OPTIMAL SLEWING OF A CONSTRAINED TELESCOPE USING SEVENTH ORDER POLYNOMIAL INPUT TORQUES

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Julia Bush

September 2012

COMMITTEE MEMBERSHIP

TITLE:	Optimal Slewing of a Constrained Telescope Using Seventh Order Polynomial Input Torques
AUTHOR:	Julia Bush
DATE SUBMITTED:	September 2012
COMMITTEE CHAIR:	Dr. Kira Abercromby
COMMITTEE MEMBER:	Dr. Eric A. Mehiel
COMMITTEE MEMBER:	David Levinson

ABSTRACT

Optimal Slewing of a Constrained Telescope Using Seventh Order Polynomial Input Torques

Julia Bush

Two-axis gimbals are frequently used to point cameras and telescopes at various points of interest for surveillance, science, and art. The rotation of a two-axis gimbal system is governed by nonlinear angular momentum equations of motion. This paper presents a method for slewing a telescope in space with a gimbaled sensor attached to a nominally non-rotating spacecraft using two seventh order polynomial input functions to characterize torques. To accomplish this task, picking the optimal coefficients of the seventh order polynomial was necessary. It was also desired to use constraint equations to limit the excursion, angular velocity, angular acceleration, and jerk of the gimbal. A Matlab code was developed for this purpose. Matlab's fmincon was used to do the optimization, and a comparison to a previously validated one-degree-of-freedom (DOF) model was presented for validation of the nonlinear, two-degree-of-freedom model. Results for a fully constrained 2 DOF slew maneuver were also shown. This thesis demonstrates that seventh order polynomial torques can be used to accurately slew a telescope in space using nonlinear equations of motion.

Keywords: 2-axis gimbal, nonlinear optimization, 7th-order polynomial, Matlab, telescope

ACKNOWLEDGMENTS

I would like to thank David Levinson and Andy Peronto for their contributions to this project. Without Dave's crash course in Dynamics this project would not have gotten off the ground and without Andy's vast experience and familiarity with Control Systems and Matlab this project would not have come to completion. Many thanks for the hours you both put into this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

Two-axis gimbals are frequently used to point cameras and telescopes at various points of interest for surveillance, science, and art. The rotation of a two-axis gimbal system is governed by nonlinear angular momentum equations of motion. They can often times be approximated as linear, but the usefulness of models that use these equations is limited. The only situations when it might be appropriate to linearize the angular momentum equations is if there is only rotation about one axis, or if the body is symmetrical about all three axes (for example, a sphere or cube), where the products of inertia are all equal to zero. Therefore, the fully nonlinear angular momentum equations need to be the basis of a useful model for slewing a telescope on a two-axis gimbal. There are many ways to slew a telescope from one orientation and angular velocity state to another. This paper presents a method for slewing a telescope in space with a gimbaled sensor attached to a nominally non-rotating spacecraft using two seventh order polynomial input functions to characterize torques. A seventh order polynomial has eight coefficients, which is enough to accommodate a wide range of slewing motions. To accomplish this task, picking the optimal coefficients for the seventh order polynomial was necessary. It was also desired to use constraint equations to limit the excursion, angular velocity, angular acceleration, and jerk of the gimbal.

The equations of motion developed in this thesis lay the groundwork for a full control system design that can stabilize the telescope (i.e., keep it pointing at a particular azimuth and elevation). The dynamics model reveals what happens when either a pitch or yaw torque is applied. The results from Matlab script show which seventh order polynomial torques are needed (in pitch and yaw) to enable a telescope to slew from one orientation to another if the dynamics model were a perfect representation of reality. However in the real world, no model is perfect. So ideally one

would design a control system that will keep the telescope stabilized by fighting any disturbances that occur that cannot be perfectly accounted for, such as solar wind, aerodynamic effects at high altitude, and internal vibrations [7].

The model presented in this thesis could be used to do a coarse slewing of the telescope from one orientation to another (which is a large angular maneuver that a control system based on linear models for small perturbations could have trouble executing); however, after the slew, some other control system logic would take over that would correct the small errors that came up during the coarse slew and would also keep the telescope pointing where the user wants it after the slew is finished. The approach is similar to that employed by an airplane's guidance and control system. The pilot has to take off and land the plane, where there can be highly nonlinear dynamics occurring, but once the plane is cruising in straight and level flight, an autopilot can be turned on to keep it going straight and level. The equations of motion for an airplane about a trim condition are well-approximated by linear models. Autopilots have no difficulty returning the plane to its trim condition if the plane encounters turbulence [7].

