicated below.

HFLMS.C

HCOM.C

HMOM.C

HMIN.C

HMAX.C

HMINPROD.C

HMAXPROD.C

HPUBVARS.C

These files must be compiled to object files. The following command line can be used.

TCC –c –DFTLIBC16 H*.C

After compiling all the files they must be added to the library called FTC16.LIB.

The command line for this is:

Tlib c:\FTMCU96\LIB\FTC16.LIB +<OBJECT FILE NAME>

(2) The next step is to design the fuzzy logic system in the GUI environment of Fuzzy Tech. Once the system has been designed the "C " code can be generated by selecting the "compile" option in the pull down menu.

The compiler generates two files. An example might be "FUZZY.C" and "FUZZY.H"

(3) The next step is to load the "C" file generated above into any text editor so that the path for the "FTLIBC.H" file can be edited. An example of this is indicated below.

#include "c:\ftmcu96\include\FTLIBC.H"

(4) Now compile "FUZZY.C" to "FUZZY.OBJ"

(5) The main program has the skeletal structure as indicated in Chapter 5 with the fuzzy engines embedded in the programme.

(6) The next step is to develop a "project" in the "C" environment.
This project must contain the following files.
(a) FTC16.LIB
(b) FUZZY.OBJ

(7) The project can now be compiled and linked to form an executable file.

3.5 SUMMARY OF CHAPTER

This chapter has highlighted the procedure for the design and real time implementation of a fuzzy controller. Emphasis was placed upon the formulation of the rules by inspecting different regions around the setpoint. In fuzzy controller design, rule formulation is generally done on an adhoc basis, as there are no real formal methods for rule generation Hence a detailed explanation was given to justify the generation of the rules, which form the knowledge base.

Finally the development of the controller with the "fuzzyTECH®" software package was explained with a step-by-step procedure to develop "C" code. The designed controller is a generic fuzzy setpoint controller that has been implemented to control the setpoint of two different processes. In the subsequent chapters the nature and type of two processes will be explained and how the generic fuzzy controller was implemented to control the specified setpoint in each process.

CHAPTER FOUR

FUZZY SET-POINT CONTROL OF LEVELS IN A HEAT EXCHANGER

The first plant of interest is essentially a heat exchanger that was developed to study the gold extraction process on the mines. This rig was developed by a previous post graduate student (De Waal,1990) and was deemed the best model to mimic this gold extraction process. In reality this rig is a heat exchanger. With reference to fig. 4.1 it can be seen that there are four tanks cascaded together from the top to the bottom.

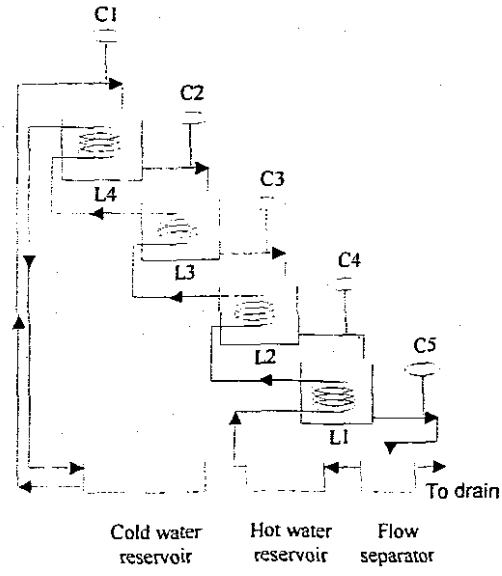


Figure: 4.1: Schematic of the counter current process

At the bottom there is a cold water reservoir from which cold water, normally at room temperature, is pumped to the top tank and flows down to the bottom tank under the influence of gravity.There is a pneumatically controlled valve on the outflow pipe of each tank. These valves can regulate the levels.

A helical wound copper coil through which hot water flows from a hot water reservoir is mounted inside each tank. This hot water is pumped in an opposite direction to the cold water. There is no physical contact between the two streams, however there is thermal contact. This thermal contact results in heat exchange between the two streams. As a result a temperature gradient is developed across the two streams.

The water flows from both the hot and cold reservoirs can be controlled by means of valves. Each tank also has a stirrer, temperature sensors and a level sensor.
A previous study (De Waal, 1990) ascertained that to control the temperature gradient as mentioned above two control loops has to be developed. Level control would be the inner loop and temperature control the outer loop. It was also found that the level response is much faster than the temperature response and thus the water levels can be used to control the temperature gradients. Hence, the inner control loop has to be fast and stable. Although the process is a heat exchanger the study in the dissertation was confined to level control only. The reason for this is twofold.

(1) The inner loop of level control is fundamental, as the setpoints for the levels have to be reached quickly to achieve effective heat transfer between the hot and cold streams.
(2) The emphasis of this study is on the fuzzy methodology for controller synthesis and the level control problem is convenient for fuzzy logic methodology.

4.1 PLANT DYNAMICS

Because this study is confined to level control we will only consider the level dynamics of this plant. To gain a better understanding of the level dynamics of the plant a technique known as the 'binary interaction matrix' was utilised.

4.1.1 Binary Interaction Matrix

As the name implies it is a matrix that indicates the relationship between all the inputs and outputs. This is indicated by either a "1" or "0" entry where a "1" is indicative of an interaction between the corresponding input and output whereas a zero entry indicates that the particular input variable does not influence the output variable at all. This matrix is merely an indication of whether there is an interaction between an input and output variable, it does not specify how or what effect the input variable has on that particular output variable. The appropriate entry can be easily obtained from a schematic of the process. This technique is useful in aiding and facilitating the intuitive approach to fuzzy control. Hence referring to fig. 4.1 the following matrix was obtained for the heat exchanger.

M1	C1	C2	C3	C4	C5
L1	1	1	1	1	1
L2	1	1	1	1	0
L3	1	1	1	0	0
L4	1	1	0	0	0

Table 4.1: Binary Interaction Matrix (BIM)for all valves and levels in the tanks

where

Ci, $i = 1,...5$ are valves

Li, $i = 1,..4$ are levels in the respective tanks.

M1, is matrix 1.

From the entries in the matrix it is evident that controlling the level of each tank and ultimately the whole process falls into the category of being complex in the sense that the levels in each tank are dependant on the both the upper and lower valves. This dependency and compensation thereof will be discussed in section 4.4.

As mentioned previously the reservoir situated at the bottom of the rig is the source of the cold water. It is pumped into the upper tank and flows under the influence of gravity into the tanks below. From fig.4.1 it can be seen that there is a valve that controls the water flow. This valve, which is C1 is set at a constant value that ensures that the tanks will neither overflow or run dry. Because C1 is set to a constant value, it further simplifies the (BIM) above. The simplified matrix is given in table 4.2.

M2	C2	C3	C4	C5
L1	1	1	1	1
L2	1	1	1	0
L3	1	1	0	0
L4	1	0	0	0

Table 4.2: Reduced BIM with the exclusion of C1.

4.1.2 Input and Output Variables for the Controller

From table 4.2 the choice of input and output variables can be obtained. The controller will accept variables associated with the levels of the tanks as its inputs and the controller outputs will effect changes in valves C2 to C5. For the closed loop control system the levels are controlled outputs. The valve changes are the controller actions.

4.1.3 Quantitative Data

Previous work done has ascertained that the level time responses are linear and transfer functions in the s-domain have been formulated. According to previous studies(De Waal,1990) the model for level control is as depicted below.

$$\begin{bmatrix} L_1(s) \\ L_2(s) \\ L_3(s) \\ L_4(s) \end{bmatrix} = \begin{bmatrix} G_{L11}(s) & G_{L12}(s) & 0 & 0 \\ 0 & G_{L22}(s) & G_{L23}(s) & 0 \\ 0 & 0 & G_{L33}(s) & G_{L34}(s) \\ 0 & 0 & 0 & G_{L44}(s) \end{bmatrix} \begin{bmatrix} C_5(s) \\ C_4(s) \\ C_3(s) \\ C_2(s) \end{bmatrix} \tag{4.1}$$

Table 4.3: Model for level control

where G_{Lii} are transfer functions for every term and G_{Lij} are transfer functions representing the influence of the *j-th* term over *i-th.*

Open loop step tests (De Waal,1990) also reveal that the level time responses in the tanks have integral characteristics and are depicted in the matrix below.Their transfer functions are represented as

$$\begin{bmatrix} L_1(s) \\ L_2(s) \\ L_3(s) \\ L_4(s) \end{bmatrix} = \begin{bmatrix} \frac{0.009}{s} & \frac{-0.01}{s} & 0 & 0 \\ 0 & \frac{0.0105}{s} & \frac{-0.01}{s} & 0 \\ 0 & 0 & \frac{0.0105}{s} & \frac{-0.01}{s} \\ 0 & 0 & 0 & \frac{0.012}{s} \end{bmatrix} \begin{bmatrix} C_5(s) \\ C_4(s) \\ C_3(s) \\ C_2(s) \end{bmatrix} \tag{4.2}$$

Table 4.4: Model for level control depicting the transfer functions for each tank

4.2 OPERATING CONDITIONS

With the data accumulated the operating conditions with reference to the effects the valves have on the respective levels of the tanks can be discussed.

4.2.1 Influence Of The Valves

From Table 4.1 and Table 4.2 it can be deduced that the changes in the valve positions are responsible for the level responses in the tanks. It is necessary to further investigate the effect each valve has on all the levels. From the entries it can be observed that the valves of the upper tanks affect the levels of the lower tanks as well. However, to understand how the levels are affected, reference has to be made to Figure.4.1 as well. The following tables detail to what effect each valve has on the corresponding levels.

	Effect of C2
L4	Changes level by varying outflow rate
L3	Changes level by varying inflow rate
L2	Change in L3 effects L2
L1	Change in L2 effects L1

Table 4.5: Influence of C2 on the levels

	Effect of C3
L4	No effect
L3	Changes level by varying outflow rate
L2	Changes level by varying inflow rate
L1	Change in L2 effects L1

Table 4.6: Influence of C3 on the levels

	Effect of C4
L4	No effect
L3	No effect
L2	Changes level by varying outflow rate
L1	Changes level by varying inflow rate

Table 4.7: Influence of C4 on the levels

	Effect of C5
L4	No effect
L3	No effect
L2	No effect
L1	Changes level by varying outflow rate

Table 4.8: Influence of C5 on the levels

To further simplify the BIM table 4.5 to table 4.8 can be used. Here the valves that have a direct effect on the respective levels, i.e. valves that effect the inflow or outflow rate are considered. Hence the matrix can be reduced to the following.

	C2	C3	C4	C5
L4	1	0	0	0
L3	1	1	0	0
L2	0	1	1	0
L1	0	0	1	1

Table 4.9: Direct effect of all valves

4.2.2 Desired Control Structure

With reference to Figure.4.1 and Table 4.9 the ideal situation will be to control the level in each tank with the valve on the output of each tank only and to compensate for the influence of the valve controlling the inflow into each tank. With compensation the ideal matrix should take on the following format indicated in Table 4.10.

	C2	C3	C4	C5
L4	1	0	0	0
L3	0	1	0	0
L2	0	0	1	0
L1	0	0	0	1

Table 4.10: Fully decentralised BIM

Table 4.10 is representative of the ideal matrix that should be used for the closed loop control but the design of the regulator has to take the matrix represented by Table 4.9 into account. The rationale behind this is as follows. A regulator designed on the basis of Table 4.9 will be centralised or hierarchical ie.

$$\begin{aligned} &U_i(k) = H_i \overline{e_i(k)} + \overline{H_{ij}} L_i(k), \quad i = \overline{1,4} \quad j = \overline{1,4} \quad i \neq j \quad k = \overline{0,\infty} \\ &e_i(k) = L_{sp_i} - L_i(k) \end{aligned} \tag{4.3}$$

where U_i is the control action, H_i is the controller coefficient for the *ith* tank, and H_{ij} is the controller coefficient compensating for the influence of the *jth* tank level over the *ith* tank level. Also Lsp_i is the level setpoint, k is the discrete time and e_i is the level error.

A regulator designed on the basis of Table 4.10 will be decentralised i.e.

$$U_i(k) = H_i . e_i(k) \quad i = 1,4 \tag{4.4}$$

where the coefficient H_i is designed in such a way that the influence of the *jth* tank level over the *ith* is compensated.

A decentralised approach is opted for because the dimensionality of the rules of the fuzzy controller will be reduced and control of the levels in each tank will be dependent upon one actuator only. This implies that the complexity of level control for a single tank is reduced.

4.3 IMPLEMENTATION OF FUZZY CONTROLLER

The control structure is fully decentralised. Four independent controllers were designed to control the level in each tank. The rationale behind this is to provide a local solution to be later combined to form a global solution of level control for the entire system. The decentralised controller only by local control actions has to compensate for all interactions between the tanks.

4.4 COMPENSATION BETWEEN TANKS

It was observed that the water level in each tank was affected by both the upper and lower valves situated on each tank. Because it was decided to use the lower valve to control the level in each tank, a compensation technique had to be employed to compensate for the effect of the upper valve situated on each tank. Upon inspection of the rule base of the proposed fuzzy controller it can be seen that compensation is included in the design of the controller. This compensation is taken care of by the linguistic variable, "error change". Here error change will represent level error change. The compensation effect of this variable can be explained by considering the following table and diagram.

C1	C2	LEVEL
0	0	constant
0	1	Level decreases
1	0	Level increases
1	1	constant

Table 4.11: Level response for 'open' and 'close' combinations of C1 and C2

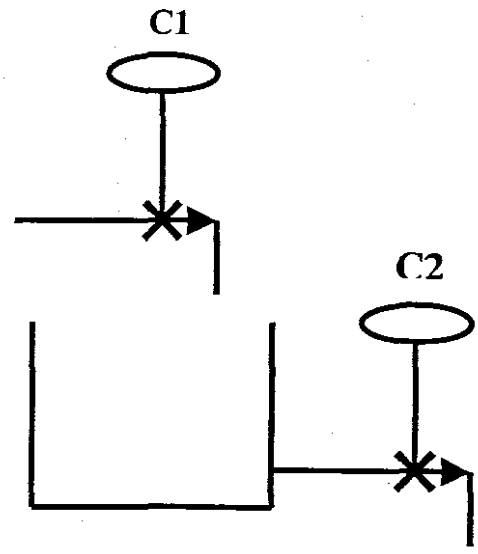


Figure 4.2 :Schematic of a single tank

Here a "1" signifies that the valve is completely open and a "0" completely shut. From the table it can be easily deduced that when both valves are either completely open or shut the level in the tank will remain constant. This is based upon the assumption that the inflow and outflow are equal. If "C1" is fully open and "C2" is completely shut, the level will rise. If "C1" is completely shut and "C2" is fully open, the level will decrease. Now if one considers differnt degrees of "openess" for the upper valve("C1"), the rate at which the level rises is also affected. Thus the inflow rate is dependant upon the degree to which the valve aperture is open and thus it will affect the rate of level change.

4.5 FUZZY CONTROL ALGORITHM GENERATION

As mentioned in chapter three the fuzzy controller was developed with the assistance of a commercial software package called "fuzzyTECH®". The text below describes the development of the controller for level controller for the considered cascaded tank system.

In chapter 3 the design of the fuzzy controller within the "fuzzyTECH®" environment was described. The next procedure develops the actual software code to realise the real-time implementation of the fuzzy controller. The algorithm was coded using the "C" programming language. It is composed of essentially three parts, each performing a specific function. These parts can be classified into the following three sections

(1) Sampling of data

(2) Writing of values to actuators.

(3) Fuzzy algorithm

4.5.1 Sampling of Data

Data sampling was accomplished using an analogue to digital interface card, the DT2801. The code below forms the skeletal structure for data acquisition.

```
/*  ------------------------------------------
    Procedure library for Data Translation Card DT2801
    ------------------------------------------
    Ref: DT2801 manual */

#include <dos.h>

#include <dos.h>

#define base_address 0x2ec    /*DT2801 uses port $2EC & $2ED*/
#define comm_wait 0x4         /*DT2801 commands*/
#define write_wait 0x2
#define read_wait 0x5
#define cstop 0xf
#define cclear 0x1
#define cerror 0x2
#define cadin 0xc
#define cdaout 0x8

#include "adc2801.h"

/*Set DT2801 port addresses*/
int command_reg = base_address + 1;
int data_reg = base_address;


/*-------------------------------------*/
/***------procedure to read from card---------***/
/*Reads the voltage from A/D channel number <chan_no>.
  Result returned in the integer variable <value>
  in the range -2048 to 2047 [Counts] corresponding to
  the voltage input -10,000 to 9,9951 [Volt] */

double adc_read(int chan_no)
        {

        double value;
```

```
        unsigned char high=0, low=0;

        do{
          }   while((inportb(command_reg) & comm_wait) <= 0);
   outportb(command_reg,cadin);
   do {
          }while(((inportb(command_reg) ^ write_wait)&write_wait) <= 0);
   outportb(data_reg, 0);
   do {
          }while(((inportb(command_reg) ^ write_wait)&write_wait) <= 0);
   outportb(data_reg,chan_no);
   do {
          }while ((inportb(command_reg) & read_wait) <= 0);
   low = inportb(data_reg);
   do {
          }while ((inportb(command_reg) & read_wait) <= 0);
   high = inportb(data_reg);

   value = (double)((int)high * (int)256 + (int)low);
         value = value-2048;
        value = value/2047*10;
        return value;
        }

/*-----------------------------------*/
/***------procedure to setup card-------***/
#pragma argsused
void set_up_2801(void)
 {
//       int temp;                              /*Discarded data*/
        outportb(command_reg,cstop);          /*Stop DT2801 card */

        /*      Clear data register*/
/*      temp = */inportb(data_reg);
        do {
              }while((inportb(command_reg) & comm_wait) <= 0);
        /*Clear DT2801 card*/
        outportb(command_reg,cclear);
 }
```

4.5.2 Data Translation to Actuators

Writing to the actuators was accomplished with a separate digital to analogue interface card, the DT8215. The code below forms the skeletal structure for data acquisition.

```
/*  ------------------------------------------
        Procedure library for Data Translation Card DT2815
        ------------------------------------------
        Ref: DT2815 manual */

#include <dos.h>
#include <math.h>

#define  Data_DA 0x224        /* Data register */
#define  Status_DA 0x225      /* Status register */

#include "DAC2815B.H"

/*-------------------------------------*/
/***------procedure to write to card-------***/
double dac_write(int chan_no,double value)
{
 unsigned char hi,lo;

 value = (int)(value*4096/5);
 /* Poll Status reg until it = 0 */
 do {
        } while(inportb(Status_DA) != 0x0);
 /* Keep value in range */
 if(value>4095)
        value = 4095;
 else if(value<0)
 value = 0;
 /*Find high & low byte values*/
 hi = floor(value/16);
 lo = 16*((int)value - 16*(int)hi) + 2*chan_no + 1;
 /* Send low byte to data reg */
 outportb(Data_DA,lo);
 /* Poll status reg until ready for high byte */
 do {
        } while(inportb(Status_DA) != 0x10);
 /* Send high byte to data reg */
 outportb(Data_DA,hi);
 return value;
}


/*-----------------------------------*/
```

```
/***------procedure to setup card------***/
void reset_2815(void)
 {
        /* Set status register to zero */
        outportb(Status_DA,0x0);
        /* Poll until status reg = 4 */
        do { //outportb(Status_DA,0x0);
                } while(inportb(Status_DA) != 4);
        /* Set data reg to 7, selecting program 1 */
        outportb(Data_DA,0x7);
 }

/* NB : 1 is added to the lower byte to be
        written to the data register. This is to
        diasable the firmware offset - by setting
        the mode bit */
```

4.5.3 Fuzzy Algorithm

After the design procedure as highlighted in chapter 3, section 3.4, the "C" code is generated by selecting the "compile" option from the pull down menu. Section 3.4.1 of chapter 3 contains a step by step procedure to generate the executable file for the fuzzy controller. For level control the following files were generated: from "Level_1.C" and "Level_1.H" through to "Level_4.C" and "Level_4.H".

The fuzzy system is called as a "C" function within the main program which in this case is the main control program.

It will be useful to note that the control algorithm contains four fuzzy functions, each one controlling a separate level. Here the fuzzy functions are appropriately named "Level_1","Level_2","Level_3" and "Level_4". These functions are all exported from the their corresponding "C" modules i.e. "Level_1.C", "Level_2.C", "Level_3.C " and "Level_4.C" by including their matching header file as shown in the code below.

The sampled input values and the output values are accessed by the generated fuzzy logic systems as global variables. Here the variable names have the name of the fuzzy logic system appended with an underscore after each variable name. Before the fuzzy logic inference system is called the global variables representing the inputs have to be initialized. All global variables are of type "FUZZY" which is defined in the header

files. Below is the skeletal structure of the control algorithm with only the code of the fuzzy algorithm shown.

```
#include "Level_1.h"        /*
#include "Level_2.h"        /*      Include knowledge base of fuzzy kernels
#include "Level_3.h"        /*
#include "Level_4.h"        /*

#include .....              /*
#include .....              /*      Other "include" files

FLAGS rv1;                  /*
FLAGS rv2;                  /*      Declaration of return variables
FLAGS rv3;                  /*
FLAGS rv4;                  /*

Void main(void) {           /*      Start of main program

.....                       /*
.....                       /*      Other code of control algorithm
.....                       /*

initLevel_1( );             /*
initLevel_2( );             /*      Initialization of fuzzy engines
initLevel_3( );             /*
initLevel_4( );             /*

level1_Level_1=.....;       /*
level2_Level_2=.....;       /*      Setting of input variables for the fuzzy
level3_Level_3=.....;       /*      engine
level4_Level_4=.....;       /*
```

```
change1_Level_1=.....;   /*
change2_Level_2=.....;   /*   Setting of input variables for the fuzzy
change3_Level_3=.....;   /*   engine
change4_Level_4=.....;   /*

.....                    /*
.....                    /*   More program code
.....                    /*

rv1=Level_1( )           /*
rv2=Level_2( )           /*   Calling of the fuzzy engines
rv3=Level_3( )           /*
rv4=Level_4( )           /*

.....= valve1_Level_1;   /*
.....= valve2_Level_2;   /*   Transfer of the output data processed
.....= valve3_Level_3;   /*   by the fuzzy engines to the process
.....= valve4_Level_4;   /*

.....                    /*   More program code
.....                    /*

}                        /*   End of control loop

}                        /*   End of main program
```

4.6 TEST OF THE DESIGNED FUZZY CONTROLLER

Before the tests were performed it was necessary to bring the plant into a steady state condition. To achieve this the following steps were carried out.

The experiments performed essentially is comprised of three parts.

1. Preparation for the test.
2. Providing the test
3. Analysis of the results of the tests.

Table 4.12 below outlines the preparation and testing procedure.

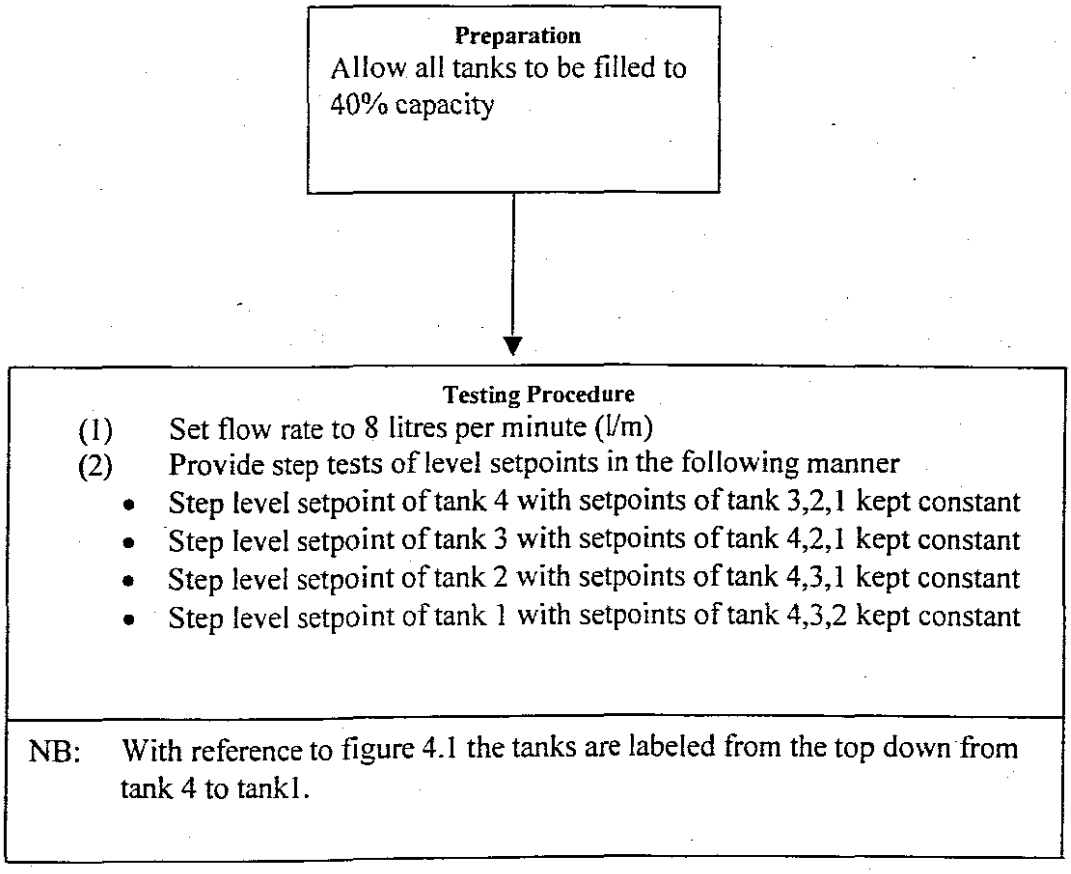


Table 4.12: Block diagram of the testing procedure

4.6.1 Preparation Procedure

Before the tests were conducted all the tanks were allowed to be filled to approximately 40% of its capacity. This was to ensure that the tanks were not too full

or too empty. Essentially this was done to prevent the tanks from overrunning or running the risk of running dry.

4.6.2 Testing Procedure

The testing procedure was then performed in the following manner.The flow rate of the water was set at approximately 8 litres per minute (l/m) Then the level of the uppermost tank was stepped to achieve a level of 80% capacity, to ensure that the tanks do not overflow, while the remaining setpoints of the tanks below were kept constant. The effect this had on the entire process was observed and documented. This procedure was also performed on the remaining three tanks in the system.
The following table shows a summary of the testing procedure.

4.6.3 Analysis of Results

4.6.3.1 Level Setpoint of Tank 4 Stepped With Constant Setpoints of Tanks 3,2,1

Figures 4.3 to Figure 4.6 documents the results obtained when the step test is first initiated when stepping the level setpoint of the uppermost tank (tank 4) while the level setpoints of the other tanks below are kept constant. It can be seen that the fuzzy controllers provide effective control based upon the following observations.

With the flow rate at approximately 8 l/m the controller provides excellent level response with regard to rise time. Figure 4.3 shows that when the level is stepped from 40% to 80% tank capacity the water in the tank increases by approximately 8 liters. Upon inspection of figure 4.3 the setpoint is reached in approximately 60 seconds indicating that the fuzzy controller enables the setpoint to be reached in minimum time.

With regards to overshoot it can be seen that there is hardly any indication of overshoot of the level. This is due to the fact that the process response is much slower than the controller response with the result that the controller has almost a predictive nature.

Figures 4.4 to Figure 4.6 show that the levels in the three lower tanks are hardly effected by the step in level setpoint of the uppermost tank. The fuzzy controllers for these tanks provide good stabilization of their level setpoints.

4.6.3.2 Level Setpoint of Tank 3 Stepped With Constant Setpoints of Tanks 4,2,1

Figure 4.7 to Figure 4.10 documents the response of the levels in the tanks when the setpoint of tank 3 is stepped and the setpoint of the rest of the tanks are kept constant. Note that Figure 4.7 indicates that the level in tank 4 is maintained at the setpoint value for 80% capacity (the result of the previous step test). Tank 3 is stepped to the 80% mark and tank 2 and tank 1 maintained at the 40% mark. Once again it can be seen that there is a minimum rise time with no overshoot for the level in tank3. Also the fuzzy controllers provide good stabilization of setpoint for tanks 4,2,1 as indicated by Figures 4.7,Figure 4.9 and Figure 4.10 respectively.

4.6.3.3 Level Setpoint of Tank 2 Stepped With Constant Setpoints of Tanks 4,3,1

Figure 4.11 to Figure 4.14 documents the response of the levels in the tanks when the setpoint of tank 2 is stepped and the setpoint of the rest of the tanks are kept constant. Note that Ffigure 4.11 and Figure 4.12 indicates that the level in tank 4 and tank 3 respectively is maintained at the setpoint value for 80% capacity (the result of the previous step test). Tank 2 is stepped to the 80% mark and tank 1 maintained at the 40%mark with tank 3 and tank 4 at the 80% mark. The result of the step is minimum risetime with no overshoot of the level in tank 2. The fuzzy controllers provide good stabilization of setpoint for tanks 4,3,1 as indicated by Figures 4.11,Figure 4.12 and Figure 4.14 respectively.

4.6.3.4 Level Setpoint of Tank 1 Stepped With Constant Setpoints of Tanks 4,3,2

Figure 4.15 to Figure 4.18 documents the response of the levels in the tanks when the setpoint of tank 1 is stepped and the setpoint of the rest of the tanks are kept constant. Note that Figure 4.15 and Figure 4.17 indicates that the level in tank 4 , tank 3 and tank 2 respectively is maintained at the setpoint value for 80% capacity (the result of the previous step test).). Tank 1 is stepped to the 80% mark and tank 4,tank3 and tank

2 maintained at the 80% mark. The result of the step is minimum risetime with no overshoot of the level in tank 1. The fuzzy controllers provide good stabilization of setpoint for tanks 4,3,2 as indicated by Figures 4.15,Figure 4.16 and Figure 4.17 respectively.

Based upon the results obtained for the step tests it can be seen that the fuzzy control algorithm has provided satisfactory setpoint control of the levels in the cascaded tank system.

