1. INTRODUCTION

Artificial Neural Networks (ANNs) and Fuzzy Logic (FL) have been increasingly in use in many engineering fields since their introduction as mathematical aids by McCulloch and Pitts, 1943, and Zadeh, 1965, respectively. Being branches of Artificial Intelligence (AI), both emulate the human way of using past experiences, adapting itself accordingly and generalizing. While the former have the capability of learning by means of parallel connected units, called neurons, which process inputs in accordance with their adaptable weights usually in a recursive manner for approximation; the latter can handle imperfect information through linguistic variables, which are arguments of their corresponding membership functions.

Although the fundamentals of ANNs and FL go back as early as 1940s and 1960s, respectively, significant advancements in applications took place around 1980s. After the introduction of back-propagation algorithm for training multi-layer networks by Rumelhart and McClelland, 1986, ANNs has found many applications in numerous inter-disciplinary areas [1-3]. On the other hand, FL made a great advance in the mid 1970s with some successful results of laboratory experiments by Mamdani and Assilian [4]. In 1985, Takagi and Sugeno [5] contributed FL with a new rule-based modeling technique. Operating with linguistic expressions, fuzzy logic can use the experiences of a human expert and also compensate for inadequate and uncertain knowledge about the system. On the other hand, ANNs have proven superior learning and generalizing capabilities even on completely unknown systems that can only be described by its input-output characteristics. By combining these features, more versatile and robust models, called “neuro-fuzzy” architectures have been developed, [6-7].

In a control system the plant displaying nonlinearities has to be described accurately in order to design an effective controller. In obtaining the model, the designer has to follow one of two ways. The first one is using the knowledge of physics, chemistry, biology and the other sciences to describe an equation of motion with Newton’s laws, or electric circuits and motors with Ohm’s, Kirchhoff’s or Lentz’s laws depending on the

plant of interest. This is generally referred to as *mathematical modeling*. The second way requires the experimental data obtained by exciting the plant, and measuring its response. This is called *system identification* and is preferred in the cases where the plant or process involves extremely complex physical phenomena or exhibits strong nonlinearities.

Conventional control methods rely upon strong mathematical modeling, analysis, and synthesis. In case where mathematical models are available conventional control theory acts as a powerful tool for controlling even complex systems. On the other hand obtaining a mathematical model for a system can be rather complex and time consuming as it often requires some assumptions such as defining an operating point and doing linearization about that point and ignoring some system parameters, etc. This fact has recently led the researchers to exploit the neural and fuzzy techniques in modeling and control of complex systems.

Although fuzzy logic allows one to model (control) a system using human knowledge and experience with if-then rules, it is not always adequate on its own. This is also true for ANNs, which only deal with numbers rather than linguistic expressions. This deficiency can be overcome by combining the superior features of the two methods, as is performed in ANFIS architecture introduced by Jang, 1993 [8-11].

ANFIS architecture which was used in here as the controller of the dynamic system is generally encountered in the areas of function approximation, fault detection, medical diagnosis and control, [12-17].

The thesis is organized as follows. Section 2 starts with a brief introduction to *UAV*-Unmanned Air Vehicle system as a model to be controlled. In section 3, theory of the proposed control schemes, conventional PID, PID type fuzzy and ANFIS controllers are described. Simulation results are presented in Section 4. The performances and potentials of these control methods are evaluated by using *MATLAB*'s standard configuration and the *Aerosim Aeronautical Simulation Block Set,* the aircraft simulated being *Aerosonde UAV.* Finally concluding remarks and information about future work are given in section 5 and section 6.

2. UAV-UNMANNED AIR VEHICLE

"UAVs are remotely piloted or self-piloted aircrafts that can carry many different types of payloads such as cameras, sensors and communications equipment. They have a very wide range of applications including both civil and military areas. Missions which are dull, dirty, or dangerous are considered ideal for UAVs. Some important features that make them very popular are their low cost, smaller size and their extended maneuver capability because of the absence of a human pilot.[18]" Besides, The Department of Defense Dictionary defines an unmanned air vehicle (UAV) as "A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non-lethal payload [18]."

2.1. The Aerosonde UAVs

In this study, Aerosonde is used as a simple UAV model. The Aerosonde (shown in Figure 2.1) is a small (33 pound weight, 12 pound payload capacity, 9.4 foot wingspan) long endurance (30 hours) UAV. It was the first UAV to complete a transatlantic flight when it flew from Newfoundland, Canada to Scotland in 26 hours and 45 minutes in August 1998 [20].

Figure 2.1. A picture of the Aerosonde UAV in flight

The Aerosonde is too small to carry a satellite data link so the UAV must be autonomous when it is over the horizon from its ground station, which it was for most of its flight across the Atlantic Ocean. The Aerosonde has flown in many harsh environments, including Alaska. In September of 2005, the Aerosonde was flown into Hurricane Ophelia [21].

2.2. UAV Dynamics

The **AeroSim** aeronautical simulation blockset [22] provides a complete set of tools for the rapid development of nonlinear 6-degree-of-freedom aircraft dynamic models. In addition to the basic aircraft dynamics blocks, the library also includes complete aircraft models which can be customized through parameter files.

The model used here is Aerosonde UAV 6-dof aircraft model which can be found in the AeroSim blockset. It is a small autonomous airplane designed for weather-reconnaissance and remote-sensing missions. The 6-dof aircraft model-body frame EOM is shown in Figure 2.2.

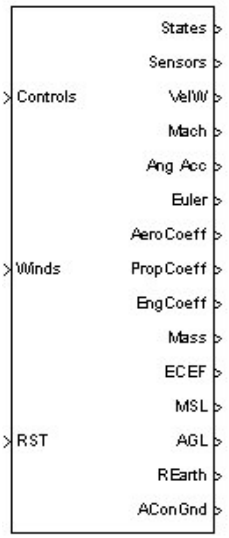


Figure 2.2. 6-Dof Aircraft Model Block

Block characteristics:

1. *Parameters:*

- Initial velocities = the 3×1 vector of initial aircraft velocity components (groundspeed in body axes) $[u\ v\ w]T$.
- Initial angular rates = the 3×1 vector of initial aircraft angular rates (in body axes) $[\ p\ q\ r\ ]T$.
- Initial attitude = the 4×1 vector of initial aircraft attitude provided as Euler-Rodrigues quaterions $[e_0\ e_x\ \ e_y\ \ e_z\ ]T$.
- Initial position = the 3×1 vector of initial aircraft location $[Lat\ Lon\ Alt\ ]T$, in [rad rad m].
- Initial fuel mass = the initial mass of the fuel quantity available on-board the aircraft, in kg.
- Initial engine speed = the initial engine shaft rotation speed, in rad/s.
- Ground altitude = the altitude of the terrain relative to mean-sea-level, at aircraft location, in meters.
- Sample time = the sample time at which the aircraft model will run.

2. *Input s:*

- Controls = the 7×1 vector of aircraft controls $[\ f\ lap\ elevator\ aileron\ rudder\ throttle\ mixture\ ignition\ ]T$ in [rad rad rad rad frac ratio bool].
- Winds = the 3×1 vector of background wind velocities, in navigation frame $[W_N\ W_E\ W_D\ ]^T$, in m/s.
- RST = the integrator reset flag (can take values of 0 or 1, all integrators reset on rising-edge).

3. *Outputs:*

- States = the 15×1 vector of aircraft states $[u\ v\ w\ p\ q\ r\ e_0\ e_x\ e_y\ e_z\ Lat\ Lon\ Alt\ m_{fuel}\ \Omega_{eng}\ ]^T$
- Sensors = the 18×1 vector of sensor data $[Lat\ Lon\ Alt\ V_N\ V_E\ V_D\ a_x\ a_y\ a_z\ p\ q\ r\ p_{stat}\ p_{dyn}\ OAT\ H_x\ H_y\ H_z\ ]^T$.
- VelW = the 3×1 vector of aircraft velocity in wind axes $[V_a\ \ \beta\ \alpha\ ]^T$ in [m/s rad rad].

• Mach = the current aircraft Mach number.

• Angular Acc = the 3×1 vector of body angular accelerations $[\dot{p}\ \dot{q}\ \dot{r}]^T$.

• Euler = the 3×1 vector of the attitude of the aircraft given in Euler angles $[\phi\ \theta\ \psi]^T$, in radians.

• AeroCoeff = the 6×1 vector of aerodynamic coefficients $[C_D\ C_Y\ C_L\ C_l\ C_m\ C_n]^T$, in rad^{-1}.

• PropCoeff = the 3×1 vector of propeller coefficients $[J\ C_T\ C_P]^T$.

• EngCoeff = the 5×1 vector of engine coefficients $[MAP\ \dot{m}_{air}\ \dot{m}_{feul}\ BSFC\ P]^T$ given in [kPa kg/s kg/s g/(W*hr) W].

• Mass = the current aircraft mass, in kg.

• ECEF = the 3 × 1 vector of aircraft position in the Earth-centered, Earth-fixed frame $[X\ Y\ Z]^T$.

• MSL = the aircraft altitude above mean-sea-level, in m.

• AGL = the aircraft altitude above ground, in m.

• REarth = the Earth equivalent radius, at current aircraft location, in m.

• AConGnd = the aircraft-on-the-ground flag (0 if aircraft above ground, 1 if aircraft is on the ground).

The internal layout of the complete aircraft model is shown in Fig.2.3. All sub-systems were designed using the AeroSim blocks.

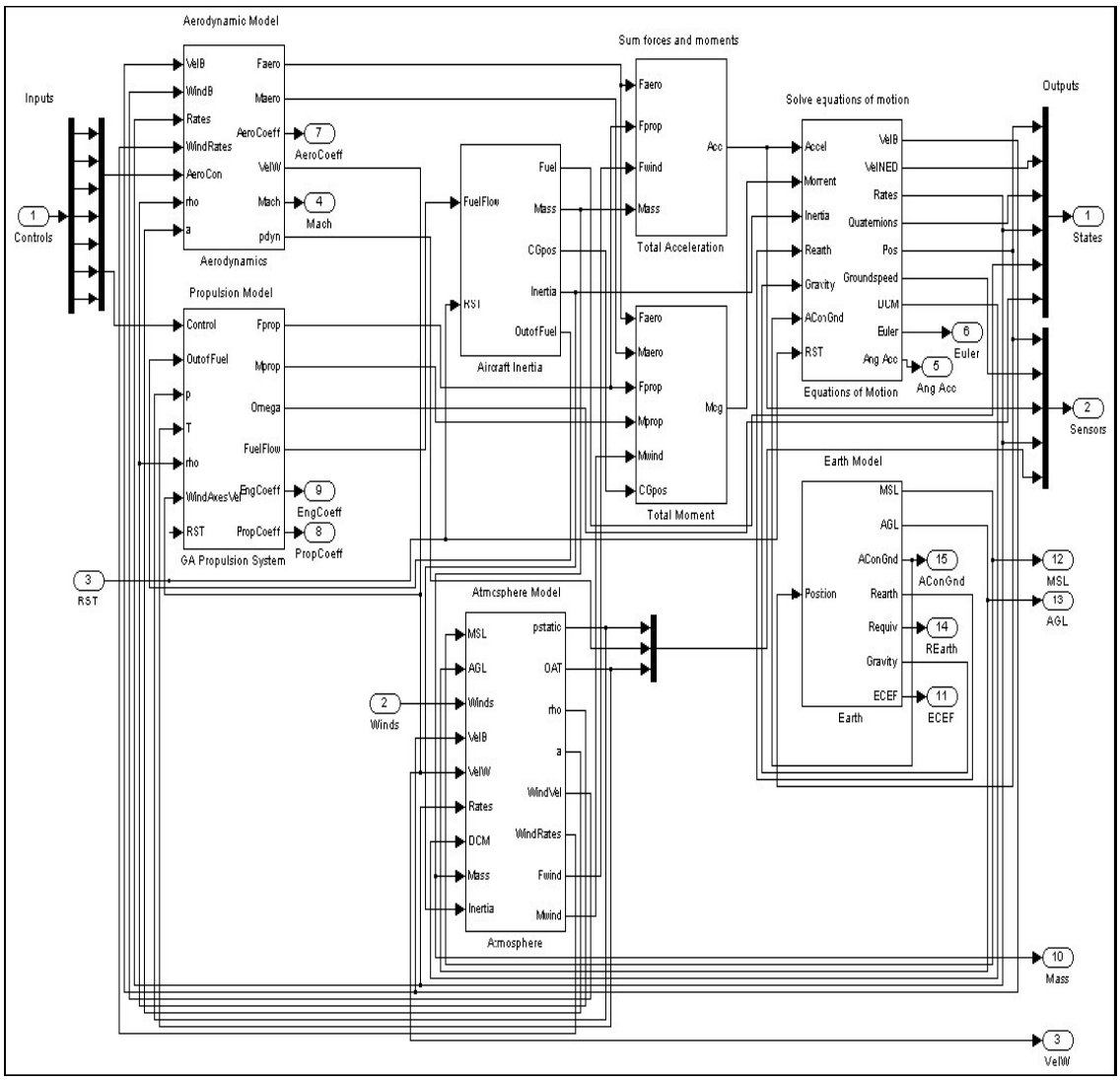


Figure 2.3. Internal Structure Of The Complete Aircraft Model

A simplified diagram is presented in Fig2.4. The aerodynamics, propulsion, and inertia models compute the airframe loads (forces and moments) as functions of control inputs and environment (atmosphere and Earth) effects. The resulting accelerations are then integrated by the Equations of Motion to obtain the aircraft states (position, velocity, attitude, angular velocities). The aircraft states then will affect the output of the environment blocks at the next iteration (for example altitude changes result in atmospheric pressure changes, latitude and longitude variations result in gravity

variations). Also, the aircraft states are used in the computation of sensor outputs (GPS, inertial measurements, etc.).

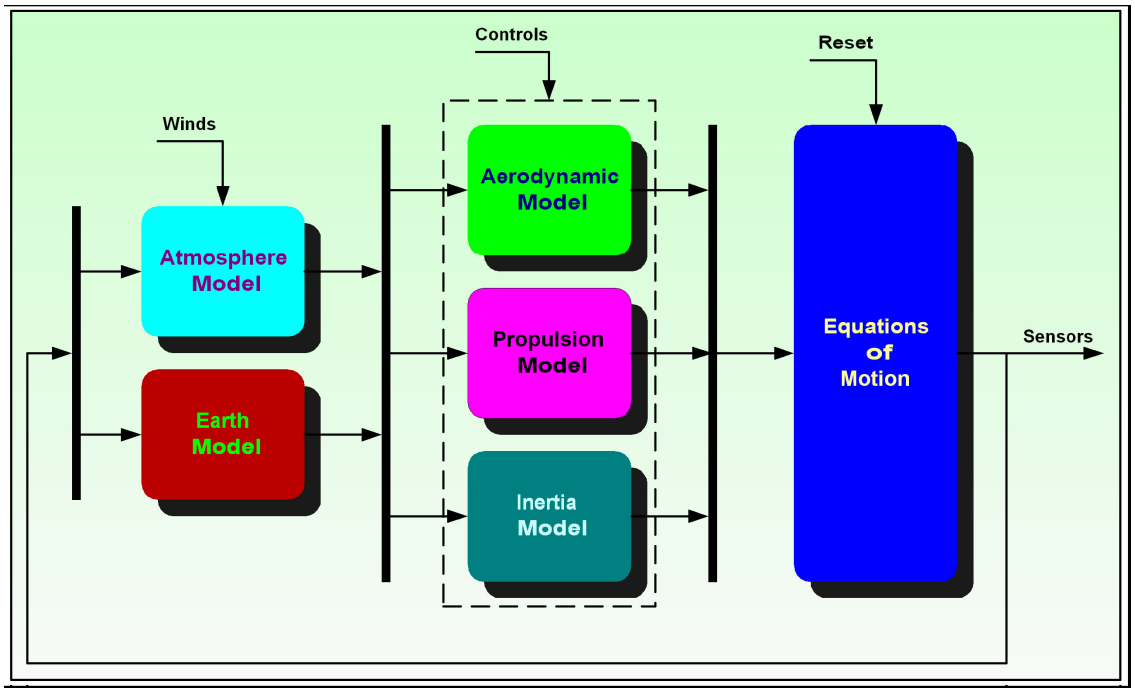


Figure 2.4. Simplified Internal Structure Diagram of The Aircraft Model

To illustrate the functionality of the Aerosim blockset, 3 different simulation results are given below. All the simulations are done for 1 minute with 20 ms sample time.

2.3. Open-Loop Flight

The Simulink model shown in Figure 2.5 represents a simulation of the aircraft in open-loop flight. That is, all aircraft control inputs are set to fixed values, independent of the aircraft states. In Figure 2.5 the controller values are set to 0.001 N/m for elevator, -0.005 N/m for aileron and -0,7 N/m for throttle. The unbalanced roll moment caused by the propulsion system excites the spiral mode and the aircraft settles in a constant bank angle turn, as shown in Figure 2.6.

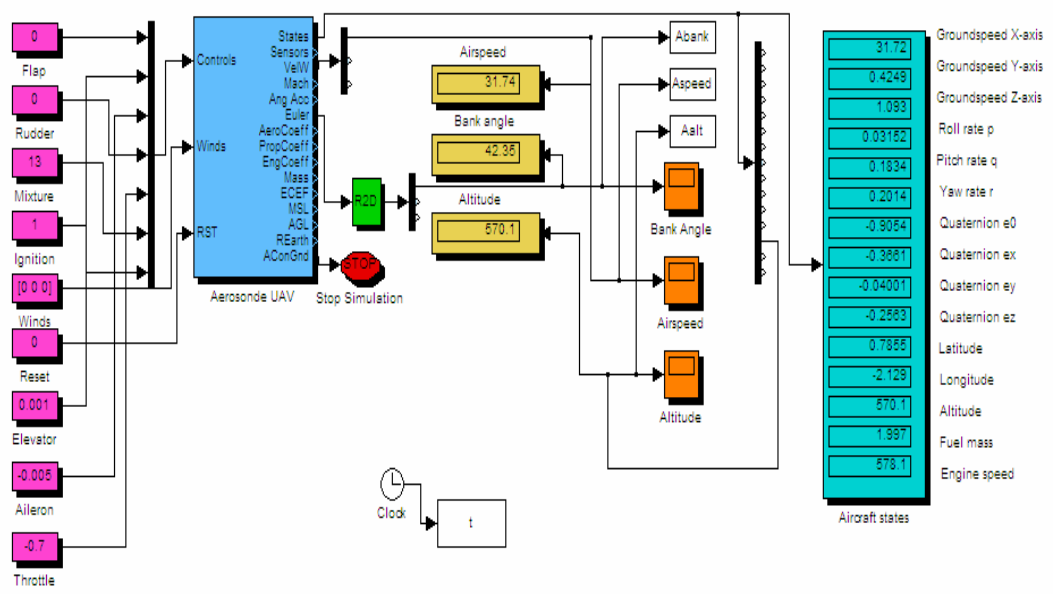


Figure 2.5. Aerosonde Open-Loop Flight Model

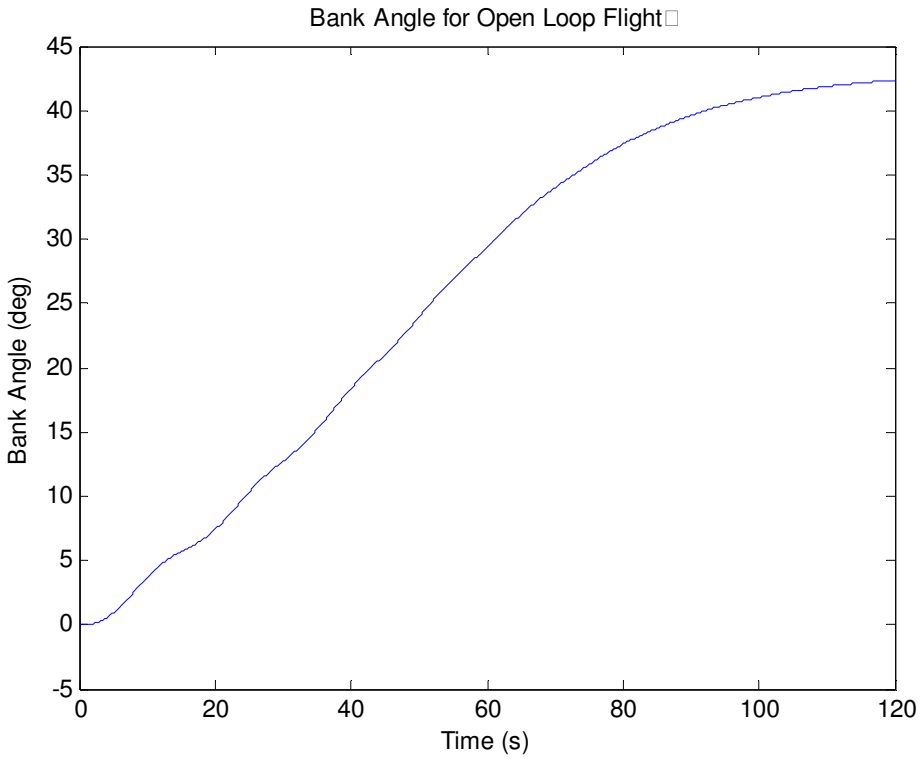


Figure 2.6. Bank Angle Output

2.3.1. Lateral Control

In this example, we stabilize the lateral dynamics of the aircraft by adding a wing leveler. The Simulink diagram is shown in Figure 2.7. This is implemented using proportional and integral feedback from bank angle to ailerons. It can be seen in Figure 2.8. that the bank angle now settles to zero (wings level attitude).