Background

The work of Yoon and Lundberg [1] in modeling two-axis yaw-pitch gimbal configurations has heavily influenced the mathematical model presented in this thesis. In their paper, Yoon and Lundberg [1] examine the dynamics and derive the equations of motion for a two-axis gimbal system. Simplified versions of these equations of motion are used in this thesis to model mathematically a telescope mounted to a nominally non-rotating spacecraft in space. To solve the equations numerically, a Matlab code was written and an optimizer iterates on the inputs to the .code to find the optimal coefficients of two seventh order polynomials to drive the telescope

from one state to another.

Many others have done research in this area, using Yoon and Lundberg as a jumping off point, although the majority of the work is related to Unmanned Aerial Vehicles (UAVs) rather than spacecraft. The driving concern is to remove the need for a human being to be in an airplane taking pictures (this would also be a driving concern for space-based telescopes). Automating the function of pointing cameras/telescopes and capturing pictures means less risk and better use of available resources.

In his paper, O'Connor [5] examines spacecraft control systems. According to O'Connor, a three-axis spacecraft can be accurately simulated using independent motion on two axes, as long as rotation around the third axis remains approximately zero. The unknown in this scenario is the torque input required to satisfy the time boundary conditions. One relatively simple approach to solving this problem is a least squares minimization. This requires that the input be some sort of linear, definite, multivariate function that is simple to evaluate [5]. For this case, the easiest solution is a polynomial, which when chosen with high order, can take most any form and thus provide the least error in the final state. The goals of O'Connor's exploration are to determine the error and the time it takes his simulation to run and to calculate the polynomial input coefficients required, given initial and final conditions of a two-state system as well as numerical values of the system parameters.

In order to achieve these ends, O' Connor linearized the equations of motion of one axis of a spacecraft and put them into state-space form. He chose initial and final states in order to test the accuracy of the calculated input. Additionally, he added full state feedback to the system to

simulate the spacecraft having closed loop damping on a level below the main command algorithm. The input was modeled as a seventh order polynomial with variable coefficients. From this, the time-domain analytic solution could be reduced to an over-determined Least Squares Minimization (LSM) problem. The LSM solution was found, which consisted of the coefficients for the polynomial input function. O'Connor coded the entire algorithm in Matlab, and a Simulink model was created with the same equations of motion and initial state for validation. From the calculated input the simulation determines the actual final state. He compared the commanded final state in Matlab to the actual final state from Simulink to determine the error between the two states, and the state time history from Simulink was used to gain insight into system behavior.

Equations of Motion for a Two-Axis Gimbal System

System description:

Figure 1 is a schematic representation of the system under consideration, consisting of a rigid spacecraft B, a gimbal K, and a telescope A. For purposes of this analysis, B is regarded as fixed in a Newtonian reference frame N. K is connected to B by means of a revolute joint, and another revolute joint connects A to K. The axes L_{BK} and L_{KA} of these joints are perpendicular to each other and intersect, the point of intersection here taken to be both the mass center A_O of A and the mass center K_O of K. $\underline{B}_1, \underline{B}_2, \underline{B}_3$ form a right-handed set of mutually perpendicular unit vectors fixed in B, with $\underline{B}_3$ parallel to L_{BK}. A second right-handed set of mutually perpendicular unit vectors $\underline{K}_1, \underline{K}_2, \underline{K}_3$ is fixed in K with $\underline{K}_3 = \underline{B}_3$ and $\underline{K}_2$ parallel to L_{KA}, as shown. Finally, a third right-handed set of mutually perpendicular unit vectors $\underline{A}_1, \underline{A}_2, \underline{A}_3$ is fixed in A, with $\underline{A}_1$ parallel to the optical axis of A, and $\underline{A}_2 = \underline{K}_2$. The angle between $\underline{B}_2$, and $\underline{K}_2$ (as well as between $\underline{B}_1$ and $\underline{K}_1$) is denoted by ν_1, and the angle between $\underline{A}_1$ and $\underline{K}_1$ (as well as between $\underline{A}_3$ and $\underline{K}_3$) is designated as ν_2.