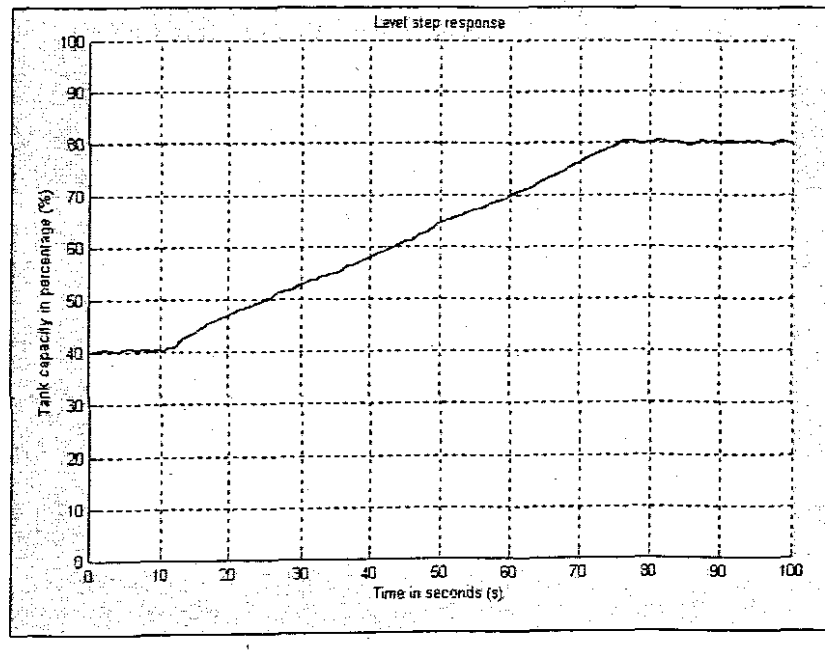


Figure 4.3: step response in tank 4

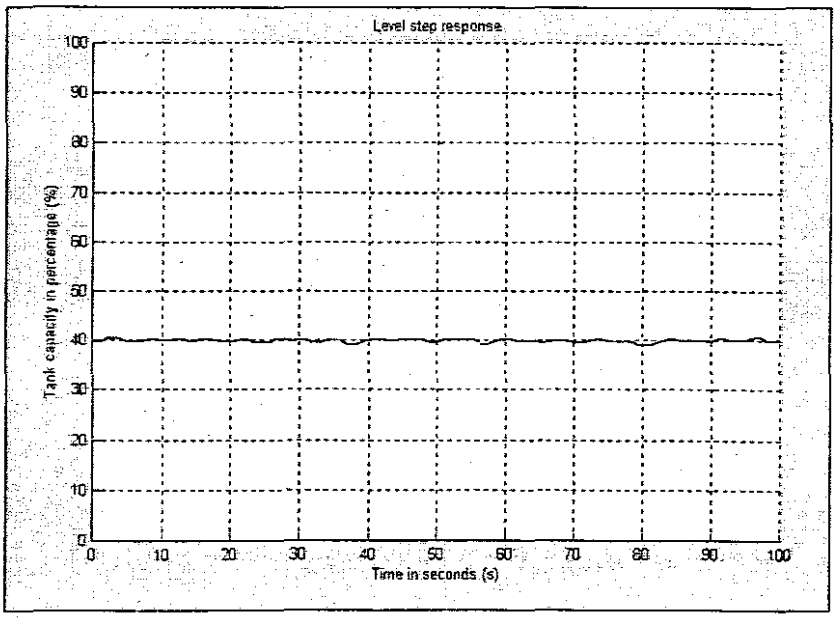


Figure 4.4: level response in tank 3

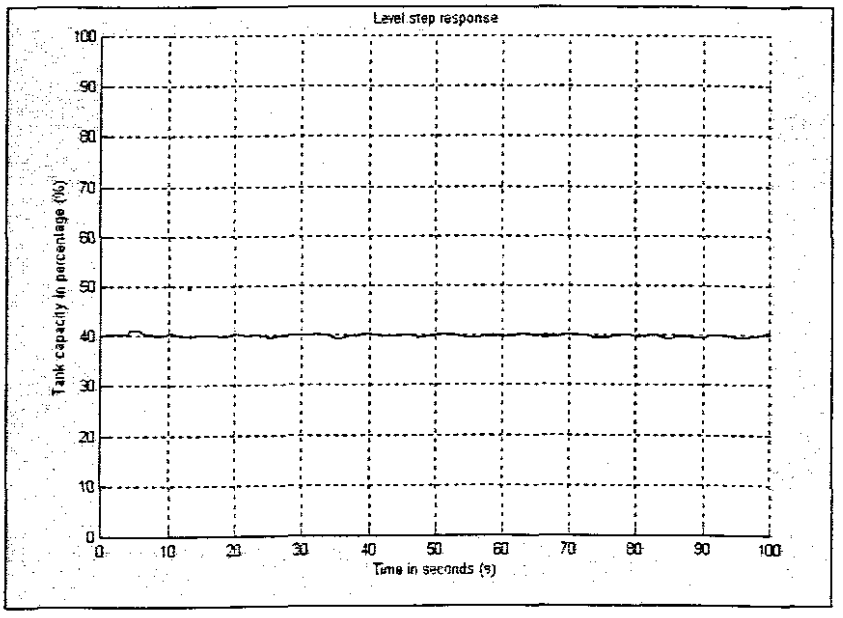


Figure 4.5: level response in tank 2

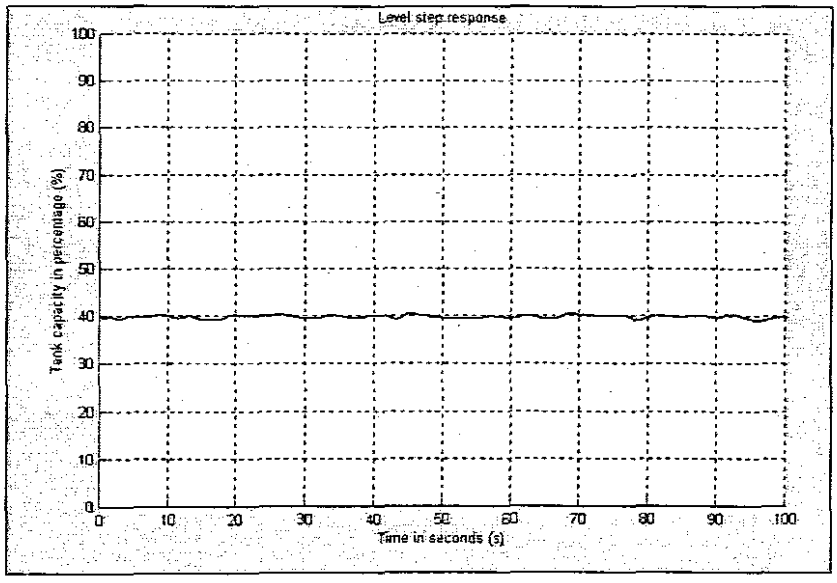


Figure 4.6: level response in tank 1

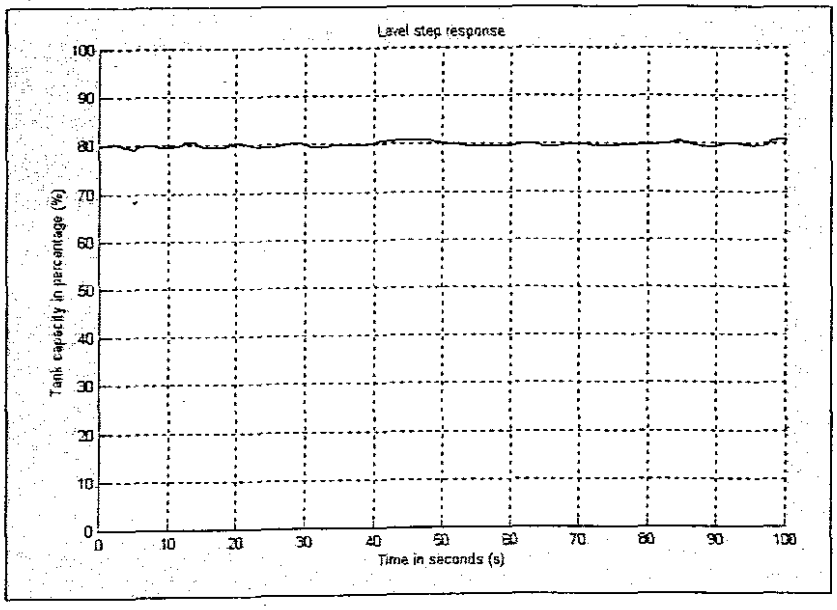


Figure 4.7: level response in tank 4

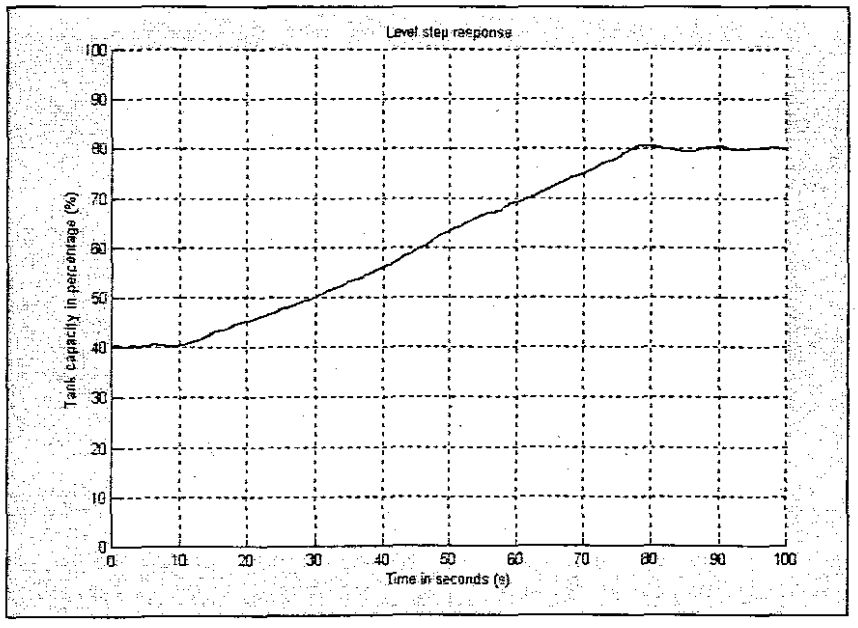


Figure 4.8: step response in tank 3

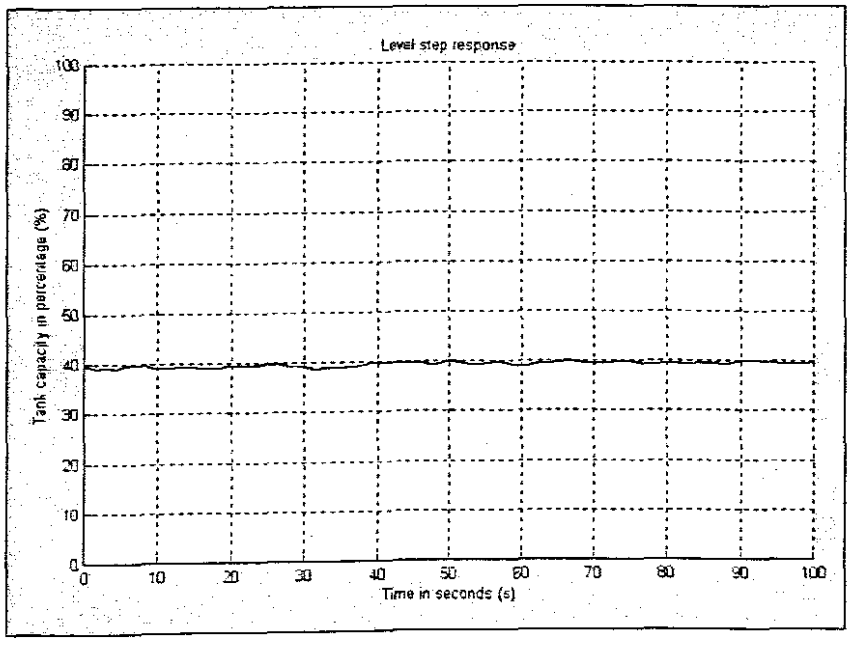


Figure 4.9: level response in tank2

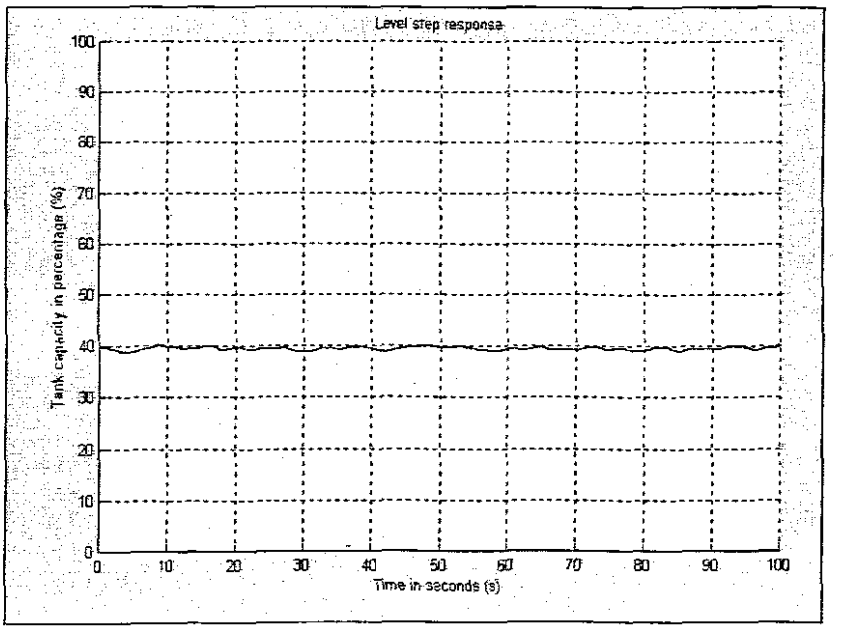


Figure 4.10: level response in tank 1

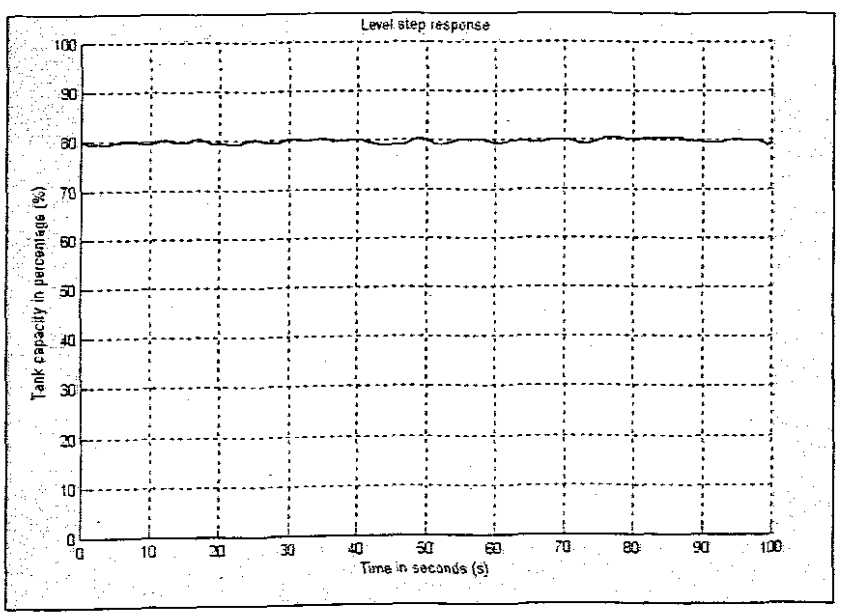


Figure 4.11: level response in tank 4

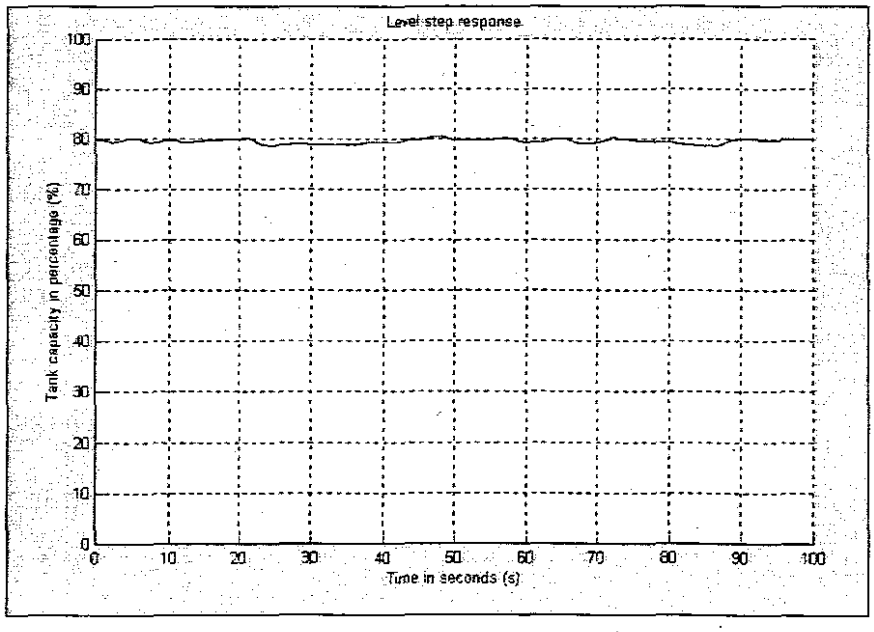


Figure 4.12: level response in tank 3

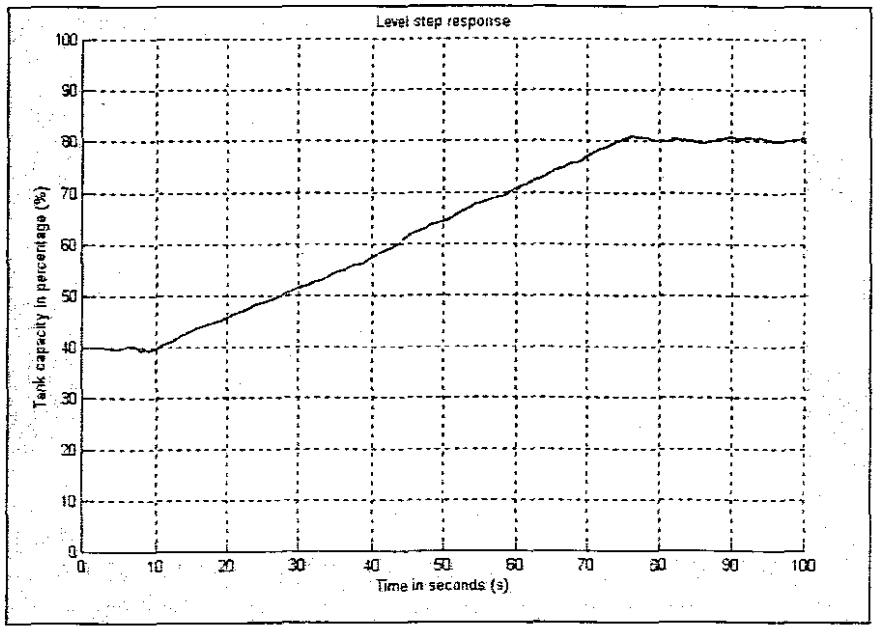


Figure 4.13: step response in tank 2

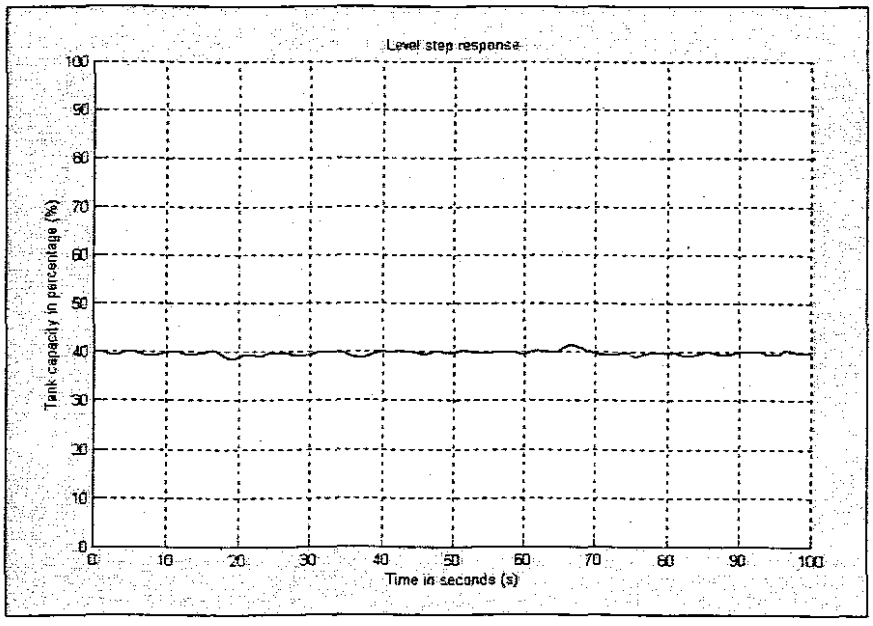


Figure 4.14: level response in tank 1

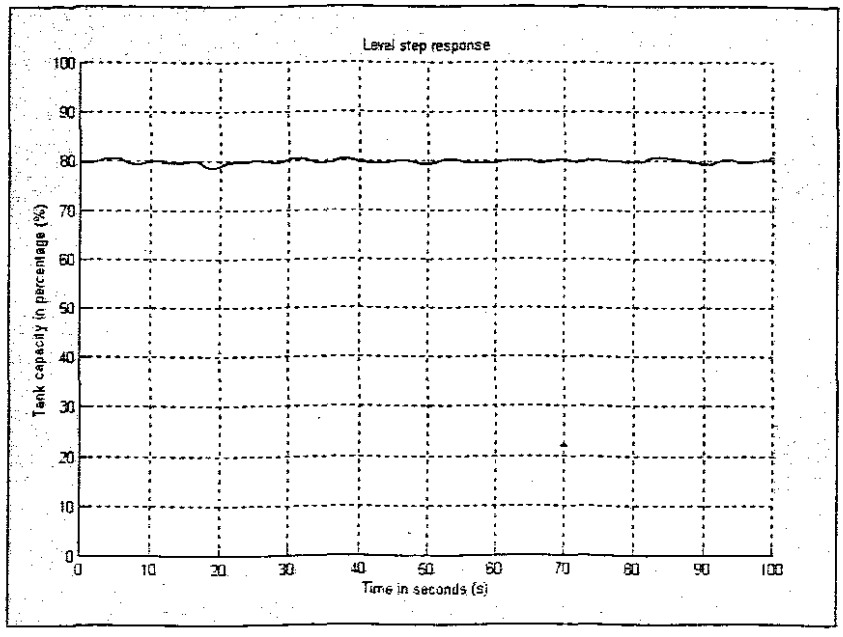


Figure 4.15: level response in tank 4

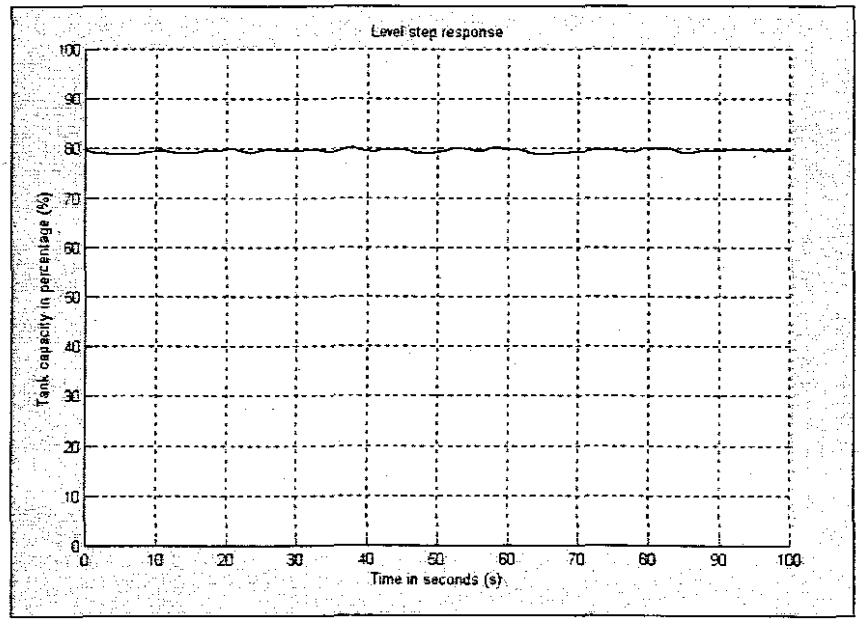


Figure 4.16: level response in tank 3

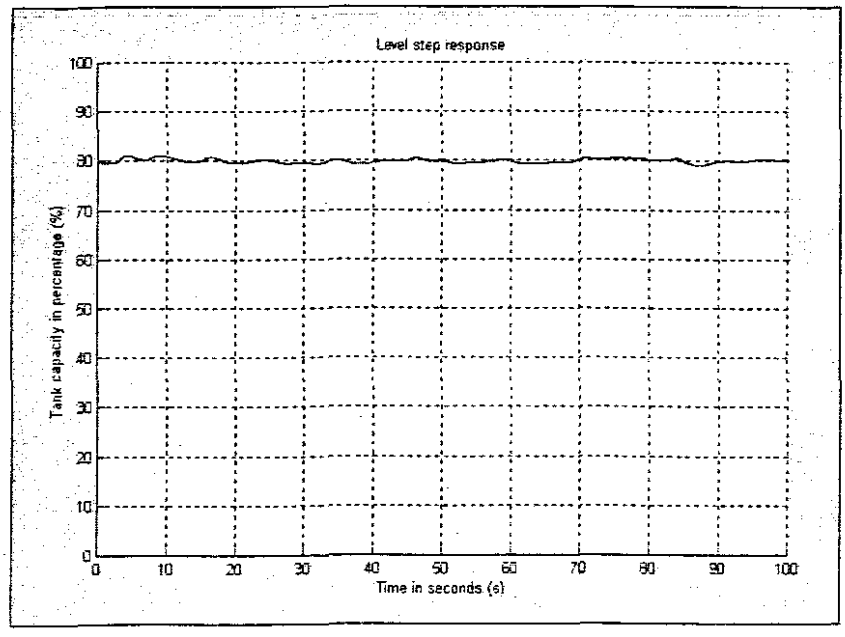


Figure 4.17: level response in tank 2

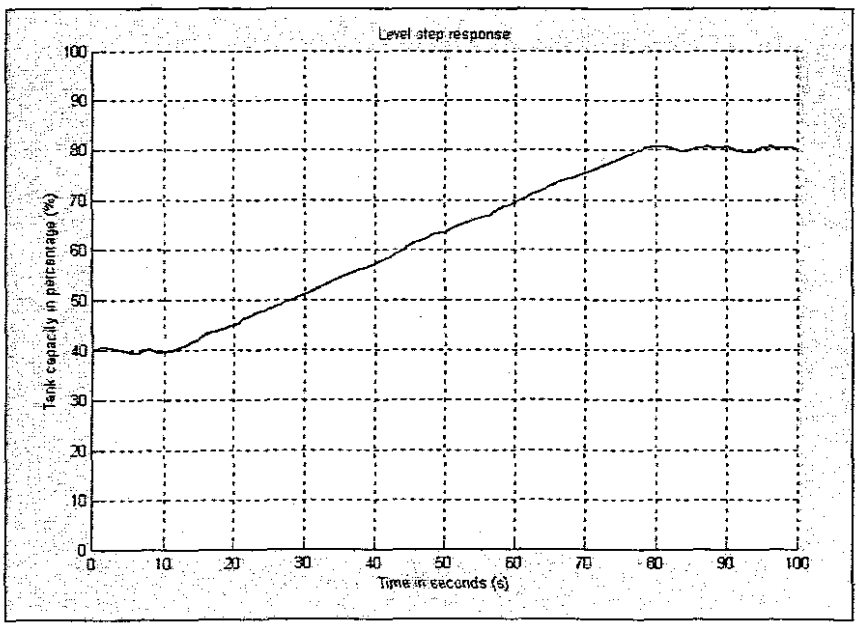


Figure 4.18: step response in tank 1

4.7 SIMULATION IN LABVIEW

A simulation of the cascaded tank system with the implementation of the designed fuzzy controllers to control the levels in the tanks was implemented in using LABVIEW. LABVIEW is a software package that has various toolboxes to assist with the simulation of various engineering applications including control engineering. It is packaged with pre-defined blocks for display panels and simulations.

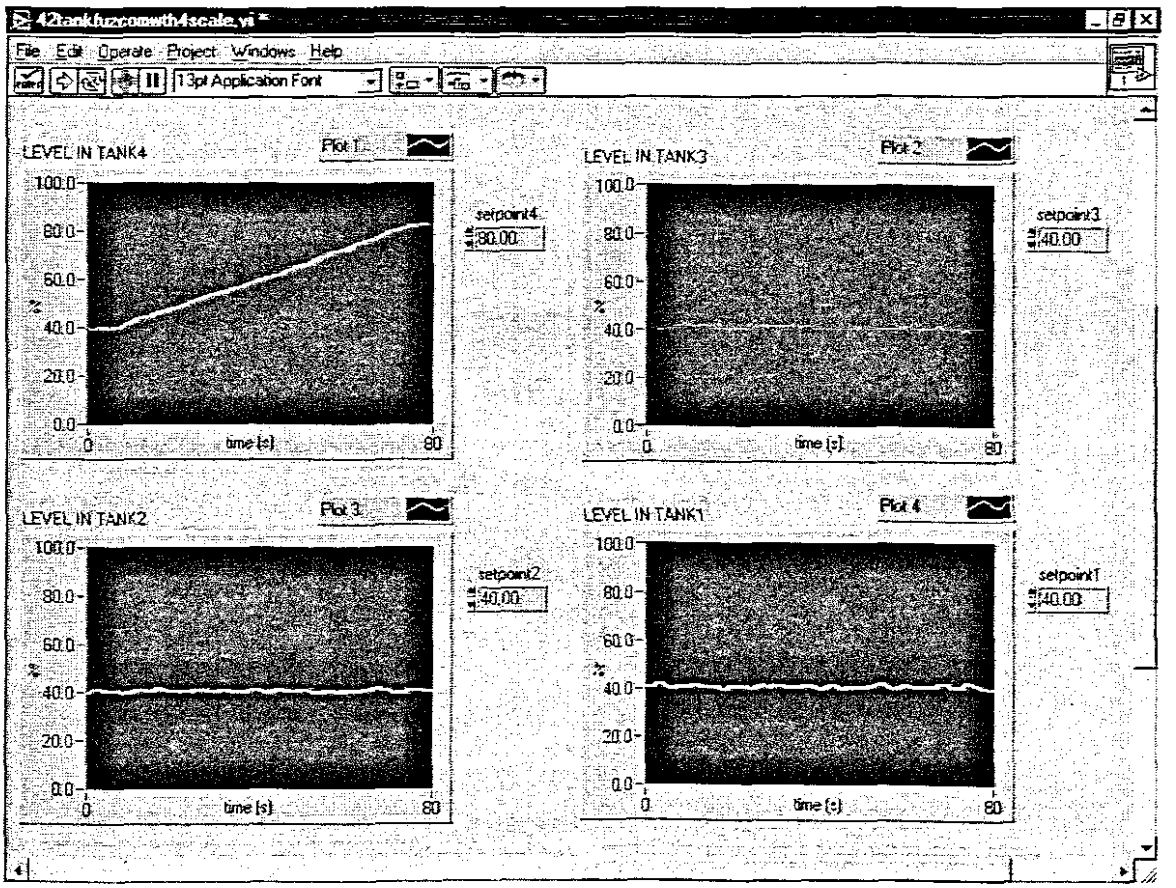


Figure 4.19 : Simulation showing a step from 40% to 80% capacity in tank4 with other tanks remaining unchanged.

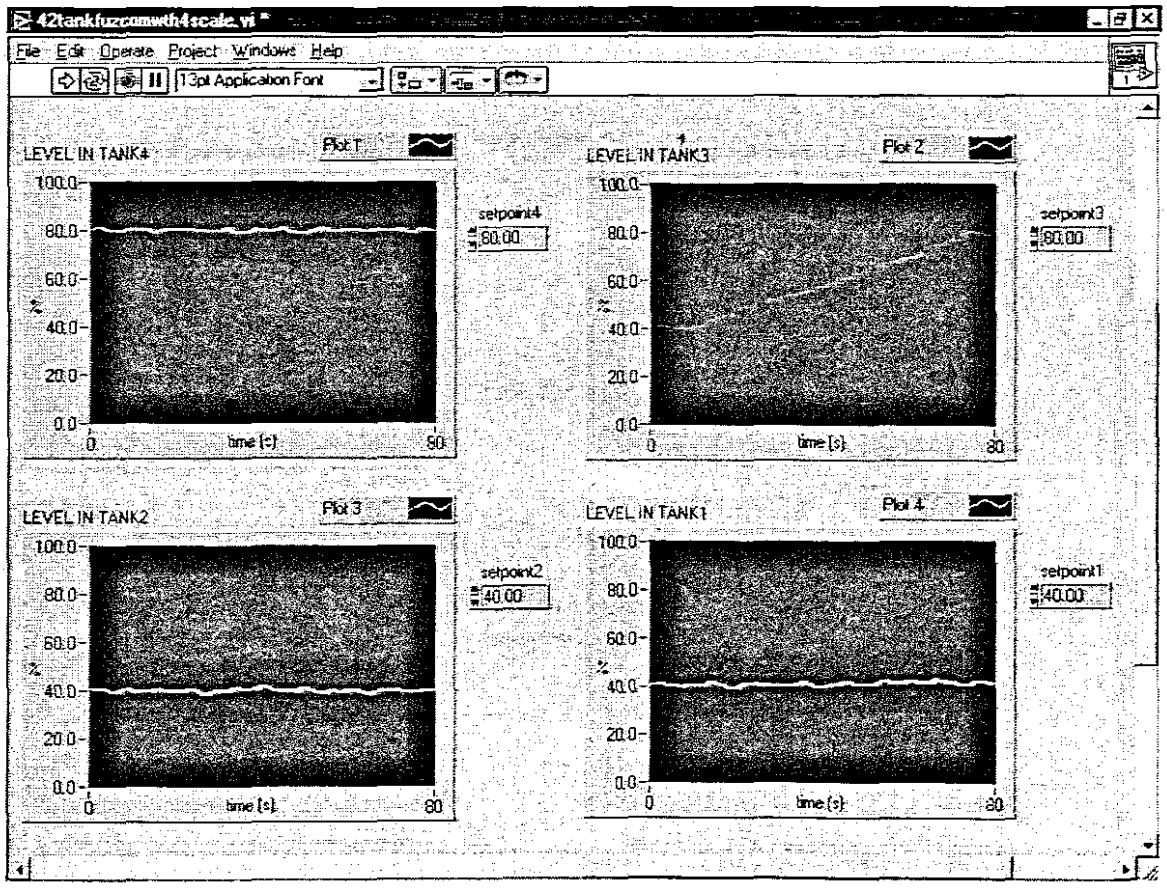


Figure 4.20: Simulation showing a step from 40% to 80% capacity in tank3 with other tanks remaining unchanged.

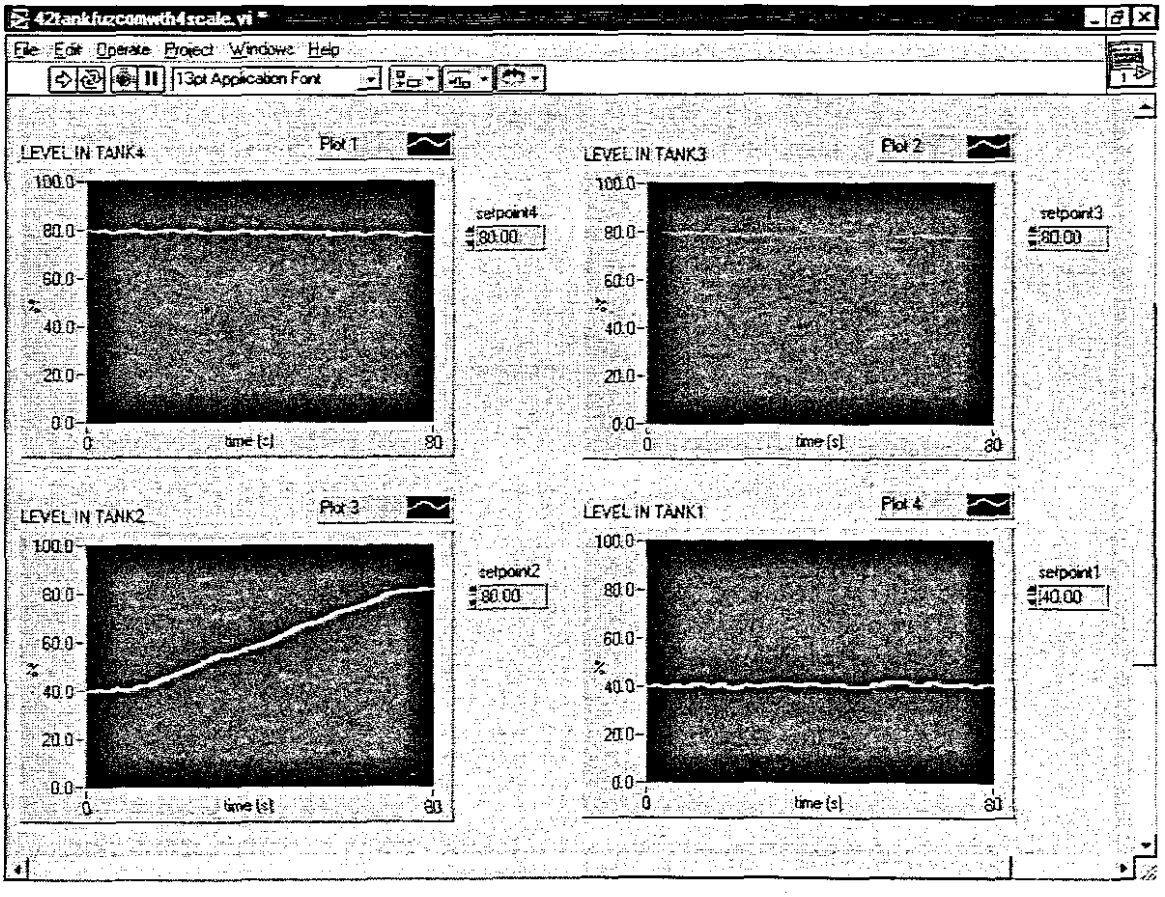


Figure 4.21: Simulation showing a step from 40% to 80% capacity in tank2 with other tanks remaining unchanged.

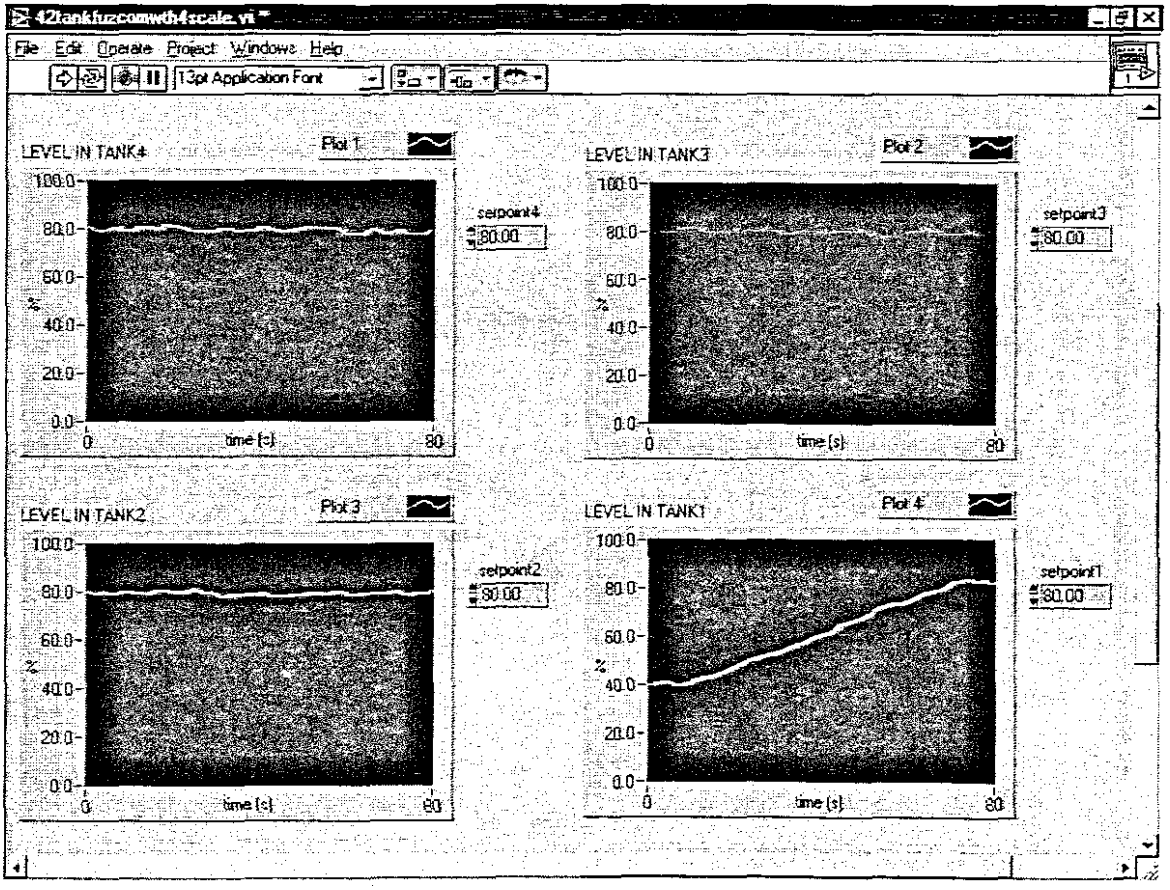


Figure 4.22: Simulation showing a step from 40% to 80% capacity in tank1 with other tanks remaining unchanged.

4.7.1 Simulation Model

The model for the simulation is based upon the model derived by (De Waal,1990) as mentioned in section 4.2.3 and depicted by table 4.4.

4.7.2 Labview front Panel

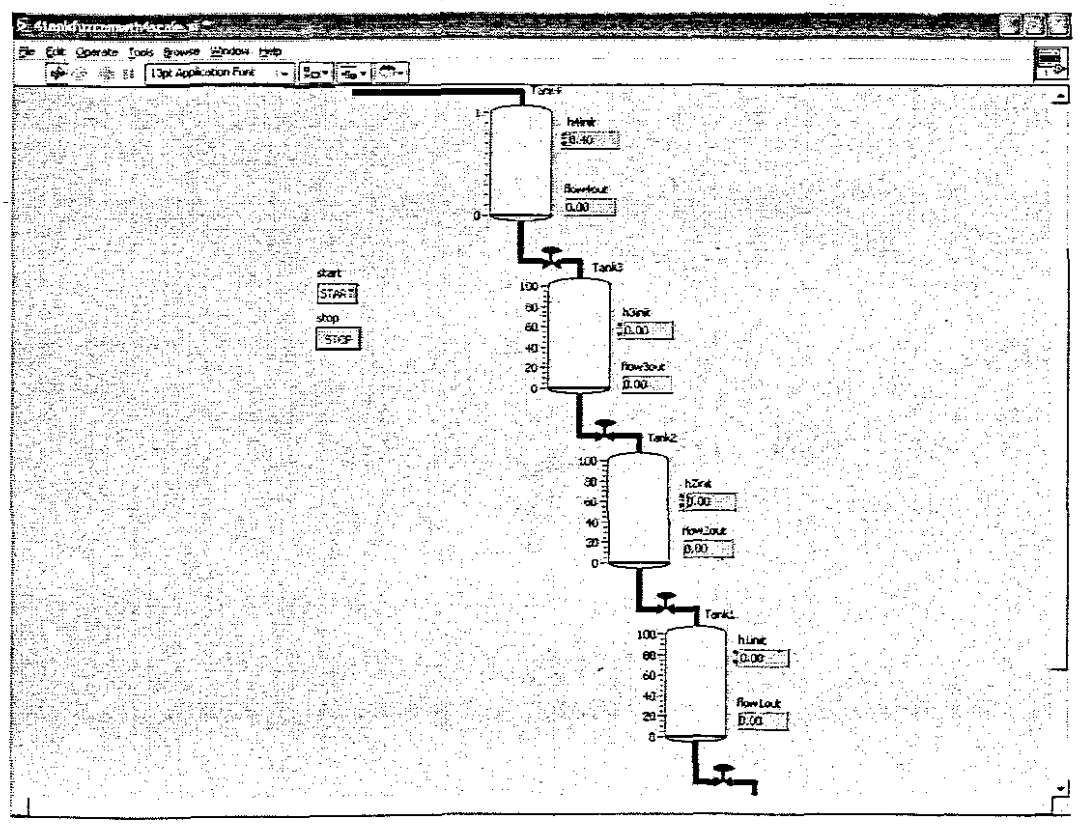


Figure 4.23: Simulation of tank system in LabVIEW

4.8 TUNING OF THE FUZZY CONTROLLERS

Tuning involved iterative "tweaking" of various fuzzy controller parameters such as the scaling gains of the input and output variables. This was done offline and in the absence of certain adaptive techniques such as that employed by the integration of neural networks and genetic algorithms with the fuzzy control methodology. This was decided upon as the optimisation criteria were satisfied with minimum effort with manual tuning.

4.8.1 Optimisation Criteria

The following criteria have been identified as the requirements for obtaining an optimised controller.

(1) Minimum time response of the controlled output. i.e. the water level in the tanks.

(2) Reduction or elimination of steady state error.

Based upon the criteria specified above the strategy that was adopted was to have a combination of on-off control for large error values and finer PI control near the setpoint. To formalise this, error values that do not fall within the "zero" fuzzy set will be regarded as "large". This applies to both negative and positive values.

