Modelling and Evaluation of Adaptive Control Techniques in Satellite Orientation during Large Actuator Gain Changes

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Electronic Engineering
at the
University of Stellenbosch

Jacques Rossouw

Promoter: Mr. J. Treurnicht

April, 2004

Declaration

I, Jacques Stéfan Rossouw, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Abstract

Improvements in the area of satellite orientation control in the presence of large actuator gain changes are investigated. Gain changes primarily originate from actuator failures, but may also result from intermittent sensor availability and geomagnetic field effects.

The stability and performance of a classic LQR control solution under these conditions is analyzed through simulation, and two adaptive schemes are developed to improve the response.

The adaptive schemes mix elements from bang-bang control to increase performance, and banded control to increase robustness. These control schemes are thoroughly tested through simulation and the results are compared with those obtained for the classic solution.

Extensive modelling of the system in the MATLAB™ environment is done to support the analysis of the controllers, and hardware sensors are built to provide realistic orientation measurements to the controllers.

Oorsig

Verbeterings in die veld van satelliet oriëntasie beheer in die gevalle waar die aktueerders groot veranderinge in aanwins toon, word ondersoek. Aanwins veranderinge word primêr veroorsaak deur aktueerder falings, maar kan ook deur diskontinue sensor metings en magneetveld effekte van die aarde veroorsaak word.

Die stabiliteit en gehalte van 'n klassieke LQR beheer strategie onder hierdie omstandighede word ondersoek deur simulasie, en twee aanpasbare beheer strategieë word ontwikkel om die oordrag te verbeter.

Die aanpasbare beheer strategieë meng elemente van "bang-bang" beheer om die gehalte te verbeter, en gebande beheer tegnieke om die robuustheid van die stelsel te verbeter. Hierdie beheer strategieë is deeglik in simulasies getoets en die resultate is vergelyk met dié van die klassieke beheerder.

Ekstensiewe modelleering van die stelsel is in die MATLAB™ omgewing gedoen om die beheerders te analiseer, en hardeware sensore is gebou om realistiese orientasie metings aan die beheerders te verskaf.

Acknowledgements

I would like to thank the following people who helped make this work possible:

- Eanette Maass and Jaco Vosloo, my collegues and friends, who shaped me as much during these long years as I have shaped this work;
- Mary-Anne Friend, my lover and partner, for putting up with the long hours and sour moods, and for acting as protector and guardian of the English language in this thesis;
- Mnr. J Treurnicht, my study leader and advisor, who always seemed to know when to push, when to pull, and when to let me fight it out on my own;
- To the most important women in my life. My mother, Elisma Rossouw, and my grandmother, Elizabeth Cilliers, for everything they have done to help make me the man I am today. And for the "lucky packets" of rusks and goodies that pulled me through the worst of times!
- To my family and friends, for all their support, love and understanding.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The successful commissioning of arguably the most complex student-developed satellite in space, has opened the door for South Africa and the University of Stellenbosch to future space co-operation" [6].

South Africa's first satellite, SUNSAT, was launched on the USA Air Force P91-1 Argos Delta II on February 23, 1999. As part of the University of Stellenbosch's continued drive to develop its space technology, this thesis addresses possible improvements in the area of satellite orientation control.

1.1 The Field of Research

Because of the inherent impracticality of servicing a satellite during its mission, including redundant systems is an integral part of the design criteria. The possibility that some of these systems may fail remain a challenge to control system design. Failures and other effects, discussed in section 2.3.1, cause large gain variations that affect the response of the satellite to control commands.

This thesis investigates the possibility of using adaptive control techniques to improve the quality and robustness of the design under these conditions.

1.2 Current Status

SUNSAT implemented a system consisting of separate controllers [10] to control different phases of the mission. This approach is related to an adaptive control method called "Controller Scheduling" [9].

The scheme relies on the design of a number of different classic control systems, one design for each of the configurations that may result from failures and measurement availability. Choosing which control system to use at any given moment is based on auxiliary observer measurements giving information about the state of the system.

The reason this scheme is so popular in practice is because well understood tools exist with which to design and test each of the controllers. Also, controller updates can be made relatively quickly, as quickly as the auxiliary measurements reveal failures and changes. Importantly, one would expect changes between the controllers to occur only infrequently, depending on the design.

Variations on this scheme exist, and may include the linear interpolation between discrete control systems to "soften" the switching between them.

The main drawback to this approach is that the extent of the required design can be enormous. This was not so much a problem on SUNSAT, where a limited amount of actuator redundancy was available, and the controllers were not designed specifically with failure in mind. Also, extensive simulation is required to verify that switching between the controllers happens smoothly and does not cause instability.

1.3 Thesis Goals

This thesis will develop a MATLAB™ model on which to test alternative control schemes.

Two adaptive schemes will be developed, tested and compared to a classical controller representing the current solution to control design in terms of stability and performance.

Due to the various sources of non-linearities that form a large part of the control environment, the approach is to design new schemes and thoroughly test them using simulation, rather than extensive mathematical analysis.

This strategy will give future satellite engineers a good idea of how well more complex adaptive schemes will function as a replacement to current designs, as well as highlight some complications that will require special attention.

1.4 Chapter Summary

Chapter 1 Present the reasoning behind this thesis and explain why the work is important.

Chapter 2 Introduce the Satellite environment and the factors that play a role in the control of the satellite.

Chapter 3 Show the development of the simulation environments used to test the proposed control strategies.

Chapter 4 Develop two new adaptive control strategies.

Chapter 5 Present and discuss the simulation results of the classical controller and the two adaptive schemes.

Chapter 6 Draw conclusions based on the results, and suggest areas for continued research.

Chapter 2

Modelling the Satellite Environment and Motion

2.1 Introduction

The following sections will describe the aspects of the satellite and its environment that are taken into account during the control system design and subsequent simulation.

Section 2.2 will present the satellite orbit, it's attitude description and the sensors and actuators included in the simulation.

Section 2.3 describes the major factors that influenced the design of the control system.

2.2 The Satellite Model

2.2.1 Keplerian Orbit Description

Kepler Orbit Parameters

The Keplerian parameters used to define the satellite orbit, shown in Figure 3.1 on pg. 18, are summarised in table 2.1.

Perigee altitude refers to the altitude of the satellite above the surface of the Earth at its nearest point. *Orbit eccentricity* is a measure of how elliptical the shape of the orbit is. It is defined as the ratio

$$e = \frac{c}{a} = \frac{\sqrt{(a^2 - b^2)}}{a} \qquad (2\text{-}1)$$

where a is the semimajor axis of the ellipse, b is the semiminor axis, and c is half the distance between the two foci of the ellipse, as shown in Figure 2.1.

The *right ascension of the ascending node* is the angle measured eastwards from the *vernal equinox* to the ascending node of the orbit. The ascending node of the orbit is the point where the satellite's orbit would cross the Earths equator going from South to North. The vernal equinox is the ascending node of the Earth's orbit about the Sun. For more information on the vernal equinox and other astronomical definitions, see Appendix A.

The *inclination* of the orbit is the angle between the orbital plane and the plane of the Earth's Equator and, lastly, the *argument of perigee* is the an-

Orbit Parameter	**Symbol**	**Value**	**Units**
Orbit Eccentricity	e	0.3	$0 \leq$ e <1
Perigee Altitude		800	km
Right Ascension of the Ascending Node	Ω	110	deg
Orbit Inclination	i	90	deg
Argument of Perigee	ω	0	deg

Table 2.1: Kepler Parameters of Satellite Orbit

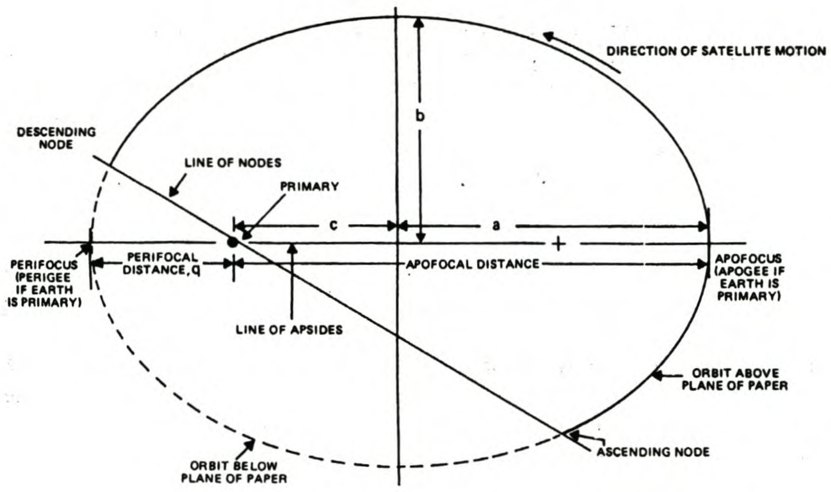

DIRECTION OF SATELLITE MOTION
DESCENDING NODE
LINE OF NODES
PRIMARY
b
c
a
PERIFOCUS (PERIGEE IF EARTH IS PRIMARY)
PERIFOCAL DISTANCE, Q
APOFOCAL DISTANCE
APOFOCUS (APOGEE IF EARTH IS PRIMARY)
LINE OF APSIDES
ORBIT ABOVE PLANE OF PAPER
ASCENDING NODE
ORBIT BELOW PLANE OF PAPER

Figure 2.1: Elliptical Orbit Parameters [11]

gle at the barycentre, measured in the orbital plane in the direction of the satellite's motion from the ascending node to perigee. The barycentre, in our case, is at the centre of the Earth.

These parameters are shown graphically on Figure 2.2. The plane of paper refers to the plane of the Earth's equator. These concepts are defined in more detail in Wertz [11].

Calculating the Satellite's Orbital Position

In order to determine the position of the satellite in the orbit defined by these parameters, we must determine the *true anomaly*, v. The true anomaly is the angle measured at the barycentre between the perigee point and the satellite.

To help calculate the true anomaly, we define the *mean anomaly*, M. The mean anomaly is $360 \cdot (\Delta t/P)$ deg, where P is the orbital period and Δt is the time since perigee passage of the satellite.

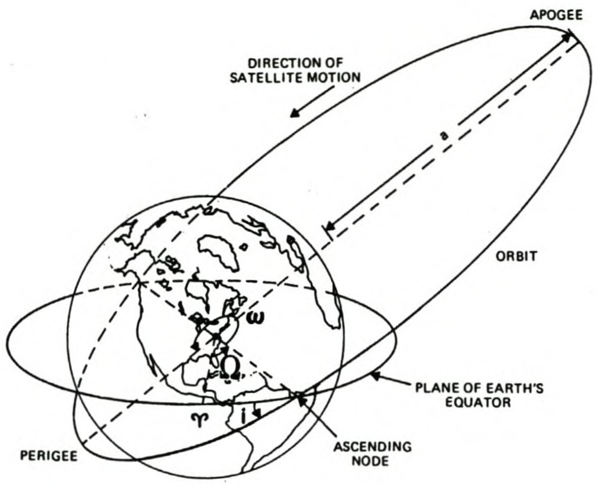


Figure 2.2: An Orbit defined by Kepler Parameters [11]

For circular orbits the eccentricity is zero, and the calculation is trivial, with

$$v = M \qquad \text{and} \tag{2-2}$$

$$r = q \tag{2-3}$$

where r is the distance of the satellite from the barycentre at the time in question, and q is the distance from the barycentre of the satellite at perigee.

For the non-trivial case with $e > 0$,

$$v \simeq M + 2e\sin(M) + \frac{5}{4}e^2\sin(2M) \qquad \text{and} \tag{2-4}$$

$$r \simeq \frac{q(1-e^2)}{1+e\cos(v)} \tag{2-5}$$

These equations are second order approximations to reality. For the complete solution, refer to Wertz [11]

2.2.2 Equations of Motion

Introduction

The central differential equations used in this work to describe the motion of the satellite and it's actuators are

$$\dot{\mathbf{h}}(t) = \mathbf{N_{rw}} \tag{2-6}$$

$$\dot{\omega}(t) = \mathbf{I}^{-1}\left(\mathbf{N_{dist}} - \mathbf{N_{rw}} - \mathbf{N_{mt}} - \omega \times \mathbf{I}\omega - \omega \times \mathbf{h}\right) \tag{2-7}$$

where $\mathbf{h}$ is the angular momentum of the reaction wheels and ω is the satellite rates about it's body axes.

$\mathbf{N_{dist}}$, $\mathbf{N_{rw}}$ and $\mathbf{N_{mt}}$ represent the total external disturbance torques' action upon the satellite and the two control torques from the reaction wheels and the magnetorquers, respectively.

Euler Numerical Integration Methods

The satellite rates and the angular momentum of the reaction wheels are updated in the simulation at every time interval using the Euler integration method,

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \Delta t\dot{\mathbf{h}}(t) \tag{2-8}$$

$$\omega(n+1) = \omega(n) + \Delta t\dot{\omega}(t) \tag{2-9}$$

2.2.3 Satellite Orientation Description

Introduction

The satellite orientation is described by three rotation angles, called the roll, pitch and yaw angles. Roll refers to the attitude rotation about it's body X-axis, pitch is the rotation about it's body Y-axis and Yaw is the rotation about the body Z-axis.

By altering the rotation order of these angles, any given attitude can be described in 24 different ways using these three parameters. Due to this ambiguity and the discontinuities in the inverse of certain trigonometric functions at or near 90°, the orientation is stored within the simulation as Euler symmetric parameters, otherwise known as quaternions. For a complete description on Euler parameters, direction cosine matrixes and quaternions, see Wertz [11].

Numerical Integration of Quaternions

The update equation for the attitude in terms of quaternions used in the simulations, is

$$\mathbf{q}(t_{n+1}) = \left[\cos\left(\frac{\bar{\omega}T}{2}\right)\mathbf{1} + \frac{1}{\bar{\omega}}\sin\left(\frac{\bar{\omega}T}{2}\right)\Omega_n\right]\mathbf{q}(t_n) \tag{2-10}$$

where

$$\tilde{\omega} = \frac{\omega(t_n) + \omega(t_{n+1})}{2} \tag{2-11}$$

$$\bar{\omega} = \sqrt{\tilde{\omega}_x^2 + \tilde{\omega}_y^2 + \tilde{\omega}_z^2} \tag{2-12}$$

and

$$\Omega_n = \begin{bmatrix} 0 & \tilde{\omega}_3 & -\tilde{\omega}_2 & \tilde{\omega}_1 \\ -\tilde{\omega}_3 & 0 & \tilde{\omega}_1 & \tilde{\omega}_2 \\ \tilde{\omega}_2 & -\tilde{\omega}_1 & 0 & \tilde{\omega}_3 \\ -\tilde{\omega}_1 & -\tilde{\omega}_2 & -\tilde{\omega}_3 & 0 \end{bmatrix} \tag{2-13}$$

2.2.4 Sensors

The Sun sensor

The Sun sensor measures the position of the Sun relative to the satellite. It provides a very accurate measurement, but it can only see a window of space above itself, and so becomes useless when the Sun is not in view or is obstructed by the body of the Earth. It is possible to make use of multiple Sun sensors so that the sunangle may be measured regardless of the orientation of the satellite, but in the case of SUNSAT, the accurate information was only needed while taking pictures of the earth, and other, less accurate, sensors were used during the rest of the mission.

The Horizon Sensor

The horizon sensor measures the direction of the earth relative to the satellite by finding the edges of the earth on the dark background of space. It shares many of the difficulties that plague the Sun sensor, and introduces a few new ones, like the difficulty of determining one of the edges of the earth when that edge is cloaked in night. Like the Sun sensor, it is only used during a small part of the mission, and normally in conjunction with the Sun sensor.

The Magnetometer

The magnetometer measures the magnetic field of the earth at a point in space, exactly like a compass, but in three dimensions, and compares it to a model of the earth's magnetic field at that altitude in its databanks. This measurement is not as accurate as either the sun- and horizon sensor, but the data is available irrespective of the satellite's orientation or it's position in it's orbit. One of the problems with this sensor is that the data is corrupted while the magnetorquers, introduced in section 2.2.5, are in operation. The data may also be influenced by magnetic field effects created by currents flowing in the satellite, and by how accurately the satellite knows it's position in orbit.

2.2.5 Actuators

Reaction Wheels

The reaction wheels are the stronger of the two types of actuators, relying on the basic principle of momentum exchange. They provide 30 mN·m of torque each, but are limited to a maximum angular momentum storage capacity of 0.3 kg·m^2/s. Once this capacity is reached, the reaction wheels are turning at the maximum speed of their motors, and they lose all further ability to exert torque in that direction.

To prevent this from happening, reaction wheels are ideally operated at or near zero bias, and the stored momentum is dumped from time to time using torques external to the satellite, such as those generated by the magnetorquers. These torques are external to the satellite because they are, in fact, caused by the Earths magnetic field.

Two reaction wheels per axis are modelled for simulation analysis later on in the thesis. SUNSAT flew a set of four reaction wheels, one per axis and the fourth one at an angle to the other three. This is a very practical solution when space constraints are severe, but the purposeful introduction of cross coupling without a communal controller for all three axes is an unnecessary complication to this investigation.

All values for maximum torque and the momentrum storage capacity of the actuators quoted here and below are loosely based on the actual specifications of the actuators flown aboard SUNSAT in 1999.

Magnetorquers

Magnetorquers, or magnetic coils, operate by creating a magnetic field that interacts with the magnetic field of the Earth, creating a moment and causing the satellite to turn. The torques created in this way are limited to around 3 mN·m peak, much smaller than those created by the reaction wheels. Unlike the reaction wheels, however, they have no limit to their capacity to change the angular momentum of the satellite.

2.3 Attitude Control Considerations

2.3.1 Origins of Gain Changes

System Failures

The possibility that some of the sensors and/or actuators aboard the satellite may lose functionality or cease to operate at all is always a consideration, and the effects of such a failure on the control system dynamics should be well understood.

Failing actuators pose the greatest threat to control system stability, having the greatest effect on system gain. In a system with two reaction wheels per channel, a single failure causes the gain of the channel and the maximum available torque to halve, severely retarding the ability of the control system to complete it's mission.

In the event of a sensor failure, it no longer provides any information, or if a sensor providing corrupted data is identified and turned off, the situation becomes similar to that of intermittent sensor data availability, discussed in the next section.

Sensor Bandwidth

Another condition exists that effects large gain transients in the control system. The bandwidth of the control system is determined by many factors, one of which is the accuracy with which the satellite states, such as orientation, may be measured or determined. Some of the more accurate sensors only function during parts of the mission. The Sun sensor, for example, can only provide accurate orientation data when it can actually see the sun. During certain maneuvers, or when the satellite is in the shadow of the earth, this is not possible.

Making use of the extra information when it is available effectively allows for more accurate orientation control based on more accurate orientation information. This translates into larger control gains, and this switch between more accurate and less accurate control is another source of system gain changes.

Magnetic Field Effects

The magnetorquers rely on interaction with the magnetic field of the Earth in order to generate a torque. Although the magnetic field of the earth exists everywhere the satellite is likely to go, the strength and direction of this field depends on the satellite's position above the Earth.

This means that the strength of the magnetic interaction will vary with time and affect the gain of the whole control system. This is the smallest gain variation of the three listed here, because it affects only the weaker of the two actuators, the magnetorquers. It is also a much more gradual change than the other two.

2.3.2 Sources of Errors and Disturbances

Overview

To study the performance of the control systems in a realistic environment, the main sources of errors and disturbances have to be taken into account.

Spacecraft are subject to a number of random disturbance torques during its mission lifetime. Some of these torques have environmental origins, while others originate within the satellite itself. Solar radiation, atmospheric drag and micro meteorite collisions are all examples of environmental disturbances. The magnetic interaction between the earth's magnetic field and the magnetic polarization of the satellite by the electric current flowing through its electronics, or the cross-coupling of the control torques caused by the misalignment of the actuators have their origin within the satellite.

Misaligned Actuators

An unavoidable consequence of the manufacturing process is that the actuators will be, usually only slightly, misaligned with the designed body axis of the satellite.

This misalignment causes *cross-talk* between the axis, meaning that an applied torque in one axis will also cause the other two axes to move. This effect is considered noise, and because its origin is the controller itself, it is also *coupled* noise.

Coupled noise refers to errors introduced to measurements taken and control torque applied to the system that in some way correlates to actions taken by the system itself. Coupled noise is much more troublesome in control systems because it rarely has a mean of zero, and it counteracts the corrective power of control, particularly when the control system itself is the source of the noise.

Delayed Actuator Response

A complex control problem to solve is the effect of delayed actuator response. Normally, the time difference between when the controller issues a command and when the actuator reacts to it is much shorter than $\mathrm{T_s}$, the sampling period of the controller's computer processor.

When this delay becomes near to or larger than $\mathrm{T_s}$, the controller may interpret this lack of response as an actuator or sensor failure. To prevent this, the controller should be programmed to expect a certain amount of delay, and this delay should be estimated along with other system parameters.

Disturbance Torques

Many other disturbance torques exist [11], both environmental and satellite generated. Models exist for some of these, and it would be prudent to include more complete models once a proper control system has been designed and the satellite specifications are available.

In the absence of any real specifications for the satellite in question, and the fact that most of the disturbances we are interested in in this study are included above, the remainder of the external and internal disturbance torques are represented by a Gaussian model with zero mean.

Chapter 3

The Development of a Simulation Environment

3.1 Introduction

The development of an environment in which to test the performance of the proposed control systems was done in three phases.

The first and second phases consist of the creation of a simulation model in the MATLAB™ environment. The first phase includes a satellite orbit propagator amongst the celestial bodies of the Earth and the Sun, while the second phase models the attitude of the satellite in a stationary position above the Earth. It focuses on the sources and roles of disturbance and control torques, and their effect on the attitude of the satellite.

The third phase comprises the design of a Sun sensor as part of a hardware-in-the-loop test bed for the orientation controllers.

The control strategies to be tested in these simulations will be developed in Chapter 4.

3.2 Phase One: Simulating the Space Environment

3.2.1 Overview

In order to design as comprehensive a test environment as possible, and to serve as the foundation of the study to understand the major aspects of the problem at hand, phase one includes models of all the major elements that affect the actuators and sensors introduced in Chapter 2.

The orientation of each individual sensor on board the satellite is specified and, together with the orientation of the satellite, is used to determine the actual input data and output measurements from these sensors during the simulation.

The MATLAB™ code for this simulation is included in Appendix C.

3.2.2 The Satellite Orbit

Figure 3.1 shows the trace of one complete orbital period of the satellite, including the relative positions of the Earth and the Sun. See Table 2.1 on pg. 5 for a summary of the orbital parameters. The sphere of the Sun is only meant to indicate direction, and does not represent it's size as seen from the satellite. The actual angular diameter of the Sun at it's mean distance from the satellite is 0.53 deg, compared with the angular diameter of the Earth from an altitude of 800km, 125.38 deg.