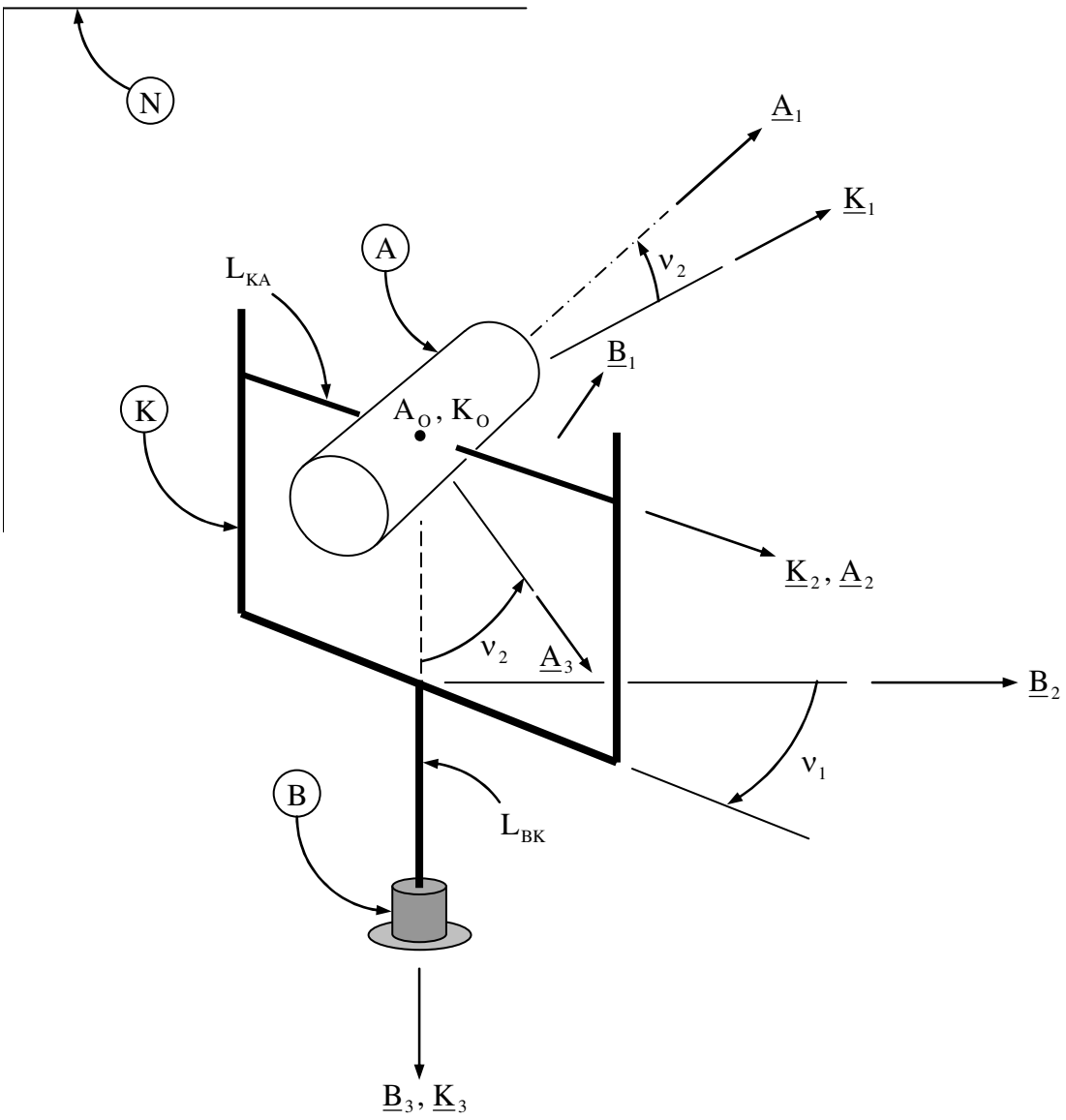

Figure 1: Schematic Representation of Telescope and Gimbal

Direction cosine tables:

Table 1 contains the direction cosines relating $\underline{B}_1, \underline{B}_2, \underline{B}_3$ to $\underline{K}_1, \underline{K}_2, \underline{K}_3$. Similarly Table 2 contains the direction cosines relating $\underline{K}_1, \underline{K}_2, \underline{K}_3$ to $\underline{A}_1, \underline{A}_2, \underline{A}_3$.

Table 1. Direction Cosines Relating $\underline{B}_1, \underline{B}_2, \underline{B}_3$, to $\underline{K}_1, \underline{K}_2, \underline{K}_3$

	$\underline{K}_1$	$\underline{K}_2$	$\underline{K}_3$
$\underline{B}_1$	$\cos\nu_1$	$-\sin\nu_1$	0
$\underline{B}_2$	$\sin\nu_1$	$\cos\nu_1$	0
$\underline{B}_3$	0	0	1

Table 2. Direction Cosines Relating $\underline{K}_1$, $\underline{K}_2$, $\underline{K}_3$ to $\underline{A}_1$, $\underline{A}_2$, $\underline{A}_3$

	$\underline{A}_1$	$\underline{A}_2$	$\underline{A}_3$
$\underline{K}_1$	$\cos\nu_2$	0	$\sin\nu_2$
$\underline{K}_2$	0	1	0
$\underline{K}_3$	$-\sin\nu_2$	0	$\cos\nu_2$

