CHAPTER 5

DESIGN AND SIMULATION OF MAMDANI-FLC BASED DRIVE SYSTEM

A novel design of an FL controller to control the IM speed using SVPWM technique for voltage source inverter is presented in this chapter. The IM drive's performance largely depends on the usage of electronics coupled with excellent control strategies in the hardware implementation. In this context, the FLC has been found to yield impressive results in drives as the control technology.

The conventional control method requires system mathematical model and the system behavior is not satisfactory due to parametric variations, but FLC based system is designed without using mathematical model. Further, a rule-based FLC is designed using the SVPWM concept and applied to control the speed of an IM, as proposed in this chapter. It has very good accuracy, low cost and simplicity. Flexibility is present while selecting the switching modes of the inverter. In this context, a Matlab / Simulink model is developed for simulation and further validated for speed control through simulations.

Using SVPWM techniques, the duty cycle of the inverter can be calculated. Thus, this method definitely improves the performance of IM compared to the other methods. The proposed method's effectiveness is proved by the simulation results shown at the end of this chapter.

5.1 INTRODUCTION

IMs are maintenance free, robust and have various applications in the industry and hence such motors are used for achieving perfect control in industrial drives. The variable-speed IMs require both a wide operating range of speed and fast torque response, regardless of load variations. Drives are non-linear systems and control of IMs using classical/conventional control may decrease the performance of the system since the accurate model of the plant may not be obtained **[5]**.

SVPWM is one of the advanced PWM methods that could be used for variable speed applications in the industry. In the recent years, this method has gained widespread applications in many of the control-related problems due to its excellent performance characteristics. The SVPWM is used to control the firing angle of the inverter; this, in turn, controls the speed of the IM. Once the speed deviates from the set value due to some parametric variations (thermal/noise /disturbance, etc), this error in speed is considered and a Mamdani-based fuzzy technique is used to bring back the deviated speed to the desired set speed.

A number of researchers have carried out extensive work on the control of various parameters using the Mamdani-based FLC, as shown in the exhaustive literature survey provided in the Literature survey chapter of this thesis. The work presented by various researchers using the PI control methods discussed in Chapter 2 had a couple of disadvantages. One among them is the speed of response i.e., the settling times of the various parameters. In the work presented in this chapter, we have considered this as a sensitive parameter and designed sophisticated controller called as the Mamdani-based fuzzy logic controller, thus improving the dynamic performance of the system compared to the PI method.

5.2 REVIEW OF MAMDANI-BASED FLC DESIGN

A Mamdani FLC is based on a set of control rules, called the fuzzy rules among the linguistic variables **[72]**. These rules are expressed in the form of conditional statements. The internal structure of the developed controller is shown in Fig. 5.1.

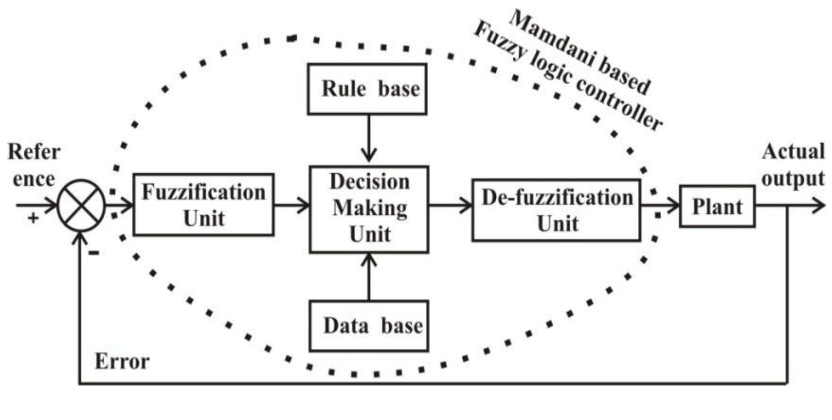


Fig. 5.1: A diagrammatic view of the Mamdani-based FLC

The necessary inputs to the decision-making unit blocks are the rule-based units and the data-based block units. The fuzzification unit converts the crisp data into linguistic formats. The decision-making unit decides in the linguistic format with the help of logical linguistic rules supplied by the rule base unit and the relevant data supplied by the data base **[24], [73]**. The error and the change in error are modeled using Equations. (5.1) and (5.2) as

$$e(k) = \omega_{ref} - \omega_r \tag{5.1}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{5.2}$$

where ω_{ref}→Set speed, ω_r →actual speed, *e(k)* →error, Δ*e(k)* → change in error

The decision-making unit uses the conditional rules of 'IF-THEN-ELSE'. In the first stage, the crisp variables *e(k)* and Δ*e(k)* are converted into fuzzy variables **[72]**. The fuzzification maps the error, and the change in error to linguistic labels of the fuzzy sets. The proposed controller uses the following linguistic labels: {(*nb*→ *negative big*), (*nm* →*negative medium*), (*ns*→ *negative small*), (*ze*→ *zero*), (*ps*→ *positive small*), (*pm*→ *positive medium*) and (*pb*→ *positive big*)}. In this case triangular membership function is used. The rule base for the decision-making unit **[36]** is in Table 5.1.

M-FLC is one of the most commonly used fuzzy methodologies for control applications. Mamdani controller was the first control system developed on the basis of the fuzzy set theory.

