4.1 Introduction

Conventionally, PID algorithms control the servomotors. Such an algorithm will be effective enough if the speed and accuracy requirements of the control system are not critical. The major approach to optimize the control action is to tune the PID coefficients, which can hardly cope with the control environment due to system non-linearity.

The Model Reference Adaptive Control (MRAC) technique [103] is an approach for coping up with environmental variation and system nonlinearities. Its function is to compare the output from the process with that from a process reference model. The error then is used for adjusting the parameters of the controller for the process through suitable adaptation algorithm either based on physical/chemical laws or parameter estimation method; these are usually very complex and require large amount of the computation time. This restricts its application when fast response is desirable. The common problem to the said approach lies in the attempt to formulate the input-output relationship by means of the mathematical models, which may be very difficult to realize in most cases. Even when such model is developed, it will be too complex to compute in real-time.

Faced with these problems, many investigators realized that implementing human intelligence into automatic control systems could be a more efficient solution, which led to the development of the fuzzy control algorithm [68]. The fuzzy algorithm is based on intuition and experience, and can be regarded as a set of heuristic decision rules or "rules of thumb". Such non-mathematical control algorithms can easily be implemented in a computer and they are straightforward and should not involve any computational problems.

Mamdani and S. Assilian [70,104] reported the application of the fuzzy set theory to control a small laboratory steam engine. The purpose was to regulate the engine speed and boiler steam pressure by means of the heat applied to the boiler and the throttle setting on the engine. At the same time, Kickert and Van Nauta Lemke
examined the performance of the fuzzy controller on a warm water plant. The success of these studies led King and Mamdani [107] to attempt to control the temperature of a chemical reactor using fuzzy algorithms. Many applications of the fuzzy control to process control and motor drives published in the literature have already been presented in chapter 1 and chapter 2.

The results of the above mentioned experiments showed that the fuzzy controllers were better than or at least, as good as the performance of a PID controller. They have the common features of not requiring a detailed mathematical model. However those experiments are mainly concerned with slow physical or chemical processes. So far fuzzy logic controllers (FLC’s) in motor drives have been designed by trial and error. The motor drives are inherently nonlinear. The causes of nonlinearities in the motor drives include power converter (switching period, saturation inductance, voltage clamping, etc.), gearbox (backlash and friction), motor internal friction, etc. With advent of PWM converters and resonant converters power converters are getting complicated, resulting in complex mathematical models. The FLC seems to be a viable controller for PMDC servomotor drive.

This chapter explores the possibility of applying fuzzy algorithms in faster and more accurate controllers such as a PMDC servomotor controller. The basic design procedures of FLC is presented along with types of fuzzy knowledge based controller (FKBC). Design of fuzzy-PI controller for PMDC motor based on the new methodology introduced by Han-Xiong Li [108] for designing and tuning the scaling gains of the conventional fuzzy logic controller is presented. Subsequently, a fuzzy logic controller is developed to improve the control of PMDC motor drive for which relevant heuristic knowledge of system dynamics is known. Observing the system response characteristics over a specified operating range the FLC rules are generated. The controller is designed after studying the various responses obtained from the classical PI controller and fuzzy-PI controller.

This chapter also presents the implementation of a fuzzy controller for PMDC motor drive using PC/AT and a high performance data acquisition and control card (DAS). Then the implementation aspect of fuzzy logic controller using an inexpensive 16-bit microcontroller (Intel’s 80C196KB) is also presented. A high performance (PC-Based) Data Acquisition and Control Card is designed and developed for above [109,110]. To emphasize the merits and demerits of the FLC, some comparisons have also been made with the fuzzy-PI controller and linear PI controller under load and supply disturbances.
4.2 Basic Design Procedures of FLC

Complete and thorough description of the design procedure for an FLC has been presented in [111, 112]. Therefore only essential concepts and notations are stated here an introduction.

4.2.1 Fuzzification

Fuzzy logic uses linguistic variables instead of numerical variables. In a closed loop control system, error between reference input (set point) and output could be labeled as zero (Z), positive small (PS), negative small (NS), etc. In real world, measured quantities are real numbers (crisp). The process of converting a numerical variable (real number) into a linguistic variable (fuzzy number) is called fuzzification. For a given crisp input, fuzzifier finds the degree of membership in every linguistic variable. The shape of the fuzzy set is quite arbitrary which depends on the preference of the user. For simplicity, triangular or trapezoidal shapes are usually used.

4.2.2 Rule base

The rule base is formed by a family of linguistic rules that describes the relationship between the input and output variables of the controller. In case of I-inputs single-output fuzzy system, for example, the jth control rule in the rule base with J rules can be expressed as

\[ R_j: \text{IF } x_1 \text{ is } A_{1j} \text{ AND } x_2 \text{ is } A_{2j} \text{ AND } \ldots \text{ AND } x_i \text{ is } A_{ij} \text{ THEN } y \text{ is } B_j \]  

(4.1)

Where \( x_i \) (\( i = 1 \sim I \)) is the process state linguistic variable defined on the universe of discourse (UoD) \( U_x \) with the term set \( T_x \). \( A_i \) (\( j = 1 \sim J \)) is the linguistic value of \( x_i \) with \( A_{ij} \in T_x \). \( Y \), \( U_y \), \( T_y \) and \( B_i \) are the counterparts of \( x_i \), \( U_x \), \( T_x \) and \( A_i \) for control action.

The Equation (4.1) describes the method of the sequence representation of the rule base. Thus the rule base stores the linguistic control rules required by rule evaluator (decision making logic).

4.2.3 Decision making logic (Rule evaluator)

The individual-rule based inference [111] is employed and modified here to cope with the inference of fuzzy sets. In this approach, a four-step operation is proposed to accomplish the inference procedure [112].
Step 1: Computing the level of matching $A_{ij}(x_i^*)$ between the crisp input value $x_i$ and linguistic value $A_i^*$

$$A_{ij}(x_i^*) = M(x_i^*, A_{ij})$$  (4.2)

$M$ maps a crisp input value to a membership degree subjected to specific membership function. Following [113] (let $x_i = \inf U_{x_i}$, $x_u = \sup U_{x_u}$), triangular function is used to construct the membership functions in this research.

Step 2: Finding the firing level $\Phi_j$ of each of the rules in the rule base.

$$\Phi_j = F(A_{ij}(x_i^*), ..., A_{ij}(x_i^*))$$  (4.3)

In the Mamdani-FLC, $F$ is the MIN aggregating operator. However, it can be replaced by other trapezoidal norms (t-norms for short) [114].

Step 3: Deciding the output fuzzy set $B_j$ of each rules.

The formulation that determines how $\Phi_j$ and the fuzzy set $B_j$ interact to form the rule output is called a fuzzy implication. Here $I$ denote this operation.

$$B_j(y) = I(\Phi_j, B_j(y))$$  (4.4)

$I$ is usually implemented as the MIN or the product through other t-norms.

Step 4: Aggregating the individual output fuzzy sets $B_j$s to form the overall system output fuzzy set $B$. Operator $A$, usually implemented as the MAX or the summation, is used here.

$$B(y) = A(B_1(y), ..., B_J(y)) = A(B_j^*(y))$$  (4.5)

4.2.4 Defuzzification

The reverse of fuzzification is called defuzzification. The rules of the FLC produce required output in a linguistic variable (fuzzy number). According to the real world requirements, linguistic variables have to be transformed to crisp output (real number). The choices available for defuzzification are numerous. So far the choice of the strategy is a compromise between accuracy and computational intensity. Center of area/gravity (COA or COG for short) method is the most popular scheme used to calculate the crisp value [111]. The defuzzification operator $D$ converts the fuzzy set $B$ into a single crisp value $y^*$, i.e.

$$Y^* = D(B(y))$$  (4.6)

$$Y_{COA}^* = \frac{\int_{U_x} B(y)ydy}{\int_{U_x} B(y)dy}$$  (4.7)
In short, the crisp controls action $y^*$ of an FLC can be obtained by a series of operations.

$$ y^* = D \left( \sum_{i=1}^{A} \left[ I \left( i \cdot \left( M \left( x_i^*, A_{ii} \right), B_i(y) \right) \right) \right] \right) \quad (4.8) $$

4.2.5 Database

The Database stores the definition of the membership function required by fuzzifier and defuzzifier. Storage format is a compromise between available memory and speed of the digital controller chip.

