Chapter 7

Preliminary Analysis

7. Preliminary Analysis

For successful implementation of any control problems, a proper mathematical model is prime requirement. The only advantage of designing control systems with evolutionary algorithm is that it does not require complex and tedious mathematical manipulations by way of solving differential equations. In case of uncertain nonlinear systems these equations are normally too complex to handle and representing them into the exact mathematical model is almost impossible.

In case of evolutionary algorithms, mathematical models with reasonable complexity, also leads to desired performance because of their capability to learn in real time environment. We know that it is very difficult to model exact nonlinear model, the uncertainties due to external disturbances etc., which attracts the researchers toward the evolutionary algorithms.

7.1 Fuzzy Model Reference Learning Controller (FMRLC)

A "learning system" possesses the capability to improve its performance over time by interacting with its environment. A learning control system is designed so that its "learning controller" has the ability to improve the performance of the closed loop system by generating command inputs to the plant and utilizing feedback information from the plant. The "fuzzy model reference learning controller" (FMRLC) is a (direct) model reference adaptive controller [197]. The term "learning" is used as opposed to "adaptive" to distinguish it from the approach to the conventional model reference adaptive controller for linear systems with unknown plant parameters. In particular, the distinction is drawn since the FMRLC will tune and to some extent remember the values that it had tuned in the past, while the conventional approaches for linear systems simply continue to tune the controller parameters. Hence, for some applications when a properly designed FMRLC returns to a familiar operating condition, it will already know how to control for that condition. Many past conventional adaptive control techniques for linear systems would have to retune each time a new operating condition is encountered.

The functional block diagram for the FMRLC is shown in Figure 7.1[198]. It has four main parts: a. the plant, b. the fuzzy controller to be tuned, c. the reference model, and d. the

learning mechanism or an adaptation mechanism. Discrete time signals are used to explain the operation of the FMRLC for discrete time systems.

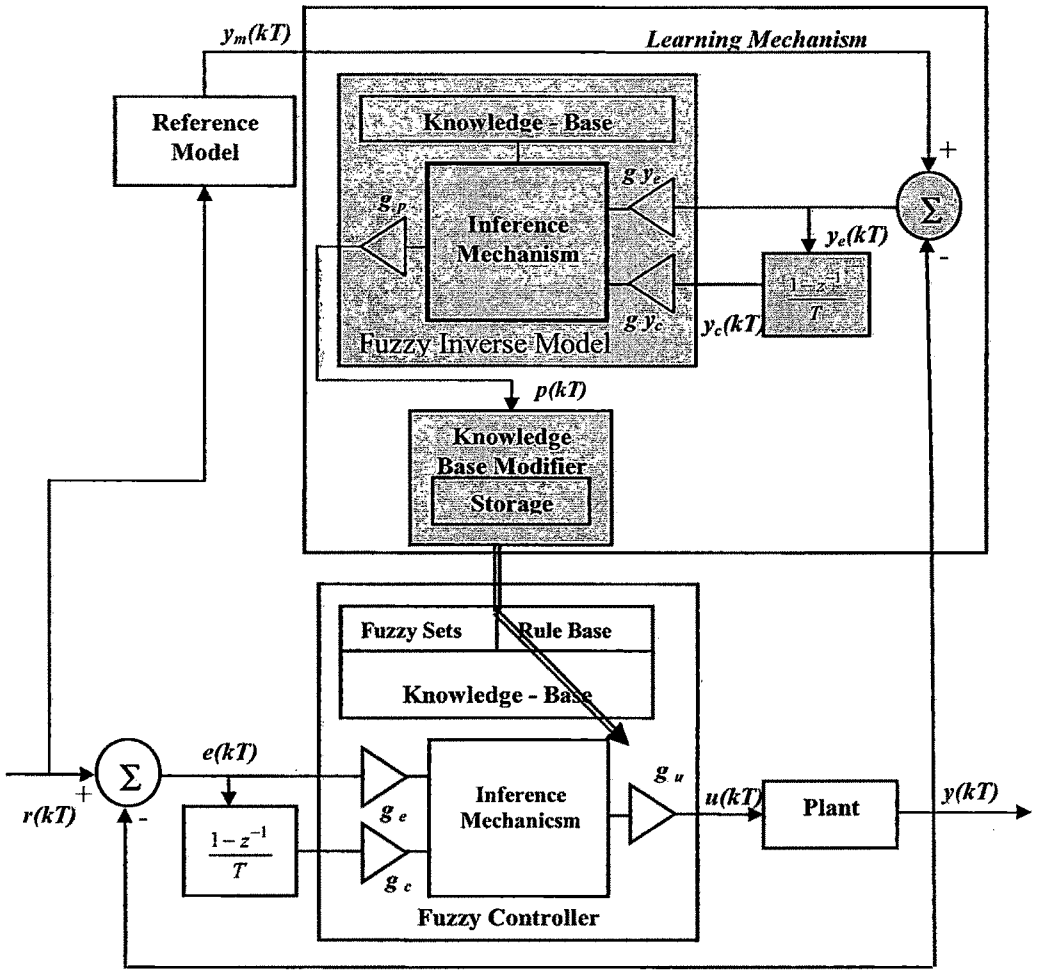

Figure 7.1 FMRLC Architecture

The FMRLC uses the learning mechanism to observe numerical data from a fuzzy control system (i.e., $r(kT)$ and $y(kT)$ where T is the sampling period). Using this numerical data, it characterizes the fuzzy control system's current performance and automatically synthesizes or adjusts the fuzzy controller so that some given performance objectives are met. These performance objectives are characterized via the reference model shown in Figure 7.1. The learning mechanism seeks to adjust the fuzzy controller so that the closed-loop system (the map from $r(kT)$ to $y(kT)$) acts like the given reference model (the map from $r(kT)$ to $y_m(kT)$). Basically, the fuzzy control system loop operates to make $y(kT)$ track $r(kT)$ by manipulating

$u(kT)$, while the upper-level adaptation control loop seeks to make the output of the plant $y(kT)$ track the output of the reference model $y_m(kT)$ by manipulating the fuzzy controller parameters.

