Design and Analysis of Power System Stabilizer Using NERO-FUZZY Controller

نموذج التصميم وتحليل استقرار النظام الكهربائي باستخدام التحكم العصبي الغامض

A Thesis Submitted in Partial Fulfillment to the Requirements for the Degree of M.Sc. in Electrical Engineering (Microprocessor and Electronic Control)

Presented by:

Maysoon Eltayeb Abd Al Wahab Abd Alla
(B.Sc. Hons. in Electronics Engineering (control) - Al Neelain University 2003)

Supervised by:

Dr. Awadalla Taifour Ali

April 2014
سورة الفاتحة (1-7)
DEDICATIONS

The words and measures can never express my deepest gratitude to my parents. They have been a force of strength all along, and without them it would have been an uphill task for me to complete this work. I am deeply indebted to my children those who are patient, and I took some of their time in order to complete this project.

Last but not the least, I am deeply indebted to my husband, family and my friends; their incessant support made me achieve new heights in life and built my character and career.
ACKNOWLEDGEMENT

I wish to express my profound gratitude to my Supervisor assistant professor Awadalla Taifor Ali, for his valuable guidance, continuous encouragement, worthwhile suggestions and constructive ideas throughout this research. His support, pragmatic analysis and understanding made this study a success and knowledgeable experience for me.
This thesis presents a comparative study of various controllers for the Power system stabilizer. The most commonly used controller for the Power System Stabilizer is conventional Proportional-Integral-Derivative (PID) controller. However, the PID controller has some disadvantages such as: the high starting overshoot, sensitivity to controller gains and sluggish response due to sudden disturbance. There relatively design PID controller with computational optimization approach method is proposed to overcome the disadvantages of the conventional PID controller. Further, two Fuzzy logic based controllers namely; Fuzzy control and Neuro-fuzzy control are proposed in this study and the performance of these controllers are compared with PID controller performance. Simulation results are presented and analyzed for all controllers. It was observed that Neuro-fuzzy controller gives a better response than other controllers for the Power System Stabilizer.
المستخص ص

تقدم هذه الدراسة مقارنة بين مختلف وحدات التحكم في نظام استقرار الطاقة. أكثر المتحكمات استخداماً للتحكم في نظام استقرار الطاقة هي المتحكمات التقليدية المعروفة بالمتحكم التناسبي التكافلي التفاضلي (PID) اختصاراً على الرقم من أن المتحكم التناسبي التكافلي التفاضلي لديه بعض العيوب مثل: بدأ بمجاوزة عالية للهدف، حساسية للتحكم في الكسب والإستجابة البطيئة للتغيرات المفاجئة عليه فإن تصميم المتحكمPID (بالطريقة الحسابية التقريبية الأمثل للتلعب على بعض عيوب المتحكم التقليدي). عليه تم استخدام إثنين من المتحكمات تعتمد على المناطق الغامض وهي المتحكم الغامض والمتحكم العصبي الغامض أقترحاً في هذه الدراسة للمقارنة مع المتحكم التناسبي التكافلي التفاضلي. تم عرض نتائج المحاكاة وتحليلها لكافة المتحكمات. لوحظ أن المتحكم العصبي الغامض له الاستجابة الأفضل من بين المتحكمات الأخرى للتحكم في نظام استقرار الطاقة.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>آلية</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEDICATIONS</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>ﺑﻤﺴﺘﺨﻠﺺ</td>
<td>v</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xiii</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xiv</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>xv</td>
</tr>
</tbody>
</table>

## CHAPTER ONE: INTRODUCTION

<table>
<thead>
<tr>
<th>1.1 General overview</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 Statement of Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Methodology</td>
<td>2</td>
</tr>
<tr>
<td>1.5 Layout</td>
<td>2</td>
</tr>
</tbody>
</table>

## CHAPTER TWO: THEORETICAL BACKGROUND AND LITERATURE REVIEW

<table>
<thead>
<tr>
<th>2.1 Overview</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1 Conventional Power System Stabilizer</td>
<td>3</td>
</tr>
<tr>
<td>2.1.2 Construction of Power System Stabilizer</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Proportional Integral Derivation Controller</td>
<td>6</td>
</tr>
<tr>
<td>Section</td>
<td>Content</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Proportional term</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Integral term</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Derivative term</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Stability</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Tuning Methods of PID</td>
</tr>
<tr>
<td>2.2.5.1</td>
<td>Manual tuning</td>
</tr>
<tr>
<td>2.2.5.2</td>
<td>Ziegler–Nichols method</td>
</tr>
<tr>
<td>2.2.5.3</td>
<td>PID tuning software</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Controller types</td>
</tr>
<tr>
<td>2.3</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Fuzzy sets, membership functions and logical operators</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Linguistic variable and rules bases</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Fuzzy logic models</td>
</tr>
<tr>
<td>2.3.3.1</td>
<td>Mamdani modeling</td>
</tr>
<tr>
<td>2.3.3.2</td>
<td>Sugeno modeling</td>
</tr>
<tr>
<td>2.4</td>
<td>Fuzzy Controller</td>
</tr>
<tr>
<td>2.4.1</td>
<td>P Fuzzy controller</td>
</tr>
<tr>
<td>2.4.2</td>
<td>PD Fuzzy controller</td>
</tr>
<tr>
<td>2.4.3</td>
<td>PI Fuzzy controller</td>
</tr>
<tr>
<td>2.4.4</td>
<td>PID Fuzzy Controller</td>
</tr>
<tr>
<td>2.5</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Single-input</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Neural control</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Neural networks in direct neural control</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Neural networks in indirect neural control</td>
</tr>
</tbody>
</table>
2.6 Neuro-Fuzzy Systems ................................................................. 30
2.6.1 Adaptive network fuzzy inference systems ............................... 30
2.6.2 Neuro – fuzzy controller .......................................................... 31
2.6.3 Adaptive Network Fuzzy Inference System (ANFIS) as an estimator…. 31

CHAPTER THREE: CONTROL SYSTEM DESIGN

3.1 Introduction .................................................................................... 35
3.2 Power System Stabilizer Parameters ........................................... 35
3.3 Design of Fuzzy adaptive controller ............................................ 35
3.3.1 Fuzzy control .............................................................................. 37
3.3.2 Fuzzy controller design .............................................................. 38
3.3.3 Structure of fuzzy logic .............................................................. 39
3.3.3.1 Pre-processing ....................................................................... 39
3.3.3.2 Fuzzification .......................................................................... 39
3.3.3.3 Rule Base ................................................................................ 40
3.3.3.4 Defuzzification ...................................................................... 40
3.3.3.5 Post processing ...................................................................... 40
3.3.4 Fuzzy Logic Toolbox ................................................................. 40
3.3.4.1 FIS Editor .............................................................................. 41
3.3.4.2 Membership function editor ................................................... 42
3.3.4.3 Rule Editor ............................................................................. 45
3.3.4.4 Rule Viewer ........................................................................... 46
3.3.4.5 Surface viewer ....................................................................... 47
3.4 Linear Fuzzy PID Control ............................................................. 47
3.4.1 Fuzzy Proportional controller ................................................... 48
3.4.2 Fuzzy proportional- derivative controller ................................... 49