4.3 Types of Fuzzy Knowledge Based Controller (FKBC)

Using fuzzy control methods one can fall back on well-known control structures which, however, are varied, extended, and even mixed with conventional technique. In fuzzy control there are analogous PD type FLC (Fuzzy-PD), PI type FLC (Fuzzy-PI), and PID type FLC (Fuzzy-PID) [116,117]. Their basic structures are shown in Figures 4.1 and 4.2. The rule base used is a two-dimensional linear rule base [115,116], and shown in Table 4.1 with seven labels for each input and output variable. It has been mentioned that PI/PD/PID type FLC and sliding mode fuzzy control (SMFC) are Mamdani fuzzy knowledge based controllers (FKBC). The PID type FLC is widely used for second order system with both linear and nonlinear characteristics. The SMFC and its variants are extension of the PID like FKBC, and are used with nonlinear systems of higher order with considerable model uncertainties and disturbances. The Sugeno FKBC is used with highly nonlinear plants where control actions can be locally described by linear or nonlinear control consequence.

4.3.1 Gain Structures of FLC

Membership functions (MF’s) of input/output variables can be chosen as the standard triangle for simplicity of the calculations. The resolution of each variable mainly depends on the fuzziness of its MF’s, that can be controlled by its scaling gain [115]. As linear rule base is used, the relationship between two input scaling gains can be approximated as a constant $\alpha$ [118].

$$ K_d = \alpha K_c. \quad (4.9) $$

For designing and tuning the scaling gains of the conventional (PI/PID type) FLC, based on its well-tuned linear counterpart Han-Xiong Li [108] introduced a new methodology. In this, he introduced a new concept of “fuzzy transfer function” to connect these fuzzy
gains with corresponding scaling gains. The simulation results for a third-order linear model presented in [108] demonstrates the viability of the new concept. But the requirement of well-tuned linear counterpart (PI/PID control) could not be met in most of the cases. The controller designed may have conditional stability, which means restrictions on the gains ($K_p$ and $K_i$) of the linear PI controller; further the sensitivity of these gains for various disturbances makes the tuning of the controller tedious.

As described in [108] the fuzzy $K_p$ and $K_i$ of fuzzy PI can be expressed with the fuzzy transfer function.

As described in [108] the fuzzy $K_p$ and $K_i$ of fuzzy PI can be expressed with the fuzzy transfer function.

![Diagram](image)

Figure 4.1 Digital structure of (A) fuzzy-PD and (B) fuzzy-PI

![Diagram](image)

Figure 4.2 Structure of simplified Fuzzy PID

Table 4.1 A linear two-dimensional rule base

<table>
<thead>
<tr>
<th>$\Delta E/E$</th>
<th>NL</th>
<th>NM</th>
<th>NS</th>
<th>ZR</th>
<th>PS</th>
<th>PM</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>ZR</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PM</td>
<td>NS</td>
<td>ZR</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PS</td>
<td>NM</td>
<td>NS</td>
<td>ZR</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>ZR</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZR</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
</tr>
<tr>
<td>NS</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZR</td>
<td>PS</td>
<td>PM</td>
</tr>
<tr>
<td>NM</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZR</td>
<td>PS</td>
</tr>
<tr>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZR</td>
</tr>
</tbody>
</table>
These fuzzy gains \(K_P\) and \(K_i\) should have qualitative similarity to the gains \(K_p\) and \(K_i\) of linear PI control.

4.3.2 Comparative Gain Design

There is always an input limitation for FLC so that the inputs and outputs of the FLC are always normalized into \([-1, 1]\) interval, by using proper scaling factor. For conventional FLC in Figures 4.1 and 4.2, two inputs error \((E)\) and change of error \((\dot{E})\) are available. The maximum scaling gain for unsaturated \(E\) is \(K_e = 1\). It is not easy to get the maximum scaling gain \(K_d = N_{\text{max}}\) for unsaturated \(\dot{E}\). Where \(N_{\text{max}}\) is the maximum value of the scaling gain for the unsaturated input. Then based on definition of fuzzy transfer function,

\[
F\{K_e\} = 1 \text{ and } F\{\alpha K_e\} \leq 1
\]  

(4.11)

4.3.3 Fuzzy PI Control for PMDC Motor Drive

Even though the symbolic representation of the Fuzzy PI, PD, PID is same as conventional PID's, in the case of the fuzzy PID's, gain varies depending on operating point. Fuzzy PD like conventional PD has steady state error. Fuzzy PID needs acceleration error; measuring or estimating acceleration terms are difficult and inaccurate. Due to the above-mentioned disadvantages with PD and PID, Fuzzy PI is chosen here.

The task of the control algorithm is to rotate the shaft of the motor to a set point without overshoot. It is necessary to write a set of fuzzy control statements based on the error signal between the preset and the measured shaft position, and the change of the error so as to adjust the output of the drive unit. The controller was designed after studying the various responses obtained from the classical controllers. In place of linear controller, FLC was inserted in the control system. The block diagram of fuzzy-Pi based controller for PMDC servomotor is shown in Figure 4.3.

All the inputs to the controller (error - \(e\), change of error - \(\dot{e}\)) are normalized to \([-1, 1]\) by using appropriate scaling factors. Thus the maximum gain for unsaturated \(E\) is \(K_e = 1\) that will be used as the initial gain. For simplicity of calculations the standard triangle membership functions shown in Figure 4.4 are chosen to represent inputs and output of FLC. The linear rule base presented in Table 4.1 is used for fuzzy control.
algorithm. The output variable is $u^{PI}$, which after signal conditioning and amplification is used to control the duty cycle of the PWM signal. This PWM (output) signal ranges from 0% to 100% with a resolution of 0.4% which finally drives the MOSFET bridge to control the motor shaft position. The developed PI controller is a fourth-order system with conditional stability. That means restriction on the gains ($K_p$, $K_i/T_i$) of the developed PI controller.

Using the fuzzy transfer function concept the analogy between the initial scaling gains ($\alpha$, $K$) and gains ($K_p$, $T_i/T_j$) of linear counterpart is derived as

$$K^{PI} = \frac{K_p}{F'(\alpha K_e)} \geq K_p, \quad T_i = \frac{F'(\alpha K_e)}{F'(K_e)} \propto \alpha$$

(4.12)

A) Analogy between Scaling Gains of Fuzzy PI and Gains of Linear PI

- The output scaling gain $K$ is more analogous to the proportional gain $K_p$.
- The input gain ratio $\alpha$ is more analogous to the integral time constant $T_i$ of linear PI control.

After a further approximation, the comparative design for the initial scaling gains of fuzzy PI control is obtained as

$$Fuzzy\ PI: \quad K = \frac{K_p}{F'(\alpha K_e)} \geq K_p, \quad \alpha \approx T_i$$

(4.13)

The initial scaling gain of fuzzy-PI is chosen as

$$K = \frac{K_p}{F'(\alpha K_e)} \geq K_p = 23.5$$

$$\alpha = T_j = 0.1818$$

(4.14)

Figure 4.3 Block diagram of a fuzzy-PI compensated PMDC motor drive
Figure 4.4 Membership functions of fuzzy-PI controller
B) Tuning of Fuzzy PI Control

Based on the theory of tuning PI/PD control [119] and relationship between scaling gains and fuzzy gain equation (4.13), the influence of scaling gains on the performance could be reasoned as below.

1) The Influence of Scaling Gains to the Performance

- Increasing $K$, similar to increasing $K_p$ (linear PI proportional gain), will speed up the response and reduce the steady state error. But large value of $K$ will cause the oscillation or instability.
- Decreasing $\alpha$ in fuzzy PI, similar to increasing $K_i$ (linear PI integral gain) will speed up the response and reduce the steady state error. Too small values of $\alpha$ will increase the overshoot and tend to destabilize the system.

2) Comparative Tuning Method for Fuzzy PI Control [108]

The following heuristic tuning method is used for tuning of the Fuzzy PI (two-term) controller.

- Use gains $K_p$ and $\frac{T}{K_i}$ of a well-tuned linear PI controller as the initial fuzzy PI gains $K_p/\frac{T}{T_1}$, the initial scaling gains $K_c/\alpha /K$ can be calculated using equation (4.13).
- Tune the value of $K_c /K$ to achieve a faster response and small steady state error without input saturation. Adjusting $K$ and $K_c$ a better control resolution [115] can be obtained.
- Tune $\alpha$ to achieve a faster response and a smaller steady state error in fuzzy PI.

