BLACKBOARD ARCHITECTURE FOR AN UNMANNED AERIAL VEHICLE CONTROLLER USING FUZZY INFERENCE SYSTEMS

by

SWETHA PANDHITI

(Under the Direction of Walter D. Potter)

ABSTRACT

The objective of this research is to design a controller for low-cost fixed-wing unmanned aerial vehicles (UAV's) using fuzzy rules and fuzzy inference systems. The control flow in the architecture design process is considered in four levels - Guidance, Navigation, Control & Stability and Communications. The main function of the Fuzzy Logic Control System (FLCS) is to control and stabilize the aircraft while navigating towards different waypoints defined according to the waypoint plan. This idea forms the base for an AI approach to the development of an autopilot unlike the traditional way of using PID controllers for control and navigation.

INDEX WORDS: Unmanned Aerial Vehicle, Fuzzy Inference Systems, Flips, Fuzzy Logic.

BLACKBOARD ARCHITECTURE FOR AN UNMANNED AERIAL VEHICLE CONTROLLER USING FUZZY INFERENCE SYSTEMS

by

SWETHA PANDHITI

B.Tech, Jawaharlal Nehru Technological University

Hyderabad, Andhra Pradesh, India.

2008.

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2011

BLACKBOARD ARCHITECTURE FOR AN UNMANNED AERIAL VEHICLE CONTROLLER USING FUZZY INFERENCE SYSTEMS

by

SWETHA PANDHITI

Major Professor:	Walter D. Potter
Committee:	Suchendra M. Bhandarkar Kyle Johnsen

ACKNOWLEDGEMENTS

I gratefully acknowledge my major professor Dr. Potter for all the help and support provided to me regarding my research and for guiding me throughout this process. I would also like to thank my committee members, Dr.Bhandarkar and Dr.Johnsen for the numerous suggestions that I received during the course of my thesis work. I would also like to acknowledge the UAV project team members for all the different ideas and comments during the weekly meetings. A special thanks to Robert Eunice for the idea behind the project and for letting me use FLIPS in my research.

I would also like to thank my parents and my aunt & uncle for their continuous love and support they have provided all along my life. Finally, I would like to thank all my friends for all the fun times I had during my stay in Athens.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This section introduces a brief overview of the present research project and the research topic. In this project, a rule-based system is being developed, where the control flows from the higher levels to the lower levels. The present hierarchical four-layered architecture follows the statement: “Increasing intelligence with decreasing precision” [16] as we move from the lower to the higher layers.

The architecture can be described in different operating levels, which are assigned to perform four different kinds of functions. The user specifies his/her goals in a high-level (human readable) language, which are then converted to tasks with the help of the FLIPS parser (chapter 3). The autopilot converts these instructions or tasks to fuzzy rules. These fuzzy rules, combined with fuzzy inference systems, send out action commands to the actuators. The actuators then communicate with the hardware (micro controllers) on-board to execute the mission.

The main focus of this thesis is the Fuzzy Logic Controller design for Unmanned Aerial Vehicles, which is solely responsible for the Navigation and Control & Stability tasks. Hence, the emphasis of this thesis would be the process of conversion of the tasks to actions which is depicted in Figure 1.

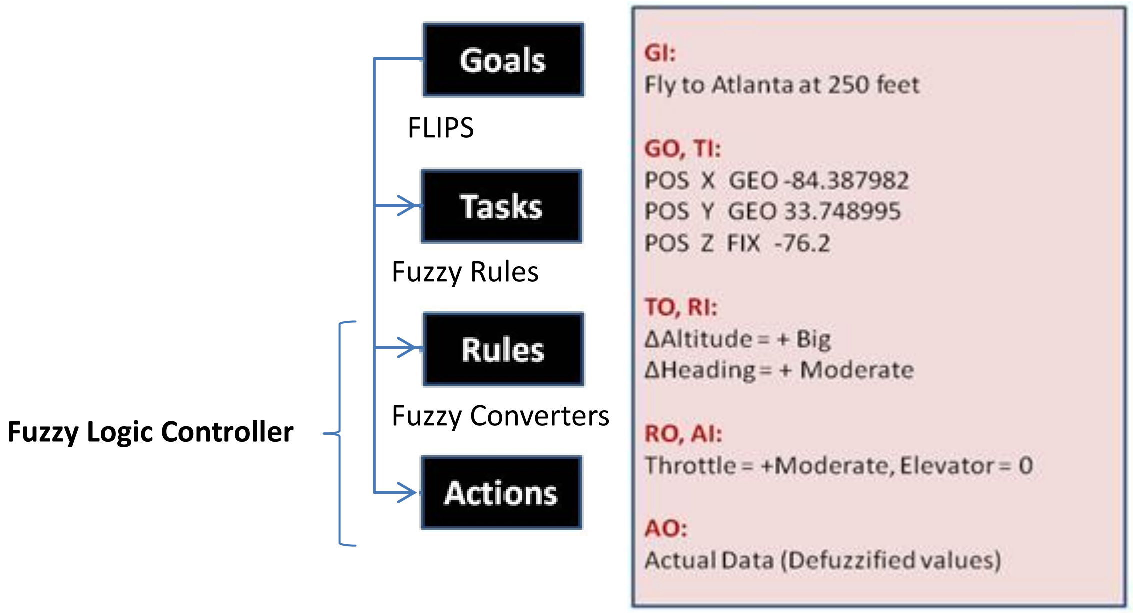


Figure 1: UAV System Overview

Though the practical implementation involves much more than this, the above architecture can be compared to a virtual black-board. The black-board idea is such that the values from every level are posted onto the black-board. Every level gets its instructions or tasks from its corresponding higher level's output values. With the help of the fuzzy inference system (controller) and the values posted on the black board, the autopilot sends commands to the actuators for task execution. Since this is a closed-loop feedback control system, the current state of the aircraft is measured and stored. The controller compares these values (current state and the desired state of the aircraft) and executes the tasks accordingly.

To define the autonomy of this system, consider a black box, a white box and a grey box. The black box is something whose functionality is completely defined and is fully autonomous. The white box is such that only the inputs, outputs and attributes are defined and the complete functionality is transparent (visible to the user) and dependent on the user. These systems can be termed as non-autonomous systems. The grey box is said to be a combination of both where the functionality of the system is neither dependent on the user nor the system and is said to be semi-autonomous. The present Fuzzy Logic Control System (the controller) is considered as a black box, which is said to be fully autonomous.

CHAPTER 2
BACKGROUND

2.1 What are UAV's?

An Unmanned Aerial Vehicle, commonly known as UAV, is an aircraft that is capable of flying without the physical presence of a pilot. Remotely piloted UAV's are aircrafts that are controlled from a remote location via a radio link. There can be many people involved in remote operations on the ground ranging from those operating the vehicle from a ground control station, to the people coordinating multiple UAV's in air operations or air traffic control center. On the other hand, autonomous and semi-autonomous aircraft are enabled to fly based on pre-programmed flight plans or more complex dynamic automation systems. They are capable of handling low level and high level decisions. The degree of autonomy depends on the level of decisions that the UAV is capable of making.

Fixed-wing mini-UAV is an emerging class of UAV's which has potential applications in ground traffic control, crop monitoring, telecommunications, mineral exploration, reconnaissance, attack roles, etc. Due to the low manufacturing and operational costs, the flexibility to adjust to the particular needs of the consumers and the elimination of the risk of human lives (pilots), the demand for UAV's has been rising over the last thirty years. In addition to the various applications in the military and civilian market, UAV's are also being used to measure atmospheric composition by flying pre-programmed flight paths to obtain precise scientific data

[8] [17]. In [9], a fixed-wing mini UAV and a gimbaled camera have been used for target acquisition, localization and surveillance tasks. Virtual Reality aided control of unmanned vehicles [10] has also been designed for better enhancement of the operator's situational awareness and effectiveness.

In the recent years, a lot of research is being carried out on autonomous and semi-autonomous UAV's [31-33] and the various autopilot architectures employed in these UAV's [35-37]. In [34], a FPGA and DSP based autopilot which involves a new control system is being designed to achieve higher functionalities of aircraft behaviors. Evolutionary algorithms and genetic programming techniques [38] have also been used for the design of autonomous navigation controllers.

2.2 What is Fuzzy Logic?

Fuzzy Logic, introduced by Lofti Zadeh [29] deals with reasoning that is fixed or appropriate rather than fixed and exact. It emulates the human deductive process which can be defined as the process where people infer conclusions from what they know (adequate or inadequate) [23]. Fuzzy Logic can be used to control or describe a system by using "commonsense" rules that refer to inadequate quantities [30] [3]. It incorporates a simple rule-based "**if X and Y, then Z**" approach to solving a control problem rather than attempting to model a system mathematically [5]. Due to the use of linguistic information, stimulation of human thinking and the ability to capture the approximate and inexact nature of the real world, fuzzy rule-based systems are said to perform very effectively in a wide variety of practical problems [27]. In [22] and [41], fuzzy logic controllers have been proven to be universal approximators and can approximate any non-linear function to have an arbitrarily small error bound.

Fuzzy Set:

A fuzzy set has been derived as an extension of the classical notion of a set. According to classical set theory, an element either belongs or does not belong to the set. In contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set.

Membership Function:

A membership function can be described as a graph that defines how each point in the input space is mapped to the membership value between 0 and 1.

Fuzzy Inference Systems:

Fuzzy Inference Systems are designed based on the concept of Fuzzy Logic. Fuzzy inference can be defined as the process of formulating the mapping from a given input to an output. A Fuzzy Logic System or a Fuzzy Inference System maps the crisp inputs into crisp outputs. It contains five components: Rule-base, Data-base, Fuzzifier, Inference and Defuzzifier.

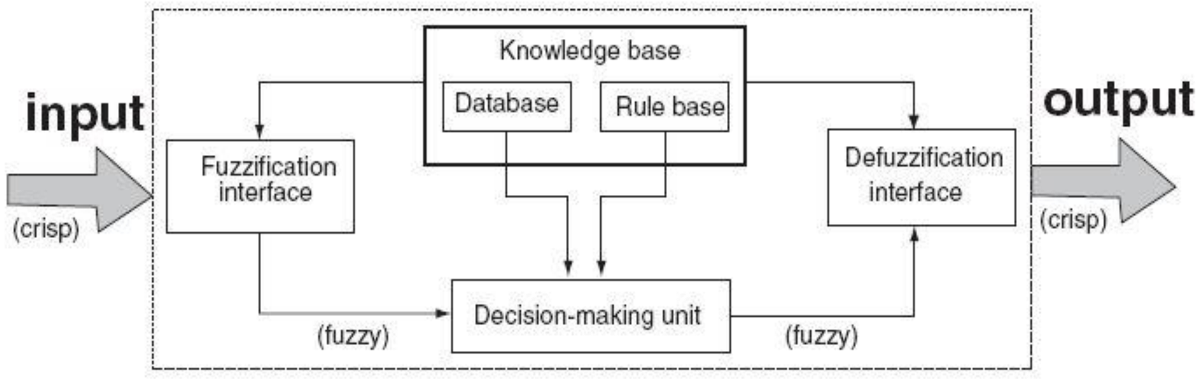


Figure 2: Fuzzy Inference System [15]

Fuzzy IF-THEN rules form the ‘rule base’ for the fuzzy inference mechanism, which makes decisions that are most suitable for the current situation based on these rules. A ‘database’ defines the membership functions of the fuzzy sets used in the fuzzy rules. ‘Rule base’ and ‘database’ together constitute the ‘knowledge base’. The ‘fuzzification interface’ is used to convert the “crisp” input to a fuzzy value which will be used by the fuzzy inference mechanism and these are converted back to “crisp” output values using the ‘defuzzification interface’. The ‘decision-making unit’ performs the inference operations on the rules.

A fuzzy inference process comprises five parts: fuzzification of the input variables, application of the fuzzy operator (AND in this case) in the antecedent, implication from the antecedent to the consequent, aggregation of the consequents across the rules, and defuzzification.

Step 1: Fuzzification of inputs is done by determining the degree to which they belong to each of the appropriate fuzzy sets via membership functions. This process takes the crisp input values which are sent to the controller and determines the degree to which these inputs belong to each of the appropriate fuzzy sets. In many practical applications, there are chances of obtaining imprecise data due to the negligible error present in some quantities. This imprecision can be represented by the membership functions in the fuzzification process. There are various methods to assign the membership values or the membership functions to fuzzy variables. The assignment of the membership functions can be done by intuition or by using some algorithms or logical procedures. Intuition is based on a human’s own intelligence and understanding to develop the membership functions. Thorough knowledge of the problem has to be known and the knowledge regarding the linguistic variable should also be known [15].

Step 2: After all the inputs are fuzzified, the degree to which each part of the antecedent is satisfied for each rule is known. Since the given rules have more than one part in the antecedent, the corresponding fuzzy operator (AND) is applied to obtain a number that represents the result of the antecedent for a particular rule.

Step 3: The number that is obtained after the logical operator is applied determines the weight associated with that particular rule. After all the rules have been weighted, the implication method is implemented. The consequent in a rule is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. Therefore, the input for the implication process is a single number given by the antecedent, and the output is a fuzzy set.

Step 4: Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a fuzzy set. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable.

Step 5: The fuzzy set (the aggregated output fuzzy set) is the input for the defuzzification process. The defuzzified output is always a calculated crisp value based on the defuzzification method used in the process. The most widely used Centroid Method has been selected for defuzzification, which returns the center of area under the curve.

The following diagram displays the above five steps:

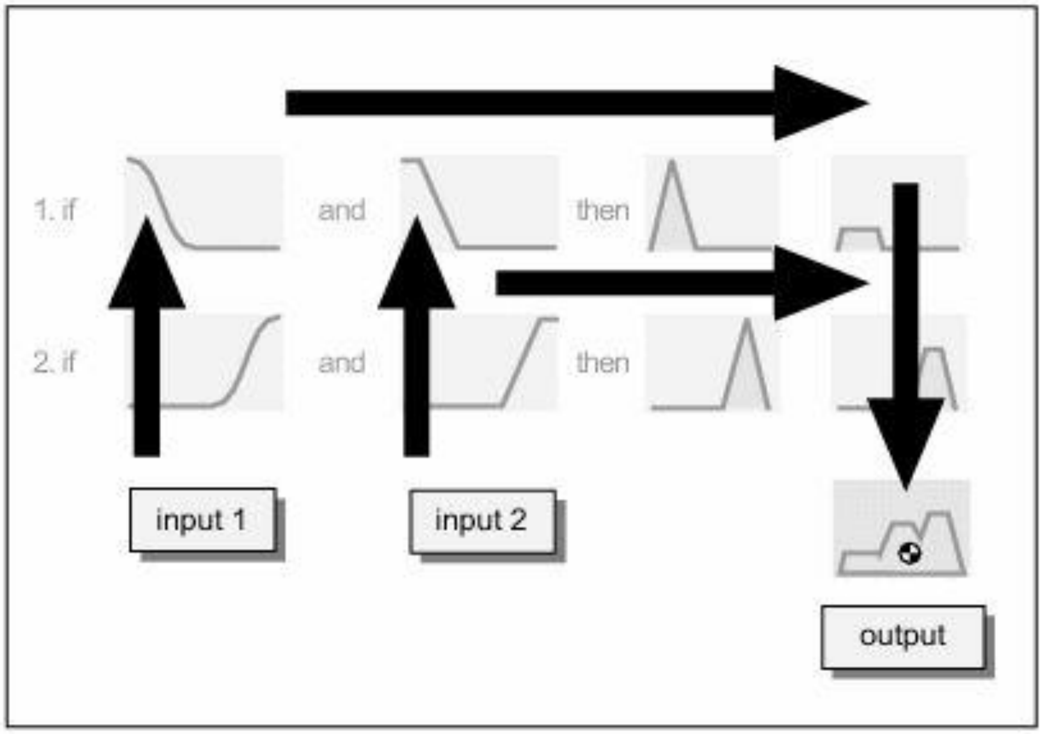


Figure 3: Fuzzy Inference Process (Source: Matlab)

There are two kinds of fuzzy inference controllers provided by MATLAB: Mamdani and Sugeno. The inference process explained in the above section is the Mamdani's Fuzzy Inference process. The first two parts of the fuzzy inference process, fuzzyfying the inputs and applying the fuzzy operator are the same for both the controller types. The main difference lies in the output membership functions and Sugeno's output membership functions are either linear or constant.

A typical rule in a Sugeno fuzzy model has the form –

If Input 1 = **x** and Input 2 = **y**, then Output is **z = ax + by + c**

A zero-order Sugeno fuzzy model's output is thus a constant, which can be represented as a singleton spike of a Mamdani's fuzzy output.

