NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ATTITUDE DETERMINATION USING STAR TRACKER DATA WITH KALMAN FILTERS

by

Henry D. Travis

December 2001

Thesis Advisor:	Harold A. Titus
Second Reader:	Brij N. Agrawal

Approved for public release; distribution is unlimited

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
19 Dec 2001	N/A	-

Title and Subtitle Attitude Determination Using Star Tracker Data with Kalman Filters	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Travis, Henry	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Naval Postgraduate School Monterey, California	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes The original document contains color images.	
Abstract	
Subject Terms	
Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 69	

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Attitude Determination Using Star Tracker Data with Kalman Filters			**5. FUNDING NUMBERS**
6. AUTHOR(S) Henry D. Travis			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

ABSTRACT (maximum 200 words)

This study adapts some established attitude determination techniques for use with star tracker measurements on satellites. Other work in this area has utilized gyro measurements with star tracker updates. Today's star trackers are giving measurements with accuracies of less than 6 arcseconds, and are therefore of high enough fidelity to be used alone. Computer simulation of a Linear Kalman Filter to process these measurements is presented. The Filter uses a linear, constant coefficient state matrix with the Optimal Control Law to provide negative feedback control. The control law uses information developed through the equations of motion of the spacecraft in a Molnyia orbit. Modifications to the Filter, including glitch rejection and various covariance manipulation techniques are discussed as possible sources for performance enhancement.

14. SUBJECT TERMS Attitude Determination, Linearized Dynamics, Kalman Filters			**15. NUMBER OF PAGES** 55
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ATTITUDE DETERMINATION USING STAR TRACKER DATA WITH KALMAN FILTERS

Henry D. Travis
Lieutenant, United States Navy
B.S., Oregon State University, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2001

Author: _____
Henry D. Travis

Approved by: _____
Harold Titus, Thesis Advisor

_____
Brij N. Agrawal, Second Reader

_____
Jeffrey B. Knorr, Chairman
Electrical and Computer Engineering Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This study adapts some established attitude determination techniques for use with star tracker measurements on satellites. Other work in this area has utilized gyro measurements with star tracker updates. Today's star trackers are giving measurements with accuracies of less than 6 arcseconds, and are therefore of high enough fidelity to be used alone. Computer simulation of a Linear Kalman Filter to process these measurements is presented. The Filter uses a linear, constant coefficient state matrix with the Optimal Control Law to provide negative feedback control. The control law uses information developed through the equations of motion of the spacecraft in a Molnyia orbit. Modifications to the Filter, including glitch rejection and various covariance manipulation techniques are discussed as possible sources for performance enhancement.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge Don Kolve and his associates at The Boeing Company for their inspiration and enthusiasm. The month I spent working with them was a once-in-a-lifetime opportunity and I appreciate all that they taught me about spacecraft operations.

Most of all, I wish to offer a special, heartfelt thank you to my advisor, Professor Hal Titus, and my co-advisor, Professor Brij Agrawal. Their tireless dedication to the students at the Naval Postgraduate School has had a profound effect both on the lives of those who they have taught as well as the quality of academics at NPS. I have benefited tremendously from the Space Systems Engineering program, and will be forever grateful for their encouragement, wisdom and leadership.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Until the recent advances in star trackers, past attitude determination algorithms have almost always relied on rate gyros. Unfortunately, rate gyros deteriorate over time and the results degrade system performance dramatically, as evidenced by the complete replacement of the gyros onboard the Hubble Space Telescope. Years of research have gone into overcoming these limitations. The greatest advance thus far has been improved star trackers. Unlike gyros, the Euler Angles or quaternions from star tracker measurements do not have to be backed out of rate measurements, but rather can be obtained directly.

The objective of this thesis is to utilize the accuracy of these new star trackers to provide estimates of the spacecraft's attitude. What sets this algorithm apart is its simplicity. Because today's star trackers have excellent tracking capability, we are not concerned with lengthy star gaps in this paper. Therefore, unlike previous work in this area, we do not need to develop a highly detailed dynamic system model that would attempt to describe the non-linear behavior of the spacecraft. Instead, we will make some general assumptions that will allow us to use a linear model of the system. Naturally, this will lead to a filter that is less computational intensive than a non-linear model. And in so doing, we are attempting to achieve satisfactory results with less computing power.

For the baseline orbital model, a linear, constant coefficient system model is used to represent the orbit in the state-space, with all of the non-linearities of the orbital dynamics treated as control inputs. This will generate a matrix of reference Euler Angles for the entire orbit. Then for the attitude control itself, two more linear, constant coefficient system models are used within a Linear

Kalman Filter (LKF) - one for the roll and yaw of the spacecraft and a second for the pitch. Then by using the principles of the Optimal Control Law, the spacecraft's attitude is controlled in discrete steps through the entire orbit. Results of this new algorithm indicate that filtering star tracker measurements provides a simple yet effective means of determining and controlling spacecraft attitude.

Most significantly, modifications to the Linear Kalman Filter (LKF) may reduce the steady-state errors even further. Three modifications will be discussed: 1) Glitch/bias rejection, 2) Covariance Manipulation, and 3) Rate Gyro measurements. The first method attempts to preemptively discard erroneous data prior to it entering the filter and being processed. Secondly, a time-varying covariance matrix can be utilized to account for specific conditions expected to occur during the spacecraft's flight. Lastly, additional measurements can be sent to the LKF for processing. Each modification has the potential for improving the LKF.

B. THESIS OUTLINE

The remainder of this thesis is organized as follows: Chapter II will discuss the astrodynamics of a Molnyia orbit and work through the equations of motion that will govern the satellite's behavior. Chapter III will detail the Linear Kalman Filter, including a review of the equations that the filter employs to estimate states. Chapter IV will showcase the various simulations and the resulting filter performance. Chapter V will focus on the conclusions drawn from the simulations and present ideas for future work in this area.

II. ASTRODYNAMICS

A. EQUATIONS OF MOTION

To begin, a review of the dynamics at work on the spacecraft in the orbit reference frame is shown. In this case, the satellite is traveling along a Molnyia orbit in the plane described by the radius and velocity vectors. The spacecraft's position in the orbit is defined by its distance in kilometers from the center of the earth, r, and the true anomaly, ν, which is measured from perigee in radians. Using the basic equations of motion,

$$F_r = m(\ddot{r} - r\dot{n}^2) \tag{1}$$

$$F_n = m(r\ddot{n} + 2\dot{r}\dot{n}) \tag{2}$$

the forces in both the radial and tangential direction can be determined [1]. Since the mass of the spacecraft is considered a point mass, the force per unit mass can be written as

$$F_r / m = -m / r^2 \tag{3}$$

And because of the nature of a two-body problem, the only force acting on the point mass is in the radial direction, the angular force, F, is zero. This leaves:

$$\ddot{r} = r\dot{n}^2 - m / r^2 \tag{4}$$

$$\ddot{n} = -2\dot{r}\dot{n} / r \tag{5}$$

B. SYSTEM MODEL

We first model the system in the form

$$\dot{x} = Ax + Bu \tag{6}$$

$$y = Cx + Du \tag{7}$$

by defining the state vector, x, as:

$$x \equiv [\, r \; \dot{r} \; n \; \dot{n} \,]^T \;, \tag{8}$$

and the control vector, u, from (4) and (5) as:

$$u \equiv \begin{bmatrix} r\dot{n}^2 - m/r^2 \\ -2\dot{r}\dot{n}/r \end{bmatrix} \tag{9}$$

With all non-linearities included in the control laws, the system coefficients can be easily written as:

$$A_{orbit} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

$$B_{orbit} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{11}$$

$$C_{orbit} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}$$

$$D_{orbit} = \begin{bmatrix} 0 \end{bmatrix} \tag{13}$$

With the linear system coefficient matrices defined, the next step is to compute the state transition matrix, Φ, and the convolution matrix, Δ, using the 'c2d' function in MATLAB. Thus the entire Molnyia orbit can be described using the discrete equation

$$x_{k+1} = \Phi x_k + \Delta u_k \tag{14}$$

This orbital information is stored for future reference with the pitch controller.

III. THE DISCRETE KALMAN FILTER

A. DEFINITIONS

The Kalman Filter is a recursive algorithm for estimating a state vector given past estimates and current measurements with noise. With a system model of the plant dynamics and sensor noise, the filter will minimize the mean square error. Since it was first development in 1960 by R.E. Kalman, the filter has been used in numerous fields of study and many sources exist that walk through the derivation of his work. For simplicity, only the resulting equations will be shown here. Following modern convention, the following definitions and notation will be used:

$\hat{x}_{k|k-1} \equiv$ state vector estimate at time k given measurements up to time k-1

$\hat{x}_{k|k} \equiv$ state vector estimate at time k given measurements up to time k

$e = x - \hat{x}_{k|k} \equiv$ state estimate error

$P_{k|k-1} \equiv$ prediction of the covariance of the state vector

$P_{k|k} \equiv$ update of the covariance of the state vector

$G \equiv$ Kalman gain

$Q \equiv$ Plant covariance matrix

$R \equiv$ Measurement covariance matrix

$\Phi \equiv$ state transition matrix

$H \equiv$ observation matrix

$w \equiv$ zero mean white Gaussian plant noise

$v \equiv$ zero mean white Gaussian measurement noise

$z \equiv$ noisy measurement

B. SYSTEM MODEL

1. Dynamics Equations

Before we can build the filter, a system must be developed that will adequately describe the behavior of the spacecraft. For simplicity, we have linearized the attitude dynamics equations of motion and used the small angle approximation. Looking at the equations of angular velocity in a rotating frame with a $y \rightarrow q \rightarrow f$ transformation,

$$w_{BI} = w_{BR} + w_{RIB} \text{ , where} \tag{15}$$

$$w_{BR} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{f} - \dot{y}\sin(q) \\ \dot{q}\cos(f) + \dot{y}\cos(q)\sin(f) \\ \dot{y}\cos(f)\cos(q) - \dot{q}\sin(f) \end{bmatrix} \tag{16}$$

which for small angles becomes

$$w_{BR} = \begin{bmatrix} \dot{f} \\ \dot{q} \\ \dot{y} \end{bmatrix} \text{, and} \tag{17}$$

$$w_{RIB} = \begin{bmatrix} 1 & y & -q \\ -y & 1 & f \\ q & -f & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -w_o \\ 0 \end{bmatrix} = \begin{bmatrix} -y w_o \\ -w_o \\ f w_o \end{bmatrix} \text{, therefore} \tag{18}$$

$$w_{BI} = \begin{bmatrix} \dot{f} - y w_o \\ \dot{q} - w_o \\ \dot{y} + f w_o \end{bmatrix} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{19}$$

Thus, the time rate of change of ω follows as:

$$\begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = \begin{bmatrix} \ddot{f} - \dot{y} w_o \\ \ddot{q} - w_o \\ \ddot{y} + \dot{f} w_o \end{bmatrix} \tag{20}$$

2. State Space Representation

Using the equations developed in the last section, we further assume that the second time derivatives of the angles are small enough to be ignored. This gives the following linear, constant coefficient matrices for the first system, which deals with only roll and yaw:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -w \\ 0 & 0 & 0 & 1 \\ 0 & w & 0 & 0 \end{bmatrix} \tag{21}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{22}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{23}$$

$$D = [0] \tag{24}$$

The final step is to compute the state transition matrix, Φ and the convolution integral, Δ using the MATLAB command,

$$[\ \Phi\ ,\ \Delta\ ] = \text{c2d (A,B,dt);} \tag{25}$$

which are used to promulgate the state vector using the equation:

$$x_{k+1} = \Phi x_k + \Delta u_k \tag{26}$$

3. Controllability

This linear time-invariant system is considered controllable if an input, u(t) will transfer the initial state of the system $\mathbf{x}(0)$ to the origin, $\mathbf{x}(t_f)=0$ with t_f finite [2]. Setting the state equation from the previous section to zero, it can be

shown that the system is controllable if it satisfies the condition that the controllability matrix

$$C_m = [B \quad AB] \tag{27}$$

has an inverse. Computing C_m for our system, it is seen that this condition is satisfied.