During the orbit, the satellite orientation is controlled to match the orbital coordinate system, i.e. with it's body Z-axis pointed towards the centre of the Earth and it's body X-axis in the direction of motion.

3.2.3 The Sun Sensor

In Figure 3.1, the body axes of the satellite is plotted at equidistant intervals along it's orbit to show the changing orientation of the satellite, but also to show the functionality of the Sun sensor. At points where the view of the Sun is totally obscured from the satellite, the display of the body axes are shortened. At points of partial obscurity, the axes are lengthened a little, and during points where the Sun comes into view of the Sun sensor, it is shown with the longest of the four line types.

Figure 3.2 shows the output measurements from the Sun sensor during this orbit. The sensor is mounted to view space in the negative nadir direction, with a view angle of 40 degrees. For most of the orbit, no measurements are available from the sensor, until the Sun comes into view 106 minutes into the simulation. Accurate orientation information is then available for approximately the next 32 minutes.

By adding more Sun sensors or by increasing their view angles, the duration of the availability of this information may be extended.

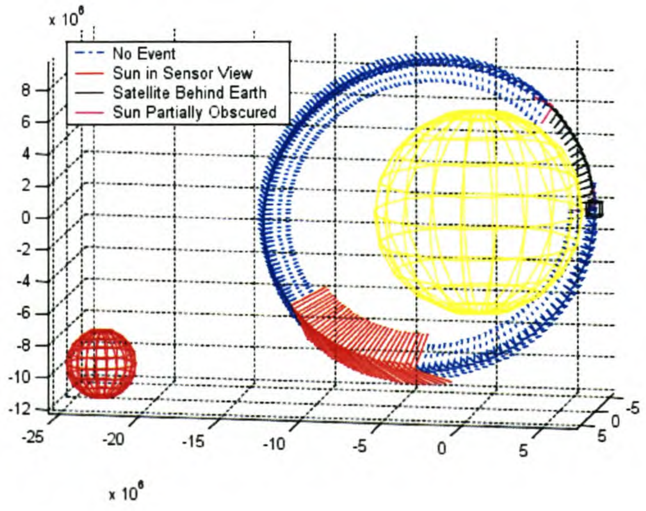


Figure 3.1: Simulation of Satellite in Orbit

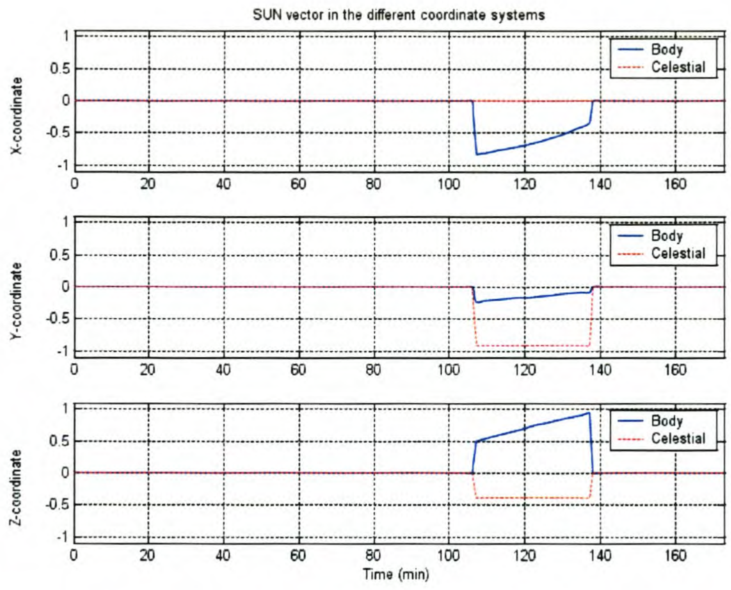


Figure 3.2: Sun Sensor Measurements

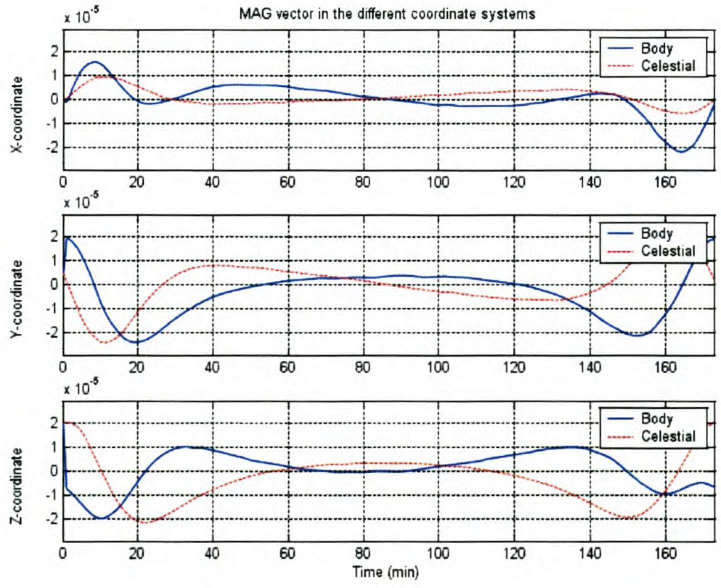


Figure 3.3: IGRF Magnetic Field Vector

3.2.4 The Magnetometer

A 10th order model of the International Geomagnetic Reference Field, IGRF, is implemented to provide data for the magnetometer, and to determine the changes in magnetic coupling between the magnetorquer actuators and the magnetic field of the Earth, causing a change in the effective gain of the subsystem.

Figure 3.3 shows the profile of the modelled geomagnetic field during the simulation in Body and Celestial Coordinates.

3.2.5 Shortcomings

Unfortunately, as controllers were designed for this system, it quickly became obvious that phase one is not very well suited to testing new control strategies.

The measurement data acquired during the simulation is insufficient to provide orientation information when the Sun is not in view of the Sun sensor. More sensors need to be modelled for such information to be useful and will very likely require the implementation of a state-estimator or Kalman filter, which is beyond the scope of this thesis.

The calculations required to determine the magnetic field at every interval and to calculate the effective torque generated by the magnetorquers from this was overly cumbersome and unnecessary for the investigation into the stability and performance of new designs.

This environment is better suited as a second stage testing ground after initial designs have been completed.

The simulation was extensively modified and later became "Phase Two: The Simulation of Attitude Dynamics".

3.3 Phase Two: The Simulation of Attitude Dynamics

3.3.1 Overview

Phase two abandons several irrelevant complications that plagued the first phase and extends other features to provide a complete environment under which to test the orientation controllers.

The celestial bodies of the Earth and Sun become abstract orientation targets, and the relative propagation of the satellite in its orbit is inconsequential. Maintaining the proper attitude is achieved by regularly updating the orientation commands.

The individual models of the magnetometer, Sun sensor and horizon sensor are all reduced to their logistical effects on the control system, while the models of the actuators are expanded to include orientation information and extended functional profiles.

The MATLAB™ code for this simulation is included in Appendix D.

3.3.2 Modelling Sensor Effects

Attitude determination is circumvented by assuming an estimator exists in the system with sufficient working sensors to provide orientation information as needed by the controller.

Only the effects of the sensor switching and failure is still taken into account, and will form part of the test cases considered in Chapter 5.

3.3.3 Modelling the Actuators

The actuators are modelled to include the body alignment, gain, angular momentum, maximum angular momentum and maximum torque, amongst others.

Since the magnetic field vector is no longer available, it's coupling with the magnetorquers at every point in orbit can no longer be determined. The magnetorquers has a relatively small effect on attitude dynamics, and the strength of this coupling was set to the largest value it is likely to reach during a normal orbit. When of interest, a gain profile for the magnetorquers during a typical orbit from phase one was recorded and included in the simulation.

3.3.4 Results

The results obtained from this simulation form the core of the work presented here, and will be discussed in detail in Chapter 5.

3.4 Sun/Horizon Sensor Hardware Design

3.4.1 Introduction

In parallel with the mathematical simulations, a test bed was proposed with which to correlate the results of the simulations with the performance of the controller in a physical environment.

A Sun and horizon sensor set is developed to provide orientation information for the controller, and eventually to be included in a hardware-in-the-loop simulation run.

A colleague [5] working on satellite attitude determination was invited to write the software to do post-processing of the sensor data and to extract

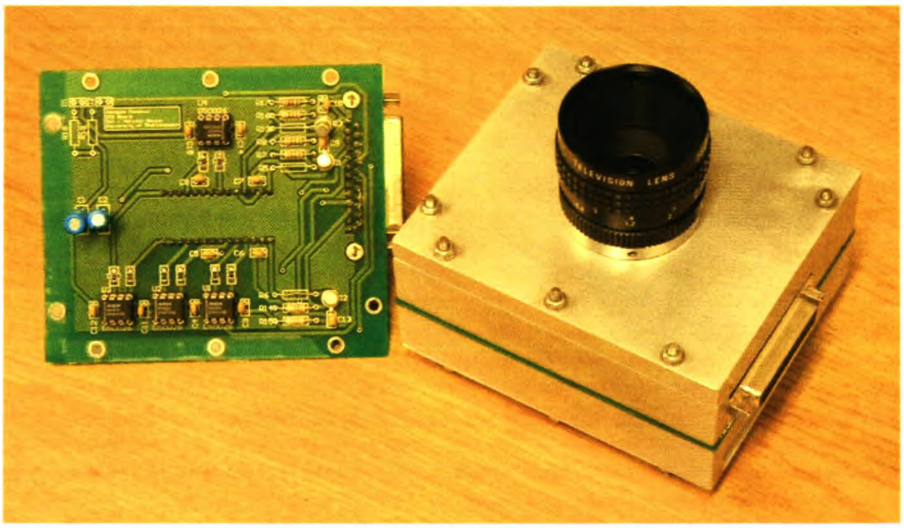


Figure 3.4: Photo of the CCD Driver Board and Housing

the attitude parameters.

The schematics and printed circuit board layouts are included in Appendix B for reference.

3.4.2 CCD Camera

A 2098 element Kodak CCD (KLI-2113) image sensor was selected to do the image capturing. It is housed in an aluminium box to serve as a mounting for the lens and to block ambient light. A PC board was designed to drive it's electronics, and control signals are received from the data capture board via a 40-pin connector at its side.

Figure 3.4 shows the completed system.

3.4.3 FPGA Controller and Data Capture Board

An add-on card, shown in Figure 3.5 was built to serve as an interface to the various components. It connects a DSP data processing board (ADSP2189M

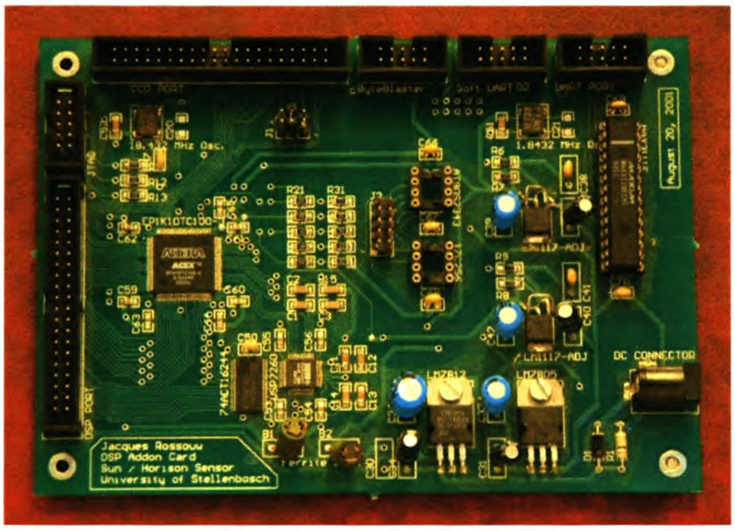


Figure 3.5: Photo of the Add-On Card

EZ-KIT LITE) with the CCD camera and the PC running the controller.

An FPGA on the board generates the timing waveforms to drive the CCD imager, and coordinates the capture and digitization of the image with read requests from the DSP board. The DSP board then extracts orientation information from the image and delivers it's results to the PC via a serial link.

3.4.4 Status and Results

The hardware was built and tested, and raw data was received from the CCD camera.

Unfortunately, extensive problems with the integration of the post-processing software with the sensor hardware prevented the design from being completed to the point where controllers may be tested on it.

A commercial company affiliated with the University of Stellenbosch showed an interest in the project, and the design was delivered to them for further development.

Chapter 4

Control: Concept and Design

4.1 Introduction

This chapter will describe the development of the adaptive control systems investigated in this thesis.

In the first section a classical controller is developed to serve as a measure against which to test the performance of the adaptive schemes in Chapter 5.

The following sections describe the initial concept and logical development that led to the final forms of the adaptive schemes.

4.2 Classical Control

4.2.1 Overview

A great many tools exist with which to design modern controllers in the satellite environment. Many of the control methods applied to satellite orientation control make use of passive control, like the gravity gradient boom [6] flown aboard SUNSAT, damping the motion of the satellite without regard for it's orientation. This method is virtually certain to assure the stability of the satellite, but is not well suited to serve as a comparison for the proposed adaptive controllers.

A modern design tool, called a linear quadratic regulator (LQR), which specifically addresses the issue of achieving a balance between good system response and the control effort required, is used to design an active controller with the same function and goals as our adaptive schemes.

4.2.2 About LQR Design

A linear quadratic regulator aims to find control gains **K** that minimize a cost function

$$\Gamma = \int_0^\infty [\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}]dt \tag{4-1}$$

where $\mathbf{Q}$ is an $n \times n$ state weighting matrix, $\mathbf{R}$ is an $m \times m$ control weighting matrix, and m is the number of control inputs in a multi-input system. Our system is SISO with two states, the error angle and the error rate, and so $m = 1$ and $n = 2$.

4.2.3 State Space Model and Design Parameters

A controller was designed for each of the three axis separately shown in Figure 4.1. The state-space description of the system,

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ \frac{k_\theta}{I} & \frac{k_\omega}{I} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{k_\theta}{I} \end{bmatrix} r_\theta \tag{4-2}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} r_\theta \tag{4-3}$$

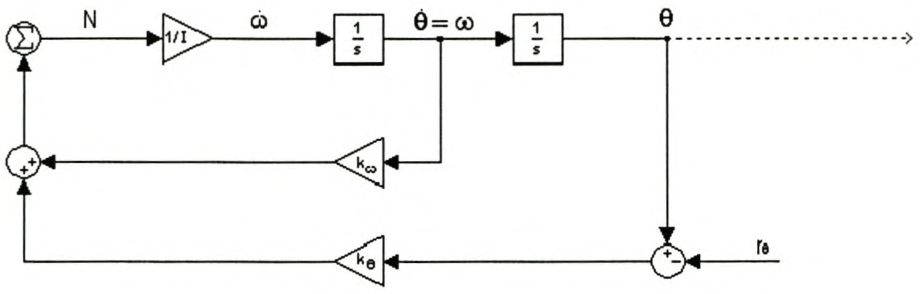


Figure 4.1: Block Diagram of Single Axis Satellite Model

and with weights $Q = [1 \quad 0; \quad 0 \quad 0]$ and $R = 1$, the LQR design yields feedback gains of

$$k_\omega = 0.0722 \tag{4-4}$$

$$k_\theta = 0.8497 \tag{4-5}$$

4.3 Adaptive Schemes

4.3.1 Introduction

Various adaptive schemes were considered, primarily adaptive variations of classic PID control theory [2, 3].

These controllers, as are many of the more advanced adaptive control schemes, are heavily dependent on the on line estimation of system states and system parameters.

It was therefore decided to custom build an adaptive strategy by using control principles that require knowledge of the information we *do* have, and that specifically focus on the performance criteria of interest here, namely the settling time and system overshoot.

4.3.2 Conventional Bang-Bang Control

The adaptive schemes presented in this thesis are all based on modifications to the so-called bang-bang control method. The method is based on the controllers accurate knowledge of the maximum available controller torque and the system parameters.

The control scheme is a time-optimal solution and uses all available control energy to drive the state errors to zero as fast as possible.

The solution of the state-space trace for such time-optimal control follows a curve starting at the centre and leaving at an exponential rate. It is given by

$$\theta_{optimal} = \frac{I}{2N_{max}} * \omega^2 \tag{4-6}$$

and is shown on all further state-space figures, as in Figure 5.1.

When the system trace reaches this line and all available torque is applied in the reverse direction, the system will reach zero error angle as it reaches zero speed.

It can also be described as a border for the point of no return, for when the satellite crosses this border insufficient energy remains in the system to prevent an angle overshoot.

4.3.3 Introducing Banded Control

Applying this solution to satellite control is difficult when neither the system parameters nor the available torque is fully known.

To adapt the application of bang-bang control to a feasible solution for orientation control, we introduce the concept of banded control.

Banded control is a way to design for both the best and worst case scenarios that need to be considered, and to gradually switch between the solutions as information about the state of the system becomes available.

The idea relies on an interaction between applying a solution and then measuring the response of the satellite to determine whether the response is sufficient to reach our goals.

The area between the borders of optimal control will be referred to as the active region, as this is the region during which the adaptive controllers will take control and try to improve the system dynamics. These borders are shown in magenta on all state-space graphs in Chapter 5. For an example, see Figure 5.1 on pg. 38.

During the remainder of the control response, the standard LQR controller will determine the dynamics of the satellite.

4.3.4 The Adaptive Control Algorithm

Both adaptive schemes measure the degree of penetration of the system states into the active region to determine the amount of adaption required. The system states here refer to the satellite rates and orientation angles.

At every sampling interval, the boundaries to the active region are determined from the current estimation of the satellite rates, ω, by the following equations:

$$\theta_a = \frac{I\omega^2}{2(N_{RW} + N_{MT})} \tag{4-7}$$

$$\theta_b = \lambda\theta_a \tag{4-8}$$

θ_a defines the best case error angle, where all actuators originally included in the design will be required to bring the satellite to a halt at it's orientation destination. θ_b denominates the worst case error angle being designed for. At this angle the minimum remaining actuators, operated at full torque, will bring the satellite to rest at it's destination. λ is some factor greater than one that will define the width of the active region. During the simulations of

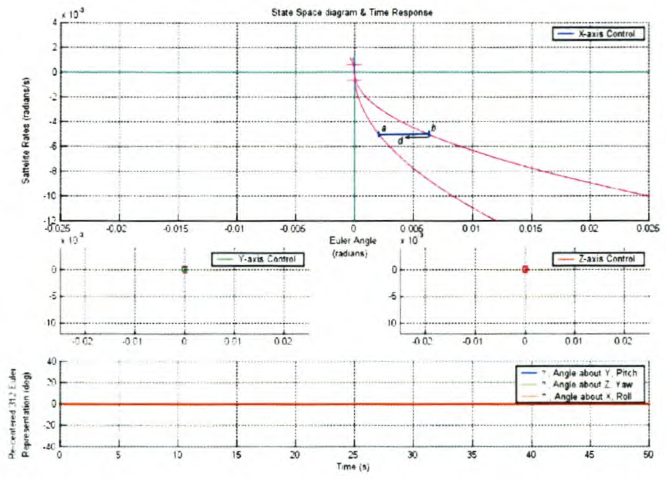


Figure 4.2: Torque Adaption in the Active Region

Chapter 5, this value was set to 3. I represents the satellite inertia, including the maximum inertia contributions from spinning reaction wheels.

N_{RW} and N_{MT} refers to the maximum reaction wheel torque and the maximum magnetorquer torque, respectively, that is available to the fully functional satellite.

If it is found that the current estimate of the error angle lies somewhere between θ_a and θ_b, the system is inside the active region and the adaptive controllers activate.

At the first such a sampling interval, the classic controller's torque command is recorded and any future adaption torque is added to that base. This means that the first adaptive control command will be very nearly the same as the previous classic controller command, ensuring a smooth transition between the two controllers.

Also, both adaptive controllers have an initial torque contribution of zero, meaning that at or near the worst case boundary, the trajectory of the system states will never cause them to cross back over θ_b into the classic region, because the trajectory there will be equal to what the classic controller commanded originally.

The active adaption scheme then calculates the difference between the recorded torque command and the system's maximum torque command to estimate the surplus torque available for adaption. It then uses the degree of penetration, d, shown in Figure 4.2, to calculate the adaptive torque contribution according to each scheme's respective adaption profiles, discussed below.

4.3.5 Torque Adaption Profiles

The difference between the two proposed schemes lies in the rate and manner in which they adapt inside the active region.

Linear Interpolation

The first adaptive scheme uses a linear scaling function to gradually increase the torque of the actuators until an equilibrium is reached. At this point the applied torque will cause the states of the system to remain at a certain distance between the two extremes and guide the satellite rates and error angle to zero simultaneously.

See Figure 5.2 on page 39 for an illustration of this adaption.

A possible problem with this scheme is that such rapid changes in actuator torque could shorten the hardware's life expectancy.

Linearizing Points of Intersection

The second scheme attempts to smooth the transition between the LQR control response and the adaptive scheme at the borders of the active region and also to tries to eliminate the current spikes associated with rapid changes in reaction wheel acceleration.

The function shown in Figure 4.3 was proposed to meet these requirements,

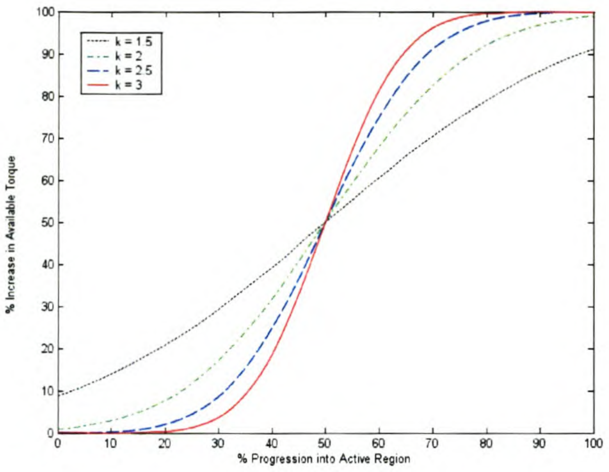


Figure 4.3: Profile of Adaption Function for Scheme Two

and is given by

$$N = 1/2\, erf(k\sqrt{-\ln(k^{-1})}\,(2\,x-1)) + 1/2 \tag{4-9}$$

where $k > 0$ and x is a value between 0 and 1 indicating the insurgence of the states into the active region. The constant $k = 3$ was used during simulations.

4.3.6 Dead-Zone

Both adaptive control schemes return control to the classic controller when the system states are very close to zero. This is done to avoid the situation that will arise when consecutive state samples are taken on either side of the orientation destination, resulting in the adaptive controllers first commanding full torque in the forward direction and, after the following sample shows that it has overshot it's target, commanding full torque in the backwards direction. This situation will continue indefinitely, and is known as a limit-cycle.

