ABSTRACT

Title of thesis: DEVELOPMENT OF A TEST-BED FOR CLOSED-LOOP MAV FLIGHT CONTROL

Nitin Kumar Gupta, Master of Science, 2006

Thesis directed by: Professor Inderjit Chopra
Department of Aerospace Engineering

A novel ground based test-bed for closed loop flight control of MAVs is developed. The emphasis is on easy re-programability, measurement of internal states of the vehicle, fail-safe features, and modular design so that control algorithms can be efficiently evaluated on a real MAV in flight. The sensors transmit data to the ground station using a wireless link. All processing is carried out on the ground station, and the final commands are up-linked back to the MAV. The control algorithm on the ground station is implemented in LabVIEW. A microcontroller is used on the ground station to take care of the time-critical tasks that cannot be handled by a PC. Starting with the early experiments on yaw-control, this report presents the results for the 3-DOF attitude control experiments done on an MAV using this test-bed. Three independent PID control loops are used to implement automation. Ziegler-Nichols method is used for tuning the PID controller gains. Simple proportional controllers in pitch and roll are shown to give good performance. The roll and pitch attitudes are maintained within ± 0.1 radians by the controller.

DEVELOPMENT OF A TEST-BED FOR CLOSED-LOOP MAV FLIGHT CONTROL

by

Nitin Kumar Gupta

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:

Professor Inderjit Chopra, Chair/Advisor
Professor Darryll Pines
Professor Sean Humbert

ACKNOWLEDGMENTS

I owe my gratitude to all the people who have made this thesis possible and who have made my graduate experience one that I will cherish forever. First and foremost I would like to thank my advisor, Professor Inderjit Chopra for giving me an opportunity to work at the rotorcraft center at the University of Maryland. I am grateful to him for being patient and understanding with me on several occasions and for his support and encouragement.

I would like to thank Dr. Jayant Sirohi for his suggestions and advice on various occasions during my graduate life. He has always made himself available for help and has taken great pains to go through my work and suggest improvements. I am also grateful to Joe for introducing me to several ideas, especially during the starting phase of my research. Working with him for a couple of months was a great learning experience.

My colleagues at the graduate research office of the rotorcraft center have made my graduate life a most enriching experience. In particular, I would like to thank Peter who has been very closely associated with my research during the last six months and has provided invaluable help in successfully conducting the flight experiments. It would have taken me much much longer to finish, but for his help in fixing the Giant after every crash! He also provided a great help by conducting experiments to validate the complementary filter. I will always remember the lively discussions I have had with Vikram and Carlos over the last year. Their keenness to explore different subjects and re-learn things starting

from the basic principles, has greatly inspired me. I am thankful to Vikram for his help during several experiments and theoretical derivations. John's help as a pilot for the Giant during initial experiments is gratefully acknowledged. His love for flying things turned out to be quite contagious!

I am highly indebted to Abhishek who has always been there (both at work and at home!) for guidance and help as an elder brother to me. I take this opportunity to also thank Arun, Smita, Asitav, Anne, Sandeep, Shreyas, Sudarshan, and all others at the rotorcraft center whose presence has made my graduate experience all the more lively and enjoyable. I would also like to express my gratitude to Amardip, Chinna and Milind for being such wonderful friends and house-mates.

I owe my deepest thanks to my family - my mother and father who have always stood by me and encouraged me to do my best. Words cannot express the gratitude I owe them. I shall always remain indebted to them for the many sacrifices they have made to make me what I am.

Contents

List of Figures

List of Tables

Nomenclature

θ, ϕ, ψ	Pitch, roll and yaw attitudes, respectively (Euler angles)
e_0, e_1, e_2, e_3	The four Euler parameters
g	Acceleration due to gravity
p, q, r	(Euler) body rates along x, y, and z body axes, respectively
$G_a(s)$	Transfer function for the low-pass filter (accelerometer branch)
$G_g(s)$	Transfer function for the high-pass filter (gyroscope branch)
$H_a(s)$	Accelerometer transfer function
$H_g(s)$	Gyroscope transfer function
K_p	Proportional gain
K_i	Integral gain
K_d	Derivative gain
K_{cr}	Critical gain
$K_{P\theta}$, K_θ	Proportional gain for pitch axis
$K_{P\phi}$, K_ϕ	Proportional gain for roll axis
$K_{P\psi}$, K_ψ	Proportional gain for yaw axis
$K_{I\theta}$	Integral gain for pitch axis

$K_{I\phi}$	Integral gain for roll axis
$K_{I\psi}$	Integral gain for yaw axis
$K_{D\theta}$, K_q	Derivative gain for pitch axis
$K_{D\phi}$, K_p	Derivative gain for roll axis
$K_{D\psi}$, K_r	Derivative gain for yaw axis
P_{cr}	Critical period
T_d	Derivative time
T_i	Integral time

Abbreviations

A/D	Analog to Digital
ASCII	American Standard Code for Information Interchange
CCPM	Cyclic Collective Pitch Mixing
DOF	Degree-Of-Freedom
FIR	Finite Impulse response
GPS	Global Positioning System
IIR	Infinite Impulse Response
IMU	nomenInertial Measurement Unit
INS	Inertial Navigation System
LiPo	Lithium-Polymer
MAV	Micro Air Vehicle
MEMS	Micro Electro Mechanical Sensor
P	Proportional control

PD	Proportional-Derivative control
PI	Proportional-Integral control
PID	Proportional-Integral-Derivative control
PCM	Pulse Coded Modulation
PPM	Pulse Position Modulation
PWM	Pulse Width Modulation
R/C	Radio Control
RPM	Rotations Per Minute
SISO	Single Input. Single Output
VI	Virtual Instrument

Chapter 1

Introduction

1.1 Micro Air Vehicles

Micro Air Vehicles (MAVs) are defined by DARPA as six-degree-of-freedom aerial robots measuring less than 15 cm in length, width or height [1]. Such aircraft are expected to be small, inexpensive and expendable. These can be used for surveillance and measurement missions in urban environments and other situations, where larger vehicles are not practical. Their applications range from small unit battlefield surveillance, mapping out chemical/radiation spills or viral outbreaks to more routine applications such as, monitoring crops or wildlife distributions. With the ever-decreasing size and weight of the payload components that can include video cameras, chemical sensors, electronics, and communication devices, such applications are becoming more achievable.

In the last decade, several concepts for MAVs have been developed. These can be broadly classified into fixed-wing based MAVs, like *Black Widow, Hornet* and *Wasp* [2–4] (Fig. 1.1), and rotary-wing based MAVs, like *Giant* [5] (Fig. 1.2) and *Micor* [6] (Fig. 1.3).

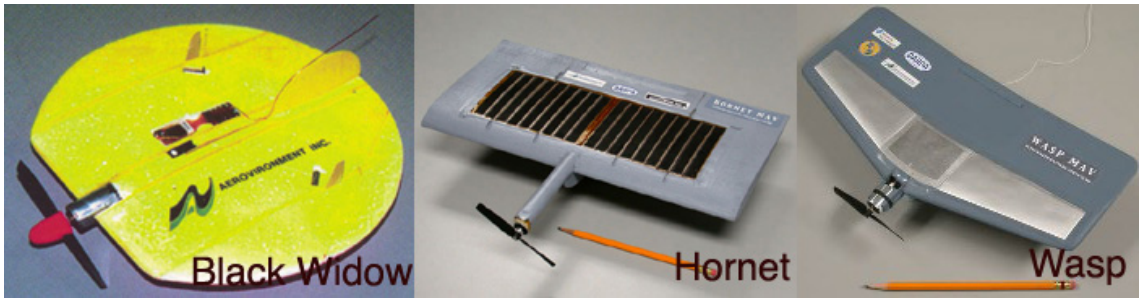

Figure 1.1: Fixed-wing based MAVs developed by Aerovironment

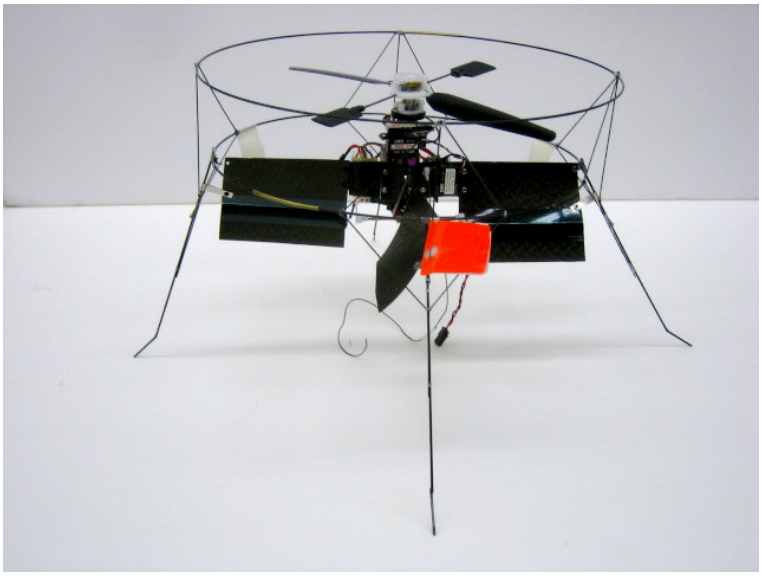

Figure 1.2: *Giant* rotary-wing based MAV developed at the University of Maryland

Rotary-wing based MAVs have capabilities that may allow missions to be performed that cannot be considered for fixed-wing MAVs. In order to generate enough lift to sustain weight, the fixed-wing MAVs have to fly fairly fast. Rotary-wing based MAVs, on the other hand, can hover, and can take off and land vertically. This allows them to operate in more confined environments, such as, small indoor spaces and dense urban settings.

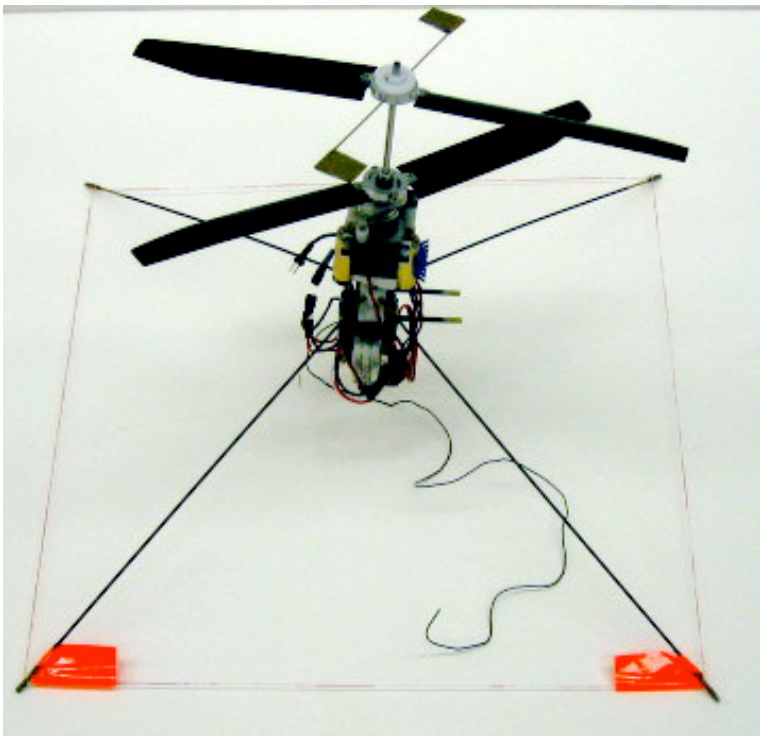

Figure 1.3: MIcro COaxial Rotor (*MICOR*) rotary-wing based MAV (UMD)

1.2 MAV Research Challenges

The small size requirements of MAVs generate a variety of challenges in their development, not seen in their full-scale counterparts. For the rotary-wing based MAVs, these challenges fall into two main categories: (1) improving aerodynamic efficiency or figure of merit, and (2) achieving stability and control [7]. Solutions for improving figure of merit of these vehicles are currently been developed in the form of design of rotor blades suitable for low Reynolds number flight, ducted rotors, and other novel concepts [8–11]. As a human pilot is not present onboard the MAVs, either a remote pilot or an autonomous controller is required in order to fly these vehicle. Even if a remote pilot controls the MAV, a basic stability-augmentation system is desired to reduce pilot workload. Design of a

test-bed to implement and evaluate closed-loop control during flight, and development of an attitude stabilization system using this test-bed are discussed in this thesis.

1.3 Motivation and Challenge in Automation

Traditionally, the UAVs and MAVs were tele-operated by remote human-pilots. These UAVs were sometimes equipped with onboard cameras and other sensors to assist the human pilot by providing visual cues on his monitor [12]. Recently there has been much research effort to automate MAVs, which would not only eliminate the remote human pilot but also allow higher degree of maneuverability to accomplish missions which would be extremely difficult, if not impossible, to fly for a remote human pilot. An important requirement of such automation systems is to augment stability. This is expected to enable untrained pilot to operate an MAV by issuing high-level commands.

The smaller size and lower inertia, relative to the available thrust, of rotary-wing based MAVs makes them more maneuverable and agile than the full-scale helicopters. This also makes the control and stabilization of MAVs a more challenging task than that of full-scale helicopters. Additional requirements like control and navigation in constrained environments, and low weight and size of the avionics package makes this problem even more involved [13]. Previous work on autonomous flight control of model helicopters has investigated outdoor flight of much larger models (5 feet rotor diameter and above) with higher payload carrying capability [14–18]. This research has resulted in successful autonomous flight of these large helicopter models. Typical payload carrying capability of MAV is around 20 grams. The design of an avionics package that is both light and

provides reliable state estimation is still a challenge.

1.4 The *Giant* MAV

We propose to use the Giant MAV (Fig. 1.4), developed at the Micro Air Vehicles Laboratory of University of Maryland, as the test-bed for our MAV control experiments because it is representative of the dynamics of a typical hover-capable rotary-wing MAV, and still has a sufficient payload carrying capacity to lift the avionics package designed using commercial off-the-shelf components. This enables us to focus on issues related to implementation and algorithms related to flight stability and control.

The Giant consists of a single fixed pitch rotor, with a set of vanes in the downwash of the rotor to provide the required anti-torque. The main rotor has a diameter of 25 cm. It is a two bladed rotor with a teetering hinge and a Bell/Hiller servo-paddle. Giant is powered by a brushless DC motor, and a 700mAh Lithium-polymer battery pack gives it an endurance of about 10 minutes. Yaw control is achieved by aerodynamic control surfaces in the vanes and lateral control is achieved by a swashplate connected to the servo-paddle. The Giant is equipped with an off-the-shelf gyro that provides automatic vane inputs in order to stabilize the vehicle in yaw. The entire system weighs 250 grams and has a payload carrying capacity of about 30 grams.

The actuator configuration of the Giant is very similar to a conventional helicopter. It has four controls: the lateral and longitudinal cyclic inputs control the swashplate or the roll and pitch moments produced by the rotor; the thrust (rotor RPM) is controlled by the throttle input; the yaw input controls the angle of the vanes, thereby providing the anti-

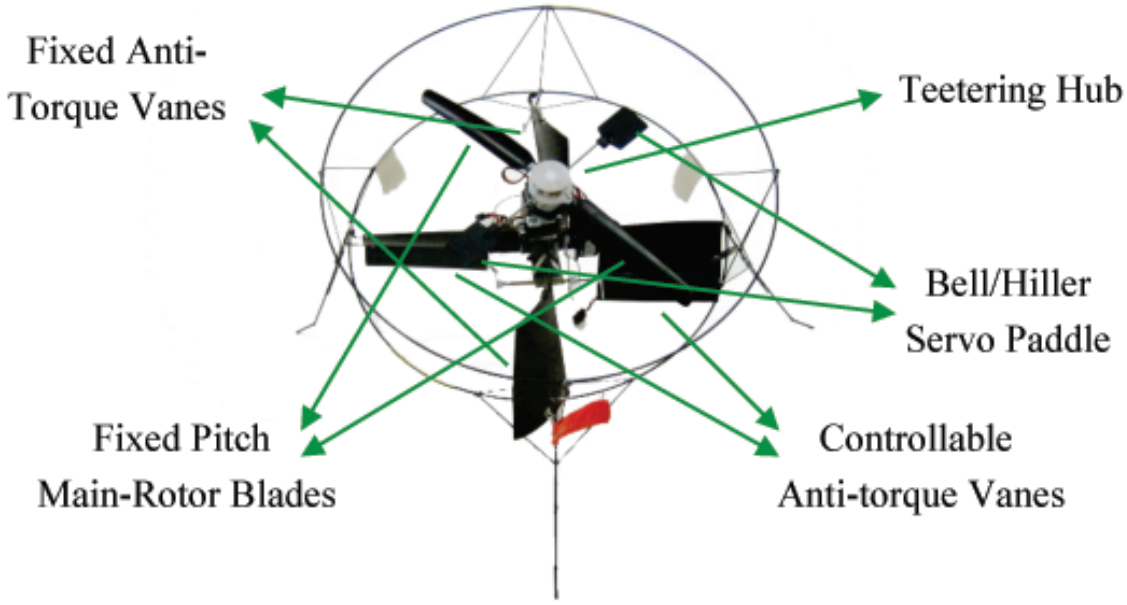

Figure 1.4: Configuration of the *Giant*

torque to counter the rotor torque. Thus the controls have a direct effect on the Giant's roll and pitch attitude rate, vertical velocity, and the heading rate, respectively.

The pilot does not control Giant's position and velocity directly. These are controlled through a chain of effects as follows. The cyclic control inputs result in control moments about the rotor hub via a tilting motion of the rotor tip-path-plane. The rotor control moments produce a fuselage rolling or pitching motion. The tilting of the tip-path-plane produces horizontal thrust components. For example, holding a constant pitch angle, the Giant will accelerate until the aerodynamic drag force balances the horizontal thrust component. In the steady state, a pitch or roll angle translates into a steady longitudinal or lateral velocity. Figure 1.5 shows a block diagram of the subsystems involved in the longitudinal-lateral control problem.

Of course, different cross couplings also need to be compensated, such as the effect

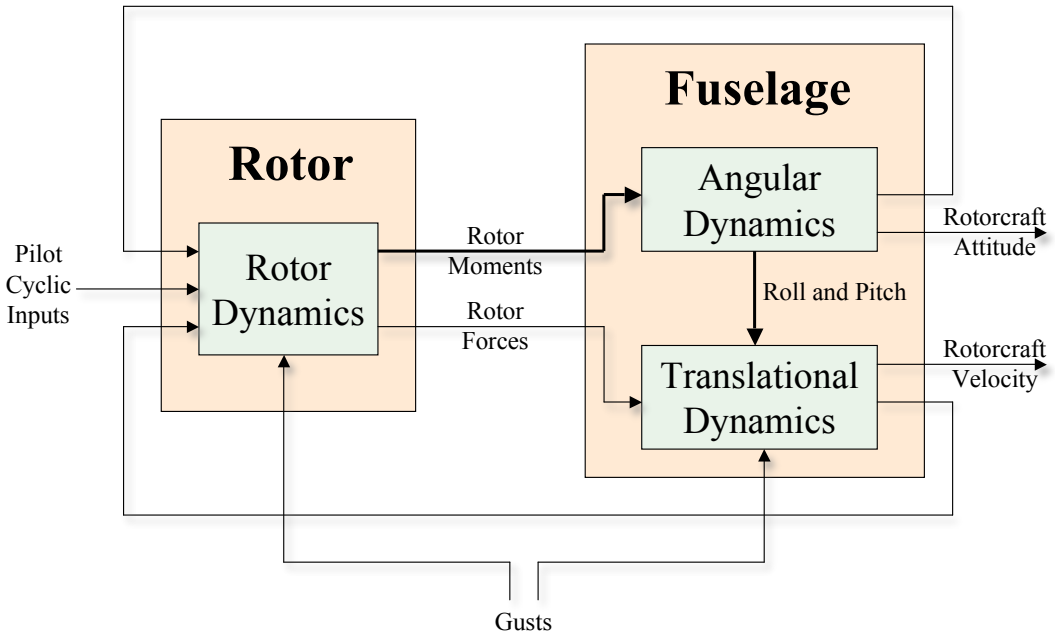

Figure 1.5: Subsystems involved in rotorcraft control

of the translation or rotational fuselage motion on the rotor and fuselage attitude dynamics, or the secondary effects of the controls. For example, when the Giant is pitched, the vertical thrust component will decrease, requiring an increase in the thrust magnitude to maintain altitude. This increase in thrust, however, will produce a reaction torque at the rotor shaft and thus vanes will need adjustment to provide more anti-torque. The Giant because of its relatively small size, is also quite sensitive to atmospheric disturbances that affect the rotor and fuselage dynamics.

1.5 Proposed Stability and Control Scheme

We propose to implement feedback control in order to stabilize and control an MAV. Basic architecture of such a control scheme is shown in figure 1.6.

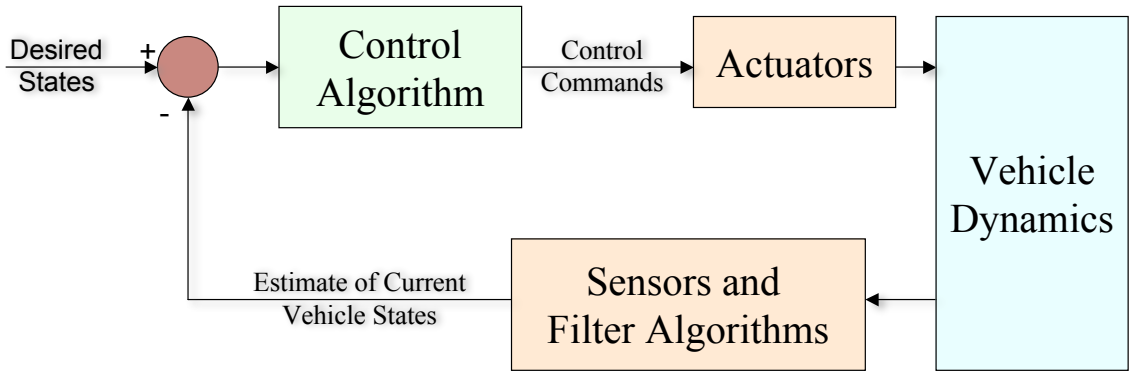

Figure 1.6: Generic feedback control

The sensors (attached to the vehicle) and the associated filter algorithms give us an estimate of the current states of the vehicle. These states are dictated by the control algorithm and the control objective. For 6 DOF control (3 rotational and 3 translational degrees-of-freedom) of the vehicle, the states would typically be the attitude and position of the vehicle. The control algorithm then uses this information to compute the commands for the actuators in order to stabilize and control the vehicle as desired. These actuators act on the vehicle and influence its motion. The sensors, in turn, again estimate the new values of the states, and the entire cycle is repeated.

The first step in implementing this scheme is the selection of a control algorithm. This decision is based on the kind of vehicle, its dynamics, availability of its flight dynamics model, the actuator configuration, and of course the control objective. The states required by the control algorithm will dictate the sensors and the filter algorithms to be used. The sensor selection is also constrained by payload carrying capability of the vehicle, onboard power available and other operational constraints (e.g., GPS cannot be used if the vehicle has to operate indoors). If no suitable sensors and filter algorithms could be

found to provide the states required by the chosen control algorithm, we need to select a different control algorithm for which suitable sensors and filter algorithms exist.

The control algorithms can be broadly classified into modern model-based control techniques and the classical techniques. Given a highly accurate vehicle dynamics model of a vehicle, the model-based techniques, like the H-infinity control, would have significantly better performance than a classical controller, like the PID (Proportional-Integral-Derivative) control [19]. While a PID based controller is expected to achieve basic automation, it will not be optimal, and will not be able to fully utilize the available degree of maneuverability of the vehicle. As there does not exist a sufficiently accurate model for the Giant and developing such a model for any rotorcraft-based vehicle is a challenging task in itself, we opt to first implement a classical PID based control and demonstrate the working of the generic test-bed developed to enable this implementation. Such a test-bed can later be used for implementing a model-based controller when required.

1.6 Overview of the Thesis

This thesis presents the development of an innovative test-bed for closed loop MAV flight control experiments. The basic concepts from control theory, state estimation and an overview of hardware components used for this setup is given in Chapter 2. Chapter 3 descibes the preliminary experiments performed to develop understanding of these basic concepts. The yaw-control experiment and development of a complete onboard stabilization system are discussed in detail. The software and hardware aspects of the final test-

bed developed to overcome the shortcomings of the onboard stabilization system form the subject of Chapter 4. The results of the 3-DOF hover control experiment are discussed in Chapter 5. Chapter 6 presents the conclusions and recommendations for future work. A quick guide to setting up the test-bed for future experiments and solutions to common problems encountered are documented as an appendix to this thesis.

Chapter 2

The Hardware Components

2.1 Sensors and their Operation

Inertial measurement unit (IMU) provides one of the most established methods for estimating the motion (change in attitude and position) of a vehicle [20, 21]. This is a self-contained system and does not need information from external source, like GPS, to determine position.

An accelerometer measures acceleration along a single axis. Its output can be integrated to get velocity. The second integration gives the position, or change in position. If the direction of travel is known, the current position with respect to the original position can thus be obtained. The inertial navigation system (INS) is a form of 'dead reckoning', that is, it gives the present position with respect to the initial position, and it cannot directly find the absolute position in the inertial frame.

Three accelerometers can be put together with orthogonal sensing directions to give a measure of displacement in the three-dimensional space. In order to maintain the ori-

entation of these accelerometers when the vehicle maneuvers, the accelerometers are suspended in a set of three gimbals which are gyro-stabilized to maintain the direction. Such an arrangement forms the 'gimballed' configuration of the inertial navigation system.

The gyros, used in the gimballed inertial navigation system are of a type known as 'integrating' gyros, that is, they give output proportional to the angle through which they have been rotated (about their sensing axis). They serve as the sensing elements in the null-seeking servos. The output of each gyro drives the servo motor connected to the appropriate gimbal, thus keeping the gimbal in a constant orientation in inertial space.

The gimbals are a set of three rings, each with a pair of bearings, initially at right angles (fig. 2.1). Each gimbal also consists of a motor, built around one of the bearings, and at the other end a synchro (an electromagnetic angle-measuring device) [22]. The innermost gimbal always maintains its orientation in inertial space, irrespective of how the vehicle maneuvers. The synchro on the innermost gimbal thus measures the azimuth (heading), the synchro on the middle gimbal measures the pitch, and the one on the outer gimbal measures roll.

The innermost gimbal provides a 'stable platform' which holds the gyros and the accelerometers. The entire system is called a 'gimballed platform'. Figure 2.2 shows the Marconi FIN1000 inertial platform, which is used in virtually all the RAF's combat aircraft and many others world-wide. Such systems can measure vehicle's position, velocity, acceleration, attitude, and heading. Over the last 30 years, the gimballed inertial navigation technology has fully matured and very reliable and accurate IMUs are available.

The gimballed arrangement is mechanically very complex. It contains delicate sliprings, the motors dissipate power; mechanical resonances are unavoidable. Due to

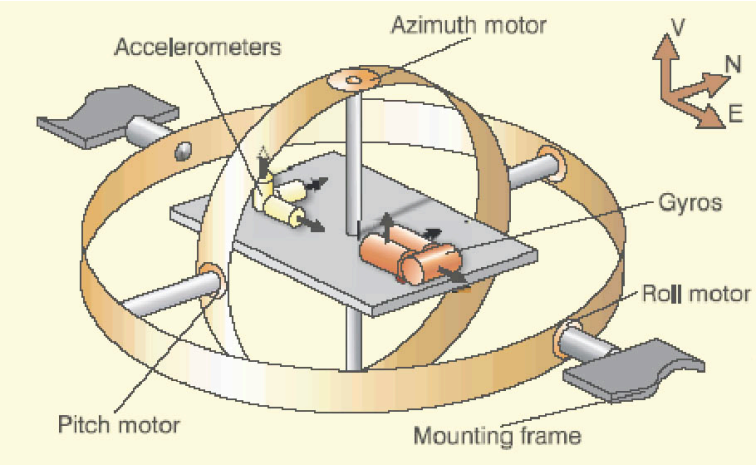

Figure 2.1: Internals of a typical gimballed inertial navigation system

high part count, these are too heavy and bulky to be carried by MAVs.

Research in the direction of simplifying the complexity of these systems lead to the development of 'strapdown' systems. These do away with the gimbal altogether and the gyros and accelerometers are mounted on the platform directly. The gyros are used to measure the rotations in space instead of null-seeking sensors. Using the gyro information, the system knows which direction the accelerometer axis is pointing in at any instant. Thus, the mechanical gimbals are replaced by a 'mathematical gimbal set' (fig. 2.3, 2.4).

Realization of such a system became possible due to recent advances in light weight, powerful and cheap digital computers (or microcontrollers), and MEMS (Micro Electro-Mechanical Sensor) technology. A software solution is used to keep track of the orientation of the IMU (and vehicle) and transform the measurements from the body frame to the navigational frame. This method overcomes the problems encountered with the gimbaled system, and most importantly reduces the size, cost, power consumption, and complexity of the system. The strapdown INS measures the orientation of a vehicle by

Figure 2.2: A typical gimballed inertial navigation system

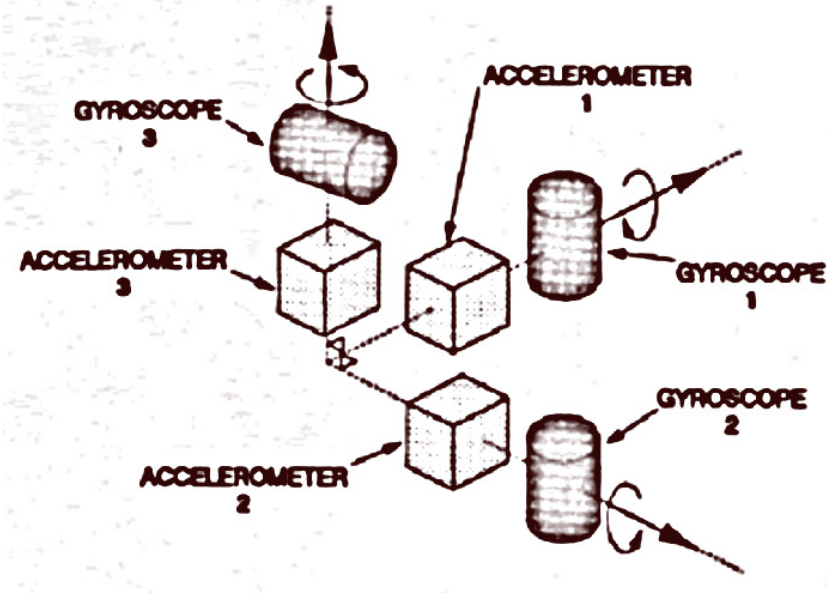

Figure 2.3: Internals of a typical strapdown inertial navigation system

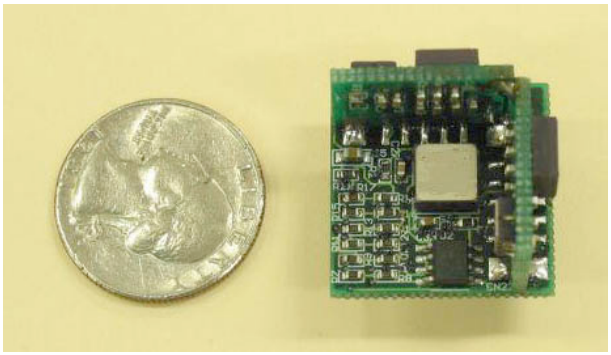

Figure 2.4: A typical strapdown inertial navigation system

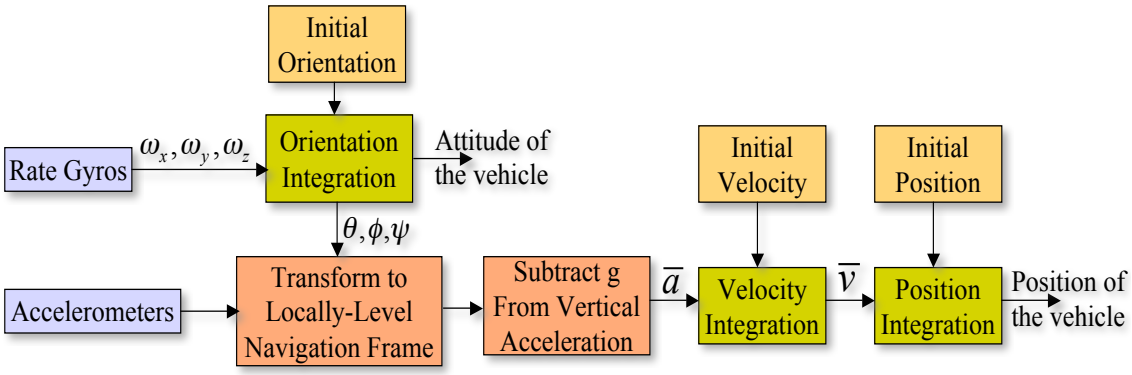

Figure 2.5: Schematic of strapdown inertial navigation system

integrating the angular rates from three orthogonal angular rate sensing gyroscopes (rate gyros) strapped down to the frame of the vehicle. To get position, 3 accelerometers, also affixed to the orthogonal axes of the moving body, measure the total acceleration vector of the body relative to the inertial space. This acceleration vector can be converted from the body coordinates to inertial coordinates using the known instantaneous orientation of the body determined by the gyros. Position is then obtained by subtracting the effect of gravity from the measured acceleration and then performing double integration starting from a known initial position (fig. 2.5).

Figure 2.6: The standard servo used on R/C aircraft models

2.2 Actuators and their Interface

The actuators used on the Giant are the standard servos used on most commercially available radio-controlled (R/C) aircraft models (fig. 2.6). A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. These servos although quite small, can generate very high torque. The servos used on the Giant are rated at 10 oz-inch.

2.2.1 Working of a Servo

A servo works on the principle of negative feedback. Figure 2.7 shows the internal parts of a servo. Here, the current to the motor is provided by a potentiometer that is itself driven by the same motor. This potentiometer also serves as the output shaft. The motor when turned on, drives the potentiometer till the potentiometer provides no more current to the

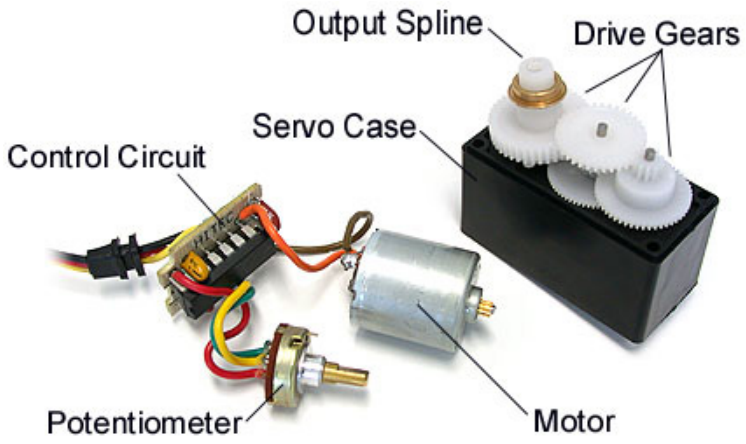

Figure 2.7: The internals of a standard servo

motor. A circuitry allows for changing this zero-current position of the potentiometer, based on the input signal. So, when a new position of the potentiometer (or output shaft) is desired, this zero position is altered by the circuitry (based on input signal) and the motor turns the potentiometer till this zero position is reached.

The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation (PCM). The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse determines how far the motor turns. A 1.5 ms pulse, for example, will make the motor turn to the 90-degree position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degrees. If the pulse is longer than 1.5 ms, the shaft turns closer to 180 degrees (fig. 2.8). The position of a servo can therefore be theoretically updated not more than 50 times per second (20 ms for each update). This imposes a fundamental limit on the servo bandwidth besides the mechanical performance issues.

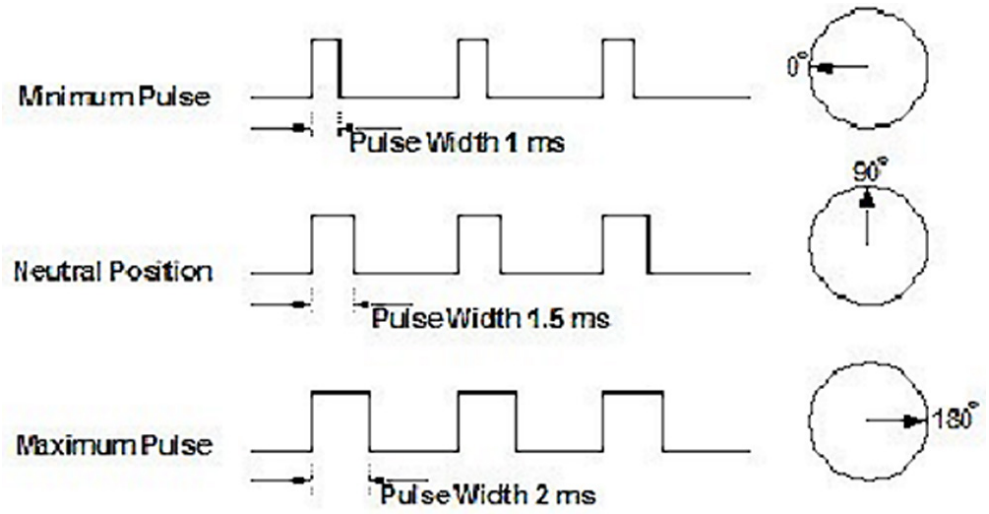

Figure 2.8: The PCM input signal to a servo

2.2.2 The Standard R/C Transmitter and Receiver

Giant can be flown manually by a human pilot using the standard radio-control transmitter and receiver. The radio-control transmitter (fig. 2.9) converts movements of the control sticks and switches into a coded radio signal, which is transmitted by radio to the radio-control receiver (fig. 2.10) installed on Giant. The signal is received and then decoded by the micro-controller on the receiver into the servo-control signals.

Giant needs a transmitter and a receiver with at least 4 channels, one each for controlling longitudinal cyclic, lateral cyclic, rotor RPM and yaw-control vanes. While longitudinal cyclic, lateral cyclic and yaw-control vanes are directly driven by the servos (which are connected to different channels on the receiver), the rotor RPM is controlled through a motor-driver circuit (fig. 2.11), which converts the servo-control signals into high frequency PWM signals.

Radio-control transmitters are usually provided with a data port. The signals being

Figure 2.9: A typical R/C transmitter

Figure 2.10: A typical 4-channel micro R/C receiver

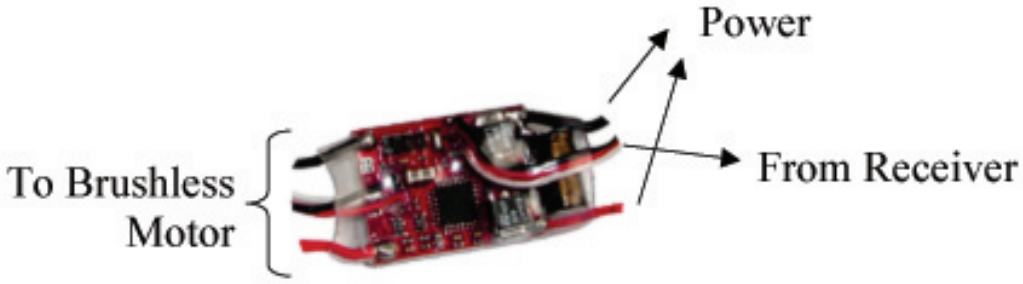

Figure 2.11: The motor driver circuit

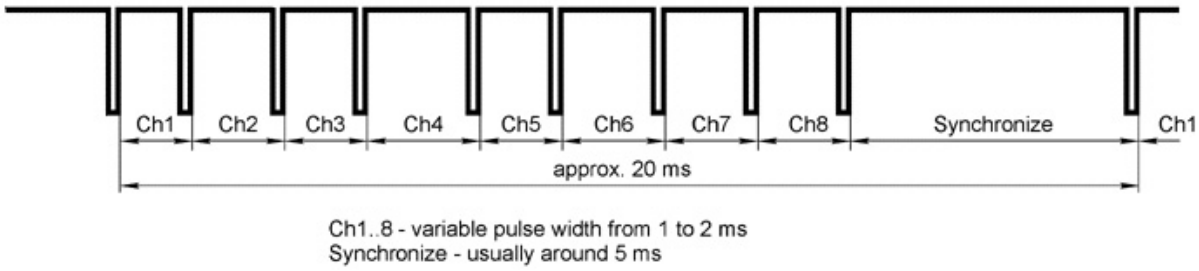

Figure 2.12: The PPM signal at the data port of an 8-channel transmitter

sent by the transmitter, corresponding to the stick positions can be read on the output pin of this port. This port also has an input pin, and any desired signal can be written to this pin for transmission to the receiver. The input and output signals on this port follow a modulation very similar to the servo-control signal. The normal state of the pin is high. Each frame has a width of 20 ms and has as many inverted-pulses as the number of channels. The width of the first pulse (inverted pulse) corresponds to the position of first servo, and so on as shown in figure 2.12. This modulation is called the inverted Pulse Position Modulation (PPM) as different channels are represented by different positions of the pulses in each frame.

It is important to note that although usually each stick corresponds to one servo channel (as in case of Giant), there maybe configurations where each stick movement

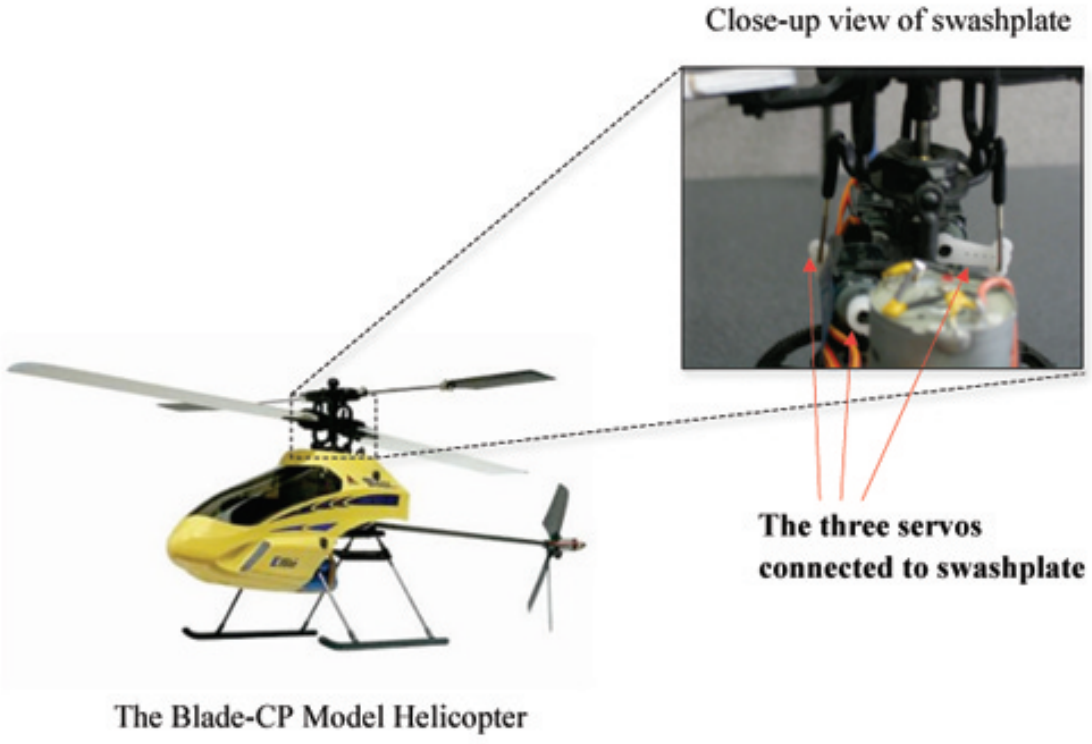

Figure 2.13: The swashplate connections on a Blade-CP helicopter

may lead to changes in more than one servo channel. For example, in a typical CCPM (Cyclic/Collective Pitch Mixing) type of RC helicopter (like, Blade-CP [23]), the swashplate is controlled by three servos (fig. 2.13). The motion of all three servos is affected even if only one of the two cyclic or collective controls is applied. Therefore, although pilot moves only one stick at a time, the width of all three pulses corresponding to swashplate servos changes.