4. Control Gains

Two methods are presented for computing the control gain. The first is the pole placement method, and the second is a Linear Quadratic Regulator method, or LQR.

a. Pole Placement

This method has been relied on for decades as a means of ensuring the poles of the closed-loop system are at desirable locations. Ackermann developed a procedure for computing the control gain and for single input systems this algorithm is performed using the 'acker' command in MATLAB. However, for a multiple input case, the MATLAB 'place' command is used instead. The inputs for the place command are the state transition matrix, the convolution matrix, and the desired eigenvalues.

b. LQR

Using a Linear-quadratic regulator design for discrete-time systems is another method of computing the control gain. The gains from this method are considered optimal since the state-feedback law u[n] = -kx[n] minimizes the cost function

$$J = \sum \left(x'Qx + u'Ru + 2x'Nu \right) \tag{28}$$

subject to the state dynamics

$$x[n+1] = \Phi x[n] + \Delta u[n]. \tag{29}$$

The matrix N represents a relationship between the system noise, Q, and the measurement noise, R, and is set to zero for our system. Also returned are the Riccati equation solution and the closed-loop eigenvalues.

C. FILTER MODEL

We begin developing the Kalman Filter by modeling the random process:

$$x_{k+1} = \Phi_k x_k + w_k \tag{30}$$

The process is observed at discrete points in time by the following relation:

$$z_k = H_k x_k + v_k \tag{31}$$

As defined earlier, the covariance matrices Q and R are given by:

$$E[w_k w_i^T] = Q_k \tag{32}$$

$$E[v_k v_i^T] = R_k \tag{33}$$

And the plant covariance is derived from the errors in position. Assuming a constant acceleration over one time step, dt, using Newton's force equations,

$$x = x_o + at^2/2 \tag{34}$$

$$\dot{x} = \dot{x}_o + at \tag{35}$$

results in an error of $[dt\,2/2 \quad dt]$, which is squared and then entered into the covariance matrix:

$$Q = q^2 \times \begin{bmatrix} dt^4/4 & dt^3/2 & 0 & 0 \\ dt^3/2 & dt^2 & 0 & 0 \\ 0 & 0 & dt^4/4 & dt^3/2 \\ 0 & 0 & dt^3/2 & dt^2 \end{bmatrix} \tag{36}$$

and the measurement covariance is the square of the star tracker error:

$$R = \begin{bmatrix} ste^2 & 0 \\ 0 & ste^2 \end{bmatrix} \tag{37}$$

Since pitch is decoupled from roll and yaw, we omit pitch and pitch rate from the first state vector. The state vector, x, is then :

$$x \equiv [f \; \dot{f} \; y \; \dot{y}] \tag{38}$$

A second filter is therefore necessary for pitch and the corresponding state vector is defined here as:

$$x_p \equiv [q \; \dot{q}] \tag{39}$$

Similarly, the plant covariance follows as

$$Q_p = q_p^2 \times \begin{bmatrix} dt^4/4 & dt^3/2 \\ dt^3/2 & dt^2 \end{bmatrix} \tag{40}$$

and the measurement covariance is reduced to:

$$R = \lfloor ste^2 \rfloor \tag{41}$$

D. ALGORITHM

1. Kalman Equations

The filter itself is a two step process, a prediction followed by an update. In this simulation, a single measurement is used as the initial prediction. The code therefore first computes the Kalman Gain with the initial covariance prediction before updating the covariance prediction and then making a new prediction.

$$G = P_{k|k-1} H^T (H P_{k|k-1} H^T + R)^{-1} \tag{42}$$

$$P_{k|k} = (I - GH) P_{k|k-1} \tag{43}$$

$$P_{k|k-1} = \Phi P_{k|k} \Phi^T + Q \tag{44}$$

Similarly, the initial state vector is first updated with the new Kalman Gain before a new prediction of the state vector is made based on the measurement residual.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + G(z - z_{k|k-1}) \tag{45}$$

The control, u, is then determined using the Optimal Control Law

$$u = -k(x - \hat{x}_{k|k-1}) \tag{46}$$

before updating the state vector and predicting the next state estimate using the state transition matrix, Φ, and convolution integral, Δ.

$$x_{k+1} = \Phi x + \Delta u \tag{47}$$

$$\hat{x}_{k+1|k} = \Phi \hat{x}_{k|k} + \Delta u \tag{48}$$

The same procedure is followed to update and predict the pitch and pitch rate estimates with the new state vector, x_p.

2. Initialization Methods

There are several ways to initialize the Kalman Filter, including using an assumed state, a measurement, or a batch-processed state. An assumed state would be used either when the state is generally known without measurement or when the error is permitted to be large because there is sufficient time for the larger transient. A measurement approach is used when the error is expected to be small to begin with. A batch processed method would be used when the dynamics of the system would cause large changes from one measurement to the next. The expected changes will dictate the number of measurement processed to initialize the filter. Due to the accuracy of the star trackers and assumed low rate of change of the Euler Angles, the Kalman Filter could use the first measurement of the star tracker to initialize the filter. To measure the responsiveness of our filter, we will assume a zero state initially.

E. MODIFICATIONS

The Kalman Filter does an excellent job of seeing through the noise to provide reliable state estimates. By taking a closer look at the filter, we see that

there are some variables that can be adjusted. Most of these adjustments will involve the system covariance matrix.

1. Covariance Manipulation

The system covariance is an assumption of how much noise exists in the system. A small covariance correlates to a small amount of noise, and will naturally result in a better estimate. However, if the system is subjected to a large noise, such as from an unexpected noise source, the filter will be unable to track the transient and the estimate will deteriorate. It should be evident then that the desired covariance would be the smallest possible while still tracking any expected transient. The covariance manipulation methods examined here attempt to achieve that end.

a. Reset Threshold

The first method of manipulating the covariance is to test the estimate error, also called the residual bias, against some predefined threshold. Should the error exceed that threshold, the covariance matrix would be reset to a higher value. This method is a safety net to the Kalman Filter and some form of it will exist in each of the following methods.

b. Discretized Residual Bias

The second method is a variation of the first. Instead of having a single threshold, the residual bias is compared against multiple discrete levels that are defined with corresponding covariance matrices.

c. Alpha-Beta

This method is actually a function of the residual bias and returns a covariance matrix. The alpha-beta filter function is a sum of two parts, each

with a weighting coefficient. Clearly this is a more computationally intensive approach, and may not be appropriate for legacy systems that are severely limited on system resources.

d. Star Gap Response

This is another variation of the reset threshold. The difference is that there is no star tracker measurement available with which to compute a residual bias. This essentially blinds the Kalman Filter and the quality of the state estimate will depend on the accuracy of the dynamic model.

e. Glitch/Bias Rejection

This is a variation of the star gap response. In this case there is a star tracker measurement, but it is outside the expected range and is thus discarded. Subsequent measurments will also be discarded until they return to a predetermined error range. The reset threshold covariance matrix may also be used concurrently should the error grow too large while waiting for a valid measurement.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SIMULATION

A. OVERVIEW

The attitude determination algorithm was tested using MATLAB, and the code is included in Appendix A. Special attention will be given to the covariance matrices, control gains, eigenvalues, filter parameters, and residual bias. Throughout the following simulations, improvements to the Kalman Filter will be documented and the residual bias will be plotted to show the smaller error in the state estimate.

B. INITIALIZATION

1. Parameters

The code begins by defining some well known constants, and then establishing the number of time steps and the duration of each time step. In each of the cases presented, the simulation will run for 600 seconds with measurements every second. This is only a fraction of the orbital period, but is enough to evaluate how well the filter is following changes in trajectory.

The covariance matrices are initialized next in the code. The system covariance matrix is defined as derived in the previous chapter, with a scalar proportionality constant to modify how much error is assumed to exist in the system model. The observation covariance is simply the identity matrix multiplied by the square of the star tracker error. The error covariance matrix is set to the identity matrix.

Many of the variables used later in the code must be initialized to zero before being used for the first time. There are also several dummy variables that are created for use in storing data for plotting and comparison and would be omitted in a fielded version. In fact, a fielded version of this algorithm would most certainly be converted to the C programming language.

2. Molnyia Orbit

The next section of code defines the Molnyia orbit to provide a look up table of pitch angles and angular rates throughout the orbit. This clever algorithm allows the non-linear elements of the orbital dynamics to be included in the control, which keeps the state transition matrix linear and the subsequent computational costs to a minimum. The resulting orbital parameters are then converted to rectangular co-ordinates for plotting purposes.

3. Profile Loading

This one line of code retrieves an entire trajectory from a separate MATLAB function. Since we wish to test the filter against a variety of orbital trajectories, multiple profile functions were written. Each defines the truth states and corresponding star tracker measurements for the entire length of the simulation.

The first case is a steady state trajectory. The satellite is assumed by the filter to be at the zero state (all Euler Angles equal to zero), but the profile will define truth to be near the zero state at some fixed attitude. This code is titled, 'pss.m'.

The second case is a step trajectory. The satellite is assumed by the filter to be at the same state as truth, such as the zero state. At some predefined point, the truth will be stepped to a new value. Variations of this code include multiple steps. This code is titled, 'pstep.m'.

The third case is a sinusoidal trajectory. The satellite is assumed by the filter to be at the zero state, but a sinusoidal motion is added by the truth trajectory to model an oscillatory motion of the spacecraft. This code is titled, 'psin.m'.