This dead-zone can be seen on Figure 4.2 as the area between the two short horizontal lines crossing the y-axis, on either side of its intersection with the x-axis.

Chapter 5

Simulation Results

5.1 Introduction

The following sections will compare the performance and stability of the standard linear quadratic regulator design with the two proposed adaptive schemes.

Section 5.2 presents the state-space diagrams and time-based error traces of the systems under the respective control schemes. Their distinguishing features will be identified and preliminary comparisons made.

Section 5.3 shows and compares the performance of the three systems under both ideal and noisy conditions. The conditions discussed in the introduction to this thesis are set up and tested and the increase in performance is measured.

Section 5.4 shows the results of an empirical Lyapunov stability study of the various control schemes.

The "State-Space and Time Trace" Figure

For the sake of clarity, all profile traces in this thesis were made on only two types of figures. Figure 5.1 on page 38 is an example of the first kind, the state-space and time trace.

The figure comprises of four graphs. The three at the top are the X-, Y- and Z-axis state-space traces showing the satellite error angles and error rates for the motion under consideration. As discussed in Chapter 4, one of the priorities of the control system is to keep the error overshoot to a minimum. This can be seen on the state-space graph by the amount the trace travels past zero in the negative x-direction.

Also displayed on all state-space graphs are two magenta traces leaving the origin at an exponential rate. They differ from the blue trace representing satellite motion in that they never return to the x-axis of the graph.

The area between these traces represent the region during which the adaptive schemes would activate. They are shown in Figure 5.1 purely for illustrative reasons, and no controller other than the LQR regulator activates on this particular figure.

The last graph, shown at the bottom of the figure, plots the orientation error angles of the satellite over time.

The "Control Torques" Figure

Figure 5.2 on page 39 is an example of the second type of figure, the control torques. The top graph is an enlarged view of torques applied around the X-axis, while the bottom graph shows a composite view of the Y- and Z-axis torques.

These graphs show the control torques applied during satellite maneuver. This figure contains multiple traces which will be briefly discussed below. Please note that Figure 5.2 is a special case of the figure and does not contain all the traces mentioned below. Figure 5.7 on page 46 is an example of the general case containing all the traces.

Adjusted torque is the combined effective control torque finally applied to the satellite by all the actuators. In most cases it follows the "suggestion" traces closely, but it may deviate significantly under fault conditions.

Classic Torque shows the suggested torque output of the LQR regulator. The controller will follow this suggestion as best it can while not in the adaptive regions.

Adaptive Torque shows the effective torque suggested by the adaptive strategies, when in the active regions. During all other times this trace will be zero to avoid unnecessary confusion.

Maximum RW Torque is a reference line showing the available torque of the reaction wheels when they are functioning properly.

Maximum MT Torque shows the reference line of the available magnetorquer power.

5.2 Control Profiles in an Ideal Environment

5.2.1 Linear Quadratic Regulator

Control Profile

Figure 5.1 shows the state-space diagram that represents the error angle and error rates resulting from the system under the control of an LQR regulator.

The system is initially at rest when, at time zero, a command is given to the controller to turn 30° around the X-axis.

The state-space trace starts on the X-axis marked by a '∘'. The controller identifies that it needs to generate an angular rate in order to reduce this error to zero, and activates the actuators.

As the rates increase, the trace moves down, generating a negative speed to negate the positive error angle. With a non-zero rate, the angle changes, causing the trace to move in the negative X-direction. This interplay continues until the controller starts slowing the satellite down in preparation for the stop at it's destination, causing the trace to start moving back up, returning to the X-axis.

The three controllers differ in the manner in which they approach zero, but the LQR regulator designed for this thesis has a small amount of overshoot, meaning that it moves past the Y-axis (the required orientation angle), and then returns to zero.

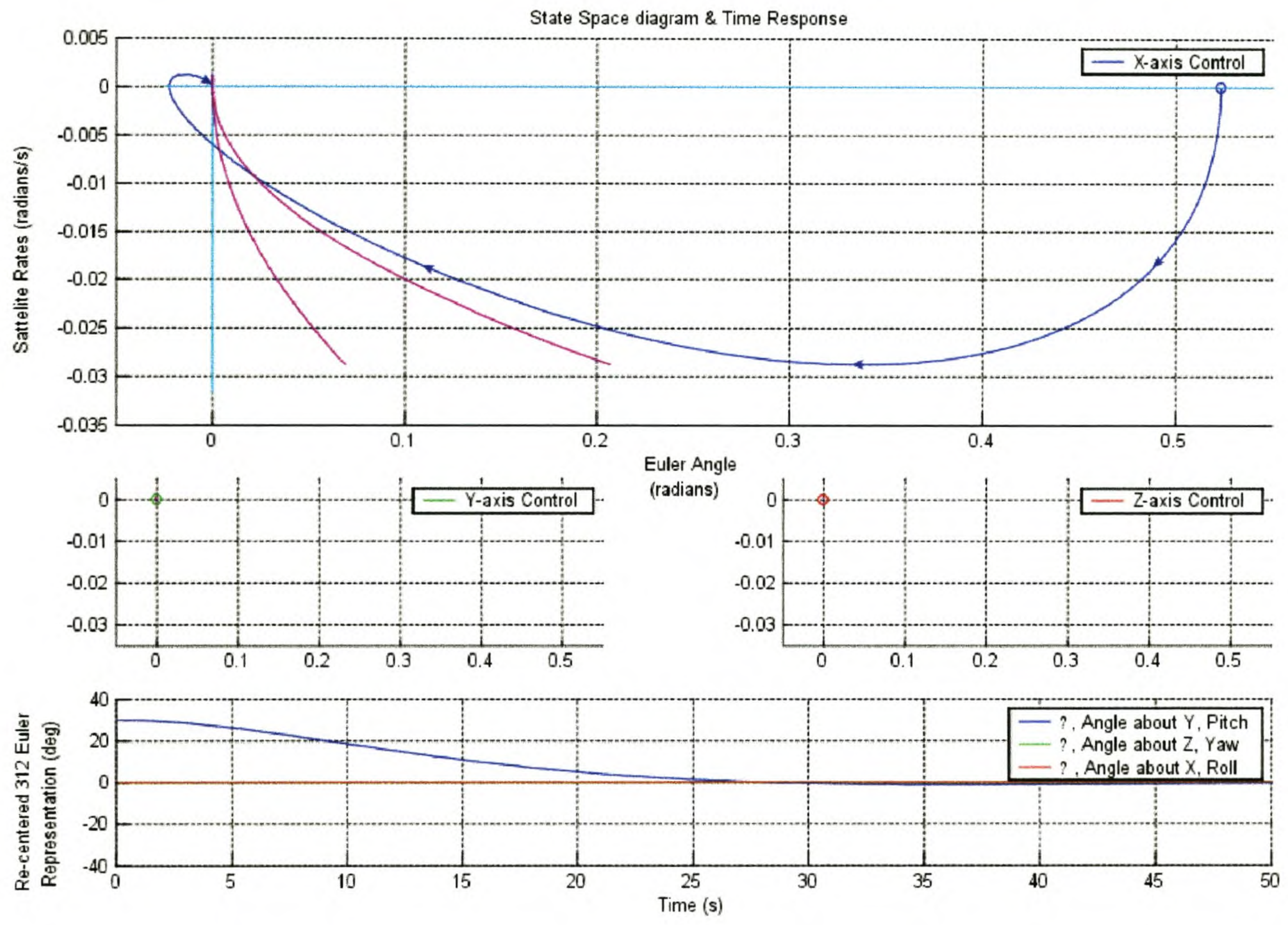


Figure 5.1: Linear Quadratic Regulator: Profile in Ideal Environment

5.2.2 Adaptive Scheme One

Control Profile

Figure 5.2 shows how the torque profile changes in the first adaptive scheme as the trace enters the active region.

This figure was generated in an open loop simulation run to avoid feedback distorting the graph. The maximum torque limitation was also ignored. The torque profile of all three schemes in the closed loop systems will be shown and discussed in Section 5.3.

Figure 5.3 shows the state-space and time trace graph of the first adaptive scheme. The trace is identical to Figure 5.1 until it enters the adaptive region where the adaptive scheme activates. The reduced overshoot is already visible.

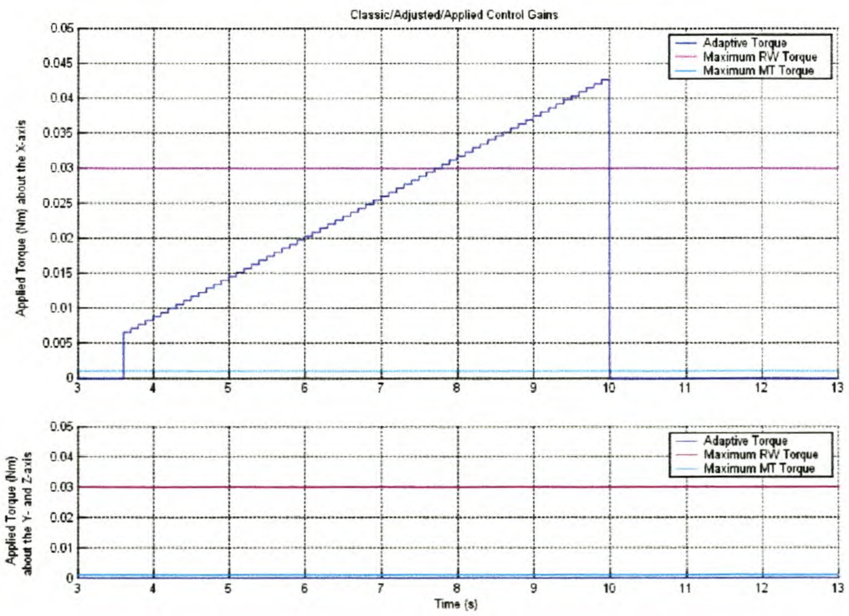


Figure 5.2: Adaptive Scheme One: Zoom of Gain Adaption through Active Region

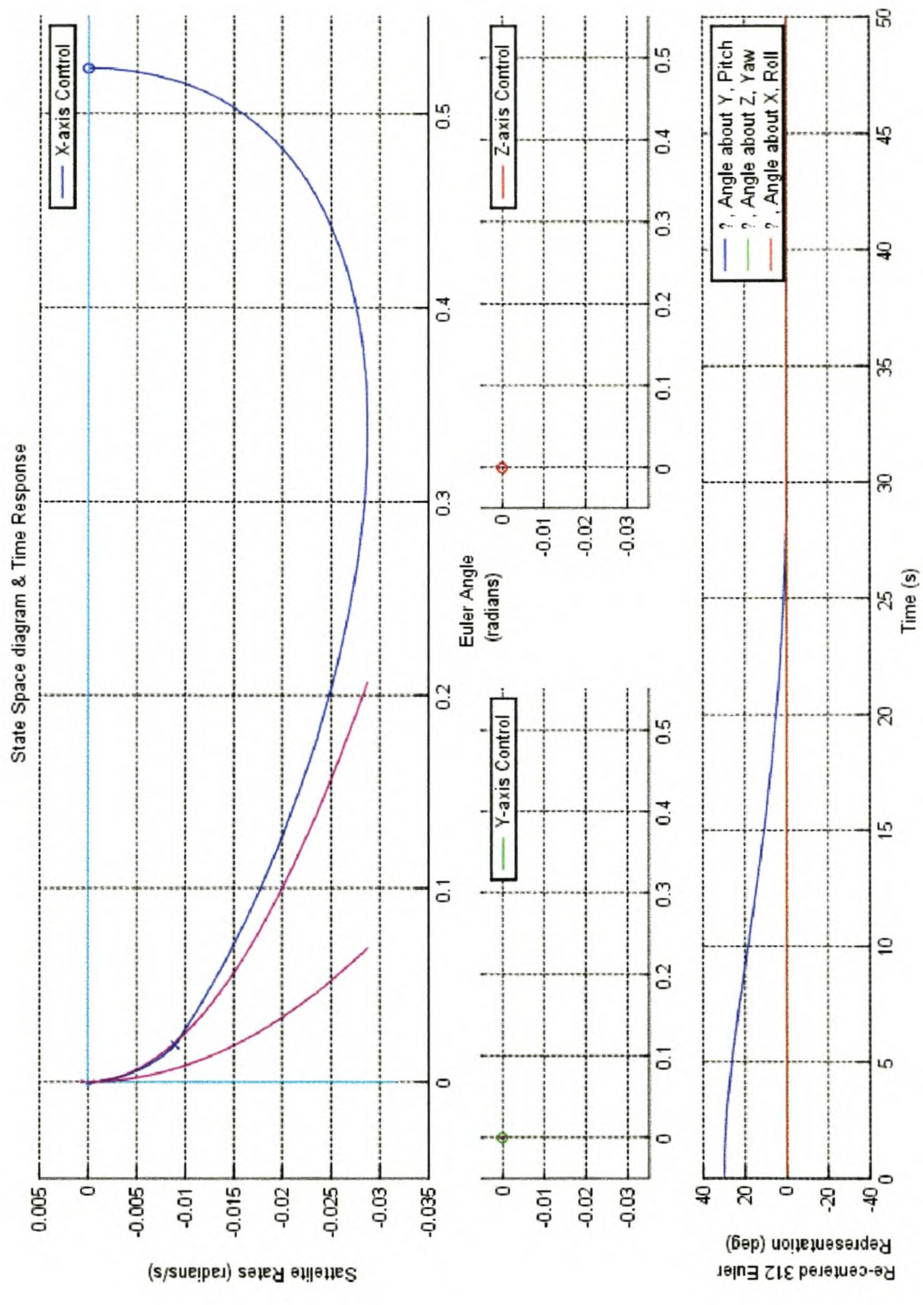


Figure 5.3: Adaptive Scheme One: Profile in Ideal Environment

5.2.3 Adaptive Scheme Two

Control Profile

Figure 5.4 shows how the torque profile adapts in the second scheme. Once again, this trace was generated using an open loop simulation.

Figure 5.5 shows the state-space and time trace graph of the second adaptive scheme. As expected, this trace is almost indistinguishable from the first adaptive scheme. The important differences will be highlighted in the next section.

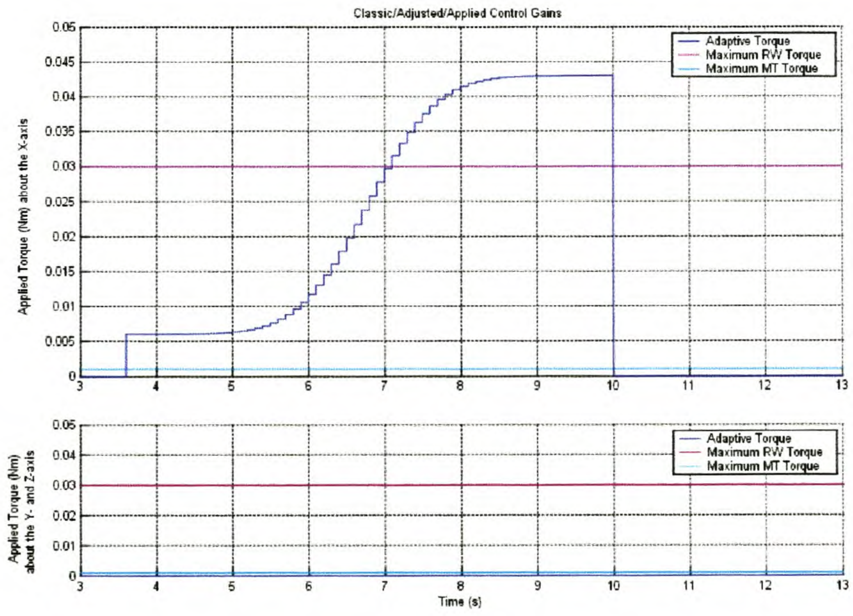


Figure 5.4: Adaptive Scheme Two: Zoom of Gain Adaption through Active Region

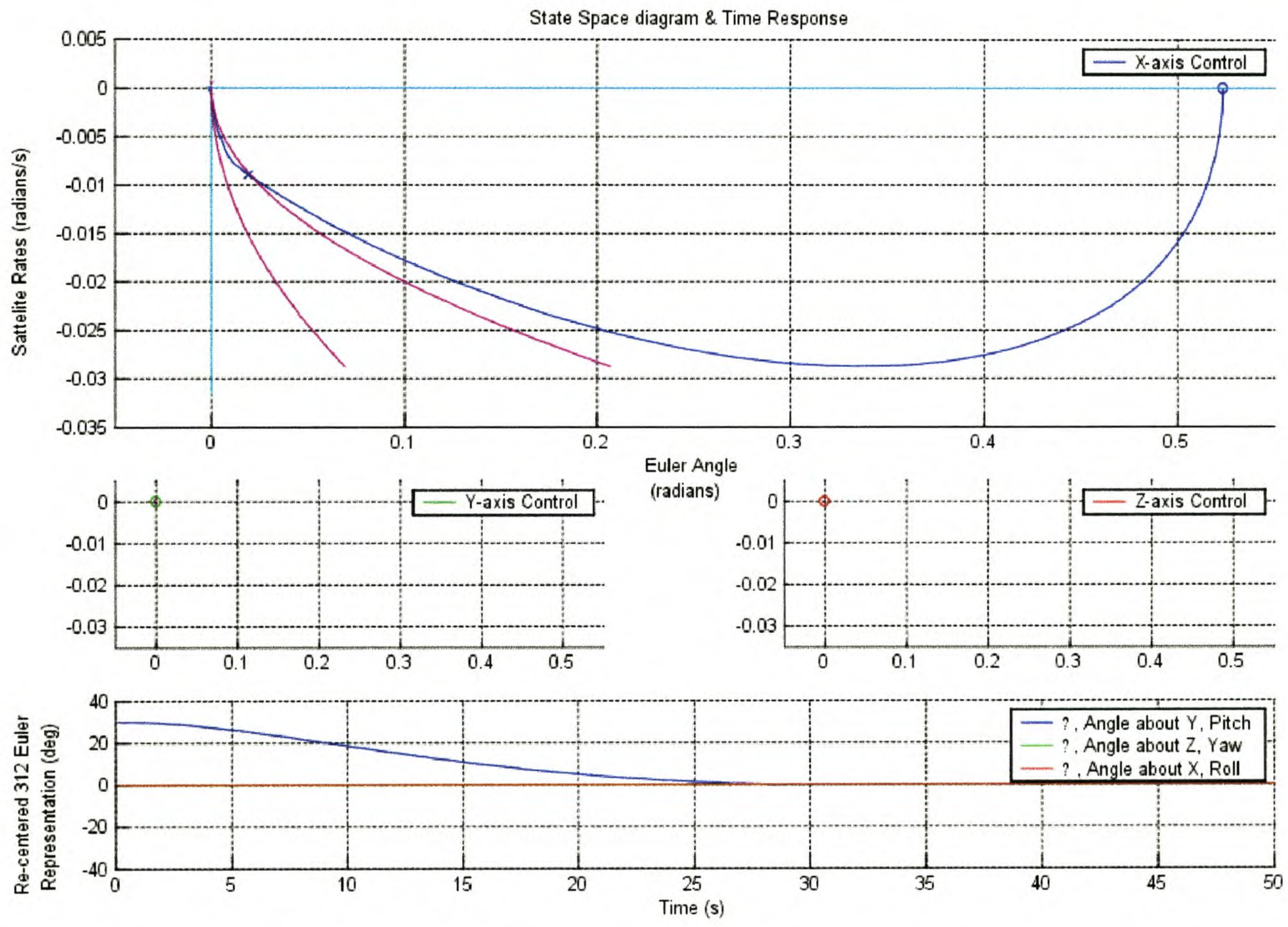


Figure 5.5: Adaptive Scheme Two: Profile in Ideal Environment

Table 5.1: Comparison of Simulation Results under Ideal Conditions

		2% Settling Point	**Overshoot**		
Figure	**Control Scheme**	**at**	**Maximum Overshoot**	**at**	**Percentage of Initial Value**
5.1	LQR	49.6s	1.2951°	37.1s	4.3170 %
5.3	AS1	26.3s	0.1062°	38.5s	0.3541 %
5.5	AS2	26.3s	0.1014°	38.2s	0.3379 %

5.3 Comparing Performance

5.3.1 Overshoot

The control profiles are exactly the same in all three cases until the trace enters the active region. Figure 5.6 shows a close-up of what happens in the three cases after the adaptive schemes activate.

The reduction in overshoot is significant. The dead-zone close to the centre can also be clearly seen in this figure as marked by two short horizontal lines across the Y-axis. It is only after the trace enters this region and the adaptive schemes return control to the LQR regulator that any overshoot is visible at all.

The difference between the two adaptive schemes in terms of their state-space traces can be seen on Figure 5.6. Adaptive scheme one deviates faster from the LQR trace and stays closer to the near boundary of the active region.

Adaptive scheme two shows a more gradual transition between the two controllers, and remains almost perfectly in the middle of the active region until it reaches the dead-zone at the centre.

Table 5.1 summarises the results of the initial tests. The two adaptive schemes perform much better than the standard design.

5.3.2 Effect of Adaptive Schemes on Control Torque

Figure 5.7 shows the actuator control torques during this comparison. The increase in control torque resulting from the two adaptive schemes can be seen in the interval from 25 to 30 seconds.

The increase in torque is much less than the maximum torque applied at the beginning of the maneuver, and so does not place unreasonable amounts of stress on the actuators. It is also clear that the adaptive schemes are active for relatively short periods of time.

Adaptive scheme one is active for a slightly longer period of time than scheme two, but it's increase in control torque is less.

Adaptive scheme two uses more control energy, but is active for a shorter period of time and the transition between the LQR curve and the adaptive curve is much smoother.