Kinematical Equations:

The angular velocity ${}^N\underline{\omega}^K$ of K in N can be written as

$$ {}^N\underline{\omega}^K = p_K\underline{K}_1 + q_K\underline{K}_2 + r_K\underline{K}_3 \tag{1}$$

where p_K, q_K, and r_K are scalar functions of time, while the angular velocity ${}^N\underline{\omega}^K$ of B in N can be expressed as

$$ {}^N\underline{\omega}^B = p\underline{B}_1 + q\underline{B}_2 + r\underline{B}_3 \tag{2}$$

and the angular velocity ${}^B\underline{\omega}^K$ of K in B is given by (see Figure 1)

$$ {}^B\underline{\omega}^K = \dot{\nu}_1\underline{K}_3 \tag{3}$$

The Addition Theorem for Angular Velocities states that:

$$ {}^N\underline{\omega}^K = {}^N\underline{\omega}^B + {}^B\underline{\omega}^K \tag{4}$$

Substituting from Equations (1) – (3) into Equation (4) gives:

$$p_K \underline{K}_1 + q_K \underline{K}_2 + r_K \underline{K}_3 = p\underline{B}_1 + q\underline{B}_2 + r\underline{B}_3 + \dot{\nu}_1 \underline{K}_3 \tag{5}$$

Dot multiplying Equation (5) successively with $\underline{K}_1$, $\underline{K}_2$, $\underline{K}_3$ yields, with the aid of Table 1, the kinematical equations:

$\underline{K}_1$: $$p_K = p\cos\nu_1 + q\sin\nu_1 \tag{6}$$

$\underline{K}_2$: $$q_K = -p\sin\nu_1 + q\cos\nu_1 \tag{7}$$

$\underline{K}_3$: $$r_K = r + \dot{\nu}_1 \tag{8}$$

The angular velocity ${}^N\underline{\omega}^A$ of A in N can be expressed as

$${}^N\underline{\omega}^A = p_A \underline{A}_1 + q_A \underline{A}_2 + r_A \underline{A}_3 \tag{9}$$

The angular velocity ${}^K\underline{\omega}^A$ of A in K is given by (see Figure 1)

$${}^K\underline{\omega}^A = \dot{\nu}_2 \underline{K}_2 \tag{10}$$

The Addition Theorem for Angular Velocities states that

$${}^N\underline{\omega}^K + {}^K\underline{\omega}^A = {}^N\underline{\omega}^A \tag{11}$$

Substituting from Equations (1), (9), and (10) into Equation (11) gives

$$p_K \underline{K}_1 + q_K \underline{K}_2 + r_K \underline{K}_3 + \dot{\nu}_2 \underline{K}_2 = p_A \underline{A}_1 + q_A \underline{A}_2 + r_A \underline{A}_3 \tag{12}$$

Dot multiplying Equation (12) successively with $\underline{A}_1$, $\underline{A}_2$, $\underline{A}_3$ yields, with the aid of Table 2, the kinematical equations:

$\underline{A}_1$: $$p_A = p_K\cos\nu_2 - r_K\sin\nu_2 \tag{13}$$

$\underline{A}_2$: $\quad q_A = q_K + \dot{\nu}_2$ (14)

$\underline{A}_3$: $\quad r_A = p_K \sin\nu_2 + r_K \cos\nu_2$ (15)

Dynamical Equations:

Pitch gimbal

The mass centers A_0 of A and K_0 of K are presumed to be coincident and located at the intersection of L_{BK} and L_{KA}.

The inertia dyadic $\underline{\underline{I}}^{A/A_0}$ of A for A_0 can be expressed as

$$\begin{aligned}\underline{\underline{I}}^{A/A_0} &= J_{ax}\underline{A}_1\underline{A}_1 + J_{ay}\underline{A}_2\underline{A}_2 + J_{az}\underline{A}_3\underline{A}_3 \\ &+ D_{xy}(\underline{A}_1\underline{A}_2 + \underline{A}_2\underline{A}_1) + D_{yz}(\underline{A}_2\underline{A}_3 + \underline{A}_3\underline{A}_2) + D_{zx}(\underline{A}_3\underline{A}_1 + \underline{A}_1\underline{A}_3)\end{aligned} \tag{16}$$

where J_{ax}, J_{ay}, and J_{az} are the moments of inertia of A about lines passing through A_0 and parallel to $\underline{A}_1$, $\underline{A}_2$, $\underline{A}_3$, respectively, and D_{xy}, D_{yz}, and D_{zx} are the associated products of inertia.