2.3 Why consider Fuzzy Logic over conventional techniques?

An important problem in autonomous navigation of any system is the need to cope with the large amount of uncertainty that is inherent of natural environments. Fuzzy logic has features that make it an adequate tool to address this problem [28] [20]. Unlike classical logic, fuzzy logic is said to be tolerant to imprecision, uncertainty and partial truth. This makes it easier to implement fuzzy logic controllers to non-linear models than other conventional techniques [39].
Due to the inherent ability to deal with imprecise inputs and their low computational complexity, fuzzy logic based systems have found various applications in recent times. Fuzzy logic has various applications like the design of flight control and navigation systems [12] for autonomous UAV's using three subsystems, the analysis of very large industrial investments [15], monitoring of induction motors stator, and many more. There are also some applications where the fuzzy logic controllers have been developed with fuzzy rules and fuzzy membership functions being evolved using evolutionary techniques [11] [13] [21] [26] and self tuning techniques [25]. Fuzzy Logic is also flexible in the sense that additional functionality can be imposed on the system without building the system from scratch.

The Institute for AI at UGA had developed a model for the control and navigation of a UAV using PID (Proportional-Integral-Derivative) controllers initially. A PID controller is a control loop feedback mechanism (controller), which calculates the error as the difference between a measured process variable and a desired set point. There have been many developments lately with the combination of fuzzy and PID controls, where the PID controllers are used as low-level controllers and fuzzy systems as high-level controllers [19]. The working of a PID controller is demonstrated in Figure 4.

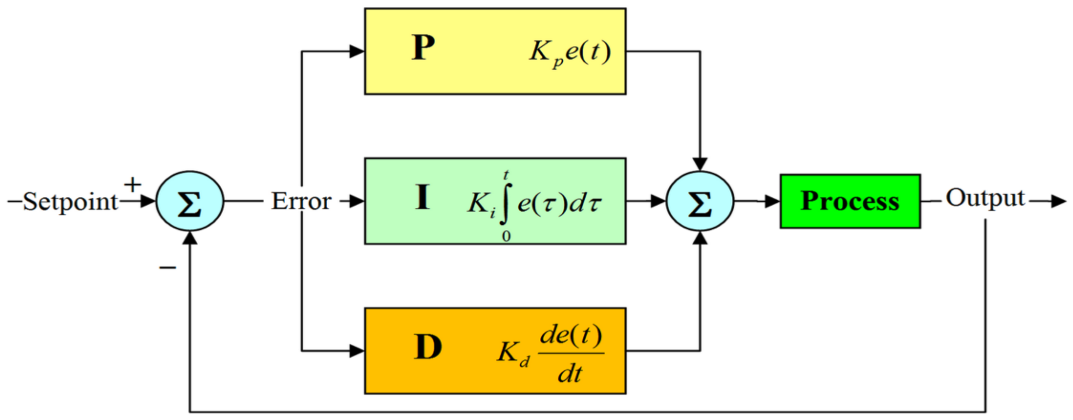


Figure 4: Working of a PID (Source: Wikipedia)

The PID controller calculation involves three constant terms and is accordingly sometimes called three-term control: P - Proportional, I - Integral and D - Derivative. In terms of time, it can be explained as – ‘P’ depends on the present error, ‘I’ depends on the accumulation of the past errors and ‘D’ is a prediction of the future errors, based on current rate of change. The weighted sum of these three factors is used to adjust the process via a control element (in this case - parameter control).

The coefficients K_p , K_i , and K_d are the ones tunable for a PID. Although PID’s are widely used and seem to be simple, they can be hard in practice if multiple (and often conflicting) objectives such as short transient and high stability are to be achieved [40]. PID controller’s are said to be unmanageable to handle a large set of attributes. They could however work independently for each single attribute but a combination of different attributes is difficult to deal with [4].

CHAPTER 3

THE UAV AND ITS ARCHITECTURE

3.1 Basic Terms

An aircraft mainly operates on three principal axes. They are:

Vertical axis (Yaw): An axis drawn from top to bottom, and perpendicular to the other two axes.

Lateral axis (Pitch): An axis running from the pilot's left to right in piloted aircraft, and typically parallel to the wings of a winged aircraft.

Longitudinal axis (Roll): An axis drawn through the body of the vehicle from tail to nose in the normal direction of flight, or the direction the pilot faces.

The following are some of the terms used in this project:

Heading (Course): An aircraft's heading is the angle that the intended path of the vehicle makes with a fixed reference object (typically true north). This is typically measured in degrees from 0 clockwise to 360 in compass convention. For example, heading 90 deg is facing east.

Altitude: The vertical distance of an airplane above the mean sea level (MSL).

Airspeed: True Airspeed is the speed at which the aircraft is traveling with respect to the surrounding air [6]. The wind moving with or against the aircraft directly affects the speed of the air moving under the wings and thus directly affects the true airspeed. If there were no wind, then the true airspeed of the aircraft would be equal to the ground speed of the aircraft.

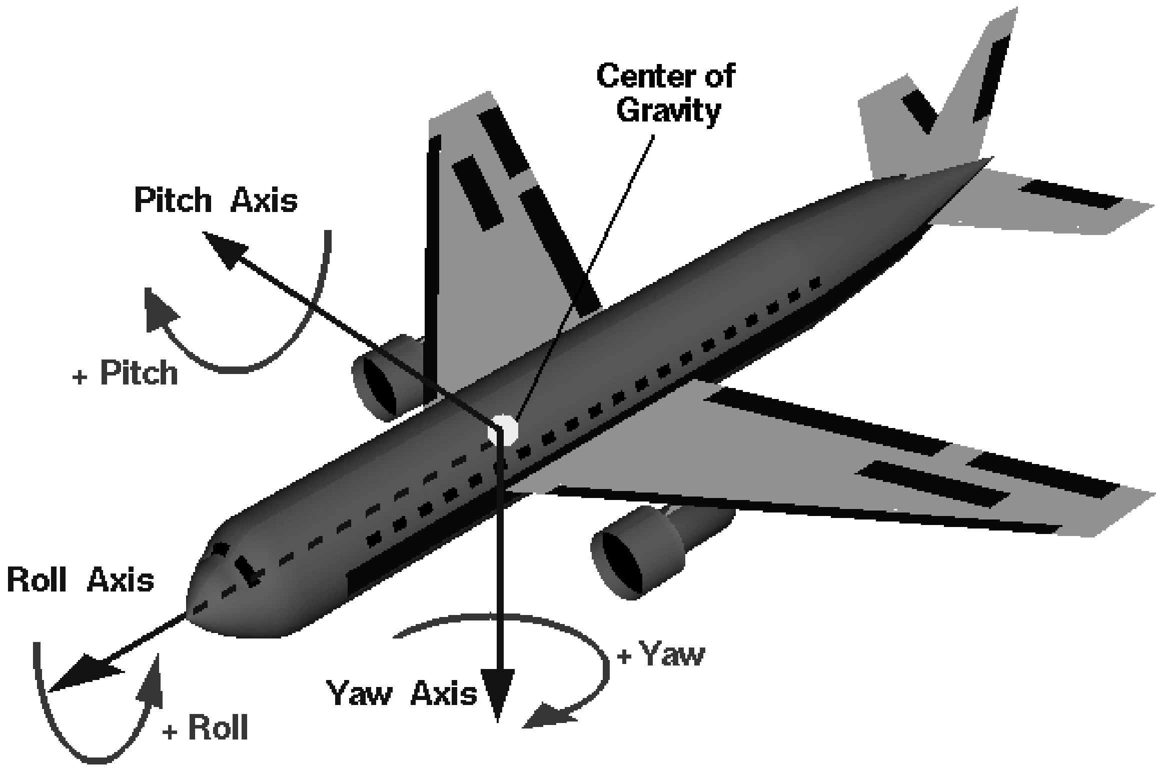


Figure 5: Control Axes of an aircraft (Source: Wikipedia)

<u>Angle of bank:</u> The angle between the aircraft's normal axis and the earth's vertical plane containing the aircraft's longitudinal axis.

<u>Shallow turn:</u> A shallow turn is that in which the bank (less than approximately 20°) is so shallow that the inherent lateral stability of the airplane is acting to level the wings unless some aileron is applied to maintain the bank.

<u>Moderate turn:</u> A moderate turn is that resulting from a degree of bank (approximately 20° to 45°) at which the airplane remains at a constant bank.

<u>Steep turn:</u> A steep turn is that resulting from a degree of bank (45° or more) at which the "over banking tendency" of an airplane overcomes stability, and the bank increases unless aileron is applied to prevent it [1].

Throttle: The valve in a carburetor or fuel control unit determines the amount of fuel-air mixture that is fed to the engine resulting in various engine power output.

Elevator: The horizontal, movable primary control surface in the tail section of an airplane. The elevator is hinged to the trailing edge of the fixed horizontal stabilizer.

Ailerons: Primary flight control surfaces mounted on the trailing edge of an airplane wing, near the tip. Ailerons control roll about the longitudinal axis.

Rudder: The movable primary control surface mounted on the trailing edge of the vertical fin of an airplane. Movement of the rudder rotates the airplane about its vertical axis.

Elevons: Aircraft control surfaces that combine the functionality of the elevator and ailerons.

Ground Track: The aircraft's path over the ground when in flight.

3.2 Basic Flight Maneuvers

All flying tasks are based on four fundamental basic flight maneuvers: straight-and-level-flight, turns, climbs and descents. These four basic maneuvers are used in the implementation of this project.

Straight and level flight:

Flight in which constant heading and altitude are maintained. It is accomplished by making immediate and measured corrections for deviations in direction and altitude from unintentional slight turns, descents and climbs [1].

Level Turns:

A turn is made by banking the wings in the desired direction of turn. A specific angle of bank and heading has to be selected in order to make a turn. The turns have been classified into two types with three groups in each type. The type of turns may be defined as left turn and right turn.

The groups can be defined as steep turn, moderate turn and shallow turn. To stop the turns, the wings are returned to level flight by the coordinated use of the ailerons and rudder applied in the opposite direction [1].

Climbs:

When an airplane enters a climb, it gains altitude and changes its flight path from level flight to an inclined plane or climb attitude [1].

Descents:

An airplane is said to descend when it loses altitude and changes its flight path from level flight to an inclined plane or descent attitude [1].

3.3 System Architecture

The UAV autopilot is said to operate in four levels. They are:

- Guidance (via FLIPS)
- Navigation
- Control and Stability
- Communications

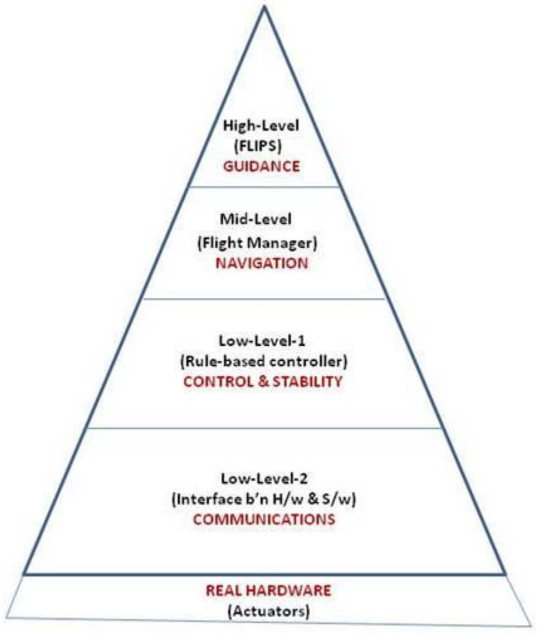


Figure 6: UAV System Architecture

Guidance (High Level):

This level consists of high level operations (goals and missions) specified by the user. Any kind of Guidance system can be implemented with the present FLCS. FLIPS (A Flight Instruction Processing System for Unmanned Aerial Vehicles) grammar developed by Robert Eunice has been chosen for this particular project in order to give instructions / specify waypoints (goals) [7]. With a pre-defined library of flight behaviors and missions, FLIPS helps the users to experiment with new missions and flight behaviors easily. A detailed description of FLIPS is explained in Section 3.4.

3.4 FLIPS:

FLIPS is a grammar used to define UAV mission commands in a high-level language that can be used as a hardware-independent instruction set architecture (ISA) for implementation of commands. The goal of FLIPS is to provide a better abstraction between the user and the flight hardware. The working of FLIPS is as given below -

1. The mission is specified by the user in the high-level FLIPS language.
2. It is then compiled into FLIPS Assembly which contains low-level flight instructions according to the FLIPS ISA.
3. The FLIPS assembly code is then translated into hardware-specific radio packets or C/C++ code which is specific to the UAV platform.
4. The UAV platform implements the FLIPS ISA, which provides instructions to update the fuzzy control parameters, set points, actuator positions, and others to execute the mission [7].

This project utilizes the FLIPS assembly language instructions to specify the high-level tasks. These instructions are presented to MATLAB in a text file. The commands and waypoints used for controller input are specified in the example below to create a better understanding for the user. An example of FLIPS conversion is as follows:

<u>Example1.uav</u>

Commands:

Normal = 0

Take-off = 1

Inverted = 2

Hover = 4

Roll left = 8

Roll right = 16

Waypoints		Latitude	Longitude
Atlanta	=	33.748995	-84.387982
New York	=	40.714269	-74.005973
London	=	51.00652	-0.126708

Goals: (High-level instructions)

a. Take-off

b. Fly to Atlanta at 50 meters

c. Fly to New York

d. Fly at 1000 feet

e. Loiter and Land

Example1.uav.asm (FLIPS Assembly)

```
a.    CMD 1                          // Take-off
b.    POS  X  GEO  -84.387982        // Atlanta W Longitude
      POS  Y  GEO  33.748995         // Atlanta N Latitude
      POS  Z  FIX  -50.0             // 50m Altitude
      FLY
c.    POS  X  GEO  -74.005973        // New York W Longitude
      POS  Y  GEO  40.714269         // New York N Latitude
      FLY
d.    POS  Z  FIX  -304.8            // 1000 feet = 304.8m
```

FLY

e. CMD 192 // Loiter and Land

These FLIPS assembly commands are the output commands of the Guidance level.

Navigation (Mid-Level):

The mid-level coordinates the task execution by receiving the task information (FLIPS Assembly) from the high level. The Flight Manager tries to identify the current & desired states of the aircraft and sends commands to the next level. These commands describe the kind of maneuver and the type of maneuver the aircraft needs to perform at that moment.

Ex: From the above example, **(d.) says - POS Z FIX -304.8 (climb to 1000 feet)** and suppose the altitude sensor says that the **current altitude is 200 feet**. The **difference is 800 feet** and is **very big**. So, the Flight Manager sends an instruction such as

ΔAltitude = + High => Perform a steep climb

Control is then handed over to the Control & Stability level.

Control and Stability (Low-Level1):

The lower level receives the instructions from the mid-level and then tries to match these to the fuzzy rules designed for control & stability of the aircraft. This fires the actual rules that perform the maneuver and type of maneuver action. Examples of the low-level rules for a steep climb are as follows:

Steep Climb: **(ΔAltitude = + High)**

If Pitch = Low, Airspeed = Low

Then => Throttle = +Moderate

Elevator = + Moderate

If Pitch = Moderate, Airspeed = High

Then =>   Throttle = 0

Elevator = + Low

Control and Stability (Low-Level2):

This level is responsible for the defuzzification of the fuzzy output values from the low-level rules fired. These defuzzified values are then converted to electrical signals to be sent to the actuators. This is then sent to Flight Gear for visualization purposes.

The 'Communications' level is not being discussed here since the system has been implemented and tested in a simulation environment only.