4.9 SUMMARY OF CHAPTER

This chapter has dealt with the real time implementation of the optimized fuzzy controller for the level control plant. The controller is comprised of essentially a hardware and software component. The hardware is composed of a host computer that communicates with the process under control via input and output interface cards. The software component can be subdivided into the fuzzy algorithm, data read algorithm and data translation algorithm. Together all these component parts are integrated to form the controller in the feedback control loop.

The implementation of the controller has yielded results that can be regarded as satisfactory for effective closed loop level control because the level responses to a step input compare favorably with those obtained from the implementation of conventional PI controllers in a previous study (de Waal,1990). From the results for both the practical implementation as well as the simulation it can be seen that with the implementation of the decentralized fuzzy controllers there is a minimum time response for the controlled output as well as negligible overshoot and steady state error.

CHAPTER FIVE

FUZZY LOGIC CONTROL OF THE DISSOLVED OXYGEN CONCENTRATION IN AN ACTIVATED SLUDGE PROCESS

Water, in whatever form, is a necessary ingredient for virtually all living organisms. This in essence is what makes our planet fundamentally unique. More than 70% of the planet is covered with this vital resource, however only a small percentage is fit for human consumption. Nature has always had the ability to cleanse itself and that includes the water present in our environment.

However over the decades urbanization has increased and people have flocked to the major city centers. This massive influx put a major burden on the disposal of wastewater. Nature could tolerate the small communities that existed yesteryear and could handle the natural cleansing of water. Communities close to the city center grew at a phenomenal rate and the disposal of wastewater was no longer a trivial process.

Our sewerage disposal systems deposit the effluent water in large water bodies such as lakes and the surrounding oceans. It was thought that due to the vast expanse of these bodies of water that the amount of effluent being disposed of is insignificant and that the lakes and oceans could tolerate this wastewater. However, this is not the case.

Mankind has reached a point where we realize that more attention has to be paid to the preservation of our environment lest we seek our own destruction. Globally stricter measures are being instituted to ensure that the effluent wastewater from the treatment process is at a level of acceptable standards. As a result there is a need for advanced control and supervision strategies for these wastewater treatment plants.

Operators of the plants play a vital role in the operation of these systems and have to perform regular checks on a wide range of aspects to ensure that the quality of the effluent is within acceptable limits. However, due to stricter limits, margins of tolerance become smaller and under certain conditions, such as large input disturbances in input flows and input concentrations, operators require some form of assistance to ensure that the plant is still operating within the normal specifications.

Hence, control strategies are needed because of unpredictable input flows and unpredictable input concentrations. This then poses the question of "What is good

control?" Well the answer to that question could possibly be a financial one. If a control strategy can ensure that the effluent quality stays with the acceptable limits, then financially this is viable, as there will be no imposition of fines. Also in terms of the actual plant, if running costs can be reduced then the control strategy is welcomed. As an example if the control strategy reduces the following without compromising the effluent quality then the control strategy is also welcomed.

- No unnecessary use of chemicals
- No excess oxygen transferred to water
- No unnecessary recycling of water.

5.1 WASTE WATER TREATMENT PLANT

Treatment plants are wide and varied but generally they all attempt to accomplish the following.

- Removal of suspended solids
- Removal of organic material
- Removal of phosphate (plant dependant)
- Removal of nitrogen

The treatment process can be divided essentially into 4 stages as indicated below.

- Mechanical treatment
- Biological treatment
- Chemical treatment
- Sludge treatment

5.1.1 Mechanical Treatment

In this stage large solids that can cause potential physical damage to the process is removed. On route to the wastewater treatment plant many particles of different sizes is swept along with the flow. At the front end of the process are grids, which remove solids such as wood, paper even motor vehicle tyres. After this stage are sand filters that remove the smaller particles. As a last stage there might be a presedimentation process where smaller heavier particles are removed.

5.1.2 Chemical Treatment

Chemical treatment is an optional process as not all plants have been designed for this part of the process. However this will be mentioned for purposes of completeness. The process is mainly concerned with phosphorus removal. Phosphorus contributes to the proliferation of algae and if the effluent is to be deposited in standing water bodies the rapid growth of algae can cause oxygen depletion in the water. Chemicals are added which cause the phosphorus present in the water to precipitate and flocculate which can then be removed through sedimentation or floatation. Not all of the phosphorus is removed in this process as some is present in the organic material that was removed in the previous physical removal stage and some is removed in the biological treatment stage where they are locked in the microorganisms.

5.1.3 Biological Treatment

Ideally, after the water has passed through the previous two stages most of the material that is left in the water is organic. Now, to remove the various organic substances microorganisms are used to essentially transform the organic material into biomass. In particular the process of interest is specifically designed for the removal of nitrogen. In these processes nitrogen is usually in the form of ammonia and nitrate. An interesting aspect of the wastewater treatment is that this part of the plant is a natural process. As mentioned in the introduction this is where nature takes over to start the cleansing process. These microorganisms occur naturally in the bodies of water however in the wastewater plant their numbers are in abundance. The environment for the organisms also contains nourishment in very large quantities. As the microorganisms propagate and grow and as nitrogen gas is being produced the water starts its cleansing process. Ultimately the growing biomass is removed with the subsequent processes and the water contains a minimum of effluent substances. This biological treatment forms the core process and is commonly known as the activated sludge process.

5.2 THE ACTIVATED SLUDGE PROCESS

In its basic form the activated sludge process (ASP) consists of an aerated tank and a settler. By pumping air into the tank the microorganisms utilize oxygen to oxidize the organic material. As a result some of this material is transformed into carbon dioxide and some is incorporated into the cell mass. This cell mass is the sludge that is then removed in the secondary settlers and passed on to the sludge treatment process. The success of this process is heavily dependent upon the recycling of sludge from the secondary settler. This is required to ensure that the biomass population is maintained at a proper level.

As mentioned above the process can also be configured for the removal of nitrogen. This is a relatively complex procedure with the entire process involving the actions of both autotrophic as well as heterotrophic bacteria. Nitrogen is divided into two stages namely nitrification and denitrification.
Nitrogen can occur in the form of ammonium (NH_4^+) as well as in the form of nitrate (NO_3^-). In order to remove these compounds the process should be configured to have both aerobic as well as anoxic tanks. Anoxic is a term that implies that there is an absence of dissolved oxygen. With ammonium the generation of nitrogen gas is essentially a two-stage process. In the aerated tanks the ammonium reacts with oxygen and is transformed into nitrate. Incidentally, ammonium is not converted directly to nitrate. When ammonium reacts with oxygen the product is nitrite. The nitrite in turn reacts with oxygen to produce nitrate. This is the first stage of the conversion process and is known as nitrification. The success of the nitrification process is dependent upon both a rich supply of dissolved oxygen and a sufficiently long sludge age. Sludge age refers to the length of time the sludge is retained in the system. Nitrification is a relatively slow process and hence a long sludge age is required to ensure the success of this first stage. The first stage is accomplished by the actions of autotrophic bacteria.

In the second stage of the process nitrate is converted into nitrogen gas (N_2) by heterotrophic bacteria. This is known as denitrification. Because there is an absence of oxygen in the anoxic tanks heterotrophic bacteria utilize the nitrate to oxidize the organic and inorganic compounds. It is during this process that nitrate is chemically

reduced to nitrogen gas. For a successful denitrification process the requirements are in direct opposite to that of the nitrification process. Because heterotrophic bacteria have a higher growth rate than autotrophic bacteria denitrification is not dependent on a long sludge age. Also it requires low dissolved oxygen levels to ensure that the nitrate is used for the oxidation process as opposed to using the oxygen. Also, there should be an abundance of readily biodegradable substrate.

In terms of plant configuration for nitrogen removal it can either be designed for pre-denitrification or post-denitrification. In pre-denitrification the anoxic tanks are placed before the aerobic tanks and there is a requirement to have internal recycling of water from the aerobic tanks to ensure that high concentration of nitrates is passed on to the anoxic tanks. The advantage of this configuration is that the denitrifying bacteria has an abundance of organic material from the incoming water. The disadvantage is that the recycled water contains oxygen, which makes the denitrification process less efficient. As a result an external carbon source is required to reduce the oxygen concentration. In a system that has post-denitrification biodegradable substrate will be oxidized and therefore the addition of carbon will be required in the anoxic tanks.

5.3 MODELLING OF WASTEWATER TREATMENT PROCESS (WWTP)

5.3.1 Description Of The Activated Sludge Model No. 1 (ASM no.1)

Wastewater treatment plants are characterized by dynamic behavioral characteristics. This dynamic behavior can often be more complex than most industrial processes because the dynamics of WWTP are under the influence of living organisms. Also there can be large variations in the composition and concentration of these organisms, which further contributes to the ever-changing nature of the WW characteristics. In order to improve the performance of these plants certain control strategies have to be employed. However these strategies are dependent upon the ability to model the process in some way. Hence one has to ask, "what is a model". The definition of a model is wide and varied but in essence a model can be seen as a representation of some reality. In this instance that reality is the ASP. Although this dissertation deals with fuzzy logic control, the process of interest will still be mathematically modeled. The reason for this is that it has become accepted practice to utilize the IAWQ activated sludge model No.1 (ASM1), see Henze *et al.*(1987).

The ASM1 has been characterized by having thirteen state variables as indicated in the table 5.1 and eight processes. Each reactor is defined by differential equations comprising the state variables and eight processes. These processes are:

- For heterotrophic biomass
  1. aerobic growth
  2. anoxic growth
  3. decay
- For autotrophic biomass which do not grow in an anoxic environment
  4. growth
  5. decay
- For the substrate
  6. hydrolysis of slowly biodegradable substrate to readily biodegradable substrate
  - For the nitrogen
  7. Hydrolysis of suspended organic nitrogen to soluble organic nitrogen.
  8. Ammonification of soluble organic nitrogen to ammonia.

Organic (COD)	Nitrogen(N)	Other
soluble inert (S_I)	ammonia (S_{NH})	oxygen (S_O)
suspended inert (X_I)	nitrate (S_{NO})	alkalinity (S_{ALK})
suspended inert products (X_P)	soluble organic nitrogen (S_{ND})	
readily biodegradable substrate (S_S)	suspended organic nitrogen (X_{ND})	
slowly biodegradable substrate (X_S)		
heterotrophic biomass(X_{BH})		
autotrophic biomass (X_{BA})		

Table 5.1: State variables as defined by the ASM 1 model with abbreviations indicated in brackets

5.3.2 Simulation Model in MATLAB/SIMULINK™

In the ASP each tank is regarded as a bioreactor. For these reactors by applying material and energy balances using the fundamental transport, stoichiometric, thermochemical, and kinetic relationships can develop models. Based upon this the models comprise of a set of differential equations which can be easily solved with the aid of powerful simulation packages such as MATLAB/SIMULINK™. In this study a generic model for a bioreactor for both an aerobic and an anoxic tank was developed and simulated in MATLAB/SIMULINK™. The entire ASM NO.1 model has been used to model each bioreactor compartment with the exception of the reactions related to alkalinity because it does not have a significant effect on the other components. Also it is common practice to combine the compounds for particulate inert matter X_I and X_P into one variable, X_{IP}. As a result every reactor is modeled in MATLAB/SIMULINK™ is modeled using 11 differential equations. Also to simplify the model the sedimentation process is considered to be ideal. For model development it is assumed that each bioreactor is completely mixed and each compound defined in terms of a mass-balance equation. The mass balance equation has the following structure.

$$\frac{dP}{dt} = R_P(t) + D(t)(P_{in}(t) - P(t))$$

where,

- $P_{in}(t)$ is influent concentration of the component
- $P(t)$ is the concentration of the component in the effluent
- $D(t)$ is the dilution rate (flow/volume)
- $R_P(t)$ denotes reaction rate for the component.

The reaction rate $R_P(t)$ has different forms in dependence of the type of the process and components. The different reaction rates are described below according to the ASM1 model. They describe reactions occurring in both anoxic and aerobic environments for the respective compounds. For model development however it should be noted that when modeling an aerobic tank the reactions that are attributed to

anoxic conditions should not be incorporated into the model. In a similar vein when modeling anoxic tanks reactions associated with aerobic conditions should not be taken into consideration. Also the hydrolysis and decay processes are ignored given the poor knowledge of the hydrolysis process.(Henze et al,1987).

According to the IAWQ model the reaction rates for the various compounds are defined as follows.

$$\frac{dS_S}{dt} = -\frac{\hat{\mu}_H}{Y_H}\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{O,H} + S_O}\right)X_{B,H} - \frac{\hat{\mu}_H}{Y_H}\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{B,H} + k_h \frac{X_S / X_{B,H}}{K_X + X_S / X_{B,H}}\left(\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_h\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right)X_{B,H} \tag{5.1}$$

$$\frac{dX_S}{dt} = (1 - f_p)(b_H X_{B,H} + b_A X_{B,H}) - k_h \frac{X_S / X_{B,H}}{K_X + X_S / X_{B,H}}\left(\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_h\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right)X_{B,H} \tag{5.2}$$

$$\frac{dX_I}{dt} = f_p(b_H X_{B,H} + b_A X_{B,A}) \tag{5.3}$$

$$\frac{dX_{B,H}}{dt} = \hat{\mu}_H\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{O,H} + S_O}\right)X_{B,H} + \hat{\mu}_H\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{B,H} - b_H X_{B,H} \tag{5.4}$$

$$\frac{dX_{B,A}}{dt} = \hat{\mu}_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}} \right) \left(\frac{S_O}{K_{O,A} + S_O} \right) X_{B,A} - b_A X_{B,A} \tag{5.5}$$

$$\begin{aligned} \frac{dS_O}{dt} = & -\frac{1 - Y_H}{Y_H} \hat{\mu}_H \left(\frac{S_S}{K_S + S_S} \right) \left(\frac{S_O}{K_{O,H} + S_O} \right) X_{B,H} - \\ & \frac{4.57 - Y_A}{Y_A} \hat{\mu}_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}} \right) \left(\frac{S_O}{K_{O,A} + S_O} \right) X_{B,A} + K_L a \left(S_{O,sat} - S_O \right) \end{aligned} \tag{5.6}$$

$$\begin{aligned} \frac{dS_{NH}}{dt} = & -i_{XB} \hat{\mu}_H \left(\frac{S_S}{K_S + S_S} \right) \left(\frac{S_O}{K_{O,H} + S_O} \right) X_{B,H} - \\ & i_{XB} \hat{\mu}_H \left(\frac{S_S}{K_S + S_S} \right) \left(\frac{K_{O,H}}{K_{O,H} + S_O} \right) \left(\frac{S_{NO}}{K_{NO} + S_{NO}} \right) \eta_g X_{B,H} - \\ & \left(i_{XB} + \frac{1}{Y_A} \right) \hat{\mu}_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}} \right) \left(\frac{S_O}{K_{O,A} + S_O} \right) X_{B,A} + k_a S_{ND} X_{B,H} \end{aligned} \tag{5.7}$$

$$\begin{aligned} \frac{dS_{ND}}{dt} = & -k_a S_{ND} X_{B,H} + \\ & \frac{X_{ND}}{X_S} k_h \frac{X_S / X_{B,H}}{K_X + X_S / X_{B,H}} \left(\left(\frac{S_O}{K_{O,H} + S_O} \right) + \eta_h \left(\frac{K_{O,H}}{K_{O,H} + S_O} \right) \left(\frac{S_{NO}}{K_{NO} + S_{NO}} \right) \right) X_{B,H} \end{aligned} \tag{5.8}$$

$$\begin{aligned} \frac{dX_{ND}}{dt} = & \left(i_{XB} - f_p i_{XP} \right) \left(b_H X_{B,H} + b_A X_{B,H} \right) - \\ & \frac{X_{ND}}{X_S} k_h \frac{X_S / X_{B,H}}{K_X + X_S / X_{B,H}} \left(\left(\frac{S_O}{K_{O,H} + S_O} \right) + \eta_h \left(\frac{K_{O,H}}{K_{O,H} + S_O} \right) \left(\frac{S_{NO}}{K_{NO} + S_{NO}} \right) \right) X_{B,H} \end{aligned} \tag{5.9}$$

$$\frac{dS_{NO}}{dt} = -\frac{1-Y_H}{2.86}\hat{\mu}_H\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{K_{O,H}}{K_{O,H}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right)\eta_g X_{B,H} + \frac{\hat{\mu}_A}{Y_A}\left(\frac{S_{NH}}{K_{NH}+S_{NH}}\right)\left(\frac{S_O}{K_{O,A}+S_O}\right)X_{B,A} \quad (5.10)$$

The parameters present in the reaction rates above are standard values according to the list as compiled by the IAWQ model parameters. A listing of these parameters are available in the appendix.

5.4 CONTROL OF THE DISSOLVED OXYGEN CONCENTRATION

The dissolved oxygen concentration in the aerobic bioreactors should be adequate enough to ensure that the microorganisms efficiently do the removal of organic compounds and ammonium conversion to nitrate. On the other hand a high concentration of dissolved oxygen will require a high airflow rate, which will result in excessive consumption of electricity as well as possible degradation of sludge quality. Also a high concentration of dissolved oxygen present in the internally recycled water can result in poor denitrification. Hence from the previous discussion it can be seen that proper control of the dissolved oxygen concentration is a necessary requirement both from an economical and process point of view.

Because of the importance of dissolved oxygen control many different control strategies have been proposed in the past. Many of the control strategies have been from a classical point of view. Many researchers (Flanagan et al, 1977), (Wells, 1979), (Ko et al, 1982), (Holmberg et al, 1989)(Bocken et al, 1989)(Rundqwist, 1986)(Carlsson et al, 1994) have contributed to the development of control strategies to maintain proper levels of dissolved oxygen concentration in the aerobic reactors. This study has however concentrated upon the development of a control strategy based on fuzzy logic control theory.

The fuzzy logic control strategy is ideal for control of the dissolved oxygen concentration because of the ability of this paradigm to handle the nonlinear dynamics of the dissolved oxygen concentration.

5.4.1 Description of Mass Balance Equation for Dissolved Oxygen Concentration

As described in section 5.3.1 the compounds present in the bioreactors are defined in terms of a mass balance differential equation. Because the objective is to control the dissolved oxygen concentration a more detailed explanation of the equation governing the dissolved oxygen concentration will be given. The mass balance equation describes the rate of change of the dissolved oxygen. Consider the terms on the right hand side of the equation. The first term is a mass balance comprising of the difference between the influent oxygen and effluent oxygen multiplied by the dilution rate. The next term describes the addition of oxygen and the last term describes the respiration rate of the microorganisms.

$$\frac{dy(t)}{dt} = \frac{Q(t)}{V}(y_{in}(t) - y(t)) + K_L a(u(t))(y_{sat} - y(t)) - R(t) \tag{5.11}$$

Where

$y(t)$	*is the DO concentration in the aerobic tank*
$y_{in}(t)$	*is the DO concentration of the input flow*
y_{sat}	*is the saturated value of the DO concentration*
$Q(t)$	*is the flow rate of the wastewater.*
V	*Volume of wastewater*
$K_L a(u)$	*is the oxygen transfer function*
$u(t)$	*airflow rate in the aerobic tank*
$R(t)$	*respiration rate*

What is of importance is the oxygen transfer function. The mass balance equation that governs the dissolved oxygen concentration in the aerobic tanks has the oxygen transfer function $K_L a$ as part of the equation. This parameter is essentially nonlinear in nature and for an effective control strategy should be incorporated into the design of

the controller. The oxygen transfer function describes the rate at which oxygen is transferred to the water by the aeration system and it is a function of the airflow rate. Various methods been devised to define the transfer function but the most common way to describe it is by the following equation. The relationship between this parameter and the airflow rate is a non-linear one. A typical model to represent the relationship between the airflow rate and the oxygen transfer function is an exponential model as depicted in the equation below.

$$K_L a(u(t)) = k_1(1 - e^{-k2 \cdot u(t)}) \tag{5.12}$$

Where,

k_1 and k_2 are parameters associated with the aerobic tanks.

Based upon this the DO concentration can now be controlled by controlling the airflow rate. In equation (5.11) above $K_L a$ is replaced by equation (5.12) above.

The type of control action employed in this study is one of set-point control. Previous studies (Nielsen and Lynggard, 1993) (Lindberg, 1997) have found this type of control to be effective for efficiently controlling the dissolved oxygen concentration and its subsequent effects such as lowering the nitrate concentration in the effluent.

5.5 SI MULATION SETUP IN MATLAB

A block diagram of the simulation as implemented in MATLAB/SIMULINK is shown in figure 5.1 below. The process simulated consists of five biological reactors similar to the one as specified by the IAWQ benchmark for evaluating control strategies. The process is configured for pre-denitrification with the first two tanks being anoxic and the rest of the tanks configured as aerobic.

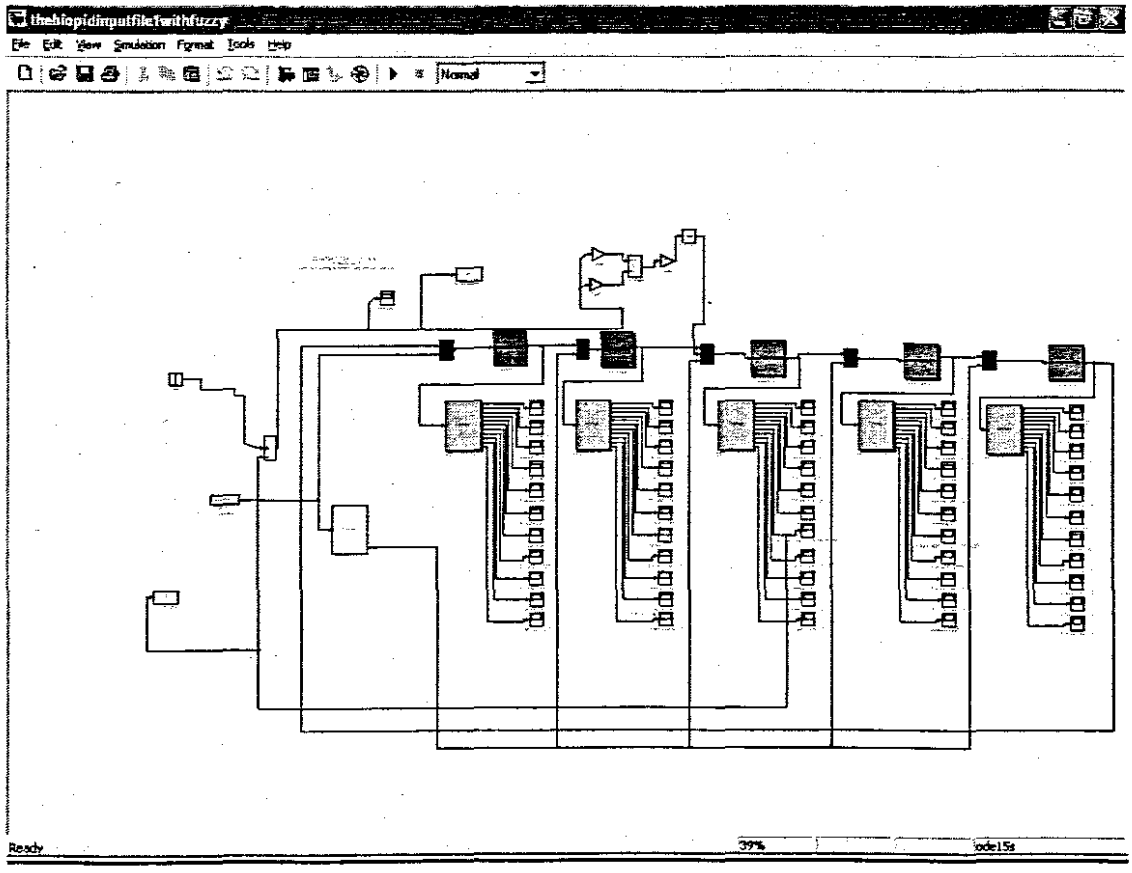


Figure 5.1: The simulation as setup in the MATLAB/SIMULINK environment

5.5.1 Code Description of Bioreactors in Matlab

The anoxic and aerobic tanks in the simulink simulation has been setup as S-functions. An S-function essentialy enables users to define custom simulink blocks using either MATLAB or C-language code. This is a useful feature as it enables existing code in MATLAB or C to be incorporated into the simulink environment. It is also useful when portions of code is easier to define algorithmically as opposed to graphical methods with the use of block diagram notation. It is particularly useful in this instance as it is easier to define the biological reactors algorithmically considering that each reactor is comprised of approximately 11 differential equations.

Below is a skeletal structure of the code defining each bioreactor

function [sys,x0,str,ts] = bioreactmodinc(t,x,u,flag) %define function name
%CSFUNC An example M-file S-function for defining a continuous system.

```
switch flag,
%%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================
==========

% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================
==========
```

```
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 11;
sizes.NumInputs = 20;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,8.4,1552,81.0,1439,50.6,0,0.6,9.7,0.7,4.3];
str = [];
ts = [0 0];
% end mdlInitializeSizes
```

Initialization and setup of the S function occurs here

```
%=============================================================
=============

% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================
=============

function sys=mdlDerivatives(t,x,u)

% PARAMETERS
%Include all values for parameters to be used in differential equations.

%INPUTS
%Define all inputs to the system.

% FLOW RATES AND TANK VOLUMES
%Include all flow rates and tank volumes

% DEFINE THE SETTLER CHARACTERISTICS
%Include items such as sludge age.
```

%DEFINE PROCESSES ACCORDING TO IAWQ MODEL

%Include equations describing the various processes occurring in the reactor

%DEFINE DIFFERENTIAL EQUATIONS ACCORDING TO IAWQ MODEL

%AND PLANT LAYOUT

%Include all the differential equations that describes the bioreactor.

% end mdlDerivatives

%

%==========

% mdlOutputs

% Return the block outputs.

%==========

%

function sys=mdlOutputs(t,x,u)

sys = x;

% end mdlOutput

The use of S-functions also allows for flexibility to enable physical parameters such as influent concentrations, flow rate, and bioreactor size to be changed.

5.5.2 Simulation Parameters

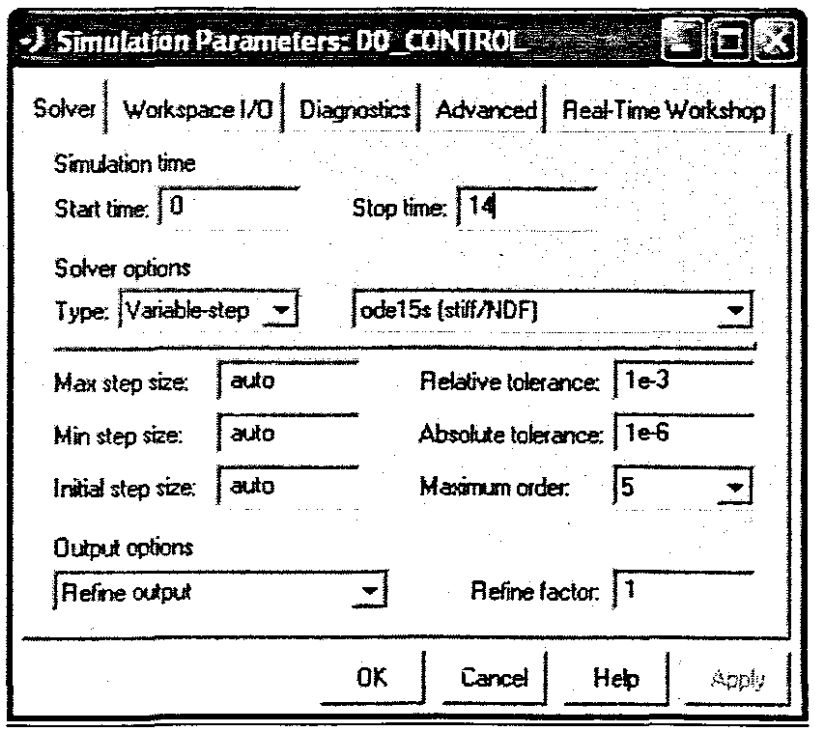


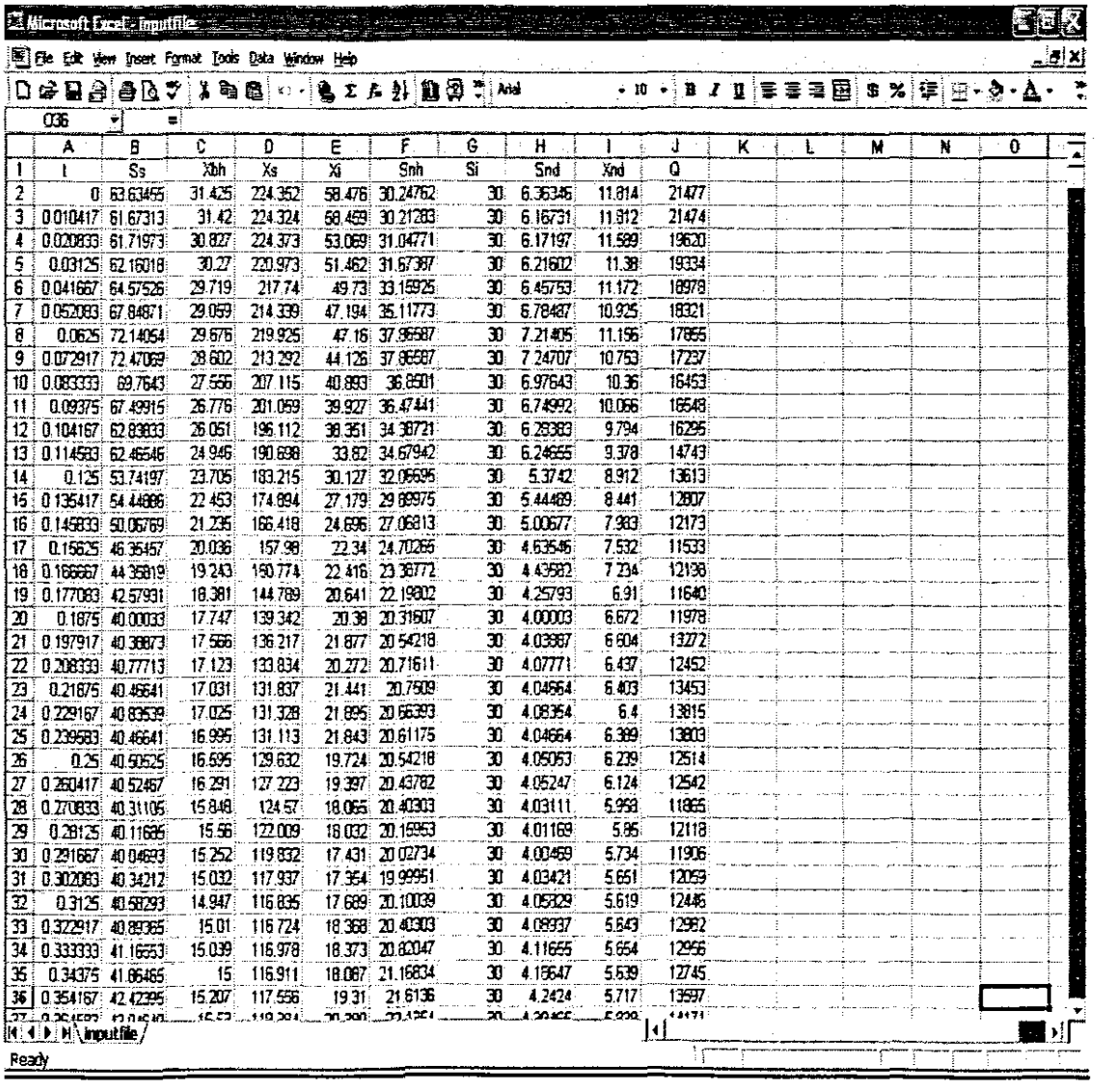
Figure 5.2: Simulation parameters for simulation of fuzzy dissolved oxygen control

5.5.2.1 Integration Method

The selected method of integration is of vital importance. The differential equations are characterized as stiff and hence the integration method chosen in the simulation environment is ODE15S for solving ordinary stiff differential equations. Even with this choice of integration method the simulation runs fairly slow.

5.5.2.2 Period of simulation

The simulation period represents fourteen days. This is dictated by the data that is input in the system. This data is a diurnal pattern representing the inflow constituents over a period of fourteen days. This pattern was obtained from the benchmark model for testing control strategies of activated sludge plants.



	A	B	C	D	E	F	G	H	I	J
1	t	Ss	Xbh	Xs	Xi	Snh	Si	Snd	Xnd	Q
2	0	63.63455	31.425	224.352	58.476	30.24762	30	6.36346	11.814	21477
3	0.010417	61.67313	31.42	224.324	58.459	30.21283	30	6.16731	11.812	21474
4	0.020833	61.71973	30.827	224.373	53.069	31.04771	30	6.17197	11.599	19620
5	0.03125	62.16018	30.27	220.973	51.462	31.67397	30	6.21602	11.38	19334
6	0.041667	64.57526	29.719	217.74	49.73	33.15925	30	6.45753	11.172	18978
7	0.052083	67.84871	29.059	214.339	47.194	35.11773	30	6.78487	10.925	18321
8	0.0625	72.14054	29.676	219.925	47.16	37.96587	30	7.21405	11.196	17855
9	0.072917	72.47069	28.602	213.292	44.126	37.86587	30	7.24707	10.753	17237
10	0.083333	69.7643	27.556	207.115	40.893	36.8501	30	6.97643	10.36	16453
11	0.09375	67.49915	26.776	201.059	39.927	36.47441	30	6.74992	10.066	16548
12	0.104167	62.83833	26.051	196.112	38.351	34.38721	30	6.28383	9.794	16295
13	0.114583	62.46546	24.946	190.698	33.82	34.67942	30	6.24655	9.378	14743
14	0.125	53.74197	23.705	183.215	30.127	32.06695	30	5.3742	8.912	13613
15	0.135417	54.44886	22.453	174.894	27.179	29.89975	30	5.44489	8.441	12807
16	0.145833	50.06769	21.235	166.418	24.696	27.06813	30	5.00677	7.983	12173
17	0.15625	46.36457	20.036	157.98	22.34	24.70266	30	4.63646	7.532	11533
18	0.166667	44.35819	19.243	150.774	22.416	23.38772	30	4.43582	7.234	12138
19	0.177083	42.57931	18.381	144.789	20.641	22.19802	30	4.25793	6.91	11640
20	0.1875	40.00033	17.747	139.342	20.38	20.31607	30	4.00003	6.672	11978
21	0.197917	40.36873	17.566	136.217	21.877	20.54218	30	4.03687	6.604	13272
22	0.208333	40.77713	17.123	133.834	20.272	20.71611	30	4.07771	6.437	12452
23	0.21875	40.46641	17.031	131.837	21.441	20.7909	30	4.04664	6.403	13453
24	0.229167	40.83539	17.025	131.328	21.895	20.66393	30	4.08354	6.4	13815
25	0.239583	40.46641	16.995	131.113	21.843	20.61175	30	4.04664	6.399	13803
26	0.25	40.50525	16.595	129.632	19.724	20.54218	30	4.05053	6.239	12514
27	0.260417	40.52467	16.291	127.223	19.397	20.43782	30	4.05247	6.124	12542
28	0.270833	40.31105	15.848	124.57	18.066	20.40303	30	4.03111	5.958	11865
29	0.28125	40.11685	15.56	122.009	18.032	20.15953	30	4.01169	5.85	12118
30	0.291667	40.04693	15.252	119.832	17.431	20.02734	30	4.00469	5.734	11906
31	0.302083	40.34212	15.032	117.937	17.354	19.99951	30	4.03421	5.651	12059
32	0.3125	40.58293	14.947	116.835	17.689	20.10039	30	4.05829	5.619	12446
33	0.322917	40.89365	15.01	116.724	18.368	20.40303	30	4.09937	5.643	12992
34	0.333333	41.16553	15.039	116.978	18.373	20.82047	30	4.11655	5.654	12956
35	0.34375	41.86485	15	116.911	18.087	21.16834	30	4.18647	5.639	12745
36	0.354167	42.42295	15.207	117.556	19.31	21.6136	30	4.2424	5.717	13597

Table 5.2: A snapshot of part of the diurnal pattern used in the simulation

5.6 Design and Implementation of fuzzy controller

For purposes of the simulation the controller was developed with the fuzzy logic toolbox in MATLAB. The controller was designed on the basis of the work done in chapter 3. Figures 5.2 to figure 5.4 show the input and output linguistic variables for the designed fuzzy controller.