The angular momentum of A for A_0 in N is defined as

$${}^N\underline{H}^{A/A_0} \overset{\Delta}{=} \underline{\underline{I}}^{A/A_0} \cdot {}^N\omega^A \tag{17}$$

Substituting into Equation (17) from (9) and (16) yields

$$\begin{aligned}{}^N\underline{H}^{A/A_0} &= J_{ax}p_A\underline{A}_1 + D_{xy}p_A\underline{A}_2 + D_{zx}p_A\underline{A}_3 + J_{ay}q_A\underline{A}_2 + D_{xy}q_A\underline{A}_1 + D_{yz}q_A\underline{A}_3 \\ &\quad + J_{az}r_A\underline{A}_3 + D_{yz}r_A\underline{A}_2 + D_{zx}r_A\underline{A}_1 \\ &= \left(J_{ax}p_A + D_{xy}q_A + D_{zx}r_A\right)\underline{A}_1 + \left(D_{xy}p_A + J_{ay}q_A + D_{yz}r_A\right)\underline{A}_2 \\ &\quad + \left(D_{zx}p_A + D_{yz}q_A + J_{az}r_A\right)\underline{A}_3\end{aligned} \tag{18}$$

The system of forces exerted by K on A is assumed to be equivalent to a couple of torque, $\underline{T}^{A}$, given by:

$$\underline{T}^{A} = T_X^A \underline{A}_1 + T_Y^A \underline{A}_2 + T_Z^A \underline{A}_3 \tag{19}$$

The angular momentum principle states that

$$^{N}\frac{d}{dt}\left(^{N}\underline{H}^{A/A_0}\right) = \underline{T}^{A} \tag{20}$$

where $^{N}\frac{d}{dt}$ denotes time-differentiation in N.

Since $^{N}\underline{H}^{A/A0}$ is expressed in terms of $\underline{A}_1$, $\underline{A}_2$, $\underline{A}_3$ [see Equation 18], it is convenient to rewrite Equation (20) in terms of time-differentiation in A rather than N. To this end, note that, for any vector $\underline{z}$, the time derivative of $\underline{z}$ in a reference frame F_1 is related to the time derivative of $\underline{z}$ in a reference frame F_2 in accordance with the formula

$$^{F_1}\frac{d\underline{z}}{dt} = ^{F_2}\frac{d\underline{z}}{dt} + ^{F_1}\underline{\omega}^{F_2} \times \underline{z} \tag{21}$$

where $^{F1}\underline{\omega}^{F2}$ is the angular velocity of F_2 in F_1. One can replace Equation (20) with

$$\underline{T}^{A} = ^{A}\frac{d}{dt}\left(^{N}\underline{H}^{A/A_0}\right) + ^{N}\underline{\omega}^{A} \times ^{N}\underline{H}^{A/A_0} \tag{22}$$

Substituting from Equation (9) and (18) into Equation (22) leads to

$$
\begin{aligned}
\underline{T}^{A} &= {}^{A}\frac{d}{dt}\left({}^{N}\underline{H}^{A/A_0}\right) + {}^{N}\underline{\omega}^{A}\times{}^{N}\underline{H}^{A/A_0} \\
&= \left(J_{ax}\dot{p}_A + D_{xy}\dot{q}_A + D_{zx}\dot{r}_A\right)\underline{A}_1 + \left(D_{xy}\dot{p}_A + J_{ay}\dot{q}_A + D_{yz}\dot{r}_A\right)\underline{A}_2 + \left(D_{zx}\dot{p}_A + D_{yz}\dot{q}_A + J_{az}\dot{r}_A\right)\underline{A}_3 \\
&+ \left[q_A\left(D_{zx}p_A + D_{yz}q_A + J_{az}r_A\right) - r_A\left(D_{xy}p_A + J_{ay}q_A + D_{yz}r_A\right)\right]\underline{A}_1 \\
&- \left[p_A\left(D_{zx}p_A + D_{yz}q_A + J_{az}r_A\right) - r_A\left(J_{ax}p_A + D_{xy}q_A + D_{zx}r_A\right)\right]\underline{A}_2 \\
&+ \left[p_A\left(D_{xy}p_A + J_{ay}q_A + D_{yz}r_A\right) - q_A\left(J_{ax}p_A + D_{xy}q_A + D_{zx}r_A\right)\right]\underline{A}_3
\end{aligned}
$$