The hardware components described in this chapter are all commercial off-the-shelf components. These enable the Giant to be flown manually by a human pilot. The next task is to use the knowledge of the protocols and internal working of these components to interface these (along with sensors) with a processing element (microcontroller or a PC)

in order to automate the flight of the Giant.

Chapter 3

Preliminary One DOF Experiments

3.1 Yaw Control Experiment

As the first step, in order to explore the requirements of a stability augmentation system, a yaw stabilization experiment was conducted on the Giant. For this purpose, a test stand that allowed only one degree-of-freedom (yaw) was prepared, and the Giant was mounted on it, as shown in figure 3.1.

A single degree of freedom PD (proportional-derivative) control scheme as shown in figure 3.2, in conjunction with a commercial off-the-shelf IMU, was implemented for stabilization. The yaw angle is measured and the yaw-vanes on the Giant are controlled in a way so as to maintain the same yaw attitude.

3.1.1 The Hardware

After an exhaustive search, the MAG^3 IMU from Memsense [24] was selected for the sensor. This IMU (fig. 3.3) has 3 gyroscopes each with a range of $\pm$150 deg/sec, 3

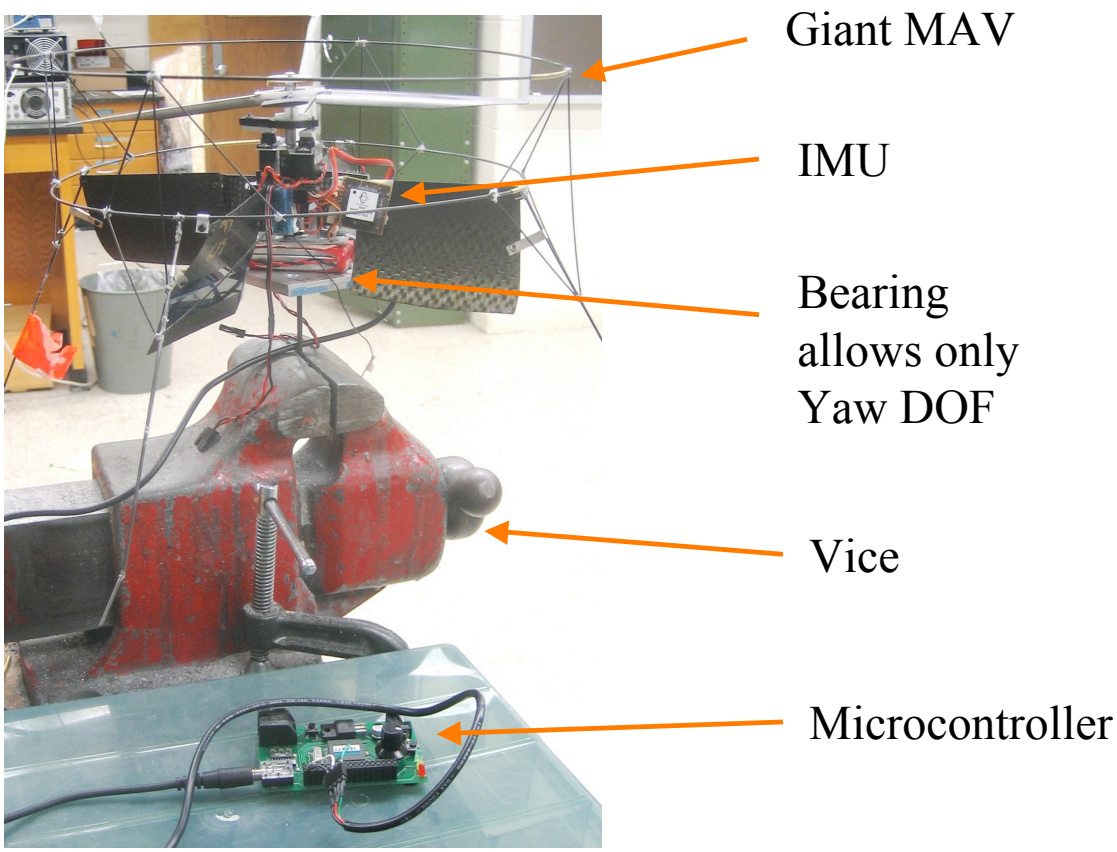

Figure 3.1: The yaw test stand

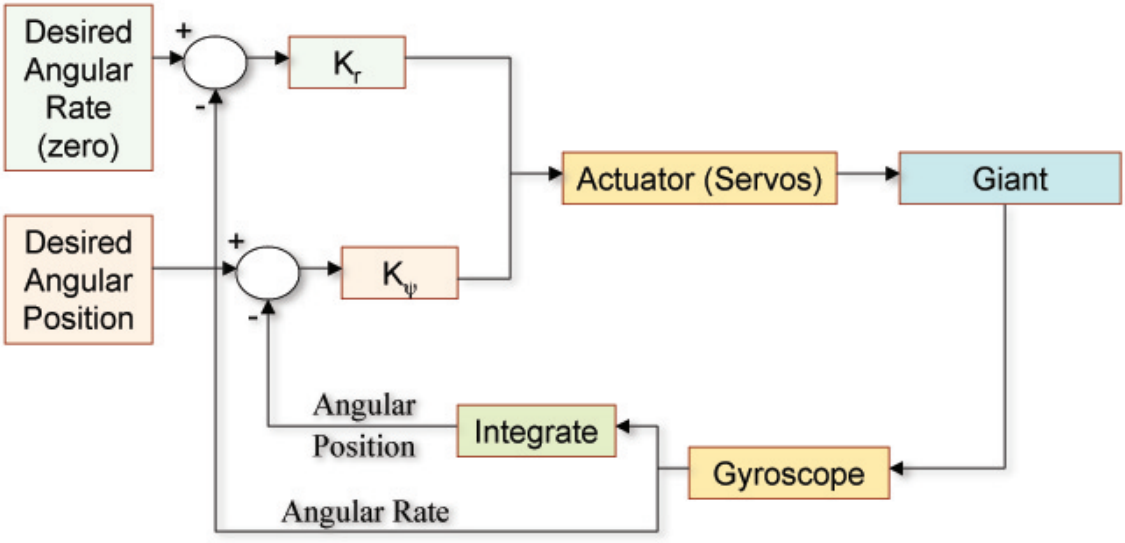

Figure 3.2: The yaw control scheme

Figure 3.3: The MAG^3 IMU from Memsense

accelerometers each with a range of $\pm$2g and 3 magnetometers each with a range of $\pm$1.9 gauss. All sensor outputs are available as analog voltages varying from zero to five volts. This is one of the smallest and lightest IMUs available, weighing only 5 grams and having a size of 0.7 in x 0.7 in x 0.4 in.

A development board (fig. 3.4) based on Microchip PIC 16F877A microcontroller [25] was obtained from Custom Computer Services, Inc. (CCS) [26] to serve as an interface between the sensors and the actuators, and to carry out the control computations. 16F877A microcontroller from PIC is an 8-bit microcontroller running at 20 MHz, and has 256 bytes of EEPROM data memory. It has 2 capture/compare/PWM functions, support for I^{2}C bus and 8 channels of 10-bit Analog-to-Digital (A/D) converter. The development board from CCS gives easy access to all these input/output pins, provides an interface to easily program the microcontroller and has a potentiometer connected to one of the A/D channels. Figure 3.5 shows the interface between different hardware components.

3.1.2 Yaw Stabilization

The numerical integration of yaw-rate (to get yaw angle), PD controller and servo command generation are implemented on the microcontroller as shown in figure 3.6. Here the

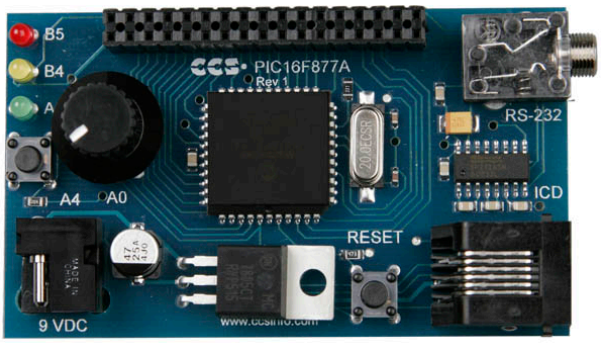

Figure 3.4: The PIC 16F877A development board from CCS

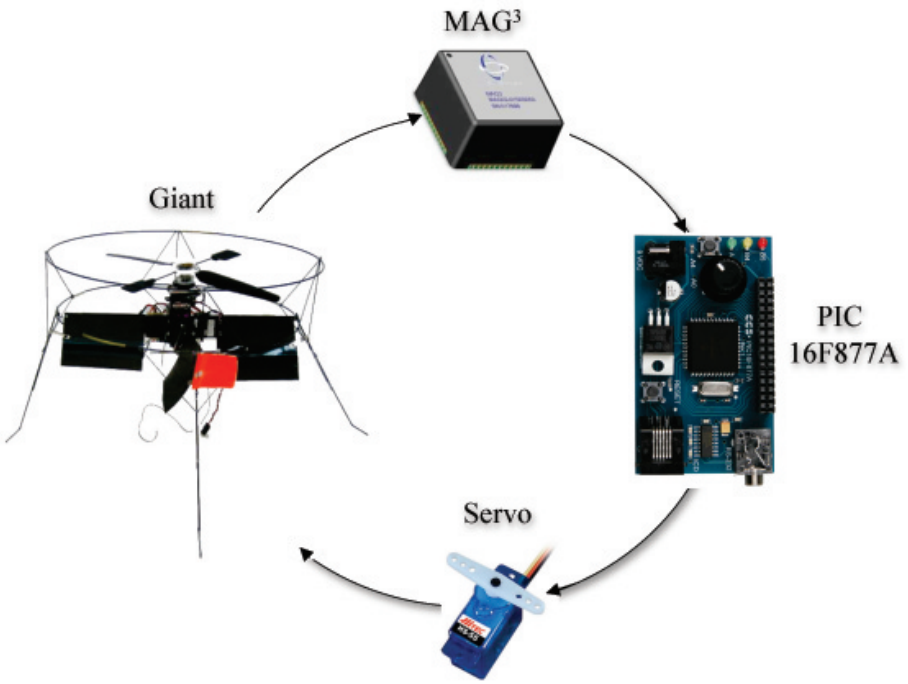

Figure 3.5: The hardware interfacing schematic for yaw control experiment

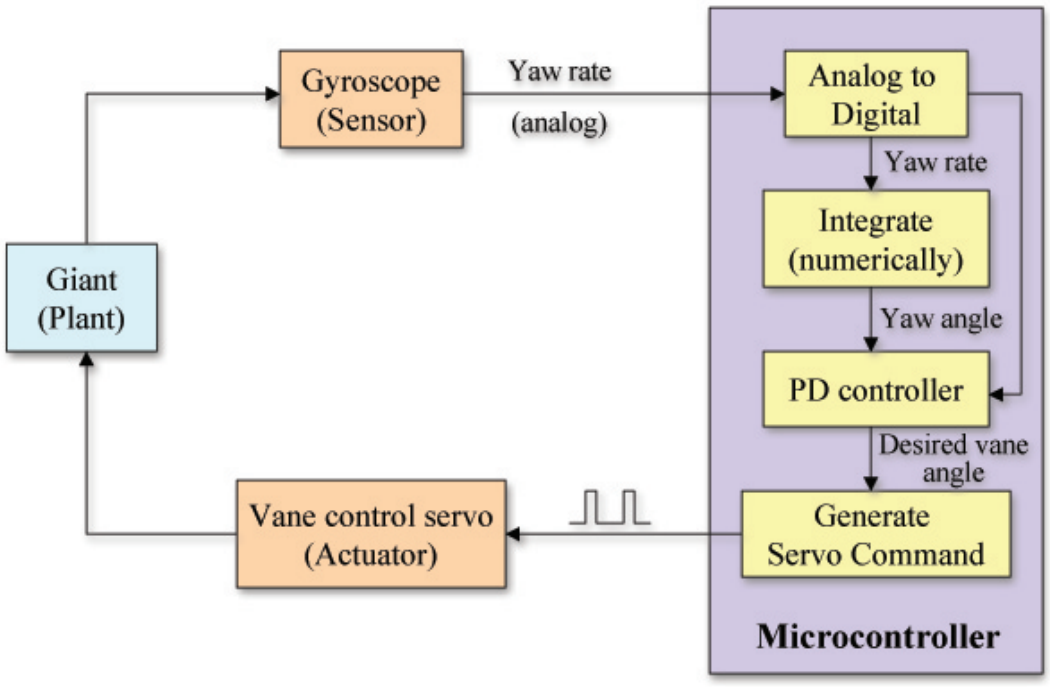

Figure 3.6: The PD control scheme for yaw control

analog output from yaw gyro is continuously sampled. This is then added to a counter, which is initialized to zero in the beginning of the code, to give a quantity proportional to the yaw angle (determination of exact yaw angle is not important). The output angle of the vanes is computed using the value of this angle and the angular rate at that instant. A separate interrupt routine generates appropriate signal for the vane servo, based on this desired angle of vanes, once every 20 ms (as explained in section 2.2.1). It is important to use interrupt routine for this task as it is time critical. The microcontroller suspends all other tasks, once every 20 ms, and generates the servo signal. The flowchart for this is given in figure 3.7 and the code is attached as an appendix to this report.

The proportional and derivative gains (K_ψ and K_r, respectively) used for the computation of output angle of vanes are experimentally determined. Here, the *HighTime* equa-

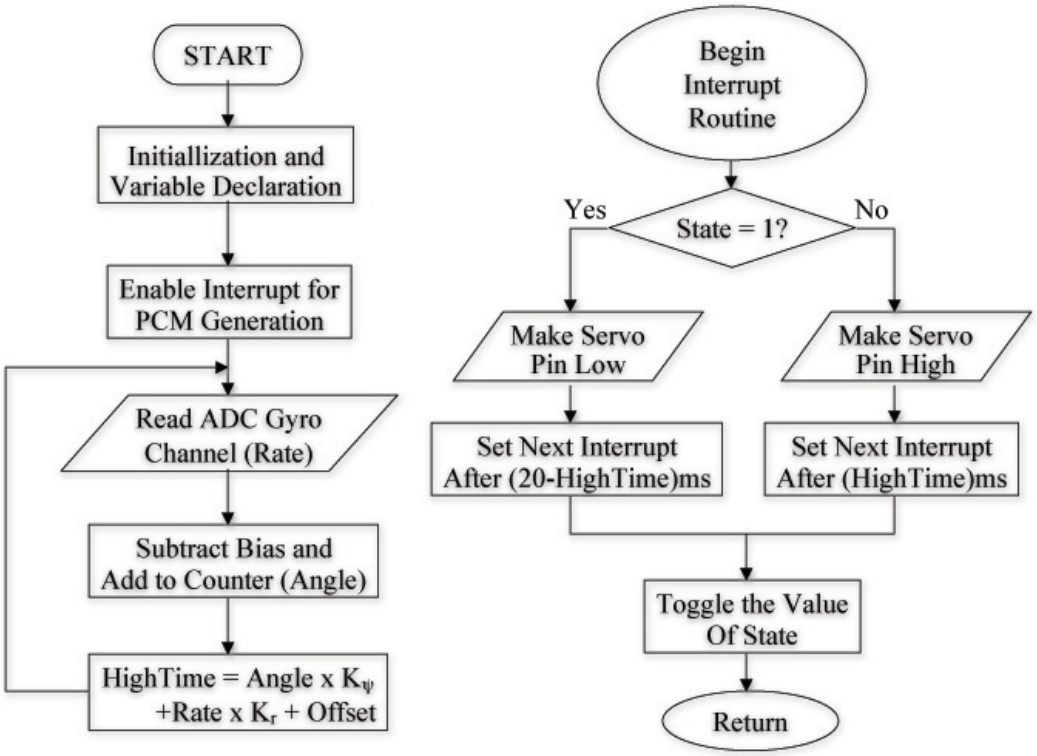

Figure 3.7: The yaw stabilization flowchart

tion represents the PD controller. This controller continuously moves the vanes (changes the value of *HighTime*) so as to hold a constant yaw attitude when the main rotor is spun or its RPM changed.

3.1.3 Yaw Control

The setup is later modified for yaw *control* (from yaw *stabilization*). A potentiometer provided on the CCS development board is used to command the desired yaw angle. The code is upgraded to sample the analog voltage from the potentiometer and use it for computation of the vanes angle. The interrupt routine in this case remains the same as that for the previous case. On turning the potentiometer, the vehicle is observed to follow the command and turn by the desired yaw angle.

In the next step, this setup is further modified and was interfaced with an RC receiver (discussed in section 2.2.2). This way the human-pilot can command the desired yaw angle remotely using the R/C transmitter. This is different from normal radio control, as in the normal case, the pilot would control the angle of the vanes, which would change the torque, and hence the yaw *acceleration*. By using the yaw attitude controller, he can however directly command the desired yaw *angle* using the knob on the R/C transmitter. The controller now continuously adjust the vanes to *achieve* and *maintain* that yaw attitude. The new flowchart is given in figure 3.8. This uses two interrupt routines, one to generate the servo signal and other to read the signal from the R/C receiver. The code is listed as an appendix to this report.

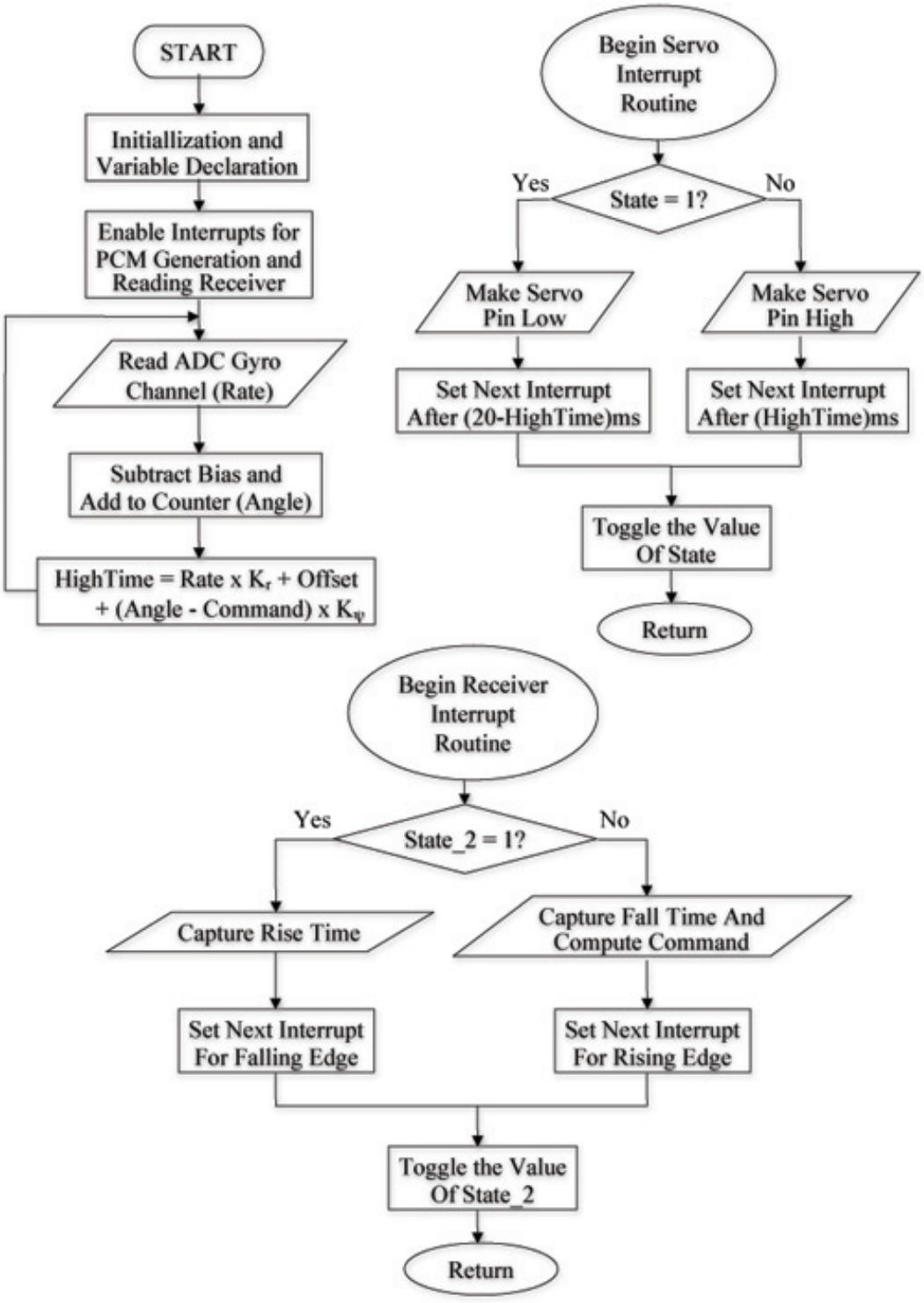

Figure 3.8: The flowchart for yaw control setup

3.2 Issues Involved in Implementation of Free-Flight

The goal of this project is to enable controlled free-flight of the MAV, that is, the Giant should not be constrained to the yaw-stand. The electronics therefore needs to be packaged and mounted on the vehicle. The source of power for this electronics should also be carried on the vehicle. Once the yaw-control is demonstrated in free-flight, it would be desirable if the same setup can later be modified for additional pitch and roll control, along with yaw control. The setup used for yaw control in the previous section (section 3.1.3) had all the basic components required for the 3-DOF control; it had an appropriate sensor package (with 3 gyros, 3 accelerometers and 3 magnetometers), a microcontroller to do the computations, and it was interfaced with the RC receiver so that control commands can be sent to it remotely. The microcontroller used here is on a development board which is designed for ease of use and not for least weight and size.

3.2.1 The Test Stand

The test stand used in the yaw control experiment allowed the use of microcontroller development board (bulky but easy to use), as only the IMU and the servos were directly mounted on the vehicle; rest of the hardware was kept on the table just below the Giant and was connected to the sensors and the actuators using long wires. A wireless telemetry link was not required for this experiment as the vehicle was not free to fly. The development of the miniature onboard stabilization system to enable free-flight of the Giant is discussed in this section.

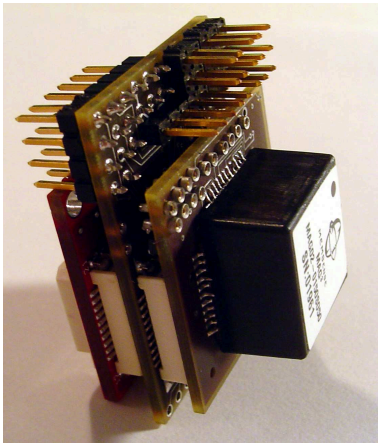

Figure 3.9: The avionics package for 3 DOF onboard controller

3.2.2 The Avionics Package

An avionics package (fig. 3.9) consisting of an IMU, three PIC 18F6722 microcontrollers and a transceiver, designed by Conroy [27], is used for onboard control implementation. This package, including the IMU and all other components weighs 30 grams. It consists of 3 boards (sensor board, actuator board and the transceiver board) that can be snapped together in different configurations.

3.2.3 System Performance

The autonomous yaw stabilization with pilot controlling the other degrees of freedom was successfully demonstrated on the Giant in free-flight using this setup. The 3 DOF attitude control, although based on the same basic principle as the yaw control setup, has several significant differences. The main issue is that of sufficiently accurate estimation of attitude states. The control can only be as good as the state estimation. While minor errors in estimation of yaw heading are not critical, relatively small errors in pitch and roll

estimation are not acceptable, as the vehicle would then develop large linear velocities and cease to hover. The 3 DOF system also involves the transformation of information from body to navigation frame (section 4.1). This was not required for the yaw control experiment as only one DOF was involved. We therefore need to work towards filter and transformation algorithms before a 3 DOF control can successfully be implemented.

3.2.4 Requirements for the MAV Control Test-bed

The experience with the onboard avionics package helped in the evolution of a list of features required in a setup for effective development of control laws and evaluation of different algorithms. The test-bed is required to have the following basic features:

- **Visibility of states; easy to debug:** The main problem with the onboard avionics package is that sufficient data is not being sent back to ground. It is therefore difficult to locate the problem and to debug the system. It is required that the test-bed allows several more intermediate states (e.g., raw sensor data, filtered data, pilot inputs, servo commands generated) to be continuously monitored for faster and effective evaluation of an algorithm.

- **Easy re-programability:** Re-programming the different microcontrollers on the avionics package is a very time consuming job. Firstly, all the algorithms need to be programmed in PIC-C (a variant of standard C language) and then optimized for execution on a microcontroller. Secondly, a number of wires need to be connected from the avionics package to the computer and power supply, each time a microcontroller has to re-programmed. This makes the entire process very cumber-

some and significantly reduces the number of different cases that one would like to experimentally evaluate.

- **More processing power:** The microcontrollers have very limited processing power and other resources. The programs therefore need to be carefully optimized to make sure they are executed as desired. Most of the times, it becomes difficult to locate if the source of error is in the algorithm itself or in the way it has been coded for the microcontroller. In order to first evaluate the algorithm, we would like to use a test-bed with much more processing power.

- **Safety features:** The design of onboard avionics package did not incorporate sufficient safety features. In case of malfunction of any electronics, there was no way to bypass the setup and take over manual control of the vehicle. This resulted in at least one major accident that led to some serious damage to Giant. It is therefore important that the test setup for such experiments incorporates better safety features.

Chapter 4

Attitude Estimation

In order to implement a 3 DOF attitude stabilization and control system, the primary task is to correctly determine and represent the attitude of the vehicle. The rotations in three dimensional space are not commutative. It is therefore important to be consistent with the conventions. Further, the attitude can be estimated using both the accelerometers and the gyroscopes; but both have their limitations and therefore filtering algorithms are necessary to get reliable attitude information. These issues are discussed in detail in this chapter.

4.1 Attitude Representation

Inertial navigation uses several reference frames [28], which are shown in figure 4.1. We restrict our discussion to the earth fixed navigation frame and the body fixed frame.

The origin of the navigation frame is located on the surface of the earth such that the Oz_e axis is directly towards the center of the spherical Earth. The Ox_e axis points

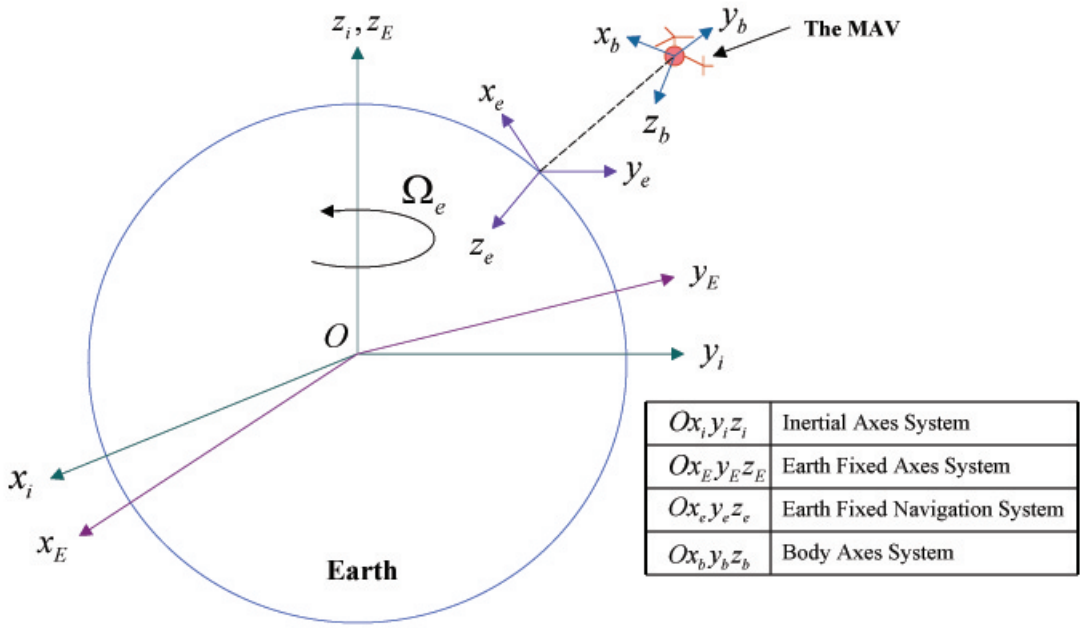

Figure 4.1: Reference frames used in inertial navigation

towards the local north and Oy_e points towards the local east to form a right-hand system. The location of the origin of this coordinate is chosen so that it lies beneath the vehicle at $t = 0$. Such a system is useful for defining the position of the vehicle with respect to the launch point and orientation of the vehicle, and with respect to the gravity vector and local north.

The body axes system on the Giant is defined in accordance with standard aerospace convention. The center of gravity of the Giant is the origin of this system. The Ox_b axis goes out the nose of the vehicle, the Oy_b axis is perpendicular to the Ox_bz_b plane and points towards the right side, as shown in figure 4.2. The Oz_b axis is the symmetry axis and points downwards so as to form a right-hand system. Since the vehicle is axi-symmetric about Oz_b axis, an orange sticker is placed on one side of the vehicle and nose is defined as being directly opposite to this orange sticker, as shown in figure 4.2.

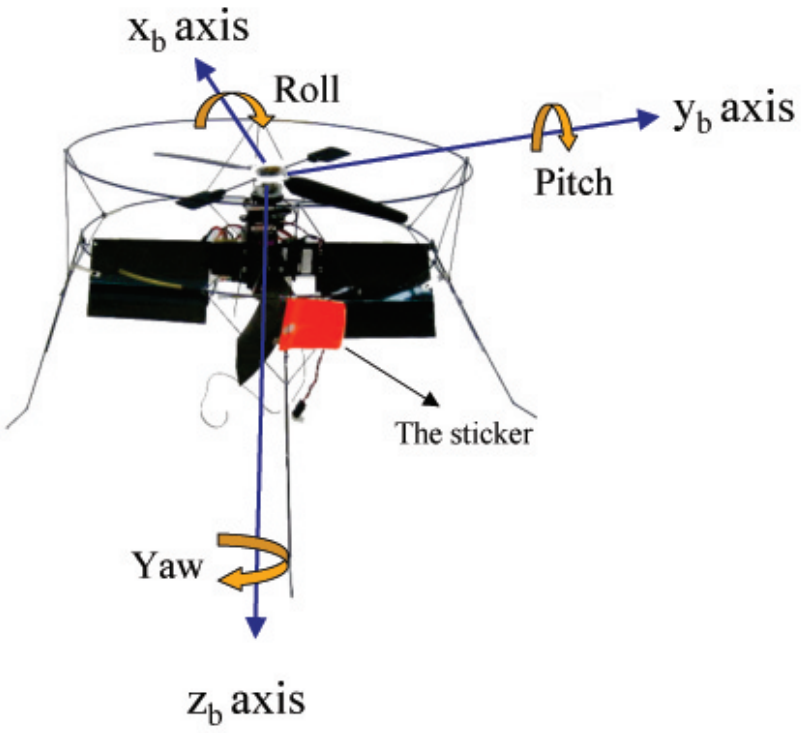

Figure 4.2: The body fixed reference frame for the Giant

Since we do not plan to use a magnetic heading sensor for initial experiments, we need not associate the Ox_e and Oy_e axes with north and east directions. At $t = 0$, the vehicle is on ground and is stationary. Hence, Oz_b axis is aligned with Oz_e. Therefore, to simplify interpretation of results, the Ox_e axis is defined to be parallel to Ox_b axis at $t = 0$; similarly, Oy_e axis is defined to be parallel to Oy_b at $t = 0$.

As the sensors are strapped to the body of the vehicle, the raw data is obtained in the body frame. This needs to be converted into navigation frame in order to determine the current position and attitude with respect to the initial states in earth-fixed reference frame. There are two main methods of carrying out this transformation: (1) using Euler angles, and (2) using quaternions (Euler parameters).

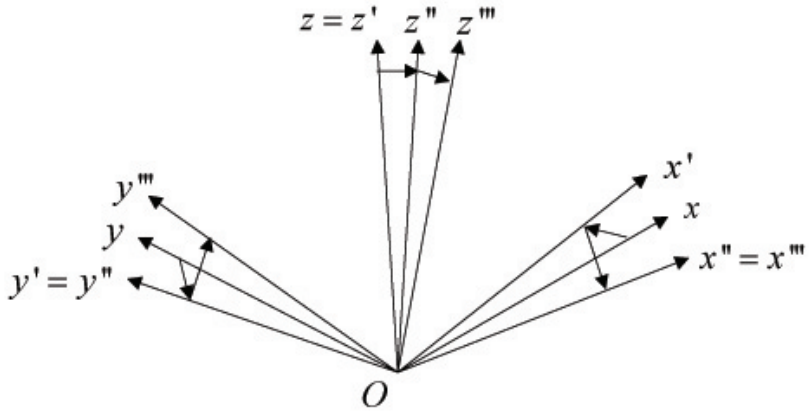

Figure 4.3: The Euler angle transformations

4.1.1 Euler Angles

The orientation of a given reference frame can be related to another reference frame by three Euler angles: ϕ, θ, ψ, or, the roll, pitch and yaw angles, respectively (fig. 4.2). In the present application, Euler angles give the orientation of the Giant in space with respect to the Earth. ψ is called the heading or yaw angle, θ the inclination or pitch angle, and ϕ the roll or bank angle. The orientation change is divided into three consecutive rotations, first about the body z-axis by angle ψ, then about the new body y-axis by angle θ, and finally about the newer still body z-axis by angle ϕ. In figure 4.3 the x, y, z are the original body axes, x', y', z' are the new body axes after first rotation, x'', y'', z'' are the new still body axes after second rotation, and x''', y''', z''' are the final body axes after third rotation.

We now derive the transformation matrix for such a sequence of rotations. Before the first rotation, the body-referenced coordinates match those of the navigation frame, i.e., $\vec{x}_b^0 = \vec{x}$. For first rotation, yaw about z-axis by angle ψ, the rotation matrix is given

by.

$$\vec{x}_b^1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{x}_b^0 = R(\psi)\vec{x}_b^0 \tag{4.1}$$

The transformation about z-axis does not modify the z-coordinate of the point. Other axes are modified according to the basic trigonometry. Now apply the second rotation, pitch about the new y-axis by angle θ.

$$\vec{x}_b^2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \vec{x}_b^1 = R(\theta)\vec{x}_b^1 \tag{4.2}$$

Finally, rotate (roll) the system about the newest x-axis by ϕ:

$$\vec{x}_b^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \vec{x}_b^2 = R(\phi)\vec{x}_b^2 \tag{4.3}$$

This represents the location of the original point in the fully-transformed body-reference frame $\vec{x}_b^3$ (henceforth called just $\vec{x}_b$). The equivalent transformation can be written as:

$$\begin{aligned} \vec{x}_b &= R(\phi)R(\theta)R(\psi) \\ &= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\sin\psi\cos\phi + \sin\theta\sin\phi\cos\psi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \sin\phi\cos\theta \\ \sin\psi\sin\phi + \sin\theta\cos\phi\cos\psi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\phi\cos\theta \end{bmatrix} \vec{x} \\ &= R(\phi,\theta,\psi)\vec{x} \end{aligned} \tag{4.4}$$

All the transformation matrices including $R(\phi,\theta,\psi)$ are orthonormal, i.e., their inverse is

equal to their transpose [28]. Therefore we can get the inverse transform as:

$$\vec{x} = R^{-1}\vec{x}_b = R^T\vec{x}_b \tag{4.5}$$

Similarly, a transformation matrix for angular rates can be derived. Given the angular rates in the body frame, the Euler rates can be computed using the current Euler angles as follows [28]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.6}$$

The most significant drawback of Euler angle scheme is the existence of singularity when pitch angle is $\pm$ 90 degrees, since at this angle both roll and yaw have similar effects. Thus, in applications where one has to encounter high values of pitch angle, a different way of measuring orientation is required.

4.1.2 Quaternions

Quaternions present a practical alternative to the Euler angles for attitude representation. These have the advantage that no singularity is encountered for any orientation. Fundamental to this method is Euler's theorem that body frame can be made to coincide with navigation frame by a single rotation D about a fixed axis in space making angles A, B, C

with the navigation frame [28]. Four parameters can be defined, e_0, e_1, e_2, e_3 as follows:

$$e_0 = \cos\frac{D}{2} \tag{4.7}$$

$$e_1 = \cos A \sin\frac{D}{2} \tag{4.8}$$

$$e_2 = \cos B \sin\frac{D}{2} \tag{4.9}$$

$$e_3 = \cos C \sin\frac{D}{2} \tag{4.10}$$

It can be verified that these four parameters are constrained by:

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{4.11}$$

The transformation matrix in equation (4.4) can now be written as:

$$R(\phi,\theta,\psi) = \begin{bmatrix} e_0^2+e_1^2-e_2^2-e_3^2 & 2(e_1e_2+e_0e_3) & 2(e_1e_3-e_0e_2) \\ 2(e_1e_2-e_0e_3) & e_0^2-e_1^2+e_2^2-e_3^2 & 2(e_2e_3+e_0e_1) \\ 2(e_0e_2+e_1e_3) & 2(e_2e_3-e_0e_1) & e_0^2-e_1^2-e_2^2+e_3^2 \end{bmatrix} \tag{4.12}$$

The relation between Euler angles and Euler parameters can be derived as [28]:

$$e_0 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \tag{4.13}$$

$$e_1 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \tag{4.14}$$

$$e_2 = \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \tag{4.15}$$

$$e_3 = -\cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} \tag{4.16}$$

The rate equations for the four quaternion parameters, given the present value of these parameters are [28]:

$$\dot{e_0} = -\frac{1}{2}(e_1 p + e_2 q + e_3 r) \tag{4.17}$$

$$\dot{e_1} = \frac{1}{2}(e_0 p + e_2 r - e_3 q) \tag{4.18}$$

$$\dot{e_2} = \frac{1}{2}(e_0 q + e_3 p - e_1 r) \tag{4.19}$$

$$\dot{e_3} = \frac{1}{2}(e_0 r + e_1 q - e_2 p) \tag{4.20}$$

4.2 Attitude Determination using Gyroscopes

The gyroscopes are strapped to the body and they give a direct measure of angular rates in body frame, i.e., p, q and r. These body rates are transformed into navigation frame (Euler rates) using equation (4.6). For this computation step, the values of Euler angles at the previous instant are used (dead-reckoning) and the result is then numerically integrated to obtain the new Euler angles. The algorithm is initialized with zero Euler angles when the vehicle is on ground, to start with.

Gyro bias, defined as the output produced by gyro at rest, leads to drift in determination of orientation. Fixed bias, if uncompensated, leads to a constant rate of drift after integration. However, the startup bias can be measured before take-off and corrected. What matters then is bias stability. The typical drift performance of a ring laser gyro is about 0.001 deg/hr, which is sufficient to get accurate position within about a mile for one hour [29]. On the other hand, the recently developed very low cost and miniature Coriolis vibratory gyroscopes (CVGs) have drift rates ranging from several degrees per hour to a

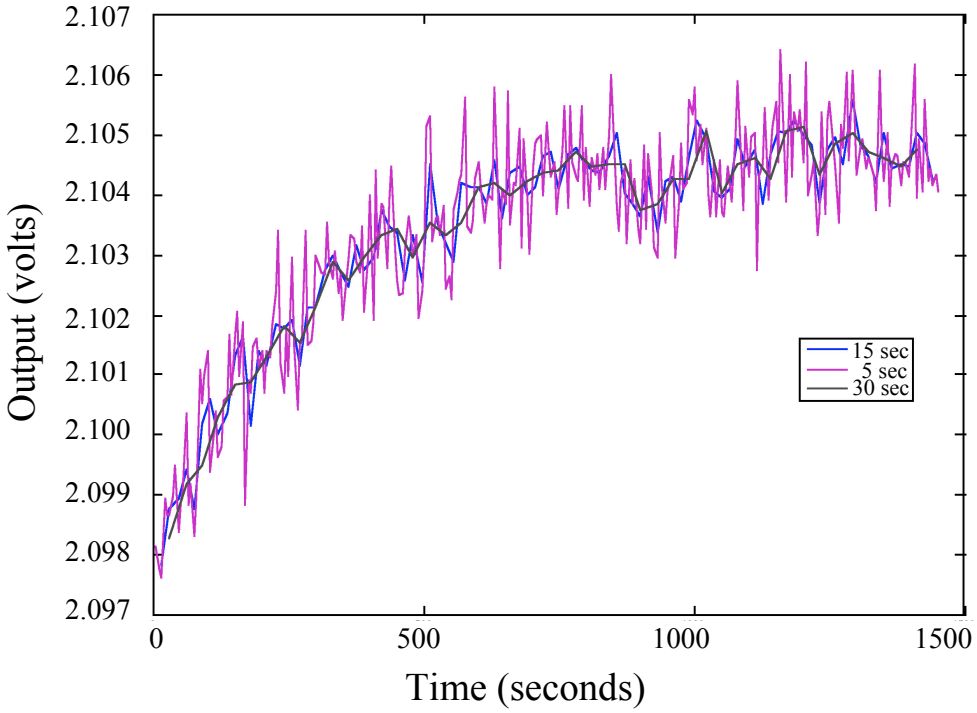

Figure 4.4: Determining gyro bias

degree per second!