viii
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.1)</td>
<td>Conventional Power System Stabilizer</td>
<td>5</td>
</tr>
<tr>
<td>(2.2)</td>
<td>PID controller</td>
<td>7</td>
</tr>
<tr>
<td>(2.3)</td>
<td>fuzzy logic</td>
<td>13</td>
</tr>
<tr>
<td>(2.4)</td>
<td>Boolean logic</td>
<td>13</td>
</tr>
<tr>
<td>(2.5)</td>
<td>Example fuzzy set</td>
<td>14</td>
</tr>
<tr>
<td>(2.6)</td>
<td>Sample Gaussian shaped MF</td>
<td>15</td>
</tr>
<tr>
<td>(2.7)</td>
<td>Mapping of input space to output space</td>
<td>17</td>
</tr>
<tr>
<td>(2.8)</td>
<td>Mamdani Fuzzy Control System</td>
<td>18</td>
</tr>
<tr>
<td>(2.9)</td>
<td>Diagram showing aggregation and defuzzification</td>
<td>20</td>
</tr>
<tr>
<td>(2.10)</td>
<td>Implementation of Sugeno model</td>
<td>21</td>
</tr>
<tr>
<td>(2.11)</td>
<td>Fuzzy controller for a plant</td>
<td>22</td>
</tr>
<tr>
<td>(2.12)</td>
<td>PD-like fuzzy control system</td>
<td>23</td>
</tr>
<tr>
<td>(2.13)</td>
<td>PI-like fuzzy control system</td>
<td>24</td>
</tr>
<tr>
<td>(2.14)</td>
<td>PI-like fuzzy control system</td>
<td>25</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>(2.15)</td>
<td>The structure for a PID-like fuzzy controller</td>
<td>19</td>
</tr>
<tr>
<td>(2.16)</td>
<td>Single - input neuron</td>
<td>28</td>
</tr>
<tr>
<td>(2.17)</td>
<td>Direct design</td>
<td>30</td>
</tr>
<tr>
<td>(2.18)</td>
<td>Neuro – Fuzzy Control scheme</td>
<td>32</td>
</tr>
<tr>
<td>(3.1)</td>
<td>Fuzzy controller architecture</td>
<td>36</td>
</tr>
<tr>
<td>(3.2)</td>
<td>FIS editor</td>
<td>38</td>
</tr>
<tr>
<td>(3.3)</td>
<td>Fuzzy controller</td>
<td>40</td>
</tr>
<tr>
<td>(3.4)</td>
<td>Structure of fuzzy logic controller</td>
<td>40</td>
</tr>
<tr>
<td>(3.5)</td>
<td>Fuzzy inference system</td>
<td>42</td>
</tr>
<tr>
<td>(3.6)</td>
<td>Membership Function Editor</td>
<td>43</td>
</tr>
<tr>
<td>(3.7)</td>
<td>FIS Editor for Membership Function variable</td>
<td>44</td>
</tr>
<tr>
<td>(3.8)</td>
<td>membership function editor for input 1&quot;e&quot;</td>
<td>45</td>
</tr>
<tr>
<td>(3.9)</td>
<td>membership function editor for input 2&quot;ce&quot;</td>
<td>45</td>
</tr>
<tr>
<td>(3.10)</td>
<td>membership function editor for output &quot;ci&quot;</td>
<td>45</td>
</tr>
<tr>
<td>(3.11)</td>
<td>Rule Editor</td>
<td>46</td>
</tr>
<tr>
<td>(3.12)</td>
<td>Rule Viewer</td>
<td>47</td>
</tr>
<tr>
<td>(3.13)</td>
<td>Surface viewer</td>
<td>48</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>(3.14)</td>
<td>Fuzzy proportional controller</td>
<td>49</td>
</tr>
<tr>
<td>(3.15)</td>
<td>fuzzy PD Controller (FPD)</td>
<td>50</td>
</tr>
<tr>
<td>(3.16)</td>
<td>Fuzzy PD+I controller (FPD+I)</td>
<td>51</td>
</tr>
<tr>
<td>(3.17)</td>
<td>ANFIS editor</td>
<td>52</td>
</tr>
<tr>
<td>(3.18)</td>
<td>ANFIS model structure</td>
<td>52</td>
</tr>
<tr>
<td>(4.1)</td>
<td>Model of PID controller for power system stabilizer</td>
<td>53</td>
</tr>
<tr>
<td>(4.2)</td>
<td>Unit step response of PID controller for power system stabilizer</td>
<td>54</td>
</tr>
<tr>
<td>(4.3)</td>
<td>Control signal of PID controller</td>
<td>54</td>
</tr>
<tr>
<td>(4.4)</td>
<td>Simulink diagram of PD-like Fuzzy for power system stabilizer</td>
<td>55</td>
</tr>
<tr>
<td>(4.5)</td>
<td>Unit step response of FLC for power system stabilizer</td>
<td>55</td>
</tr>
<tr>
<td>(4.6)</td>
<td>Control signal of Fuzzy controller</td>
<td>56</td>
</tr>
<tr>
<td>(4.7)</td>
<td>Block diagram of Neuro-Fuzzy controller</td>
<td>56</td>
</tr>
<tr>
<td>(4.8)</td>
<td>Unit step response of Neuro-Fuzzy controller for system</td>
<td>57</td>
</tr>
<tr>
<td>(4.9)</td>
<td>Control signal of Neuro-Fuzzy controller</td>
<td>57</td>
</tr>
<tr>
<td>(4.10)</td>
<td>Unit step system response for all controllers</td>
<td>58</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.1)</td>
<td>Methods tuning a PID</td>
<td>9</td>
</tr>
<tr>
<td>(2.2)</td>
<td>Effects of increasing a parameter independently</td>
<td>10</td>
</tr>
<tr>
<td>(2.3)</td>
<td>Ziegler–Nichols method</td>
<td>11</td>
</tr>
<tr>
<td>(3.1)</td>
<td>Fuzzy rule base table</td>
<td>46</td>
</tr>
<tr>
<td>(4.1)</td>
<td>Comparison Between PIDPSS, FPSS and NFPSS</td>
<td>59</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSS</td>
<td>Power System Stabilizer</td>
</tr>
<tr>
<td>ANFIS</td>
<td>Adaptive Neural Fuzzy Interface System</td>
</tr>
<tr>
<td>FIS</td>
<td>Fuzzy Interface System</td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>FLC</td>
<td>Fuzzy Logic Control</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>MF</td>
<td>Membership Function</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integral Derivative controller</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
</tbody>
</table>
# LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>Transfer function</td>
</tr>
<tr>
<td>( K_p )</td>
<td>Proportional gain</td>
</tr>
<tr>
<td>( K_i )</td>
<td>Integration gain</td>
</tr>
<tr>
<td>( K_d )</td>
<td>Derivative gain</td>
</tr>
<tr>
<td>( Ce )</td>
<td>Error change</td>
</tr>
<tr>
<td>MO</td>
<td>Maximum Overshoot</td>
</tr>
<tr>
<td>( T_S )</td>
<td>Settling Time</td>
</tr>
<tr>
<td>Tr</td>
<td>Rise Time</td>
</tr>
</tbody>
</table>
CHAPTER ONE

INTRODUCTION

1.1 General overview

Power stabilizers are used to produce controlling signals for the system and to decrease the oscillation of low frequency power system. A variety of methods are proposed to overcome the faults of common power system stabilizer of which can named fuzzy logic, genetic algorithm, neural networks, PID and PI controller. The project regarded the Neuro-fuzzy controller to control the system based on simulations, it is defined that the Neuro-fuzzy controller make better answers to common stabilizer.

Fuzzy Logic Controller (FLC) is chosen also as a controller for this project because it has several advantages compared to the other classical controller. The advantages of FLC are such as simplicity of control, low cost and the possibility to design without knowing the exact mathematical model of the process. Nero Fuzzy also used as a controller because of it advantages that a combination of neural networks and fuzzy control enhances the performance of the controller.

1.2 Statement of the problem

Power system stabilizers (PSS) must be capable of providing appropriate stabilization signals over a broad range of operating conditions and disturbances. A traditional PSS provides a positive damping torque in phase with the speed signal to cancel the effect of the system negative damping torque. Because the gains of this controller are determined for a particular operating condition, they may not be valid for a wide range of operating conditions. Considerable efforts have been directed towards developing adaptive PSS in an attempt to cover a wide range of operating conditions.
1.3 Objectives

- To cover a wide range of operating conditions.
- Several controllers have been developed for PSS.
- The proposed method also pursues small signal stability analysis which provides the opportunity to design a system with adjustable controller parameters.

1.4 Methodology

Literature review of related study to understand the fuzzy logic controller, neuro-fuzzy control and its application. The mathematical and computer models of PSS under study are developed. MATLAB- GUI simulation software is used to obtain the response of the system.

1.5 Layout

This thesis consists of five chapters including chapter one. Chapter one gives an introduction to the principles of work, the reasons and motivation and also discusses the objectives and outlines methodologies of the study. Chapter two gives a theoretical background of Power System Stabilizer, PID controller, Fuzzy and Neuro fuzzy system. Chapter three presents the system control design of power system stabilizer system with Fuzzy controller and Neuro-Fuzzy controller. Chapter four presents the simulation results obtained and comparison between different results. Finally, Chapter five provides the conclusions and recommendations.
CHAPTER TWO
THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 Overview

Power systems have usually many disturbances. These disturbances lead to bringing low frequency oscillation in power system. The dynamic unstability is caused if the torque of the system is not sufficient. In the past two decades, the controlling signal complementing stimulate generator was used in order to improving dynamic stability.

Today, classic PSS is widely used with stimulate system in powerhouse. The modulation of classic stabilizer coefficient is based on a linear model of system. Approximation of the plant of power system components is changed in the conditions of system construction due to the faults and operation of protection relay. The classic stabilizers would not offer high efficiency in idle range of functioning. In recent years, different methods are proposed based on controlling techniques of nonlinear systems, such as adaptive controlling techniques and artificial intelligence techniques to design power system stabilizer. Recent development in design and construction of stimulation system, not only made the application of these techniques in real system possible, but also made it easy to use. The techniques based on artificial intelligence include fuzzy logic, application of artificial neural networks and intelligent searching algorithm such as genetic algorithm, tabor searching algorithm and the algorithm of developing particle swarm [1].

2.1.1 Conventional power system stabilizer

The Conventional Power System Stabilizer is modeled by nonlinear system as shown in Figure (2.1). To ensure a robust damping, the conventional PSS should provide a moderate phase advance at frequencies of interest in order to compensate
for the inherent lag between field excitation and electrical torque induced by the conventional PSS action. The model shown in Figure (2.1) consists of a low-pass filter, a general gain, a washout high-pass filter, a phase-compensation system and an output limiter. The general gain $K$ determines the amount of damping produced by the stabilizer. The washout high-pass filter eliminates low frequencies that are present in the $dw$ signal and allows the PSS to respond only to speed changes. The phase-compensation system is represented by a cascade of two first-order lead-lag transfer functions used to compensate the phase lag between the excitation voltage and the electrical torque of the synchronous machine.

![Figure 2.1: Conventional Power System Stabilizer](image)

Two types of signals can be used at the input, the synchronous machine speed deviation $dw$ signal (in p.u.) and the synchronous machine acceleration power $dPa$, the output is the stabilization voltage (in p.u.). Connecting the $V_{stab}$ to the input of the excitation system is used to control the terminal voltage of the synchronous machine [2].

### 2.1.2 Construction of power system stabilizer

Power system stabilizer is a control system whose output is utilized as an input to the voltage regulator of the excitation system of a generator. Since PSS work on only one element of the electrical spring mass system, it is necessary to have a large number of PSSs working in unison to develop the necessary composite increase in damping required for stable operation. The ability of PSS to provide an input to the voltage regulator which can result in an increase in system damping is dependent on more than just the ability of PSS to sense the oscillation. It must be capable of modifying the generator field so that generator terminal voltage varies in phase with system frequency swings.
A step change into the voltage regulator circuit will not result in an instantaneous change in generator terminal voltage. Excitation systems are designed to compensate for the delays of generators via parameter variation. The generator response is a then a function of the time constant of the generator and excitation system. The ability of PSS to improve system damping is dependent on the excitation system ability to provide a quick generator response.