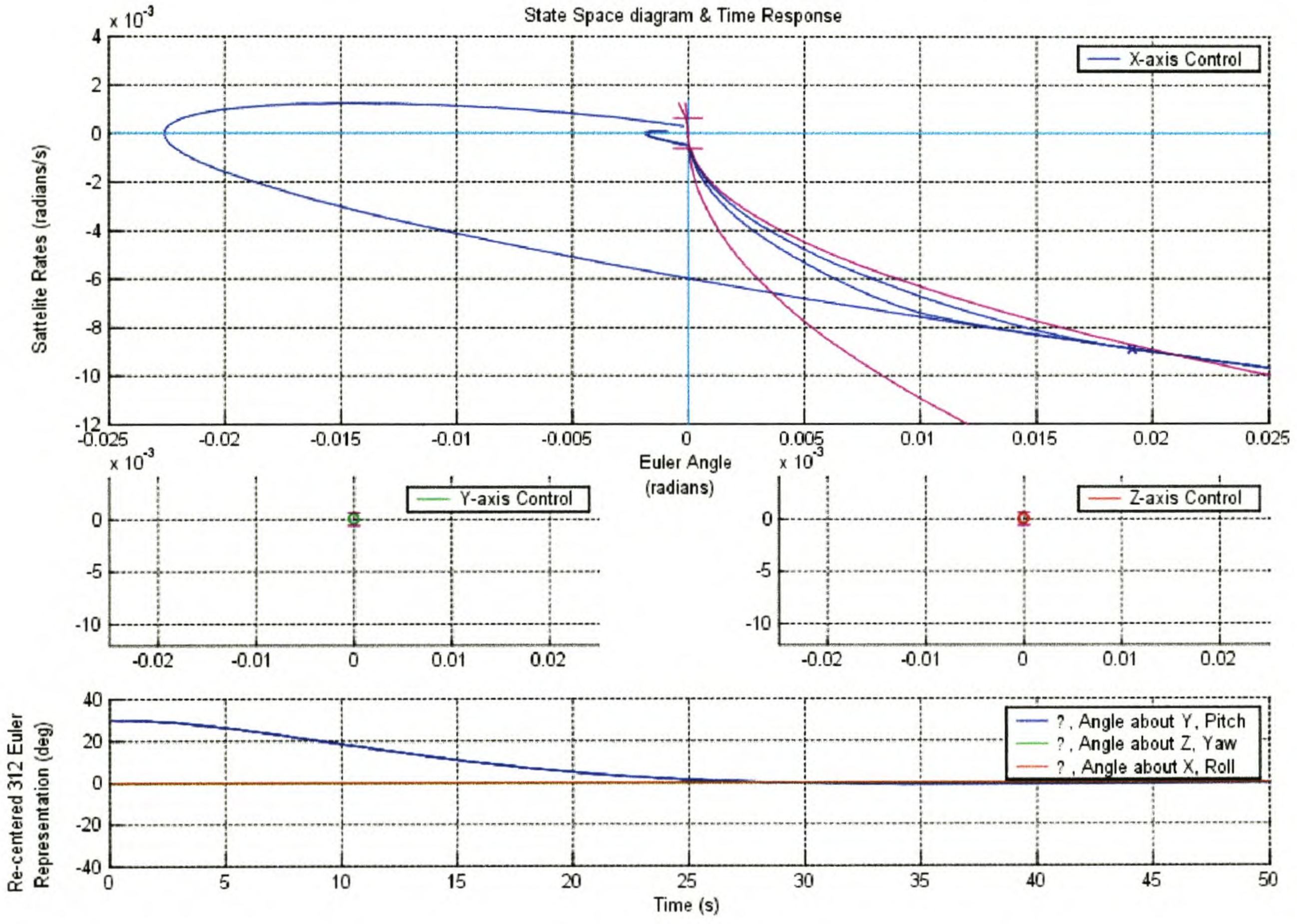


Figure 5.6: Comparing Control Profiles in Ideal Environment

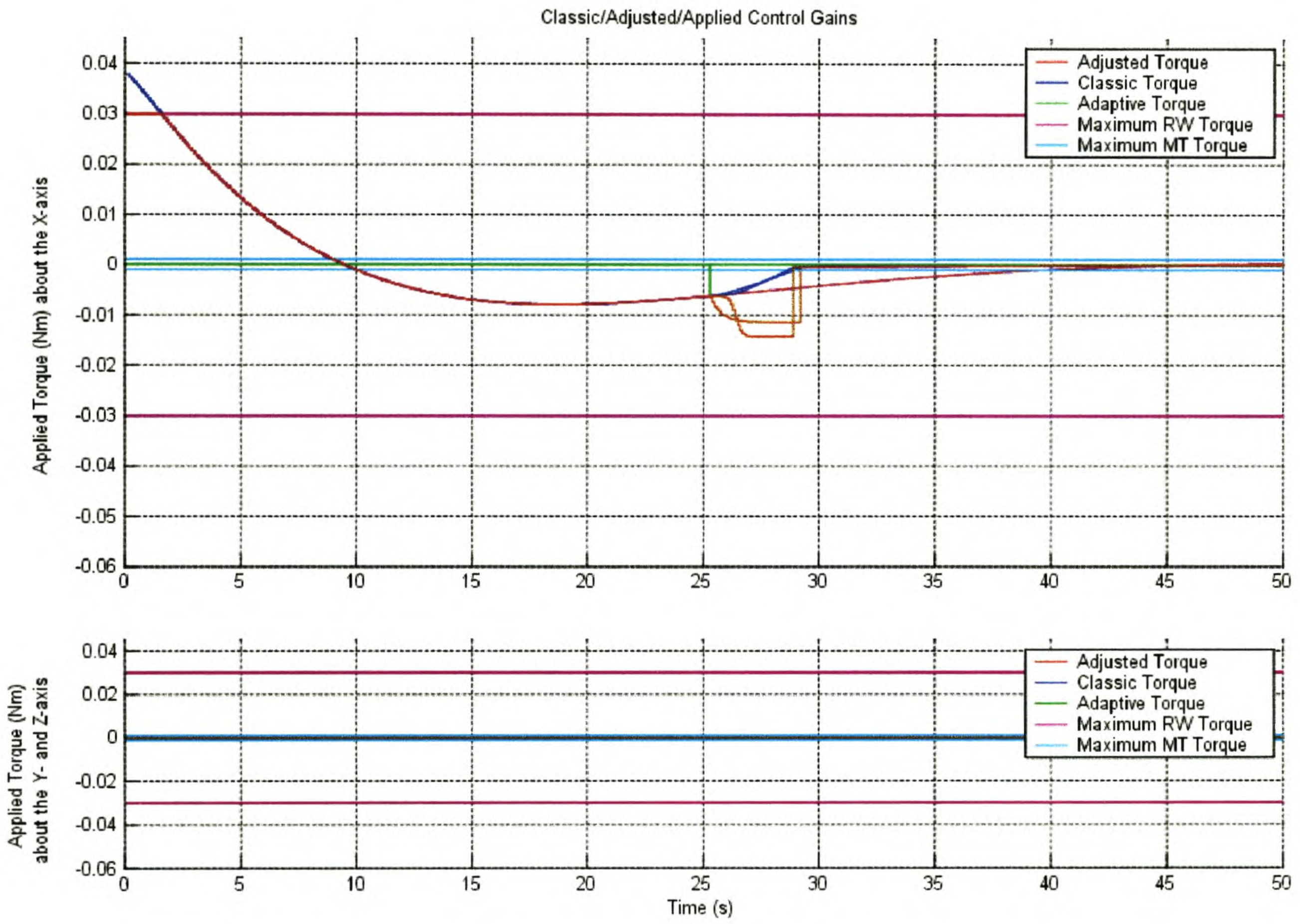


Figure 5.7: Actuator Torques During Comparison

Table 5.2: Summary of Noise Conditions during Tests

Noise Source	**Value**	**Units**
Reaction Wheel Misalignment	[10°,10°,0°]	3-1-2 Euler Rotation
Gaussian Noise	0 ± 10	mN
Actuator Delay	2	seconds

5.3.3 Comparison of Performance under Noise Conditions

In all cases where comparisons under noise conditions were done, the noise profile was recorded and applied equally to all simulations to negate the possibility that the comparison could be influenced by random factors. Table 5.2 gives a summary of these conditions.

Figure 5.8 shows the resulting traces of the satellite orientation under the control of the individual three schemes.

No irregularities unique to the adaptive schemes were discovered that could potentially lead to instability in noisy environments. This can be attributed to the fact that for both very large and very small error angles, the system is still under control of the stable LQR regulator in all three cases. The adaptive schemes just minimize the time spent crossing the middle boundary.

A more rigorous stability analysis will be done in Section 5.4.

Comparisons of the settling times and overshoot of the different schemes under these conditions were complicated by the fact that the noise constantly pushed the error angle outside of the 2% boundary and also affected the overshoot measurement.

In all the simulations that were done, however, the two adaptive schemes consistently outperformed the standard design in both settling time and overshoot reduction.

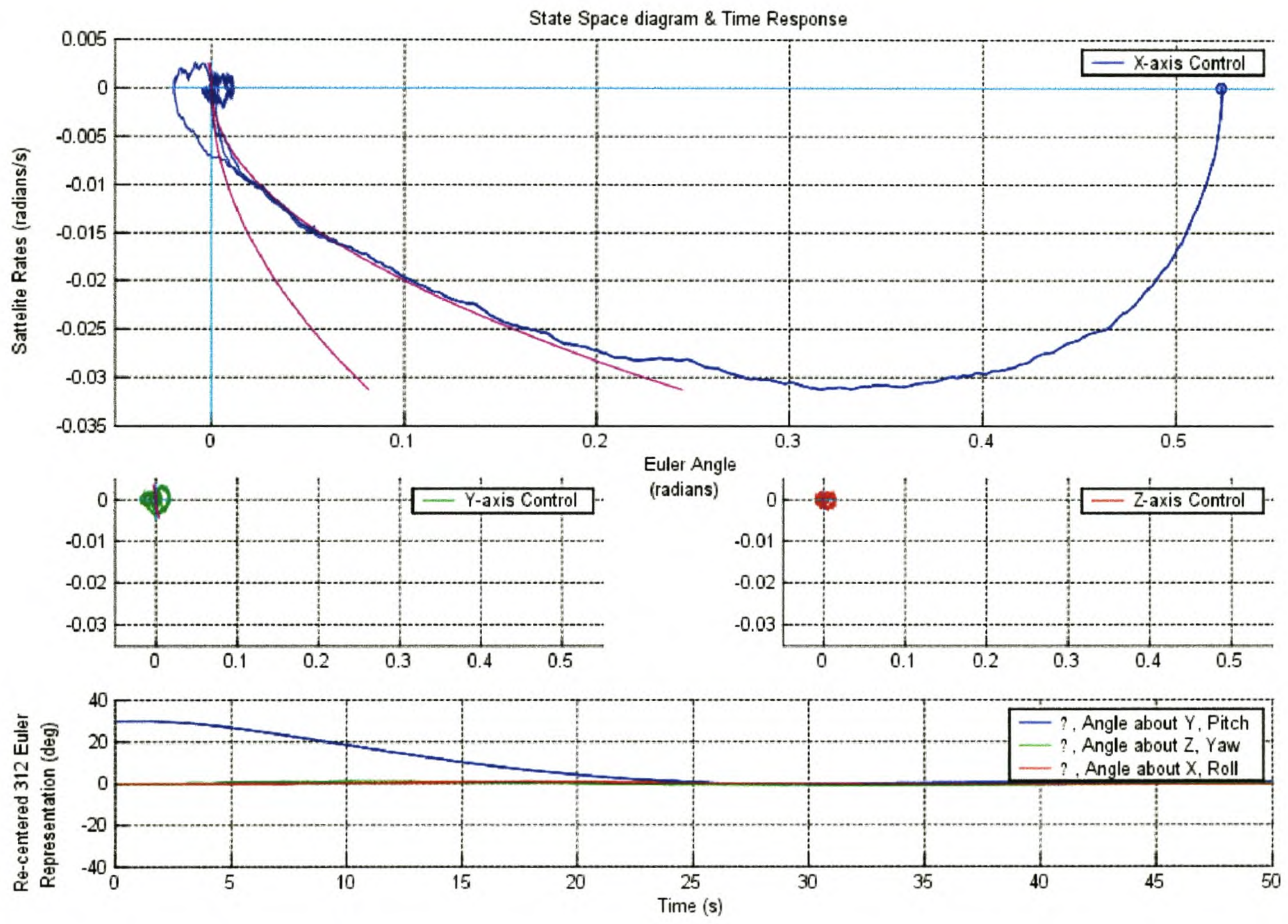


Figure 5.8: Comparing Control Profiles in a Noisy Environment

Table 5.3: Comparison of Simulation Results under Fault Conditions

Figure	Control Scheme	2% Settling Point	Overshoot		
		at	Maximum Overshoot	at	Percentage of Initial Value
5.9	LQR	92.7s	5.7951°	37.9s	19.3169 %
5.9	AS1	24.3s	0.2092°	41.1s	0.6975 %
5.9	AS2	56.0s	1.3078°	40.0s	4.3595 %

5.3.4 Investigating the Effects of Reaction Wheel Failure

The primary goal of this thesis is to determine whether adaptive control can improve the stability and performance of the orientation controller during large actuator gain changes.

The three main causes of these gain changes were introduced in Section 2.3.1.

Actuator failure was simulated by dropping the reaction wheel torque to 50 percent of its maximum, 10 seconds into the maneuver. This affected both the reaction wheel's response to command signals and it's maximum available torque.

This simulates the case where there are two reaction wheels per axis, and one of them fails.

Figure 5.9 shows the resulting state-space and time traces of all three schemes. The LQR is now clearly underdamped, shooting past the command angle to reach an overshoot of almost 20% .

The two adaptive schemes perform much better than the reference design, their performance at least *comparable* with earlier tests. The results of the test are summarized in Table 5.3.

Figure 5.10 show the control torques during the test for the three schemes.

The LQR regulator never requests enough torque during the maneuver to reach the reaction wheel maximum, but the drop in actuator gain causes the satellite to overshoot its mark significantly.

Adaptive scheme one increases torque output as it recognises that the satellite is not responding as fast as it should, and manages to reach the dead-zone just as the reaction wheel clips at maximum torque.

Adaptive scheme two does not manage quite as well, pressing the reaction wheel a bit harder and reaching maximum torque earlier. Without sufficient energy available to bring the satellite to a halt at it's goal, it finally falls out of the active region and returns control to the LQR regulator. The result is a significant increase in overshoot, but it is still less than that of the reference design.

5.3.5 Investigating the Effects of Actuator Switching

In the second investigation, the actuator gain was suddenly increased while the controller was operating. This would happen in the event that the reaction wheels were turned on during a maneuver, or if a very accurate sensor comes into operation, causing a sudden increase in available bandwidth.

In the simulation environment, this situation was created by bringing the reaction wheels on line 72 seconds into the motion. Initially, only the magnetorquers are in operation. The time was chosen to coincide with the trace's approach to the active region in order to examine the reaction of the different controllers during this time.

The effect is the same at any other time, but the dynamics are then dominated by the response of the LQR regulator, making distinguishing features difficult to see on the graph.

Figure 5.11 shows the traces. As is the case with all the comparative graphs, the LQR regulator trace is furthermost to the left, resulting in the most

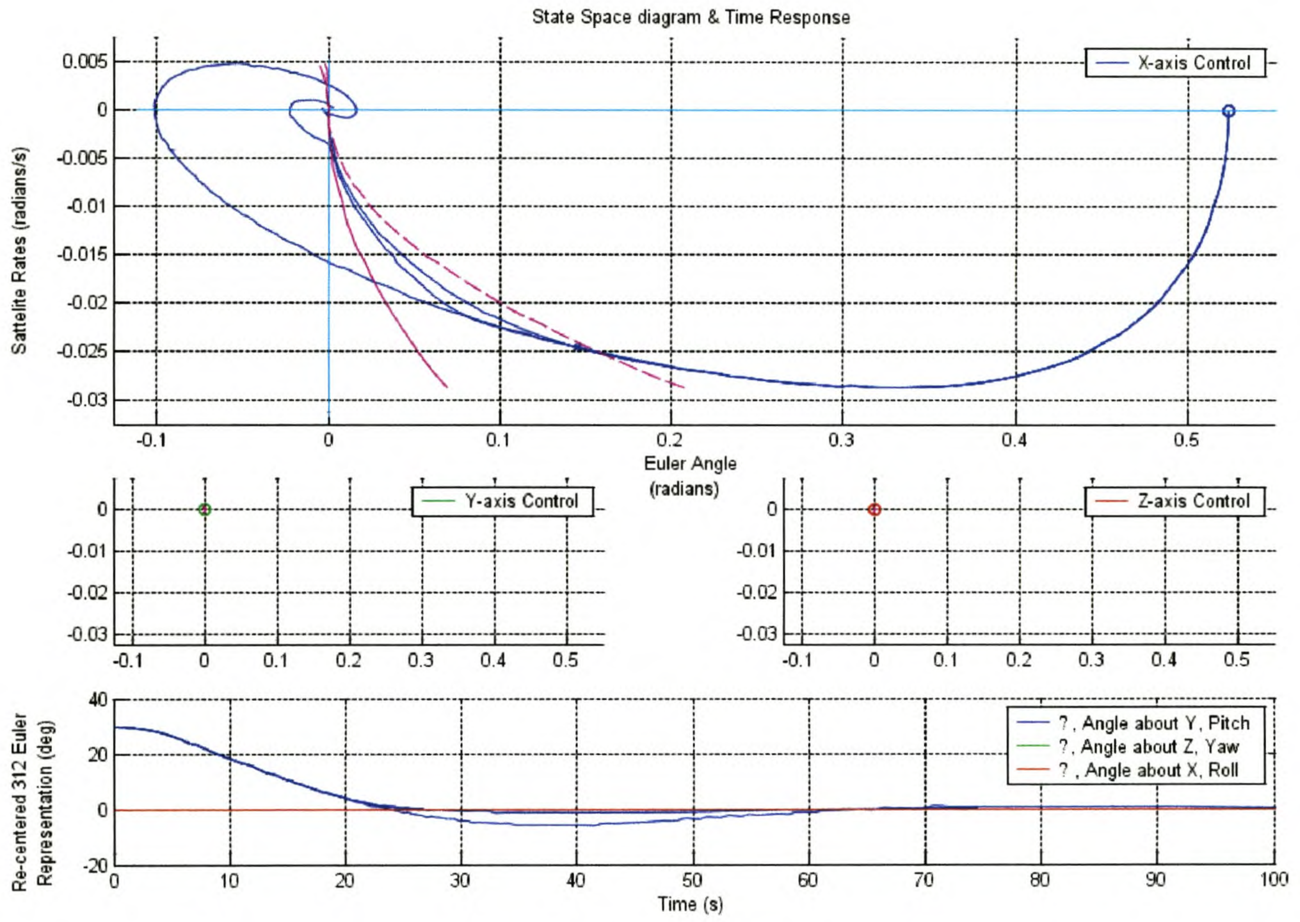


Figure 5.9: Comparing Control Profiles during Actuator Failings

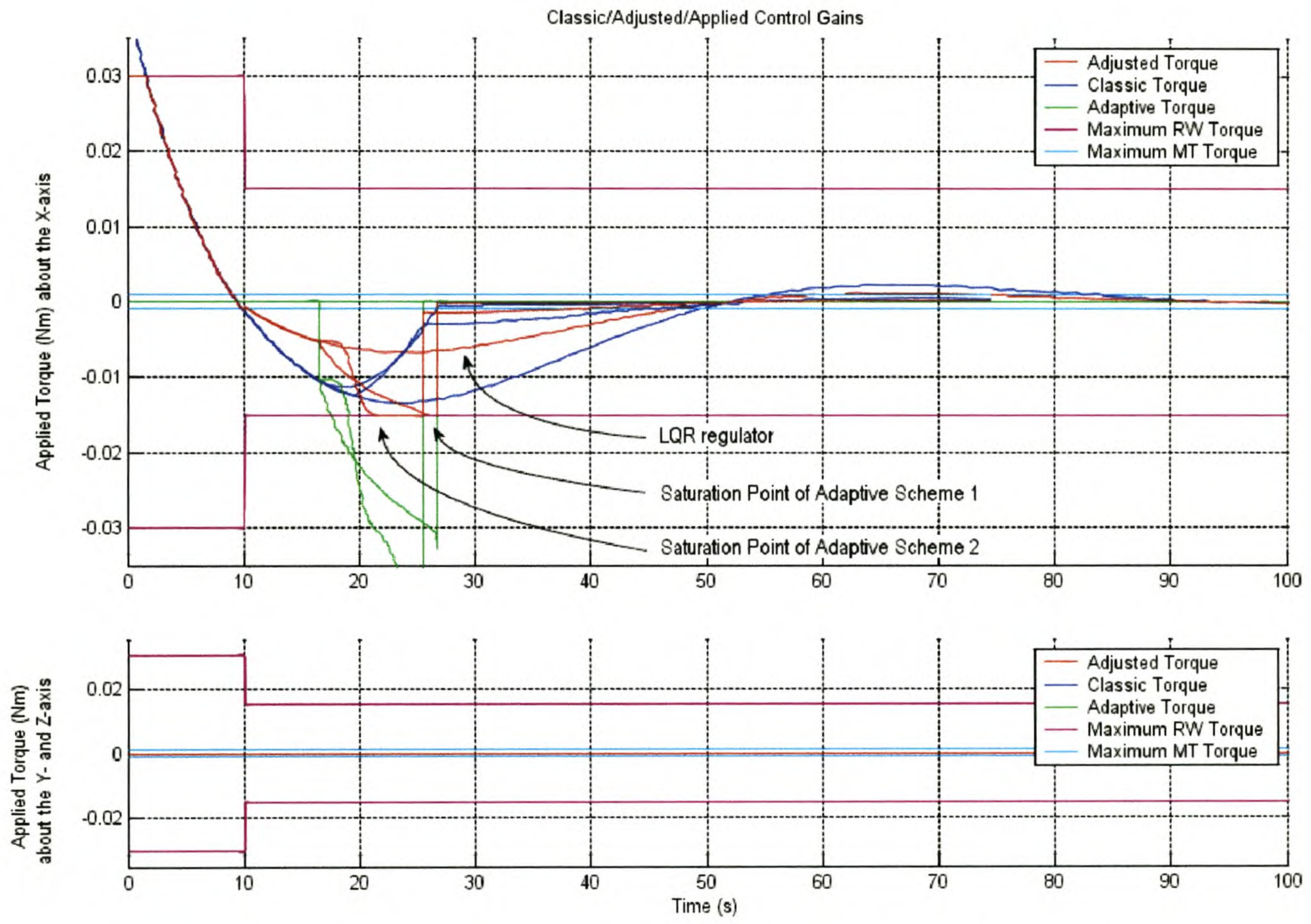


Figure 5.10: Comparing Control Torques during Actuator Failures

Table 5.4: Comparison of Simulation Results during Actuator Switching

Figure	Control Scheme	2% Settling Point	Overshoot		
		at	Maximum Overshoot	at	Percentage of Initial Value
5.11	LQR	90.6s	0.9924°	82.7s	3.3079 %
5.11	AS1	73.9s	0.0534°	80.2s	0.1779 %
5.11	AS2	73.8s	0.0374°	79.3s	0.1248 %

overshoot. The adaptive controllers follow close together and remain in the active region, approaching the zero point almost directly, with scheme two slightly to the left of scheme one.

The dynamics very close to zero are similar to that shown close-up in Figure 5.6.

None of the controllers, including the standard LQR design, have any problems in dealing with this situation, diminishing its usefulness in selecting a control scheme.

It is clear, however, that no serious questions about system stability arose from such an event.