The above steps are repeated until the satisfactory performance is achieved.

C) Simulation Results

The simulation of the fuzzy-PI controller is done using Fuzzy Toolbox of the MATLAB-SIMULINK software package. Using fuzzy toolbox the membership function definition of the inputs and output of fuzzy-PI controller, fuzzification, rule base evaluation and defuzzification are defined offline. Then fuzzy-PI controller block is integrated with the system (Figure 4.3) and system simulation is carried out using the fifth-order Runge-Kutta method.

The quantitative criteria for measuring the performance chosen are settling time and steady state error of the step response of the system. The initial performance of fuzzy-PI controller is shown in Table 4.2 and compared with PI in Figure 4.5. From the simulation results it is observed that the fuzzy-PI with a linear rule base has similar...
characteristics to its linear counterpart (PI). The separation of the proportional and the integral effect in linear PI makes the proportional contribution significant at the beginning; while the fuzzy combined proportional and integral effect eliminates the proportional effects in the beginning. Therefore fuzzy-PI is slower in the beginning compared with PI control.

Table 4.2 Fuzzy-PI simulation results

<table>
<thead>
<tr>
<th>No.</th>
<th>K_c</th>
<th>( \alpha )</th>
<th>K</th>
<th>Settling time ( t_s ) - ms</th>
<th>Position Error (Deg.)</th>
<th>( M_p ) - Overshoot (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1</td>
<td>0.2</td>
<td>50</td>
<td>180</td>
<td>0.06</td>
<td>&lt; 1.0</td>
</tr>
<tr>
<td>2\textsuperscript{nd} run</td>
<td>1</td>
<td>0.2</td>
<td>30</td>
<td>170</td>
<td>0.08</td>
<td>Nil</td>
</tr>
<tr>
<td>Final</td>
<td>1</td>
<td>0.1818</td>
<td>23.5</td>
<td>140-145</td>
<td>0.1</td>
<td>Nil</td>
</tr>
</tbody>
</table>

Figure 4.5 Performance evaluation of fuzzy-PI controller

Step Response: 10 Deg. Load: Rated Load
1- Input
2- Initial - \( \alpha=0.2, K=50 \)
3- \( \alpha=0.2, K=30 \)
4- Final - \( \alpha=0.1818, K=23.5 \)
4.4 Fuzzy Logic Controller – A Proposed Scheme

The block diagram of the proposed FLC for the PMDC motor drive is shown in Figure 4.6. It consists of the motor drive, position translator (gearbox, etc.), position feedback (encoder interface), PWM amplifier and the fuzzy logic controller (FLC).

The proposed fuzzy logic controller is designed and simulated using Fuzzy Toolbox of the MATLAB-SIMULINK software package. Using fuzzy toolbox the membership function definition of the inputs and output of fuzzy controller, fuzzification, rule base evaluation and defuzzification are defined offline. Using the concept of the shrinking-span membership function [120] the membership functions of the proposed FLC are generated. The fuzzy control rules are developed based on the expert experience and from observing the system (PI controller) response characteristics over a specified operating range. The fuzzy controller block is integrated with the system (Figure 4.7) and system simulation is carried out using the fifth-order Runge-Kutta method. The design of the proposed FLC is briefly described as follows.

4.4.1 Fuzzification

The system-input variables are defined as the position error $e_o$ (difference between the preset and the measured shaft position) and the “rate of change of error” or how quickly the feedback position at time $t$ is changing with respect to the previous sample at time $t-1$. The output control variable $C_I$ determines the action to be performed upon evaluation of the rules.

$$e_o(k) = \theta_s(k) - \theta_o(k)$$  \hfill (4.15)

$$\Delta e_o(k) = \frac{d \theta_o(k)}{dt} = e_o(k) - e_o(k-1)$$  \hfill (4.16)

Where

- $\theta_o(k)$ = Output of the shaft encoder at kth sampling instant
- $\theta_s(k)$ = Set point value at the kth sampling instant
- $e_o(k)$ = Error of the servo system at kth sampling instant
- $\Delta e_o(k)$ = Rate of change of error.

The output control variable is $C_I$ (voltage), which after signal conditioning and amplification is used to control the duty cycle of the PWM signal. This PWM (output) signal ranges (duty cycle) from 0% to 100% with a resolution of 0.4% which finally drives the MOSFET Bridge to control the motor shaft position.
As the maximum position input limit to the controller was 20 degrees, the range of error input is taken to be [-20, 20]. Similarly by taking the derivative of the error input (in PI controller) and in order to take care of additional change in error caused by noise the range of the rate of change of error is taken as [-420, 420]. This range is obtained after some trial and error procedure.

Thus, the universe of the discourse of the error, rate of change of error, and the control variable are

\[ e_0 = -20 \text{ to } +20 \] (position error range)

\[ \Delta e_0 = -420 \text{ to } +420 \] (rate of change of error range)

\[ V = -1.5 \text{ to } +1.5 \] (control voltage for PWM amplifier)

The universe of discourse for all the variables (linguistic) viz., error, rate of change of error and the control output are confined to the range [-1, 1] by properly selecting the scaling factors. In this investigation, to start with the following linguistic variables are used (Table 4.3). Obviously, increasing the number of labels of the input and output variables will increase the number of rules needed to perform a proper control action.

Table 4.3 Linguistic Fuzzy Variables

<table>
<thead>
<tr>
<th>Linguistic variable</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Big</td>
<td>PB</td>
</tr>
<tr>
<td>Positive Small</td>
<td>PS</td>
</tr>
<tr>
<td>Zero</td>
<td>Z</td>
</tr>
<tr>
<td>Negative Small</td>
<td>NS</td>
</tr>
<tr>
<td>Negative Big</td>
<td>NB</td>
</tr>
</tbody>
</table>

4.4.2 Membership Functions

In the real application of the FLC, the membership functions are constructed by using knowledge of the domain experts and then are modified by laboriously surveying the control response of the process. Majority of the researchers has chosen the equal-span isosceles triangular or trapezoidal membership functions for their FLCs [121, 122, 123, and 124]. The main advantage of choosing this type of membership function is that these membership functions ease the difficulties in analysing the structure of the FLC. However, in practical applications involving nonlinear processes an FLC with equal span triangular membership function is not adequate to achieve a good control result. To overcome these problems C. L. Chen and C. T. Hsieh [120] introduced the concept of shrinking-span membership function. The method allows to
construct a set of membership functions, called shrinking-span membership functions (SSMFs), for a specific linguistic variable systematically by using only two parameters; number of elements of the term set and the shrinking factor for that linguistic variable.

For the triangular membership function $\Lambda(x; a, c, d)$ is defined in equation (4.17).

$$\Lambda(x; a, c, d) = \begin{cases} 0 & \text{for } x \leq a \\ \frac{(x-a)}{(c-a)} & \text{for } a \leq x \leq c \\ \frac{(d-x)}{(d-c)} & \text{for } c \leq x \leq d \\ 0 & \text{for } d \leq x \leq x_u \end{cases}$$

The SSMFs $A^{\beta(i, l)}(x_i)$ for linguistic variable $x_i$:

$$A^{\beta(i, l)}(x_i) = \left\{ x_i : \frac{(1+\beta)\Lambda^{\star(i,l)}-1}{2} + (1-\beta)\Lambda^{\star(i,l)} - \frac{(1-\beta)\Lambda^{\star(i,l)} + (1+\beta)\Lambda^{\star(i,l)+1}}{2} \right\}$$

For $l_i \in I_m$  

(4.18)

Here $A^{\star(i, l)}$ is the principle value of $A(i, l)(x_i)$ defined by

$$A^{\star(i,l)} = \frac{l_i}{m_i} S_i, \quad l_i \in I_{m_i}$$

(4.18)

Where $S_i \in [0, 1]$ is the shrinking factor for linguistic variable $x_i$ and $\beta$ is the overlapping factor. The $I_{m_i} = \{-m_i, ..., -1, 0, 1, ..., m_i\}$ is the index set with $M_i$ ($M_i = 2m_i + 1$) terms for linguistic variable $x_i$.