Table 5.1: Rule base for controlling the speed of IM using FLC

$e\Rightarrow$ $\Delta e\Downarrow$	*nb*	*nm*	*ns*	*ze*	*ps*	*pm*	*pb*
nb	*nb*	*nb*	*nb*	*nb*	*nm*	*ns*	*ze*
nm	*nb*	*nb*	*nm*	*nm*	*ns*	*ze*	*ps*
ns	*nb*	*nm*	*ns*	*ns*	*ze*	*ps*	*pm*
ze	*nb*	*nm*	*ns*	*ze*	*ps*	*pm*	*pb*
ps	*nm*	*ns*	*ze*	*ps*	*ps*	*pm*	*pb*
pm	*ns*	*ze*	*ps*	*pm*	*pm*	*pb*	*pb*
pb	*ze*	*ps*	*pm*	*pb*	*pb*	*pb*	*pb*

The triangular membership function shown in Fig. 5.2, uses a 2-input Mamdani FIS with 2 rules. A set of linguistic rules was used for control purposes, obtained from the experience of the users. The output MFs are fuzzy sets in the Mamdani controller. Once fuzzification has taken place after the aggregation process, the process of de-fuzzification has to be started, which is a much efficient method.

This concept is called output MF, which is a pre-defuzzified fuzzy set. Computations are greatly reduced when the Mamdani control is used, which makes use of the concept of 2-dimensional function to determine the centroid. Mamdani FLC is shown in the form of a block diagram in Fig. 5.1. This designed Mamdani fuzzy logic controller can be used to control the various parameters of the IM by considering a requisite number of rules. With reference to Fig. 5.2, to design a Mamdani controller, let us consider 2 inputs and 2 rules **[74]**.

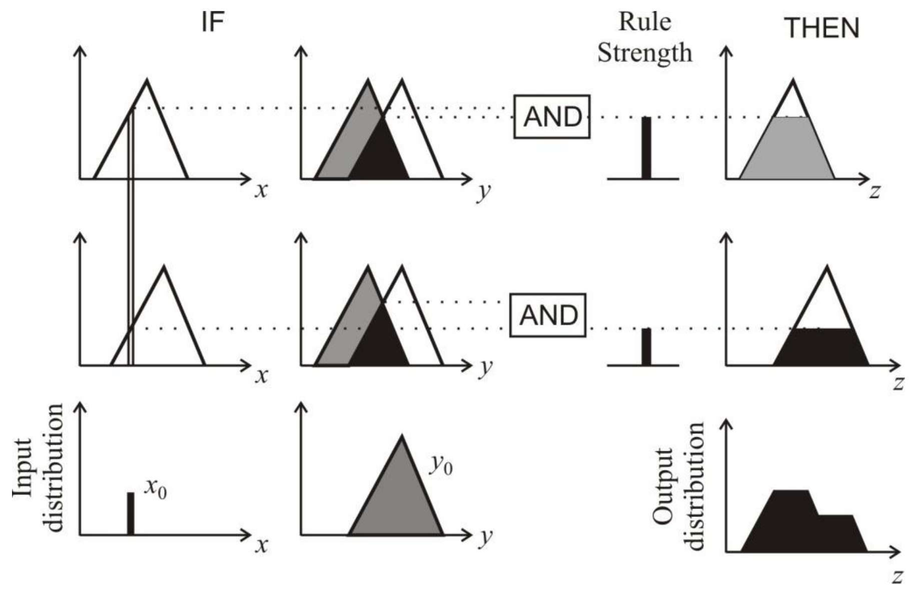


Fig. 5.2: A 2-input, 2-rule Mamdani model

Let the 2 linguistic parameters be *x* and *y*. If *x* is L_x, then *y* is L_y **[75]** be the only control rule where L_x and L_y are the linguistic values taken by the processes state variable *x* and the control output variable *y*. Furthermore, the MFs are given by μ_{lx}: $X\rightarrow$ [0 1] and μ_{ly}: $Y\rightarrow$ [0 1]. Consider 2 crisp inputs y_1 and y_2 with the crisp outputs being given by y_1 and y_2. The crisp outputs using the sources of non-linearity are obtained using the above-mentioned 6 steps (scaling, denormalization, fuzzification, rule firing, defuzzification and generation of crisp outputs) as follows.

Scaling: It is a linear element since it multiplies the inputs with a scalar value P_X

$$P_x.x_1 + P_x.x_1 = P_x.(x_1 + x_2) \text{ and } \rho.P_x.x_1 = P_x.(\rho.x_1). \qquad (5.3)$$

<u>De-normalization</u>: Note that normalization and de-normalization are also linear elements.

<u>Fuzzification</u>: The MF μ_{lx}: of the linguistic value *LX* is considered to be a non-linear function. The fuzzification of x_1 and x_2 results in finding μ_{lx}: (x_1) and μ_{lx}: (x_2). Using the superposition property, this requires

$$\mu_{lX}(x_1) + \mu_{lX}(x_2) = \mu_{lX}.(x_1 + x_2). \tag{5.4}$$

This cannot be fulfilled because of the non-linear characteristic property of $\mu_{lx.}$

<u>Rule firing</u>: Here, we consider the MF μ_{ly} of the linguistic variable *LY* also to be a non-linear function. After firing, the rule for input x_1 is

$$\forall y : \mu'_{CLY}(y) = \mu_{LX}(x_1) \wedge \mu_{LY}(y). \tag{5.5}$$

Similarly, for the input x_2, the rule is obtained as

$$\forall y : \mu''_{CLY}(y) = \mu_{LX}(x_2) \wedge \mu_{LY}(y). \tag{5.6}$$

The superposition property requires

$$\forall y : \mu'_{CLY}(y) + \mu''_{CLY}(y) = \mu_{LX}(x_1 + x_2) \wedge \mu_{LY}(y). \tag{5.7}$$

Note that Equation (5.7) cannot be fulfilled as μ_{LY}, $\mu'_{CLY}(y)$ and $\mu''_{CLY}(y)$ are non-linear functions. The operation '∧' is non-linear for ∧=*min.*