$$
\begin{aligned}
&= \left[J_{ax}\dot{p}_A + D_{xy}\dot{q}_A + D_{zx}\dot{r}_A + D_{zx}p_Aq_A + D_{yz}q_A^2 + J_{az}q_Ar_A - D_{xy}p_Ar_A - J_{ay}q_Ar_A - D_{yz}r_A^2\right]\underline{A}_1 \\
&+ \left[D_{xy}\dot{p}_A + J_{ay}\dot{q}_A + D_{yz}\dot{r}_A - D_{zx}p_A^2 - D_{yz}p_Aq_A - J_{az}p_Ar_A + J_{ax}p_Ar_A + D_{xy}q_Ar_A + D_{zx}r_A^2\right]\underline{A}_2 \\
&+ \left[D_{zx}\dot{p}_A + D_{yz}\dot{q}_A + J_{az}\dot{r}_A + D_{xy}p_A^2 + J_{ay}p_Aq_A + D_{yz}p_Ar_A - J_{ax}p_Aq_A - D_{xy}q_A^2 - D_{zx}q_Ar_A\right]\underline{A}_3
\end{aligned}
$$

$$
= T_X^A\,\underline{A}_1 + T_Y^A\,\underline{A}_2 + T_Z^A\,\underline{A}_3 \tag{23}
$$

The scalar equation of interest is obtained by dot-multiplying Equation (23) by $\underline{A}_2$

$$
T_Y^A = J_{ay}\dot{q}_A + D_{xy}\left(\dot{p}_A + q_Ar_A\right) + D_{yz}\left(\dot{r}_A - p_Aq_A\right) + D_{zx}\left(r_A^2 - p_A^2\right) + \left(J_{ax} - J_{az}\right)p_Ar_A \tag{24}
$$

Yaw gimbal

The inertia dyadic $\underline{\underline{I}}^{K/K_0}$ of K for K_0 can be written as

$$
\begin{aligned}
\underline{\underline{I}}^{K/K_0} &= J_{kx}\underline{K}_1\underline{K}_1 + J_{ky}\underline{K}_2\underline{K}_2 + J_{kz}\underline{K}_3\underline{K}_3 \\
&+ d_{xy}\left(\underline{K}_1\underline{K}_2 + \underline{K}_2\underline{K}_1\right) + d_{yz}\left(\underline{K}_2\underline{K}_3 + \underline{K}_3\underline{K}_2\right) + d_{zx}\left(\underline{K}_3\underline{K}_1 + \underline{K}_1\underline{K}_3\right)
\end{aligned} \tag{25}
$$

where J_{kx}, J_{ky}, and J_{kz} are the moments of inertia of K about lines passing through K_0 and parallel to $\underline{K}_1$, $\underline{K}_2$, $\underline{K}_3$, respectively, and d_{xy}, d_{yz}, and d_{zx} are the associated products of inertia.

The angular momentum of K for K_0 in N is defined as

$$ {}^{N}\underline{H}^{K/K_0} \overset{\Delta}{=} \underline{\underline{I}}^{K/K_0} \cdot {}^{N}\omega^{K} \tag{26} $$

Substituting into Equation (26) from Equations (1) and (25) yields

$$ \begin{aligned} {}^{N}\underline{H}^{K/K_0} &= J_{kx} p_K \underline{K}_1 + d_{xy} p_K \underline{K}_2 + d_{zx} p_K \underline{K}_3 + J_{ky} q_K \underline{K}_2 + d_{xy} q_K \underline{K}_1 + d_{yz} q_K \underline{K}_3 \\ &\quad + J_{kz} r_K \underline{K}_3 + d_{yz} r_K \underline{K}_2 + d_{zx} r_K \underline{K}_1 \\ &= \left(J_{kx} p_K + d_{xy} q_K + d_{zx} r_K\right)\underline{K}_1 + \left(d_{xy} p_K + J_{ky} q_K + d_{yz} r_K\right)\underline{K}_2 \\ &\quad + \left(d_{zx} p_K + d_{yz} q_K + J_{kz} r_K\right)\underline{K}_3 \end{aligned} \tag{27} $$

The system of forces exerted by B on K is assumed to be equivalent to a couple of torque, $\underline{T}^K$, given by

$$ \underline{T}^K = T_X^K \underline{K}_1 + T_Y^K \underline{K}_2 + T_Z^K \underline{K}_3 \tag{28} $$

The angular momentum principle states that (recall that K_0=A_0):

$$ \underline{T}^K = {}^{N}\frac{d}{dt}\left({}^{N}\underline{H}^{K/K_0} + {}^{N}\underline{H}^{A/A_0}\right) = {}^{N}\frac{d}{dt}\left({}^{N}\underline{H}^{K/K_0}\right) + {}^{N}\frac{d}{dt}\left({}^{N}\underline{H}^{A/A_0}\right) \tag{29} $$

where $ {}^{N}\frac{d}{dt} $ denotes time-differentiation in N.