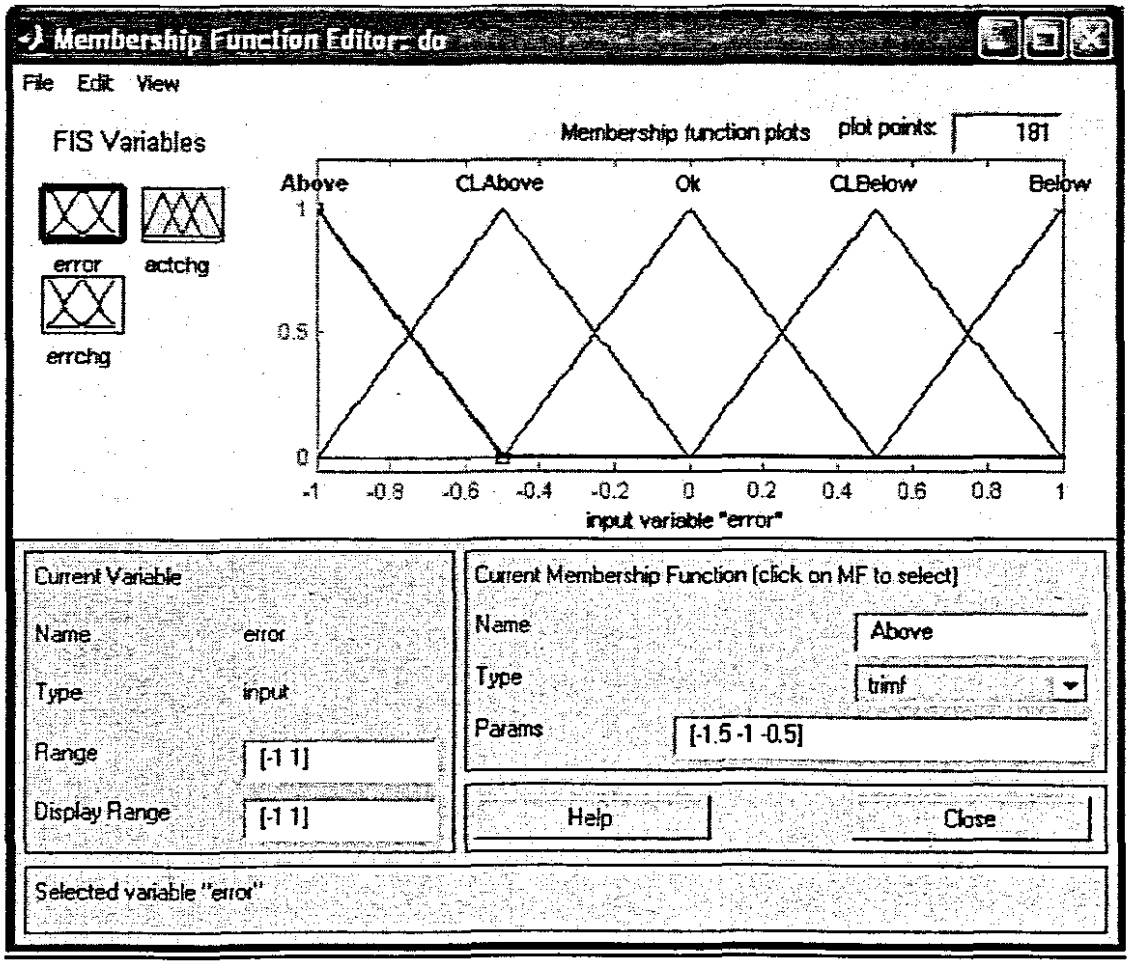


Figure 5.3: A snapshot of the screen showing the linguistic variable "error" as designed with the fuzzy logic toolbox

Figure 5.2 shows the dialogue box for defining and editing the input linguistic variable "error". From the figure it can be seen that the fuzzy sets are placed on a normalized universe. Editing of the fuzzy sets can be done by either entering the parameters of the fuzzy sets in the "Params" field or by clicking on the fuzzy set and dragging the corners of the fuzzy set with the mouse. The name of the fuzzy set that is being edited appears in the "Name" field.

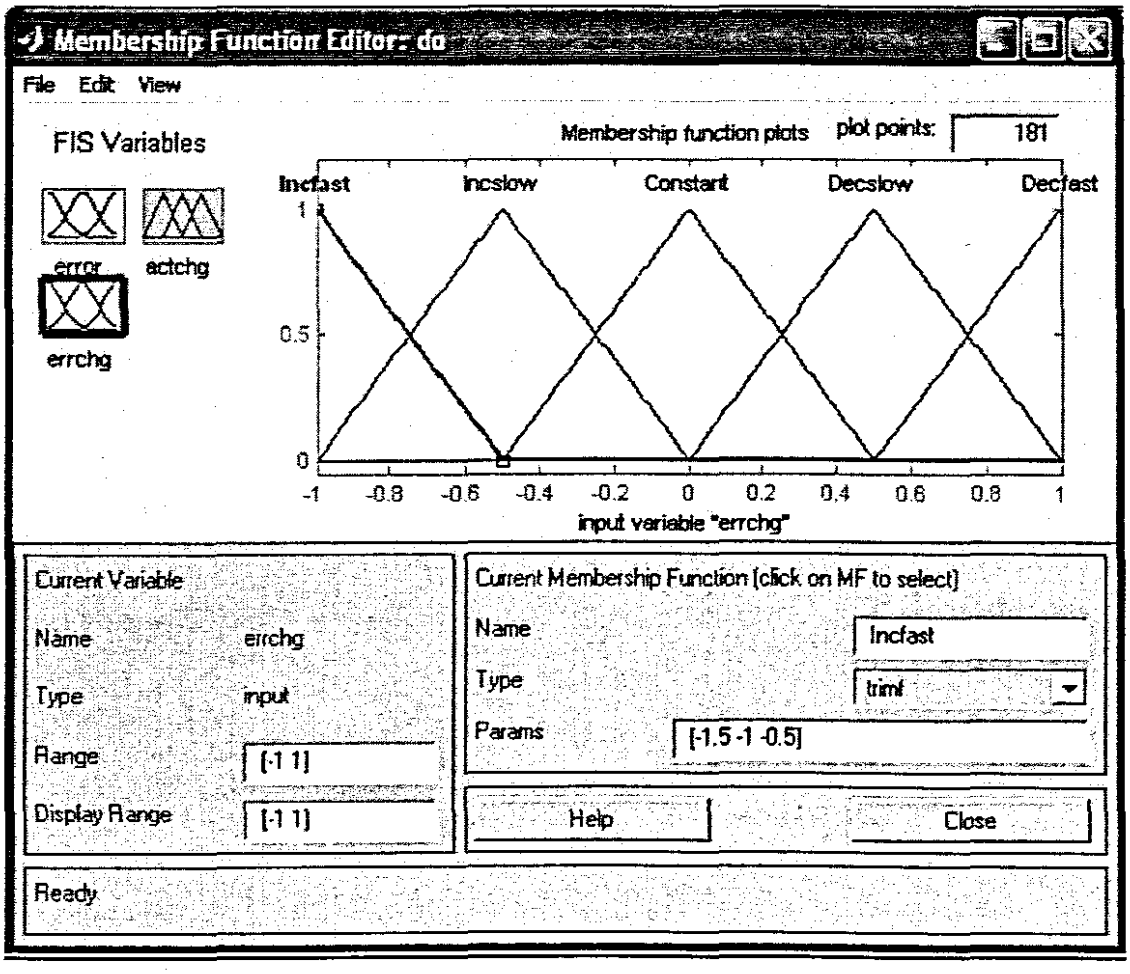


Figure 5.4: A snapshot of the screen showing the linguistic variable "error change" as designed with the fuzzy logic toolbox

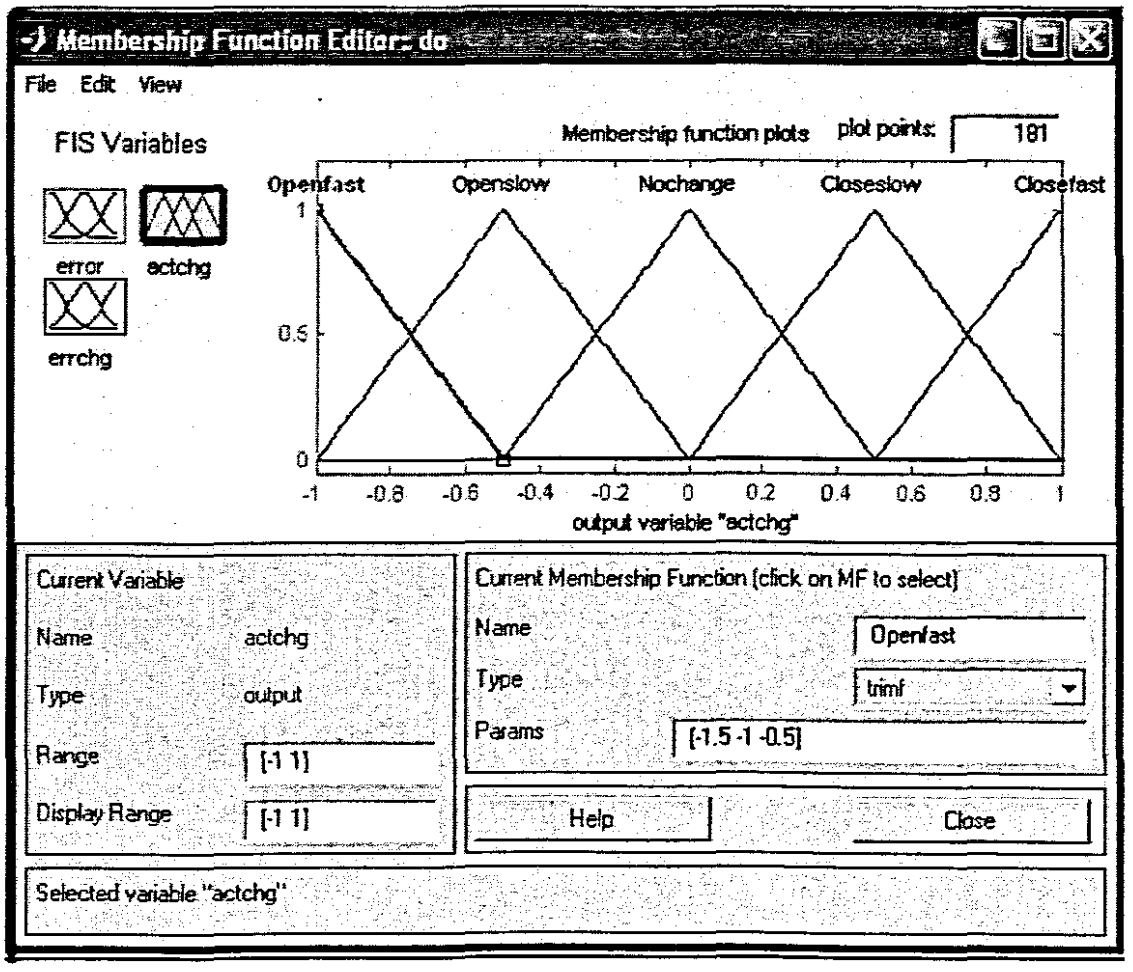


Figure 5.5: A snapshot of the screen showing the linguistic variable "actuator change" as designed with the fuzzy logic toolbox

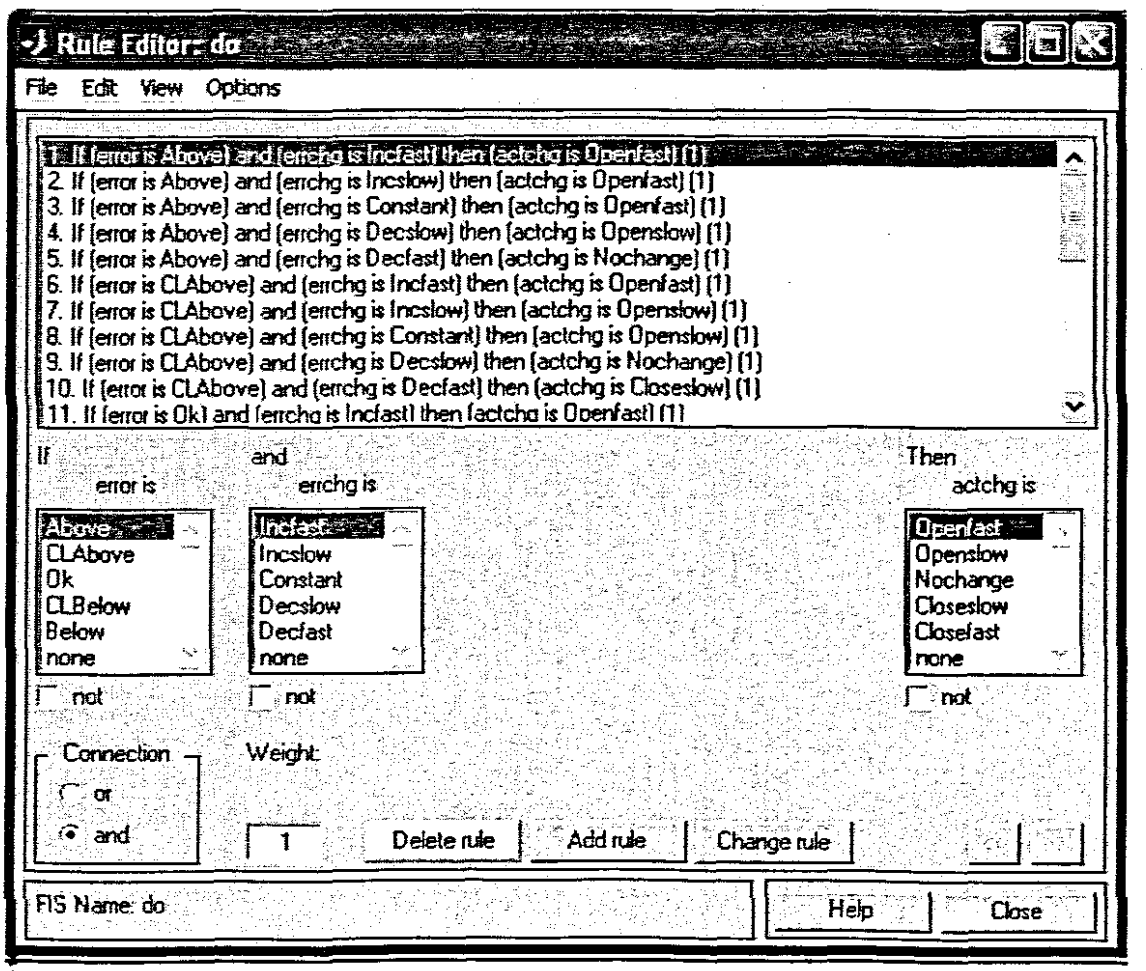


Figure 5.6: A snapshot of the screen showing the rules as designed with the fuzzy logic toolbox

5.6.1 Procedure for simulation

The following procedure was followed to simulate the fuzzy control of the dissolved oxygen concentration in an aerobic reactor.

- At the MATLAB command prompt the input diurnal pattern was loaded
  >> load inputfile
- The fuzzy designed fuzzy controller was then loaded.
  fuzcon=readfis('filename')
  This creates a fuzzy inference system (fis) matrix in the workspace corresponding to the fis file 'filename' on disk.
- Start "simulink", the simulation environment
  >> simulink
- In simulink the designed simulation was loaded. This is done using the pull down menu of simulink and loading the file "DO_CONTROL".
- Once the simulation design is loaded the setpoint value for the dissolved oxygen concentration is set to 2mg/l.This is done in the block representing the step input in the simulation schematic.
- The simulation is then started by selecting start from the simulation menu or simultaneously pressing the keys CTRL_T.

5.6.2 Results obtained

The simulation yielded positive results that showed that even in its untuned state the fuzzy controller performed admirably. Using the diurnal pattern as input to the system the dissolved oxygen concentration will show fluctuations as the input concentrations to the system vary. Under the control of the fuzzy controller the dissolved oxygen concentration in the aerobic tank showed a reduction in fluctuations. The important aspect of maintaining the proper dissolved oxygen concentrations is evident in the concentrations of the effluent such as nitrogen.

The results show that even in its untuned state the fuzzy controller performs admirably yielding results that show a reduction in the fluctuations of the dissolved oxygen concentration in an aerobic tank. Coupled with this it can be seen that there is also a reduction in the concentration of nitrogenous compounds in the effluent wastewater. This can be seen by comparing the results obtained in figures 5.8 to figures 5.11 with those obtained under uncontrolled conditions in figure 5.13 to figure 5.16.

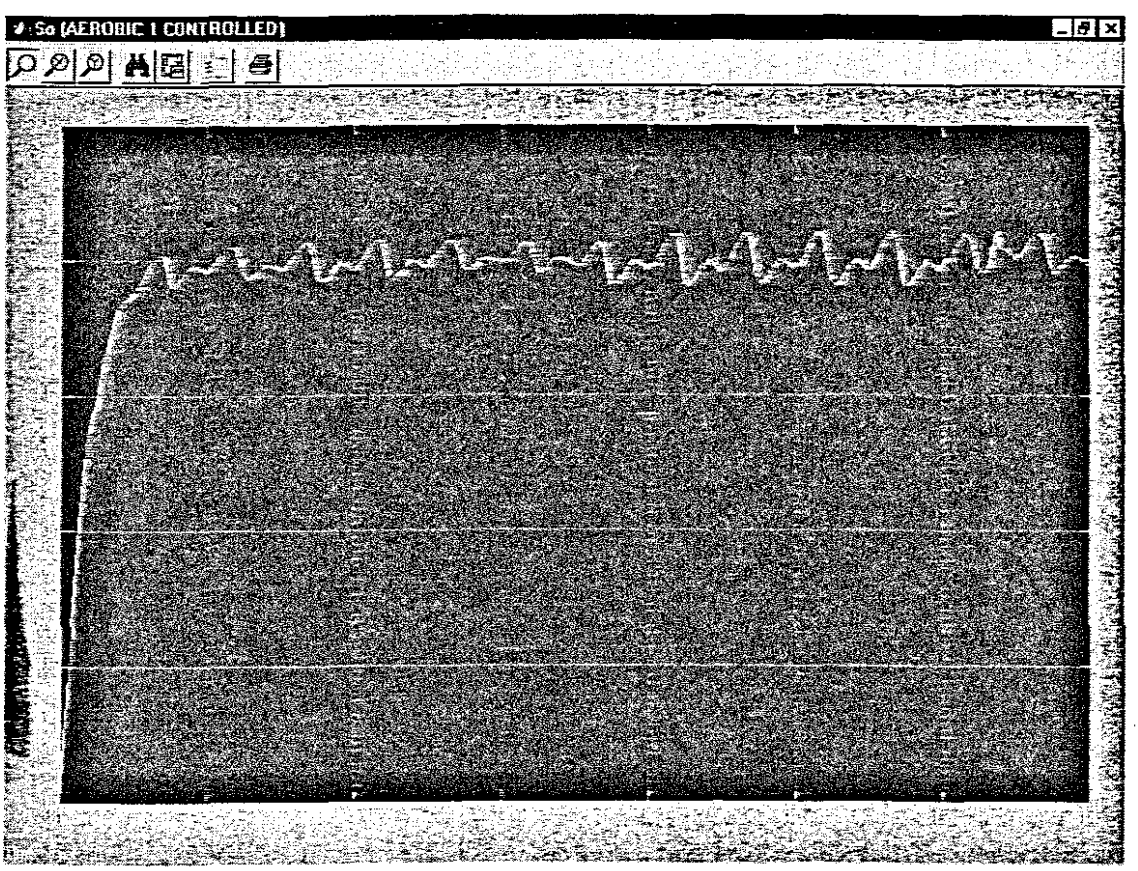


Figure 5.7: Response of dissolved oxygen concentration under fuzzy setpoint control

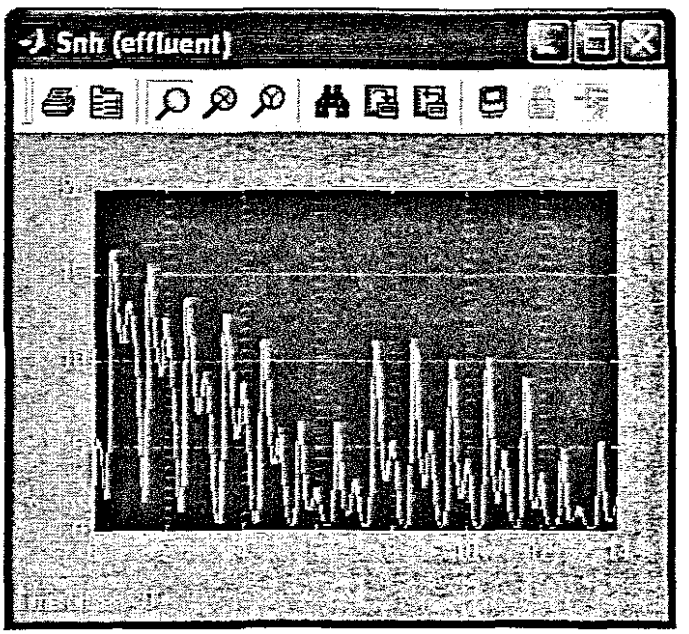


Figure 5.8: Plot of ammonia concentration in the effluent under controlled conditions

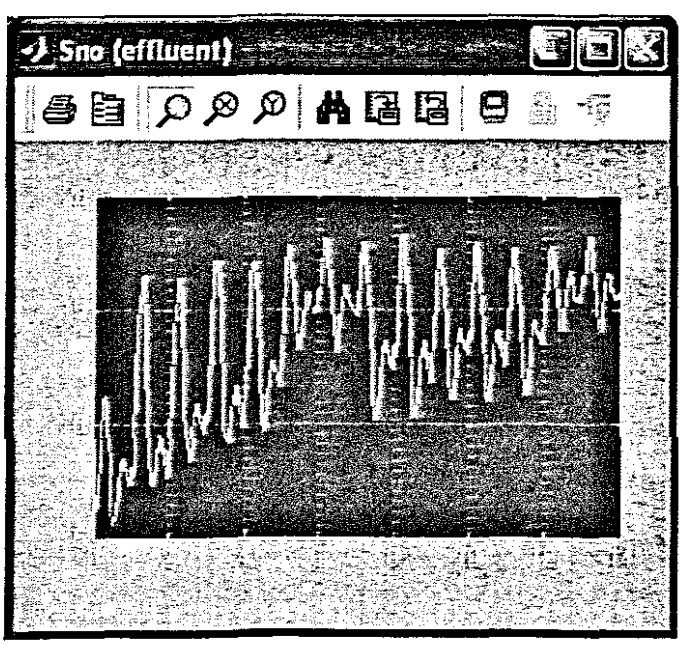


Figure 5.9: Plot of nitrate concentration in the effluent under controlled conditions

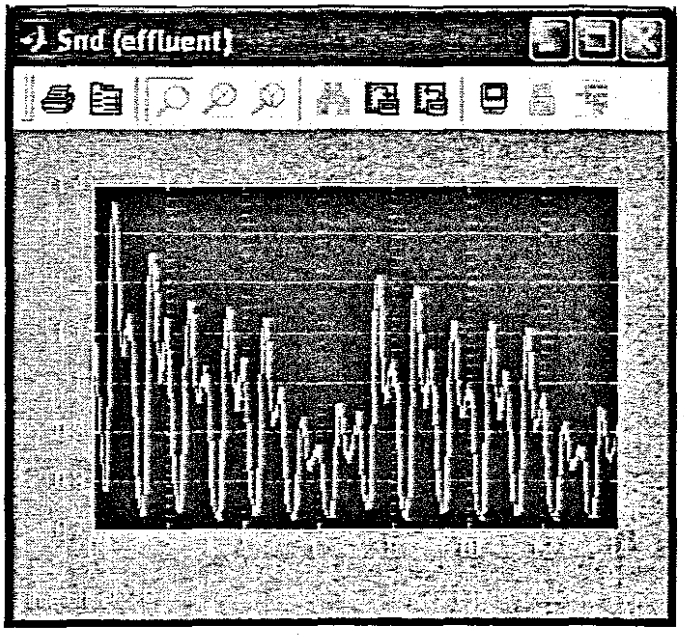


Figure 5.10: Plot of soluble organic nitrogen concentration in the effluent under controlled conditions

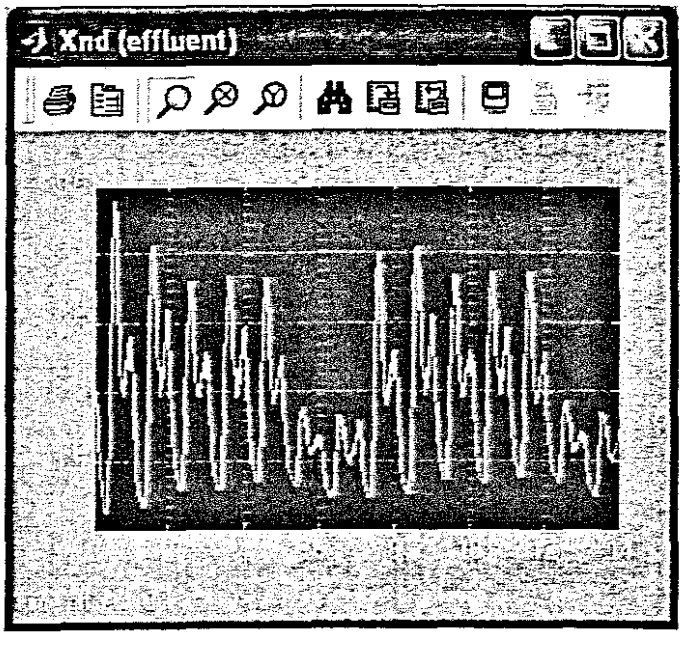


Figure 5.11: Plot of soluble organic nitrogen concentration in the effluent under controlled conditions

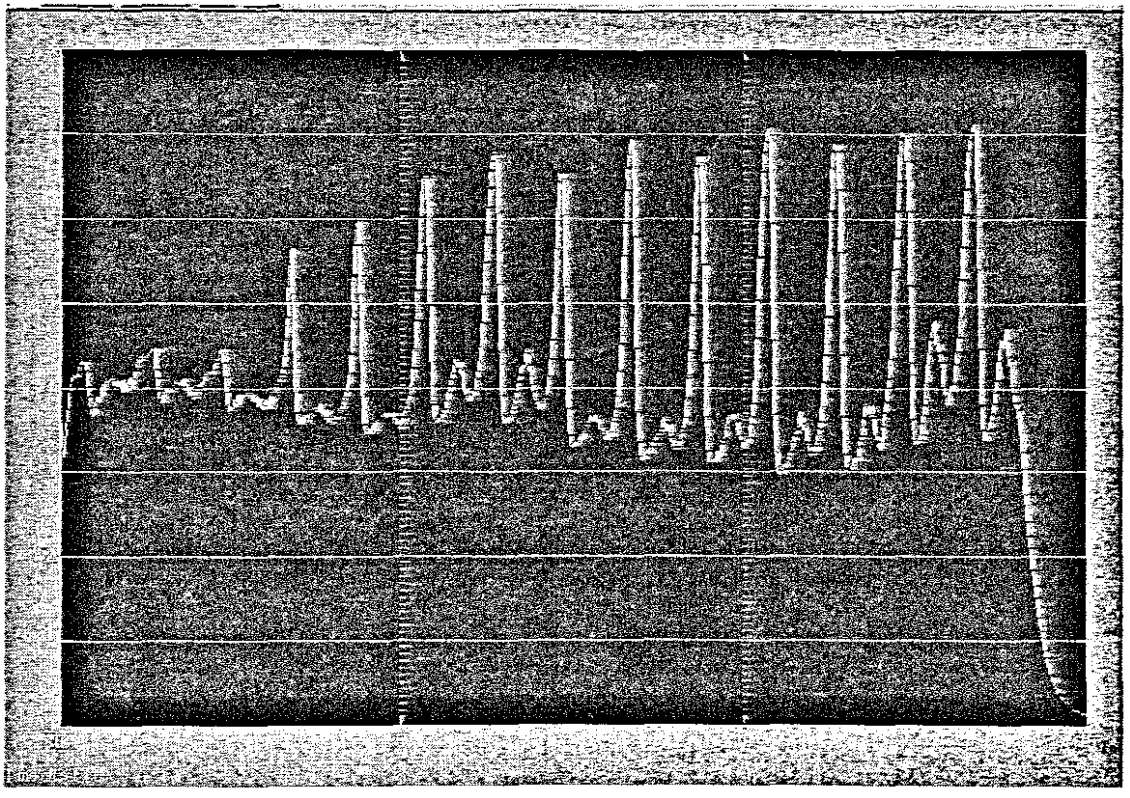


Figure 5.12: Response of dissolved oxygen concentration without control

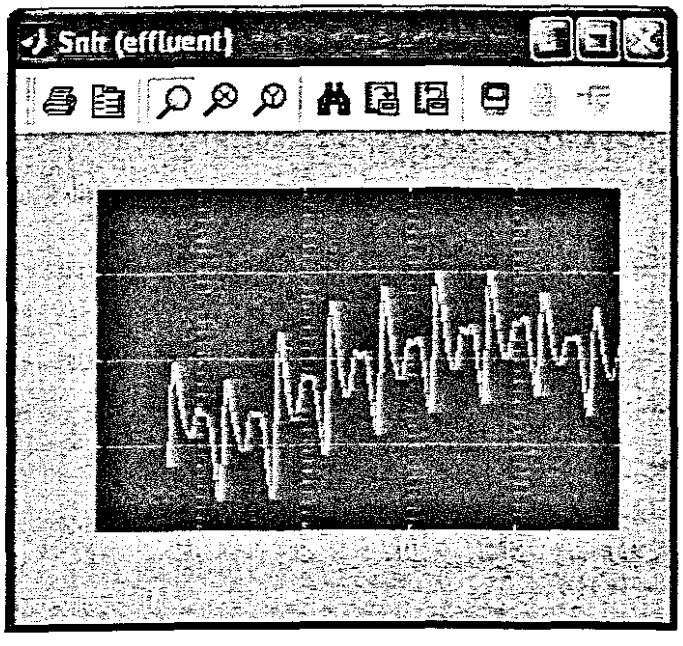


Figure 5.13: Plot of ammonia concentration in the effluent under uncontrolled conditions

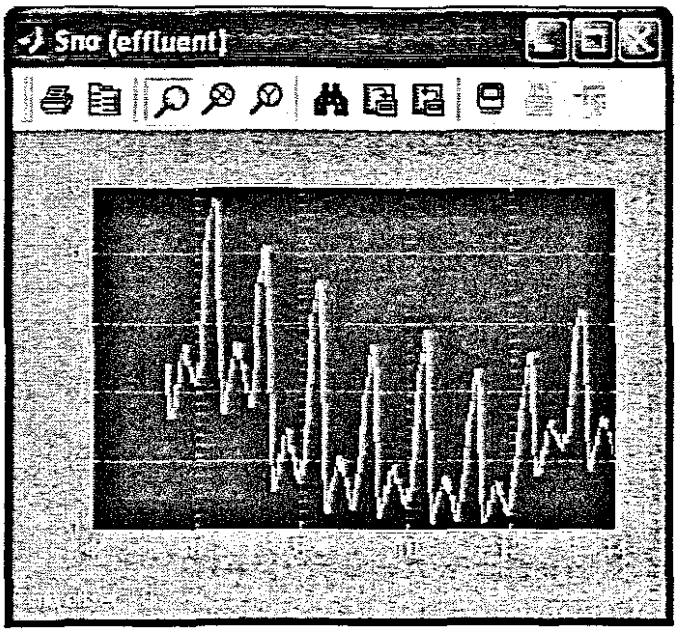


Figure 5.14: Plot of nitrate concentration in the effluent under uncontrolled conditions

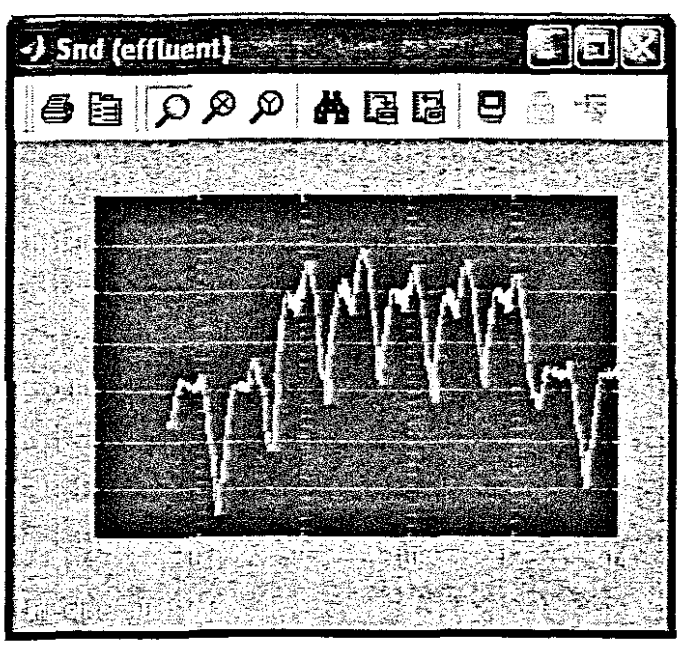


Figure 5.15: Plot of soluble organic nitrogen concentration in the effluent under uncontrolled conditions

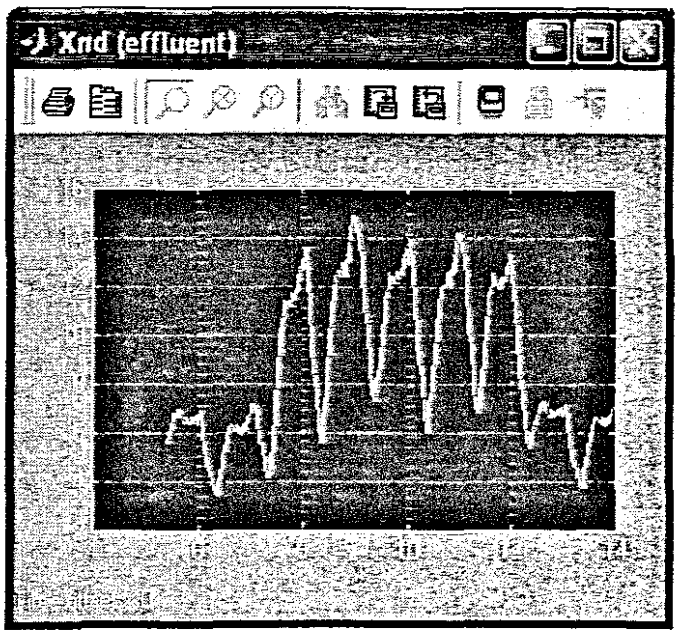


Figure 5.16: Plot of soluble organic nitrogen concentration in the effluent under uncontrolled conditions

5.7 SUMMARY OF CHAPTER

In this chapter a brief overview of a wastewater treatment plant was given with the emphasis placed upon the activated sludge plant where water is biologically treated. A description of the modeling of the ASP was given and the various components present in wastewater were described. Based upon this and the ASM1 NO1. model is described and it is shown how a simulation of the ASP was accomplished in the MATLAB/SIMULINK simulation environment. Finally the results of the implementation of the designed controller to control the setpoint of the dissolved oxygen concentration in an aerobic tank is displayed and compared with the results obtained when no control for dissolved oxygen is implemented. It has been shown that the implementation of the designed controller has yielded positive results even before it has been fine-tuned for optimal control. In the next chapter the issue of tuning and controller optimization will be investigated and implemented.

CHAPTER SIX

TUNING OF THE FUZZY CONTROLLER

This chapter deals with the aspect of fuzzy controller tuning. The tuning of fuzzy controllers has always been a topic of great debate and research because of the wide range of choice of quantities that can be used for tuning. Hence, when designing a fuzzy controller the designer has to decide upon a suitable method or technique for tuning the controller. To improve the behavior of fuzzy controllers the parameters can either be manually tuned or an optimization technique can be employed. This chapter gives a brief overview of the options available to the designer when considering optimization techniques. The method employed in this study is explained and shown how it was utilized for optimizing the fuzzy controller to control the dissolved oxygen concentration in the wastewater treatment plant.

6.1 A PROBLEM FOR TUNING OF THE FUZZY CONTROLLER

6.1.1 Fuzzy Parameters

A fundamental problem with fuzzy controller implementation is related to tuning of its parameters, which includes the placement, and shape of fuzzy sets. Other elements of the fuzzy controller such as amount of fuzzy sets also affect the behavior of the controller but these choices are included in the design stage rather than being included in the tuning stage. This essentially constitutes the major drawback of fuzzy controllers because of the large amount of parameters to be tuned. Initial approximate adjustment is particularly difficult as there is no prescribed way to accomplish this. It is also known that good convergence of optimization methods is strongly dependent on initial settings (Pivonka and Blaha,1999).

6.1.2 Non-linear and Linear Fuzzy Controllers

There exists few formal tuning methods. Fuzzy controllers are essentially characterized as nonlinear controllers with a lack of tuning methods. Because their inherent non-linearity they can pose difficulties with regards to tuning and analysis. This inherent characteristic of non-linearity stems from

- the rule base
- inference method
- the defuzzification method.

6.1.2.1 Non-linear Surface of the Fuzzy Controller

Non-linearity caused by the rule base is caused by both the fuzzy sets and the rules because when fuzzy sets are shifted and reshaped the control strategy expressed by the rules can be non linear. A plot of the control surface provides a very quick way of assessing the controller as this surface encompasses useful information about the controller and gives the designer an overall view of its parameters. By changing scaling gains the effect can be immediately noticed on the surface plot. The rippled surface present is as a result of the rules and fuzzy sets. Increasing the amount of fuzzy sets of each variable on the input and output variables will result in a decrease in ripple and amplitude. Also increasing the gain of the variable such as the output will result in an increase in the slope and hence the controllers gain. As mentioned previously adjusting the fuzzy sets will enable gain tuning of the controller in specific regions of the control space. This effect is easily noticeable from a surface plot and is characterized by different slope gradients in certain areas of the plot. Using this concept an ideal situation would be to have a very small gain when the signals are small and larger gains as the signal increases in magnitude. Hence small gains will be represented by a shallow slope and larger gains by steeper slopes. This can be achieved by adjusting the fuzzy sets of the output fuzzy variable by making the sets near the origin more closely spaced with

increasing broadness of fuzzy sets further away from the origin. This can be seen in figure 6.5 and figure 6.6 which are surface plots of two fuzzy controllers which have the same parameters for the fuzzy sets of the input linguistic variables but different parameters for the output linguistic variable theses are shown by Figures 6.1 to Figure 6.4 respectively.