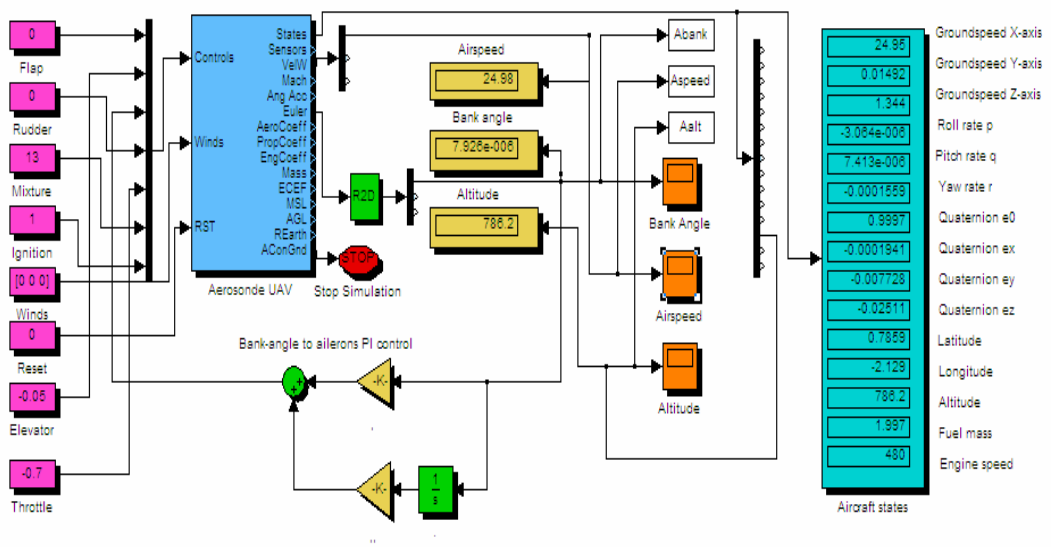


Figure 2.7. Aerosonde Lateral Control Model

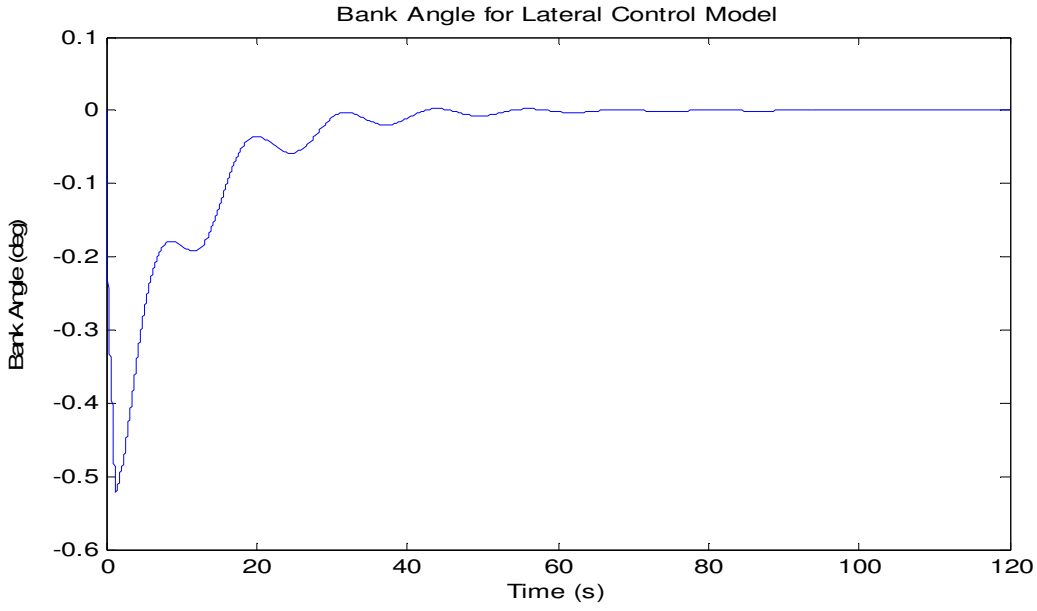


Figure 2.8. Bank Angle Output

2.3.2. Airspeed Control

With the lateral dynamics of the aircraft now under control, we take a quick look at the longitudinal dynamics, in Figure 2.9.

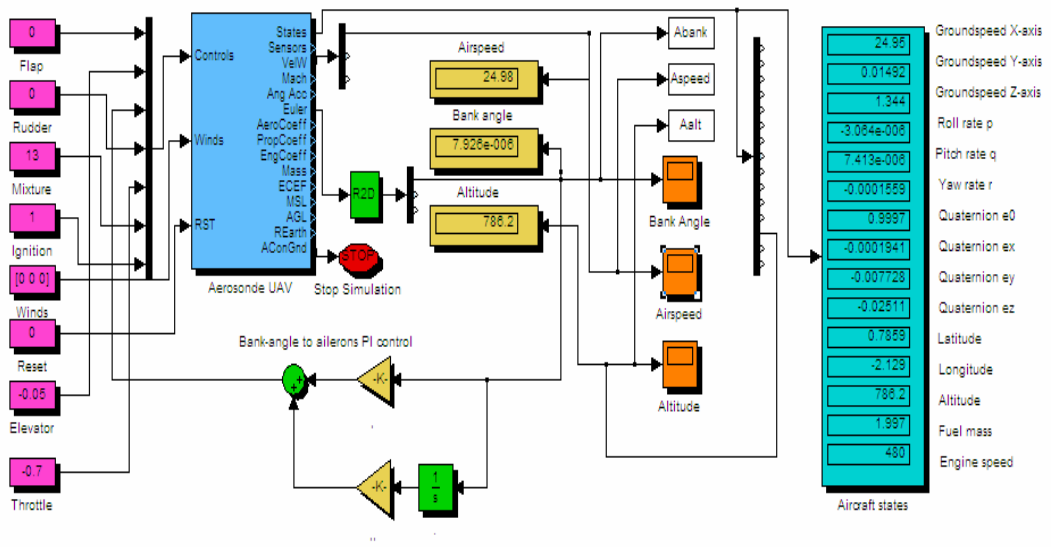


Figure 2.9. Longitudinal & Open-Loop and Lateral & Closed-Loop Control Model

In Figure 2.10. we can see the time history of airspeed. The initial phugoid oscillations (long period oscillation damping in the longitudinal motion of an aircraft) take approximately 80 seconds to damp out. At the same time, the airspeed settles to a value which depends on the elevator setting.

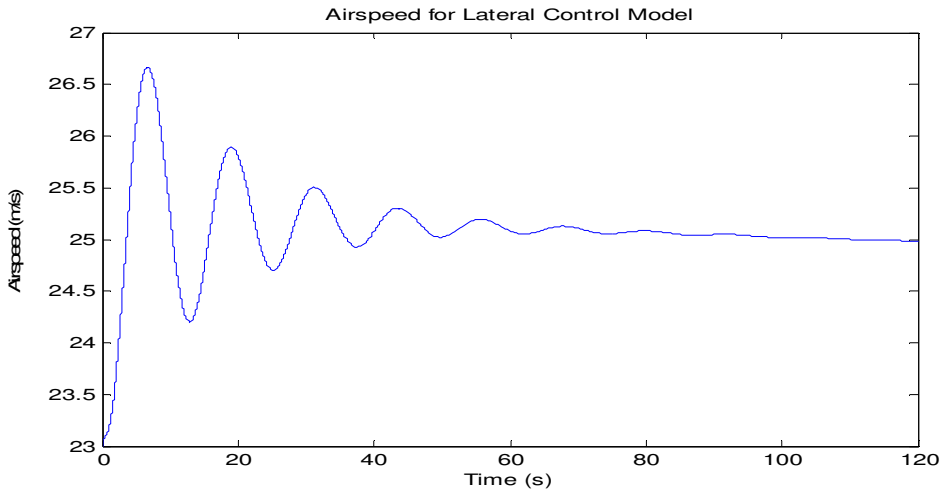


Figure 2.10. Airspeed Output

By adding a PID (proportional, integral, derivative) control loop from airspeed to elevator - see Figure 2.11. we can improve the phugoid damping (long-period oscillation damping in the longitudinal motion of the aircraft) and provide the aircraft with the ability to track airspeed commands. The airspeed command is set to a constant value of 23 m/s.

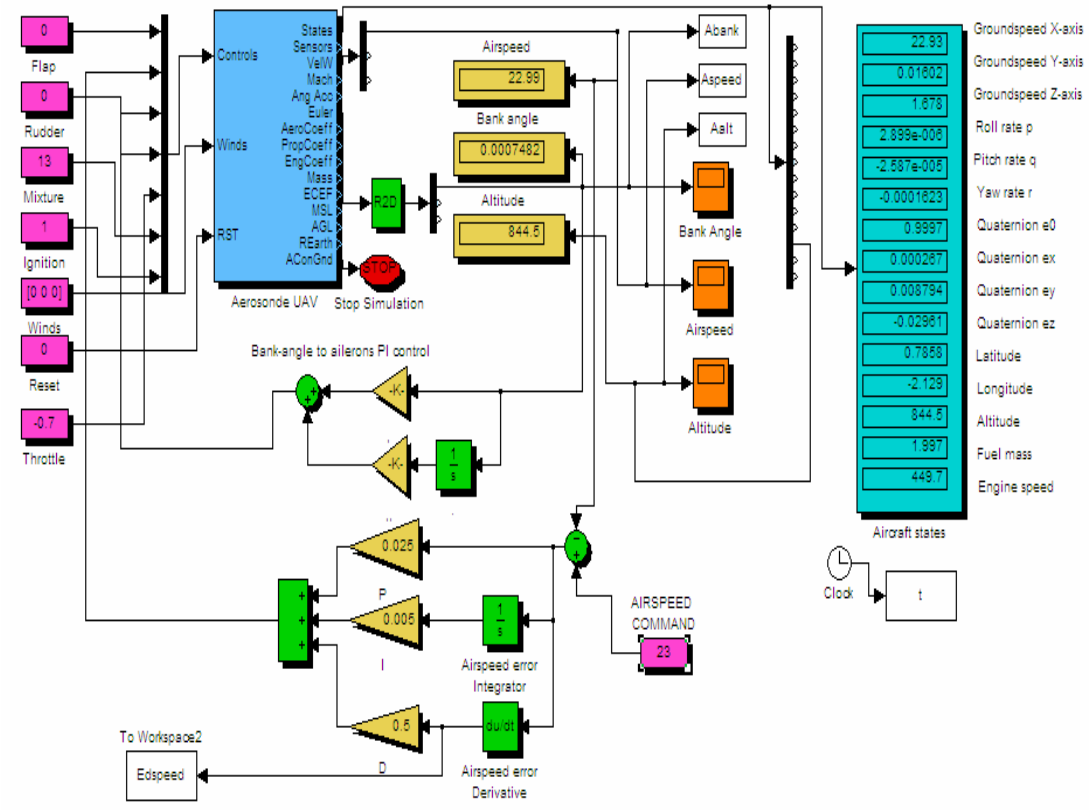


Figure 2.11. Aerosonde Longitudinal Control

By running the simulation, you should see a plot of the airspeed similar to Figure 2.12. The airspeed does indeed settle to the commanded value, while the phugoid oscillations are virtually eliminated

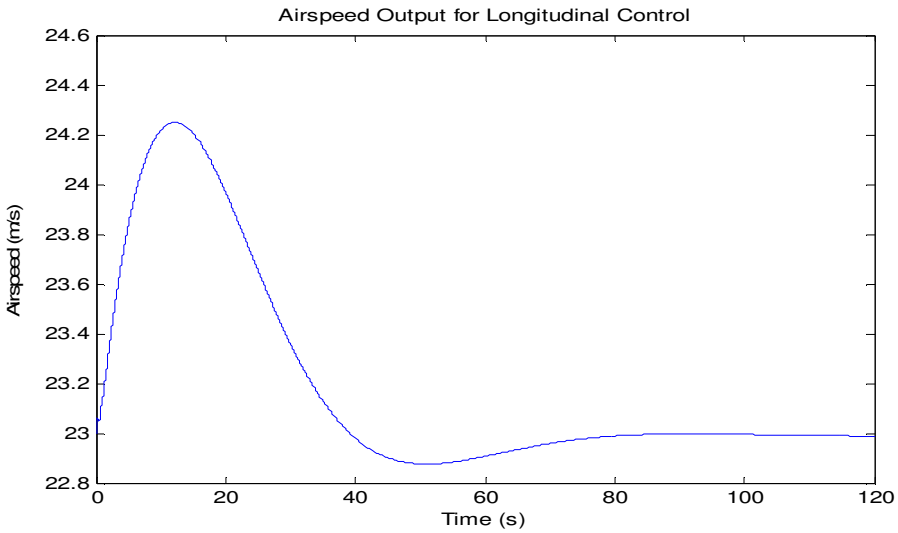


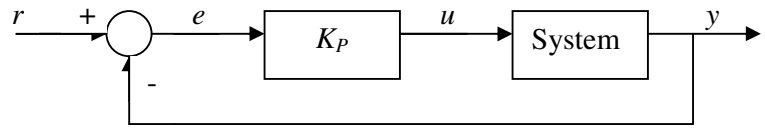
Figure 2.12. Airspeed Outputs

3. CONTROL SCHEMES

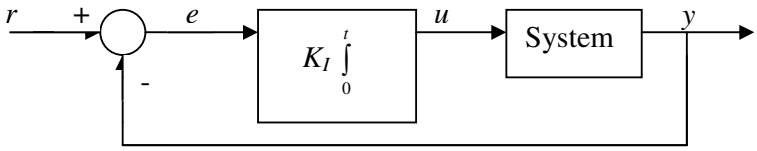
3.1. PID Control

In this section, we give a brief review on several types of conventional PID controllers; their configuration, design methods and stability analysis, which will be needed in introducing the fuzzy PID controller later, where the fuzzy PID controllers are natural extensions of the conventional ones: they have the same structure but are defined based on fuzzy mathematics and fuzzy control strategies.

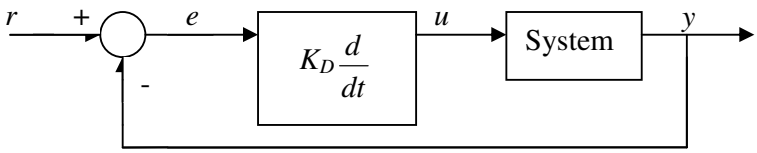
The conventional proportional (P), integral (I), and derivative (D) controllers for controlling a given system (plant, process) have the structures shown in Figure 3.1.(a), (b) and (c), respectively, where $r = r(t)$ is reference input (set-point), $y = y(t)$ is the controlled system's output, $e = e(t) := r(t) - y(t)$ is the set point tracking error, and $u = u(t)$ is the output of the controller which is used as the input to the system.



(a) Proportional Controller



(b) Integral Controller



(c) Derivative Controller

Figure 3.1 Proportional-integral-derivative controllers

In the time domain, they can be expressed as follows:

- The P-controller $u(t)=K_p e(t)$
- The I-controller $u(t)=K_I \int_0^t e(\tau)\,d\tau$
- The D-controller $u(t)= K_D \frac{d}{dt} e(t)$

Here, the three control gains, K_P, K_I and K_D, are constants to be determined in the design for the set-point tracking performance and stability consideration. To increase their control capabilities, they are usually used in combinations, which will be further discussed below.

In the frequency domain, they can be interpreted as follows:

- The P-controller $U(s)=K_p E(s)$
- The I-controller $U(s)=\frac{K_I}{s} E(s)$
- The D-controller $U(s)= K_D\, s\, E(s)$

Here, we use capital letters for the laplace transform $L\{.\}$ of a continuous time signal. Thus,

$$U(s)=L\{u(t)\} \text{ and } E(s)=L\{e(t)\},$$

with zero initial conditions.

The followings are some typical combinations of P, I and D controllers:

- The PI-controller $u(t)=K_p e(t)+K_I \int_0^t e(\tau)\,d\tau$
- The PD-controller $u(t)= K_P\, e(t)+K_D \frac{d}{dt} e(t)$

- The PID-controller $$u(t)=K_P e(t)+K_I\int_0^t e(\tau)\,d\tau+K_D\frac{d}{dt}e(t)$$
- The PI+D-controller $$u(t)=K_P e(t)+K_I\int_0^t e(\tau)\,d\tau-K_D\frac{d}{dt}y(t)$$

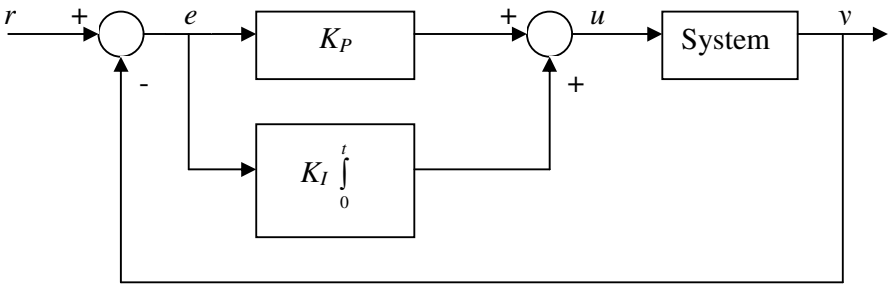
Corresponding frequency domain representations are as follows:

- The PI-controller $$U(s)=K_p E(s)+\frac{K_I}{s}E(s)$$
- The PD-controller $$U(s)=K_P E(s)+K_D sE(s)$$
- The PID-controller $$U(s)=K_P E(s)+\frac{K_I}{s}E(s)+K_D sE(s)$$
- The PI+D-controller $$U(s)=K_P E(s)+\frac{K_I}{s}E(s)-K_D sY(s)$$

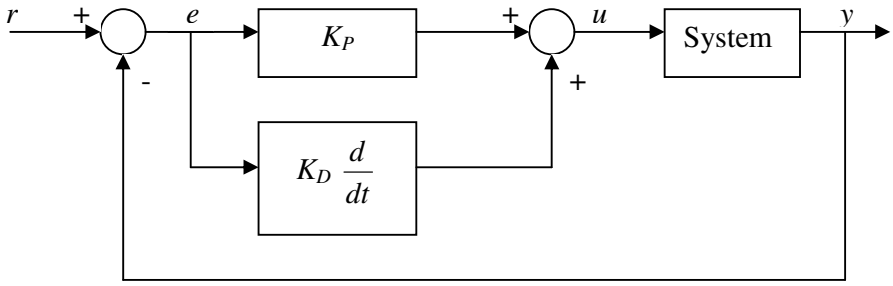
In the above,

$$Y(s)=L\{y(t)\}=L\{r(t)-e(t)\}=R(s)-E(s)$$

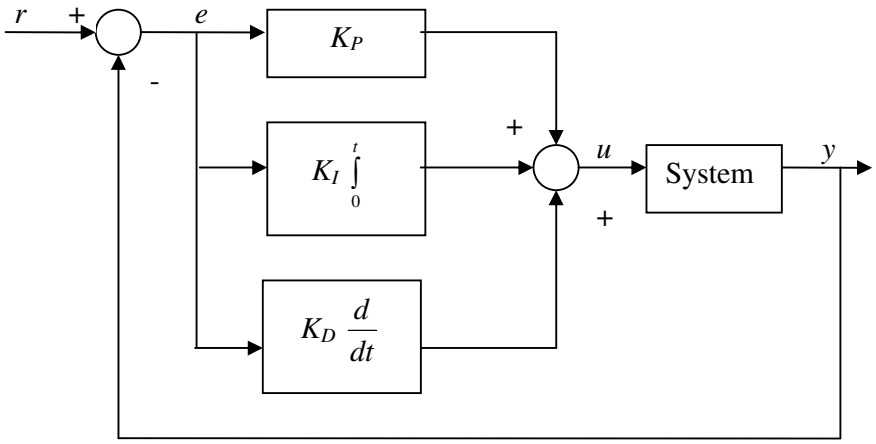
The PID controller shown in Figure3.2.(c) is not a good combination of the three controllers in practice, since if the error signal $e(t)$ has discontinuities then the D-controller will produce very bad (even unbounded) responses. This combination is good only for illustration since it has a neat formulation, and hence is called a "textbook PID controller." A practical combination of the three controllers is the PI+D controller shown in Figure 3.2 (d), where the system output signal $y(t)$ is usually smoother than the error signal (though the I-controller and the system). More importantly, this structure of the closed loop control systems has been validated to be efficient by many practical examples and case studies.