The factor $\beta$ can take values greater than unity as long as the resultant membership functions are rational in applications. The overlapping region increases monotonically as $\beta$ increases. For $\beta = 0$, there is no overlap between the SSMFs and if $\beta = 1$ the support for the SSMFs have proper overlapping regions. By applying various shrinking factors to the same linguistic variable, different membership functions can be obtained. For the proposed controller (FLC) the number of linguistic members are set to five for both the inputs (error and rate of change of error) and the output control variables (CI). The triangular membership functions are used in this application to reduce complexity in calculation.

The effect of the shrinking factors on the control performance was simulated by applying different combinations of $S_c$ (for $e_0$), $S_{\Delta e_0}$ (for $\Delta e_0$), $S_v$ (for output $V$) to the SSMFs with constant parameters ($m = 2$) using Fuzzy toolbox of Matlab-simulink software package. The resultant membership functions used in this application with
optimized values of the shrinking factor \( S = 0.7 \) and overlapping factor \( \beta = 1 \) is shown in Figure 4.8.

![Figure 4.6 Block diagram of a FLC compensated PMDC motor drive](image)

Figure 4.6 Block diagram of a FLC compensated PMDC motor drive

![Figure 4.7 Position control of PMDC motor using fuzzy logic controller (FLC)](image)

(A) Block diagram. (B) FLC structure
4.4.3 Procedure for Deriving Fuzzy Control Rules

There are four main approaches that can be employed to derive fuzzy control rules.

- Based on empirical knowledge of skilled human operator.
- Based on the desired response of the process.
- Based on implicit fuzzy Model of the process.
- Based on learning algorithm.
The first approach is based on empirical knowledge of skilled human operator, if the plant has previously been manually controlled. Mamdani suggested this and the knowledge being represented as fuzzy algorithm, synthesizes the linguistic control protocol of skilled human operator. The knowledge acquisition problem that may arise by adopting this approach is as follows. The operators may not be able to completely explain the control strategy that they adopt or they may be uncommunicative. Different control strategies may be offered by individual based on the control engineering sense. Control engineers can derive a rule base from his or her working knowledge of the plant on the basis of a desired control objective (a desired process response curve) as in Fig. 4.9 [75]. Such an approach is limited to applications in which plant exists in model, prototype or operational form. The same approach is followed in this research to derive the control rules from the step response obtained of the PI controller under static and dynamic tests. Mac Vicar-whelan [125] tried to overcome some of the limitations by providing few general rules for the structure of fuzzy controller [126] [127]. The fourth approach is that the controller itself generates its own control rules based on its performance. This falls into the category of fuzzy self-organizing controller [128].

4.4.3.1 Fuzzy Control Rules Based on Desired Response Curve

The specifications for a control system are often expressed in the time domain and these specifications usually include a certain transient response and steady-state error for a specific input such as a step or a ramp input. The transient response specifications for step input are shown in Figure 4.9 and include the following.

(i) Rise time ($t_r$)

(ii) Percent overshoot and settling time ($t_s$)

The rise time is defined as the time for the output to rise from 10% to 90% of the step input. The settling time is the time required for the output to settle within a given percentage of the desired final value. The typical number for this percentage is 2%. The steady-state specification in the time domain for step input is also shown in Figure 4.9. So to improve the step response, the rise time, percent overshoot, and steady-state error should all be minimized.
A typical desired closed-loop response of a feedback controlled first order process is shown in Figure 4.10 with response divided into various zones on the time axis. From this response curve, heuristics and control engineering sense can be used to derive rules. In order to improve the response the rules are derived by considering an action to be taken at a certain point in different zones marked in the Figure 4.10. The Fuzzy Control Rules can be written for the term sets of input/output variables having the cardinality of three or more (Table 4.4 and Table 4.5). But as the rule base (number of rules) increases the computational time of the control algorithm increases thus degrading the dynamic performance of the system. For the present research, with five linguistic variables (labels) for the inputs and output twenty-five rules (listed in Table 4.6, Table 4.7) are derived based on the step response and frequency response results of the PI controller under step load change conditions. However, in order to optimize the rule base (number...
(of rules) some of the control actions in the rule table are also developed using “trial and error” and from an “intuitive” feel of the system being controlled.

### Table 4.4 Fuzzy control rules with term set 3

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>$E_o$</th>
<th>$\Delta E_o$</th>
<th>$C_i$</th>
<th>Reference Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>P</td>
<td>Z</td>
<td>P</td>
<td>a</td>
</tr>
<tr>
<td>2.</td>
<td>Z</td>
<td>N</td>
<td>N</td>
<td>b</td>
</tr>
<tr>
<td>3.</td>
<td>N</td>
<td>Z</td>
<td>N</td>
<td>c</td>
</tr>
<tr>
<td>4.</td>
<td>Z</td>
<td>P</td>
<td>P</td>
<td>d</td>
</tr>
<tr>
<td>5.</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>set point</td>
</tr>
<tr>
<td>6.</td>
<td>P</td>
<td>N</td>
<td>Z (P)</td>
<td>5 (1)</td>
</tr>
<tr>
<td>7.</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2.6</td>
</tr>
<tr>
<td>8.</td>
<td>N</td>
<td>P</td>
<td>Z (N)</td>
<td>7. (3)</td>
</tr>
<tr>
<td>9.</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>4. 8</td>
</tr>
</tbody>
</table>

### Table 4.5 Fuzzy control rules with term set 7

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>$E_o$</th>
<th>$\Delta E_o$</th>
<th>$C_i$</th>
<th>Reference Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>PB</td>
<td>ZE</td>
<td>PB</td>
<td>a</td>
</tr>
<tr>
<td>2.</td>
<td>ZE</td>
<td>NB</td>
<td>NB</td>
<td>b</td>
</tr>
<tr>
<td>3.</td>
<td>NB</td>
<td>ZE</td>
<td>NB</td>
<td>c</td>
</tr>
<tr>
<td>4.</td>
<td>ZE</td>
<td>PB</td>
<td>PB</td>
<td>d</td>
</tr>
<tr>
<td>5.</td>
<td>ZE</td>
<td>ZE</td>
<td>ZE</td>
<td>set point</td>
</tr>
<tr>
<td>6.</td>
<td>PM</td>
<td>ZE</td>
<td>PM</td>
<td>c</td>
</tr>
<tr>
<td>7.</td>
<td>ZE</td>
<td>NM</td>
<td>NM</td>
<td>f</td>
</tr>
<tr>
<td>8.</td>
<td>NM</td>
<td>ZE</td>
<td>NM</td>
<td>g</td>
</tr>
<tr>
<td>9.</td>
<td>ZE</td>
<td>PM</td>
<td>PM</td>
<td>h</td>
</tr>
<tr>
<td>10.</td>
<td>PS</td>
<td>ZE</td>
<td>PS</td>
<td>1</td>
</tr>
<tr>
<td>11.</td>
<td>ZE</td>
<td>NS</td>
<td>NS</td>
<td>j</td>
</tr>
<tr>
<td>12.</td>
<td>NS</td>
<td>ZE</td>
<td>NS</td>
<td>k</td>
</tr>
<tr>
<td>13.</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>1</td>
</tr>
<tr>
<td>14.</td>
<td>PB</td>
<td>NS</td>
<td>PM</td>
<td>1 (rise time)</td>
</tr>
<tr>
<td>15.</td>
<td>PS</td>
<td>NB</td>
<td>NM</td>
<td>1 (overshoot)</td>
</tr>
<tr>
<td>16.</td>
<td>NB</td>
<td>PS</td>
<td>NM</td>
<td>3</td>
</tr>
<tr>
<td>17.</td>
<td>NS</td>
<td>PB</td>
<td>PM</td>
<td>3</td>
</tr>
<tr>
<td>18.</td>
<td>PS</td>
<td>NS</td>
<td>ZE</td>
<td>9</td>
</tr>
<tr>
<td>19.</td>
<td>NS</td>
<td>PS</td>
<td>ZE</td>
<td>10</td>
</tr>
</tbody>
</table>