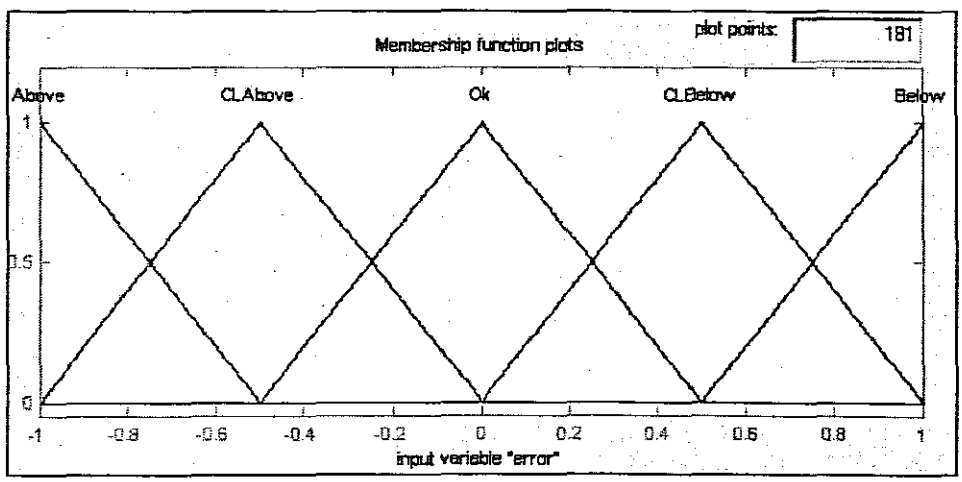


Figure 6.1: Input linguistic variable "error"

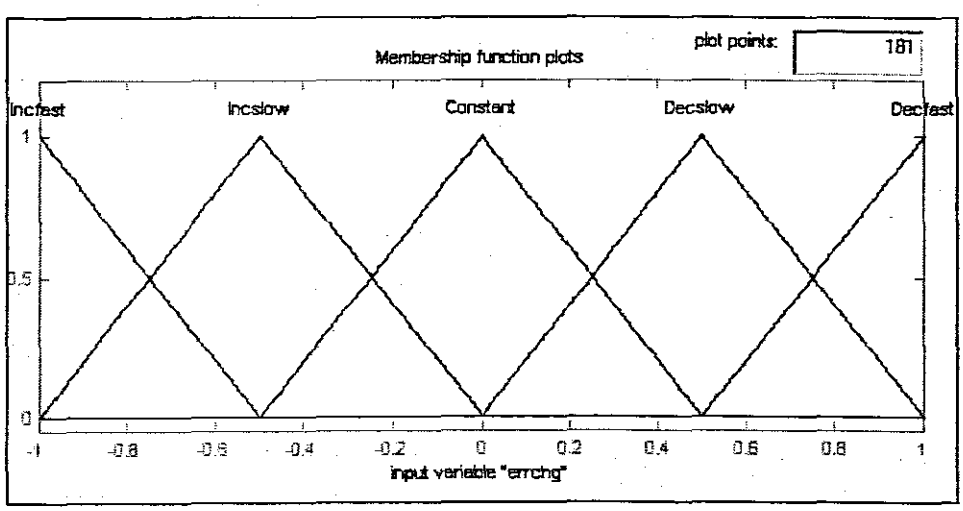


Figure 6.2: Input linguistic variable "error change"

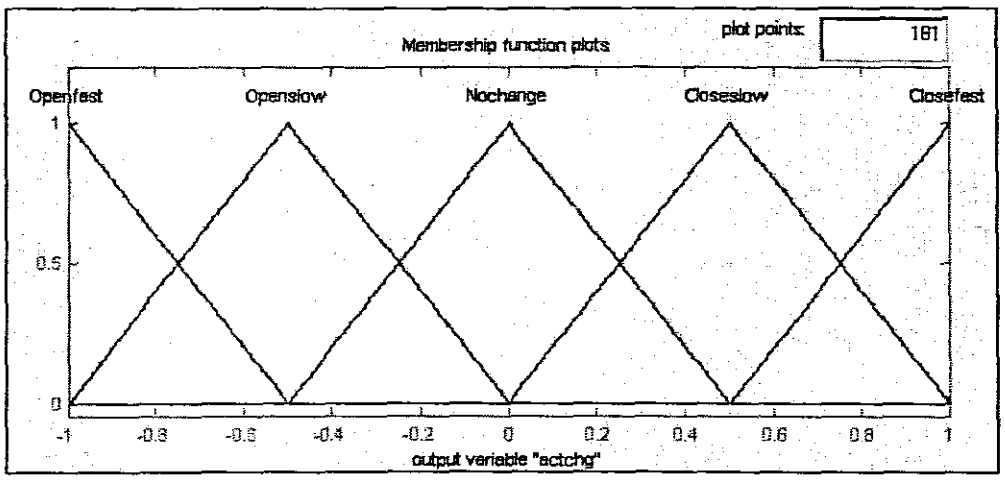


Figure 6.3: Output linguistic variable "actuator change" with all fuzzy sets Equally distributed on the universe of discourse

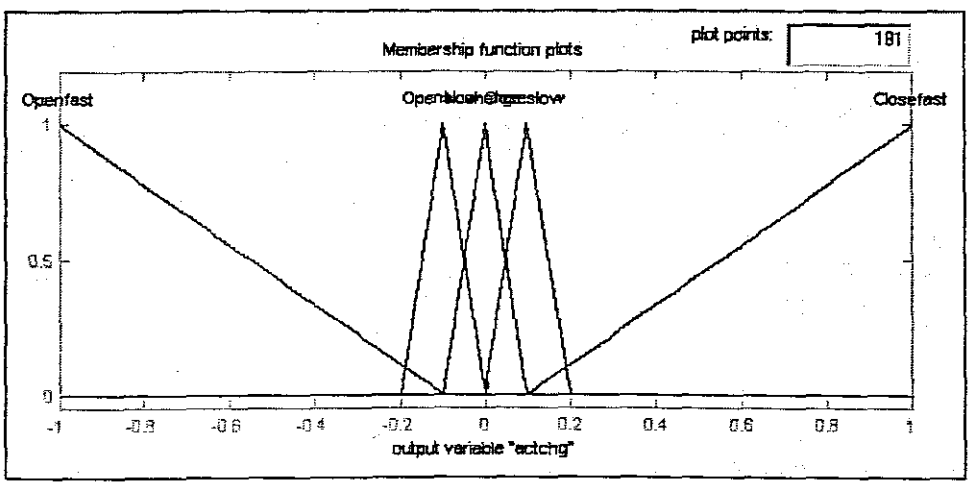


Figure 6.4: Output linguistic variable "actuator change" with contracted fuzzy sets close to the origin

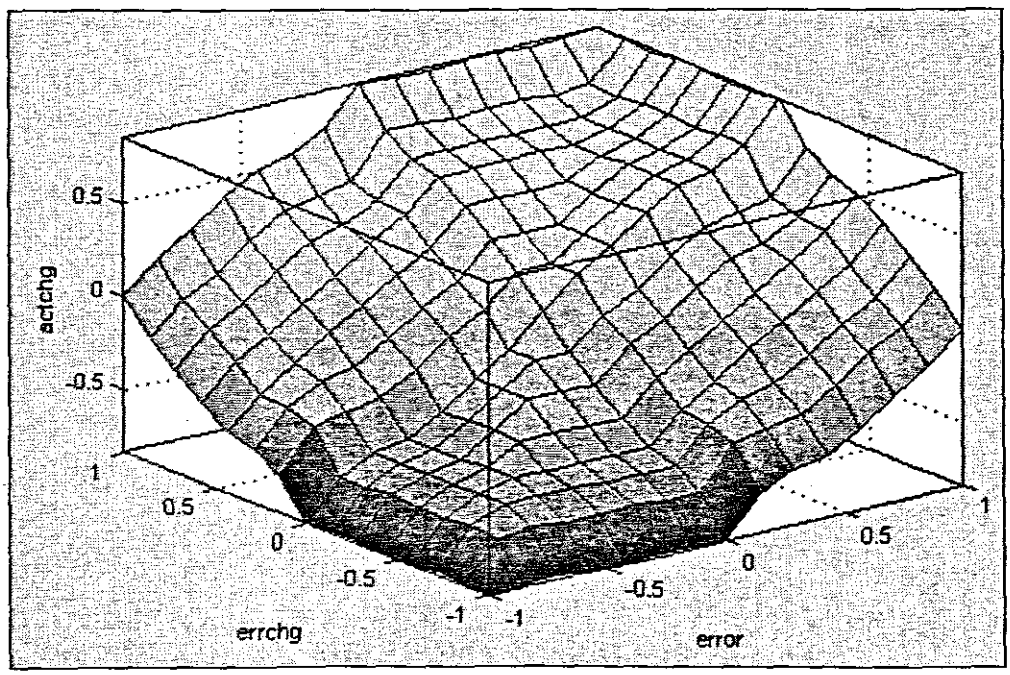


Figure 6.5 : Plot of control surface with the output as defined by the parameters in figure 6.3

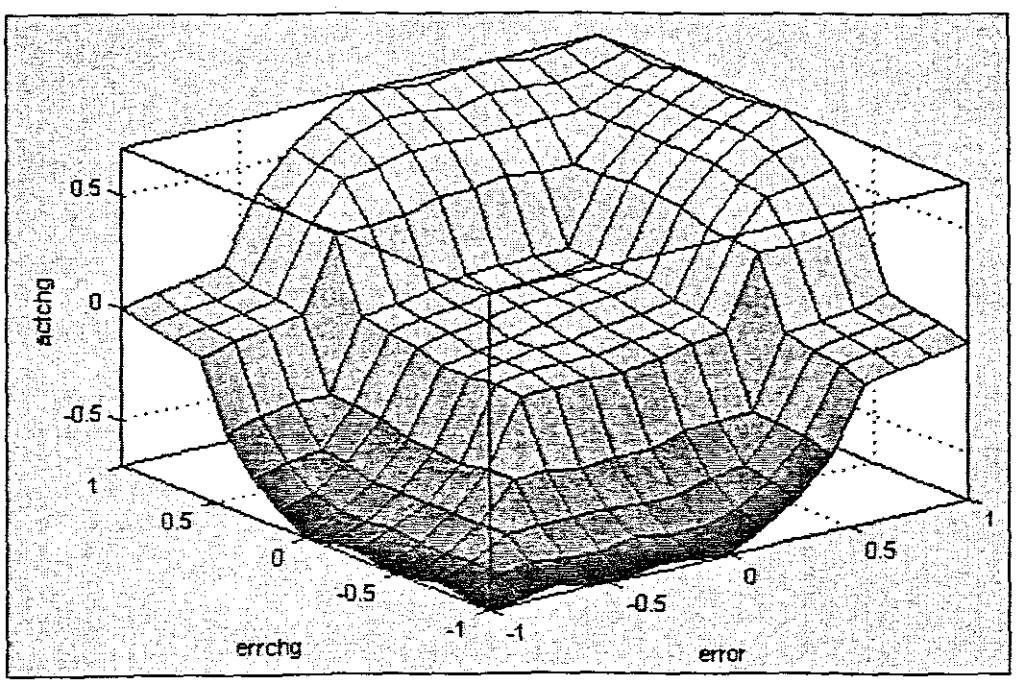


Figure 6.6 : Plot of control surface with the output as defined by the parameters in figure 6.4

When considering the inference method connectives such as 'and' and 'or' are non linear. Also defuzzification methods can be a source of non-linearity as well. Fuzzy controllers can however be designed to have linear characteristics. The following steps defines a procedure for achieving this (Jantzen, 1998)

- Fuzzy sets on the linguistic universes of the input and output variables of the controller should overlap each other at the membership value of 0.5.
- The 'and' connective should be implemented with the algebraic product.
- The rule base must be the outer 'and' product of the input families
- Singletons should be used for the output variable with their positions determined by the sum of the peak positions of the input sets.
- For defuzzification use the Center of Gravity method.

6.1.3 Techniques For Improving Fuzzy Controllers

To improve the behavior of fuzzy controllers the parameters can either be manually tuned or an optimization technique can be employed. By adjusting the placement and shape of fuzzy sets optimum values for the parameters of the fuzzy controller can be reached. Optimization can require a vast amount of time if there are many quantities to adjust in the fuzzy controller. Also optimization in many cases is not performed directly on real processes and the success of the optimization becomes heavily dependent upon the accuracy of the mathematical model of the process (Pivonka & Blaha, 1999). Amongst the different techniques employed for tuning experimentation, optimization techniques and development of corresponding conventional controllers can be employed to assist with the adjustment of the parameters of fuzzy controllers. In this chapter a brief overview of various techniques used for tuning of fuzzy controllers will be reviewed. They are

- Conventional controller techniques
- Neural fuzzy systems
- Genetic algorithm systems.

6.1.4 Overview Of Tuning Methods

6.1.4.1 Using Conventional Techniques

One technique is to derive the tuning parameters for a fuzzy controller from an equivalent conventional controller, which may be a PI, PID, or PD type. Fuzzy controllers are essentially nonlinear and more cumbersome to set controller gains compared to its conventional counterpart (Jantzen,1998). It was shown that there does exist similarities between fuzzy controllers and conventional controllers taking certain conditions into consideration(Siler&Ying,1989;Mizumoto,1992;Qiao&Mizumoto,1996;Tso&Fung,1997) .The starting point is to use a conventional controller, replace it with a linear fuzzy controller and then make the fuzzy controller nonlinear and to fine tune it. The rationale behind the technique explained above is that it offers the designer some systematic procedure to tune the fuzzy controller. By using techniques such as Ziegler-Nichols, Kappa-Tau, pole placement etc. the gains of the conventional controller can be easily set up and transferred to the equivalent fuzzy controller.

The following technique for tuning a fuzzy controller for controlling a step change in the setpoint is given below (Jantzen,1999)

- Develop a crisp controller, and tune it using known methods(Ziegler-Nichols, Kappa-Tau, optimization, hand tuning etc.)
- Insert a linear fuzzy contoller
- The parameters such as K_p, T_d and $1/T_i$ should be transferred to GE, GCE, GIE and GU using the table 6.1.
- Then insert a linear rule base
- And finally fine tune the controller using hand tuning; GE for improving rise time, GCE for damping, and GIE to remove steady state error.

where, GE is error gain, GCE is error change gain and GIE is integral gain

Fuzzy controller	K_p	$1/T_i$	T_d
P	*GE*GU*		
PI	*GCE*GCU*	*GE/GCE*	
PD	*GE*GU*		*GCE/GE*
PID	*GE*GU*	*GIE/GE*	*GCE/GE*

Table 6.1: Relationship between fuzzy and PID gains

Also to demonstrate the relationship between conventional and fuzzy controllers a research project was embarked upon at the Dresden Medical Academy in Germany in 1994 and a number of software files was written which was later updated by Victor Colazzo as a first version to run under the Matlab environment. A detailed and useful document can be found at the following website; http://www.csc.umist.ac.uk/ .

6.1.4.2 Alternative Tuning Methods

6.1.4.2.1 Neural Networks

There also exists a large amount of work on the integration of other technologies and techniques with fuzzy logic for the purposes of tuning. The integration of neural networks with fuzzy logic to form hybrid systems has been intensively investigated with Lee and Lee being one of the pioneers in this field (Lee & Lee,1975). Basically in a fuzzy neural network (Gupta & Rao,1994;Pal & Mitra,1992;Pedrycz,1992) there exists both a fuzzy system and a neural network where the fuzzy system is a fuzzy inference block converting linguistic information for the neural network or the arrangement is such that the neural network drives the fuzzy inference block. Ultimately the main idea of integration of theses two technologies is to combine the strengths of each, which results in the ability of the controller to adjust its performance by learning and accumulating experience. A plethora of information exists on work done in this field and many different neural network paradigms have been formulated (Huntsberger & Ajjimarangsee,

1990 ;Tsao et al,1994; Bezdek et al,1992;Chung & Lee,1994;Carpenter et al,1991;Carpenter et al,1993).

6.1.4.2.2 Genetic Algorithms

Genetic algorithms are modeled after the natural optimization process present in nature where the principles of natural selection, evolution, and genetics are simulated to parameterize the many variables of a fuzzy controller. One of the pioneering work in this field was done by Karr (1991). Karr's method parameterizes both the weighting of output sets as well as fuzzy set shapes. There was then an attempt to improve upon the method of Karr by including rule minimization in the search (Lee & Takagi,1993). The next improvement on this was to allow the number of rules and the size of the search space to dynamically vary during the search (Cooper & Vidal,1993). As with fuzzy neural systems mentioned above a vast amount has been done in this field and research is ongoing.

6.2 TUNING OF FUZZY SET PARAMETERS

The utilization of the techniques as described in the previous section really implies an implementation of some adaptive mechanism, where the mechanism has the principal role of altering the parameters of the controller to improve its performance so that the process output satisfies some criterion.

It was previously mentioned that tuning can be accomplished by adjusting a vast range of quantities of the fuzzy controller. It was also mentioned that there is no recommended method for tuning and the designer has the option of choosing any of the quantities for tuning. In this study the tuning procedure is concentrated upon the adjustment of the fuzzy sets of the input and output controller variables. The advantage of this is that altering of fuzzy set definitions allows for gain tuning within specific regions of the respective universes whereas tuning of scaling gains will alter the gain uniformly across the entire universe. An example of the effect it has on the control surface is shown in section 6.1.2.1. Hence the adaptive mechanism that effects a change in the scaling or

fuzzy set parameters are called self-tuning whereas the adaptive mechanism that modifies the rule base are called self-organizing where a large amount of initial research in this area was done by Mamdani and colleagues (Mamdani & Baaklini,1975;Mamdani et al ,1976;Procyk & Mamdani,1979 ; Yamazaki & Mamdani, 1982)

Tuning of fuzzy set definitions has met with objection (Mamdani,1976;Tong,1976). The argument is that fuzzy set definitions are not chosen at random but chosen to reflect the meaning of the linguistic values taken by the variables. However, research has shown that modifying fuzzy sets can provide a means for effective tuning (Nomura,Hayashi & Wakami,1991;Maeda,Someya & Funabashi,1991;Glorennec,1991;Bartolini et al,1982).

6.3 TUNING OF THE DESIGNED FUZZY CONTROLLER

Finally the technique implemented is a fairly simple iterative process that does not make use of any of the techniques described in section 6.1.4.2 above. This was done, as it was not required to implement a hybrid system (neuro-fuzzy or genetic algorithms and fuzzy logic integrated) to optimize the parameters of the fuzzy controller. The algorithm for the tuning procedure is based upon the work done by Bartolini and co-authors (Bartolini et al,1982). The algorithm modifies the parameters of the fuzzy sets based upon a number of specified performance measures such as the average steady state error. The algorithm employed is concerned with improving the controller's ability to track the setpoint as close as possible. In this algorithm there are essentially three performance indices of concern. They are the average error and maximum and minimum error. Figure 6.7 shows a flowchart of the algorithm The controller should try to maintain the setpoint in the region of +- 2% of the desired setpoint and it has been found that the performance indices stated above are sufficient for this purpose.

6.3.1 Description Of Procedure For Tuning

The algorithm dictates the tuning procedure in the following manner.

- Compute the average error, maximum and minimum error
- Determine whether the average error is within the specified limit indicated by α
- If it is not within the specified limit determine whether the average value is positive or negative
- If it is positive perform ACTION A or if it is negative perform ACTION B and then repeat the process.
- If the average error is within the specified limit check whether the max error and minimum error is within the specified limit.
- If the maximum error is not within the limit perform ACTION D and then check minimum error
- If the minimum error is not within the limit perform ACTION C, then repeat the sequence. If the minimum error is within the limit END the tuning procedure.

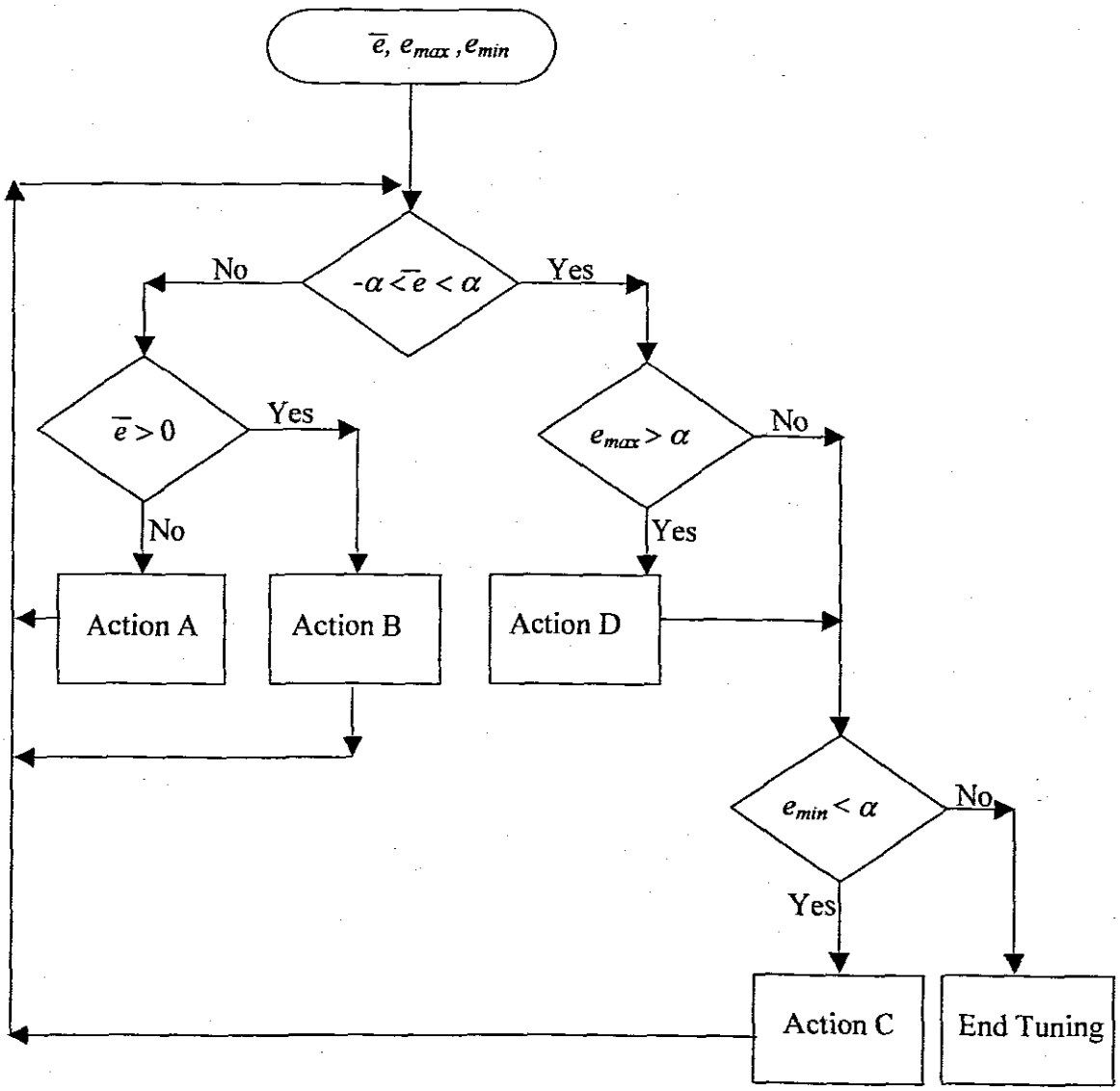


Figure 6.7: Flowchart of the algorithm employed or tuning

6.3.2 Description of the Tuning Actions

With reference to Figure 6.7 the tuning actions are described as follows

- If the average error is too large then the fuzzy sets are modified as depicted in the figures below.

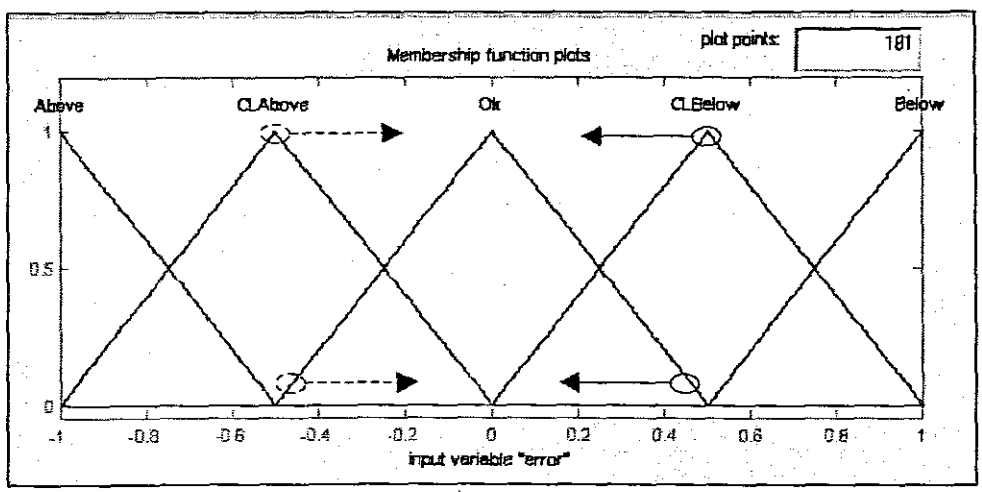


Figure 6.8: Tuning action A and action B

Action A

Action B

- If the error is predominantly negative then fuzzy sets CLAbove and Ok are modified. CLAbove will have its apex moved towards the origin and Ok will have its left side moved towards the origin. This is indicated by the dashed arrow in figure 6.8 and is ACTION A. The effect that this has is to increase the degree to which the error values are recognized as negative.

- If the error is predominantly positive then fuzzy sets CLBelow and Ok are modified. CLBelow will have its apex moved towards the origin and Ok will have its right side moved towards the origin. This is indicated by the solid arrow in figure 6.8 and is ACTION B. The effect that this has is to increase the degree to which the error values are recognized as positive.

- Now once the average error is within the acceptable range the maximum and minimum error has to be checked. If it is exceeds the tolerance as stipulated by alpha (α) above, then the fuzzy sets on the extreme ends are modified.

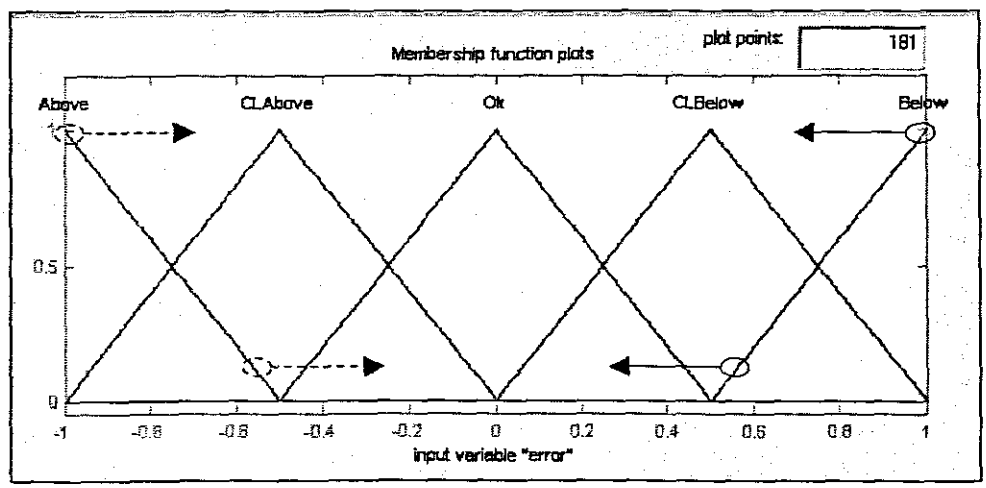


Figure 6.9: Tuning action C and action D

Action C

Action D

- With ACTION C the fuzzy set Above has its apex and rightmost base parameter moved closer towards the origin. This action further helps to increase the degree to which error values are recognized as negative. With ACTION D the fuzzy set Below has its apex and leftmost base parameter moved closer towards the origin. This action further helps to increase the degree to which error values are recognized as positive.
- The tuning action is stopped if the maximum and minimum error values are within the specified tolerance range.

The tuning algorithm is an iterative process and the parameters of the fuzzy sets are modified by small incremental changes at a time as large changes can cause oscillations

to occur. It was found that changes need only be made to the linguistic variable error. The output fuzzy set was manually adjusted according to the description above. With only one variable to tune the optimization procedure can be executed much quicker than tuning of three variables.

6.3.3 Description of Tuning Algorithm in the MATLAB environment

Below is a description of the MATLAB code for the tuning procedure. It is essentially comprised of three sections.

1. Fuzzy controller generation
2. Simulation of the process
3. Parameter tuning

```
load inputfile;                 /* loading of diurnal pattern as input to system

tel=1;
tol=0.5;
tn=0.5;
tp=0.5;                         generic parameters for the fuzzy controller
tor=0.5;
ter=1;
diss7nov;                       /* calling of m-file to generate fuzzy controller

options = simset('FinalStateName','filename','InitialState',[State]);   run simulation
[t,x,y]=sim('thebiopidinputfile1withfuzzy',[tx ty],options);

avg=mean(err);                  Calculation of average error , minimum and maximum
mx=max(err);                    error
mn=min(err);
```

```
while(mx > 0.04),
if  0.04 > abs(avg),
   mx = max(err);
   mn = min(err);
   if mx > 0.04,
      tor = 0;
      ter = ter-0.05;
   end

   if mn < -0.04,
      tol = 0;
      tel = tel-0.05;
   end
```

Checking of average , minimum and maximum error values and then adjusting of fuzzy parameter variables if performance indices are not satisfied.

```
else
   if avg < 0,
      tn = abs(avg);
   end
   if avg > 0;
     tp = avg;
   end
end
```

Checking of average , minimum and maximum error values and then adjusting of fuzzy parameter variables if performance indices are not satisfied.

```
   diss7nov;                    /*Generation of fuzzy controller with new parameters
   options = simset('FinalStateName','filename','InitialState',[State]);
   [t,x,y]=sim('thebiopidinputfile1withfuzzy',[tx ty],options);
   avg=mean(err);
   mx=max(err);
   mn=min(err);
end
```

Simulation rerun to test updated controller

6.3.4 Description of The Algorithm for Fuzzy Controller Generation

Below is the algorithm that generates the fuzzy controller. This portion of code is called by the main tuning algorithm described in section 6.3.3 with the filename "diss7nov". This is an important portion of the tuning code as it has to accept the new parameters for the fuzzy sets and modify them accordingly

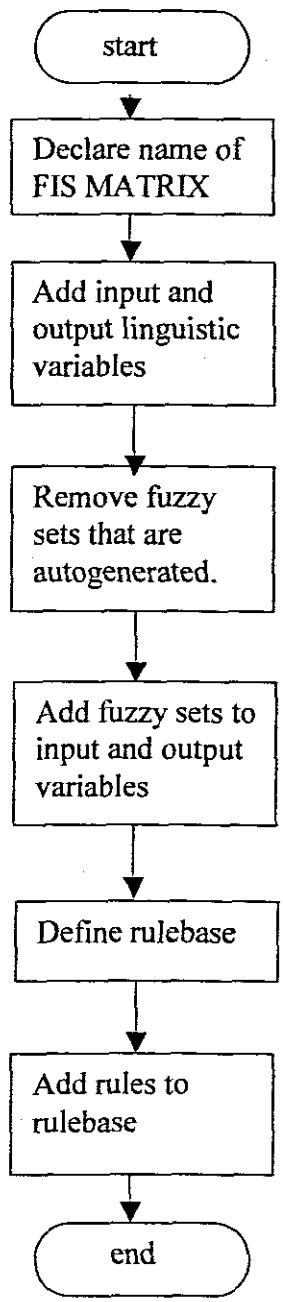


Figure 6.10: Flowchart depicting fuzzy controller generation in MATLAB

In figure 6.10 it can be seen that a portion of code in the algorithm has to remove fuzzy sets that are auto generated. If this is not done then the user defined fuzzy sets that are added to the system will overlap with the auto generated fuzzy sets causing the structure of the linguistic variable to be corrupted. Below is the code for fuzzy controller generation.

```
%Generation of Fuzzy Inference System (FIS)
a=newfis('do');
%Addition of input and output variables
a=addvar(a,'input','error',[-1 1]);
a=addvar(a,'input','errchg',[-1 1]);
a=addvar(a,'output','actchg',[-1 1]);
%In this version of Matlab the fuzzy sets that are auto generated should be removed
a=rmmf(a,'input',1,'mf',1);
a=rmmf(a,'input',1,'mf',2);
a=rmmf(a,'input',1,'mf',1);
a=rmmf(a,'input',2,'mf',1);
a=rmmf(a,'input',2,'mf',2);
a=rmmf(a,'input',2,'mf',1);
a=rmmf(a,'output',1,'mf',1);
a=rmmf(a,'output',1,'mf',2);
a=rmmf(a,'output',1,'mf',1);
%Addition of fuzzy sets for input variable NO.1
a=addmf(a,'input',1,'Above','trimf', [-1.5 -tel -tol]);
```

```
a=addmf(a,'input',1,'CLAbove','trimf',[-1 -tn 0]);

a=addmf(a,'input',1,'Ok','trimf',[-tn 0 tp]);

a=addmf(a,'input',1,'CLBelow','trimf', [0 tp 1]);

a=addmf(a,'input',1,'Below','trimf', [tor ter 1.5]);
```

%Addition of fuzzy sets for input variable NO.2

```
a=addmf(a,'input',2,'Incfast','trimf', [-1.5 -1 -0.5]);

a=addmf(a,'input',2,'Incslow','trimf',[-1 -0.5 0]);

a=addmf(a,'input',2,'Constant','trimf', [-0.5 0 0.5]);

a=addmf(a,'input',2,'Decslow','trimf', [0 0.5 1]);

a=addmf(a,'input',2,'Decfast','trimf', [0.5 1 1.5]);
```

%Addition of fuzzy sets for output variable NO.1

```
a=addmf(a,'output',1,'Openfast','trimf', [-1.5 -1 -0.1]);

a=addmf(a,'output',1,'Openslow','trimf',[-0.2 -0.1 0]);

a=addmf(a,'output',1,'Nochange','trimf', [-0.1 0 0.1]);

a=addmf(a,'output',1,'Closeslow','trimf', [0 0.1 0.2]);

a=addmf(a,'output',1,'Closefast','trimf', [0.1 1 1.5]);
```

%Definition of Rules in indexed format

```
rulelist=[1 1 1 1 1;1 2 1 1 1;1 3 1 1 1;1 4 2 1 1;1 5 3 1 1;21 1 1 1;2 2 2 1 1;2 3 2 1 1;
  2 4 3 1 1;2 5 4 1 1;3 1 1 1 1;3 2 2 1 1;3 3 3 1 1;3 4 4 1 1;3 5 5 1 1;4 1 2 1 1;
  4 2 3 1 1;4 3 4 1 1;4 4 4 1 1;45 5 1 1;5 1 3 1 1;5 2 4 1 1;5 3 5 1 1;5 4 5 1 1;
  5 5 5 1 1];
```

%Add Rules to Rulebase

```
a = addrule(a,rulelist);
```

6.3.5 Simulation Procedure

The procedure of tuning is repetitive during the time horizon of the process and the time intervals were selected to be one day as this time period is adequate for such a slow process. Also it was shown that the optimized parameters were well determined to compensate for the daily disturbances input from the diurnal pattern. During the tuning procedure the simulation was performed with command line directives as this offered greater flexibility. Following is the code that sets the conditions and runs the simulation.

```
options = simset('FinalStateName','filename','InitialState',[State]);
[t,x,y]=sim('thebiopidinputfile1withfuzzy',[tx ty],options);
```

The "options" part of the code is used to setup the initial simulation conditions and to save the final state of the simulation. The final state that is saved is used as the initial conditions for the next period in the tuning procedure.

The "sim" function then runs the actual simulation with the selected options and the time interval set by the variables "tx" and "ty" in this function.

6.4 RESULTS FROM THE TUNING PROCEDURE

As indicated section 6.3.5 the time interval is one day. The first time period is regarded as the startup period until steady state conditions are reached. From Figure 6.11 it can be seen that steady state conditions are reached after approximately 7 hours with slight variations about the setpoint value of 2mg/l. The diurnal pattern is over a period of 14 days and hence the controller parameters will vary each day. These parameters are shown in Tables 6.2 through Table 6.15. The results for the optimized controller for each day over the 14-day period is shown in Figure 6.11 through Figure 6.52.

6.4.1 Results for the Period day 0 to day 1

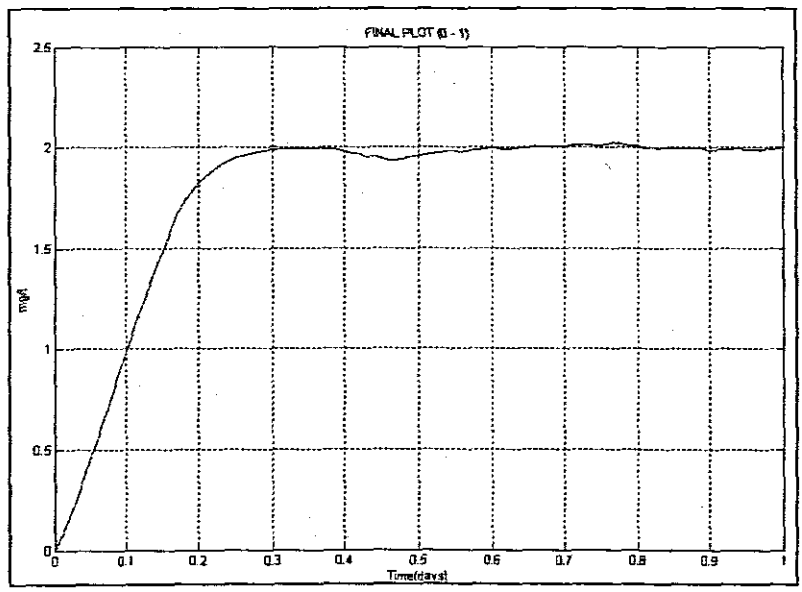


Figure 6.11: Final plot of dissolved oxygen concentration value for the period day 0 to day 1.