PSS consists of input modules, washout, compensation, gain, limit, and protection to provide a control system which is flexible and reliable.

- **Input**

PSS must first detect when a power swing is taking. This detection is accomplished using several possible inputs: generator terminal voltage, stator current or rotor speed. These inputs are translated into frequency deviation, power deviation or a combination of both.

- **Washout or reset time constant**

The purpose of washout module in a PSS is to minimize the impact of off-nominal frequency operation on the PSS output signal and remove the effects of drift. In absence of washout module any off-nominal frequency operation of the system will produce a corresponding dc output from the transducer and propagates to the voltage regulator through PSS.

- **Phase compensation**

The fundamental function of the phase shifting network in the PSS is to compensate the phase lag in response of the excitation system being controlled. For example, when the response of the machine terminal voltage lags the input voltage of the voltage regulator by 90 degrees, the power system stabilizer should provide 90 degrees of phase lead at the same frequency.
• **Gain**

The gain of the PSS is the ratio of the change in generator terminal voltage to the change of system frequency. For example, again of four would be reflected by a change in terminal voltage of 4 percent for a 1 percent change of speed or frequency (0.6HZ)

### 2.2 Proportional integral derivation controller

The Proportional Integral Derivation (PID) Control scheme shown in Figure (2.2) is named after its three correcting terms, whose sum constitutes the Manipulated Variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller

---

![PID controller diagram](image)

**Figure (2.2): PID controller**

Defining $y(t)$ as the controller output final form of the PID algorithm is:

$$y(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

... (2.1)

where:

- $K_p$ is proportional gain
- $K_i$ is integral gain
- $K_d$ is derivation gain
- $e(t)$ is the error

---
2.2.1 Proportional term

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant $K_p$, called the proportional gain constant.

The proportional term is given by:

$$ P_{out} = K_p e(t) $$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable, contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.

2.2.2 Integral term

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain ($K_i$) and added to the controller output.

The integral term is given by:

$$ I_{out} = K_i \int_0^t e(t) dt $$

The integral term accelerates the movement of the process towards set point and eliminates the residual steady-state error that occurs with a pure proportional
controller. However, since the integral term responds to accumulated errors from the past, it causes the present value to overshoot the set point value.

### 2.2.3 Derivative term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain $K_d$. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain $K_d$

The derivative term is given by:

$$D_{out} = K_d \frac{d}{dt} e(t)$$  \hspace{1cm} \text{(2.4)}

The derivative term slows the rate of change of the controller output. Derivative control is used to reduce the magnitude of the overshoot produced by the integral component and improve the combined controller-process stability. However, the derivative term slows the transient response of the controller. Also, differentiation of a signal amplifies noise and thus this term in the controller is highly sensitive to noise in the error term, and can cause a process to become unstable if the noise and the derivative gain are sufficiently large. Hence an approximation to a differentiator with a limited bandwidth is more commonly used [3].

### 2.2.4 Stability

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e., its output diverges, with or without oscillation, and is limited only by saturation or mechanical breakage. Instability is caused by excess gain.

Generally, stabilization of response is required and the process must not oscillate for any combination of process conditions and set points, though sometimes marginal stability (bounded oscillation) is acceptable or desired.
2.2.5 Tuning methods of PID

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, and then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters [3]. Table (2.1) shows advantage and disadvantage of methods tuning of PID.

Table (2.1) Methods of tuning a PID

<table>
<thead>
<tr>
<th>Choosing a tuning method</th>
<th>Disadvantages</th>
<th>Advantages</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual tuning</td>
<td>Requires experienced personnel.</td>
<td>No math required; online.</td>
<td></td>
</tr>
<tr>
<td>Ziegler–Nichols</td>
<td>Process upset, some trial-and-error, very aggressive tuning.</td>
<td>Proven method; online.</td>
<td></td>
</tr>
<tr>
<td>Software tools</td>
<td>Some cost and training involved.</td>
<td>Consistent tuning; online or offline; may include valve and sensor analysis; allows simulation before downloading; can support non-steady-state (NSS) tuning.</td>
<td></td>
</tr>
<tr>
<td>Cohen–Coon</td>
<td>Some math; offline; only good for first-order processes.</td>
<td>Good process models.</td>
<td></td>
</tr>
</tbody>
</table>
2.2.5.1 Manual tuning

If the system must remain online, one tuning method is to first set $K_i$ and $K_d$ values to zero. Increase the $K_p$ until the output of the loop oscillates, then the $K_p$ should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase $K_i$ until any offset is corrected in sufficient time for the process. However too much $K_i$ will cause instability. Finally, increase $K_d$, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much $K_d$ will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the set point more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a $K_p$ setting significantly less than half that of the $K_p$ setting that was causing oscillation.[citation needed] Table (2.2) shows effects of increasing a parameter independently.

Table (2.2) Effects of increasing a parameter independently

<table>
<thead>
<tr>
<th>Stability</th>
<th>Steady-state error</th>
<th>Settling time</th>
<th>Overshoot</th>
<th>Rise time</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degradese</td>
<td>Decrease</td>
<td>Small change</td>
<td>Increase</td>
<td>Decrease</td>
<td>$K_p$</td>
</tr>
<tr>
<td>Degradese</td>
<td>Eliminate</td>
<td>Increase</td>
<td>Increase</td>
<td>Decrease</td>
<td>$K_i$</td>
</tr>
<tr>
<td>Improve if $K_d$small</td>
<td>No effect in theory</td>
<td>Decrease</td>
<td>Decrease</td>
<td>Minor change</td>
<td>$K_d$</td>
</tr>
</tbody>
</table>
2.2.5.2 Ziegler–Nichols method

Another heuristic tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols in the 1940s[4]. As in the method above, the $K_i$ and $K_d$ gains are first set to zero. The P gain is increased until it reaches the ultimate gain, $K_u$, at which the output of the loop starts to oscillate. $K_u$ and the oscillation period $p_u$ are used to set the gains as shown in Table (2.3):

Table (2.3) Ziegler–Nichols method

<table>
<thead>
<tr>
<th></th>
<th>$K_d$</th>
<th>$K_i$</th>
<th>$K_p$</th>
<th>Control Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>0.50 $K_u$</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1.2$K_p/P_u$</td>
<td>0.45 $K_u$</td>
<td>PI</td>
</tr>
<tr>
<td>$K_pP_u/8$</td>
<td>2$K_p/P_u$</td>
<td>0.60 $K_u$</td>
<td>PID</td>
<td></td>
</tr>
</tbody>
</table>

These gains apply to the ideal, parallel form of the PID controller. When applied to the standard PID form, the integral and derivative time parameters $T_i$, and $T_d$, are only dependent on the oscillation period $p_u$.

2.2.5.3 PID tuning software

Most modern industrial facilities no longer tune loops using the manual calculation methods shown above. Instead, PID tuning and loop optimization software are used to ensure consistent results. These software packages will gather the data, develop process models, and suggest optimal tuning. Some software packages can even develop tuning by gathering data from reference changes [3].

2.2.6 Controller types

For a given control task, it is obviously not necessary to adopt all the three actions. Thus, the choice of the controller type is an integral part of the overall controller design, taking into account that the final aim is to obtain the best cost/benefit ratio and therefore the simplest controller capable to obtain a satisfactory performance should be preferred [4].

2.3 Fuzzy Logic

The Concept of fuzzy logic was developed in 1964 by Professor Lotfi A. Zadeh at the University of California; it emerged from the development of the theory of fuzzy sets [5]. As shown in the Figure (2.3) and Figure (2.4) the fuzzy logic is a superset of conventional (Boolean) logic. This logic was developed to solve mathematical problems which conventional logic couldn’t handle because of its confinement to absolute true or false values. The advent of this logic was inhibited for some time as it wasn’t quickly accepted by some scientists, who stated that it threatened the integrity of scientific thought. However as time passed the acceptance of the logic grew and its applications became numerous. The logic provides a way to handle the concept of partial truth, by assigning different degrees of absolute truth and absolute false, these values are between zero and one.
2.3.1 Fuzzy sets, membership functions and logical operators

Fuzzy sets are sets without clear or crisp boundaries. The elements they contain may only have a partial degree of membership; they are therefore not the same as classical sets in the sense that the sets are not closed. Some examples of vague fuzzy sets and their respective units include the following: Loud noises (sound intensity), Ambient (brightness), High speeds (velocity), Desirable actions (decision of control space). Fuzzy sets can be combined through fuzzy rules to represent specific actions/behavior and it is this property of fuzzy logic that will be utilized when implementing a fuzzy logic controller in subsequent chapters. A membership function (MF) is a curve that defines how each point in the input space is mapped to the set of all real numbers from 0 to 1. A classical set may be for example written as $A = \{X \mid X > 3\}$. Now if $X$ is the universe of discourse with elements $x$ then a fuzzy set $A$ in $X$ is defined as a set of ordered pairs $A = \{ x, \mu A(x) \mid x \in X \}$. Note that in the above expression $\mu A(x)$ may be called the membership function of $x$ in $A$ and that each element of $X$ is mapped to a membership value between 0 and 1. Typical membership function shapes include triangular, trapezoidal and Gaussian functions. The shape is chosen on the basis of how well it describes the set it represents. Figure (2.5) and Figure (2.6) some example fuzzy sets can be observed.