C. SYSTEM SETUP

1. System Model

The system model, as developed in the previous chapter, is inserted here in the code. As shown again here, one interesting aspect of the system matrix is the time-varying angular frequency.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -w \\ 0 & 0 & 0 & 1 \\ 0 & w & 0 & 0 \end{bmatrix} \tag{49}$$

Since we are using a Molnyia orbit, the angular frequency is always changing. Therefore, to model the system more accurately, we would need to update this matrix at every time step before it could be used in other calculations. However, as stated in the introduction, our goal in this thesis is simplicity. Thus, one alternative would be to update this matrix less often than every time step, or go even further and assume a constant system matrix. Naturally, one of the primary results of interest will be the effect of using different system matrices.

2. Controllability

The next four lines of code quickly calculate the determinant of the Controllability and Observability matrices to check for singularity. Since both matrices have non-zero determinants, the system is both controllable and observable. However, should the state matrix change, this condition would need to verified again.

3. State Transition Matrix

To discretize our system, we input the continuous-time coefficients into the MATLAB command ‘c2d’ with the given time step, and we then have a discrete state transition matrix and convolution matrix for our system. As mentioned previously, if the state matrix changes, this computation would need to be repeated.

4. Eigenvalues and Control Gains

The ‘dlqr’ command returns the optimal eigenvalues and control gains for a given system matrix and covariance matrices. Those eigenvalues could be used in the ‘place’ command to obtain the same control gains. However, the ‘place’ command gives us additional flexibility to account for the non-linearities that we omitted when we simplified the system dynamics. We start by using the eigenvalues returned by ‘dlqr’, and make slight changes to see if better results can be obtained with more “realistic” control gains.

D. KALMAN FILTER

It is important to note once more that there are two separate Kalman Filters used in the code. This first estimates the roll and yaw of the spacecraft, while the second estimates the pitch. In the original code, the filter estimates of both states and measurements are set to zero, with the singular exception of the angular frequency. Then the filter loop begins by incrementing the time variable, calculating the Kalman Gain, updating the error covariance and predicting the error covariance for the next time step.

Here the truth states and truth measurements are promulgated using the state transition matrix and the convolution matrix. The observation error is calculated and stored for future plotting before being used to update the state estimate.

The control law used for roll and yaw is different than the control law used for pitch. Since no external disturbances are assumed for roll and yaw, the control law is the optimal control law,

$$u = -k(\hat{x} - x) \tag{50}$$

and this control effort is used to predict the state estimate. The new state estimate is then used to update the measurement estimate. The control law for the pitch filter is different because of the expected external disturbance due to the

Coriolis acceleration. Aside from that, the pitch filter performs the same algorithm as described above.

E. RESULTS

Even in a simplified piece of code as the one we have developed, there are several areas that can be worked with in the future to improve performance. However, before improvements are attempted, we need to develop a better understanding of the code and the effects of changing various parameters. The first part of the simulation establishes the orbit and stores the desired trajectory for future use. The Molnyia orbit pictured here covers one 12 hour orbit.

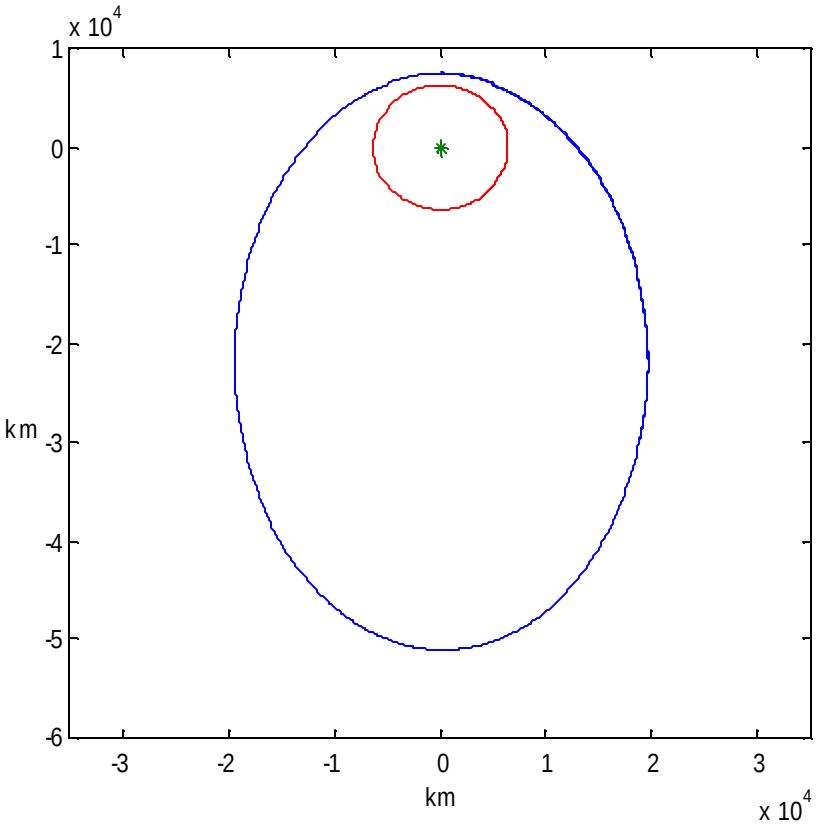

Figure 1. Molnyia Orbit.

The red circle represents the Earth and is shown for scale. The expected pitch angle during the orbit is stored in the 'xo' variable and is used to obtain a pitch estimate in the second Kalman Filter.

The next step in evaluating the algorithm is to look at the steady state estimates. Since the steady state performance of the filter is within the given six arcsecond accuracy of the star tracker, the focus of this thesis will be on the transient. A quick simulation shows filter convergence on the desired steady state roll and yaw angles within 60 seconds.

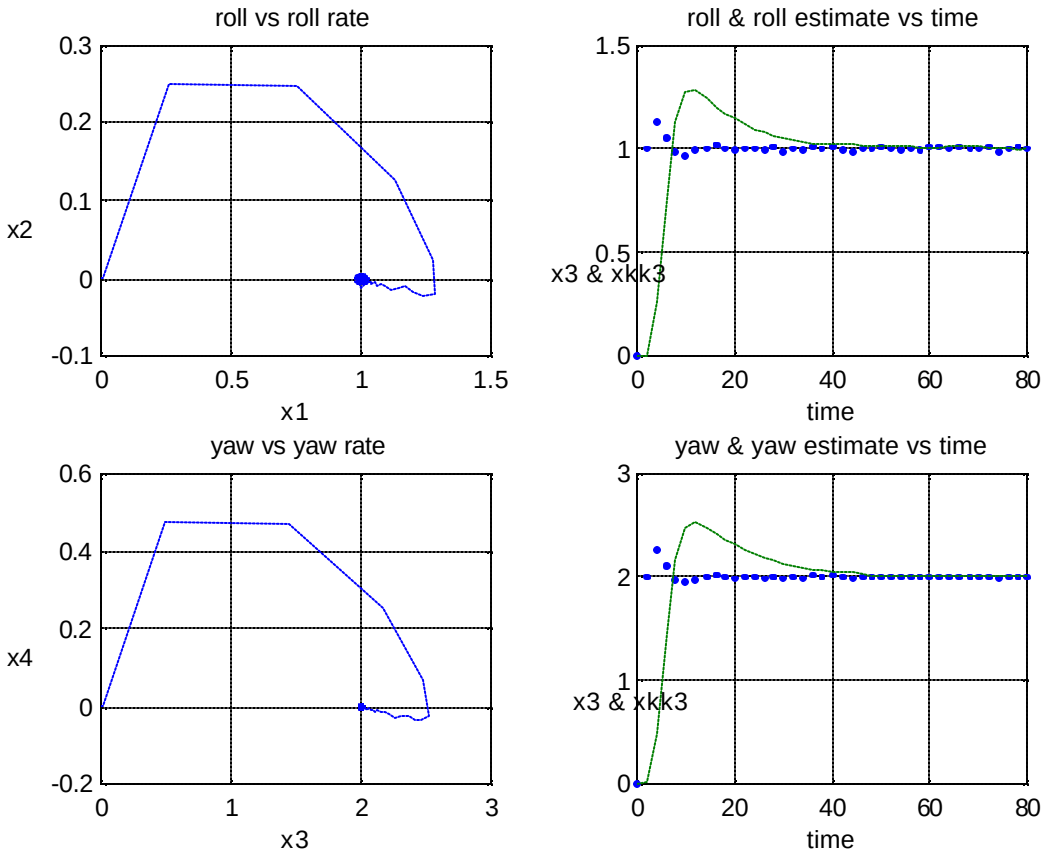

Figure 2. Roll and Yaw.

Thus, the simulation time will be reduced to the first 600 seconds, and even that will allow enough time to affect several transients in each run. The rest of this chapter will examine the effects of changes in the System Covariance matrix, the System matrix, Error Covariance matrix, eigenvalues, time steps, and the Control Law.

1. System Covariance Effects

The System Covariance matrix defined in Chapter III has a scalar, q, which can be modified easily. A smaller q represents less error in the system, but if the actual error is too great, the filter will not be able to acquire the star tracker signal. With $q = 10^{-6}$, the filter has no chance of acquiring the desired signal.

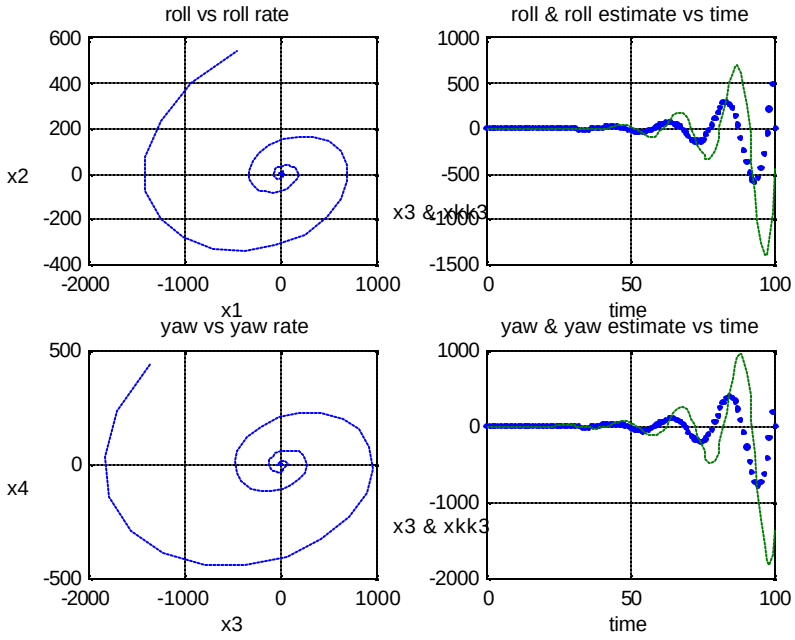

Figure 3. Roll and Yaw ($q = 10^{-6}$).

