Three Degree-of-Freedom Simulator Motion Cueing Using Classical Washout Filters and Acceleration Feedback

C. Jason Gutridge

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute & State University
In partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Dr. Wayne Durham – Chair
Dr. Craig Woolsey
Dr. Naira Hovakimyan

April 27, 2004
Blacksburg, Virginia

Keywords: Motion Cueing, Washout Filters, System Identification, Feedback Controller

Three Degree-of-Freedom Simulator Motion Cueing Using Classical Washout Filters and Acceleration Feedback

C. Jason Gutridge

(ABSTRACT)

Good motion cueing in a flight simulator serves to enhance the overall simulation environment. However, poor motion cueing can greatly detract from the simulation and serve solely to distract the pilot. The latter was the case for Virginia Tech's three degree-of-freedom motion-base.

The most common method of motion cueing is to use washout filters to produce the best motion cues within the physical limitations of the motion system. This algorithm is named the classical washout algorithm and its filters were studied first in this research, but initially yielded undesirable results. In efforts to greatly improve the acceleration response in the pitch axis, the concept of an acceleration feedback controller in conjunction with washout filters was investigated.

In developing a mathematical model of the motion-base and its corresponding circuitry, corrections and modifications were made to the circuitry which served to improve the dynamic response of the motion-base and enhance motion sensations. Next, design and implementation of the acceleration feedback controller for the pitch axis was performed and tested using a pilot rating scale and time history responses. The parameters for the acceleration feedback algorithm and the classical washout algorithm were varied to find the most favorable algorithm and set of parameters.

Results of this paper have demonstrated the successful implementation of acceleration feedback and that the motion system at Virginia Tech now serves to greatly enhance the simulation environment.

Acknowledgments

My deepest thanks go out to my parents, who have instilled in me all the great qualities they possess. Thank you for your continual encouragement and guidance in all of my endeavors and all the support you have given me in the past, for without it none of this would have been possible. I thank you for giving me the freedom and drive to pursue and strive to be the best at whatever I desire. To my brother Bobby, your support and advice have been the essence of what a big brother represents. With the successful path you followed, I was inspired to achieve what many thought impossible. To Erin, your peppy attitude and continual encouragement have lifted my spirits in the roughest times.

I would like to also thank my advisor, Dr. Wayne Durham, for his support and input to this project and my graduate career at Virginia Tech. I have gained a great deal of knowledge from his experiences in the aerospace field, and I now consider him a good friend. To the remaining members of my committee, Dr. Craig Woolsey and Dr. Naira Hovakimyan, thank you for spending the time to help me with this project. Your knowledge and advice are very much appreciated. To Bill Oetjens, thank you for your enormous amount of help with my project and for being my guinea pig for the countless number of tests I had to perform.

Thanks also go out to my many friends who make my life much more enjoyable. Thank you for all the great memories; you truly made my time at Virginia Tech unforgettable.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms:

3DOF	Three Degree-of-Freedom
6DOF	Six Degree-of-Freedom
AF1	Acceleration Feedback algorithm with one in-loop HP filter
AF2	Acceleration Feedback algorithm with two in- and outer-loop HP filters
CASTLE	Control Analysis and Simulation Test Loop Environment
CW	Classical Washout algorithm
MANGEN	Maneuver Generator
OFT	Operational Flight Trainer
PID	Proportional, Integral, and Derivative Controller
SISO	Single-Input Single-Output

Variables:

$a_{accelLF}$	Low-pass filtered accelerometer measurements
a_{BIAS}	Accelerometer voltage bias
a_{raw}	Raw accelerometer measurements
a_{sensed}	Simulator z-axis acceleration sensed at the pilot station
a_x	Cockpit body x-axis acceleration
a_{xLF}	Low-pass filtered body x-axis accelerations
a_y	Cockpit body y-axis acceleration

a_{yHF}	High-pass filtered body *y*-axis accelerations
a_{yLF}	Low-pass filtered body *y*-axis accelerations
a_z	Cockpit body *z*-axis acceleration
a_{zHF}	High-pass filtered body *z*-axis accelerations
a_{zOL}	Acceleration/Error input into the feedback loop
e	Error for proportional gain
f_{AA}	Specific aircraft forces
g	Acceleration due to gravity
G_{PID}	PID control law
G_{XX}	System identification input auto-spectrum
G_{XY}	System identification input-output cross-spectrum
H_{HPx}	High-pass filter *x*-axis transfer function
H_{HPz}	High-pass filter *z*-axis transfer function
$H_{LP\text{-}ACC}$	Low-pass filter for accelerometer noise
H_{XY}	Motion system transfer function
K_D	Derivative feedback gain
K_I	Integral feedback gain
K_P	Proportional feedback gain
L_{IB}	Body to inertial rotational matrix
M	Magnitude of complex pair
p	Body-axis roll rate
p_{HF}	High-pass filtered body-axis roll rate
p_{volts}	Roll actuator command in volts
q	Body-axis pitch rate
$\bar{q}$	Dynamic Pressure
$\dot{q}_{HF}$	Pitch angular acceleration commanded
r	Body-axis yaw rate
$\dot{r}_{HF}$	Yaw angular acceleration commanded
s	Lapalcian operator
Sc_1	Initial scaling for similar variable comparison in error production

Sc_2	Scaling factor for classical washout
S_I	Translational motion command
T_B	Body to inertial angular rotation matrix
u	State-Space input variable
x	State-Space state variables
X	System identification input spectrum
X_{BEAM}	Distance from gimbal to pilot station
y	State-Space output variable
y_{volts}	Yaw actuator command in volts
Y	System identification output spectrum
z_{volts}	Pitch actuator command in volts
β_B	Tilt-coordination angle
β_S	Angular motion command
ϕ	Phase angle of complex pair
ϕ_{LF}	Tilt-coordination roll angle
ϕ_{MOT}	Commanded roll motion
Ψ_{MOT}	Commanded yaw motion
σ	Real part of complex pair
τ	General time constant
τ_{IL}	In-loop high-pass filter time constant
τ_{OL}	Out-of-loop high-pass filter time constant
θ_{ACT}	Actual pitch angle of cockpit
θ_{LF}	Tilt-coordination pitch angle
θ_{MOT}	Commanded pitch motion
ω	Imaginary part of complex pair
ω_{AA}	Aircraft angular rates
ω_c	Origin frequency for asymptotic approximations
ζ	Damping ratio

Chapter 1

Introduction

1.1 Motion Simulation

Flight simulators are composed of several subsystems that contribute to the overall replication of the senses felt in an aircraft. The subsystem of interest in this study is the motion-base, which serves to move the simulator cockpit in a manner that produces similar acceleration sensations to that of an aircraft. It is widely recognized that a properly cued motion-base can greatly enhance the simulation, while a motion-base with incorrect cues can be more detrimental than no motion at all [1].

One of the main ingredients to realistic motion is the software that transforms aircraft accelerations into simulator motion cues. This software typically uses a washout algorithm that is responsible for calculating the best commanded motion without attempting to extend the motion-base beyond its physical limitations. The purpose of this study is to analyze the washout filters for the pitch-axis of the three degree-of-freedom (3DOF) motion system and to investigate the use of feedback control in conjunction with washout filters.

1.2 Motion System

In March of 1996, the Department of Aerospace and Ocean Engineering at Virginia Tech acquired a three degree-of-freedom motion-based A-6E device 2F122A Operational Flight Trainer (OFT) from Oceana Naval Air Station in Virginia Beach, VA. During its service life with the Navy, the 2F122A OFT, shown in figure 1.1, had been primarily used for night time carrier landing training. Because of the small maneuvers experienced during a carrier approach, a 3DOF motion system like that of the 2F122A was used rather than the more common and complex six degree-of-freedom (6DOF) motion systems.

Figure 1.1 – Virginia Tech's 2F122A Flight Simulator

The OFT's motion cues are produced by moving the cockpit through three independent angular degrees-of-freedom around a gimbal located approximately twelve feet behind the pilot station. A photograph of the motion-base along with its schematic is shown in figures 1.2 and 1.3 [2]. The cockpit is rigidly fastened to a cantilever beam whose position is controlled by a series of hydraulic actuators. The pitch-axis is operated by a single hydraulic actuator that serves to raise and lower the cockpit as shown in figure 1.4. The roll and yaw axes are each operated by two hydraulic actuators that act in opposite directions about a moment arm to rotate the cockpit

about the desired axis. Due to actuator position limits, the cockpit may only be moved +20 and -10 degrees for pitch, +/- 20 degrees for yaw, and +/- 20 degrees for roll.

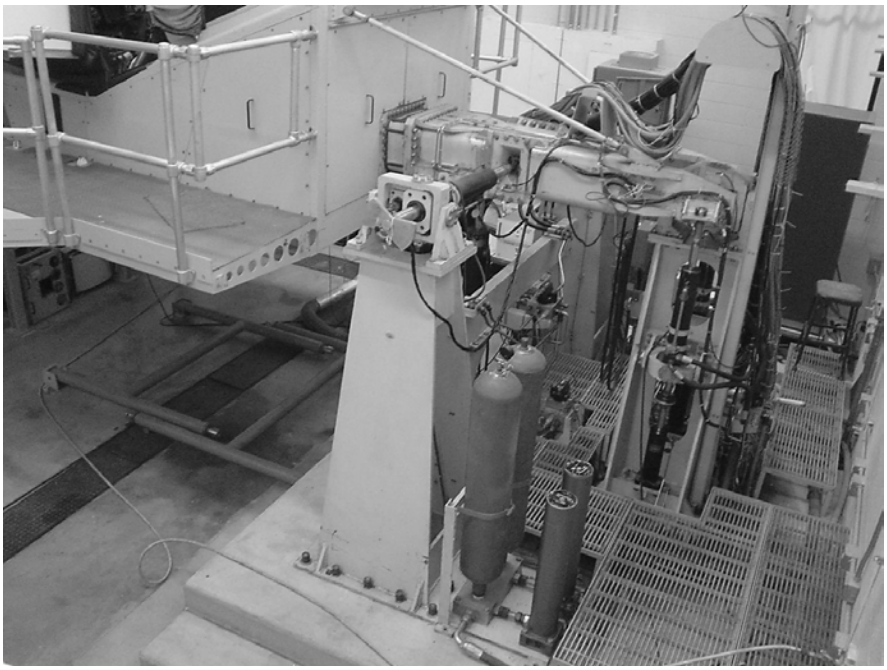

Figure 1.2 – The 2F122A's Motion-Base

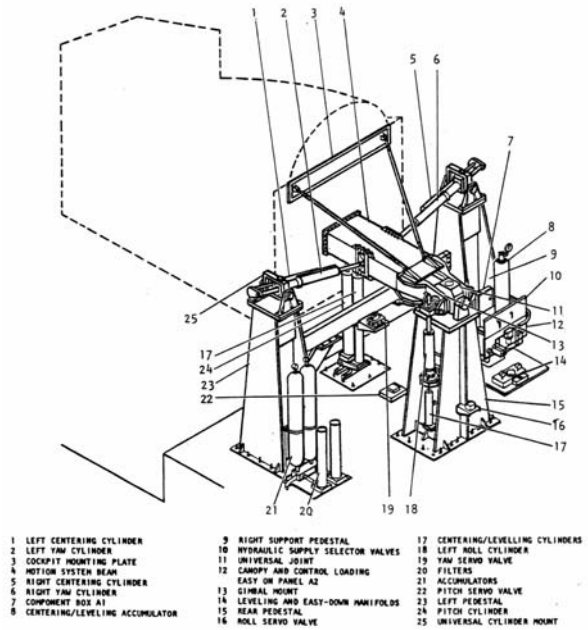

Figure 1.3 – Motion-Base Schematic

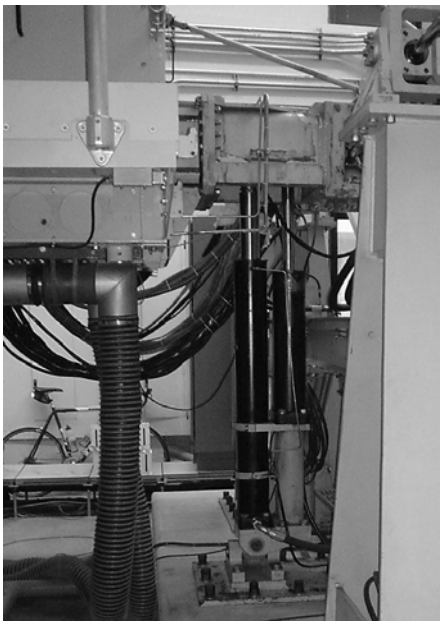

Figure 1.4 – Pitch-Axis Hydraulic Actuator

The normal method of operation of the motion system is through cockpit commands (e.g. stick deflection, throttle, etc.) which are converted into motion cues using the simulation software and washout algorithms. This method of operation is used during normal simulation and is run under *CASTLE* (Control Analysis and Simulation Test Loop Environment), a simulation environment developed by the Naval Air Warfare Center's Manned Flight Simulator branch in Patuxent River, Maryland. In terms of motion cueing, *CASTLE* first calculates cockpit aircraft accelerations from pilot inputs via the equations of motion. It then uses the washout algorithm to calculate the actuator position commands that theoretically simulate aircraft accelerations within the constraints of the motion system. The actuator position commands are then sent out of the computer in the form of analog outputs (AO) that serve as input to the servo cards. These cards serve to close two built-in feedback loops of pressure and actuator position that control the dynamic response of the motion system. The signal is then amplified and sent to corresponding control valves responsible for directing hydraulic fluid to the appropriate cylinder.

1.3 Research Objectives

Since the simulator's inception at Virginia Tech and prior to this research, the motion sensations by the 2F122A OFT had been deemed distracting, poor, and unrealistic. The unrealistic motion cues may be attributed to several different changes in the motion system dynamics from the simulator's inception at Virginia Tech. First, due to noticed cracks in the welds that hold the cockpit to the cantilever motion beam, structural reinforcements were added in the form of steel plates. Second, an additional 500 lb visual monitor was incorporated into the simulator to allow the pilot to have a side-view. These two changes altered the mass and stiffness of the motion system structure. Finally, re-hosting the system from the original Gould 32/67 to a Silicon Graphics Origin 2000 may have been a cause of the poor motion cues. These modifications effectively changed the dynamic modes inherent in the system that the original motion system was designed to control. It is believed that the modifications above were the sources of the unrealistic motion cues experienced.

The purpose of this research was to investigate the use of the classical washout algorithm with and without an outer-loop of acceleration feedback. The main function of the classical washout algorithm is to high-pass filter the accelerations such that the motion system will not extend beyond its inherent limits and will washout to its neutral point. The acceleration feedback algorithm uses the classical washout filters, but also measures the accelerations at the cockpit, feeds these values back, and compares the measured accelerations with the aircraft accelerations. This error, multiplied by a feedback gain, represents the new acceleration input to the algorithm.

Chapter 2 discusses the classical washout algorithm in detail and its implementation into the simulation environment. Acceleration feedback and the steps to designing the controller are also introduced.

Chapter 3 represents the first step in designing the feedback controller: system identification. This chapter presents the steps in identifying the system transfer function between accelerations sensed in the simulator cockpit (output) to commanded hydraulic piston position (input) from the computer. This transfer function represents the servo card circuitry with its inherent inner feedback loops, the servo valves, the hydraulic pistons, and the cockpit/beam structural dynamics. This chapter also presents the modification of the inner feedback loops

inherent in the servo cards to obtain the best dynamic response possible without the aid of acceleration feedback.

Chapter 4 discusses the design of the feedback controller and the methods used to determine its theoretical feedback gains. The implementation is also presented and experimentally discovered issues are discussed.

Chapter 5 explains and presents the study of finding which set of parameters allows for maximum performance of the given algorithm. The method for finding the most favorable algorithm is discussed. Gain scheduling is also studied and implemented into the algorithm of choice.

Finally, Chapter 6 summarizes the work and draws conclusions discovered during this research.

Chapter 2

Motion-Drive Algorithms

2.1 Introduction

A critical step in the generation of accurate motion cues is the software that transforms aircraft accelerations into commanded simulator motion cues. The software that performs the necessary computations to complete this task is commonly called the washout algorithm [3]. The goal of this software is to provide the pilot with the best motion cues within the physical limitations of the motion-base, throughout a specified flight envelope. Before acceleration feedback was analyzed and implemented, the classical washout algorithm was studied and is the subject of this chapter.

This chapter deals with the implementation of the Toronto Institute for Aerospace Studies classical washout algorithm as reported by Reid and Nahon [4, 5, 6] into Virginia Tech's flight simulator. This algorithm was chosen because it is mathematically simple, and is transparent to the user, and therefore pilot complaints can often be alleviated by changing parameter values [3].

The algorithm itself was altered to compensate for the fact that the aforementioned study was performed on a 6DOF motion system, rather than a 3DOF motion system that Virginia Tech's simulator employs.

2.2 Classical Washout Algorithm

The classical washout algorithm is the most common motion-drive algorithm used in simulators [3]. Because of such constraints as physical limits, and actuator rate limits, filtering between the aircraft accelerations and the calculated motion cues is essential. These washout filters serve to attenuate the low frequency accelerations that cause the motion-base to reach its limitations [7], while keeping high frequency accelerations unchanged. The high frequency accelerations last for a small duration of time, and thus will not drive the motion-base to its physical limits if a proper high-pass filter break frequency is chosen. The filters also serve to bring the motion-base back to its zero reference point, hence the term "washout" filter.

Low-pass filters are also used to reproduce the low frequency accelerations sensed in an aircraft. This method is called tilt-coordination, and utilizes the gravity vector by tilting the cockpit to a commanded angle such that sustained accelerations are simulated by gravity.

Reid and Nahon's 6DOF classical washout algorithm is shown in figure 2.1 [3] for a translational and angular degree-of-freedom. This algorithm will be described first for the 6DOF case as done in the report referenced. Following this, an explanation of how this algorithm was adapted to the 3DOF cantilevered beam motion system will be given.

The inputs to the 6DOF algorithm shown in figure 2.1 are the three body-axis accelerations, a_x, a_y, and a_z-g (represented as f_{AA}), along with the three aircraft angular rates, p, q, and r (represented as ω_{AA}). The specific forces are then scaled down to a desired fraction of the actual motion due to the physical limitations of the system. The forces are then transformed from the body-axis to the inertial-axis using the Euler angle transformation matrix. These inertial specific forces are high-pass filtered to obtain the simulator translational accelerations. The accelerations are then integrated twice in order to transform commanded accelerations to commanded actuator positions [4].

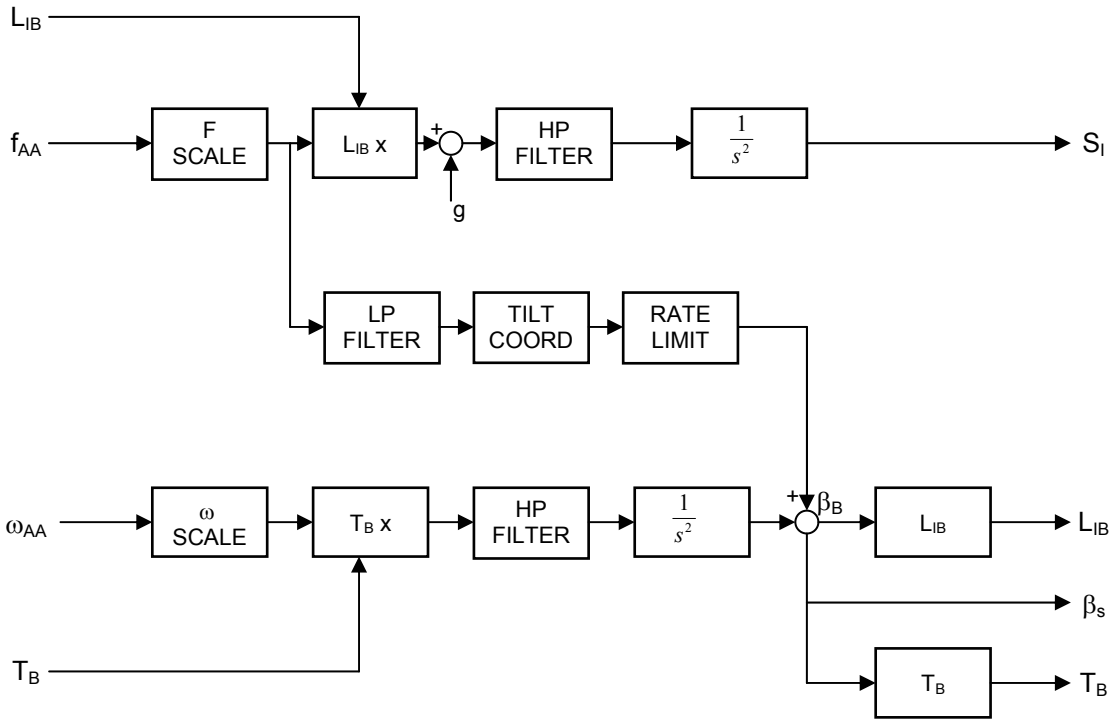

Figure 2.1 – Reid and Nahon Classical Algorithm

The forces are also passed to low-pass filters, scaled, and rate limited to create pitch and roll tilt-angles. This concept is called tilt-coordination, and its objective is to orient the gravity vector such that the pilot feels a sustained acceleration while the visual display remains constant. For instance, in order to feel translational acceleration in the body-axis *x*-direction, or out the nose of the aircraft, the simulator cockpit will tilt nose up such that the gravity vector will allow the pilot to feel acceleration in the simulator's *x*-axis (i.e. a force pushing on the pilot's back). As long as the cockpit does not tilt up too far the pilot will not notice the decrease in the normal acceleration in the *z*-direction due to tilting. This is because the threshold value of acceleration, or the minimum acceleration a pilot can sense, will not be reached. The threshold values are obtained from pilot tests and range from 0.17 to 0.28 m/s^2 for linear accelerations [8]. Using these threshold values, a tilt of more than ten degrees may cause a noticeable difference in normal acceleration.

Similar to the specific forces, the angular motion is high-pass filtered to obtain the high frequency component of cockpit angular motion. These angular rates are then integrated to find the motion angle and added to the low frequency angular motion due to tilt-coordination.

2.3 Three Degree-of-Freedom Adaptation

Because the motion-base used in this study was a 3DOF cantilevered motion-base, modifications were necessary to implement the classical washout algorithm. The inputs used in this code consisted of three accelerations at the pilot's station a_x, a_y, a_z, and one angular roll rate p. The outputs were hydraulic actuator positions for pitch, roll, and yaw. Figure 2.2 represents the entire algorithm schematic.

The z-direction aircraft pilot station acceleration, a_z, is first scaled by a factor of 0.5, and then fed into a first order high-pass filter with a break frequency of 6.66 rad/s. These scaling factors and break frequencies were derived from those used in Reid and Nahon's 6DOF study [3]. The output of the high-pass filter is then transformed into an angular acceleration through equation 2.1, where a_{zHF} represents the high frequency accelerations, X_{BEAM} represents the length from the motion pivot point to the pilot station, and ϕ_{MOT} is the motion roll position. The cosine of this angle transforms the z-direction body accelerations to z-direction inertial accelerations. The angular accelerations are then integrated twice to obtain the high frequency output of pitch position.

$$\dot{q}_{HF} = \frac{-(180/\pi)a_{zHF}}{X_{BEAM}\cos(\phi_{MOT})} \tag{2.1}$$

The x-direction aircraft pilot station acceleration, a_x, is used for simulating low frequency accelerations in this direction by using the tilt-coordination mechanism. This acceleration is scaled by a factor of 0.4 and fed into a low-pass filter with a break frequency of 0.2 rad/s. Equation 2.2 is used to transform the low frequency acceleration into a tilt-angle of pitch position. Here, a_{xLF} is the low frequency x-direction accelerations and g represents gravity. This equation uses the gravity vector to simulate sustained accelerations in the x-direction. It must also be noted that a rate limit of 3 deg/s was implemented. This allows the tilt-coordination to seem realistic without the pilot noticing the actual cockpit movement. The low frequency pitch position is then added to the high frequency pitch position from a_z, converted to volts with a digital-to-analog converter, and finally sent to the servo cards.

$$\theta_{LF} = \sin^{-1}\left(\frac{a_{xLF}}{g}\right) \tag{2.2}$$

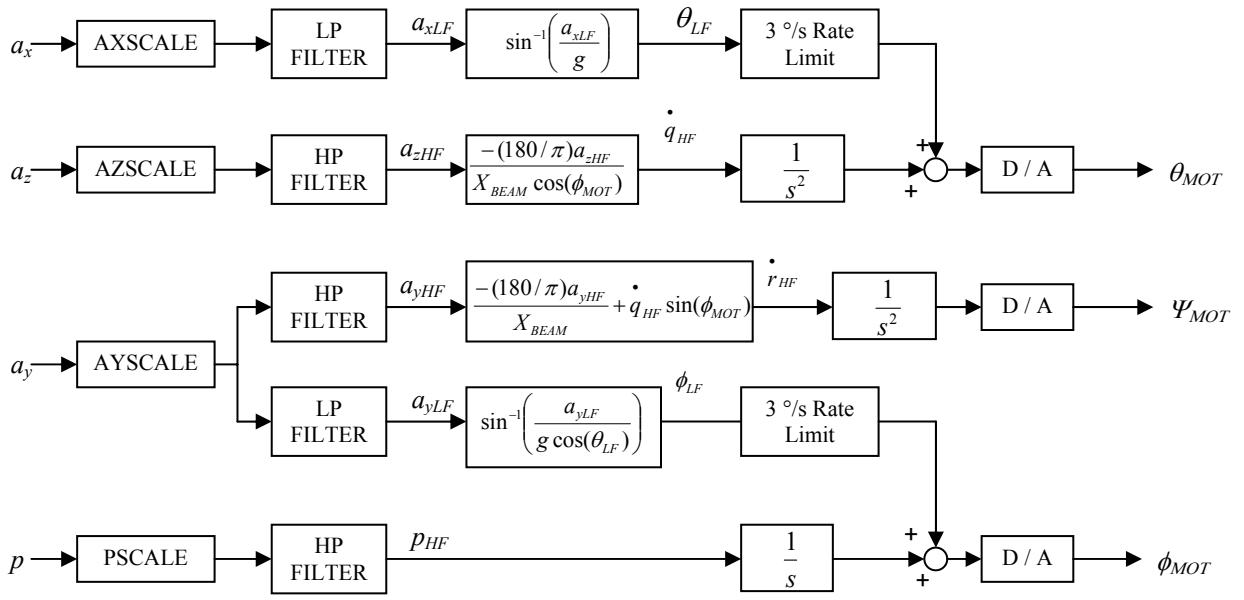

Figure 2.2 – 3DOF Motion Algorithm Schematic

The *y*-direction aircraft pilot station acceleration, a_y, is the only input that serves two functions of (1) tilt-coordination for motion roll position and (2) high frequency *y*-direction accelerations for motion yaw position. First, the *y*-direction acceleration is scaled by a factor of 0.1. The large attenuation of yaw accelerations was due to the poor structural characteristics inherent in the yaw-axis. The poor structural characteristics, described more in the following chapter, caused the cockpit to have high frequency oscillations with poor damping after a roll or yaw maneuver had ceased. This was not considered a major problem due to the fact that very small yawing maneuvers are performed in common flight.

After scaling, the accelerations are high-pass filtered with a break frequency of 1 rad/s. Equation 2.3 calculates the yaw acceleration given the high frequency *y*-direction accelerations, a_{yHF}, the distance from the motion pivot to the pilot station, X_{BEAM}, the angular accelerations from the pitch-axis, $\dot{q}$, and the motion roll position, ϕ_{MOT}. The first term in the equation represents the angular acceleration in the yaw direction, while the second term represents the component of pitch angular acceleration in the inertial *y*-direction. Yaw acceleration is then integrated twice, converted from digital to analog, and sent to the servo cards.

$$\dot{r}_{HF} = \frac{-(180/\pi)a_{yHF}}{X_{BEAM}} + \dot{q}_{HF}\sin(\phi_{MOT}) \tag{2.3}$$

The low frequency component of *y*-direction acceleration that is sustainable through tilt-coordination is sent to the roll position. It is calculated by scaling the a_y accelerations by a factor of 0.1, low-pass filtering with a break frequency of 0.2 rad/s, and performing the calculation in equation 2.4. In this equation a_{yLF} is the low frequency *y*-direction accelerations, g is gravity, and θ_{LF} is the low frequency motion pitch position. The low frequency roll position is also rate limited such that 3 deg/s was the fastest the tilt-coordination moved the motion system. The final low frequency roll position is added to the high frequency roll positions described in the following paragraph.

$$\phi_{LF} = \sin^{-1}\left(\frac{a_{yLF}}{g\cos(\theta_{LF})}\right) \tag{2.4}$$

Finally, the roll rate, p, is input to incorporate high frequency roll maneuvers. The roll rate was scaled by a factor of 0.15 and then high-pass filtered with a break frequency of 1 rad/s. Similar to the yaw-axis, the roll rate is heavily attenuated due to poor structural characteristics. The high frequency roll rate is then integrated once to obtain roll position. This signal was then added to the aforementioned low frequency motion roll position, converted from digital to analog, and then sent to the servo cards.