### Table 4.6 Fuzzy control rules with term set 5

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>$E_o$</th>
<th>$\Delta E_o$</th>
<th>$C_i$</th>
<th>Reference Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>PB</td>
<td>ZE</td>
<td>PB</td>
<td>a</td>
</tr>
<tr>
<td>2.</td>
<td>ZE</td>
<td>NB</td>
<td>NB</td>
<td>b</td>
</tr>
<tr>
<td>3.</td>
<td>NB</td>
<td>ZE</td>
<td>NB</td>
<td>c</td>
</tr>
<tr>
<td>4.</td>
<td>ZE</td>
<td>PB</td>
<td>PB</td>
<td>d</td>
</tr>
<tr>
<td>5.</td>
<td>ZE</td>
<td>ZE</td>
<td>ZE</td>
<td>set point</td>
</tr>
<tr>
<td>6.</td>
<td>PS</td>
<td>ZE</td>
<td>PS</td>
<td>1</td>
</tr>
<tr>
<td>7.</td>
<td>ZE</td>
<td>NS</td>
<td>NS</td>
<td>j</td>
</tr>
<tr>
<td>8.</td>
<td>NS</td>
<td>ZE</td>
<td>NS</td>
<td>k</td>
</tr>
<tr>
<td>9.</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>1</td>
</tr>
<tr>
<td>10.</td>
<td>PS</td>
<td>NS</td>
<td>ZE</td>
<td>1</td>
</tr>
<tr>
<td>11.</td>
<td>NS</td>
<td>PS</td>
<td>ZE</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4.7 Rule table for FLC

<table>
<thead>
<tr>
<th>$E_o / \Delta E_o$</th>
<th>NB</th>
<th>NS</th>
<th>Z</th>
<th>PS</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>NB</td>
<td>NB</td>
<td>NB</td>
<td>NS</td>
<td>Z</td>
</tr>
<tr>
<td>NS</td>
<td>NB</td>
<td>NB</td>
<td>NS</td>
<td>Z</td>
<td>PB</td>
</tr>
<tr>
<td>Z</td>
<td>NB</td>
<td>NS</td>
<td>Z</td>
<td>PS</td>
<td>PB</td>
</tr>
<tr>
<td>PS</td>
<td>NS</td>
<td>Z</td>
<td>PS</td>
<td>PB</td>
<td>PB</td>
</tr>
<tr>
<td>PB</td>
<td>Z</td>
<td>PS</td>
<td>PB</td>
<td>PB</td>
<td>PB</td>
</tr>
</tbody>
</table>

4.4.3.2 Inference Engine

The Inference engine (IE) uses the data driven (forward chaining) strategy to make the inferences. All chains of reasoning are only one inference long, because in process control there is no need to generate and test hypothesis [129].

The Max-min type implication (proposed by Mamdani) is used to represent the meaning of the individual rules. The individual rule based inference is used here, because it is computationally very efficient and saves a lot of memory.

4.4.3.3 Metarules

First version of the fuzzy algorithms (with 25 rules) were time consuming, because all the rules were executed, each sample time, consuming about 1.2 ms. New versions use metarules to decrease the time consumed by IE. They actuate over the rule base decreasing the number of rules, which is needed to infer to obtain the control output. Metarules select, first according to the error value and then according to the rate of change of error value, which rules are convenient to infer and which are not.

Error metarules are of type:

"**IF** error $(e_o)$ is bigger than -0.5 (NS or NB)  

**THEN** only consider rules 1 to 8."

Next, inside this subset, metarules of similar form to the rate of change of error are introduced.

"**IF** $-0.25 < \Delta e_o < -0.1$  

**THEN** only consider rules 1,2,6,7."

The computing time with these metarules reduced to 0.3 ms approximately.
4.4.4 Defuzzification

The fuzzy algorithm use the center of gravity as defuzzification method:

\[ u(t) = \frac{\sum [E_i * Y_i]}{\sum E_i} \] (4.19)

Where, \( E_i \) represents the fulfilment grade of the premises, as the result of applying minimum or product implication function. \( Y_i \) represents the linguistic output variable. The normalized output (\( U_n \)) is also restricted to the interval \([-1, +1]\). Thus it is necessary to multiply it by output gain \( G_o \) in order to obtain a required appropriate signal level to drive the PWM amplifier. Because of the special properties (discontinuity) of the triangular membership functions, discontinuity in the FLC output is observed. The 2\textsuperscript{nd} order lowpass filter (\( f_c \sim 5 \text{ kHz} \)) is used to avoid the discontinuity.

Figure 4.11 shows the detailed behavior of an entire fuzzy inference system (using Matlab- fuzzy toolbox facility) and the surface plot of the fuzzy controller showing the behavior of the fuzzy output signal as function of error and rate of change in error.

![Figure 4.11 FLC behavior](image-url)

(A) Control surface
(B) Rule Inferencing

![Typical rule inferencing](image-url)
4.5 Control Implementation

Many hardware implementations have been proposed in the past ten years for high speed fuzzy information processing. Togai and Watanabe proposed a design for a special purpose VLSI chip for processing fuzzy logical rules with a speed of 80KFLIPS (Fuzzy Logic Inference per Second) [130]. After this approach, faster and more elaborate fuzzy inference hardware systems have been developed [131]-[135]. The active rule driven architecture improved the inferencing speed [136,137] and several fuzzy logic controllers with the optimized memory organization were proposed [138]-[138]. Appendix-I lists the details of some of the available fuzzy hardware and systems. However, they are limited in generality, that is, there is no flexibility in inference methods. The numbers of inputs/outputs (I/P) are limited and support only small numbers of fuzzy operations.

The standard PC based control approach using MATLAB (Simulink and Fuzzy LogicToolbox) software package and high performance data acquisition and control card (DAS) is adopted for the implementation of the developed fuzzy logic algorithms. Then the implementation aspect of fuzzy logic controller using an inexpensive 16-bit microcontroller (Intel’s 80C196KB) is also presented.

The Fuzzy Logic Toolbox lets engineers create and edit fuzzy inference system by hand, with interactive graphical tools or command-line functions, or by generating them automatically with clustering or adaptive neuro-fuzzy techniques. Simulink, the simulation tool that runs alongside MATLAB, makes it easy to test the developed fuzzy system in a block diagram simulation environment. The Real-Time Workshop generates the portable C (ANSI compliant) code from the Simulink environment for use in realtime or non-realtime applications. Using the above facility the developed FLC is simulated under the MATLAB (Simulink) environment using fuzzy logic toolbox for static and dynamic test (step response and frequency response) conditions. The controller is tuned (offline) to get satisfactory response. Using the Real-Time workshop the executable ANSI C code is generated to link with Turbo C (ver. 1.0) software package for realtime implementation.

Figure 4.12 shows the block diagram of the realtime implementation of FLC using high performance PC/AT -80486DX4 (@ 100 MHz) system. This PC-based structure could be easily extended to a multi-axis and/or multi-motor application with the installation of multiple DAS and PWM amplifier cards. The overall execution interval for the position loop with the FLC is 0.4 ms which is less than the required 0.5 ms sampling interval.
4.5.1 Hardware

The FLC system consists of the following components:

- IBM-PC AT 80486DX4 (@ 100 MHz) computer.
- Data acquisition and control (DAS) card consisting of timer / counters, ADC, DAC, etc.
- PWM Amplifier including base drive and MOSFET bridge.
- PMDC servomotor (18P417, Muirhead Vactric make), gearbox with shaft encoder (Hewlett Packard HEDS – 5310).

The DAS card generates interrupt (IRQ3) every 0.5 ms (sample time), to call the position control routines. The details of PWM amplifier including base drive circuit and MOSFET-bridge has already been explained in chapter 3 (section 3.5.1). The details of the DAS card are given below.

PC / AT 80486DX4 (@ 100 MHz)

![Block diagram of the FLC system for PMDC servomotor drive](image)

Figure 4.12 Block diagram of the FLC system for PMDC servomotor drive
4.5.1.1 Data Acquisition and Control (DAS) Card

The hardware implementation of the DAS depends on the specifications of the DAS which, in turn, is governed by instrumentation requirement for the control of PMDC servomotor drive. For any DAS, the three basic specifications are resolution, sampling rate, and accuracy. A market survey indicated that the variety of DAS designs available from the number of manufacturers do not provide a single cost-effective solution to the requirement of the implementation of FLC. Variety of definitions of the specifications in the absence of standards further complicates the selection of an optimum DAS for the FLC application. Therefore a thorough understanding of DAS for FLC application to PMDC motor drive is needed. In view of above, a DAS as an add-on card to PC for implementation of FLC to PMDC motor drive is developed. The main specifications of the DAS are given in Table 4.8. The full details of the DAS design is beyond the scope of the thesis, only brief design procedures and important component selection are described. For details, refer [109,110].