CHAPTER 4

IMPLEMENTATION IN MATLAB

4.1 Rules

In a rule-based system, each rule represents a unit of knowledge and all the knowledge is expressed in the same format. In order to attain the readability and flexibility of the present control system, the controller has been designed as a rule-based system whose architecture is hierarchically structured. Every layer in this architecture is based on the function of rules performed by the autopilot. Every layer consists of a set of rules and is thus easy to understand and activate. MATLAB's Fuzzy Logic Toolbox has been used for the development of a rule-base to navigate the aircraft. The rules have been categorized into four major categories which are further divided into subcategories of steep, moderate and shallow. They are –

Climbs:	Steep climbs	=	**ΔAltitude = + High**
	Moderate climbs	=	**ΔAltitude = + Moderate**
	Shallow climbs	=	**ΔAltitude = + Low**
Descents:	Steep descents	=	**ΔAltitude = - High**
	Moderate descents	=	**ΔAltitude = - Moderate**
	Shallow descents	=	**ΔAltitude = - Low**
Left Turn:	Steep turn	=	**ΔHeading = - High**
	Moderate turn	=	**ΔHeading = - Moderate**
	Shallow turn	=	**ΔHeading = - Low**

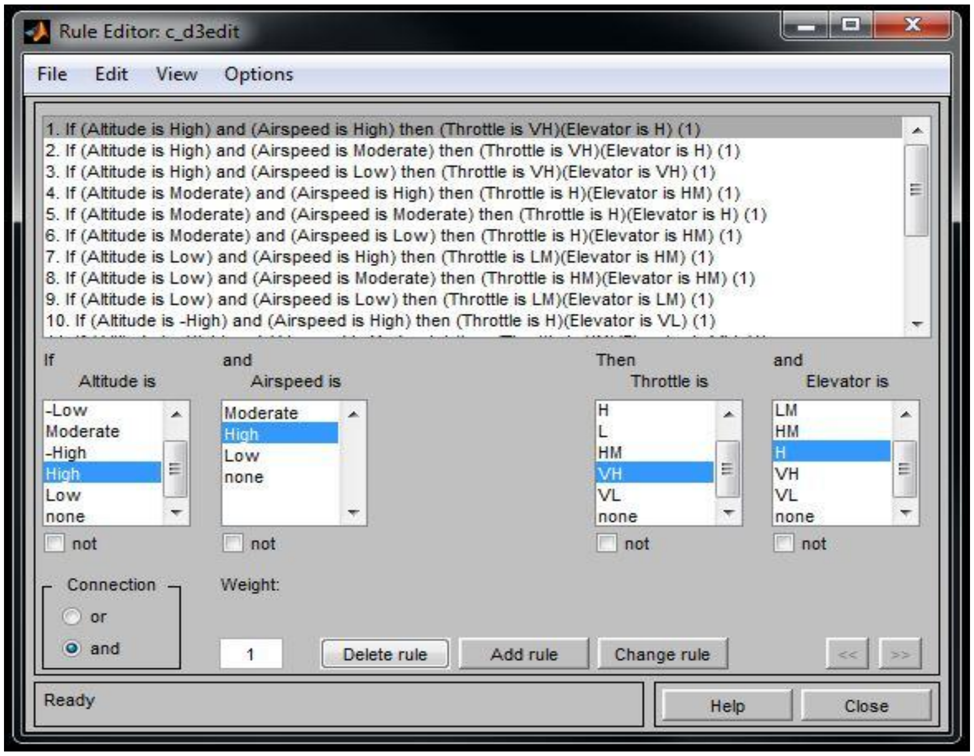
Right Turn:	Steep turn	=	**ΔHeading = + High**
	Moderate turn	=	**ΔHeading = + Moderate**
	Shallow turn	=	**ΔHeading = + Low**

No additional rules for "Straight and Level flight" are required since the straight and level flight is performed by default while using these rules.

In addition to the altitude, during the climbs and descents, another factor needs to be considered, none other than "Airspeed". The airspeed varies drastically while the plane gets to an inclined position. So, the rules are designed in such a way that the altitude and airspeed factors contribute to the climb and descent paths. Along with heading, another factor is considered for the turns and that is "Angle of Bank" or the roll angle. There are always additional options to add or remove the number of inputs for a smoother control of the aircraft. Here are some sample rules from climbs and descents –

If (Altitude is High) & (Airspeed is Moderate)	**=>**	**(Throttle is VH) & (Elevator is H)**
If (Altitude is Moderate) & (Airspeed is Low)	**=>**	**(Throttle is VH) & (Elevator is H)**
If (Altitude is Low) & (Airspeed is High)	**=>**	**(Throttle is LM) & (Elevator is HM)**
If (Altitude is -High) & (Airspeed is Low)	**=>**	**(Throttle is H) & (Elevator is VL)**
If (Altitude is -Moderate) & (Airspeed is High)	**=>**	**(Throttle is HM) & (Elevator is L)**
If (Altitude is -Low) & (Airspeed is Low)	**=>**	**(Throttle is L) & (Elevator is LM)**

The membership functions assigned to each of the input and output parameters vary according to their function. ‘Altitude’ is expressed with six membership functions: **High**, **Moderate**, **Low**, **-High**, **-Moderate** and **–Low**. The first three represent Steep, Moderate and Shallow Climbs where as the latter three represent Steep, Moderate and Shallow Descents. ‘Airspeed’ is defined by three membership functions which represent **High**, **Moderate** and **Low ranges**. ‘Throttle’ and ‘Elevator’ are the main actuator controls during a climb or a descent. They are assigned with six membership functions each which are defined as **VL – Very Low**, **L – Low**, **LM – Low Moderate**, **HM – High Moderate**, **H – High** and **VH – Very High** of which three are used during the climbs and three during the descents. MATLAB’s Fuzzy Logic Toolbox provides a very convenient user interface to execute these rules. The “Rule Editor” in Matlab which stores the required rules for a FIS is shown in Figure 7.



1. If (Altitude is High) and (Airspeed is High) then (Throttle is VH)(Elevator is H) (1)
2. If (Altitude is High) and (Airspeed is Moderate) then (Throttle is VH)(Elevator is H) (1)
3. If (Altitude is High) and (Airspeed is Low) then (Throttle is VH)(Elevator is VH) (1)
4. If (Altitude is Moderate) and (Airspeed is High) then (Throttle is H)(Elevator is HM) (1)
5. If (Altitude is Moderate) and (Airspeed is Moderate) then (Throttle is H)(Elevator is H) (1)
6. If (Altitude is Moderate) and (Airspeed is Low) then (Throttle is H)(Elevator is HM) (1)
7. If (Altitude is Low) and (Airspeed is High) then (Throttle is LM)(Elevator is HM) (1)
8. If (Altitude is Low) and (Airspeed is Moderate) then (Throttle is HM)(Elevator is HM) (1)
9. If (Altitude is Low) and (Airspeed is Low) then (Throttle is LM)(Elevator is LM) (1)
10. If (Altitude is -High) and (Airspeed is High) then (Throttle is H)(Elevator is VL) (1)

If Altitude is	and Airspeed is	Then Throttle is	and Elevator is
-Low	Moderate	H	LM
Moderate	High	L	HM
-High	Low	HM	H
High	none	VH	VH
Low		VL	VL
none		none	none
not	not	not	not

Selected: Altitude is High; Airspeed is High; Throttle is VH; Elevator is H

Connection: or [✓ and]

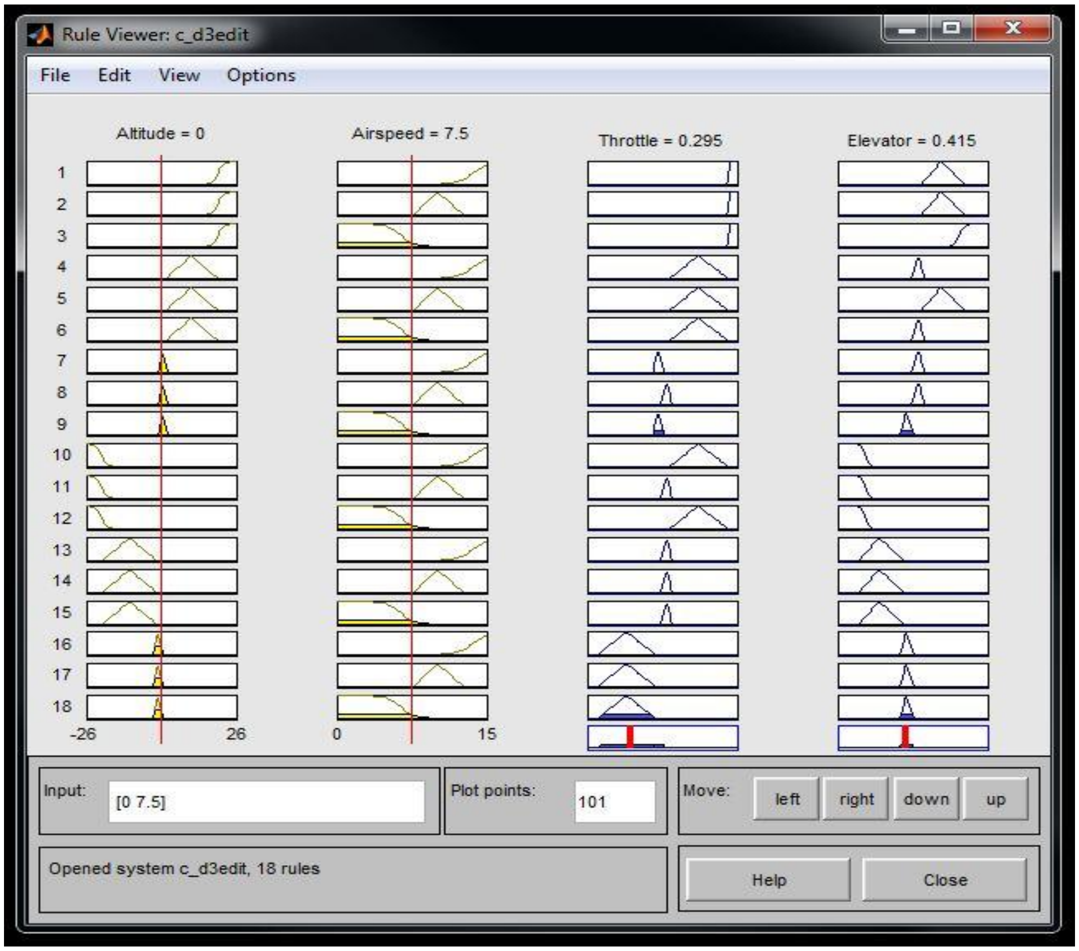
Weight: 1

Ready

Figure 7: Rule Editor (from MATLAB Fuzzy Logic Toolbox)

This rule editor is used to modify the rules of a FIS structure stored in a file. All the inputs, outputs and their corresponding membership functions are defined before the use of the rule editor. These are then combined with the corresponding logical operator and then implication methods to form the actual rule-base.

A fuzzy inference diagram for a FIS stored in a file can be depicted with the help of the "Rule Viewer" as shown in the Figure 8 -



[0 7.5]

101

Opened system c_d3edit, 18 rules

Figure 8: Rule Viewer (from MATLAB Fuzzy Logic Toolbox)

The rule viewer is used to view the entire implication process from the beginning to the end. The four plots across the top of the figure shown in Figure 8 represent the antecedent and consequent of a particular rule. Each column represents a variable and each row of plots represents a rule.

The first two columns which are in yellow represent the antecedent of a rule (the IF part) and the columns which are in blue (third and fourth columns) represent the consequent of a rule (the THEN part). Since the decision will depend on the input values for the system, the line indices that correspond to the input can be moved around and the system can be seen readjusting and computing the new output. The defuzzified output is represented as a bold vertical (red) line at the bottom in the columns corresponding to the consequent.

4.2 Simulink Blocks

The purpose of developing a simulation environment for the UAV is to provide a platform for testing and tuning algorithms and to foresee the behavior of the UAV prototype in different scenarios. The simulation can also be used as part of the ground station software to control the UAV.

Using PID controllers for control, stability and navigation has been considered as one of the traditional approaches for autopilot development in Unmanned Aerial Vehicles. This approach has also been followed by the Institute for Artificial Intelligence at UGA. A simulation environment has been developed by former UAV team members (Uthayasanker Thayasivam – a current Computer Science Ph.D graduate and Rahul Lakshmanan – a former Artificial Intelligence Master’s student) in order to provide a platform for testing and tuning algorithms and observing the behavior of ongoing UAV prototypes in different possible scenarios. In order to build a base for the development of intelligent systems, the Fuzzy Logic approach has been considered by replacing the traditional PID controllers with Fuzzy Inference Systems. Figure 9 depicts the current UAV Simulink model in Matlab -

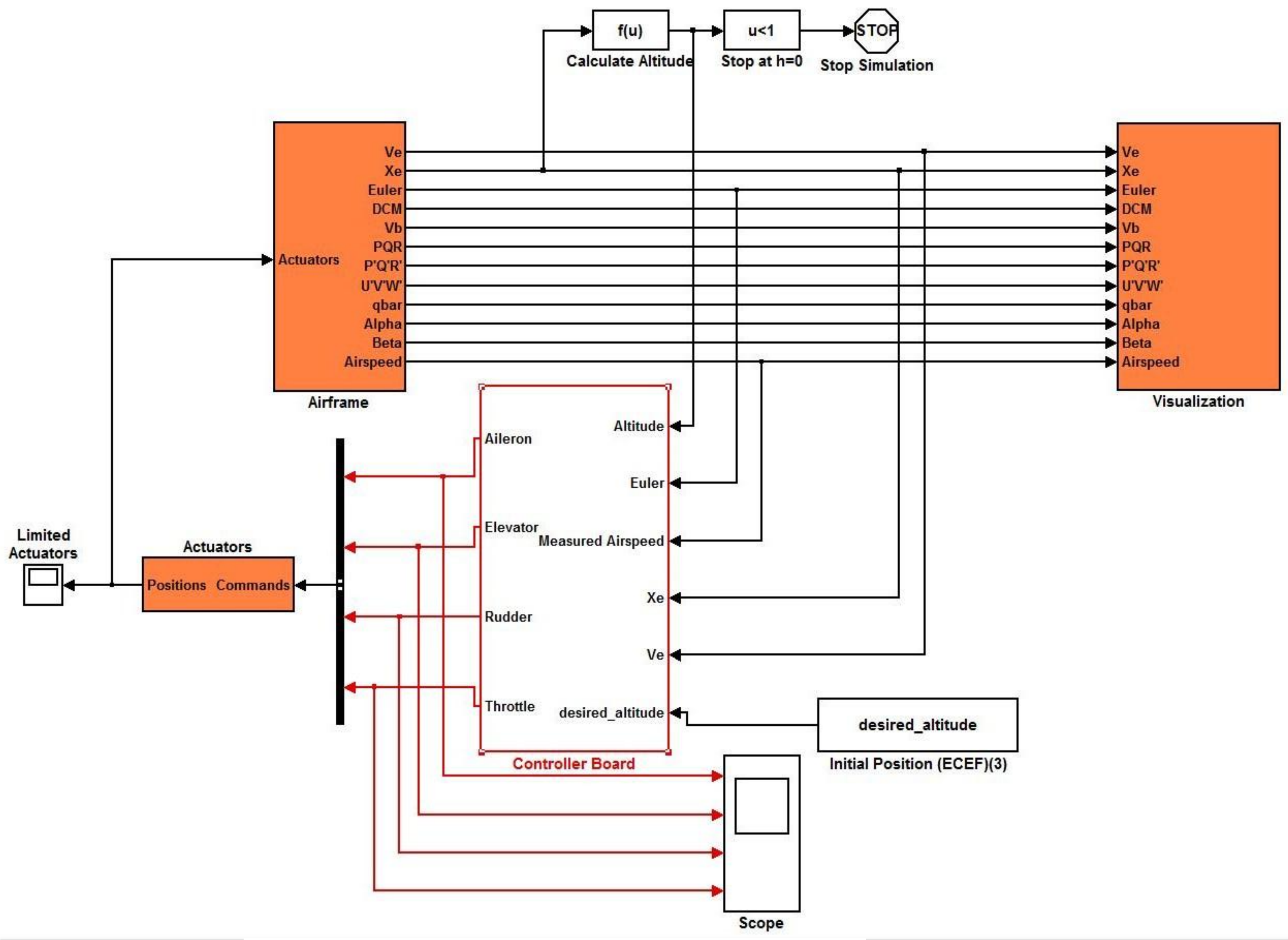


Figure 9: MATLAB Simulink Model

This section describes the development of a simulation of a UAV, its controller and the environment in which it operates. The simulation is developed using Mathworks' Simulink software and FlightGear. Construction of the simulation satisfies the need for a simulation environment that researchers can use to implement and analyze their models and algorithms. FlightGear is used to animate and visualize the simulation and Simulink is used to model the physics and dynamics of the airframe, actuators, sensors, environment (wind, noise and other disturbances), and the controller.

The simulation environment has been considered as ground software to control the UAV and has been classified into three key components:

- **Aerodynamics and Environment model** which contains the 'Airframe' and 'Actuators' blocks. This model is used to characterize the dynamics of a UAV in its environment.
- **Controller Model** which consists of the 'Controller Board', a combination of two fuzzy controllers, one for Turns (lateral controller) and one for Climbs&Descents (longitudinal controller). This model is used for Guidance, Navigation, Control & Stability, and Communications.
- **Visualization Model** communicates with Flight Gear and produces useful graphical outputs to analyze and monitor the motion of the aircraft.

The Aerodynamics & Environment Model:

The effect of the environment on the motion of the simulated UAV is modeled in this simulation block. The actuator outputs from the UAV controller are passed to the Actuators which are modeled as Second Order Nonlinear Actuators (explained in later sections). The actuators in the UAV are throttle, elevator, aileron, and rudder. The effects of the actuators and the environment including gravitational forces, wind forces, aerodynamic forces and moments are taken into account to generate the total force and moment on the Airframe of the UAV. The resultant force and moment acting on an aircraft are calculated using the power generated by the actuators and the physical attributes of the vehicle. This resultant force and moment on the aircraft determines its position and orientation. The position of the aircraft and its orientation are the values of the six degrees of freedom (6-DOF) that the UAV enjoys. The six degrees of freedom of the UAV are its latitude, longitude, altitude and the angular positions of roll, pitch and yaw. Since our UAV does not have a rudder, the rudder output in the simulation model will always be null or 0. The Cessna C172P being used for simulation does have a rudder but it is set to 0 to suit the current UAV model.