De-fuzzification: Using the centre of area method, defuzzification is performed; as a result, we obtain the crisp outputs *y1* and *y2* as

$$y_1 = \frac{\sum_u \mu''_{CLY}(y).y}{\sum_u \mu'_{CLY}(y)} \quad and \quad y_2 = \frac{\sum_u \mu''_{CLY}(y).y}{\sum_u \mu''_{CLY}(y)} \tag{5.8}$$

Outputs: By adding *y1* and *y2* we will get

$$y_1 + y_2 = \frac{\sum_u \{\mu'_{CLY}(y) + \mu''_{CLY}(y)\}.y}{\sum_u \{\mu'_{CLY} y) + \mu''_{CLY}(y).\}} \tag{5.9}$$

Equation (5.9) can be written using the concept of non-linearity and normalizations as

$$y_1 + y_2 = \frac{\sum_u \mu'_{CLY}(y).y}{\mu'_{CLY}(y)} + \frac{\sum_u \mu''_{CLY}(y).y}{\mu''_{CLY}(y)} \tag{5.10}$$

From Equations (5.9) and (5.10), the knowledge-based controllers use the concepts of non-linearity, membership functions, rule firing and defuzzification. The methodology shown above uses 2 inputs to design the Mamdani-based FLC. The M-FLC is designed with 2 inputs and 1 output. The error in speed was considered as first input, change in error is treated as second input along with one output.

The two input and one output are fuzzified with seven membership function further 49 rule base are written on the basis of knowledge which in turn used for decision making.

The centre of gravity (CG) method is used for defuzzification. The output of the defuzzification unit will give the control commands for generating gating signal. The inverter terminal voltage controlled by these gating signals in turn control the speed of drive. The developed fuzzy rules (7×7=49) are given below:

1 if (speederror is *nb*) and (changeinerror is *nb*) then (output1 is *ns*)(1)
2 if (speederror is *nb*) and (changeinerror is *nm*)then (output1 is *ns*)(1)
3 if (speederror is *nb*) and (changeinerror is *ns*) then (output1 is *ns*)(1)
4 if (speederror is *nb*) and (changeinerror is *ns*) then (output1 is *ns*)(1)
5 if (speederror is *nb*) and (changeinerror is *ps*)then (output1 is *nm*)(1)
6 if (speederror is *nb*) and (changeinerror is *pm*)then (output1 is *ns*)(1)
7 if (speederror is *nb*) and (changeinerror is *pb*) then (output1 is *z*) (1)
8 if (speederror is *nm*) and (changeinerror is *nb*)then (output1 is *ns*)(1)
9 if (speederror is *nm*) and (changeinerror is *nm*)then (output1 is*ns*)(1)
10 if (speederror is *nm*) and (changeinerror is *ns*) then (output1 is *nb*(1)
11 if (speederror is *nm*) and (changeinerror is *z*) then (output1 is *nm*)(1)
12 if (speederror is *nm*) and (changeinerror is *ps*) then (output1 is *ns*)(1)
13 if (speederror is *nm*) and (changeinerror is *pm*) then (output1 is *z*) (1)
14 if(speederror is *nm*) and (changeinerror is *pb*) then (output1 is *ps*)(1)
15 if (speederror is *ns*) and (changeinerror is *nb*) then (output1 is *ns*) (1)
16 if (speederror is *ns*) and (changeinerror is *nm*) then (output1 is *nb*) (1)
17 if (speederror is *ns*) and (changeinerror is *ns*) then (output1 is *nm*) (1)
18 if (speederror is *ns*) and (changeinerror is *z*) then (output1 is *ns*) (1)
19 if (speederror is *ns*) and (changeinerror is *ps*) then (output1 is *z*) (1)
20 if (speederror is *ns*) and (changeinerror is *pm*) then (output1 is *ps*) (1)

21 if (speederror is *ns*) and (changeinerror is *pb*) then (output1 is *pm*) (1)

22 if (speederror is *z*) and (changeinerror is *nb*) then (output1 is *nb*) (1)

23 if (speederror is *z*) and (changeinerror is *nm*) then (output1 is *nm*) (1)

24 if (speederror is *z*) and (changeinerror is *ns*) then (output1 is *ns*) (1)

25 if (speederror is *z*) and (changeinerror is *pb*) then (output1 is *pb*) (1)

26 if (speederror is *z*) and (changeinerror is *z*) then (output1 is *z*) (1)

27 if (speederror is *z*) and (changeinerror is *ps*) then (output1 is *ps*) (1)

28 if (speederror is *z*) and (changeinerror is *pm*) then (output1 is *pm*) (1)

29 if (speederror is *ps*) and (changeinerror is *nb*) then (output1 is *nm*) (1)

30 if (speederror is *ps*) and (changeinerror is *nm*) then (output1 is *ns*) (1)

31 if (speederror is *ps*) and (changeinerror is *ns*) then (output1 is *z*) (1)

32 if (speederror is *ps*) and (changeinerror is *z*) then (output1 is *ps*) (1)

33 if (speederror is *ps*) and (changeinerror is *ps*) then (output1 is *pm*) (1)

34 if (speederror is *ps*) and (changeinerror is *pm*) then (output1 is *pb*) (1)

35 if (speederror is *ps*) and (changeinerror is *pb*) then (output1 is *ps*) (1)

36 if (speederror is *pm*) and (changeinerror is *nb*) then (output1 is *ns*) (1)

37 if (speederror is *pm*) and (changeinerror is *nm*) then (output1 is *z*) (1)

38 if (speederror is *pm*) and (changeinerror is *ns*) then (output1 is *ps*) (1)

39 if (speederror is *pm*) and (changeinerror is *z*) then (output1 is *pm*) (1)