Considering the systems required bandwidth and in order to complete the control task before arrival of the next information from the position sensor (encoder) the sample period \( T_s \) is selected as 500\( \mu \)s. Further the PWM amplifier is operated at 8 kHz switching frequency, so the current and voltage waveforms of the converter are non-sinusoidal and are of variable duty cycle. With a 4th order butterworth presampling low pass filter, an error of 0.025 % (72 db) occurs, when the sampling frequency is 9 times the maximum frequency of interest. Therefore the minimum required sampling frequency for analog to digital converter (ADC) is nearly 80 kHz. The minimum signal to noise ratio (S/N) for industrial operation of the drive to achieve the desired accuracy (0.1 %) is assumed to be 60 db. The resolution of the ADC not only determines the dynamic range but it also limits the accuracy. Since the maximum range of the parameters with superposed ripple is 100:1 and S/N ratio is 60 db, a dynamic range of 72 db (12 bit ADC) is chosen.

The allocation of processing time to individual components is based on total allowable acquisition time and experience. Since the actual (instantaneous) error of the discrete data sample is throughput error of DAS and digital error, the allocation of error to individuals is based on Route Square Sum (RSS) and total allowable error. Based on the individual allocation, 8 channel MUX AD7501, programmable gain instrumentation amplifier AD524, sample and hold amplifier SHA AD585 (3\( \mu \)S acquisition time), 12 bit ADC AD774B (7\( \mu \)S conversion time), and DAC, HA10B (10 bit) are selected. Error budgeting is carried out to confirm the total error to be less than the specified.
The block diagram of the DAS is shown in the Figure 4.13. The other necessary hardware, viz: Control and Timing logic, Address decoding logic, bus interface, interrupt controller interface, DAC, and first in first out (FIFO) static RAM for data storage with direct memory access (DMA) interface, are designed for implementing the additional features (Table 4.8). To achieve the sampling rate of 80 KHz with selected components, facility of auto channel selection, auto start of conversion, and overlap mode is provided. In overlap mode, the total acquisition time is sum of the acquisition and settling time of sample & hold amplifier (SHA) and conversion time of ADC only. As sum of the times of the remaining components are equal to or less than this time and they are process to adjacent samples simultaneously. The FIFO-RAM and DMA interface reduces the burden of the main processor while accessing the converted (ADC) data. The position command input (potentiometer), motor current and voltage are accessed through the ADC. Two 16 bit, up/down counters are used in cascade to form 32 bit up/down counter for counting the number of pulses from the encoder to determine current motor (shaft) position and velocity. 10 bit DAC HA10B along with amplifier and low pass filter is used to convert FLC digital output to analogue output signal to drive the PWM amplifier. The DAS is developed as an integral module of PC/AT based control system for PMDC servomotor.

Table 4.8 Main Specifications of the DAS Card

<table>
<thead>
<tr>
<th>Specification</th>
<th>Specification Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog I/P range ( (jumper ) selectable)</td>
<td>Unipolar: 10V, 1V, 100mV, 10mV Bipolar: ±10V, ±100mV, ±10mV</td>
</tr>
<tr>
<td>Number of channels ( (jumper ) selectable)</td>
<td>16 single ended / 8 differential</td>
</tr>
<tr>
<td>Resolution ( (jumper ) selectable)</td>
<td>12 bit / 8 bit</td>
</tr>
<tr>
<td>Conversion type</td>
<td>Successive approximation</td>
</tr>
<tr>
<td>Conversion speed</td>
<td>80 KHz ( @ ) 12 bit resolution</td>
</tr>
<tr>
<td>Accuracy</td>
<td>&lt; 0.1 % at 25°C</td>
</tr>
<tr>
<td>Trigger mode</td>
<td>Software / on board programmable or external trigger (TTL)</td>
</tr>
<tr>
<td>Data transfer</td>
<td>Polling / Interrupt driven / DMA</td>
</tr>
<tr>
<td>Card addressing ( (jumper ) selectable)</td>
<td>I/O mapping / Memory mapping</td>
</tr>
<tr>
<td>Operational mode ( (software ) selectable)</td>
<td>Serial / Overlap, Auto SOC, Auto Scan. One shot sampling, etc.</td>
</tr>
<tr>
<td>Common mode rejection</td>
<td>70 db at gain G = 1, 110 db at G = 1000</td>
</tr>
<tr>
<td>Digital Input / Output</td>
<td>24</td>
</tr>
<tr>
<td>Analog output</td>
<td>10 bit 2 channel, 0-10V</td>
</tr>
</tbody>
</table>
4.5.2 Software

The languages used were Turbo C++ and Assembly (Macro Assembler). The C++ routines include a file to define the system variables, a module to initialize them, a module to implement the user-friendly interface, a module to manipulate PC interrupts, and module to calculate performance indices. The real time C code of the FLC generated by using Real-Time workshop from the MATLAB Simulink environment is linked to the above C++ file before execution. The time critical routines are developed in Assembly language, making intensive use of the Basic Input / Output subroutines (BIOS).

Figure 4.14 and Figure 4.15 giving the program flow charts, shows that a fuzzy logic development can be divided in two parts: the DAS control environment and fuzzy logic algorithm.

The main subroutine DAS-control includes:

- Initialization of the DAS card, viz. Number of channels, gain selection, ADC data acquisition control mode, timers and counters, and other on-board peripherals.
- Acquisition of new motor position command, motor current and voltage through Data acquisition subroutine.
- Motor shaft position acquisition – reading of the 32 bit up/down counter,
- Calculation of position error value and position error variation.
- Calling the FLC algorithm.
- Generation of output through DAC to control the duty cycle of the PWM signal.

The timer is programmed to generate an interrupt (IRQ3) every 0.5ms. FIFO memory is cleared before acquisition of new ADC data values. The DAS card is initialized for auto scanning of the 3 ADC channels with programmable gain of 1, selection of auto start of conversion (SOC) and Overlap mode with DMA control for ADC data collection.

The fuzzy algorithm consist of the executable C code generated by Real-Time workshop of the MATLAB (SIMULINK) software package. This part is made of fuzzification of the input variables, execution of the activated rules (Rule Inferencing), and defuzzification producing the output variable.
Figure 4.14 Flow chart of the main subroutine
4.5.3 Implementation of Microcontroller Based Fuzzy Controller

With microcontrollers being available in different configurations, low cost, small die sizes, power management features and high clock rates, it is reasonable to apply fuzzy logic in an inexpensive microcontroller to implement the fuzzy control loop. Few applications of FLC using 8-bit microcontrollers [141] and digital signal processors (DSP) [142] are reported in literature. It is noted that, most DSP systems are not optimized for nonlinear arithmetic, despite being able to compute very quickly and with high precision. Also, most DSP's do not contain the full complement of functions needed to implement a feedback system, i.e., functions like on chip analog-to-digital converter (ADC), pulse width modulation (high speed I/O processing). On the other
hand, low-cost microcontrollers often include a wealth of control and communication functions in hardware, but with limited computation power and memory. Unlike earlier 8-bit microcontroller architectures with limited math capability, the Intel high performance, low cost, 16-bit 8XC196Kx microcontroller (architecture) is used to implement a closed loop fuzzy control system because of the following reasons:

- Improved math capability (execution timing).
- The powerful instruction set makes use of the register-register-based architecture with various addressing modes and a rich set of peripherals.
- Data acquisition and processing is done easily and efficiently.
- Software can be implemented with minimum memory and fast execution (upto 50MHz) by locating the code/data in internal chip memory.

Rules constitute the base of the algorithm and are evaluated in sequence, one after the other. Upon completion of processing of all rules, the final system output is calculated as previously described. In contrast, if a custom dedicated fuzzy parallel processor were to be used, rules could be evaluated in parallel. The parallel processing method suggests a fast processing cycle. However, in this case data acquisition and data output continues using conventional peripherals. The time gained in parallel rule processing can be lost in acquiring and manipulating data via external peripherals.