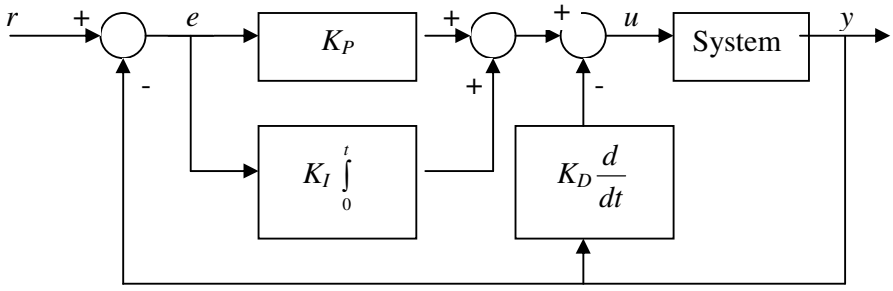
(a) PI controller



(b) PD controller



(c) PID controller



(d) PI+D controller

Figure 3.2. Some typical combinations of P, I and D controllers

3.2. Fuzzy Control

Fuzzy controllers are used to control consumer products, such as washing machines, video cameras, and rice cookers, as well as industrial processes, such as cement kilns, underground trains, and robots. Fuzzy control is a control method based on fuzzy logic. Just as fuzzy logic can be described simply as “computing with words rather than numbers”, fuzzy control can be described simply as ’’control with sentences rather than equations’’. A fuzzy controller can include empirical rules, and that is especially useful in operator controlled plants. Take for instance a typical fuzzy controller:

If error is Neg and change in error is Neg then output is NB
If error is Neg and change in error is Zero then output is NM (3.1)

The collection of rules is called a *rule base*. The rules are in the familiar if-then format, and formally the if-side is called the *condition* and the then-side is called the *conclusion*(more often, perhaps, the pair is called *antecedent* - *consequent* or *premise* - *conclusion*). The input value ’’Neg’’ is a *Linguistic Term* short for the word *Negative,* the output value ’’NB’’ stands for *Negative Big* and ’’NM’’ for *Negative Medium.* The computer is able to execute the rules and compute a control signal depending on the measured inputs *error* and *change in error.* The objective here is to identify and explain the various design choices for engineers.

In a rule based controller the control strategy is stored in a more or less natural language. The control strategy is isolated in a rule base opposed to an equation based description. A rule based controller is easy to understand and easy to maintain for a non-specialist end-user. An equivalent controller could be implemented using conventional techniques. It is just that it is convenient to isolate the control strategy in a rule base for operator controlled systems.

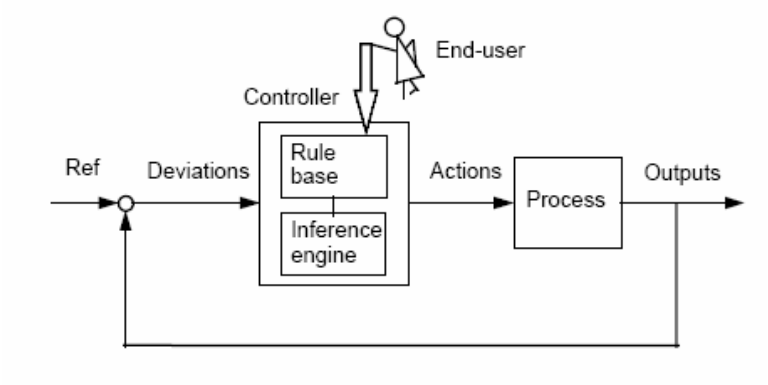


Figure 3.3. Direct control

Fuzzy controllers are being used in various control schemes. The most obvious one is *direct control*, where the fuzzy controller is in the forward path in a feedback control system (Figure 3.3). The process output is compared with a reference, and if there is a deviation, the controller takes action according to the control strategy. In the figure, the arrows may be understood as hyper-arrows containing several signals at a time for multiloop control. The controller is here a fuzzy controller, and it replaces a conventional controller, a *PID,* (proportional-integral-derivative) controller.

In *feedforward control* (Figure 3.4.) a measurable disturbance is being compensated. It requires a good model, but if a mathematical model is difficult or expensive to obtain, a fuzzy model may be useful. Figure 3.4. shows a controller and the fuzzy compensator, the process and the feedback loop are omitted for clarity. The scheme, disregarding the disturbance input, can be viewed as a collaboration of linear and nonlinear control actions; the controller C may be a linear PID controller, while the fuzzy controller F is a supplementary nonlinear controller.

Fuzzy rules are also used to correct tuning parameters in *parameter adaptive control* schemes (Figure 3.5). If a nonlinear plant changes operating point, it may be possible to change the parameters of the controller according to each operating point. This is called *gain scheduling* since it was originally used to change process gains. A gain scheduling controller contains a linear controller whose parameters are changed as a function of the operating point in a preprogrammed way. It requires thorough knowledge of the plant, but it is often a good way to compensate for nonlinearities and parameter

variations. Sensor measurements are used as scheduling variables that govern the change of the controller parameters, often by means of a table look-up.

Whether a fuzzy control design will be stable is a somewhat open question. Stability concerns the system's ability to converge or stay close to an equilibrium. A *stable* linear system will converge to the equilibrium asymptotically no matter where the system state variables start from. It is relatively straight forward to check for stability in linear systems,

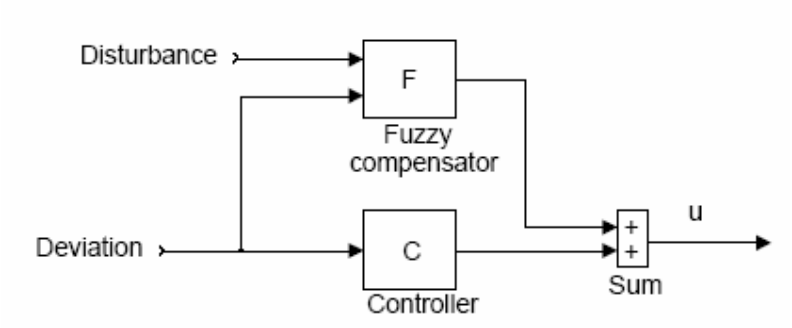


Figure 3.4. Feedforward control

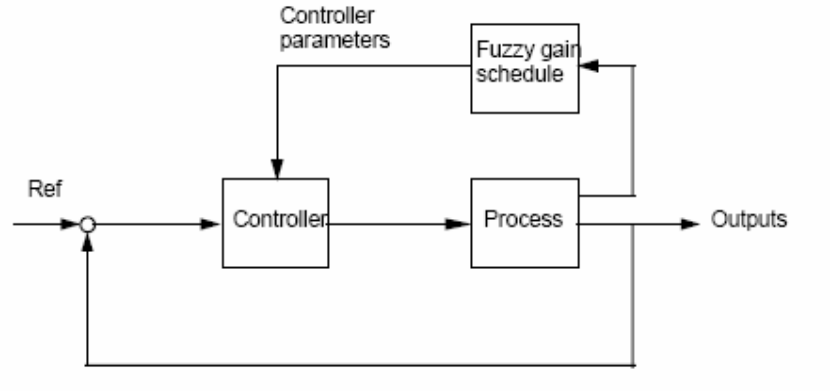


Figure 3.5. Fuzzy parameter adaptive control

for example by checking that all eigenvalues are in the left half of the complex plane. For nonlinear systems, and fuzzy systems are most often nonlinear, the stability concept is more complex. A nonlinear system is said to be *asymptotically stable* if, when it starts

close to an equilibrium, it will converge to it. Even if it just stays close to the equilibrium, without converging to it, it is said to be *stable* (in the sense of Lyapunov). To check conditions for stability is much more difficult with nonlinear systems, partly because the system behavior is also influenced by the signal *amplitudes* apart from the *frequencies*. The literature is somewhat theoretical and interested readers are referred to [23-24]. They report on four methods (Lyapunov functions, Popov, circle, and conicity), and they give several references to scientific papers. It is characteristic, however, that the methods give rather conservative results, which translate into unrealistically small magnitudes of the gain factors in order to guarantee stability.

Another possibility is to approximate the fuzzy controller with a linear controller, and then apply the conventional linear analysis and design procedures on the approximation. It seems likely that the stability margins of the nonlinear system would be close in some sense to the stability margins of the linear approximation depending on how close the approximation is. This paper shows how to build such a linear approximation, but the theoretical background is still unexplored.

There are at least four main sources for finding control rules [25].

- *Expert experience and control engineering knowledge.* One classical example is the operator's handbook for a cement kiln [26]. The most common approach to establishing such a collection of rules of thumb, is to question experts or operators using a carefully organised questionnaire.
- *Based on the operator's control actions* Fuzzy *If-then* rules can be deduced from observations of an operator's control actions or a log book. The rules express input-output relationships.
- *Based on a fuzzy model of the process.* A linguistic rule base may be viewed as an inverse model of the controlled process. Thus the fuzzy control rules might be obtained by inverting a fuzzy model of the process. This method is restricted to relatively low order systems, but it provides an explicit solution assuming that fuzzy models of the open and closed loop systems are available [25]. Another approach is *fuzzy identification* [25, 27] or fuzzy model-based control (see later).
- *Based on learning* The self-organising controller is an example of a controller that finds the rules itself. Neural networks is another possibility.

There is no design procedure in fuzzy control such as root-locus design, frequency response design, pole placement design, or stability margins, because the rules are often nonlinear. Therefore we will settle for describing the basic components and functions of fuzzy controllers are described in order to recognise and understand the various options in commercial software packages for fuzzy controller design.

There is much literature on fuzzy control and many commercial software tools but there is no agreement on the terminology, which is confusing. There are efforts, however, to standardise the terminology, and the following makes use of a draft of a Standard from the International Electrotechnical Committee [28]. Throughout, letters denoting matrices are in bold upper case, for example **A**; vectors are in bold lower case, for example **x** scalars are in italics, for example *n*; and operations are in bold, for example **min**.

3.2.1. Structure of A Fuzzy Controller

There are specific components characteristic of a fuzzy controller to support a design procedure. In the block diagram in Figure 3.6. the controller is between a preprocessing block and a postrocessing block. The following explains the diagram block by block.

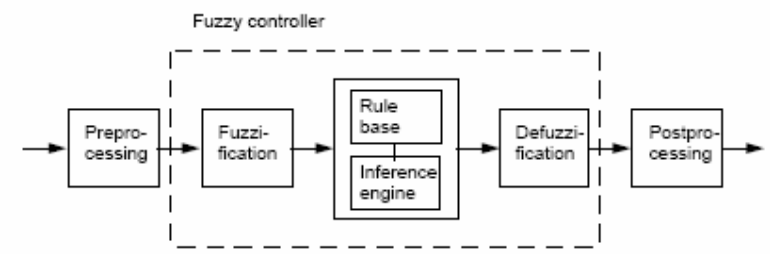


Figure 3.6. Blocks of a fuzzy controller

Preprocessing

The inputs are most often hard or *crisp* measurements from some measuring equipment, rather than linguistic. A preprocessor, the first block in Figure 3.6., conditions the measurements before they enter the controller. Examples of preprocessing are:

- Quantization in connection with sampling or rounding to integers;
- normalization or scaling onto a particular, standard range;
- filtering in order to remove noise;
- averaging to obtain long term or short term tendencies;
- a combination of several measurements to obtain key indicators; and
- differentiation and integration or their discrete equivalences.

A *quantiser* is necessary to convert the incoming values in order to find the best level in a discrete *universe*. Assume, for instance, that the variable *error* has the value 4.5, but the universe is u=(-5, -4, ….0,….4, 5). The quantiser rounds to 5 to fit it to the nearest level. Quantization is a means to reduce data, but if the quantization is too coarse the controller may oscillate around the reference or even become unstable.
Nonlinear *scaling* is an option (Figure 3.7). In the *FL Smidth* controller the operator is asked

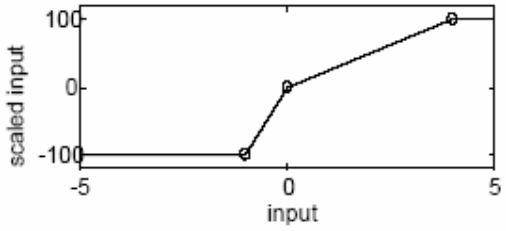


Figure 3.7. Example of nonlinear scaling of an input measurement

to enter three typical numbers for a small, medium and large measurement respectively [26]. They become break-points on a curve that scales the incoming measurements (circled in the figure). The overall effect can be interpreted as a distortion of the primary fuzzy sets. It can be confusing with both scaling and gain factors in a controller, and it makes tuning difficult.

When the input to the controller is *error*, the control strategy is a static mapping between input and control signal. A dynamic controller would have additional inputs, for example derivatives, integrals, or previous values of measurements backwards in time. These are created in the preprocessor thus making the controller multi-dimensional, which requires many rules and makes it more difficult to design.
The preprocessor then passes the data on to the controller.

Fuzzification

The first block inside the controller is *fuzzification*, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block thus matches the input data with the conditions of the rules to determine how well the condition of each rule matches that particular input instance. There is a degree of membership for each linguistic term that applies to that input variable.

Rule Base

The rules may use several variables both in the condition and the conclusion of the rules. The controllers can therefore be applied to both multi-input-multi-output (MIMO) problems and single-input-single-output (SISO) problems. The typical SISO problem is to regulate a control signal based on an error signal. The controller may actually need both the *error*, the *change in error*, and the *accumulated error* as inputs, but we will call it single-loop control, because in principle all three are formed from the error measurement. To simplify, we assume that the control objective is to regulate some process output around a prescribed setpoint or reference.

Rule formats

Basically a linguistic controller contains rules in the *If-then* format, but they can be presented in different formats. In many systems, the rules are presented to the end-user in a format similar to the one below,

1. If error is Neg and change in error is Neg then output is NB
2. If error is Neg and change in error is Zero then output is NM
3. If error is Neg and change in error is Pos then output is Zero
4. If error is Zero and change in error is Neg then output is NM
5. If error is Zero and change in error is Zero then output is Zero
6. If error is Zero and change in error is Pos then output is PM
7. If error is Pos and change in error is Neg then output is Zero
8. If error is Pos and change in error is Zero then output is PM
9. If error is Pos and change in error is Pos then output is PB

(3.2)

The names *Zero, Pos, Neg* are labels of fuzzy sets as well as *NB, NM, PB and PM* (negative big, negative medium, positive big, and positive medium respectively). The same set of rules could be presented in a *relational* format, a more compact representation.

Table 3.1. Compact representation for set of rules

Error	Change in error	Output
Neg	Pos	Zero
Neg	Zero	NM
Neg	Neg	NB
Zero	Pos	PM
Zero	Zero	Zero
Zero	Neg	NM
Pos	Pos	PB
Pos	Zero	PM
Pos	Neg	Zero

(3.3)

The top row is the heading, with the names of the variables. It is understood that the two leftmost columns are inputs, the rightmost is the output, and each row represents a rule. This format is perhaps better suited for an experienced user who wants to get an overview of the rule base quickly. The relational format is certainly suited for storing in a relational database. It should be emphasised, though, that the relational format implicitly assumes that the connective between the inputs is always logical **and** - or logical **or** for that matter as long as it is the same operation for all rules_and not a mixture of connectives. Incidentally, a fuzzy rule with an **or** combination of terms can be converted into an equivalent **and** combination of terms using laws of logic (DeMorgan's laws among others). A third format is the tabular linguistic format.

		Change in error		
		Neg	Zero	Pos
Error	Neg	NB	NM	Zero
	Zero	NM	Zero	PM
	Pos	Zero	PM	PB

(3.4)

This is even more compact. The input variables are laid out along the axes, and the output variable is inside the table. In case the table has an empty cell, it is an indication of a missing rule, and this format is useful for checking completeness. When the input variables are *error* and *change in error*, as they are here, that format is also called a *linguistic phase plane*. In case there are $n > 2$ input variables involved, the table grows to an *n*-dimensional array; rather user-*un*friendly.

To accommodate several outputs, a nested arrangement is conceivable. A rule with several outputs could also be broken down into several rules with one output.

Lastly, a graphical format which shows the fuzzy membership curves is also possible (Figure3. 9). This graphical user-interface can display the inference process better than the other formats, but takes more space on a monitor.

Membership function

Every element in the universe of discourse is a member of a fuzzy set to some grade, maybe even zero. The grade of membership for all its members describes a fuzzy set, such as *Neg*. In fuzzy sets elements are assigned a *grade of membership*, such that the transition from membership to non-membership is gradual rather than abrupt. The set of elements that have a non-zero membership is called the *support* of the fuzzy set. The function that ties a number to each element {of the universe is called the *membership function* $\mu(x)$

The designer is inevitably faced with the question of how to build the term sets. There are two specific questions to consider: (i) How does one determine the shape of the sets? and (ii) How many sets are necessary and sufficient? For example, the *error* in the position controller uses the family of terms *Neg, Zero* and *Pos*. According to fuzzy set theory the choice of the shape and width is subjective, but a few rules of thumb apply.

- A term set should be sufficiently wide to allow for noise in the measurement.

- A certain amount of overlap is desirable; otherwise the controller may run into poorly
  defined states, where it does not return a well defined output.

A preliminary answer to questions (i) and (ii) is that the necessary and sufficient number of sets in a family depends on the width of the sets, and vice versa. A solution could be to ask the process operators to enter their personal preferences for the membership curves; but operators also find it difficult to settle on particular curves.

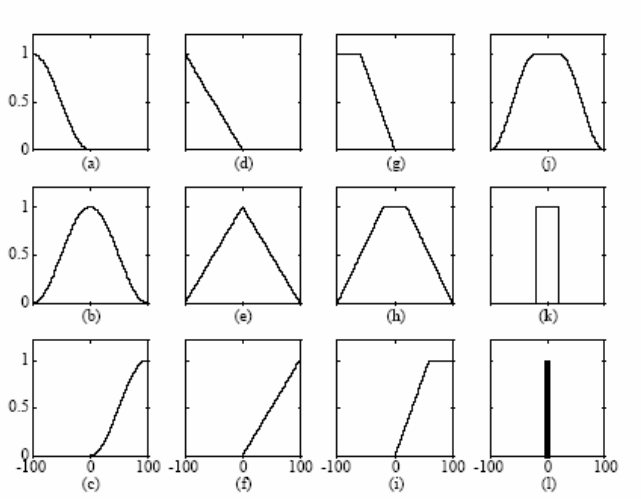


Figure 3.8. Examples of membership functions. Read from top to bottom, left to right: (a) s- function, (b) π function, (c) z function, (d-f) triangular versions, (g-i) trapezoidal versions, (j) flat π function, (k) rectangle, (l) singleton

The manual for the TILShell product recommends the following [29]

- *Start with triangular sets.* All membership functions for a particular input or output should be symmetrical triangles of the same width. The leftmost and the rightmost should be shouldered ramps.
- *The overlap should be at least 50 %.* The widths should initially be chosen so that each value of the universe is a member of at least two sets, except possibly for elements at the extreme ends. If, on the other hand, there is a gap between two sets no rules fire for values in the gap. Consequently the controller function is not defined.

Membership functions can be flat on the top, piece-wise linear and triangle shaped, rectangular, or ramps with horizontal shoulders. Figure 3.8. shows some typical shapes of membership functions.