Most parameter values were chosen based first on the Reid and Nahon's 6DOF study [3]. Some of the scaling factors produced an excess or insufficient amount of motion, and as a result, were scaled accordingly.

2.4 Numerical Implementation

The algorithm described above was implemented into the *CASTLE* environment. The FORTRAN file with the classical washout algorithm appears in Appendix A with all flags for acceleration feedback turned off. The following are some numerical methods used in the code.

2.4.1 High-Pass Filtering

The first type of filter used was the first order high-pass filter. The purpose of this type of filter was to attenuate all low frequency accelerations that tend to drive the motion-base to its physical limits, while keeping the high frequency accelerations unchanged. Equations 2.5 – 2.11 describe

how this type of filter's numerical implementation was derived for the *z*-direction accelerations using the simple lead equations [9], while equations 2.12 and 2.13 show the actual implementation. Here *x* is the state variable, *u* is the input, or scaled aircraft accelerations, and *y* is the output of measured accelerations.

$$\dot{x} = \frac{u - x}{\tau} \tag{2.5}$$

$$y = u - x \tag{2.6}$$

Equation 2.5 can be derived by using the lead transfer function and equation 2.6 as described in equations 2.7 through 2.11.

$$H_{HPz}(s) = \frac{a_{zHF}}{a_z} = \frac{s\tau}{1 + s\tau} \tag{2.7}$$

$$a_{zHF} = a_z\left(1 - \frac{1}{1 + s\tau}\right) \tag{2.8}$$

$$a_{zHF} = a_z - x \tag{2.9}$$

$$x = \frac{a_z}{1 + s\tau} \Rightarrow x(1 + s\tau) = a_z \tag{2.10}$$

$$sx = \frac{1}{\tau}(a_z - x) \Rightarrow \dot{x} = \frac{a_z - x}{\tau} \tag{2.11}$$

Figure 2.3 shows the Bode plot of the high-pass filter with break frequency of 3.33 rad/s. The attenuation of low frequency accelerations is apparent in the magnitude plot. Figure 2.4 shows a sample of how this filter numerically affects *z*-direction accelerations.

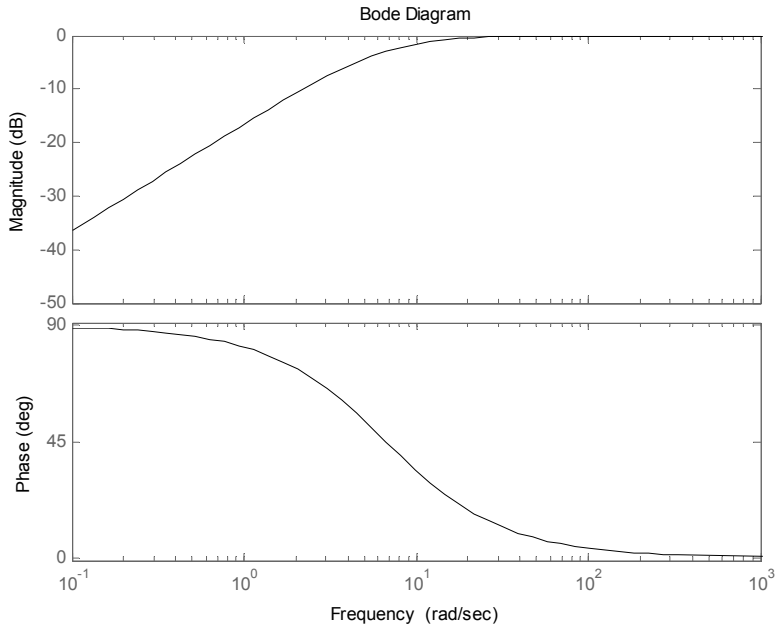

Figure 2.3 – High-Pass Filter Bode Plot

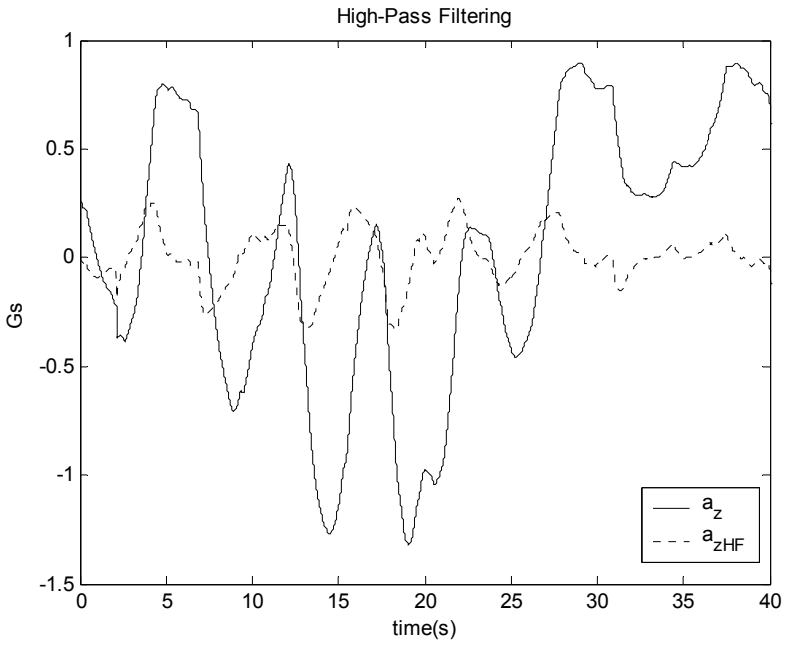

Figure 2.4 – High-Pass Filtering of Sample Run

2.4.2 Low-Pass Filtering

The second type of filter used was the first order low-pass filter used during tilt-coordination. The purpose of this type of filter was to attenuate all high frequency accelerations and leave the low frequency accelerations unchanged. Equations 2.12 – 2.17 describe how this type of filter was numerically implemented for the *x*-direction accelerations using the simple lag transfer function [9].

$$\dot{x} = \frac{u - x}{\tau} \tag{2.12}$$

$$y = x \tag{2.13}$$

$$H_{LPx}(s) = \frac{a_{xLF}}{a_x} = \frac{1}{1 + s\tau} \tag{2.14}$$

$$a_{xLF}(1 + s\tau) = a_x \tag{2.15}$$

$$a_{xLF} + \dot{a}_{xLF}\,\tau = a_x \tag{2.16}$$

$$\dot{a}_{xLF} = \frac{(a_x - a_{xLF})}{\tau} \tag{2.17}$$

Equation 2.17 was then numerically integrated using the two-step Adams-Bashforth integration method. The a_{xLF} term on the right hand side represents the previous value calculated, in which an initial value was selected to begin the simulation. Here τ may be input from the user, but a time constant of five seconds was chosen based on the Reid and Nahon 6DOF study [3].

Figure 2.5 shows the Bode plot of the low-pass filter with a break frequency of 0.2 rad/s. The drop-off in the magnitude shows that the first order filter will slowly attenuate the higher frequencies, while leaving the lower frequencies unchanged. This becomes more evident in figure 2.6, where time-domain filtered and unfiltered data are shown for a sample run.

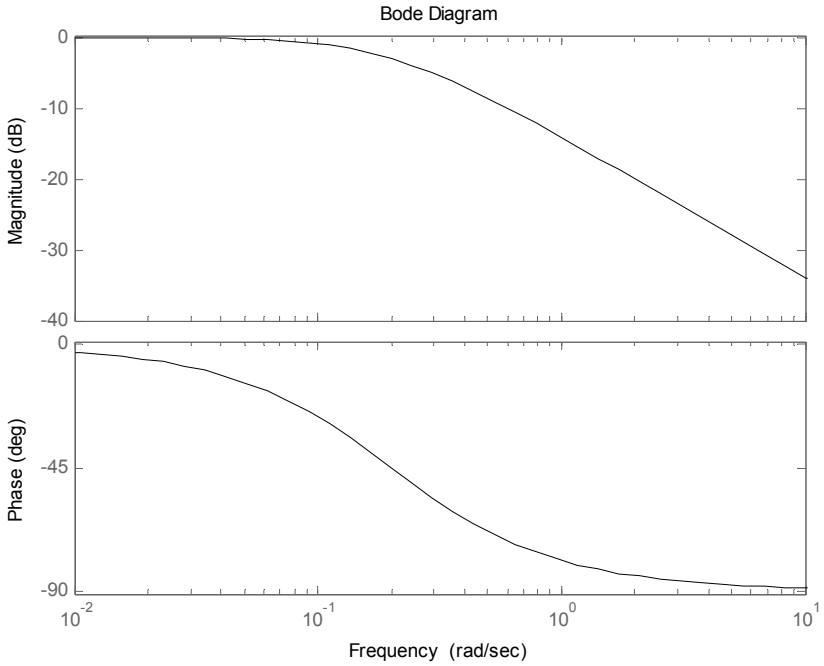

Figure 2.5 Low-Pass Filter Bode Plot

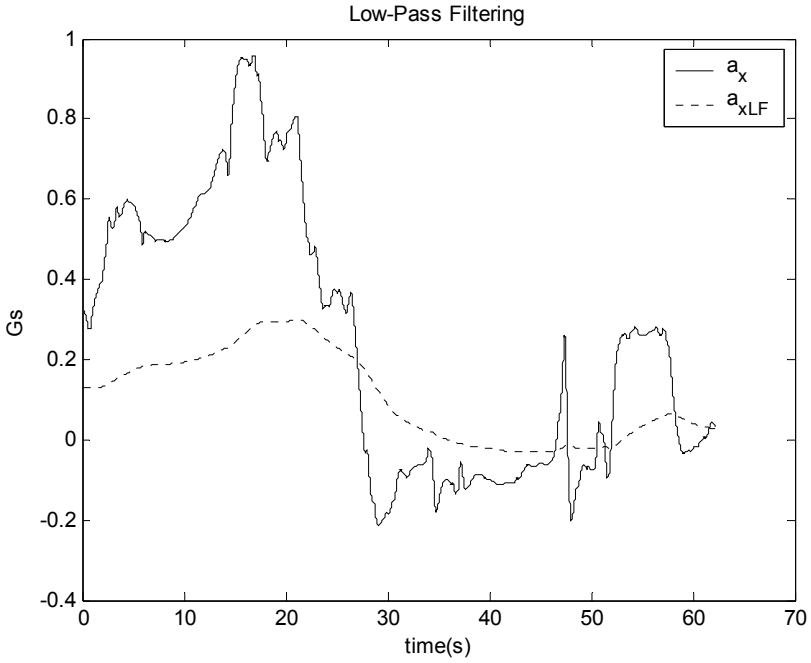

Figure 2.6 – Low-Pass Filtering of Sample Run

2.5 Acceleration Feedback

After the algorithm was implemented and tested, pilot complaints and poor time history correlation of scaled aircraft accelerations and measured accelerations in the *z*-direction led to the exploration of different options. Because the algorithm implementation seemed to be correct, the system characteristics (servo cards, hydraulics, and structural dynamics) were suspected as having poor motion system dynamics. In an attempt to better the motion cueing, servo card adjustments were made and are discussed in the following chapter. Along with these physical adjustments, a new hybrid algorithm was also investigated. Due to the availability of an accelerometer for our time history comparison and the wide use of the concept in industry control applications, a PID controller using acceleration feedback was investigated for the *z*-axis. The subsequent chapters deal with the theoretical design, implementation, and testing of this algorithm. After implementation, the algorithm was then compared with the classical washout algorithm after the aforementioned servo card modifications were performed.

The acceleration feedback algorithm was designed as a hybrid using the classical washout algorithm with an outer loop of acceleration feedback. Figure 2.7 shows a basic diagram of the algorithm for the *z*-direction. The PID controller block consists of proportional, integral, and derivative gains, and the washout filters block represents the classical washout filters discussed previously. The last block represents the transfer function between commanded actuator positions (input to the servo cards) to sensed accelerations at the cockpit. This transfer function includes the servo card circuitry, actuator, and structural dynamics inherent in the motion-base. In order to theoretically analyze this system such that desirable feedback gains could be selected, this transfer function must first be approximated via system identification. This analysis is the subject of the following chapter.

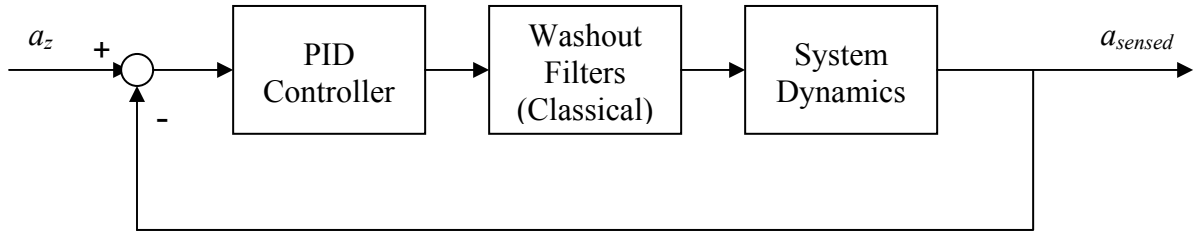

Figure 2.7 – General Acceleration Feedback Algorithm

Chapter 3

System Identification with Servo Card Modifications

3.1 Introduction

When acceleration feedback was chosen as an algorithm to investigate, it was evident that a mathematical model of the hardware and structural dynamics was needed in order to perform the theoretical analysis necessary for controller design. The algorithms in the motion code could be modeled in the Laplace domain as separate transfer functions, but there was a critical unknown transfer function with input of motion position in volts (sent from the motion algorithm) and output of acceleration at the pilot's station (accelerometer readings). This transfer function includes the dynamics of the servo card circuitry that contains inner position and pressure feedback loops, the hydraulic actuator, and the structural dynamics of the beam and cockpit. Figure 2.7 shows the block diagram with the unknown block being labeled "System Dynamics". In order to select theoretical gains using root locus theory, a complete model representing the motion code's classical washout algorithm and the motion system's physics must be available.

The method for identifying the unknown transfer function was based on the asymptotic approximation of Bode plots given experimental frequency response data [10]. Given this data, a Bode plot was constructed and asymptotes were fit to construct an approximate transfer function

of the motion system's dynamics. In order to record the frequency response that creates the Bode plot, a digital signal analyzer was used.

3.2 Servo Card Modifications

Before the experimental data was gathered, it was imperative to understand the hardware and the processes that take place between output of pitch position from the motion code and the actual hydraulic piston position.

After the motion code calculates a commanded piston position, a voltage is sent as an analog signal to the servoamplifiers in the servo cards. These servo cards are almost identical for all axes, each having a command input in the form of DC voltage from the computer (analog outputs). There are two inner-loop feedback channels sent to the amplifier: a position feedback from the angular position potentiometer located on the back of the motion beam pivot, and two pressure signals from transducers measuring hydraulic cylinder pressure. The position feedback serves to stop the command from the computer as the beam reaches the desired position. The pressure feedback loop acts to dampen the accelerations of the system [11].

In the inner feedback loops described above, there are corresponding gains that can be adjusted with a pin and screwdriver. Due to some lightly damped oscillations felt during some maneuvers, it was decided to reevaluate the servo cards and make the modifications necessary so that the motion-base would perform the best it could without the aid of acceleration feedback. In looking over the prints supplied by the Navy when the simulator was obtained, it also became evident that there were some discrepancies in the transducer wiring. The 5 volts that the prints instructed to use for the transducers did not match the 24 volts that the transducers recommended. Also, some of the transducer inputs into the servo cards were in the wrong location on the prints. After making the correct changes to the wiring, the inner feedback gains could be tweaked to get a desired response outlined by standard procedures from the Navy [11]. The ringing test was one such procedure performed using a strip chart to measure the response time given a square wave input. This trial-and-error process was done to find the gains where the response had a small overshoot and very fast response time (minimum ringing without hunting).

These modifications proved to be very beneficial in that the motion system fidelity was already greatly improved. The improvement in the motion system greatly enhanced the classical washout algorithm and alleviated a good portion of the pilot complaints. Acceleration feedback was then used to improve upon the benefits of the servo card adjustments.

3.3 Frequency Response Data

After the servo cards were tweaked to obtain the best system characteristics possible, the frequency response data to approximate the transfer function was gathered. A Hewlett-Packard digital signal analyzer was used to record the frequency response. Rather than using sine waves at different frequencies to record the magnitude and phase, low amplitude random noise was input into the servo cards. Random noise was selected as the input because it sweeps the entire frequency range in a very small amount of time. This allowed for recording magnitude and phase over a selected frequency range of 0.1 to 25 Hz in one sweep. Another reason was because the energy being input into the system was spread out over the entire frequency range rather than being concentrated into one frequency (e.g. a sine wave). This helped prevent any damage being done to the system when the natural frequency was reached.

The spectrums of input X, and output Y were recorded and a Bode plot of the entire system transfer function, H_{xy}, was plotted. Averaging was used for better accuracy and to obtain coherence data. Coherence data is a dimensionless measure of the response quality that represents the fraction of output that is linearly related to the input, and values greater than 0.6 are generally acceptable [12]. Since this relationship measures linearity of the input-output relationship, several averages were needed to determine the correct coherence values over the frequency range. Table 3.1 displays the spectrum nomenclature while equation 3.1 shows how these spectrums were averaged. The letter n represents the number of averages performed.

Table 3.1 – Recorded Spectrum Information

G_{xx}	Input autospectrums. X^*X - the product of the complex conjugate of the input spectrum and the input spectrum
G_{xy}	Input-Output cross spectrums. X^*Y – the product of the complex conjugate of the input spectrum and the output spectrum
H_{xy}	Transfer function. Relationship between the input and output. Equation 3.1.
G^2_{xy}	Coherence – Fraction of output that is linearly related to the input.

$$H_{xy} = \frac{\Sigma G_{xy}}{\Sigma G_{xx}} = \frac{\Sigma X * Y}{\Sigma X * X} = \frac{X_1 * Y_1 + X_2 * Y_2 + ... + X_n * Y_n}{X_1 * X_1 + X_2 * X_2 + ... + X_n * X_n} \tag{3.1}$$

The input spectrum of random noise was recorded and injected into the servo cards in the same location as the analog motion cues are injected. The output spectrum was recorded using a high resolution AC coupled accelerometer having a resolution of 5 volts per G. The signal analyzer performed the averaging and the transfer function data was recorded in real and imaginary parts ($\sigma + j\omega$) and converted into a Bode plot using equation 3.2 and 3.3.

$$M = \sqrt{\sigma^2 + \omega^2} \tag{3.2}$$

$$\phi = \frac{180}{\pi} \tan^{-1}\left(\frac{\omega}{\sigma}\right) \tag{3.3}$$

Here, frequency response magnitude (dBVrms), phase angle (deg), and coherence were plotted against frequency (rad/s). Figure 3.1 -3.3 show the frequency response data for the pitch-, roll-, and yaw-axes. Although pitch was the axis of interest in this study, it was of interest to analyze the modes of the roll and yaw axes also.

The sign conventions used for the frequency response data are identical, except the *z*-direction, to those adopted for the body-axis in flight dynamics. The *z*-direction acceleration sensed, used for pitch data, was measured by an accelerometer facing in the positive upward direction. The positive-up convention was used because of the way the accelerometer was mounted as illustrated in figure 3.4. The pitch command in volts, z_{volts}, was taken as positive for upward extension of the piston. The side acceleration, a_y, used for roll and yaw data retains the body-axis sign convention as positive out of the right wing. The roll commands in volts, $p_{volts,}$ used the right piston extension down and left piston up as positive extensions. The yaw

commands, y_{volts}, used the right piston movement left and left piston movement right as positive sign conventions.

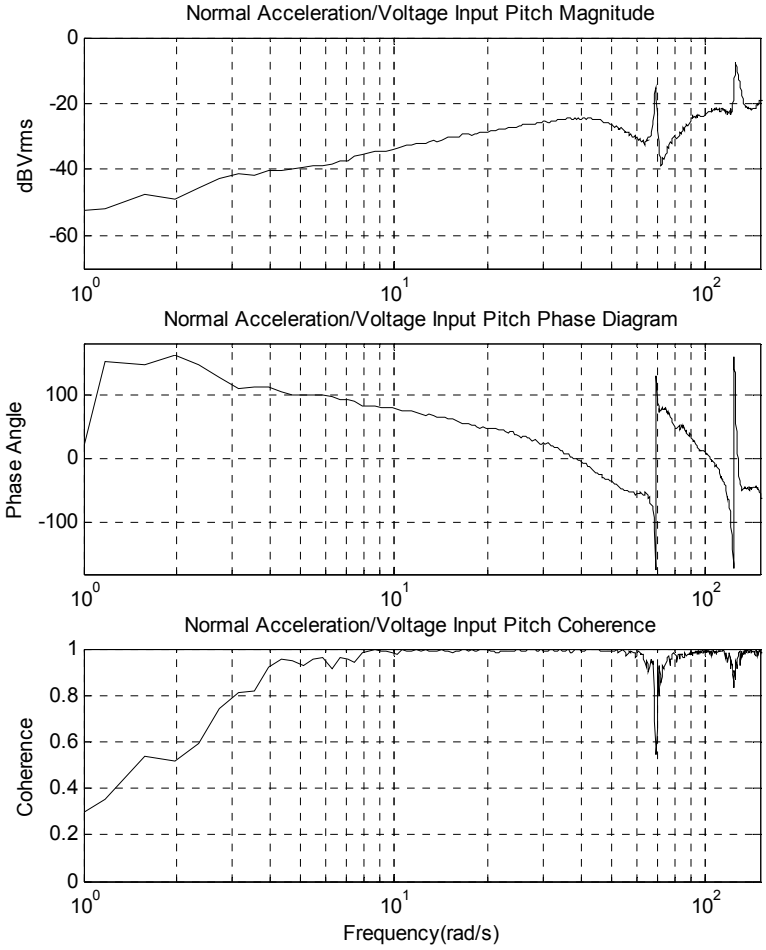

Figure 3.1 – Pitch-Axis Frequency Response

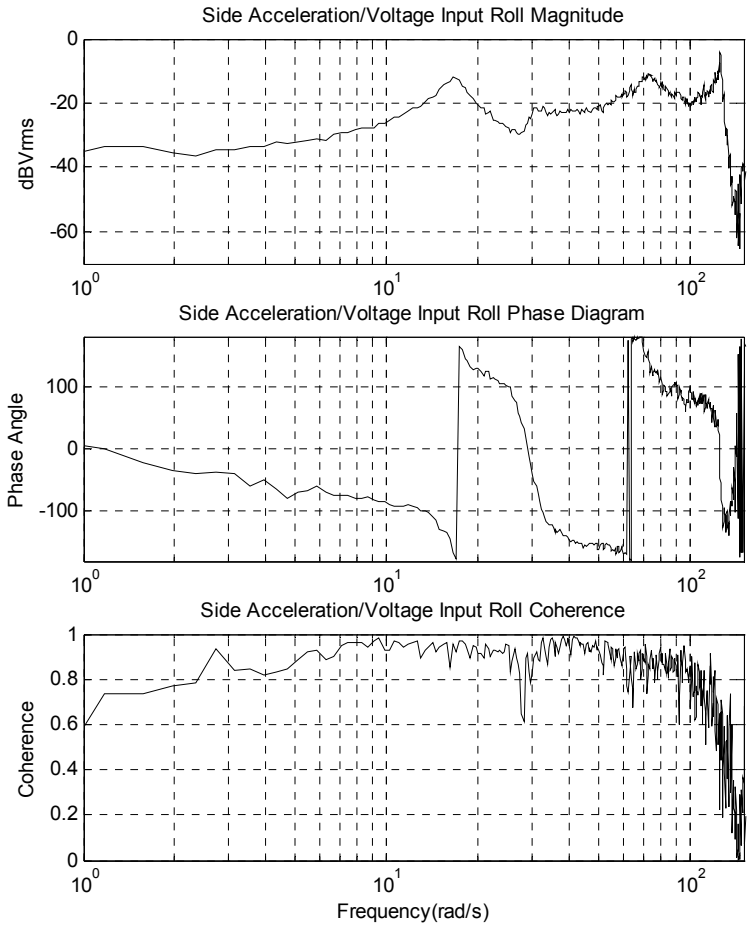

Figure 3.2 – Roll-Axis Frequency Response

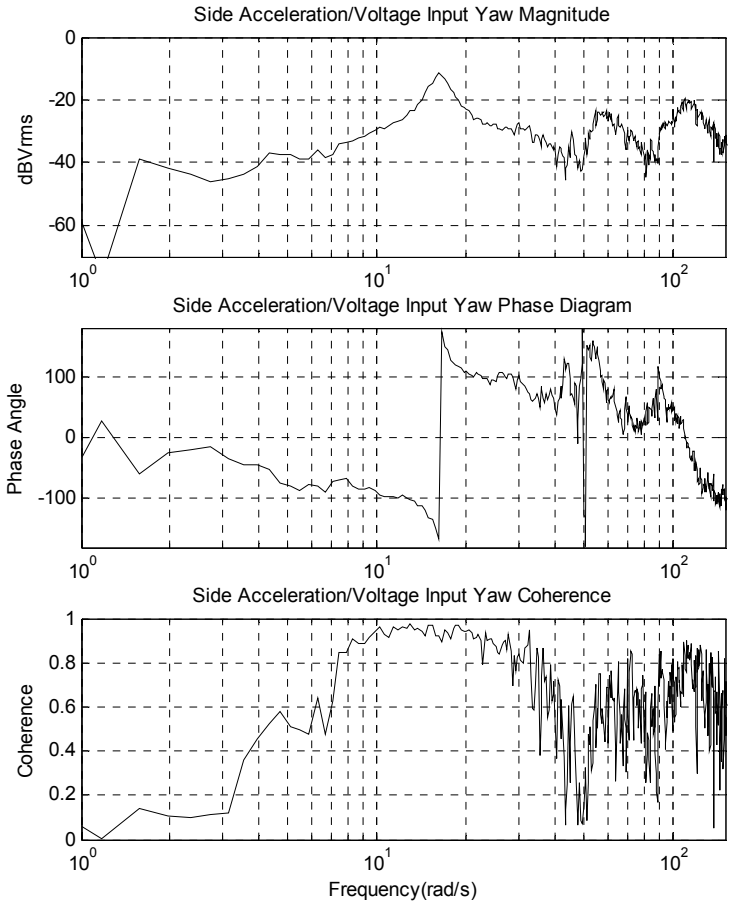

Figure 3.3 – Yaw-Axis Frequency Response

Figure 3.4 – Accelerometer Position

Figure 3.1 shows the pitch-axis frequency response plot with the accelerometer facing upward. A resonance is exhibited around 40 rad/s, 70 rad/s and 125 rad/s. These indicate system modes and natural frequencies. The last two natural frequencies seem like stronger resonances due to the drop in coherence values at these frequencies. The lower frequency mode is of most importance since most motion maneuvers performed are low frequency (around 1 Hz).

Figure 3.2 shows the roll-axis frequency response plot with the accelerometer mounted facing the positive *y*-direction and above the motion beam. Another natural frequency is exhibited at 18 rad/s and is responsible for the aforementioned vibrations, labeled poor structural dynamics in the previous chapter, that occur at this low frequency. Due to problems intrinsic to the simulator's structure, the roll mode was attenuated such that when the natural frequency is reached the simulation will not produce distracting cues. It is also evident that the roll-axis does not correlate well with the input above frequencies of 120 rad/s.

Figure 3.3 displays the yaw-axis frequency response with the accelerometer facing the positive *y*-axis. The system has a strong natural frequency mode at 17 rad/s. Also, the coherence is very poor for this axis. Since the yaw cues are very small compared to the pitch cues, the yaw cues were attenuated so that even very large accelerations in this direction resulted in very small cues.

3.4 Transfer Function Approximation

After the frequency response data was gathered and plotted, an approximate transfer function was then be created. Using a method outlined in Ogata [10], a transfer function was found that fit the experimental data with a reasonable degree of accuracy.

The method begins with fitting asymptotes to the Bode magnitude plot. As specified in Ogata [10], it is often more accurate to use the Bode magnitude measurements to determine the transfer function than it is to use the phase measurements. The asymptotes' slopes were at multiples of +/- 20 dB per decade. Each +/- 20 dB per decade corresponds to a first order polynomial of the form $(s + \omega_c)$, where ω_c represents the frequency at which the asymptote began. For multiples of +/- 20 dB per decade (e.g. +40 dB per decade), the multiple is the power to which the polynomial was raised. For instance, if the slope of the asymptote is +40 dB per decade, the corresponding polynomial would be $(s + \omega_c)^2$. When two asymptotes meet, the slope

of the second asymptote is taken as the slope change *between* the second and the first asymptote. An example of this is a +20 dB per decade asymptote meeting with a -40 dB per decade asymptote. In this case, the second asymptote (-40 dB per decade) would be taken as having a slope change of -60 dB per decade.

In order to form the transfer function, one must place all positive sloped asymptotes in the numerator to create zeroes, and all negative sloped asymptotes in the denominator to create poles. Once this was done all polynomials were multiplied together to form the approximate transfer function.