The focus here is limited to the feasibility of implementing a microcontroller based fuzzy logic controller (FLC) in a cost-effective manner. The important issues of performance advantage and design approaches remain open for further study. The following section discusses the implementation aspect of the FLC using Intel 80C196KC microcontroller.

4.5.3.1 Control Implementation

The direct implementation of fuzzification, rules inferencing or decision-making, and defuzzification equations in the basic fuzzy algorithm is computationally prohibitive on the low-cost microcontroller platform. Another approach is to use of the development system, like Fuzzy logic Toolbox with MATLAB (Simulink), fuzzyTECH MCU-96, etc. for development of optimized FLC algorithm and generation of real-time executable code suitable for selected microcontroller (Intel 8xC196 family). The Fuzzy logic Toolbox with MATLAB (Simulink) environment is selected for the FLC implementation on Intel’s 80C196KX family microcontrollers. Figure 4.16 shows a possible implementation aspect of FLC using 80C196KC microcontroller.
The Intel 80C196KC microcontroller is the next step up in the CHMOS 196 family. It is available in CPU only, 16 Kbyte ROM and 16 Kbyte OTPROM versions. All versions feature 488 bytes of Register RAM. The 80C196KC is offered in a 20 MHz version, allowing an immediate 25% increase in performance. The 80C196KC has all the same peripherals as the 80C196KB, but adds the following features:

- There are now a total of three hardware PWM generators,
- The A/D converter has both 8- and 10-bit conversion modes with programmable sample and conversion times, and
- Peripheral Transaction Server (PTS) has been added.

The PTS acts as a microcoded interrupt handler, which greatly reduces CPU overhead during interrupt servicing. The High Speed Input Output unit can be used to effectively handle I/O without impacting precious on-chip timer resources. The high-speed input unit (HSI) is used to measure position and speed information from the encoder pulses. The PWM unit is used to generate PWM pulses for driving the PMDC servomotor. The on-chip serial port is used to communicate with the host (PC/AT-486DX4) for execution of the FLC algorithm. The use of on-chip peripherals viz., A/D, Pulse Width Modulator (PWM) unit, memory unit, and serial port reduces the chip count, thus reducing the overall cost of the system.

The ADC is used to sense reference command input, motor voltage and motor current. Program instructions and data (knowledge base, i.e., the rules and the membership functions) can be stored on-chip for optimized execution. No long external bus cycles are required to read data due to the large register based architecture. This feature is extremely beneficial to fuzzy logic applications.

![Figure 4.16 Implementation of FLC using 80C196KC](image-url)
4.5.3.2 Optimization of Fuzzy Code

Offline optimization allows system performance analysis using model simulation while on-line optimization allows process hardware to be connected to the host system and optimization of the fuzzy controller during runtime. Since simulations tend to be approximations of actual system behavior as they are models based, appropriate optimization of the fuzzy system is done on-line taking feedback into consideration. The 80C196KC-evaluation kit is connected to the host (PC/AT-486DX4) system enabling “on the fly” optimization by changing system parameters and visualizing inference flow in real-time. The generated ANSI C code is recompiled using MATLAB – C compiler without the on-line option and integrated into the 80C196KC hardware system in either ANSI C or assembly by using the 80196-C (Ver. 5.0) compiler from Tasking Software, BV. The recompiled code is compact, and portable to implement on 80C196KC-evaluation kit.

Table 4.9 shows the typical performance of several test systems on a 20 MHz, 80C196KC device using the FLC, and Table 4.10 shows the worst case overall execution times for execution of the control algorithms for driving PMDC servomotor.

Table 4.9 Typical test system performance of FLC on 80C196KC [143]

<table>
<thead>
<tr>
<th>7 Rules</th>
<th>20 Rules</th>
<th>20 FAM Rules</th>
<th>80 FAM Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 in/1 out</td>
<td>2 in/1 out</td>
<td>2 in/1 out</td>
<td>3 in/1 out</td>
</tr>
<tr>
<td>0.22 ms</td>
<td>0.33 ms</td>
<td>0.34 ms</td>
<td>0.50 ms</td>
</tr>
</tbody>
</table>

Table 4.10 Worst case execution timing of the control algorithms

<table>
<thead>
<tr>
<th>Software Timer Interrupt Routine</th>
<th>25 µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLC Algorithms:</td>
<td></td>
</tr>
<tr>
<td>PC/AT486DX4, @100 MHz</td>
<td></td>
</tr>
<tr>
<td>80C196KC, @ 20 MHz</td>
<td>Approx. 300 µs</td>
</tr>
<tr>
<td>HSI Interrupt Processing</td>
<td>16 µs</td>
</tr>
<tr>
<td>HSO Generate PWM Routine</td>
<td>12 µs</td>
</tr>
<tr>
<td>Receive Interrupt and Serial data Processing</td>
<td>42 µs</td>
</tr>
<tr>
<td>DAS card control</td>
<td>15 µs</td>
</tr>
<tr>
<td>Data Acquisition (position data &amp; ADC data)</td>
<td>25 µs</td>
</tr>
<tr>
<td>DAC output</td>
<td>10 µs</td>
</tr>
</tbody>
</table>
4.5.3.3 Conclusive Remarks

The FLC can be designed and implemented in a much easier and quicker way than a classical controller. The 80C196KC 16-bit microcontroller is designed to handle high-speed calculations and fast I/O operations. The large register file with single cycle multiply/accumulate instructions was used to efficiently implement a low cost, simple, model free fuzzy controller. Transfer functions were not used and a mathematical model was not considered. Table 4.9 and Table 4.10 show the worst case execution timings for the control algorithm by using combination of PC/AT-486DX4 (for FLC) with 80C196KC evaluation kit and using 80C196KC evaluation kit only. It is noted that by using 80C196KC evaluation Kit the overall control execution time (nearly 600 µs) may exceed the 0.5 ms sampling time requirement. While using the combination of PC/AT-486DX4 and 80C196KC-evaluation kit the execution time (nearly 400 µs) is well within the limit. The extra time may be utilized for implementation of intelligent fault diagnostic application.

4.6 Simulation and Experimental Results

The designed PC/AT 80486DX-4 based FLC is simulated for various static and dynamic (step and frequency response) test conditions under MATLAB Simulink environment using the fuzzy logic toolbox. Runge kutta -5 algorithm is selected for simulation of the designed controller. The designed controller was verified by simulation. Developed FLC was tested for following test conditions and test results obtained are reported in Figures 4.17 to Figure 4.22. The results obtained viz. position error, settling time, and phase lag, etc. are also depicted in the figures.
4.6.1 FLC Controller Results

- Step response results of FLC (Figure 4.17)
  (a) - Motor current, (b) - motor torque,
  (c) - Speed output and (d) - Position output

- Step response results of FLC (Figure 4.18)
  a) 1 Deg., b) 5 Deg., c) 10 Deg., d) 20 Deg., with load and
  e) 10 Deg., with step change in load

- Frequency response results for 2 Deg., sinusoidal input, (Figure 4.19)
  a) 1 Hz, b) 2 Hz, c) 3 Hz, d) 4 Hz, e) 5 Hz with load

- Frequency response results for 10 Deg., sinusoidal input, (Figure 4.20)
  a) 1 Hz, b) 2 Hz, c) 3 Hz, d) 4 Hz, e) 5 Hz with load

- Frequency response results for 1 degree, sinusoidal input, (Figure 4.21)
  a) 1 Hz, b) 2 Hz, c) 5 Hz without load

- Frequency response results for 1 degree, (Figure 4.22)
  a) 1 Hz ramp input, no load, b) 1 Hz pulse input, with load
Figure 4.17 Step response results of FLC
(a) – Motor current, (b) – Motor torque,
(c) – Speed output and (d) – Position output
a) 1 Deg., step input  
Load: Rated  
Pos. error: 0.1 Deg.  
$\tau_2$: 100-110 ms  
Overshoot: Nil

b) 5 Deg., step input  
Load: Rated  
Pos. error: 0.11 Deg.  
$\tau_2$: 110-120 ms  
Overshoot: Nil

c) 10 Deg., step input  
Load: Rated  
Pos. error: 0.1 Deg.  
$\tau_2$: 155-160 ms  
Overshoot: Nil

d) 20 Deg., step input  
Load: Rated  
Pos. error: 0.6 Deg.  
$\tau_2$: 250-260 ms  
Overshoot: Nil

e) Step change in load  
applied at 0.21 sec.  
Pos. error: < 0.1 Deg.  
Recovery: 50 ms