Let us discuss each module of the FMRLC, for SISO system.

7.1.1 The Fuzzy Controller

The plant in Figure 7.1 has an input $u(kT)$ and output $y(kT)$. Most often the inputs to the fuzzy controller are generated via some function of the plant output $y(kT)$ and reference input $r(kT)$. For this, the inputs to the fuzzy controller are the error $e(kT)$ and change in error $c(kT)$, i.e. PD Fuzzy controller.

Where,

$$e(kT) = r(kT) - y(kT) \quad and$$

$$c(kT) = \frac{e(kT) - e((k-1)T)}{T} \qquad \ldots(7.1)$$

In the figure 7.1 we use scaling gains g_e, g_c, and g_u for the error $e(kT)$, change in error $c(kT)$, and controller output $u(kT)$, respectively. Selection of these gains is done in the heuristic way as follows: The gain g_e can be chosen so that the range of values that $e(kT)$ typically takes on will not make it so that its values will result in saturation of the corresponding outermost input membership functions. The gain g_c can be determined by experimenting with various inputs to the fuzzy control system, without the adaptation mechanism, to determine the normal range of values that $c(kT)$ will take on. Taking these range in consideration the gain g_c is chosen so that normally encountered values of $c(kT)$ will not result in saturation of the outermost input membership functions. g_u is also chosen the same way, so that the range of outputs that are possible is the maximum one possible yet still so that the input to the plant will not saturate. It is also possible to tune these gains when we tune the overall FMRLC.

Rule base

The rule base of the fuzzy controller has rules of the form

$$If\ \tilde{e}\ is\ \tilde{E}^j\ and\ \tilde{c}\ is\ \tilde{C}^l\ then\ \tilde{u}\ is\ \tilde{U}^m$$

where, $\tilde{e}$ and $\tilde{c}$ denote the linguistic variables associated with controller inputs $e(kT)$ and $c(kT)$, respectively, $\tilde{u}$ denotes the linguistic variable associated with the controller output u, $\tilde{E}^j$ and $\tilde{C}^l$

denote the j^{th} (l^{th}) linguistic value associated with $\tilde{e}$ ($\tilde{c}$), respectively, and $\tilde{U}^m$ denotes the consequent linguistic value associated with $\tilde{u}$.

Hence, as an example, one fuzzy control rule could be

If error is positive-large ***and*** *change-in-error is negative-small*

Then *plant-input is positive-big*

Standard triangular shaped membership functions are used for all the membership function of the inputs of universe of discourse as shown in the figure 7.2, with scaled horizontal axis for *c(kT)*. All the possible combinations of rules are used for the rule base.

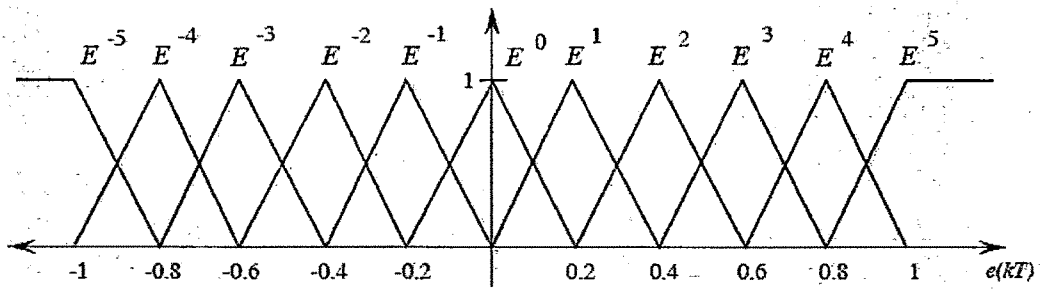

Figure 7.2: Membership function for input universe of discourse

Rule base Initialization

The input membership functions are defined to characterize the premises of the rules that define the various situations in which rules should be applied. The input membership functions are left constant and are not tuned by the FMRLC. The membership functions on the output universe of discourse are assumed to be unknown. They are what the FMRLC will automatically synthesize or tune. Hence, the FMRLC tries to fill in what actions ought to be taken for the various situations that are characterized by the premises. Hence, initial values for each of the output membership functions are required to be defined.

For example, for an output universe of discourse [−1, 1], triangular shaped membership functions with base widths of 0.4 and centers at zero can be taken. This choice represents that the fuzzy controller initially knows nothing about how to control the plant so it inputs $u = 0$ to the plant initially. Of course, one can often make a reasonable best guess at how to specify a fuzzy controller that is "more knowledgeable" than simply placing the output membership function centers at zero. For example, we could pick the initial fuzzy controller to be the best one that we

can design for the nominal plant. Notice, however, that this choice is not always the best one. Really, what you often want to choose is the fuzzy controller that is best for the operating condition that the plant will begin in (this may not be the nominal condition). Unfortunately, it is not always possible to pick such a controller since you may not be able to measure the operating condition of the plant, so making a best guess or simply placing the membership function centers at zero are common choices.

Finally, minimum or product can be used to represent the conjunction in the premise and the implication. For the application on hand, minimum is used to represent the conjunction. Standard center-of-gravity method is used as defuzzification technique. Other methods can also be employed.