Many times the above steps are not enough to fit the transfer function to the Bode plot accurately. In order to control the peak (i.e. often where two asymptotes meet), a second order polynomial may be expanded by equation 3.4 for $\zeta = 1$. Once expanded, the frequency at which the asymptote begins remains constant, but the damping ratio of the characteristic equation may be modified. Sharper peaks were results of lower damping ratios. Figure 3.5 illustrates this concept with a +20 dB per decade asymptote meeting a -80 dB per decade asymptote.

$$(s + \omega_c)^2 = (s^2 + 2\zeta\omega_c s + \omega_c{}^2) \tag{3.4}$$

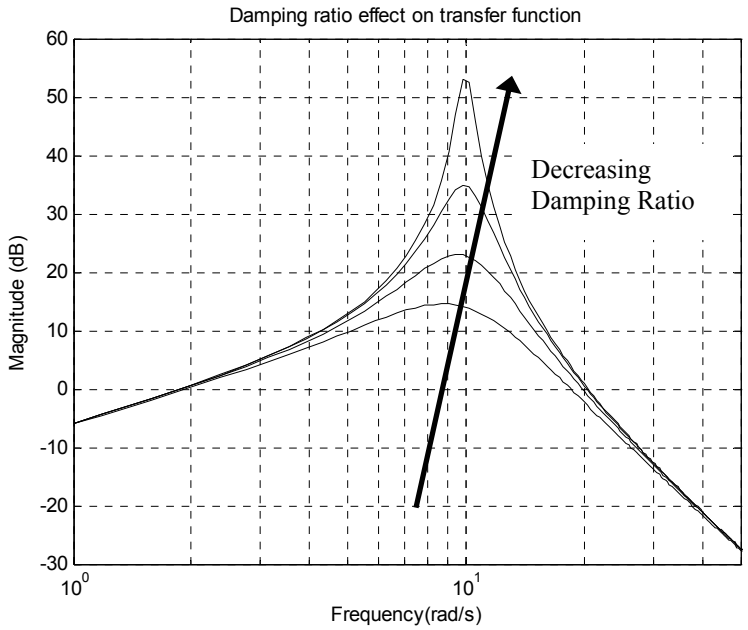

Figure 3.5 - Damping Ratio Effect on Transfer function

Along with modification of damping ratio, a scalar multiple was also used to fit the transfer function to the experimental data. This multiple simply shifted the transfer function up and down on the Bode plot.

In fitting the given experimental data of acceleration output and voltage input the above process was used to create a simple asymptotic approximation. First a +20 dB per decade began near zero frequency. This asymptote created a first order polynomial in the numerator of the form $(s + 0)$. Next, a -80 dB per decade change in slope began at 43 rad/s. This created a polynomial of fourth order $(s + 43)^4$ or $(s^2 + 2\zeta(43)s + 43^2)^2$. Figure 3.6 displays the asymptotes on the Bode magnitude plot. The fact that the asymptotes do not line up exactly with the experimental data is attributed to the various corrections that can be made involving damping ratio and the scalar multiple. The process was more of a trial-and-error eyeballing procedure than exact fitting of asymptotes. The MATLAB code used for this process is displayed in Appendix B.

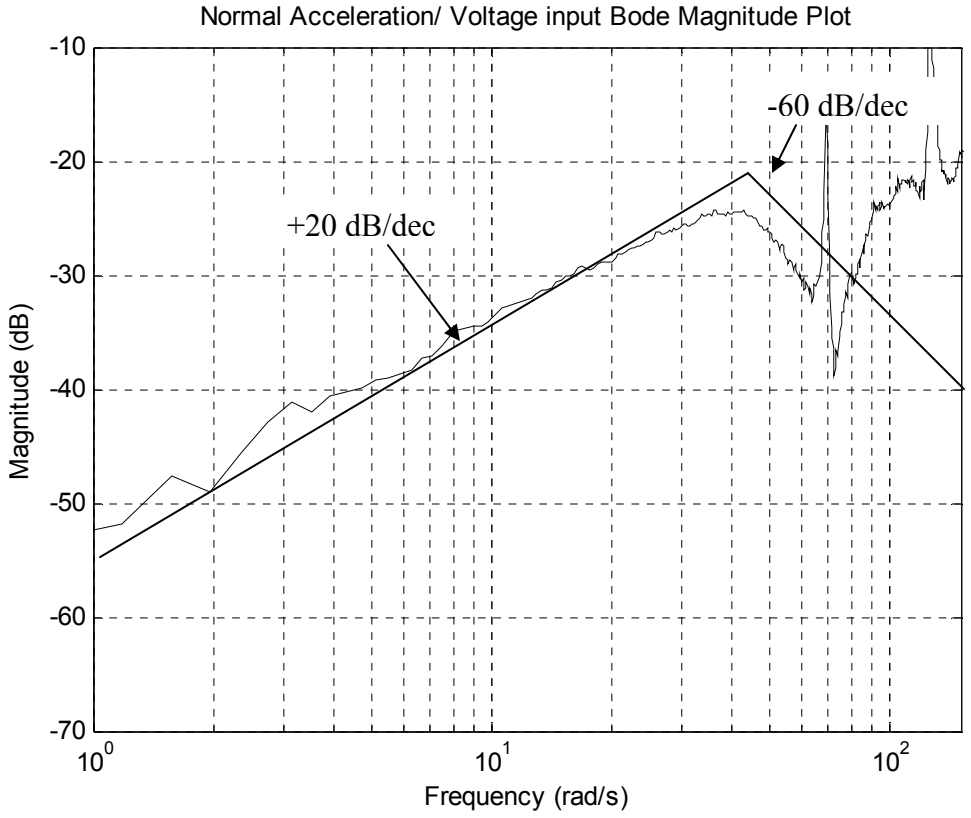

Figure 3.6 - Asymptotic Approximations

After fitting the asymptotes it was then necessary to tweak the starting frequencies, damping ratio, and scalar multiple to find the best match to the experimental data. Following the outlined steps above an accurate transfer function approximation was produced. Equation 3.5 displays the final transfer function representing an output of acceleration at the simulator pilot station and an input of pitch position into the servo cards. Figure 3.7 shows the accuracy of the transfer function approximation. The approximation fits the magnitude plot very well, while the phase diagram is slightly different. This difference was taken as an accurate enough approximation for the theoretical analysis.

$$H_{xy} = \frac{a_{sensed}}{z_{volts}} = \frac{6500s}{(s + 24.4 \pm j37.58)^4} \tag{3.4}$$

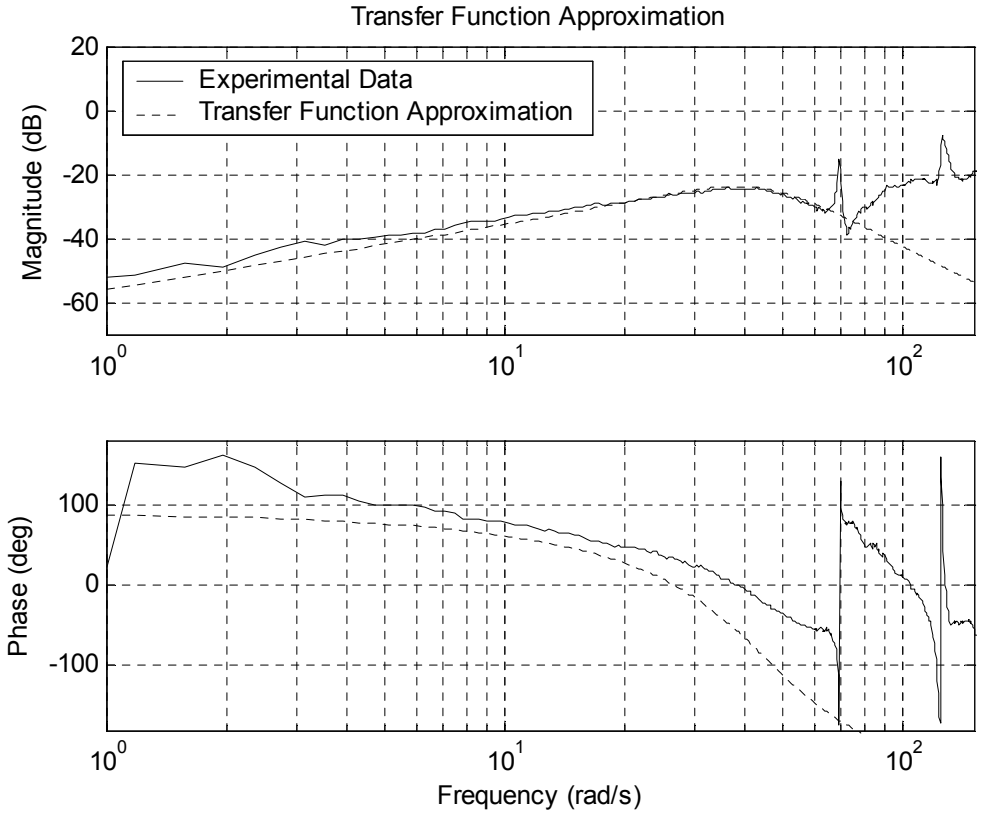

Figure 3.7 – Transfer Function Approximation Accuracy Comparison

Chapter 4

Controller Design and Implementation

4.1 Introduction

Industry has proven that PID controllers can be an invaluable tool for control applications. This control method was selected to enhance motion cueing, alleviate the remaining pilot complaints, and improve the response characteristics of accelerations that were felt in the cockpit. In addition, it is well suited for the single input-single output (SISO) system that this problem represents. A theoretical analysis was done to find desirable feedback gains and a numerical implementation was performed.

4.2 Theoretical Analysis

4.2.1 PID Control Law

The control law $G_{PID}(s)$ is given in equation 4.1.

$$G_{PID}(s) = K_p + \frac{K_I}{s} + K_D s \tag{4.1}$$

In the above equation K_P is the proportional gain, and takes the role of a scalar gain of the measured error between accelerations sensed in the cockpit and the aircraft accelerations the

simulator attempts to replicate. K_I is the integral gain and is responsible for driving the error to zero. K_D is the derivative gain and can be a major factor in system response.

4.2.2 Accelerometer Information

The instrument used to measure the accelerations was a Crossbow triple-axis accelerometer mounted directly above the pilot's head with a resolution of 0.5 volts per G. Because the sensor had a great deal of noise, a first order low-pass filter was used to create a smoother measurement. A break frequency of 3 rad/s was selected due to its smooth response without compromising the true acceleration measurement. Figure 4.1 shows the Bode plot of the low-pass filter. The low sloped drop off in the magnitude shows that the first order filter will slowly attenuate the signal as the frequency increases, while leaving lower frequency signals unchanged.

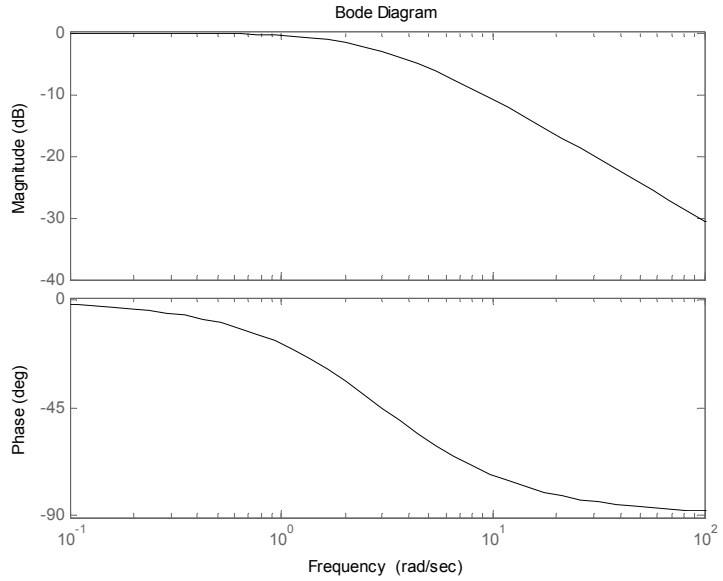

Figure 4.1 – Bode Plot of Low-Pass Filter with Break Frequency of 3 rad/s

Equations 4.2 through 4.5 describe how this low-pass filter was implemented numerically using the simple lag transfer function from chapter 2 [9].

$$H_{LP-ACC}(s) = \frac{a_{accelLF}}{a_{raw}} = \frac{1}{1+s\tau} \tag{4.2}$$

$$a_{accelLF}(1+s\tau) = a_{raw} \tag{4.3}$$

$$a_{accelLF} + \dot{a}_{accelLF}\,\tau = a_{raw} \tag{4.4}$$

$$\dot{a}_{accelLF} = \frac{(a_{raw} - a_{accelLF})}{\tau} \tag{4.5}$$

The above equations were used in the numerical implementation. Equation 4.5 was integrated to obtain the filtered accelerometer data. Figure 4.2 shows the degree of filtering present for a given simulation. The heavy dark line represents the filtered data while the dotted line represents the unfiltered data.

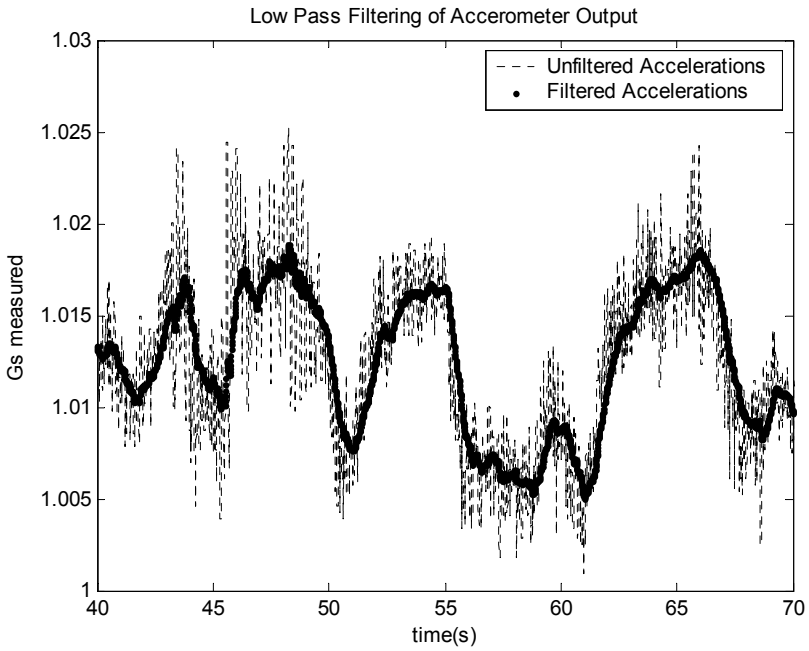

Figure 4.2 – Low-Pass Filtering of Accelerometer Output

Due to the fact that the filter simply attenuates the high frequency maneuvers, leaving low amplitude high frequency data, derivative control would not be applicable. The remaining noise would cause the derivative error to be extremely noisy due to the fact that noise is greatly amplified when the derivative is taken. Thus, this study was reduced to the design of a PI controller.

4.2.3 Complete Transfer Function Approximation

In order to find the feedback gains that give a desirable acceleration response, the full mathematical model must be formed. In the previous chapter the transfer function approximation of *z*-direction acceleration measured at the pilot station (output) to voltage input into the servo cards (input) was formed. However, this transfer function does not account for the dynamics inherent in the motion code. The complete transfer function has an output of *z*-direction sensed acceleration at the simulator pilot station (accelerometer readings) and input of z-direction aircraft cockpit acceleration (input to the motion code from *CASTLE*). In order to produce this transfer function, one must break down the washout filters described in chapter 2 and combine them with the transfer function found in chapter 3.

Transfer functions in the motion code include the high-pass filter with a break frequency of 6.66 rad/s, the two integrators that convert acceleration to position, the low-pass filter to account for measurement noise, and the unit conversion factors. The transfer functions inherent in the motion code for the pitch-axis were combined with the structural, hydraulic, and electronics transfer functions that were collectively represented by the single transfer function H_{xy} (equation 3.4) found in chapter 3. Equation 4.6 and 4.7 show the complete transfer function of the system with output of filtered accelerometer data ($a_{accelLF}$) and input of aircraft cockpit accelerations from the equations of motion (a_z).

$$\frac{a_{accelLF}}{a_z} = [HP\ Filter][Integrators][Scaling\ Factors / Unit\ Conversions][H_{xy}][LP\ Filter] \quad (4.6)$$

$$\frac{a_{accelLF}}{a_z} = \left[\frac{0.15s}{0.15s+1}\right]\left[\frac{1}{s^2}\right][0.025]\left[\frac{6500s}{(s+24.4\pm j37.58)^4}\right]\left[\frac{1}{0.33s+1}\right] \quad (4.7)$$

By inspection, it was obvious that there were two exact pole-zero cancellations. The zero of the transfer function H_{xy} and the zero of the high-pass filter cancel with the two integrators, reducing the system to 4th order. This indicated that the two integrator states were both uncontrollable and unobservable. This was acceptable because the integrators are kinematic and would not affect system internal stability. Because the acceleration output is much smaller than the input due to the motion system constraints, a scaling was necessary to account for the steady-state error between input and output. The final value theorem was applied with a step function to the transfer function as shown in equation 4.8, and the inverse of this value was multiplied by the

numerator of the transfer function to obtain the reduced and scaled complete transfer function for use in controller design as shown in equation 4.9.

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} s\frac{a_{accelLF}}{a_z}\frac{1}{s} = \frac{23.96}{4.101\text{e}006} \tag{4.8}$$

$$\frac{a_{sensed}}{a_z} = \frac{a_{accelLF}}{a_z} = \frac{4.101\text{e}006}{(\text{s}+6.66)(s+24.4\pm j37.93)^2(\text{s}+3)} \tag{4.9}$$

4.2.4 Controller Design

MATLAB was used in conjunction with root locus theory, as coded in Appendix B, to design the PI controller illustrated in figure 4.3. The design objectives were for the acceleration response to a given input to have a damping ratio of 0.707 and zero steady state error between the input and output for a step response. This damping ratio was selected because of its single small overshoot as shown in figure 4.4. As a secondary objective, an attempt was made to place the new eigenvalues such that they exhibit a faster rise time than the original system.

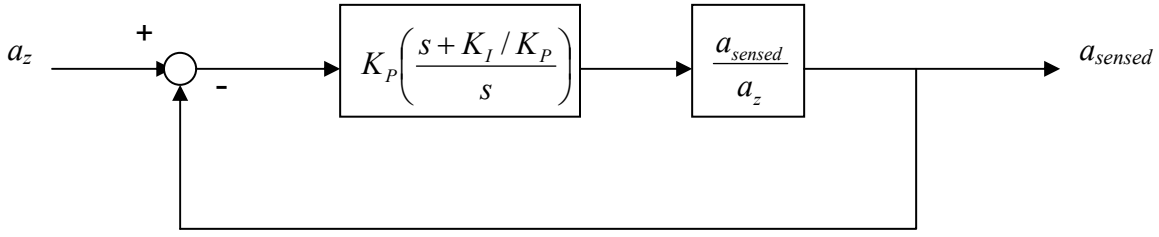

Figure 4.3 – PI Controller Block Diagram

To begin the process of selecting proportional and integral gains, it was useful to first study a P controller with $K_I = 0$. Selecting the necessary value for K_P was simply a matter of selecting the eigenvalues on the open loop root locus plot, shown in figure 4.5, which gave the desired system characteristics.

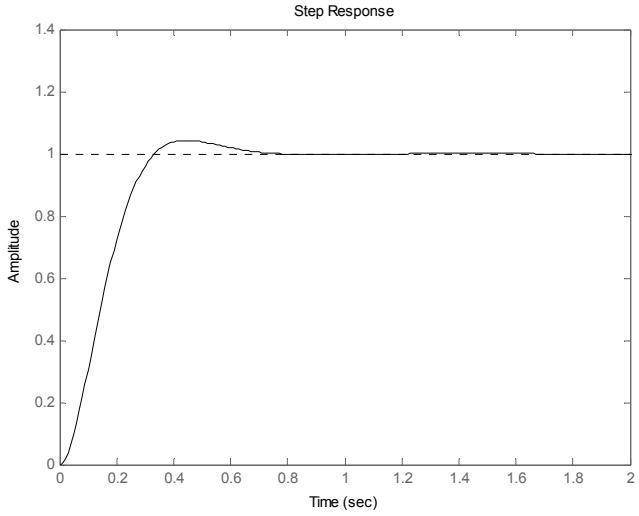

Figure 4.4 – Sample Response with Damping Ratio of 0.707

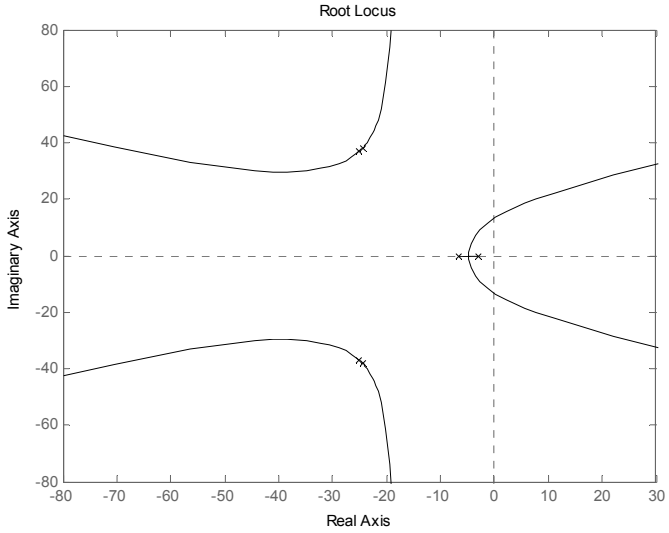

Figure 4.5 – Root Locus of Open-Loop System

The dominant eigenvalues were the low-pass and high-pass filter poles. These poles begin at s = -3 and s = -6.66 respectively. For various feedback gains, the poles move toward each other and then form a complex pair as shown in a magnified root locus plot in Figure 4.6. On this complex pair, a feedback gain of K_P = 0.91 was selected such that the damping ratio was 0.707. However, as shown in Figure 4.7, the steady state value for a step input was nearly half

the commanded value. The steady-state error led to the investigation of a proportional and integral controller.

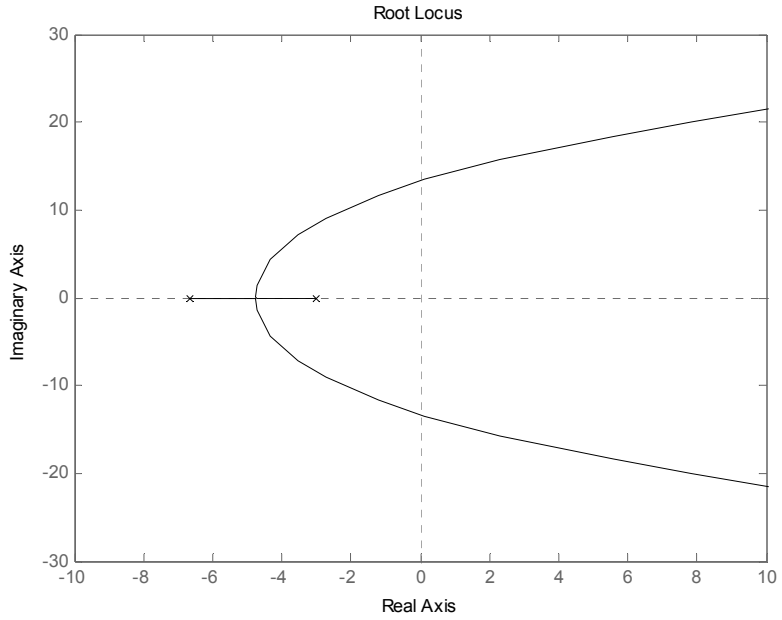

Figure 4.6 – Magnified Root Locus of Dominate Eigenvalues

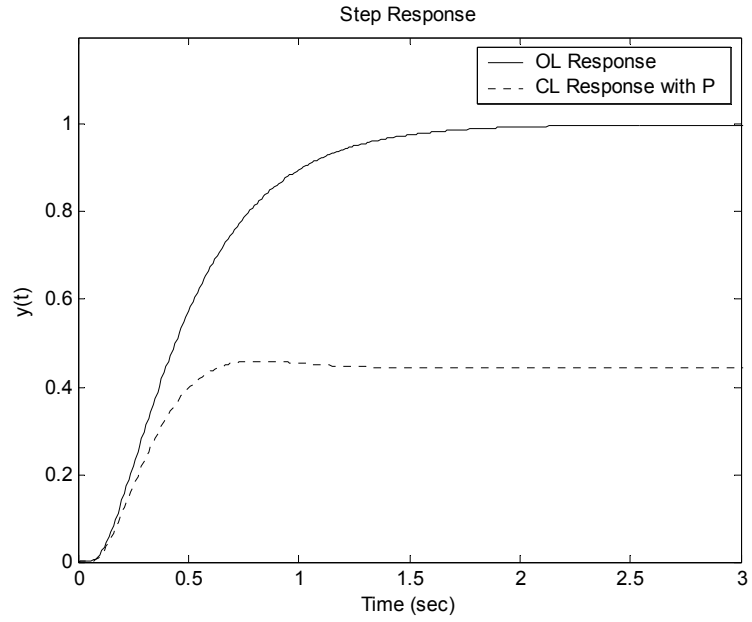

Figure 4.7 – Response to Step Input with P Feedback

In order to design and tune proportional and integral gains, a method outlined in Stevens and Lewis [9] was adopted. Using this method, equation 4.1 becomes equation 4.10 where $s = -K_I / K_P$ is the compensator zero. Upon selecting a compensator zero, a value for K_P was selected from the root locus much in the same way as it was for the P controller. What resulted was an iterative process in which the values for $s = K_I / K_P$ and K_P were eventually tuned to give the desirable dominant eigenvalues.

$$G_{PI}(s) = K_P\left(\frac{s + K_I / K_P}{s}\right) \tag{4.10}$$

Initially a compensator zero was chosen at s = -10 to the left of both filters. The root locus plot was then used to find the K_P that gave a damping ratio of 0.707 with the given compensator zero. Figure 4.8 shows the response characterized by zero steady state error, but an extremely slow rise time compared to the open loop response. A compensator zero to the right of both filters at s = -2 yielded a response in Figure 4.9 with a faster rise time, but very slow convergence to steady state.

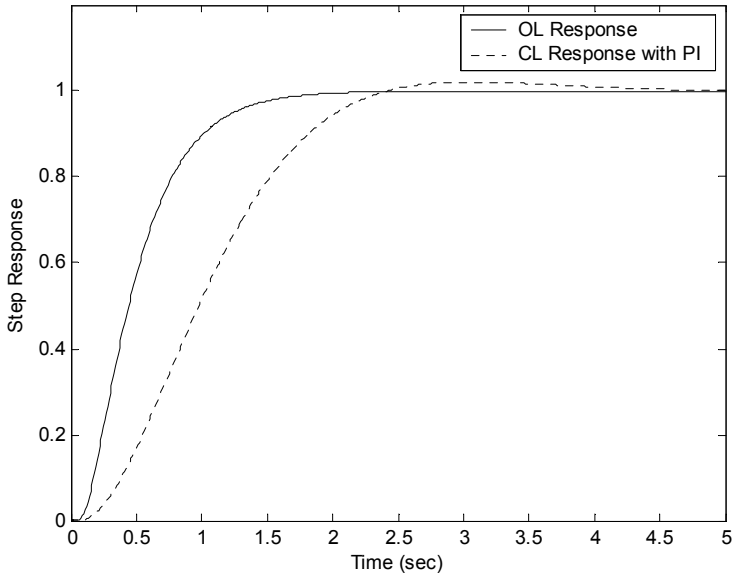

Figure 4.8 – Response to Step Input with K_I / K_P = -10

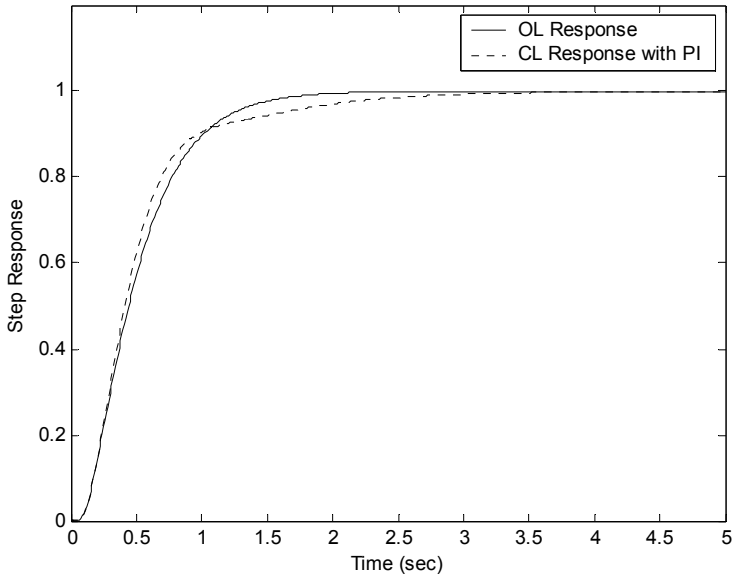

Figure 4.9 – Response to Step Input with K_I / K_P = -2

Several more iteration steps yielded a very desirable response. A compensator zero at s = -3.1 yielded the best response that was found. The K_P selected was 0.82 while the corresponding K_I was found to be 2.5. Figure 4.10 shows the comparison of the PI controller with these feedback gains and the P controller designed previously.