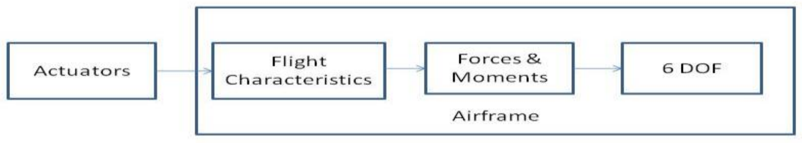


Figure 10: Aerodynamics and Environment Model

Airframe:

The UAV employs a sophisticated set of pre-calculated physics equations to calculate rigid-body parameters for use in modeling the 6-DOF flight dynamics of the aircraft. The dynamics of the UAV aircraft used have been decoupled into lateral and longitudinal dynamics. The lateral dynamics are the aircraft's response along the roll and yaw axes and the longitudinal dynamics are the aircraft's response along the pitch axis. The 'Airframe' block which is present on the left side in Figure 9 is responsible to deliver the current state of the aircraft to the controller forming a feedback loop control system. This block had been developed by a team of three (Robert Eunice, Uthayasanker Thayasivam and Rahul Lakshmanan) who initially planned to test the model using PID controllers. The *thrust, force & momentum, airspeed and dynamic pressure* are all employed with functional blocks where the parameters can be specified.

The **International Standard Atmosphere (ISA)** is an atmospheric model of how pressure, temperature, density, and viscosity of the Earth's atmosphere change over a wide range of altitudes. Here, the ISA model has been computed for altitudes between 0Km and 20Km using a lapse rate method. The WGS84 gravity model has been used to calculate the earth's gravity at a certain location. The position, altitude and latitude are all again defined by the functional block parameters. The 6-DOF characteristics will be calculated using the pre-defined *6-DOF (Quaternion) block* provided by the Simulink Block Library, taking the resultant momentum and force in.

Actuators:

The ‘Actuators’ block (at the left corner in Figure 9) accepts input commands from the controller and returns the position of the actuators to the airframe. It appears as shown below in Figure 11 when expanded –

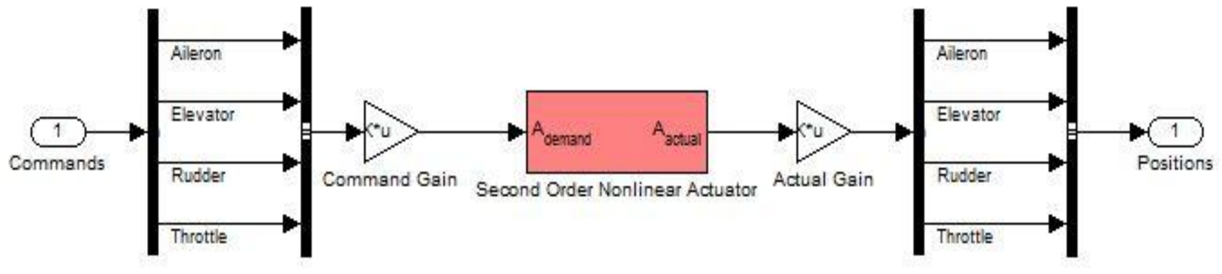


Figure 11: Actuators Block

<u>Controller Model:</u> (Controller Board)

The ‘Controller Board’ block present in Figure 9 appears as depicted in Figure 12 when expanded. It reads the current state of the aircraft from the airframe through the feedback system and sends output signals to the control actuators to meet its goals. This block which was a network of PID controllers has been replaced with Fuzzy Controllers.

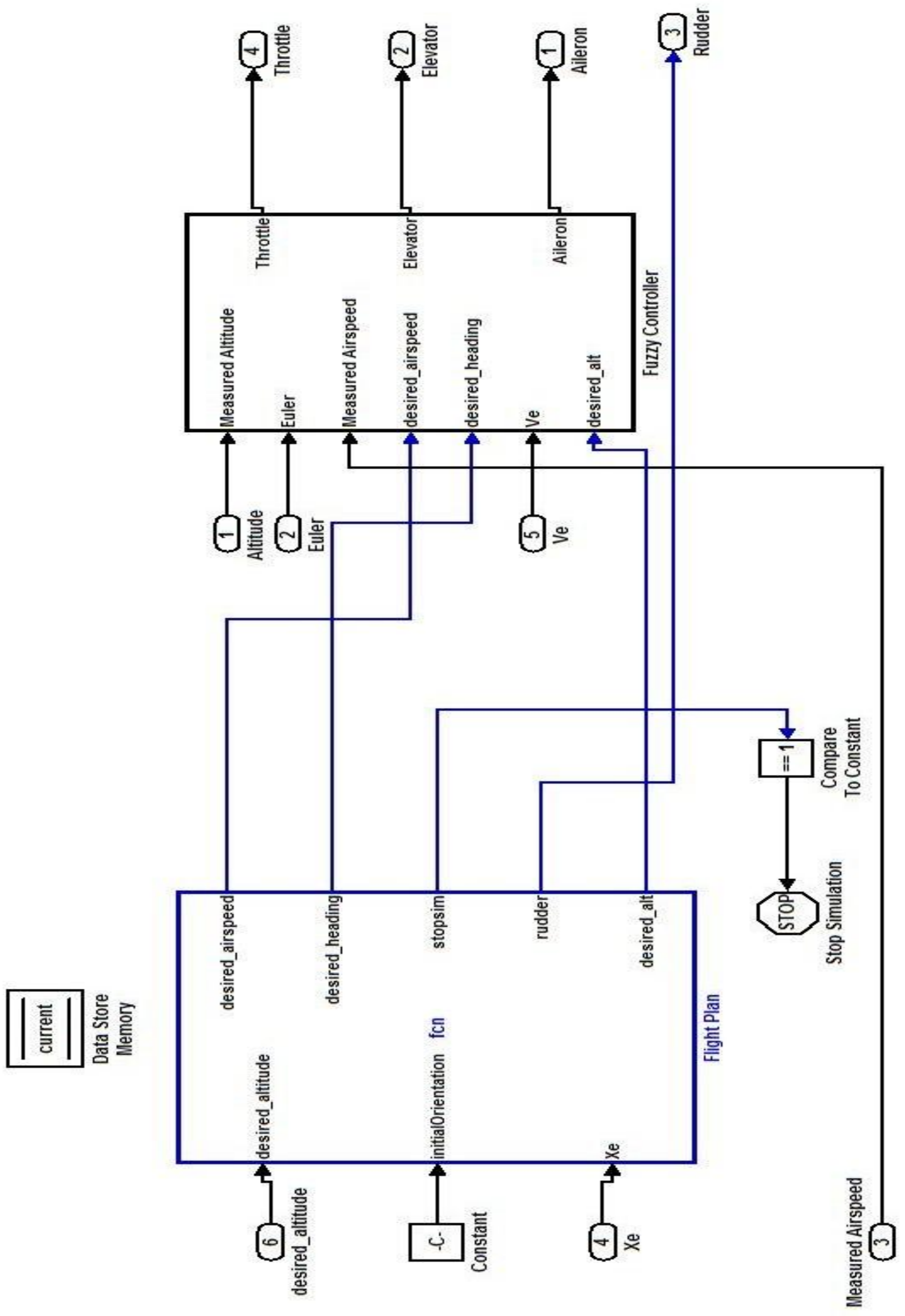


Figure 12: Controller Board

The controller board consists of two subsystem blocks – The ‘Flight Plan’ block and the ‘Fuzzy Controller’ block. The top-level set points for the fuzzy controllers are the three variables: desired altitude, desired heading and desired airspeed. These are set by the Flight Plan block based on the current plan and fed into the controller (present in the ‘Fuzzy Controller’ block). Waypoint Navigation is also being managed by the Flight Plan. The waypoints are defined as parameters (and not inputs or outputs) to the Flight Plan block in order to calculate the above three set-points. “Stop Simulation” is a MATLAB block which is used to end the simulation when it reaches the last waypoint mentioned in the FLIPS output file.

The desired airspeed varies according to the type of maneuver and tries to attain the desired constant value while moving in straight and level flight. Desired altitude can be set to a constant value or can be varied according to the user’s choice.

Fuzzy Controller Block:

The Fuzzy Controller consists of the Flight Manager block and the Controller block which are shown in Figure 13. As discussed earlier, the Flight Manager block sends commands to the controller by calculating the error between the desired and current state of the aircraft. It calculates the difference in altitude, difference in airspeed and the difference in heading between the desired and measured values.

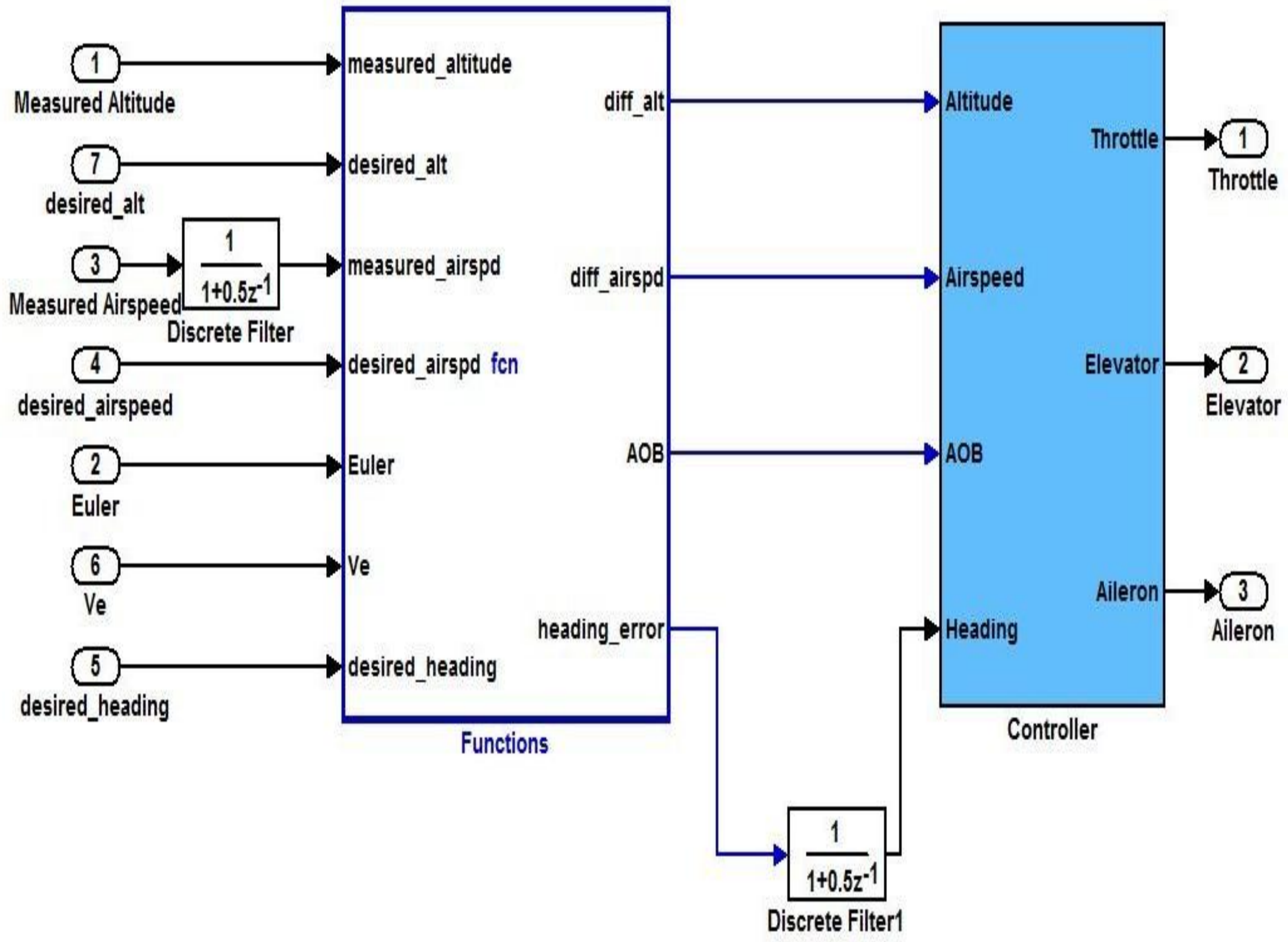


Figure 13: Fuzzy Controller Block

These are then passed on to the controllers as inputs. Based on these input values, the actuator control values will be sent to the actuators. The Fuzzy Inference Systems present in the controller block are made up of rules and fuzzy membership functions pertaining to the inputs and outputs. Figure 14 shows the fuzzy controllers used in this project (expansion of the ‘Controller Block’ in Figure 13)-

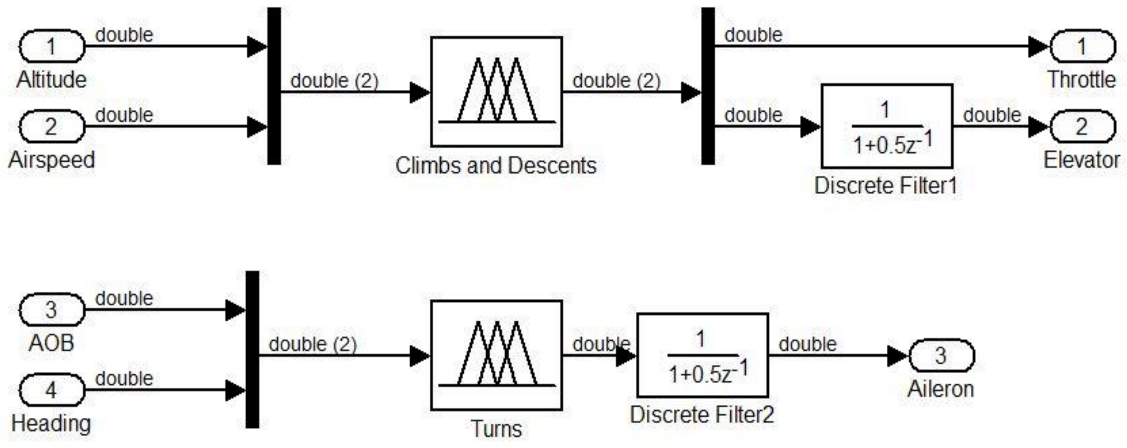


Figure 14: Fuzzy Controllers in the Controller Block

The Fuzzy Logic Controller blocks (Climbs and Descents, Turns) implement the fuzzy inference systems (FIS) in Simulink. ‘Climbs and Descents’ deals with the longitudinal control of the aircraft and ‘Turns’ deals with the lateral control of the aircraft. The discrete filters have been used in order to smooth out the values and for better control of the aircraft. They act as low pass filters filtering out the high frequency (out of the safety range) values to maintain the stability of the aircraft. The FIS is automatically generated as a hierarchical block diagram representation in the fuzzy logic controller. This automatic model generation ability is called the ‘Fuzzy Wizard’.

Visualization: Flight Gear

The visualization block (developed by Uthayasanker Thayasivam and Rahul Lakshmanan) is associated with the Flight Gear software. This block takes in the 12 input values needed to display the state of the aircraft and the 12 inputs are shown on the leftmost corner of the diagram in Figure 16. Like the one used in [14], the visualization block in this model has the 6-DOF block (top right corner in Figure 15) which displays the graphical representation of the 6-DOF parameters during simulation. The 'Flight Gear pre-configured 6-DOF animation' block (also on the right in Figure 15) gives the drive position and attitude values of the vehicle to a Flight Gear flight simulator. Units are degrees West/North for longitude and latitude; meters above mean sea level for altitude; and radians for attitude values. The 'Generate Run Script' block (rightmost in Figure 15) generates a custom run-script file on the current platform. Other blocks shown in the figure are the physical dynamics related to the aircraft and its environment. The aircraft used in this project is a Cessna C172P. The runway 28R at KSFO (San Francisco International) airport has been selected as the Flight Gear environment. The Flight Gear flight simulator environment can be seen in Figure 16.