Learning, Memorization, and Controller Input Choice

The inputs to the fuzzy controller can be either direct or preprocessed. Sometimes the same guidelines that are used for the choice of the inputs for a non-adaptive fuzzy controller are useful for the FMRLC. It has been observed in many of the cases that time where it is advantageous to replace part of a conventional controller with a fuzzy controller and use the FMRLC to tune the same. In such cases the complex preprocessing of inputs to the fuzzy controller is achieved via a conventional controller. Sometimes there is also the need for post-processing of the fuzzy controller outputs. Some time, it may happen that choice of the inputs itself involves the issues related to the learning dynamics of the FMRLC. As the FMRLC operates, the learning mechanism will tune the fuzzy controller's output membership functions i.e. for each different combination of $e(kT)$ and $c(kT)$ inputs, it will try to learn what the best control actions are.

In general, there is a close connection between what inputs are provided to the controller and the controller's ability to learn to control the plant for different reference inputs and plant operating conditions. The FMRLC is designed such that it will learn and remember different fuzzy controllers for all the different plant operating conditions and reference inputs; hence, the fuzzy controller needs information about these. Often, however, it is difficult to measure the operating condition of the plant, so the FMRLC does not know exactly what operating condition it is learning the controller for. Moreover, it then does not know exactly when it has returned to an operating condition. Clearly, then, if the fuzzy controller has better information about the

plant's operating conditions, the FMRLC will be able to learn and apply better control actions. If it does not have good information, it will continually adapt, but it will not properly remember.

For instance, for some plants $e(kT)$ and $c(kT)$ may only grossly characterize the operating conditions of the plant. In this situation the FMRLC will not be able to learn different controllers for different operating conditions; it will use its limited information about the operating condition and continually adapt to search for the best controller. It degrades from a learning system to an adaptive system that will not properly remember the control actions.

Generally the inputs to the fuzzy controller specify what conditions are; we need to learn different controllers. A competing objective is, however, to keep the number of fuzzy controller inputs low due to concerns about computational complexity. In fact, to help with computational complexity, multiple fuzzy controllers are used with fewer inputs to each of them rather than one fuzzy controller with many inputs; then we can sum the outputs of the individual controllers.

7.1.2 The Reference Model

Next, important is to decide on what to choose for the reference model that quantifies the desired performance. Basically, one has to specify a reasonable desirable performance. Certain characteristics of real-world plants place practical constraints on what performance can be achieved. It is not always easy to pick a good reference model and its choice is purely based on the application on hand. In general, the reference model may be discrete or continuous time, linear or nonlinear, time-invariant or time-varying, and so on. For example, suppose that we would like to have the response track the continuous time model

$$G(s) = \frac{1}{s+1} \qquad \ldots(7.2)$$

For discrete time implementation of same with T=0.1 and using bilinear transformation to find discrete equivalent to the continuous time transfer function *G(s)*. Bilinear transformation is given by

$$s = \frac{2}{T}\frac{z-1}{z+1} \qquad \ldots(7.3)$$

Substituting eq. 7.3 into eq. 7.2 ...

$$\frac{y_m(z)}{R(z)} = H(z) = \frac{\frac{1}{21}(z+1)}{z-\frac{19}{21}} \qquad \ldots(7.4)$$

Where, $y_m(z)$ and $R(z)$ are the Z-transform of $y_m(kT)$ and $r(kT)$, respectively. Hence, for discrete time implementation of the same will be

$$y_m(kT+T) = \frac{19}{21}y_m(kT) + \frac{1}{21}r(kT+T) + \frac{1}{21}r(kT) \qquad \ldots(7.5)$$

This choice would then represent that output $y(kT)$ is to track a smooth, stable, first-order type response of $y_m(kT)$. A similar approach can be used to, for example, track a second-order system with a specified damping ratio ζ and undamped natural frequency ω_n.

The performance of the overall system is computed with respect to the reference model by the learning mechanism by generating an error signal

$$y_e(kT) = y_m(kT) - y(kT). \qquad \ldots(7.6)$$

Given that the reference model characterizes design criteria such as rise-time and overshoot and the input to the reference model is the reference input $r(kT)$, the desired performance of the controlled process is met if the learning mechanism forces $y_e(kT)$ to remain very small for all time no matter what the reference input is or what plant parameter variations occur. Hence, the error $y_e(kT)$ provides a characterization of the extent to which the desired performance is met at time kT. If the performance is met, i.e. $y_e(kT)$ is small, then the learning mechanism will not make significant modifications to the fuzzy controller. On the other hand if $y_e(kT)$ is big, the desired performance is not achieved and the learning mechanism must adjust the fuzzy controller.

7.1.3 Learning Mechanism

The learning mechanism tunes the rule-base of the direct fuzzy controller so that the closed-loop system behaves like the reference model. These rule-base modifications are made by observing data from the controlled process, the reference model, and the fuzzy controller. The learning mechanism consists of two parts: a "fuzzy inverse model" and a "knowledge-base modifier." The fuzzy inverse model performs the function of mapping $y_e(kT)$, representing the

deviation from the desired behavior, to changes in the process inputs $p(kT)$ that are necessary to force $y_e(kT)$ to zero. The knowledge-base modifier performs the function of modifying the fuzzy controller's rule-base to affect the needed changes in the process inputs.

Fuzzy Inverse Model

A fuzzy system is used to map $y_e(kT)$ and possibly functions of $y_e(kT)$ such as $y_c(kT)=\frac{1}{T}y_e(kT)-y_e(kT-T)$ to the necessary changes in the process inputs $p(kT)$. This fuzzy system is sometimes called the "fuzzy inverse model" since information about the plant inverse dynamics is used in its specification. Similar to the fuzzy controller, the fuzzy inverse model shown in Figure 7.1 contains scaling gains, denoted as g_{ye}, g_{yc}, and g_p. Selection of these scaling gains is also important.