For comparison, the results of the test are summarized in Table 5.4 on page 53.

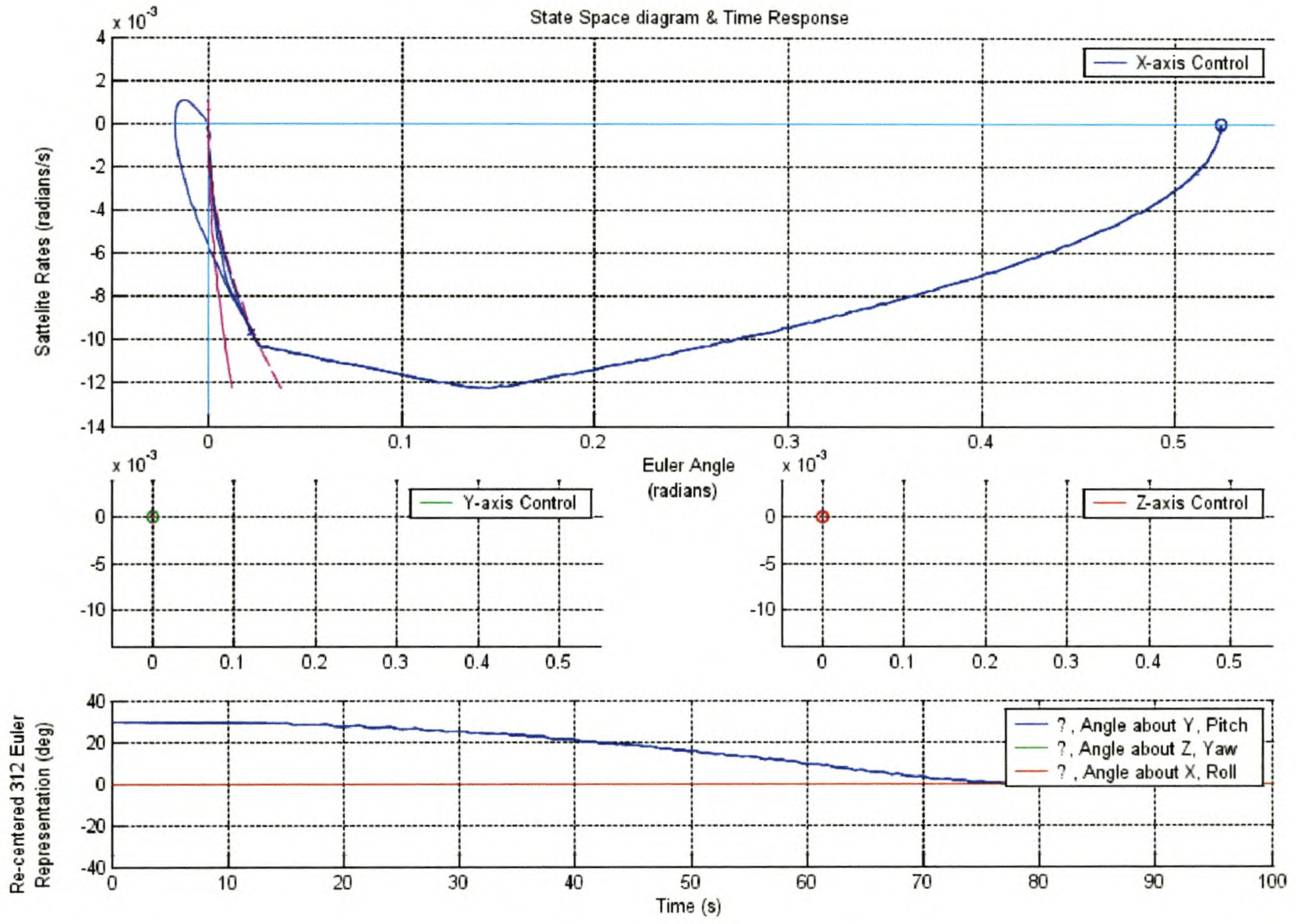


Figure 5.11: Comparing Control Profiles during Actuator Activation

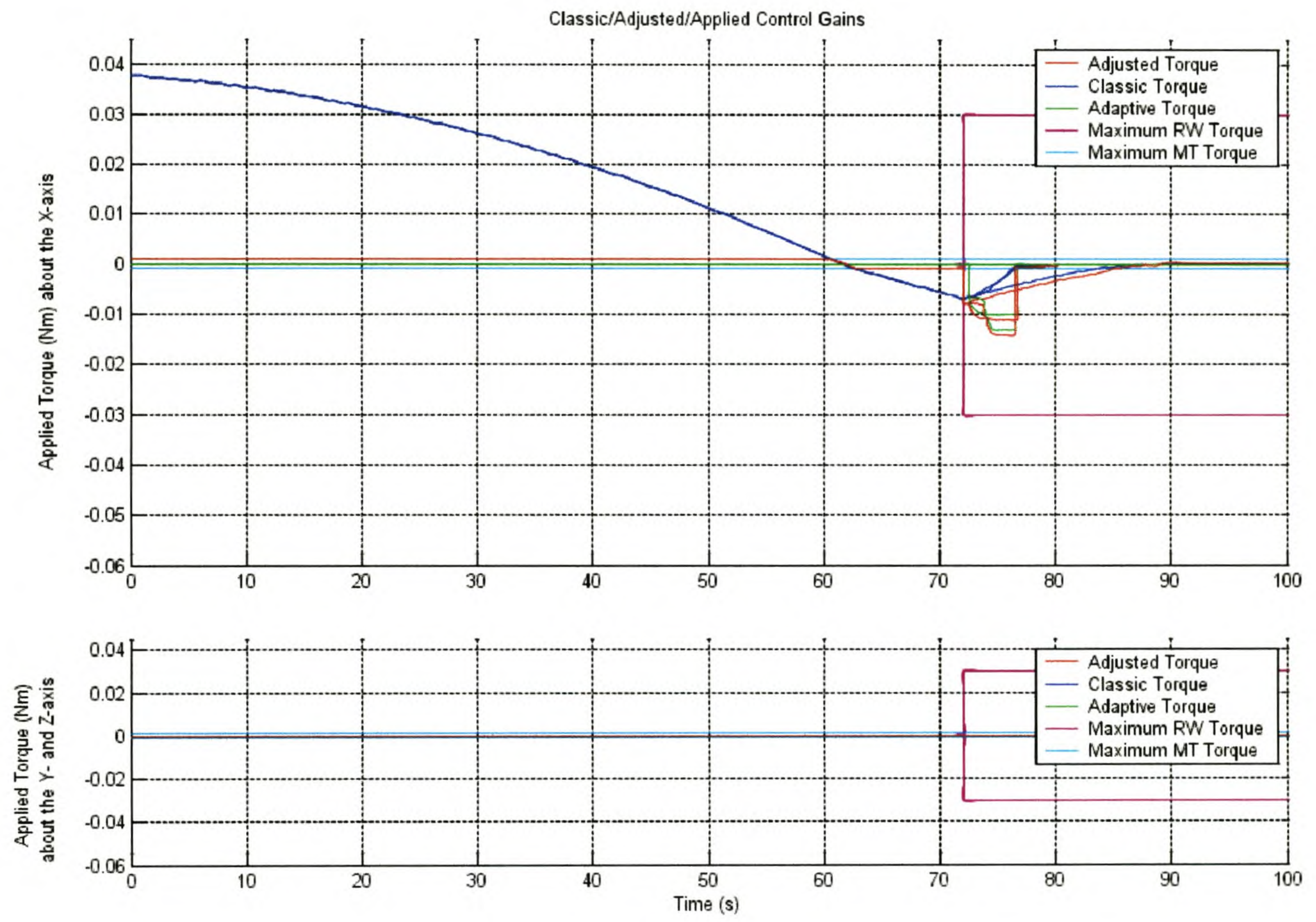


Figure 5.12: Comparing Control Torques during Actuator Activation

5.4 Lyapunov Stability Analysis

In 1892, A.M. Lyapunov presented two methods (called the first and second methods) for determining the stability of dynamic systems described by ordinary differential equations.

The first method consists of all procedures in which the explicit form of the solutions of the differential equations are used for the analysis.

The second method, on the other hand, does not require the solutions of the differential equations. That is, by using the second method of Lyapunov, we can determine the stability of a system without solving the state equations.

This is ideal for our situation in which we have to analyse the stability of a time-*variant* system.

Lyapunov's second method holds that an equilibrium state of a system is stable if, for any trace starting at a point in a region around the state in question (in our case the zero point on the state-space graph), the trace will remain closer than a certain finite distance from the centre for all time, i.e. the initial errors in angle and rate will not grow without bound. This upper bound may be dependant on both the starting point of the trace and the profile of how the system dynamics change over time.

For a more general, formal definition of Lyapunov's methods see Ogata [7].

As a point of reference, Figure 5.13 shows the stability analysis of the system under LQR control in an Ideal environment. Any starting state results in a trace that strives to zero control error. The system is clearly stable in the region under consideration, as expected.

With respect to the region for which these tests are conducted, we expect the system to be able to recover from any initial error angle. Even though the saturation of the reaction wheels can be clearly seen on the figure, the satellite should be able to recover from any initial error rate due to the infinite capacity of the magnetorquers.

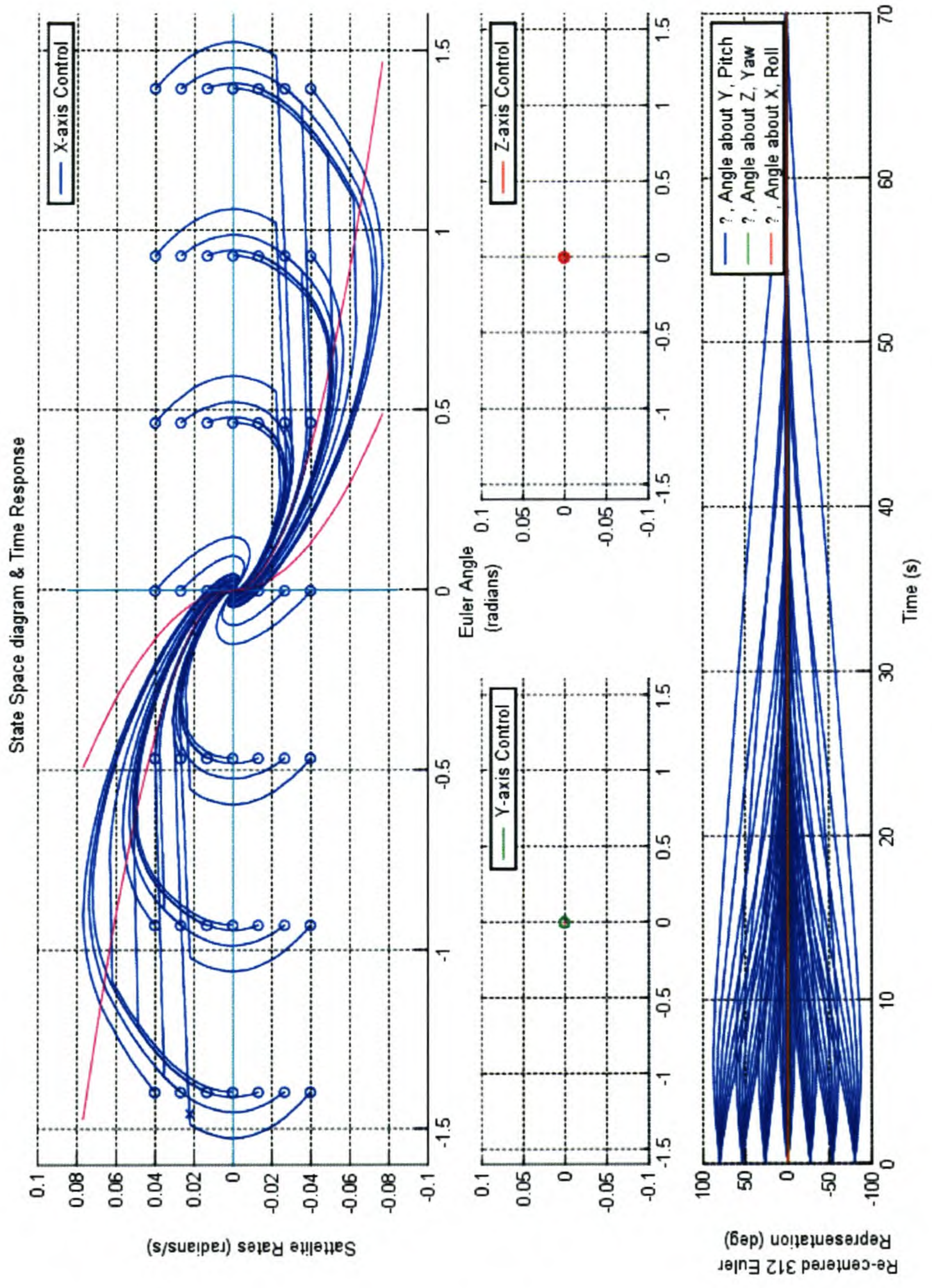


Figure 5.13: Linear Quadratic Regulator: Lyapunov Stability Test without Noise

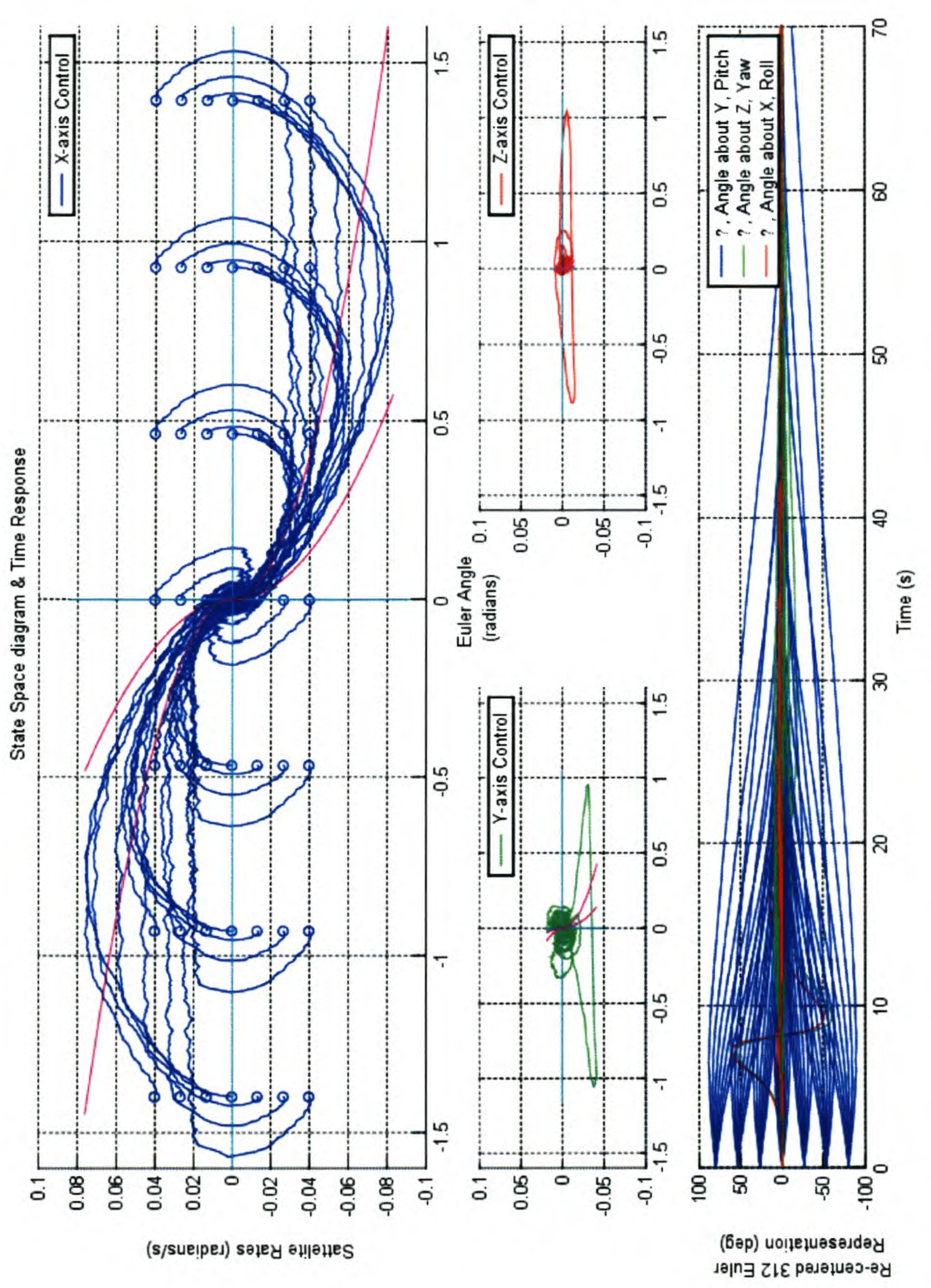


Figure 5.14: Linear Quadratic Regulator: Lyapunov Stability Test

Table 5.5: Summary of Noise Conditions during Lyapunov Trials

Noise Source	**Value**	**Units**
Reaction Wheel Misalignment	[10°,10°,0°]	3-1-2 Euler Rotation
Gaussian Noise	0 ± 30	mN
Actuator Delay	2	seconds

The saturation of the actuator causes an abrupt halt to the smooth recovery normally associated with LQR control and the trace from then on runs almost horizontally. What happens at this point is that all torque from the reaction wheels in one direction is lost, but the magnetorquers continue to function. They still obey the controller command to increase the satellite rate in order to correct the error in orientation angle, but cannot do so as quickly as the reaction wheels and magnetorquer combined, resulting in a flattening of the slope of the trace.

When the saturated system approaches zero orientation angle, the controller commands the actuators to apply torque in the reverse direction, causing the reaction wheels to come out of saturation (similar to Figure 5.11) and normal operation continues.

We are interested in stability in the real environment, however. The noise levels shown in Table 5.5 were added to all further tests.

It should be noted that no test can prove stability in the presence of unpredictable and unbound external torques. The Lyapunov trials shown here were included solely to help identify unexpected dynamics that would lead to further investigation.

Figures 5.14, 5.15 and 5.16 show the performance of the LQR regulator and the two adaptive schemes respectively, under these conditions.

All three systems pass the stability tests satisfactorily and no new questions are raised.