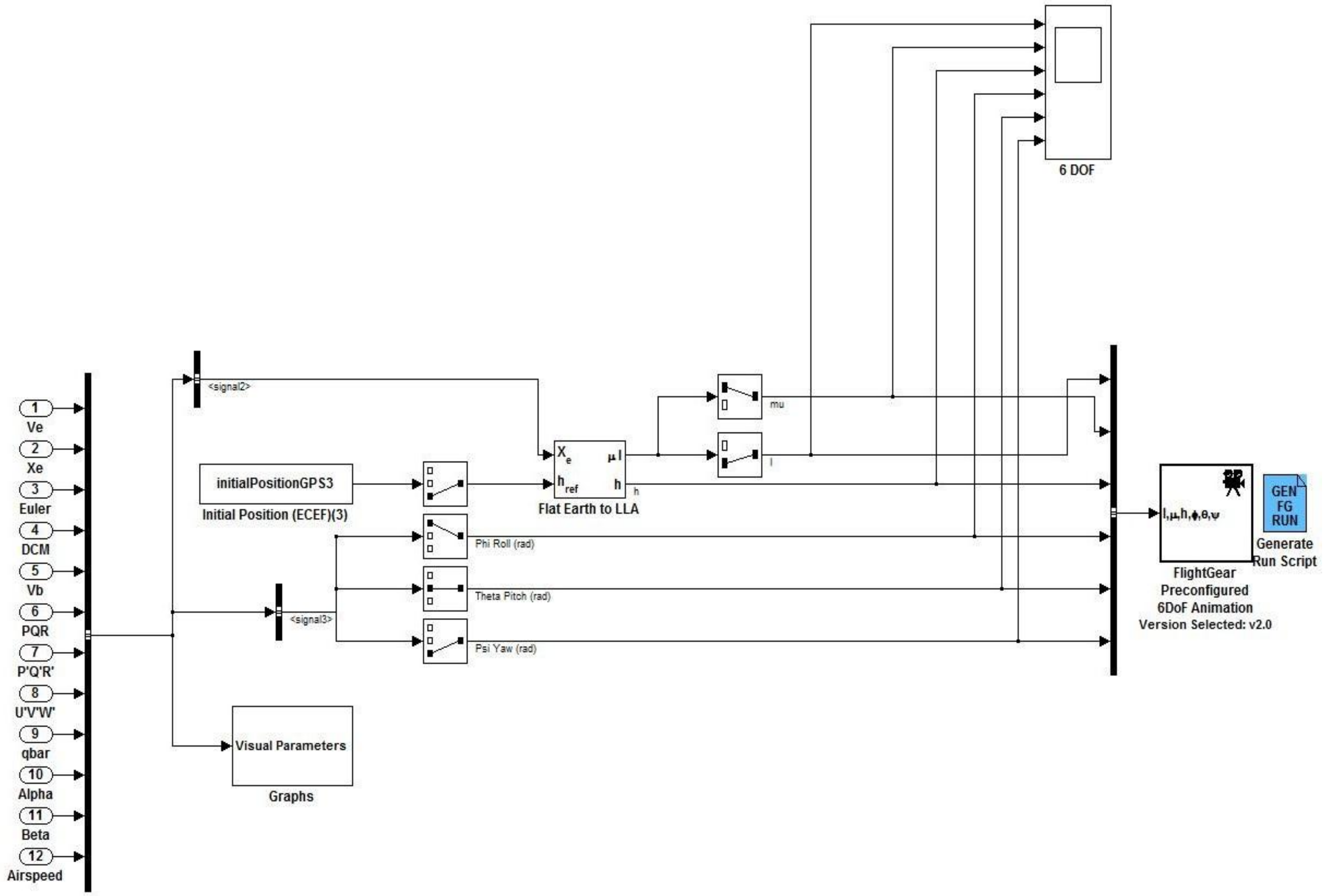


Figure 15: Visualization Block

FlightGear
File View Location Autopilot Environment Equipment ATC/AI Network Debug Help Cessna C172P

Figure 16: Flight Gear Flight Simulator

The whole model is simply a feedback loop control system with subsystems built into it. This feedback loop system reports the state of the aircraft every 0.005 sec which is said to be its simulation step time. The step time can be set according to the user's choice. It can be of fixed-type (the one used here is fixed) or variable-type. The step size changes for every step during the simulation in a variable type step time. The variable step solvers in Simulink vary the step time dynamically during the simulation. These solvers (variable and fixed step) increase or decrease the step size using their local error control to achieve the tolerances that the user specifies. The accuracy and the length of time of the resulting simulation depend on the size of the steps taken by the simulation: the smaller the step size, the more accurate the results are but the longer the simulation takes. A fixed step solver has been implemented in this project for faster simulation results. The ‘Scope’ block used in the simulation model (Figure 6) is used for visualizing the actuator control actions during the simulation. It displays the output with respect to the simulation time.

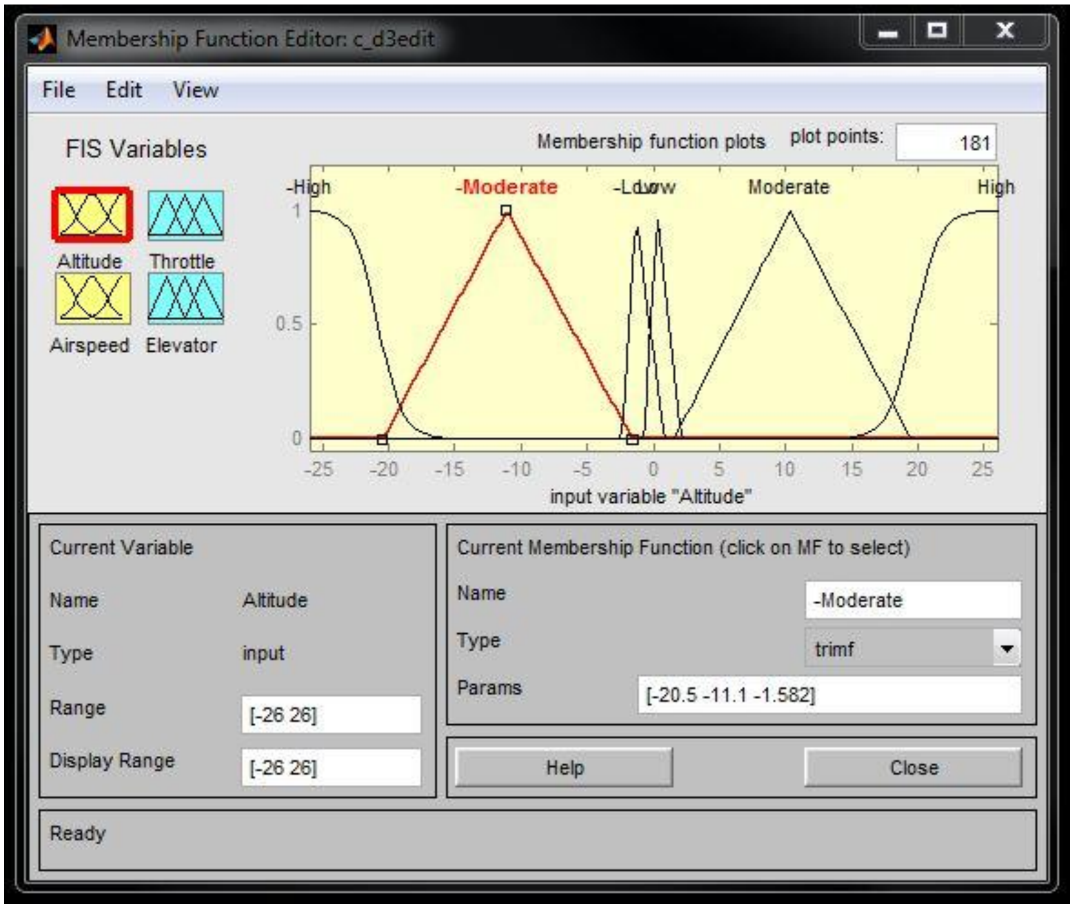
In the fuzzy controllers used, the membership functions are properly chosen and are assigned with membership values with intuition by observing the universes of discourse of the input and output variables. The aircraft would become highly unstable if the values out of the specified ranges (Table 1) are used. The input variables, output variables and their ranges for the two controllers are as shown in Table 1:

Variable	Range
Inputs	
Altitude	[-26 26]
Airspeed	[0 15]
Heading	[-1.5 1.5]
Angle of Bank	[-0.4 0.4]
Outputs	
Throttle	[0 1]
Elevator	[0 0.9]
Aileron	[-0.075 0.075]
Rudder	0

Table 1: Input and Output variables and their ranges

The above values were determined from the minimum and maximum ranges that can be handled by the actuators. These can be set or varied according to the safety ranges mentioned in the manufacturer's manual for the particular aircraft being used for implementation. A Cessna C172P has been selected as the aircraft for simulation purposes in this research. A typical low-cost mini-UAV is much smaller and lighter when compared to the one used for simulation.

Matlab's Fuzzy Logic Toolbox provides the implementation ground for fuzzy inference systems. This is further linked to the Simulink model explained in the previous sections to complete the feedback loop control system. Figure 17 depicts the 'membership function editor,' which stores the membership functions associated with the fuzzy sets being used in the controller.



plot points: 181

Current Variable
Name: Altitude
Type: input
Range: [-26 26]
Display Range: [-26 26]

Current Membership Function (click on MF to select)
Name: -Moderate
Type: trimf
Params: [-20.5 -11.1 -1.582]

Ready

Figure 17: Membership Function Editor

Figure 17 displays the input 'Altitude' and the membership function range with its defined fuzzy sets.

The type of fuzzy inference system used in this project is the Mamdani-type fuzzy inference due to its widespread acceptance and its intuitive nature suits human input. In Sugeno FIS, a large number of fuzzy rules must be employed to approximate periodic or highly oscillatory functions. The rule set used in this project suits the Mamdani style of FIS and hence the Mamdani method (Section 2.2) has been implemented.

CHAPTER 5

RESULTS

After the user defined goals are converted to tasks by the FLIPS parser, the autopilot begins execution by getting these tasks from a text file. MATLAB reads the text file and converts these tasks to fuzzy IF-THEN rules in the Fuzzy Inference System. According to the current state of the aircraft and the desired state of the aircraft, the FIS fires the rules for task execution. These are then applied to the control actuators (rules to actions) and fed back into the system according to the specified step time. This process is also visualized simultaneously with the help of Flight Gear. The X-Y graph (Ground Track) plot displays an X-Y plot of the position of the aircraft. These are represented as the latitude and longitude calculated from the X_e (in all XYZ directions) values (orientation of the aircraft) during the simulation.

The fuzzy logic approach appears to be presenting with interesting results in the sense that the autopilot obeys the instructions precisely according to the waypoints defined. It also corrects itself from small roll and pitch angle errors during straight and level flight. The autopilot has been tested with various trajectory paths and not just one to test for precision. A comparison of the results with the PID controller model has also been made in order to determine its performance. The vertical Figure 8 with the fuzzy control model and the PID control model are as shown in Figures 18 and 19:

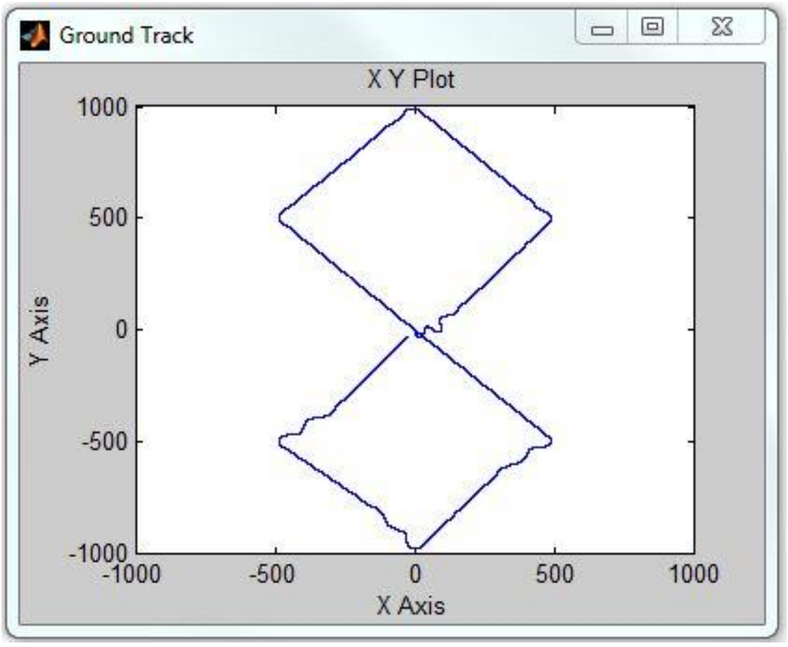


Figure 18: Vertical Figure 8 – Fuzzy Model

X-Axis – Matlab Units, Y-Axis – Matlab Units

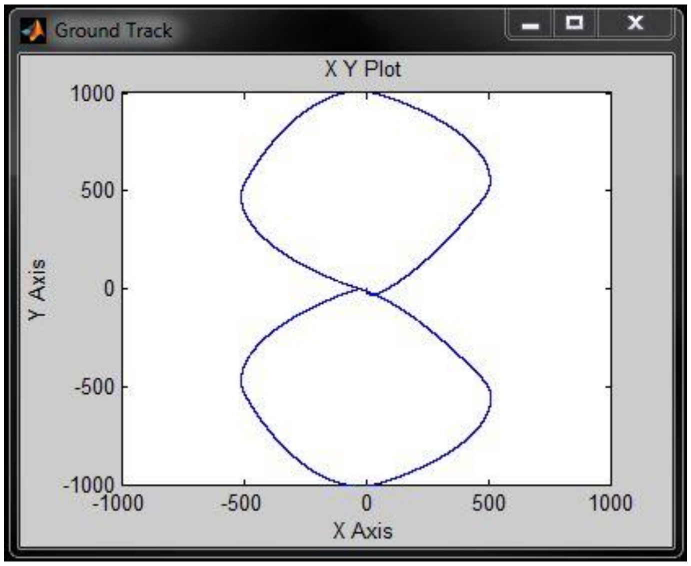


Figure 19: Vertical Figure 8 – PID Model

X-Axis – Matlab Units, Y-Axis – Matlab Units

The 3-dimensional perspective view of the above Figure 8 is as shown in Figure 20 –

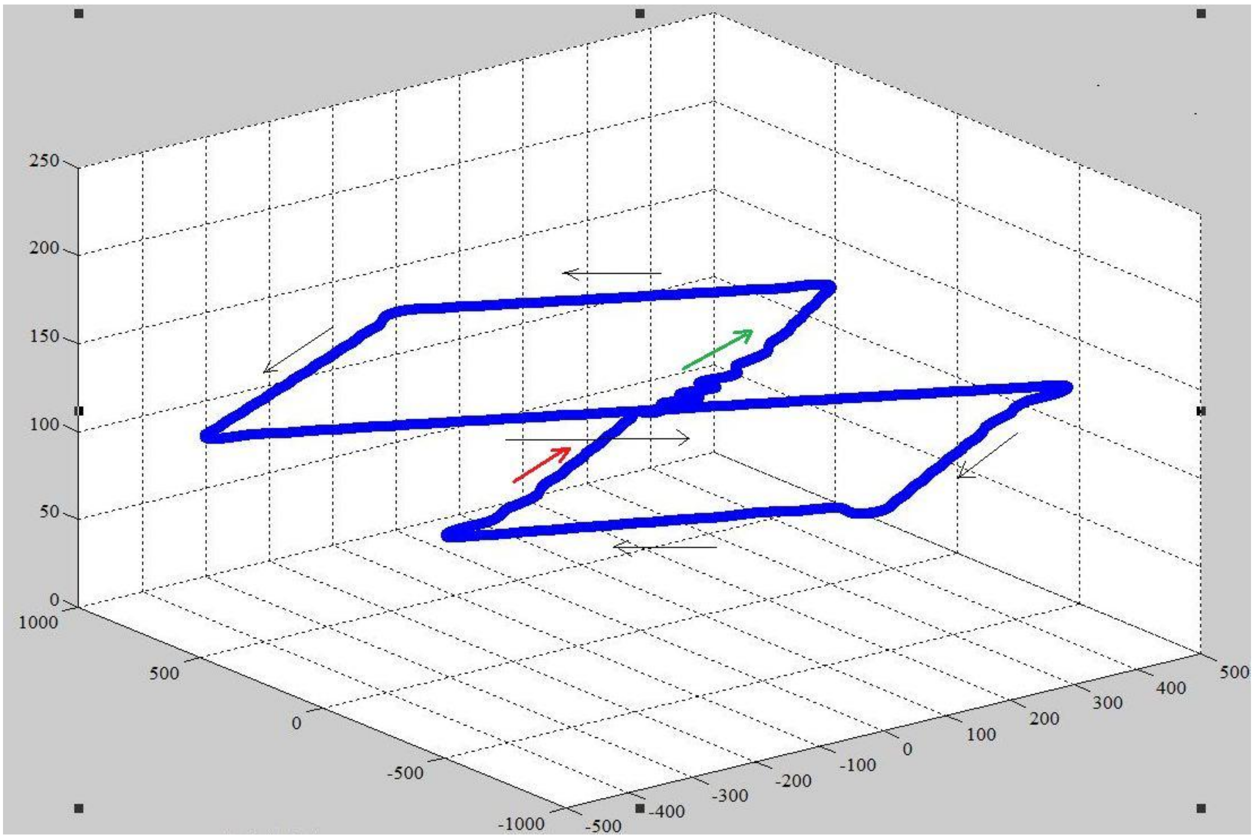


Figure 20: 3D perspective view of Vertical Figure 8 – Fuzzy Model
X-Axis – Matlab Units, Y-Axis – Matlab Units, Z-Axis – Altitude (m)

As depicted in Figure 20, the aircraft starts from position (0, 0) and continues to fly according to the waypoints defined in the goal. The axes in all the Figures depicting the aircraft trajectory define orientation of the aircraft. The XY plots display the Y and X orientation axes respectively and the 3D plots display all the three X (Longitudinal Axis), Y (Lateral Axis) and Z (Vertical Axis) axes respectively. The beginning and the end of the trajectory taken by the aircraft are marked with green and red arrows respectively.