A sine wave was also simulated to evaluate response characteristics. The PI feedback improved the response for low frequency sine waves, but as the frequency increases to around 0.5 Hz, the PI controller did not improve the response; in fact there was a larger lag in the PI controller response than in the open loop response. Figures 4.11 and 4.12 illustrate this concept with sine waves of frequency 0.2 Hz and 0.75 Hz.

It can thus be stated that the PI controller has the potential to improve the dynamic response of the motion system and its algorithms for low frequency maneuvers, while the P controller or open loop response produce favorable results for higher frequency maneuvers. The fact that various scaling factors were created such that the input and output display similar acceleration magnitudes caused the steady state error for the P controller to be a minor problem. The fact that the acceleration magnitudes of input and output were scaled such that the input and output are of similar magnitudes leads to the conclusion that steady state error may be alleviated by simply scaling these values by a different factor. Nonetheless, the PI controller was

implemented to search for any beneficial features that the integral control may provide. If the user wishes to use just the proportional controller, K_I may simply be set to zero.

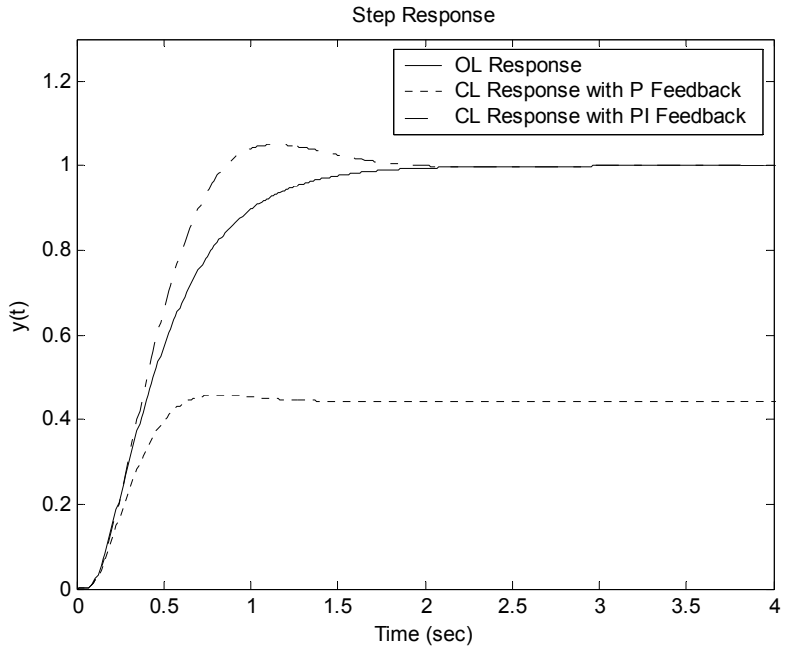

Figure 4.10 – Response to Step Input for Selected P and PI Feedback

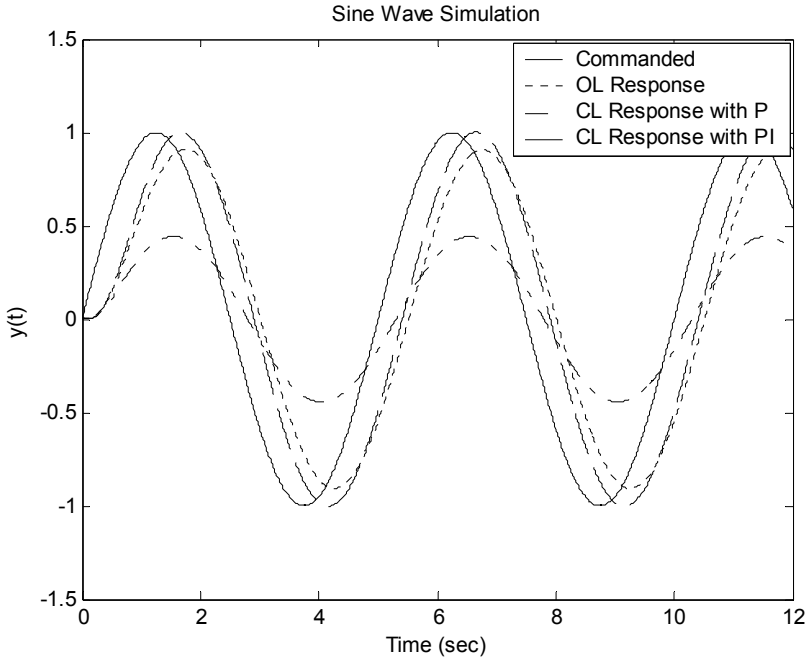

Figure 4.11 – Response to a Sine Wave of Frequency 0.2 Hz

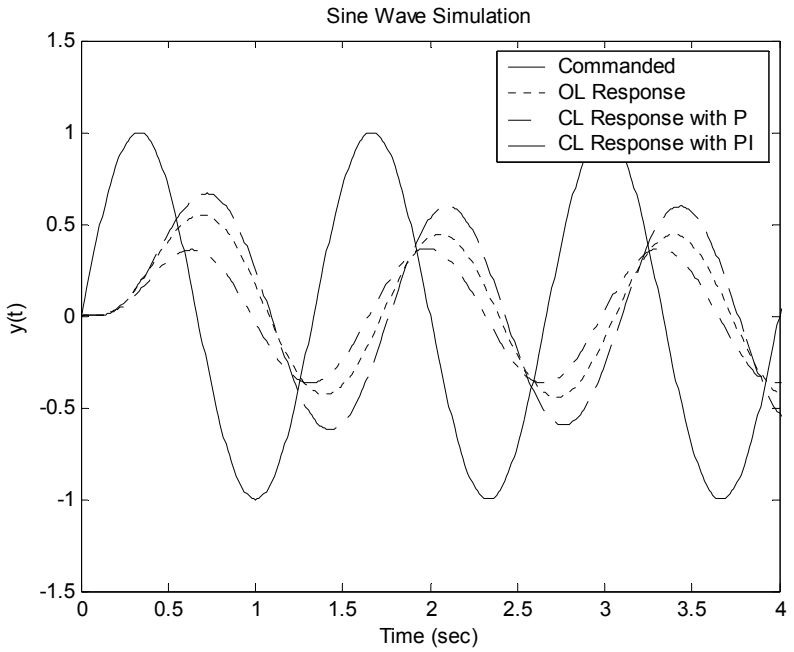

Figure 4.12 – Response to a Sine Wave of Frequency 0.75 Hz

4.3 Implementation

4.3.1 Block Diagram

Figure 4.13 illustrates the entire block diagram for the acceleration feedback PI controller for the *z*-axis. The algorithm was implemented into FORTRAN as shown in Appendix A with all flags for acceleration feedback turned on. As one can see, this algorithm contains all of the classical washout blocks with the addition of the outer loop of acceleration feedback. Table 4.1 describes all the variables and their corresponding names as programmed in the motion code displayed in Appendix A.

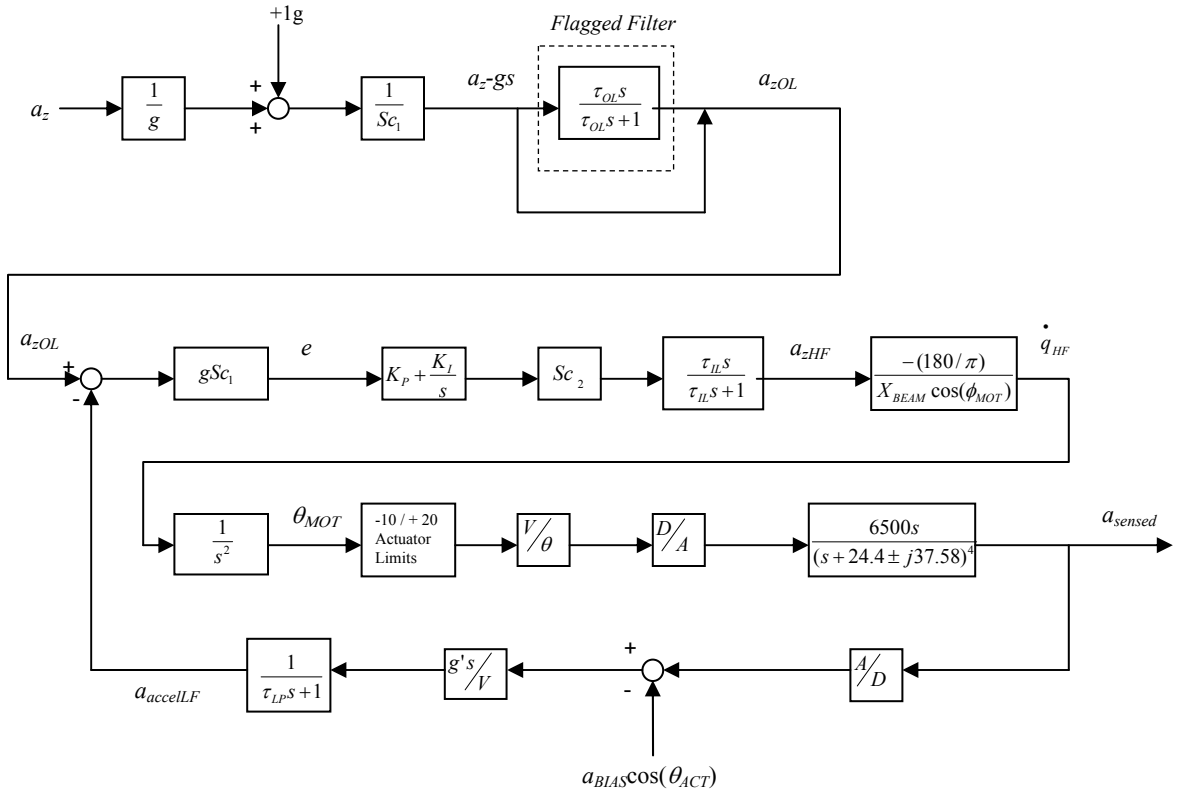

Figure 4.13 – Acceleration Feedback Algorithm

Table 4.1 – Description of Variables and Corresponding Code Variables

Variable Name	Variable Name in Code	Description
a_z	AZP	Cockpit *z*-direction aircraft accelerations obtained from the equations of motion.
g	GMOT	Acceleration due to gravity in ft/s^2.
Sc_1	ACC_SCALE	Scaling factor to reduce the cockpit z-direction accelerations to a similar magnitude as accelerometer reading.
a_z-gs	AZP_GS	Scaled cockpit accelerations converted to *g's.*
τ_{OL}	HIRATEZOL	Time constant for high-pass filter outside of acceleration feedback loop.
s	n/a	The Laplacian symbol. In the Code an s represents the derivative while the inverse is an integral.
a_{zOL}	AZPOUTHIOL	Outer loop high-pass filtered (or unfiltered based on flag) scaled accelerations in g's. This variable is equivalent to the reference variable.
e	ERR	The error between the accelerometer reading and the scaled accelerations after being scaled back up to a similar magnitude as a_z.

K_P	KP	Proportional gain in PI controller.
K_I	KI	Integral gain in PI controller.
Sc_2	AZSCALE	Tuning variable that simultaneously scales the two gains by the same multiple in feedback algorithms. Used mainly as the scaling factor in the classical washout algorithm.
a_{zIL}	AMOTZ	Acceleration error multiplied by feedback gains before in-loop high-pass filtering.
τ_{IL}	HIRATEZ	Time constant for high-pass filter inside of acceleration feedback loop.
a_{zHF}	AZPOUTHI	In-loop high-pass filtered acceleration errors.
X_{BEAM}	XMP	The distance from the gimbal and the cockpit seat.
ϕ_{MOT}	ROLLPOS	The roll position of the cockpit in degrees.
$\dot{q}_{HF}$	QDOTMOT	The angular acceleration commanded in the pitch-axis.
θ_{MOT}	THETHI	The angular position to be commanded to the motion system.
θ_{ACT}	PPOSD	The actual angle position of the cockpit measured by the potentiometer.
V / θ	-0.04	Conversion factor from angular position to volts.
D / A	n/a	A digital to analog converter.
A / D	n/a	An analog to digital converter.
a_{BIAS}	ACC_BIAS	The accelerometer zero offset value.
$g's / V$	2	Conversion from accelerometer reading of volts to *g's*.
τ_{LP}	1 / LPBF	Time constant for low-pass filtering of accelerometer readings.
$a_{accelLF}$	ACC_FILT_SCALE	The low-pass filtered accelerometer readings.

4.3.2 Algorithm Description

The algorithm receives input of pilot station *z*-direction accelerations in ft/s^2 computed from the equations of motion in *CASTLE*. These accelerations are then converted to *g*'s and the neutral point was changed zero g's rather than negative one g. The accelerations are then scaled down by a factor of 50 in order to obtain a comparable value to the accelerations read by the accelerometer. The block labeled "Flagged Filter" will be discussed later, but for now the case of the filter being turned off will be discussed. The acceleration a_{zOL} is the reference variable in the standard control diagram.

An error is produced by the comparison between the reference variable, a_{zOL}, and the filtered accelerometer reading, $a_{accelLF}$. This error is thought of to be the error between the accelerations the motion-base produces and the scaled down accelerations of an aircraft. Ideally,

the error will be very small or zero for perfect motion response. After this error is created, it is then multiplied by gravity and the scaling factor of 50 in order to translate the accelerations into the correct units of ft/s^2. This error is then multiplied by the proportional gain designed in the previous section. The error is also numerically integrated for use in integral control and then multiplied by the integral gain. These two multiplications (error multiplied by K_P and the integral of the error multiplied by K_I) are then added together to create the new overall commanded acceleration. A tuning factor that serves the purpose of scaling both feedback gains up or down by the same factor is also implemented for user convenience. The commanded accelerations are then high-pass filtered such that the motion-base does not reach its physical limits.

As described in chapter 2, the high-pass filtered accelerations are then converted to angular accelerations and integrated twice to obtain commanded motion travel in degrees. The next block in Figure 4.13 represents the physical limits of +20 and -10 degrees exhibited by the motion-base. This was implemented into the code by limiting the commanded motion positions such that the motion-base would reach user imposed limits within the code before actually reaching its physical limits and possibly causing damage to the motion-base. Next, a conversion factor scales the commanded position to a corresponding voltage and is sent out to the servo cards after being converted to an analog signal. The next block corresponds to the transfer function found in chapter 3 consisting of the servo card circuitry with two inner loops of position and pressure feedback, the structure of the motion-base and cockpit, and the accelerometer itself. Obviously, this transfer function was not coded because it is represented in the physics of the motion system.

The accelerometer output is centered about its zero point by subtracting the product of the instrument's bias and the cosine of the actual pitch angle as measured by the same potentiometer used for inner-loop position feedback. The cosine of the actual pitch position is incorporated to account for the tilting of the gravity vector's effect on accelerometer readings when the cockpit is tilted but stationary. The measurements are converted to *g*'s, and low-pass filtered as described in section 4.2.2. These filtered measured accelerations represent the lower variable, $a_{accelLF}$, in the summing block shown in figure 4.13.

4.4 Experimentally Discovered Issues

4.4.1 Nonlinearities

Since the theoretical analysis performed in section 4.2 was for a linear system when in fact the motion system is nonlinear due to actuator position and rate limits, some unforeseen issues may arise in the numerical implementation and testing of the classical washout algorithm, the P, and PI controllers. One such issue may be the fact that the limits inherent in the motion base may be reached and the controller will not account for this problem.

4.4.2 Integral Removal

One problem arose with the use of integral control to alleviate any steady state issues. When integral control was used it sometimes seemed to extend the motion-base to its limits despite the high-pass filter. This problem can be attributed to several separate reasons. First, if the accelerometer drifted away from its zero point, which it tended to do occasionally, then the integral of this error would tend to wind up and the commands would drive the motion system to its imposed limits. This occurred most of the time when the pilot was flying a low magnitude *z*-direction acceleration run. For instance when the pilot flew a carrier approach, the magnitudes of the pilot station aircraft accelerations would be relatively small due to the low speed and small pilot corrections. These smaller accelerations led to less error from the proportional standpoint, but the same amount of drift error from the accelerometer. Thus, integral control would in a sense take over and cause the motion-base to extend to its limits because the integral of the error would build up over time.

Figure 4.14 displays how the integral error tended to wind up because the motion-base cannot reach the commanded accelerations due to physical limitations. This causes the motion-base to reach the user defined limits inherent in the motion code and remain at the lower or upper limit until the simulation was stopped or the integral error began to wind up in the opposite direction. During higher magnitude aircraft acceleration runs, this seemed not to be a large issue because the controller was typically dominated by the proportional control. This was due to the fact that the higher magnitude aircraft accelerations produced larger deviations from the zero point, and thus led to the drift being more negligible. Since the integral control was not aiding in

the motion simulation based on pilot opinion and time history response, the conclusion was made that integral control was not producing beneficial results because of two main reasons.

First, the drift in the accelerometer created a constant error that would build up when integrated. This caused poor motion cues because the algorithm was trying to compensate for a steady state error that wasn't actually present.

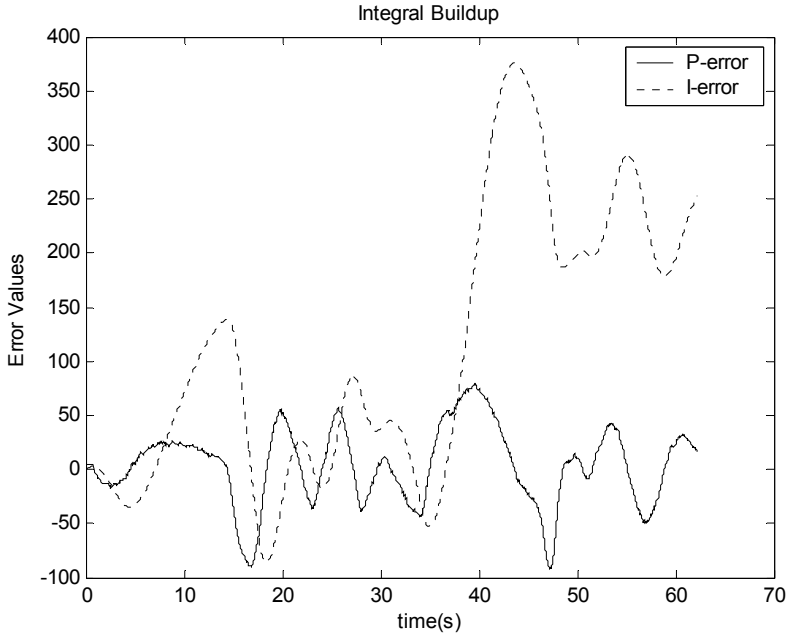

Figure 4.14 – Integral Error Windup

Second, because of the high-pass filtering of the commanded accelerations, the scaled aircraft accelerations differed from the commanded motion accelerations. In turn, this caused the measured accelerations in the simulator to differ from the reference variable of aircraft accelerations. The reason this was not a problem with the proportional gain was because it was multiplied by the error value at that particular time and then high-pass filtered. It became a problem with integral control because the integral of the error is dependent on past values of error and thus would build up over time and extend the motion-base to its limits.

Integral control removal was not a large issue due to the reasoning of why it was incorporated in the first place. The reason it was introduced was to alleviate any steady-state error in the response plots. In further evaluation, this is somewhat trivial in this application because of the fact that aircraft pilot station accelerations were scaled to match more closely to

the accelerometer readings. In other words, steady state error will always exist in reality because the accelerations achieved in an aircraft cannot be reproduced by a land-based simulator. A user could simply scale the input or output accordingly such that there is not a difference in the commanded acceleration for one scaling and a measured acceleration for another scaling.

Due to the physical limits along with the removal of integral control, it was decided to use the proportional feedback gain of 0.91 as a starting point in testing. Simulated test flights using a pilot rating scale and time history responses of accelerations are later discussed as the final deciding factors in finding the best combination of parameters for a given flight condition.

4.4.3 Effect on Tilt-Coordination

One of the main limiting factors discovered while implementing the acceleration feedback controller was that tilt-coordination had to be removed. This feature, which allows the pilot to sense sustained accelerations (mostly used for *x*-direction accelerations in this simulator) by tilting the cockpit affected the accelerometer readings in an adverse way. The problem lay within the fact that in order to move the cockpit to a given tilt angle, an initial acceleration was required. Although this acceleration was measured by the accelerometer, it could not be separated from the *z*-direction accelerations. When this initial acceleration occurred, the P controller read this as an error because it is using only *z*-direction aircraft forces as a comparison. This caused the controller to fight the tilt-coordination in an adverse manner. Thus, in order for acceleration feedback to be useful, the feature of tilt-coordination was removed.

4.4.4 High-Pass Filter Location

For acceleration feedback, the idea of where the high-pass filter should be located became a topic of interest. Since the high-pass filter was located inside the loop, the comparison between the reference variable of scaled aircraft cockpit accelerations and sensed acceleration from the accelerometer was in question. Because the error created was high-pass filtered, the reference a_{zOL} would not usually match the output variable of $a_{accelLF}$, therefore the error would usually be nonzero. An experiment was performed to examine the effects of placing a high-pass filter outside the loop. This would have the characteristics of comparing the high frequency aircraft

accelerations with the sensed accelerations. The theory behind this idea was to match the reference variable with the accelerometer readings more comparably. Shown in Figure 4.13 as the "Flagged Filter", tests were performed to investigate the results of filter placement.

Initially, the high-pass filter was removed from inside the loop and placed outside the loop. It became quickly apparent that this caused the motion system to reach its inherent physical limits often. Due to the fact that filtering was done on the reference variable, there was nothing to filter the low frequency errors that were periodically created and input into the feedback algorithm. Consequently, the motion system would diverge from its zero reference point and never washout. It was decided that a separate high-pass filter was needed inside the loop as well. Since this seemed to contradict the reason a filter was placed outside of the loop, the filter inside the loop was required to have a much smaller break frequency, such that the filter outside the loop does most of the filtering work, while the filter inside the loop serves only to washout the motion-base to its neutral position during very low frequency error commands. This configuration was tested in the following chapter along with the feedback algorithm with one high-pass filter in the loop and the classical washout algorithm.

4.4.5 Gain Scheduling

Since the magnitude of accelerations an aircraft achieves is dependent upon its flight condition, the final step in the controller design was to vary the feedback gains according to flight condition. Because density and velocity affect the magnitude of the accelerations felt, dynamic pressure was used to schedule the feedback gains.

Two flight conditions were tested to determine the best feedback gain for the two dynamic pressures. A low dynamic pressure flight condition was a formation flying task with a tanker at 205 knots and 2,000 ft (dynamic pressure of 134 lb/ft^2). A higher dynamic pressure flight condition was formation flying at 440 knots and 15,000 ft (dynamic pressure of 411 lb/ft^2). These two flight conditions yielded two different ideal feedback gains because of the differing magnitudes of aircraft accelerations attained. A function was then made that will pick off a feedback gain given a dynamic pressure. The selection of these feedback gains is discussed in detail in the following chapter.

Chapter 5

Validation

5.1 Introduction

The purpose of this chapter is to analyze the three algorithms in the longitudinal axis as previously mentioned. The first step in analyzing the algorithms was to label them for shorthand notation. The algorithm that uses acceleration feedback with one high-pass filter in the loop was AF1. The algorithm that uses acceleration feedback with two high-pass filters, one outside of the loop and one inside the loop as discussed in the previous chapter, was labeled AF2. Finally, the classical washout algorithm without acceleration feedback was named CW.

Each algorithm was analyzed by experimentally varying the parameters involved in each one using a trial-and-error method. Each algorithm was tested under two flight conditions of 205 knots at 2,000 ft, labeled flight condition I, and 440 knots at 15,000 ft, labeled flight condition II, in order to obtain the aforementioned gain scheduling. Both flight conditions were tested under clean conditions (no flaps, landing gear down, etc.). Over 120 different parameter configurations spanning all three algorithms were tested (flown in the simulation environment) by a simulator engineer and the configurations were narrowed down to just twenty-two: ten for flight condition I and twelve for flight condition II. The process of narrowing down the 120 initial

configurations consisted of eliminating configurations for reasons such as too much motion, phase lag, oscillatory movements (tracking problems), or simply unpleasant sensations.

The final twenty-two configurations found from this initial elimination process are further analyzed in this chapter using a pilot opinion rating scale and time history plots of accelerations to find the best configuration for each flight condition.

5.2 Tested Parameter Configurations

5.2.1 Flight Condition I Configurations

Each algorithm was tested at the flight condition of 205 knots and 2,000 ft. Different parameter configurations for each algorithm were narrowed down to ten using the trial-and-error elimination process described in the previous section. Table 5.1 lists the parameter configurations that were subject to final testing using the AF1 algorithm. Table 5.2 presents the list of the configurations subject to final testing using the AF2 algorithm and table 5.3 presents the configurations tested using the CW algorithm. All AF1 and AF2 configurations used a Sc_2 of 0.1 as shown in Figure 4.13.

Table 5.1 – AF1 Configurations (F.C. I)

Configuration Number	K_P	τ_{IL} (s)
1	2.5	0.3
2	1.5	1
3	1.2	0.8
4	7	0.06

Table 5.2 – AF2 Configurations (F.C. I)

Configuration Number	K_P	τ_{IL} (s)	τ_{OL} (s)
1	1	4	2
2	5.5	1	0.02

Table 5.3 – CW Configurations (F.C. I)

Configuration Number	Sc_2	τ_{IL} (s)
1	0.6	0.15
2	0.08	1
3	0.025	2
4	1.4	0.05

5.2.2 Flight Condition II Configurations

Each algorithm was also tested at the flight condition of 440 knots and 15,000 ft. Different parameter configurations for each algorithm were narrowed down to twelve using the trial-and-error elimination process described in the previous section. Table 5.1 presents the list of the parameter configurations that were subject to final testing using the AF1 algorithm. Table 5.2 lists the configurations subject to final testing using the AF2 algorithm. All AF1 and AF2 configurations used a Sc_2 of 0.1. Finally, Table 5.3 presents the configurations tested using the CW algorithm.

Table 5.4 – AF1 Configurations (F.C. II)

Configuration Number	K_P	τ_{IL} (s)
1	25	0.02
2	0.6	0.5
3	0.3	1
4	0.5	0.3

Table 5.5 – AF2 Configurations (F.C. II)

Configuration Number	K_P	τ_{IL} (s)	τ_{OL} (s)
1	0.5	2	1
2	4	1	0.1
3	0.5	6	1

Table 5.6 – CW Configurations (F.C. II)

Configuration Number	Sc_2	τ_{IL} (s)
1	0.1	0.3
2	0.25	0.1
3	0.1	0.2
4	0.025	1
5	0.075	0.4

5.3 Pilot Ratings

5.3.1 Pilot Rating Scale and Questionnaire

The most important factor in selecting the best algorithm and parameter configuration for permanent use in the simulator was pilot opinion. Although time history data was also analyzed, this data was used only as a tool to analyze pilot opinions and aid in the evaluation of evenly rated configurations.

Pilot rating scales are widely used to evaluate aircraft flying qualities. For instance, handling qualities of an aircraft are measured by a test pilot using the Cooper-Harper rating scale [13]. Pilot opinion for aircraft handling qualities, although qualitative, is quantified by the rating scale. Similar to this evaluation, a pilot rating scale was incorporated as the main tool in evaluating the motion system's fidelity.

Many pilot opinion evaluation methods were experimented with, some involving a decision tree much like the Cooper-Harper rating scale, and others that required the pilot to evaluate motion sensations with several categories. It was concluded that a rating scale and questionnaire would be used based on Reid and Nahon's pilot evaluations study [6]. Figure 5.1 presents the rating scale/questionnaire used to evaluate the twenty-two parameter configurations.

The rating scale consisted of seven attributes for the pilot to analyze. A rating of one to five was given based on the descriptions shown in figure 5.1. The smoothness attribute served to analyze how smooth or jerky the motion-base performed. The sense attribute accounted for the pilot's overall sense of how the motion system felt compared to that of an actual aircraft. The amplitude attribute was a very important category in that it measured if the motion scaling was correct. If the feedback gain or scaling factor was too high, a lag would become present and the washing out of the motion system to its zero position could be felt. Conversely, if the gain or scaling factor was too low, the pilot would not feel enough motion cues to aid in the simulation. The phase lag attribute served to rate the motion sensation correlation to the visual cues. If the pilot felt that the motion seemed to be catching up to the visual cues, a phase lag would be quantified. The discomfort attribute provided a documentation of any unpleasant motion cues sensed during the task. The disorientation attribute was much like the discomfort attribute except it served the purpose of quantifying any unpleasant cues that seemed to disorient the pilot. Finally, an overall rating of the motion cues was given.