40 if (speederror is *pm*) and (changeinerror is *ps*) then (output1 is *pb*) (1)

41 if (speederror is *pm*) and (changeinerror is *pm*) then (output1 is *ps*) (1)

42 if (speederror is *pm*) and (changeinerror is *pb*) then (output1 is *pb*) (1)

43 if (speederror is *pb*) and (changeinerror is *nb*) then (output1 is *z*) (1)

44 if (speederror is *pb*) and (changeinerror is *nm*) then (output1 is *ps*) (1)

45 if (speederror is *pb*) and (changeinerror is *ns*) then (output1 is *pm*) (1)

46 if (speederror is *pb*) and (changeinerror is *z*) then (output1 is *pb*) (1)

47 if (speederror is *pb*) and (changeinerror is *ps*) then (output1 is *pb*) (1)

48 if (speederror is *pb*) and (changeinerror is *pm*) then (output1 is *pb*) (1)

49 if (speederror is *pb*) and (changeinerror is *pb*) then (output1 is *pb*) (1)

The above-mentioned rules are written in the form of a file (*.fis*) and then called upon in the developed Simulink block-diagram.

5.3 DEVELOPMENT OF THE SIMULINK MODEL

The developed Simulink model for the control of various parameters of the SCIM is shown in Fig. 5.3 and its development is similar to the one developed for the speed control using the PI method, except for incorporation of the fuzzy controller. The specifications of the SCIM used for simulation purposes have been described in Chapter 3 (table 3.1).

5.4 SIMULATION RESULTS AND DISCUSSIONS

In order to start the simulations using Matlab, the fuzzy rule set has to be invoked first from the command window. Initially, the fuzzy file where the rules are written with the incorporation of the Mamdani control strategy is opened in the Matlab command window, after which the fuzzy editor (FIS) dialogue box opens. The *.fis* file is imported using the command window from the source file and then opened in the fuzzy editor dialog box using the file open command. Once the file is opened, the Mamdani rules file gets activated as shown in Fig. 5.4.

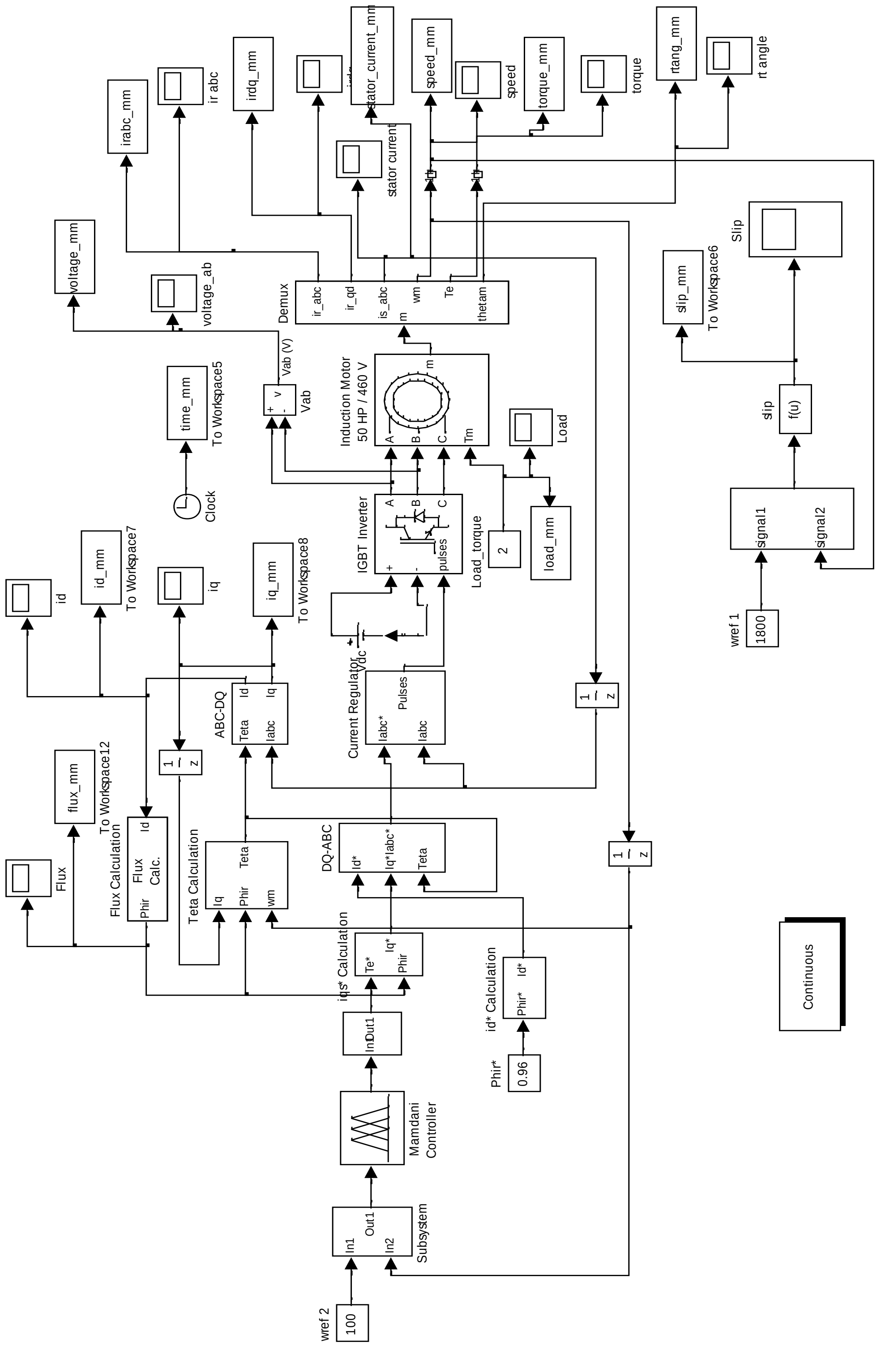
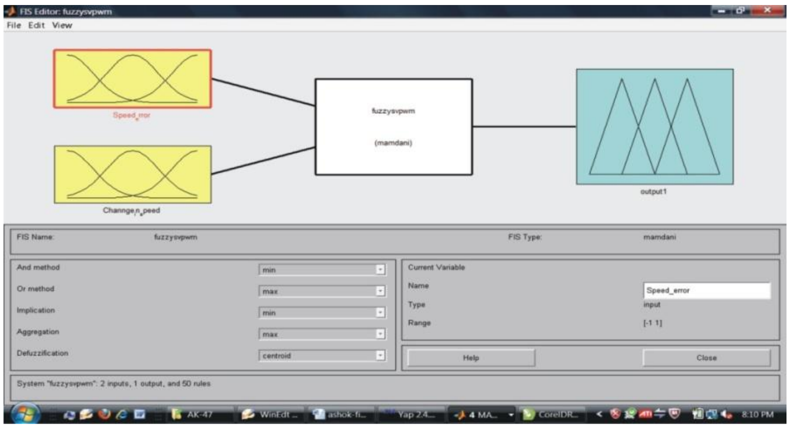