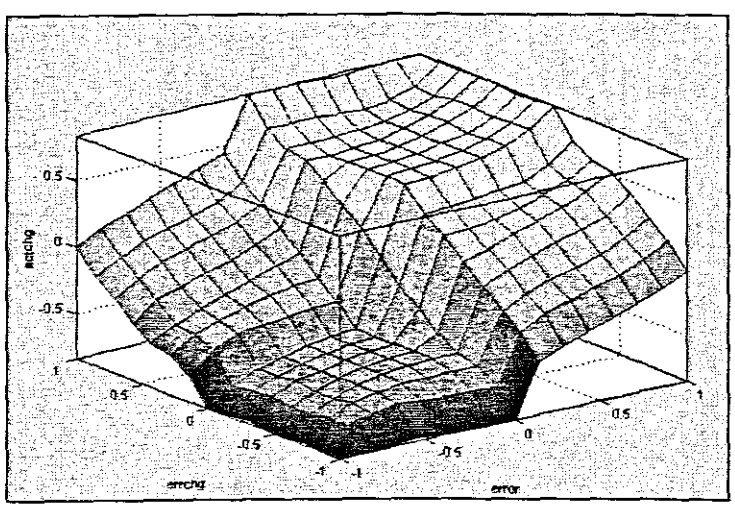


Figure 6.12: Plot of fuzzy controller surface for the period day 0 to day 1

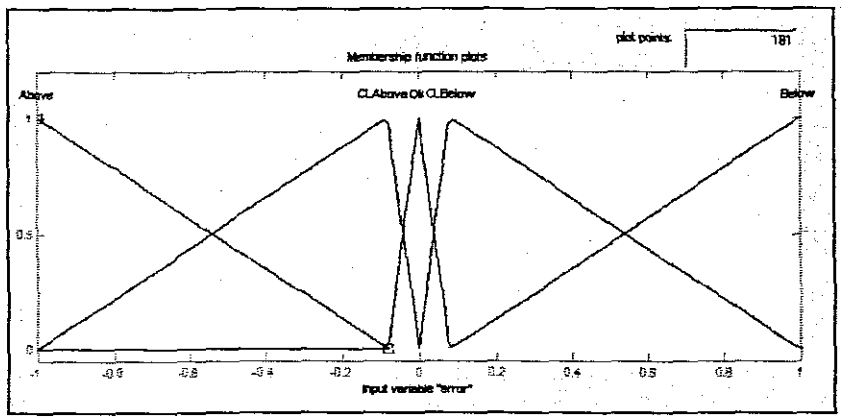


Figure 6.13: Plot of tuned fuzzy sets of "error" linguistic variable for period day 0 to day 1

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -1 -0.08 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.08 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.08 0 0.08 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.08 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0.08 1 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.2: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 0 to day 1

6.4.2 Results for the Period day 1 to day 2

Although the plot in figure 6.14 looks very oscillatory it should be observed that the plot shows a zoomed in view and that the maximum deviation from the setpoint is approximately 0.04, which is negligible. This is the case for all subsequent plots of the dissolved oxygen concentration.

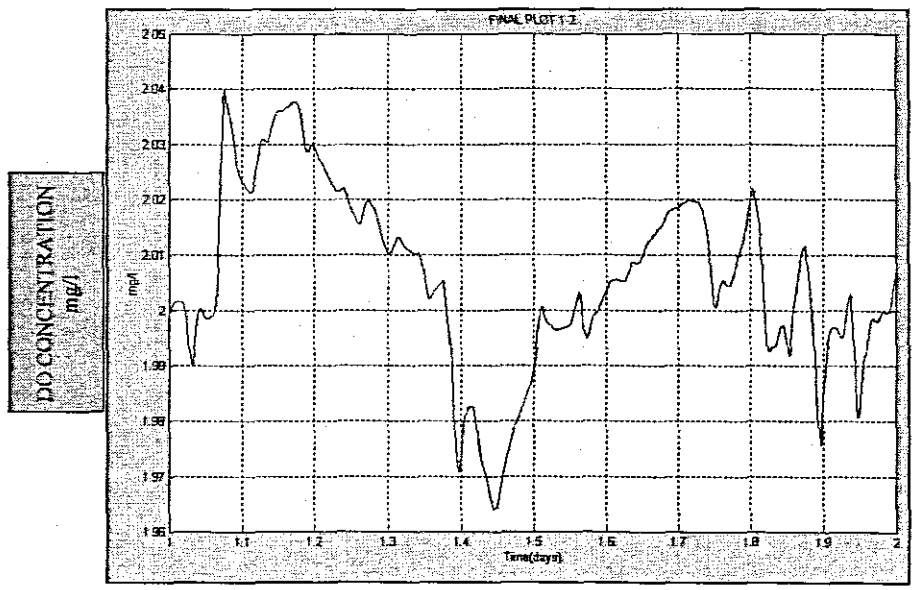


Figure 6.14: Final plot of dissolved oxygen concentration value for the period day 1 to day 2

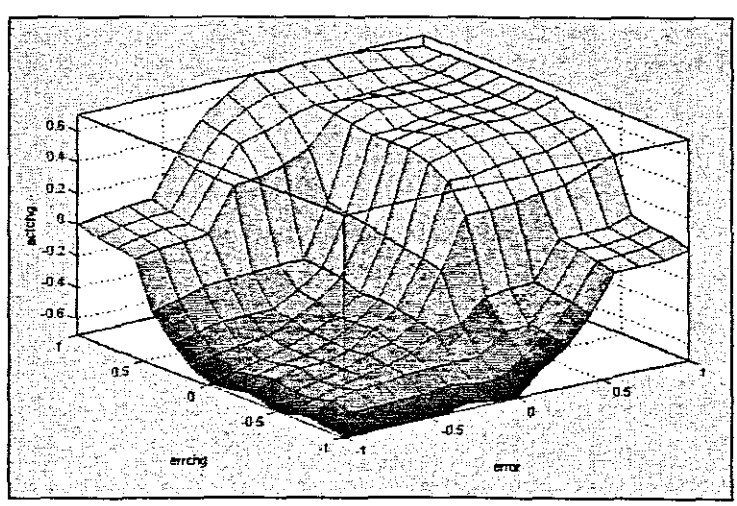


Figure 6.15: Plot of fuzzy controller surface for the period day 1 to day 2

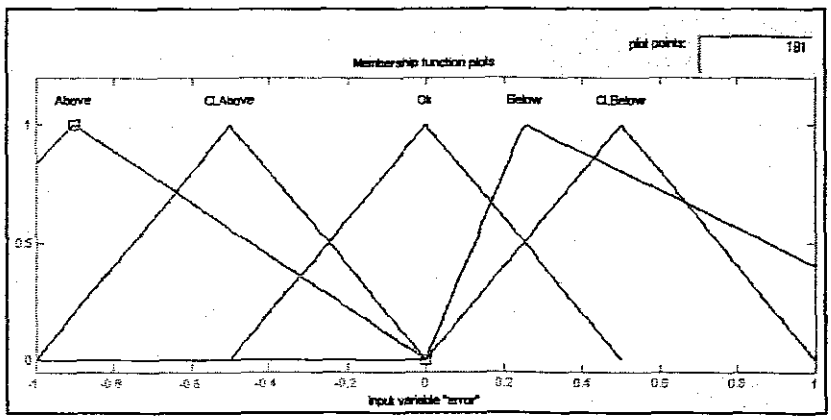


Figure 6.16: Plot of tuned fuzzy sets of "error" linguistic variable for period day1 to day 2

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.9 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.5 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.5 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.25 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.3: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 1 to day 2

Figure 6.14 to Figure 6.16 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.3 Results for the Period day 2 to day 3

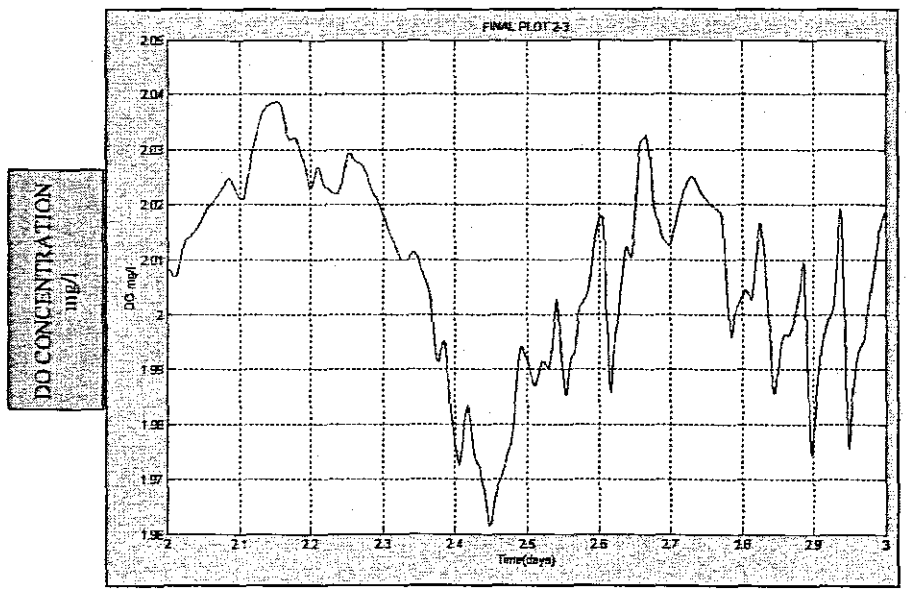


Figure 6.17: Final plot of dissolved oxygen concentration value for the period day 2 to day 3

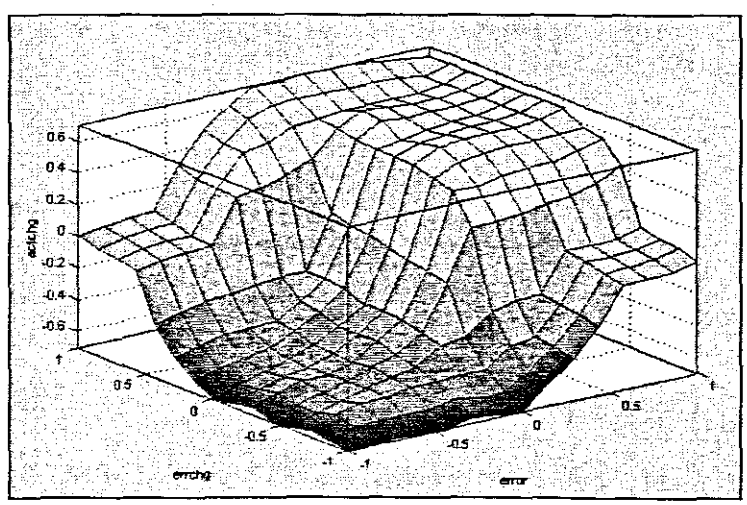


Figure 6.18: Plot of fuzzy controller surface for the period day 2 to day 3

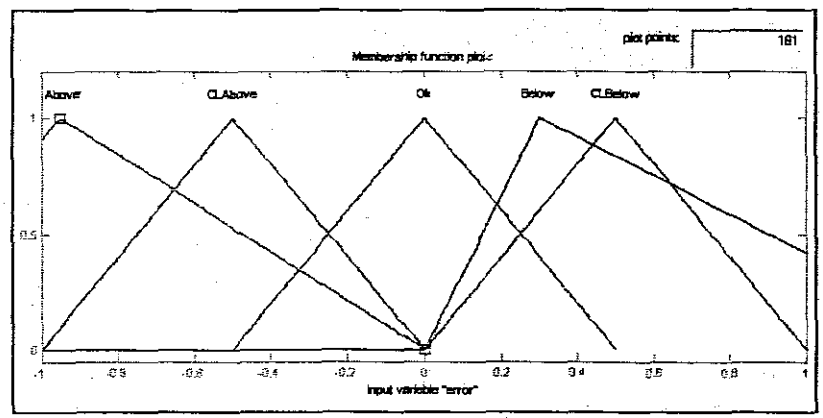


Figure 6.19: Plot of tuned fuzzy sets of "error" linguistic variable for period day 2 to day 3

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.95 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.5 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.5 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.3 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.4: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 2 to day 3

Figure 6.17 to Figure 6.19 show the optimized trajectory, the controller surface, and the graphical display of the controller parameters respectively.

6.4.4 Results for the Period day 3 to day 4

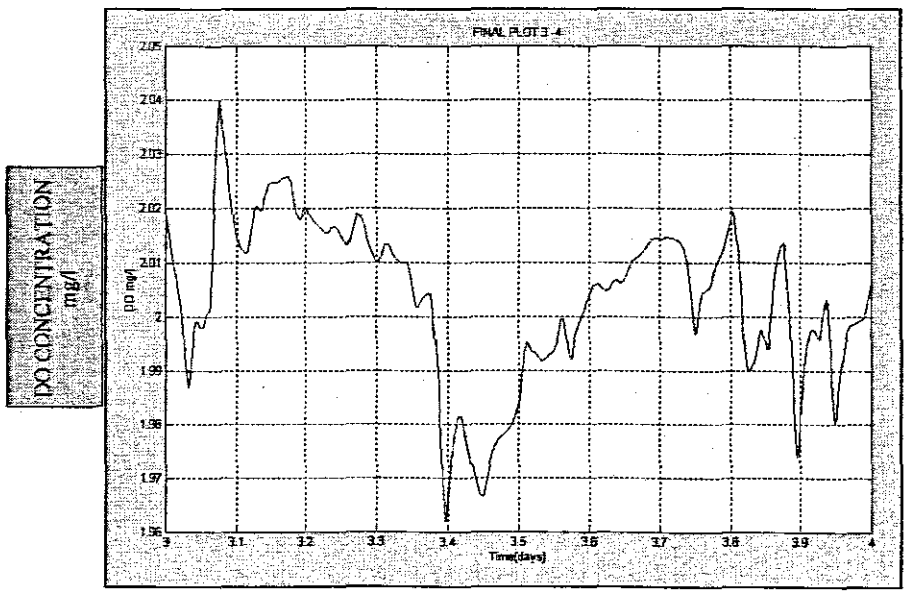


Figure 6.20: Final plot of dissolved oxygen concentration value for the period day 3 to day 4

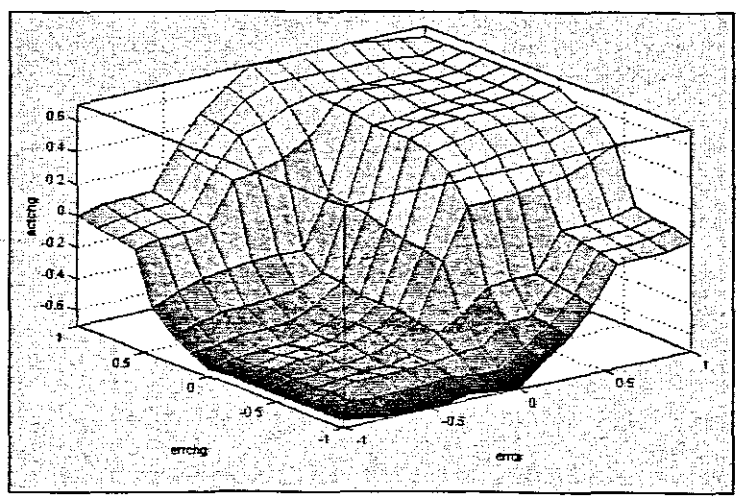


Figure 6.21: Plot of fuzzy controller surface for the period day 3 to day 4

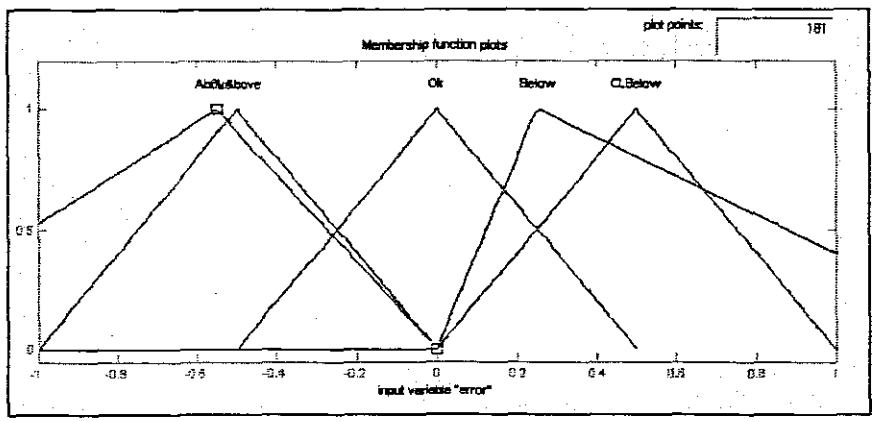


Figure 6.22: Plot of tuned fuzzy sets of "error" linguistic variable for period day 3 to day 4

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.55 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.5 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.5 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.25 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.5: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 3 to day 4

Figure 6.20 to Figure 6.22 show the optimized trajectory, the controller surface, and the graphical display of the controller parameters respectively.

6.4.5 Results for the Period day 4 to day 5

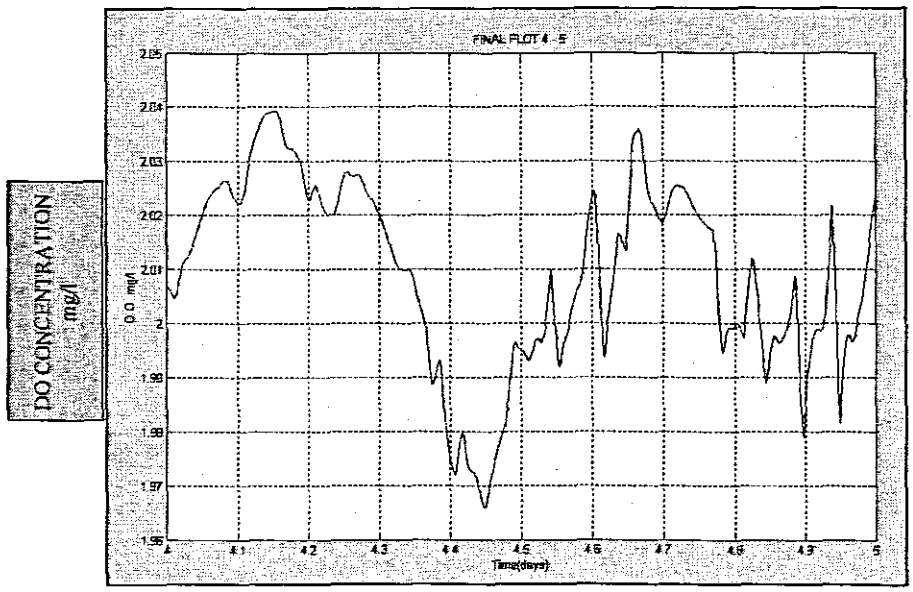


Figure 6.23: Final plot of dissolved oxygen concentration value for the period day 4 to day 5

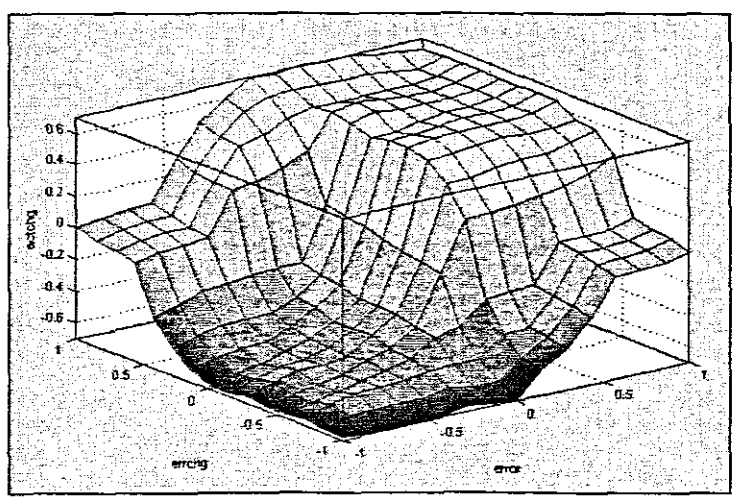


Figure 6.24: Plot of fuzzy controller surface for the period day 4 to day 5

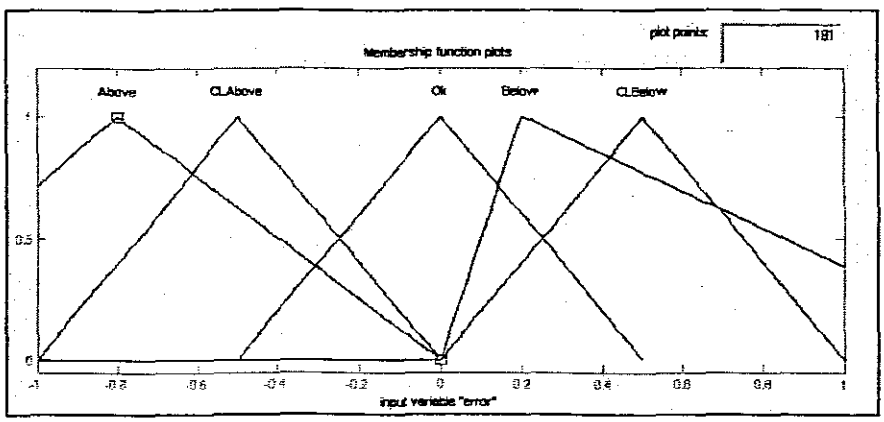


Figure 6.25: Plot of tuned fuzzy sets of "error" linguistic variable for period day 4 to day 5

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.8 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.5 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.5 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.2 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.6: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 4 to day 5

Figure 6.23 to Figure 6.25 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.6 Results for the Period day 5 to day 6

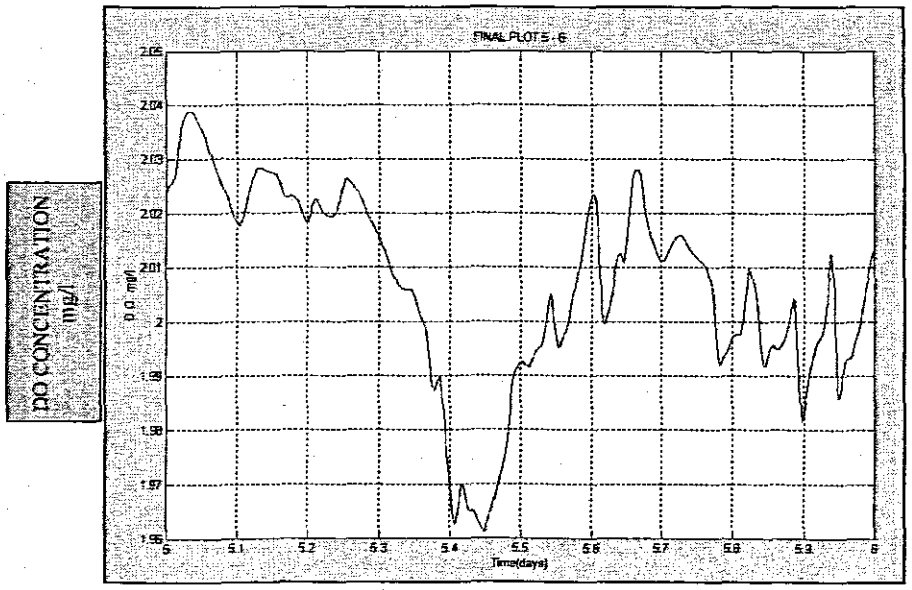


Figure 6.26: Final plot of dissolved oxygen concentration value for the period day 5 to day 6

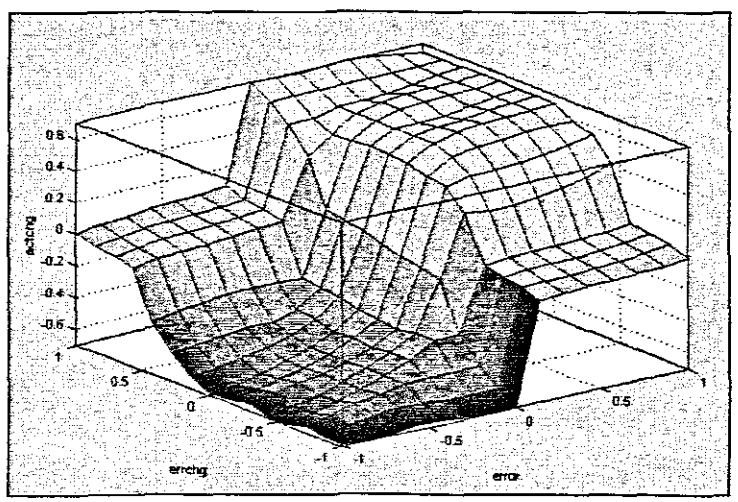


Figure 6.27: Plot of fuzzy controller surface for the period day 5 to day 6

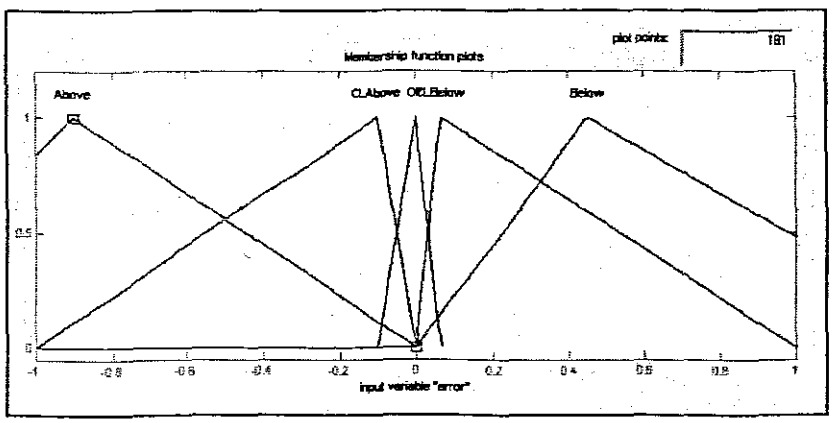


Figure 6.28: Plot of tuned fuzzy sets of "error" linguistic variable for period day 5 to day 6

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.9 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.09757 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.09757 0 0.0638 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.0638 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.45 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.7: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 5 to day 6

Figure 6.26 to Figure 6.28 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.7 Results for the Period day 6 to day 7

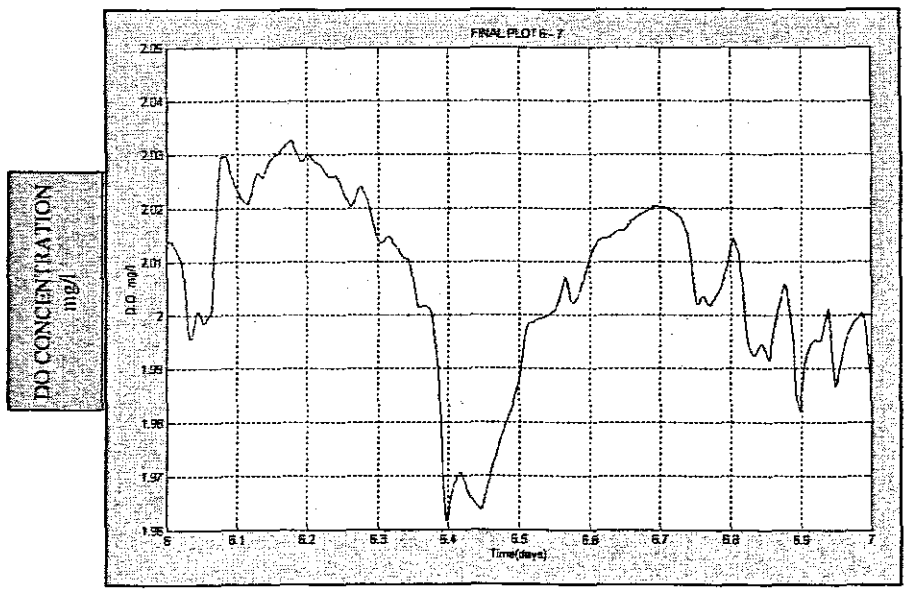


Figure 6.29: Final plot of dissolved oxygen concentration value for the period day 6 to day 7

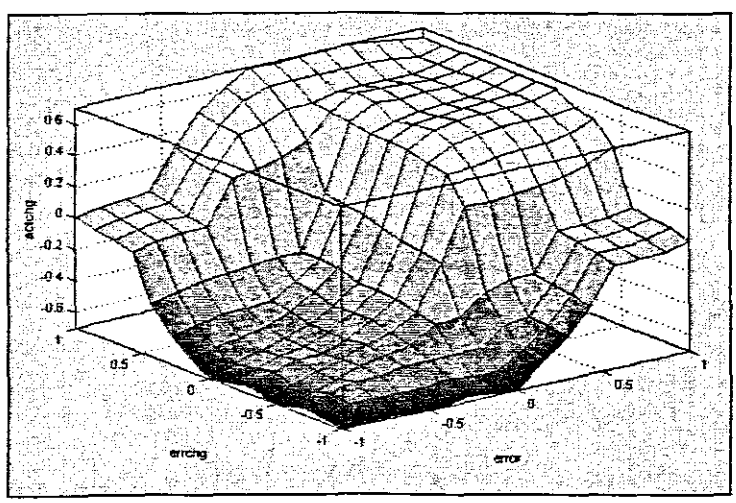


Figure 6.30: Plot of fuzzy controller surface for the period day 6 to day 7

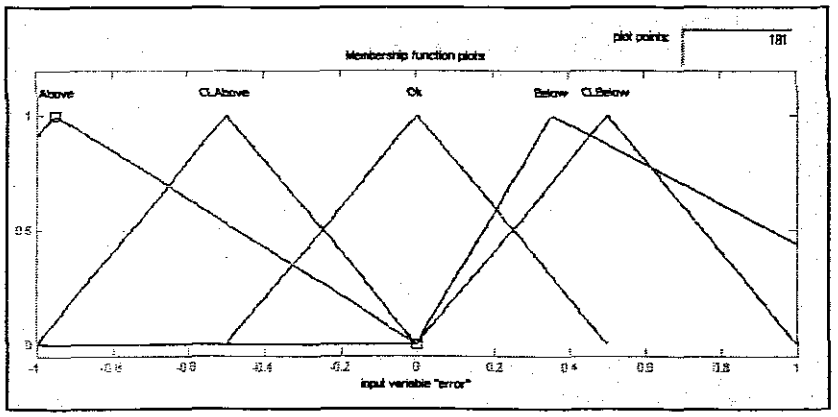


Figure 6.31: Plot of tuned fuzzy sets of "error" linguistic variable for period day 6 to day 7

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.95 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.5 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.5 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.35 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.8: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 6 to day 7

Figure 6.29 to Figure 6.31 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.8 Results for the Period day 7 to day 8

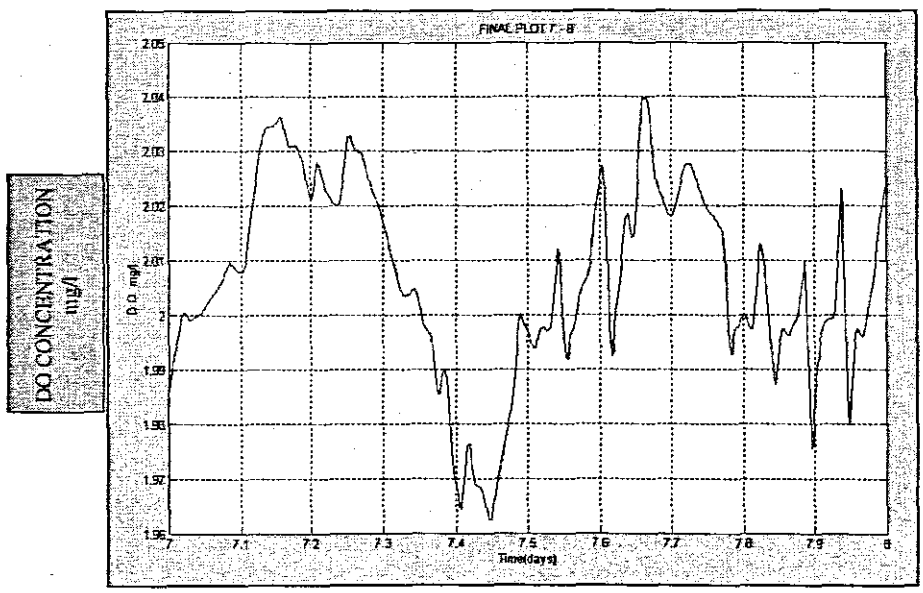


Figure 6.32: Final plot of dissolved oxygen concentration value for the period day 7 to day 8

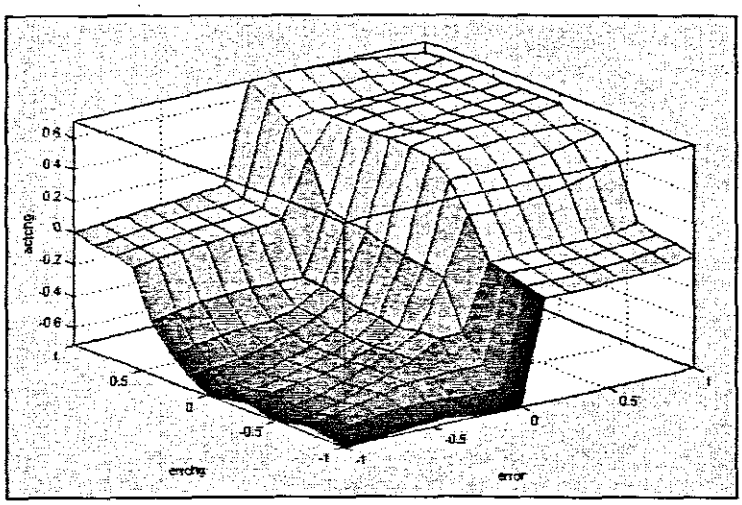


Figure 6.33: Plot of fuzzy controller surface for the period day 7 to day 8

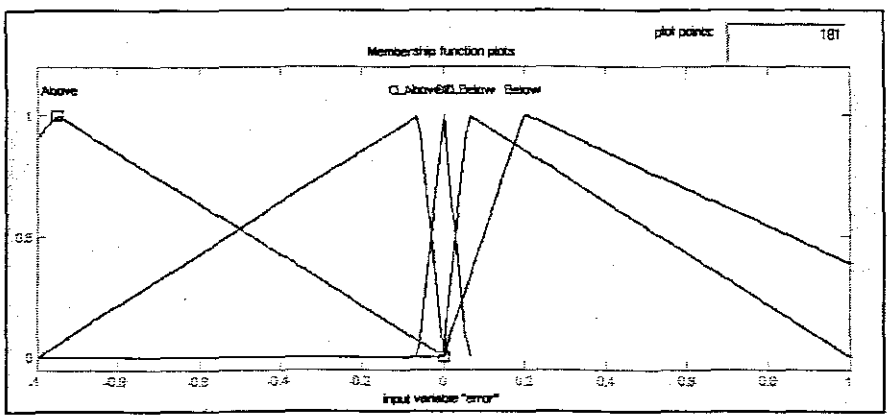


Figure 6.34: Plot of tuned fuzzy sets of "error" linguistic variable for period day 7 to day 8

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.95 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.0605 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.0605 0 0.06027 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.06027 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.2 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.9: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 7 to day 8

Figure 6.32 to Figure 6.34 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.9 Results for the Period day 8 to day 9

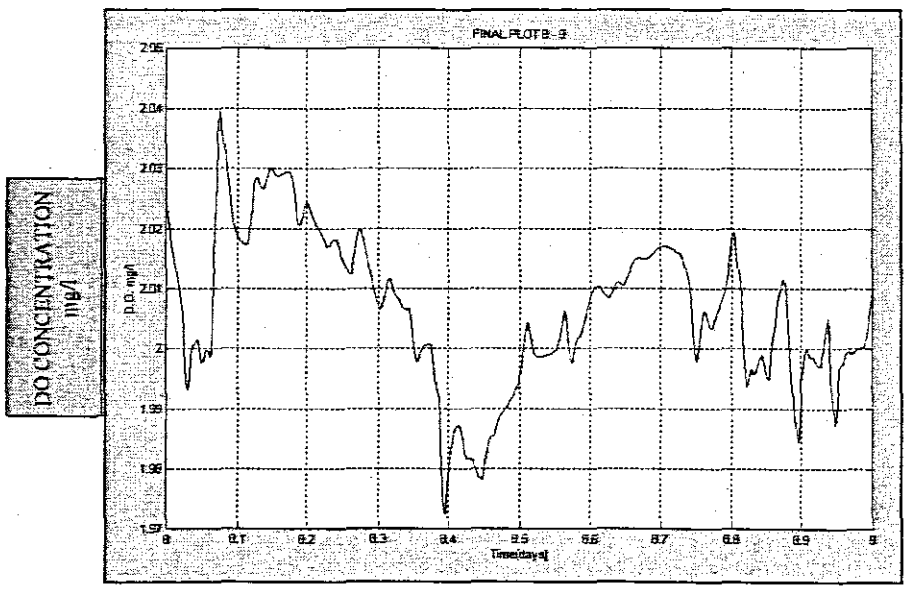


Figure 6.35: Final plot of dissolved oxygen concentration value for the period day 8 to day 9

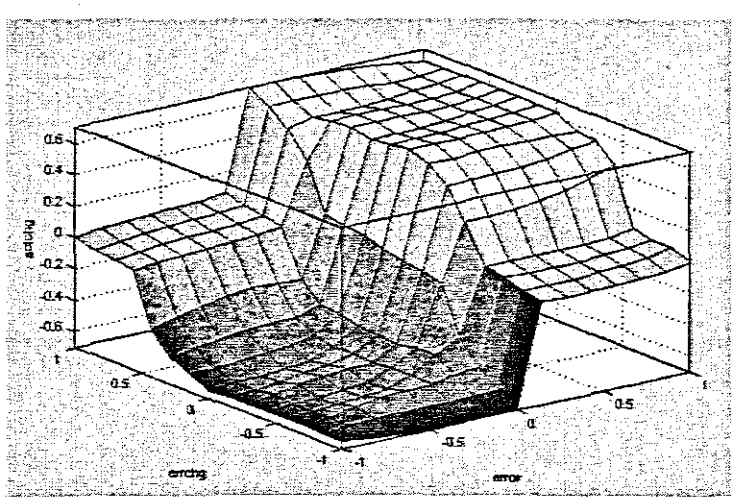


Figure 6.36: Plot of fuzzy controller surface for the period day 8 to day 9

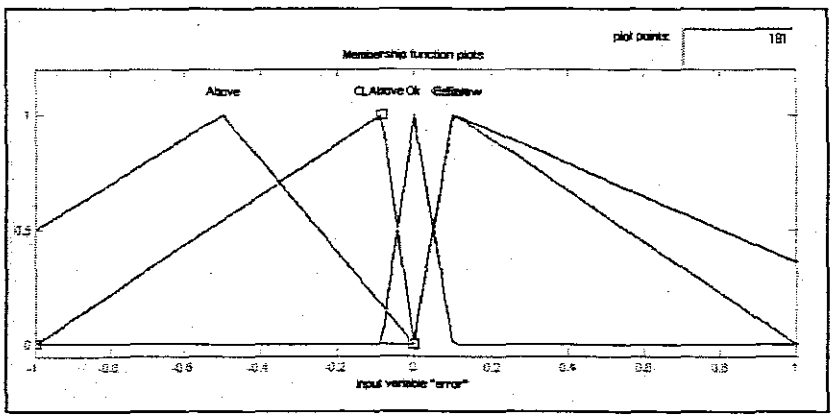


Figure 6.37: Plot of tuned fuzzy sets of "error" linguistic variable for period day 8 to day 9