The richness of FL is that there are enormous numbers of possibilities that lead to lots of different mappings [18], [12] and hence the mapping to the actuators gives out a different value for every simulation time step (though straight and level comes from the same fuzzy set). The adjustment of this rule-based controller for every time step creates the little distortion in its trajectory. This can be further improved by using some kind of tuning algorithms or by adding additional constraints with additional rules [24] [25].

Since the control for take-off and landing is currently not handled by the autopilot, the rules take action once the take-off is set and the aircraft reaches it desired altitude. The Figure 20 depicts the trajectory with a constant altitude of 125 m. If the altitude is not asked to be changed for every waypoint, the controller simply assigns the previous altitude value to the corresponding waypoint.

Figure 21 shows the deflection of the control surface when the aircraft is said to move in straight and level flight. This shows us the way each time the controller fires the rules according to the current state and the desired state of the aircraft. The signals represented in Figure 21 show the control surface (actuators) deflections during a straight and level flight. There are no special rules employed for the straight and level flight and that is the reason the small disturbance (up and down) in the aileron action is being displayed due to the aircraft's adjustment. Noise has not been included in this model since it is a simple demonstration of how an Unmanned Aerial Vehicle can be controlled using fuzzy logic controllers at the very basic level without the help of any traditional tuning techniques.

As the results show, the aircraft is performing the exact actions specified by the user. This one can be said to be a precision controlled aircraft system. Since this is a basic model of a fuzzy logic controller for aircraft control, it can be said that the drawback here is the perfect turning mechanism of this feedback loop control system. The turns taken by the fuzzy controller might get slightly complicated during practical implementations. However, this can be further modified to smoothen out the turns by adding some constraints on the input values, by tuning the membership values, or by adding additional rules.

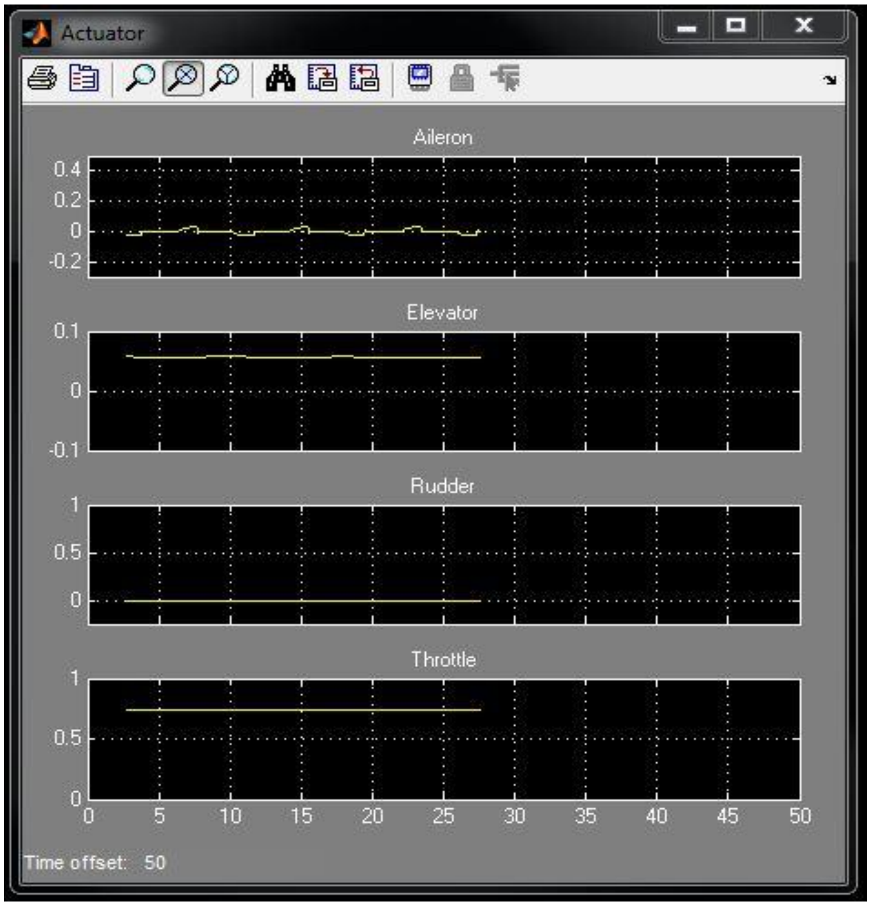


Figure 21: Control surface deflections during straight and level flight

The Figures 22 and 23 show another trajectory followed by the aircraft (all the trajectories depend on the goals specified by the user. The goals here imply latitude, longitude, altitude, airspeed etc.). The figure 24 shows a holding pattern for the Square and it is said to follow the second trajectory displayed in the figure if it is asked to follow the same path for more number of times. The figure shows only the two trajectories for a clearer view. All the desired information is gathered by the autopilot from the text file. The simulation is set in such a way that the aircraft always starts from the origin i.e., position (0, 0). For example, the desired latitude and longitude values for a simple trajectory which forms a square are: (500, 0), (500, 500), (0, 500), (0, 0).

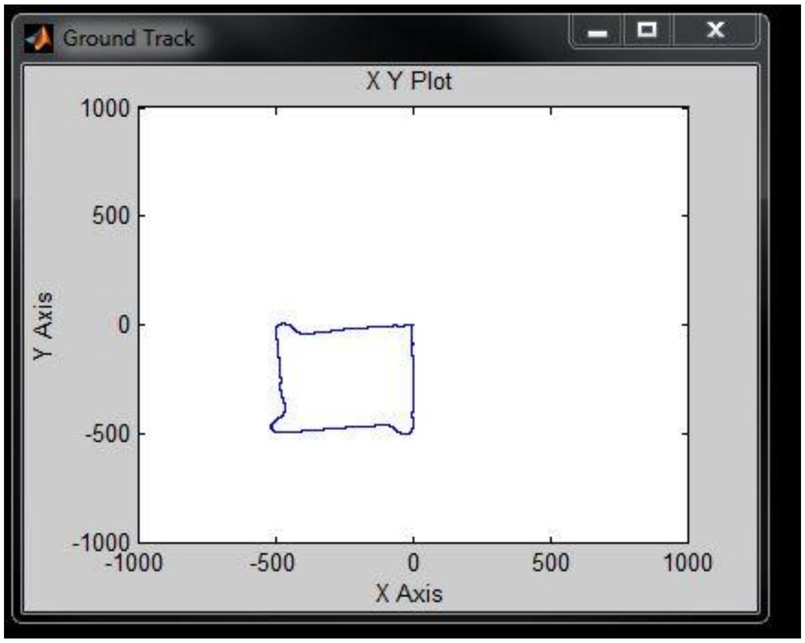


Figure 22: Square – Fuzzy Model

X-Axis – Matlab Units, Y-Axis – Matlab Units

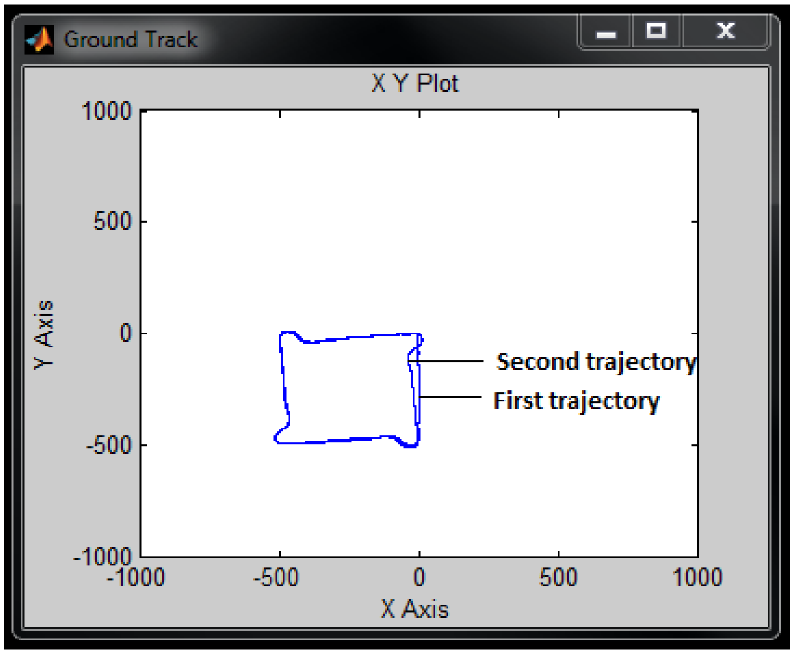


Figure 23: Holding pattern of a square

X-Axis – Matlab Units, Y-Axis – Matlab Units

Figure 24 displays the path of the trajectory followed by the UAV with change in altitude for every waypoint. The figure displays the 3D prospective view of the trajectory followed by the aircraft. As it is clear from the figure, the altitude increases from (0, 0) for the first half of the Figure 8 and starts decreasing after for the second half. The turns, as discussed earlier, appear to be distorted since there a change in altitude and change in the waypoint (heading) at the same time and it is a challenging task for any controller to handle the two and maintain stability at the same time. If there is additional tuning provided and more number of attributes involved, there might be a better turn scenario.

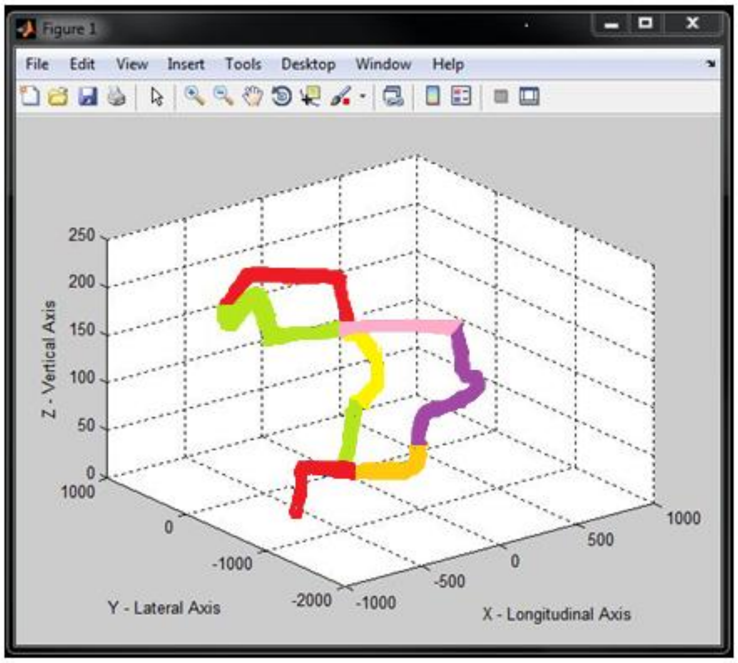


Figure 24: Figure 8 with change in altitude

X-Axis – Matlab Units, Y-Axis – Matlab Units, Z-Axis – Altitude (m)

Trajectory path – Green, Yellow, Red, Green, Pink, Purple, Mustard and Red

CHAPTER 6

CONCLUSION

The purpose of this research was to develop navigation and control based fuzzy logic controller for unmanned aerial vehicles. The simulation results depict an adequate overall performance of the autopilot. The results show some deflections during the turns which are due to the adjustment of the controller with the corresponding rules during every time step. Trial and error methods to fine tune them during the turns have been performed but it was disappointing to note that not much tuning was possible to make them smoother. Every method has its own disadvantages and this can be considered as a disadvantage of using fuzzy rules for smoothing the turns. However, this might be improved by employing a very few PID's for just tuning purposes during the turns. The perfect control flow of the architecture at present helps in developing a fuzzy model on the hardware much easier without the PID's.

CHAPTER 7

FUTURE WORK

The model aircraft being developed at the Institute for AI at UGA is a ‘Zagi’. Zagi is a low-cost, fast, light-weight aerobatic flying wing aircraft designed and recommended for all flyers, experts and beginners. Zagi has 2 elevons, one on each side of the wing. The target weight of a Zagi is 25.5oz. The airplane is designed to balance at 8” measured back from the nose. In order to achieve these two objectives, the Zagi pusher motor, micro servos, and an 1800mA NiMh 8 cell high rate battery pack must be used [2]. The Zagi model has not been used for simulation purposes due to lack of elevon controlled aircraft in the simulation environment. A similar model can be developed in the future with elevon controls. This can be further extended by using it as a surveillance aircraft system by connecting camera-like sensors on board.

The use of fuzzy controllers for the navigation and control along with PID’s for fine tuning purposes could be a good research topic. Also, the take-off, landing and tasks other than the navigation and control could be appended to this. This research follows waypoint navigation like most of the aircraft control systems. New navigation control methodologies also can be employed like learning the trajectory path while traversing it and coming up with a new set of points of interest.

Bibliography

1. FAA. (2004). *Airplane Flying Handbook.* U.S. Department of Transportation, Federal Aviation Administration, Airman.
2. (Assembly Manual) Retrieved July 2011, from http://www.zagi.com/pdf/zxs.pdf
3. *Fuzzy Logic*. (n.d.). Retrieved July 2011, from http://en.wikipedia.org/wiki/Fuzzy_logic
4. Chang, C.-L. (1997). *Fuzzy Logic Based Programming.* World Scientific Publishing Co. Pte. Ltd.
5. Kaehler, S. D. (n.d.). Retrieved July 2011, from seattlerobotics.org: http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part1.html#INTRODUCTION
6. Eunice, R. E. (2007, Oct 31). Development of Low-Cost Nonlinear Embedded Flight Control Architecture for Autonomous Unmanned Aerial Vehicles. *Institute for Artificial Intelligence, University of Georgia* .
7. Eunice, R. E. (n.d.). *Google Code*. Retrieved from FLIPS-UAV: http://code.google.com/p/flips-uav/
8. Grabowski, T. G., Frydrychewicz, A., Goraj, Z., & Suchodolski, S. (2006). MALE UAV design of an increased reliability level. *Aircraft Engineering and Aerospace Technology* , Vol. 78 Iss: 3, pp.226 - 235.
9. M. Quigley, Michael A. Goodrich, Stephen Griffiths, Andrew Eldredge, Randal W. Beard (2005). Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera. *Proceedings of the 2005 IEEE International Conference on In Robotics and Automation*, (pp. 2600-2605).

10. Bryan E. Walter, Jared S. Knutzon, Adrian V. Sannier, James H. Oliver. (2004). VR Aided Control of UAVs. *3rd AIAA Unmanned Unlimited Technical Conference.* Chicago.
11. F. Hoffmann, Gerd Pfister. (1996). Evolutionary Learning of Fuzzy Control Rule Base for an Autonomous Vehicle. *Sixth Int. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (pp. pp. 1235-1240). Granada: (IPMU'96).
12. Omer Cetin, Sefer Kurnaz, Okyay Kaynak. (January 2011). Fuzzy Logic Based Approach to Design of Autonomous Landing System for Unmanned Aerial Vehicles. *Journal of Intelligent and Robotic Systems* , Volume 61 Issue 1-4.
13. Marin, J., Radtke, R., Innis, D., Barr, D., & Schultz, A. (1999). Using a genetic algorithm to develop rules to guide unmanned aerial vehicles. *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, (pp. 1055 - 1060 vol.1). Tokyo.
14. Sorton, E. F., & Hammakeer, S. (September 2005). Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle using Flight Gear. *American Institute of Aeronautics and Astronautics, AIAA 2005-7083.*
15. Sivanandam, S., S.Sumathi, & S.N.deepa. (n.d.). Introduction to Fuzzy Logic. Retrieved August 2011, from Introduction to Fuzzy Logic.
16. Passino, K. (June 1995). Intelligent Control for Autonomous Systems. *Spectrum, IEEE* , Volume: 32 Issue: 6, pp.55 - 62.
17. B. T. Clough, “Unmanned Aerial Vehicles: Autonomous Control Challenges, a Researcher’s Perspective,” in Cooperative Control and Optimization, R. Murphey and P. M. Pardalos, eds., pp. 35—53, Kluwer Academic Publishers, 2000.