At $q2 = 10^{-4}$, a large overshoot is observed, but the signal is at least acquired. The phase plots on the left of the figure show the inward spiral that is characteristics of having a stable focus for the system [3].

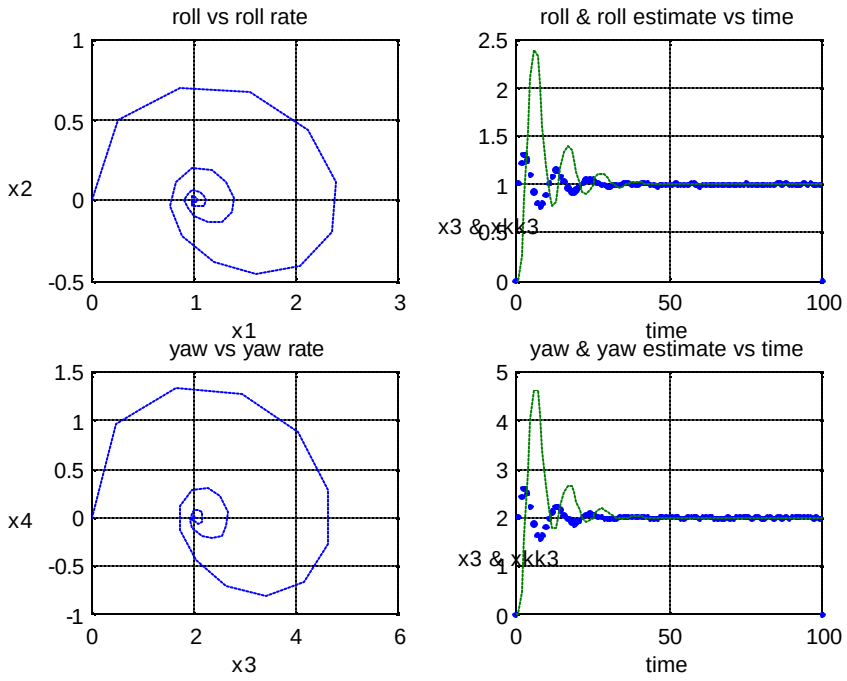

Figure 4. Roll and Yaw ($q = 10^{-4}$).

With $q2 = 10^{-2}$, the signal is acquired much more rapidly. Instead of the spiral of Figure 4, it takes two time steps before the estimate has reached the observation.

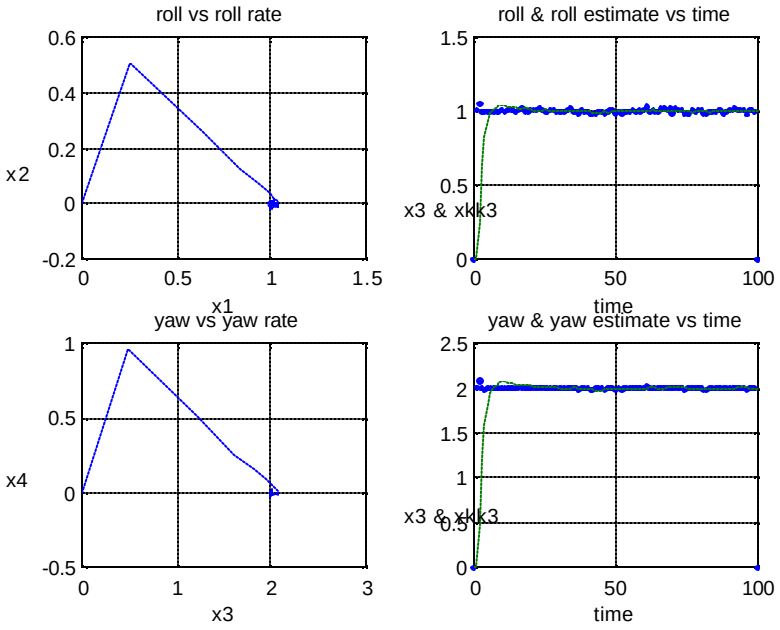

Figure 5. Roll and Yaw ($q = 10^{-2}$).

Further increases in q become counter productive. While the phase plots in Figure 6 are of similar quality, the time response in roll and yaw is degraded.

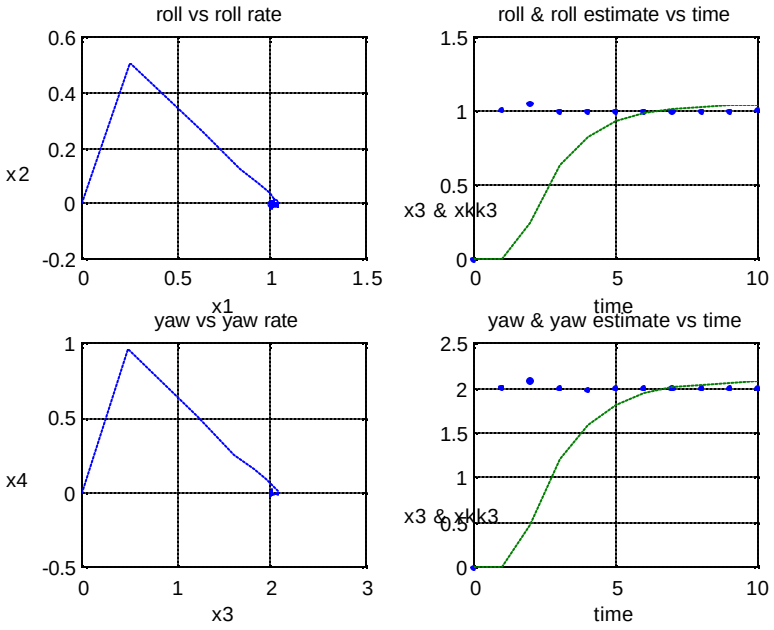

Figure 6. Roll and Yaw ($q = 10^{-2}$ zoom in).

Continuing to arbitrarily enlarge q2 to 1 or greater introduces overshoot again, but still acquisition. This is shown clearly when $q2 = 10^{6}$. The overshoot introduces a chattering in the phase plot.

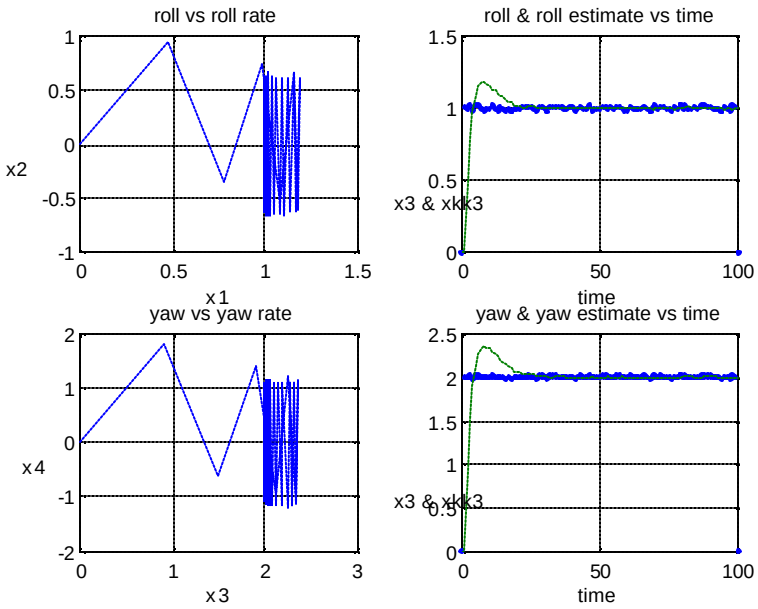

Figure 7. Roll and Yaw ($q = 10^6$).

2. System Matrix Effects

One of the ways to attempt to add realism would be to update the system matrix with the current angular frequency, ω. After all, this is not a constant coefficient as we have thus far assumed. From our early development of the orbit, we have stored the values of angular frequency for every step of the orbit. Adding the following three lines of code into the roll/yaw filter was a natural progression.

```
w = xo(4,i);                                % update angular frequency
A=[0 1 0 0;0 0 0 -w;0 0 0 1;0 w 0 0];  % update state matrix
[Phi,Del]=c2d(A,B,dt);                      % compute new transition matrix
```

However, as the next plot shows, these extra lines of code do not offer any improvement in performance, only computational cost. With q2 = .01, Figure 8 is indistinguishable from the constant coefficient version in Figure 5.

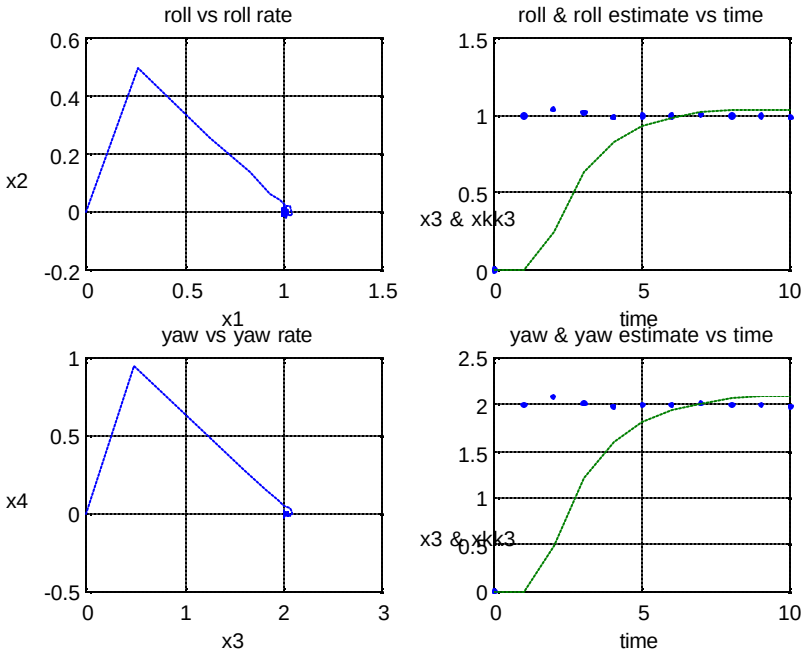

Figure 8. Roll and Yaw ($q = 10^{-2}$).

A similar approach can be taken with the pitch filter. However, running several cases shows that the best results come with a much lower q than used in the roll filter.