A gyro at rest is sampled for a period of time and the average of all the samples collected gives the bias of the gyro. An experiment to demonstrate non-constant gyro bias was conducted. The ADXRS150 gyroscope [30] from Analog Devices is used for this experiment. Figure 4.4 shows three different lines corresponding to the averages taken over 5, 15 and 30 seconds repeatedly in order to estimate bias. The gyro is sampled at a rate of 100 samples per second using a 12 bit analog-to-digital converter. The calibration factor for the gyroscope is 60 degrees per volt. This data is collected after the sensor has been on for over 2 hours so that a stable temperature has been reached. This shows that it is not possible to assign a single exact value to gyro bias as the output for zero angular rate keeps changing over the time. Any choice of gyro bias will lead to drift in the integrated

result. Further discussion on the sources of gyro drift is included in the appendix to this report.

4.3 Attitude Determination using Accelerometers

The accelerometers, strapped to the vehicle, measure the tilt by registering changes in the component of gravity along their axes. Assuming, linear accelerations are negligible compared to the acceleration due to gravity, the pitch and roll angles can be computed from the accelerometers. Using transformation equations 4.4, 4.5, we get:

$$a_x = -g\sin\theta \tag{4.21}$$

$$a_y = g\sin\phi\cos\theta \tag{4.22}$$

$$a_z = g\cos\phi\cos\theta \tag{4.23}$$

We therefore get:

$$\theta = \sin^{-1}\frac{-a_x}{g} \tag{4.24}$$

$$\phi = \tan^{-1}\frac{a_y}{a_z} \tag{4.25}$$

where, a_x, a_y and a_z represent the accelerations (due to gravity) measured by the three orthogonal accelerometers aligned along the body x, y and z axes, respectively, and g represents the acceleration due to gravity.

A serious drawback of the accelerometer is its sensitivity to vibrations. To eliminate this, a low frequency filter is required which limits the bandwidth of the accelerometers and introduces a phase delay in the filtered result.

4.4 Complementary Filter

We have seen that either gyroscopes or accelerometers can be used to determine attitude. However, each sensor has serious limitations when used alone; a gyroscope has long term divergence problem due to drift (bad for measuring low frequency rates) and an accelerometer has bandwidth problem due to low pass filter (bad for measuring high frequency rates). These limitations are demonstrated by the results of the experiments described in the next section.

4.4.1 The Pendulum Experiment

A thin aluminum beam is suspended from the rotational axis of a potentiometer to form a pendulum. The potentiometer used is a contactless Hall effect based angle sensor [31]. This potentiometer gives analog output proportional to the pendulum angle within a range of ± 40 degrees. The output of the potentiometer is nonlinear outside this range. The IMU discussed in section 5.3.1 is mounted on the aluminum beam 4.5. The data from the gyroscope is integrated to get the attitude. We also get attitude from the accelerometers using equation 4.24. The data from the accelerometers and gyroscopes on the IMU can now be compared with the attitude given by the potentiometer (the true angular position of pendulum).

The IMU is setup with the bluetooth transmission of sensor data. The potentiometer is sampled separately at a rate of 500 Hz using a 12-bit analog-to-digital converter. All data is displayed on a chart in LabVIEW. To filter the noise, 5 potentiometer readings are read at a time and averaged. Similarly, a moving average of last 10 samples is used on

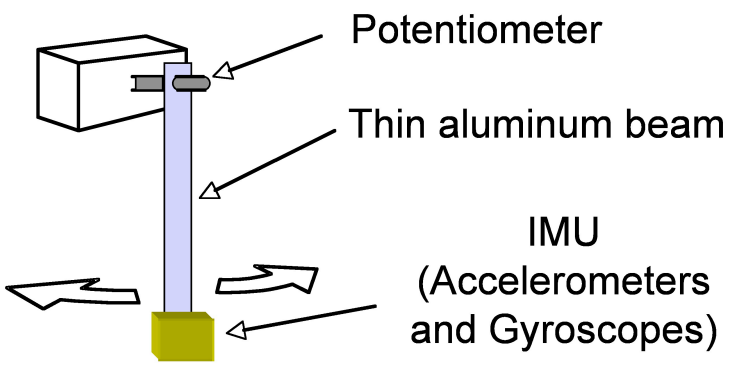

Figure 4.5: Pendulum Experiment Setup

the accelerometer data. The angle data from the gyroscopes is quite smooth because of integration; so no filtering is required for the gyroscope.

In the first case, the IMU is mounted on the axis of rotation. This makes sure that the accelerometer is not subjected to any linear accelerations and it thus measures only the change in the component of gravity along its axis (hence the angular position of pendulum). The pendulum is now given a push by the hand and the data from 3 different sensors, i.e., the potentiometer, the gyroscope and the accelerometer is recorded. This is shown in figure 4.6. The accelerometer data shows a good match with the potentiometer data. The attitude from the gyroscope is seen to drift away from the true attitude (from potentiometer) with time.

The IMU is placed at a distance of 14 inches away from the axis of rotation on the pendulum for the second test. The results for this test are shown in figure 4.7. When the pendulum is held at some attitude with hand, the accelerometer is seen to give correct attitude, but when in motion, it is no longer able to give correct attitude estimate. This is because now it is subject to linear acceleration along with changes in the magnitude of gravity along its axis. In fact, when the pendulum is allowed to swing freely, the ac-

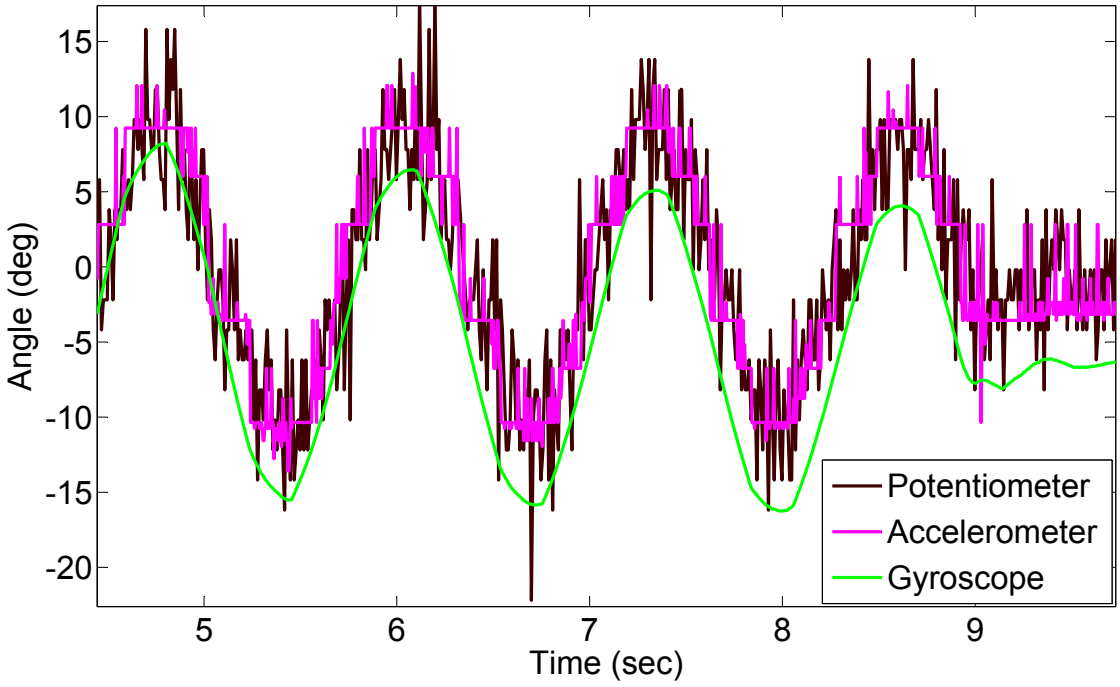

Figure 4.6: Performance of accelerometer and gyro for attitude estimation (on axis)

celerometer does not measure any angle. This is because the linear accelerations exactly cancel the change in the magnitude of the gravity along its axis (in the linear approximation, both equal to $g\theta$ in different directions; where θ is the angle of the pendulum with the vertical). The attitude from the gyroscope is again seen to drift away from the true attitude with time. The pendulum was given large oscillatory perturbations by hand towards the end of experiment (as seen towards the right end of figure 4.7). We therefore notice that the attitude form the accelerometers is way off during such situations, i.e., in the presence of high linear accelerations, or vibrations.

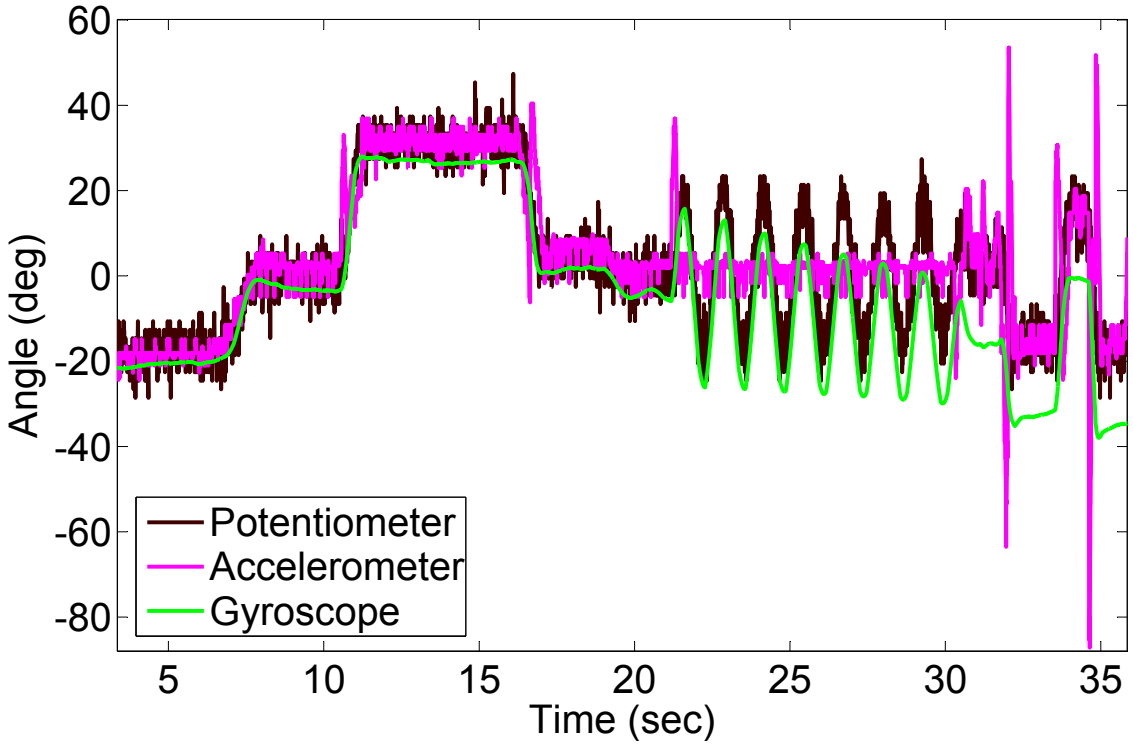

Figure 4.7: Performance of accelerometer and gyro for attitude estimation (off axis)

4.4.2 Design of Complementary Filter

The complementary filter is used to fuse the outputs of different sensors with complementary noise characteristics [32]. In this case, rate gyros provide a good high frequency estimate of attitude and the accelerometers provide a good low frequency estimate, as discussed before. The best estimate of the orientation is thus obtained as the sum of the signals from two measurement branches as shown in figure 4.8. The accelerometers feed output signal into the filter $G_a(s)$, which provide the contribution of the accelerometer branch to the low frequency estimate of attitude. The output of rate gyros is fed into the filter $G_g(s)$, which provides the high frequency estimate of attitude.

In order to have constant amplification of the entire system, the filter transfer func-

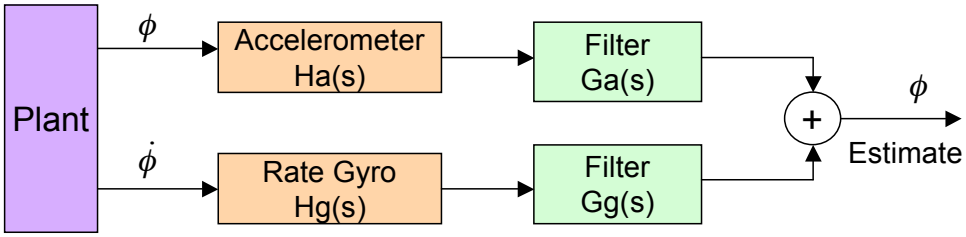

Figure 4.8: The complementary filter schematic

tions $G_a(s)$ and $G_g(s)$ must be chosen according to [32]:

$$H_a(s)G_a(s) + sH_g(s)G_g(s) = 1 \tag{4.26}$$

where, $H_a(s)$ and $H_g(s)$ denote transfer functions of the accelerometer and gyro respectively. We assume the sensors to be ideal and hence use:

$$H_a(s) = H_g(s) = 1 \tag{4.27}$$

This is assumption is valid because the dynamics of sensors is much faster than that of the vehicle itself. Therefore, equation (4.26) is reduced to:

$$G_a(s) + sG_g(s) = 1 \tag{4.28}$$

A second order filter with a double pole is now selected:

$$G_a(s) = \frac{4s+1}{(2s+1)^2} \tag{4.29}$$

$$G_g(s) = \frac{4s}{(2s+1)^2} \tag{4.30}$$

Here, the transfer function $G_a(s)$ works as a first order low-pass filter (-20 db per decade) on the signal coming from the accelerometer (fig. 4.9). The other transfer function, $G_g(s)$ works as a second order high pass filter with respect to the orientation (-40 db per decade), on the signal from the gyroscope (fig. 4.10).

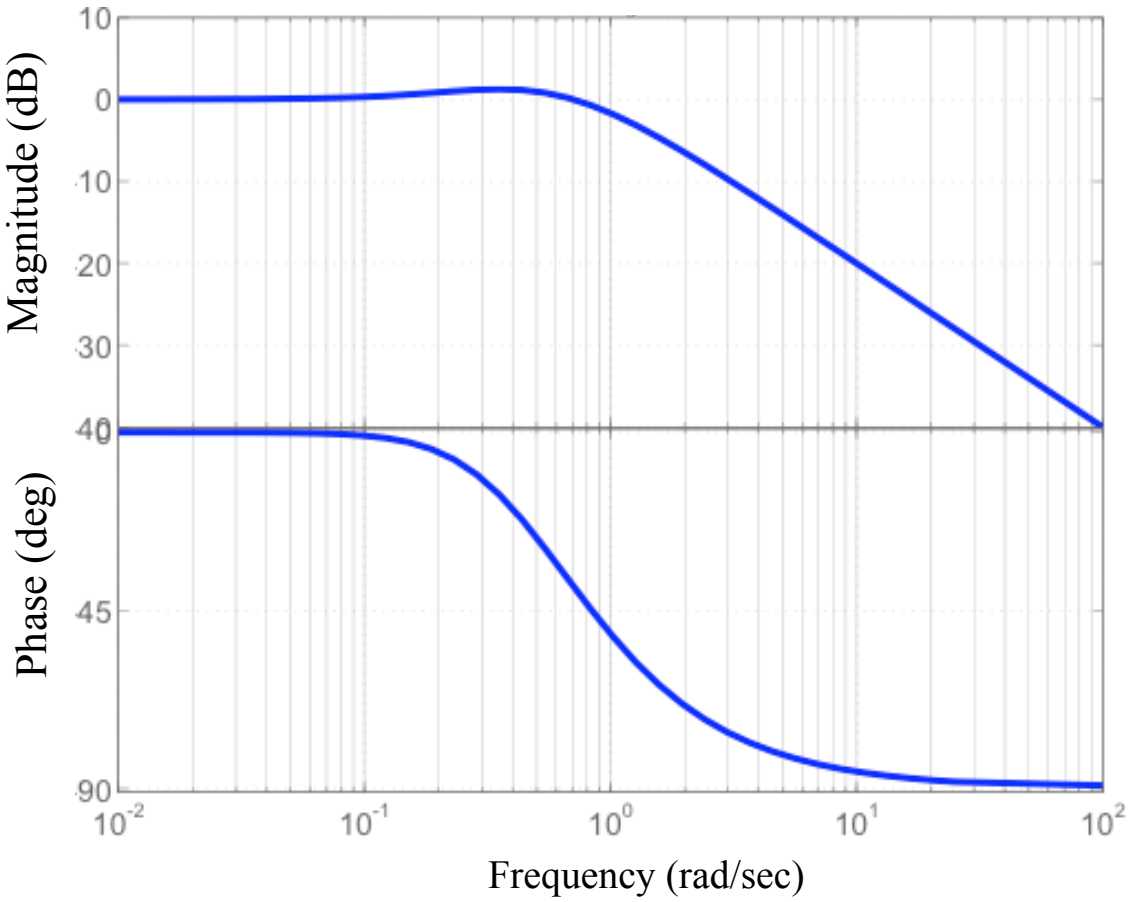

Figure 4.9: Bode plot of the accelerometer filter transfer function (low pass)

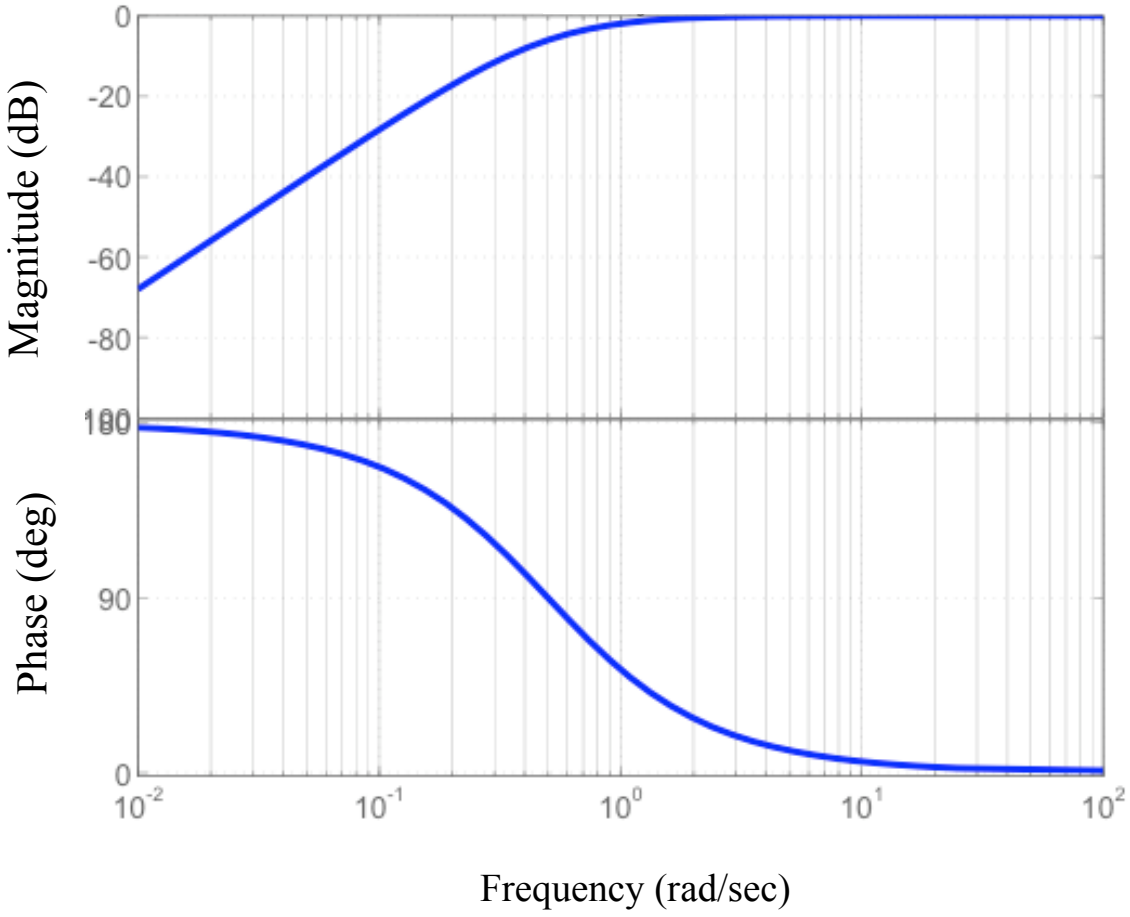

Figure 4.10: Bode plot of the gyroscope filter transfer function (high pass)

4.4.3 Performance of Complementary Filter

The pendulum experiment described in section 4.4.1 is now repeated with the complementary filter designed above. The results for this case are shown in figure 4.11 (no accelerometer data) and figure 4.12 (with accelerometer data). We see that although the attitudes from both the accelerometer and the gyroscope do not match very well with the true attitude (potentiometer), for some or the other region of the chart, the attitude from the complementary filter shows a very good match for all regions. It takes care of both the drift in the gyros (low frequency phenomenon) and the linear accelerations in the accelerometers (mostly due to vibrations, a high frequency phenomenon). Close-up views of of different regions of this plot are shown in figures 4.13, 4.14.

4.5 Position Determinition using INS

Linear position determination is a much harder problem than attitude estimation. Firstly, there are accelerometer instrument errors such as bias stability, scale factor stability, nonlinearity and misalignment. Inertial grade accelerometers must keep these errors to a few micro-g to get an accuracy of 1 mile/hr. Since position is obtained by double integrating acceleration, leads to a position drift error that grows quadratically in time. It is therefore especially critical to accurately estimate and eliminate any persistent bias errors. The second critical cause of error in position measurement is error in the orientation determined in the gyros. Since the INS interprets the direction of the measured acceleration according to the computed orientation of the platform, and error in this computer orientation will cause it to integrate the accelerations in the wrong direction, thus deviating from

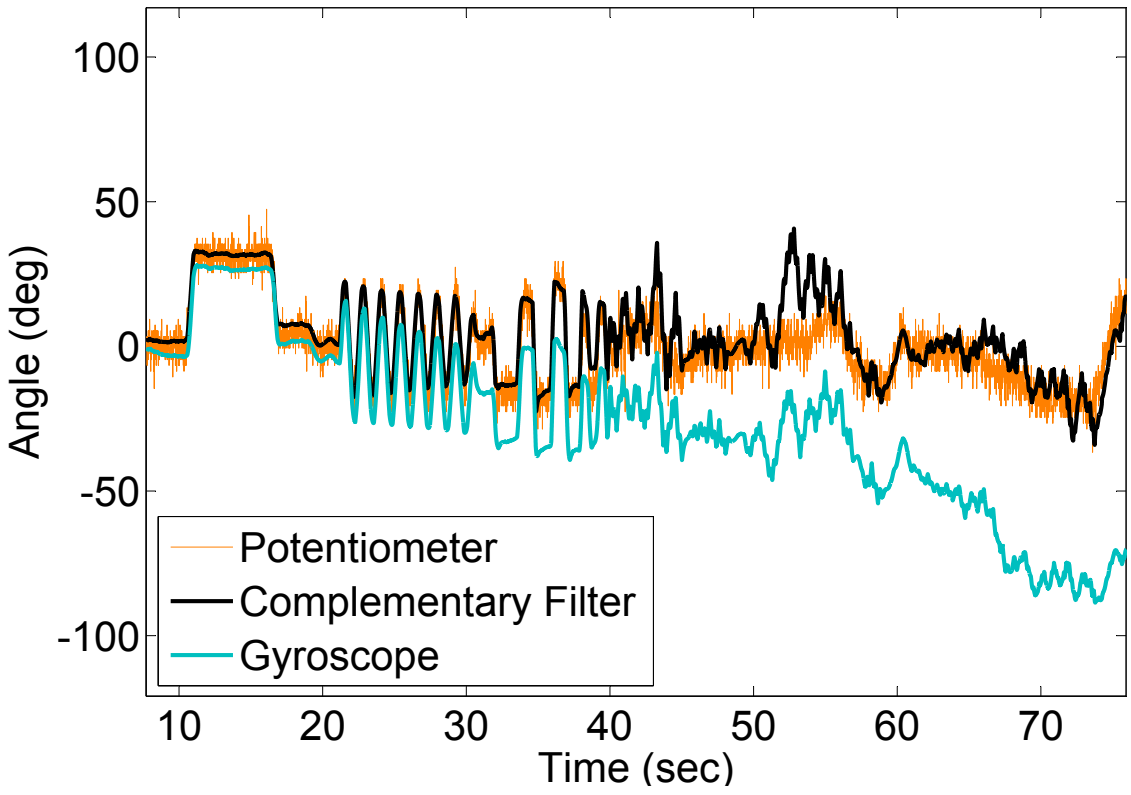

Figure 4.11: Performance of complementary filter for attitude estimation (no accelerometer)

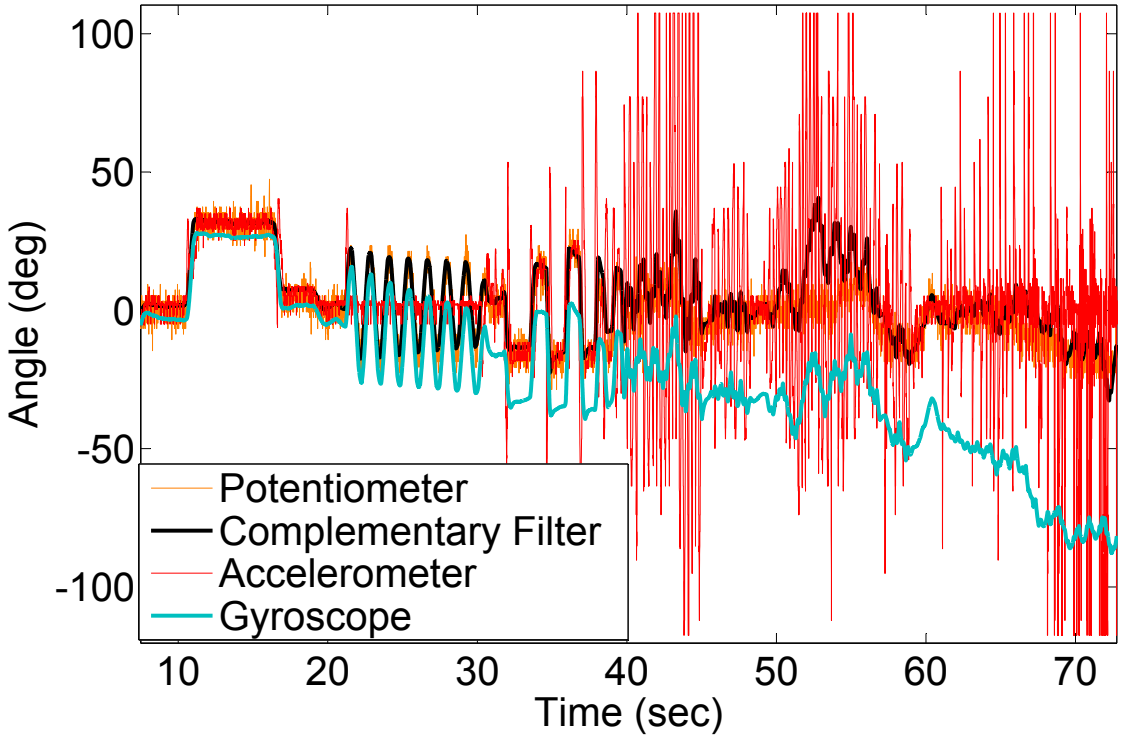

Figure 4.12: Performance of complementary filter for attitude estimation (with accelerometer)

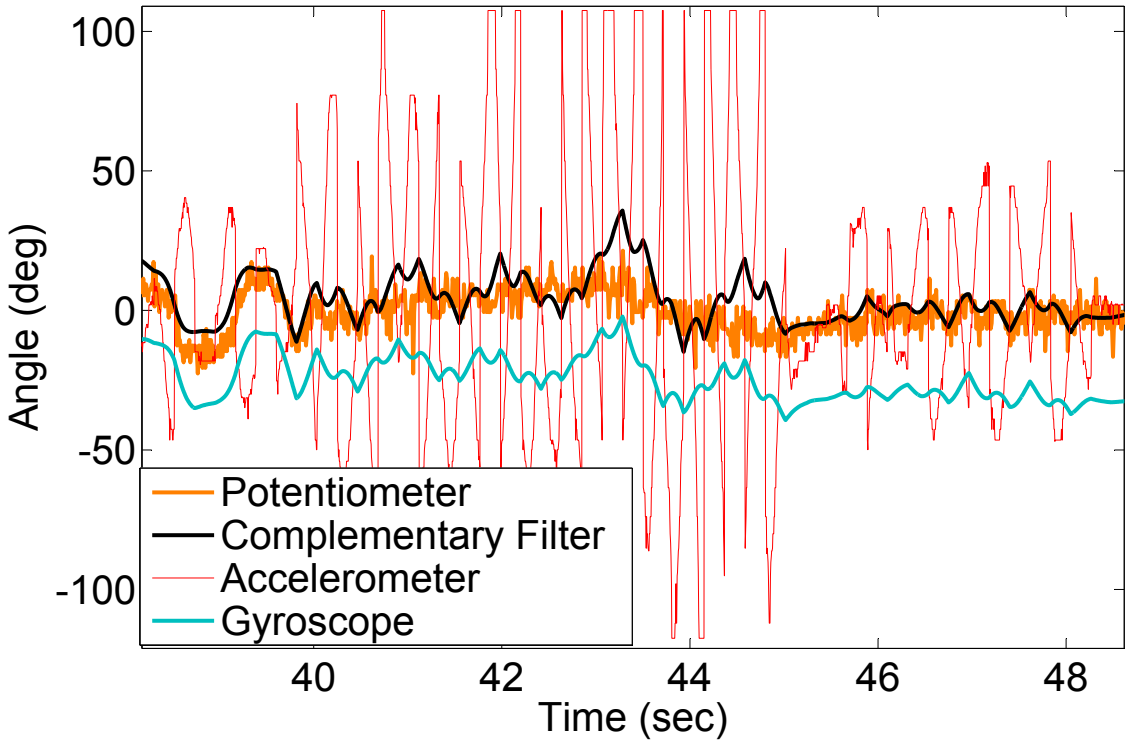

Figure 4.13: Complementary filter performance (accelerometer saturation)

the true course of the vehicle. More importantly, the cancellation of gravity will be performed imperfectly by the navigation computer. This will cause a horizontal acceleration component to be erroneously added to the navigation-frame acceleration vector. Thus, even gyros have a more critical accuracy requirement in order to get an accurate position estimate using INS.

4.6 IMU Aiding and Kalman Filtering

Although figure 2.5 shows how IMU is theoretically self-sufficient for the purpose of both attitude and position determination, we now realize that practically it can only be reliably used for attitude determination (provided linear acceleration are negligible compared to

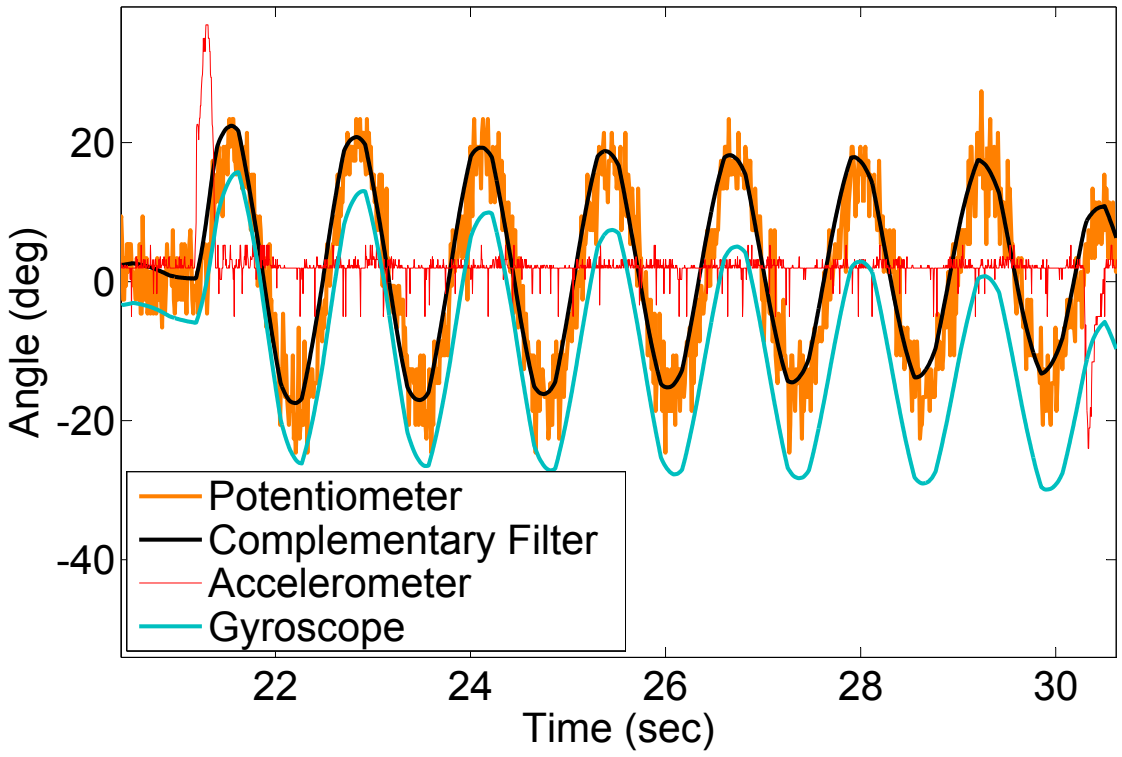

Figure 4.14: Complementary filter performance (free oscillations)

acceleration due to gravity). Therefore, IMUs are normally one component of a navigation system. Other systems such as GPS (used to correct for long term drift in position), a barometric system (for altitude correction), or a magnetic compass (for attitude correction) compensate for the limitations of an IMU [33]. Note that most other systems have their own shortcomings which are mutually compensated for.

In case of accelerometers and gyroscopes, there is a clear distinction in the frequency range where each gives a reliable estimate of attitude. We are therefore able to exploit this property and devise a complementary filter. Where many different systems are used, the region of reliability can always not be marked as clearly in terms of frequency or magnitude. Such situations need more advanced filtering algorithms, like Kalman filter, that can optimally fuse all the available data, utilizing sensor and their noise characteristics [34].

Chapter 5

The Ground Based Setup

5.1 PID Configuration

We need to find a suitable control algorithm that can be used for autonomous flight of the Giant, and then build a test-bed for its implementation and performance evaluation. A human pilot applies lateral cyclic in order to control the roll degree-of-freedom. Similarly, one applies longitudinal cyclic or vane input to control pitch or yaw degrees-of-freedom, respectively. For a full-envelope flight, it is important to consider coupling between different axes and nonlinear dynamical effects; thus a SISO (single input, single output) PID framework is clearly not suitable. But, for a near-hover flight, roll, pitch and yaw can be treated as uncoupled and a linear SISO PID controller is expected to give satisfactory results [19]. The schematic of the PID control scheme that we plan to implement on the Giant is shown in figure 5.1.

The inner attitude-control loop (higher bandwidth) forms the core of the controller and directly interacts with the vehicle to achieve the required attitudes. The outer translation-

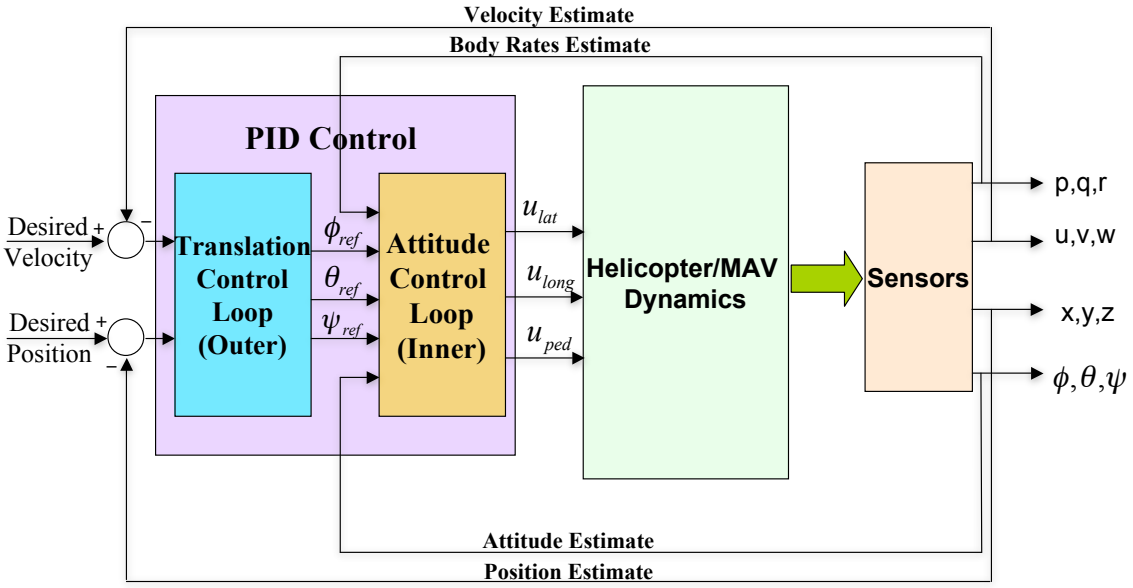

Figure 5.1: Schematic of PID control for autonomous flight of Giant

control loop (lower bandwidth) is the secondary controller, which takes translation commands from the user and generates equivalent attitude commands for the inner loop (which in turn controls the vehicle).

Next, we need to look for appropriate sensors to implement these two control loops. The attitude-control loop needs the sensors that can provide information regarding the 3-DOF attitude of the vehicle, and the translation-control loop additionally requires the 3-DOF position information. In the present work, we focus on the attitude control loop of the controller and discuss in detail how to obtain a good estimate of attitude of the vehicle using low cost sensors. The detailed schematic of this controller is shown in figure 5.2.

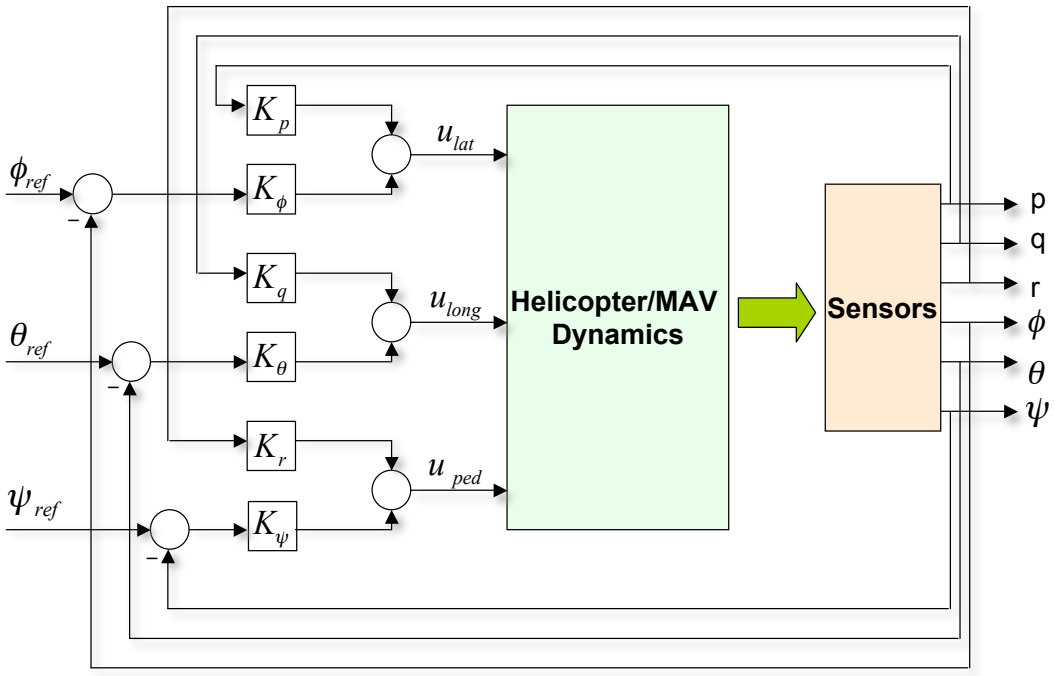

Figure 5.2: Schematic of attitude control loop (inner-loop)

5.2 Overview of the New Test-Bed

The design of the new test-bed, incorporating the features listed in section 3.2.4, is now discussed. This is a ground based setup with the main idea to do minimal work onboard and to transfer all the complex jobs to the ground station. Ground station can employ much faster processors and can help graphical monitoring of much more data than what can be done using onboard microcontrollers. The architecture for such a system is shown in figure 5.3.

Here, only the sensors and the actuators are placed on the vehicle. The sensors send the raw sensed data to the ground station using a wireless transmitter. All the processing is done on ground and the final command are linked back to the vehicle using the wireless equipment.

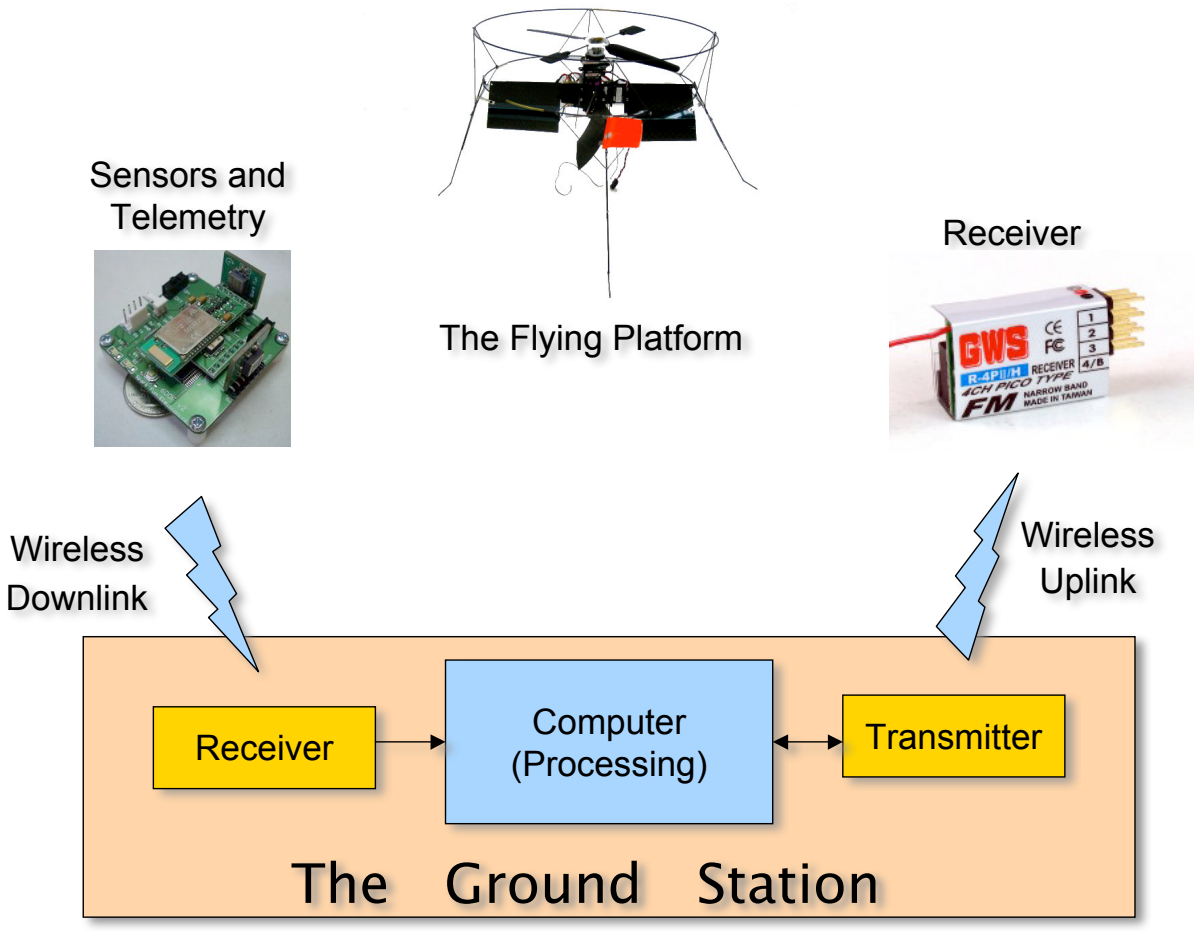

Figure 5.3: Architecture of the ground based test-bed

One of the most important requirement for such a system is redundancy. That is, should the control system fail, the Giant should revert back to normal manual controls. The wireless links employed need to be very reliable, as any communication failure could lead to loss of computer control of flight. Since the standard R/C transmitter-receiver is known to provide very reliable communication, is inexpensive, easily available and redundancy can be easily built using it, hence it was decided to use the same for uplinking the commands from the ground station. As mentioned earlier in section 2.2.2, the input pin on the data port of the transmitter can be used to transmit data received from a source external to transmitter. The servos and the speed controller on the Giant (or any other MAV in use) are connected to the same R/C receiver as in the case of normal manual flight configuration.