![Figure (2.5): Example fuzzy set.](image120x145.png)
Fuzzy logic reasoning is a superset of standard Boolean logic yet it still needs to use logical operators such as AND, OR and NOT. Firstly note that fuzzy logic differs from Boolean yes/no logic in that although TRUE is given a numerical value ‘1’ and FALSE numerical values ‘0’, other intermediate values are also allowed. For example the values 0.2 and 0.8 can represent both not-quite-false and not-quite-true respectively.

It will be necessary to do logical operations on these values that lie in the [0,1] set, but two-valued logic operations like AND, OR and NOT are incapable of doing this. For this functionality, the functions min, max and additive complement will have to be used.

### 2.3.2 Linguistic variable and rules bases

Linguistic variables are values defined by fuzzy sets. A linguistic variable such as ‘High Speed’ for example could consist of numbers that are equal to or between 50km/h and 80km/h. The conditional statements that make up the rules that govern fuzzy logic behavior use these linguistic variables and have an if-then syntax. These if–then rules are what make up fuzzy rule bases. A sample if-then rule where A and B represent linguistic variables could be:

\[
\text{if } x \text{ is } A \text{ then } y \text{ is } B
\]

The statement is understood to have both a premise, if ‘x is A’, and a conclusion, then ‘y is B’. The premise also known as the antecedent returns a single number
between 0 and 1 whereas the conclusion also known as the consequent assigns the fuzzy set B to the output variable y. Another way of writing this rule using the symbols of assignment

\( (=) \) and equivalence \( (\equiv) \) is:

\[
\text{if } x = A \text{ then } y = B
\]

(2.6)

An if-then rule can contain multiple premises or antecedents. For example, if velocity is high and road is wet and brakes are poor then…

Similarly, the consequent of a rule may contain multiple parts. If temperature is very hot then fan is on and throughput is reduced interpreting these rules involves a number of distinct steps: Firstly, the inputs must be fuzzified; to do this all fuzzy statements in the premise are resolved to a degree of membership between 0 and 1. This can be thought of as the degree of support for the rule. At a working level this means that if the antecedent is true to some degree of membership, then the consequent is also true to that same degree. Secondly, fuzzy operators are applied for antecedents with multiple parts to yield a single number between 0 and 1. Again this is the degree of support for the rule. Thirdly, the result is applied to the consequent, this step is also known as implication. The degree of support for the entire rule is used to shape the output fuzzy set. The outputs of fuzzy sets from each rule are aggregated into a single output fuzzy set. This final set is evaluated (or defuzzified) to yield a single number.

Immediate advantages of this approach become apparent. Fuzzy sets can be combined using fuzzy rules to define system behavior and thus complex non-linear systems can be expressed linguistically. In fact, as will be shown later, rule tables can represent fuzzy controllers.

The process of fuzzifying a single crisp input, applying fuzzy operators and then defuzzifying to produce a single crisp output is known as fuzzy inference. This progression of modeling is discussed in detail in the next section.
2.3.3 Fuzzy logic models

Standard control techniques use numerical data to relate input and output signals. In Figure (2.7) fuzzy logic systems can use both numerical information and linguistic information to perform a mapping of an input to an output space.

Many different control mechanisms could reside within the black box but in this case the mechanism will be confined to a fuzzy logic system. Since the objective is to map inputs to outputs it becomes possible to model non-linear systems, even complex ones.

This is one of fuzzy logics greatest advantages. Differently fuzzy logic systems are tolerant of imprecise data. When considered this suits many real-world applications well because as real-world systems become increasingly complex often the need for highly precise data decreases. The rules that govern this mapping can be acquired through two methods. The first is a method called the direct approach and the second is by using system identification.

The direct approach involves the manual formulation of linguistic rules by a human expert. These rules are then converted into a formal fuzzy system model. The problem with this approach is that unless the human expert knows the system well it is very difficult to design a fuzzy rule base and inference system that is workable, let
alone efficient. For complex systems (non-linear for example) tuning these membership functions would require the adjustment of many parameters simultaneously. Understandably no human expert could accomplish this. Fuzzy models that are designed using system identification are based on the use of input/output data. System identification was introduced to overcome the difficulties involved in the direct approach of choosing the fuzzy set’s membership functions using a search/optimization technique to aid the selection.

The system identification method is covered in Chapter four. All of the previous elements of fuzzy logic that have been discussed up to this point are put together to form a fuzzy inference system (FIS). Two main types of fuzzy inference system exist – the Mamdani and Sugeno type, they are both introduced in the following sections.

### 2.3.3.1 Mamdani modeling

Owing its name to Ebrahim Mamdani the Mamdani model was the first efficient fuzzy logic controller designed and was introduced in 1975[5]. As shown in the Figure (2.8) below the controller consists of a fuzzifier, fuzzy rule base, an inference engine and a defuzzifier.

![Figure (2.8): Mamdani Fuzzy Control System](image-url)
Conventional control systems require crisp outputs to result from crisp inputs. The above representation shows how a crisp input in R can be operated on by a fuzzy logic system to yield a crisp output in Q.

The Mamdani controller is realized using the following steps: Firstly, Fuzzification of Inputs the fuzzifier maps crisp input numbers into fuzzy sets. The value between 0 and 1 each input is given represents the degree of membership that input has within these output fuzzy sets. Fuzzification can be implemented using lookup tables or as in this report, using membership functions. Second, Application of Fuzzy Operators, in the case where multiple statements are used in the antecedent of a rule, it is necessary to apply the correct fuzzy operators as explained above. This allows the antecedent to be resolved to a single number that represents the strength of that rule. Third, Application of Implication Method; this part of the Mamdani system involves defining the consequence as an output fuzzy set. This can only be achieved after each rule has been evaluated and is allowed contribute its ‘weight’ in determining the output fuzzy set. Fourth, Aggregation of all Outputs; the fuzzy outputs of each rule needs to be combined in a meaningful way to be of any use. Aggregation is the method used to perform this by combining each output set into a single output fuzzy set. The order of rules in the aggregation operation is unimportant as all rules are considered.

The three methods of aggregation available for use include sum (sum of each rules output set), max (maximum value of each rule output set) and the probabilistic OR method (the algebraic sum of each rules output set). Fifth, Defuzzification of Aggregated Output, the aggregated fuzzy set found in the previous step is the input to the defuzzifier.

As indicated in the model shown in Figure (2.9) this aggregated fuzzy set in Q is mapped to a crisp output point in Q. This crisp output is a single number that can usefully be applied in controlling the system.
A number of methods of defuzzification are possible and these include the mean of maximum, largest of maximum, smallest of maximum, bisector and centroid (centre of area) methods. The centroid method is the most widely used.

Figure (2.9): Diagram showing aggregation and defuzzification

2.3.3.2 Sugeno modeling

In 1985, Takagi and Sugeno modified the consequence of implication from fuzzy sets to linear functions and developed the so-called "Takagi-Sugeno fuzzy systems" which were applied to parking control of a model car. The format of their fuzzy rules is

\[ \text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ and...and } x_n \text{ is } A_n \text{, then } y = a_0 + a_1 x_1 + ..... + a_n x_n \ldots \ldots \ldots (2.7) \]

The structure of these systems varies significantly from that of the previous ones (the conventional ones). As a consequence of implication, they contain a linear function by which the output can be computed..) The aim of the linear function in Takagi-Sugeno fuzzy systems is to describe the local linear behavior of the system. Fuzziness, which appears only in the premise part of the fuzzy rule, indicates the uncertainty about which the output range of the linear function varies.

Takagi-Sugeno fuzzy systems have a number of advantages by their nature. The systems can be easily understood and the local system equations can be directly
related to the local behavior of the system. Each local system can be clearly described and the dynamics are separately modeled. Takagi-Sugeno fuzzy systems include two kinds of knowledge: one is the qualitative knowledge represented by the if-then rules, and the other is the quantitative knowledge represented by the local functions. The systems allow us to formulate these two kinds of knowledge into unified mathematical framework [9]. In many respects it is identical to the Mamdani method except that the output membership functions for the Sugeno method are always linear or constant. The output membership functions can be thought of as singleton spikes that undergo a simple aggregation instead of other aggregation methods such as max, or sum. Figure (2.10) shows the application of three basic rules for a Sugeno model, All three rules have been written using the OR connector for example.

\[ \text{if } \text{input} \_1 \text{ is } x \text{ or } \text{input} \_2 \text{ is } y \text{ then } \text{output} \_1 \text{ is } z \ldots \ldots \ldots \ldots (2.8) \]

![Figure (2.10): Implementation of Sugeno model](image)

The method of defuzzification is the weighted average (as marked by the thin line in the bottom right corner of the figure). The output is always a single number and in this case having the inputs 0.5 input1, 0.8 input2 and the output is 6.64 output1. The Sugeno system is computationally efficient and its ability to interpolate multiple linear models makes it particularly suited to modeling non-linear systems.
2.4 Fuzzy Controller

Fuzzy control was first introduced and applied in an attempt to design controllers for systems that are structurally difficult to model. Since then, fuzzy control has become one of the most active and fruitful research areas in fuzzy set theory, and many practical applications to industrial processes.

The primary objective of fuzzy logic is to map an input space to an output space. The way of controlling this mapping is to use if-then statements known as rules. The order these rules are carried out insignificant since all rules run concurrently.

A fuzzy controller can be constructed using error, change of error and sum of error \( (e, \Delta e \text{ and } \Sigma e) \) as inputs and control input \( u \) as output depending on the type of controller e.g. PD-, PI- or PID-type. Figure (2.11) shows the block diagram of a typical 3-input single-output fuzzy controller for plant.