MOTION RATING SCALE:

Attribute	**Rating**		
	1		**5**
Smoothness	Extremely smooth – no bumps or jerks		Extremely jerky – limit of tolerance
Sense	Definitely correct – as in aircraft		Totally reversed
Amplitude	No motion experienced		At least twice that expected
Phase Lag	None experienced		At least 180°
Discomfort	None experienced		Cannot continue maneuver
Disorientation	None experienced		Cannot perform maneuver
Overall	Excellent		Extremely poor

PILOT QUESTIONNAIRE

1. Was the timing of the motion sensations relative to the visual and instrument cues satisfactory?

2. Was the magnitude of the motion sensations relative to the visual and instrument cues satisfactory?

3. Did you experience any distracting motion sensations?

4. Was the motion sensation helpful in your piloting task?

5. Did the motion sensation add to the simulation?

6. Was this your favorite so far?

7. If not, what was better about the current favorite?

Figure 5.1 – Pilot Rating Scale and Questionnaire

The questionnaire's purpose was to gain information from the pilot of what led to the given attribute ratings. Question one asked the pilot if any motion cues that were not in-sync with the visual cues were present, while question two allowed the pilot to note any motion cues that were not of an expected magnitude based on the visual cues. Question three asked the pilot if there were any distracting cues that may have led to poor ratings for any of the seven attributes. Question four and five are very important in that they ask the pilot if the motion cues were helpful to the task and if they added to the simulation environment. This was very important because if the sensations did not add to the simulation then a conclusion was made that no motion cues were better than distracting motion cues. Question six and seven asked the pilot if the parameter configuration was their favorite so far and to explain what made the configuration better or worse than their previous favorite.

5.3.2 Evaluation Flights

Since there were two flight conditions of interest due to the later implementation of gain scheduling, two separate pilot tasks were created to analyze the motion sensations. Initially, a carrier approach and landing was used for the task, but not enough information was being produced due to the low amplitude accelerations felt during an approach. A formation flying task was then created. For flight condition I the pilot started at 1,000 ft and 205 knots and was asked to rendezvous and fly formation with a refueling tanker at 2,000 ft and 205 knots positioned 2,000 ft in front of the pilot's aircraft. This exercised the longitudinal axis during the initial maneuver as the pilot transitions from 1,000 to 2,000 ft. The pitch-axis was also exercised when the pilot needed to make corrections to remain in formation with the tanker.

The same task was used to evaluate flight condition II except the altitude and airspeeds were different. The pilot began at 14,000 ft and 440 knots while the tanker began at 15,000 ft and 440 knots and was positioned 2,000 ft in front of the pilot's aircraft.

A retired Navy test pilot was the subject to the final tests done using the rating scale and questionnaire. The test pilot is currently an Associate Professor in the Department of Aerospace and Ocean Engineering at Virginia Tech. After a distinguished career as a carrier based fighter pilot, he served as a test pilot instructor at the United States Naval Test Pilot School at Patuxent River Naval Air Station. It is important to note that the pilot was asked to evaluate the longitudinal axis only, for the pitch axis is the subject of this report.

The aircraft simulated was an F-18 and was flown without flaps or landing gear down for both tasks. A headset with a microphone was used to communicate the attribute ratings and questionnaire answers. Each task was flown until the pilot was comfortable in evaluating the given configuration. Typically, the flights lasted around two minutes.

5.3.3 Flight Condition I Pilot Ratings

The ten parameter configurations for flight condition I were randomly sorted and tested using the rating scale and questionnaire. The results of the rating scale and questionnaire for all ten configurations are presented in table 5.7 and table 5.8.

It is apparent from the rating scale and questionnaire results that there were several good parameter configurations. Six configurations received an overall rating of one. The favorite configuration, although five others were very similarly rated, was an AF1 with a washout filter time constant of 0.06 seconds and a feedback gain of 7. Several other notable AF1 configurations had washout filter time constants of 0.3 seconds and 1 second with respective feedback gains of 2.5 and 1.5. The AF2 configurations received poor ratings for both configurations at this flight condition. The CW configurations received very favorable remarks for the cases with high-pass filter time constants of 0.15, 2, and 0.05 seconds with respective scaling factors of 0.6, 0.025, and 1.4. The final configuration for gain scheduling was picked based on the five favorable configurations in this flight condition and the subsequent favorable configurations found in the second flight condition.

Table 5.7 – Flight Condition I Rating Scale Results

Algorithm	Number	Smoothness	Sense	Amplitude	Ph. Lag	Discomfort	Disorientation	Overall
AF1	1	1	1	2	1	1	1	1
AF1	2	1	1	2	1	1	1	1
CW	1	1	1	2	1	1	1	1
CW	2	1	1	2	2	1	1	2
AF2	1	1	3	2	2	2	2	3
AF2	2	1	2	2	1	2	2	2
AF1	3	1	1	2	2	2	1	2
AF1	4	1	1	2	1	1	1	1
CW	3	1	1	2	1	1	1	1
CW	4	1	1	3	1	1	1	1

Table 5.8 - Flight Condition I Questionnaire Results

Question	**Configuration AF1 – 1**	**Configuration AF1 – 2**
1	Yes	Yes
2	Yes	Yes
3	No	No
4	Yes, it helped	Yes
5	Yes	Yes
6	Yes	Felt very similar to the first
7	n/a	Felt very similar
Question	**Configuration CW – 1**	**Configuration CW – 2**
1	Yes	No
2	Yes	Yes, but it could have been better
3	No	Yes, the motion peaked milliseconds after the visual did
4	Yes	No
5	Yes	No
6	Yes	No
7	More amplitude was felt. This was a good thing.	It felt like it was behind me. I was very much aware of the motion system

Question	Configuration AF2 – 1	Configuration AF2 - 2
1	No	Yes
2	Yes, but definitely not perfect	Yes
3	Yes, just didn't feel right at all. Didn't seem to correlate well.	Yes, it felt real jerky or oscillatory
4	No	No
5	No	No
6	No	No
7	It was very distracting. Maybe I could feel it washing out more than I should	It was very bouncy. NOTE: low amplitude oscillations were seen
Question	**Configuration AF1 – 3**	**Configuration AF1 – 4**
1	No	Yes, it felt real good
2	Yes	Yes
3	Yes, I could feel a phase lag	No
4	No	Yes
5	No	Yes
6	No	No
7	No, because of the phase lag	This was the best so far
Question	**Configuration CW – 3**	**Configuration CW – 4**
1	Yes	Yes
2	Yes	Yes
3	No	No
4	Yes	Yes
5	Yes	Yes
6	No	No
7	Felt a bit low on amplitude	Something felt better about the current favorite

5.3.4 Flight Condition II Pilot Ratings

The twelve parameter configurations for flight condition II were randomly sorted and tested using the rating scale and questionnaire. The results of the rating scale and questionnaire are presented in Table 5.9 and Table 5.10.

Five configurations received an overall rating of one. The favorite configuration was a CW configuration with a time constant of 1 second and a scaling factor of 0.025. Another CW configuration with favorable results had a high-pass filter time constant of 0.2 seconds and a scaling factor of 0.1. The AF1 configurations that received excellent ratings had high-pass filter

time constants of 1 and 0.3 seconds, with corresponding feedback gains of 0.3 and 0.5. One AF2 configuration received very favorable ratings. Its parameters consisted of an in-loop high-pass filter constant of 2 seconds, an outer-loop high-pass filter constant of 1 second, and a feedback gain of 0.5.

Table 5.9 – Flight Condition II Rating Scale Results

Algorithm	Number	Smoothness	Sense	Amplitude	Ph. Lag	Discomfort	Disorientation	Overall
AF1	1	1	1	2	1	1	1	2
AF1	2	1	3	2	3	3	2	3
CW	1	1	3	2	3	2	1	3
CW	2	1	2	1	1	2	1	2
AF2	1	1	1	3	1	1	1	1
AF2	2	1	1	2	2	2	1	2
AF1	3	1	1	3	1	1	1	1
AF1	4	1	1	2	1	1	1	1
CW	3	1	1	3	1	1	1	1
CW	4	1	1	3	1	1	1	1
CW	5	1	1	2	2	2	1	2
AF2	3	1	1	3	3	2	2	3

Table 5.10 - Flight Condition II Questionnaire Results

Question	**Configuration AF1 – 1**	**Configuration AF1 – 2**
1	Yes	No
2	No, maybe not enough amplitude	Yes
3	No	Yes, felt a phase lag
4	No	No
5	Yes	No
6	Yes	No
7	n/a	Felt a phase lag
Question	**Configuration CW – 1**	**Configuration CW – 2**
1	No	Yes
2	Yes	Yes

3	Yes, a phase lag	Yes, the motion was distracting when flying close to the aircraft
4	No	Yes
5	No	Yes
6	No	No
7	The phase lag detracted from the simulation	Because when I was close in to the tanker it was distracting
Question	**Configuration AF2 – 1**	**Configuration AF2 - 2**
1	Yes	No
2	Yes	Yes
3	No	Yes, I could feel a phase lag
4	Yes	No
5	Yes	No
6	Yes, but motion cues were still on smaller maneuvers	No
7	n/a	Because of the sensed phase lag
Question	**Configuration AF1 – 3**	**Configuration AF1 – 4**
1	Yes	Yes
2	Yes	Yes
3	No	No
4	Yes	Yes
5	Yes	Yes
6	No	No
7	The amplitude felt better on my favorite	This just seemed better than the current favorite
Question	**Configuration CW – 3**	**Configuration CW – 4**
1	Yes	Yes
2	Yes	Yes
3	No	No
4	Yes	Yes
5	Yes	Yes
6	Yes	Yes
7	Seemed to react a bit better	Liked this one a little more
Question	**Configuration CW – 5**	**Configuration AF2 – 3**
1	No	No
2	No	Yes
3	Yes, a phase lag	Yes, a phase lag
4	No	No
5	No	No
6	No	No
7	Because of the lag	I really did not like that one

5.4 Time Histories

The next step in the evaluating the final configurations was to investigate the time history responses of variables inherent in the implemented motion code. These responses served to aid the pilot opinion ratings in the analysis of the different configurations.

5.4.1 Maneuver Generator

A useful feature that *CASTLE* has to offer is the maneuver generator (MANGEN). This feature allows the user to input different functions to overwrite a certain variable that is calculated within *CASTLE*'s structure. Among the function inputs available are a sine wave, step, or time history. During testing, a time history of cockpit *z*-direction accelerations was recorded for a certain task and then MANGEN's time history input feature was used to reproduce the same *z*-direction acceleration inputs into the motion code. Since the subject of this report concentrates on longitudinal motion and the only input into the motion code for this degree-of-freedom is cockpit z-direction accelerations, MANGEN serves to recreate a recorded test flight without a pilot in the cockpit.

This feature was an invaluable tool in comparing the response characteristics of the final parameter configurations. The task for the two flight conditions of interest was flown and two separate time histories of a_z were recorded as shown in Figure 5.2 and 5.3. These time histories were then used as input for the twenty-two final parameter configurations, allowing time history comparisons to be made for the variables of interest. The advantage of using MANGEN was that the different configurations tested could be easily compared because all configurations were tested with the same acceleration input.

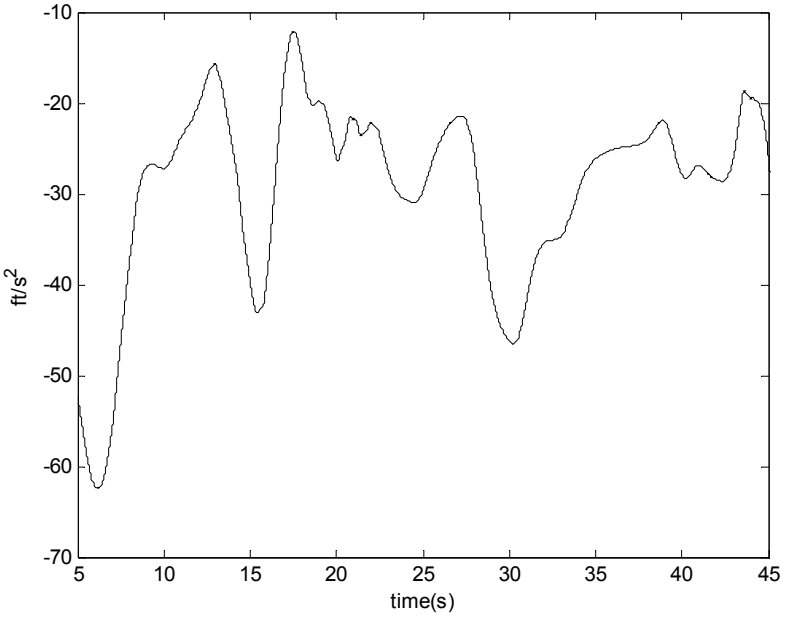

Figure 5.2 – Flight Condition I a_z Input

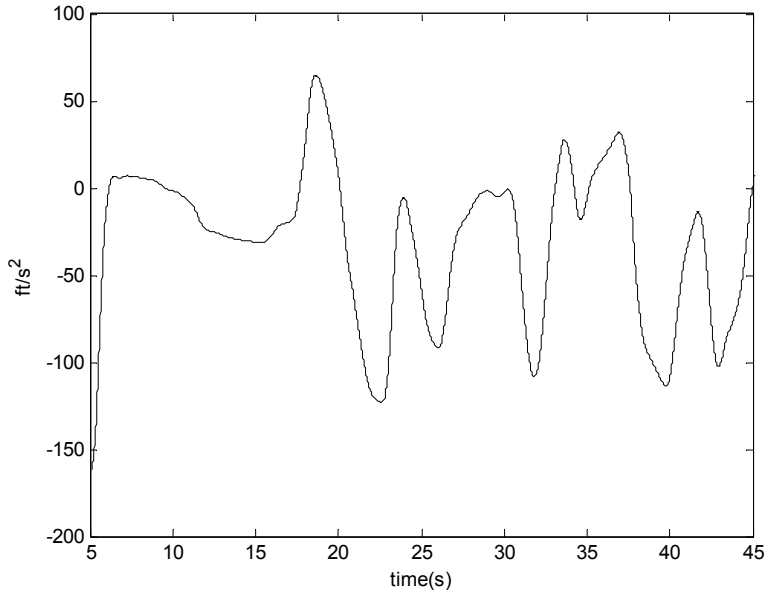

Figure 5.3 – Flight Condition II a_z Input

5.4.2 Plot Descriptions

The parameter configurations were analyzed in the order at which the pilot ratings were taken. One way in which a particular configuration was analyzed was to compare scaled-up accelerometer data, referred to as measured accelerations (a_{sensed}), and aircraft cockpit accelerations from the equations of motion that are referred to as aircraft accelerations (a_z). This was a good starting point, but it was important to realize what washout filters and gains or scaling factors were used, as they greatly affect the system response. The accelerations calculated by the motion algorithm after the high-pass filter, named commanded accelerations (a_{zHF}), were plotted along with the measured accelerations (a_{sensed}) in the simulator cockpit. This served to analyze how the motion system was able to respond to algorithm commanded accelerations for a particular algorithm. It must be noted that because of the fact that the measured accelerations were typically two to four times less than that of the commanded, the measured accelerations were multiplied by a factor of two in order to obtain more informative plots of the response characteristics.

5.4.3 Flight Condition I Time Histories

The *z*-direction cockpit accelerations in figure 5.2 were used as input for the ten parameter configurations for this flight condition. Only the six configurations with the overall rating of one were analyzed because, as stated previously, the time history responses served only to help select between very evenly matched pilot rated configurations.

5.4.3.1 Configuration AF1 – 1 (F.C. I)

Configuration AF1 – 1 (K_p = 2.5, τ_{IL} = 0.3 s) received favorable pilot ratings and thus should have fairly good time history correlations. Figure 5.4 shows the measured accelerations matched fairly well for a good portion of the task, but were also somewhat different for many instances. This was noticed in many of the configurations and was attributed to the filtering out of some of the lower frequency accelerations as shown in figure 5.5. This figure shows the filtering of the acceleration commands and how there was very high attenuation during the lower frequency maneuvers.

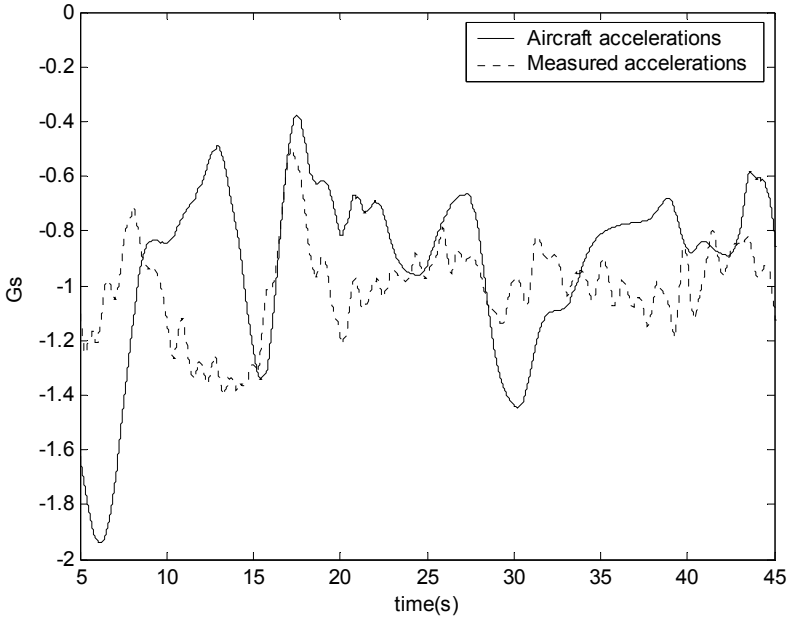

Figure 5.4 – Aircraft/Simulator Acceleration Comparison for AF1 - 1 (F.C. I)

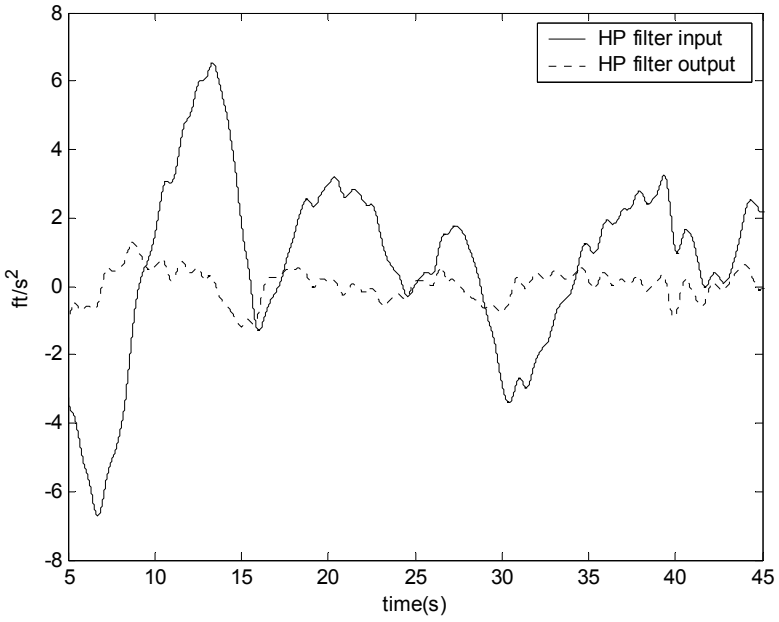

Figure 5.5 – High-Pass Filtering for AF1 - 1 (F.C. I)

Figure 5.6 displays the relationship of accelerations commanded by the motion algorithm after high-pass filtering (a_{zHF}) and the corresponding accelerations measured (a_{sensed}). As one

can see, they were slightly out of phase after thirty seconds because the motion system could not remain in phase with high frequency commands.

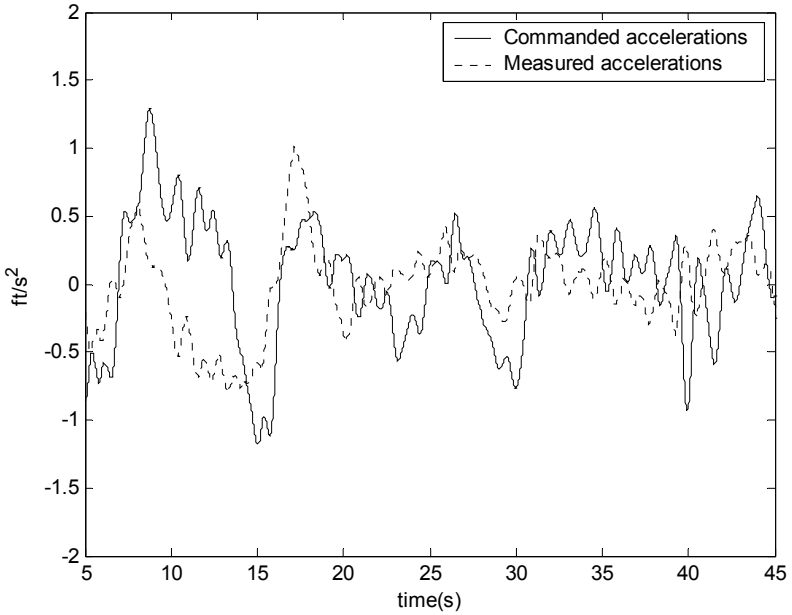

Figure 5.6 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF1 - 1 (F.C. I)

5.4.3.2 Configuration AF1 – 2 (F.C. I)

Configuration AF1 – 2 (K_p = 1.5, τ_{IL} = 1 s) also received favorable pilot ratings and thus should have good time history correlations. Figure 5.7 shows the scaled-up measured accelerations matched fairly well with the aircraft accelerations for a good portion of the task. This time history seemed no better or worse than the previous configuration and this statement matches what the pilot stated in the questionnaire that the two configurations felt very similar. Figure 5.8 shows the commanded accelerations and the measured accelerations. In this figure, configuration AF1 – 1 may have had slightly better correlation than AF1 - 2, but this difference was minor.

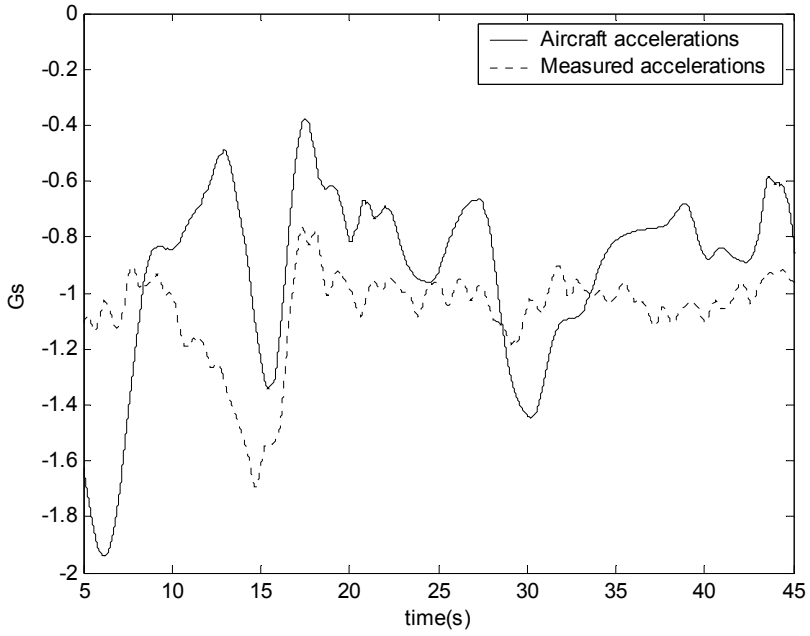

Figure 5.7 – Aircraft/Simulator Acceleration Comparison for AF1 - 2 (F.C. I)

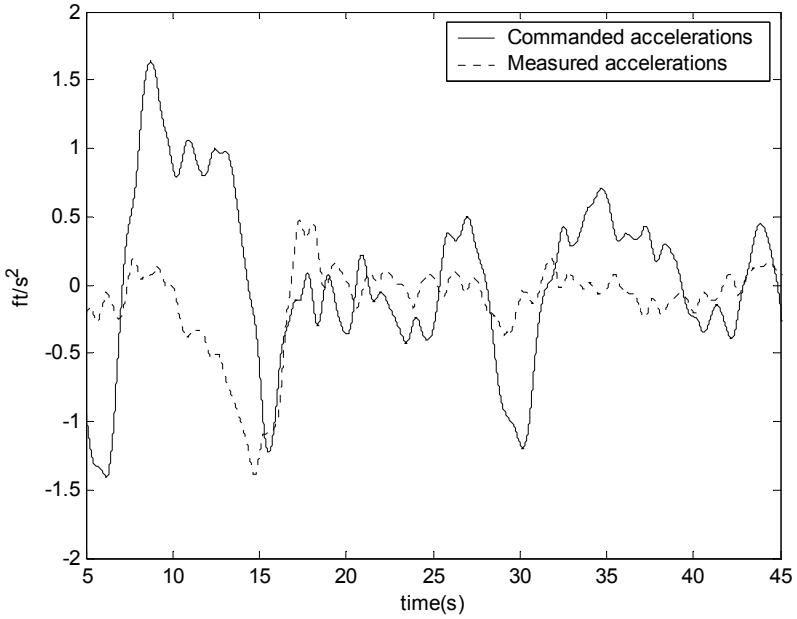

Figure 5.8 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF1 - 2 (F.C. I)

5.4.3.3 Configuration CW – 1 (F.C. I)

Configuration CW – 1 (Sc_2 = 0.6, τ_{IL} = 0.15 s) seemed to have a slightly worse than average correlation in aircraft accelerations and scaled accelerometer readings as shown in figure 5.9. Figure 5.10 shows the commanded accelerations and the measured accelerations had a very good correlation in phase and magnitude.

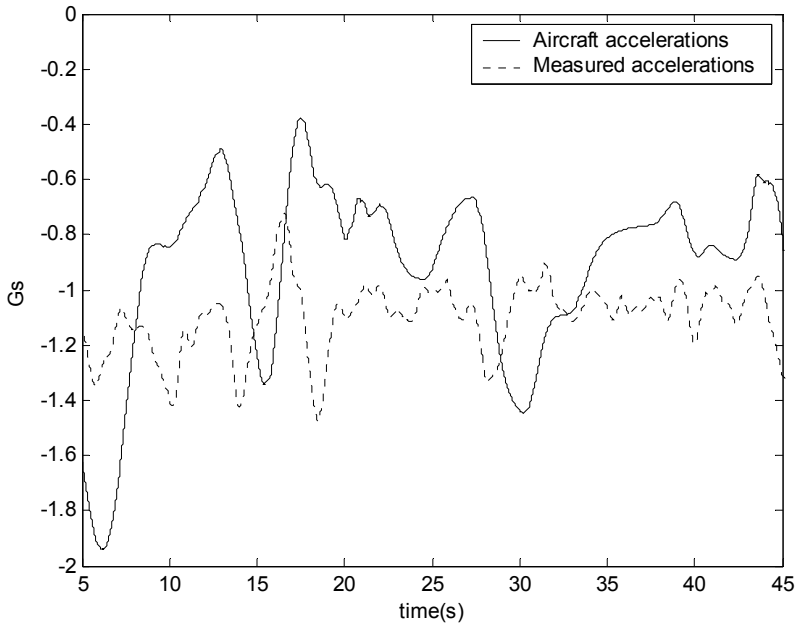

Figure 5.9 – Aircraft/Simulator Acceleration Comparison for CW - 1 (F.C. I)

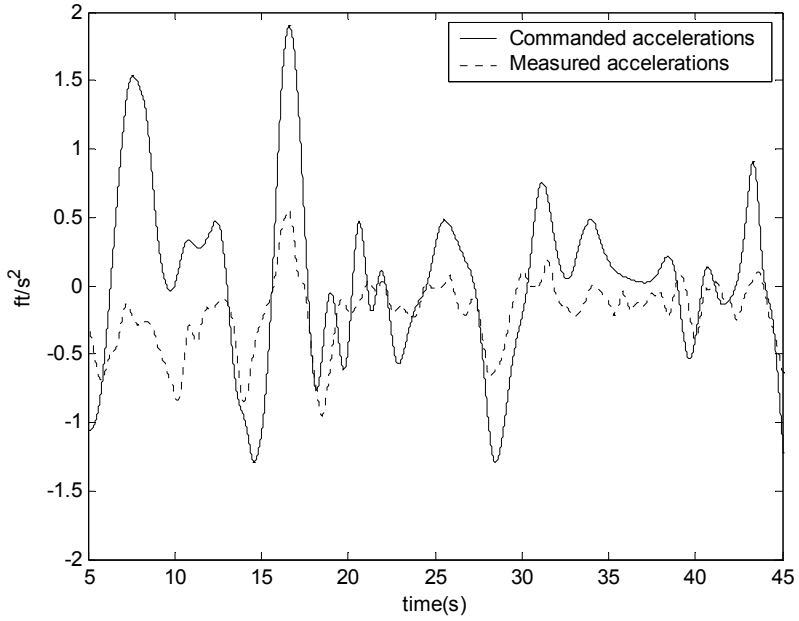

Figure 5.10 – Algorithm Commanded/Twice the Measured Acceleration Comparison for CW - 1 (F.C. I)

5.4.3.4 Configuration AF1 – 4 (F.C. I)

Figure 5.11 displays a reasonable time history of *z*-direction aircraft accelerations and accelerometer readings for configuration AF1 – 4 ($K_p = 7,\ \tau_{IL} = 0.06$ s). This plot shows that this configuration was very similar to the previous AF1 configurations in that the acceleration response measured in the cockpit closely matches figures 5.4 and 5.7. Figure 5.12 shows the commanded accelerations and the measured accelerations. A phase lag was noticeable between the commanded accelerations and the measured accelerations. This may be attributed to the dynamics of the system and the increased feedback gain that must compensate for the high attenuation of low-frequency accelerations.