5.3 Subsystems of the Setup (Hardware)

This setup consists of 3 main subsystems (fig. 5.3); first, consisting of the onboard sensors and the wireless transmission of this data; second, reception of this data on ground and its processing to generate the commands for the actuators; and third, consisting of the interface with the R/C transmitter to uplink these commands to the Giant and to read pilot inputs.

5.3.1 The Sensors and Telemetry

A sensor-board with Bluetooth data transmission capability was obtained from Spark Fun Electronics [35]. It consists of a MMA7260Q single IC triple axis accelerometer [36]

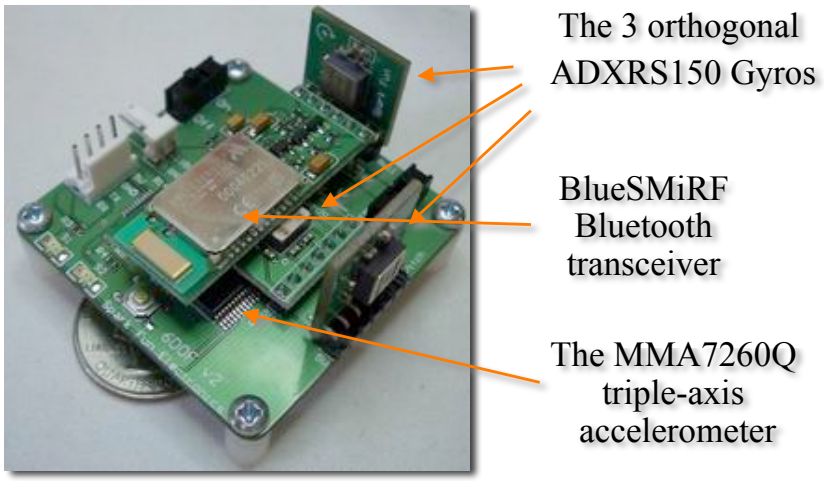

Figure 5.4: The wireless IMU from Spark Fun Electronics [35]

from Freescale and three ADXRS150 gyroscopes [30] from Analog Devices, mounted along three orthogonal axes as shown in figure 5.4. This sensor-board weighs about 22 grams. The analog data from these six sensors is sampled by PIC16F88 microcontroller and it transmitted to ground through an onboard BlueSMiRF Bluetooth transceiver [37].

The firmware on microcontroller allows for selection of appropriate data channels to be sampled and the range for accelerometers, as described in the datasheet [38]. Six channels corresponding to sampled data from the 3 accelerometers and 3 gyroscopes are selected from the firmware menu. A range of $\pm 2g$ is selected for accelerometers. The gyroscopes have a range of ± 150 deg/sec. All the analog channels are sampled using a 10-bit ADC. Therefore, each sample consists of 2 bytes in the data-stream. The lower 10 bits are occupied by the sample and the upper 6 bits read as zeros. Each data packet, consisting of one sample each from all 6 channels, starts with an 'A, then has 12 bytes corresponding to readings from 6 sensors and concludes with an end byte 'Z. Therefore, 14 consecutive bytes, starting with an 'A' and ending with a 'Z', make a packet. The

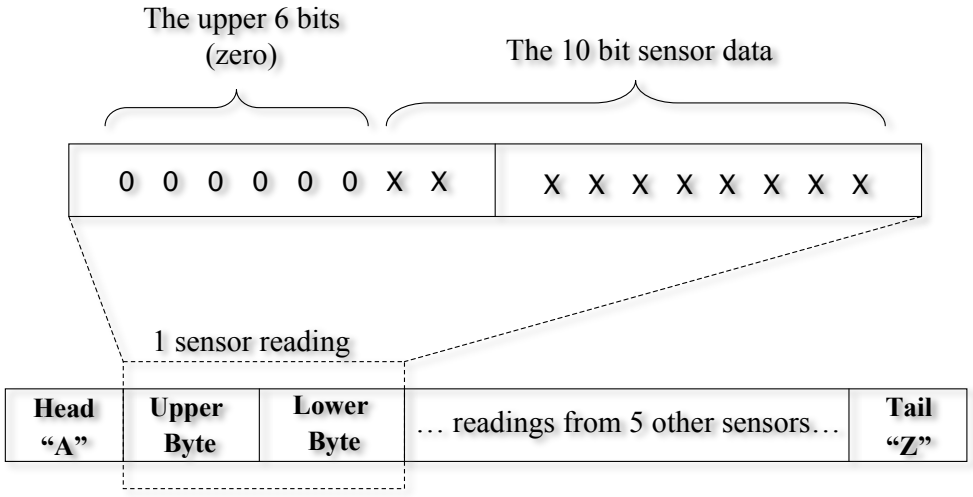

Figure 5.5: The structure of a sensor data packet

structure of the packet is illustrated in figure 5.5. A data rate of about 230 packets/second is achieved using the 57.6 kbps bluetooth wireless link. A bluetooth dongle is used on the ground station PC to receive the data from the wireless IMU.

5.3.2 The R/C Transmitter Interface

The generation and sampling of high frequency pulses, as required by the transmitter for communication (discussed in section 2.2.2), is time critical and cannot be directly done by a PC running a normal non-real-time operating system (e.g., MS Windows, Linux, or Mac OS). We, therefore, either need a PC with real-time operating system, or a microcontroller to interface the transmitter with a PC running a non-real-time operating system. The latter option is chosen, as it simplifies application development using LabVIEW and is also faster and cheaper to implement. The microcontroller communicates with the PC using serial port (RS232) and with the transmitter using PPM pulses (fig. 2.12), as shown in figure 5.6.

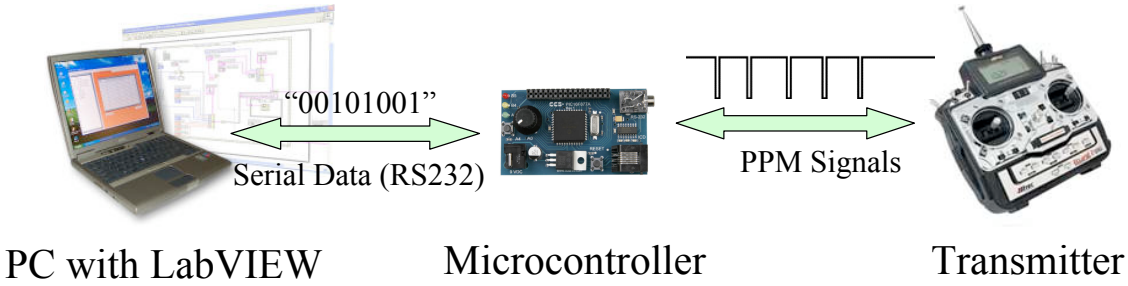

Figure 5.6: The interface of the transmitter with the PC

The radio control system used has 6 channels. Therefore, each frame of PPM (discussed in section 2.2.2) consists of 6 pulses corresponding to each of these channels. The system can easily be modified to use transmitters with 4, 8 or any other number of channels. The transmitter is used both for sending commands to the vehicle, as well as, as an input device, i.e., to get pilot inputs through the sticks. Therefore, the microcontroller in this interface does two jobs:

1. **Writing data to transmitter**: The PC sends a stream of six 8-bit numbers to the microcontroller over the serial port, corresponding to the width of each of the six pulses in a PPM frame. For servos to hold position and for the effective working of the system, a new frame is required every 20 ms. The microcontroller is programmed to continue to repeat the same PPM frame till data for a new frame is received from the PC. In order to synchronize, the decimal number '255' (a byte with all 1's) is sent by the PC before every frame. Once the microcontroller receives this synchronization byte, the next 6 bytes are read as data corresponding to the 6 channels, respectively.

2. **Reading data from transmitter**: The data from the transmitter is continuously

sampled and the width of each pulse in a frame is calculated by the microcontroller. This data is then sent to the PC as six 8-bit numbers over the serial port, corresponding to each of the six channels. A pulse of width larger than 2.5 ms marks the beginning of a new frame. For communication with PC, the microcontroller first sends a byte with all bit as 1's (for synchronization) and then 6 bytes corresponding to the 6 channels of data, in order.

Figure 5.7 shows the flowchart for the microcontroller code. The PIC-C code is included in the appendix to this report.

5.3.3 The Virtual Instrumentation on PC

The Virtual Instrumentation (VI) on the PC, using the National Instrument's LabVIEW software [39], forms the core of this experimental setup and the rest of discussion will primarily concentrate on its implementation. The sensor data is received, processed and commands for the transmitter are generated through this VI. Almost all the processing is therefore done on the ground station PC using this VI. This provides a very efficient way to make changes to the controller and conduct different experiments.

5.4 Organization of the Virtual Instrument (Software)

The in-house developed VI can be divided into 3 distinct modules as discussed below. These modules communicate using a set of well-defined protocols, i.e., the type and format of input to, and output from each module is defined. This allows for replacement or up-gradation of any of the module without disturbing the working of rest of the VI. Figure

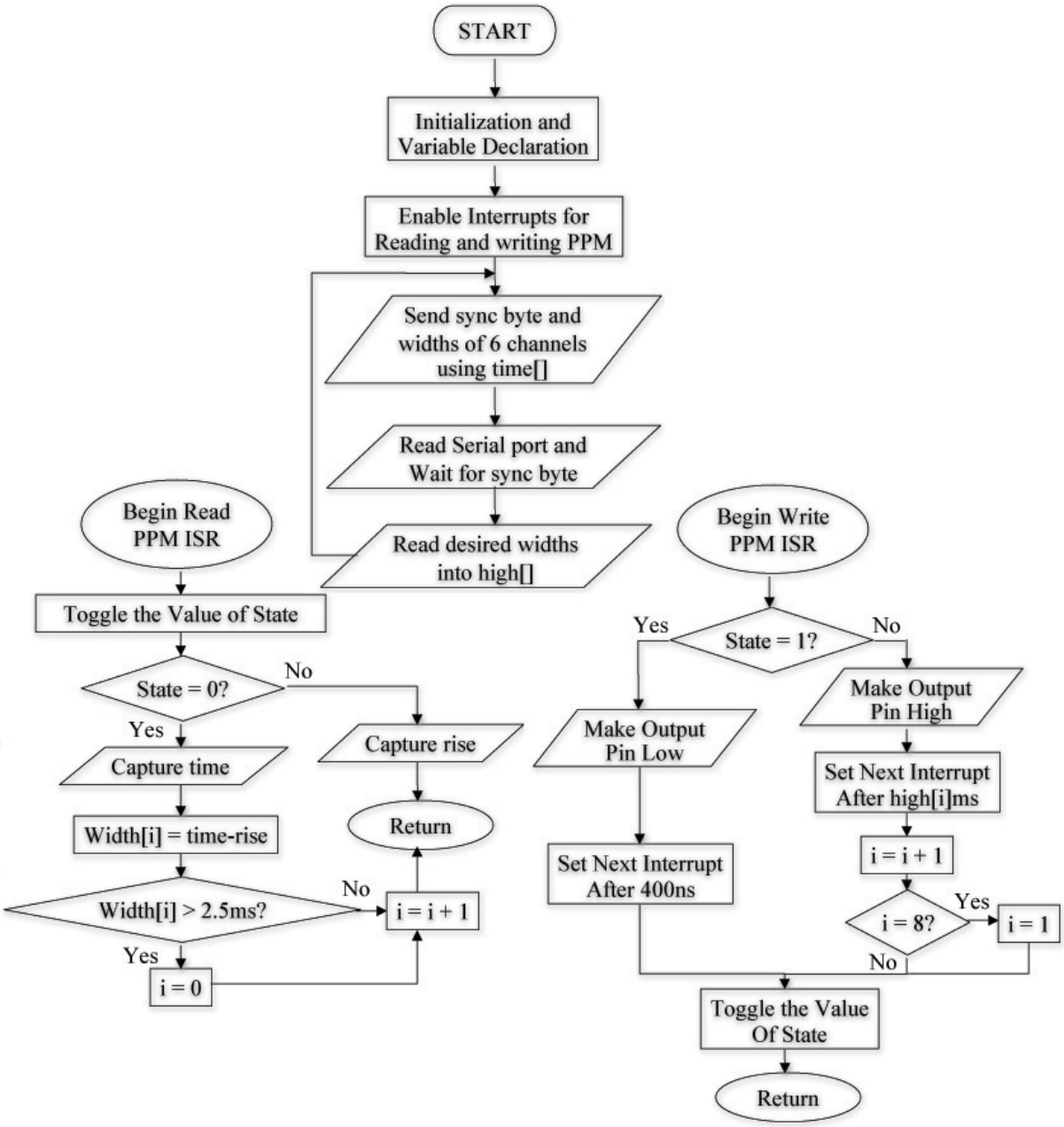

Figure 5.7: The flowchart for the transmitter interface code

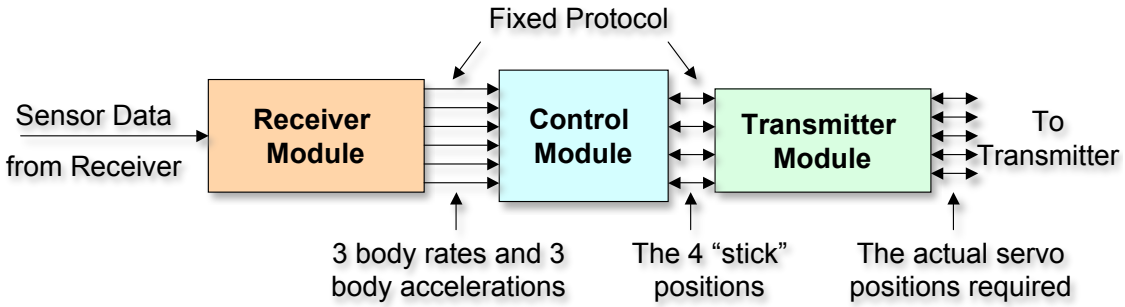

Figure 5.8: The 3 modules of the VI and their interfaces

5.8 shows a schematic of the interface between these modules.

5.4.1 The Receiver Module

The receiver module is coded as a separate sub-VI. It reads the data from the Bluetooth receiver via serial port. The data is received as a packet of 14 bytes (discussed in section 5.3.1). The receiver module makes use of the packet delimiters, 'A' and 'Z', to check for validity of data. If the packet does not have appropriate delimiters, it is discarded and the module waits for next valid packet. After successful validity check, the 12 middle bytes of a packet are fused together, taken 2 at a time to retrieve the 6 sensor readings of 10-bit each. The raw sensor data thus obtained is converted into meaningful physical quantities, i.e., angular body rates in radians per second and linear accelerations in gs, using the calibration data, and passed on to the processing module. The working of the receiver module is shown in figure 5.9.

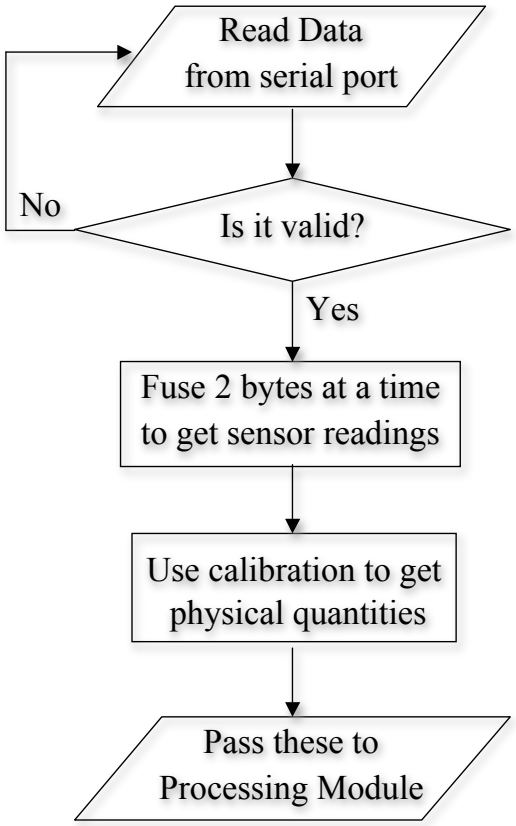

Figure 5.9: The receiver module

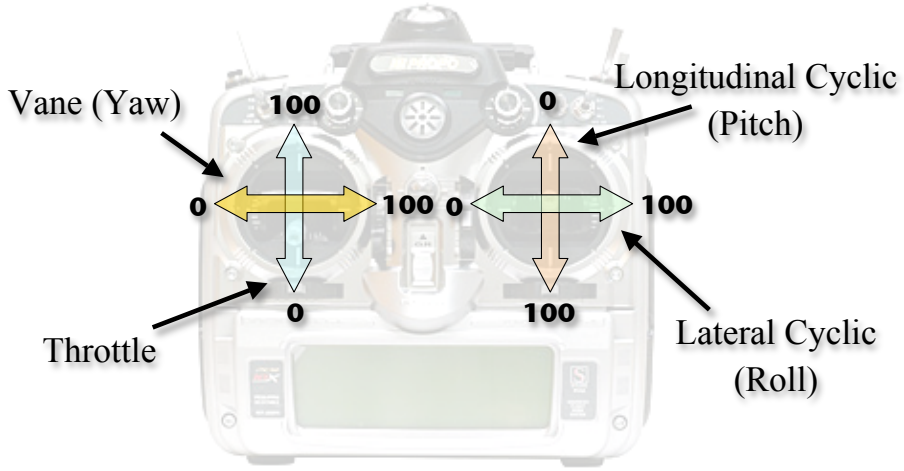

Figure 5.10: The definition of stick positions

5.4.2 The Processing Module

The processing module first transforms the data received from the receiver module into the navigation frame (section 4.1). The transformed data is then used to obtain the body attitude using a complementary filter. Finally, a PID controller computes the desired stick positions, or the amount of cyclic and anti-torque, and passes on this information to the transmitter module. The value of stick positions correspond to the physical positions where the sticks of the transmitter should be in order to generate the desired output. These are normalized on a scale of 0 to 100; 0 corresponding to one extreme position of stick and 100 to the other extreme. The convention adopted is that the increase in the value of stick position should lead to increase in thrust or yaw, pitch, roll attitudes of the vehicle, as shown in figure 5.10

The stick positions discussed above may or may not have a one-to-one relation with servo channel pulse-widths, as explained in section 2.2.2. The distinction between

servo positions and stick positions is important. The idea is that the controller (or human pilot) is interested in issuing generic lateral cyclic, longitudinal cyclic, collective and anti-torque commands. The vehicle dynamics or controller design does not depend on how these commands are conveyed to the vehicle or how the servos are actually connected to the swashplate. Therefore, the processing module essentially remains the same even when different MAVs or transmitters are used. These effects are isolated in the form of transmitter module. The transmitter module takes care of the variations in the way these commands are actually implemented on a certain design of an MAV.

5.4.3 The Transmitter Module

The transmitter module sub-VI converts the stick positions received from the processing module into actual commands for different servo channels and writes it to the transmitter via the microcontroller connected to the serial port (section 5.3.2). This sub-VI also reads the pilot inputs (the servo positions from the transmitter output), translates them into stick positions and communicates these back to the processing module. The transmitter is thus used both as an input and an output device.

Once the transmitter is transmitting information received on the data port through the microcontroller, it (obviously) stops transmitting the stick positions directly. Very often, it is required that the pilot still control one or more channels manually, when selective channel autonomy is implemented (semi-autonomous vehicle). For example, in our experiments, the throttle stick position is controlled manually by the pilot and is not autonomously controlled because of lack of altitude sensor. This is implemented by reading

the throttle stick position from the transmitter and sending that data back to the transmitter as output by the processing module. The same concept is also used for implementing the 'throttle curve', limits on motion of the cyclic servos, and to receive attitude commands from the pilot (via the cyclic sticks) in the attitude follow mode of control.

5.5 Implementation of the VI

The sensor data is received and processed (complementary filter) at a much higher sampling frequency (230 Hz) than the output frequency or the servo bandwidth (50 Hz). Two parallel loops are therefore used in the implementation of the VI (fig. 5.11); one for reading sensor data and state estimation, and the second for PID controller and RC transmitter interface. While the first loop is a 'while loop' running at the maximum possible rate, determined by the sampling rate of sensor data, the second loop is a 'timed loop' and runs once in every 20 ms as the servo update is required only at that rate. This makes sure that the processing resources are optimally utilized. These two loops are analogous to the state equation and the output equation in the state-space form of control. It is therefore appropriate to refer to the first loop as the 'state estimation loop' and to the second as the 'output loop'.

The code can thus be divided into 3 sections, i.e., 1) tasks done outside both these loops for initialization, 2) the tasks done under the first loop, and 3) the tasks done under the second loop.

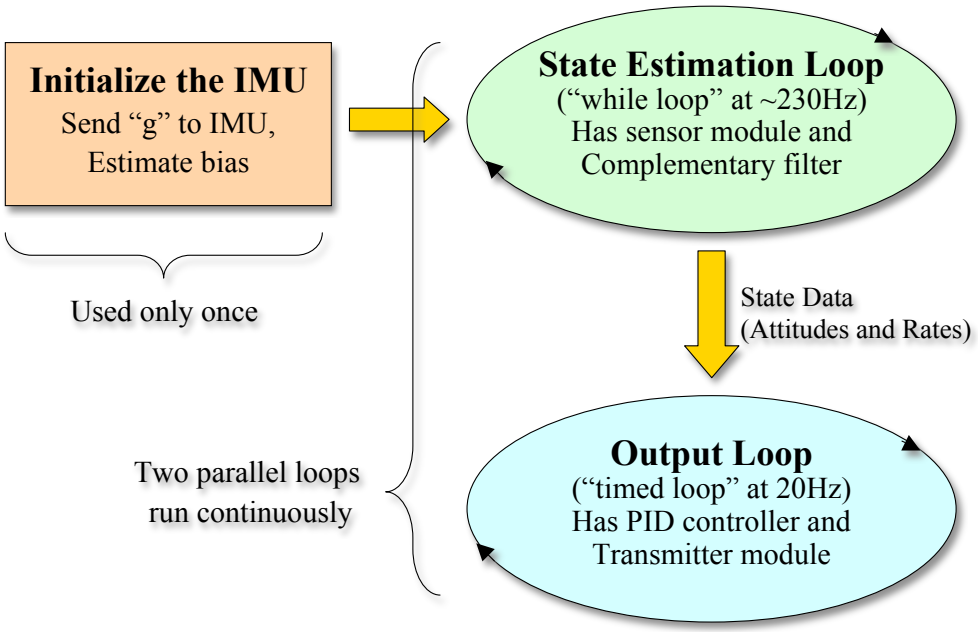

Figure 5.11: The implementation of the VI

5.5.1 Initializing the IMU

The first task of the VI is to send the command to the IMU to start sending data. The ASCII character "g" is sent to the IMU, as specified in the IMU datasheet [38]. The data from 6 sensors is now made available by the sensor module. The MEMS sensors are quite sensitive to temperature. It is therefore important to wait for the temperature of sensors to stabilize before bias estimation. Every time the IMU is powered on, the analog-to-digital convertors, and other electronics need some time to stabilize. The first 5000 data packets received are thus discarded.

Accurate determination of gyroscope bias (the gyro output for zero angular rate) is critical, as any error in this will lead to large drift in the estimation of attitude over long time (without filtering). The bias of a gyroscope is not constant between different runs. Therefore, the bias needs to be estimated every time before the experiment. The IMU is

kept stationary and the next 5000 samples from each of the 3 gyros are averaged to get their biases. Further discussion on gyro bias is presented in the appendix to this report.

5.5.2 State Estimation Loop

State estimation loop is responsible for continuously sampling the IMU, and then filtering the raw sensor data to provide the output loop with required states, viz. the roll, pitch and yaw attitudes. The sensor data is read through the sensor module (described above), and the angular rates and translational accelerations are made available in navigation frame. The angular rates are then converted into inertial frame (section 4.1) and the accelerations into roll and pitch attitudes (section 4.3). These are finally processed by the complementary filter (section 4.4) to provide the attitudes.

5.5.3 Output Loop (PID controller)

The output loop uses the transmitter module to read the pilot inputs (via the sticks). The attitudes and body rates are made available by the state estimation loop. The pilot inputs and attitudes are then used along with the user supplied gains to generate output based on the selected mode, as shown in figure 5.12. The 3 modes of operation are as follows:

1. Loop-back mode: In this mode all the data received through the transmitter module (pilot stick positions) is directly wired back to the transmitter module. It is useful to check if the transmitter interface is working correctly.

2. Attitude hold mode: This mode uses the gains supplied by the user to generate the cyclic and tail-rotor commands in order to keep the attitudes close to zero (PID

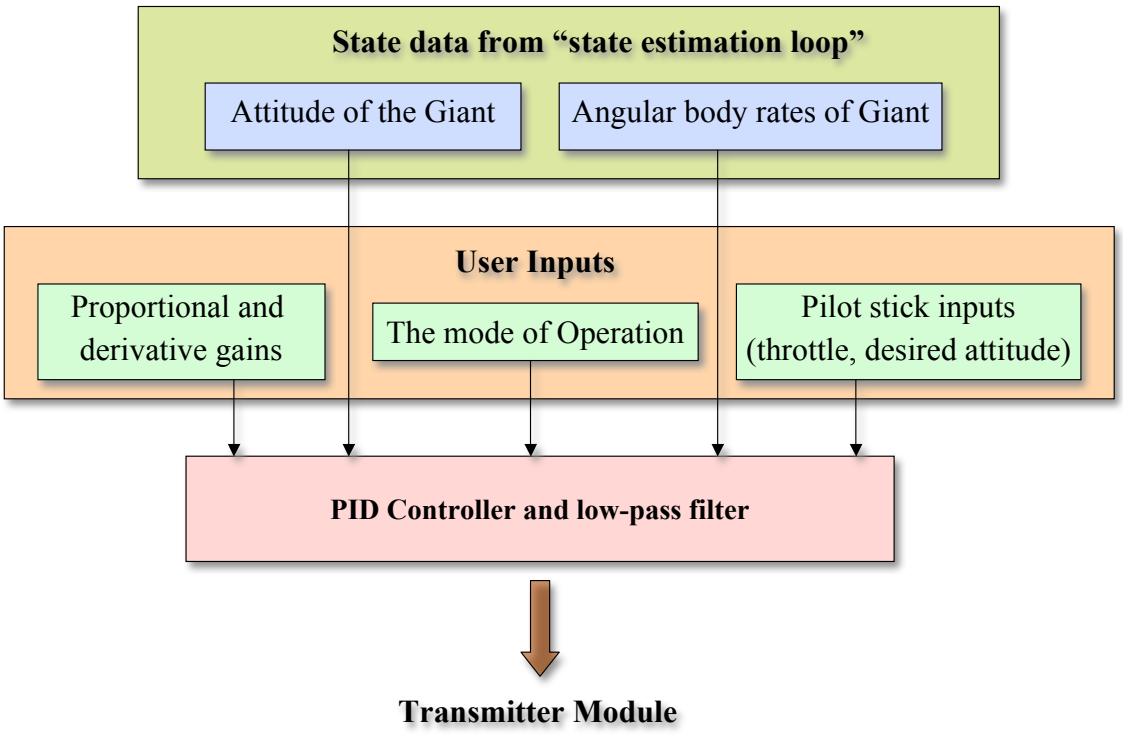

Figure 5.12: The output loop

controller). The Giant is thus expected to hover in this mode.

3. Attitude control mode: It is similar to the attitude hold mode, except that the pilot can now demand roll and pitch attitudes through the cyclic sticks on the transmitter. The controller will make the Giant follow those attitude commands.

The gains can be adjusted by the user anytime during the experiment. The servos have a limited bandwidth. The frequency of the output signal should be below 5Hz for effective working of servos. A low pass filter is therefore incorporated in the output of PID controller.

Chapter 6

Results and Discussion

6.1 Vibrations and Sensor Saturation

The IMU is mounted on the Giant as shown in figure 6.1. During the initial test flights, the vibrations were found to lead to saturation of gyroscopes aligned with the body x and y axes as shown in figure 6.2. The reliability of attitude estimation is severely affected due to saturation of sensors. The vibrations are isolated from the IMU by proving a padding between the LiPo battery and the IMU. Different blade geometries are also seen to affect the vibration level. Typical gyro data with no saturation is shown in figure 6.3

6.2 Manual Flight

In order to fly the Giant with PID controller, it is first required to get an estimate of the gains required. In order to gain confidence in the system and obtain an estimate of gains required, a manual instrumented hover flight is conducted. The data from the IMU and

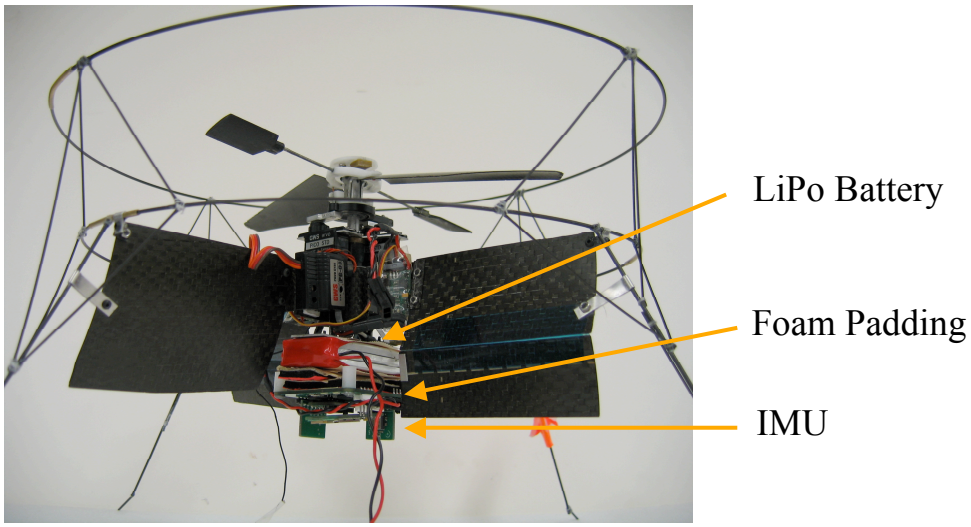

Figure 6.1: IMU mounted on the Giant with foam padding

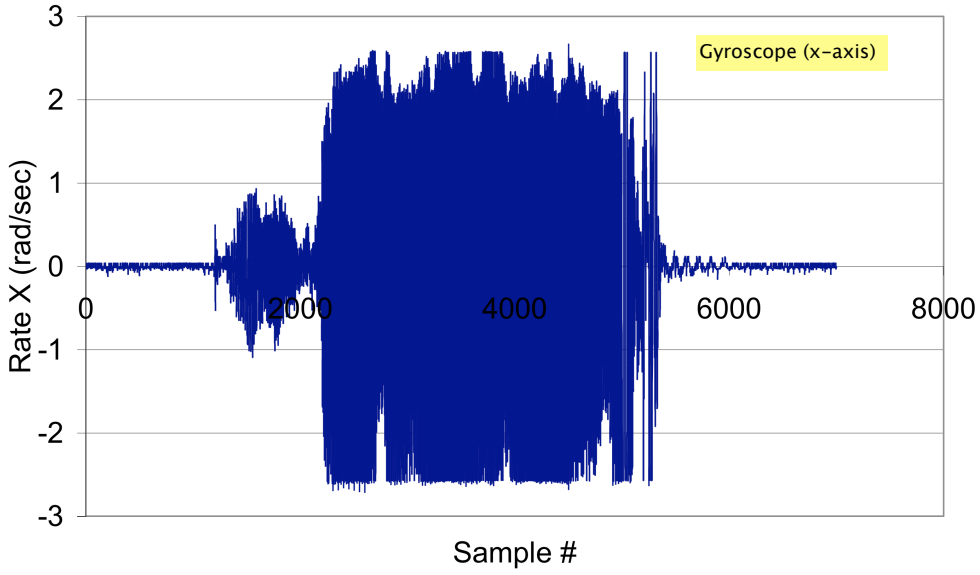

Figure 6.2: Angular rate about body x-axis during flight (gyro saturated)

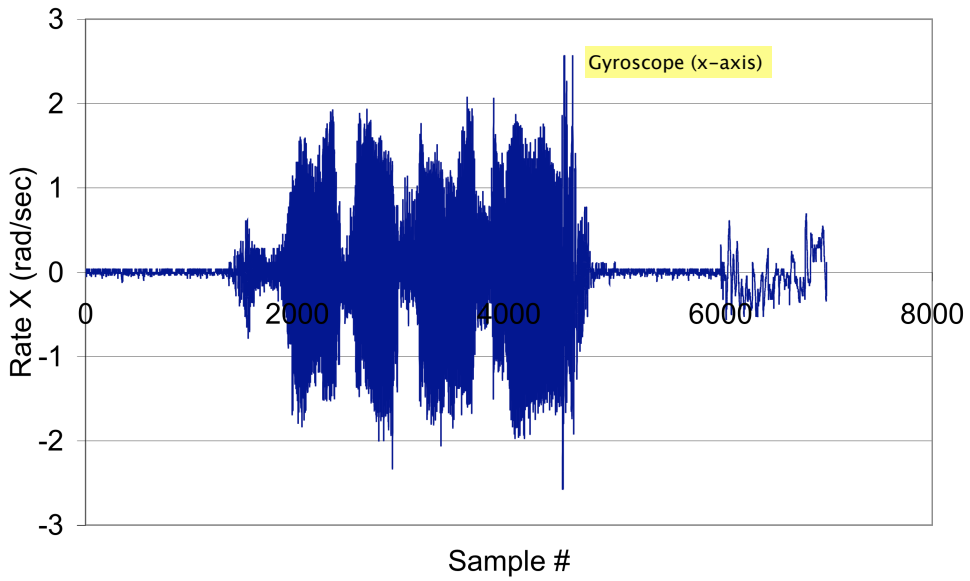

Figure 6.3: Angular rate about body x-axis during flight (gyro not saturated)

the pilot stick inputs are recorded during the flight. Figure 6.4 shows the pitch attitude as estimated by the accelerometers, gyroscopes and the complementary filter. We see that the attitude is maintained within $\pm$ 0.1 radian. The time history of pilot longitudinal cyclic inputs during this flight is shown in figure 6.5.

From this data, it can be seen that the pilot effectively uses a gain of around 20% to 25% stick range per radian. It can also be seen that cyclic is applied at a maximum frequency of about 2Hz by the pilot. Based on the complementary filter data, output proportional to the attitude, with a gain of 25% stick range per radian is computed and plotted in figure 6.6. A low-pass filter is also incorporated in the output of controller so that the servos are actuated at a frequency similar to that used by a human pilot.

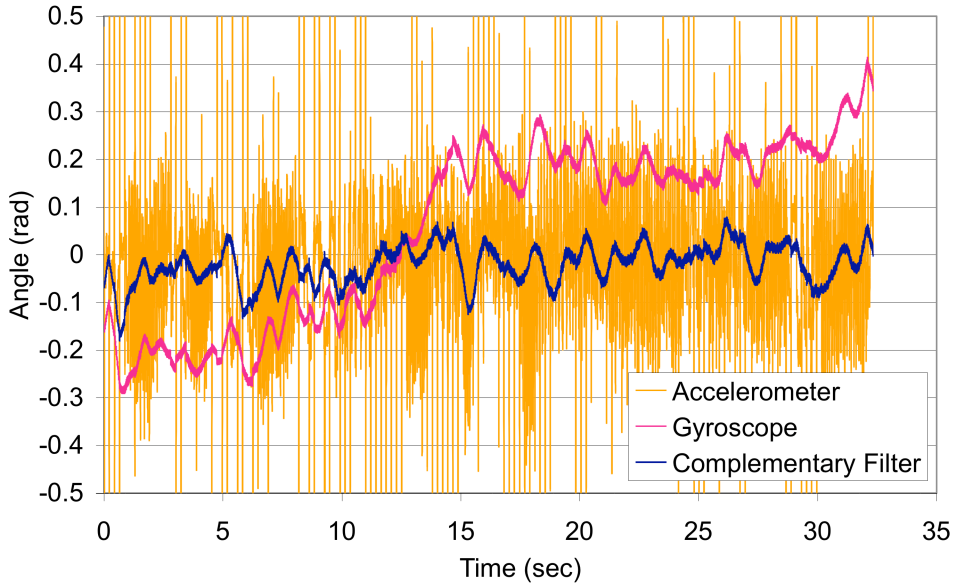

Figure 6.4: Pitch attitude estimate during manual flight

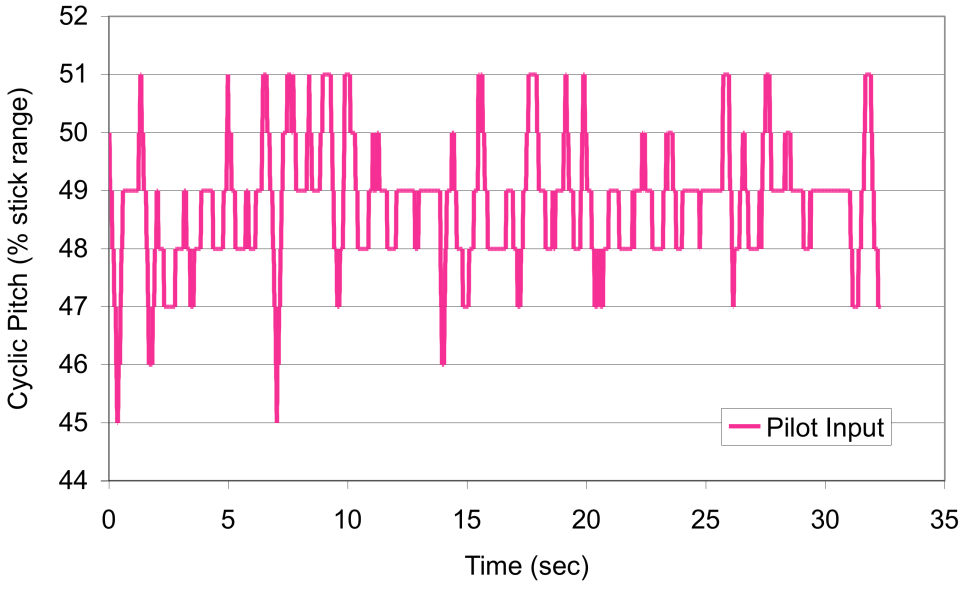

Figure 6.5: Longitudinal cyclic during manual flight

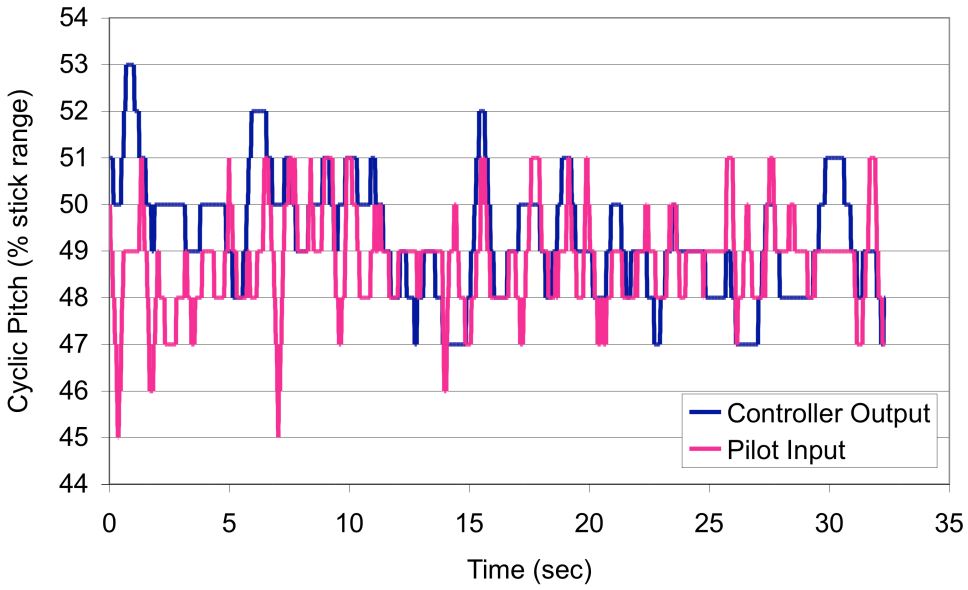

Figure 6.6: Comparison of computed longitudinal cyclic with actual pilot command

6.3 Controlled Flight Results

The results for the controlled hovering flight of the Giant are presented next. The roll, pitch and yaw of the vehicle are controlled by the autonomous controller. Throttle is the only input provided by the pilot. A dual slope is used on the throttle curve to reduce the sensitivity near the hovering RPM of rotor. Once the throttle exceeds 60% stick range, a slope of 0.4 is used in place of unity, to enable finer control of rotor RPM. As can been seen in figure 6.7, both the pilot-input and the controller-output match when throttle is below 60% stick range, but the controller output increases by only 40% of pilot-input beyond that.