Fig. 5.3: Developed Simulink model for speed control using the M-FLC scheme

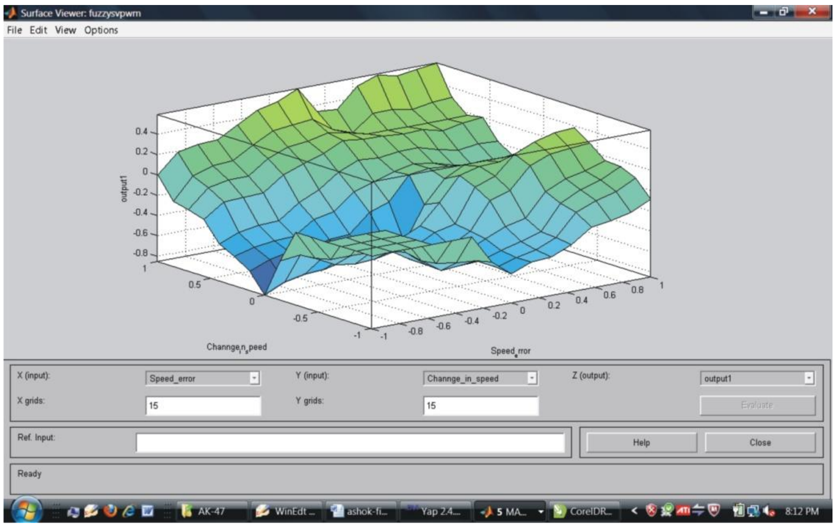
Furthermore, the data is exported to the workspace and the simulations are run for a specific amount of time (say 2 to 3 second). The fuzzy membership function editor is then obtained using the view membership command from the menu bar.



FIS Name: fuzzysvpwm
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: Speed_error
Type: input
Range: [-1 1]
System "fuzzysvpwm": 2 inputs, 1 output, and 50 rules

Fig. 5.4: 2 inputs and 1 output Fuzzy Editor

The written 49 fuzzy rules using Mamdani control strategy also can be viewed from the rule view command. The rule viewer for the 2 inputs and 1 output can be observed pictorially.



X (input): Speed_error
Y (input): Channge_in_speed
Z (output): output1
X grids: 15
Y grids: 15
Ref. Input:
Ready

Fig. 5.5: Surface plot for Speed error, change in error and output

The surface plot for the error in speed and change in error with the output is shown in the Fig. 5.5. The simulations are run for a period of 3 second in Matlab 7 with a reference speed of 100 rads / second $\left(\frac{100\times 60}{2\pi}\right)=955$ rpm and with a load torque of 2 *N-m*. After the simulation is run, the performance characteristics are observed on the respective scopes. The response curves of various parameters such as voltage, stator current, torque, speed, etc. are shown in the Figs. 5.6-5.17 respectively **[76]**.

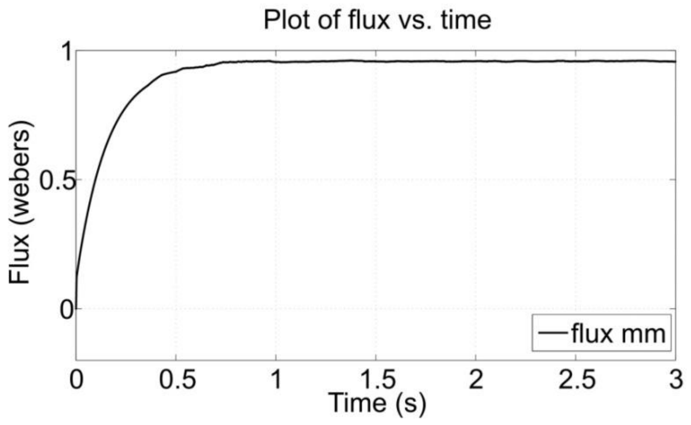


Fig. 5.6: Plot of flux vs. time using the M-FLC scheme

From the variation of flux with time as shown in Fig. 5.6, it can be observed that when the motor speed increases (during the transient period), more stator current is required to develop the requisite flux in the air gap. Hence, the flux also starts increasing during the transient period (0 to 0.9 second) exponentially.

Once the motor attains the set rated speed, the flux required to develop the torque almost remains constant after ≥ 0.9 second. Once the saturation of the flux takes place in the air gap, the variation of the load torque and speed will not disturb the flux curve. Hence, the IM will operate at a constant flux.