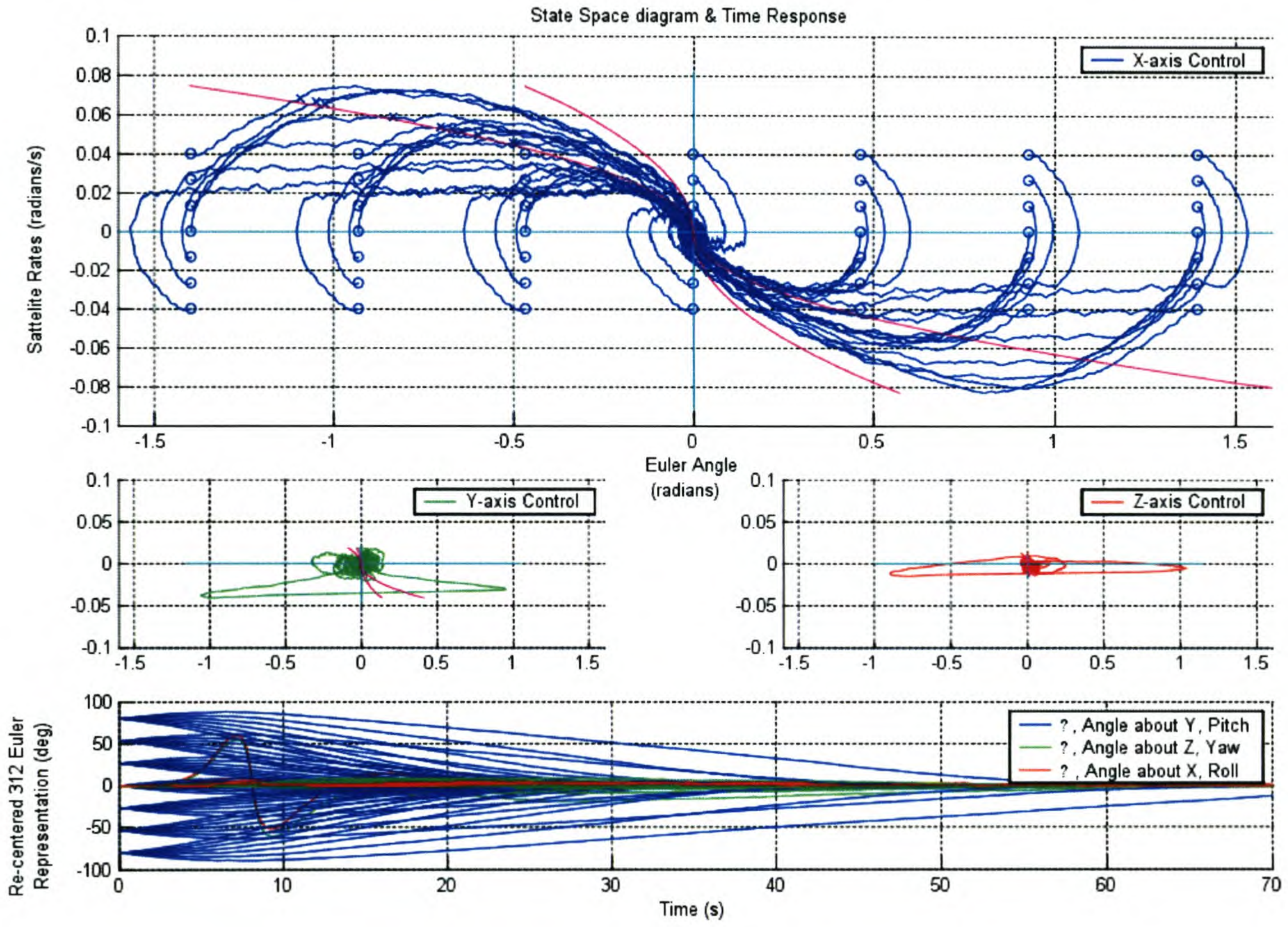


Figure 5.15: Adaptive Scheme One: Lyapunov Stability Test

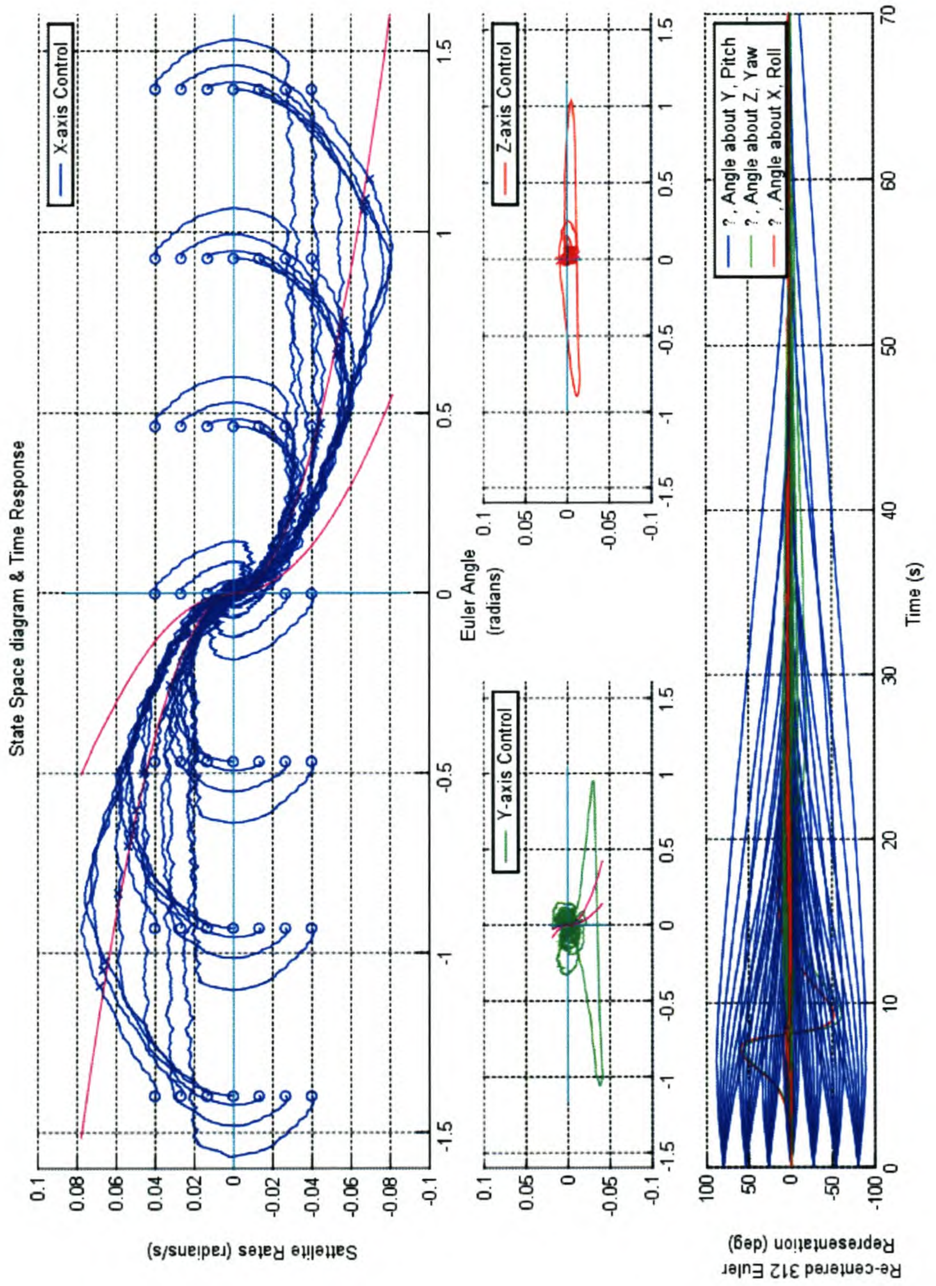


Figure 5.16: Adaptive Scheme Two: Lyapunov Stability Test

5.5 Summary of Simulation Results

Both adaptive schemes performed consistently better than the standard LQR regulator in all the test cases, exhibiting improved overall response times and reduced overshoot of the orientation error angles.

All three schemes are robust and perform well in the presence of external disturbance torques. No significant degradation in performance was observed with disturbance levels well above their expected values.

The adaptive schemes showed considerably better performance than the standard controller in situations of severe actuator gain changes. Adaptive scheme one maintained performance levels in situations where only half the available torque was available. Adaptive scheme two fared marginally worse in this design, but is expected to match or even exceed the performance of scheme one with only minor modifications to the active region.

All three control schemes exhibit stable behaviour during the extensive Lyapunov trials. These trials cannot be considered proof of stability, but lends strong credibility to an argument supporting such a claim.

Chapter 6

Conclusion

6.1 Control Schemes in Perspective

Two adaptive schemes were presented to deal the reduction in performance of the orientation controller and with stability issues arising from large actuator gain changes during the satellite mission.

The results of the simulations presented in Chapter 5 show that this increase of performance have been achieved without reducing the stability of the system.

It is clear that many benefits may be gained from using compound control schemes such as those of the adaptive schemes developed here, but that extensive simulation is necessary to investigate the limits and dangers of using such techniques.

Many formal adaptive techniques have been developed in the literature, but their application depend on a thorough understanding of the system and the extent and nature of changes in it's dynamics, often supported by close integration with state estimators like Kalman filters.

This thesis presented a comparatively simple adaptive scheme to solve the problem of large gain changes, and in the process produced a good understanding of the system elements involved in such a design. The simulation models that were developed are generic and can serve as a thorough testing ground for future control schemes.

Any further research in this field will be well served by this work.

6.2 Recommendations

6.2.1 Improvements to the Control Scheme

Secondary Control Scheme

Improvements to the control profile in areas outside the active region can greatly increase performance of the satellite orientation control.

Possibly applying a similar banded control methodology to identify system response even before the system enters the active region could both enhance the systems response times and increase it's stability.

Inclusion of a State Estimator

All of the more efficient and robust adaptive control schemes depend heavily on the continuous and accurate identification of critical system parameters.

For any serious adaptive scheme to be implemented for orientation control, a state estimator, like an extended Kalman filter, needs to form an integral part of the control design.

Integrating Three Axis Control

One of the most formidable challenges lie in the integration of the separate controllers of the three axes of the satellite.

The benefits of using such an integrated system is that any orientation may be achieved by using only two of the three actuator axes. All aircraft, for example, use only their pitch and roll axes to achieve any desired orientation.

This means that control can be maintained to a degree even when an entire actuator channel is lost. Orientation control may also take into account the available torque of each reaction wheel channel and choose a path that will avoid or limit reaction wheel saturation.

6.2.2 Improvements to Simulation Environment

Accurate Noise Models

A Gaussian Noise model is in many ways a very forgiving source of noise. More accurate noise models for the torques acting upon the satellite during it's mission may highlight new complications and lead to better controllers.

Hardware-in-the-Loop

A hardware test-bed can help identify some physical aspects to the control solution that may not be apparent in the computer simulations. Controller sampling times and delays may be accurately measured and the effects of control schemes on supply currents and actuator lifetimes may be better understood.

A low-cost spacecraft simulator was presented in IEEE Control Systems Magazine, August 2003 [4] that could greatly benefit future research in this area.

Bibliography

[1] Gene F. Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems.* Addison-Wesley Longman, Inc., third edition, 1998.

[2] R. Ghanadan. Adaptive pid control of nonlinear systems. Master's thesis, University of Maryland, 1990.

[3] S.N. Huang et al. A combined pid/adaptive controller for a class of nonlinear systems. *Automatica*, 37, 2001.

[4] ByungMoon Kim et al. Designing a low-cost spacecraft simulator. *IEEE Control Systems Magazine*, August 2003.

[5] E. Maass. Integrated attitude determination system using a combination of magnetometer and horizon sensor data. Master's thesis, University of Stellenbosch, 2003.

[6] Garth W. Milne et al. Sunsat - launch and first six month's orbital performance. *13th Annual AIAA/USU Conference on Small Satellites*, 1999.

[7] Katsuhiko Ogata. *Modern Control Engineering.* Prentice-Hall, Inc., third edition, 1997.

[8] Joseph P. Pickett et al., editors. *The American Heritage Dictionary of the English Language.* Houghton Mifflin Company, fourth edition, 2000. www.bartleby.com/61/.

[9] Shankar Sastry and Marc Bodson. *Adaptive Control: Stability, Convergence, and Robustness.* Prentice-Hall International, Inc., 1989.

[10] W. H. Steyn. *A Multi-Mode Attitude Determination and Control System for Small Satellites.* PhD thesis, University of Stellenbosch, 1995.

[11] James R. Wertz, editor. *Spacecraft Attitude Determination and Control.* D. Reidel Publishing Company, 1978.

Appendix A

Cooridinate Systems

Various systems exist with which the positions of celestial objects are described relative to each other. In spacecraft work, where almost all aspects of the mission are specifically designed and tailored to each individual satellite, the definitions of well known systems are not always the same. In the rest of this section, the coordinate systems used in this thesis are briefly described, along with an explanation of where they are commonly used. For a more complete definition of these standards, see *Spacecraft Attitude Determination and Control* [11] by Wertz.

A.1 About Coordinate Systems

In orbital work, it is preferred to specify position and orientation in spherical coordinates, because it is easier for people to visualize. These spherical coordinate systems have a number of properties in common.

Each spherical coordinate system has two *poles* diametrically opposite each other on the celestial sphere and an *equator*, or great circle, halfway between the poles. This is shown visually in Figure A.1. A great circle of a sphere is defined as the circle of intersection when a flat plane separates the sphere into

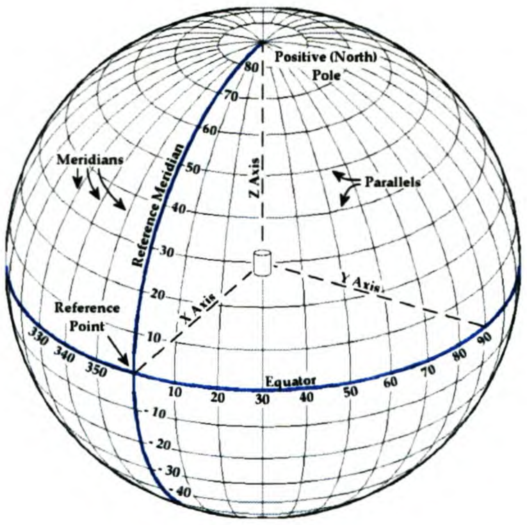


Figure A.1: Spherical Geometry in Space

two parts of equal size. A small circle is any other circle created when a flat plane divides the sphere into unequal parts. The great circles through the poles and perpendicular to the equator are called *meridians* and the small circles a fixed distance above and below the equator are called *parallels.*

A spherical coordinate system is therefore fully specified by indicating it's positive pole and it's *reference* meridian or *reference point.* The reference point is the point of intersection between the reference meridian and the equator. Sometimes a *primary axis* is indicated instead of the positive pole, referring to the axis connecting the two poles of the system. The specification should then also indicate which side of the axis is considered to be the positive side.

The position of any point on the sphere can be described by two components, the *azimuth* and *elevation.* These same two components are called the *longitude* and *latitude* when used to describe positions on the surface of the Earth and *right ascension*, α, and *declination*, δ, when describing direction in the spacecraft centred inertial system discussed in Section A.4.

Azimuth is measured along any of the parallels from 0°at the reference meridian to 360°, while elevation is measured along a meridian i.e. perpendicular to the equator from 0°at the equator to 90°at the north pole and -90°at the south pole, respectively. At either pole, the azimuth component is undefined.

Six coordinate systems were used in this thesis for describing various relations. These systems are summarized in Table A.1.

A.1.1 Geocentric Inertial Coordinates

The *geocentric inertial coordinate system* GEO is primarily used to describe the position of the spacecraft in its orbit about the Earth. This position is used by the simulation to calculate the Earth's magnetic field strength at the satellite's position. The position of the Sun relative to the Earth is also described in geocentric inertial coordinates. No consideration has been given

Table A.1: Coordinate Systems used in this Thesis

Coordinate System	**Reference Name**	**Fixed w. respect to**	**Centre**	**Z-axis or Pole**	**X-axis or Reference Point**
Geocentric Inertial	GEO	Inertial Space[a]	Earth	Celestial or Earth Pole	Vernal Equinox[b]
Earth Fixed	EAR	Earth	Earth	Earth Pole	Greenwich Meridian
Orbital	ORB	Orbit	Spacecraft	Nadir	Perpendicular to Nadir toward velocity vector
Spacecraft Centred Inertial[c]	SCI	Inertial Space	Spacecraft	Celestial Pole	Vernal Equinox
Body	BOD	Spacecraft	Spacecraft	Spacecraft axis toward nadir	Spacecraft axis in direction of velocity vector
Local Tangent	TAN	Oblate Earth	Spacecraft	*See Text*	*See Text*

[a]Actually rotating slowly with respect to inertial space. See text for discussion.
[b]Also known as the First Point of Aries
[c]Also known as Celestial Coordinates

to the Earth's motion about the Sun, due to it's relatively small variation during a typical simulation period, so this position remains constant.

The system is centred on the Earth and fixed with respect to inertial space, with it's positive pole at the north pole of the Earth, and it's reference meridian running through the vernal equinox, as shown in Figure A.2.

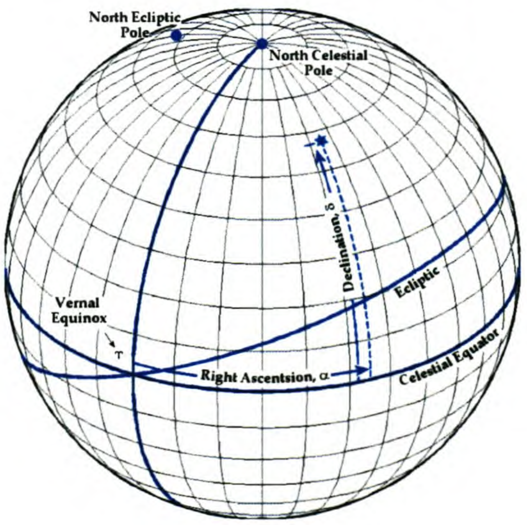
The vernal equinox is the point where the *ecliptic*, or plane of the Earth's orbit about the Sun, crosses the Earths equator going from south to north. It is also the direction parallel to the line connecting the Earth and the Sun on the first day of spring.

A.2 Earth Fixed Coordinates

In conjunction with the *local tangent coordinates* described in Section A.6, *Earth fixed coordinates* is used to determine the Earth's magnetic field strength at the satellite's position.

The system is identical to the geocentric inertial coordinates defined earlier, with the exception that it's reference meridian is fixed relative to the surface

Figure A.2: Geocentric Inertial Coordinates



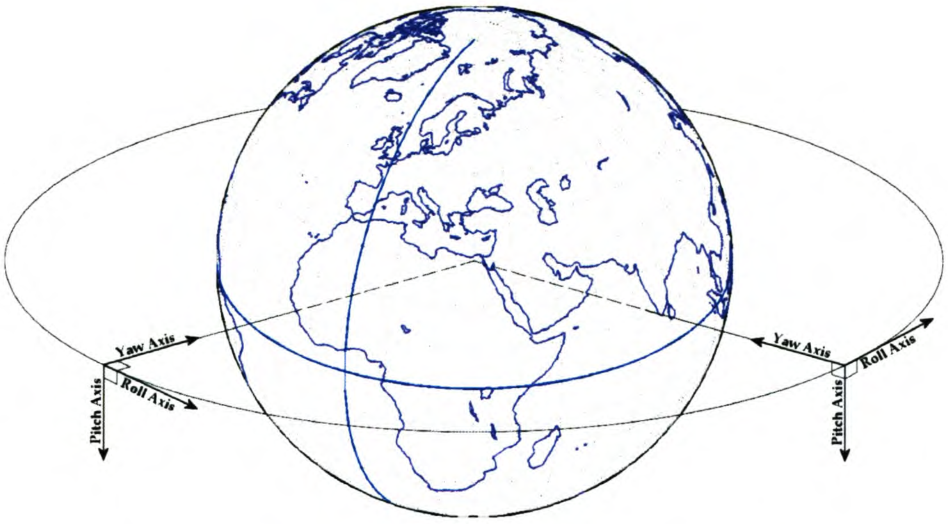


Figure A.3: Roll, Pitch and Yaw RPY Coordinates

of the Earth. The Greenwich meridian is universally used as the reference meridian for this coordinate system. The Greenwich meridian is defined as running through the former Royal Greenwich Observatory in metropolitan London.

A.3 Orbital Coordinates

Orbital coordinates is one of the primary coordinate systems used in the simulation. All control commands are given in orbital coordinates, and the orientation of the satellite is described in it.

In spherical and cartesian systems, where the coordinates are used to describe the position of points in the system, at least two separate points are needed to specify the orientation of objects in the system unambiguously, since rotation about any single vector is still possible. Calculations using this definition is cumbersome, and so another reference system, called *Euler angles*, is used to specify orientation.

Euler angles, as the name suggests, consist of a set of three rotation angles about the centre of mass of the spacecraft, called the *roll*, *pitch* and *yaw*. These rotations are made about three respective axes, following the right had rule. The yaw axis is defined to always point towards the centre of the Earth. The pitch axis lies tangent to the orbital plane in a manner so that when the roll axis completes the coordinate set, it points towards the velocity vector of the satellite. Figure A.3 shows these axes.

In the special case of circular orbits, the roll axis coincides perfectly with the velocity vector. All the orbits considered in this thesis were of this class.

A.4 Spacecraft Centred Inertial Coordinates

In order to simulate the satellites attitude response to internal and external torques, the calculations need to be performed in an inertial reference system. This system is called the *spacecraft centred inertial system* SCI.

Once the simulation updates are done, the satellite orientation is translated into orbital coordinates, which then forms the basis for the *body coordinates*, discussed in Section A.5.

The system is centred on the spacecraft, with the orientation chosen somewhat arbitrarily. Some literature define SCI coordinates to coincide with orbit coordinates at the time the satellite starts its primary mission phase, known as *epoch* time, or at some point in the orbit, like perigee the nearest point to the Earth or apogee the furthermost point from Earth. This thesis defines SCI coordinates in the same manner that GEO coordinates were defined in Section A.1.1, that is with it's axis parallel to the Earth's axis of rotation and it's positive pole in the same direction as the Earth's north pole. It's reference point lies parallel to the line connecting the centre of the Earth with the vernal equinox, also facing the Sun on the first day of spring.

A.5 Body Coordinates

Body coordinates is used to define sensor orientations, therefore all measured data are passed to the control algorithm in body coordinates.

Again, the choice of coordinate system orientation is arbitrary, and here it is defined to coincide with the orbit coordinates described in Section A.3 when the controller error is zero.

The positive pole lies on the line connecting the satellite with the centre of the Earth the yaw axis, and the reference point lies in the direction of the roll axis.

In the Cartesian system, the x-, y- and z-axes coincide with the roll, pitch and yaw axes, respectively.

A.6 Local Tangent Coordinates

The model used to describe the Earths magnetic field, the *International Geomagnetic Reference Field* IGRF, describes the field strength at any point in the *Earth fixed coordinates* as three components along the axis of a locally defined coordinate system called the *local tangent coordinate system.*

The system has it's centre at the point at which the calculation is done the satellite's centre, and it's three axes are called B_r, B_Θ and B_Φ.

B_r always point directly away from the centre of the Earth while B_Θ and B_Φ always point East and North, respectively.