Yaw control is provided by both the onboard (off-the-shelf) gyro as well as the control system. The off-the-shelf gyro provides the basic derivative control and it is assisted by a proportional control implemented in LabVIEW. The off-the-shelf gyro is left

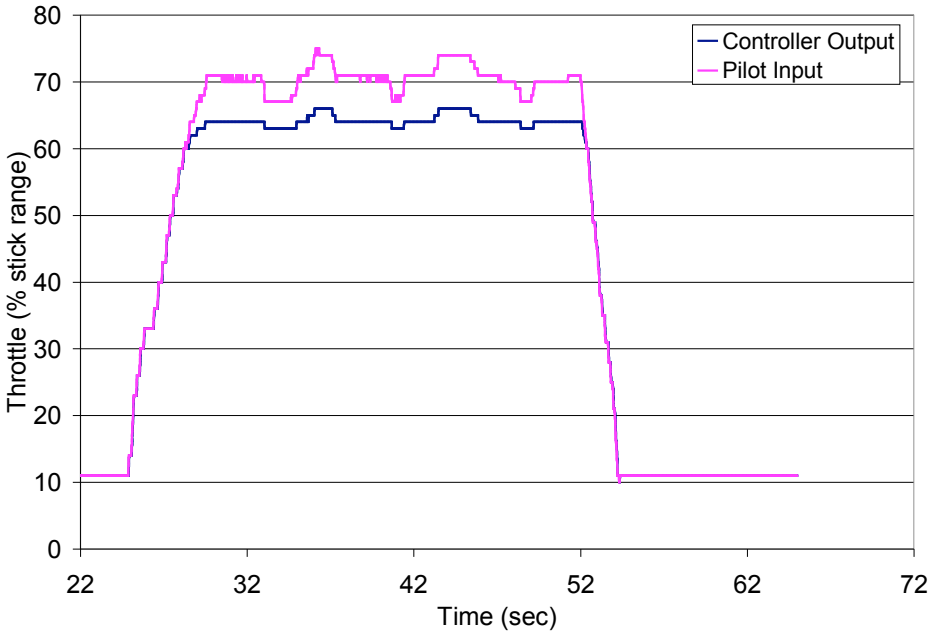

Figure 6.7: Dual-curve in throttle to reduce sensitivity near hover RPM

in place, because in case the flight mode is switched to manual (or if there is a failure of autonomous control), it would not be possible to control the Giant without some level of yaw stabilization. A fixed gain of 30% stick range per radian is used for the yaw axis for all experiments. It is important to note that from the Euler-angle transformation point of view, the yaw attitude is not important as it does not affect the roll or pitch attitudes (this is not true for roll and pitch though; roll affects both pitch and yaw, and pitch affects yaw (see equation 4.6)). However, dynamics in yaw may still influence the roll and pitch motion due to the presence of coupling terms in the dynamics of the Giant.

We follow the Ziegler-Nichols method of tuning the PID controller [40]. Ziegler and Nichols suggested rules for tuning PID controllers base on experimental step response or based on the value of proportional gain that results in marginal stability when only proportional control action is used. Ziegler-Nichols tuning rules give an educated guess

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Table 6.1: Ziegler-Nichols Tuning Method

for the gain values and provide a starting point for fine tuning, rather than give the final settings for the gains in a single shot. We use the second Ziegler-Nichols method, i.e., the one based on the value of proportional gain that results in marginal stability, for tuning the PID controller for autonomous flight of Giant [40].

6.3.1 Ziegler-Nichols PID Tuning

According to the Ziegler-Nichols second method of PID gain tuning [40], we we increase the proportional gain, K_p, upto a critical value K_{cr} at which the output exhibits sustained oscillations or marginal stability. Once the critical gain K_{cr} and the corresponding period P_{cr} are experimentally determined, Ziegler and Nichols suggested that we set the values of the proportional gain K_p, integral time T_i, and derivative time T_d according to the formula given in table 6.1.

The resulting controller, $G_c(s)$, in s-domain is given by:

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{6.1}$$

In terms of gains, the resulting controller action (in time domain) is:

$$G_c(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt} \tag{6.2}$$

where, $e(t)$ is the error signal given by:

$$e(t) = set\,point - measurement(t) \tag{6.3}$$

We therefore see that the integral gain, K_i, and derivative gain, K_d, are related to the integral time, T_i, and derivative time, T_d, as follows:

$$K_i = \frac{K_p}{T_i} \tag{6.4}$$

$$K_d = K_p T_d \tag{6.5}$$

Determining Critical Gain and Critical Period

As required for the Ziegler-Nichols gain tuning method, we first conduct experiments to determine the critical gain and the corresponding critical period. As mentioned before, we are primarily interested in tuning the roll and pitch loops. For simplicity, the yaw gain is kept fixed for all experiments. The critical gains for both the roll and pitch axes is expected to be different because of the non-symmetric servo-swashplate assembly, i.e., the two swashplate servos need to be deflected by different amounts to get equal longitudinal and lateral cyclics. We apply the Ziegler-Nichols method to this two axes system as follows. The idea is to keep the system well-tuned in one axis at a time, and marginally stable in the other. This way, the first axis will have minimal influence during determination of critical gain and period of the other axis. The same process is then repeated with the axes swapped, in order to determine the other set of critical gain and period.

We first determine approximate gains for which the vehicle is stable in both axes. We then keep one gain fixed at this value and increase the other till it becomes marginally

unstable. This value of second gain gives us a first approximation of critical gain for that axis. We now fix the second gain to half of this value and change the first gain (which was kept fixed earlier) till the vehicle again becomes marginally unstable to get approximation of critical gain for first axis. The second gain is reduced to *half* of critical value because according to table 6.1, half of critical gain is expected to give a well tuned system. We now iterate till consistent results are obtained, i.e., the system becomes marginally unstable when we use critical gain for one axis and half of critical gain for the other. We thus obtain critical gains for both the axes that can be used to tune the proportional (P), proportional-integral (PI) and proportional-integral-derivate (PID) controllers.

From the analysis of manual flight (section 6.2), we expect the vehicle to be stable for proportional gains of around 25% stick range per radian for both roll and pitch axes. The pitch and roll attitudes during the autonomous flight for these values of gains are shown in figures 6.8 and 6.9, respectively. The corresponding control inputs are shown in figures 6.10 and 6.11, respectively. Figures 6.12 and 6.13 show the roll attitude and corresponding controller inputs, respectively. The large peaks in all these plots at $T \approx 48$ seconds corresponds ot the landing of the vehicle. The vehicle can be seen to diverge much more in yaw than roll or pitch. This is because of relatively much lower inertia in yaw. As mentioned before, for simplicity, a fixed gain was used in yaw as any errors in yaw do not affect pitch and roll estimation.

We now increase the pitch gain. The system is found to reach the critical condition for a gain of 36% stick range per radian in pitch (the roll gain is kept fixed at 25% stick range per radian). The pitch and roll attitudes for this case are shown in figures 6.14 and 6.15, respectively. Although only the pitch gain was increased, the instability in pitch

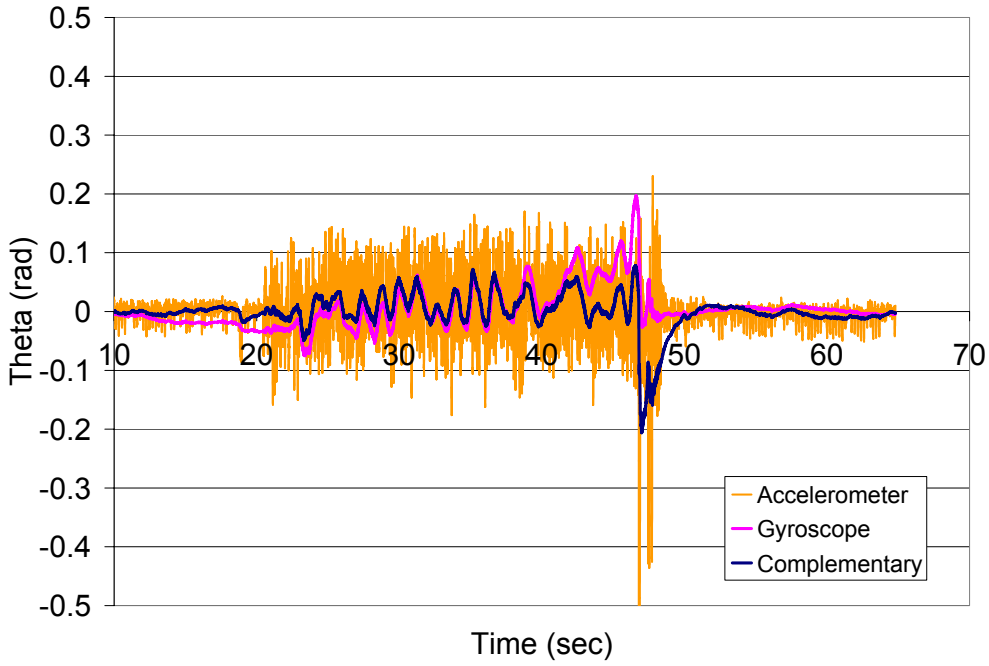

Figure 6.8: Pitch attitude vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

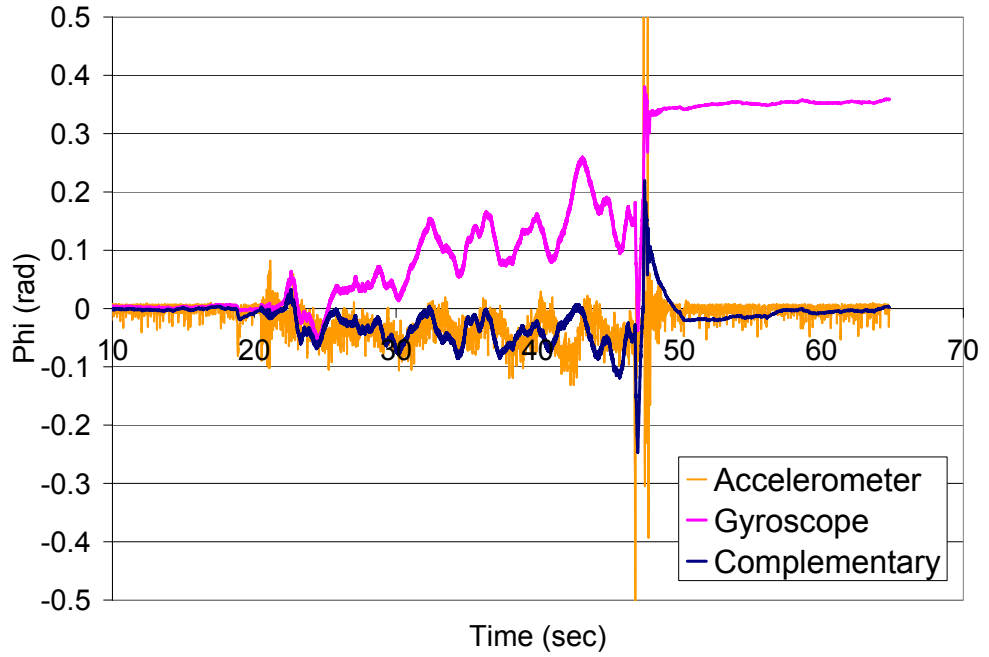

Figure 6.9: Roll attitude vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

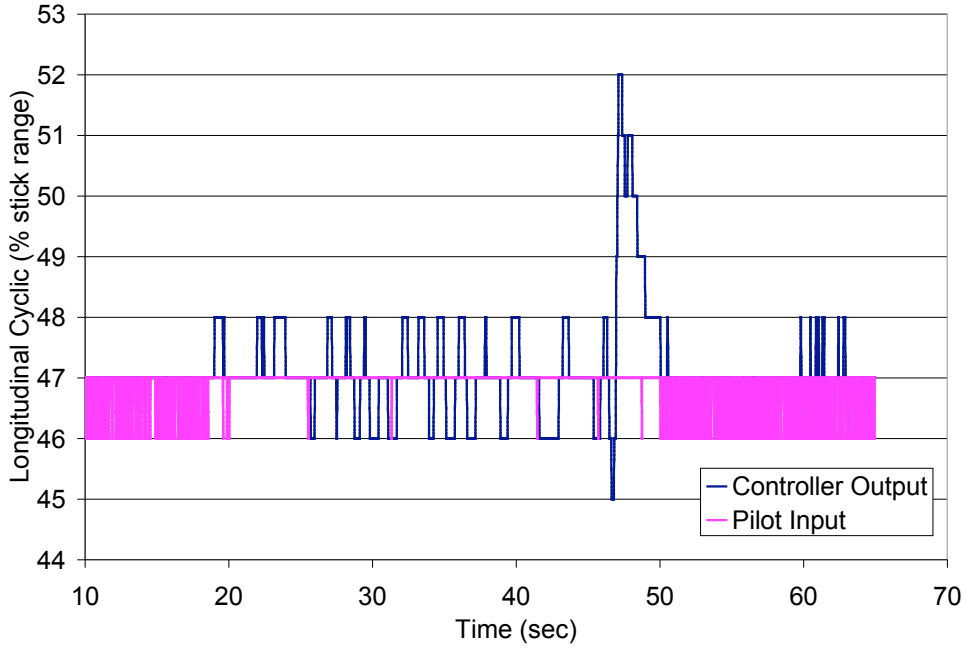

Figure 6.10: Longitudinal cyclic vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

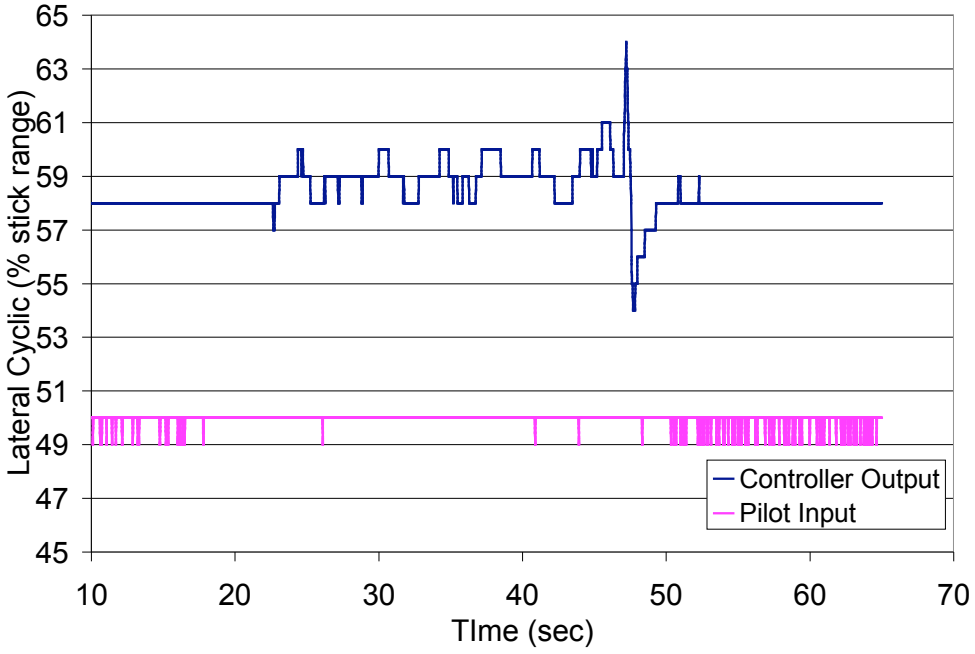

Figure 6.11: Lateral cyclic vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

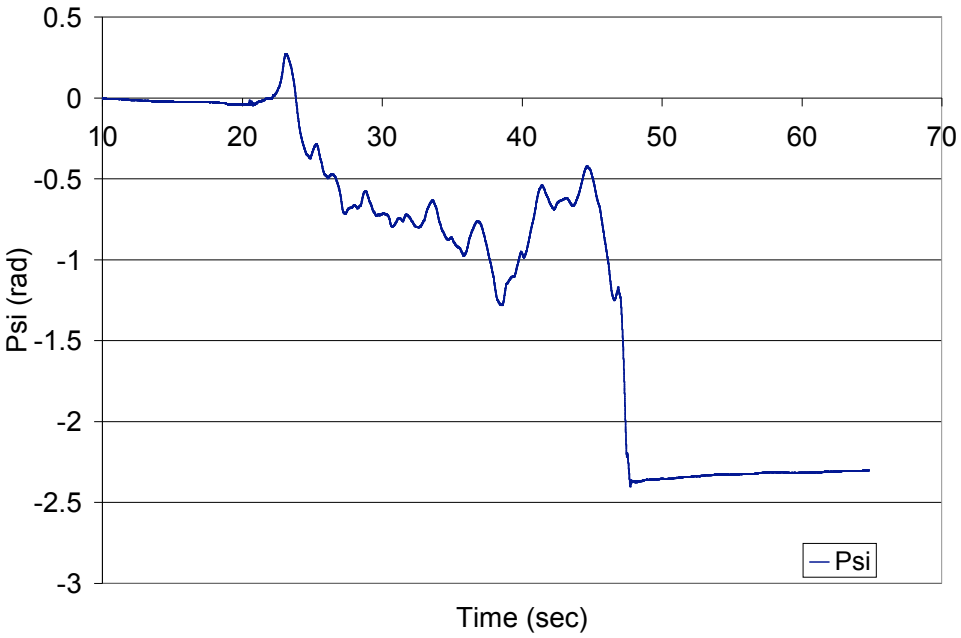

Figure 6.12: Yaw attitude vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

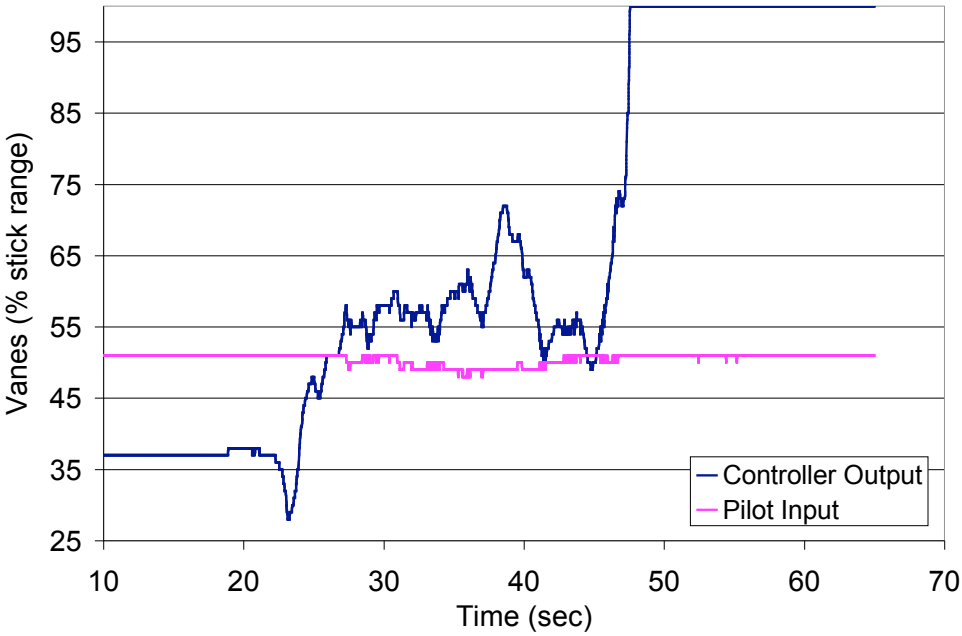

Figure 6.13: Vane command vs. time ($K_{P\theta} = 25, K_{P\phi} = 25, K_{P\psi} = 30$)

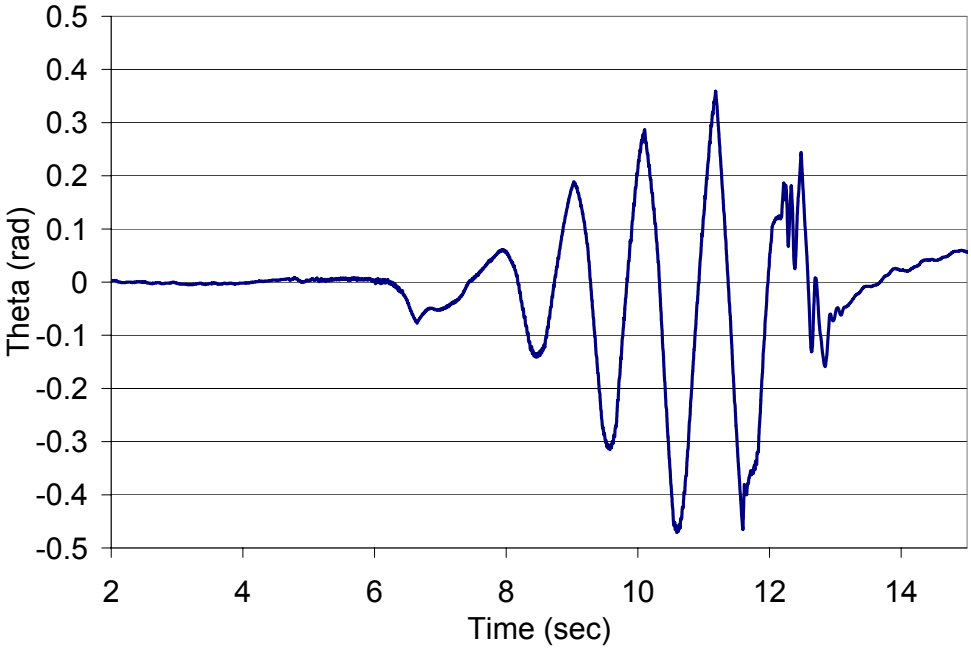

Figure 6.14: Pitch attitude vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

is reflected also in roll due to coupling. The corresponding cyclic inputs are shown in figures 6.16 and 6.17. The vane control inputs and yaw attitude are shown in figures 6.18 and 6.19, respectively.

We now reduce the proportional gain in pitch to 18% stick range per radian (half of estimated critical value) and increase the roll gain till the system becomes marginally stable. At a gain of 130% stick range per radian in roll, the system is found to be unstable as shown in figures 6.20 and 6.21. The gain is then reduced and marginal stability is found at the gain of 124% stick range per radian in roll (with 18% stick range per radian in pitch). The pitch and roll attitudes for this flight test are shown in figures 6.22 and 6.23, respectively. The longitudinal and lateral cyclics provided by the controller for same flight are shown in figures 6.24 and 6.25, respectively. The yaw attitude and the vane input provided by the controller are shown in figures 6.26 and 6.27, respectively.

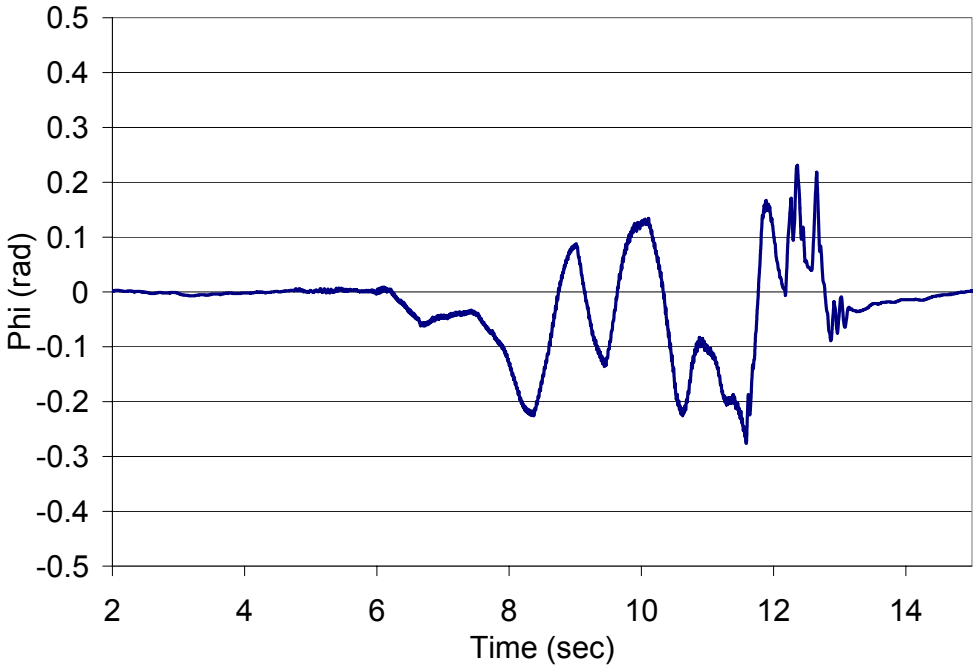

Figure 6.15: Roll attitude vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

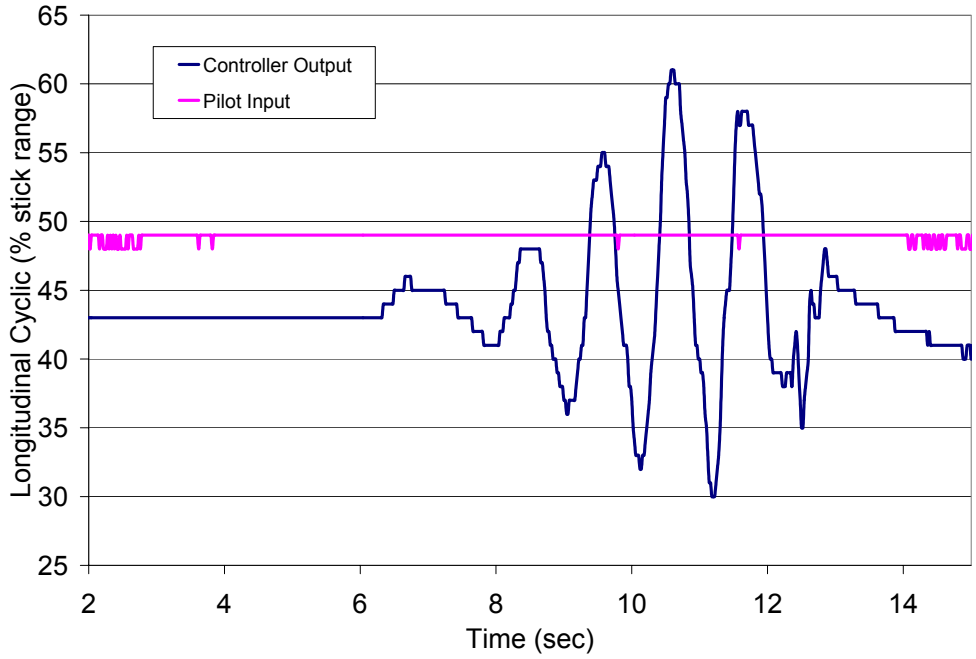

Figure 6.16: Longitudinal cyclic vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

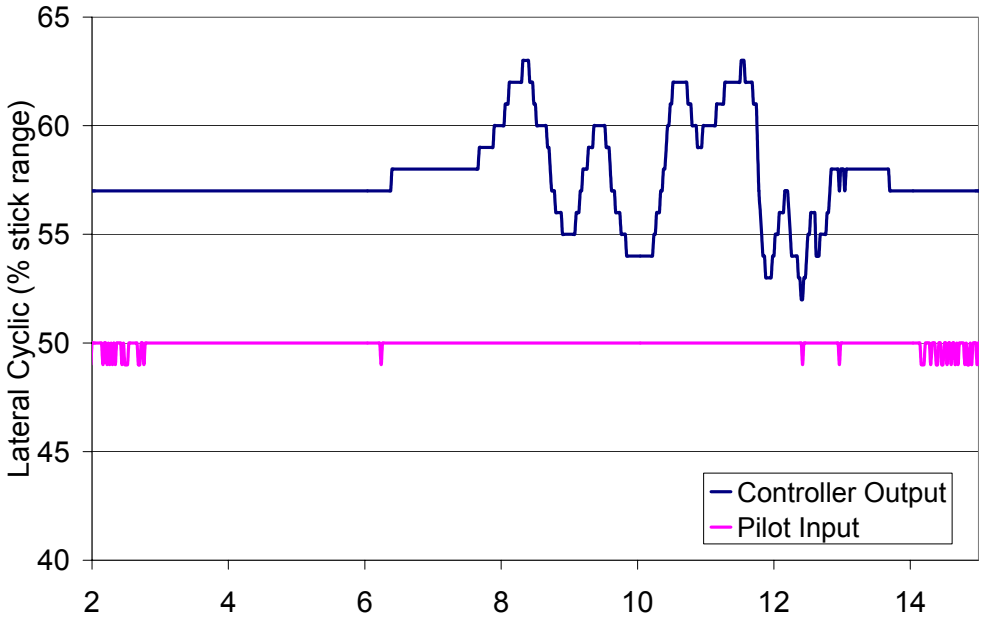

Figure 6.17: Lateral cyclic vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

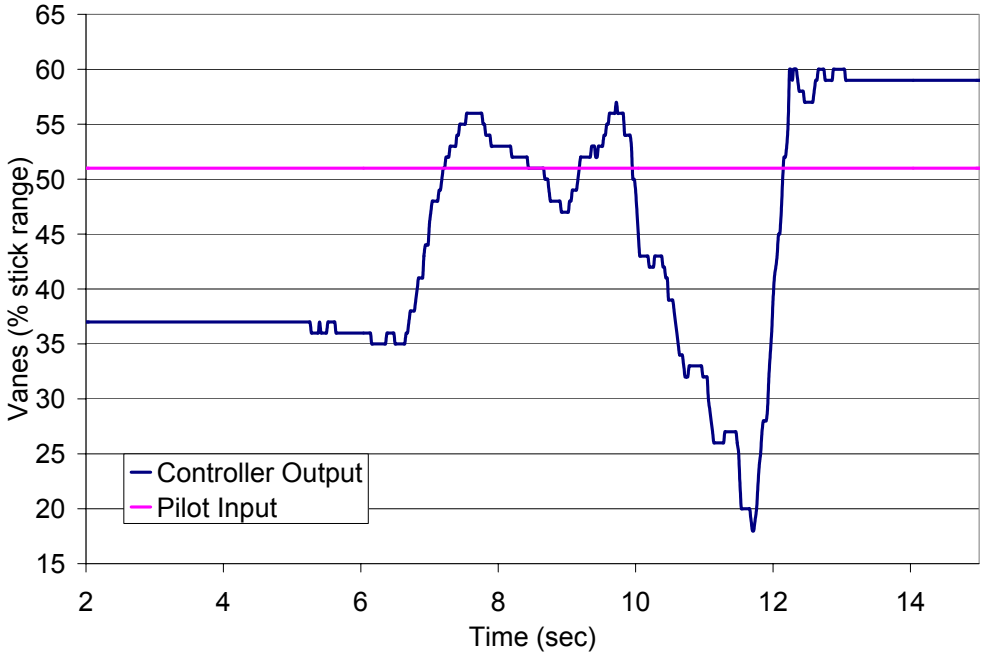

Figure 6.18: Vane command vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

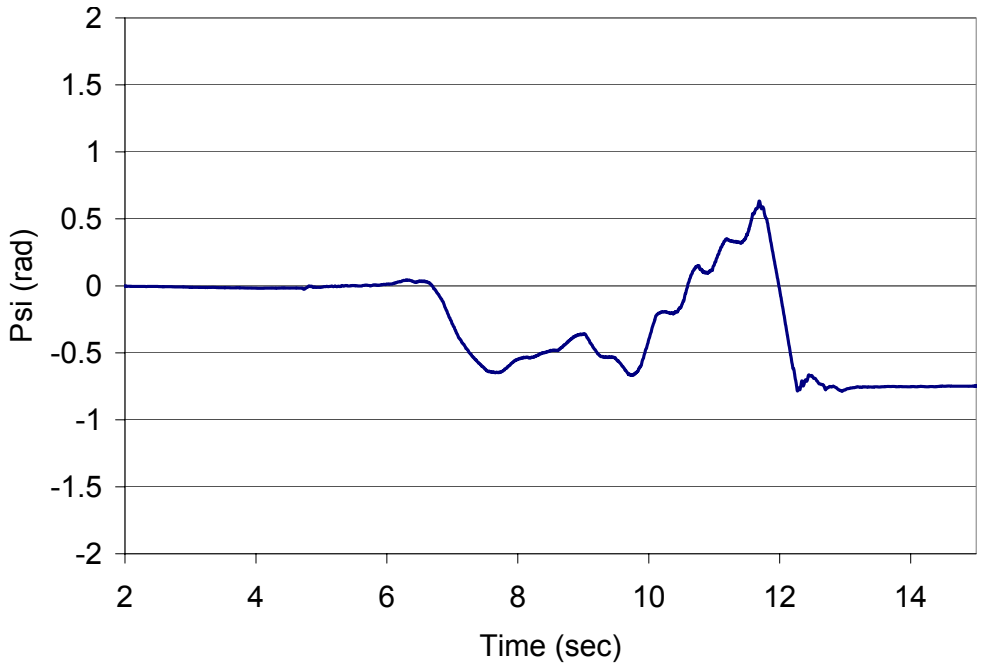

Figure 6.19: Yaw attitude vs. time ($K_{P\theta} = 36, K_{P\phi} = 25, K_{P\psi} = 30$)

The large spikes towards the end of all these plots corresponds to the touch down of the Giant.

We now check if the system is marginally stable for a pitch gain of 36% stick range per radian and roll gain of 62% stick range per radian. If this is found to be marginally stable, then the estimates of critical gains for the two axes is correct, else we need to iterate. The system was found to be unstable for these gains as shown by the pitch and roll attitudes in figures 6.28 and 6.29, respectively. We now reduce the pitch gain to 32% stick range per radian to get the marginal stability condition. The pitch and roll attitudes for this case are shown in figures 6.30 and 6.31, respectively. The longitudinal and lateral inputs for the same flight are shown in figures 6.32 and 6.33, respectively. Figures 6.34 and 6.35 show the yaw attitude and vane servo commands given by the controller, respectively.

The critical gains and period are thus obtained as shown in table 6.2.

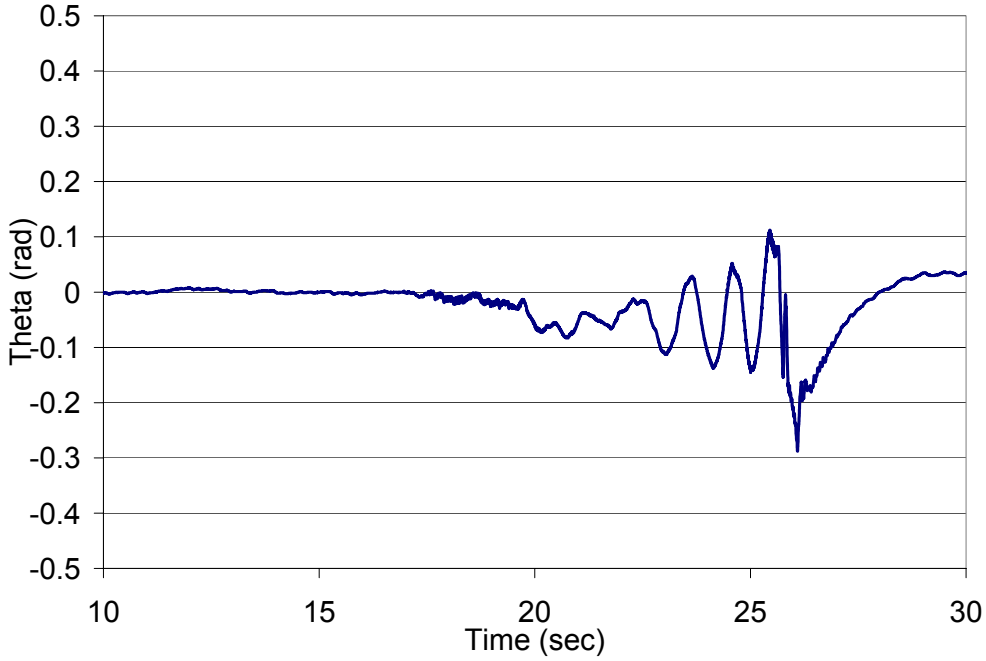

Figure 6.20: Pitch attitude vs. time ($K_{P\theta} = 18, K_{P\phi} = 130, K_{P\psi} = 30$)

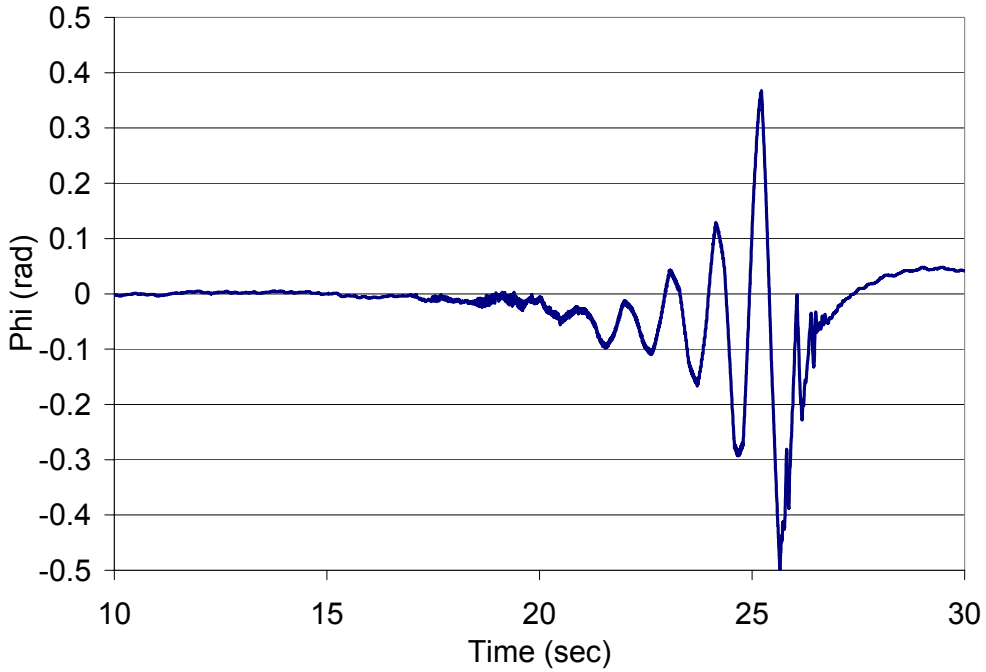

Figure 6.21: Roll attitude vs. time ($K_{P\theta} = 18, K_{P\phi} = 130, K_{P\psi} = 30$)

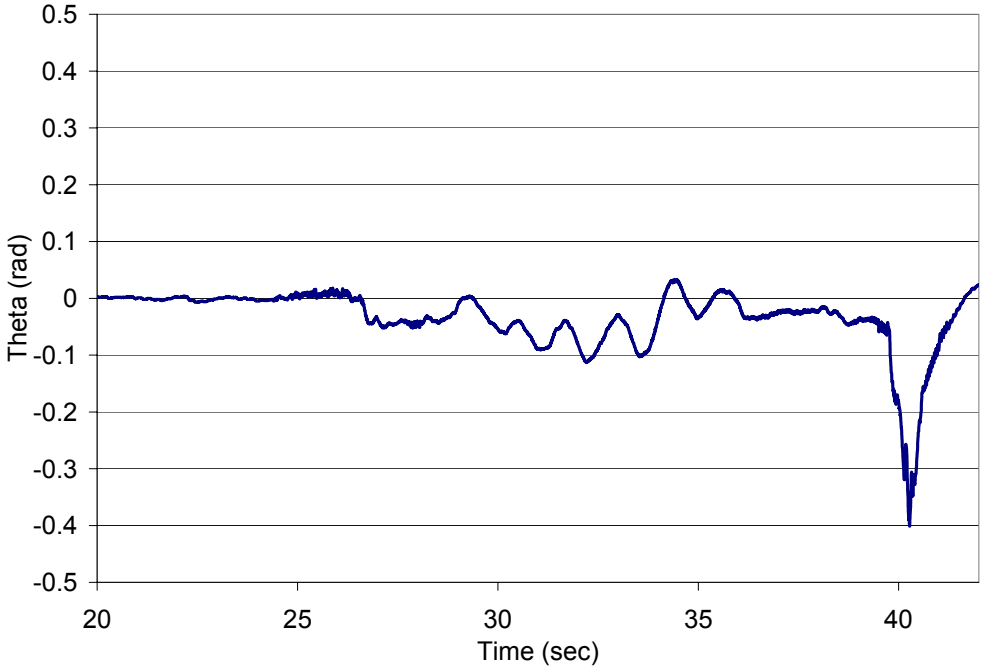

Figure 6.22: Pitch attitude vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

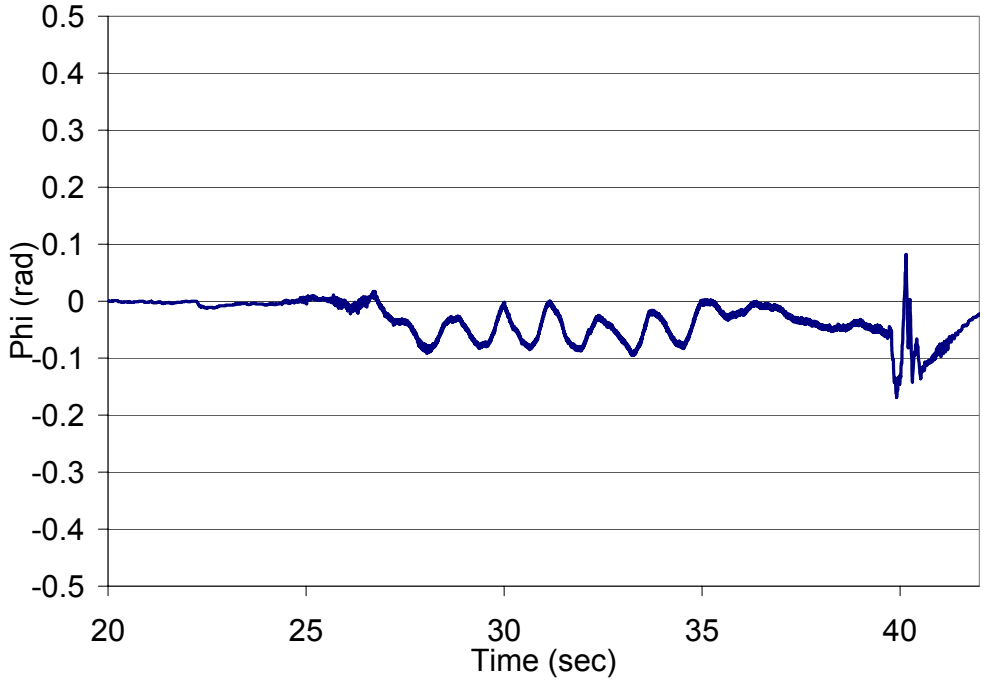

Figure 6.23: Roll attitude vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