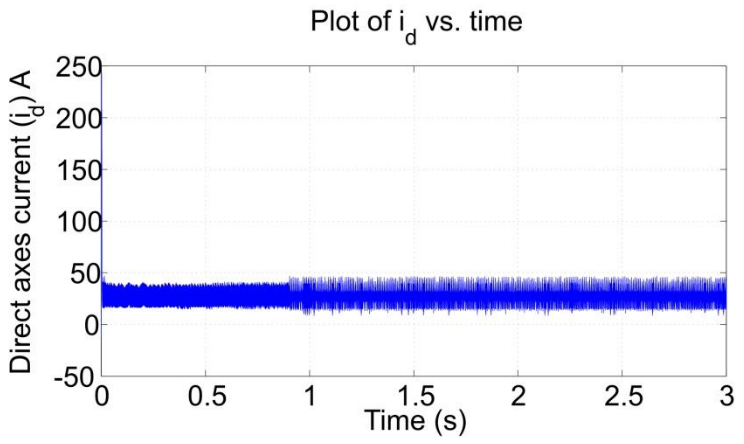


Fig. 5.7: Plot of i_d vs. time using the M-FLC Scheme

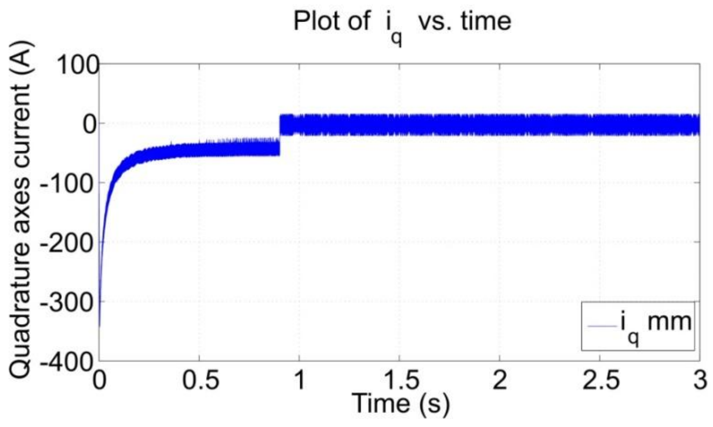


Fig. 5.8: plot of i_q vs. time using M-FLC scheme

The plots of the direct axes *(i_d)* and quadrature axes currents *(i_d)* versus time are shown in Figs. 5.7 and 5.8, respectively. It can be inferred that the machine reaches the set reference speed of **955 rpm** in a time interval of **0.9 second**.

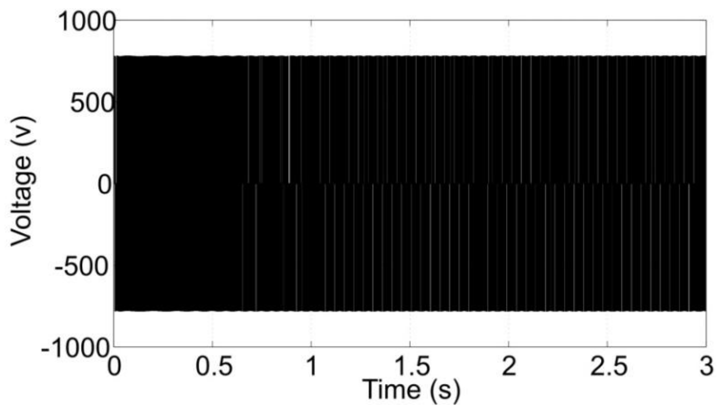


Fig. 5.9: Plot of voltage vs. time using the M-FLC scheme

The terminal voltage of the IM is shown in Fig. 5.9. The variation of 3Φ rotor currents ($i_{r\text{-}abc}$) with time is shown in Fig. 5.10. It can be inferred that at lower speeds, the slip is more, and hence the flux required to develop the suitable torque is also more.

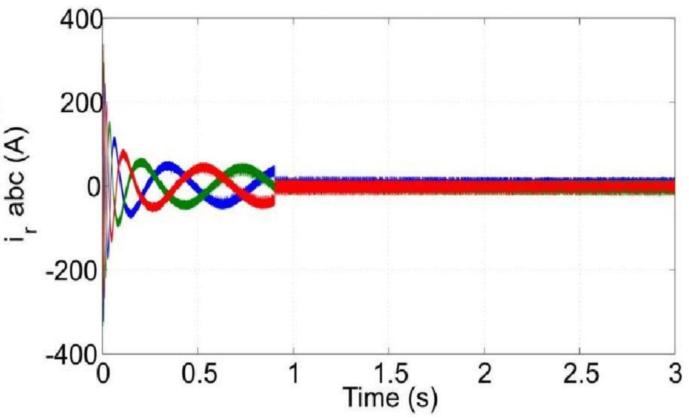


Fig. 5.10: Plot of Rotor current (*abc* frame) vs. time using M-FLC

Moreover, the torque required to reach the set speed is also more. Hence, the magnitude of the rotor currents will also be more during the transient periods (starting periods) of the IM.

After speed reaches the set value from zero, the *3Φ* rotor currents decrease exponentially and then stabilize, i.e., reach a constant value once the desired speed is reached. The 3*Φ* rotor currents ($i_{r\text{-}abc}$) are transformed to direct axes and quadrature axes currents using the *d*–*q* transformation techniques and the variation of the transformed currents with time is shown in Fig. 5.11. Here, only two phases of the currents can be observed in the characteristic curve. In this case, also, once the motor achieves the set speed at **0.9 second**, it requires a nominal current to drive the IM system.