Strictly speaking, a fuzzy set **A** is a collection of ordered pairs

$$A=\{(x,\mu(x))\} \tag{3.5}$$

Item x belongs to the universe and $\_\mu(x)$, is its grade of membership in **A**. A single pair $(x, \mu(x))$ is a fuzzy *singleton; singleton output* means replacing the fuzzy sets in the conclusion by numbers (scalars). For example

1. If error is Pos then output is 10 volts
2. If error is Zero then output is 0 volts
3. If error is Neg then output is -10 volts

There are at least three advantages to this:

- The computations are simpler;
- it is possible to drive the control signal to its extreme values; and
- it may actually be a more intuitive way to write rules.

The scalar can be a fuzzy set with the singleton placed in a proper position. For example 10 *volts*, would be equivalent to the fuzzy set (0, 0, 0, 0, 1), defined on the universe (-10, -5, 0, 5, 10) *volts.*

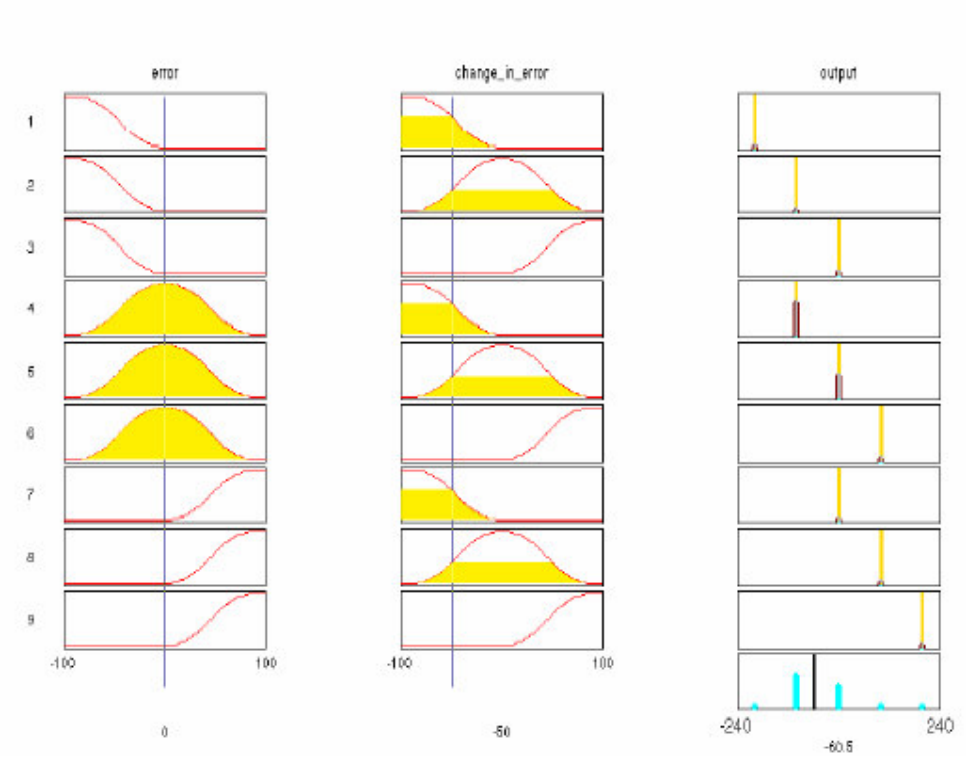


Figure 3.9. Graphical construction of the control signal in a fuzzy PD controller (generated in the Matlab Fuzzy Logic Toolbox).

Inference Engine

In Figure 3.9., each of the nine rows refers to one rule. For example, the first row says that if the *error* is negative (row 1, column 1) and the *change in error* is negative (row 1, column 2) then the output should be negative big (row 1, column 3). The picture corresponds to the rule base in (3.2.). The rules reflect the strategy that the control signal should be a combination of the reference error and the change in error, a fuzzy proportional-derivative controller. We shall refer to that figure in the following. The instances of the *error* and the *change in error* are indicated by the vertical lines on the first and second columns of the chart. For each rule, the inference engine looks up the membership values in the condition of the rule.

Defuzzification

The resulting fuzzy set (Figure 3.9., bottom right; Figure 3.10., extreme right) must be converted to a number that can be sent to the process as a control signal. This operation is called *defuzzification*, and in Figure 3.10. the *x*-coordinate marked by a white, vertical dividing line becomes the control signal. The resulting fuzzy set is thus defuzzified into a crisp control signal. There are several defuzzification methods.

Centre of gravity (COG): The crisp output value x (white line in Figure 3.10.) is the abscissa under the centre of gravity of the fuzzy set,

$$u = \frac{\sum_i \mu(x_i)\, x_i}{\sum_i \mu(x_i)} \tag{3.6}$$

Here x_i is a running point in a discrete universe, and $\mu(x_i)$ is its membership value in the membership function. The expression can be interpreted as the weighted average of the elements in the support set. For the continuous case, replace the summations by integrals. It is a much used method although its computational complexity is relatively high. This method is also called *centroid of area.*

Center of gravity method for singletons (COGS): If the membership functions of the conclusions are singletons (Figure 3.7.), the output value is

$$u = \frac{\sum_i \mu(s_i) s_i}{\sum_i \mu(s_i)} \tag{3.7}$$

Here s_i is the position of singleton *i* in the universe, and $\mu(s_i)$ is equal to the firing strength α_i of rule *i*. This method has a relatively good computational complexity, and *u* is differentiable with respect to the singletons s_i, which is useful in neurofuzzy systems.

Bisector of area (BOA): This method picks the abscissa of the vertical line that divides the area under the curve in two equal halves. In the continuous case,

$$u = \left\{ x \middle| \int_{Min}^{x} \mu(x)\, dx = \int_{x}^{Max} \mu(x)\, dx \right\} \tag{3.8}$$

Here x is the running point in the universe, $\mu(x)$ is its membership, *Min* is the leftmost value of the universe, and *Max* is the rightmost value. Its computational complexity is relatively high, and it can be ambiguous. For example, if the fuzzy set consists of two singletons any point between the two would divide the area in two halves; consequently it is safer to say that in the discrete case, BOA is not defined.

Mean of Maxima (MOM): An intuitive approach is to choose the point with the strongest possibility, i.e. maximal membership. It may happen, though, that several such points exist, and a common practice is to take the *mean of maxima* (MOM). This method disregards the shape of the fuzzy set, but the computational complexity is relatively good.

Leftmost maxima (LM), and rightmost maxima (RM): Another possibility is to choose the leftmost maximum (LM), or the rightmost maximum (RM). In the case of a robot, for instance, it must choose between left or right to avoid an obstacle in front of it. The defuzzifier must then choose one or the other, not something in between. These methods are indifferent to the shape of the fuzzy set, but the computational complexity is relatively small.

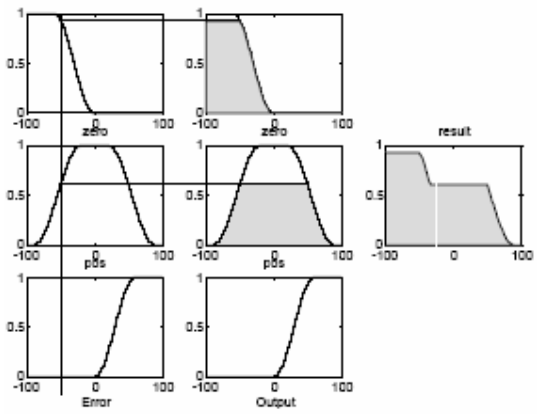


Figure 3.10. One input, one output rule base with non-singleton output sets

Postprocessing:

Output scaling is also relevant. In case the output is defined on a standard universe this must be scaled to *engineering units* for instance, volts, meters, or tons per hour. An example is the scaling from the standard universe [-1, 1] to the physical units [-10, 10]

volts. The postprocessing block often contains an output gain that can be tuned, and sometimes also an integrator.

3.2.2. Table Based Controller

If the universes are discrete, it is always possible to calculate all thinkable combinations of inputs before putting the controller into operation. In a *table based controller* the relation between all input combinations and their corresponding outputs are arranged in a table. With two inputs and one output, the table is a two-dimensional look-up table. With three inputs the table becomes a three-dimensional array. The array implementation improves execution speed, as the run-time inference is reduced to a table look-up which is a lot faster, at least when the correct entry can be found without too much searching. Below is a small example of a look-up table corresponding to the rulebase (2) with the membership functions in Figure 3.7.,

Table 3.2. Look-up table corresponding to the rulebase

		change in error				
		-100	-50	0	50	100
error	-100	-200	-160	-100	-40	0
	-50	-160	-121	-61	0	40
	0	-100	-61	0	61	100
	50	-40	0	61	121	160
	100	0	40	100	160	200

(3.9)

A typical application area for the table based controller is where the inputs to the controller are the *error* and the *change in error*.

Table regions: A negative value of *error* implies that the process output *y* is above the reference *Ref*, because the error is computed as *error* = *Ref* - *y* A positive value implies a process output below the reference. A negative value of *change in error* means that the process output increases while a positive value means it decreases.

Certain regions in the table are especially interesting. The centre of the table corresponds to the case where the *error* is zero, the process is on the reference.

Furthermore, the *change in error* is zero here, so the process stays on the reference. This position is the stable point where the process has settled on the reference. The anti-diagonal (orthogonal to the main diagonal) of the table is zero; those are all the pleasant states, where the process is either stable on the reference or approaching the reference. Should the process move away a little from the zero diagonal, due to noise or a disturbance, the controller will make small corrections to get it back. In case the process is far from the reference and also moving away from it, we are in the upper left and lower right corners. Here the controller calls for drastic changes.

The numerical values on the two sides of the zero diagonal do not have to be antisymetric; they can be any values, reflecting asymmetric control strategies. During a response with overshoot after a positive step in the reference, a plot of the point *(error, change in error)* will follow a trajectory in the table which spirals clockwise from the lower left corner of the table towards the centre. It is similar to a *phase plane* trajectory, where a variable is plotted against its derivative. A clever designer may adjust the numbers manually during a tuning session to obtain a particular response.

Bilinear interpolation If the resolution in the table is too coarse it will cause *limit cycling* that is, oscillations around the reference. The table allows the *error* to drift away from zero until it jumps into a neighboring cell with a nonzero control action. This can be avoided with *bilinear interpolation* between the cells instead of rounding to the nearest point. In the case of a two-dimensional table, an error E satisfies the relation $E_1 \le E \le E_2$ where E_1 and E_2 are the two neighboring points. The change-in-error CE will likewise satisfy $CE_1 \le CE \le CE_2$. The resulting table value is then found by interpolating linearly in the E axis direction between the first pair $u_1 = (F(E_1, CE_1), F(E_2, CE_1))$ and the second pair $u_2 = (F(E_1, CE_2), F(E_2, CE_2))$ and then in the CE-axis direction between the pair (u_1, u_2)

n-Dimensional Tables: A three input controller has a three-dimensional look-up table. Assuming a resolution of, say, 13 points in each universe, the table holds 2197 elements. It would be a tremendous task to fill these in manually, but it is manageable with rules. A three dimensional table can be represented as a two-dimensional table using a relational representation. Rearrange the table into three columns one for each of the three inputs (E_1, E_2, E_3) and one for the output (U) for example Table 3.1. Each input can take five values,

and the table thus has $5\times5\times5 = 125$ rows. The table look-up is now a question of finding the right row, and picking the corresponding U value.

Table 3.3. Equivalent of a 3D look-up table

E_1	E_2	E_3	U
−100	−100	−100	−100
−100	−100	−67	−89
−100	−100	0	−67
−100	−100	67	−178
−100	−100	100	−33
−100	−67	−100	−89
−100	−67	−67	−189
−100	−67	0	−56
...	...	...	...
100	100	100	100

3.2.3. INPUT-OUTPUT MAPPING

Two inputs and one output result in a two dimensional table, which can be plotted as a surface for visual inspection. The relationship between one input and one output can be plotted as a graph. These plots are a design aid when selecting membership functions and constructing rules. The shape of the surface can be controlled to a certain extent by manipulating the membership functions. In order to see this clearly, we will use the one-input-one-output case (without loss of generality). The fuzzy proportional rule base

If error is Neg then output is Neg
If error is Zero then output is Zero (3.10)
If error is Pos then output is Pos

produced the six different mappings in Figure 3.11. The rightmost column is the input-output mapping, and each row is a different controller. The controllers have the input families in the *if*-column and the output families in the *then* column. The results depend on the choice of design parameters, which in this case are the following: the * (*product*) operation for *activation* because it is continuous, the **max** operation for *accumulation* since it corresponds to set union, and *centre of gravity* for *defuzzification* since it is continuous, unambiguous, and it degenerates to COGS in the case of singleton output. If there had been two or more inputs, the * operation for **and** would be chosen since it is continuous. These choices are also necessary and sufficient for a linear mapping.

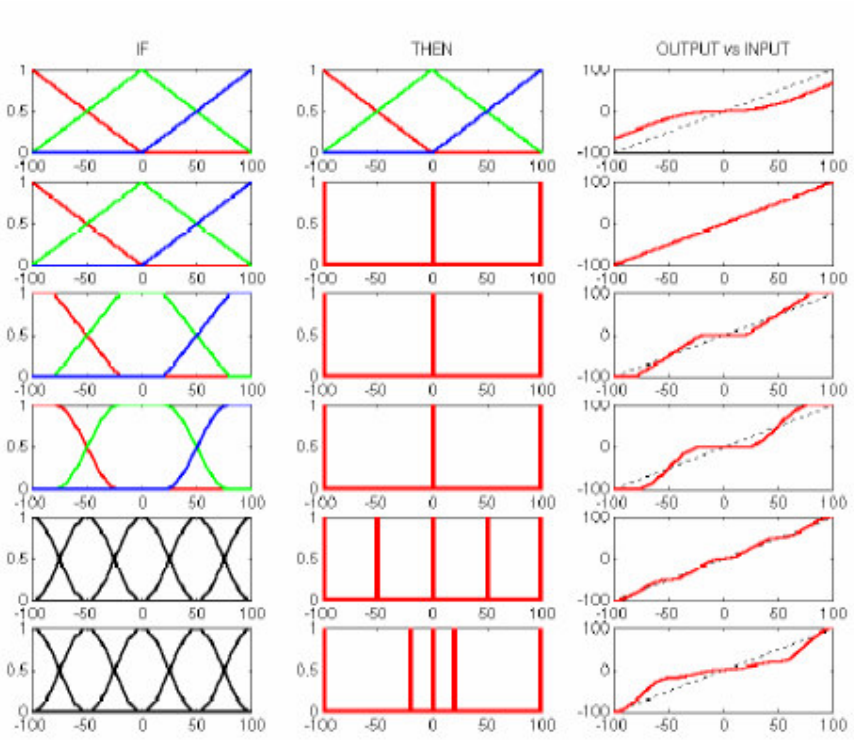


Figure 3.11. Input-Output maps of proportional controllers. Each row is a controller

The following comments relate to the figure, row by row:

1. Triangular sets in both condition and conclusion result in a winding input-output mapping. Compared to a linear controller (dotted line) the gain of the fuzzy controller varies. A slight problem with this controller is that it does not use the full output range; it is impossible to drive the output to 100%. Another problem is that the local gain is always equal or lower than the linear controller.

2. Singleton outputs eliminate the problem with the output range. The set **pos** corresponds to 100, **zero** to 0, and **neg** to -100. The input terms are the same as before. Now the input-output mapping is linear.

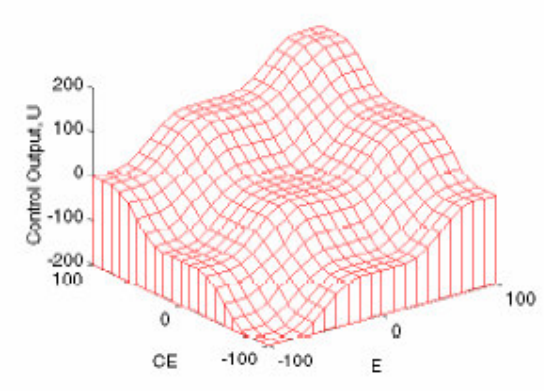


Figure 3.12. Example of control surface

3. Flat input sets produce flat plateaus and large gains far away from the reference. This is similar to a dead zone with saturation. Increasing the width of the middle term results in a wider plateau around the reference. Less overlap between neighboring sets will result in steeper slopes.
4. If the sharp corners cause problems, they are removed by introducing nonlinear input sets. The input-output relationship is now smooth.
5. Adding more sets only makes the mapping more bumpy.
6. On the other hand with more sets it is easier to stretch the reference plateau by moving the singletons about.

The experiment shows that depending on what the design specifications are, it is possible to control, to a certain extent, the variation of the gain. Using singletons on the output side makes it easier. The results can easily be generalised to three dimensional surfaces. In all cases the activation operator was * (product), the accumulation operator was **max** and the defuzzification method was **COG** or **COGS** other operations may give slightly different results.

Control Surface: With two inputs and one output the input-output mapping is a surface. Figure 3.12., is a mesh plot of an example relationship between *error E* and *change in error CE* on the input side, and controller output *u* on the output side. The plot results from a rule base with nine rules, and the surface is more or less bumpy. The horizontal plateaus are due to flat peaks on the input sets. The plateau around the origin implies a low sensitivity towards changes in either *error* or *change in error* near the reference. This is an

advantage if noise sensitivity must be low when the process is near the reference. On the other hand, if the process is unstable in open loop it is difficult to keep the process on the reference, and it will be necessary to have a larger gain around the origin.
There are three sources of nonlinearity in a fuzzy controller.

- *The rule base* The position, shape and number of fuzzy sets as well as nonlinear input scaling cause nonlinear transformations. The rules often express a nonlinear control strategy.
- *The inference engine* If the connectives **and** and **or** are implemented as for example **min** and **max**[respectively, they are nonlinear.

- *The defuzzification* Several defuzzification methods are nonlinear.

It is possible to construct a rule base with a linear input-output mapping [30, 31, 32]. The following checklist summarises the general design choices for achieving a fuzzy rule base equivalent to a summation (details in the appendix):

1. Use triangular input sets that cross at $\mu = 0.5$
2. Use the algebraic product (*) for the **and** connective;
3. The rule base must be the complete **and** combination (cartesian product) of all input families;
4. Use output singletons, positions determined by the sum of the peak positions of the input sets;
5. Use *COGS* defuzzification.

With these design choices the control surface degenerates to a diagonal plane (Figure 3.13.). A flexible fuzzy controller, that allows these choices, is two controllers in one so to speak. When linear, it has a transfer function and the usual methods regarding tuning and stability of the closed loop system apply.

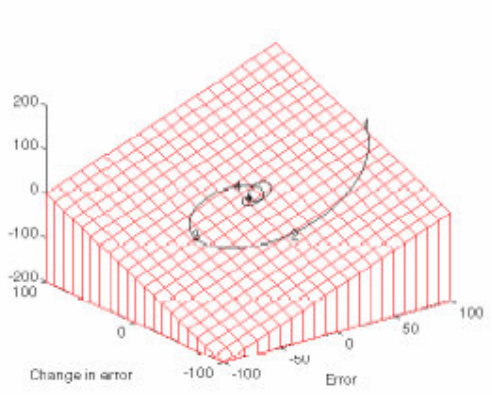


Figure 3.13. Linear surface with trajectory of a transient response

3.2.4. Takagi-Sugeno Type Controller

We saw that the output sets can be singletons, but they can also be linear combinations of the inputs, or even a function of the inputs [5]. The general Takagi-Sugeno rule structure is

$$If\ (e_1\ is\ A_1, e_2\ is\ A_2, ..., e_k\ is\ A_k)\ then\ y = g(e_1, e_2 ....) \tag{3.11}$$

Here f is a logical function that connects the sentences forming the condition, y is the output, and g is a function of the inputs. A simple example is

If error is Zero and change in error is Zero then output $y = c$

where c is a crisp constant. This is a *zero-order* model, and it is identical to singleton output rules. A slightly more complex rule is

If error is Zero and change in error is Zero then output

$$y = a*error + b*(changeinerror) + c$$

where *a, b* and *c* are all constants. This is a *first-order* model. Inference with several rules proceeds as usual, with a firing strength associated with each rule, but each output is linearly dependent on the inputs. The output from each rule is a moving singleton, and the

defuzzified output is the weighted average of the contributions from each rule. The controller interpolates between linear controllers; each controller is dominated by a rule, but there is a weighting depending on the overlap of the input membership functions. This is useful in a nonlinear control system, where each controller operates in a subspace of the operating envelope. One can say that the rules interpolate smoothly between the linear gains. Lastly, Higher order models are also possible.