Figure 4.18 Step response results of FLC  
a) 1 Deg., b) 5 Deg., c) 10 Deg., d) 20 Deg., with load  
and e) 10 Deg., with step change in load
Figure 4.19 Frequency response of FLC for 2 Deg. sinusoidal input

a) 2 Deg., 1 Hz
Pos. error: 0.1 Deg.
Phase lag: 0.36 Deg.
A - Input, B - Output

b) 2 Deg., 2 Hz
Pos. error: 0.14 Deg.
Phase lag: 1.52 Deg.
A - Input, B - Output

c) 2 Deg., 3 Hz
Pos. error: 0.15 Deg.
Phase lag: 4.32 Deg.
A - Input, B - Output

d) 2 Deg., 4 Hz
Pos. error: 0.16 Deg.
Phase lag: 7.2 Deg.
A - Input, B - Output

e) 2 Deg., 5 Hz
Pos. error: 0.2 Deg.
Phase lag: 10.8 Deg.
A - Input, B - Output
a) 10 Deg., 1 Hz
Pos. error: 0.14 Deg.
Phase lag: 2.52 Deg.
A - Input, B - Output

b) 10 Deg., 2 Hz
Pos. error: 0.67 Deg.
Phase lag: 7.2 Deg.
A - Input, B - Output

c) 10 Deg., 3 Hz
Pos. error: 1.8 Deg.
Phase lag: 21.6 Deg.
A - Input, B - Output

d) 10 Deg., 4 Hz
Pos. error: 3.37 Deg.
Phase lag: 57.6 Deg.
A - Input, B - Output

e) 10 Deg., 5 Hz
Pos. error: 4.76 Deg.
Phase lag: 90 Deg.
A - Input, B - Output

Figure 4.20 Frequency response of FLC for 10 Deg., sinusoidal input

- a) 1 Hz, b) 2 Hz, c) 3 Hz, d) 4 Hz, e) 5 Hz with load
Figure 4.21 Frequency response of FLC for 1 degree, sinusoidal input

a) 1 Deg., 1 Hz
Position error: 0.11 Deg.
Phase lag: 0.6 Deg
Load: No-Load
Backlash: nil

b) 1 Deg., 2 Hz
Position error: 0.12 Deg.
Phase lag: 1.44 Deg
Load: No-Load
Backlash: nil

A- Input, B - Output

A- Input, B - Output

A- Input, B - Output

Figure 4.21 Frequency response of FLC for 1 degree, sinusoidal input

a) 1 Hz, b) 2 Hz, c) 5 Hz without load

A- Input, B - Output

Figure 4.22 Frequency response of FLC for 1 degree, ramp and pulse input

a) 1 Deg., 1 Hz, Ramp Input, with load
Position error: 0.14 Deg
Phase lag: 7.2 Deg
Load: No-Load

b) 1 Deg., 1 Hz, Pulse Input, with load
Position error: 0.14 Deg
Load: No-Load

A- Input, B - Output

A- Input, B - Output

Figure 4.22 Frequency response of FLC for 1 degree, ramp and pulse input

a) 1 Hz ramp input, no load, b) 1 Hz pulse input, with load
4.6.2 Experimental Results

Various tests conducted on the FLC are discussed below and results of step response, frequency response tests are presented below.

4.6.2.1 Step Response

The system response for step input of 1 Deg, 2 Deg, 5 Deg, 10 Deg and 20 Deg were measured. These tests were conducted on all three control surfaces (Lower course rudder, Depth rudder, and upper course rudder) for rated load. The experimental set-up used is shown in Figure 4.23 and results obtained are given in Table 4.11.

![Figure 4.23 Experimental set-up for step response test](image)

4.6.2.2 Frequency Response

This test was conducted only on the lower course rudder. To determine the frequency response of the system sinusoidal input of fixed amplitude was applied. Amplitude of position output and phase lag with respect to input was measured. Test setup is shown in Figure 4.24 and the results obtained are given in Table 4.12.

![Figure 4.24 Experimental set-up for frequency response test](image)

The Figure 4.25 shows the photographs of DAS card and experimental test setup for testing of FLC.
Figure 4.25 Photographs of DAS card and experimental test set-up of FLC
| f (Hz) | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δθ | Δtheta...
<table>
<thead>
<tr>
<th>Frequency (Hz)</th>
<th>Input Amplitude (Peak-Peak)</th>
<th>Deg. Error</th>
<th>Deg. Phase Lag</th>
<th>Deg. Error</th>
<th>Deg. Phase Lag</th>
<th>Deg. Error</th>
<th>Deg. Phase Lag</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5.5</td>
<td>0.16</td>
<td>0.14</td>
<td>0.10</td>
<td>0.18</td>
<td>0.15</td>
<td>0.17</td>
</tr>
<tr>
<td>10</td>
<td>11.0</td>
<td>0.6</td>
<td>0.10</td>
<td>0.10</td>
<td>0.13</td>
<td>0.01</td>
<td>0.12</td>
</tr>
<tr>
<td>20</td>
<td>21.0</td>
<td>0.18</td>
<td>0.02</td>
<td>0.01</td>
<td>0.12</td>
<td>0.01</td>
<td>0.12</td>
</tr>
<tr>
<td>40</td>
<td>41.0</td>
<td>0.36</td>
<td>0.10</td>
<td>0.14</td>
<td>0.22</td>
<td>0.16</td>
<td>0.14</td>
</tr>
<tr>
<td>50</td>
<td>51.0</td>
<td>0.36</td>
<td>0.10</td>
<td>0.14</td>
<td>0.22</td>
<td>0.16</td>
<td>0.14</td>
</tr>
</tbody>
</table>

**Table 4.12 Frequency Response with Raled Load**
4.7 Conclusion

In this chapter the possibility of applying fuzzy algorithms in faster and more accurate controllers such as a PMDC servomotor controller was explored. The basic design procedures of FLC was presented along with types of fuzzy knowledge based controller (FKBC). Design details of fuzzy-PI controller for PMDC motor drive based on the new methodology introduced by Han-Xiong Li for designing and tuning the scaling gains of the conventional fuzzy logic controller was presented. Then design aspect of the fuzzy logic controller (FLC) to improve the performance of PMDC motor drive was presented. Using the concept of the shrinking-span membership function the membership functions of the proposed FLC were generated. The fuzzy control rules were developed based on the expert experience and after studying the various responses obtained from the classical PI controller and fuzzy-PI controller.

The implementation details (hardware and software) of the PC/AT (80486DX4, @ 100 MHz) based fuzzy controller (FLC) using the high performance data acquisition and control card (DAS) for PMDC servomotor drive were presented. The details of the DAS card were also presented in short. Then the feasibility of implementing the microcontroller based fuzzy logic controller (FLC) in a cost-effective manner was presented. It was also found that (Table 4.11) the execution time required for control algorithms using PC/AT-486DX4 based DAS card system is much less (approx. 350μs) compare to using 80C196KC microcontroller option.

Finally, the results of various static and dynamic tests conducted on the implemented FLC were presented. It can be seen that the experimental results were sufficiently close (±6 %) to the simulation results. The difference is mainly due to assumptions / approximation made in simulation and modelling of the FLC system. Further, to emphasize the merits and demerits of the FLC, some comparisons have also been made with the fuzzy-PI controller and linear PI controller under load and supply disturbances. It was found that use of FLC results in smoother step responses with response time comparable to PI controller system and much better than the lead compensator (Table 4.11). It should be noted that smoother response can be obtained by decreasing the constant gains of the original PI controller, but resulting increase in response time is unacceptably large. The frequency response results of FLC are much better than the PI controller and the lead compensator particularly at the small reference inputs (Table 4.12). Further, effect of the load disturbances is less in FLC system compared to PI and lead compensated systems.
A comparative study of proportional-integral (PI), adaptive and fuzzy control for friction compensation approach to improve the performance of the PMDC motor drive is presented in Chapter 5. The design and development methodologies of adaptive fuzzy controller for friction compensation to improve the performance of a PMDC motor speed control system are presented. Results of analysis, simulation and modelling of adaptive fuzzy friction compensator for PMDC motor drive using MATLAB are presented.