Given that $g_{ye}y_e$ and $g_{yc}y_c$ are inputs to the fuzzy inverse model, the rule-base for the fuzzy inverse model contains rules of the form

$$If\ \tilde{y}_e\ is\ \tilde{Y}_e^j\ and\ \tilde{y}_c\ is\ \tilde{Y}_c^l\ then\ \tilde{p}\ is\ \tilde{P}^m$$

Where, $\tilde{Y}_e^j$ and $\tilde{Y}_c^l$ denote linguistic values and $\tilde{P}^m$ denotes the linguistic value associated with the m^{th} output fuzzy set. For the application under consideration, the membership functions for the input universes of discourse are as shown in Figure 7.2, symmetric triangular-shaped membership functions for the output universes of discourse, minimum to represent the premise and implication, and COG defuzzification.

Knowledge-Base Modifier

Given the information about the necessary changes in the input, which are represented by $p(kT)$, to force the error y_e to zero, the knowledge-base modifier changes the rule-base of the fuzzy controller so that the previously applied control action will be modified by the amount $p(kT)$. Let the previously computed control action $u(kT - T)$, and assume that it contributed to the present good or bad system performance i.e., it resulted in the value of $y(kT)$ such that it did not match $y_m(kT)$. $e(kT - T)$ and $c(kT - T)$ are the error and change in error that were input to the fuzzy controller at that time. By modifying the fuzzy controller's knowledge-base, the fuzzy controller will now force to produce a desired output $u(kT - T) + p(kT)$, which should have put in at time $kT - T$ to make $y_e(kT)$ smaller. The next time when similar values for the error and change in error

occur, the input to the plant will be one that will reduce the error between the reference model and plant output.

Learning, Memorization, and Inverse Model Input Choice

The changes made to the rule-base are only *local* ones, i.e. the entire rule-base is not updated at every time step, just the rules that needed to be updated to force $y_e(kT)$ to zero. This local learning is important since it allows the changes that were made in the past to be remembered by the fuzzy controller. The type and amount of memory depends critically on the inputs to the fuzzy controller. Different parts of the rule-base are "filled in" based on different operating conditions for the system and when one area of the rule-base is updated, other rules are not affected. Hence, if the appropriate inputs are provided to the fuzzy controller so that it can distinguish between the situations in which it should behave differently, the controller adapts to new situations and also remembers how it has adapted to past situations.

Just as the choice of inputs to the fuzzy controller has a fundamental impact on learning and memorization, so does the choice of inputs to the inverse model e.g. one may want to choose the inputs to the inverse model so that it will adapt differently in different operating conditions, to adapt more slowly than in another, the direction of adjustment of the output membership function centers may be the opposite of that in another etc. If there are multiple fuzzy controllers, it is preferable to have multiple inverse models to adjust them, which may help with computational complexity as use of fewer inputs per fuzzy inverse model.

The choice of inputs to the fuzzy inverse model shown in Figure 7.1 indicates that different errors and error rates between the reference model and plant output are required to adapt differently. The inverse model may be designed so that, for example, if the error is small, then the adjustments to the fuzzy controller should be small, and if the error is small but the rate of error increase is high, then the adjustments should be larger. It is rules such as these that are loaded into the fuzzy inverse model.

Let us now discuss the mathematical models of the application on hand, namely Ship and Aircraft dynamics, followed by their design implementations.

7.2 Ship Dynamics

For proper mathematical representation of ship dynamics, we need to fix the reference frame to be used. One can use either earth reference frame or body reference frame. In case of earth reference frame, x & y coordinates are defined with reference to true earth ellipsoid, where x axis points towards the North and y axis towards the East. Body reference frame is moving coordinate system but it is fixed to the ship. The origin coincides with the centre of gravity. The axes are defined as: x axis – the longitudinal axis, directed from aft to fore, y axis – the transversal, directed towards the starboard.

The position of the ship, defined by GPS, is normally described by earth reference frame but its motion is generally described by a body reference frame, figure 7.3. Ship dynamics are obtained by applying Newton's laws of motion to the ship. For very large ships, the motion in the vertical plane may be neglected since the "bobbing" or "bouncing" effects of the ship are small for large vessels.

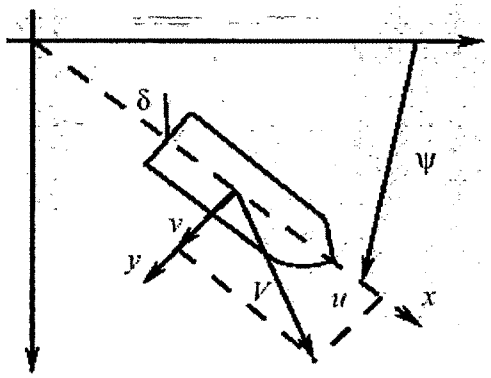

Figure 7.3: Cargo Ship Coordinate System

A Simple model of ship's motion is given by

$$\dddot{\psi}(t)+\left(\frac{1}{\tau_1}+\frac{1}{\tau_2}\right)\ddot{\psi}(t)+\left(\frac{1}{\tau_1\tau_2}\right)\dot{\psi}(t)=\frac{K}{\tau_1\tau_2}\left(\tau_3\,\dot{\delta}(t)+\delta(t)\right) \qquad \ldots(7.7)$$

Where, ψ is the heading of the ship and δ is the rudder angle.