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.5 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.08501 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.08501 0 0.1016 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.1016 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.1 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.10: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 8 to day 9

Figure 6.35 to Figure 6.37 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.10 Results for the Period day 8 to day 9

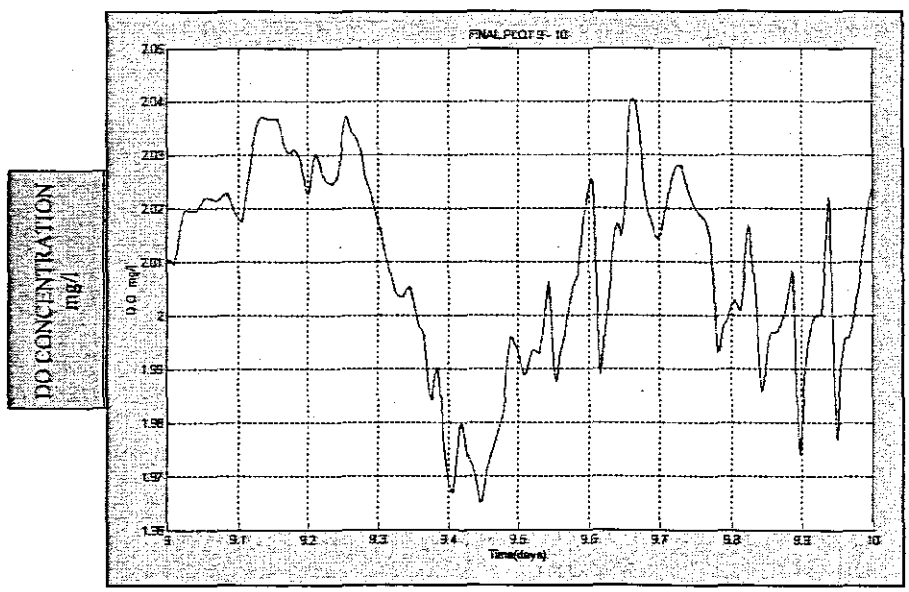


Figure 6.38: Final plot of dissolved oxygen concentration value for the period day 9 to day 10

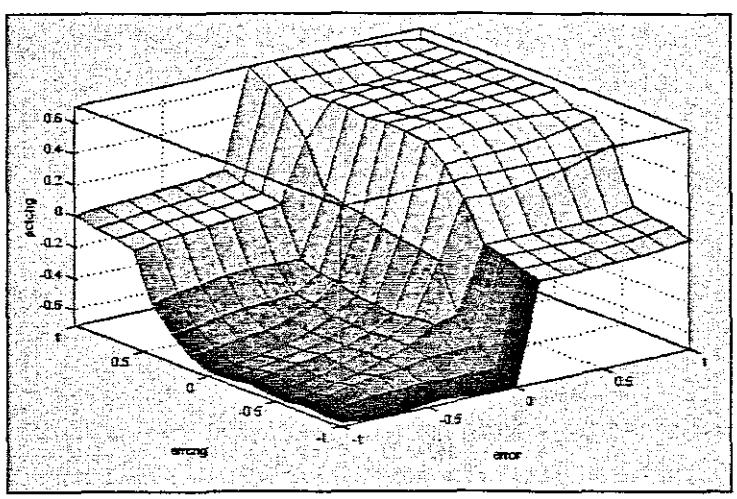


Figure 6.39: Plot of fuzzy controller surface for the period day 9 to day 10

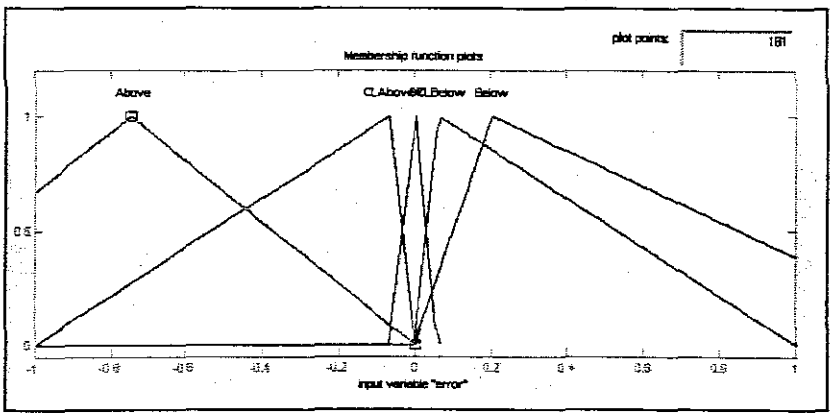


Figure 6.40: Plot of tuned fuzzy sets of "error" linguistic variable for period day 9 to day 10

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.75 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.06629 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.06629 0 0.06006 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.06006 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.2 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.11: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 9 to day 10

Figure 6.38 to Figure 6.40 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.11 Results for the Period day 10 to day 11

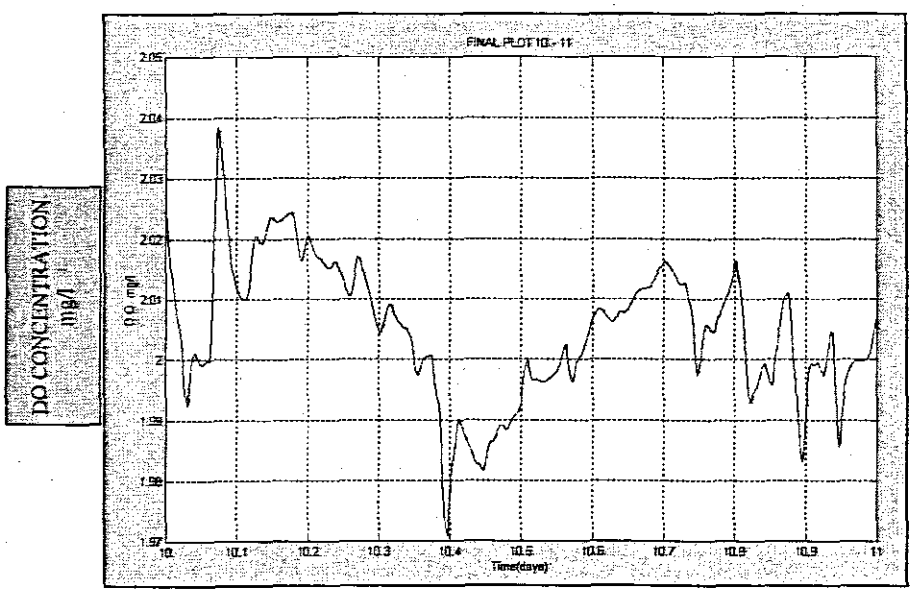


Figure 6.41: Final plot of dissolved oxygen concentration value for the period day 10 to day 11

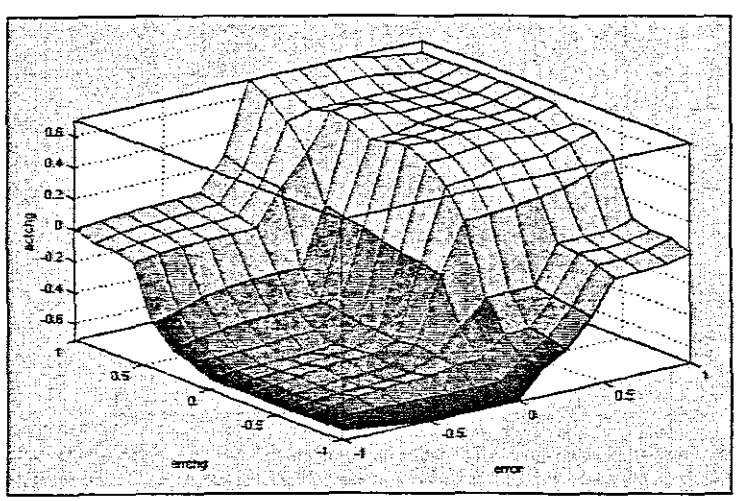


Figure 6.42: Plot of fuzzy controller surface for the period day 10 to day 11

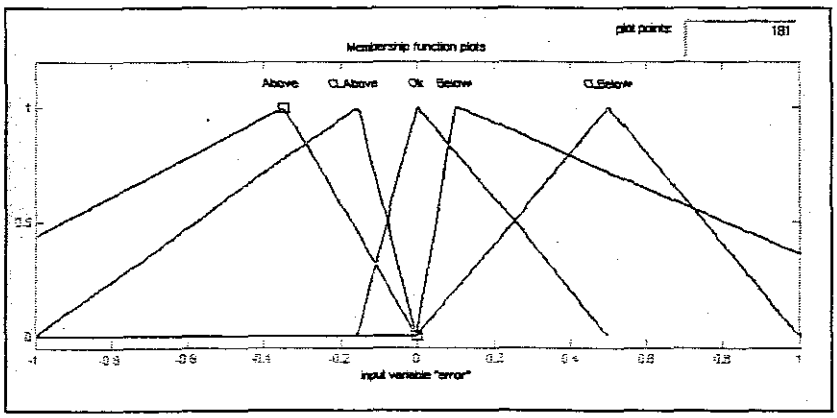


Figure 6.43: Plot of tuned fuzzy sets of "error" linguistic variable for period day 10 to day 11

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.35 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.1578 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.1578 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.1 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.12: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 10 to day 11

Figure 6.41 to Figure 6.43 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.12 Results for the Period day 11 to day 12

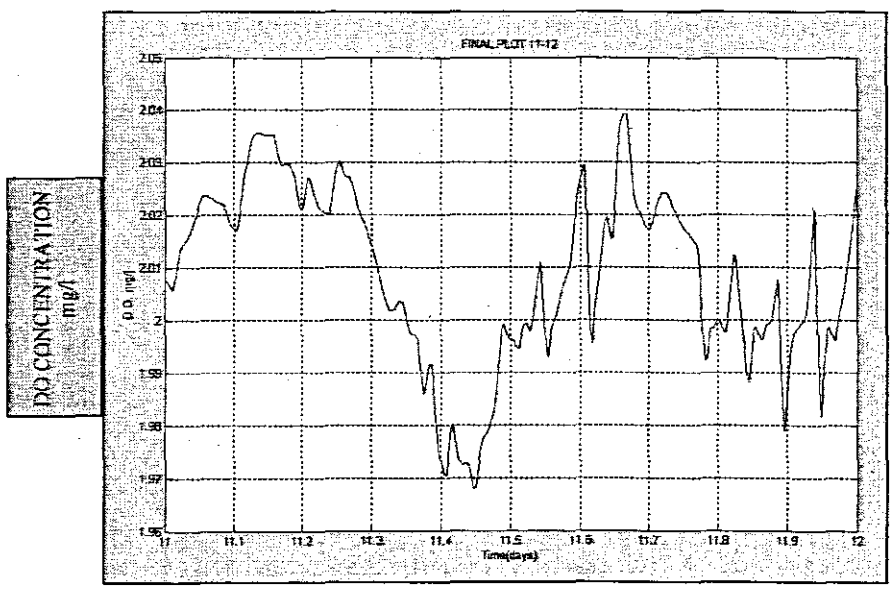


Figure 6.44: Final plot of dissolved oxygen concentration value for the period day 11 to day 12

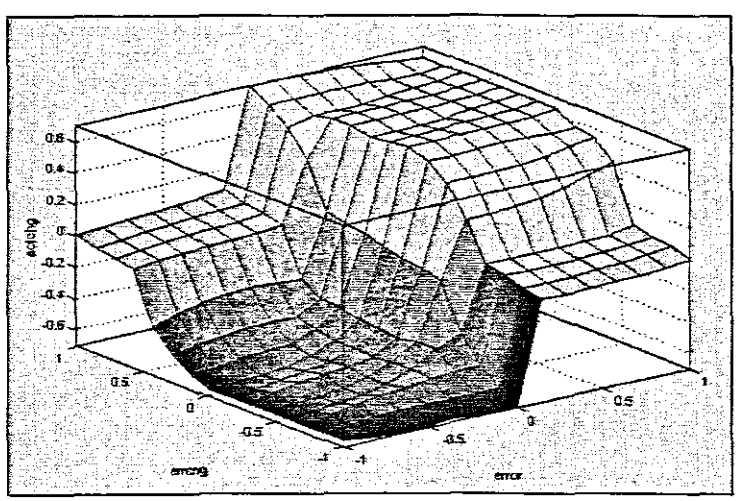


Figure 6.45: Plot of fuzzy controller surface for the period day 11 to day 12

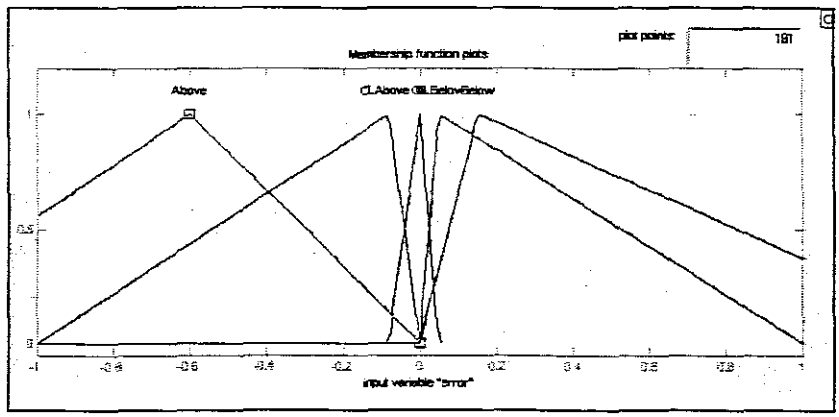


Figure 6.46: Plot of tuned fuzzy sets of "error" linguistic variable for period day 11 to day 12

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.6 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.08153 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	0.08153 0 0.04811 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.04811 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.15 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.13: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 11 to day 12

Figure 6.44 to Figure 6.46 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.13 Results for the Period day 12 to day 13

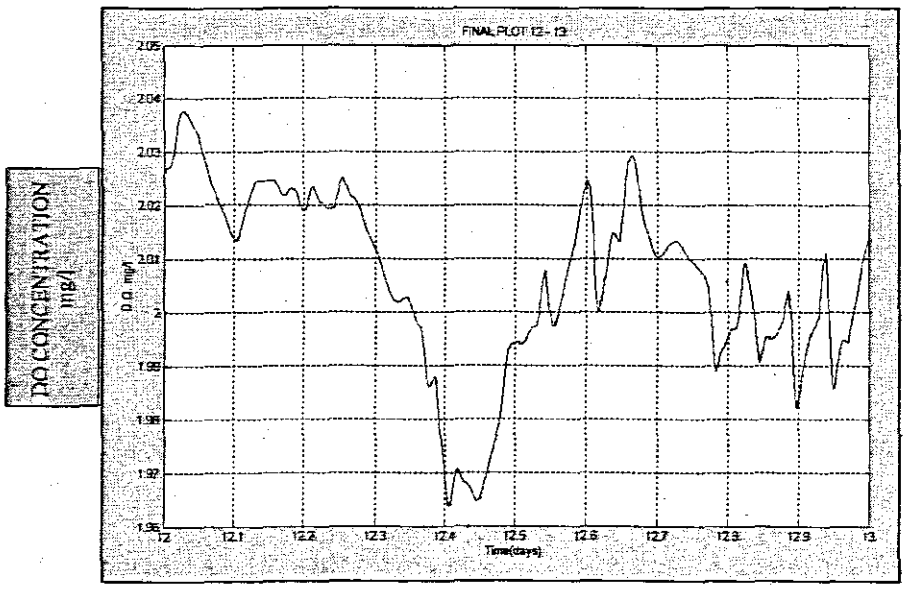


Figure 6.47: Final plot of dissolved oxygen concentration value for the period day 12 to day 13

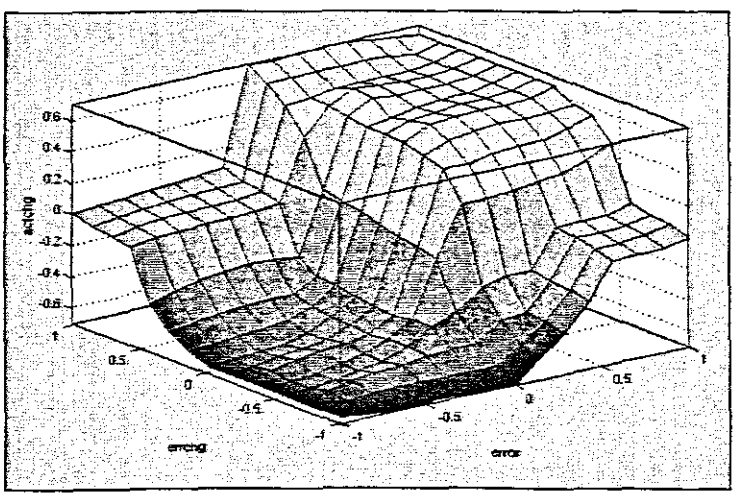


Figure 6.48: Plot of fuzzy controller surface for the period day 12 to day 13

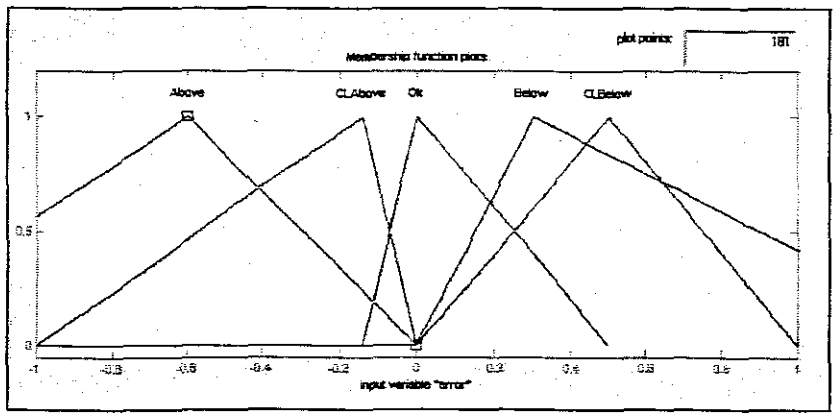


Figure 6.49: Plot of tuned fuzzy sets of "error" linguistic variable for period day 12 to day 13

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.6 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.1411 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.1411 0 0.5 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.5 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.3 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.14: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 11 to day 12

Figure 6.47 to Figure 6.49 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.4.14 Results for the Period day 13 to day 14

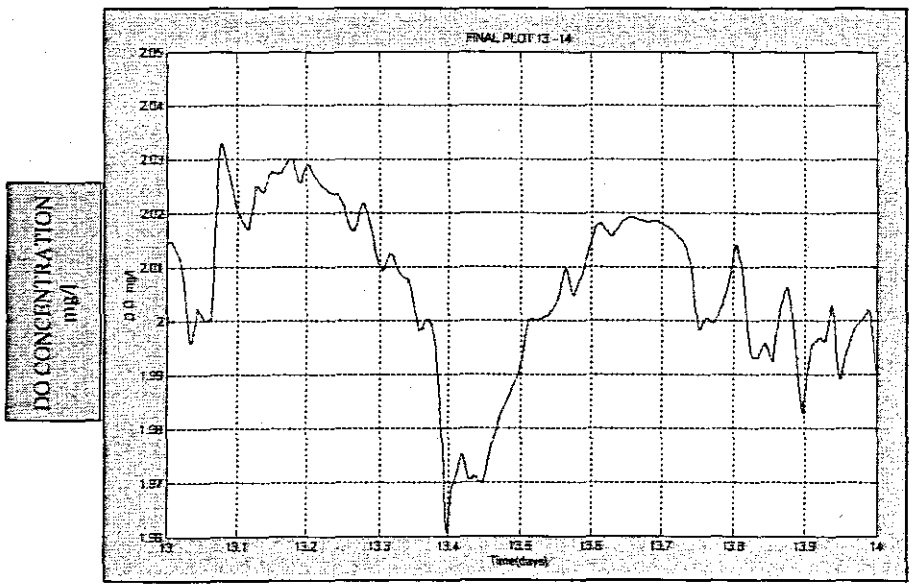


Figure 6.50: Final plot of dissolved oxygen concentration value for the period day 13 to day 14

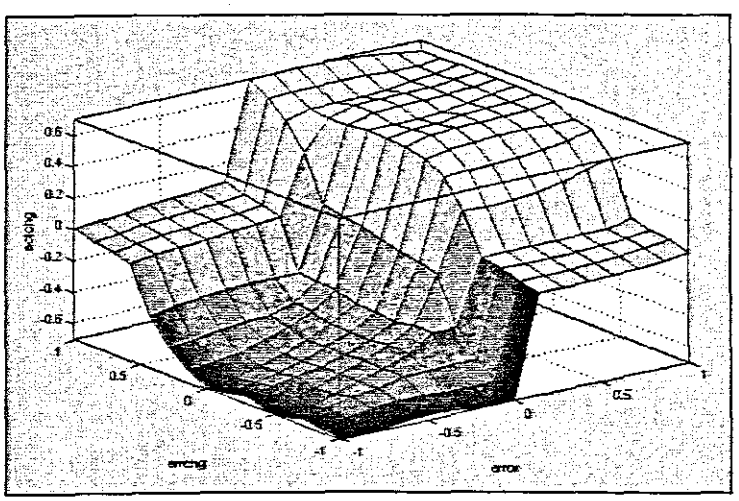


Figure 6.51: Plot of fuzzy controller surface for the period day 13 to day 14

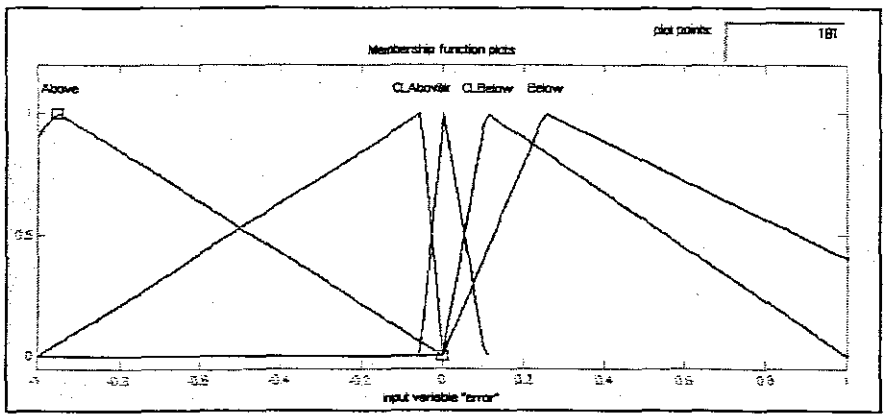


Figure 6.52: Plot of tuned fuzzy sets of "error" linguistic variable for period day 13 to day 14

CONTROLLER INPUT				CONTROLLER OUTPUT	
Error		Error_change		Actuator_change	
Above	[-1.5 -0.95 0 0]	Incfast	[-1 -0.5 0 0]	Openfast	[-1.5 -1 -0.5 0]
CLAbove	[-1 -0.05351 0 0]	Incslow	[-0.5 0 0.5 0]	Openslow	[-1 -0.5 0 0]
Ok	[-0.05351 0 0.1046 0]	Constant	[0 0.5 1 0]	Nochange	[-0.5 0 0.5 0]
CLBelow	[0 0.1046 1 0]	Decslow	[0.5 1 1.5 0]	Closeslow	[0 0.5 1 0]
Below	[0 0.25 1.5 0]	Decfast	[-1 -0.5 0 0]	Closefast	[0.5 1 1.5 0]

Table 6.15: Parameters of tuned fuzzy sets for optimized fuzzy controller over the period day 13 to day 14

Figure 6.50 to Figure 6.52 show the optimized trajectory, the controller surface, and a graphical display of the controller parameters respectively.

6.5 SUMMARY OF CHAPTER

This chapter has dealt with a very crucial aspect of fuzzy controller design which essentially is fuzzy parameter tuning. In many ways it can be deemed the most important part of fuzzy controller design and implementation. The complexity of tuning of fuzzy controllers was highlighted stating that the fundamental problem with tuning is the large amount of parameters that can be used for tuning. Also there exists no formal method for initial approximations of parameters and hence an adhoc approach is adopted for initial approximate adjustment.

Section 6.1.2 stated that tuning is further complicated because of the fact that fuzzy controllers are essentially non-linear controllers which further makes the tuning procedure more complex The non-linear control surface of the fuzzy controller was also briefly investigated and it was shown how the adjustment of certain parameters such as scaling gains,shape and position of fuzzy sets can affect the control surface. In essence the plot of the control surface enables one to obtain a global idea of how certain parameters effect the controller's performance and this can be a vital tool to analyse a fuzzy controller.

Although fuzzy controllers are inherently non-linear it was mentioned that certain design techniques can enable the linear fuzzy controllers to be designed. Research in the area of fuzzy controller tuning is an ongoing procedure with many techniques for tuning being developed especially with the aid techniques borrowed from the artificial intelligence field.

In this study an iterative tuning technique was developed which is based upon previous work done by Bartolini et al. It was shown that this technique performed admirably and managed to adjust the parameters of the fuzzy controller so that the performance criteria which essentially stipulated a steady state error of 2% were satisfied.

CHAPTER SEVEN

CONCLUSIONS

Fuzzy modeling provides a very flexible framework for integrating all forms of information whether it is of a qualitative or quantative nature. It also provides a very transparent interface with the operator or designer as the IF-THEN structure of the rules in the knowledge base shows the relationship between input and output variables in linguistic terms. Fuzzy control has come up against a lot of opposition but has still managed to find its niche in the field of control. What is quite evident is that it can provide control solutions in areas where conventional control is not effective enough. Academics have realized the potential of this technology and there is a concerted effort on their part to develop the theory so that there are tools to justify the application of the fuzzy logic methodology in the field of control. Software houses have capitalized upon the idea of fuzzy control and many software packages for the design of fuzzy controllers is available. FUZZYTECH appears to be the leading software package for controller design and has indeed been used for literally hundreds of real time applications especially in the field of industrial applications.

In this dissertation a generic fuzzy PI controller was developed and implemented on two applications deemed to be non-linear and complex. The first process was a 4-level cascaded tank system upon which real time control was performed. The second was a simulation of the activated sludge process of a sewage wastewater treatment plant where the fuzzy controller was used to control the setpoint of the dissolved oxygen concentration in an aerobic reactor of the process.

7.1 ACHIEVEMENTS OF THE OBJECTIVES OF THE DISSERTATION

The problem in the dissertation was to investigate the fuzzy logic methodology as a tool for controller development to control complex processes. This problem was decomposed into the following sub-problems

- Investigating the fuzzy logic methodology for controller development

- The development of a generic fuzzy PI controller for setpoint control

- Analysis of a 4-stage cascaded tank system for application of the developed controller for decentralized level control

- The development of software for simulation and real time control of the levels in the cascaded tank system.

- Analysis of the Activated Sludge Process of sewage wastewater treatment plants with emphasis on the dynamics of the dissolved oxygen concentration in the aerobic reactors.

- The development of a simulation of the Activated Sludge Process in the MATLAB/SIMULINK environment

- The development of a fuzzy controller based, upon the generic PI controller developed, for setpoint control of the dissolved oxygen concentration in the simulation of the activated sludge process.

- The development of a tuning process for determining the parameters of the fuzzy PI controller for dissolved oxygen concentration.

7.2 INVESTIGATION OF THE FUZZY LOGIC METHODOLOGY FOR CONTROL

An initial study was conducted to investigate the fuzzy methodology as applied to the field of control engineering. There exists a plethora of information for fuzzy controller development. Important was the understanding of the fundamental concepts of the fuzzy

logic methodology and fuzzy control. This is important, as many commercial software packages for fuzzy control tend to provide excellent graphical user interfaces for controller development but hide the low level computations involved in realizing the fuzzy controller. Chapter two can be read as a tutorial to show how the fuzzy methodology works and how a fuzzy controller can be realized. It provides the necessary theory for those who are unfamiliar with the concept of fuzzy logic in the field of control and lays the foundation for understanding the development of the fuzzy controller which is further expounded in chapter three.

7.3 THE DEVELOPMENT OF A GENERIC FUZZY PI CONTROLLER FOR SETPOINT CONTROL

With the foundation of the fuzzy control methodology explained in chapter two, the synthesis and procedure of the development of the generic fuzzy PI controller is dealt with in chapter three. Firstly, a theoretical explanation of the relationship between a conventional PI and fuzzy PI controller is given. It is shown how, based upon a conventional controller, the linguistic variables are chosen to realize a fuzzy PI controller. It is also shown how, once the linguistic variables are chosen, the rules are formulated to realize the action of a PI controller. Chapter three shows a method developed for fuzzy controller rule formulation, which can be applied to any process where the emphasis is on the development of a fuzzy controller for setpoint control.

It was also shown how software algorithms for fuzzy control could be developed utilizing a software package called "fuzzyTECH®". These algorithms are for the real-time implementation of a fuzzy PI controller, which was used for level control of a 4-stage cascaded tank system as explained in chapter four. A step-by-step procedure for the generation of code in the "C" programming language for controller implementation was explained.

7.4 ANALYSIS OF A 4-STAGE CASCADED TANK SYSTEM FOR APPLICATION OF THE DEVELOPED CONTROLLER FOR DECENTRALISED LEVEL CONTROL

It is generally assumed that fuzzy controllers can be designed entirely from a heuristic point of view. Hence in chapter four a heuristic analysis was conducted on a 4-stage cascaded tank system so that the fuzzy approach could be adopted to independently control the levels in each tank. To assist with the heuristic analysis, a method employing a technique utilizing a binary interaction matrix was used to find the relationship between the input and output variables. This was done without the aid of mathematical models. The final result of the analysis yielded a matrix showing that a decentralized fuzzy controller could be used to independently control the levels in each tank. To further assist with the development of the controller a simulation of the cascaded tank system was performed with the levels under the control of the simulated fuzzy PI controller. This simulation was done using the labVIEW software package. LabVIEW is a software package that is widely used in industry for both simulations as well as for real-time implementation of control systems. Chapter four also dealt with the development of code for real-time implementation of the fuzzy controller on the cascaded tank system. The code was developed using the "C" programming language and integrated with the fuzzy code developed in the **"fuzzyTECH®"** environment. Upon implementation the results show that the controller and control strategy provides effective and satisfactory control of the level setpoints in the actual process. With the success of this controller implementation the next step was the investigation of a more complex process for fuzzy controller implementation.

7.5 ANALYSIS OF THE ACTIVATED SLUDGE PROCESS

In chapter five a more complex process, the activated sludge process, was investigated. This investigation included a fundamental study, simulation and development of a fuzzy PI controller for control of the dissolved oxygen concentration in an aerobic tank of the process.

The activated sludge process is a highly non-linear process and a suitable candidate for fuzzy control. A study was conducted on the dynamics of the process with the emphasis placed on the importance of maintaining the proper dissolved oxygen concentrations in the aerobic tanks. As part of the investigation a simulation of the process was done in the MATLAB/SIMULINK environment where code was developed to simulate an activated sludge process consisting of two anoxic tanks and three aerobic tanks. For controller implementation a fuzzy PI controller was developed with the aid of the fuzzy logic toolbox in MATLAB/SIMULINK and used as the setpoint controller for dissolved oxygen concentration in an aerobic tank. It is shown in chapter five that the controller yielded positive results even though the controller parameters were not tuned for optimal control. To further improve the controller the next step was to develop a technique for tuning of the parameters of the fuzzy controller.

7.6 DEVELOPMENT OF A TUNING PROCESS FOR FUZZY PARAMETERS

The next and final stage of fuzzy controller development involves the optimization of the controller parameters. In chapter six the aspect of tuning of fuzzy controllers was investigated and a method for tuning of the fuzzy PI controller was developed and implemented on the simulated activated sludge plant. The tuning algorithm, consisting of MATLAB code, is an iterative process that tunes the parameters to achieve ideal process output conditions.

7.7 BENEFITS OF FUZZY CONTROL

The benefits of this control strategy provide improvements in the performance of the plant. Improvement in plant performance results in greater productivity, increased throughput and possibly less demand on energy resources. It also provides an alternative control paradigm to handle processes that are highly non-linear and complex. The results obtained have shown that the application of the fuzzy methodology in the field of control can yield positive results. The fuzzy control methodology is also very versatile as it can

be integrated with other technologies such as neural networks and genetic algorithms as explained in chapter six.

7.8 FUTURE DEVELOPMENTS AND APPLICATIONS OF DEVELOPED METHODS, ALGORITHMS AND PROGRAMS

Future developments can be subdivided in the following manner:

- Application of the developed fuzzy PI controller on a pilot plant of an activated sludge process to control the dissolved oxygen concentration.
- Testing of the tuning algorithm for optimizing parameters of the fuzzy controller controlling the dissolved oxygen concentration of the pilot plant.
- Investigating the integration of the fuzzy methodology with other artificial intelligence techniques such as neural networks.
- Development of a user-friendly control system for the activated sludge plant using labVIEW.
- Applications of the various methods and techniques developed in this dissertation including software can be applied to wastewater treatment plants and any plant that deals with the treatment of water. The programs are versatile and can be easily adapted to suit the particular application of interest.
- The software can also be used to train operators of wastewater treatment plants and activated sludge processes.
- The software can also be used in educational institutions that offer courses in wastewater treatment.

REFERENCES

ASTROM, K. J., DUGARD, L., M'MSAAD, M., & LANDAU, I.D. ,1992:Intelligent Tuning. Adaptive Systems in Control and Signal Processing , Pergamon Press, Oxford, pp. 360-370

BABUSKA, R., 1998: Fuzzy Modeling for Control. Kluwer Academic Publishers

BARTOLINI, G., CASALINO, G., DAVOLI, F.,MASTRETTA, M.,MINCIARDI, R., & MORTEN, E., 1982: Development of performance adaptive controllers with application to continuous casting plants. In : R. Trappl (ed) , Cybernetics and Systems Research, Amsterdam, North Holland, pp. 721-728

BELLMAN, R.E. & ZADEH, L. A., 1970: Decision-making in a Fuzzy Environment. Management Science, 17, no. 4, pp. 141-164

BESTAOUI, Y., 1989: Decentralised PD and PID robotic regulators. IEE Proceedings – D, 136, pp.133-145

BEZDEK, J.C., TSAO, E.C.K, & PAL, N.R., 1992: Fuzzy Kohonen clustering networks. Proceedings of the 1st IEEE International Conference on Fuzzy Systems, Fuzz-IEEE'92, pp. 1035-1043

BOCKEN, S.M., BRAAE, M., & DOLD, P.L., 1989: Dissolved Oxygen Control and oxygen utilization rate estimation: Extension of the Holmberg/Olsson method. Wat. Sci. Tech., 21, pp.1197-1208

BROWN, M., & HARRIS, C., 1994: Neurofuzzy Adaptive Modeling and Control. New York. Prentice Hall

BUCKLEY, J., & YING, H., 1990: Linear Fuzzy Controller: It is a linear nonfuzzy controller. Information Sciences, 51, pp. 183-192

CARLSSON, B.S., LINDBERG, C.F., HASSELBLAD, S, & XU, S., 1994:On-line estimation of the respiration rate and the oxygen transfer rate at Kungsangen wastewater plant in Uppsala. Wat. Sci. Tech., 30, pp. 255-263

CARPENTER, G.A., GROSSBERG, S., & ROSEN, D.B., 1991: Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, Vol.4, pp. 759-771

CARPENTER, G.A., GROSSBERG, S., MARKUZON, N., REYNOLDS, J.H., ROSEN, D.B., 1993: Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multi-dimensional maps. IEEE Transactions on Neural Networks, Vol.3, no. 5, pp. 698-713

CHUNG, F.L., & LEE, T., 1994: Fuzzy competitive learning. Neural Networks, Vol.7 no.3, pp.539-551

COOPER, M.G., & VIDAL, J.J., 1993: Genetic Design of Fuzzy Controllers. Proceedings of the 2nd International Conference on Fuzzy theory and Technology, Durham, NC

De Waal, A., 1990: Performance Index Analysis of Control Systems Synthesised for a Counter Current Process. M.Sc Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town.

DRIANKOV, D.,HELLENDOORN, H.,REINFRANK, M., 1996: An Introduction to Fuzzy Control, second edn, Springer-Verlag, Berlin

FLANAGAN, M.J., BRACKEN, B.D., ROESLER, J.F., 1977: Automatic dissolved oxygen control. Journal of the Environmental Engineering Division EE4, pp. 707-722

FRANKS, R., 1972: Modeling and Simulation in Chemical Engineering. New York: Wiley

GEORGESCU, G., AFSHARI, A., & BORNARD, G., 1993: Fuzzy predictive PID controllers: A heating control application. Proceedings of the 2nd IEEE Conference on Fuzzy Systems, pp. 1091-1098

GLORENNEC, P.Y., 1991: Adaptive Fuzzy Control . Proceedings of the IFSA '91, pp. 33-36

GOMEZ, G.E., & DUQUE, E., 1998: Automatic startup of a high rate anaerobic reactor using a fuzzy logic control system. 5th Latin American workshop-seminar, Vina del mar, Chile

GUPTA, M.M., & RAO, D.H., 1994: On the principles of fuzzy neural networks. Fuzzy Sets and Systems, Vol.61, pp. 1-18.