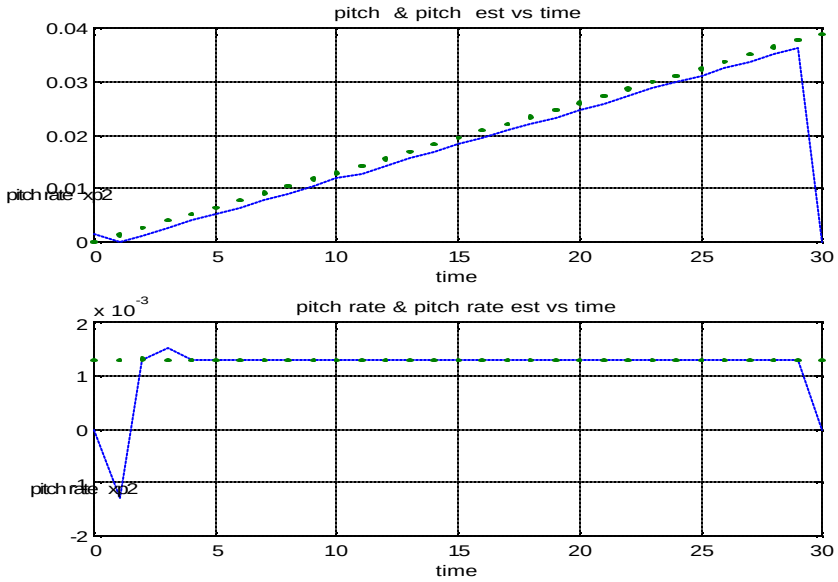

Figure 9. Pitch and Pitch Rate.

3. Error Effects

The Error Covariance matrix, P, is also subject to change. However, only when increasing or decreasing by tens of orders of magnitudes can one notice any impact. Such is the case in Figure 10, after P has been arbitrarily raised to 5^{20}.

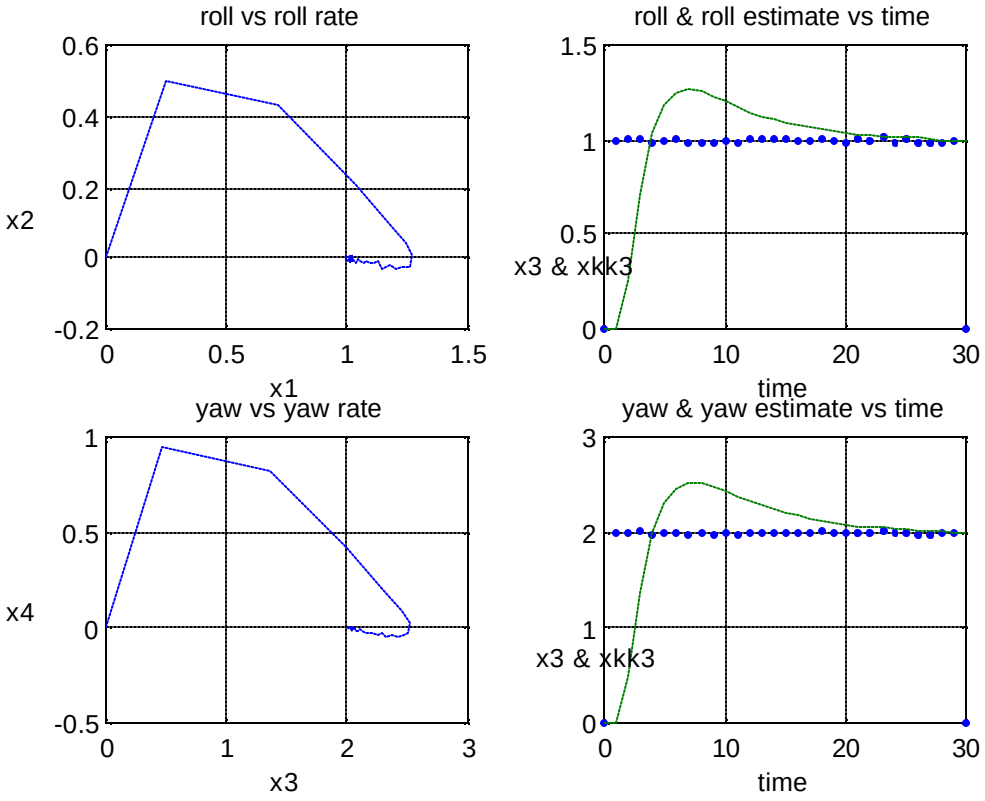

Figure 10. Roll and Yaw ($P = 5^{20}$).

4. Eigenvalue Effects

The simulations run so far have employed the 'place' command in MATLAB. Substituting the 'dlqr' command will return eigenvalues that are optimal for the give system matrix. With the same q = .01, the new Control Gains result in a distinct overshoot. Even after experimenting with new values for the System Covariance matrix, the overshoot can only be mitigated, not removed.

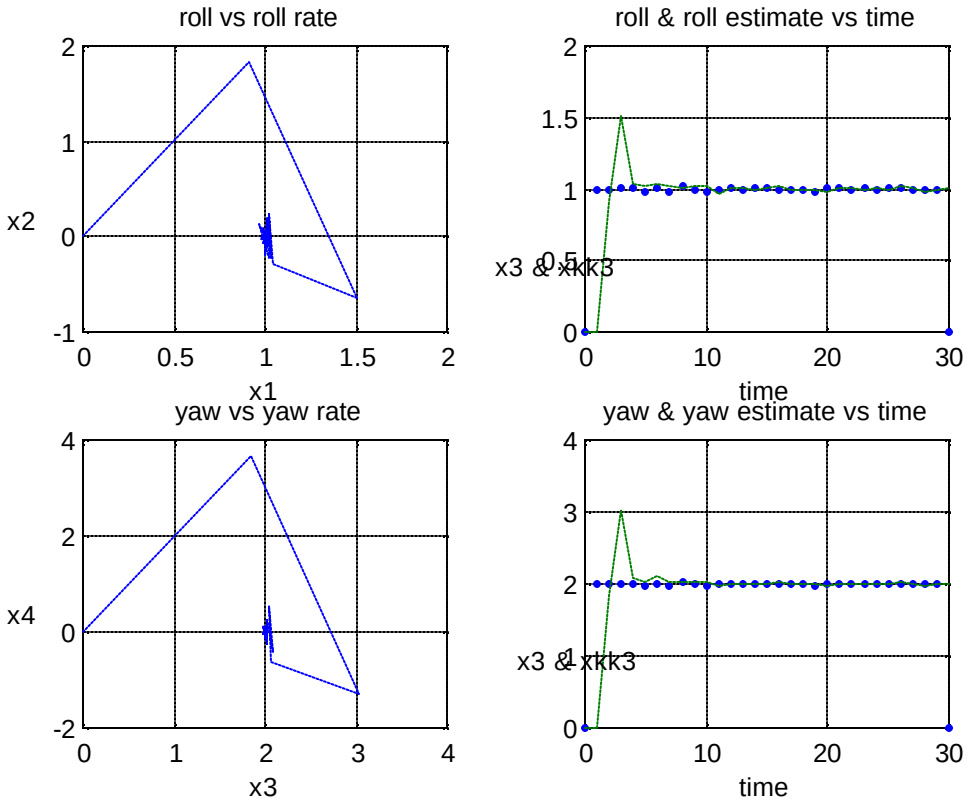

Figure 11. Discrete Linear Quadratic Regulator.

5. Time Step Effects

Changing the time step length changes the System Covariance matrix by definition. However, when running the simulation with varying time steps, the results were consistent. As seen in Figures 12 and 13, even after changing the time step by an order of magnitude, the filter still requires the same number, in these cases about 5, of time steps to settle to its steady state value.

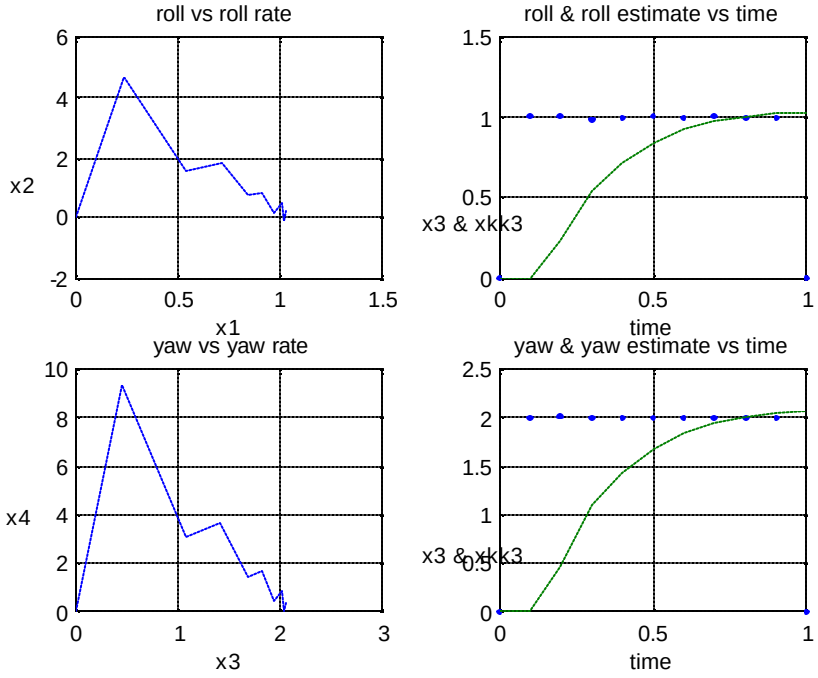

Figure 12. Roll and Yaw (dt = .1).

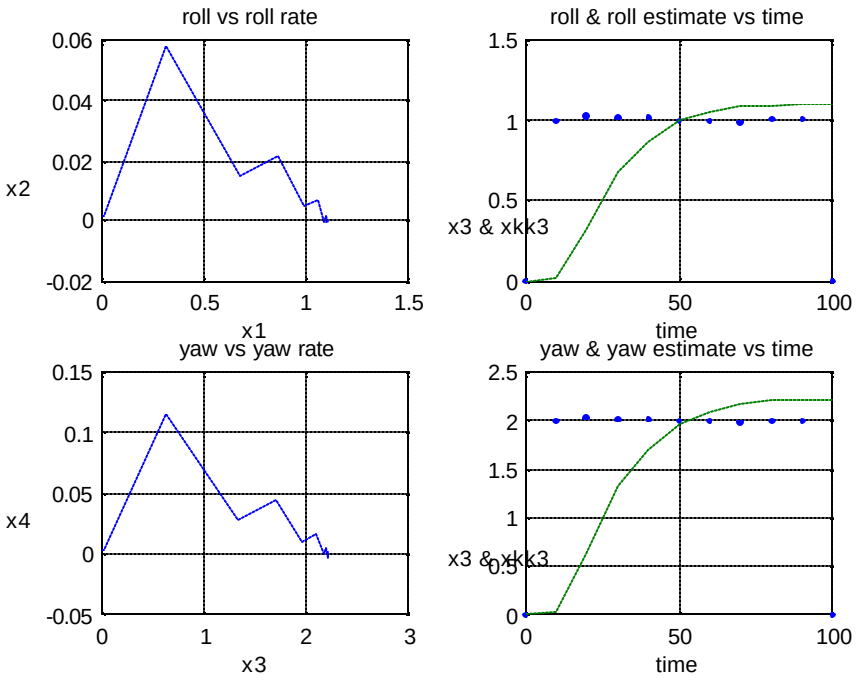

Figure 13. Roll and Yaw (dt = 10).