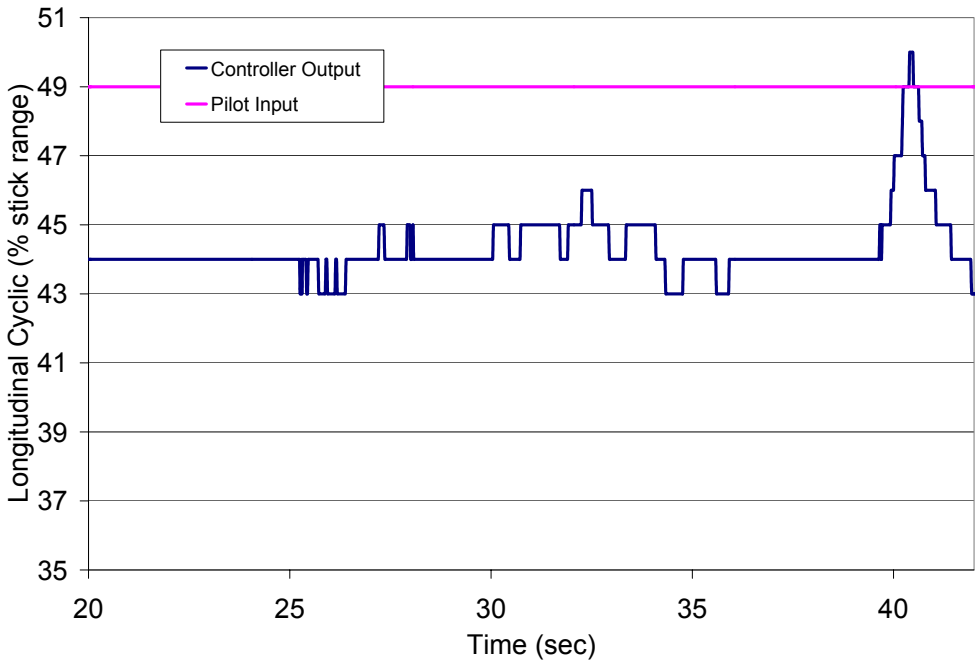

Figure 6.24: Longitudinal cyclic vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

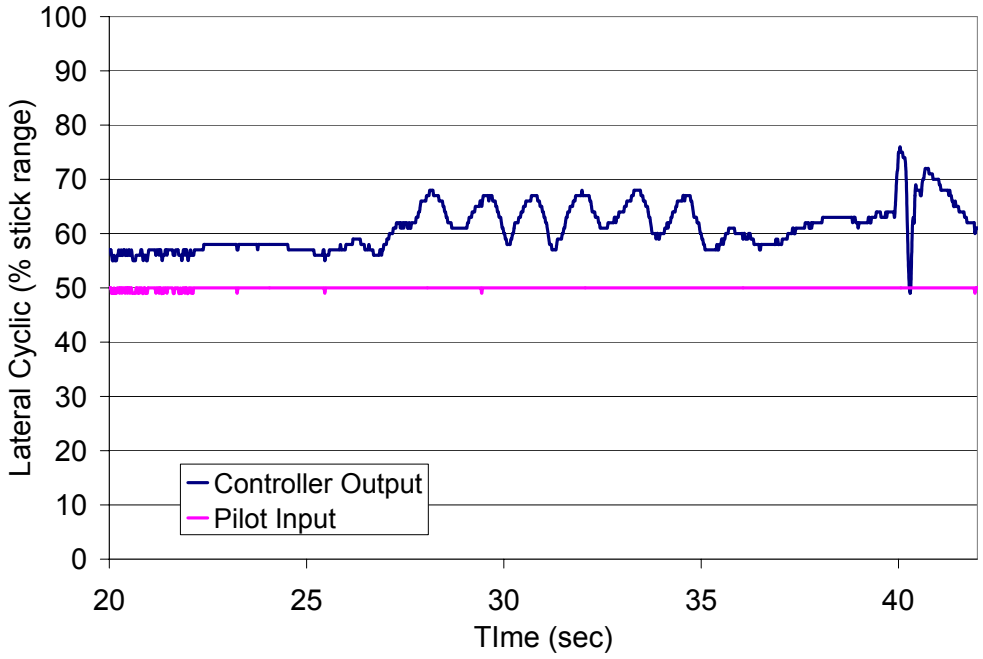

Figure 6.25: Lateral cyclic vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

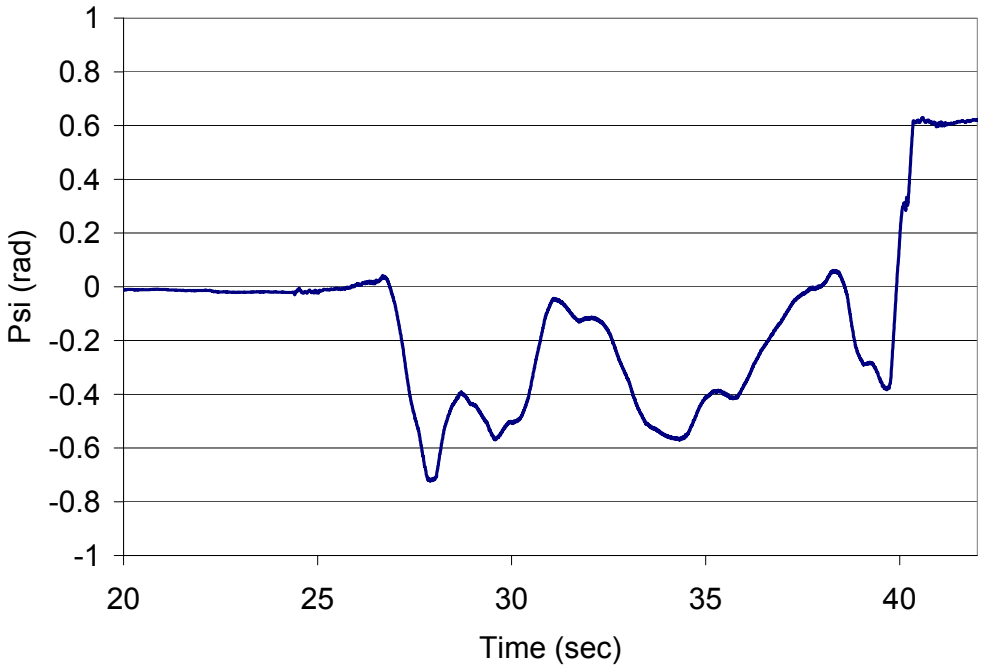

Figure 6.26: Yaw attitude vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

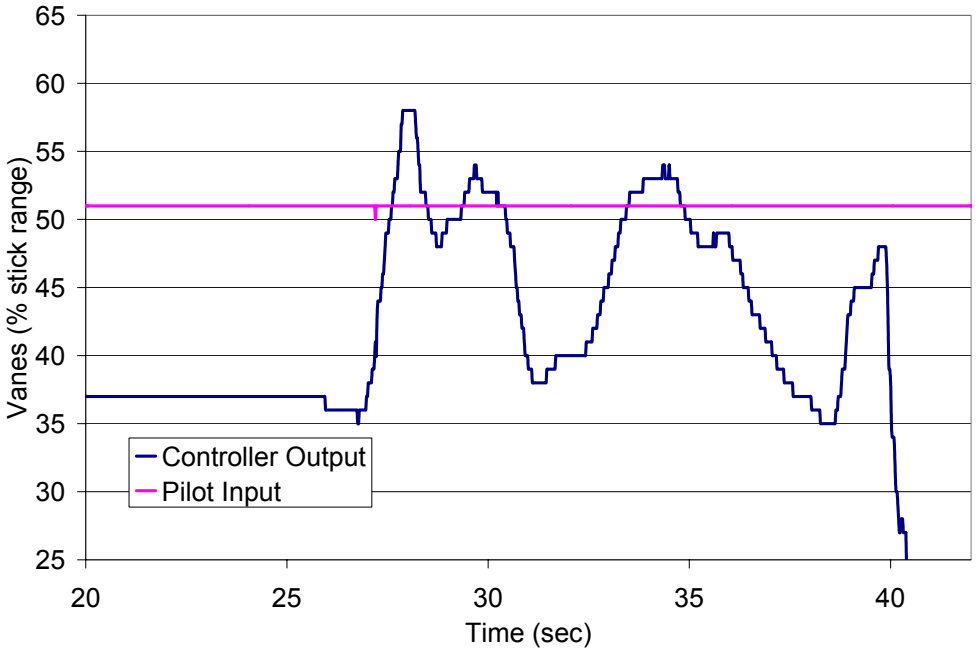

Figure 6.27: Vane command vs. time ($K_{P\theta} = 18, K_{P\phi} = 124, K_{P\psi} = 30$)

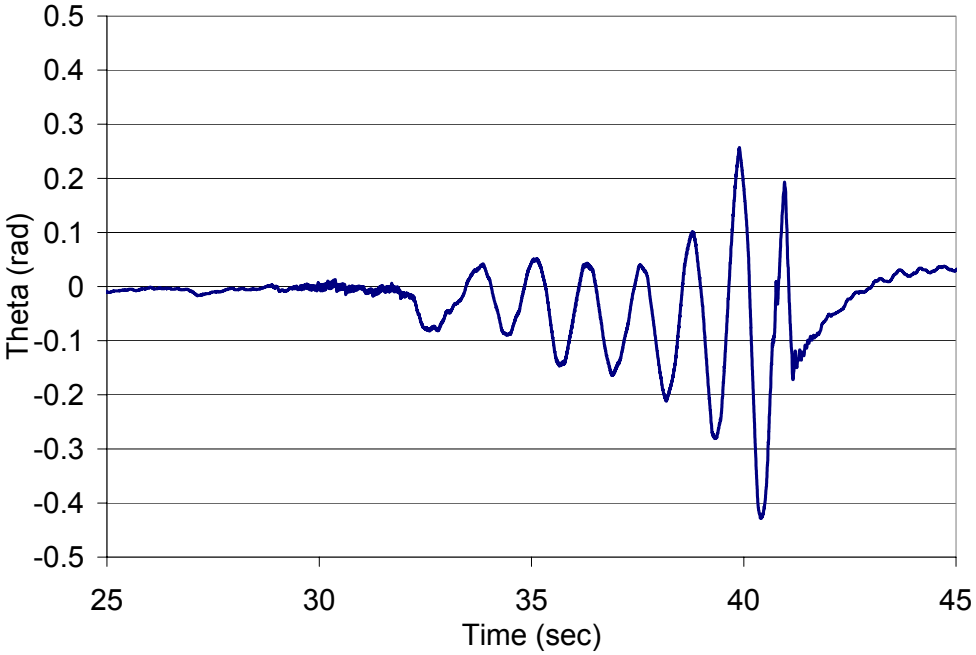

Figure 6.28: Pitch attitude vs. time ($K_{P\theta} = 36, K_{P\phi} = 62, K_{P\psi} = 30$)

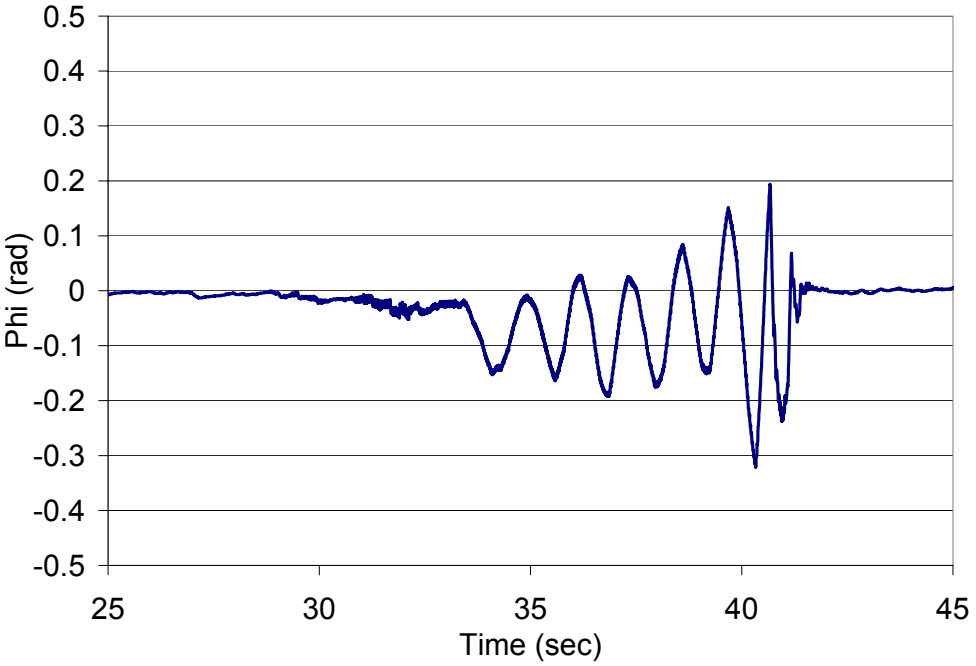

Figure 6.29: Roll attitude vs. time ($K_{P\theta} = 36, K_{P\phi} = 62, K_{P\psi} = 30$)

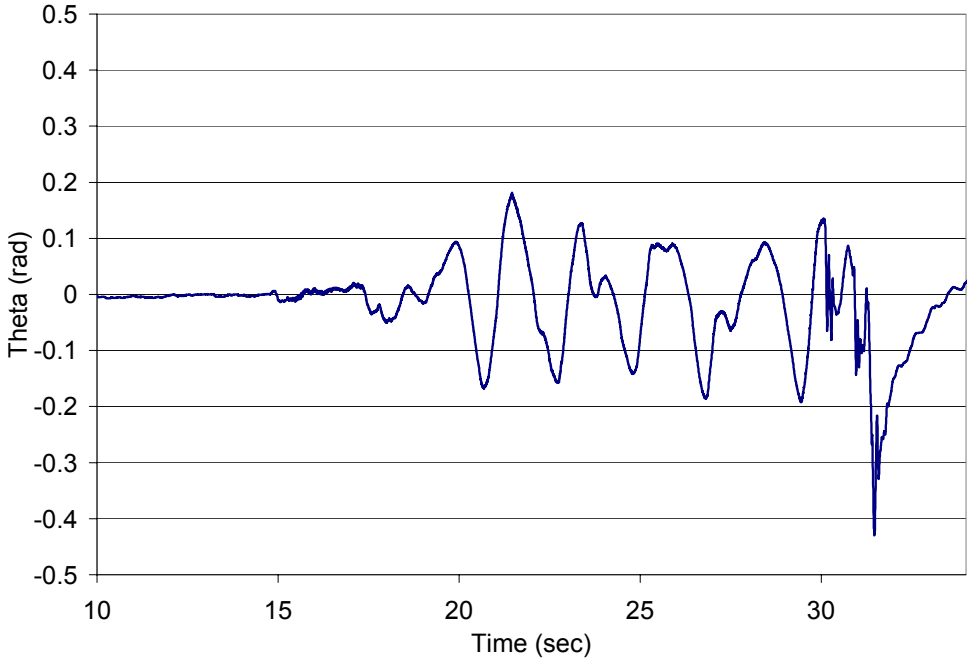

Figure 6.30: Pitch attitude vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

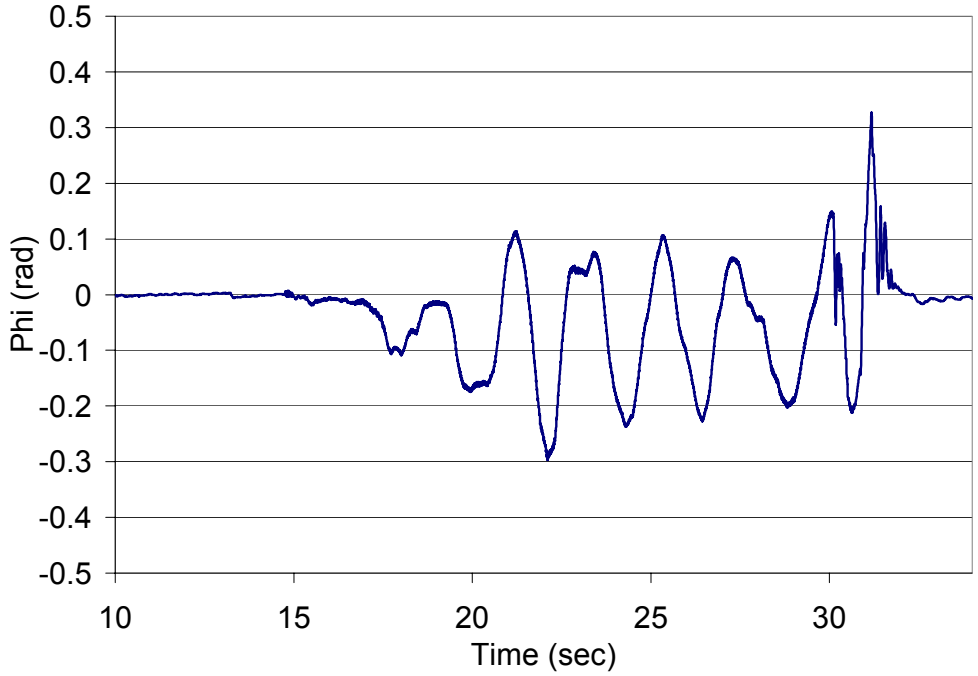

Figure 6.31: Roll attitude vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

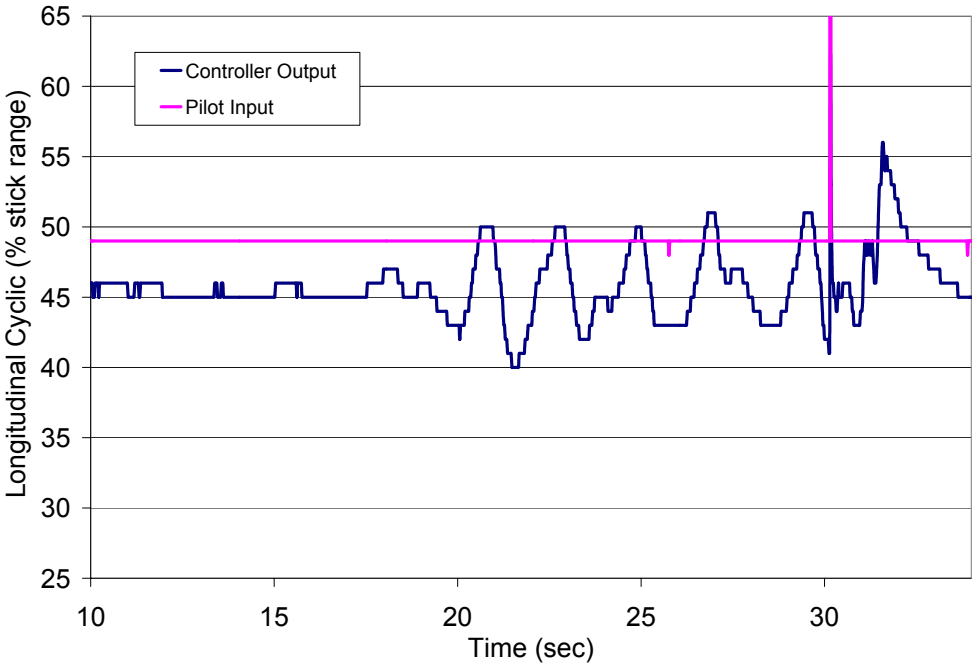

Figure 6.32: Longitudinal cyclic vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

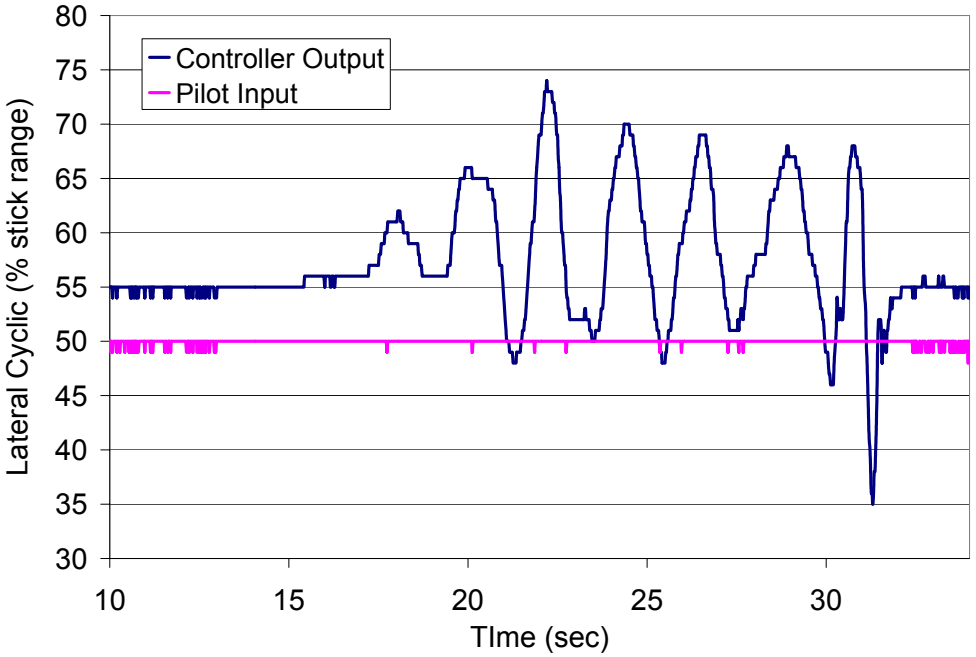

Figure 6.33: Lateral cyclic vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

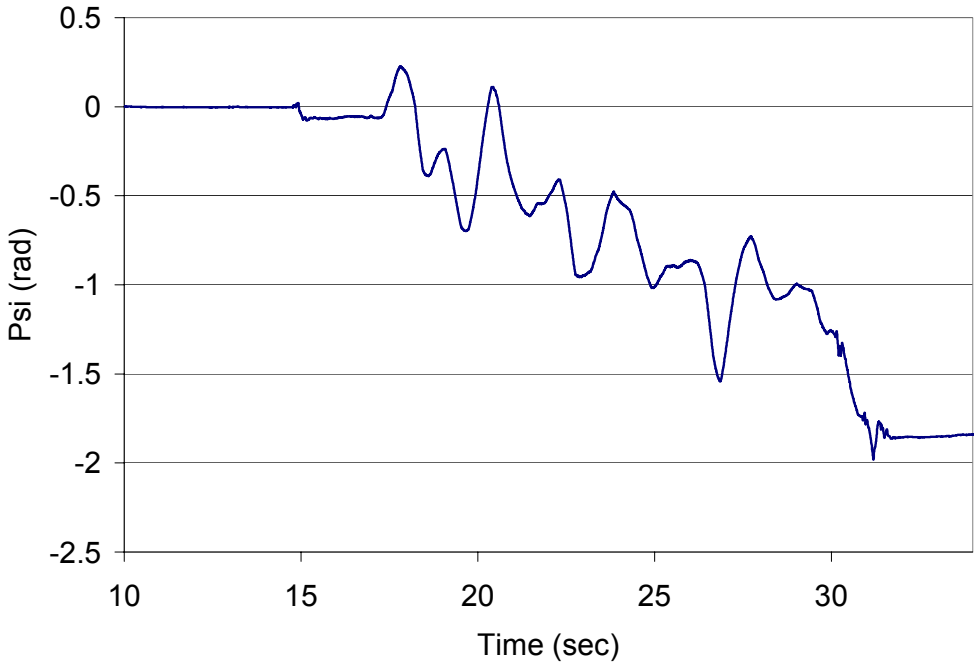

Figure 6.34: Yaw attitude vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

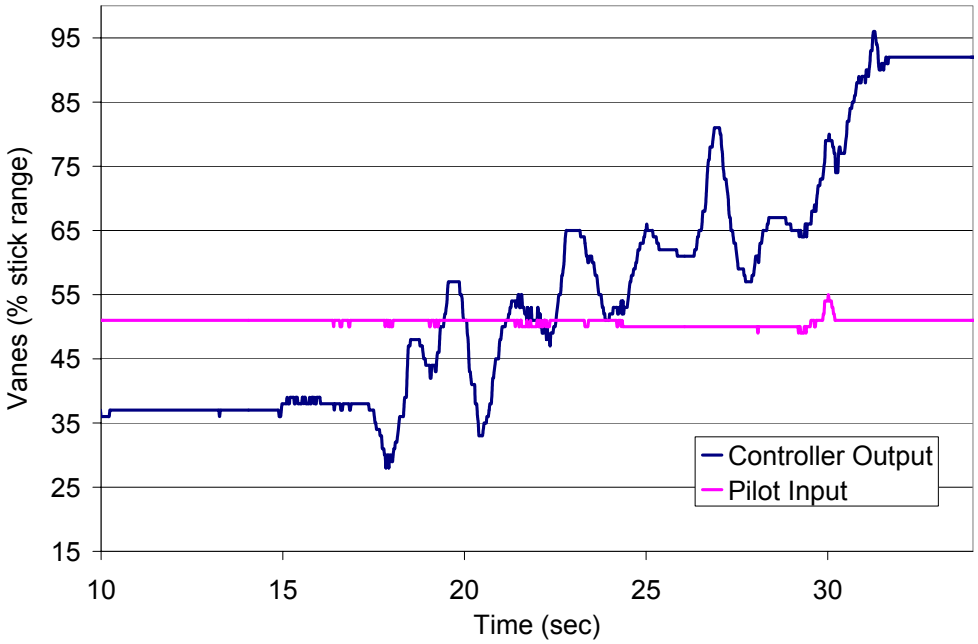

Figure 6.35: Vane command vs. time ($K_{P\theta} = 32, K_{P\phi} = 62, K_{P\psi} = 30$)

Axis	K_{cr}	P_{cr}
Pitch	32	1.4 sec (from fig. 6.30)
Roll	124	1.2 sec (from fig. 6.23)

Table 6.2: Critical gain and period for pitch and roll

Type of Controller	K_p	K_i	K_d
P	16	∞	0
PI	14.4	12.3	0
PID	19.2	27.4	3.36

Table 6.3: Ziegler-Nichols Gains (Pitch)

From table 6.1, data in table 6.2 and relations given in equations 6.4 and 6.5, we get tables 6.3 and 6.4 for pitch and roll axes, respectively.

Proportional Controller

We now use gains calculated in first rows (corresponding to proportional control) of tables 6.3 and 6.4 to implement a proportional controller. The longitudinal and lateral attitude

Type of Controller	K_p	K_i	K_d
P	62	∞	0
PI	55.8	55.8	0
PID	74.4	124	11.16

Table 6.4: Ziegler-Nichols Gains (Roll)

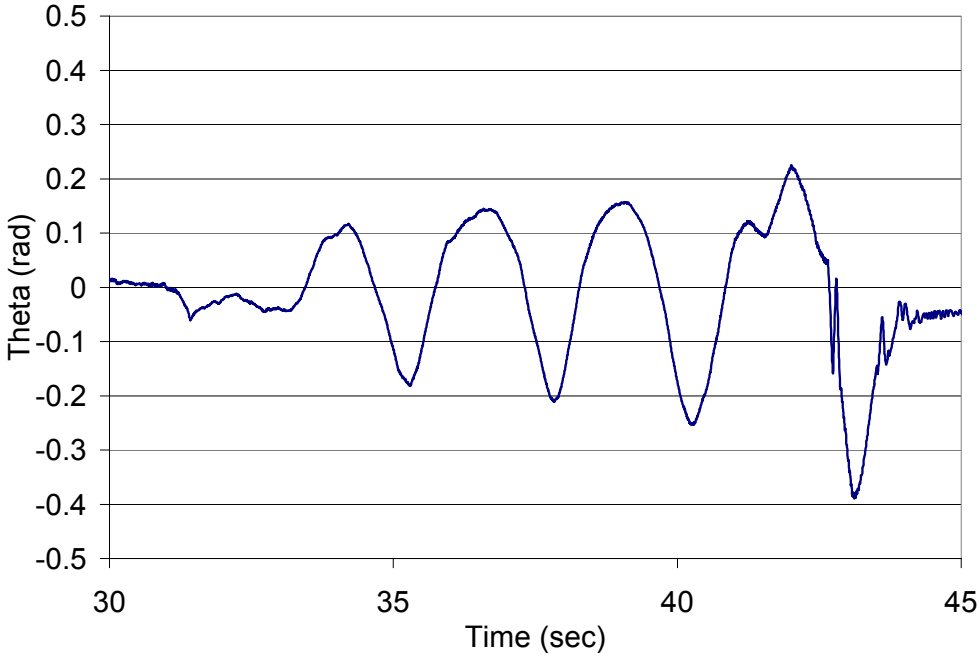

Figure 6.36: Pitch attitude vs. time ($K_{P\theta} = 16, K_{P\phi} = 62, K_{P\psi} = 30$)

results for this flight test are shown in figures 6.36 and 6.37, respectively. The corresponding commands given by the controller are shown in figures 6.38 and 6.39. The performance of the controller is satisfactory but it is found to slowly diverge, i.e., the vehicle is seen to fly in slowly growing horizontal circles.

With some manual tweaking the performance of the controller is found to improve. This setting corresponds to a gain of 20% stick range per radian in pitch and 55% stick range per radian in roll. The longitudinal attitude and the controller inputs are shown in figures 6.40 and 6.41, respectively. Figures 6.42 and 6.43 show the lateral cyclic commands and the lateral attitude of the vehicle.

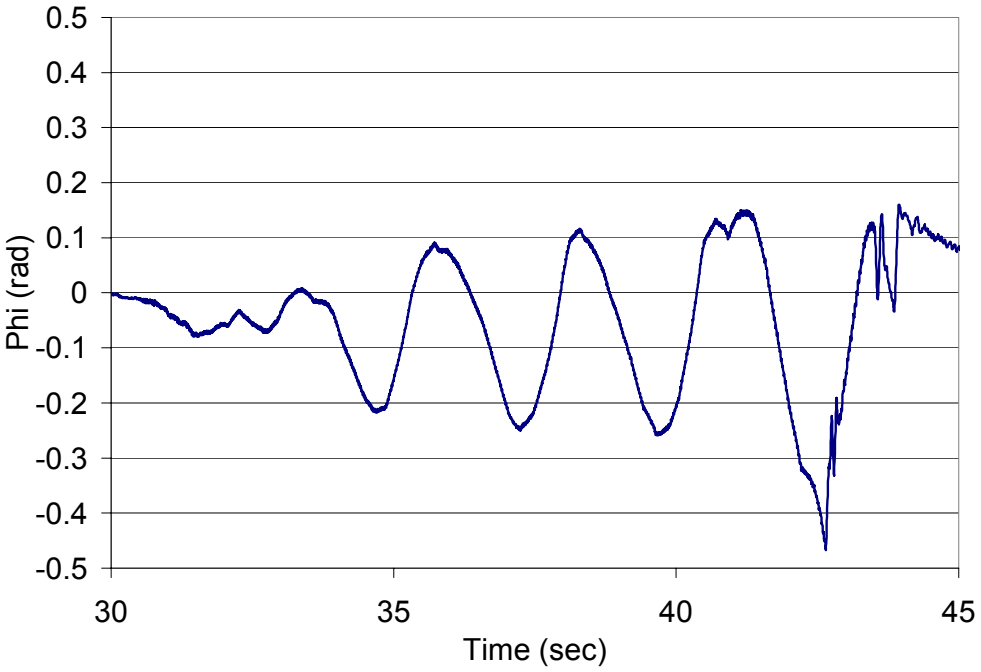

Figure 6.37: Roll attitude vs. time ($K_{P\theta} = 16, K_{P\phi} = 62, K_{P\psi} = 30$)

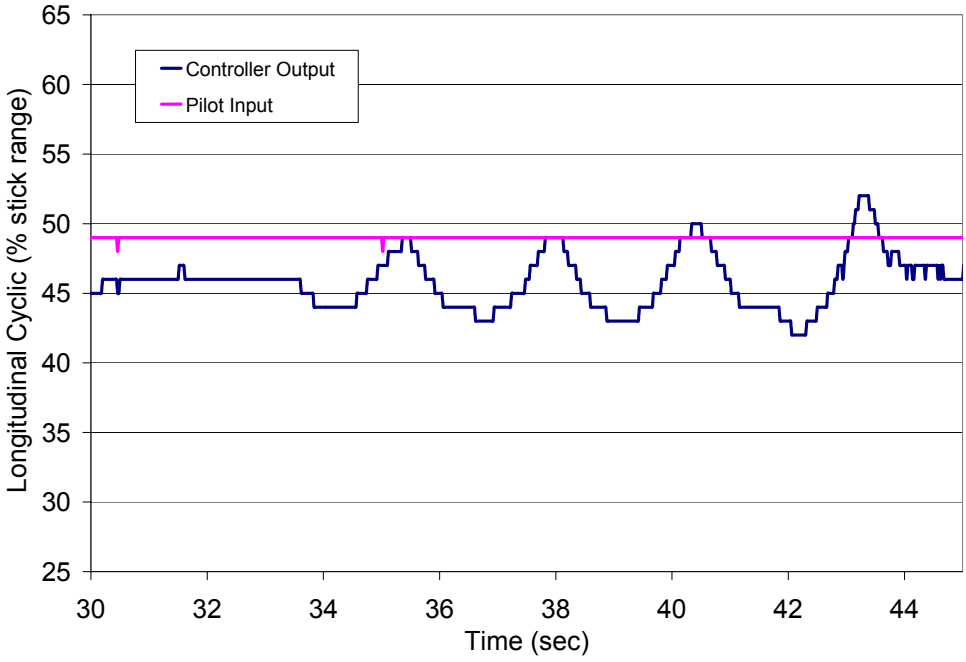

Figure 6.38: Longitudinal cyclic vs. time ($K_{P\theta} = 16, K_{P\phi} = 62, K_{P\psi} = 30$)

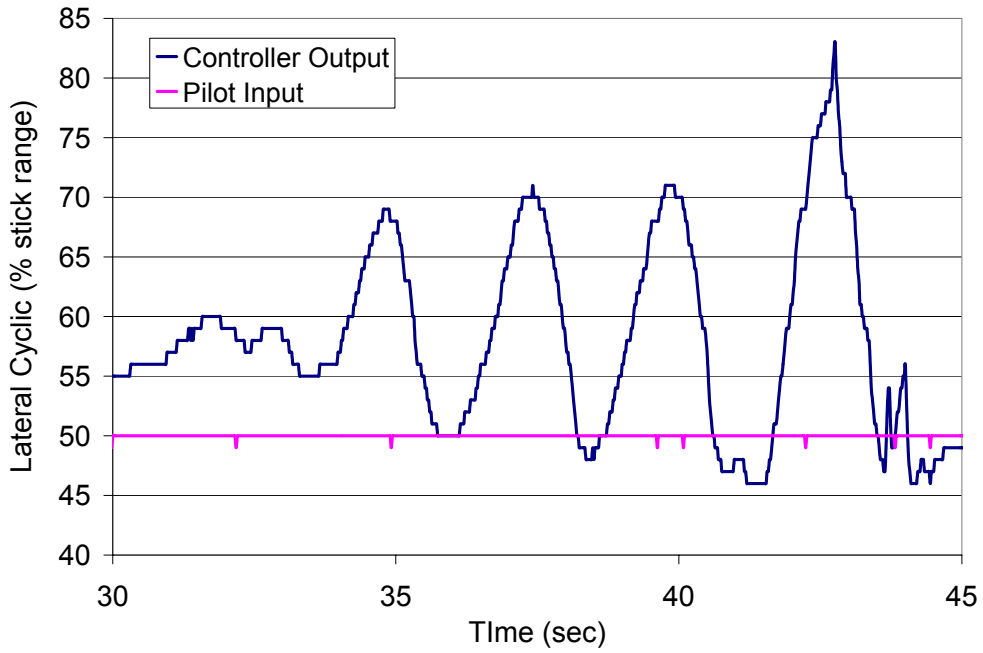

Figure 6.39: Lateral cyclic vs. time ($K_{P\theta} = 16, K_{P\phi} = 62, K_{P\psi} = 30$)

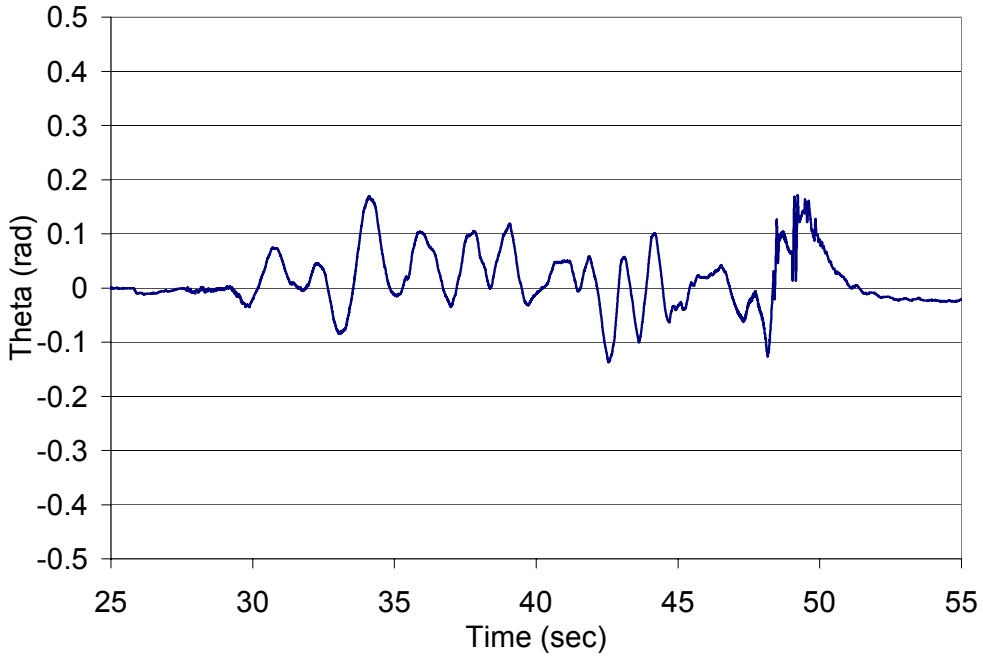

Figure 6.40: Pitch attitude vs. time ($K_{P\theta} = 20, K_{P\phi} = 55, K_{P\psi} = 30$)

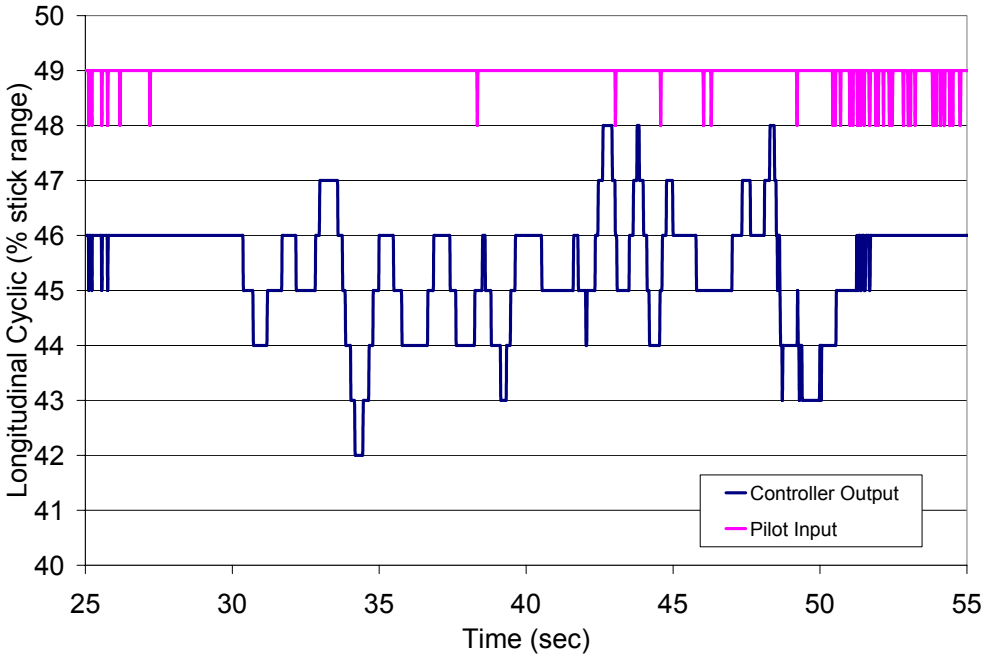

Figure 6.41: Longitudinal cyclic vs. time ($K_{P\theta} = 20, K_{P\phi} = 55, K_{P\psi} = 30$)

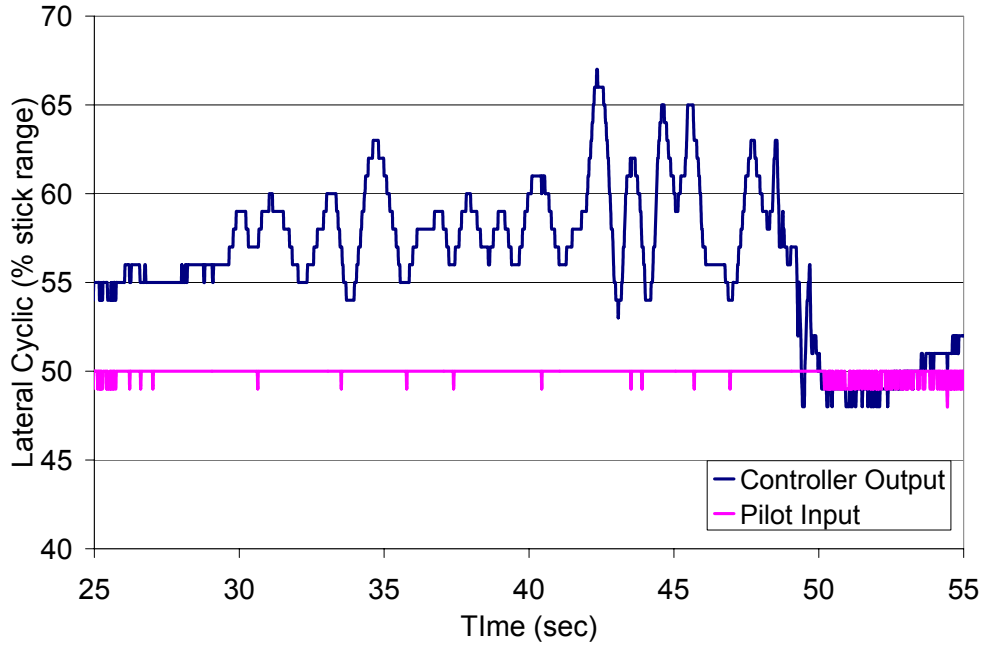

Figure 6.42: Lateral cyclic vs. time ($K_{P\theta} = 20, K_{P\phi} = 55, K_{P\psi} = 30$)

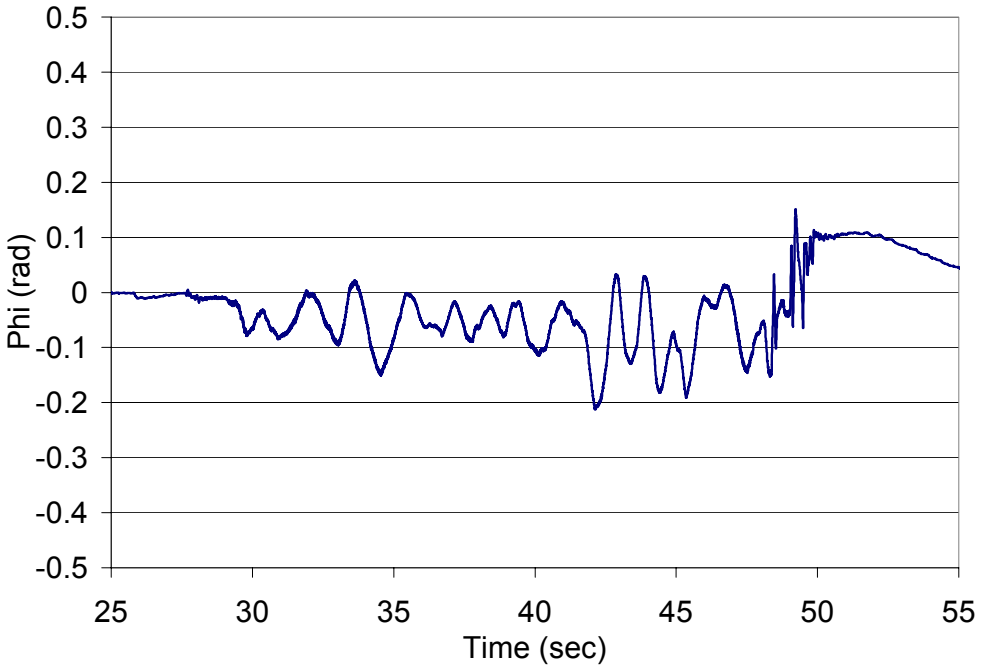

Figure 6.43: Roll attitude vs. time ($K_{P\theta} = 20, K_{P\phi} = 55, K_{P\psi} = 30$)

Proportional-Integral Controller

Using the gains in the second rows (corresponding to PI control) of tables 6.3 and 6.4, we implement the proportional-integral controller. The longitudinal and lateral attitudes are shown in figures 6.44 and 6.45, respectively. The corresponding controller commands are shown in figures 6.46 and 6.47. It is found to give a satisfactory performance.