3.3 PID Type Fuzzy Control

In this study, a controller with two-input and one-output (MISO) is used. The inputs are the error, e, and the change rate of error, $\dot{e}$. The linguistic values of inputs are denoted as A_i and B_j, respectively. These membership functions are referred to as $Ai(e)$ and $Bj(\dot{e})$. The centre of gravity method is applied in the defuzzification process to obtain the controller output, u. An example fuzzy controller rule is expressed below:

If e is A_i and $\dot{e}$ is B_j , then u is u_{ij}.

Here, all linguistic expressions of the error and the change rate of error are expressed with triangular membership functions as seen in Figure 3.14. The cores of fuzzy sets A_i and B_j are denoted as e_i and $\dot{e}_j$, respectively. The supporting set of A_i is shown as $[e_{i-1}, e_{i+1}]$ and the set of B_j is $[\dot{e}_{j-1}, \dot{e}_{j+1}]$.

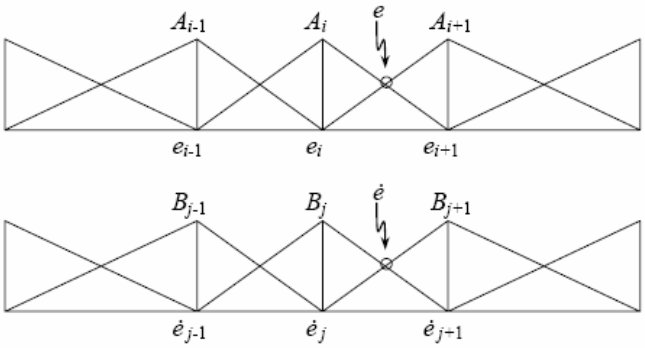


Figure 3.14 Membership functions of A_i and B_j

Each node on the e and $\dot{e}$ plane, as indicated in Figure 3.15, represents an output of the controller, which is a nonlinear function. There is no exact analytical solution for this kind of nonlinear structure. Besides, with the aid of the linearization method applied in

classical and modern control theory, for the small deviations about nominal values of e, $\dot{e}$ and u, an approximation can be done.

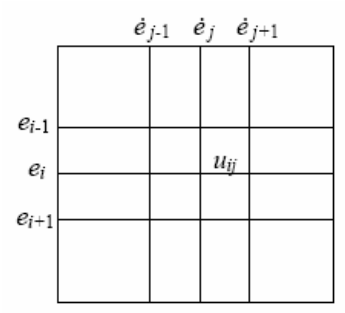


Figure 3.15. Network structure of (e, $\dot{e}$) plane

The relation between the input and output of the controller designed to control a nonlinear model is below:

$$u = f(e, \dot{e}, t) \tag{3.12}$$

When $e = e_i$ and $\dot{e} = \dot{e}_j$, in other words, if the node is (e_i, $\dot{e}_j$), then the output of the controller will be,

$$u = u_{ij} \tag{3.13}$$

According to this, the output will become

$$u = f(e_i, \dot{e}_j, t) = u_{ij} \tag{3.14}$$

A linearization can be done around a node u_{ij} of the $e - \dot{e}$ plane for small excursions from e, $\dot{e}$ and u as

$$\begin{aligned} \delta e &= e - e_i \\ \delta \dot{e} &= \dot{e} - \dot{e}_j \\ \delta u &= u - u_{ij} \end{aligned} \tag{3.15}$$

Consequently if the values of δe, $\delta \dot{e}$ and δu are small enough, Equation (3.12) can be assumed as a linear equation,

$$\delta u = \left[\frac{\delta f}{\delta e}\right]_n \delta e + \left[\frac{\delta f}{\delta \dot{e}}\right]_n \delta \dot{e} \tag{3.16}$$

Every neighbourhood of each node-*n* (a nominal point) will be divided to four different quadrants. For simplicity, only the case of the first quadrant is presented:

$$\text{If } \delta e \geq 0 \text{ and } \delta \dot{e} \geq 0 \text{ then } (e_i + \delta e, \dot{e}_j + \delta \dot{e}) \in [e_i, e_{i+1}] \times [\dot{e}_j, \dot{e}_{j+1}] \tag{3.17}$$

$$\delta u = \left[\frac{\delta f}{\delta e}\right]_n \delta e + \left[\frac{\delta f}{\delta \dot{e}}\right]_n \delta \dot{e} = \frac{u_{(i+1)j} - u_{ij}}{e_{i+1} - e_i} \delta e + \frac{u_{i(j+1)} - u_{ij}}{\dot{e}_{j+1} - \dot{e}_j} \delta \dot{e} \tag{3.18}$$

$$u - u_{ij} = \frac{u_{(i+1)j} - u_{ij}}{e_{i+1} - e_i}(e - e_i) + \frac{u_{i(j+1)} - u_{ij}}{\dot{e}_{j+1} - \dot{e}_j}(\dot{e} - \dot{e}_j \tag{3.19}$$

Consequently the equation below is obtained.

$$u = \left[\ u_{ij} - \frac{u_{(i+1)j} - u_{ij}}{e_{i+1} - e_i} e_i - \frac{u_{i(j+1)} - u_{ij}}{\dot{e}_{j+1} - \dot{e}_j} \dot{e}_j \ \right] + \frac{u_{(i+1)j} - u_{ij}}{e_{(i+1)} - e_i} e + \frac{u_{i(j+1)} - u_{ij}}{\dot{e}_{j+1} - \dot{e}_j} \dot{e} \tag{3.20}$$

The right hand side of the equation is in the form of

$$A + Pe + D\dot{e} \tag{3.21}$$

where

$$A = \left[u_{ij} - \frac{u_{(i+1)i} - u_{ij}}{e_{(i+1)} - e_i} e_i - \frac{u_{i(j+1)} - u_{ij}}{\dot{e}_{j+1} - \dot{e}_j} \dot{e}_j \right] = u_{ij} - Pe_i - D\dot{e}_j$$

$$P=\frac{u_{(i+1)i}-u_{ij}}{e_{(i+1)}-e_i}$$

$$D=\frac{u_{i(j+1)}-u_{ij}}{\dot{e}_{j+1}-e_j} \tag{3.22}$$

As it is seen, the fuzzy controller designed behaves like a PD controller. If the controlled system is type "0", P or PD type controller cannot eliminate the steadystate error. Although PI controller improves theu steady-state error, it can deteriorate the transient characteristics, i.e. it slows the response down. Three input variables such as the error, the change rate of error and the integral of error are required to design a PID controller to get desired responses in terms of improving the transient characteristics and the steady-stare error. These three variables are also needed when PID type fuzzy controller is concerned. Handling the three variables however, in practice, is quite difficult. When the rules are defined, an operator's manual control experiences are utilized. An operator can manipulate a control process intuitively by observing the error and the change rate of error but s/he cannot observe the integral of error. Besides, adding another input to the controller will increase the number of rules exponentially. This causes more computational effort and a larger time consumption. Because of these drawbacks, a PID type fuzzy controller consisting of only the error and the change rate of error is used in the method. This system allows a PD and PI type fuzzy controller to work in parallel [32]. An equivalent structure is shown in Figure 16, where β and α are the weights of PI and PD type controllers, respectively [33]. As the α / β ratio becomes larger, the effect of the derivative control increases with respect to integral control. Using Equation (3.21), the output of the controller can be expressed as,

$$\begin{aligned} u_c &= \alpha u + \beta \int u\,dt \\ &= \alpha(A + PK_1 e + DK_2 \dot{e}) + \beta \int (A + PK_1 e + DK_2 \dot{e})\,dt \\ &= \alpha A + \beta At + (\alpha K_1 P + \beta K_2 D)e + \beta K_1 P \int e\,dt + \alpha K_2 D\dot{e} \end{aligned} \tag{3.23}$$

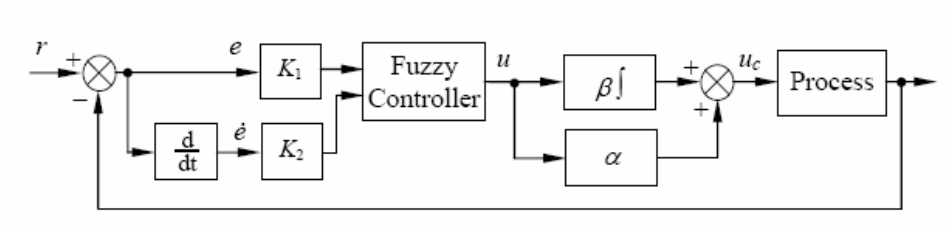


Figure 3.16. PID type fuzzy controller structure

Consequently, the fuzzy controller behaves like a PID controller. The equivalent proportional, integral and derivative components are $\alpha K_1 P + \beta K_2 D$, $\beta K_1 P$ and $\alpha K_2 D$. This controller is referred to as PID type fuzzy controller.

3.4. ANFIS: Adaptive-Network-Based Fuzzy Inference System

Several types of fuzzy reasoning [25] have been proposed in the literature. Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types (see Figure 3.17.):

Type 1:

The overall output is the weighted average of each rule's crisp output induced by the rule's firing strength (the product or minimum of the degrees of match with the premise part) and output membership functions. The output membership functions used in this scheme must be monotonic functions

Type 2:

The overall fuzzy output is derived by applying "max" operation to the qualified fuzzy outputs (each of which is equal to the minimum of firing strength and the output membership function of each rule). Various schemes have been proposed to choose the final crisp output based on the overall fuzzy output; some of them are centroid of area, bisector of area, mean of maxima, maximum criterion, etc [25].

Type 3:

Takagi and Sugeno's fuzzy if-then rules are used. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output. Figure 3.17 [8] utilizes a two-rule two-input fuzzy inference system to show different types of fuzzy rules and fuzzy reasoning mentioned above. Be aware that most of the differences come from the specification of the consequent part (monotonically non-decreasing or bell-shaped membership functions, or crisp function) and thus the defuzzification schemes (weighted average, centroid of area, etc) are also different.

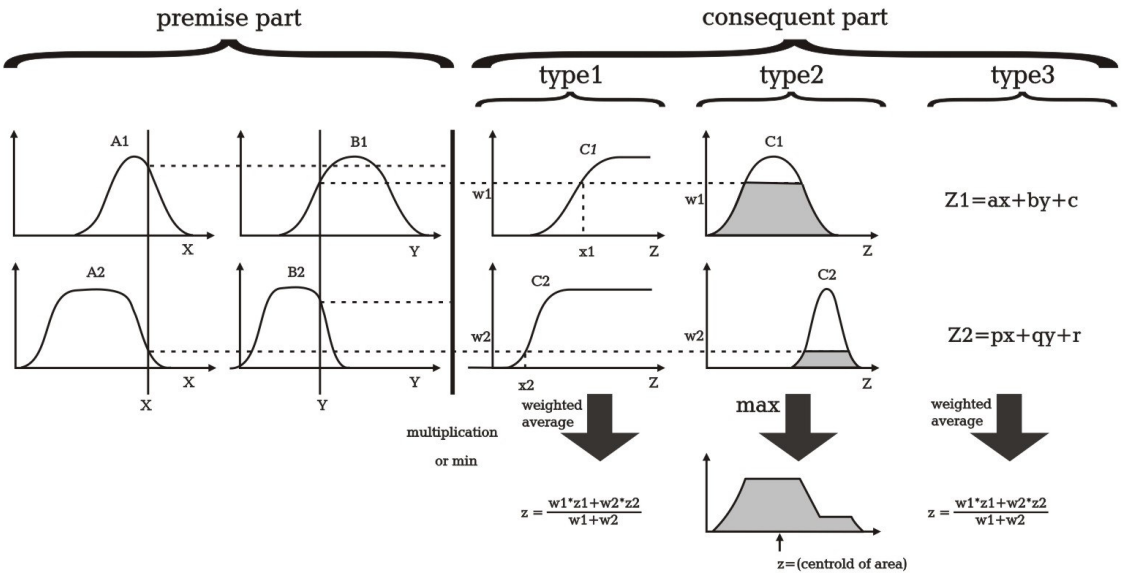


Figure 3.17. Commanly used fuzzy if-then rules and fuzzy reasoning mechanism

A first order Sugeno model can be expressed with two rules as follows and the related inference method is shown in Figure 3.18.

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1 x + q_1 y + r_1$

Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2 x + q_2 y + r_2$

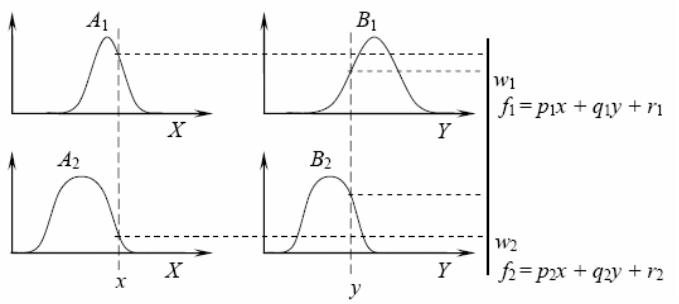


Figure 3.18. The inference method of Sugeno model

Here, x and y inputs constitute the premise variables of the fuzzy rules. $A1$, $A2$, $B1$and $B2$ are the premise parameters. $p1$, $p2$, $q1$, $q2$, $r1$ and $r2$ are the consequence parameters. Using the $f1$ and $f2$ functions, which are combinations of consequent parameters, and the weight values (now referred to as firing strengths) of $w1$ and $w2$, the overall output function, f, for this Sugeno model is expressed as,

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \overline{w_1} f_1 + \overline{w_2} f_2$$

From Sugeno Fuzzy Model, Adaptive Neural-Fuzzy Inference System (ANFIS) was proposed by Roger Jang in 1993 [8]. and the corresponding ANFIS model for Sugeno's fuzzy structure is given in Figure 3.19.

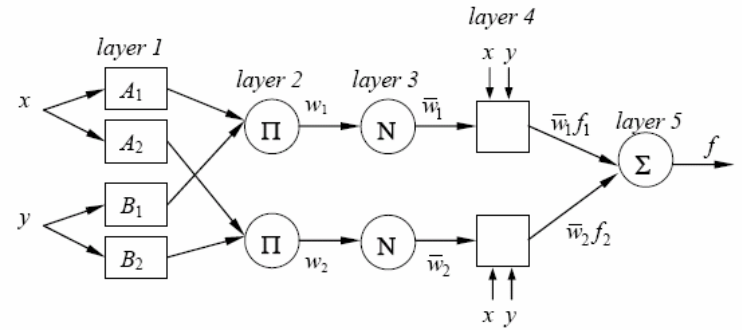


Figure 3.19. The ANFIS model using Sugeno's fuzzy inference method for two rules (Π: multiplies the incoming signals and sends the product out, N: normalizes the weights, Σ: sums up the incoming signals and produces the overall output).

Functionally, there are almost no constraints on the node functions of an adaptive network except piecewise differentiability. Structurally, the only limitation of network configuration is that it should be of feedforward type. Due to these minimal restrictions, the adaptive network's applications are immediate and immense in various areas. In this section, a class of adaptive networks which are functionally equivalent to fuzzy inference systems are used. The architecture is referred to as ANFIS, standing for adaptive-network-based fuzzy inference system.

3.4.1. ANFIS Architecture

For simplicity, we assume the fuzzy inference system under consideration has two inputs x and y and one output z. suppose that the rule base contains two fuzzy if-then rules of Takagi and Sugeno's type.

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1 x + q_1 y + r_1$

Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2 x + q_2 y + r_2$

Then the type-3 fuzzy reasoning is illustrated in Figure 3.18, and the corresponding equivalent ANFIS architecture *(type-3*ANFIS) is shown in Figure 3.19. The node functions in the same layer are of the same function family as described below:

Layer 1: Every node i in this layer is a square node with a node function

$$O_i^1 = \mu_{A_i}(x)$$

where x is the input to node i, and A_i is the linguistic label *(small, large,* etc.) associated with this node function. In other words, O_i^1 is the membership function of A_i and it specifies the degree to which the given $\boldsymbol{x}$ satisfies the quantifier A_i. Usually we choose $\mu_{A_i}(x)$ to be bell-shaped with maximum equal to 1 and minimum equal to 0, such as

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right] b_i}$$

or

$$\mu_{A_i}(x) = \exp\left\{-\left(\frac{x-c_i}{a_i}\right)^2\right\}$$

Layer 2: Every node in this layer is a circle node labeled $\prod$ which multiplies the incoming signals and sends the product out. For instance,

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), \quad i = 1, 2.$$

Each node output represents the firing strength of a rule. (In fact, other *T-norm* operators that perform generalized AND can be used as the node function in this layer.)

Layer 3: Every node in this layer is a circle node labeled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strengths:

$$\overline{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2.$$

For convenience, outputs of this layer will be called *normalized firing strengths.*

Layer 4: Every node *i* in this layer is a square node with a node function

$$O_i = \overline{w}_i f_i = \overline{w}_i \left(p_i x + q_i y + r_i\right)$$

where $\overline{w}_i$ is the output of layer 3, and *{pi, q_i, ri}* is the parameter set. Parameters in this layer will be referred to as *consequent parameters.*

Layer 5: The single node in this layer is a circle node labeled $\sum$ that computes the overall output as the summation of all incoming signals, i.e.,

$$O_1^5 = overall\ output = \sum_i w_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

Thus we have constructed an adaptive network which is functionally equivalent to a type-3 fuzzy inference system. For type–1 fuzzy inference systems, the extension is quite straightforward and for type–1 ANFIS the output of each rule is induced jointly by the output membership funcion and the firing strength. For type–2 fuzzy inference systems, if we replace the centroid defuzzification operator with a discrete version which calculates the approximate centroid of area, then type–3 ANFIS can still be constructed accordingly. However, it will be more complicated than its type–3 and type–1 versions and thus not worth the efforts to do so.

3.4.2. *Hybrid Learning Algorithm*

From the type-3 ANFIS architecture (see Figure 3.19.) it is observed that given the values of premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output ***f*** in Figure 3.19 can be rewritten as

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \overline{w}_1 f_1 + \overline{w}_2 f_2 \\ &= (\overline{w}_1 f_1) p_1 + (\overline{w}_1 y) q_1 + (\overline{w}_1) r_1 + (\overline{w}_2 x) p_2 + (\overline{w}_2 y) q_2 + (\overline{w}_2) r_1 \end{aligned}$$

which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2, r_2).

In the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. Table 3.2. summarizes the activities in each pass.

Table 3.4. Two passes in the hybrid learning procedure for ANFIS

	Forward Pass	Backward Pass
Premise Parameters	Fixed	Back Propagation
Consequent Parameters	Recursive-Least Squares Estimate	Fixed
Signals	Node Outputs	Error Signals

The hybrid approach is much faster than the strict gradient descent. However, it should be noted that the computation complexity of the least squares estimate is higher than that of the gradient descent. In fact, there are four methods to update the parameters, as listed below according to their computation complexities:

1) ***Gradient Descent Only:*** All parameters are updated by the gradient descent.

2) ***Gradient Descent and One Pass of LSE:*** The LSE is applied only once at the very beginning to get the initial values of the consequent parameters and then the gradient descent takes over to update all parameters.

3) ***Gradient descent and LSE: This*** is the hybrid learning rule.

4) ***Sequential (Approximate) LSE Only:*** The ANFIS is linearized w.r.t. the premise parameters and the extended Kalman filter algorithm is employed to update all parameters. This has been proposed in the neural network literature [34-36]. The choice of above methods should be based on the trade-off between computation complexity and resulting performance.