Assuming zero initial conditions, we can write transfer function of the system as

$$\frac{\psi(s)}{\delta(s)} = \frac{K \cdot (s\tau_3 + 1)}{s \cdot (s\tau_1 + 1) \cdot (s\tau_2 + 1)} \qquad \ldots(7.8)$$

Where, K, $\tau 1$, $\tau 2$, and $\tau 3$ are parameters that are a function of the ship's constant forward velocity u and its length l.

In particular,

$$K = K_0\left(\frac{u}{l}\right) \quad and \quad \tau_i = \tau_{i0}\left(\frac{u}{l}\right) \; i = 1,2,3 \qquad \ldots(7.9)$$

In normal steering, a ship often makes only small deviations from a straight-line path. Therefore, the model in Equation (7.7) is obtained by linearizing the equations of motion around the zero rudder angle ($\delta = 0$). As a result, the rudder angle should not exceed approximately 5 degrees; otherwise the model will be inaccurate. For our purposes, we need a model suited for rudder angles that are larger than 5 degrees; hence, we use the model proposed in [208].

The extended model of the ship's motion is given by

$$\dddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\tau_2}\right)H(\dot{\psi}(t)) = \frac{K}{\tau_1\tau_2}\left(\tau_3\dot{\delta}(t) + \delta(t)\right) \qquad \ldots(7.10)$$

Where, $H(\dot{\psi}(t))$ is a nonlinear function of $\dot{\psi}(t)$. The function $H(\dot{\psi}(t))$ can either be derived from the experiment or can be approximated to any n^{th} order equation, while keeping in mind the relationship between δ and $\dot{\psi}(t)$ in steady state such that $\dddot{\psi} = \ddot{\psi} = \dot{\delta} = 0$. Higher the order, performance is better but this will be at the cost of computation time, as far as implementation with evolutionary algorithm is concerned. To be reasonable, $H(\dot{\psi}(t))$ is approximated as

$$H(\dot{\psi}(t)) = \bar{a} \cdot \dot{\psi}^3 + \bar{b} \cdot \dot{\psi}. \qquad \ldots(7.11)$$

Where, $\bar{a}$ and $\bar{b}$ are real-valued constants such that $\bar{a}$ is always positive.

While evaluating the controllers, nonlinear model is used for simulation. Note that to do this the *nth*-order nonlinear ordinary differential equations representing the ship are converted to n first-order ordinary differential equations; for convenience, let

$$a = \left(\frac{1}{\tau_1} + \frac{1}{\tau_2} \right)$$

$$b = \left(\frac{1}{\tau_1 \tau_2} \right)$$

$$c = \frac{K \cdot \tau_3}{\tau_1 \tau_2}; \text{ and} \qquad \ldots(7.12)$$

$$d = \frac{K}{\tau_1 \tau_2}$$

The model is required to be presented in the state space form given as

$$\begin{aligned} \dot{x}(t) &= F(x(t), \delta(t)) \\ y(t) &= G(x(t), \delta(t)) \end{aligned} \qquad \ldots(7.13)$$

Where $x(t) = [x_1(t), x_2(t), x_3(t)]^T$ and

$F = [F_1, F_2, F_3]^T$ for use in nonlinear simulation program.

We need to choose $\dot{x}_i(t)$ so that F_i depend only on $x_i(t)$ and δ for *i=1,2,3.*

We have

$$\dddot{\psi}(t) = -a \cdot \ddot{\psi}(t) - b \cdot H(\dot{\psi}(t)) + c \cdot \dot{\delta}(t) + d \cdot \delta(t) \qquad \ldots(7.14)$$

Taking $\dot{x}_3(t) = \dddot{\psi}(t) - c \cdot \dot{\delta}(t)$

so that F_3 will not depend on $c \cdot \dot{\delta}(t)$ and $x_3(t) = \ddot{\psi}(t) - c \cdot \delta(t)$

Take $\dot{x}_2(t) = \ddot{\psi}(t)$

so that $x_2(t) = \dot{\psi}(t)$.

Finally take $x_1(t) = \psi(t)$.

This gives ...

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) = F_1(x(t),\delta(t)) \\ \dot{x}_2(t) &= x_3(t) + c\dot{\delta}(t) = F_2(x(t),\delta(t)) \\ \dot{x}_3(t) &= -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + d\dot{\delta}(t)\end{aligned} \qquad \ldots(7.15)$$

But,

$$\begin{aligned}&\ddot{\psi}(t) = x_3(t) + c\delta(t), \\ &\dot{\psi}(t) = x_2(t) \text{ and} \\ &H(x_2) = x_2^3(t) + x_2(t)\end{aligned} \qquad \ldots(7.16)$$

So,

$$\dot{x}_3(t) = -a(x_3(t) + c\delta(t)) - b(x_2^3(t) + x_2(t)) + d\dot{\delta}(t) = F_3(x(t),\delta(t)) \qquad \ldots(7.17)$$

These are the desired equations for the simulation. For discrete time implementation, we have to simply discretize the differential equations given above. The initial conditions used for the simulation are ...

$$\psi(0) = \dot{\psi}(0) = \ddot{\psi}(0) = 0$$

which implies that

$$\begin{aligned}&x_1(0) = x_2(0) = 0 \text{ and} \\ &x_3(0) = \ddot{\psi}(0) - c\dot{\delta}(0) \\ &\qquad = -c\dot{\delta}(0)\end{aligned}$$

7.3 Aircraft Dynamics

The equations of motion relate the forces and moments to the accelerations for the three translational and three rotational directions. For a rigid body referenced to a body fixed frame, the nonlinear equations of motion are [207]: (Refer Figure 7.4)