![Figure (2.11): Fuzzy controller for a plant](image)

To construct a PID-like fuzzy controller the input and output variables and the rules of the controller should be chosen properly.

If one has made a choice of designing a P-, PD-, PI- or PID like fuzzy controller, this already implies the choice of process state and control output variables, as well as the content of the rule-antecedent and the rule-consequent parts for each rule. The process state variables representing the contents of the rule-antecedent (if-part of a rule) are selected among: error signal \( e \), change-of-error \( \Delta e \), sum-of-errors or an integral error denoted by \( \Sigma e \). The control output (process input) variables
representing the contents of the rule-consequent (then-part of the rule) are selected among: change-of-control output $\Delta u$ or control output $u$. The error is the difference between the desired output of the object or process under control or the set-point and the actual output. This is one of the basic milestones in conventional feedback control [5].

\[ e(t) = y_{sp} - y_{act} \] \hspace{1cm} (2.9)

\[ \Delta e(t) = e(t) - e(t-1) \] \hspace{1cm} (2.10)

\[ \Delta u(t) = u(t) - u(t-1) \] \hspace{1cm} (2.11)

2.4.1 P fuzzy controller

P-like fuzzy controller, the deviation can be considered as an error and the turn as a control output. Then this table represents a choice of the control output roughly proportional to the error that means a P-like controller designed.

2.4.2 PD fuzzy controller

To design the PD-like Fuzzy controller shown in Figure (2.12), the equation giving a conventional PD-controller is

\[ u(k) = K_p \times e(t) + K_d \times \Delta e(t) \] \hspace{1cm} (2.12)

![Figure (2.12): PD-like fuzzy control system](image_url)

The PD controller for any pair of the values of error ($e$) and change-of-error $\Delta e(t)$ calculates the control signal ($u$). The fuzzy controller should do the same thing. For
any pair of error and change-of-error, it should work out the control signal. Then a PD-like fuzzy controller consists of rules, and a symbolic description of each rule is given as if \( e(t) \) is \(<\text{property symbol}>\) and \( \Delta e(t) \) is \(<\text{property symbol}>\) then \( u(t) \) is \(<\text{property symbol}>\), where \(<\text{property symbol}>\) is the symbolic name of a linguistic value.

The natural language equivalent of the above symbolic description reads as follows, for each sampling time \( t \). If the value of error is \(<\text{linguistic value}>\) and the value of change of- error is \(<\text{linguistic value}>\) then the value of control output is \(<\text{linguistic value}>\). We will omit the explicit reference to sampling time \( t \), since such a rule expresses a casual relationship between the process state and control output variables, which holds for any sampling time \( t \). The symbolic name of a linguistic value mean that this is one of the linguistic qualifiers, determined for the proper variable: error, change-of-error or control signal, for example: high, low, medium, etc.

### 2.4.3 PI fuzzy controller

PI-like fuzzy controller, A block diagram for a fuzzy control system looks like Figure (2.13) the equation giving a conventional PI-controller is

\[
u(t) = K_p \times e(t) + K_i \int e(t)dt...
\] (2.13)

![Figure (2.13): PI-like fuzzy control system](image)

The diagram Figure (2.13) has a different from the previous one Figure (2.12). By replaced differentiation with integration and a change-of-error with an integral error.
Now the fuzzy controller and the rules table have other inputs. It means that the rules themselves should be reformulated. Sometimes it is difficult to formulate rules depending on an integral error, because it may have the very wide universe of discourse. In this case the integration can move from the part proceeding to a fuzzy controller to the part following it. Output of a controller will integrate, not the input. Then the error and the change of error inputs and still realize the PI-control. When the derivative, with respect to time, of the Equation (2.8) is taken, it is transformed into an equivalent expression

\[ \frac{du(t)}{dt} = K_p \times \frac{de(t)}{dt} + K_i e(t) \]  

or in the discrete form

\[ \Delta u(t) = K_p \times \Delta e(t) + K_i e(t) \]  

One can see here that one has the error and the change-of-error inputs and one need just to integrate the output of a controller. One may consider the controller output not as a control signal, but as a change in the control signal. The block diagram for this system is given in Figure (2.14). The rule format proposed for this controller written as:

if e is <property symbol> and \( \Delta e \) is <property symbol> then \( \Delta u \) is <property symbol>.

In this case, to obtain the value of the control output variable \( u(t) \), the change-of-control output \( \Delta u(t) \) is added to \( u(t-1) \). It is necessary to stress here that this takes place outside the PI-like fuzzy controller, and is not reflected in the rules themselves.

Figure (2.14): PI-like fuzzy control system
PID-like fuzzy controller, the equation for a PID-controller is as follows:

\[ u(t) = K_p \times e(t) + K_d \Delta e(t) + K_i \int e(t) dt \] (2.16)

Thus, in the discrete case of a PID-like fuzzy controller one has an additional process state variable, namely sum-of-errors, denoted \( \sigma e \) and computed as:

\[ \sigma e = \sum_{i=1}^{n} e(i) \] (2.17)

The symbolic expression for a rule of a PID-like fuzzy controller is:

if \( e \) is < property symbol > and \( \Delta e \) is < property symbol > and \( \sigma e \) is < property symbol> then \( u \) is < property symbol >.

The main difference between the rules for these controllers, the last one has three conditions in the antecedent part but the previous ones had just two [13]. Formulating many more rules to describe the PID-controller will be required. If any input is described with seven linguistic values, as was before, then because the PID-controller has three inputs and any rule has three conditions we will need \( 7 \times 7 \times 7 = 343 \) rules.

Previously we had just \( 7 \times 7 = 49 \) rules. It is too much work to write 343 rules.

2.4.4 PID fuzzy controller

The PID-like fuzzy controller can be constructed as a parallel structure of a PD-like fuzzy controller and a PI-like fuzzy controller Figure (2.15) with the output approximated as [10]:

\[ u = \left( \frac{K_p}{2} \times e + K_d \times \frac{de}{dt} \right) + \left( \frac{K_p}{2} \times e + K_i \times \int edt \right) \] (2.18)
2.5 Artificial Neural Network

It is well known that biological systems can perform complex tasks without recourse to explicit quantitative operations. In particular, biological organisms are capable of learning gradually over time. This learning capability reflects the ability of biological neurons to learn through exposure to external stimuli and to generalize. Such properties of nervous systems make them attractive as computation models that can be designed to process complex data. For example, the learning capability of biological organisms from examples suggests possibilities for machine learning [6].

2.5.1 Single-input

A single-input neuron shown in figure (2.16) is the one type of different neuron model the scalar input \( b \) is multiplied by the scalar weight \( w \) to form \( wp \), one of the terms that is sent to the summer. The other input \( 1 \), is multiplied by a bias \( b \) and then passed to the summer. The summer output \( n \), often referred to as the net input, goes into a transfer function \( f \), which produces the scalar neuron output \( a \). (Some authors use the term “activation function “rather than transfer function and “offset” rather than bias.) The neuron output is calculated as:

\[
a = f(wp + b)……………………………………………………………………………….. (2.19)
\]

Note that \( w \) and \( b \) are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the designer and then the parameters \( w \) and \( b \) will be
adjusted by some learning rule so that the neuron input/output relationship meets some specific goal.

The transfer function may be a linear or a nonlinear function. A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve [7].

![Diagram of a single-input neuron](image)

Figure (2.16): Single - input neuron

### 2.5.2 Neural control

Neural control refers both to a methodology in which the controller itself is a neural network, and to a methodology in which controllers are designed based on a neural network model of the plant. These two basically different approaches for implementing neural networks in control are referred to as direct and indirect design methods.

We have seen that fuzzy control is a control method relying on perception-based information expressed in fuzzy logic. This is the case where the available data is in the form of a collection of linguistic If... then... rules. In other words, fuzzy control is a mathematical method for implementing control strategies expressed in a natural language. This situation arises mostly in the control of complex systems, a situation that human operators handle well and for which natural language is an appropriate means for describing control strategies.
As its name indicates, neural control refers to another control method when available data are in the form of measurements (observed numerical data) of the plant’s behavior. This is the case where information is only in the form of system behavior, either of the real plant or of its simulated model, expressed as input-output measurements.

In view of the generality of neural networks as function approximation devices, it is natural to use neural networks in control situations such as this. Specifically, when mathematical models of the plant dynamics are not available, neural networks can provide a useful method for designing controllers, provided we have numerical information about the system behavior in the form of input-output data. In other words, a neural network can be used as a “black box” model for a plant. Also, controllers based on neural networks will benefit from neural networks’ learning capability that is suitable for adaptive control where controllers need to adapt to changing environment, such as for time-variant systems. In practice, neural network controllers have proved to be most useful for time-invariant systems.

Basically, to build a neural network-based controller that can force a plant to behave in some desirable way; we need to adjust its parameters from the observed errors that are the difference between the plant’s outputs and the desired outputs. Adjustment of the controller’s parameters will be done by propagating back these errors across the neural network structure. This is possible if the mathematical model of the plant is known.

When the mathematical model of the plant is not known, we need to know at least an approximate model of the plant in order to do the above. An approximate (known) model of the plant is called an identified model. When we use input-output data from the plant to train a neural network to provide an approximate model to the plant, we obtain a neural network identified model of the plant. Neural network identified models are used in indirect neural control designs. After a general
discussion of inverse dynamics, we will first discuss direct neural control designs and then indirect control.