4. SIMULATION RESULTS

4.1 The System

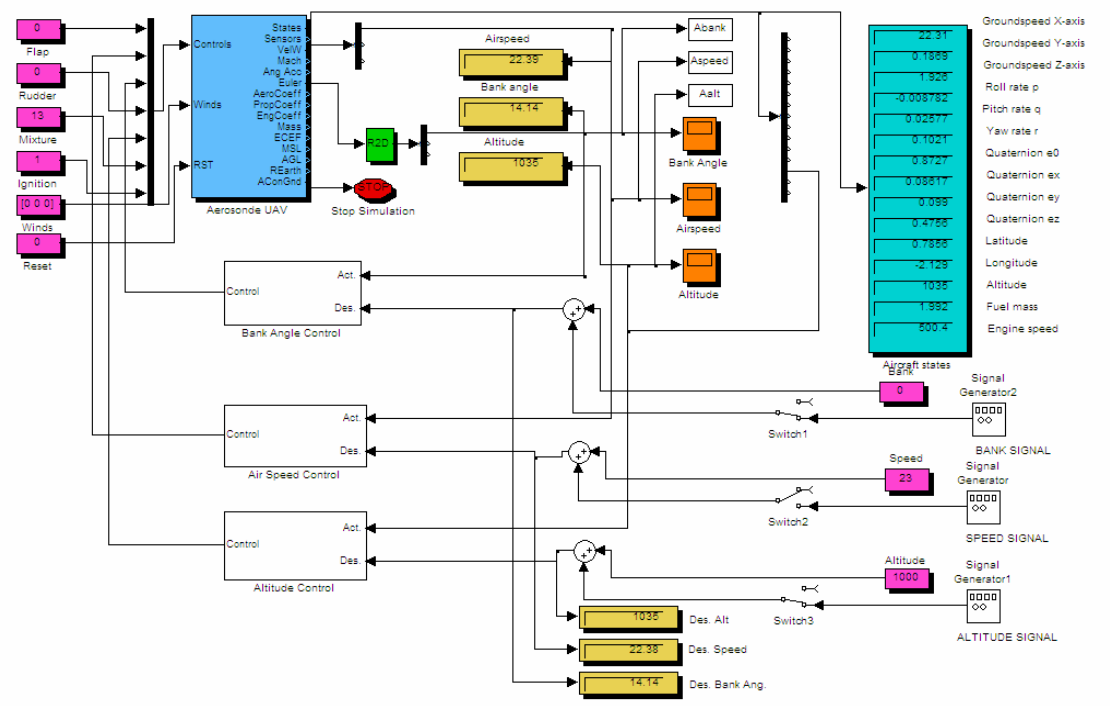


Figure 4.1. Aerosonde UAV control system block diagram

The aerial vehicle, in Figure 4.1., considered in this thesis as the test vehicle is a 6 degree-of-freedom nonlinear aerial vehicle, called Aerosonde UAV, for which a model is provided in *Aerosim Aeronautical Simulation Block*. The performance and potential of the control approaches were evaluated by using *MATLAB*'s standard configuration and the *Aerosim Aeronautical Simulation Block Set,* the aircraft simulated being *Aerosonde UAV.* Simulation studies are performed using the sinusoidal references given below:

$$[x_d(t)] = \begin{bmatrix} x_{d1}(t) \\ x_{d2}(t) \\ x_{d3}(t) \end{bmatrix} = \begin{bmatrix} 20\sin(0.0005\pi t) \\ 23 + 5\sin(0.0005\pi t) \\ 1000 + 50\sin(0.0005\pi t) \end{bmatrix}$$

Where $X_{d1}(t)$, $X_{d2}(t)$ and $X_{d3}(t)$ represent desired bank angle, airspeed and altitude, respectively.

In Conventional PID control method, all the control gains were found by trial-error. In bank angle control, better results were obtained with PI control rather than PID. Proportional control gain was chosen as -0.5 and integral control gain was -0.05. In air-speed control, proportional, integral and derivative control gains were 0.025, 0.005 and 0.5 respectively. Lastly, in altitude control, gains were chosen as 1, 0.01 and 0.1

In PID type fuzzy control method, three fuzzy logic controllers were designed for Aerosonde in order to control the altitude and the latitude-longitude as shown below. These three controllers acting in combination enable the navigation of the aerial vehicle (Figure 4.2.)

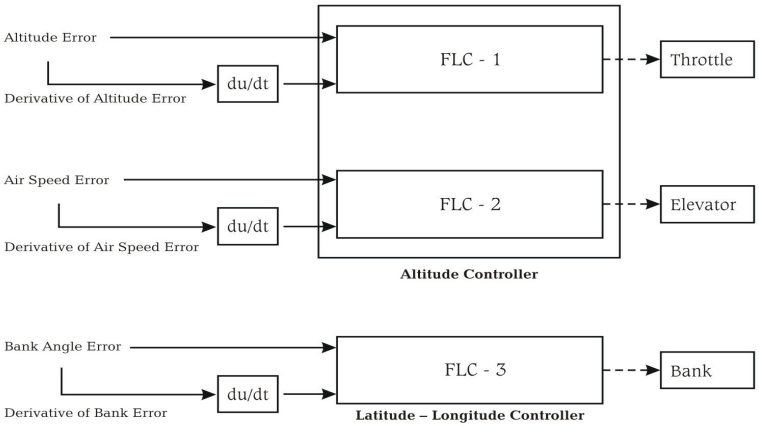


Figure 4.2. Fuzzy Logic Controllers

"The altitude controller has two sub-control systems, namely the throttle controller and the elevator controller. The throttle fuzzy logic controller has two inputs: the altitude error (i.e. the difference between the desired altitude and the actual altitude) and its rate of change. The latter indicates whether the UAV is approaching to the desired altitude or diverging away. Like the throttle controller, the elevator control has two inputs, the air speed error and its derivative. The control outputs of the block are the throttle and the elevator respectively, responsible for the head going up or down [18]." The objective of the control was to minimize the difference between the reference signals and the closed loop system actual responses. Hence we could define the error signal as performance criteria.

In design stage of PID type fuzzy controllers for bank angle control, number of membership functions for 2 inputs and 1 output were chosen as 7. All the MFs were symmetric and triangular. Fuzzy rule-base was constructed as below,

Table 4.1 Rule Base of PID Type Fuzzy Controller for Bank Angle

edot \ e	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	Z
NM	NB	NM	NM	NM	NS	Z	PS
NS	NB	NM	NS	NS	Z	PS	PM
Z	NB	NM	NS	Z	PS	PM	PB
PS	NM	NS	Z	PS	PS	PM	PB
PM	NS	Z	PS	PM	PM	PM	PB
PB	Z	PS	PS	PM	PB	PB	PB

In Air-speed control, each MFs were 5 and symmetric triangular. Rule-table is given below,

Table 4.2 Rule Base of PID Type Fuzzy Controller for Air-Speed

edot \ e	NB	NS	Z	PS	PB
NB	NB	NB	NS	NS	Z
NS	NB	NS	NS	Z	PS
Z	NS	NS	Z	PS	PS
PS	NS	Z	PS	PS	PB
PB	Z	PS	PS	PB	PB

Altitude control was also realized with 7 symmetric triangular functions. The table formed with forty-nine rule is below,

Table 4.3 Rule Base of PID Type Fuzzy Controller for Altitude

edot \ e	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	Z
NM	NB	NM	NM	NM	NS	Z	PS
NS	NB	NM	NS	NS	Z	PS	PM
Z	NB	NM	NS	Z	PS	PM	PB
PS	NM	NS	Z	PS	PS	PM	PB
PM	NS	Z	PS	PM	PM	PM	PB
PB	Z	PS	PS	PM	PB	PB	PB

In ANFIS control structure, triangular membership functions were chosen to represent the inputs which were error and change rate of error respectively. First, desired error tolerance was given to the system instead of epoch number, and the proper epoch number was left to ANFIS. Error tolerance values between 0.1–0.00000001 are tried but no satisfactory results can be achieved. Then the method was changed and epoch numbers were given to the system instead of error tolerance. Also the epoch numbers were adjusted by trial and error method. All the epoch numbers between 1 and 10 were tried. But the learning performance was always changing in an unexpected manner. Actually, it could be explained by over learning or local minima. After 7 epochs, the error extremely decreased. But small error does not mean that ANFIS learned better. While the epoch numbers were changed, data pair numbers were also changed. And system was reloaded with this set and trained again. As appreciated, it took for a long time and sometimes became an exhausting activity. All the reference signals are given in first part of this section. Their periods are 2000seconds. However, in training procedure we took the same signals but a different period. All the signal periods were 1000 seconds and one-fourth of these data sets are used as the training set.

Each learning process was starting from a random initial value because of that even with the same parameters; system would present different results which made the process longer. Furthermore, inadequate CPU's were also extended the period. In this thesis, every parameters such as number of membership functions, epoch numbers and input-output data set were interacting because of that only a small change in any of the parameters were effecting the choice of the others. Due to these facts, finding optimum values of the

parameters for learning procedure took a long time. All the membership functions and the rule base provided by ANFIS are given below,

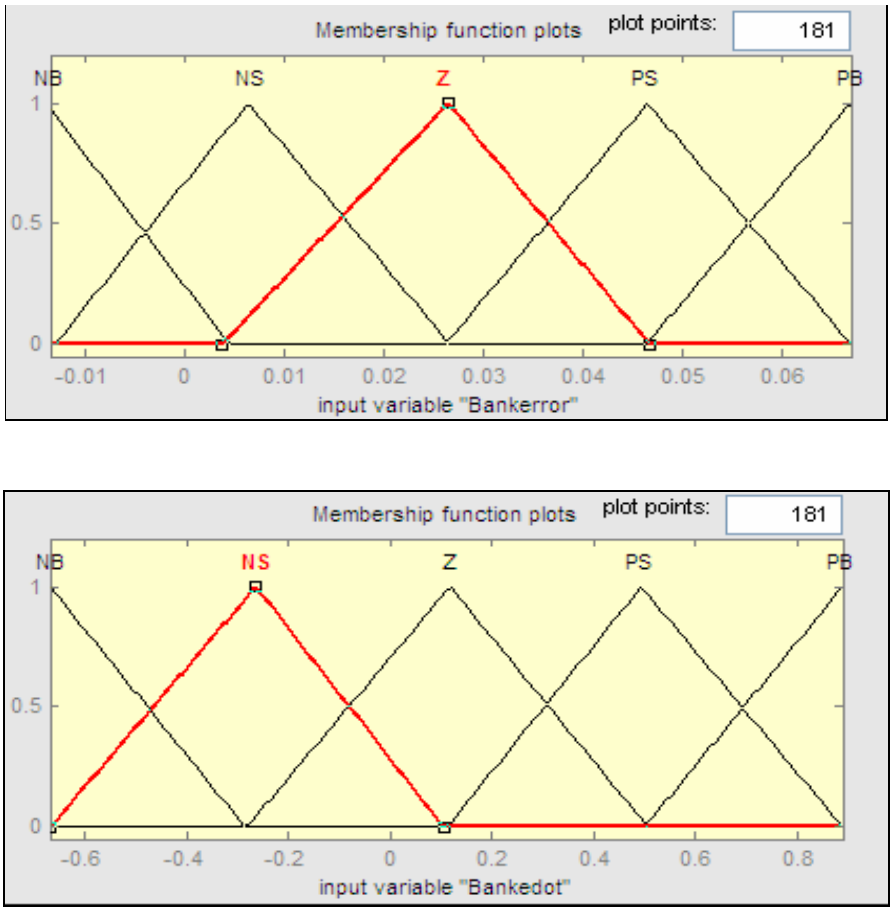
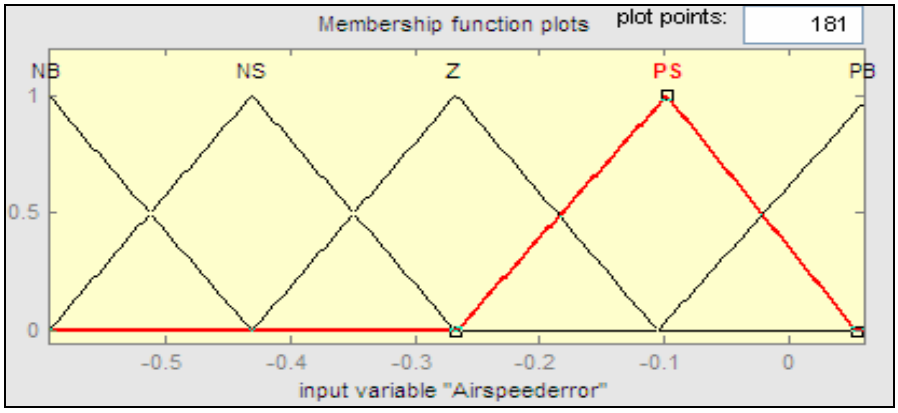


Figure 4.3. Membership Functions of ANFIS Bank Angle Controller



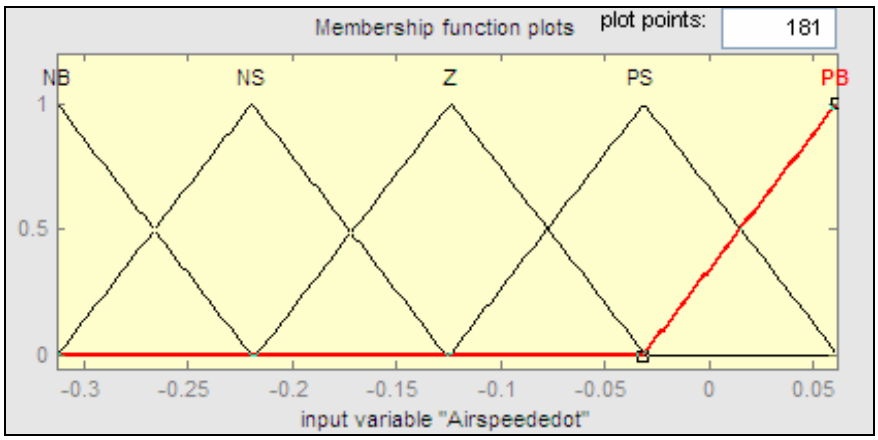
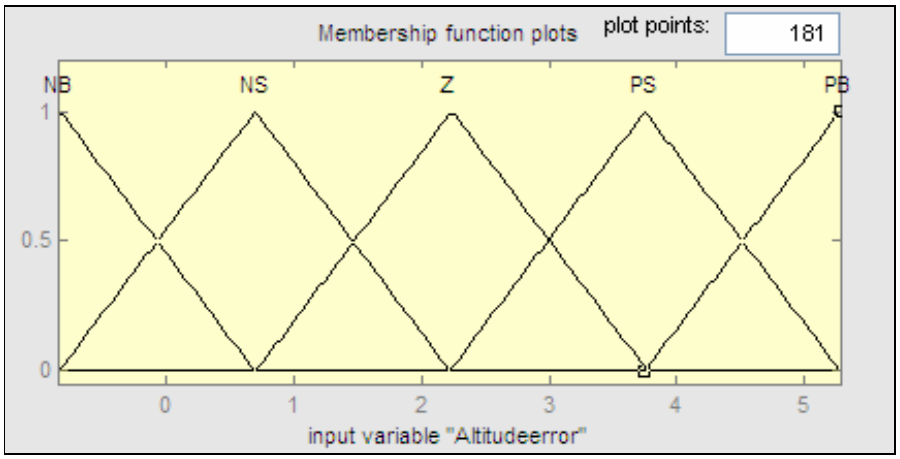


Figure 4.4. Membership Functions of ANFIS Air-Speed Controller



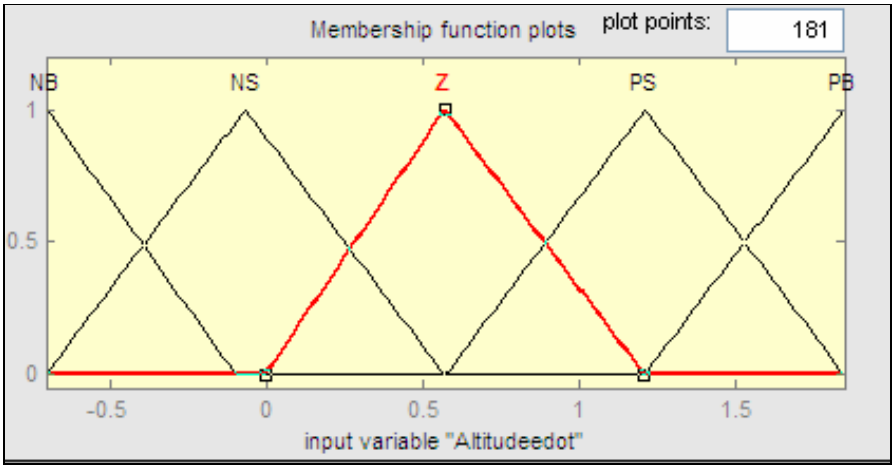


Figure 4.5. Membership Functions of ANFIS Altitude Controller

Table 4.4 Rule-Base Constructed by ANFIS

BANK ANGLE

edot \ e	NB	NS	Z	PS	PB
NB	-0.00165	-0.001613	-0.001595	-0.001599	-0.001616
NS	-0.0007721	-0.0007786	-0.0007464	-0.0007196	-0.0007175
Z	0.0003008	0.000316	0.0003292	0.0003247	0.0003204
PS	0.00138	0.00131	0.001381	0.001272	0.001344
PB	0	0.00157	0.00203	0.002037	0.001979

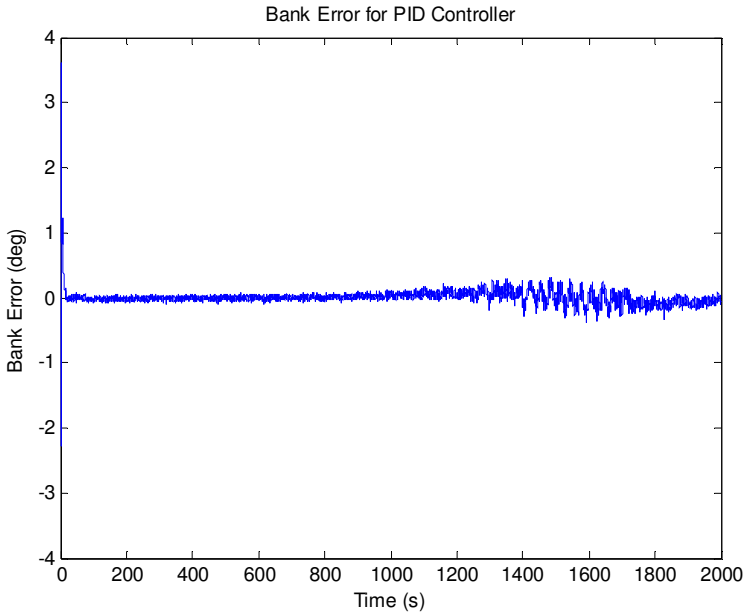
AIRSPEED

edot \ e	NB	NS	Z	PS	PB
NB	-0.1371	-0.1338	-0.1267	-0.1238	-0.1208
NS	-0.1425	-0.1438	-0.103	-0.1094	-0.1166
Z	-0.1444	-0.1688	-0.06039	-0.08026	-0.02357
PS	-0.1555	-0.1519	-0.1457	-0.1377	0.06566
PB	-0.1269	-0.1081	-0.07577	0.03462	0.01514

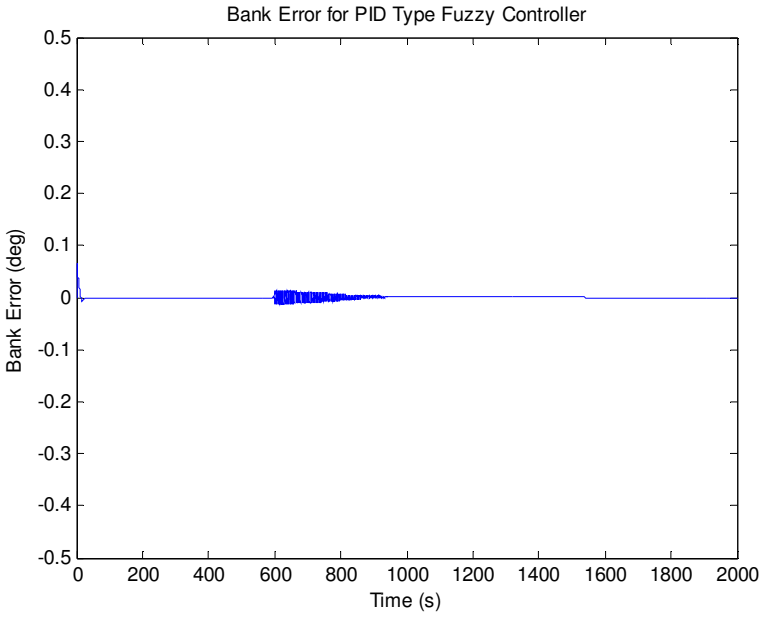
ALTITUDE

edot \ e	NB	NS	Z	PS	PB
NB	-0.002003	-0.001186	-0.0002356	0.0002826	0.0003464
NS	-0.0005489	0.0004269	0.004185	0.001604	0.002265
Z	0.0008969	0.001501	0	0.0001933	0.002944
PS	0.002107	0.002236	0.002259	0.001855	0.002843
PB	0.00383	0.003711	0.003879	0.003905	0.004096