Since ${}^{N}\underline{H}^{K/K0}$ is expressed in terms of $\underline{K}_1$, $\underline{K}_2$, $\underline{K}_3$, it is helpful to use once again the formula relating time derivatives of a vector in two reference frames, Equation (21), which gives

$$ \underline{T}^K = {}^{K}\frac{d}{dt}\left({}^{N}\underline{H}^{K/K_0}\right) + {}^{N}\underline{\omega}^{K} \times {}^{N}\underline{H}^{K/K_0} \tag{30} $$

Substituting from Equations (27), (1), and (23) into Equation (30) leads to

$$
\begin{aligned}
\underline{T}^{K} &= {}^{K}\frac{d}{dt}\left({}^{N}\underline{H}^{K/K_0}\right)+{}^{N}\underline{\omega}^{K}\times{}^{N}\underline{H}^{K/K_0}+\frac{d}{dt}\left({}^{N}\underline{H}^{A/A_0}\right)\\
&=\left(J_{kx}\dot{p}_K+d_{xy}\dot{q}_K+d_{zx}\dot{r}_K\right)\underline{K}_1+\left(d_{xy}\dot{p}_K+J_{ky}\dot{q}_K+d_{yz}\dot{r}_K\right)\underline{K}_2+\left(d_{zx}\dot{p}_K+d_{yz}\dot{q}_K+J_{kz}\dot{r}_K\right)\underline{K}_3\\
&+\left[q_K\left(d_{zx}p_K+d_{yz}q_K+J_{kz}r_K\right)-r_K\left(d_{xy}p_K+J_{ky}q_K+d_{yz}r_K\right)\right]\underline{K}_1\\
&-\left[p_K\left(d_{zx}p_K+d_{yz}q_K+J_{kz}r_K\right)-r_K\left(J_{kx}p_K+d_{xy}q_K+d_{zx}r_K\right)\right]\underline{K}_2\\
&+\left[p_K\left(d_{xy}p_K+J_{ky}q_K+d_{yz}r_K\right)-q_K\left(J_{kx}p_K+d_{xy}q_K+d_{zx}r_K\right)\right]\underline{K}_3\\
&+\left[J_{ax}\dot{p}_A+D_{xy}\dot{q}_A+D_{zx}\dot{r}_A+D_{zx}p_Aq_A+D_{yz}q_A^2+J_{az}q_Ar_A-D_{xy}p_Ar_A-J_{ay}q_Ar_A-D_{yz}r_A^2\right]\underline{A}_1\\
&+\left[D_{xy}\dot{p}_A+J_{ay}\dot{q}_A+D_{yz}\dot{r}_A-D_{zx}p_A^2-D_{yz}p_Aq_A-J_{az}p_Ar_A+J_{ax}p_Ar_A+D_{xy}q_Ar_A+D_{zx}r_A^2\right]\underline{A}_2\\
&+\left[D_{zx}\dot{p}_A+D_{yz}\dot{q}_A+J_{az}\dot{r}_A+D_{xy}p_A^2+J_{ay}p_Aq_A+D_{yz}p_Ar_A-J_{ax}p_Aq_A-D_{xy}q_A^2-D_{zx}q_Ar_A\right]\underline{A}_3
\end{aligned}
$$

$$
\begin{aligned}
&=\left[J_{kx}\dot{p}_K+d_{xy}\dot{q}_K+d_{zx}\dot{r}_K+d_{zx}p_Kq_K+d_{yz}q_K^2+J_{kz}q_Kr_K-d_{xy}p_Kr_K-J_{ky}q_Kr_K-d_{yz}r_K^2\right]\underline{K}_1\\
&+\left[d_{xy}\dot{p}_K+J_{ky}\dot{q}_K+d_{yz}\dot{r}_K-d_{zx}p_K^2-d_{yz}p_Kq_K-J_{kz}p_Kr_K+J_{kx}p_Kr_K+d_{xy}q_Kr_K+d_{zx}r_K^2\right]\underline{K}_2\\
&+\left[d_{zx}\dot{p}_K+d_{yz}\dot{q}_K+J_{kz}\dot{r}_K+d_{xy}p_K^2+J_{ky}p_Kq_K+d_{yz}p_Kr_K-J_{kx}p_Kq_K-d_{xy}q_K^2-d_{zx}q_Kr_K\right]\underline{K}_3\\
&+\left[J_{ax}\dot{p}_A+D_{xy}\dot{q}_A+D_{zx}\dot{r}_A+D_{zx}p_Aq_A+D_{yz}q_A^2+J_{az}q_Ar_A-D_{xy}p_Ar_A-J_{ay}q_Ar_A-D_{yz}r_A^2\right]\underline{A}_1\\
&+\left[D_{xy}\dot{p}_A+J_{ay}\dot{q}_A+D_{yz}\dot{r}_A-D_{zx}p_A^2-D_{yz}p_Aq_A-J_{az}p_Ar_A+J_{ax}p_Ar_A+D_{xy}q_Ar_A+D_{zx}r_A^2\right]\underline{A}_2\\
&+\left[D_{zx}\dot{p}_A+D_{yz}\dot{q}_A+J_{az}\dot{r}_A+D_{xy}p_A^2+J_{ay}p_Aq_A+D_{yz}p_Ar_A-J_{ax}p_Aq_A-D_{xy}q_A^2-D_{zx}q_Ar_A\right]\underline{A}_3
\end{aligned}
$$