2.5.3 Neural networks in direct neural control

Direct design means that a neural network directly implements the controller that is, the controller is a neural network as in Figure (2.17). The network must be trained as the controller according to some criteria, using either numerical input-output data or a mathematical model of the system.

A natural question that arises in this type of neural control is the selection of the type of neural network needed for the controller. There are several types of neural network architectures. Multi-layered perceptron (MLP) neural networks are composed of configurations of simple perceptrons in a hierarchical structure forming a feed forward network. They have one or more hidden layers of perceptrons between the input and output layers. It is permissible to have any prior layer nodes connected to subsequent layer nodes via a corresponding set of weights.

Different learning algorithms can be used for MLPs, but the most common ones have been the delta rule and error-back propagation algorithms. These algorithms do work fairly well but they tend to be slow. Faster and more efficient algorithms have been developed [8, 20, 32, 37], and ongoing research is continually discovering further improvements.

2.5.4 Neural networks in indirect neural control

Indirect neural control design is based on a neural network model of the system to be controlled. In this case, the controller itself may not be a neural network, but it is
Derived from a plant that is modeled by a neural network.

This is similar to standard control in that a mathematical model is needed, but here the mathematical model is a neural network.

Indirect neural control designs involve two phases. The first phase consists of identifying the plant dynamics by a neural network from training data that is, system identification. In the second phase, the control design can be rather conventional even though the controller is derived, not from a standard mathematical model of a plant, but from its neural network identified model.

Since the identified neural network model of the plant is nonlinear, one way to design the controller is to linearize its identified neural network model and apply standard linear controller designs. Another way is through instantaneous linearization [7].

2.6 Neuro-Fuzzy Systems

Both neural networks and fuzzy system are motivated by imitating human reasoning process. It utilizes human expertise. In fuzzy systems, relationships are represented explicitly in the form of the if-then rules. In neural networks, the relations are not explicitly given, but are encoded in the networks and parameters designed. Neuro-fuzzy systems combine semantic transparency of rule-based fuzzy systems with a learning capability of neural networks [8].

2.6.1 Adaptive network fuzzy inference systems

To illustrate the use of neural networks for fuzzy inference, we present some successful adaptive neural network fuzzy inference systems, along with training algorithms known as ANFIS. These structures, also known as adaptive neuro-fuzzy inference systems or adaptive network fuzzy inference systems, were proposed by Jang. It should be noted that similar structures were also proposed independently by
Lin and Lee and Wang and Mendel. These structures are useful for control and for many other applications [7].

2.6.2 Neuro – fuzzy controller

The neural predictive controller can be extended with Neuro-fuzzy controller, connected in parallel Figure (2.18). Neuro-fuzzy systems, which combine neural networks and fuzzy logic, have recently gained a lot of interest in research and application. A specific approach in neuro-fuzzy development is the ANFIS (Adaptive Network-based Fuzzy Inference System).

ANFIS uses a feed forward network to search for fuzzy decision rules that perform well on a given task. Using a given input output data set; ANFIS creates a Fuzzy Inference System for which membership function parameters are adjusted using a combination of a back propagation and least square method [9].

Figure (2.18) Neuro – Fuzzy Control scheme

2.6.3 Adaptive network fuzzy inference systems as an estimator

Adaptive Network Fuzzy Inference System (ANFIS) can be used for the estimation of some dependent variables in chemical process. The designed ANFIS estimator is used to infer the compositions from measurable tray temperatures distillation
column. In estimator design process, different ANFIS structure are constructed and trained to find the architecture that gives the best performance as an estimator.

As a first step to design an estimator, training data sets should be generated to train the estimator networks. These data sets consist of estimator inputs and desired output values. They are produced from the process input output data. Since, ANFIS is a data processing method, it is important that the input-output data must be within the sufficient operational range including the maximum and minimum values for both input and output variables of the system. If this is not provided, estimator performance cannot be guaranteed and thus the designed estimator will not be accurate. Having generated the training data, estimators that have different architectures are trained with the obtained data sets.

Performances of the trained estimators are evaluated with model simulations and best estimator architecture is obtained. These simulations are made to verify and to generalize the ANFIS structures. Verification is done to show how good the estimator structure learned the given training data. This is carried out by simulating the column models with specific initial process inputs used in obtaining training data sets. Generalization capabilities of the estimators are found with other simulations in which input process variables are in operational range but not used in training data formation.

ANFIS estimator design consists of two parts: constructing and training. In constructing part, structure parameters are determined. These are type and number of input Membership Functions (MFs), and type of output MF. Any of several MFs such as Triangular, Trapezoidal and Gaussian can be used as an input MF. Frequently used MFs in literature are the Triangular and Gaussian. For this reason, they are chosen as input MF type in this study. Number of MFs on each input can be chosen as 3, 5, and 7 to define the linguistic labels significantly. Effective partition of the input space is important and it can decrease the rule number and thus increase
the speed in both learning and application phase. Output MFs can be either a constant or in linear form.

Both of these two forms are used for the output MF in this study. Having described the number and type of input MFs, the estimator rule base is constituted. Since, there is no standard method to utilize the expert knowledge; automatic rule generation method is usually preferred. According to this method, for instance, an ANFIS model with two inputs and five MFs on each input would result in \(3^2 = 25\) Takagi-Sugeno fuzzy if-then rules automatically. Although this method can require much computational knowledge especially in systems that have to be defined with many inputs, it is used in this study due to advantage of MATLAB software. Therefore, rule bases of the estimators are formed automatically with the number of inputs and number of MFs. After the ANFIS structure is constructed, learning algorithm and training parameters are chosen. As mentioned in the earlier in this chapter, back propagation or hybrid learning can be used as a learning algorithm. The hybrid learning algorithm is used in this study.

Parameters in the algorithm are epoch size (presentation of the entire data set), error tolerance, initial step size, step size decrease rate, and step size increase rate. Since there is no exact method in literature to find the optimum of these parameters a trial and error procedure is used. MATLAB fuzzy logic toolbox is used to design ANFIS estimators’ structures. Using the given training data set, the toolbox constructs an ANFIS structure using either a back propagation algorithm alone, or in combination with least squares type of method (hybrid algorithm). ANFIS model can be generated either from the command line, or through the ANFIS editor GUI. In this study, ANFIS Editor GUI is used to generate the ANFIS models with the chosen design parameters in construction phase. Written MATLAB code is used to train the ANFIS structure in the training step.

The steps in ANFIS estimator design in this study utilizing the MATLAB fuzzy logic toolbox are as follows:
• Generated training data is loaded to the Editor GUI.
• Design parameters, number of input MF, type of input and output MF, are chosen. Thus, initial ANFIS structure is formed.
• The code for the training is run with the initial structure.
• ANFIS structure constituted after training is saved to use as an estimator [9].
CHAPTER THREE

CONTROL SYSTEM DESIGN

3.1 Introduction

Abstract- Power stabilizers are used to produce controlling signals for simulation system and to decrease the oscillation of low frequency power system. A variety of methods are proposed to overcome the faults of common power system stabilizer of which PID, fuzzy logic, genetic algorithm and neural networks. PID, Fuzzy logic and Neuro-fuzzy controller are used to control the system based on simulations, it is defined that the Neuro-fuzzy controller make better answers to common stabilizer.

3.2 Power system stabilizer Parameters

Generator&load = \( \frac{1}{0.8s+1} \)………………………………………………………………………..(3.1)

Governor = \( \frac{1}{0.2s+1} \)………………………………………………………………………..(3.2)

Turbine = \( \frac{1}{0.5s+1} \)………………………………………………………………………..(3.3)

3.3 Design of fuzzy adaptive controller

Fuzzy adaptive controller system design essentially amounts to choosing the fuzzy controller inputs and outputs, choosing the preprocessing that is needed for the controller inputs and possibly post processing that are needed for the outputs, and to designing each of the four components of the fuzzy controller shown in Figure (3.1).

Figure (3.1): Fuzzy controller architecture.
There are standard choices for the fuzzification and defuzzification interfaces. Moreover, most often the designer settles on an inference mechanism and may use this for many different processes. Hence, the main part of the fuzzy controller that we focus on for design is the rule-base.

The rule-base is constructed so that it represents a human expert “in-the-loop.” Hence, the information that we load into the rules in the rule-base may come from an actual human expert who has spent long time learning how best to control the process. In other situations there is no such human expert, and the control engineer will simply study the plant dynamics (perhaps using modeling and simulation) and write down a set of control rules that makes sense.

As an example, in the cruise control problem discussed above it is clear that anyone who has experience driving a car can practice regulating the speed about a desired set-point and load this information into a rule-base. For instance, one rule that a human driver may use is “If the speed is lower than the set-point, then press down further on the accelerator pedal.” A rule that would represent even more detailed information about how to regulate the speed would be “If the speed is lower than the set-point AND the speed is approaching the set-point very fast, then release the accelerator pedal by a small amount.” This second rule characterizes our knowledge about how to make sure that we do not overshoot our desired goal (the set-point speed). Generally speaking, if we load very detailed expertise into the rule-base, we enhance our chances of obtaining better performance [10].