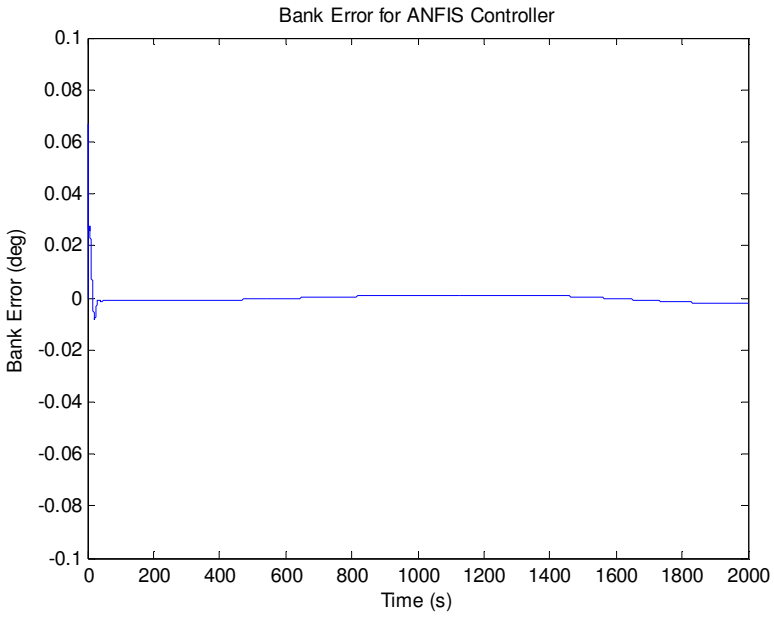
4.2 Simulation Results of Bank Angle Control



(a)



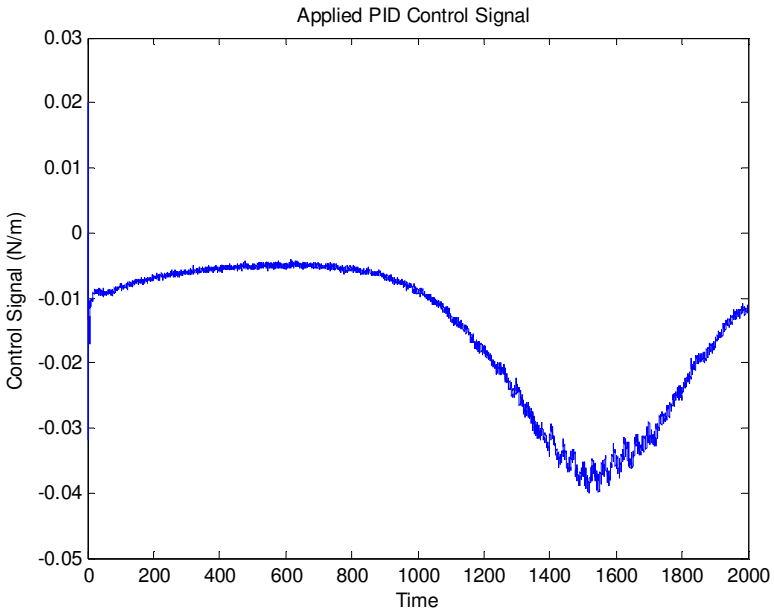
(b)



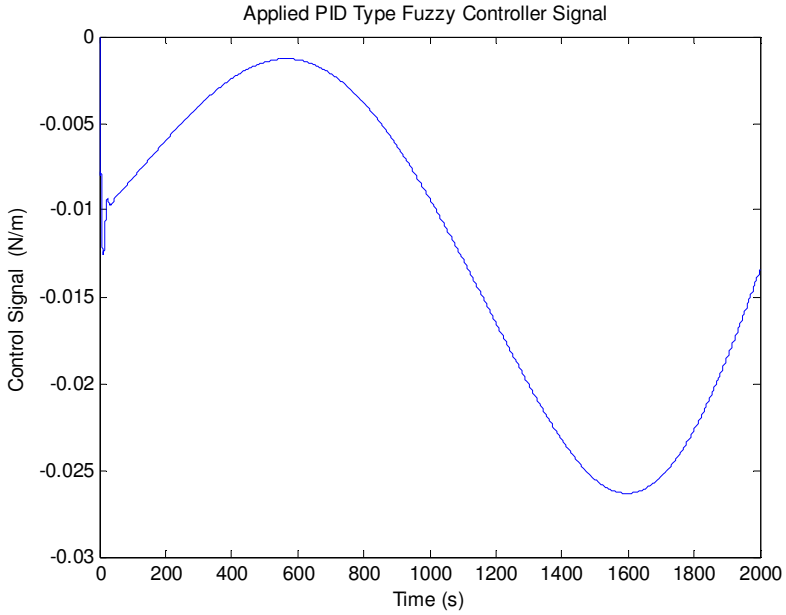
(c)

Figure 4.6. Bank Errors of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

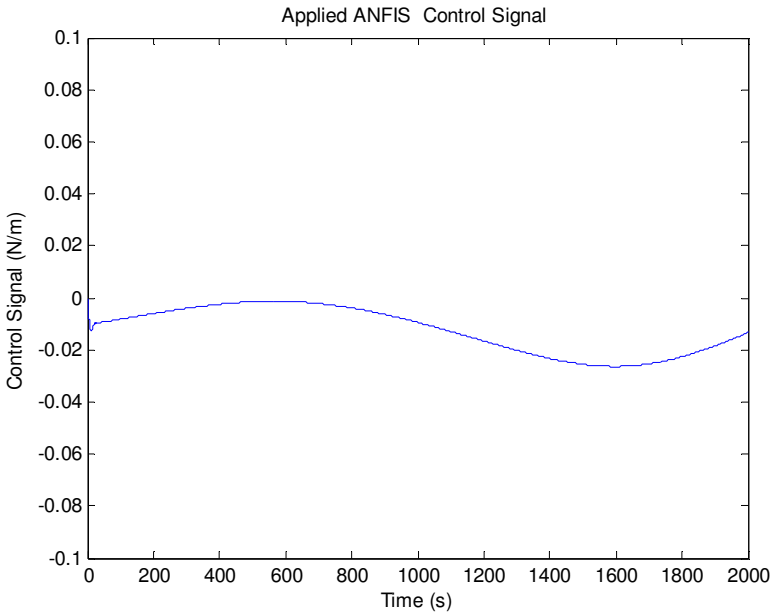
In Figure 4.6.(a) the error signal is not very smooth and the initial error value is 3.6 but steady state error is in an acceptable range of [0.3,-0.3]. And settling time can be considered as 20 sec In Figure 4.6.(b) error signal is not still very smooth but very small and the range of error signal is changing with in range of [0.07,-0.0015]. the initial error value is 0.07 and settling time of the system is 40sec. In Figure 4.6.(c) initial error value is 0.06 and the steady state error is changing in [0.0012, -0.002] in addition the signal is quite smooth and settling time of the system is approximately 50 sec.


Applied PID Control Signal
Control Signal (N/m)
Time
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(a)


Applied PID Type Fuzzy Controller Signal
Control Signal (N/m)
Time (s)
0
-0.005
-0.01
-0.015
-0.02
-0.025
-0.03
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(b)

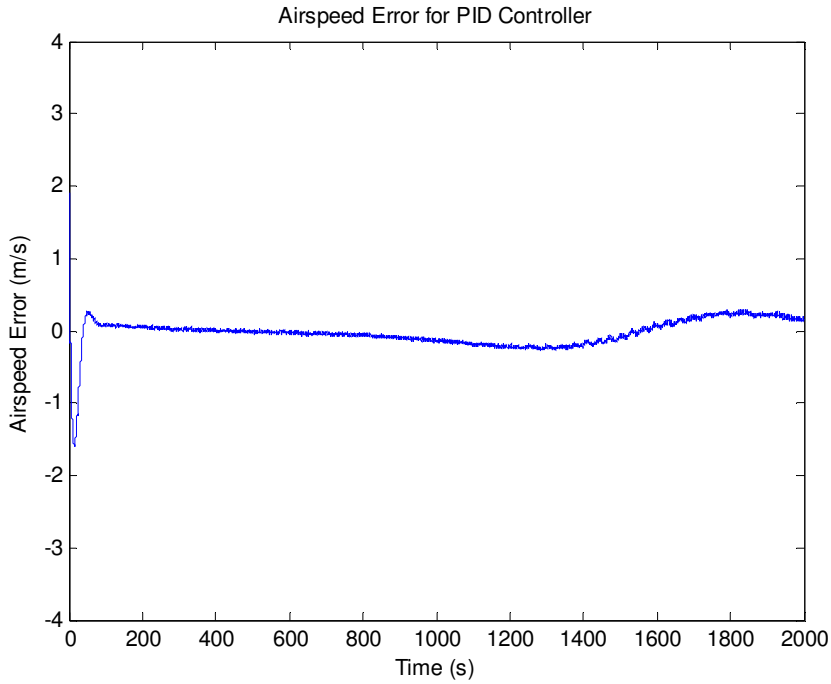


(c)

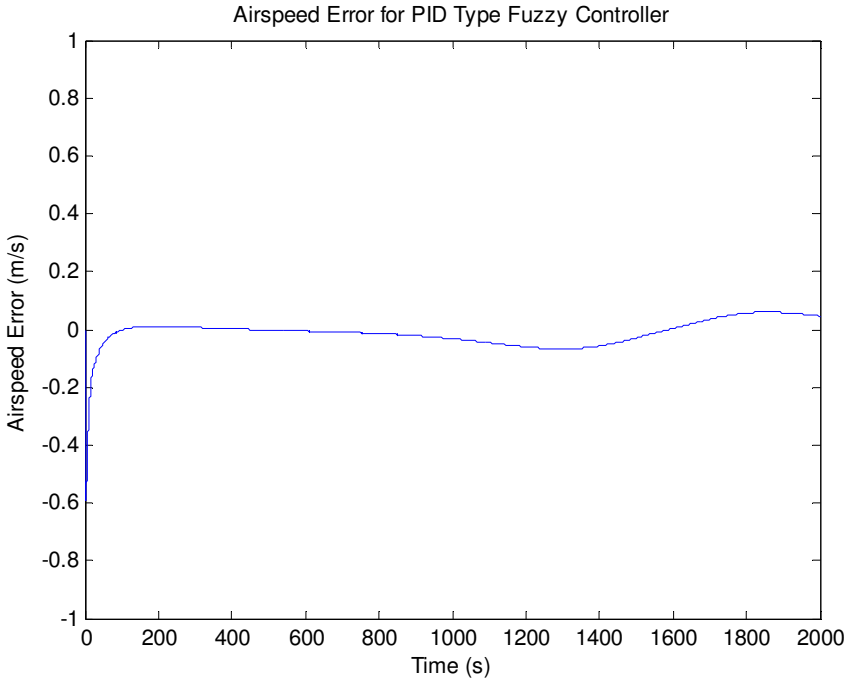
Figure 4.7. Bank Angle Control Signals of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

In Figure 4.7.(a) the control signal is not very smooth but it is in an acceptable range. The other two control signals in Figure 4.7.(b) and Figure 4.7.(c) are quite smooth and bounded. L_2 norm of the control signal can be assumed as an indicator of the energy which is applied to the system. Energies given to the system by these controllers are 0.845, 0.730 and 0.686 Nm respectively. It shows that ANFIS structure gives the minimum error with minimum energy consumption.

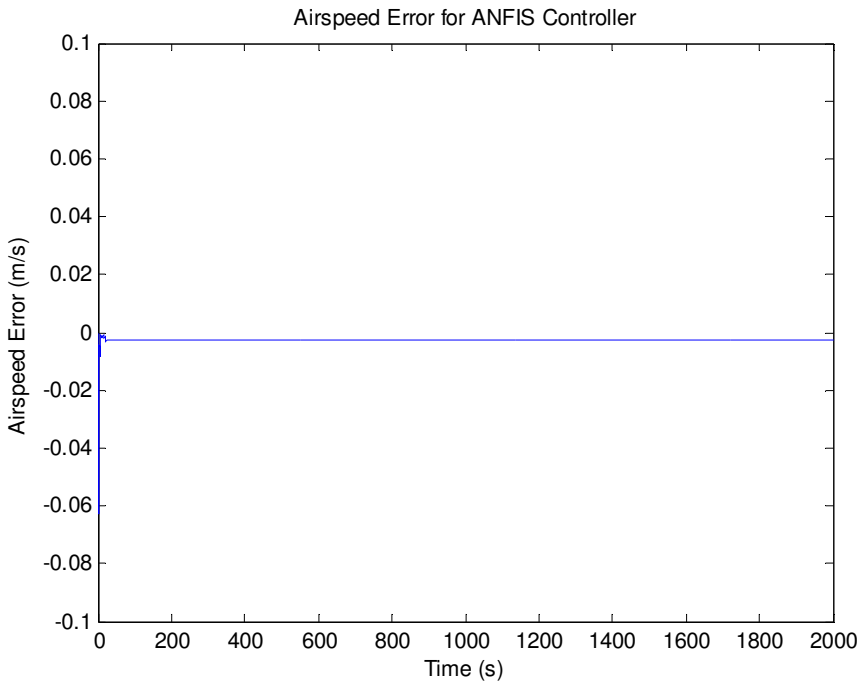
4.3 Simulation Results of Air-Speed Control



(a)



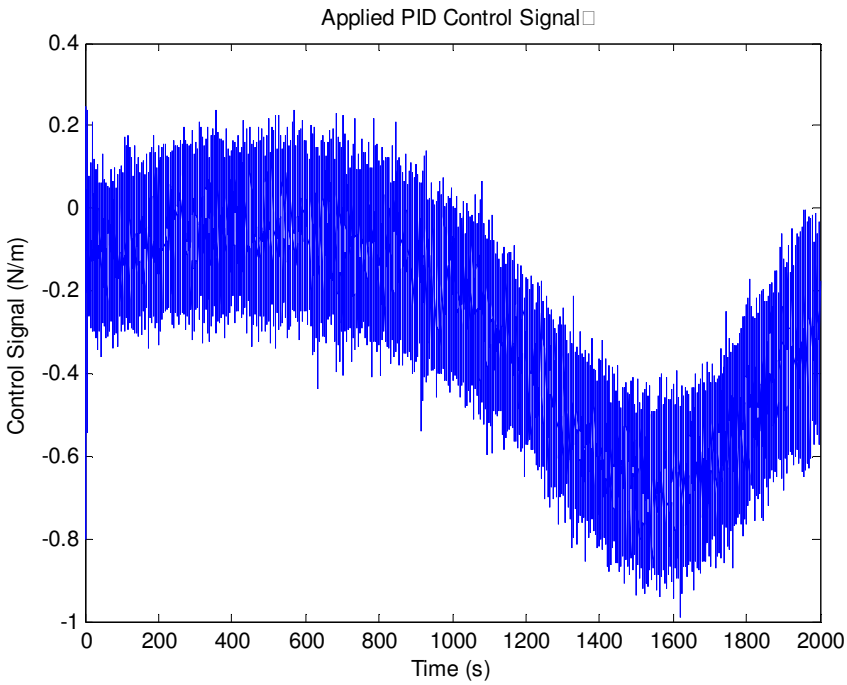
(b)



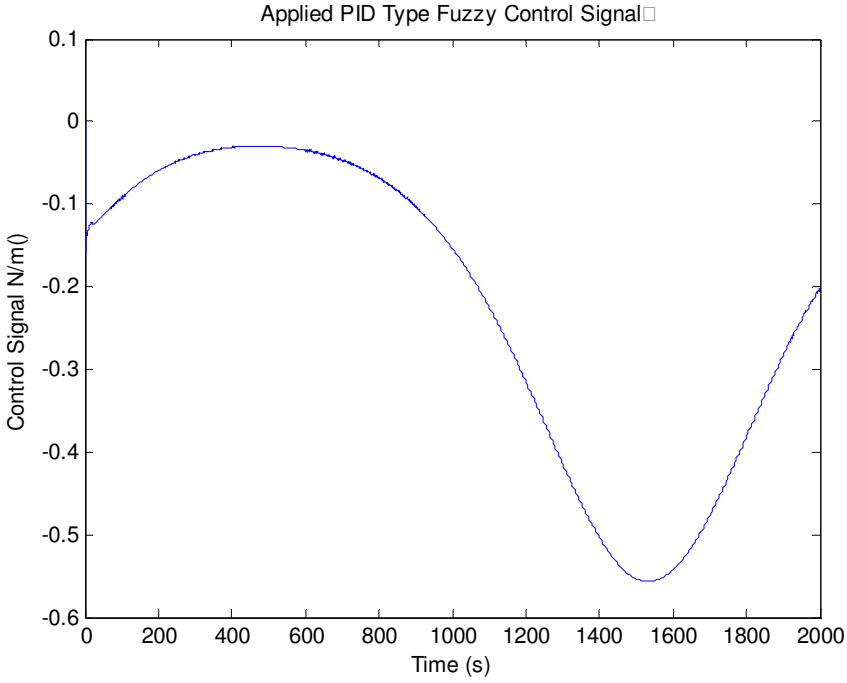
(c)

Figure 4.8. Air-Speed Errors of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

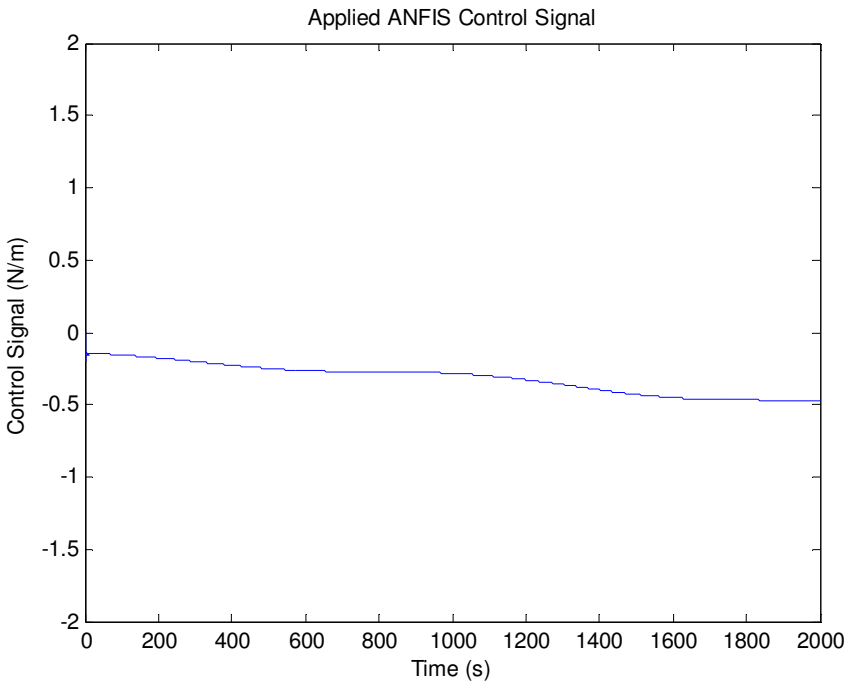
In PID control method, shown in Figure 4.8.(a), the Air speed error signal is not also very smooth but acceptable. Initial error value is 1.9. Steady state error range is [0.3, -0.3] and its settling time is 300 sec. In PID Type fuzzy control system, initial error is 0.6. Steady state error range is [0.07, -0.07]. and the settling time is 400 sec. Initial error of the system, controlled by ANFIS is 0.6. Maximum steady state error is 0.025 and settling time is 40 sec.