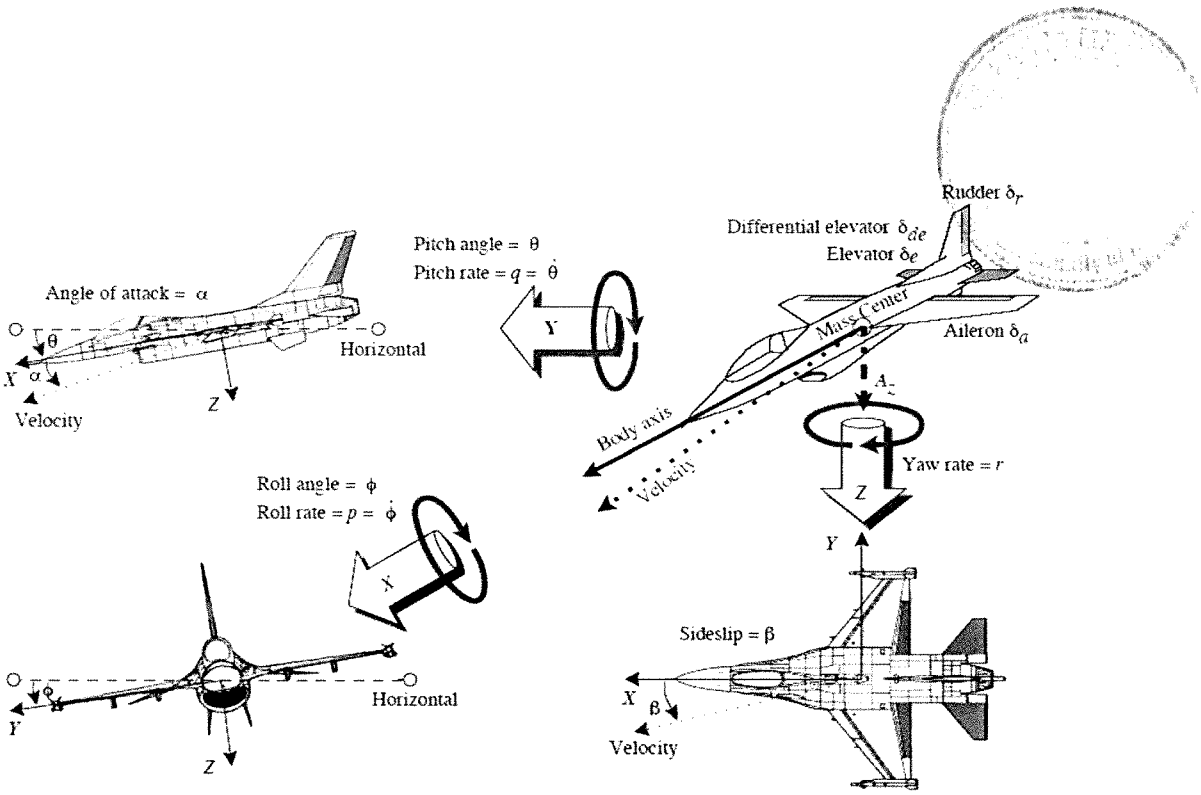

Figure 7.4: F-16 Aircraft

$$\dot{u} = rv - qw - g\sin\theta + \frac{\bar{q}S}{m}C_X + \frac{T}{m}$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi + \frac{\bar{q}S}{m}C_Y$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + \frac{\bar{q}S}{m}C_Z$$

$$\dot{p}I_X - \dot{r}I_{XZ} = pqI_{XZ} - qr(I_Z - I_Y) + \bar{q}S_bC_l \qquad \ldots(7.18)$$

$$\dot{q}I_Y = pr(I_Z - I_X) - (p^2 - r^2)I_{XZ} + \bar{q}S_{\bar{c}}C_m - rH_{eng}$$

$$\dot{r}I_Z - \dot{p}I_{XZ} = pq(I_X - I_Y) - qrI_{XZ} \_ \bar{q}S_bC_m + qH_{eng}$$

The aerodynamic coefficients in the equations of motion are written as a sum of contributing force and moment coefficients, some related to the control surfaces, others to the basic aerodynamic properties of the aircraft itself.

Force Coefficients are given as...

$$C_X = C_{X_0}(\alpha,\delta_e) + \left(\frac{q\bar{c}}{2V_T}\right)C_{X_q}$$

$$C_Y = -0.02\beta + 0.021\left(\frac{\delta_a}{20}\right) + 0.086\left(\frac{\delta_r}{30}\right) + \left(\frac{b}{2V_T}\right)\left(C_{X_p}(\alpha)p + C_{Y_r}(\alpha)r\right)$$

$$C_Z = C_{Z_0}(\alpha)\left[1-\left(\frac{\beta\pi}{180}\right)^2\right] - 0.019\left(\frac{\delta_e}{25}\right) + \left(\frac{q\bar{c}}{2V_T}\right)C_{Z_q}(\alpha)$$

...(7.19)

Where as Moment Coefficients are ...

$$C_l = \begin{aligned} & C_{l_0}(\alpha,\beta) + \Delta C_{l,\delta_a=20\deg}\left(\frac{\delta_a}{20}\right)\Delta C_{l,\delta_r=30\deg}\left(\frac{\delta_r}{30}\right) \\ & + \left(\frac{b}{2V_T}\right)\left(C_{l_p}(\alpha)p + C_{l_r}(\alpha)r\right) \end{aligned}$$

$$C_m = C_{m_0}(\alpha,\delta_e) + \left(\frac{q\bar{c}}{2V_T}\right)C_{m_q}(\alpha) + \left(x_{c.g.ref} - x_{c.g}\right)C_Z$$

$$C_n = \begin{aligned} & C_{n_0}(\alpha,\beta) + \Delta C_{n,\delta_a=20\deg}\left(\frac{\delta_a}{20}\right)\Delta C_{n,\delta_r=30\deg}\left(\frac{\delta_r}{30}\right) \\ & + \left(\frac{b}{2V_T}\right)\left(C_{n_p}(\alpha)p + C_{n_r}(\alpha)r\right) - \left(\frac{\bar{c}}{b}\right)\left(x_{c.g.ref} - x_{c.g}\right)C_Y \end{aligned}$$