By editing "fuzzy" in workspace a window of (FIS) editor appears as in Figure (3.2).
3.3.1 Fuzzy control

Roughly speaking, fuzzy control is ‘control with rules’. A fuzzy controller can include Empirical rules and that is especially useful in operator-controlled plants. Consider the typical fuzzy controller:

i. If error is Neg and change in error is Neg then control is NB
ii. If error is Neg and change in error is Zero then control is NM

The rules are in the familiar if–then format, with the premise on the if-side and the conclusion on the then-side. The premise value ‘Neg’ is a linguistic term short for the word ‘negative’, the conclusion value ‘NB’ stands for ‘negative big’ and ‘NM’ for ‘negative medium’. The collection of rules is a rule base. A computer can execute the rules and compute a control action depending on the measured inputs error and change in error. The inclusion of fuzzy rules in a controller raises more design questions than usual. The objective here is to identify and explain those design choices. In a rule-based controller the control strategy is in a more or less
natural language. A rule-based controller is intelligible and maintainable for a non-specialist. An equivalent controller could be implemented using conventional techniques – it is just more convenient to isolate the control strategy in a rule base when operators control the plant. In the direct control scheme in Figure 3.1 the fuzzy controller is in the forward path of a feedback control system. The plant output $y$ is compared with a reference $\text{Ref}$, and if there is a deviation $e = \text{Ref} - y$, the controller takes action according to the control strategy embedded in the rule base. In the figure, the arrows can be hyper-arrows containing several signals at a time for multi-loop control.

There are at least four main sources for finding control rules (Takagi and Sugeno in Lee 1990):

### 3.3.2 Fuzzy controller design

Fuzzy control is a practical alternative for a variety of challenging control applications since it provides a convenient method for constructing nonlinear controllers via the use of heuristic information. Such heuristic information may come from an operator who has acted as a “human-in-the-loop” controller for a process. In the fuzzy control design methodology, we ask this operator to write down a set of rules on how to emulate the decision-making process of the human. In other cases, the heuristic information may come from a control engineer who has performed extensive mathematical modeling, analysis, and development of control algorithms for a particular process. Again, such expertise is loaded into the fuzzy controller to automate the reasoning processes and actions of the expert. Regardless of where the heuristic control knowledge comes from, fuzzy control provides a user-friendly formalism for representing and implementing the ideas we have about how to achieve high-performance control [11]. The fuzzy controller takes input values from the real world. These values are referred to as “crisp” values since they are represented as single number, not a fuzzy one. There are three parts to a fuzzy controller, the fuzzification of the inputs, the defuzzification of the outputs, and the rule base. Figure (3.3) shows these plants in order for the fuzzy controller to understand the inputs; the crisp input has to be converted to a fuzzy number.
3.3.3 Structure of fuzzy logic

There are specific components characteristic of a fuzzy controller to support a design procedure. Figure (3.4) shows the controller between the pre-processing block and post processing block. [12]

3.3.3.1 Pre-processing

The inputs are most often hard or crisp measurement from some measuring equipment rather than linguistic. A pre-processor, the first block in Figure (3.4) shows the conditions the measurements before enter the controller [13].

3.3.3.2 Fuzzification

The first block inside the controller is fuzzification, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block matches the input data with the conditions of the
rules to determine. There is degree of membership for each linguistic term that applies to the input variable [13].

3.3.3.3 Rule base

The collection of rules is called a rule base. The rules are in “If Then” format and formally the If side is called the conditions and the Then side is called the conclusion. The computer is able to execute the rules and compute a control signal depending on the measured inputs error (e) and change in error.(dE). In a rule based controller the control strategy is stored in a more or less natural language. A rule base controller is easy to understand and easy to maintain for a non-specialist end user and an equivalent controller could be implemented using conventional techniques [13].

3.3.3.4 Defuzzification

Defuzzification is when all the actions that have been activated are combined and converted into a single non-fuzzy output signal which is the control signal of the system. The output levels are depending on the rules that the systems have and the positions depending on the non-linearities existing to the systems. To achieve the result, develop the control curve of the system representing the I/O relation of the systems and based on the information; define the output degree of the membership function with the aim to minimize the effect of the non-linearity [14].

3.3.3.5 Postprocessing

The postprocessing block often contains an output gain that can be tuned and also become as an integrator [14].

3.3.4 Fuzzy logic toolbox

There are five primary graphical user interface (GUI) tools for building, editing and observing fuzzy inference systems in the toolboxes shown in Figure(3.5):

- Fuzzy Inference System (FIS) editor
- Membership Function editor
- Rule Editor
- Rule Viewer
- Surface Viewer

These GUI are dynamically linked and if the changes make to the FIS to the one of
the toolbox, the effect can be seen in other GUIs. In addition to these five primary
GUIs, the toolbox includes the graphical ANFIS Editor GUI, which is used for
building and analyzing Sugeno-types adaptive neural fuzzy inference systems [13].

![Fuzzy inference system](image)

**Figure (3.5): Fuzzy inference system**

### 3.3.4.1 FIS editor

The FIS editor handles the high level issues of the system. Fuzzy Logic toolbox
does not limit The FIS editor displays general information about fuzzy inference
systems. There is a simple diagram at the top shows the name of each input variable
on the left and the output on the right. The step below is show how fuzzy controller
is build [12].

i. System are start from scratch, by typing fuzzy at the MATLAB prompt.

ii. By Selected  Edit > Add Variable > Input two input are chosen (because the
system need two inputs)

iii. By click the yellow box input1. Box is highlighted with a red outline

iv. The Name field from input1 is Edit to "e", and press Enter.
v. By Clicked the yellow box input2. This box is highlighted with a red outline/vi. The Name field from input2 is Edi to "ce", and press Enter.
vii. By Clicked the blue box output1.
viii. The Name field from output1 is Edi to "ci", and press Enter.
Ix. By Selected File > Export > To workspace
x. By Enter the Workspace variable name, and click OK.

3.3.4.2 Membership function editor

The membership function editor shown in Figure (3.6) shares some features with the FIS editor. The membership function editor is the tool that lets the programmer displays and edits all of the membership functions associated with all inputs and output variables for entire fuzzy inference system [12].

When you open the Membership Function Editor to work on a fuzzy inference system that does not already exist in the workspace, there is not yet any membership functions associated with the variables that you have just defined with the FIS Editor.

The step below shows how to open the membership function editor which are shown in Figure (3.7):

• Within the FIS editor windows, select Edit > Membership functions
• Within the FIS editor, double click the blue icon called ci
• At the command line, type mfedit

![Add membership functions](image)

Figure (3.6): Membership Function Editor
Figure (3.7): FIS Editor for Membership Function variable

The process of specifying the input membership function for two inputs is as follows:

- Selected input variable, 'e', double clicking on it. Set both the Range and the Display Range to the vector [0 10]
- Selected Remove All MFs from Edit menu. It removes all the existing Membership Function from the Membership Function Editor.
- Selected Add MFs from Edit menu. The window shown in Figure (3.6) is opens:
- Tab is used to choose “trimf” for MF Type and 7 Number of MFs. This choice adds seven Trimf curves to the input variable service.
- Click once on the curve with the left-most hump. Change the name of the curve to "nb". To adjust the shape of the membership function, click on the membership function. The desired parameter Params listing for this will appear. The two inputs of Params represent the standard deviation and centre for the Trimf curve.
- After editing all the value and adjusted the membership function, the system will look similar Figure(3.8) the step are repeated for "ce" and "ci" as are shown in Figure(3.9) and (3.10) respectively:-

43
Figure (3.8) Membership function editor for input 1"e"

Figure (3.9) Membership function editor for input 2"ce"

Figure (3.10) Membership function editor for output "ci"
3.3.4.3 Rule editor

Rule editor are call up, by choose edit menu rules edit selected.

![Rule Editor](image)

Based on the description of the input and output variable defined with the FIS Editor, the Rule Editor allows to construct the rule statements automatically. From GUI as shown in Figure (3.11):

- Rules are create by selecting an item in each input and output variable box and one Connection item and clicking Add Rule. You can choose none as one of the variable qualities to exclude that variable from a given rule and choose not under any variable name to negate the associated quality.
- Rules are deleting by selecting the rule and clicking Delete Rule.
- Rules are editing by changing the selection in the variable box and clicking Change Rule.
- Specify weight to a rule by typing in a desired number between 0 and 1 in Weight. If you do not specify the weight, it is assumed to be unity (1).the rule which used are determine in Table (3.1).
Table (3.1): A fuzzy rule base table.

<table>
<thead>
<tr>
<th>Change in error</th>
<th>NL</th>
<th>NM</th>
<th>NS</th>
<th>NO</th>
<th>PO</th>
<th>PS</th>
<th>PM</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NS</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NM</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>NO</td>
<td>NO</td>
<td>PO</td>
</tr>
<tr>
<td>NS</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>NO</td>
<td>NO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
</tr>
<tr>
<td>NO</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>NO</td>
<td>NO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
</tr>
<tr>
<td>PO</td>
<td>NM</td>
<td>NS</td>
<td>NO</td>
<td>PO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
</tr>
<tr>
<td>PS</td>
<td>NS</td>
<td>NO</td>
<td>PO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PM</td>
<td>NO</td>
<td>PO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PL</td>
<td>PO</td>
<td>PO</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
</tbody>
</table>

3.3.4.4 Rule viewer

The Rule Viewer shown in Figure (3.12) displays a roadmap of the whole fuzzy inference process. It is based on the fuzzy inference. The three plots across the top of the Figure (3.7) represent the antecedent and consequent of the first rule. Each rule is a row of plots, and each column is a variable. The rule numbers are displayed on the left of each row. To view the rule in the status line click the rule number [12].

Figure (3.12): Rule viewer
3.3.4.5 Surface viewer

Upon opening the Surface Viewer shown in Figure (3.13), we are presented with a two-dimensional curve that represents the mapping from service quality to tip amount. Since this is a one-input one-output case, we can see the entire mapping in one plot. Two-input one-output systems also work well, as they generate three-dimensional plots that MATLAB can adaptively manage. When we move beyond three dimensions overall, we start to encounter trouble displaying the results. Accordingly, the Surface Viewer is equipped with pop-up menus that let you select any two inputs and any one output for plotting. Just below the pop-up menus are two text input fields that let you determine how many x-axis and y-axis grid lines you want to include. This allows you to keep the calculation time reasonable for complex problems.