Applied PID Control Signal
Control Signal (N/m)
Time (s)
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(a)


Applied PID Type Fuzzy Control Signal
Control Signal N/m()
Time (s)
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(b)

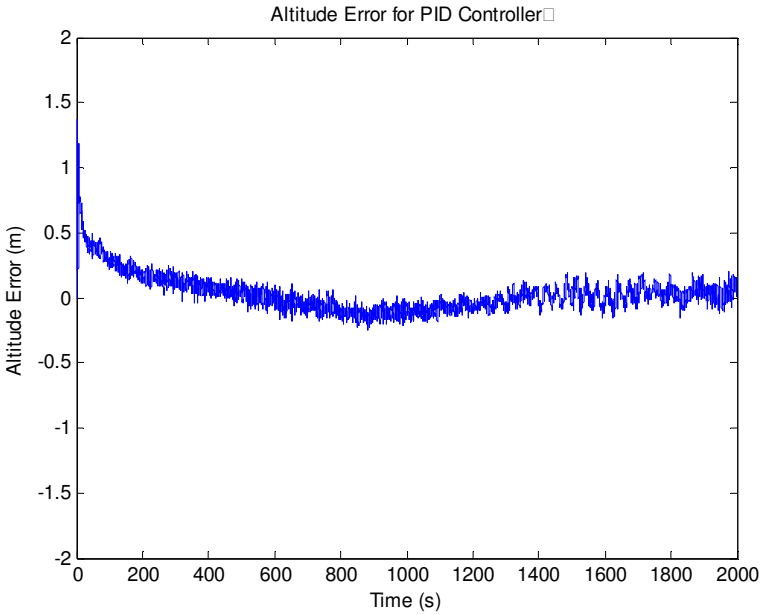


(c)

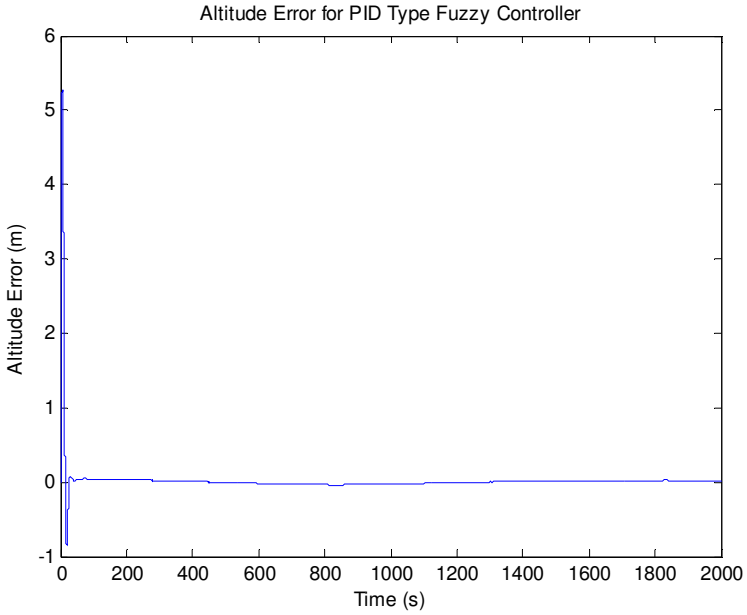
Figure 4.9. Air-Speed Control Signals of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

In Figure 4.9.(a) PID control signal is acting like a sliding mode controller. It can be seen from the Figure that it is not very smooth. There is chattering in the signal but they are all acceptable. And the energy given to the system by the controller is 16.84. In Figure 4.9.(b) signal of the PID type fuzzy controller is seen. It is a quite smooth and clear signal. Energy consumption is 13.1. In Figure 4.9.(c) ANFIS structure is also ehhibiting a smooth signal and at the end of 120^{th} second it is fully clear. The signal is bounded and changing in an acceptable range. Energy given to the system is 14.96

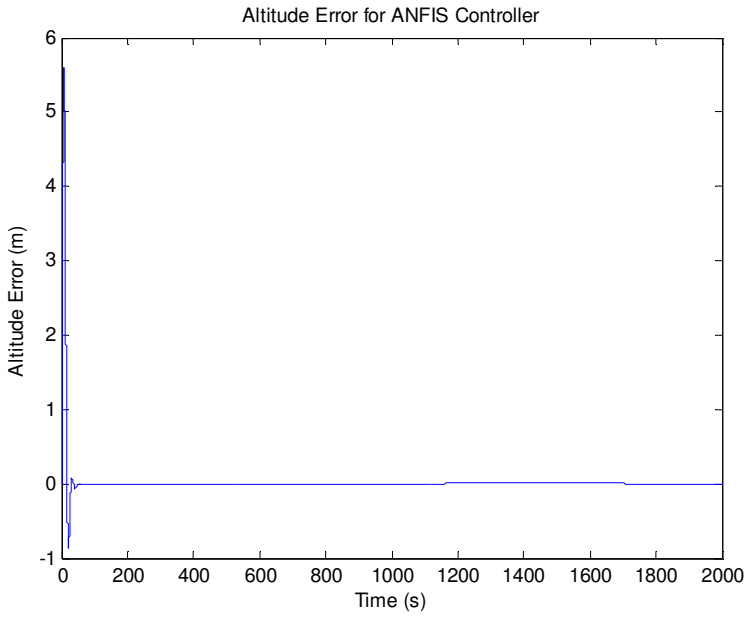
4.4 Simulation Results of Altitude Control



(a)



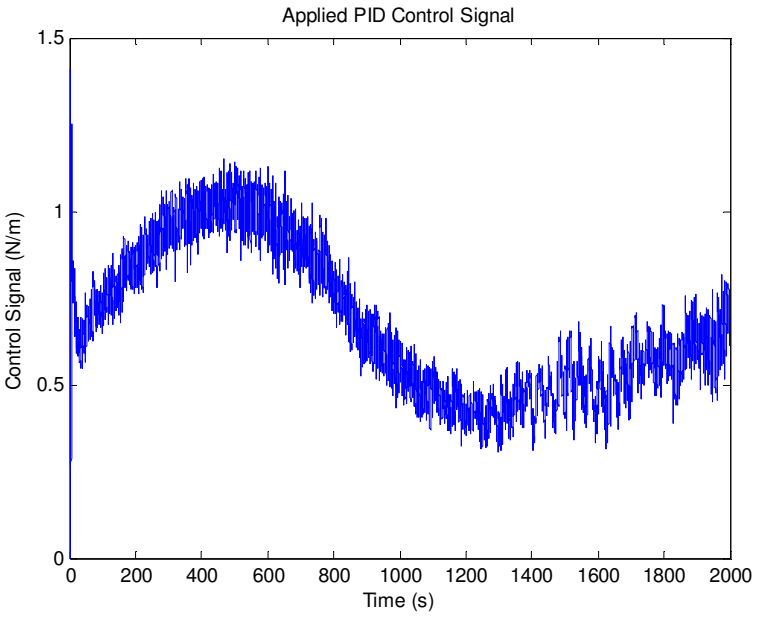
(b)



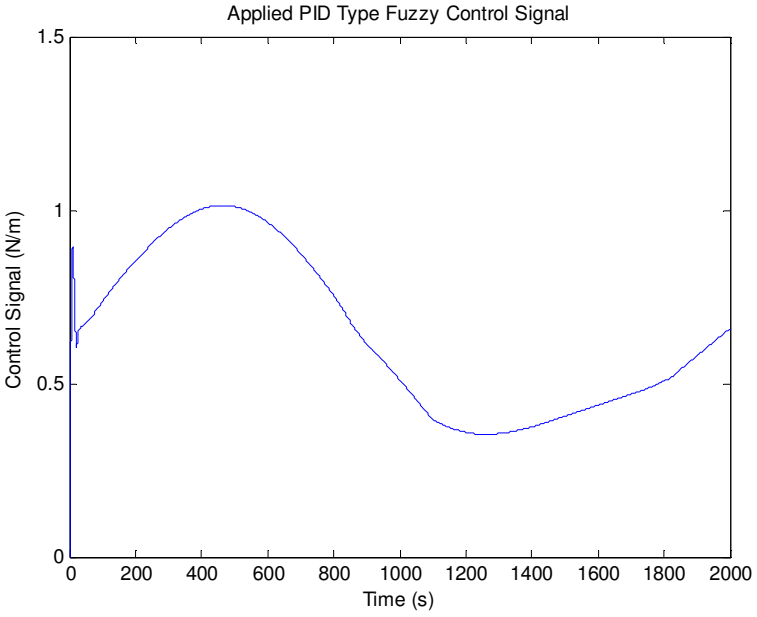
(c)

Figure 4.10. Altitude Errors of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

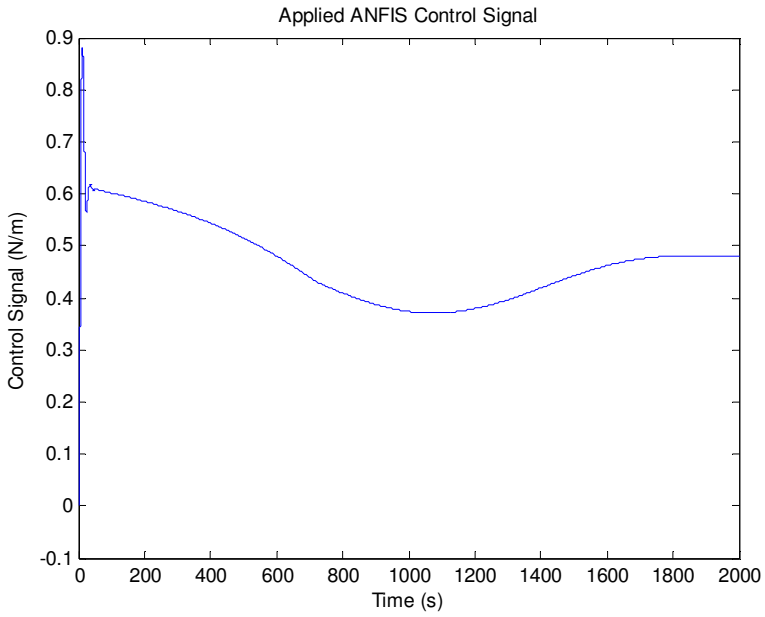
In Figure 4.10.(a) the error signal has a chattering problem but acceptable. Initial error is 1.35. Maximum steady state error is 0.25 and the settling time is 800 sec. In Figure 4.10.(b) error of the PID type fuzzy controller is seen. The signal is quite smooth and desirable with 5.3 initial error. Steady state error of the system is zero. The settling time of the system is 90 sec. In Figure 4.10.(c) initial error value is 5.6.and the steady state error is zero. The signal is quite smooth and settling time of the system is nearly 60 sec.


Applied PID Control Signal
Control Signal (N/m)
Time (s)
0
0.5
1
1.5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(a)


Applied PID Type Fuzzy Control Signal
Control Signal (N/m)
Time (s)
0
0.5
1
1.5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

(b)



(c)

Figure 4.11. Altitude Control Signals of (a) PID Controller, (b) PID Type Fuzzy Controller, (c) Anfis Controller

It can be seen from the Figure 4.11(a) that the control signal is not very smooth but quite well. There is chattering in the signal but they are all acceptable and the energy given to the system by the controller is 31.81. In Figure 4.11.(b) signal of the PID type fuzzy controller is seen. It is a quite smooth, clear and a bounded signal. Energy given to the system is 30.40. When the Figure 4.11.(c) is taken into consideration, it can be seen that control signal is very smooth and small. Total energy applied to the system is 21.21.

Table 4.5 Proposed Control Schemes Performance Chart

Bank Angle Control	Conventional PID	PID Type Fuzzy	ANFIS
Error Bound	0.3, -0.3	0.07, -0.0015	0.0012, -0.002
Settling Time	20	40	50
Control Signal Norm	0.84	0.73	0.68

Air Speed Control	Conventional PID	PID Type Fuzzy	ANFIS
Error Bound	0.3, -0.3	0.07, -0.07	0.025
Settling Time	300	400	40
Control Signal Norm	16.84	13.14	14.92

Altitude Control	Conventional PID	PID Type Fuzzy	ANFIS
Error Bound	0.25	0.00	0.00
Settling Time	800	90	60
Control Signal Norm	31,81	30,40	21,21

It can be seen from Table 4.5. that although all the control schemes gave satisfactory results, ANFIS control method exhibits the best performance and potential than the others. It provides the minimum error with minimum energy.

5. CONCLUSIONS

It is generally not possible to derive an accurate model of a process or plant especially with nonlinearities. If a reliable model is not available, it is quite difficult to design a controller producing desired outputs. Moreover, traditional modelling techniques are rather complex and time consuming.

If the system involves strong nonlinearities, conventional PID controller cannot present desired responses. In such cases, fuzzy controller can be a solution because it compensates for the shortcomings of the model, which cannot be estimated when modeling. Operating with linguistic expressions, fuzzy logic can use the experiences of a human expert and also compensate for inadequate and uncertain knowledge about the system. On the other hand, ANNs have proven superior learning and generalizing capabilities even on completely unknown systems that can only be described by its input-output characteristics. By combining these features, more versatile and robust model, called ANFIS “Adaptive Neuro Fuzzy Inference System” has been developed.

In this thesis, three different control algorithms, conventional PID, PID type fuzzy and ANFIS, were tested in order to control of altitude and the latitude-longitude of Unmanned Aerial Vehicle system. And the results were compared and discussed.

During the simulation stage, in bank angle control, it is seen that ANFIS using Sugeno model with two inputs and one output, each having five membership functions was superior to the others. It provides minimum error giving minimum energy to the system. In air speed control, ANFIS with a shortest settling time was still giving the smallest error. In altitude control, although both ANFIS and PID type fuzzy controllers make the error zero, ANFIS achieves this result giving less energy than the other gives

From the results of these experiments, it is seen that, although all the control schemes gave satisfactory results, ANFIS control method exhibits the best performance and potential than the others.

As mentioned before, ANFIS is an advantageous structure for modeling and control of systems that contain high nonlinearities. One of the most difficulties of Anfis learning procedure is the constitution part of input-output data set. Too many data pairs may cause over learning while few data pairs are not enough for the system to learn. On the other hand, most of the learning parameters of ANFIS are still determined by trial-error method. There is no systematic way to obtain these parameters. This is still an open question and it should be somehow recovered from trial-error method.

As a further study, it is planned to obtain better responses from ANFIS by using an online learning algorithm with an extensive system identification study.

REFERENCES

1. Patterson, D. W. Artificial Neural Networks – Theory and Applications, Prentice Hall, 1996.

2. Rumelhart, D. E. and J. L. McCelland, “On Learning The Past Tenses of English Words.” In J. L. McCelland, D. E. Rumelhart, & PDP Research Group (Eds), Parallel Distrubuted Processing: Exploratios in The Microstructeres of Cognition, pp.216-271. Cambridge, MA: Bradford / MIT Press, 1986.

3. Staszewski, W. J. and K. Worden, “Classification of Faults in Gearboxes – Preprocessing Algorithms and Neural Networks”. Neural Computing and Applications 5(3): 160-183, 1997.

4. Mamdani, E. H. and S. Assilian, “An Experiment in Linguistic Synthesis with a Logic Controller”, Int. J. Man – Machine Studies 8, pp. 1 – 13, 1975.

5. Takagi, S. and M. Sugeno, “Fuzzy identification of fuzzy systems and it’sapplication to modelling and control”, IEEE Trans. Systems Man Cybern., 15 pp 116-132, 1985.

6. Culliere, T., A. Titli, J. Corrieu, “Neuro-fuzzy modelling of nonlinear systems for control purposes”, In Proc. IEEE INT. Conf. on Fuzzy Systems, pp 2009-2016, Yokohama, 1995.

7. Nauck, D., “Fuzzy neuro systems: An overview”, In R. Kruse, J. Gebhardt and R. Palm, eds, Fuzzy Systems in Computer Science, pp 91-107, Vieweg, Braunschweig, 1994.

8. Jang, J. ANFIS: Adaptive-Network Based Fuzzy Inference System. IEEE Trans. On Systems, Man and Cybernetics, Vol. 23. No. 3 pp. 665-685, 1993.

9. Jang, J. and C. T. Sun, “Neuro-Fuzzy Modeling and Control”, IEEE Proc., Vol. 83. No. 3, pp 378 – 406, 1995.

10. Jang, J., “Input Selection for ANFIS Learning”, IEEE Fuzzy Systems, pp. 1493 – 1499, 1996.

11. Jang, J., “Neuro – Fuzzy Modeling for Dynamic System Identification”, IEEE Fuzzy Systems Symposium, pp. 320 – 325, 1996.

12. Altug, S. and M. Chow, “Fuzzy Inference Systems Implemented on Neural Architectures for Motor Fault Detection and Diagnosis”, IEEE Trans. on Ind.Electronics, Vol. 46, No. 6, December 1999.

13. Zhou, C. and K. Jagannathan, “Adaptive Network Based Fuzzy Control of a Dynamic Biped Walking Robot”, IEEE 1996 Int. Joint Symposia on Intelligence and Systems (IJSIS’96) Nov. 04 – 05 1996.

14. Djukanović, M.B. and M.S. Ćalović, and B.V. Vešović, and D.J. Šobajć, “Neuro – Fuzzy Controller of Low Head Hydropower Plants Using Adaptive – Network Based Fuzzy Inference System”, IEEE Trans. on Energy Conversion, Vol. 12, No. 4, December 1997.

15. Niestroy. M., “The use of ANFIS for Approximating an optimal controller”, World Cong. on Neural Networks, San Diego, CA, Sept. 15- 18, pp 1139 – 1142, 1996.

16. Jensen, E. W. and A. Nebot, “Comparision of FIR and ANFIS Methodologies for Prediction of Mean Blood Pressure and Auditory Evoked Potentials Index During Anaesthesia”, Proceedings of the IEEE Engineering in Medicine and Biology Society, Vol. 20, No. 3, 1998.

17. Oonsivilai, A. and M.E. El–Hawary, “Power System Dynamic Modeling using Adaptive – Network Based Fuzzy Inference System”, Proceedings of the 1999

IEEE Canadian Conf. on Electrical and Computer Engineering Canada May 9 – 12 1999.

18. Kurnaz, S., E. Eroglu, O. Kaynak, U. Malkoc, "A Frugal Fuzzy Logic Based Approach for Autonomous Flight Control of Unmanned Aerial Vehicles". MICAI 2005: 1155-1163.

19. Pardesi, M. S., "Unmanned Aerial Vehicles", Air & Space Power Journal Vol19 , No: 3, pp 46, 2005.

20. Laurence R. Newcome. "Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles". AIAA, 2004.

21. http://www.noaanews.noaa.gov/stories2005/s2508.htm. October 24, 2005.

22. AeroSim Aeronautical Simulation Blockset User's Guide, www.u-dynamics.

23. Driankov, D., H. Hellendoorn, and M. Reinfrank, "An Introduction to Fuzzy Control", Second Edn, Springer-Verlag, Berlin 1996.

24. Passino, K. M. and S. Yurkovich, *Fuzzy Control*, AddisonWesley Longman, Inc, Menlo Park, CA, USA 1998.

25. Lee, C. C. (1990). "Fuzzy logic in control systems: Fuzzy logic controller", IEEE Trans. Systems, Man & Cybernetics 20 (2): 404–435.

26. Holmblad, L. P. and J.-J. Østergaard, "Control of a cement kiln by fuzzy logic", in Gupta and Sanchez (eds), Fuzzy Information and Decision Processes, North-Holland, Amsterdam, pp. 389–399. 1982. (Reprint in: FLS Review No 67, FLS Automation A/S, Høffdingsvej 77, DK- 2500 Valby, Copenhagen, Denmark).

27. Pedrycz, W., *Fuzzy Control and Fuzzy Systems*, second edn, Wiley and Sons, New York. 1993.

28. IEC., Programmable controllers: Part 7 fuzzy control programming, Technical Report IEC 1131 International Electrotechnical Commission. (Draft.), 1996.

29. Hill, G., E. Horstkotte, and J. Teichrow, *Fuzzy-C Development System- User's Manual,* Togai Infralogic, 30 Corporate Park, Irvine, CA 92714, USA. 1990.

30. Siler, W. and H. Ying, "Fuzzy control theory: The linear case", Fuzzy Sets and Systems 33: 275–290, 1989.

31. Mizumoto, M. "Realization of PID controls by fuzzy control methods", in IEEE (ed.), First. Int. Conf. on Fuzzy Systems, number 92CH3073-4, The Institute of Electrical and Electronics Engineers, Inc, San Diego, pp. 709–715, 1992.

32. Qiao,W. and M. Mizumoto, "PID type fuzzy controller and parameters adaptive method", Fuzzy Sets and Systems 78: 23–35, 1996.

33. Engin, S., J. Kuvulmaz and V. Omürlü, "Fuzzy Control of an ANFIS Model Representing a Nonlinear Liquid Level System", Neural Computing and Applications, Vol.13, 2004, pp. 202–210.

34. Shah S., F. Palmieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks," *Neural Networks,* vol. *5,* no. 5**,** pp. 779-787, 1992.

35. Sbar S. and F. Palmieri, "MEKA-a fast, local algorithm for training feedforward neural networks," in *Proc. Int. Joint Conf Neural Networks,* pp. 111 4146, 1990.

36. Singhal S. and L. Wu, "Training multilayer perceptrons with the extended kalman algorithm," *Advances in Neural Information Processing Systems-I,* in D. *S.* Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 133-140.