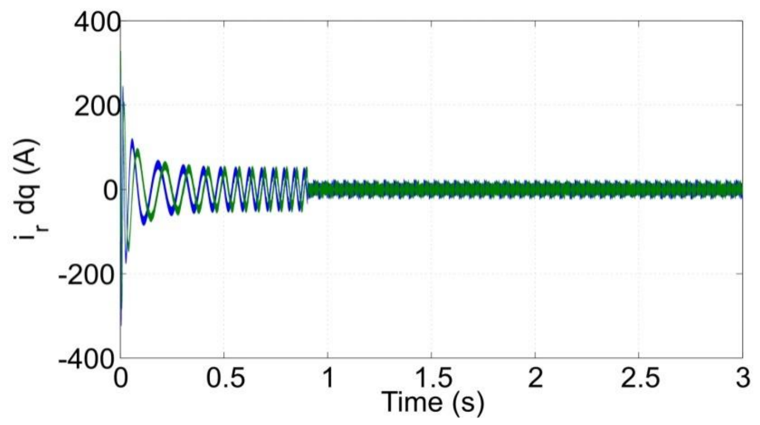


Fig. 5.11: Plot of rotor current (*d-q* frame) vs. time using M-FLC

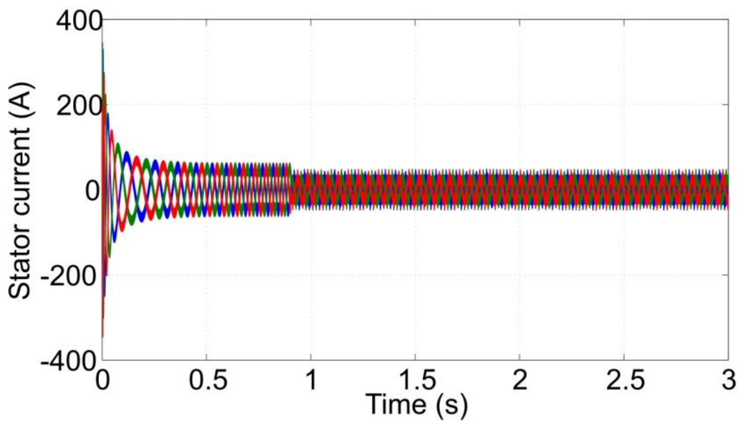


Fig. 5.12: Plot of 3-phase stator current vs. time using M-FLC

The variation of the *3Φ* stator currents *($i_{s\text{-}abc}$)* with time is shown in Fig. 5.12. It can be clearly observed from this figure, that at lower speeds, the slip is more, and hence the flux required to develop the suitable torque is also more. Moreover, the torque required to reach the set speed is also more. Hence, the magnitude of the stator currents will also be more during the transient periods (starting periods) of the induction motor. When the speed is reaches the set value from zero, the *3Φ* stator currents decrease exponentially. Once it attains the set speed at **0.9 second**, it requires a nominal stator current to drive the IM system. Stator current does not exhibit any overshoot or undershoot.

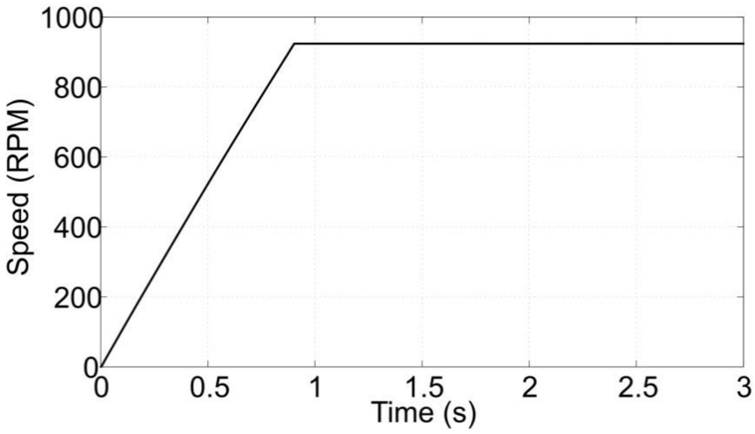


Fig. 5.13: Plot of speed vs. time using the M-FLC scheme

From the simulation results of the speed shown in Fig. 5.13 using the Mamdani control strategy, it was observed that the speed reaches its desired set value (becomes stable) at **0.9 second [76]**. They then reach the final steady-state value. The motor speed increases linearly up to the set speed of **955 rpm** in **0.9 second**.

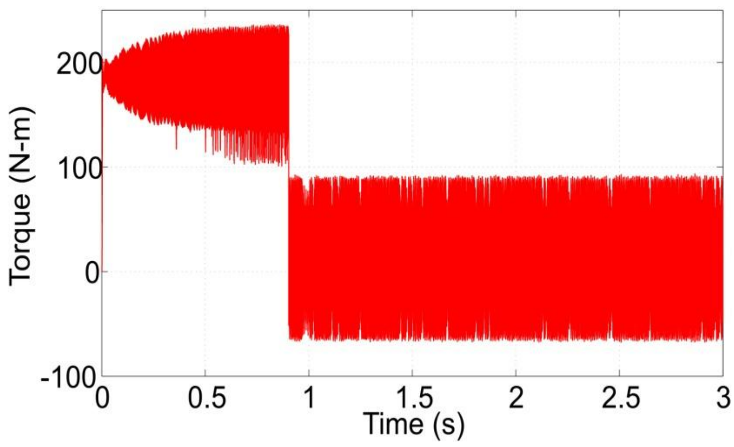


Fig. 5.14: Plot of torque vs. time using the M-FLC scheme

Torque characteristics for a set reference speed of **100 rad/second** (**955 rpm**) are shown in Fig. 5.14. From this figure, we can conclude that when the motor operates at lower speeds, the slip is more. Hence, the machine requires more torque to attain the set speed. Once the machine reaches the set speed of 955 rpm, the average torque of the machine becomes nearly zero, which is justified from the simulation result.