![Surface Viewer](image)

Figure (3.13): Surface viewer

3.4 Linear Fuzzy PID Control

A fuzzy PID controller is a fuzzified proportional-integral-derivative (PID) controller. It acts on the same input signals, but the control strategy is formulated as
fuzzy rules. If a control engineer changes the rules, or the tuning gains to be discussed later, it is difficult to predict the effect on rise time, overshoot, and settling time of a closed-loop step response, because the controller is generally nonlinear and its structure is complex. In contrast, a PID controller is a simple, linear combination of three signals: the p action proportional to the error e, the I-action proportional to the integral of the error e dt, and the D-action proportional to the time derivative of the error de/dt, or e for short. This chapter introduces a systematic tuning procedure for fuzzy PID type controllers.

3.4.1 Fuzzy proportional controller

Proportional control is the simplest form of continuous control that can be used in closed-loop system. Proportional action can reduce the steady-state error and provides a faster response, but too much of it can cause the stability to deteriorate [13].

The input to fuzzy proportional FP controller is the error “e” and the output signal U as shown in figure (3.14)

\[ E = GE \times e \]  
(3.4)

The gains are mainly, for tuning the response, but since there are two gains, the input universe to exploit it better. The controller output is the control signal U (n) which is a non-liner function of e (n) [5].
U(n) = f(GE * e(n)) .................................................................(3.5)

### 3.4.2 Fuzzy proportional- derivative controller

Proportional- derivative controller use the derivative action to improve closed-loop stability and help to predict the error (reduce overshoot, but it may be sensitive to noise as well as an abrupt change of the reference causing a derivative kick). The input to fuzzy PD controller is the error “e” and derivative of the error “ce” as shown in figure (3.15)

![Fuzzy PD Controller (FPD)](image)

The controller output is a non-liner function of error and change in error.

U(n)= f(GE *e(n), GCE *ce(n))*GU................................................................. (3.6)

### 3.4.3 Fuzzy proportional, derivative and integral controller

When the control problem is to regulate the process output around a set point, it is natural to consider error as an input, even to a fuzzy controller, and it follows that the integral of the error and the derivative of the error may be useful input as well. In a fuzzified PID controller, however, it is difficult to tell the effect of each gain factor on the rise time, overshoot, and settling time since it is most often nonlinear and has more tuning gains than a PID controller.

However it has been known that conventional PID controllers generally do not work well for nonlinear systems, high order and time-delayed liner system, and particularly complex and vague systems that have no precise mathematical models.
To overcome this difficulties, various type of modified conventional PID controller such as automatic and adaptive PID controller were developed lately[15]. Integral action will eliminate the stead-state error. I-controller are much slower in their response time than p-controller. It often used when measured variables need to remain within every narrow range and require fine-tuning control [13].

The input to fuzzy PD+I controller is the error “e” the derivative of error “ce” and integral of the error “i.e.” a rules base with three input becomes very big, and rules concerning the integral action are troublesome. There for it is common to separate the integral action as in the fuzzy PD+I controller as shown in figure (3.16).

![Fuzzy PD+I controller (FPD+I).](image)

The controller output is function of three inputs

\[ u(n) = u_1(n) + u_2(n) \]  
\[ u(n) = f_1(GE \cdot e(n), GCE \cdot ce(n)) + f_2(GIE \cdot ie(n)) \]  
\[ U(n) = f_1(GE \cdot e(n), GCE \cdot ce(n)) \cdot GU + f_2(GIE \cdot ie(n)) \cdot GU \]

This controller provides all the benefits of PID Control, but also the disadvantages regarding derivative kick and integrator windup [13].

**3.5 Neuro-Fuzzy Controller Design**

ANFIS uses a hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS). A combination of least-squares and back propagation gradient descent methods are
used for training FIS membership function parameters to model a given set of input/output data.

[FIS,ERROR] = ANFIS (TRNDATA) tunes the FIS parameters using the input/output training data stored in TRNDATA. For an FIS with N inputs, TRNDATA is a matrix with N+1 columns where the first N columns contain data for each FIS input and the last column contains the output data. ERROR is the array of root mean square training errors (difference between the FIS output and the training data output) at each epoch. ANFIS uses GENFIS1 to create a default FIS that is used as the starting point for ANFIS training [21].

Note: in this research training data was taken from PID controllers with computational optimization approach method.

3.5.1 ANFIS editor

Using anfisedit, you bring up the ANFIS editor shown in Figure (3.17) GUI from which you can load a data set and train anfis. The ANFIS Editor GUI invoked using anfisedit('a'), opens the ANFIS editor GUI from which you can implement ANFIS using a FIS structure stored as a file a.FIS anfisedit(a) operates the same way for a FIS structure a, stored as a variable in the MATLAB workspace, Figure (3.18) showing ANFIS model structure.

On the ANFIS editor GUI, there is a menu bar that allows you to open related GUI tools, open and save systems, and so on. The file menu is the same as the one found on the FIS editor

- By using the following edit menu item:
  - Undo to undo the most recent change.
  - FIS properties to invoke the FIS editor.
  - Membership functions to invoke the membership function editor.
  - Rules to invoke the rule editor.

- By using the following view menu items:
  - Rules to invoke the rule viewer.
Surface to invoke the surface viewer [21].

Figure (3.17): ANFIS editor.

Figure (3.18): ANFIS model structure
CHAPTER FOUR
SIMULATION RESULTS AND DISCUSSIONS

4.1 Simulation Results of PID Controller

This section demonstrates the simulation results of a power system stabilizer by using design of controllers with computational optimization approach method, by substituting the value of parameters: \( K_P = 5, K_D = 0.48 \) and \( K_I = 5 \)

Figure (4.1) and Figure (4.2) show the model and response of PID controller for power system stabilizer respectively for unit step input. It is clear that the output of the model have large \( T_r, T_s \) and steady state error at the case of model without controller. And Figure (4.3) illustrate the Control signal of PID controller.

Figure (4.1): Model of PID controller for power system stabilizer.
Figure (4.2): Unit step response of PID controller for power system stabilizer

Figure (4.3): Control signal of PID controller
4.2 Simulation Results of Fuzzy logic controller

The system with a unit step input is simulated with MATLAB as shown in figure (4.4). By using trial & error method of tuning to calculate the FLC parameters. The values of FLC parameters are, $K_p = 100$ and $K_D = 50$. Figure (4.5) and (4.6) illustrates the response of system and the control signal of Fuzzy controller.

![Simulink diagram of PD-like Fuzzy for power system stabilizer](image)

Figure (4.4): Simulink diagram of PD - like Fuzzy for power system stabilizer.

![Unit step response of FLC for power system stabilizer](image)

Figure (4.5): Unit step response of FLC for power system stabilizer.
4.3 Simulation Results of Neuro-Fuzzy

Figure (4.7) and figure (4.8) illustrate the block diagram and the unit step response of Neuro-Fuzzy controller for power system stabilizer and figure (4.9) illustrate Control signal of Neuro-Fuzzy controller.

Figure (4.7): Block diagram of Neuro-fuzzy controller.
Figure (4.8): Unit step response of Neuro-fuzzy controller for system

Figure (4.9): Control signal of Neuro-fuzzy controller.
The unit step system response for all controllers PID, FLC and Neuro-fuzzy shown in Figure (4.10).

Figure (4.10): Unit step system response for all controllers.

4.4 Comparison and Discussion

In order to validate the control strategies as described above, digital simulation were carried out on a power system stabilizer whose parameters are given in previous chapter. The MATLAB/SIMULINK model of system under study with all three controllers is shown in Figures (4.1), (4.3) and (4.7).

Comparison has been made between the maximums overshoot and the settling time illustrated in table (4.1).
Table (4.1) Comparison Between PIDPSS, FPSS and NFPSS

<table>
<thead>
<tr>
<th>Model</th>
<th>Ts (second)</th>
<th>Overshoot MP</th>
<th>Steady State Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIDPSS</td>
<td>7.8</td>
<td>0.013%</td>
<td>0</td>
</tr>
<tr>
<td>FPSS</td>
<td>10</td>
<td>0.016%</td>
<td>0</td>
</tr>
<tr>
<td>NFPSS</td>
<td>8.8</td>
<td>0.01%</td>
<td>0</td>
</tr>
</tbody>
</table>

Performance is evaluated on the basis of maximum Overshoot, Settling time and Steady State Error. The parameters obtained show that NFPSS achieves a better performance than PIDPSS and FPSS.
CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this study, the power system stabilizer has been reviewed from control theory perspective. The study is based on the MATLAB Simulink software platform and mathematical models of the PSS. The work basically exhibits versatility of the high performance for Neuro-fuzzy controller. Hence, the implementation of the Neuro-fuzzy controller gives a very cost effective solution to the drive control design. The work can also be effectively applied to higher order systems without any complications.

The neuro-fuzzy controller enhances the performance of drive system. Simulation results show that the error tends to be zero. Successful simulations demonstrate that neuro-fuzzy controller can achieve more desire performance than PID controller and fuzzy controller.

5.2 Recommendations

- MATLAB simulation for power system stabilizer has been done which will be implemented in real time to observe actual feasibility of the approach applied in this thesis.
- This technique can be applied for multiple numbers of power system stabilizers.
- The system parameters can be tuned by using genetic algorithm (GA).
References