HE, S.Z., TAN, S.H., XU, F.L., & WANG, P.Z., 1993: Fuzzy Self-tuning of PID Controllers. Fuzzy Sets and Systems, 56, pp. 37-46

HENZE, M., GRADY Jr, C.P.L., GUJER, W., MARAIS, G. v. R., & MATSUO, T., 1987: Activated Sludge Model no. 1. Scientific and Technical Report No1. IAWPRC, London

HIROTA, K., ARAI, A., & HACHISU, S., 1989: Fuzzy Controlled Robot Arm Playing Two-Dimensional Ping-Pong Game. Fuzzy Sets and Sytems, 32, no. 2, pp. 149-159

HOLLAND, C. & LIAPSIS, A., 1983: Computer Methods for Solving Dynamic Separation Problems. New York: McGraw Hill.

HOLMBERG, U., OLSSON, G., & ANDERSSON, B., 1989: Simultaneous DO control and respiration estimation. Wat. Sci. Tech., 21, pp.1185-1195

HUNTSBERGER, T.L., & AJJIMARANGSEE, P., 1990: Parallel self-organising feature maps for unsupervised pattern recognition. Int. J. General Systems, Vol.16, no. 4, pp.357-372

JAGER, R., 1995: Fuzzy Logic in Control, PhD thesis,Technische Universiteit Delft,

JANG, J.S.R, & SUN C. T., 1993: Functional equivalence between Radial Basis Function Networks and Fuzzy Inference Systems. IEEE Transactions on Neural Networks, 4(1), pp. 156-159

JANG, J.S.R., 1993: ANFIS: Adaptive-network-based Fuzzy Inference Systems. IEEE Transactions on Systems, Man and Cybernetics, 23(3), pp. 665-685

JANTZEN, J., 1998: Tuning of Fuzzy PID controllers. Technical report no.98-H 871(fpid). Technical University of Denmark, Department of Automation

KARR, C., 1991: Genetic Algorithms for Fuzzy Controllers. AI Expert, pp. 26-33

KIUPEL, N., FRANK, P.M., & BUX, O.,1994: Fuzzy control of steam turbines. Fuzzy Sets and Systems, 63, pp.319-327

KO, K.Y., McINNIS, B.C., GOODWIN, G.C., 1982: Adaptive control and identification of the the dissolved oxygen process. Automatica, 18(6), pp. 727-730

KOKOTOVIC, P., KHALIL, H. & O' REILLY, J., 1986: Singular Pertubation Methods in Control. Analysis and Design, Academic Press, London

KOSKO, B., 1992: Neural Networks and Fuzzy Systems.New Jersey, Prentice Hall

KOSKO, B., 1994: Fuzzy Systems as Universal Approximators. IEEE Transactions on Computers, 43, pp. 1329-1333

LEE, J., 1993: A Method of Improving the Performance of PI-type fuzzy logic controllers. IEEE Transactions on Fuzzy Systems, 1, pp. 298-301

LEE, M.A., & TAKAGI, H., 1993: Integrating design stages of Fuzzy Systems Using Genetic Algorithms. Proceedings of the 2nd IEEE International Conference on Fuzzy Systems. New York, NY, Institute of Electrical and Electronics Engineers, pp. 612-617

LEE, S.C., & LEE, E.T., 1975: Fuzzy neural networks. Mathematical Biosciences, Vol. 23, pp. 151-177

LINDBERG, C.F.,1997: Control and estimation strategies applied to the activated sludge process. Ph.D thesis. Uppsala University. Dept. of Systems and Control

MALKI, H. A., LI, H., CHEN, G., 1994: New Design and Stabilty Analysis of Fuzzy Proportional Derivative Control Sytems. IEEE Transactions on Fuzzy Systems, 2, pp. 245-254

MALKI, H. A., KWAN, A., & CHEN, G.,1995: An Application of the Fuzzy PI controller to Temperature Control using real Data from a Power Plant. Annual report to the energy laboratory , University of Houston

MAMDANI, E. H. & ASSILIAN S., 1975: An experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man Machine Studies, Vol.7 , No.1,pp.1-13.

MAMDANI, E.H., & BAAKLINI, N., 1975: Prescriptive method for deriving control policy in a fuzzy logic controller. Electronic Letters, 11, pp. 625-626

MAMDANI, E.H., 1976: Advances in the linguistic synthesis of Fuzzy Controllers. Int. J. Man-Mach. Stud., 8, pp. 669-678

MAMDANI, E.H., PROCYK, T., & BAAKLINI, N., 1976: Application of Fuzzy Logic o controller Design Based on Linguistic Protocol. In: E.H. Mamdani and B.R. Gaines (eds), Discrete Systems and Fuzzy Reasoning, Queen Mary College, University of London, pp. 125-149

MAMDANI, E. H., 1977: Application of Fuzzy Logic to Approximate Reasoning using Linguistic Sytems. Fuzzy Sets and Systems, 26, pp. 1182-1191

MENDEL J. M., 1995: Fuzzy Logic Systems for Engineering: A Tutorial. Proceedings of the IEEE, Vol.83, No.3.pp345-377.

MIZUMOTO, M., 1992: Realization of PID controls by fuzzy control methods. IEEE 1ST International Conference on Fuzzy Systems, number 92CH3073-4, The Institute of Electrical and Electronics Engineers, Inc, San Diego, pp. 709-715

MORBIDELLA, A., SEWIDA, A., STORTI, G. & CARRAS, C., 1982: Simulation of Multi-component Absorption Beds. Model Analysis and Numerical Solution. Industrial Engineering Chemical Fundamentals, Vol. 21, no. 2, pp. 123-131

NIELSEN, M.K., LYNGAARD, J., 1993: Superior tuning and reporting (STAR)- a new concept for on-line process control of waste water treatment plants. Proceedings of the 6th IAWQ Workshop on Instrumentation. Control and Automation of Water and Wastewater Treatment and Transportation Systems, pp. 560-574

NOMURA, H., HAYASHI, I., & WAKAMI, N., 1991: A Self-tuning method of Fuzzy Control by Descent Method. Proceedings of the IFSA '91, Brussels, pp. 155-158

PAL, S.K., & MITRA, S., 1992: Multilayer perceptron, fuzzy sets, and classification. IEEE Transactions on Neural Networks, Vol.3, pp. 683-697

PEDRYCZ, W., 1990: Relevancy of Fuzzy Models. Information Sciences, 52, pp. 285-302

PEDRYCZ, W., 1992: Fuzzy neural networks with reference neurons as pattern classifiers. IEEE Transactions on Neural Networks, Vol.3, no.5, pp. 770-775

PIVONKA, P., & BLAHA, P., 1999 : Comparative analysis of classical and Fuzzy PID control algorithms. http://www.neci.nec.com/, site visited June 2002.

PROCYK T.J., & MAMDANI E.H., 1979: A Linguistic Self-Organizing Process Controller. Automatica, 15(1), pp.15-30

QIAO, W., & MIZUMOTO, M., 1996: PID type fuzzy controller and parameters adaptive method . Fuzzy Sets and Systems, 78, pp.23-35

QIN, S.J., & BORDERS, G., 1994: A multiregion fuzzy logic controller for non-linear Process control. IEEE Transactions on Fuzzy Systems, 2, pp. 74-81

RANGANATHAN, R.S., MAKLKI, H.A., & CHEN, G., 2002: Fuzzy predictive PI control for processes with large time delay. Expert Systems. 19(1), pp. 21-33

RUNDQWIST, L., 1986:Self-tuning control of the Dissolved Oxygen Concentration in an Activated Sludge Process. Licentiate Thesis. Lund Institute of Technology. Department of Automatic Control

SILER, W., & YING, H., 1989: Fuzzy Control Theory: The linear case. Fuzzy Sets and Systems. 33, pp.275-290

SUGENO, M. & NISHIDA, M., 1985: Fuzzy Control of a Model Car. Fuzzy Sets and Systems, pp. 103-113

SWALLOW, J.P., 1991: The Best of the Best in Advanced Process Control. Keynote speech at the 4th International Chemical Process Control Conference, Padre Island

TAKAGI, T. & SUGENO, M., 1985: Fuzzy Identification of Systems and its Application to Modeling and Control. IEEE Trans. Systems, Man and Cybernetics, 15(1), pp. 116-132

TANG, K.S., MAN, K.F., CHEN, G., & KWON, S., 2001:An Optimal Fuzzy PID Controller. IEEE Transactions on Industrial Electronics, 48, 4, pp. 757-765

TANG, W.M., & CHEN, G., 1994: A robust fuzzy PI controller for a flexible-joint robot arm with uncertainties. Proceedings of the 3rd IEEE Conference on Fuzzy Systems, pp.1554-1559

TANG, W., CHEN, G., & LU, R., 2001: A modified fuzzy PI controller for a flexible-joint robot arm with uncertainties. Fuzzy Sets and Systems, 118, pp. 109-119

TONG, R.M., 1976: Analysis of Fuzzy Control Algorithms using the Relation Matrix. Int. J. Man-Machine Studies, 8, pp.679-686

TSAO, E.C.K., BEZDEK, J.C., & PAL, N.R., 1994: Fuzzy Kohonen clustering networks. Pattern Recognition , Vol.27, no. 5, pp 757-764

TSO, S.K., & FUNG, Y.H., 1997: Methodological development of fuzzy logic controllers from multi-variable linear control. IEEE Transactions on Systems, Man and Cybernetics, 27(3), pp. 566-572

WANG, L. X., 1992: Fuzzy Sytems are Universal Approximators. Proceedings of the IEEE of the International Conference on Fuzzy Systems 1992, San Diego, U.S.A., pp. 1163-1170

WANG, L.X., 1997: A Course in Fuzzy Systems and Control. New Jersey, Prentice Hall

WELLS, H. C., 1979: Computer control of fully nitrifying activated sludge processes. Instrumentation Technology, 4, pp. 32-36

YAGER, R. R., & FILEV, D. P., 1994: Essentials of Fuzzy Modeling and Control. New York. John Wiley

YAMAZAKI, T., & MAMDANI, E.H., 1982: On the performance of a Rule-based Self-organizing controller. Proceedings of the IEEE Conference on Applications of Adaptive and Multivariable control, Hull, England, pp. 50-55

YAMAKAWA, T., 1989: Stabilization of an Inverted Pendulum by a High-Speed Fuzzy Logic Controller Hardware Systems. Fuzzy Sets and Systems, 32, pp. 161-180

YAMAMOTO, S., & HASHIMOTO, I., 1991: Present Status and Future Needs: The view from Japanese Industry. Proceedings of the 4th International Chemical Process Control Conference, Padre Island, TX, pp. 1-28.

YING, H., SILER, W., & BUCKLEY, J.J., 1990: Fuzzy Control Theory: A non-linear case. Automatica, 26, pp.513-520

YING, H., 1993: A Two-input Two-output Fuzzy controller is the sum of two non-linear PI controllers with variable gains. Proceedings of the 2nd International Conference on Fuzzy Systems, San Francisco, CA, pp. 35-37

YU, C., CAO, Z., & KANDEL, A., 1990: Application of Fuzzy Reasoning to the control of an Activated Sludge Plant. Fuzzy Sets and Systems, 38, pp.1-14

ZADEH, L. A., 1965: Fuzzy Sets. Information and Control, Vol. 8,pp.338-353

ZADEH, L.A., 1968: Fuzzy Algorithms. Information and Control, 12, no. 2,pp. 94-102

ZADEH, L.A., 1973: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions Sytems. Man and Cybernetics, Vol.SMC-3,pp.28-44.

ZENG, X.J., & SINGH, M.G., 1995: Approximation Theory of Fuzzy Systems-MIMO case. IEEE Transactions on Fuzzy Systems, 3(2), pp. 219-235

ZIMMERMAN, H. J., 1987: Fuzzy Sets. Decision Making and Expert Systems. Boston, Kluwer Academic Publishers

APPENDIX A

A.1 Table showing model parameters for ASM no. 1 model

IAWQ model parameters	symbol	unit	20 °C	10 °C	literature
Stoichiometric parameters					
Heterotrophic yield	Y_H	g cell COD formed (g COD oxidized)$^{-1}$	0.67	0.67	0.38-0.75
Autotrophic yield	Y_A	g cell COD formed (g N oxidized)$^{-1}$	0.24	0.24	0.07-0.28
Fraction of biomass yielding particulate products	f_P	dimensionless	0.08	0.08	-
Mass N/mass COD in biomass	i_{XB}	g N (g COD)$^{-1}$ in biomass	0.086	0.086	-
Mass N/mass COD in products from biomass	i_{XP}	g N (gCOD)$^{-1}$ in endogenous mass	0.06	0.06	-
Kinetic parameters					
Heterotrophic max. specific growth rate	μ_H	day^{-1}	6.0	3.0	0.6-13.2
Heterotrophic decay rate	b_H	day^{-1}	0.62	0.20	0.05-1.6
Half-saturation coefficient (hsc) for heterotrophs	K_S	g COD m^{-3}	20	20	5-225
Oxygen hsc for heterotrophs	$K_{O,H}$	g O_2 m^{-3}	0.20	0.20	0.01-0.20
Nitrate hsc for denitrifying heterotrophs	K_{NO}	g NO_3-N m^{-3}	*0.50*	*0.50*	*0.1-0.5*
Autotrophic max. specific growth rate	μ_A	day^{-1}	0.80	0.30	0.2-1.0
Autotrophic decay rate	b_A	day^{-1}	0.20	0.10	0.05-0.2
Oxygen hsc for autotrophs	$K_{O,A}$	g O_2 m^{-3}	0.4	0.4	0.4-2.0
Ammonia hsc for autotrophs	K_{NH}	g NH_3-N m^{-3}	1.0	1.0	-
Correction factor for anoxic growth of heterotrophs	η_g	dimensionless	0.8	0.8	0.6-1.0
Ammonification rate	k_a	m^3 (g COD day)$^{-1}$	0.08	0.04	-
Max. specific hydrolysis rate	k_h	g slowly biodeg. COD (g cell COD day)$^{-1}$	3.0	1.0	-
Hsc for hydrolysis of slowly biodeg. substrate	K_X	g slowly biodeg. COD (g cell COD)$^{-1}$	0.03	0.01	-
Correction factor for anoxic hydrolysis	ηh	dimensionless	0.4	0.4	-

A.2 LISTING OF CODE FOR SIMULATION OF ANOXIC AND AEROBIC BIOREACTORS.

A.2.2 *Code Listing for the first ANOXIC tank in the system*

```
function [sys,x0,str,ts] = bioreactmodinc(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
switch flag,
%%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%
% Outputs %
%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================
=========
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================
=========
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
```

```
sizes.NumOutputs = 11;
sizes.NumInputs = 20;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,8.4,1552,81.0,1439,50.6,0,0.6,9.7,0.7,4.3];
str = [];
ts = [0 0];
% end mdlInitializeSizes
%=============================================================
=========
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================
=========
function sys=mdlDerivatives(t,x,u)
% parameters
do=0;
par(1)=0.67;par(2)=0.24;par(3)=0.08;par(4)=0.086;par(5)=0.06;par(6)=6;
par(7)=0.62;par(8)=20;par(9)=0.2;par(10)=0.5;par(11)=0.8;par(12)=0.2;
par(13)=0.4;par(14)=1;par(15)=0.8;par(16)=0.08;par(17)=3;par(18)=0.03;
par(19)=0.4;
%inputs
x(7)=0;
in1=u(1);in2=u(2);in3=u(3);in4=u(4);in5=u(5);
in6=u(6);in7=u(7);in8=u(8);in9=u(9);in10=u(10);in11=u(11);
exin(1)=u(12);exin(2)=u(13);exin(3)=u(14);exin(4)=u(15);exin(5)=u(16);exin(9)=u(17);
exin(10)=u(18); exin(11)=u(19); qin=u(20);
% influent concentrations
exin(6)=0;
exin(7)=0; exin(8)=0;
 % transpose into a column vector

% flow rates and volumes
qret=qin*0.5; qrec=qin*2;
vol=1000; Vtot = 4999;
% thinkening factor and sludge age – ideal settler
SA=7;
gamma=ones(11,1);
gamma([3:6 11])=(qin+qret-Vtot/SA)/qret * ones(1,5); % particulate states are thickened
gamma(7)=0; % DO concentration in return sluge flow = 0.

q = [qin qret qrec];
%------ AEROBIC ZONE ----------------------%
% processes, taken from the IAWQ matrix formulation
```

```
p1 = par(6) * (x(2)/(par(8)+x(2))) * (x(7)/(par(9)+x(7))) * x(5);%
p2 = par(6) * (x(2)/(par(8)+x(2))) * (par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8))) *
par(15) * x(5);%
p3 = par(11) * (x(9)/(par(14)+x(9))) * (x(7)/(par(13)+x(7))) * x(6);%
p4 = par(7) * x(5);%
p5 = par(12) * x(6);%
p6 = par(16) * x(10) * x(5);%
p7 = par(17) * ((x(4)/x(5))/(par(18)+(x(4)/x(5)))) * ((x(7)/(par(9)+x(7))) + par(19) *
(par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8)))) * x(5);
p8 = p7*(x(11)/x(4));%

% aeration parameters and Kla model

% differential equations according to IAWQ matrix and plant layout

sys(1)=1/vol*(q*[exin(1) u(1)*gamma(1) u(1)]'-sum(q)*x(1));
sys(2)=1/vol*(q*[exin(2) u(2)*gamma(2) u(2)]'-sum(q)*x(2))      -    1/par(1)*p1    -
1/par(1)*p2 + p7;
sys(3)=1/vol*(q*[exin(3) u(3)*gamma(3) u(3)]'-sum(q)*x(3))      +    par(3)*p4      +
par(3)*p5;
sys(4)=1/vol*(q*[exin(4) u(4)*gamma(4) u(4)]'-sum(q)*x(4))      + (1-par(3))*p4 + (1-
par(3))*p5 - p7;
sys(5)=1/vol*(q*[exin(5) u(5)*gamma(5) u(5)]'-sum(q)*x(5))      + p1 + p2 - p4;
sys(6)=1/vol*(q*[exin(6) u(6)*gamma(6) u(6)]'-sum(q)*x(6))      + p3 - p5;
if do > 0
        sys(7)=1/vol*(q*[exin(7) u(7)*gamma(7) u(7)]'-sum(q)*x(7))      + Kla*(sosat-
x(7)) - (1-par(1))/par(1)*p1 + (1-4.57/par(2))*p3;
else
        sys(7)=0;
end
sys(8)=1/vol*(q*[exin(8) u(8)*gamma(8) u(8)]'-sum(q)*x(8))      -                (1-
par(1))/(2.86*par(1))*p2 + 1/par(2)*p3;
sys(9)=1/vol*(q*[exin(9) u(9)*gamma(9) u(9)]'-sum(q)*x(9))      -    par(4)*p1    -
par(4)*p2 - (par(4)+1/par(2))*p3 + p6;
sys(10)=1/vol*(q*[exin(10) u(10)*gamma(10) u(10)]'-sum(q)*x(10)) - p6 + p8;
sys(11)=1/vol*(q*[exin(11)  u(11)*gamma(11)  u(11)]'-sum(q)*x(11))  +  (par(4)-
par(3)*par(5))*p4 + (par(4)-par(3)*par(5))*p5 - p8;



% end mdlDerivatives
%
%=============================================================================
% mdlOutputs
% Return the block outputs.
```

```
%=============================================================
=========
%
function sys=mdlOutputs(t,x,u)
sys = x;
% end mdlOutput
```

A.2.3 Code Listing for the second ANOXIC bioreactor in the system

```
function [sys,x0,str,ts] = bioreactmodin1(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
switch flag,
%%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 11;
sizes.NumInputs = 12;
```

```
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,8.4,1552,81.0,1439,50.6,0,0.6,9.7,0.7,4.3];
str = [];
ts = [0 0];
% end mdlInitializeSizes
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys=mdlDerivatives(t,x,u)
% parameters
do=0;
par(1)=0.67;par(2)=0.24;par(3)=0.08;par(4)=0.086;par(5)=0.06;par(6)=6;
par(7)=0.62;par(8)=20;par(9)=0.2;par(10)=0.5;par(11)=0.8;par(12)=0.2;
par(13)=0.4;par(14)=1;par(15)=0.8;par(16)=0.08;par(17)=3;par(18)=0.03;
par(19)=0.4;
%inputs
x(7)=0;
in1=u(1);in2=u(2);in3=u(3);in4=u(4);in5=u(5);
in6=u(6);in7=u(7);in8=u(8);in9=u(9);in10=u(10);in11=u(11);qin=u(12);
%exin(1)=u(12);exin(2)=u(13);exin(2)=u(14);exin(2)=u(15);exin(2)=u(16);exin(9)=u(17)
;
%exin(10)=u(18); exin(11)=u(19); qin=u(20);
% influent concentrations
%exin(6)=0;
%exin(7)=0; exin(8)=0;
 % transpose into a column vector

% flow rates and volumes

qret=qin*0.5; qrec=qin*2;
vol=1000; Vtot = 5999;
% thinkening factor and sludge age – ideal settler
SA=7;
gamma=ones(11,1);
gamma([3:6 11])=(qin+qret-Vtot/SA)/qret * ones(1,5); % particulate states are thickened
gamma(7)=0; % DO concentration in return sluge flow = 0.

q = [qin+qret+qrec];
%------ AEROBIC ZONE ----------%
% processes, taken from the IAWQ matrix formulation
```

```
p1 = par(6) * (x(2)/(par(8)+x(2))) * (x(7)/(par(9)+x(7))) * x(5);%
p2 = par(6) * (x(2)/(par(8)+x(2))) * (par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8))) *
par(15) * x(5);%
p3 = par(11) * (x(9)/(par(14)+x(9))) * (x(7)/(par(13)+x(7))) * x(6);%
p4 = par(7) * x(5);%
p5 = par(12) * x(6);%
p6 = par(16) * x(10) * x(5);%
p7 = par(17) * ((x(4)/x(5))/(par(18)+(x(4)/x(5)))) * ((x(7)/(par(9)+x(7))) + par(19) *
(par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8)))) * x(5);
p8 = p7*(x(11)/x(4));%

% aeration parameters and Kla model

% differential equations according to IAWQ matrix and plant layout

sys(1)=1/vol*(q*u(1)-sum(q)*x(1));
sys(2)=1/vol*(q*u(2)-sum(q)*x(2))  - 1/par(1)*p1 - 1/par(1)*p2 + p7;
sys(3)=1/vol*(q*u(3)-sum(q)*x(3))  + par(3)*p4 + par(3)*p5;
sys(4)=1/vol*(q*u(4)-sum(q)*x(4))  + (1-par(3))*p4 + (1-par(3))*p5 - p7;
sys(5)=1/vol*(q*u(5)-sum(q)*x(5))  + p1 + p2 - p4;
sys(6)=1/vol*(q*u(6)-sum(q)*x(6))  + p3 - p5;
if do > 0
        sys(7)=1/vol*(q*u(7)-sum(q)*x(7))  + Kla*(sosat-x(7)) - (1-par(1))/par(1)*p1 +
(1-4.57/par(2))*p3;
else
        sys(7)=0;
end
sys(8)=1/vol*(q*u(8)-sum(q)*x(8))  - (1-par(1))/(2.86*par(1))*p2 + 1/par(2)*p3;
sys(9)=1/vol*(q*u(9)-sum(q)*x(9))  - par(4)*p1 - par(4)*p2 - (par(4)+1/par(2))*p3 + p6;
sys(10)=1/vol*(q*u(10)-sum(q)*x(10)) - p6 + p8;
sys(11)=1/vol*(q*u(11)-sum(q)*x(11))  +  (par(4)-par(3)*par(5))*p4  +  (par(4)-
par(3)*par(5))*p5 - p8;



% end mdlDerivatives
%
%=============================================================
=========
% mdlOutputs
% Return the block outputs.
%=============================================================
=========
%
function sys=mdlOutputs(t,x,u)
sys = x;
```

```
% end mdlOutput
```

A.2.4 Code listing for 1st AEROBIC bioreactor in the system

```
function [sys,x0,str,ts] = bioreactmod1withpid(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
switch flag,
%%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 11;
```

```
sizes.NumInputs = 13;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,2.8,1559,24.8,1451,51.7,0,7.9,2.2,0.9,1.7];

str = [];
ts = [0 0];
% end mdlInitializeSizes
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys=mdlDerivatives(t,x,u)
% parameters
do=1;
par(1)=0.67;par(2)=0.24;par(3)=0.08;par(4)=0.086;par(5)=0.06;par(6)=6;
par(7)=0.62;par(8)=20;par(9)=0.2;par(10)=0.5;par(11)=0.8;par(12)=0.2;
par(13)=0.4;par(14)=1;par(15)=0.8;par(16)=0.08;par(17)=3;par(18)=0.03;
par(19)=0.4;
%inputs
%x(7)=2;
in1=u(1);in2=u(2);in3=u(3);in4=u(4);in5=u(5);
in6=u(6);in7=u(7);in8=u(8);in9=u(9);in10=u(10);in11=u(11);out=u(12);qin=u(13);

% influent concentrations
%exin(1)=20; exin(2)=125; exin(3)=60; exin(4)=175; exin(5)=20; exin(6)=0;
%exin(7)=0; exin(8)=1; exin(9)=30; exin(10)=5; exin(11)=5;
%exin=exin'; % transpose into a column vector

% flow rates and volumes
qret=qin*0.5; qrec=qin*2;
vol=1333; Vtot = 5999;
% thinkening factor and sludge age – ideal settler
SA=7;
gamma=ones(11,1);
gamma([3:6 11])=(qin+qret-Vtot/SA)/qret * ones(1,5); % particulate states are thickened
gamma(7)=0; % DO concentration in return sluge flow = 0.

%xin = u';
q = [qin+qret+qrec];
%------ AEROBIC ZONE ----------%
% processes, taken from the IAWQ matrix formulation
```

```
p1 = par(6) * (x(2)/(par(8)+x(2))) * (x(7)/(par(9)+x(7))) * x(5);%
p2 = par(6) * (x(2)/(par(8)+x(2))) * (par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8))) *
par(15) * x(5);%
p3 = par(11) * (x(9)/(par(14)+x(9))) * (x(7)/(par(13)+x(7))) * x(6);%
p4 = par(7) * x(5);%
p5 = par(12) * x(6);%
p6 = par(16) * x(10) * x(5);%
p7 = par(17) * ((x(4)/x(5))/(par(18)+(x(4)/x(5)))) * ((x(7)/(par(9)+x(7))) + par(19) *
(par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8)))) * x(5);
p8 = p7*(x(11)/x(4));%
qair = out*7;
k1=400; k2=-0.42; Kla=k1*(1-exp(k2*qair));
sosat=9.5; qairtot=qair*4000;

% aeration parameters and Kla model

% differential equations according to IAWQ matrix and plant layout

sys(1)=1/vol*(q*u(1)-sum(q)*x(1));
sys(2)=1/vol*(q*u(2)-sum(q)*x(2))  - 1/par(1)*p1 - 1/par(1)*p2 + p7;
sys(3)=1/vol*(q*u(3)-sum(q)*x(3))  + par(3)*p4 + par(3)*p5;
sys(4)=1/vol*(q*u(4)-sum(q)*x(4))  + (1-par(3))*p4 + (1-par(3))*p5 - p7;
sys(5)=1/vol*(q*u(5)-sum(q)*x(5))  + p1 + p2 - p4;
sys(6)=1/vol*(q*u(6)-sum(q)*x(6))  + p3 - p5;
if do > 0
        sys(7)=1/vol*(q*u(7)-sum(q)*x(7))  + Kla*(sosat-x(7)) - (1-par(1))/par(1)*p1 +
(1-4.57/par(2))*p3;
else
        sys(7)=0;
end
sys(8)=1/vol*(q*u(8)-sum(q)*x(8))  - (1-par(1))/(2.86*par(1))*p2 + 1/par(2)*p3;
sys(9)=1/vol*(q*u(9)-sum(q)*x(9))  - par(4)*p1 - par(4)*p2 - (par(4)+1/par(2))*p3 + p6;
sys(10)=1/vol*(q*u(10)-sum(q)*x(10)) - p6 + p8;
sys(11)=1/vol*(q*u(11)-sum(q)*x(11))  +  (par(4)-par(3)*par(5))*p4  +  (par(4)-
par(3)*par(5))*p5 - p8;



% end mdlDerivatives
%
%=============================================================
=========

% mdlOutputs
% Return the block outputs.
%=============================================================
=========
```

```
%
function sys=mdlOutputs(t,x,u)

sys = x;
% end mdlOutput
```

A.2.5 Code listing for 2nd AEROBIC tank in the system

```
function [sys,x0,str,ts] = bioreactmod2in(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
switch flag,
%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 11;
sizes.NumInputs = 12;
sizes.DirFeedthrough = 0;
```

```
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,2.8,1559,24.8,1451,51.7,0,7.9,2.2,0.9,1.7];
str = [];
ts = [0 0];
% end mdlInitializeSizes
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys=mdlDerivatives(t,x,u)
% parameters
do=1;
par(1)=0.67;par(2)=0.24;par(3)=0.08;par(4)=0.086;par(5)=0.06;par(6)=6;
par(7)=0.62;par(8)=20;par(9)=0.2;par(10)=0.5;par(11)=0.8;par(12)=0.2;
par(13)=0.4;par(14)=1;par(15)=0.8;par(16)=0.08;par(17)=3;par(18)=0.03;
par(19)=0.4;
%inputs
%x(7)=u(7);
in1=u(1);in2=u(2);in3=u(3);in4=u(4);in5=u(5);
in6=u(6);in7=u(7);in8=u(8);in9=u(9);in10=u(10);in11=u(11);qin=u(12);
% influent concentrations
%exin(1)=20; exin(2)=125; exin(3)=60; exin(4)=175; exin(5)=20; exin(6)=0;
%exin(7)=0; exin(8)=1; exin(9)=30; exin(10)=5; exin(11)=5;
%exin=exin'; % transpose into a column vector

% flow rates and volumes
 qret=qin*0.5; qrec=qin*2;
vol=1333; Vtot = 5999;
% thinkening factor and sludge age – ideal settler
SA=7;
gamma=ones(11,1);
gamma([3:6 11])=(qin+qret-Vtot/SA)/qret * ones(1,5); % particulate states are thickened
gamma(7)=0; % DO concentration in return sluge flow = 0.

%xin = u';
q = [qin+qret+qrec];
%------ AEROBIC ZONE ----------%
% processes, taken from the IAWQ matrix formulation

p1 = par(6) * (x(2)/(par(8)+x(2))) * (x(7)/(par(9)+x(7))) * x(5);%
p2 = par(6) * (x(2)/(par(8)+x(2))) * (par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8))) *
par(15) * x(5);%
p3 = par(11) * (x(9)/(par(14)+x(9))) * (x(7)/(par(13)+x(7))) * x(6);%
```

```
p4 = par(7) * x(5);%
p5 = par(12) * x(6);%
p6 = par(16) * x(10) * x(5);%
p7 = par(17) * ((x(4)/x(5))/(par(18)+(x(4)/x(5)))) * ((x(7)/(par(9)+x(7))) + par(19) *
(par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8)))) * x(5);
p8 = p7*(x(11)/x(4));%
%qair = 7;
%k1=300; k2=-0.42; Kla=k1*(1-exp(k2*qair));
sosat=9.5; %qairtot=qair*4000;
Kla=200;

% aeration parameters and Kla model

% differential equations according to IAWQ matrix and plant layout

sys(1)=1/vol*(q*u(1)-sum(q)*x(1));
sys(2)=1/vol*(q*u(2)-sum(q)*x(2))  - 1/par(1)*p1 - 1/par(1)*p2 + p7;
sys(3)=1/vol*(q*u(3)-sum(q)*x(3))  + par(3)*p4 + par(3)*p5;
sys(4)=1/vol*(q*u(4)-sum(q)*x(4))  + (1-par(3))*p4 + (1-par(3))*p5 - p7;
sys(5)=1/vol*(q*u(5)-sum(q)*x(5))  + p1 + p2 - p4;
sys(6)=1/vol*(q*u(6)-sum(q)*x(6))  + p3 - p5;
if do > 0
        sys(7)=1/vol*(q*u(7)-sum(q)*x(7))  + Kla*(sosat-x(7)) - (1-par(1))/par(1)*p1 +
(1-4.57/par(2))*p3;
else
        sys(7)=0;
end
sys(8)=1/vol*(q*u(8)-sum(q)*x(8))  - (1-par(1))/(2.86*par(1))*p2 + 1/par(2)*p3;
sys(9)=1/vol*(q*u(9)-sum(q)*x(9))  - par(4)*p1 - par(4)*p2 - (par(4)+1/par(2))*p3 + p6;
sys(10)=1/vol*(q*u(10)-sum(q)*x(10)) - p6 + p8;
sys(11)=1/vol*(q*u(11)-sum(q)*x(11))  +  (par(4)-par(3)*par(5))*p4  +  (par(4)-
par(3)*par(5))*p5 - p8;



% end mdlDerivatives
%
%=============================================================
=========
% mdlOutputs
% Return the block outputs.
%=============================================================
=========
%
function sys=mdlOutputs(t,x,u)
sys = x;
```

```
% end mdlOutput
```

A.2.6 Code listing for the 3rd AEROBIC bioreactor in the system

```
function [sys,x0,str,ts] = bioreactmod3in(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
switch flag,
%%%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%%%%%%%%%
case 3,
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%%%
case { 2, 4, 9 },
sys = [];
%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 11;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 11;
sizes.NumInputs = 12;
sizes.DirFeedthrough = 0;
```

```
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [20,2.8,1559,24.8,1451,51.7,0,7.9,2.2,0.9,1.7];
str = [];
ts = [0 0];
% end mdlInitializeSizes
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys=mdlDerivatives(t,x,u)
% parameters
do=1;
par(1)=0.67;par(2)=0.24;par(3)=0.08;par(4)=0.086;par(5)=0.06;par(6)=6;
par(7)=0.62;par(8)=20;par(9)=0.2;par(10)=0.5;par(11)=0.8;par(12)=0.2;
par(13)=0.4;par(14)=1;par(15)=0.8;par(16)=0.08;par(17)=3;par(18)=0.03;
par(19)=0.4;
%inputs
%x(7)=u(7);
in1=u(1);in2=u(2);in3=u(3);in4=u(4);in5=u(5);
in6=u(6);in7=u(7);in8=u(8);in9=u(9);in10=u(10);in11=u(11);qin=u(12);
% influent concentrations
%exin(1)=20; exin(2)=125; exin(3)=60; exin(4)=175; exin(5)=20; exin(6)=0;
%exin(7)=0; exin(8)=1; exin(9)=30; exin(10)=5; exin(11)=5;
%exin=exin'; % transpose into a column vector

% flow rates and volumes
qret=qin*0.5; qrec=qin*2;
vol=1333; Vtot = 5999;
% thinkening factor and sludge age – ideal settler
SA=7;
gamma=ones(11,1);
gamma([3:6 11])=(qin+qret-Vtot/SA)/qret * ones(1,5); % particulate states are thickened
gamma(7)=0; % DO concentration in return sluge flow = 0.

%xin = u';
q = [qin+qret+qrec];
%------ AEROBIC ZONE ----------%
% processes, taken from the IAWQ matrix formulation

p1 = par(6) * (x(2)/(par(8)+x(2))) * (x(7)/(par(9)+x(7))) * x(5);%
p2 = par(6) * (x(2)/(par(8)+x(2))) * (par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8))) *
par(15) * x(5);%
p3 = par(11) * (x(9)/(par(14)+x(9))) * (x(7)/(par(13)+x(7))) * x(6);%
```

```
p4 = par(7) * x(5);%
p5 = par(12) * x(6);%
p6 = par(16) * x(10) * x(5);%
p7 = par(17) * ((x(4)/x(5))/(par(18)+(x(4)/x(5)))) * ((x(7)/(par(9)+x(7))) + par(19) *
(par(9)/(par(9)+x(7))) * (x(8)/(par(10)+x(8)))) * x(5);
p8 = p7*(x(11)/x(4));%
%qair = 7;
%k1=300; k2=-0.42; Kla=k1*(1-exp(k2*qair));
sosat=9.5; %qairtot=qair*4000;
Kla=200;

% aeration parameters and Kla model

% differential equations according to IAWQ matrix and plant layout

sys(1)=1/vol*(q*u(1)-sum(q)*x(1));
sys(2)=1/vol*(q*u(2)-sum(q)*x(2))  - 1/par(1)*p1 - 1/par(1)*p2 + p7;
sys(3)=1/vol*(q*u(3)-sum(q)*x(3))  + par(3)*p4 + par(3)*p5;
sys(4)=1/vol*(q*u(4)-sum(q)*x(4))  + (1-par(3))*p4 + (1-par(3))*p5 - p7;
sys(5)=1/vol*(q*u(5)-sum(q)*x(5))  + p1 + p2 - p4;
sys(6)=1/vol*(q*u(6)-sum(q)*x(6))  + p3 - p5;
if do > 0
        sys(7)=1/vol*(q*u(7)-sum(q)*x(7))  + Kla*(sosat-x(7)) - (1-par(1))/par(1)*p1 +
(1-4.57/par(2))*p3;
else
        sys(7)=0;
end
sys(8)=1/vol*(q*u(8)-sum(q)*x(8))  - (1-par(1))/(2.86*par(1))*p2 + 1/par(2)*p3;
sys(9)=1/vol*(q*u(9)-sum(q)*x(9))  - par(4)*p1 - par(4)*p2 - (par(4)+1/par(2))*p3 + p6;
sys(10)=1/vol*(q*u(10)-sum(q)*x(10)) - p6 + p8;
sys(11)=1/vol*(q*u(11)-sum(q)*x(11))  +  (par(4)-par(3)*par(5))*p4  +  (par(4)-
par(3)*par(5))*p5 - p8;



% end mdlDerivatives
%
%=============================================================
=========
% mdlOutputs
% Return the block outputs.
%=============================================================
=========
%
function sys=mdlOutputs(t,x,u)
sys = x;
```

```
% end mdlOutput
```