...(7.20)

The engine model for the F-16 consists of three parts. The first part is the relationship between the throttle setting δ_T and the commanded power level P_c, which is a piecewise linear function taking into account the military power level:

$$P_c(\delta_T) = \begin{cases} 64.96 \cdot \delta_T & \text{if } \delta_T \le 0.77 \\ 217.38 \cdot \delta_T - 117.38 & \text{if } \delta_T > 0.77 \end{cases} \quad \ldots(7.21)$$

The second part of the model is the relationship between the commanded power level P_c and the actual power level P_a. This relationship is characterized by the engine power lag τ_{eng}:

$$\dot{P}_a = \frac{1}{\tau_{eng}(P_c - P_a)}$$

$$Here, P_c = \begin{cases} P_c & if\ P_c \geq 50\ and\ P_a \geq 50 \\ 60 & if\ P_c \leq 50\ and\ P_a < 50 \\ 40 & if\ P_c < 50\ and\ P_a \geq 50 \\ P_c & if\ P_c < 50\ and\ P_a < 50 \end{cases} \quad \ldots(7.22)$$

The engine power lag τ_{eng} is itself a function of the commanded power level and the actual power level:

$$\frac{1}{\tau_{eng}} = \begin{cases} 5.0 & if\ P_c \geq 50\ and\ P_a \geq 50 \\ \frac{1}{\tau_{eng}}^* & if\ P_c \leq 50\ and\ P_a < 50 \\ 5.0 & if\ P_c < 50\ and\ P_a \geq 50 \\ \frac{1}{\tau_{eng}}^* & if\ P_c < 50\ and\ P_a < 50 \end{cases}$$

$$\frac{1}{\tau_{eng}}^* = \begin{cases} 1.0 & if\ (P_c - P_a) \leq 25 \\ 0.1 & if\ (P_c - P_a) \geq 50 \\ 1.9 - 0.036(P_c - P_a) & if\ 25 < (P_c - P_a) < 50 \end{cases} \quad \ldots(7.23)$$

The third and final part of the engine model is the relationship between the actual power level P_a and the thrust T:

$$T = \begin{cases} T_{idle} + (T_{mil} - T_{idle})\left(\frac{P_a}{50}\right) & if\ P_a < 50 \\ T_{mil} + (T_{\max} - T_{mil})\left(\frac{P_a - 50}{50}\right) & if\ P_a \geq 50 \end{cases} \quad \ldots(7.24)$$

In the complete model of the F-16 there are thirteen states, six for position and attitude, six for their derivatives and one for the power level and four control inputs. But the for the application of Fault tolerant aircraft, the F-16 aircraft model used is further simplified based on a set of five linear perturbation models, extracted from a nonlinear model discussed above at the five different operating conditions (A_i, B_i, C_i, D_i), *where* $i \in \{1,2,3,4,5\}$[209,210], which takes into account the six states, three related to position and three of their derivatives, and all four inputs. The state space representation of the same is given as...

$$\begin{aligned} \dot{\bar{x}} &= A_i \bar{x} + B_i \bar{u} \\ \bar{y} &= C_i \bar{x} + D_i \bar{u} \end{aligned} \quad \ldots(7.25)$$

Where the state vector $\bar{x}$, input vector $\bar{u}$ and output vector $\bar{y}$ are defined as ...

- Inputs $\bar{u} = [\delta_e, \delta_{de}, \delta_a, \delta_r]^T$,
  - δ_e = elevator deflection (in degrees)
  - δ_{de} = differential elevator deflection (in degrees)
  - δ_a = aileron deflection (in degrees)
  - δ_r = rudder deflection (in degrees)
- System state $\bar{x} = [\alpha, q, \phi, \beta, p, r]^T$.
  - α = angle of attack (in degrees)
  - q = body axis pitch rate (in degrees/second)
  - ϕ = Euler roll angle (in degrees)
  - β = sideslip angle (in degrees)
  - p = body axis roll rate (in degrees/second)
  - r = body axis yaw rate (in degrees/second)
- Output $\bar{y} = [\bar{x}^T, A_z]^T$, where A_z is normal acceleration in g.

The nominal control laws for the aircraft are for the lateral channel as well as for the longitudinal channel. The inputs to the controller are pilot commands and the system feedback signal. Pilot commands for the longitudinal channel is desired pitch A_{zd} and for lateral channel are desired roll rate p_d & desired side slip β_d. The controller gains for both the channels are function of different dynamic pressures $\bar{q}$. Assuming constant speed and altitude of aircraft this can be assumed to be as *499.24 psf. (23.9 KN/sqm.).*

7.4 Helicopter Dynamics

A helicopter is mainly controlled by three operating controls. Those controls are throttle, the collective pitch controller and the cyclic pitch controller. The collective pitch angle of a rotor blade is the angle between the chord line and a reference plane determined by the rotor hub or the plane of rotation, refer Figure 7.5. The cyclic pitch angle is between the rotor disk and the air speed caused by tilting the rotor disk either up (positive) or down (negative), refer Figure 7.6. [211] The throttle main purpose is to control the angular speed of the main rotor. A constant angular speed is assumed for the implementation.