Proportional-Integral-Derivative Controller

Next, we use the gains in the third rows (corresponding to PID control) of tables 6.3 and 6.4. The longitudinal attitude and the controller commands are shown in figures 6.48, 6.49, respectively. Figures 6.50 and 6.51 show the lateral attitude and the corresponding controller commands, respectively.

As can be seen from these figures, the controller needs to be tweaked in order to

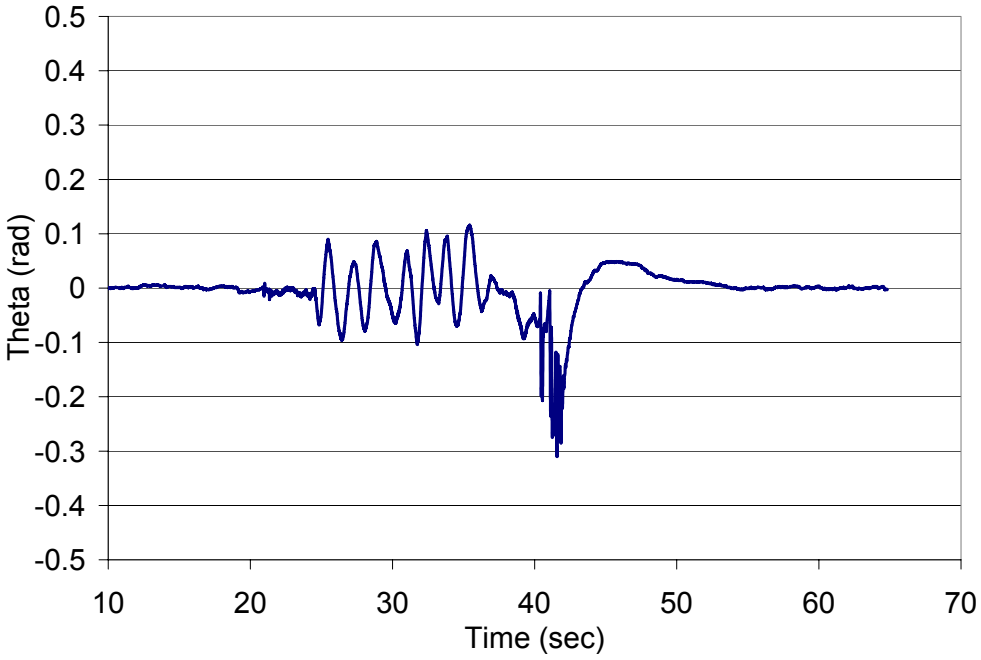

Figure 6.44: Pitch attitude vs. time ($K_{P\theta} = 14.4, K_{I\theta} = 12.3, K_{P\phi} = 55.8, K_{I\phi} = 55.8, K_{P\psi} = 30$)

improve its performance. Starting with the gains given by Ziegler-Nichols tuning method, the gains are manually tweaked and the gains in table 6.5 are found to give satisfactory performance. The longitudinal attitude and the controller commands are shown in figure 6.52 and 6.53, respectively. Figures 6.54 and 6.55 show the lateral attitude and PID controller commands, respectively.

Axis	K_p	K_i	K_d
Pitch	24	30	2
Roll	60	70	3

Table 6.5: PID Controller Gains

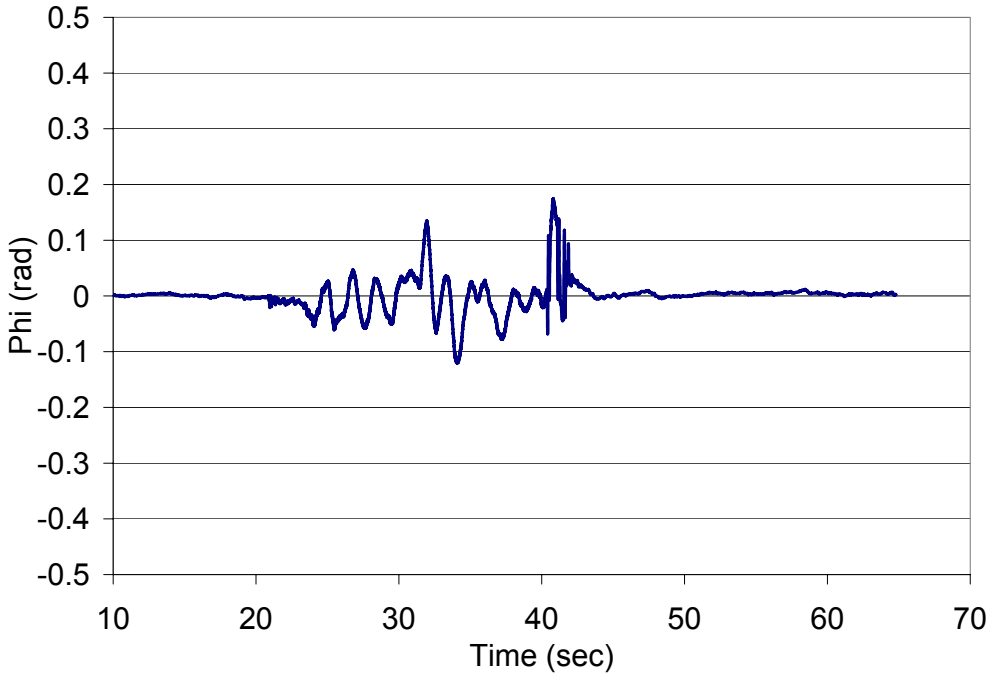

Figure 6.45: Roll attitude vs. time ($K_{P\theta} = 14.4, K_{I\theta} = 12.3, K_{P\phi} = 55.8, K_{I\phi} = 55.8, K_{P\psi} = 30$)

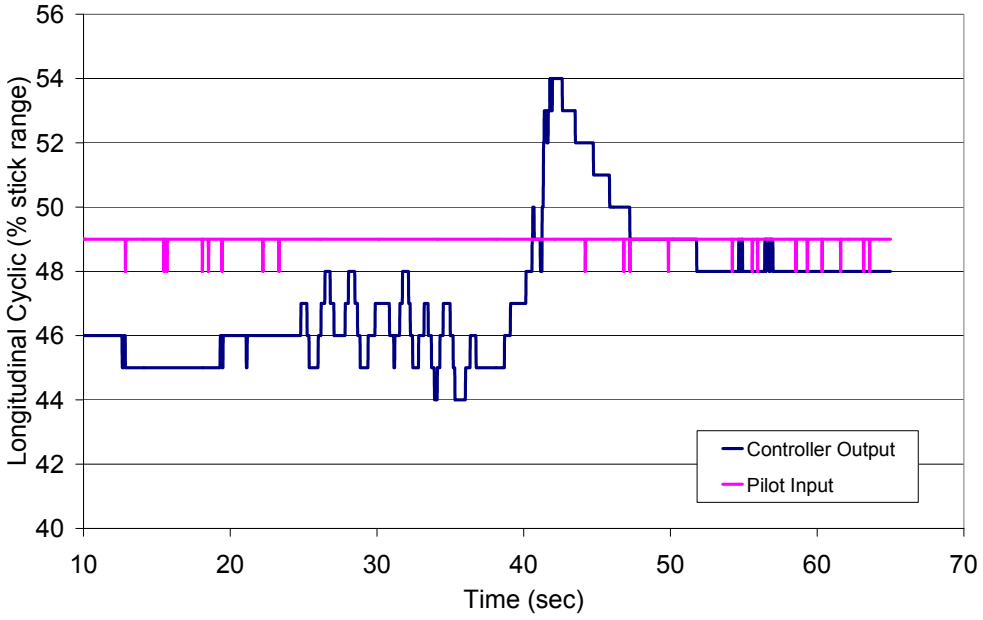

Figure 6.46: Longitudinal cyclic vs. time ($K_{P\theta} = 14.4, K_{I\theta} = 12.3, K_{P\phi} = 55.8, K_{I\phi} = 55.8, K_{P\psi} = 30$)

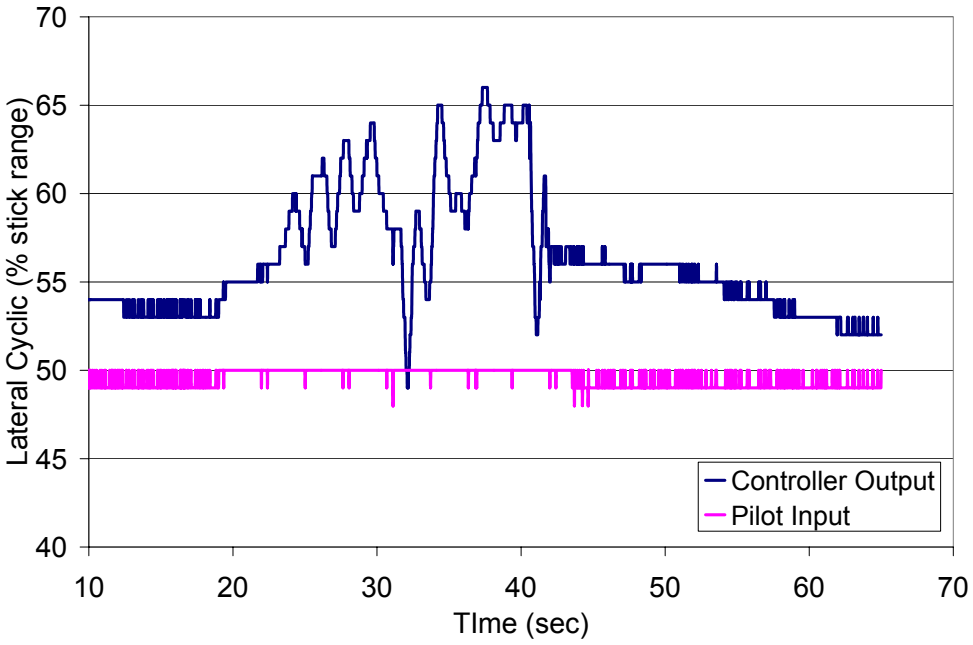

Figure 6.47: Lateral cyclic vs. time ($K_{P\theta} = 14.4, K_{I\theta} = 12.3, K_{P\phi} = 55.8, K_{I\phi} = 55.8, K_{P\psi} = 30$)

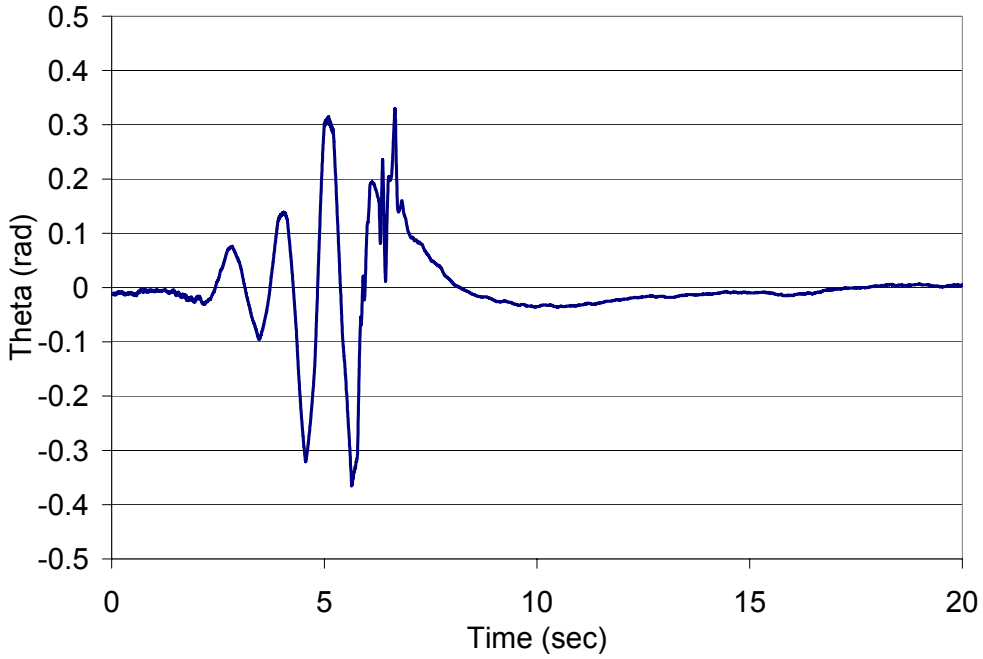

Figure 6.48: Pitch attitude vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

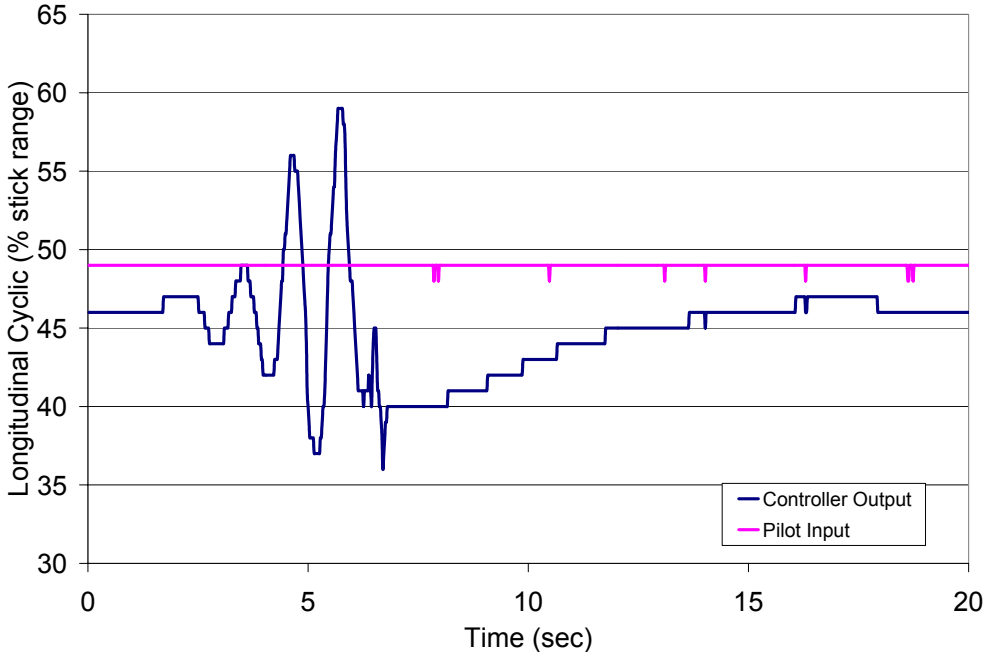

Figure 6.49: Longitudinal cyclic vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

6.3.2 Perturbation Response

In order to test the robustness of the proportional controller (with 25% stick range per radian gain), we perturb the system in pitch and study its response. A large pilot input is applied at about 36 seconds into the experiment as shown in figure 6.56. The variation of the pitch attitude for this input is shown in figure 6.57. From figure 6.56, we see that the controller applies a corrective action as soon as the attitude tries to diverge as a result of the perturbation. The controller is successful in stabilizing the attitude to within 0.1 radian in 3 seconds (fig. 6.57).

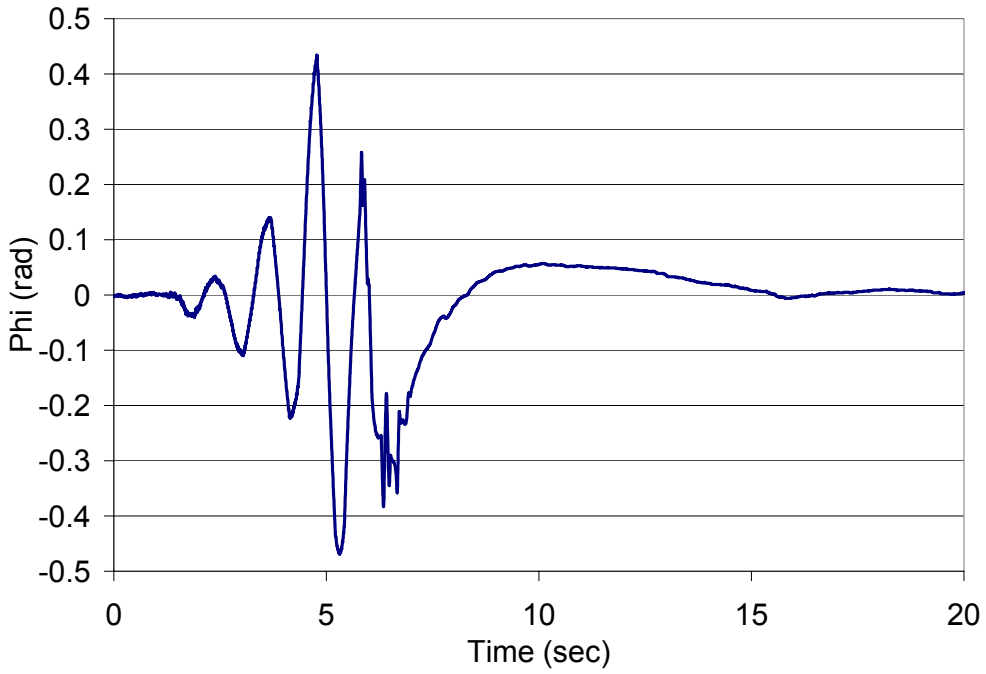

Figure 6.50: Roll attitude vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

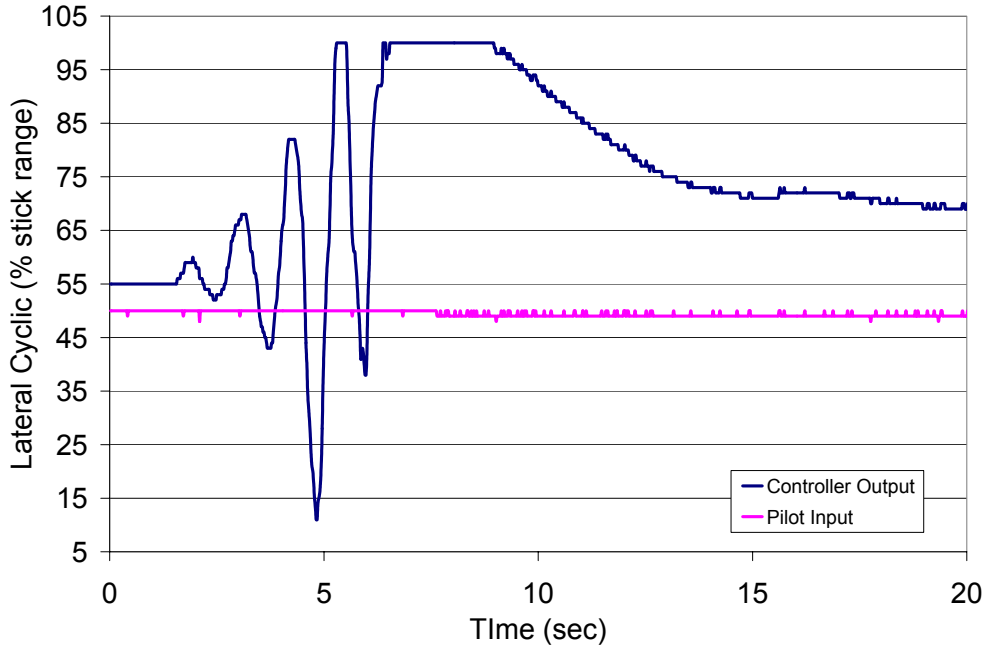

Figure 6.51: Lateral cyclic vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

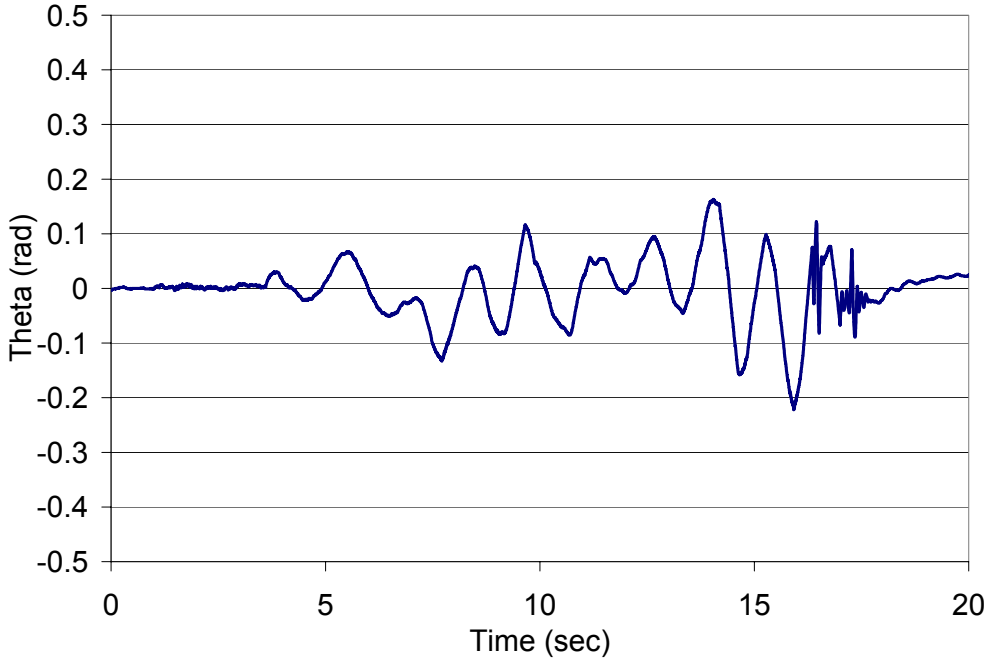

Figure 6.52: Pitch attitude vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

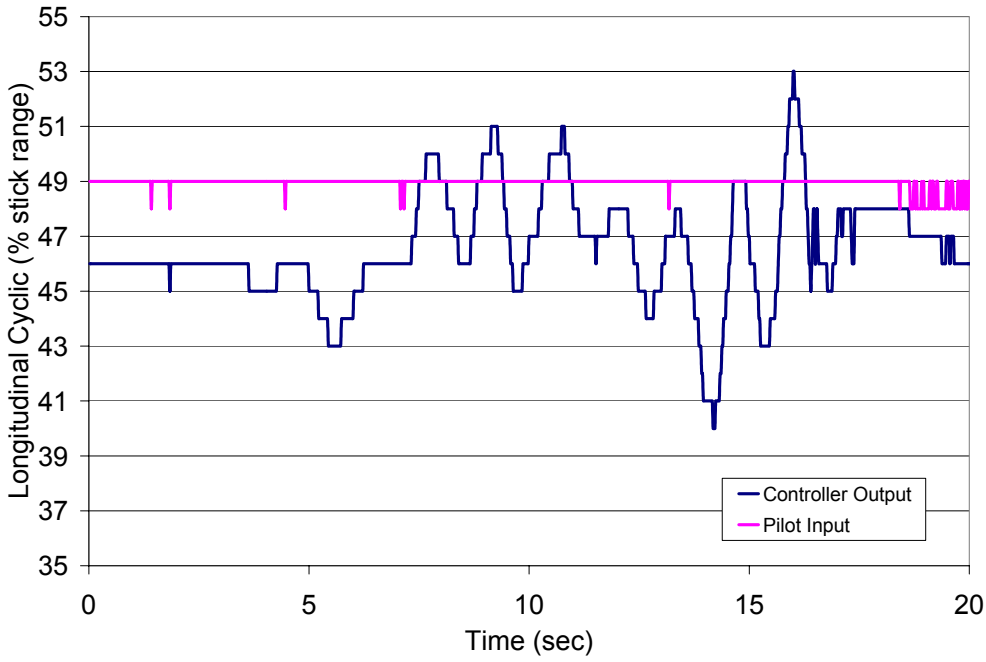

Figure 6.53: Longitudinal cyclic vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

6.4 Discussion

From the results presented in this chapter, we see that three independent SISO PID loops can be used to stabilize attitude of the Giant. We get a good estimate of proportional gains by computing the gains used by a human pilot during a manual test-flight. Ziegler-Nichols method is used to tune the roll and pitch controller gains. Since roll and pitch axes of the system are coupled, an iteration-based approach has been used in order to determine their respective critical gains and periods. One axis is stabilized at a time with the best known gain setting and the gain on the other axis is changed in order to determine its critical parameters. The value of these critical parameters is used to update the best known gain setting for one axis at a time. The process is repeated till same critical gain and period

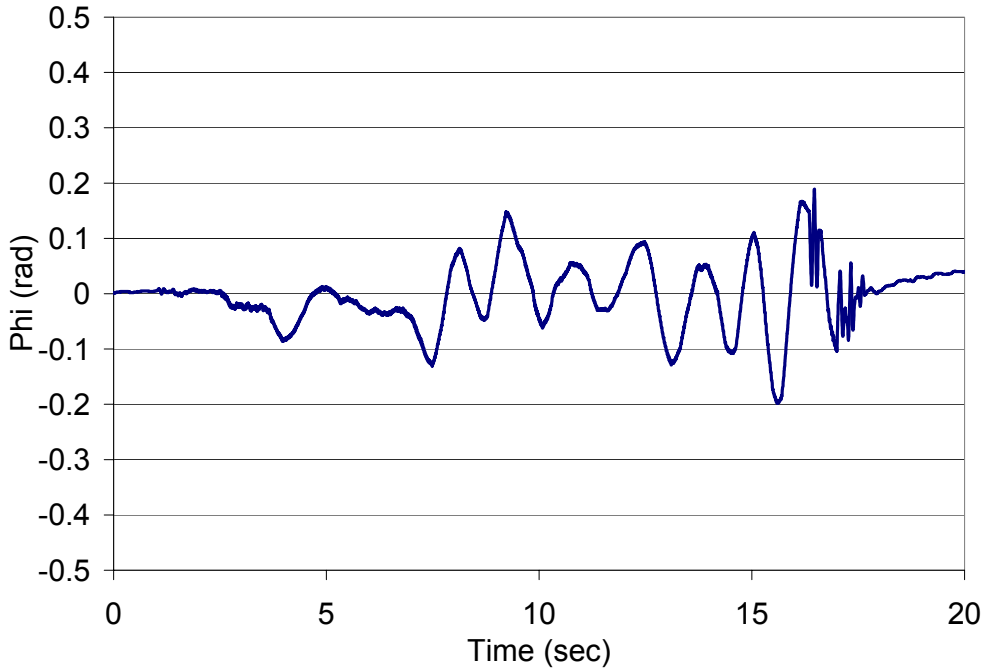

Figure 6.54: Roll attitude vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

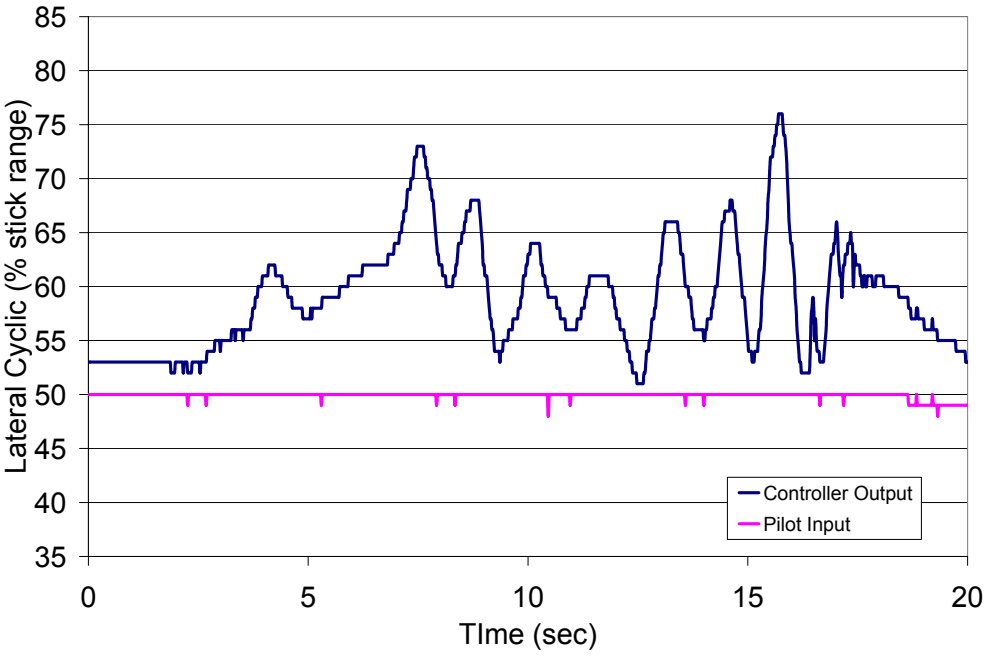

Figure 6.55: Lateral cyclic vs. time ($K_{P\theta} = 19.2, K_{I\theta} = 27.4, K_{D\theta} = 3.36, K_{P\phi} = 74.4, K_{I\phi} = 124, K_{D\phi} = 11.16, K_{P\psi} = 30$)

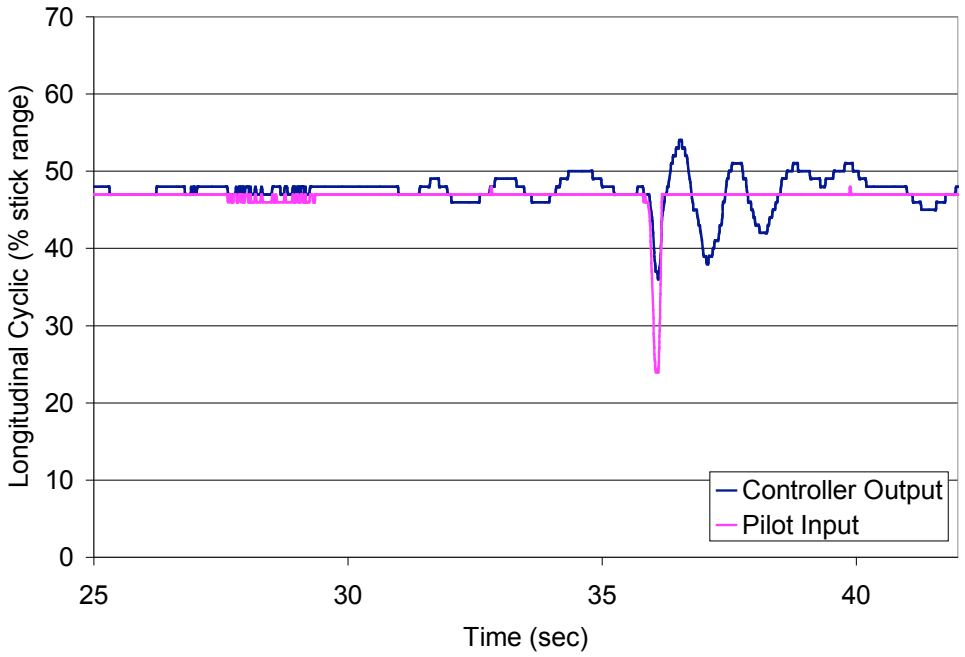

Figure 6.56: Longitudinal cyclic with perturbation at 36 sec

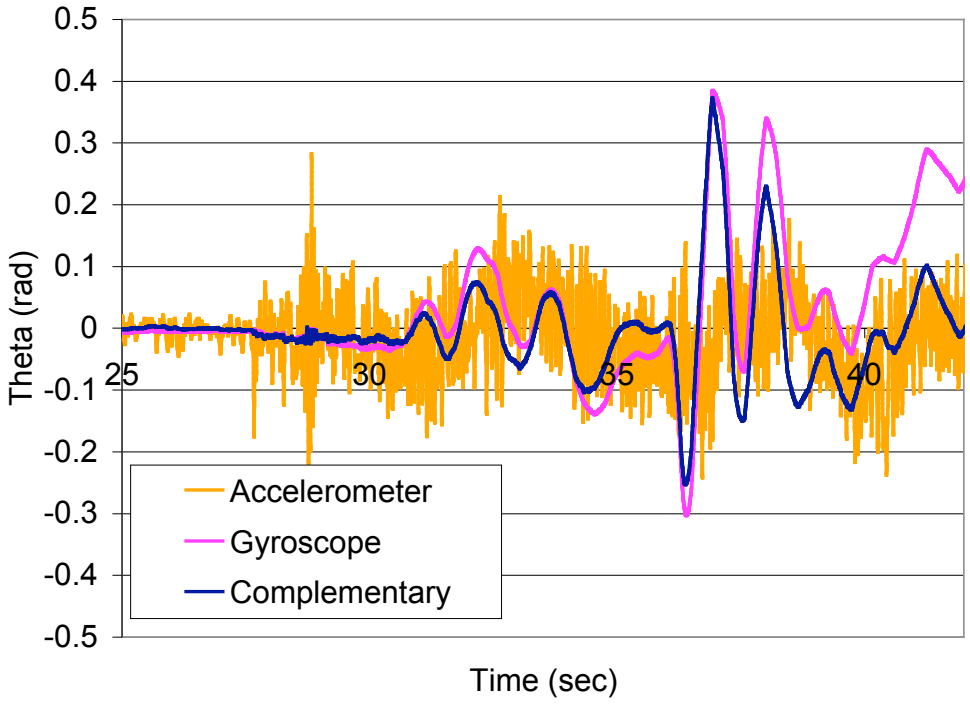

Figure 6.57: Pitch attitude response to longitudinal perturbation

values are obtained for both the axes from two successive iterations.

Having obtained the critical parameters for both axes, the gains for P, PI and PID controllers are computed following the Ziegler-Nichols method. Some adjustments are required to get good performance for each case. The final performance of the system is found to be very similar in all the three cases. Proportional controllers in pitch and roll are therefore found to be sufficient for implementing attitude stabilization on the Giant. This is able to maintain the attitude of the Giant within ± 0.1 radian in both pitch and roll. It was observed that this corresponds to the vehicle hovering over ground within a radius of about 2 meters during the one minute duration of test-flight.

Chapter 7

Conclusions and Future Work

7.1 Summary

The preliminary experiments conducted to validate the control setup were restricted to a single degree-of-freedom motion, namely, yaw (Section 3.1). The same yaw stabilization was then implemented on the Giant in free flight with the development of the onboard stabilization system (Section 3.2). It was soon realized that a number of features were desired to enable faster and efficient development of autonomous control (Section 3.2.4). A new ground-based control setup is then developed incorporating those features (Section 5.2).

With the objective of implementing attitude feedback control on Giant, an appropriate sensor package, along with a high bandwidth telemetry system is identified (Section 5.3.1). Gyroscopes and accelerometers are the two main sensors that can be used for sensing attitude. In parallel with the development of hardware, attitude transformation and estimation algorithms have been investigated (Chapter 4). The attitude information

from gyroscopes is found to drift with time (low frequency phenomenon) and the attitude from the accelerometers if found to be corrupted with high frequency noise due to vibrations. Complementary filter is used to fuse information from both these sources to give a reliable estimate of attitude. It has been experimentally validated and is found to give much better results instead of using either of accelerometers or gyroscopes alone (Section 4.4).

The hardware and the attitude algorithms have then been used to implement 3 independent SISO PID loops, one each in pitch, roll and yaw, for attitude control. In order to get an estimate of PID gains, a modified Ziegler-Nichols method is adopted. The system is then tuned in P, PI and PID modes and the results are presented in Chapter 6. The controller response to perturbation in attitude has also been investigated.

7.2 Conclusions

For efficient testing and validation of control strategies on MAVs, a ground-based system is better suited over an onboard system for the following reasons:

- **Easy to debug:** as the controller is programmed in LabVIEW on a PC, the standard probe and other debug tools are available to monitor its execution. The charts and graphs in LabVIEW also provides good visibility of different critical parameters of the system.

- **Easy to reprogram:** the controller can be modified on a PC in graphical programming environment of LabVIEW. This makes it much more efficient to evaluate different algorithms, instead of reprogramming an embedded device, as in the case

of onboard controller implementation.

- **Change parameters on the fly:** several parameters, like gain settings on different channels, can be changed as and when required, even during the flight of the vehicle.

- **Portability of the System and Modular code:** Modular design makes it easy to change transmitter-receiver or upgrade the sensor package. All such differences are captured in the different interface modules. The core of the controller remains unaffected by such changes. This makes the system portable between different vehicles with different configurations (eg., Giant and Micor).

- **Selective autonomy:** using a two way interface between the transmitter and the PC, we can select the number of channels that need to be controlled by the PC and by the human pilot. For example, for autonomous hover, we use manual throttle command but the other 3 channels (cyclics and vane angle) are autonomously controlled. This feature is also helpful in implementing maximum channel limits and scale factors on different channel outputs. For example, it is quite useful to have dual-slope or exponential-curve on the throttle channel in order to reduce the stick sensitivity near hover RPM.

- **Manual/auto switching:** this setup provides a quick way to switch between the autonomous mode and the full manual mode, by flipping a switch on the transmitter, in case of any problem or malfunction of the controller.

Other main conclusions that follow from this work are listed below:

- With the availability of advanced wireless transceivers, it is now possible to have a high bandwidth (57.6 kbps) and a long range (about 200 meters) wireless link, enabling ground-based implementation of test-bed.

- A complementary filter is able to reject high frequency noise in accelerometer data and low frequency drift in gyroscope data. It thus gives a very reliable estimate of attitude for low translational-acceleration flights.

- The algorithms for transformation and filtering of raw data received from the sensors and generating the servo commands, with real time performance, can be implemented much more efficiently on a modern PC (with dual-core processor) using LabVIEW. A lot more programming effort and tweaking is required with the older generation PCs (single-core processors) in order to get a reliable real-time performance using same algorithms.

- The actuation bandwidth of the system is only limited by the servo bandwidth; i.e., the servos are updated at the maximum rate allowed by the standard servo-control protocol (50 Hz), using the microcontroller.

- Three independent SISO PID loops can be used for control and stabilization of the Giant for low-speed flight. This enables the use of simple control scheme for attitude stabilization of the Giant.

- The pitch and roll axes need different gains because of the non-symmetric servo attachment to the swashplate.

- Modified Ziegler-Nichols tuning method gives good starting values for PID gains, but some adjustments are required to get good performance.

- All three, P, PI and PID controllers are found to give comparable performance for attitude stabilization.

- A suitably tuned PID (or P, or PI) controller can keep the roll and pitch attitude within ± 0.1 radians. From the flight tests it was observed that the vehicle remains with a radius of about 2 meters for a one minute controlled hover-flight using only attitude feedback.

- The controller is found to stabilize the vehicle against the perturbations in attitude issued by the pilot using the transmitter stick inputs. The attitude was found to reduce from ± 0.4 radians to within ± 0.1 radians in 3 seconds.

7.3 The Future Roadmap

As mentioned before, the goal of research in this area is to enable autonomous flight and operation of an MAV. Looking at the bigger picture, the controller for an autonomous air vehicle can be hierarchically decomposed into three different layers as shown in figure 7.1 [41]. At the highest level, we have the strategic control layer, which does mission planning and way-point generation. Tactical layer forms the intermediate layer and takes care of guidance and trajectory planning. At the lowest level is the reflexive layer that includes the traditional control functions like stabilization, regulation and command tracking. Development of reflexive layer needs work in two primary areas, state estima-

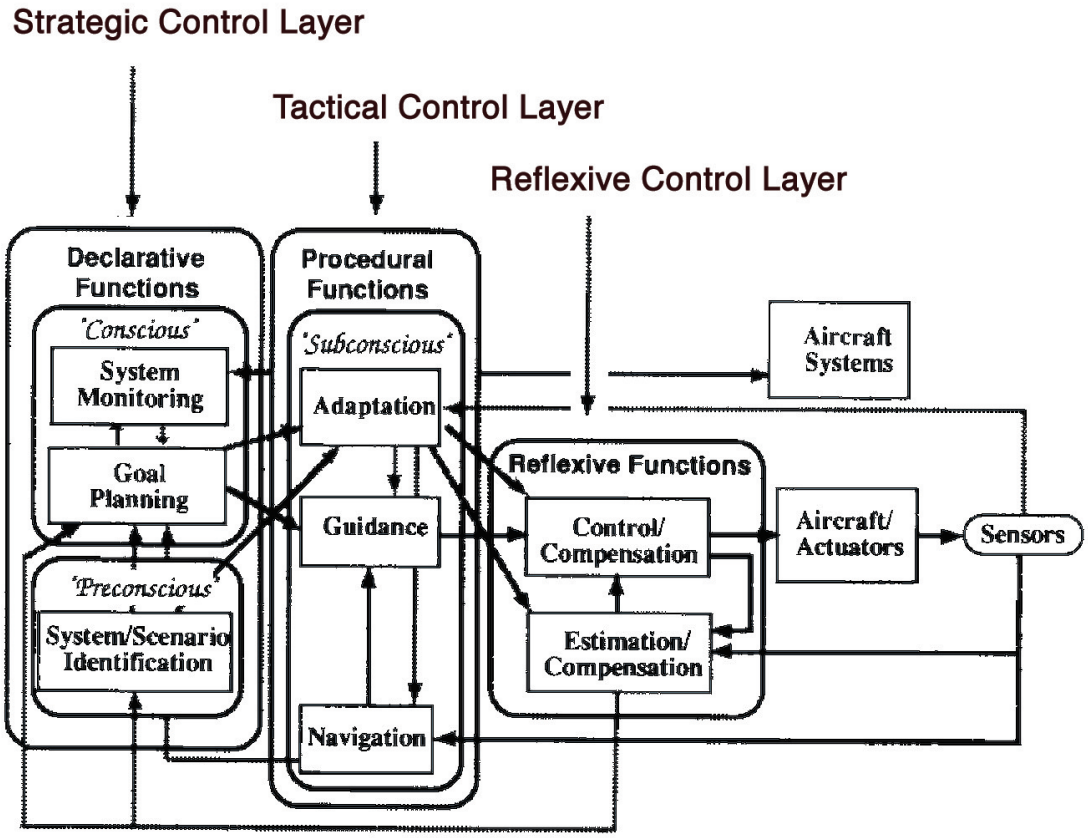

Figure 7.1: Hierarchical decomposition of autonomous control [41]

tion and stabilization or command-tracking controller design. Besides these, a test-bed to implement, test and evaluate the controller needs to be developed.