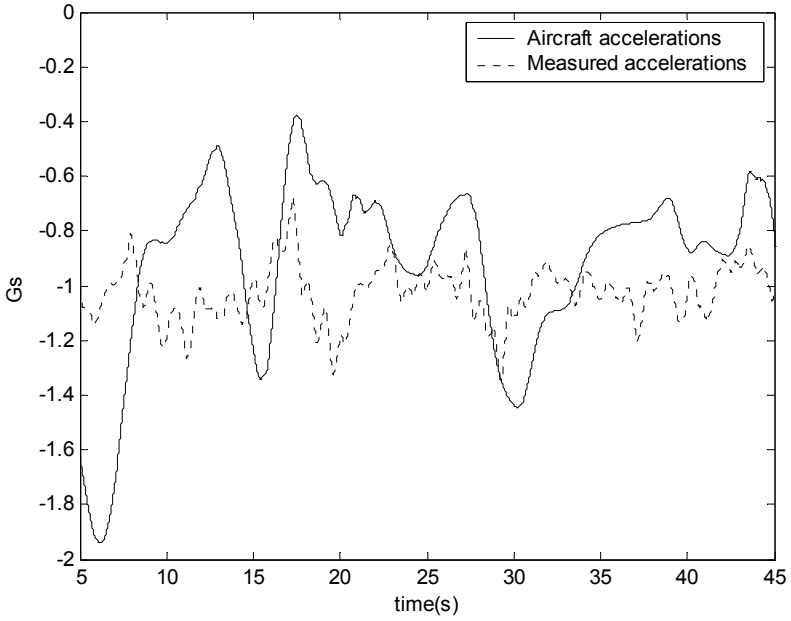

Figure 5.11 – Aircraft/Simulator Acceleration Comparison for AF1 - 4 (F.C. I)

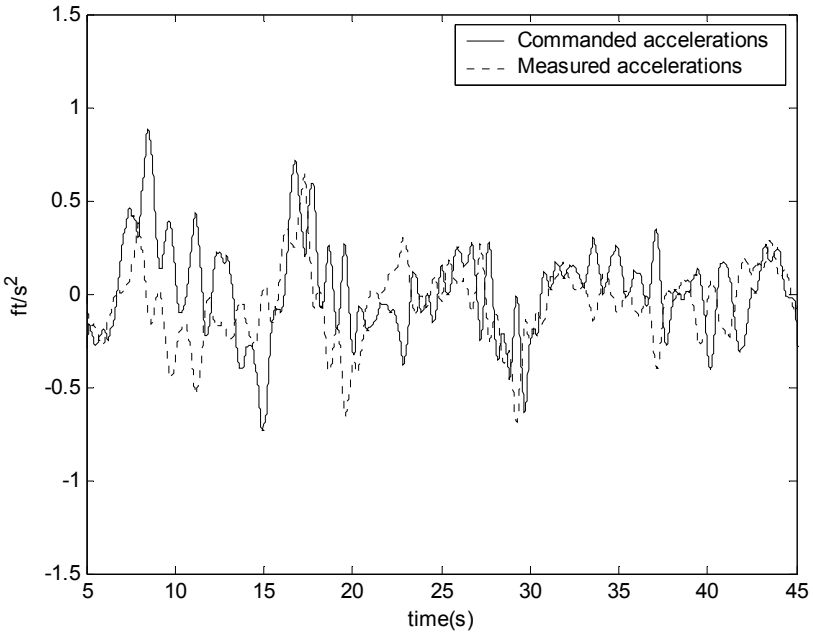

Figure 5.12 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF1 - 4 (F.C. I)

5.4.3.5 Configuration CW – 3 (F.C. I)

Figure 5.13 represents the aircraft/simulator acceleration comparison for configuration CW – 3 ($Sc_2 = 0.025$, $\tau_{IL} = 2$ s). As one can see, the amplitude for the measured accelerations was much lower for this case. This was an expected trait of this configuration as the pilot complained that the amplitude for this case was low, but not low enough to receive a rating of one (no motion experienced). Figure 5.14 displays the accelerations commanded to the motion-base and the corresponding accelerations that the motion-base produced. It was apparent that the accelerometer drifted slightly in the recording of this data because the measured accelerations seemed to be centered around 1 ft/s^2 rather than zero ft/s^2. The correlation between these two variables was poor and the fact that the accelerometer drifted gives proof of the first reason of why integral control was removed.

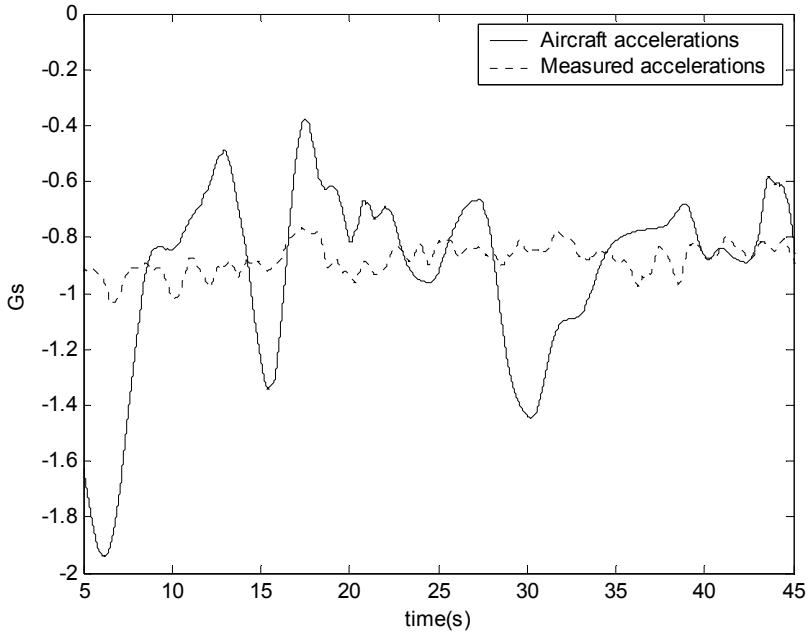

Figure 5.13 – Aircraft/Simulator Acceleration Comparison for CW - 3 (F.C. I)

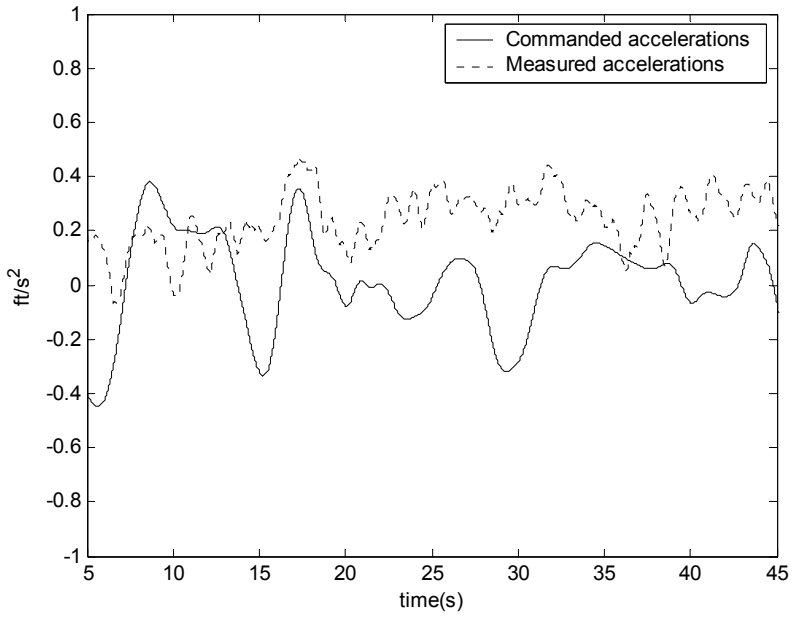

Figure 5.14 – Algorithm Commanded/Twice the Measured Acceleration Comparison for CW - 3 (F.C. I)

5.4.3.6 Configuration CW – 4 (F.C. I)

Figure 5.15 presents the aircraft/simulator acceleration comparison for configuration CW – 4 (Sc_2 = 1.4, τ_{IL} = 0.05 s). The correlation between the aircraft cockpit accelerations and the accelerations measured in the simulator cockpit were fairly good and similar to configuration CW – 1. The comparison between commanded accelerations and measured accelerations as shown in figure 5.16 were very good, particularly when the phase differences are examined.

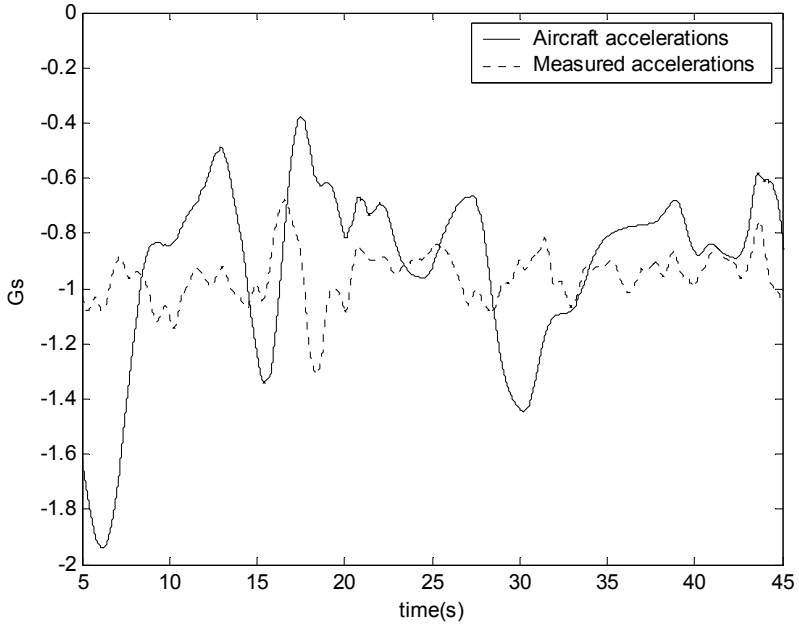

Figure 5.15 – Aircraft/Simulator Acceleration Comparison for CW - 4 (F.C. I)

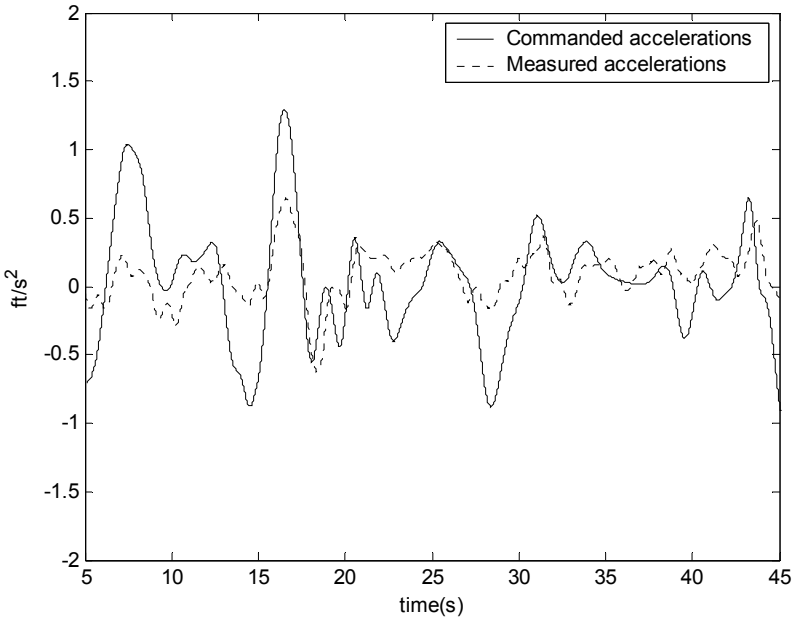

Figure 5.16 – Algorithm Commanded/Twice the Measured Acceleration Comparison for CW - 4 (F.C. I)

5.4.4 Flight Condition II Time Histories

The *z*-direction cockpit accelerations in figure 5.3 were used as input for the twelve parameter configurations for this flight condition. The parameter configurations were analyzed in the order at which the pilot ratings were taken. Similar to flight condition I, the configurations with an overall rating of one were analyzed by using two plots. First, the aircraft's cockpit *z*-direction accelerations (a_z) and the simulator's measured *z*-direction accelerations (a_{sensed}) were plotted. Secondly, the commanded accelerations (a_{zHF}) by the motion algorithm were plotted with the measured accelerations (a_{sensed}).

5.4.4.1 Configuration AF2 – 1 (F.C. II)

Figure 5.17 displays the aircraft/simulator acceleration comparison as described above for configuration AF2 - 1 (K_p = 0.5, τ_{IL} = 2 s, τ_{OL} = 1 s). The plot shows a very good correlation between the two accelerations of interest. Figure 5.18 shows that the measured acceleration response strayed a bit from the commanded, but overall matched nicely, especially in the category of phase difference.

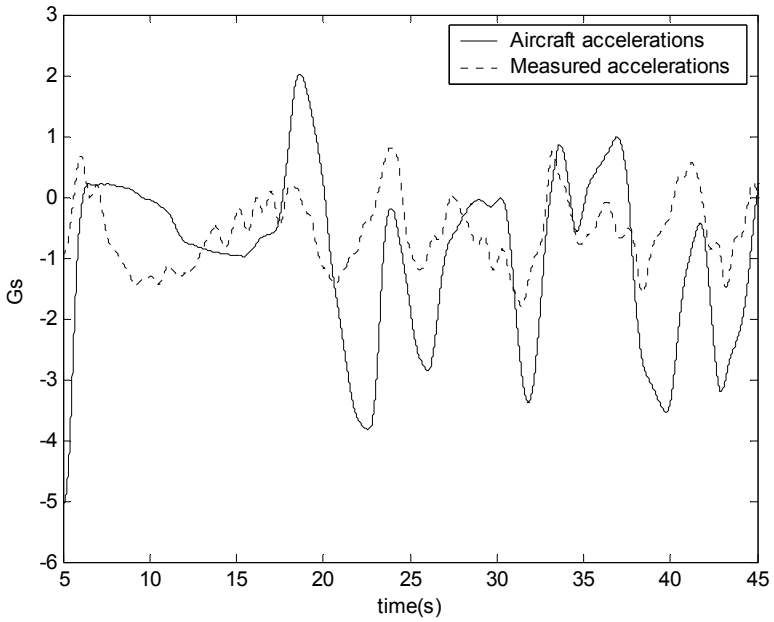

Figure 5.17 – Aircraft/Simulator Acceleration Comparison for AF2 - 1 (F.C. II)

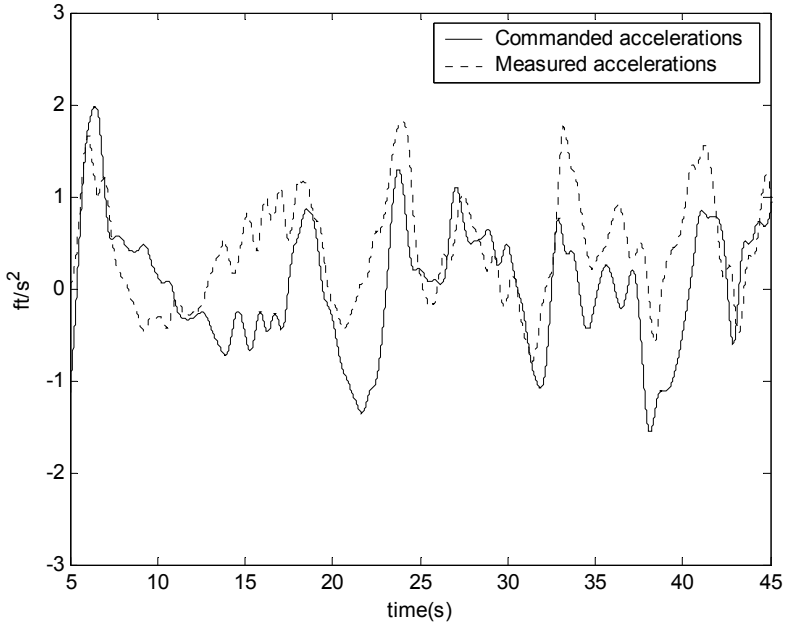

Figure 5.18 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF2 - 1 (F.C. II)

5.4.4.2 Configuration AF1 – 3 (F.C. II)

Figure 5.19 displays the aircraft/simulator acceleration comparison as described above for configuration AF1 - 3 (K_p = 0.3, τ_{IL} = 1 s). The plot shows a good correlation within the constraints of the motion-base. After about twenty seconds the accelerations measured matched very closely to the aircraft accelerations. Figure 5.20 also shows that the measured acceleration response strayed a bit more from the commanded than the B – 1 configuration.

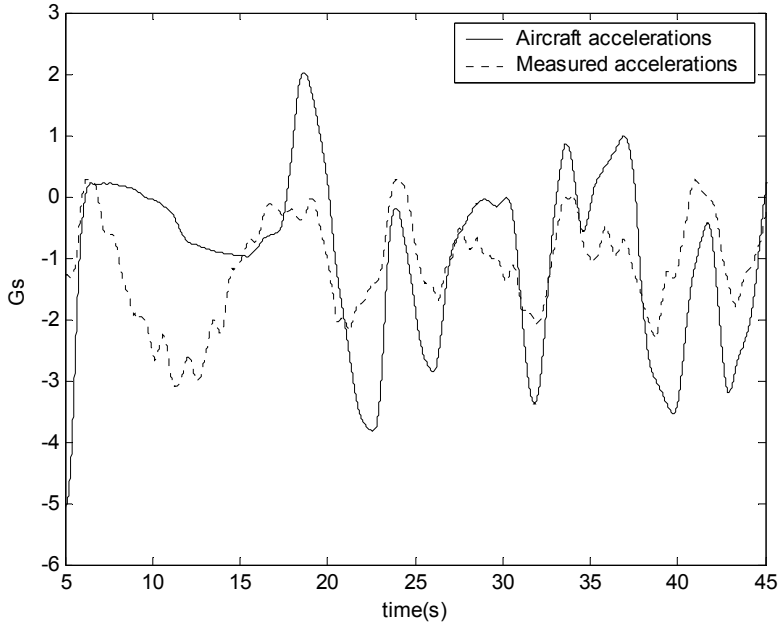

Figure 5.19 – Aircraft/Simulator Acceleration Comparison for AF1 - 3 (F.C. II)

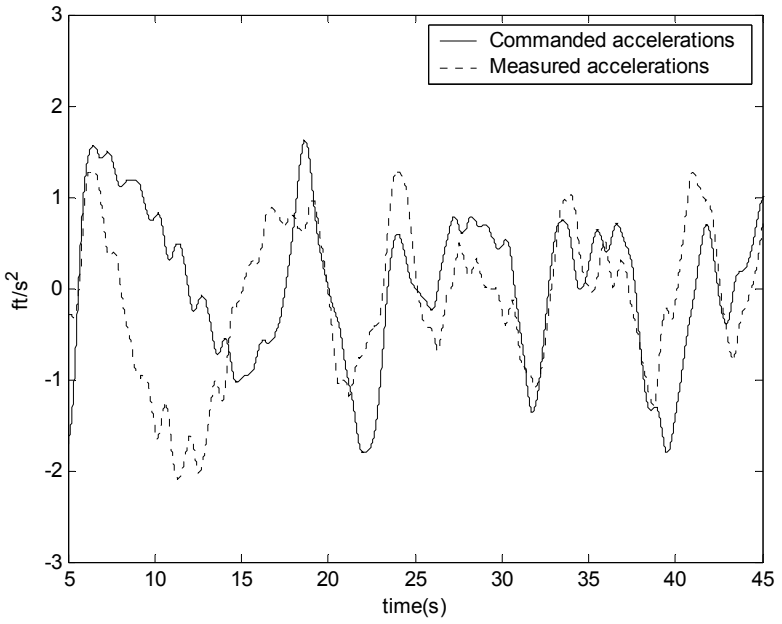

Figure 5.20 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF1 - 3 (F.C. II)

5.4.4.3 Configuration AF1 – 4 (F.C. II)

Figure 5.21 displays the aircraft/simulator acceleration comparison for configuration AF1 - 4 (K_p = 0.5, τ_{IL} = 0.3 s). The plot shows a good correlation within the constraints of the motion-base. After about twenty seconds the accelerations measured matched well with the aircraft accelerations. Figure 5.22 also shows the measured acceleration response matched the commanded accelerations very well.

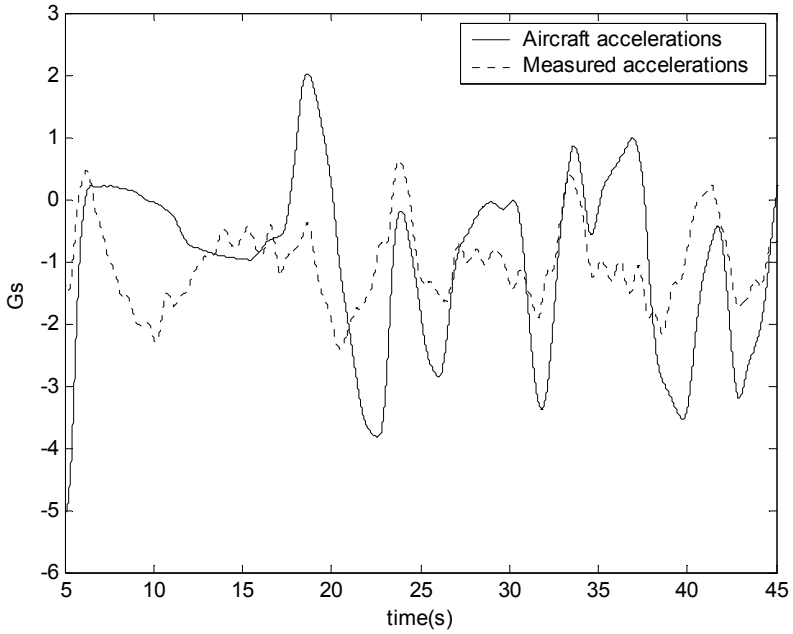

Figure 5.21 – Aircraft/Simulator Acceleration Comparison for AF1 - 4 (F.C. II)

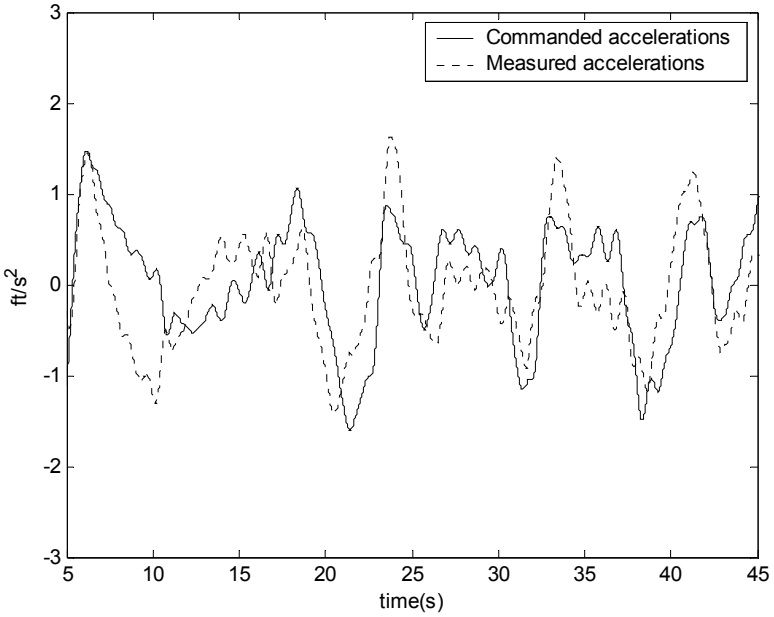

Figure 5.22 – Algorithm Commanded/Twice the Measured Acceleration Comparison for AF1 - 4 (F.C. II)

5.4.4.4 Configuration CW – 3 (F.C. II)

Figure 5.23 and 5.24 represent the aircraft/simulator accelerations and commanded/measured comparisons for configuration CW – 3 (Sc_2 = 0.1, τ_{IL} = 0.2 s). Figure 5.23 shows that the acceleration comparison was not as good for this configuration as the previous ones. Conversely, figure 5.24 shows that this configuration's measured accelerations, much like the other CW configurations, followed the commanded accelerations very well.

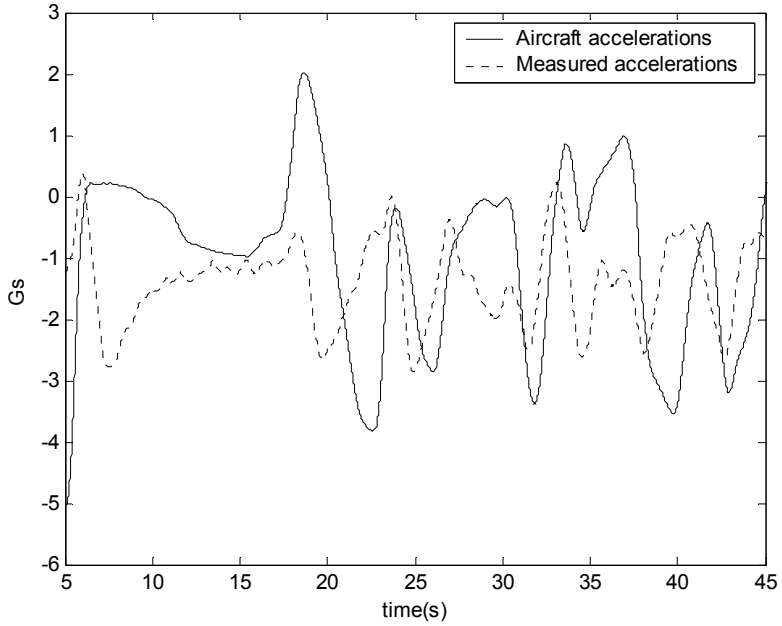

Figure 5.23 – Aircraft/Simulator Acceleration Comparison for CW - 3 (F.C. II)

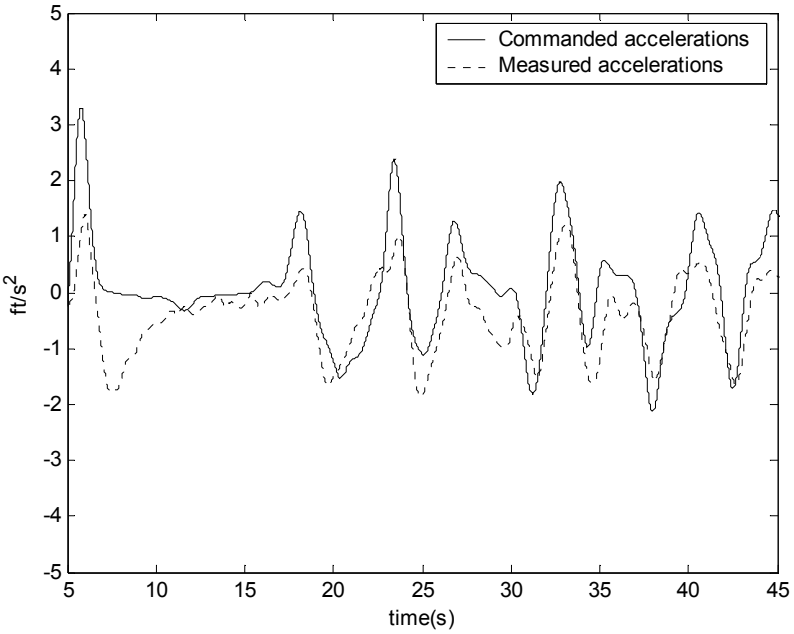

Figure 5.24 – Algorithm Commanded/Twice the Measured Acceleration Comparison for CW - 3 (F.C. II)

5.4.4.5 Configuration CW – 4 (F.C. II)

Figure 5.25 and 5.26 show the aircraft/simulator and commanded/measured accelerations for configuration CW – 4 (Sc_2 = 0.025, τ_{IL} = 1 s). The aircraft/measured comparison shows fairly good correlation. The relationship of acceleration commanded and measured displayed very good response characteristics.