18. Mendel, J. M. (March 1995). Fuzzy logic systems for engineering: A tutorial. *Proc. IEEE*, (pp. Vol. 83, pp. 345-377).

19. Vaishnav, S., & Khan, Z. (October 24-26, 2007). Design and Performance of PID and Fuzzy Logic Controller with Smaller Rule Set for Higher Order System. *Proceedings of the World Congress on Engineering and Computer Science.* San Francisco, USA: WCECS 2007.

20. Kurnaz, S., Cetin, O., & Kaynak, O. (2009). Fuzzy Logic Based Approach to Design of Flight Control and Navigation Tasks for Autonomous Unmanned Aerial Vehicles. *J Intell Robot Syst* , 54:229–244.

21. Thrift, P. (1991). Fuzzy Logic Synthesis with genetic algorithms. *Proc. Fourth Int. Conf. on Genetic Algorithms (ICGA'91)* (pp. pp. 509–513.). San Diego, USA: Morgan Kaufmann, Los Altos, CA.

22. Castro, J. L. (April 1995). Fuzzy Logic Controllers Are Universal Approximators. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, (pp. VOL 25, NO. 4).

23. Lee, C. C. (March/April 1990). Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I. *IEEE Transactons on Systems, Man and Cybernetics* , (p. Vol. 2; No.2).

24. Z. W. Woo, H. Y. Chung, and J. J. Lin, "A PID type fuzzy controller with self-tuning scaling factors," *Fuzzy Sets Syst.*, vol. 115, no. 2, pp. 321–326, 2000.

25. W. C. Daugherity, B. Rathakrishnan, and J. Yen, "Performance evaluation of a self-tuning fuzzy controller," in *Proc. IEEE Int. Conf. Fuzzy Systems*, San Diego, CA, Mar. 1992, pp. 389–397

26. A. Arslan and M. Kaya, "Determination of fuzzy logic membership functions using genetic algorithms," *Fuzzy Sets Syst.*, vol. 118, no. 2, pp. 297–306, 2001

27. Seraji, H., & Howard, A. (June 2002). Behavior-Based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach. *IEEE Transactions on Robotics and Automation* , Vol. 18, No.2.

28. A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," J. Soft Comput., vol. 1, no. 4, pp. 180–197, 1997.

29. L. A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

30. Bart K. and Satoru I. (1993), "Fuzzy Logic", retrieved from http//:Fortunecity.com/emachines/e11/86/fuzzylog.html.

31. M. Quigley, M. Goodrich, and R. Beard, "Semi-Autonomous Human- UAV Interfaces for Fixed-Wing Mini-UAVs," IROS 2004.

32. HaiYang Chao, YongCan Cao, and YangQuan Chen, "Autopilots for Small UnmannedAerial Vehicles: A Survey", International Journal of Control, Automation, and Systems (2010) 8(1):36-44.

33. R. S. Christiansen. Design of an autopilot for small unmanned aerial vehicles. Master's thesis, Brigham Young University, 2004.

34. Christophersen, H.B., Pickell, W.J., Koller, A.A., Kannan, S.K., and Johnson, E.N., "Small Adaptive Flight Control Systems for UAVs using FPGA/DSP Technology," *Proceedings of the AIAA Unmanned Unlimited Technical Conference, Workshop, and Exhibit*, 2004.

35. E.N. Johnson, D.P. Schrage,"The Georgia Tech Unmanned Aerial Research Vehicle: GTMax", *in Proceedings of the AIAA Guidance, Navigation, and Control Conference 2003*, AIAA Paper 2003-5741.

36. Mahmood, M. M., Yousaf, U. M., Chowdhury, M. S., Afifi, M. W., Ihsan, R., & Chuwdhury, I. M. (August 30 - September 2, 2009). UAV Autopilot Design for the AUVSI, UAS International Competition. *Proceedings of the ASME 2009 International Design Technical Conferences & Cpmputers and Information in Engineering Conference IDETC/CIE 2009.* San Diego, California, USA.

37. Beard, et al. (2005). Autonomous Vehicle Technologies for Small Fixed Wing UAVs'. *AIAA Journal of Aerospace, Computing, Information, and Communication*.

38. Oh, C.K., Barlow, G.J.: Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In: Proceedings of the Congress on Evolutionary Computation. (2004)

39. Peri, V. M. and Simon, D., *2005*. Fuzzy Logic Control for an Autonomous Robot, *North American Fuzzy Information Processing Society, NAFIPS 2005 Annual Meeting*, 337-342.

40. Ang, K.H. and Chong, G.C.Y. and Li, Y. (2005) PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology* 13(4):pp. 559-576. –

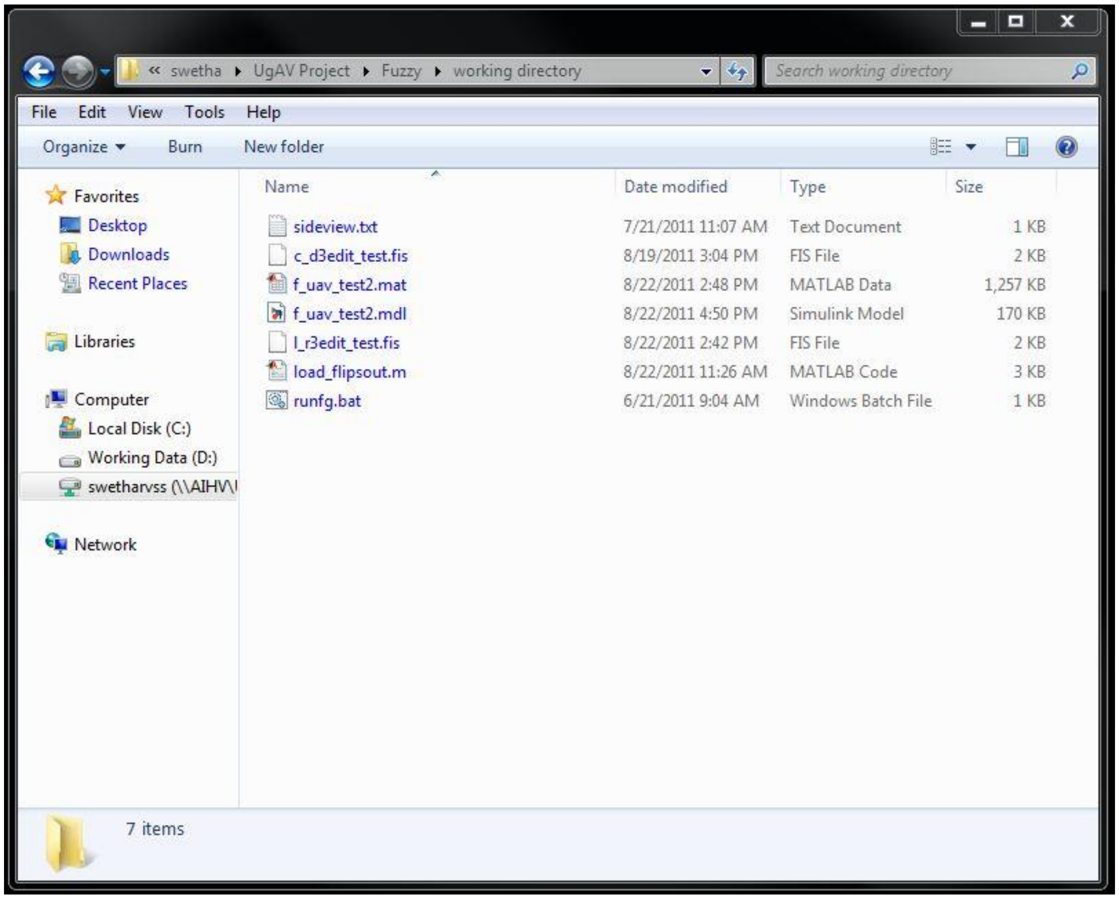
41. L. X. Wang. Fuzzy systems are universal approximators. In Proc. of the IEEE Int.Conf. on Fuzzy Systems, pages 1163–1169, San Diego, 1992.

USER'S GUIDE

RUNNING THE SIMULATION

In order to run the simulation, the following files should be in the working directory:

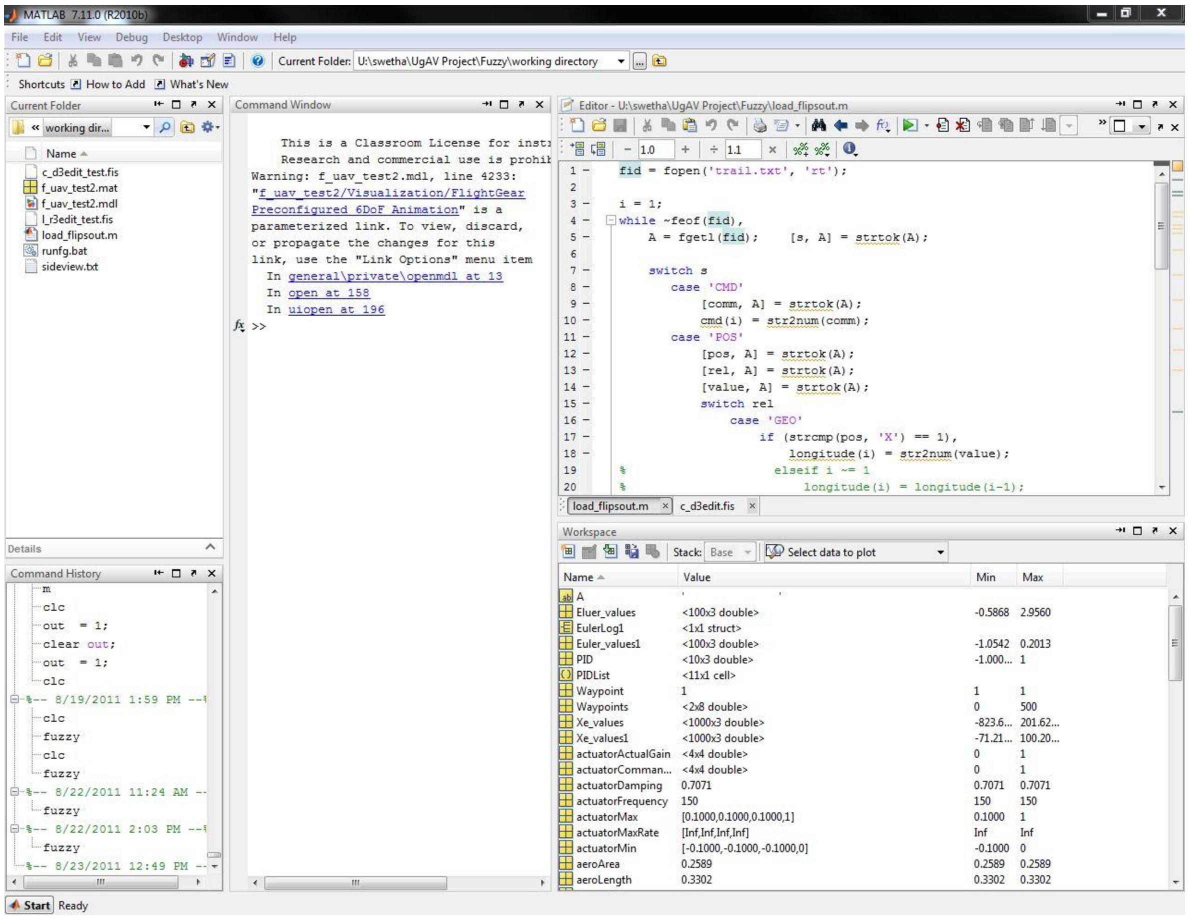
- MATLAB simulation file (extension .mdl)
- MATLAB Workspace variable file (extension .mat)
- FLIPS output file (extension .txt)
- The file that converts FLIPS output to high-level commands (extension .m)
- Fuzzy Inference System files (extension .fis)
- Flight Gear run script (runfg.bat)



Name	Date modified	Type	Size
sideview.txt	7/21/2011 11:07 AM	Text Document	1 KB
c_d3edit_test.fis	8/19/2011 3:04 PM	FIS File	2 KB
f_uav_test2.mat	8/22/2011 2:48 PM	MATLAB Data	1,257 KB
f_uav_test2.mdl	8/22/2011 4:50 PM	Simulink Model	170 KB
l_r3edit_test.fis	8/22/2011 2:42 PM	FIS File	2 KB
load_flipsout.m	8/22/2011 11:26 AM	MATLAB Code	3 KB
runfg.bat	6/21/2011 9:04 AM	Windows Batch File	1 KB

Figure 25: Required Files

- Open the MATLAB Simulation file (.mdl).
- Within MATLAB, load the workspace variables from the .mat file and the FLIPS command conversion code in the .m file by double-clicking them from the Current Folder which is located on the top left.



```
    This is a Classroom License for instr
    Research and commercial use is prohib
Warning: f_uav_test2.mdl, line 4233:
"f_uav_test2/Visualization/FlightGear
Preconfigured 6DoF Animation" is a
parameterized link. To view, discard,
or propagate the changes for this
link, use the "Link Options" menu item
  In general\private\openmdl at 13
  In open at 158
  In uiopen at 196
>>
```

```
fid = fopen('trail.txt', 'rt');

i = 1;
while ~feof(fid),
    A = fgetl(fid);    [s, A] = strtok(A);

    switch s
       case 'CMD'
           [comm, A] = strtok(A);
           cmd(i) = str2num(comm);
       case 'POS'
           [pos, A] = strtok(A);
           [rel, A] = strtok(A);
           [value, A] = strtok(A);
           switch rel
               case 'GEO'
                   if (strcmp(pos, 'X') == 1),
                       longitude(i) = str2num(value);
%                    elseif i ~= 1
%                        longitude(i) = longitude(i-1);
```

```
m
clc
out  = 1;
clear out;
out  = 1;
clc
%-- 8/19/2011 1:59 PM --%
clc
fuzzy
clc
fuzzy
%-- 8/22/2011 11:24 AM --
fuzzy
%-- 8/22/2011 2:03 PM --%
fuzzy
%-- 8/23/2011 12:49 PM --
```

Name	Value	Min	Max
A	' '		
Eluer_values	<100x3 double>	-0.5868	2.9560
EulerLog1	<1x1 struct>		
Euler_values1	<100x3 double>	-1.0542	0.2013
PID	<10x3 double>	-1.000...	1
PIDList	<11x1 cell>		
Waypoint	1	1	1
Waypoints	<2x8 double>	0	500
Xe_values	<1000x3 double>	-823.6...	201.62...
Xe_values1	<1000x3 double>	-71.21...	100.20...
actuatorActualGain	<4x4 double>	0	1
actuatorComman...	<4x4 double>	0	1
actuatorDamping	0.7071	0.7071	0.7071
actuatorFrequency	150	150	150
actuatorMax	[0.1000,0.1000,0.1000,1]	0.1000	1
actuatorMaxRate	[Inf,Inf,Inf,Inf]	Inf	Inf
actuatorMin	[-0.1000,-0.1000,-0.1000,0]	-0.1000	0
aeroArea	0.2589	0.2589	0.2589
aeroLength	0.3302	0.3302	0.3302

Figure 26: MATLAB Workspace

- Open the FlightGear run script (runfg.bat) to be able to see the visualization of the simulation.
- Run the simulation in Matlab Simulink by clicking on *Simulation* in the Menu bar of the Simulink model and then click *Start*.
- Maximum simulation run time can be set to any value or inf (for infinite). This means the simulation stops when all the waypoints have been covered.