6. New Trajectories

All previous simulations have involved a constant roll and yaw angle trajectory. The next figures will show the filter's response to other trajectories. First with a discrete step from one constant value to another, and then with a ramp trajectory where the roll and yaw angles are incremented at each time step. As expected, the discrete steps are similar to the original case.

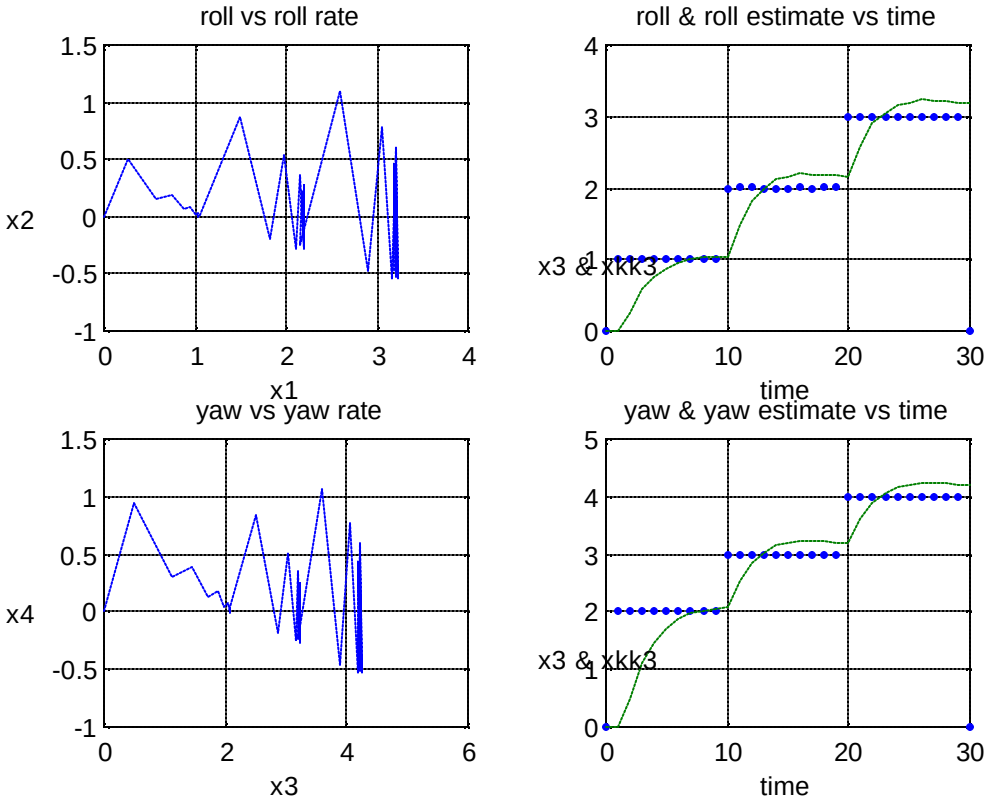

Figure 14. Discrete Step Trajectory.

The Ramp Trajectory shown in Figure 15 indicates that it takes longer for the filter to catch up with the trajectory when it is non-constant. The filter takes approximately twice as long to reach steady state, but under the circumstances this seems reasonable. Even with the exponential trajectory shown in Figure 16, the filter needs only 10 time steps to acquire and track the roll and yaw angles.

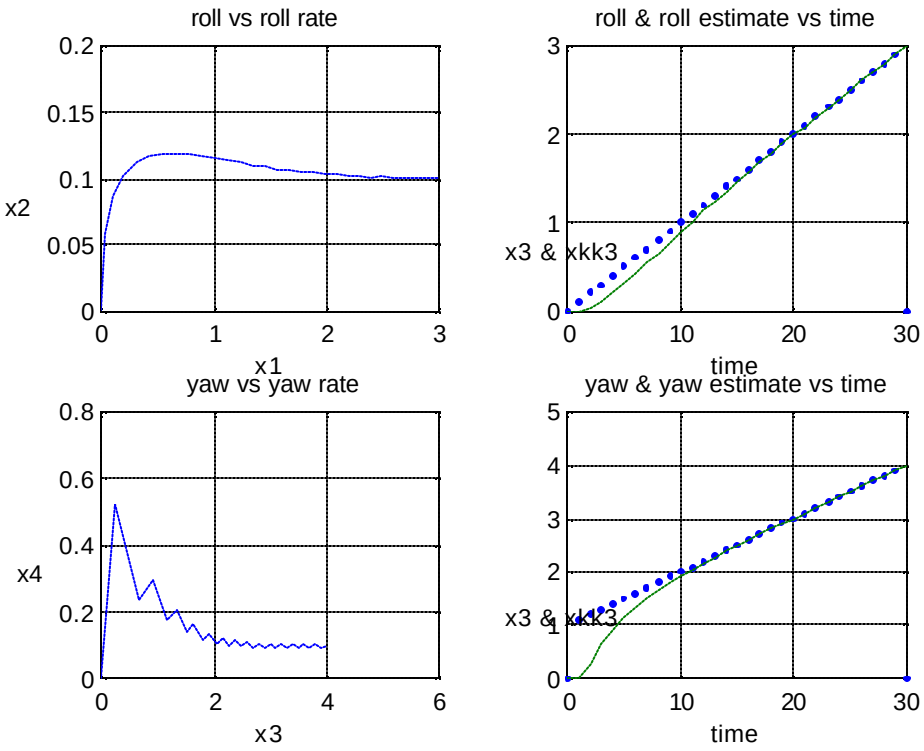

Figure 15. Ramp Trajectory.

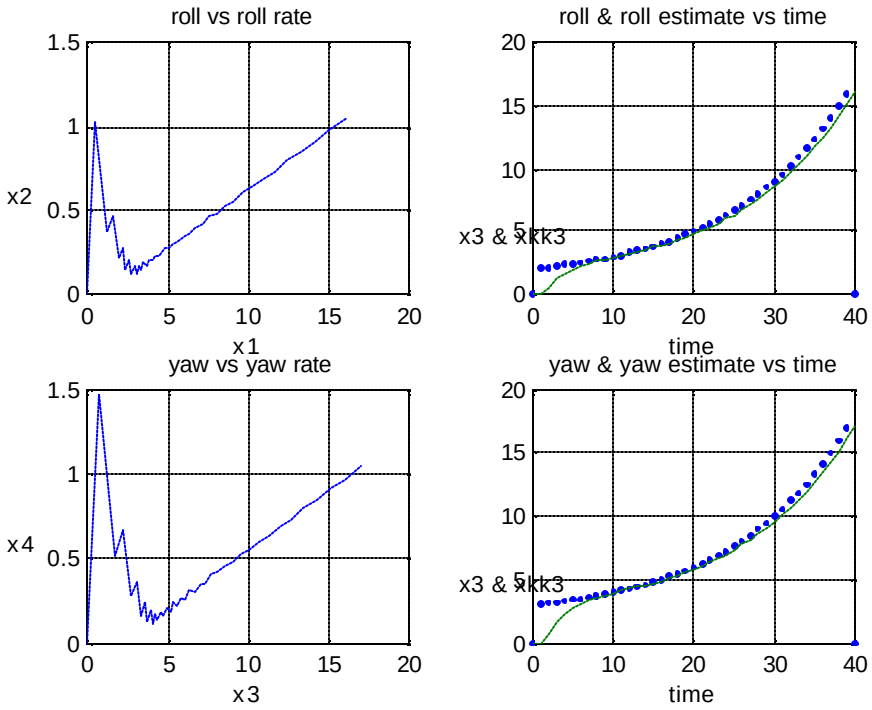

Figure 16. Exponential Trajectory.

7. Star Gap Effects

Another important area to consider is a star gap. From time to time, measurements from the star trackers may not be available. This can happen for several reasons, but in the past the biggest problem was the size of the star catalog. With a small star catalog, the stars that are in the field of view of the star tracker may not be included in the catalog and thus would be worthless. Today's star trackers have a much larger catalog and this problem has been reduced substantially. However, there may still be times in which a measurement is not available. The question we have to address is how well this filter will handle a star gap scenario.

As expected and seen in Figure 17, the attitude starts to wander off immediately when the star gap starts. At the end of the star gap, the filter reacts just as rapidly, achieving steady-state in 20 seconds.

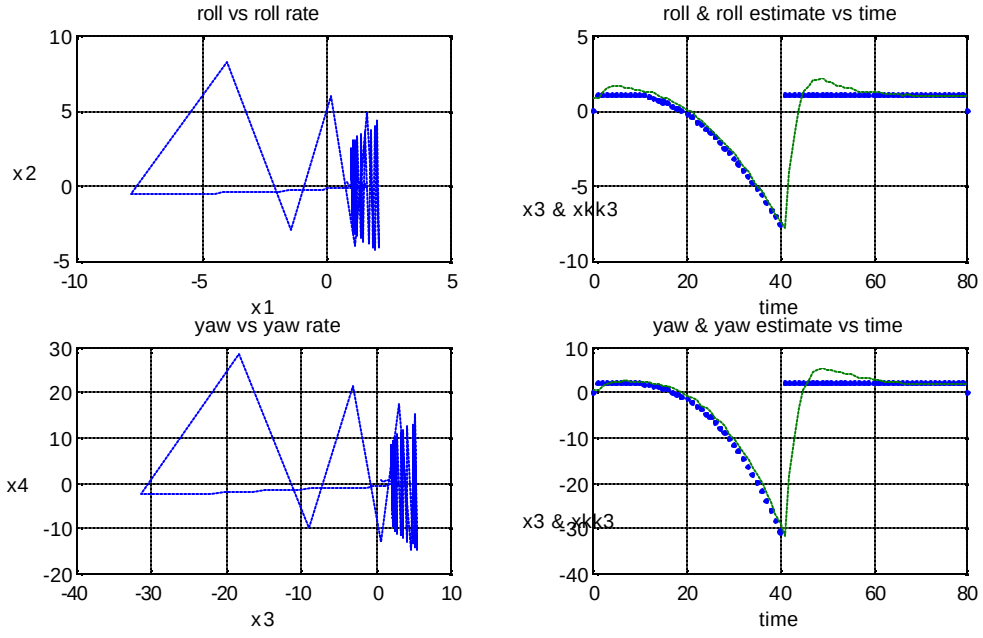

Figure 17. Star Gap (30 seconds).

While it would be desirable to see less change in the attitude during the star gap, we must accept some performance loss with our simplified dynamics equations. Furthermore, no attempt has been made so far in the code to improve this area. Such modifications should be explored in future work.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY

The accuracy and extensive catalogs of modern day star trackers make them the premier subsystem for attitude determination. The Euler Angles measured are within arcseconds of the actual attitude. This improvement in subsystem technology has allowed for the simplification of the algorithms. No longer do the limitations of the hardware require software workarounds. A basic Kalman Filter, like the ones characterized in this thesis, is able to acquire and track a spacecraft's attitude.