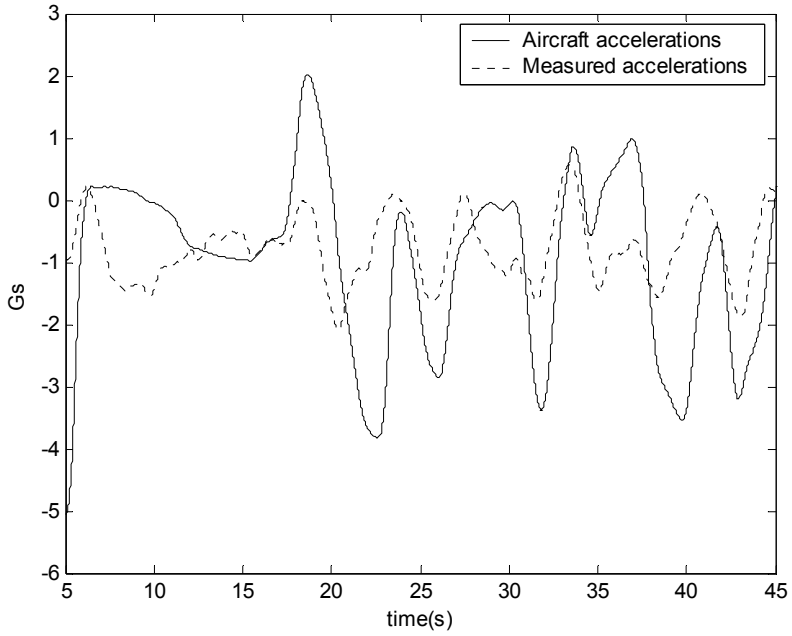

Figure 5.25 – Aircraft/Simulator Acceleration Comparison for CW - 4 (F.C. II)

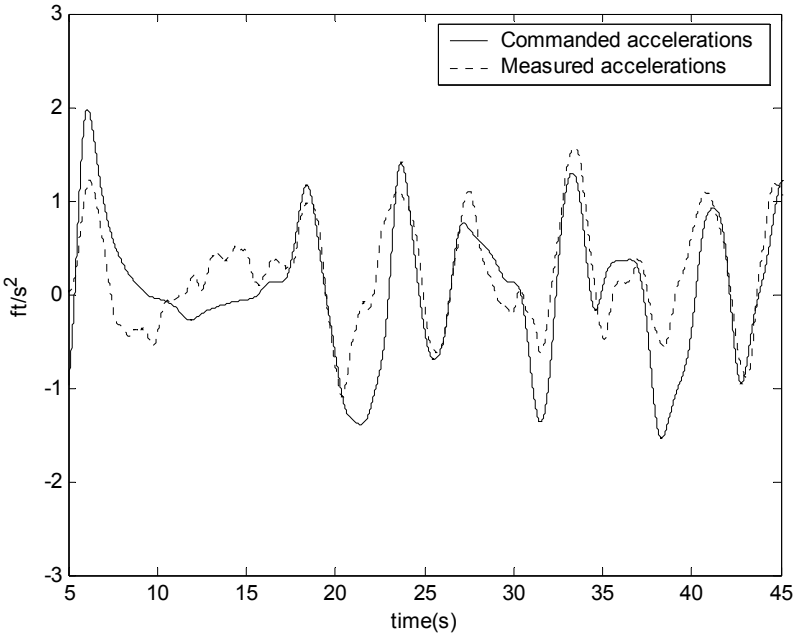

Figure 5.26 – Algorithm Commanded/Twice the Measured Acceleration Comparison for CW - 4 (F.C. II)

5.5 Gain Scheduling

5.5.1 Algorithm Analysis

After the configuration analysis was performed, a final configuration for each flight condition was selected. Because of the adaptation of a feedback gain or scaling for a given dynamic pressure, the same algorithm was used for each flight condition. Switching from one algorithm to another during a simulation would have caused major problems during the transition. A sudden jerk in the motion system would be felt due to the discontinuity that occurred during the transition.

The AF2 configuration with two high-pass filters was removed from consideration due to the fact that it behaved very poorly at the first flight condition. AF1 (acceleration feedback with one high-pass filter) and CW (classical washout) seemed to be evenly matched in that an AF1 configuration was the best for the first flight condition and a CW configuration was the favored configuration for the second flight condition.

The two time history plots that were analyzed yielded some conclusions about the algorithms in general. The AF1 algorithm matched the aircraft/simulator acceleration comparison better than the CW in most instances. Conversely, the CW algorithm achieved better correlation in the commanded/measured acceleration plots.

It must also be stated that the configurations that received an overall rating of one all exhibited acceptable configurations. Due to the closeness in the ratings and decision of a "favorite" configuration, both algorithms were tested using gain scheduling.

5.5.2 Gain Scheduling Selection

It was beneficial to find favorable configurations with the same high-pass filter at both flight conditions. This allowed for the variation of just the feedback gain or scaling factor while keeping the high-pass filter time constant at a constant value.

Two configuration combinations produced the best results. Configuration CW – 2 (F.C. I) and CW – 4 (F.C. II) were programmed to vary scaling factors between 0.08 and 0.025 with a high-pass filter time constant of 1 second. Also, configuration AF1 – 1 (F.C. I) and AF1 – 4 (F.C. II) were programmed to vary feedback gains between 2.5 and 0.6 with a high-pass filter time constant of 0.3 seconds.

After testing both of these configuration combinations throughout the entire flight envelope, the AF1 algorithm (acceleration feedback) was chosen with a high-pass filter time constant of 0.3 seconds. This configuration received better pilot comments than its counterpart. The first flight condition of 2,000 ft and 205 knots yielded a dynamic pressure of 134 lb/ft^2 and a K_P of 2.5 while the second flight condition of 15,000 ft and 440 knots yielded a dynamic pressure of 411 lb/ft^2 and a K_P of 0.6. These gains were linearly varied as shown in figure 5.27. Above a dynamic pressure of 411 lb/ft^2 the feedback gain remains constant at 0.6 to ensure the feedback gain does not become too low or negative. This was the final configuration implemented and produced very favorable comments.

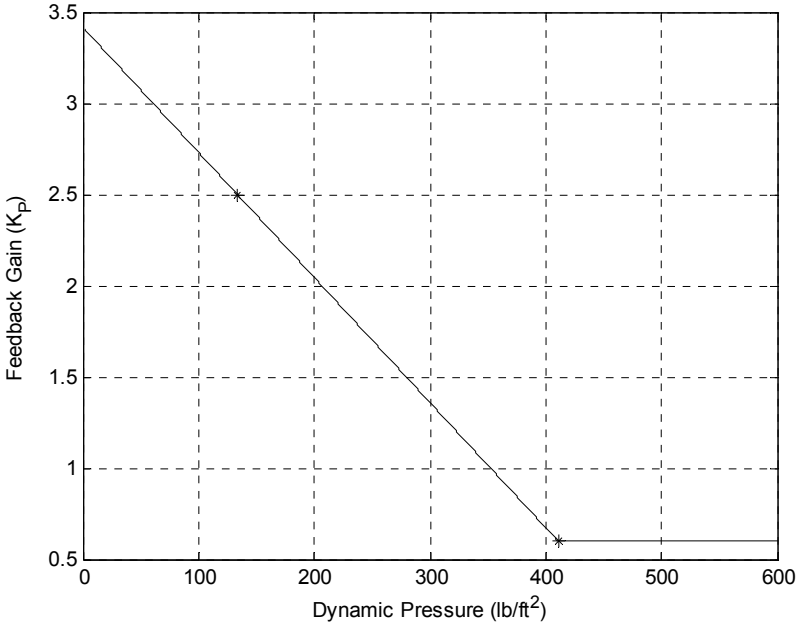

Figure 5.27 – Gain Scheduling Scheme

Chapter 6

Summary and Conclusions

Three motion cueing algorithms were investigated, designed, implemented, and tested in the *CASTLE* simulation environment of Virginia Tech's 2F122A flight simulator. The classical washout algorithm represents the conventional algorithm commonly used in flight simulator motion cueing. The purpose of the washout filters is to produce good motion sensations without extending the hydraulic actuators to their limits, but this algorithm originally produced many pilot complaints due to its undesirable motion cues. Although roll and yaw received pilot complaints too, the majority of the motion sensed in the simulation was longitudinal and thus this axis alone was studied.

Due to its usefulness in numerous control applications from control augmentation systems in aircraft to use in mechanical robot controls, an outer feedback loop was implemented for the longitudinal axis into the classical washout filter algorithm. By measuring the z-direction accelerations sensed in the cockpit via an accelerometer, an algorithm was designed such that aircraft cockpit accelerations calculated from the equations of motion were compared to scaled accelerometer readings to create an error variable. This variable then served as the input into the classical washout algorithm that calculates hydraulic actuator position.

In designing the controller associated with the feedback loop, a system identification analysis was performed in order to obtain a transfer function model of the system. In doing so,

discrepancies in transducer wiring were discovered in the servo cards and repaired. Also, adjustments were made on the inner-loop feedback gains of position and pressure inherent in the motion system to obtain the best responses possible. These two steps alone greatly enhanced the simulation for all three algorithms.

A transfer function representing the motion system and the classical washout algorithm was used to design the feedback gains. Originally, proportional and integral gains were found, but the integral controller was later removed because of integral windup due to accelerometer drift and the high-pass filters. The feedback gains found from the theoretical analysis were used as the starting point in the testing of the different algorithms and their parameter configurations.

A study of the location of the high-pass filter was completed. The idea of placing a high-pass filter with a high break frequency outside the loop and a high-pass filter with a lower break frequency inside the loop was explored and tested as a separate algorithm. The inspiration behind this theory was to achieve better tracking results of the commanded and output variables because the majority of the filtering was performed outside the loop.

Twenty-three parameter configurations encompassing all three algorithms were tested using a pilot rating scale and time history plots. The best parameter configuration for two separate flight conditions were found and gain scheduling was implemented to allow full use of the motion-base throughout all flight conditions.

It was concluded that the configurations with acceleration feedback using one in-loop high-pass filter (AF1) received favorable pilot ratings and the aircraft/simulator acceleration comparison plots produced very good results. The acceleration feedback algorithm with a high-pass filter outside and inside of the loop (AF2) was ruled out due to pilot complaints at low dynamic pressure flight conditions. The classical washout algorithm (CW) did not produce exceptionally favorable results in the aircraft/simulator acceleration comparison plots, but performed very well in the correlation of commanded acceleration from the algorithm to measured accelerations in the cockpit. When gain scheduling was implemented and tested, the test pilot favored the acceleration feedback algorithm (AF1) with proportional feedback gains varying from 2.5 to 0.6 with a high-pass filter time constant of 0.3 seconds.

In conclusion, the motion system fidelity was significantly improved by three major steps. First, the initial tuning of the inner-loop feedback gains inherent in the servo card circuitry played a major role in enhancing the motion system's ability to produce a desirable acceleration

response to inputs into the servo cards. Second, acceleration feedback was proven to improve the classical washout algorithm such that the motion cues created enhanced the acceleration sensations in the cockpit. Finally, gain scheduling served to better utilize the motion system's range of motion for any given flight condition.

Bibliography

[1] Idan, M., and Nahon, M., "Off-Line Comparison of Classical and Robust Flight Simulator Motion Control," *Journal of Guidance, Control, and Dynamics,* Vol. 22, No. 5, 1999, pp. 702-709.

[2] Scalera, K., and Durham, W., "Modification of a Surplus Navy 2F122A A-6E OFT for Flight Dynamics Research and Instruction," AIAA-98-4180, 1998.

[3] Nahon, M., and Reid, L., "Simulator Motion Drive Algorithms – A Designer's Perspective," *Journal of Guidance, Control, and Dynamics,* Vol. 13, No. 2, 1990, pp. 356-362.

[4] Reid, L. and Nahon, M. "Flight Simulator Motion-Base Drive Algorithms: Part 1 – Developing and Testing the Equations," University of Toronto, Ontario, Canada, UTIAS Rept. 296, Dec. 1985.

[5] Reid, L. and Nahon, M. "Flight Simulator Motion-Base Drive Algorithms: Part 2 – Selecting the System Parameters," University of Toronto, Ontario, Canada, UTIAS Rept. 307, May 1986.

[6] Reid, L. and Nahon, M., "Flight Simulator Motion-Base Drive Algorithms: Part 3 – Pilot Evaluations," University of Toronto, Ontario, Canada, UTIAS Rept. 319, Dec. 1986.

[7] Grant, P., and Reid, L., "Motion Washout Filter Tuning: Rules and Requirements," *Journal of Aircraft,* Vol. 34, No. 2, March-April 1997, pp. 145-151.

[8] Pouliot, N., Gosselin, C., and Nahon, M., "Motion Simulation Capabilities of Three-Degree-of-Freedom Flight Simulators," *Journal of Aircraft,* Vol. 35, No. 1, January-February 1998, pp. 9-17.

[9] Stevens, B., and Lewis, F., *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.

[10] Ogata, K., *Modern Control Engineering, Fourth Edition,* Prentice Hall, New Jersey, 2002.

[11] Operation and Maintenance Instructions – Motion System. "A-6E Operational Flight Trainer Device 2F122A," July 15, 1991.

[12] Galloway, R., and Smith, R., "Simulation cue validation using frequency response techniques," AIAA-96-3528, 1996.

[13] "Cooper-Harper Rating Scale," *United States Navy Flight Test website,* 24 September 1997, <http://flighttest.navair.navy.mil/unrestricted/ch.pdf >, (21 April 2004)

Appendix A

Motion Algorithms

A.1 Motion Code Description

The FORTRAN code vpi_motion.f serves to drive the motion-base using the algorithms discussed in this report. The code receives pilot station aircraft accelerations from the equations of motion and calculates the motion commands to be sent to the servo cards/motion system. The accelerometer readings are read through an analog output, converted from volts to G's and low-pass filtered numerically to obtain a clean signal for the controller. The dynamic pressure is also an input to the code such that gain scheduling may be used. All three aforementioned algorithms were implemented and flags were created such that a user may alternate between AF1, AF2, and CW configurations very simply.