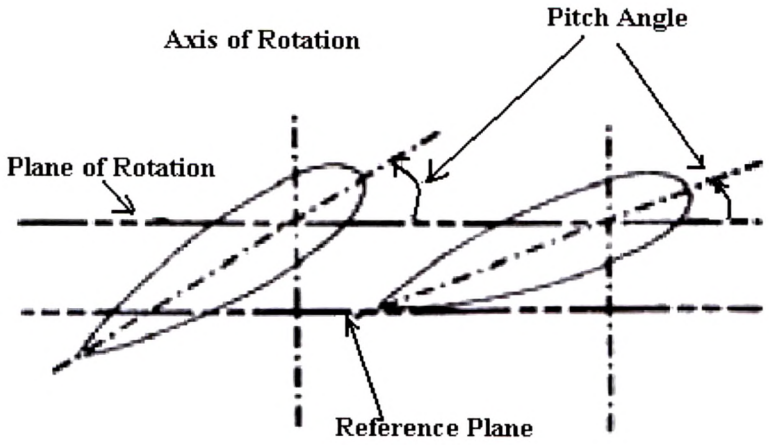

Figure 7.5: Collective Pitch angle of Rotor Blade

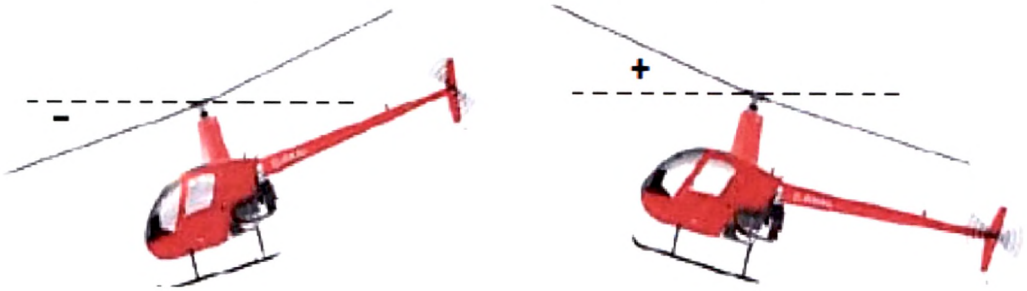

Figure 7.6: Cyclic pitch angle.

The collective pitch control angles all blades equally and simultaneously and allows the aircraft to raise vertically, figure 7.7(a). The cyclic pitch control allows each blade to be angled individually and allows the aircraft to move forward or backward, nose upward or downward, and roll from side to side, figure 7.7(b). A tail rotor is used to maintain yaw control and counteract the torque effect. By changing the pitch of the tail rotor's blades, this rotor will produce a side force that turns the helicopter nose left or right, figure 7.7(c).

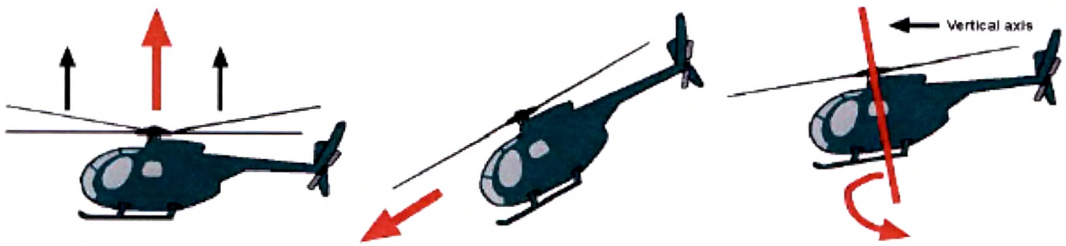

Figure 7.7 - (a) Collective Pitch, (b) Cyclic Pitch and (c) Tail collective Pitch

There is also a coupling effect when using the cyclic pitch due to the angled lift that consists of vertical and horizontal force components. In this model, the helicopter will be controlled by entering four inputs: main rotor collective pitch angle, tail rotor collective pitch angle, longitudinal cyclic pitch angle and lateral cyclic pitch angle.

Considering the helicopter equations of motion in nonlinear form given by:

$$\dot{x} = F(x,u,t) \qquad \ldots(7.26)$$

In 6 degree of freedom form, the motion states and controls are:

$$\begin{aligned} x &= \{u, w, q, \theta, v, p, \phi, r, \psi\} \\ u &= \{\theta_0, \theta_{ls}, \theta_{lc}, \theta_{0T}\} \end{aligned} \qquad \ldots(7.27)$$

The basic equation of motion for the helicopter is given in equations

$$\begin{aligned} \dot{u} &= (rv - qw) + \frac{X}{M_a} - g\sin\theta \\ \dot{v} &= (pw - ru) + \frac{Y}{M_a} + g\sin\phi\cos\theta \\ \dot{w} &= (qu - pv) + \frac{Z}{M_a} + g\cos\phi\cos\theta \\ I_{XX}\dot{p} &= (I_{YY} - I_{ZZ})qr + I_{XZ}(\dot{r} + pq) + L \\ I_{YY}\dot{q} &= (I_{ZZ} - I_{XX})rp + I_{XZ}(r^2 - p^2) + M \\ I_{ZZ}\dot{r} &= (I_{XX} - I_{YY})pq - I_{XZ}(\dot{p} - qr) + N \\ \dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta \\ \dot{\theta} &= q\cos\phi - r\sin\phi \\ \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \end{aligned} \qquad \ldots(7.28)$$

The same can be represented in state space representation with six degrees of freedom like equation 7.25 of the aircraft with state vector x and input vector u as defined in equation 7.27. In [211] helicopter data of SA330 Puma is given and the same is used for implementing the PID controller for the same and then later fine tuning of the same is employed using GA to improve its real time performance.