The simplicity of the algorithm developed in this paper is at the heart of its success. The dynamics of the spacecraft are simplified to the point where linear, time-invariant equations can be used in the model. Furthermore, it was shown that making the system coefficient matrix time-varying did not improve performance. Thus removing the computational need for additional coding.

While studying the active control of the covariance matrices is an important area of research, characterizing the filter was even more important and needed to be done first. Therefore, many of the different ways of manipulating the covariance matrices will have to be left for future work.

B. FUTURE WORK

1. Reverse-Time Smoothing

The Kalman Filter uses only past and present observations, and is therefore a causal filter. This is ideal for real time systems such as satellites. However, for improved estimates, the additional computing power of today's satellites could be utilized by post-processing old data. The smoothed past estimates could then be used in the filter.

When considering a fixed interval smoother, several methods have been developed to post-process data. Three of the most common are 1) forward-

backward smoother, 2) two-point boundary value approach, and 3) the Rauch-Tung-Streibel smoother [4]. Additional work in this area could potentially offer some accuracy improvements.

2. Rate Gyro Measurements

Utilizing star trackers for attitude determination will provide the Euler Angles that define the attitude at the time the sample is taken. Those along with the Euler Angle rates make up the state variables. The rates can be calculated more directly by taking the difference of the angles from one time step to another and dividing by the time step size. This would give much more sporadic results, so the rates estimated by the filter are preferable.

There are other techniques, not presented in this thesis, for using rate information in attitude determination. One area of future work, then, would be to incorporate some of those algorithms with the one developed here to improve the fidelity of the results. Where such algorithms to be employed, however, it may be worth considering the addition of rate gyros, which can provide rate measurements instead of just estimates.

APPENDIX

A. MATLAB CODE

1. Primary Code

```
%  Code written by Professor Hal Titus

%  Modified by LT Henry Travis 20 NOV 01

%  orbit by discrete. This becomes the pitch observable xo(3,i)

%  from Kepler  H = mr1^2*w1 = mr2^2*w2....r2 = xo(2,i) = (r1^2*w1/w2)^(0.5)

%  r2=((7439*0.62)^2*0.0013)^0.5;


% constants

mu = 3.986e5            % Gravitational coefficient (km3/sec2)

w=0.00013;              % orbital frequency (rads/sec)

Tf=29400*2;             % seconds in one 12 hour pass

Tf=50;                  % seconds in one 12 hour pass

dt=1;

kmax=Tf/dt +1;

trackerr=6*pi/(3600*180);


% Kalman Filter Covariances and observation matrices

% roll yaw

P=1*eye(4);

Q=[(dt^4)/4  (dt^3)/2 0 0;(dt^3)/2 dt^2 0 0; ...

   0 0 (dt^4)/4  (dt^3)/2;0 0 (dt^3)/2 dt^2]*.01;

R=[trackerr^2 0;0 trackerr^2];

H=[1 0 0 0;0 0 1 0];

% pitch

Pp=50*eye(2);
```

```
Qp=[(dt^4)/4  (dt^3)/2;(dt^3)/2 dt^2]*0.000001

Rp=[trackerr^2];

Hp=[1 0 ];


% initialize matrices

uo=zeros(2,kmax);ho=zeros(1,kmax);xo=zeros(4,kmax);

yo=zeros(1,kmax);time=zeros(1,kmax);xcart=zeros(2,kmax);

u=zeros(2,kmax);up=zeros(1,kmax);x=zeros(4,kmax);xp=zeros(2,kma x);

y=zeros(1,kmax);xf=zeros(4,kmax);time=zeros(1,kmax);zkkm1=zeros(2,1);

xkk=zeros(4,kmax);xkkm1=zeros(4,kmax);  z=zeros(2,1);

xkkp=zeros(2,kmax);xkkm1p=zeros(2,kmax);


% Define the Molniya orbit

Ao=[0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];

Bo=[ 0 0;1 0;0 0;0 1];

Co=[1 0 0  0];

[Phio,Delo]=c2d(Ao,Bo,dt);


% xo1 is r (km), xo2 is r dot, xo3 is theta (rad), xo4 is theta dot

xo(:,1)=[7439 0 0 .0013]';

h=xo(1)^2*xo(4)                  % angular momentum

for (i=1:kmax-1)

   uo(:,i)=[xo(1,i)*xo(4,i)^2-mu/(xo(1,i))^2;

           -(2*xo(2,i)*xo(4,i))/(xo(1,i))];

   xo(:,i+1) = Phio*xo(:,i) +Delo*uo(:,i);

  time(i+1)= time(i) + dt;

end;
```

```
% convert orbit to rectangular co-ordinates for plotting
for (i=1:kmax)
  xcart(1,i)=xo(1,i)*sin(xo(3,i));
  xcart(2,i)=xo(1,i)*cos(xo(3,i));
  xc(1,i)= 6378*sin(xo(3,i));
  xc(2,i)= 6378*cos(xo(3,i));
 end

% x1 is roll,x2 is roll rate,  x3 is yaw, x4 is yaw rate, xp is pitch
A=[0 1 0 0;0 0 0 -w;0 0 0 1;0 w 0 0];
B=[0 0; 1 0;0 0;0 1];
Ap=[0 1; 0 0];
Bp=[0 1]';

% place desired poles for roll and yaw
pz=[0.78 0.79 0.77  0.76]';
%pz=[.4 .41 .42 .43]';
% state transition matrix
[Phi,Del]=c2d(A,B,dt);

% gains and eigenvalues
k=place(Phi,Del,pz);
[klqr,s,elqr]=dlqr(Phi,Del,Q,R)        % discrete linear quadratic regulator
%k=klqr;
ppp=eig(Phi);
pppp=eig(Phi-Del*k);
```

```
% place desired poles for pitch
p=[.85 .856]';
[Phip,Delp]=c2d(Ap,Bp,dt);

% gains and eigenvalues
kd=place(Phip,Delp,p);
[klqrp,s,elqr]=dlqr(Phip,Delp,Qp,Rp)        % discrete linear quadratic regulator
%kd=klqrp;
pp=eig(Phip+Delp*kd);



% Kalman filter

% initial conditions
x(:,1)=[0.001; 0.002 ;0.003; 0.004];        % x is truth state
xkkm1(:,1)=[0.0;0.0;0;0.0013];             % z is truth measurement (includes meas. error)
zkkm1p=0;
xp(2,1)=0.0013;
xkkm1p(:,1)=[0;0];
zkkm1(:,1)=[0;0];
zp=.0013;

% run filter
for (i=1:kmax-1)
%  plant
  time(i+1)= time(i) + dt;
```

```
% for roll and yaw
G=P*H'*inv(H*P*H'+R);                    % Kalman Gain
Pk=(eye(4)-G*H)*P;                       % Covariance Update
P=Phi*Pk*Phi'+Q;                         % Covariance Prediction
xkk(:,i)=xkkm1(:,i)+G*(z-zkkm1);         % State estimate update
u(1,i)= k(1,:)*(xkk(:,i)-x(:,i));        % Control Law
u(2,i)= k(2,:)*(xkk(:,i)-x(:,i));        % Control Law
x(:,i+1) = Phi*x(:,i) + Del*u(:,i)  ;    % Update State
xkkm1(:,i+1)=Phi*xkk(:,i)+ Del*u(:,i) ;  % Predict State estimate

roller=12*(randn(1)-.5)*pi/(3600*180);   % +/- 6 arcseconds of random error
yawer=12*(randn(1)-.5)*pi/(3600*180);    % +/- 6 arcseconds of random error
pitcher=12*(randn(1)-.5)*pi/(3600*180);  % +/- 6 arcseconds of random error

z=[1+roller;2+yawer];
zkkm1=[xkkm1(1,i+1);xkkm1(3,i+1)];       % Update Measurement estimate

% for pitch
Gp=Pp*Hp'*inv(Hp*Pp*Hp'+Rp);             % Kalman Gain
Pkp=(eye(2)-Gp*Hp)*Pp;                   % Covariance Update
Pp=Phip*Pkp*Phip'+Qp;                    % Covariance Prediction
xkkp(:,i)=xkkm1p(:,i)+Gp*(zp-zkkm1p);    % State estimate update
up(i)= -(2*xo(2,i)*xkkp(2,i))/(xo(1,i)); % Control Law
xp(:,i+1) = Phip*xp(:,i)  +Delp*up(i);   % Update State
xkkm1p(:,i+1)=Phip*xkkp(:,i)+ Delp*up(i);  % Predict State estimate
zp=[xo(3,i)+pitcher];                    % Update Measurement
```

```
  zkkm1p=[xkkm1p(1,i+1)];                    % Update Measurement estimate
end % Kalman loop



clf
figure(1)
plot(xcart(1,:),xcart(2,:),0,0,'*',xc(1,:),xc(2,:))
AXIS([-35000 35000 -60000 10000])
XLABEL('km'); YLABEL('km')

figure(2)
subplot(221),plot(x(1,:),x(2,:))
title('roll vs roll rate ')
xlabel('x1'), ylabel('x2'),grid

subplot(222),plot(time(1,:),xkk(1,:),'.',time(1,:),x(1,:))
title('roll & roll estimate vs time')
xlabel('time'), ylabel('x3 & xkk3'),grid

subplot(223),plot(x(3,:),x(4,:))
title(' yaw vs yaw rate')
xlabel('x3'), ylabel('x4'),grid

subplot(224),plot(time(1,:),xkk(3,:),'.',time(1,:),x(3,:))
title('yaw & yaw estimate vs time')
xlabel('time'), ylabel('x3 & xkk3'),grid
```

```
figure(3)
subplot(211)
plot(time(1,:),xkkp(1,:),time(1,:),xp(1,:),'.')
title(['pitch  & pitch  est vs time']);
xlabel('time'),ylabel('pitch rate    xp2');
grid;

subplot(212)
plot(time(1,:),xkkp(2,:),time(1,:),xp(2,:),'.')
title(['pitch rate & pitch rate est vs time']);
xlabel('time'),ylabel('pitch rate    xp2');  grid;
```

2. Profile Generator

```
function xtrue=profile(kmax)

% Random number generator

 for i=1:kmax
 y(1,i+1)=12*(randn(1)-.5)*pi/(3600*180);    % +/- 6 arcseconds of random error
 y(2,i+1)=12*(randn(1)-.5)*pi/(3600* 180);    % +/- 6 arcseconds of random error

 xtrue(:,i)=[1;0;2;0;1+y(1,i+1); 2+y(2,i+1)];   % profile variable
          end
```