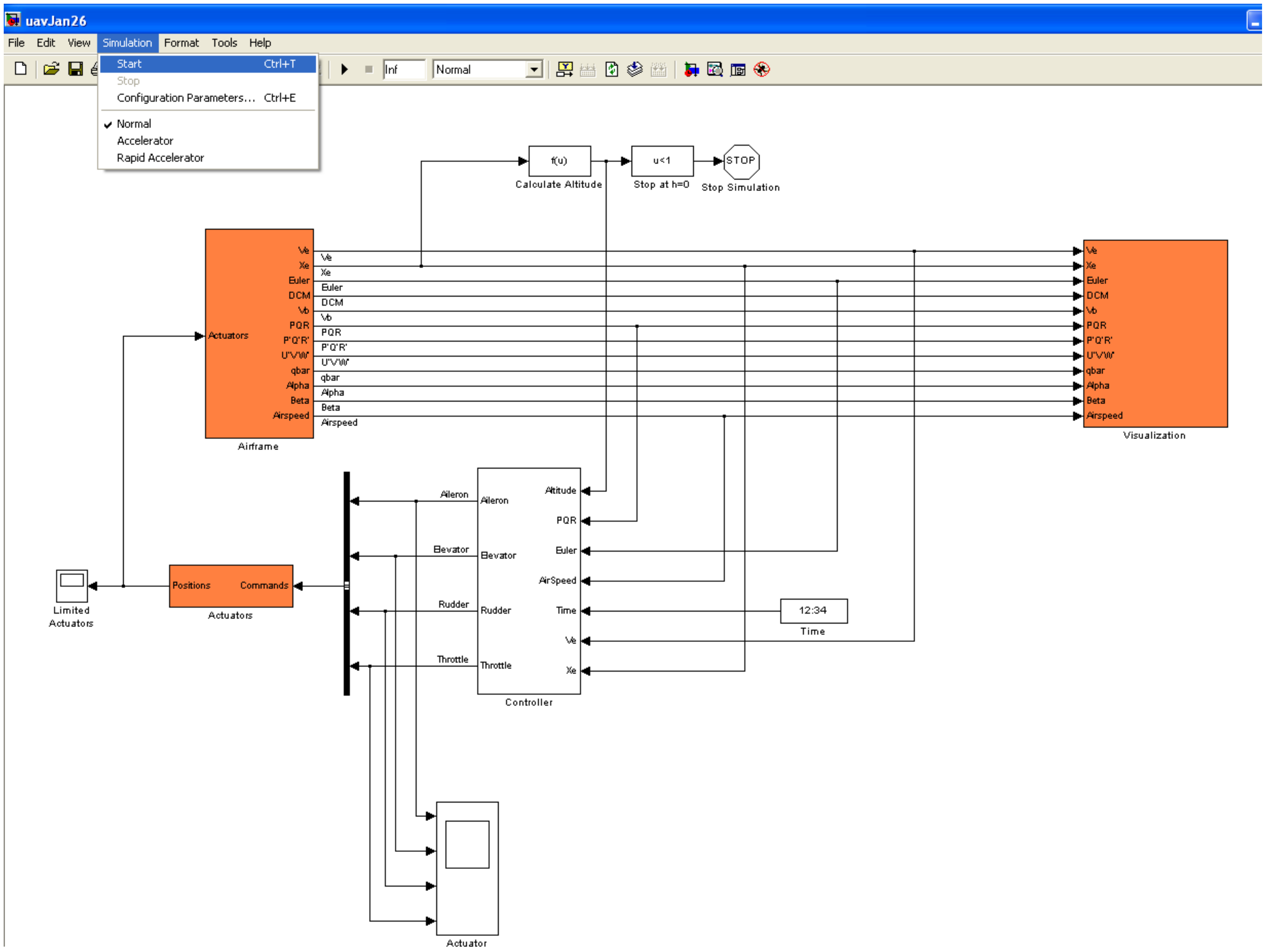


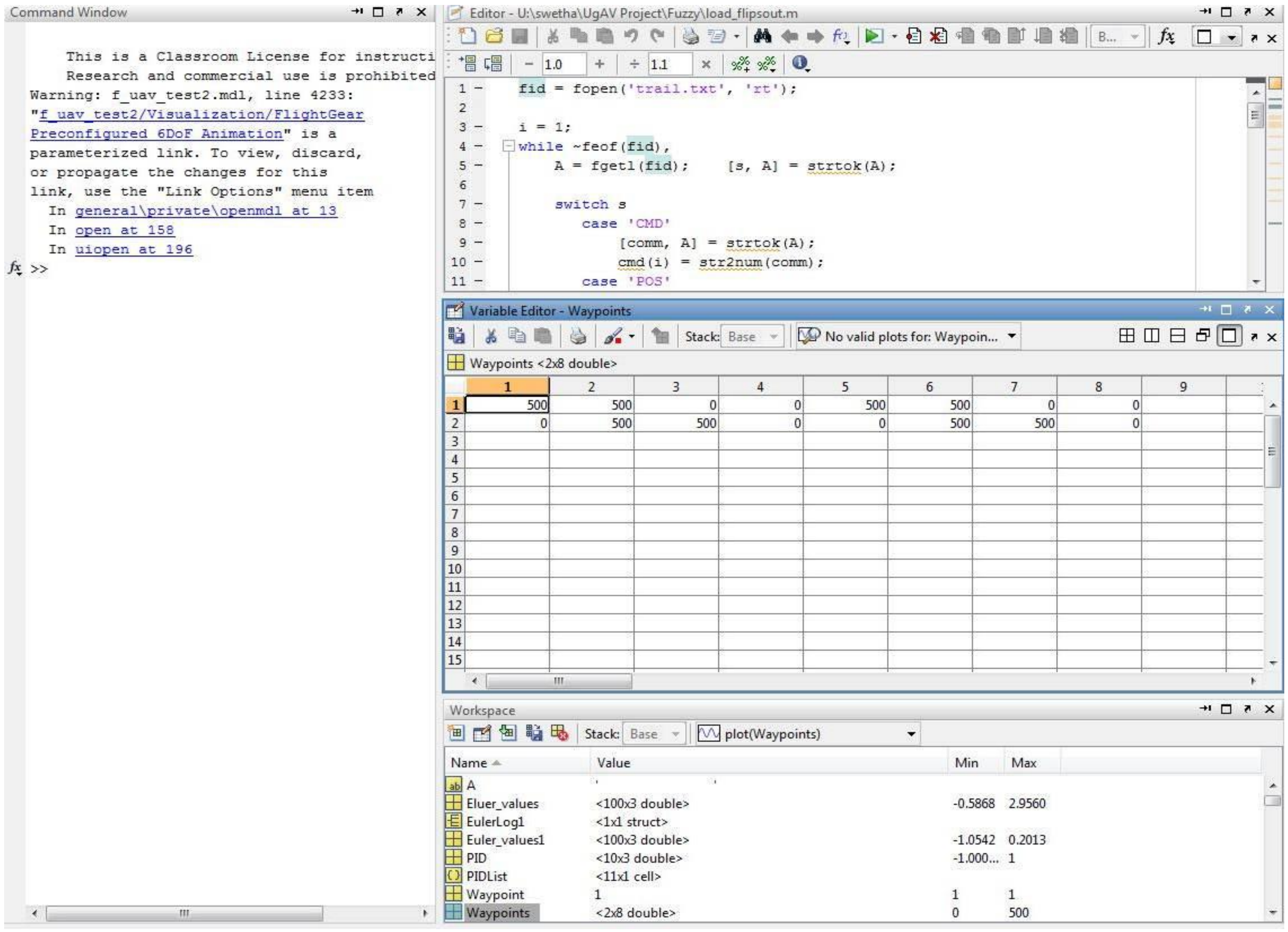
Figure 27: Start Simulation, change simulation time

Changing Parameters:

The flight plan followed in this model is a waypoint following plan. The waypoints and altitude are derived from the FLIPS output file. The waypoints are stored as a 2-by-n workspace variable and the desired altitude as a 1-by-n workspace variable (in the .mat file). These can be accessed from within the variable editor in Matlab as shown in Figure 28.

The latitude values for every point are in the first row of the variable and the longitude values in the second row. At present, the waypoint values are relative to the start position or origin (0, 0). The altitude values have been calculated for every waypoint. If there is no change in altitude, then the altitude value represented for the previous waypoint will be carried over.

The waypoints and other parameters can be varied by varying the FLIPS output file or by entering them into the workspace variable directly.


Command Window
Editor - U:\swetha\UgAV Project\Fuzzy\load_flipsout.m
Variable Editor - Waypoints
Stack: Base
No valid plots for: Waypoin...
Workspace
plot(Waypoints)

```
This is a Classroom License for instructi
Research and commercial use is prohibited
Warning: f_uav_test2.mdl, line 4233:
"f_uav_test2/Visualization/FlightGear
Preconfigured 6DoF Animation" is a
parameterized link. To view, discard,
or propagate the changes for this
link, use the "Link Options" menu item
  In general\private\openmdl at 13
  In open at 158
  In uiopen at 196
>>
```

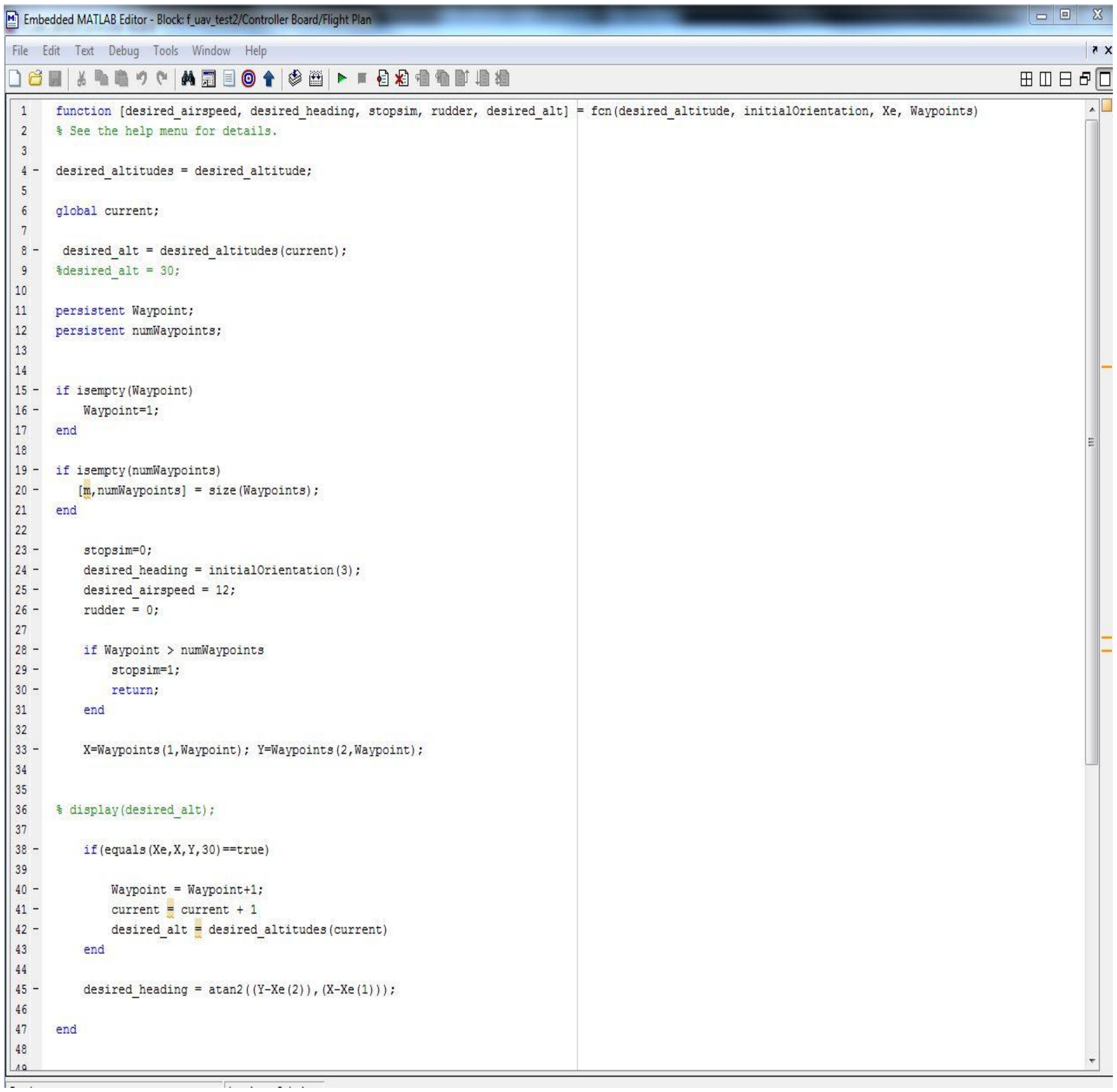
```
fid = fopen('trail.txt', 'rt');

i = 1;
while ~feof(fid),
    A = fgetl(fid);     [s, A] = strtok(A);

    switch s
       case 'CMD'
          [comm, A] = strtok(A);
          cmd(i) = str2num(comm);
       case 'POS'
```

Waypoints <2x8 double>

	1	2	3	4	5	6	7	8	9
1	500	500	0	0	500	500	0	0	
2	0	500	500	0	0	500	500	0	

Name	Value	Min	Max
A	' '		
Eluer_values	<100x3 double>	-0.5868	2.9560
EulerLog1	<1x1 struct>		
Euler_values1	<100x3 double>	-1.0542	0.2013
PID	<10x3 double>	-1.000...	1
PIDList	<11x1 cell>		
Waypoint	1	1	1
Waypoints	<2x8 double>	0	500

Figure 28: Variable Editor in MATLAB

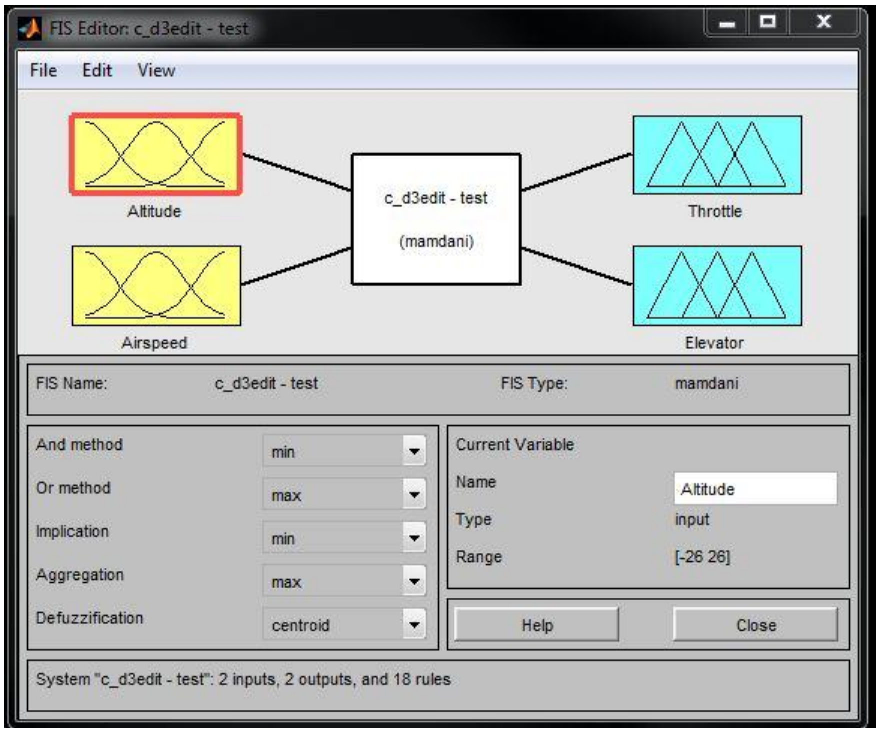

Embedded MATLAB Editor - Block: f_uav_test2/Controller Board/Flight Plan
File Edit Text Debug Tools Window Help

```
function [desired_airspeed, desired_heading, stopsim, rudder, desired_alt] = fcn(desired_altitude, initialOrientation, Xe, Waypoints)
% See the help menu for details.

desired_altitudes = desired_altitude;

global current;

 desired_alt = desired_altitudes(current);
%desired_alt = 30;

persistent Waypoint;
persistent numWaypoints;


if isempty(Waypoint)
    Waypoint=1;
end

if isempty(numWaypoints)
   [m,numWaypoints] = size(Waypoints);
end

    stopsim=0;
    desired_heading = initialOrientation(3);
    desired_airspeed = 12;
    rudder = 0;

    if Waypoint > numWaypoints
        stopsim=1;
        return;
    end

    X=Waypoints(1,Waypoint); Y=Waypoints(2,Waypoint);


% display(desired_alt);

    if(equals(Xe,X,Y,30)==true)

        Waypoint = Waypoint+1;
        current = current + 1
        desired_alt = desired_altitudes(current)
    end

    desired_heading = atan2((Y-Xe(2)),(X-Xe(1)));

end
```

Figure 29: Flight Plan

Figure 29 shows the Flight Plan present inside the Controller Board block. The desired airspeed, desired heading and desired altitude can be controlled through this Embedded MATLAB function block. A new MATLAB function can also be defined if one wants to create their own plan. In addition to changing the flight plan, other aspects of the control system may be changed. This includes the fuzzy input values, membership functions, membership function ranges, rules etc. from the FIS Editor Graphical User Interface. To go to the FIS editor,

- Type “**fuzzy**” in the MATLAB Command Window.
- The FIS Editor opens an Untitled file (a new one).
- Go to **File** -> **Import** -> **From File** and select the corresponding FIS file.



FIS Name: c_d3edit - test
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: Altitude
Type: input
Range: [-26 26]

System "c_d3edit - test": 2 inputs, 2 outputs, and 18 rules

Figure 30: FIS Editor

<u>Note:</u> Parameters cannot be changed during a simulation run.

Once the simulation has started, the graphical representation of the flight parameters can be viewed, as well as the simulated motion of the aircraft in Flight Gear.