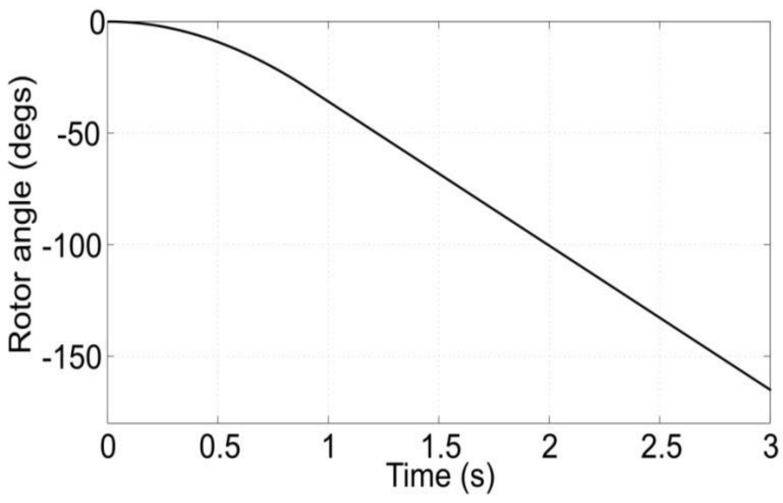


Fig. 5.15: Plot of rotor angle vs. time using the M-FLC

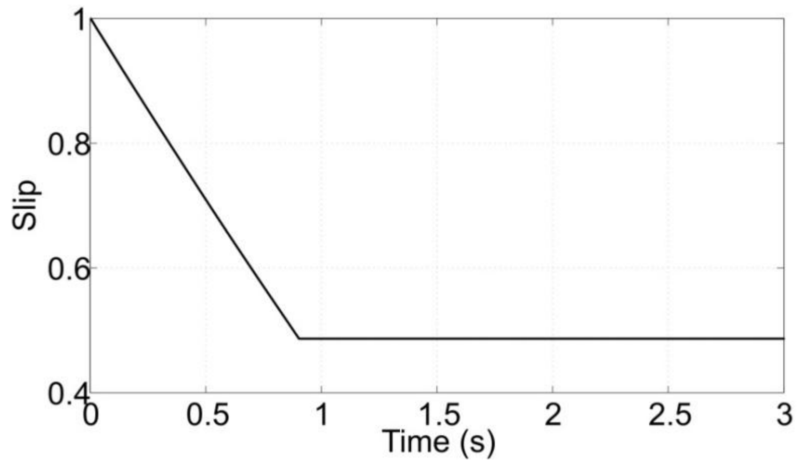


Fig. 5.16: Plot of slip vs. time using the M-FLC

The plot of rotor angle vs. time is shown in Fig. 5.15. It is clear that the rotor angle damping using the Mamdani-based FLC is more effective. Fig. 5.16 shows the variation of slip vs. time characteristics for a speed of 100 rad/second (955 rpm).

From this simulation result, we infer that the IM attains the set reference speed of 955 rpm in 0.9 second using the Mamdani-FLC scheme. At that instant, the slip is $\left(\frac{N_s - N}{N_s}\right) = \frac{1800 - 955}{1800} = 0.46$; this can be verified from the result shown. Note that the slip decreases from 1.0 to 0.46 linearly in a time span of just **0.9 second**.

The load torque is set at a constant value of 2 N-m throughout the process of simulation at the time of change in speed, as shown in Fig. 5.17.

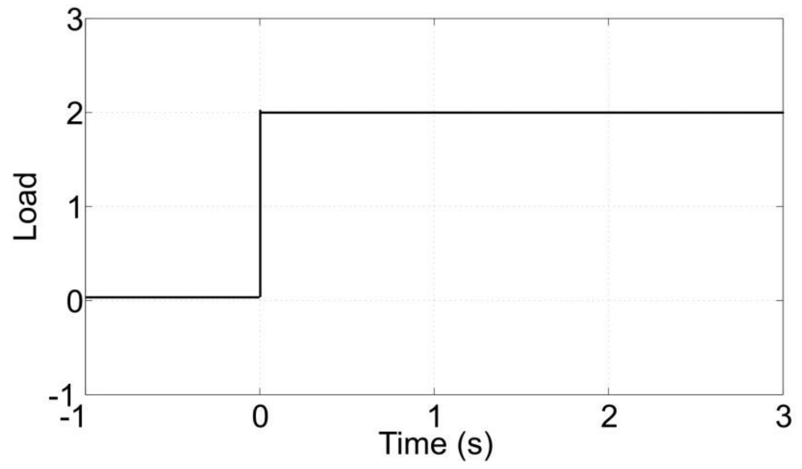


Fig. 5.17: Char. of load vs. time using the M-FLC

5.5 SUMMARY

The M-FLC based control scheme was developed in this chapter to control the speed of the IM in Matlab simulink. The Mamdani-based FLC provides control commands for generating gating signal. The inverter terminal voltage controlled by these gating signals, in turn control the speed of drive. This controller provides faster settling times, has very good dynamic response and good stabilization. The speed curve takes lesser time to reach the stable point compared to the PI method. The speed increases smoothly up to the set point (set speed of **100 rad/sec**) and then settles at **0.9 second**, thus showing the effectiveness of the developed method **[76]**. To attain better performance of the controller, a new type of controller, called the Takagi–Sugeno-based fuzzy logic controller, can be used to control the speed of the IM. This is depicted in the next chapter.