Appendix B

Hardware Test Bed: Schematics and PC Boards

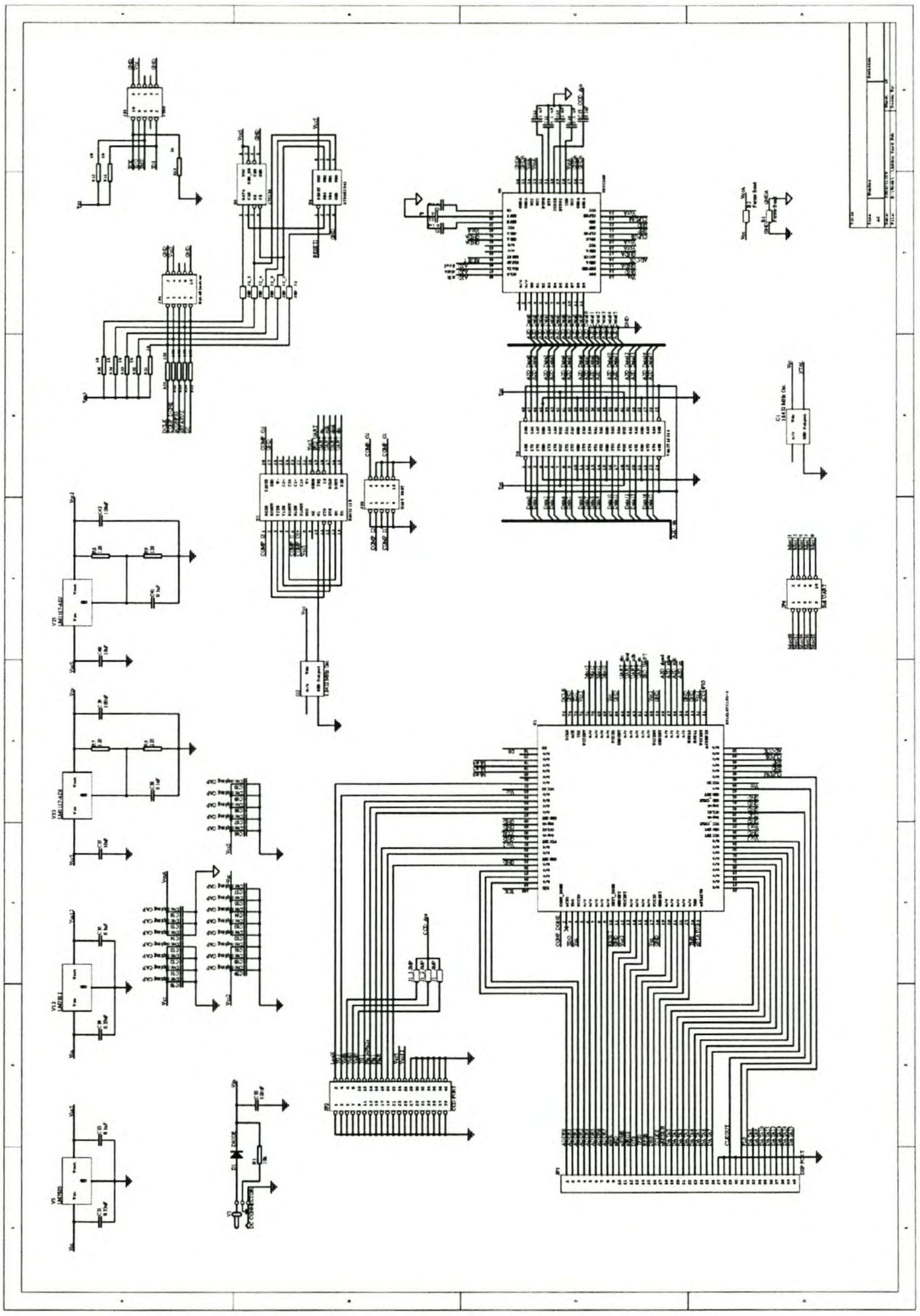


Figure B.1: DSP Add-On Card: Schematic

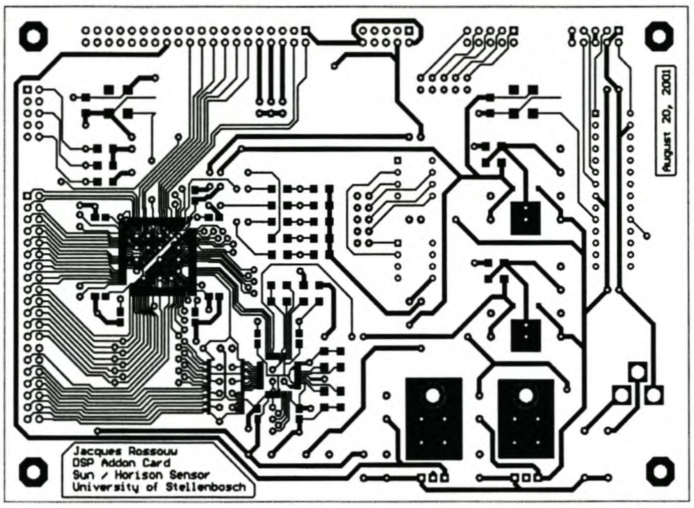


Figure B.2: DSP Add-On Card: PCB Top Layer

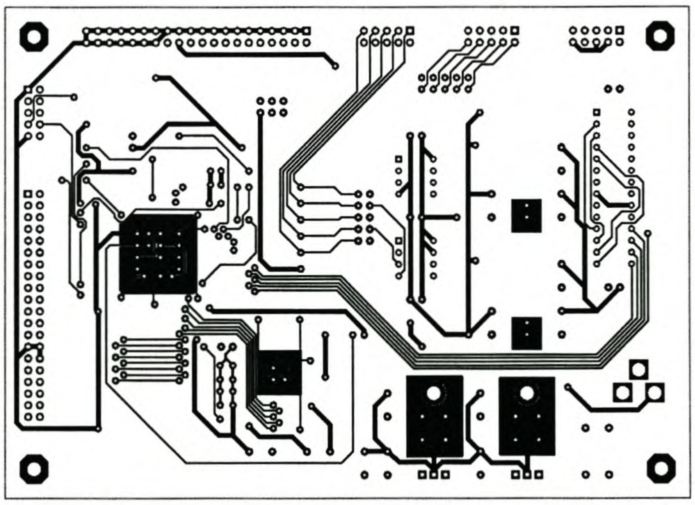


Figure B.3: DSP Add-On Card: PCB Bottom Layer

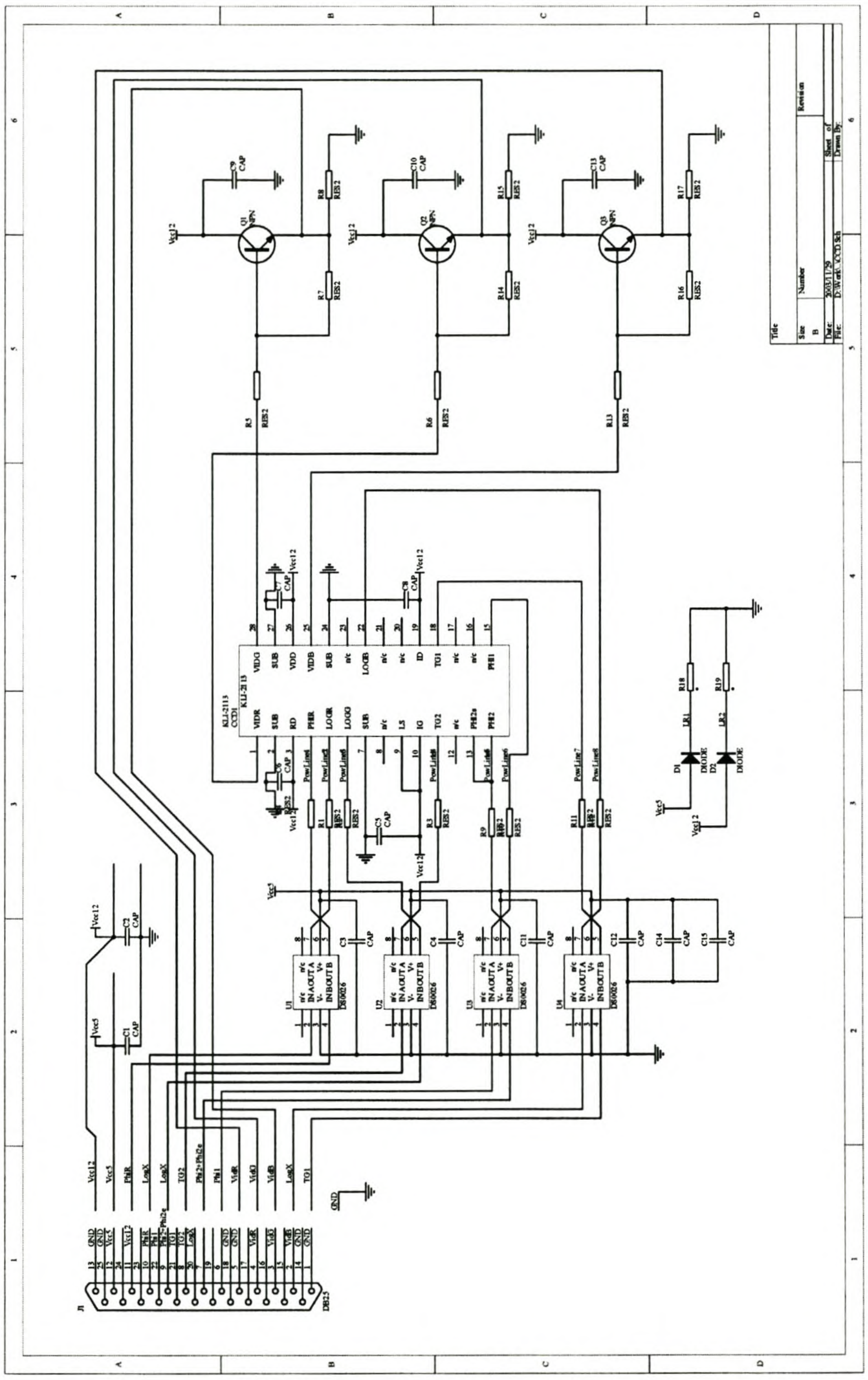


Figure B.4: CCD Board: Schematic

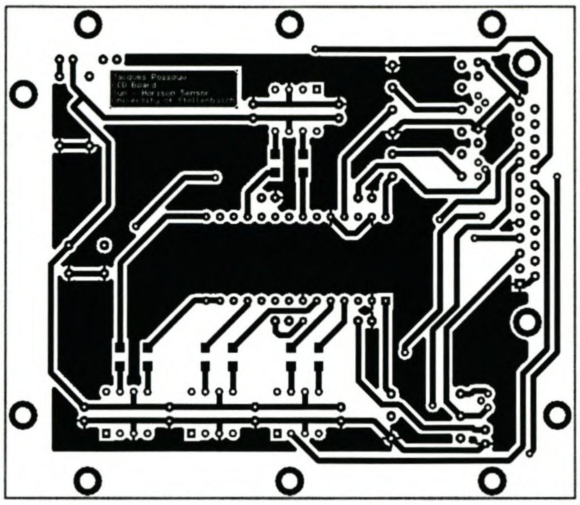


Figure B.5: CCD Board: PCB Top Layer

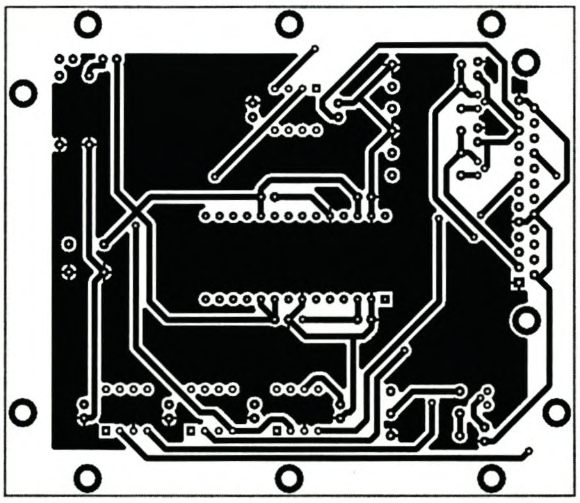


Figure B.6: CCD Board: PCB Bottom Layer

Appendix C

MATLAB$^{\text{TM}}$ Code: Celestial Satellite Environment

main.m

```
% *********************************************************************** %
% main                                                                    %
%                                                                         %
%   Main Program Run Script.  Performs different functions depending     %
%   on what is being tested.                                              %
%                                                                         %
%   v1.1:                                                                 %
%       * added another function to take the sattelite rpy rates as     %
%       input, and return the rpy angles. Sattelite.m                    %
%   v1.2:                                                                 %
%       * added function Attitude.m to take either the SUN or the MAG    %
%       vectors as input, along with an estimation of what they are      %
%       supposed to be in ORB coordinates, and returns the RPY angles    %
%       in BODY coordinates that we have to rotate thru to correct the   %
%       orientation.                                                      %
%                                                                         %
% *********************************************************************** %

% ----------------------------------------------------------------------- %
% Settings                                                                %
% ----------------------------------------------------------------------- %

format compact;
clear all;
Initialize;

drawSUN         = 1;
drawEARTH       = 1;
drawCELaxis     = 0;
drawBODaxis     = 1;
drawORBaxis     = 0;
drawSATstart    = 1;
drawORBtrace    = 0;
drawSUNvector   = 0;
drawMAGvector   = 0;

% ----------------------------------------------------------------------- %
% Imports                                                                 %
% ----------------------------------------------------------------------- %

global TIMEzero

global EARTHradius
global SUNaz_GEO SUNel_GEO
global DCMcel2bod
global DCMcel2orb
global DCMorb2bod

% ----------------------------------------------------------------------- %
% Exports                                                                 %
% ----------------------------------------------------------------------- %

global dt

% ----------------------------------------------------------------------- %
% Main                                                                    %
% ----------------------------------------------------------------------- %
```

```
figure(1); clf; hold on;

SATrpy_GEO = [0 0 0] /180*pi;        % rad

[SATsph_GEO, SATxyz_GEO, startTrueAnomal] = orbit(TIMEzero, [0 0 0]);

allSATxyz_GEO = SATxyz_GEO;
allSUNxyz_BOD = []; allSUNxyz_GEO = []; allSUNxyz_ORB = [];
allMAGxyz_BOD = []; allMAGxyz_GEO = []; allMAGxyz_ORB = [];
Time = [];

dt = 60; time = TIMEzero - dt;
TrueAnomal = startTrueAnomal;

while TrueAnomal < (startTrueAnomal + 2*pi),
    time = time + dt;
    Time = [Time, time];

    X = SATxyz_GEO(1); Y = SATxyz_GEO(2); Z = SATxyz_GEO(3);
    [SATsph_GEO, SATxyz_GEO, TrueAnomal] = orbit(time, SATrpy_GEO);

    %Keep a record of the ORBITAL path
    allSATxyz_GEO = [allSATxyz_GEO; SATxyz_GEO];

    %Determine what the Sunsensors can, and does, see
    [SUNsph_BOD, SUNxyz_BOD, INshadow, INview] = SunVector(SATsph_GEO);
    allSUNxyz_BOD = [allSUNxyz_BOD; SUNxyz_BOD'];
    allSUNxyz_GEO = [allSUNxyz_GEO; (DCMcel2bod'*SUNxyz_BOD)'];
    allSUNxyz_ORB = [allSUNxyz_ORB; (DCMcel2orb*DCMcel2bod'*SUNxyz_BOD)'];

    %Determine what the Magnetometer can, and does, see
    [MAGsph_BOD, MAGxyz_BOD] = MagField(SATsph_GEO);
    allMAGxyz_BOD = [allMAGxyz_BOD; MAGxyz_BOD'];
    allMAGxyz_GEO = [allMAGxyz_GEO; (DCMcel2bod'*MAGxyz_BOD)'];
    allMAGxyz_ORB = [allMAGxyz_ORB; (DCMcel2orb*DCMcel2bod'*MAGxyz_BOD)'];

    %Determine the Attitude command to return to Perfect Orientation...
    DCMcel2bod = DCMcel2orb; DCMorb2bod = DCMcel2bod * DCMcel2orb';
    SATrpy_GEO = dcm2eul('312', DCMcel2bod);

    %sets the drawcolor to reflect the luminosity of the sattelite
    switch INshadow,
        case 1, Color = 'k-'; l = 0.5;
        case 0, Color = 'b-.'; l = 1;
        otherwise,Color = 'm-'; l = 0.75;
    end;
    if INview, Color = 'r-'; l = 1.5; end;

    %%Various plotting routines removed...%%
end;
```

initialize.m

```
% ********************************************************************** %
% Initialize                                                             %
%                                                                        %
%      sets up all the constants used during the simulation.             %
%      this serves as the central place to make changes that affect      %
%      files in the whole program.                                       %
%                                                                        %
%   v1.1:                                                                %
%       * implemented better way of calculating when the sat is in shadow%
%       * added a variable to control the relative start time of the     %
%       simulation.                                                      %
%       * VAR changes:                                                   %
%           out :   SUNtanshadowincl SUNshadowelev SUNshadowaz_GEO       %
%           in:     SATfootprint SUNsize TIMEzero                        %
%                                                                        %
% ********************************************************************** %

% ---------------------------------------------------------------------- %
% Exports                                                                %
% ---------------------------------------------------------------------- %

global ASTRONOMICALunit TIMEzero

global EARTHradius EARTHraGM

global ORBITeccent ORBITperialt ORBITinclin ORBITraAN ORBITarg
global ORBITsemimaj ORBITmean

global SUNaz_GEO SUNel_GEO SUNalt

global SATfootprint SUNsize

global sunSENSORxyz_BOD sunSENSORviewcone

% ---------------------------------------------------------------------- %
% Main                                                                   %
% ---------------------------------------------------------------------- %

% ASTRONOMICAL parameters                                                %
%   unit is the value of one Astronomical Unit                           %
ASTRONOMICALunit    = 1.49597870e11;        % m
TIMEzero            = 0;                    % sec

% EARTH parameters                                                       %
%   radius is the radius of the earth at the equator                     %
%   raGM is the rotation of the earth around it's axis as time goes by   %
%       also known as the right ascension of the Greenwich Meridian      %
%   equincl is the inclination that the earths equator makes with the    %
%       celestial equator                                                %
EARTHradius     = 6.378140e6;               % Equatorial rad of the earth
EARTHraGM       = 90    /180*pi;            % rad

% ORBIT parameters                                                       %
%   eccent describes the shape of the ellipse                            %
%   perialt is the altitude of the orbit at perigree                     %
%   inclin the angle between the angular momentum vector and the unit    %
%       vector in the geocentric z-direction.                            %
%   raAN is the angle from the vernal equinox to the ascending node.  the%
%       ascending node is the point where the satellite passes through   %
%       the equatorial plane moving from south to north.  right ascension%
%       is measured as a right-handed rotation about the pole, Z.        %
%   arg the angle from the ascending node to the eccentricity vector     %
%       measured in the direction of the satellite's motion.  the        %
%       eccentricity vector points from the centre of the earth to       %
%       perigee with a magnitude equal to the eccentricity of the orbit. %
ORBITeccent     = 0.3;                                      % 0 <= e < 1
ORBITperialt    = 800e3;                                    % m
ORBITraAN       = 110   /180*pi;                            % rad
ORBITinclin     = 90    /180*pi;                            % rad
ORBITarg        = 0     /180*pi;                            % rad

% SUN parameters                                                         %
%   due to the scope of time involved in any simulation in this thesis,  %
%   the rotation of the earth about the sun will never have a significant%
%   affect on the environment. the sun is thus described by simple       %
%   parameters that will remain constant throughout the simulation.      %
%                                                                        %
%   radius is the radius of the sun.                                     %
%   raSV is the right ascention of the the sun vector. i.e. the component%
%       along the equator of the angle between the vernal equinox and the%
%       sun vector.                                                      %
%   alt is the distance from the earth to the sun.                       %
%   incl is the inclination angle between the plane of orbit of the earth%
%       around the sun, and the earths equatorial plane.                 %
%   size is the sun's optical size as viewed from 1AU (approx. also      %
%       the size that the sattelite will see it at). it's given as an    %
%       angle from the center of the sun to its optical border.          %
SUNradius   = 6.96000e8;                                     % m
SUNraSV     = 270   /180*pi;                                 % rad
SUNalt      = ASTRONOMICALunit;                              % m
SUNincl     = 23.44 /180*pi;                                 % rad
SUNsize     = 0.53313 /180*pi;                               % rad

% sunSENSOR parameters                                                   %
%   The orientation of the sunSENSORs on the sattelite.  If there is     %
%   more than one sensor, a matrix is specified. The orientation is given%
%   in body coordinates.                                                 %
sunSENSORaz_BOD    = [-167]   /180*pi;                       % rad
sunSENSORel_BOD    = [50]  /180*pi;                          % rad
sunSENSORviewcone  = [20]    /180*pi;                        % rad

% ---------------------------------------------------------------------- %
% CALCULATIONS                                                           %
% ---------------------------------------------------------------------- %

%   semimaj is calculated from perialt and eccent. it is the length of   %
%       the semi-major axis of the orbit ellips                          %
%   mean is a constant related to the speed of the sattelite. see orbit.m%
%       for more details.                                                %
ORBITsemimaj    = (EARTHradius+ORBITperialt) / (1-ORBITeccent); % m
ORBITmean       = 6.3132795e2*(ORBITsemimaj * 1e-3)^(-1.5);     % rad/s

%   az_GEO and el_GEO is the azimuth and elevation angles of the SUN     %
%       vector in Celestial coordinates                                  %
phi     = -SUNincl; Cp = cos(phi);   Sp = sin(phi); %angle about x axis
psi     = -SUNraSV; Cs = cos(psi);   Ss = sin(psi); %angle about z axis
DCM = ...
    [    Cs,     Ss,  0;
     -Cp*Ss,  Cp*Cs, Sp;
```

```
      Sp*Ss, -Sp*Cs, Cp]; % x-z

SUN_GEO = DCM*[SUNalt; 0; 0];
[SUNaz_GEO, SUNel_GEO, SUNalt_GEO] = ...
    cart2sph(SUN_GEO(1), SUN_GEO(2), SUN_GEO(3));

%calculate the sattelites footprint at perigee. defined as the angle     %
%between the nadir vector, and the horizon vector of a circular earth.   %
SATfootprint = asin(EARTHradius / (EARTHradius+ORBITperialt) );     %rad

%transform the sunSENSOR azimuth and elevation angles to x-, y- and      %
%z-coordinates for easy vector comparison later in the program.          %
[sunSENSORx_BOD, sunSENSORy_BOD, sunSENSORz_BOD] = ...
    sph2cart(sunSENSORaz_BOD, sunSENSORel_BOD, 1);
sunSENSORxyz_BOD = [sunSENSORx_BOD; sunSENSORy_BOD; sunSENSORz_BOD];


% ---------------------------------------------------------------------- %
% Error CHECK                                                            %
% ---------------------------------------------------------------------- %

if length(sunSENSORaz_BOD) ~= length(sunSENSORel_BOD),
    error('sunSENSOR setup ERROR');
end;
```

orbit.m

```
% ************************************************************************* %
% [SATsph_GEO, SATxyz_GEO, TrueAnomal] = Orbit(time, SATrpy_GEO)            %
%                                                                           %
%   This function takes the time input variable, and uses the globally     %
%   defined ORBIT variables to return the point along the orbit that        %
%   the sattelite currently is. It also defines the DCM needed to           %
%   convert vectors from orbital to celestial coordinates and back again.%
%                                                                           %
%   SATaz_GEO, SATel_GEO, SATalt  : current position of the SAT             %
%                                                                           %
%   v1.1:                                                                   %
%       * now takes SATrpy_GEO as input as well to determine all the        %
%       various DCM matrixes used in the rest of the program.               %
%       * Also returns the TrueAnomal angle                                 %
%                                                                           %
%   v1.2:                                                                   %
%       * Outputs the position vector in both AZ,EL,R and X,Y,Z to          %
%       prevent multiple conversions later on in the program.               %
%                                                                           %
% ************************************************************************* %
function [SATsph_GEO, SATxyz_GEO, TrueAnomal] = Orbit(time, SATrpy_GEO);

% ------------------------------------------------------------------------- %
% Imports                                                                   %
% ------------------------------------------------------------------------- %

global ORBITeccent ORBITsemimaj ORBITmean ORBITinclin ORBITraAN ORBITarg

% ------------------------------------------------------------------------- %
% Exports                                                                   %
% ------------------------------------------------------------------------- %

global DCMcel2orb
global DCMcel2bod
global DCMorb2bod

% ------------------------------------------------------------------------- %
% Constants                                                                 %
% ------------------------------------------------------------------------- %

a       = ORBITsemimaj;
e       = ORBITeccent;
n       = ORBITmean;
raan    = ORBITraAN;
inclin  = ORBITinclin;

% ------------------------------------------------------------------------- %
% Main                                                                      %
% ------------------------------------------------------------------------- %

MeanAnomal = n*time;
if e == 0,
    TrueAnomal = MeanAnomal;
    SATalt     = a;
else
    TrueAnomal = MeanAnomal + 2*e*sin(MeanAnomal) ...
        + 1.25*e^2*sin(2*MeanAnomal);
    SATalt     = a*(1-e^2) / (1 + e*cos(TrueAnomal));
```

```
end;

%convert the TrueAnomal to a position in the Celestial System, taking    %
%into account the orbit characteristics                                  %
psi   = -ORBITraAN;      Cs1 = cos(psi);   Ss1 = sin(psi);   %angle about z
phi   = pi-ORBITinclin; Cp1 = cos(phi);   Sp1 = sin(phi);   %angle about x
psi   = TrueAnomal + ORBITarg;
                        Cs2 = cos(psi);   Ss2 = sin(psi);   %angle about z
DCM = ...
    [ Cs1*Cs2-Ss1*Cp1*Ss2,  Cs1*Ss2+Ss1*Cp1*Cs2, Ss1*Sp1;
     -Ss1*Cs2-Cs1*Cp1*Ss2, -Ss1*Ss2+Cs1*Cp1*Cs2, Cs1*Sp1;
                 Sp1*Ss2,             -Sp1*Cs2,     Cp1]; % z1-x1-z2

SATxyz_GEO = DCM * [SATalt; 0; 0];
[SATaz_GEO, SATel_GEO, SATalt] = ...
    cart2sph(SATxyz_GEO(1), SATxyz_GEO(2), SATxyz_GEO(3));
SATsph_GEO = [SATaz_GEO, SATel_GEO, SATalt]';

%calculate the DCM that will be used in the rest of the program to       %
%convert vectors in the Celestial coordinate system to vectors in the    %
%Orbital coordinate system.                                              %
%NOTE: some of these were defined above and some are constants.  the     %
%   commented out parts are simply efficient, the code remains true.     %
%psi = -ORBITraAN;      Cs1 = cos(psi);   Ss1 = sin(psi);   %angle about z
%phi = pi-ORBITinclin; Cp1 = cos(phi);   Sp1 = sin(phi);   %angle about x
%phi = -90/180*pi;      Cp2 = cos(phi);   Sp2 = sin(phi);   %angle about x
psi = TrueAnomal + ORBITarg + pi/2;
                       Cs2 = cos(psi);   Ss2 = sin(psi);   %angle about z
DCMorb2cel = ...
    [ Cs1*Cs2-Ss1*Cp1*Ss2, Ss1*Sp1, -Cs1*Ss2-Ss1*Cp1*Cs2;
     -Ss1*Cs2-Cs1*Cp1*Ss2, Cs1*Sp1,  Ss1*Ss2-Cs1*Cp1*Cs2;
                 Sp1*Ss2,     Cp1,              Sp1*Cs2];  %z1-x1-z2-x2
DCMcel2orb = DCMorb2cel';

%calculate the DCM needed to convert Celestial vectors to Body vectors   %
%NOTE: i don't understand why this DCM needs to be z-y-x when it produces%
%a x-y-z rotation, but it works...                                       %
phi   = SATrpy_GEO(1); Cp = cos(phi);   Sp = sin(phi);      %angle about x
theta = SATrpy_GEO(2); Ct = cos(theta); St = sin(theta);    %angle about y
psi   = SATrpy_GEO(3); Cs = cos(psi);   Ss = sin(psi);      %angle about z
DCMcel2bod = ...
    [         Ct*Cs,           Ct*Ss,   -St;
     Sp*St*Cs-Cp*Ss, Sp*St*Ss+Cp*Cs, Sp*Ct;
     Cp*St*Cs+Sp*Ss, Cp*St*Ss-Sp*Cs, Cp*Ct];           %z-y-x

DCMorb2bod = DCMcel2bod * DCMcel2orb';
```