3. Star Gap Code

```
%  Code written by LT Henry Travis 26 NOV 01
%  orbit by discrete. This becomes the pitch observable xo(3,i)
%  from Kepler  H = mr1^2*w1 = mr2^2*w2....r2 = xo(2,i) = (r1^2*w1/w2)^(0.5)
%  r2=((7439*0.62)^2*0.0013)^0.5;

% constants
mu = 3.986e5                % Gravitational coefficient (km3/sec2)
w=0.00013;                  % orbital frequency (rads/sec)
N=-1;
Tf=29400*2;                 % seconds in one 12 hour pass

Tf=80;                      % seconds in one 12 hour pass
dt=1;
kmax=Tf/dt +1;

trackerr=6*pi/(3600*180);
zdifference=0;
zpdifference=0;
threesigma=1;


% Kalman Filter Covariances and observation matrices
% roll yaw
P=1*eye(4);
Q=[(dt^4)/4  (dt^3)/2 0 0;(dt^3)/2 dt^2 0 0; ...
   0 0 (dt^4)/4  (dt^3)/2;0 0 (dt^3)/2 dt^2]*.01;
R=[trackerr^2 0;0 trackerr^2];
H=[1 0 0 0;0 0 1 0];
% pitch
Pp=50*eye(2);
Qp=[(dt^4)/4  (dt^3)/2;(dt^3)/2 dt^2]*0.000001
Rp=[trackerr^2];
Hp=[1 0 ];
```

```
% initialize matrices
uo=zeros(2,kmax);ho=zeros(1,kmax);xo=zeros(4,kmax);
yo=zeros(1,kmax);time=zeros(1,kmax);xcart=zeros(2,kmax);
u=zeros(2,kmax);up=zeros(1,kmax);x=zeros(4,kmax);xp=zeros(2,kmax);
y=zeros(1,kmax);xf=zeros(4,kmax);time=zeros(1,kmax); zkkm1=zeros(2,1);
z=zeros(2,1);
xkk=zeros(4,kmax);xkkm1=zeros(4,kmax);
xkkp=zeros(2,kmax);xkkm1p=zeros(2,kmax);


% Define the Molniya orbit

Ao=[0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];
Bo=[ 0 0;1 0;0 0;0 1];
Co=[1 0 0  0];
[Phio,Delo]=c2d(Ao,Bo,dt);


% xo1 is r (km), xo2 is r dot, xo3 is theta (rad), xo4 is theta dot
xo(:,1)=[7439 0 0 .0013]';
h=xo(1)^2*xo(4)                 % angular momentum

for (i=1:kmax-1)
   uo(:,i)=[xo(1,i)*xo(4,i)^2-mu/(xo(1,i))^2;
           -(2*xo(2,i)*xo(4,i))/(xo(1,i))];
    xo(:,i+1) = Phio*xo(:,i) +Delo*uo(:,i);
  time(i+1)= time(i) + dt;
end;

% convert orbit to rectangular co-ordinates for plotting
for (i=1:kmax)
  xcart(1,i)=xo(1,i)*sin(xo(3,i));
  xcart(2,i)=xo(1,i)*cos(xo(3,i));
  xc(1,i)= 6378*sin(xo(3,i));
  xc(2,i)= 6378*cos(xo(3,i));
 end
```

```
% x1 is roll,x2 is roll rate,  x3 is yaw, x4 is yaw rate, xp is pitch
A=[0 1 0 0;0 0 0 -w;0 0 0 1;0 w 0 0];
B=[0 0; 1 0;0 0;0 1];
Ap=[0 1; 0 0];
Bp=[0 1]';

% place desired poles for roll and yaw
pz=[0.78 0.79 0.77  0.76]';
%pz=[.4 .41 .42 .43]';
% state transition matrix
[Phi,Del]=c2d(A,B,dt);

% gains and eigenvalues
k=place(Phi,Del,pz);
[klqr,s,elqr]=dlqr(Phi,Del,Q,R)         % discrete linear quadratic regulator
%k=klqr;
ppp=eig(Phi);
pppp=eig(Phi-Del*k);

% place desired poles for pitch
p=[.85 .856]';
[Phip,Delp]=c2d(Ap,Bp,dt);

% gains and eigenvalues
kd=place(Phip,Delp,p);
[klqrp,s,elqr]=dlqr(Phip,Delp,Qp,Rp)         % discrete linear quadratic regulator
%kd=klqrp;
pp=eig(Phip+Delp*kd);



% Kalman filter

% initial conditions
x(:,1)=[0.01; 0.02 ;0.03; 0.04];        % x is truth state
x(:,1)=[0.8; 0.2 ;0.6; 0.4];        % x is truth state
```

```
xkkm1(:,1)=[0.0;0.0;0;0.0013];          % z is truth measurement (includes meas. error)
zkkm1p=0;
xp(2,1)=0.0013;
xkkm1p(:,1)=[0;0];
zkkm1(:,1)=[0;0];
zp=.0013;
skipme=0;
% run filter
for (i=1:kmax-1)
%  plant
  time(i+1)= time(i) + dt;


  % for roll and yaw

  G=P*H'*inv(H*P*H'+R);                 % Kalman Gain
  Pk=(eye(4)-G*H)*P;                    % Covariance Update
  P=Phi*Pk*Phi'+Q;                      % Covariance Prediction
  xkk(:,i)=xkkm1(:,i)+G*(z-zkkm1);      % State estimate update
  u(1,i)= k(1,:)*(xkk(:,i)-x(:,i));     % Control Law
  u(2,i)= k(2,:)*(xkk(:,i)-x(:,i));     % Control Law
  x(:,i+1) = Phi*x(:,i) + Del*u(:,i)  ; % Update State
  xkkm1(:,i+1)=Phi*xkk(:,i)+ Del*u(:,i) ;  % Predict State estimate

  roller=12*(randn(1)-.5)*pi/(3600*180);   % +/- 6 arcseconds of random error
  yawer=12*(randn(1)-.5)*pi/(3600*180);    % +/- 6 arcseconds of random error
  pitcher=12*(randn(1)-.5)*pi/(3600*180);  % +/- 6 arcseconds of random error
  z=[1+roller;2+yawer];                    % Update Measurement
  zdifference =0;

if (i<10)|(i>40)                  % Introduce large error from time 10 to 40
  z=[3;4];                        % Arbitrarily large measurement
  zdifference=z-zkkm1;            % calulate difference from one time step to next
end
```

```
if (zdifference>threesigma)                    % if difference is outside 3 sigma, replace
  z=[xkkm1(1,i+1);xkkm1(3,i+1)];               % measurement with estimate.
end

  zkkm1=[xkkm1(1,i+1);xkkm1(3,i+1)];           % Update Measurement estimate

  % for pitch
  Gp=Pp*Hp'*inv(Hp*Pp*Hp'+Rp);                 % Kalman Gain
  Pkp=(eye(2)-Gp*Hp)*Pp;                       % Covariance Update
  Pp=Phip*Pkp*Phip'+Qp;                        % Covariance Prediction
  xkkp(:,i)=xkkm1p(:,i)+Gp*(zp-zkkm1p);        % State estimate update
  up(i)= -(2*xo(2,i)*xkkp(2,i))/(xo(1,i));     % Control Law
  xp(:,i+1) = Phip*xp(:,i)  +Delp*up(i);       % Update State
  xkkm1p(:,i+1)=Phip*xkkp(:,i)+ Delp*up(i); % Predict State estimate
  zp=[xo(3,i+1)+ ((randn(1)-.5)*trackererr)]; % Update Measurement as truth plus noise

zpdifference=0;

if (i<10)|(i>40)                               % Introduce large error from time 10 to 40
    zp=3;                                      % Arbitrarily large measurement
    zpdifference=zp-zkkm1p;                    % calulate difference from one time step to next
end

if (zpdifference>threesigma)                   % if difference is outside 3 sigma, use estimate
 zp=[xp(1,i+1)];                               % Update Measurement
end

zkkm1p=[xkkm1p(1,i+1)];                        % Update Measurement estimate

end  % ends Kalman loop

clf
figure(1)
plot(xcart(1,:),xcart(2,:),0,0,'*',xc(1,:),xc(2,:))
AXIS([-35000 35000 -60000 10000])
```

```
XLABEL('km')
YLABEL('km')

figure(2)

subplot(221),plot(x(1,:),x(2,:))
title('roll vs roll rate ')
xlabel('x1'), ylabel('x2'),grid

subplot(222),plot(time(1,:),xkk(1,:),'.',time(1,:),x(1,:))
title('roll & roll estimate vs time')
xlabel('time'), ylabel('x3 & xkk3'),grid

subplot(223),plot(x(3,:),x(4,:))
title(' yaw vs yaw rate')
xlabel('x3'), ylabel('x4'),grid

subplot(224),plot(time(1,:),xkk(3,:),'.',time(1,:),x(3,:))
title('yaw & yaw estimate vs time')
xlabel('time'), ylabel('x3 & xkk3'),grid

figure(3)
subplot(211)
plot(time(1,:),xkkp(1,:),time(1,:),xp(1,:),'.')
title(['pitch  & pitch  est vs time']);
xlabel('time'),ylabel('pitch rate   xp2');
grid;

subplot(212)
plot(time(1,:),xkkp(2,:),time(1,:),xp(2,:),'.')
title(['pitch rate & pitch rate est vs time']);
xlabel('time'),ylabel('pitch rate   xp2');
grid;
```

LIST OF REFERENCES

1. Greenwood, Donald T., *Principles of Dynamics*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ, 1988, p. 199

2. Phillips, Charles L., Harbor, Royce D., *Feedback Control Systems*, 4th ed. Prentice Hall, Upper Saddle River, NJ, 2000, p. 435

3. Mohler, R. R., *Nonlinear Systems Volume I Dynamics and Control*, Prentice Hall, Englewood Cliffs, NJ, 1991, p. 129

4. Moon, Todd K., Stirling, Wynn C., *Mathematical Methods and Algorithms for Signal Processing*, Prentice Hall, Upper Saddle River, NJ, 2000, p. 613

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

1. Brown, Robert G., Hwang, Patrick Y.C., *Introduction to Random Signals and Applied Kalman Filtering*, 3rd ed. Wiley, New York, NY, 1997

2. Ackermann, J.E., "Der Entwurf linearer regelungs Systems in Zustandstraum," Regelungstech. Prozess-Datenverarb. 7, 1972

3. Dorf, Richard C., *Modern Control Systems*, 6th ed. Addison-Wesley Publishing Company, Reading, MA, 1992

4. Kolve, D. I., "Describing an Attitude", Proceedings of the 16th Annual American Astronautical Society Guidance and Control Conference, Keystone, CO, 1993

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5000

3. Department Chairman, Code EE
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Hal A. Titus
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Brij N. Agrawal, Code AA/Ag
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. SRDC Research Library, Code AA
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943-5000

7. Henry D. Travis
   2139 Elysium Ave
   Eugene, OR 97401