During the course of this project a generic test-bed for control implementation had been developed. It has so far been used to implement an attitude-feedback PID controlled reflexive layer. Following are the suggestions for future work in this area:

Translation Feedback Control

As discussed in section 5.1, and shown in figure 5.1, the inner attitude-stabilization loop forms the core of the controller. Although the vehicle is actuated using lateral and longitudinal cyclics, which correspond to attitude commands, it is not practical or intuitive to issue attitude commands to a vehicle in order to navigate it from one point to another. This needs to be augmented with an outer translation-stabilization loop in order to directly be able to issue translation or position commands instead of attitude commands.

The challenge in implementing the translation feedback control is to identify appropriate sensors that not only give reliable position information, but are also sufficiently compact, light weight and can operate in closed environments. For initial experiments, one may use a Global Positioning System (GPS) to demonstrate the basic working of the system [42]. The GPS is of limited use for MAV applications, as it does not work in closed environments (e.g., inside a building), i.e., when it is not in line of sight of at least three GPS satellites.

The other techniques that have evolved over recent years include vision based systems [43], such as optic flow sensors [44]. The optic flow sensors are based on visual cues that insects use to navigate. It gives an estimate of a ratio of velocity to distance from a textures surface as seen by the sensor. So, with the knowledge of distance (using some other sensor, like pressure transducer or laser ranging system), the velocity of the vehicle can be estimated, and vice-versa.

Model Based Control

In the present work, we have implemented PID control in order to achieve some level of automation. In order to expand the flight envelope and make use of the maneuvering capability of the vehicle, it is important to develop a model based controller [19]. As the MAVs are small in size and have much lower inertias compared to the manned aircraft, they have quicker vehicle responses. It is therefore required to have very high bandwidth models in order to implement model based control on MAVs. A basic structure of the model of the vehicle first needs to be worked out from the first principles. The different unknown parameters in this model are then identified using system identification techniques [19, 45]. Once a sufficiently "good" model has been developed, different model-based control techniques [46] can be implemented and evaluated using this testbed.

This will complete the development of the reflexive layer and then the research effort can focus on development of higher layers of autonomy, i.e., the tactical and strategic layers.

Appendix A

Gyroscope Drift

A basic explanation of gyroscope drift is given in Section 4.2. In this appendix, a more detailed explanation is presented in a form of questions and answers.

- **Why does the attitude reading obtained from gyroscope drift?**

  The fundamental reason is the non-exact determination of bias.

- **What is bias?**

  The gyroscope is an analog angular rate sensor and gives a non-zero voltage for zero angular rate. This voltage is called the bias. The angular rate is given by:

$$Rate = (Voltage - Bias) * Calibration\_constant \tag{A.1}$$

  From this, we see that if bias is not correctly estimated, the "rate" obtained will be erroneous. As previously mentioned, the gyroscope only gives us this rate information. It needs to be integrated over time to compute the angular position (attitude).

The errors in the determination of rate (due to incorrect bias) lead to diverging error in attitude estimation (the drift)!

- **Why is bias not correctly determined?**

  There are two reasons for this. The first has to do with the fundamental physical issues in the operation of gyro itself, and the second has to do with the way they are interfaced and sampled. Let us first look at the second issue.

  Let us assume that we have a perfect gyro (with perfectly steady bias). There is still a limit to the resolution with which the bias can be determined due to unavailability of infinite-bit analog-to-digital converters! A 10-bit ADC on a 5-volt scale can only resolve 5/1024 volts, i.e., about 4 mV. The datasheet for ADXRS150 specifies the sensitivity as 12.5 mV/deg/s [30]. Therefore we can only resolve a rate of 0.32 deg/sec with this setup. An error of 0.32 deg/sec in the estimation of bias would lead to a drift of 32 deg in every 100 seconds!

  But the gyro is not noise-free. It is in some ways good for us as it introduces a 'dithering' effect which leads to a finer estimation of bias!

- **What is dithering?**

  Suppose the actual output of an analog sensor is 2.6 volts, and it is being sampled by a device that truncates the fractions and resolves the data only in unit's place. This device will show a result of 2 instead of 2.6. Now if we introduce a noise in the data such that 60% of the time the result of sampling device is 3 and remaining 40% of the time it is 2, the average will turn out to 2.6! This introduction of noise

to improve the estimation of a signal is called dithering.

- **Why do we still have problems (after having exploited dithering)?**

  Firstly, we only have a finite number of samples available at any point in time. Therefore, going by the above example, there is no guarantee that the data will be sampled as 3 for 60% of the time; sometimes it might show 3 for much less than 60% of the samples or sometimes more. Even if the bias was correctly determined by taking a large number of samples, the estimation of rate (and thus angle) for any subsequent instantaneous gyro reading will still be wrong (although for white noise, the angle would fluctuate around zero, if bias is perfectly determined).

  Secondly, the gyro is not a perfect sensor. That is, the construction of gyro and its working principle is such that the reading from the gyro is influenced by a number of factors instead of just angular rate. The MEMS gyro measures the angular rates by means of Coriolis acceleration, which is measured by estimating the displacement of a resonating mass and its frame (due to the Coriolis effect) through capacitive sensing elements attached to the resonator [48]. The ADXRS gyro electronics can resolve capacitance changes as small as $12\text{x}10^{21}$ farads (12 zeptofarads) from beam deflections as small as 0.00016 Angstroms (16 femtometers). These sub atomic displacements are meaningful as the average positions of the surfaces of the beams, even though the individual atoms on the surface are moving randomly by much more. There are about 10^{12} atoms on the surfaces of the capacitors, so the statistical averaging of their individual motions reduces the uncertainty by a factor of 10^6 (dithering effect again!). We cannot do better because of the air molecules

which cause the structure to move. Although these are also averaged, their effect is far greater. The air is not removed because the device consists of a very fine film (4 micrograms) and flexures (1.7 micron wide) that are suspended over the silicon substrate and the air molecules cushion this structure, preventing it from damage due to shocks.

- **How to get better attitude then?**

  There could be two ways of getting better attitude estimate:

  1. Study the nature of bias noise and devise an advanced filter to estimate that noise much more accurately [49]. This is an involved exercise in itself and limited success has been achieved to date.

  2. An alternate way is to use some sort of external aiding to correct for drift in the gyro data. This is typically done using an accelerometer [32]. An accelerometer has no drift as it directly gives the attitude (no integration required) by measuring the gravity vector (assuming that linear accelerations are small compared to gravity). Accelerometer and gyroscope have complementary characteristics. Accelerometer gives a very good low frequency estimate (no drift) but has high frequency noise due to vibrations. Gyros on the other hand have a good high frequency performance (higher rates lead to larger capacitance to be measured, where the air molecule effect is smaller than the Coriolis effect; also the integration used for angle computation has an effect of averaging out the signal, so that there are no jerks in the data even at high frequency) but a poor low frequency performance (drift). Therefore we use a

complementary filter to fuse the data from both these sensors to get the best estimate.

- **What are the other sources of gyro errors?**

  1. **Sensor saturation:** If the gyro is subjected to angular rates higher than what it can measure, it will saturate and in turn miss all the action during that time. Although the vehicle is not expected to move at rates higher than +- 150 deg/sec (rate limits for ADXRS 150), the vibrations often exceed that limit and lead to very fast loss of orientation.

  2. **Temperature:** The gyro is known to be sensitive to temperature. In lab experiments no conclusive fixed temperature calibration could be established.

  3. **ADC warm-up:** It has been observed that when the entire system is switched on, the analog-to-digital converter and other electronics take some time to stabilize. The bias computed from the first few minutes of data is significantly worse than that computed a little later.

- **How do we take care of these?**

  The ADC warm up and temperature corrections are taken care of by letting the system run for a few minutes before starting the experiment. The bias estimation should be done after this initial period.

  Sensor saturation (vibration isolation) is taken care of by appropriate mounting of the sensor on the vehicle. A packaging is provided to absorb high angular rate vibrations.

To reiterate, gyro drift occurs because of integration. A MEMS accelerometer does not have a constant analog bias for all of the same reasons as a gyro. The only difference is that the accelerometer data does not have to be integrated to measure the angle. If velocity was measured from integrating the accelerometer, an analogous issue as gyro drift would occur. It could be called accelerometer drift.

Appendix B

Listing of Microcontroller Codes

The microcontroller codes discussed in Sections 3.1.2, 3.1.3, 5.3.2 are listed below. All these codes are written for the CCS-C cross compiler [26] for PIC microcontrollers.

The Code for Yaw Stabilization Experiment (Section 3.1.2)

```
// Initialization
#include <16f877.h>
#fuses HS,NOLVP,NOWDT,PUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#include <stdlib.h>
#include <input.c>
#define GREEN_LED PIN_A5
#define YELLOW_LED PIN_B4
#define RED_LED PIN_B5
#define PUSH_BUTTON PIN_A4

char state;
int32 high, cycle = 12500;

// The Interrupt Service Routine
// to generate servo command once every 20ms
#int_ccp1
void isr(){
```

```
   if (state == 1)
   {
      output_low(PIN_C2);
      CCP_1 += cycle - high;
   }
   else
   {
      output_high(PIN_C2);
      CCP_1 += high;
   }
   if (CCP_1 > 65535)
      CCP_1 = 65535-CCP_1;
   state = 1-state;
}

//The Main Routine
void main()
{
int32 angle,temp1,inc,i,val1,val2,value,temp,base=124;
setup_ccp1(CCP_COMPARE_INT);   // Configure CCP1 as input compare
setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
state = 0;
high = 730;
angle = 10000;

enable_interrupts(INT_CCP1);   // Setup interrupt on compare
enable_interrupts(GLOBAL);

CCP_1 = 10; // set interrupt at 10 timer-1 ticks
set_timer1(0); // initialize timer-1

// initialize ADC
setup_port_a(ALL_ANALOG);
setup_adc(adc_clock_internal);
set_adc_channel( 1 );

while(TRUE)
{

value=read_adc(); // sample ADC

if (value < 10)
   output_low(RED_LED); // warning for saturation
else
   output_high(RED_LED);
```

```
if (value > 240)
   output_low(YELLOW_LED); // warning for saturation
else
   output_high(YELLOW_LED);

inc = value-base; // compute rate
angle += inc; // angle is integral of rate

if (angle<5000)
   angle = 5000; // lower limit on angle
if (angle>15000)
   angle = 15000; // upper limit on angle

temp = 3730 - (float)angle/4 - value*4; // servo command (PD control)

if (temp<630 || temp>1245) // check for servo saturation
   output_low(GREEN_LED);
else
   output_high(GREEN_LED);

high = temp; // final servo command
delay_us(50000);
}}
```

The Code for Yaw Control Experiment (Section 3.1.3)

```
// Initialization
#include <16f877.h>
#fuses HS,NOLVP,NOWDT,PUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#include <stdlib.h>
#include <input.c>
#define GREEN_LED PIN_A5
#define YELLOW_LED PIN_B4
#define RED_LED PIN_B5
#define PUSH_BUTTON PIN_A4

char state,state_read;
int32 time, rise, high, cycle = 12500;

// Interrupt Service Routine
```

```
// to generate servo command once every 20ms
#int_ccp1
void isr(){
   if (state == 1)
   {
      output_low(PIN_C2);
      CCP_1 += cycle - high;
   }
   else
   {
      output_high(PIN_C2);
      CCP_1 += high;
   }

   if (CCP_1 > 65535)
      CCP_1 = 65535-CCP_1;

   state = 1-state;
}

// Interrupt Service Routine
// to read R/C receiver
#int_ccp2
void isr2() {
   state_read = 1 - state_read;
   if (state_read==0)
   {
      if (CCP_2>rise)
         time = CCP_2-rise;
      else
         time = 65535 - rise + CCP_2;
      setup_ccp2(CCP_CAPTURE_RE);
   }
   else
   {
      rise = CCP_2;
      setup_ccp2(CCP_CAPTURE_FE);
   }
}

// The Main Routine
void main()
{
int32 angle,temp1,inc,i,val1,val2,value,temp,base=124, neutral;
setup_ccp1(CCP_COMPARE_INT);   // Configure CCP1 as input compare
```

```
setup_ccp2(CCP_CAPTURE_RE);
setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
state = 0;
state_read = 0;
time = 0;

high = 730; // default servo output
angle = 10000; // Initial value of angle
neutral = 0;

enable_interrupts(INT_CCP1);   // Setup interrupt on compare
enable_interrupts(INT_CCP2);
enable_interrupts(GLOBAL);

CCP_1 = 10; // set interrupt for timer-1 = 10
set_timer1(0); // initialize timer-1

// Initialize ADC
setup_port_a(ALL_ANALOG);
setup_adc(adc_clock_internal);
set_adc_channel( 1 );

while(TRUE)
{
value=read_adc(); // Read Gyro data
if (value < 10)
   output_low(RED_LED); // warning for sensor saturation
else
   output_high(RED_LED);

if (value > 240)
   output_low(YELLOW_LED); //  warning for sensor saturation
else
   output_high(YELLOW_LED);

inc = value-base;

angle += inc;

if (angle<5000)
   angle = 5000; // lower limit on angle
if (angle>15000)
   angle = 15000; // upper limit on angle

neutral = time-700; // modify input
```

```
// compute servo command
temp = 3730 + 833 - (float)angle/3 - value*4 + (float)neutral*2;

if (temp<630 || temp>1245)
   output_low(GREEN_LED); // check for servo saturation
else
   output_high(GREEN_LED);

high = temp; // set final servo command
delay_us(50000);

}}
```

The Code for R/C Transmitter Interface (Section 5.3.2)

```
// Initialization
#include <16F877.h>
#fuses HS,NOLVP,NOWDT,PUT
#use delay(clock=20000000)
#use rs232(baud=57600, xmit=PIN_C6, rcv=PIN_C7)
#include <stdlib.h>
#include <input.c>

char i=1,statew=0;
// set default values for different channels (write to transmitter)
unsigned int16 high[8]={700,696,712,418,690,468,676,7500};
char stater=0, pos = 0;
// set default values for different channels (read from transmitter)
unsigned int16 rise, time[10]={700,700,700,410,700,468,676,700,700,700};

// Interrupt Service Routine
// to read transmitter data
#int_ccp1
void write(){
   if (statew == 1)
   {
      output_low(PIN_E1);
      CCP_1 += 250; //400ns low time
   }
   else
   {
```

```
      output_high(PIN_E1);
      CCP_1 += high[i];
      i++;
      if (i==8)
         i=1;
   }
   if (CCP_1 > 65535)
      CCP_1 = 65535-CCP_1;
   statew = 1-statew;
}

// Interrupt Service Routine
// to read from the transmitter
#int_ccp2
void read() {
   stater = 1 - stater;
   if (stater==0)
   {
      if (CCP_2>rise)
         time[pos] = CCP_2-rise;
      else
         time[pos] = 65535 - rise + CCP_2;
      setup_ccp2(CCP_CAPTURE_RE);
      if (time[pos] > 1500) // sync pulse
         pos = 0;
      pos++;
   }
   else
   {
      rise = CCP_2;
      setup_ccp2(CCP_CAPTURE_FE);
   }
}

void main() {
   unsigned int16 temp;
   unsigned int8 output, j;

   setup_ccp1(CCP_COMPARE_INT);   // Configure CCP1 as output compare
   setup_ccp2(CCP_CAPTURE_RE);    // Configure CCP2 as input capture (rising)
   setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
   enable_interrupts(INT_CCP1);   // Setup interrupt on compare
   enable_interrupts(INT_CCP2);   // Setup interrupt on input capture
   enable_interrupts(GLOBAL);
   CCP_1 = 10;
```

```
   set_timer1(0);

   while(TRUE)
   {
      putc(255);
      for(j=1;j<7;j++) // write to PC
      {
         output = (time[j]-200) >> 2; // adjust data in 1 byte
         putc(output);
      }

      temp = getc(); // read from PC
      while (temp!=255)
         temp = getc();

      temp=getc();
      high[1] = (temp<<2) + 200; // expand data to 2 bytes
      temp=getc();
      high[2] = (temp<<2) + 200;
      temp=getc();
      high[3] = (temp<<2) + 200;
      temp=getc();
      high[4] = (temp<<2) + 200;
      temp=getc();
      high[6] = (temp<<2) + 200;
   }
}
```

Appendix C

Implementing Digital Filters

The raw data collected from the sensors is often corrupted by other unwanted signals, referred to as noise. It is therefore required to filter this noise in order to get meaningful information from the sensors. In most cases, such filters have to implemented on some or the other form of digital processor as software. Since the sensors are sampled at a very high rate (hundreds of readings per second) and the entire system has to react in real time, the implementation of the filters needs be to very efficient in terms of both processing time and memory usage.

There are many different ways to design filters, but the most common ones have their roots in simple averaging. Digital filters can be divided into two categories based on implementation [47]:

1. Finite Impulse Response (FIR), or the filters based on convolution: The output of these filters is affected by a finite number of samples taken in the past. That is, after a certain amount of time, the old sensor data will have no influence on the output

of the filter.

2. Infinite Impulse Response (IIR), or the filters based on recursion: The output of these filters is affected by all the sensor data samples collected in the past.

We now discuss the implementation of 3 different digital filters that are commonly used. The first two filters (generic FIR, and moving average) belong to the FIR type of filters and the last one (exponentially weighted averaging filter) is an IIR kind of digital filter.

Generic FIR Filter

The most generic way to design a digital filter is to *convolve* the input signal with the digital filter's *impulse response*. In this method, each sample in the output is calculated by weighting the samples in the input using the *filter kernel* (impulse response or convolution coefficients), and adding them together.

MATLAB's Filter Design Toolbox [50] provides several useful functions for designing, simulating and analyzing digital filters. 'FDATool' command in this toolbox provides a graphical interface to design filters. Filter kernels for generic FIR filters, can also be obtained using the 'fir' MATLAB command. This MATLAB function can be called directly from a LabVIEW VI during initialization and the kernel obtained can be repeatedly used over input data to get filtered data.

Although this is a powerful digital filtering technique, it needs a lot of computation per output sample. This imposes a serious limitation on how fast the sensor can be sampled and thus on the bandwidth of the control system.

Moving Average Filter

Moving average filter is one of the simplest and most commonly used filters in digital signal processing. It is very efficient as it can be implemented in a recursive form.

Suppose that at any instant k, the average of the latest n samples of a data sequence, x_i, is given by:

$$\bar{x}_k = \frac{1}{n} \sum_{i=k-n+1}^{k} x_i \tag{C.1}$$

Similarly, at the previous time instant, $k-1$, the average of the latest n samples is:

$$\bar{x}_{k-1} = \frac{1}{n} \sum_{i=k-n}^{k-1} x_i \tag{C.2}$$

Therefore,

$$\bar{x}_k = \bar{x}_{k-1} + \frac{1}{n}(x_k - x_{k-n}) \tag{C.3}$$

Here, the average at each k^{th} instant is based on the most recent set of n values. Or, a *moving window* of n values is used to calculate the average of the data sequence. It can be seen that this filter only needs one division, one addition and one subtraction operation. This is always the case, irrespective of the size of the window, n; although, we need to store the value of x_{k-n}, which may require n storage locations.

Exponentially Weighted Averaging Filter

In the previous filter design (moving average filter), each data point in the window is given equal importance when calculating the average. In dynamic systems, however, the

most current values from the sensor reflect better the sate of the plant. We now look at the implementation of a recursive infinite response filter (that uses all the sensor data from the past), that places more emphasis on the most recent data.

In the most simple form, the *exponentially weighted averaging filter* can be represented by the following equation:

$$\bar{x}_k = \alpha \bar{x}_{k-1} + (1-\alpha)\, x_k \tag{C.4}$$

where, $0 \leq \alpha < 1$

This implementation requires only one addition, 1 subtraction and two multiplications. There is no need to store a window of samples as was the case with moving average filter. The *filter constant*, α, dictates the *degree of filtering*, i.e., how strong the filtering action will be. When the filtering is strong, i.e., $\alpha \to 1$, then, $\bar{x}_k \to \bar{x}_{k-1}$, or, the current measurement does nor affect the output. On the other hand when the filtering is weak, i.e., $\alpha \to 0$, then, $\bar{x}_k \to x_k$, or, the measurement if not even modified by the filter.

Exponentially weighted averaging filter places more importance on the more recent data in an exponential manner. This is be seen by expanding equation C.4:

$$\bar{x}_k = \alpha \bar{x}_{k-1} + (1-\alpha)\, x_k, \tag{C.5}$$

and,

$$\bar{x}_{k-1} = \alpha \bar{x}_{k-2} + (1-\alpha)\, x_{k-1} \tag{C.6}$$

therefore,

$$\bar{x}_k = \alpha \bar{x}_{k-1} + (1-\alpha)\, x_k = \alpha \left[\alpha \bar{x}_{k-2} + (1-\alpha)\, x_{k-1}\right] + (1-\alpha)\, x_k \tag{C.7}$$

Or,

$$\bar{x}_k = \alpha^2 \bar{x}_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k \tag{C.8}$$

Expanding further,

$$\bar{x}_k = \alpha^3 \bar{x}_{k-3} + \alpha^2(1-\alpha)x_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k \tag{C.9}$$

From equation C.9 we see that the contribution of the older values of x_i is weighted by increasing powers of α. Since α is less than 1, the contribution of older values of x_i becomes exponentially smaller.

This is in fact a very simple implementation of first-order low-pass filter. The equivalence of a exponentially weighted averaging filter and a first-order low-pass filter is shown next.

Consider the Laplace transfer function of a first-order low-pass filter, with time constant τ_f. The following equation relates the filtered signal, $\bar{x}(s)$ to the measurement, $x(s)$:

$$\frac{\bar{x}(s)}{x(s)} = \frac{1}{1+\tau_f s} \tag{C.10}$$

Time equivalent of equation C.10 is:

$$\tau_f \frac{d\bar{x}(t)}{dt} + \bar{x}(t) = x(t) \tag{C.11}$$

Now, we discretise equation C.11 using the following approximation:

$$\frac{d\bar{x}(t)}{dt} \approx \frac{\bar{x}_k - \bar{x}_{k-1}}{T_s} \tag{C.12}$$

where, T_s is the interval between each measurement, i.e., the sampling interval. Thus equation C.11 leads to:

$$T_f \frac{\bar{x}_k - \bar{x}_{k-1}}{T_s} + \bar{x}_k = x_k \tag{C.13}$$

Simplification and re-arrangement gives:

$$\bar{x}_k = \left(\frac{T_f}{T_f + T_s}\right)\bar{x}_{k-1} + \left(\frac{T_s}{T_f + T_s}\right)x_k \tag{C.14}$$

By letting, $\alpha = \left(\frac{T_f}{T_f+T_s}\right)$, we get:

$$\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)\,x_k \tag{C.15}$$

which is same as equation C.4.

This is the simplest and most efficient way of implementing a first-order low-pass filter. It has been used in several places in the VI described in Chapter 5.

Appendix D

Setting up the Test-Bed

The procedure to setup the test-bed is given below:

1. Load the appropriate code (ReadWriteTrans.hex) on the microcontroller.
2. Connect the transmitter with the microcontroller and the microcontroller to the PC.
3. Power on the transmitter and the microcontroller.
4. Use “Device Manager” under “Control Panel” on the PC to identify the Serial Port to which the microcontroller is connected.
5. Launch “LabVIEW” and load the “ReadWriteTransTrial.vi” in order to test the transmitter communication.
6. Setup the correct Serial Port parameters in the VI and run it.
7. The transmitter stick positions should be correctly read by the PC/VI. If this does not work, refer to the troubleshooting section of this appendix.

8. Close the "ReadWriteTrans.vi" if the transmitter connection is working fine.

9. Connect the Bluetooth dongle to the PC.

10. Power on the Giant MAV (along with the onboard avionics box).

11. Use the "IVT BlueSoleil" software on the PC to establish connection to the bluetooth transceiver on the Giant.

12. Launch "GiantClosedLoopControl.vi" in LabVIEW.

13. Setup the correct Serial Port parameters for the transmitter and Bluetooth communication (Fig. D.1).

14. Select the "Type of Control" and enter gain settings (Fig. D.2).

15. Select "Need to Initialize" in order to enable initialization of the Bluetooth link.

16. Run the VI and stop it after 5 seconds.

17. Check if the LED on the avionics box (on Giant) starts to blink. The blinking indicates successful start of data transmission.

18. If the LED is not blinking refer to the section on troubleshooting.

19. Un-select the "Need to Initialize" option on the VI and rerun it (Fig. D.1).

20. After about 10 seconds, the data from the sensors should be available on the charts of the VI.

21. Make sure that data from all channels of the avionics box and transmitter is within expected limits.

22. Pull the trainer switch on the transmitter to go into autonomous mode and perform the experiment.

23. After the experiment, press the “Stop” button on the VI.

24. The VI will request for the names of the files (one for sensor data and other for transmitter data) where the on-screen data is to be stored for later analysis, before terminating.

Troubleshooting

- **ReadWriteTransTrial.vi does not respond to transmitter stick movements**

  This problem is usually due to some minor mistake in the setting up of the system and is usually resolved by one of the following:

  - Make sure the transmitter and the microcontroller are powered on.
  - Check the connecting wires between the transmitter and the microcontroller, and between the microcontroller and the PC. The wires on the microcontroller should be connected to correct I/O pins.
  - Reset the microcontroller.
  - Remove the USB cable from the PC and reconnect it (this way it can reinitialize).
  - Make sure that the VI is reading from the correct Serial Port (as shown by the “Device Manager”).

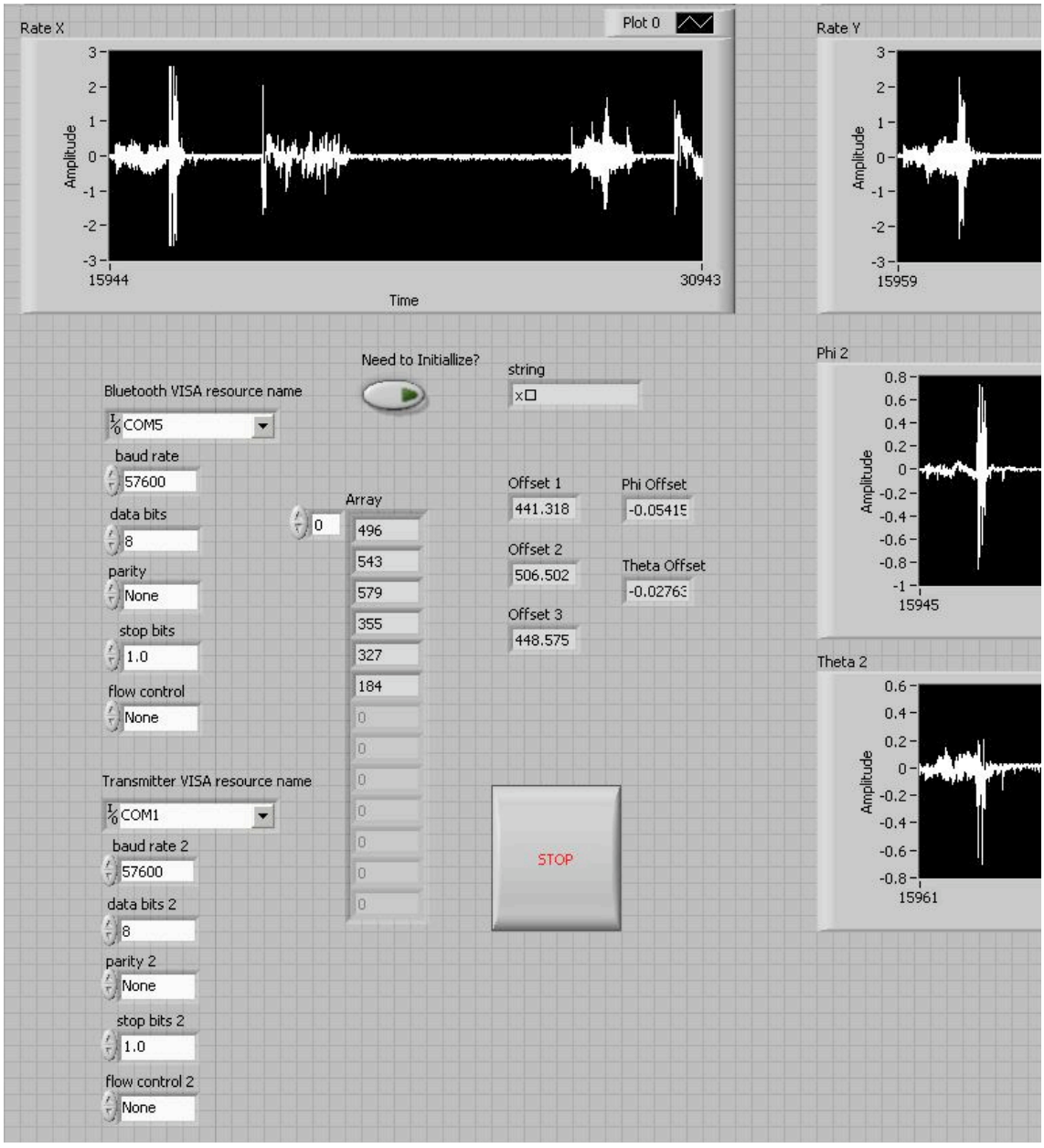

Figure D.1: Communication settings in LabVIEW

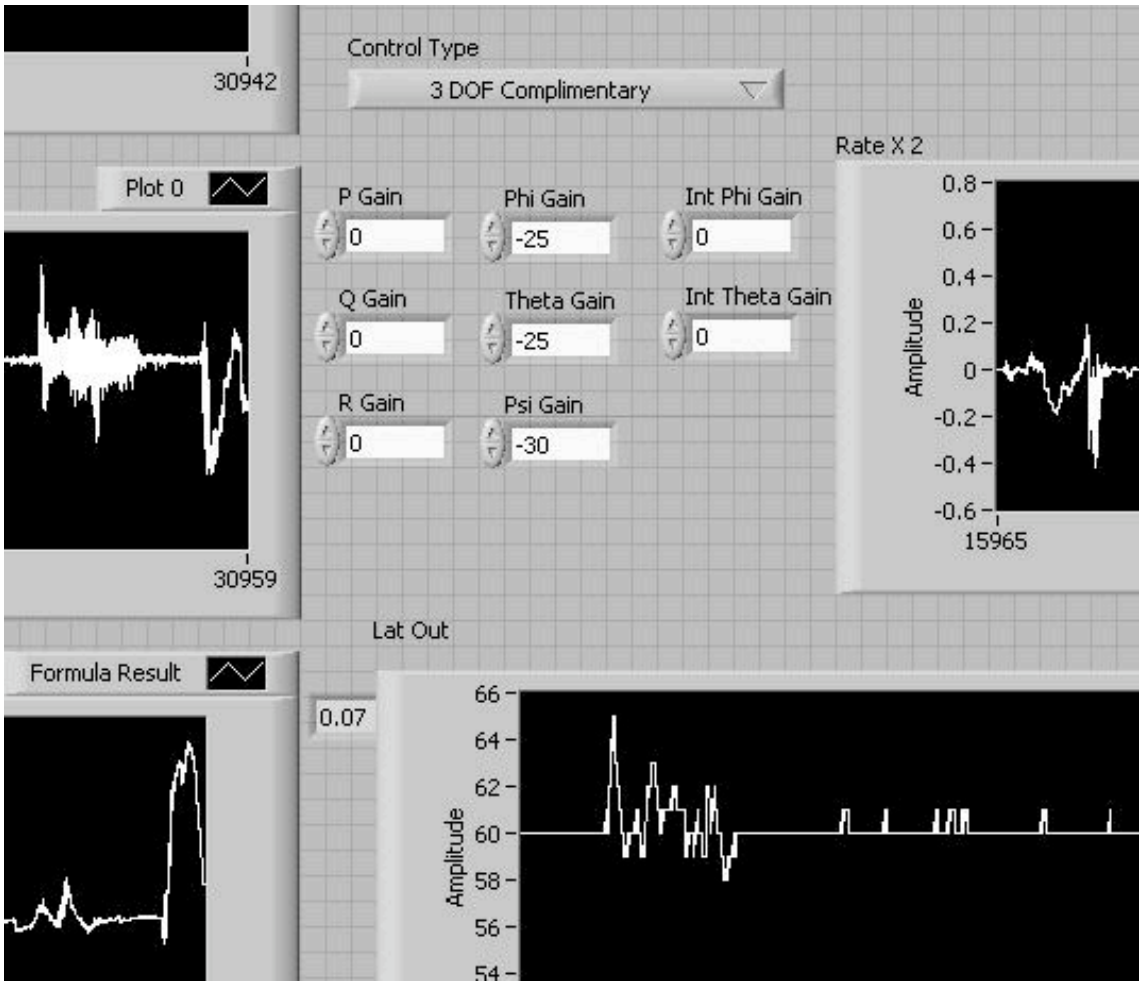

Figure D.2: Control settings in LabVIEW

  - Check the battery level of the transmitter. Recharge it, if necessary.

- **BlueSoleil software fails to connect to the Bluetooth transceiver**

  This usually happens when the onboard avionics box has already been initialized (LED is blinking) but the communication link with the PC is required to be re-established. The solution is to reset the avionics box so that first the link can be set up before it begins transmission of data.

- **LED on the avionics box is not blinking after trying to initialize it**

  The most common reason for this problem is improper link between the PC and the onboard avionics box. The solution is to reset the avionics box and re-establish link using the "IVT BlueSoleil" software on the PC.

References

[1] McMichael, J. M., Francis, M. S., "Micro Air Vehicles - Towards a New Dimension in Flight," DARPA Document, 1997.

[2] Grasmeyer, J. M., and Keennon, M. T., "Development of the Black Widow Micro Air Vehicle," AIAA Paper No. AIAA-2001-0127, Presented at the 39^{th} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2000.

[3] Anon., *Unmanned Air Vehicles Roadmap, 2002-2027,* Office of the Secretary of Defence, 2002.

[4] Anon., *Unmanned Air Vehicles Roadmap, 2005-2030,* Office of the Secretary of Defence, 2005.

[5] Sirohi, J., Tishchenko, M., and Chopra, I., "Design and Testing of A Micro-Aerial Vehicle with a Single Rotor and Turning Vanes," American Helicopter Society 61^{st}nd Annual Forum, Grapevine, TX, 1-3 June, 2005.

[6] Bohorquez, F., Samuel, P., Sirohi, J., Pines, D., Rudd, L., and Perel, R., "Design, Analysis and Performance of a Rotary Wing MAV," *Journal of the American Helicopter Society,* Vol. 48, No. 2, pp. 80-90, April, 2003.

[7] Young, L., and Aiken, E., "New Concepts and Prespectives on Micro-Rotorcraft and Small Autonomous Rotary-Wing Vehicles," AIAA-2002-2816, Presented at the 20^{th} AIAA Applied Aerodynamics Conference, St. Louis, Missouri, June 24-26, 2002.

[8] Bohorquez, F. and Pines, D., "Rotor and Airfoil Design for Efficient Rotary Wing Micro Air Vehicles," 61st Annual American Helicopter Society Forum, Grapevine, TX, June 2005.

[9] Pereira, J., and Chopra, I., "An Investigation of the Effects of Shroud Design Variables on the Hover Performance of a Shrouded-Rotor Micro Air Vehicle," American Helicopter Society International Specialists' Meeting on Unmanned Rotorcraft, January, 2005.

[10] Parsons, E., Sirohi, J., and Chopra, I., "Micro Air Vehicle; Cycloidal Rotor Concept," American Helicopter Society International Specialists' Meeting on Unmanned Rotorcraft, January, 2005.

[11] Singh, B., Ramasamy, M., Chopra, I., and Leishman, J., "Experimental Studies on Insect-Based Flapping Wings for Micro Hovering Air Vehicles," 46th Annual AIAA uctural Dynamics and Materials Conference, Austin, TX, April 2005.

[12] Newcome, L. R., *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles,* AIAA, 2004.

[13] Arning, R., and Sassen, S., "Flight Control of Micro Aerial Vehicles," AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island, Aug. 16-19, 2004.

[14] Amidi, O., Kanade, T., and Miller, J. R., "Autonomous Helicopter Research at Carnegie Mellon Robotics In stitute," Proceedings of the Heli Japan '98, April, 1998.

[15] Spraque, K., *et. al.*, "Design and Applications of an Avionics System for a Miniature Acrobatic Helicopter," The 20^{th} Digital Avionics Systems Conference, Daytona Beach, FL, Oct. 14-18, 2001

[16] Johnson, E., Schrage, D., "The Georgia Tech Unmanned Aerial Research Vehicle: GTMax," AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, Texas, Aug. 11-14, 2003.

[17] Koo, T., Hoffmann, F., Shim, H., Sinopoli, B., and Sastry, S., "Hybrid Control of an Autonomous Helicopter," IFAC Workshop on Motion Control, Grenoble, France, Sep. 21-23, 1998.

[18] Garcia-Pardo, P. J., Sukhatme, G. S., and Montgomery, J. F., "Towards vision-based safe landing for an autonomous helicopter," *Robotics and Autonomous Systems,* Vol. 38, Issue 1, January, 2002, pp. 19-29.

[19] Mettler, B., *Indentification Modeling and Characteristics of Miniature Rotorcraft,* Springer, 1^{st} edition, USA, Sep. 30, 2002.

[20] Britting, K., *Inertial Navigation Systems Analysis,* Wiley Interscience, New York, 1971.

[21] Broxmeyer, C., *Inertial Navigation Systems,* McGraw-Hill, New York, 1964.

[22] King, A. D., "Inertial Navigation - Forty Years of Evolution," GEC Review, Vol. 13, No. 3, 1998, pp. 140-149.

[23] Anon., "E-flite Blade CP," *http://en.wikipedia.org/wiki/E-flite_Blade_CP*, accessed Oct. 30, 2006.

[24] Anon., "Triaxial angular rate (gyro), acceleration and magnetic field Inertial Sensor," *http://www.memsense.com/products/mag3.asp*, accessed Oct. 30, 2006.

[25] Anon., "PIC16F877A," *http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010242*, accessed Oct. 30, 2006.

[26] Anon., "CCS, Inc. - PIC16F877A Development Kit," *http://www.ccsinfo.com/ product_info.php?products_id=16F877Akit*, accessed Oct. 30, 2006.

[27] Conroy, J. K., Samuel, P. D., and Pines, D. J., "Development of an MAV Control and Navigation System," AIAA-2005-7065, Infotech@Aerospace, Arlington, VA, Sep. 26-29, 2005.

[28] Pamadi, B. N., *Performance, Stability, Dynamics, and Control of Airplanes,* 2^{nd} edition, AIAA, Dec. 2003.

[29] Barbour, N., Schmidt, G., "Inertial Sensor Technology Trends," *IEEE Sensors Journal,* Vol. 1, No. 4, December, 2001, pp 332-339.

[30] Anon., "Analog Devices ADXRS150 - Angular Rate Sensor ADXRS150," *http://www.analog.com/en/prod/0,2877,ADXRS150,00.html*, accessed Oct. 30, 2006.

[31] Anon., "QP-2HC Datasheet," *http://www.midoriamerica.com/pdf/QP-2HC.pdf*, accessed Oct. 30, 2006.

[32] Baerveldt, Albert-Jan and Klang, R., "A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter," IEEE International Conference on Intelligent Engineering Systems, 1997, pp. 391-395.

[33] Lu, Y., Brown, A., "Performance Test Results of an Integrated GPS/MEMS Inertial Navigation Package," *Proceedings of the ION GNSS 2004,* Long Beach, California, Sep. 2004.

[34] Jun, M., Roumeliotis, S. I., Sukhatme, G. S., "State Estimation of an Autonomous Helicopter Using Kalman Filtering," *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Vol. 3, 1999, pp. 1346-1353.

[35] Anon., "Spark Fun Electronics," *http://www.sparkfun.com/commerce/product_info.php?products_id=411*, accessed Oct. 30, 2006.

[36] Anon., "MMA7260Q Product Summary Page," *http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MMA7260Q&nodeId=01126911184209*, accessed Oct. 30, 2006.

[37] Anon., "BlueSMiRF v1 Datasheet," *http://www.sparkfun.com/datasheets/RF/BlueSMiRF_v1.pdf*, accessed Oct. 30, 2006.

[38] Anon., "IMU 6-DOF v2 Datasheet," *http://www.sparkfun.com/datasheets/Sensors/IMU_6DOF-v21.pdf*, accessed Oct. 30, 2006.

[39] Anon., "LabVIEW - The Software That Powers Virtual Instrumentation," *http://www.ni.com/labview/*, accessed Oct. 30, 2006.

[40] Ogata, K., *Modern Control Engineering,* Prentice-Hall Inc., New Jersey, 2002.

[41] Stengel, R. F., "Toward Intelligent Flight Control," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 23, no. 6, 1993, pp. 1699-1717.

[42] Ananda, M. P., Bernstein, H., Cunningham, K. E., Feess, W. A., and Stroud, E. G., "Global Positioning System (GPS) Autonomous Navigation," IEEE Position Location and Navigation Symposium, Las Vegas, NV, Mar. 20-23, 1990.

[43] Amidi, O., Kanade, T., and Miller, R., "Vision-based Autonomous Helicopter Research at Carnegie Mellon Robotics Insitute 1991-1997," American Helicopter Society International Conference, Japan, April, 1998.

[44] Barrows, G. L., Chahl, J. S., and Srinivasan, M. V., "Biomimetic Visual Sensing and Flight Control," *The Aeronautical Journal*, Vol. 107, No. 1069, pp. 159-168, 2003.

[45] Wu, H., Sun, D., Zhou, Z., "Model Identification of a Micro Air Vehicle in Loitering Flight Based on Attitude Performance Evaluation," *IEEE Transactions on Robotics*, Vol. 20, No. 4, pp. 702-712, Aug. 2004.

[46] Shim, D. H., Koo, T. J., Hoffmann, F., and Sastry, S., "A Comprehensive Study of Control Design for an Autonomous Helicopter," 37th IEEE Conference on Decision and Control, 1998.

[47] Smith, S. W., *The Scientist and Engineer's Guide to Digital Signal Processing,* California Technical Pub., 1st edition, 1997.

[48] Geen, J, Krakauer, D., "New iMEMS Angular Rate-Sensing Gyroscope," *Analog Dialogue*, Vol. 37, No. 3, 2003 (available online: *http://www.analog.com/library/analogdialogue/archives/37-03/gyro.pdf*).

[49] Xiyuan, C., "Modeling Random Gyro Drift by Time Series Neural Networks and by Traditional Method," IEEE Int. Conf. on Neural Networks and Signal Processing, Nanjing, China, Dec. 14-17, 2003.

[50] Anon., "The MathWorks - Filter Design toolbox," *http://www.mathworks.com/products/filterdesign/*, accessed Oct. 30, 2006.