Appendix D

MATLAB$^{\text{TM}}$ Code: Attitude Control

main.m

```
clear all

global SAVE DISTURBANCE
DISTURBANCE = rand(3,3001)*2-1;

DATA = struct(...
        't',    0, ...
        'SAT', struct(...
                'q',          [0; 0; 0; 1], ...
                'q_next',     [0; 0; 0; 1], ...
                'w',          zeros(3,1), ...
                'w_next',     zeros(3,1), ...
                'w_dot',      zeros(3,1), ...
                'L',          zeros(3,1), ...
                'I',          eye(3,3) * 5, ...
                'invI',       zeros(3,3), ...
                'EUL123',     zeros(3,1), ...
                'EUL321',     zeros(3,1), ...
                'EUL312',     zeros(3,1), ...
                'EULER',      zeros(3,1) ...
            ), ...
        'RW', struct(...
                'gain_factor',        1, ...
                'gain_factor_list',  [1], ...
                'Alignment',          eye(3,3), ...
                'N_max',              ones(3,1) * 30e-3, ...
                'N_max_list',         [ones(3,1) * 30e-3], ...
                'h',                  zeros(3,1), ...
                'h_max',              ones(3,1) * 300e-3, ...
                'h_next',             zeros(3,1), ...
                'h_dot',              zeros(3,1) ...
            ), ...
        'MT', struct(...
                'gain_factor',        1, ...
                'gain_factor_list',  [1], ...
                'Alignment',          eye(3,3), ...
                'N_max',              ones(3,1) * 1e-3, ...
                'N_max_list',         [ones(3,1) * 1e-3] ...
            ), ...
        'CTRL', struct(...
                'ref',                    1*deg2rad([0 0 0]'), ...
                'dt',                     0.1, ...
                'Time_Delay',             0, ...
                'Time_Delay_Steps',       1, ...
                'N',                      {{ ...
                        zeros(3,1), ...
                    }}, ...
                'N_adaptive',             zeros(3,1), ...
                'N_adapt_init',           zeros(3,1), ...
                'N_classic',              zeros(3,1), ...
                'Nr',                     zeros(3,1), ...
                'Nm',                     zeros(3,1), ...
                'Nd',                     zeros(3,1), ...
                'Nd_max',                 ones(3,1) * 1e-3, ...
                'K',                      [ ...
                        0.05483113556161; ...
                        0.74036866869599; ...
                    ], ...
                'Nbar',                  [0.05483113556161], ...
                'Adaptive_Strategy',    zeros(3,1), ...
                'OptimalGrace',         3, ...
                'OptimalScale',         1, ...
                'Grade',                zeros(3,1), ...
                'near',                 zeros(3,1) ...
            ) ...
    );
STOR = struct(...
        'STATIC', struct(...
                'SAT', {{...
                        'I' ...
                    }}, ...
                'RW', {{...
                        'Alignment', ...
                        'h_max' ...
                    }}, ...
                'MT', {{...
                        'Alignment', ...
                    }}, ...
                'CTRL', {{...
                        'Nd_max', ...
                        'OptimalGrace', ...
                        'OptimalScale', ...
                        'Grade', ...
                        'near' ...
                    }} ...
            ), ...
        'DYNAMIC', struct(...
                't', [], ...
                'SAT', {{...
                        'q', ...
                        'w', ...
                        'EUL312', ...
                        'EULER' ...
                    }}, ...
                'RW', {{...
                        'gain_factor', ...
                        'N_max', ...
                        'h' ...
                    }}, ...
                'MT', {{...
                        'gain_factor' ...
                        'N_max' ...
                    }}, ...
                'CTRL', {{...
                        'N_adaptive', ...
                        'N_classic', ...
                        'Nr', ...
                        'Nm', ...
                        'Nd', ...
                        'Grade' ...
                    }} ...
            ) ...
    );

% -------------------------- %
% SIMULATION CONTROL OPTIONS %
% -------------------------- %
global SIMOPTION_FIGURE_INDEX SIMULATION_LIST_LENGTH
global SAVE
```

```
SIMOPTION_FIGURE_INDEX = 1;

% ------------------------------------------------------------------- %
% SETUP A LIST OF SIMULATION PARAMETERS TO SET DURING CONSECUTIVE RUNS %
% ------------------------------------------------------------------- %

SIMULATION_LIST = setup(1);

% -------------------------------------- %
% PARSE AND CORRECT SIMULATION PARAMETERS %
% -------------------------------------- %

SIMULATION_LIST_LENGTH = fieldlength(SIMULATION_LIST);
SIMULATION_LIST_FIELDS = fieldnames(SIMULATION_LIST);
SIMULATION_LIST = evenfields(SIMULATION_LIST, SIMULATION_LIST_LENGTH);

% -------------------------------- %
% INITIALIZE SIMULATION PARAMETERS %
% -------------------------------- %

for LOOP_INDEX = 1:SIMULATION_LIST_LENGTH,
    if LOOP_INDEX == 1, disp('Running the Simulation...'); end;
    disp([...
        'Pass ' num2str(LOOP_INDEX) '/' num2str(SIMULATION_LIST_LENGTH)...
    ]);

    for i = 1:size(SIMULATION_LIST_FIELDS, 1),
        FIELD = getfield(SIMULATION_LIST, SIMULATION_LIST_FIELDS{i});
        if isa(FIELD, 'struct'),
            FIELD = selectfield(FIELD, LOOP_INDEX);
            FIELD_FIELDS = fieldnames(FIELD);
            for m = 1:size(FIELD_FIELDS, 1),
                DATA.(SIMULATION_LIST_FIELDS{i}).(FIELD_FIELDS{m}) = ...
                        FIELD.(FIELD_FIELDS{m}) ...
                    ;
            end;
        else
            warning('Unknown Field Type');
            return;
        end;
    end;

    DATA.CTRL.N = {};
    DATA.CTRL.Time_Delay_Steps = ...
                        ceil(DATA.CTRL.Time_Delay / DATA.CTRL.dt+1);
    for i = 1:DATA.CTRL.Time_Delay_Steps,
        DATA.CTRL.N{i} = zeros(3,1);
    end;

    DATA.SAT.invI = inv(DATA.SAT.I);
    DATA.SAT.EUL312 = dcm2eul('312', qua2dcm(DATA.SAT.q_next));
    DATA.SAT.EULER = DATA.SAT.EUL312([2 3 1]);
    for i = 1:3,
        DATA.CTRL.near(i) = ( ...
                DATA.RW.N_max(i) + DATA.MT.N_max(i)
            ) / DATA.SAT.I(i,i) * DATA.CTRL.dt;
    end;

% ----------------------------------- %
% RECORD STATIC SIMULATION PARAMETERS %
% ---------------------------------- %
    SAVE_STRUCT = fieldnames(STOR.STATIC);
    for m = 1:length(SAVE_STRUCT),
        for n = 1:length(STOR.STATIC.(SAVE_STRUCT{m})),
            FIELD_NAME = STOR.STATIC.(SAVE_STRUCT{m}){n};
            SAVE.(SAVE_STRUCT{m}).(FIELD_NAME){LOOP_INDEX} = ...
                    DATA.(SAVE_STRUCT{m}).(FIELD_NAME);
        end;
    end;

    ew = 1;
    ew_max = max(abs(DATA.SAT.w_next)); ew_max2 = ew_max;
    ee = 1;
    ee_max = max(abs(DATA.SAT.EULER - DATA.CTRL.ref)); ee_max2 = ee_max;
    dt = 1; DATA.t = 0;
    i = 0;

    RW_gainchange = length(DATA.RW.gain_factor_list);
    MT_gainchange = length(DATA.MT.gain_factor_list);
    RW_clipchange = length(DATA.RW.N_max_list);
    MT_clipchange = length(DATA.MT.N_max_list);

                    % ------------------- %
                    % START THE MAIN LOOP %
                    % ------------------- %

    while (ew > 0.01 | ee > 0.01 | DATA.t < 50) & DATA.t < 300,
        DATA.t = DATA.t + DATA.CTRL.dt;
        i = i + 1;

    % ---------------------------- %
    % UPDATE SIMULATION PARAMETERS %
    % ---------------------------- %
    if (RW_gainchange ~= 1) | (MT_gainchange ~= 1),
        if RW_gainchange ~= 1,
            index = i;
            while index > RW_gainchange,
                index = index - RW_gainchange;
            end;
            DATA.RW.gain_factor = DATA.RW.gain_factor_list(index);
        end;
        if MT_gainchange ~= 1,
            index = i;
            while index > MT_gainchange,
                index = index - MT_gainchange;
            end;
            DATA.MT.gain_factor = DATA.MT.gain_factor_list(index);
        end;
    end;

    if (RW_clipchange ~= 1) | (MT_clipchange ~= 1),
        if RW_clipchange ~= 1,
            index = i;
            while index > RW_clipchange,
                index = index - RW_clipchange;
            end;
            DATA.RW.N_max = ones(3,1) * DATA.RW.N_max_list(index);
        end;
        if MT_clipchange ~= 1,
            index = i;
            while index > MT_clipchange,
                index = index - MT_clipchange;
            end;
            DATA.MT.N_max = ones(3,1) * DATA.MT.N_max_list(index);
        end;
    end;

    % -------------------------------------- %
    % RUN THE MAIN SIMULATION INCREMENT LOOP %
    % -------------------------------------- %
        [DATA] = simulate(DATA);
        if ~isreal(DATA.SAT.q), where = [LOOP_INDEX, DATA.t], end;

    % ------------------------------------ %
    % RECORD DYNAMIC SIMULATION PARAMETERS %
    % ------------------------------------ %
        SAVE_STRUCT = fieldnames(STOR.DYNAMIC);
        for m = 1:length(SAVE_STRUCT),
            if isa(STOR.DYNAMIC.(SAVE_STRUCT{m}), 'cell'),
                for n = 1:length(STOR.DYNAMIC.(SAVE_STRUCT{m})),
                   FIELD_NAME = STOR.DYNAMIC.(SAVE_STRUCT{m}){n};
                   SAVE.(SAVE_STRUCT{m}).(FIELD_NAME){i, LOOP_INDEX} = ...
                           DATA.(SAVE_STRUCT{m}).(FIELD_NAME);
                end;
            else
                SAVE.(SAVE_STRUCT{m}){i,LOOP_INDEX} = ...
                    DATA.(SAVE_STRUCT{m});
            end;
        end;

    % ------------- %
    % ERROR UPDATES %
    % ------------- %
        if ew_max == 0,
            ew_max2 = max([ew_max2 max(abs(DATA.SAT.w_next))]);
            if ew_max2 == 0,
                ew = max(abs(DATA.SAT.w_next));
            else
                ew = max(abs(DATA.SAT.w_next))/ew_max2;
            end;
        else
            ew = max(abs(DATA.SAT.w_next))/ew_max;
        end;
        if ee_max == 0,
            ee_max2 = max([ee_max2 max(abs(DATA.SAT.w_next))]);
            if ee_max2 == 0,
                ee = max(abs(DATA.SAT.w_next));
            else
                ee = max(abs(DATA.SAT.w_next))/ee_max2;
            end;
        else
            ee = max(abs(DATA.SAT.EULER - DATA.CTRL.ref))/ee_max;
        end;
    end;
end;

disp('Plotting the Results...');
plotme;
```

simulate.m

```
function [DATA] = simulate(DATA),

% ---------------------------------------------------------------------- %
% Defaults & Error check                                                 %
% ---------------------------------------------------------------------- %
    N = zeros(3,1);             %define shape of N

% ---------------------------------------------------------------------- %
% Constants                                                              %
% ---------------------------------------------------------------------- %

global DISTURBANCE

% ---------------------------------------------------------------------- %
% Main                                                                   %
% ---------------------------------------------------------------------- %

% ---------------- %
% VARIABLE UPDATES %
% ---------------- %

    DATA.RW.h = DATA.RW.h_next;
    DATA.SAT.w = DATA.SAT.w_next;
    DATA.SAT.q = DATA.SAT.q_next;

% --------------- %
% CONTROL UPDATES %
% --------------- %

    %determine euler angles from the quaternion representation
    DATA.SAT.EUL312 = dcm2eul('312', qua2dcm(DATA.SAT.q));
    DATA.SAT.EUL321 = dcm2eul('321', qua2dcm(DATA.SAT.q));
    DATA.SAT.EUL123 = dcm2eul('123', qua2dcm(DATA.SAT.q));
    DATA.SAT.EULER = DATA.SAT.EUL312([2 3 1]);

    %recenter the euler angles about the control DATA.CTRL.ref input
    for m = 1:3,
        while DATA.SAT.EULER(m) > DATA.CTRL.ref(m)+pi,
            DATA.SAT.EULER(m) = DATA.SAT.EULER(m) - 2*pi;
        end;
        while DATA.SAT.EULER(m) <= DATA.CTRL.ref(m)-pi,
            DATA.SAT.EULER(m) = DATA.SAT.EULER(m) + 2*pi;
        end;
    end;

% ------------------------ %
% DETERMINE CONTROL ACTION %
% ------------------------ %

    [DATA] = Control(DATA);
    index = floor(DATA.t./DATA.CTRL.dt);
    DATA.CTRL.Nd = diag(DATA.CTRL.Nd_max) * DISTURBANCE(:, index);

% ------------------- %
% EQUATIONS OF MOTION %
% ------------------- %

    DATA.RW.h_dot = DATA.CTRL.Nr;
    DATA.SAT.w_dot = DATA.SAT.invI * (...
            DATA.CTRL.Nd - cross(DATA.SAT.w, DATA.SAT.I*DATA.SAT.w) - ...
            [cross(DATA.SAT.w,DATA.RW.h) + DATA.CTRL.Nr + DATA.CTRL.Nm]...
        );
    DATA.SAT.L = DATA.SAT.I * DATA.SAT.w + DATA.RW.h;

% ----------- %
% INTEGRATION %
% ----------- %
    %integrate Reaction Wheel Angular Momentum...
    DATA.RW.h_next = DATA.RW.h + DATA.CTRL.dt*DATA.RW.h_dot;
    %integrate Sattelite Rates...
    DATA.SAT.w_next = DATA.SAT.w + DATA.CTRL.dt*DATA.SAT.w_dot;
    %integrate Sattelite position...
    wavr = (DATA.SAT.w + DATA.SAT.w_next)/2; wbar = sqrt(sum(wavr.^2));
    Omega = [...
         0        wavr(3) -wavr(2) wavr(1);
        -wavr(3)  0        wavr(1) wavr(2);
         wavr(2) -wavr(1)  0       wavr(3);
        -wavr(1) -wavr(2) -wavr(3) 0       ];
    %DATA.SAT.q_next = [eye(4) + 1/2 * Omega * DATA.CTRL.dt] * DATA.SAT.q;
    if wbar ~= 0,
        Omega = Omega/wbar;
        DATA.SAT.q_next = ( ...
                cos(wbar*DATA.CTRL.dt/2)*eye(4) ...
                + ...
                sin(wbar*DATA.CTRL.dt/2)*Omega ...
            ) * DATA.SAT.q;
    end;
    %end integration
```

control.m

```
function [DATA] = Control(DATA),

% ----------------------------------------------------------------------- %
% Defaults & Error check                                                  %
% ----------------------------------------------------------------------- %
    N = zeros(3,1);             %define shape of N

% ----------------------------------------------------------------------- %
% Main                                                                    %
% ----------------------------------------------------------------------- %

% ----------------------- %
% DETERMINE CONTROL ACTION %
% ----------------------- %
    for i = 1:3,
        if DATA.CTRL.Time_Delay_Steps > 1,
            [DATA.CTRL.N{1:DATA.CTRL.Time_Delay_Steps-1}] = ...
                deal(DATA.CTRL.N{2:DATA.CTRL.Time_Delay_Steps});
        end;

        EULER(i) = DATA.SAT.EULER(i) - DATA.CTRL.ref(i);

        %Classic Control Strategy
        DATA.CTRL.N_classic(i) = DATA.CTRL.K*[EULER(i); DATA.SAT.w(i)];

        DATA.CTRL.OptimalCurve(i) = DATA.SAT.I(i,i)/( ...
                2*(DATA.RW.N_max(i)+DATA.MT.N_max(i)) ...
            ) * DATA.SAT.w(i)^2;
        if ( ...
                (DATA.CTRL.Adaptive_Strategy ~= 0) ...
                & ( ...
                abs(EULER(i)) < (...
                        DATA.CTRL.OptimalCurve(i) * ...
                        DATA.CTRL.OptimalGrace) ...
                    ) ...
                & ( ...
                abs(EULER(i)) > (...
                        DATA.CTRL.OptimalCurve(i)) ...
                    ) ...
                & ...
                ( ...
                        ((EULER(i) > 0) & (DATA.SAT.w(i) < 0)) ...
                        | ...
                        ((EULER(i) < 0) & (DATA.SAT.w(i) > 0)) ...
                    ) ...
                & ...
                (abs(DATA.SAT.w(i)) > DATA.CTRL.near(i)) ...
            ),

            %Adaptive Control Strategies...
            DATA.CTRL.Grade(i) = (...
                    DATA.CTRL.OptimalCurve(i)*DATA.CTRL.OptimalGrace ...
                        - abs(EULER(i))...
                    ) / (...
                    DATA.CTRL.OptimalCurve(i)*DATA.CTRL.OptimalGrace - ...
                    DATA.CTRL.OptimalCurve(i)/DATA.CTRL.OptimalScale ...
                    ) ...
                ;
                TORQUE_RANGE = ( ...
                        DATA.RW.N_max(i) + DATA.MT.N_max(i) ...
                        + ...
                        abs(DATA.CTRL.N_adapt_init(i)) ...
                    );

                switch DATA.CTRL.Adaptive_Strategy
                case 1,
                   TORQUE_ADAPTION = DATA.CTRL.Grade(i);
                case 2,
                   k=3;
                   x = DATA.CTRL.Grade(i);
                   TORQUE_ADAPTION = 1/2*erf(k*(-log(1/k))^(1/2)*(2*x-1))+1/2;
                otherwise,
                   TORQUE_ADAPTION = 0;
                end;

                DATA.CTRL.N_adaptive(i) = ...
                       sign(DATA.SAT.w(i)) * TORQUE_RANGE * TORQUE_ADAPTION...
                       + ...
                       DATA.CTRL.N_adapt_init(i) ...
                    ;

                DATA.CTRL.N{DATA.CTRL.Time_Delay_Steps}(i) = ...
                    DATA.CTRL.N_adaptive(i);
            else
                DATA.CTRL.Grade(i) = 0;
                DATA.CTRL.N{DATA.CTRL.Time_Delay_Steps}(i) = ...
                    DATA.CTRL.N_classic(i);
                DATA.CTRL.N_adaptive(i) = 0;
                DATA.CTRL.N_adapt_init(i) = DATA.CTRL.N_classic(i);
            end;

  DATA.CTRL.Nr = DATA.RW.gain_factor * DATA.RW.Alignment * DATA.CTRL.N{1};
  DATA.CTRL.Nm = DATA.MT.gain_factor * DATA.MT.Alignment * DATA.CTRL.N{1};

% -------------------- %
% HARDWARE LIMIT TESTS %
% -------------------- %

    %Reaction Wheels maximum Torque limitation...
    index = find(...
            (DATA.CTRL.Nr > DATA.RW.N_max*DATA.RW.gain_factor) ...
                | ...
                (DATA.CTRL.Nr < -DATA.RW.N_max*DATA.RW.gain_factor)...
        );
    if index ~= 0,
        DATA.CTRL.Nr(index) = sign(DATA.CTRL.Nr(index)) .* ...
            DATA.RW.N_max(index)*DATA.RW.gain_factor;
    end;

    %Reaction Wheels maximum angular momentum (speed) limitation...
    h_next = DATA.RW.h + DATA.CTRL.dt*DATA.CTRL.Nr;
    index = find(...
            (h_next > DATA.RW.h_max) | (h_next < -DATA.RW.h_max)...
        );
    if index ~= 0,
        DATA.CTRL.Nr(index) = ( ...
                sign(h_next(index)).*DATA.RW.h_max(index) ...
                - ...
            DATA.RW.h(index) ...
        )/DATA.CTRL.dt;
end;

%Magneto Torquer maximum Torque limitation...
index = find(...
        (DATA.CTRL.Nm > DATA.MT.N_max) ...
        | ...
        (DATA.CTRL.Nm < -DATA.MT.N_max)...
    );
if index ~= 0,
    DATA.CTRL.Nm(index) = ...
        sign(DATA.CTRL.Nm(index)) .* DATA.MT.N_max(index);
end;
```