A.2 vpi_motion.f

```
C*******************************************************************************
C
C      TITLE:        VPI_MOTION
C
C-------------------------------------------------------------------------------
C
C      FUNCTION:     Drive the 2F122A Motion System
C
C-------------------------------------------------------------------------------
C
C      MODULE TYPE:  GENERIC
C
C-------------------------------------------------------------------------------
C
C      GENEALOGY:    Create from SMMOTN on Gould 32/67
C
C-------------------------------------------------------------------------------
C
C      DESIGNED BY:  Jason Gutridge using Roger Beck's CW original code
C
C      CODED BY:     Jason Gutridge using Roger Beck's CW original code
C
C      MAINTAINED BY:
C
C-------------------------------------------------------------------------------
C
C      MODIFICATION HISTORY:
C
C      DATE          PURPOSE                                    BY
C      ====          =======                                    ==
C`
C
C-------------------------------------------------------------------------------
C
C      REFERENCES:   None.
C
C-------------------------------------------------------------------------------
C
C                           GLOSSARY
C                           ========
C
C      ASSIGNMENTS:
C
C  NONE
C
C-------------------------------------------------------------------------------
C
C      INPUTS:
C
C  AXP              X Acceleration of pilot station          FT/SEC2      FWD
C  AYP              Y Acceleration of pilot station          FT/SEC2      RT
C  AZP              Z Acceleration Of pilot station          FT/SEC2      DOWN
C  DT               Simulation time step (frame time)        SEC          --------
C  IMODE            Sim. mode: -2=init,-1=reset,0=hold,1=run --------     --------
C  PBDEG            Roll rate, body axis                     DEG/SEC      RWD
C  QBAR             Dynamic pressure                         LB/FT2       N/A
C  R2D              57.2957795 Rad To Deg conversion factor  DEG/RAD      --------
C  RTP_AI(16)       Measured pitch position                  VOLTS        --------
```

```
C  RTP_AI(21)       Z-axis accelerometer                    VOLTS        --------
C  RTP_AI(17)       Measured roll position                  VOLTS        --------
C  RTP_AI(18)       Measured yaw position                   VOLTS        --------
C  VRW              Velocity w.r.t. wind (true airspeed)    FT/SEC       N/A
C
C-------------------------------------------------------------------------------
C
C     OUTPUTS:
C
C  RTP_AO(61)       Pitch motion output voltage             VOLTS        --------
C  RTP_AO(62)       Roll motion output voltage              VOLTS        --------
C  RTP_AO(63)       Yaw motion output voltage               VOLTS        --------
C  RTP_AO(64)       Output buffer frequency                 VOLTS        --------
C  RTP_AO(66)       Ouput buffer amplitude                  VOLTS        --------
C
C-------------------------------------------------------------------------------
C
C     LOCALS:
C
C  ACCZ             Unfiltered z-axis accelerometer data    G            DOWN
C  ACC_BIAS         Accelerometer bias                      VOLTS        --------
C  ACC_FILT         Filtered z-axis accelerometer data      G            DOWN
C  ACC_FILT_SCALE   Filtered and scaled z-axis accel. data  G            DOWN
C  ACC_SCALE        Z-axis feedback scale factor            --------     --------
C  ADAPT            Flag for gain scheduling                --------     --------
C  AFDOT            Z-axis accel low-pass state derivative  G/SEC        DOWN
C  AMOTX            Scaled x-motion accelerations           FT/SEC2      FWD
C  AMOTY            Scaled y-motion accelerations           FT/SEC2      RT
C  AMOTZ            Scaled z-motion accelerations           FT/SEC2      DOWN
C  AXPOUT           Motion x accelerations (Low Freq)       FT/SEC2      FWD
C  AXPOUTHI         Motion x accel (High Freq)              FT/SEC2      FWD
C  AXSCALE          Motion x accel scale                    --------     --------
C  AYPOUT           Motion y accel (Low Freq)               FT/SEC2      RT
C  AYPOUTHI         Motion y accel (High Freq)              FT/SEC2      RT
C  AYSCALE          Motion y accel scale                    --------     --------
C  AZPOUT           Motion z accel (Low Freq)               FT/SEC2      DOWN
C  AZPOUTHI         Motion z accel (High Freq) in-loop      FT/SEC2      DOWN
C  AZPOUTHIOL       Motion z accel (High Freq) out-of-loop  FT/SEC2      DOWN
C  AZP_GS           Z A/C acceleration of pilot station     G            DOWN
C  AZP_GS_DOT       Z-axis A/C accel LP state derivative    G/SEC        DOWN
C  AZP_GS_DOTP      Past LP acceleration filter variable    G/SEC        DOWN
C  AZP_GS_FILT      Filtered acceleration variable          G            DOWN
C  AZSCALE          Motion z accel scale (no gain sched.)   --------     --------
C  AZSC_200         Low dynamic press. scaling factor       --------     --------
C  AZSC_500         High dynamic pressure scaling factor    --------     --------
C  ERR              Proportional controller error           G            DOWN
C  ERRP             Past value of P controller error        G            DOWN
C  GMOT             Grav. acceleration in Hancock           --------     --------
C  GOULDMOT         ENABLE original motion                  --------     --------
C  HIRATEPB         Motion roll accel high-pass time const  SEC-1        --------
C  HIRATEX          Motion x accel high-pass time const     SEC-1        --------
C  HIRATEY          Motion y accel high-pass time const     SEC-1        --------
C  HIRATEZ          IL motion z accel high-pass time const  SEC-1        --------
C  HIRATEZOL        OL Motion z accel high-pass time const  SEC-1        --------
C  IERR             Integral controller error               G-SEC        DOWN
C  ILHP             Flag for in-loop high-pass filter       --------     --------
C  KI               Integral gain                           1/SEC        --------
C  KP               Proportional gain                       --------     --------
C  KP_HI            High dynamic press. FB gain             --------     --------
C  KP_LO            Low dynamic press. FB gain              --------     --------
C  LOWRATEX         Motion x accel low-pass time const      SEC-1        --------
C  LOWRATEY         Motion y accel low-pass time const      SEC-1        --------
C  LOWRATEZ         Motion z accel low-pass time Const      SEC-1        --------
```

```
C  LPBF             Z-axis accelerometer low-pass break freq RAD/SEC      --------
C  MOTION_PITCH     Commanded pitch motion                   VOLTS        UP
C  MOTION_ROLL      Commanded roll motion                    VOLTS        CW
C  MOTION_YAW       Commanded yaw motion                     VOLTS        LT
C  OLHP             Flag for out-of-loop high-pass filter    --------     --------
C  PBHIFILT         Roll rate high-pass filter state         --------     --------
C  PBHIFILTD        P high-pass filter state derivative      --------     --------
C  PBMOT            Scaled A/C roll rate                     DEG/S        CCW
C  PBOUTHI          Motion roll rate accel (High Freq)       DEG/S        CCW
C  PBSCALE          Motion roll scaling                      --------     --------
C  PHIHI            Roll position command                    DEG          CCW
C  PHIHIP           Roll position past value                 DEG          CCW
C  PHIMOT           Tilt-coordination for roll               DEG          CCW
C  PHIMOTC          Current tilt-coordination for roll       DEG          CCW
C  PHIMOTP          Previous roll due to tilt-coordination   DEG          CCW
C  PIMOT            ENABLE PI Controller                     --------     --------
C  PITCHPOS         Motion pitch position                    DEG          CCW
C  PMOTCUE          Slow system down when reaching the lims  --------     --------
C  PMOTCUEP         Past value                               --------     --------
C  PPOSD            Actual pitch angle                       DEG          CCW
C  PSIHI            Motion Yaw due to high frequencies       DEG          CCW
C  PSIHIDOT         Mot yaw rate due to high frequencies     DEG/S        CCW
C  PSIHIDOTP        Prev. mot yaw rate due to high freq.     DEG/S        CCW
C  PSIHIP           Prev. motion yaw due to high freq        DEG          CCW
C  PSIMOT           Motion yaw pos low freq                  DEG          CCW
C  QDOTLIM          Limit on speed of rotation for q         DEG/S        CCW
C  QDOTMOT          Motion pitch accel                       DEG/S2       CCW
C  QDOTMOTP         Motion pitch accel previous              DEG/S2       CCW
C  RDOTLIM          Limit on speed of rotation for r         DEG/S        CCW
C  RDOTMOT          Motion yaw accel                         DEG/S2       CCW
C  RDOTMOTP         Previous motion accel rate               DEG/S2       CCW
C  ROLLPOS          Motion roll position                     DEG          CCW
C  TEST             Flag for any test programmed             --------     --------
C  THETHI           Motion pitch due to high frequencies     DEG          CCW
C  THETHIDOT        Mot pitch rate due to high frequencies   DEG/S        CCW
C  THETHIDOTP       Prev mot pitch rate due to high freq.    DEG/S        CCW
C  THETHIP          Prev mot pitch due to high frequencies   DEG          CCW
C  THETMOT          Tilt-coordination angle for pitch        DEG          CCW
C  THETMOTC         Curr. tilt-coordination angle for pitch  DEG          CCW
C  THETMOTP         Past tilt-coordination angle for pitch   DEG          CCW
C  TILTLIMMODE      Mode of tilt-coordination                --------     --------
C  TLTLIM_PH        Tilt-coordination phi limit              DEG          CCW
C  TLTLIM_TH        Tilt-coordination theta limit            DEG          CCW
C  TLTRAT           Rate limit of tilt-coordination          DEG/S        CCW
C  XHIFILT          X high-pass filter state                 DEG          CCW
C  XHIFILTD         X high-pass filter state derivative      DEG/S        CCW
C  XLOWD            X low-pass filter state derivative       DEG          CCW
C  XMP              X pos of pilot station wrt mot pivot     FT           FWD
C  YAWPOS           Motion yaw position                      DEG          CCW
C  YHIFILT          Y high-pass filter state                 DEG          CCW
C  YHIFILTD         Y high-pass filter state derivative      DEG/S        CCW
C  YLOWD            Y low-pass filter state derivative       DEG          CCW
C  ZHIFILT          IL z high-pass filter state              DEG          CCW
C  ZHIFILTD         IL z high-pass filter state derivative   DEG/S        CCW
C  ZHIFILTDOL       OL z high-pass filter state              DEG/S        CCW
C  ZHIFILTOL        OL z high-pass filter state derivative   DEG          CCW
C  ZLOWD            Z low-pass filter state derivative       DEG          CCW
C
C-------------------------------------------------------------------------------
C
C-------------------------------------------------------------------------------

      SUBROUTINE   VPI_MOTION
```

```
C-------------------------------------------------------------------------------
C
C      DECLARATION SECTION
C
C-------------------------------------------------------------------------------
       IMPLICIT NONE


C             ** INPUTS:

       INTEGER*4 IMODE
       REAL*4 AYP, AZP, DT, AXP, PBDEG, QBAR, R2D, RTP_AI(48), VRW

C             ** OUTPUTS:

       REAL*4 RTP_AO(104)

C             ** LOCALS:

       INTEGER*4 ADAPT, GOULDMOT, ILHP, OLHP, PIMOT, TEST
       REAL*4 AFDOT, AMOTX, AMOTY, AMOTZ, AXPOUT, AXPOUTHI, AXSCALE
       REAL*4 AYPOUT, AYPOUTHI, AYSCALE, AZPOUT, AZPOUTHI, AZPOUTHIOL
       REAL*4 AZP_GS, AZP_GS_DOT, AZP_GS_DOTP, AZP_GS_FILT, AZSCALE
       REAL*4 AZSC_200, AZSC_500, ERR, ERRP, GMOT, ACC_BIAS, HIRATEPB
       REAL*4 HIRATEX, HIRATEY, HIRATEZ, HIRATEZOL, IERR, ACC_FILT, KI
       REAL*4 KP, KP_HI, KP_LO, LOWRATEX, LOWRATEY, LOWRATEZ, LPBF
       REAL*4 MOTION_BUFFAMP, MOTION_BUFFREQ, MOTION_PITCH
       REAL*4 MOTION_ROLL, MOTION_YAW, ACC_FILT_SCALE, PBHIFILT
       REAL*4 PBHIFILTD, PBMOT, PBOUTHI, PBSCALE, PHIHI, PHIHIP
       REAL*4 PHIMOT, PHIMOTC, PHIMOTP, ACC_SCALE, PITCHPOS, PMOTCUE
       REAL*4 PMOTCUEP, PPOSD, PSIHI, PSIHIDOT, PSIHIDOTP, PSIHIP
       REAL*4 PSIMOT, QDOTLIM, QDOTMOT, QDOTMOTP, RDOTLIM, RDOTMOT
       REAL*4 RDOTMOTP, ROLLPOS, TAU_PHI, TAU_PSI, TAU_PSID, TAU_THET
       REAL*4 TAU_THETD, ACCZ, THETHI, THETHIDOT, THETHIDOTP, THETHIP
       REAL*4 THETMOT, THETMOTC, THETMOTP, TILTLIMMODE, TLTLIM_PH
       REAL*4 TLTLIM_TH, TLTRAT, XHIFILT, XHIFILTD, XLOWD, XMP
       REAL*4 YAWDLIM, YAWLIM, YAWPOS, YHIFILT, YHIFILTD, YLOWD, YMP
       REAL*4 ZHIFILT, ZHIFILTD, ZHIFILTDOL, ZHIFILTOL, ZLOWD

C             ** OTHER LOCALS:

       BYTE CONPAR, KONSTANTS, LCLBUF, SIMPAR, S_COCKPIT_BUF, VPI
       LOGICAL*4 INITIALIZED
       INTEGER*4 J
       REAL*4 INIT7, INIT8, INIT5, INIT6, AFDOTP, INIT1, PDOTCUE_FTP
       REAL*4 QDOTNEG_FTP, QDOTPOS_FTP, RDOTCUE_FTP, INIT2, INIT3
       REAL*4 INIT4

C-------------------------------------------------------------------------------
C
C      COMMON SECTION
C
C-------------------------------------------------------------------------------
       COMMON/ SHELL1 / CONPAR(424)
       COMMON/ SHELL2 / SIMPAR(1024)
       COMMON/ KONSTANT / KONSTANTS(40)
       COMMON/ VPI / VPI(1324)
       COMMON/ BK_VPI_MOTION / LCLBUF(444)

C-------------------------------------------------------------------------------
C
C      EQUIVALENCE SECTION
```

```
C
C-------------------------------------------------------------------------------

C             ** INPUTS:

      EQUIVALENCE( SIMPAR(285), AXP)
      EQUIVALENCE( SIMPAR(289), AYP)
      EQUIVALENCE( SIMPAR(293), AZP)
      EQUIVALENCE( CONPAR(145), DT)
      EQUIVALENCE( CONPAR(1), IMODE)
      EQUIVALENCE( SIMPAR(625), PBDEG)
      EQUIVALENCE( SIMPAR(549), QBAR)
      EQUIVALENCE( KONSTANTS(9), R2D)
      EQUIVALENCE( VPI(481), RTP_AI(16))
      EQUIVALENCE( VPI(501), RTP_AI(21))
      EQUIVALENCE( VPI(485), RTP_AI(17))
      EQUIVALENCE( VPI(489), RTP_AI(18))
      EQUIVALENCE( SIMPAR(205), VRW)

C             ** OUTPUTS:

      EQUIVALENCE( VPI(245), RTP_AO(61))
      EQUIVALENCE( VPI(249), RTP_AO(62))
      EQUIVALENCE( VPI(253), RTP_AO(63))
      EQUIVALENCE( VPI(257), RTP_AO(64))
      EQUIVALENCE( VPI(265), RTP_AO(66))

C             ** LOCALS:

      EQUIVALENCE( LCLBUF(1), ACCZ)
      EQUIVALENCE( LCLBUF(5), ACC_BIAS)
      EQUIVALENCE( LCLBUF(9), ACC_FILT)
      EQUIVALENCE( LCLBUF(13), ACC_FILT_SCALE)
      EQUIVALENCE( LCLBUF(17), ACC_SCALE)
      EQUIVALENCE( LCLBUF(21), AFDOT)
      EQUIVALENCE( LCLBUF(25), AMOTX)
      EQUIVALENCE( LCLBUF(29), AMOTY)
      EQUIVALENCE( LCLBUF(33), AMOTZ)
      EQUIVALENCE( LCLBUF(37), AXPOUT)
      EQUIVALENCE( LCLBUF(41), AXPOUTHI)
      EQUIVALENCE( LCLBUF(45), AXSCALE)
      EQUIVALENCE( LCLBUF(49), AYPOUT)
      EQUIVALENCE( LCLBUF(53), AYPOUTHI)
      EQUIVALENCE( LCLBUF(57), AYSCALE)
      EQUIVALENCE( LCLBUF(61), AZPOUT)
      EQUIVALENCE( LCLBUF(65), AZPOUTHI)
      EQUIVALENCE( LCLBUF(69), AZPOUTHIOL)
      EQUIVALENCE( LCLBUF(73), AZP_GS)
      EQUIVALENCE( LCLBUF(77), AZP_GS_DOT)
      EQUIVALENCE( LCLBUF(81), AZP_GS_DOTP)
      EQUIVALENCE( LCLBUF(85), AZP_GS_FILT)
      EQUIVALENCE( LCLBUF(89), AZSCALE)
      EQUIVALENCE( LCLBUF(93), AZSC_200)
      EQUIVALENCE( LCLBUF(97), AZSC_500)
      EQUIVALENCE( LCLBUF(101), ERR)
      EQUIVALENCE( LCLBUF(105), ERRP)
      EQUIVALENCE( LCLBUF(109), GMOT)
      EQUIVALENCE( LCLBUF(113), HIRATEPB)
      EQUIVALENCE( LCLBUF(117), HIRATEX)
      EQUIVALENCE( LCLBUF(121), HIRATEY)
      EQUIVALENCE( LCLBUF(125), HIRATEZ)
      EQUIVALENCE( LCLBUF(129), HIRATEZOL)
      EQUIVALENCE( LCLBUF(133), IERR)
```

```
EQUIVALENCE( LCLBUF(137), KI)
EQUIVALENCE( LCLBUF(141), KP)
EQUIVALENCE( LCLBUF(145), KP_HI)
EQUIVALENCE( LCLBUF(149), KP_LO)
EQUIVALENCE( LCLBUF(153), LOWRATEX)
EQUIVALENCE( LCLBUF(157), LOWRATEY)
EQUIVALENCE( LCLBUF(161), LOWRATEZ)
EQUIVALENCE( LCLBUF(165), LPBF)
EQUIVALENCE( LCLBUF(169), MOTION_BUFFAMP)
EQUIVALENCE( LCLBUF(173), MOTION_BUFFREQ)
EQUIVALENCE( LCLBUF(177), MOTION_PITCH)
EQUIVALENCE( LCLBUF(181), MOTION_ROLL)
EQUIVALENCE( LCLBUF(185), MOTION_YAW)
EQUIVALENCE( LCLBUF(189), PBHIFILT)
EQUIVALENCE( LCLBUF(193), PBHIFILTD)
EQUIVALENCE( LCLBUF(197), PBMOT)
EQUIVALENCE( LCLBUF(201), PBOUTHI)
EQUIVALENCE( LCLBUF(205), PBSCALE)
EQUIVALENCE( LCLBUF(209), PHIHI)
EQUIVALENCE( LCLBUF(213), PHIHIP)
EQUIVALENCE( LCLBUF(217), PHIMOT)
EQUIVALENCE( LCLBUF(221), PHIMOTC)
EQUIVALENCE( LCLBUF(225), PHIMOTP)
EQUIVALENCE( LCLBUF(229), PITCHPOS)
EQUIVALENCE( LCLBUF(233), PMOTCUE)
EQUIVALENCE( LCLBUF(237), PMOTCUEP)
EQUIVALENCE( LCLBUF(241), PPOSD)
EQUIVALENCE( LCLBUF(245), PSIHI)
EQUIVALENCE( LCLBUF(249), PSIHIDOT)
EQUIVALENCE( LCLBUF(253), PSIHIDOTP)
EQUIVALENCE( LCLBUF(257), PSIHIP)
EQUIVALENCE( LCLBUF(261), PSIMOT)
EQUIVALENCE( LCLBUF(265), QDOTLIM)
EQUIVALENCE( LCLBUF(269), QDOTMOT)
EQUIVALENCE( LCLBUF(273), QDOTMOTP)
EQUIVALENCE( LCLBUF(277), RDOTLIM)
EQUIVALENCE( LCLBUF(281), RDOTMOT)
EQUIVALENCE( LCLBUF(285), RDOTMOTP)
EQUIVALENCE( LCLBUF(289), ROLLPOS)
EQUIVALENCE( LCLBUF(293), TAU_PHI)
EQUIVALENCE( LCLBUF(297), TAU_PSI)
EQUIVALENCE( LCLBUF(301), TAU_PSID)
EQUIVALENCE( LCLBUF(305), TAU_THET)
EQUIVALENCE( LCLBUF(309), TAU_THETD)
EQUIVALENCE( LCLBUF(313), THETHI)
EQUIVALENCE( LCLBUF(317), THETHIDOT)
EQUIVALENCE( LCLBUF(321), THETHIDOTP)
EQUIVALENCE( LCLBUF(325), THETHIP)
EQUIVALENCE( LCLBUF(329), THETMOT)
EQUIVALENCE( LCLBUF(333), THETMOTC)
EQUIVALENCE( LCLBUF(337), THETMOTP)
EQUIVALENCE( LCLBUF(341), TILTLIMMODE)
EQUIVALENCE( LCLBUF(345), TLTLIM_PH)
EQUIVALENCE( LCLBUF(349), TLTLIM_TH)
EQUIVALENCE( LCLBUF(353), TLTRAT)
EQUIVALENCE( LCLBUF(357), XHIFILT)
EQUIVALENCE( LCLBUF(361), XHIFILTD)
EQUIVALENCE( LCLBUF(365), XLOWD)
EQUIVALENCE( LCLBUF(369), XMP)
EQUIVALENCE( LCLBUF(373), YAWDLIM)
EQUIVALENCE( LCLBUF(377), YAWLIM)
EQUIVALENCE( LCLBUF(381), YAWPOS)
EQUIVALENCE( LCLBUF(385), YHIFILT)
```

```
      EQUIVALENCE( LCLBUF(389), YHIFILTD)
      EQUIVALENCE( LCLBUF(393), YLOWD)
      EQUIVALENCE( LCLBUF(397), YMP)
      EQUIVALENCE( LCLBUF(401), ZHIFILT)
      EQUIVALENCE( LCLBUF(405), ZHIFILTD)
      EQUIVALENCE( LCLBUF(409), ZHIFILTDOL)
      EQUIVALENCE( LCLBUF(413), ZHIFILTOL)
      EQUIVALENCE( LCLBUF(417), ZLOWD)
      EQUIVALENCE( LCLBUF(421), PIMOT)
      EQUIVALENCE( LCLBUF(425), ADAPT)
      EQUIVALENCE( LCLBUF(429), TEST)
      EQUIVALENCE( LCLBUF(433), OLHP)
      EQUIVALENCE( LCLBUF(437), GOULDMOT)
      EQUIVALENCE( LCLBUF(441), ILHP)

C-------------------------------------------------------------------------------
C
C     DATA SECTION
C
C-------------------------------------------------------------------------------
      DATA INITIALIZED / .FALSE. /

       DATA GMOT / 32.174 /
       DATA XMP /   12.0   /   !maybe this is right
       DATA YMP /   4.0    /   !maybe this is right
C-------------------------------------------------------------------------------
C
C     INITIALIZATION SECTION
C
C-------------------------------------------------------------------------------

      IF( IMODE.LE.-2 .OR. .NOT. Initialized ) THEN
        Initialized = .TRUE.

         TAU_THET  = 1.5
         TAU_THETD = 0.3
         TAU_PSI   = 0.3
         TAU_PSID  = 0.3
         TAU_PHI   = 0.3
         LOWRATEX = 5.0
         LOWRATEY = 5.0
         LOWRATEZ = 5.0
         HIRATEX  = 1.0
         HIRATEY  = 1.0
         HIRATEZ  = 0.3
         HIRATEZOL = 4
         HIRATEPB = 1.0
         AXSCALE  = 0.4
         AYSCALE  = 0.1
         AZSCALE  = 0.1
         PBSCALE = 0.15
         J=0
         INIT1=0
         INIT2=0
         INIT3=0
         INIT4=0
         QDOTLIM = 9999.0
         RDOTLIM = 9999.0
         MOTION_BUFFAMP = 0.0
         MOTION_BUFFREQ = 0.0
         TLTLIM_PH = 3.0
         TLTLIM_TH = 3.0
         GOULDMOT = 0
```

```
         LPBF = 3 ! RAD/S
         PIMOT = 1
         ILHP = 1
         ACC_SCALE = 50.0 ! accelerometer scale value to reduce feedback error
         ACC_BIAS = 0.0 ! accelerometer bias control for PI feedback
         AZSC_200 = 0.3
         AZSC_500 = 0.1
         KP = 1.5 ! Controller P gain
         KI = 0.0 ! Controller I gain
         ADAPT = 1
         TEST = 1
         OLHP = 0
         KP_LO = 2.5
         KP_HI = 0.6
       ENDIF

C-------------------------------------------------------------------------------
C
C      RESET SECTION
C
C-------------------------------------------------------------------------------

       IF( IMODE.LT.0 .OR. .NOT. Initialized ) THEN
            MOTION_BUFFAMP = 0.0
            MOTION_BUFFREQ = 0.0
            QDOTMOTP   = 0.0
            THETHIDOTP = 0.0
            THETHIP    = 0.0
            RDOTMOTP   = 0.0
            PSIHIDOTP  = 0.0
            PSIHIP     = 0.0
            PHIHIP     = 0.0
            PMOTCUE    = 0.0

            AXPOUT = AXSCALE*AXP
            AYPOUT = AYSCALE*AYP
            AZPOUT = 0

            XHIFILT = AXPOUT
            YHIFILT = AYPOUT
            ZHIFILT = AZPOUT
            ZHIFILTOL = AZPOUT
            PBHIFILT = PBDEG*PBSCALE
            J = 0
            INIT1=0
            INIT2=0
            INIT3=0
            INIT4=0
            ACC_FILT = -1
            AZP_GS_FILT = 0.0
            ERR = 0.0
            IERR = 0.0
            ERRP = 0.0
            AZP = -GMOT
            PITCHPOS = 0.0
            PPOSD = 0.0
       ENDIF

C-------------------------------------------------------------------------------
C
C      RUN SECTION
C
C-------------------------------------------------------------------------------
```

```
C  Read in initial values for the accelerometers rest voltage
C  Starts at 51 (such that the initial movement doesn't get averaged)
C  Reads 8 samples and takes the mean

       PPOSD = 2.473*RTP_AI(16)

       IF (J.EQ.50) THEN
           INIT1=RTP_AI(21)
        ENDIF
       IF (J.EQ.55) THEN
           INIT2=RTP_AI(21)
        ENDIF
       IF (J.EQ.60) THEN
           INIT3=RTP_AI(21)
        ENDIF
       IF (J.EQ.65) THEN
           INIT4=RTP_AI(21)
        ENDIF
       IF (J.EQ.70) THEN
           INIT5=RTP_AI(21)
        ENDIF
       IF (J.EQ.75) THEN
           INIT6=RTP_AI(21)
        ENDIF
       IF (J.EQ.80) THEN
           INIT7=RTP_AI(21)
        ENDIF
       IF (J.EQ.85) THEN
           INIT8=RTP_AI(21)
        ENDIF
C
C  Make low-pass filter to get rid of accelerometer noise
C
       AFDOT = (ACCZ-ACC_FILT)*LPBF
       IF (J.LT.85) THEN
              ACC_FILT_SCALE = 0
       ENDIF
       IF (J.GT.85) THEN
C
C  Averaging of 8 samples to obtain true bias
C
              ACC_BIAS=(INIT1+INIT2+INIT3+INIT4+INIT5+INIT6+INIT7+INIT8)/8
C
C  Conversion from volts to G's
C
       IF (TEST.EQ.0) THEN
              ACCZ = -(RTP_AI(21)- ACC_BIAS)*2-1
       ELSE
              ACCZ = -(RTP_AI(21)- (ACC_BIAS*(COSD(PPOSD))))*2-1
       ENDIF
C
C  Adams-Bashforth integration of low-pass filter
C
              ACC_FILT = ACC_FILT + DT/2.0*(3*AFDOT - AFDOTP)
              AFDOTP = AFDOT
              ACC_FILT_SCALE = ACC_FILT+1


       AZP_GS = (AZP/GMOT+1)/ACC_SCALE
       AZP_GS_DOT = (AZP_GS - AZP_GS_FILT)*LPBF
       AZP_GS_FILT = AZP_GS_FILT +DT/2.0*(3*AZP_GS_DOT - AZP_GS_DOTP)
       AZP_GS_DOTP = AZP_GS_DOT
```

```
C
C      Out of loop HP filter
C

       IF (OLHP .EQ. 1) THEN
              ZHIFILTDOL = (AZP_GS_FILT-ZHIFILTOL)/HIRATEZOL
              ZHIFILTOL = ZHIFILTOL+ZHIFILTDOL*DT
              AZPOUTHIOL = AZP_GS_FILT-ZHIFILTOL
       ELSE
              AZPOUTHIOL = AZP_GS_FILT
       ENDIF
       ENDIF

       J=J+1
        IF (GOULDMOT .EQ. 1) THEN
            CALL VPI_MOT_GOULD
            RETURN
        ENDIF
C
C  PI Controller Code for AZP
C
       IF (PIMOT .EQ. 1) THEN
          ERR = (AZPOUTHIOL - ACC_FILT_SCALE)*GMOT*ACC_SCALE ! make error
         IERR = IERR + 0.5*DT*(3*ERR-ERRP) ! Adams-Bashforth Integration
          ERRP = ERR
       ENDIF

C  Limit Constants which appear on the bottom of equations
        LOWRATEX = AMAX1(0.01,LOWRATEX)
        LOWRATEY = AMAX1(0.01,LOWRATEY)
        LOWRATEZ = AMAX1(0.01,LOWRATEZ)
        HIRATEX  = AMAX1(0.01,HIRATEX)
        HIRATEY  = AMAX1(0.01,HIRATEY)
        HIRATEZ  = AMAX1(0.01,HIRATEZ)
        HIRATEPB = AMAX1(0.01,HIRATEPB)

C
C      Do Gain Scheduling
C
       IF (PIMOT.EQ.0) THEN
       IF (ADAPT.EQ.1) THEN
            IF (QBAR.LT.411) THEN
              AZSCALE = ((AZSC_500 - AZSC_200)/277)*QBAR
     &           + AZSC_200*1.484 - 0.484*AZSC_500
            ENDIF
            IF (QBAR.GT.411) THEN
              AZSCALE = ((AZSC_500 - AZSC_200)/277)*411
     &           + AZSC_200*1.484 - 0.484*AZSC_500
            ENDIF
        ENDIF
        ENDIF
       IF (PIMOT.EQ.1) THEN
              IF (ADAPT.EQ.1) THEN
              IF (QBAR.LT.411) THEN
              KP = ((KP_HI - KP_LO)/277)*QBAR +1.484*KP_LO-0.484*KP_HI
              ENDIF
              IF (QBAR.GT.411) THEN
              KP = ((KP_HI - KP_LO)/277)*411 +1.484*KP_LO-0.484*KP_HI
              ENDIF
              ENDIF
       ENDIF

C  Scale Input Forces
```

```
       AMOTX = AXSCALE*AXP
       AMOTY = AYSCALE*AYP
       IF (PIMOT.EQ.1) THEN
C
C  Implement acceleration Feedback if PIMOT is on
C
             AMOTZ = (AZSCALE*(ERR*KP + IERR*KI))
       ELSE
             AMOTZ = AZSCALE*AZP
       ENDIF
        PBMOT = PBSCALE*PBDEG

        XLOWD = (AMOTX-AXPOUT)/LOWRATEX
        AXPOUT = AXPOUT+XLOWD*DT

        YLOWD = (AMOTY-AYPOUT)/LOWRATEY
        AYPOUT = AYPOUT+YLOWD*DT

        ZLOWD = (AMOTZ-AZPOUT)/LOWRATEZ
        AZPOUT = AZPOUT+ZLOWD*DT

C  Rate Limit Tilt Coordination
        THETMOTP = THETMOTC
        PHIMOTP = PHIMOTC
        THETMOTC = ASIND(AXPOUT/GMOT)
        PHIMOTC = ASIND(AYPOUT/(GMOT*COSD(THETMOT)))

C  Assumes motion yaw contribution is zero
C  Two cases one limits rate of output one limits
C  rate of command.

      IF (TILTLIMMODE .EQ. 1) THEN
        TLTRAT = (THETMOTC-THETMOTP)/DT
        IF (ABS(TLTRAT) .GT. TLTLIM_TH) THEN
           THETMOT = THETMOT+SIGN(TLTLIM_TH, TLTRAT)*DT
        ELSE
            THETMOT = THETMOT+TLTRAT*DT
        ENDIF
        TLTRAT = (PHIMOTC-PHIMOTP)/DT
        IF (ABS(TLTRAT) .GT. TLTLIM_PH) THEN
            PHIMOT = PHIMOT+SIGN(TLTLIM_PH, TLTRAT)*DT
        ELSE
            PHIMOT = PHIMOT+TLTRAT*DT
        ENDIF
      ELSEIF (TILTLIMMODE .EQ. 2) THEN
       THETMOT = 0.0
      ELSE
        TLTRAT = (THETMOTC-THETMOT)/DT
        IF (ABS(TLTRAT) .GT. TLTLIM_TH) THEN
            THETMOT = THETMOT+SIGN(TLTLIM_TH, TLTRAT)*DT
        ELSE
            THETMOT = THETMOT+TLTRAT*DT
        ENDIF
        TLTRAT = (PHIMOTC-PHIMOT)/DT
        IF (ABS(TLTRAT) .GT. TLTLIM_PH) THEN
            PHIMOT = PHIMOT+SIGN(TLTLIM_PH, TLTRAT)*DT
        ELSE
            PHIMOT = PHIMOT+TLTRAT*DT
        ENDIF
      ENDIF

C
C     High-Pass filtering
```

```
C

        XHIFILTD = (AMOTX-XHIFILT)/HIRATEX
        XHIFILT = XHIFILT+XHIFILTD*DT
        AXPOUTHI = AMOTX-XHIFILT

        YHIFILTD = (AMOTY-YHIFILT)/HIRATEY
        YHIFILT = YHIFILT+YHIFILTD*DT
        AYPOUTHI = AMOTY-YHIFILT

       IF (ILHP .EQ. 1) THEN
              ZHIFILTD = (AMOTZ-ZHIFILT)/HIRATEZ
             ZHIFILT = ZHIFILT+ZHIFILTD*DT
             AZPOUTHI = AMOTZ-ZHIFILT
       ELSE
             AZPOUTHI = AMOTZ
       ENDIF

       PBHIFILTD = (PBMOT-PBHIFILT)/HIRATEPB
       PBHIFILT = PBHIFILT+PBHIFILTD*DT
       PBOUTHI = PBMOT-PBHIFILT


C    Rotational Velocity
C    Slow down when the system approaches the limits

       PMOTCUE = PBOUTHI*PDOTCUE_FTP(RTP_AI(17)*2.2)

C   Tangential Accel
C   Assume Y = 0.0 and Z moment arm is zero and same psi assumtpion above
C   and all low-pass accels are low


       QDOTMOT = -R2D*(AZPOUTHI)/(XMP*COSD(ROLLPOS))
       IF (QDOTMOT.GT.0) THEN
          QDOTMOT = QDOTMOT*QDOTPOS_FTP(RTP_AI(16)* 3.4)
       ELSE
          QDOTMOT = QDOTMOT*QDOTNEG_FTP(RTP_AI(16)*3.4)
       ENDIF

       RDOTMOT = (-R2D*(AYPOUTHI)/XMP
     &          +QDOTMOT*SIND(ROLLPOS))*RDOTCUE_FTP(RTP_AI(18)*2.2)

       IF (ABS(QDOTMOT).GT.QDOTLIM) THEN
           QDOTMOT = SIGN(QDOTLIM, QDOTMOT)
       ENDIF
       IF (ABS(RDOTMOT).GT.RDOTLIM) THEN
           RDOTMOT = SIGN(RDOTLIM, RDOTMOT)
       ENDIF

C
C  Integration of High Freq results
C

       THETHIDOT = THETHIDOTP+(DT/2.0)*(3*QDOTMOT-QDOTMOTP)
       THETHI = THETHIP+(DT/2.0)*(3*THETHIDOT-THETHIDOTP)

       PSIHIDOT = PSIHIDOTP+(DT/2.0)*(3*RDOTMOT-RDOTMOTP)
       PSIHI = PSIHIP+(DT/2.0)*(3*PSIHIDOT-PSIHIDOTP)

       PHIHI = PHIHIP+(DT/2.0)*(3*PMOTCUE-PMOTCUEP)
C
C   If tilt-coord is desired at THETMOT, PSIMOT, PHIMOT to the following
```

```
C
       PITCHPOS = THETHI
       YAWPOS   = PSIHI
       ROLLPOS  = PHIHI


       PITCHPOS = AMAX1(AMIN1(PITCHPOS, 40.0), -25)
       YAWPOS   = AMAX1(AMIN1(YAWPOS,   10.0), -10.0)
       ROLLPOS  = AMAX1(AMIN1(ROLLPOS,  20.0), -20.0)
*
************************* CREATE PAST VALUES *************************
*
       QDOTMOTP   = QDOTMOT
       THETHIDOTP = THETHIDOT
       THETHIP    = THETHI
       RDOTMOTP   = RDOTMOT
       PSIHIDOTP  = PSIHIDOT
       PSIHIP     = PSIHI
       PHIHIP     = PHIHI
       PMOTCUEP    = PMOTCUE
C*********************************************************************
C*
C*  Calculate Checksums and prepare to send stuff to RTP
C*
C*********************************************************************
      MOTION_PITCH = PITCHPOS*(-0.04)
      MOTION_ROLL  = ROLLPOS*(-0.015)
      MOTION_YAW   = YAWPOS*(0.03)


      RTP_AO(61) = AMAX1(AMIN1(MOTION_PITCH,  0.999),-0.999)
      RTP_AO(62) = AMAX1(AMIN1(MOTION_ROLL,   0.999),-0.999)
      RTP_AO(63) = AMAX1(AMIN1(MOTION_YAW,    0.999),-0.999)
      RTP_AO(64) = AMAX1(AMIN1(MOTION_BUFFREQ,0.999),-0.999)
      RTP_AO(66) = AMAX1(AMIN1(MOTION_BUFFAMP,0.999),-0.999)


C-----------------------------------------------------------------------------
C
C      END OF VPI_MOTION MODULE
C
C-----------------------------------------------------------------------------

       RETURN
       END
```

Appendix B

Controller Design

B.1 Transfer Function Approximation

B.1.1 Description

FR2TF.m serves the purpose of allowing the user to create a transfer function that approximates the experimental frequency response data of a given system. The program first reads in the frequency response data in real and imaginary parts collected from the digital signal analyzer and creates a Bode plot of the data. The user then inputs values of K, ω_n, and ζ (K, wn, zeta). By trial-and-error a transfer function is created that approximates the given system dynamics. A plot is constructed comparing the two bode plots and the transfer function is output in Laplacian form.

B.1.2 FR2TF.m

```
% compares Pitch FR data and 3rd order TF approximation

close all
clear all

% read in FR data (complex) and coherence data from Spectrum Analyzer
frqdat = dlmread('V1b.dat',' ');
% Scale data for accelerometer resolution
frqdat = frqdat*5
cohdat = dlmread('V2b.dat',' ');

% compute magnitude and phase and to reproduce Bode plot
magnitude = ((frqdat(:,1)).^2+(frqdat(:,2)).^2).^.5;
phase = (180/pi)*atan2(frqdat(:,2),frqdat(:,1));
coherence = cohdat(:,1)
freqHZ = 0:(25/400):25;
freq = 2*pi*freqHZ

% The three variables K, wn, and zeta ste the paramters with which
% the trial and error process uses
K = 6500
wn =45
zeta = 1.1/2
s = tf('s')
sys = (K*s)/((s^2+2*zeta*wn*s+wn^2)^2)
[mag1,phase1]=bode(sys,freq);

for i=1:401
    mag(i)=mag1(1,1,i);
    phase2(i)=phase1(1,1,i);
end

% Plot the raw data with the transfer function approximation
figure(1)
subplot(2,1,1);
semilogx(freq,20*log10(magnitude),'k',freq,20*log10(mag),':k')
axis([1 150 -70 20]);
title('Transfer Function Approximation')
ylabel('Magnitude (dB)');
legend('Experimental Data','Transfer Function Approximation')
grid;
subplot(2,1,2); semilogx(freq,phase,'k',freq,phase2,':k')
axis([1 150 -180 180]);
ylabel('Phase (deg)');
xlabel('Frequency (rad/s)')
grid;
sys
```

B.2 PI Controller Design

B.2.1 Description

ClassicalPI.m is the code used to select feedback gains for a P, and PI controller using MATLAB. The program assembles the complete transfer function, given the smaller transfer functions in the classical washout algorithm and the transfer function approximation of the motion system found using FR2TF.m. First, the final value theorem is applied and the transfer function is scaled as explained in section 4.2.3. As one may recall this step is needed in order to account for the fact that the accelerations that the motion system can produce are much smaller than the accelerations in an actual fighter aircraft.

For the P controller design, MATLAB's rlocus command allows the user to select the desired roots off the root locus and create the new transfer function using the feedback command. The rlocus and feedback command allows the user to select the desired eigenvalues and create a new transfer function. These closed loop transfer functions were then plotted using step inputs and sin wave inputs.

B.2.2 ClassicalPI.m

```
close all
clear all

K1=0.1/50;        % initial scaling factor
K2=-180/pi/12; % ft/s2 to deg/s2 (moment arm ~ 12 ft)
K3=-0.04;      % deg to volts
K4=2;         % volts to g's
K5=32.174;     % g's to ft/s2

tf1=tf([.15 0],[.15 1]); % initial high-pass filter
tf2=tf([1],[1 0]);       % first integrator
tf3=tf2;                 % first integrator
% system transfer function (pos cmd (Volts) to accelerometer out (Volts))
tf4=tf([6500 0],[1 99 6500 200500 4101000]);
tf5=tf([1],[.333 1]);     % low-pass filter for accel out
tftot=K1*tf1*K2*tf2*tf3*K3*tf4*tf5*K4*K5; % total transfer function
[num,den]=tfdata(tftot,'v');

% get rid of pole-zero cancellations (cannot control or observe integrators)
num=[num(7)];
den=[den(1:7)];

% get steady-state value of tf (this is the actual desired magnitude - need
% to normalize accel output such that steady-state cmd of 1 ft/s gives a
% steady-state accel output equal to uncontrolled steady-state output) -
% The final value theorem
K7=den(7)/num;
num=num*K7;

% compute final transfer function, state space, root locus for open loop
tfol=tf(num,den);
ssol=ss(tfol);

% create P controller by picking desired root
figure(1);rlocus(tfol,'k');axis([-10 10 -30 30])
pause
[Kp,roots]=rlocfind(tfol)
damp(roots)

% close loop with P control
ssPfb=feedback(Kp*tfol,1); % close the loop
tfPfb=tf(ssPfb);
damp(tfPfb)

% close loop with PI control (Stevens and Lewis method)
PIzero=-3.1;
tfcomp=tf([1 -PIzero],[1 0]);
tfolnew=tfcomp*tfol;
figure(2);rlocus(tfolnew);
axis([-10 5 -20 20]);
pause
[Kp2,roots]=rlocfind(tfolnew)
Ki=-PIzero*Kp2
ssPIfb=feedback(Kp2*tfolnew,1); % close the loop
```

```
tfPIfb=tf(ssPIfb);
damp(tfPIfb)

% examine time histories of step input responses for overshoots
T=0:0.01:5;
[yol]=step(tfol,T); % step input
[yPfb]=step(tfPfb,T); % step input
[yPIfb]=step(tfPIfb,T); % step input
figure(3);plot(T,yol(:,1),'k',T,yPfb(:,1),'k:',T,yPIfb(:,1),'k-.') %
qualitative evaluation
axis([0 4 0 1.3])
legend('OL Response','CL Response with P','CL Response with PI')
xlabel('Time (sec)')
ylabel('y(t)')
title('Step Response')

% examine time histories of sin input responses
T2=0:0.01:12;
w=.75*(2*pi);
[yol2]=lsim(tfol,sin(w*T2),T2); % sin input
[yPfb2]=lsim(tfPfb,sin(w*T2),T2); % sin input
[yPIfb2]=lsim(tfPIfb,sin(w*T2),T2); % sin input
figure(4);plot(T2,sin(w*T2),'k',T2,yol2(:,1),'k:',T2,yPfb2(:,1),'k-
.',T2,yPIfb2(:,1),'k--') % qualitative evaluation
legend('Commanded','OL Response','CL Response with P','CL Response with PI')
xlabel('Time (sec)')
ylabel('y(t)')
axis([0 4 -1.5 1.5])
title('Sine Wave Simulation')
```

Vita

Jason Gutridge was born in Richmond, VA on November 18, 1980 to Robert and Elizabeth Gutridge. He lived in several locations along the east coast during his first eight years before settling back near Richmond in a town called Midlothian, VA. In his early years, Jason spent much of his time playing baseball and roaming the woods around the neighborhood with friends. Jason discovered more friends in his teenage years and he found the sport he was most meant to play, football. After graduating from Midlothian High School, Jason ventured two-hundred miles away to Blacksburg, VA, home of Virginia Polytechnic Institute and State University nestled in beautiful Southwest Virginia. Here Jason earned his Bachelor's Degree in Aerospace Engineering in May of 2002. After graduation, Jason found his way down to the Flight Simulation Laboratory where he studied dynamics and controls for two years, this time earning his Master's Degree in Aerospace Engineering. In his six year career at Virginia Tech, Jason met many great friends and has countless memories that will last him a lifetime.