$$
=T_X^K\underline{K}_1+T_Y^K\underline{K}_2+T_Z^K\underline{K}_3
$$

(31)

The scalar equation of interest can be obtained by dot-multiplying Equation (31) by $\underline{K}_3$ (see Table 2):

$$
\begin{aligned}
T_Z^K&=\left[J_{kz}\dot{r}_K+d_{zx}\left(\dot{p}_K-q_Kr_K\right)+d_{yz}\left(\dot{q}_K+p_Kr_K\right)+d_{xy}\left(p_K^2-q_K^2\right)+p_Kq_K\left(J_{ky}-J_{kx}\right)\right]\\
&+\left[J_{ax}\dot{p}_A+D_{xy}\left(\dot{q}_A-p_Ar_A\right)+D_{zx}\left(\dot{r}_A+p_Aq_A\right)+D_{yz}\left(q_A^2-r_A^2\right)+\left(J_{az}-J_{ay}\right)q_Ar_A\right]\left(-\sin\nu_2\right)\\
&+\left[J_{az}\dot{r}_A+D_{yz}\left(\dot{q}_A+p_Ar_A\right)+D_{zx}\left(\dot{p}_A-q_Ar_A\right)+D_{xy}\left(p_A^2-q_A^2\right)+\left(J_{ay}-J_{ax}\right)p_Aq_A\right]\left(\cos\nu_2\right)
\end{aligned}
$$

(32)

Since the spacecraft is assumed to be non-rotating, then p, q, r = 0. Equations (6)-(8) become:

$$p_K = 0 \tag{33}$$

$$q_K = 0 \tag{34}$$

$$r_K = \dot{\nu}_1 \tag{35}$$

Differentiating Equations (33) - (35) gives

$$\dot{p}_K = 0 \tag{36}$$

$$\dot{q}_K = 0 \tag{37}$$

$$\dot{r}_K = \ddot{\nu}_1 \tag{38}$$

Substituting from Equations (33)-(35) into Equations (13)-(15) yields

$$p_A = -\dot{\nu}_1 \sin\nu_2 \tag{39}$$

$$q_A = \dot{\nu}_2 \tag{40}$$

$$r_A = \dot{\nu}_1 \cos\nu_2 \tag{41}$$

Differentiating Equations (39)-(41) results in

$$\dot{p}_A = -\ddot{\nu}_1 \sin\nu_2 - \dot{\nu}_1 \dot{\nu}_2 \cos\nu_2 \tag{42}$$

$$\dot{q}_A = \ddot{\nu}_2 \tag{43}$$

$$\dot{r}_A = \ddot{\nu}_1 \cos\nu_2 - \dot{\nu}_1 \dot{\nu}_2 \sin\nu_2 \tag{44}$$

Taking Equation (24) and setting the products of inertia to zero (a typical telescope gimbal is built to make them as near zero as possible) produces

$$J_{ay}\dot{q}_A = T_Y^A + (J_{az} - J_{ax})p_A r_A \tag{45}$$

Substituting from Equations (39), (41), and (43) into Equation (24) yields

$$J_{ay}\ddot{\nu}_2 = T_Y^A + (J_{az} - J_{ax})(-\dot{\nu}_1^2 \sin\nu_2 \cos\nu_2) \tag{46}$$

which is a nonlinear differential equation governing the pitch motion.

Taking Equation (32) and again setting the products of inertia to zero results in

$$\begin{aligned} T_Z^K = {} & [J_{kz}\dot{r}_K + d_{zx}(\dot{p}_K - q_K r_K) + d_{yz}(\dot{q}_K + p_K r_K) + d_{xy}(p_K^2 - q_K^2) + p_K q_K (J_{ky} - J_{kx})] \\ & + [J_{ax}\dot{p}_A + (J_{az} - J_{ay})q_A r_A](-\sin\nu_2) + [J_{az}\dot{r}_A + (J_{ay} - J_{ax})p_A q_A](\cos\nu_2) \end{aligned} \tag{47}$$

Substituting from Equations (33)-(38) into Equation (47) gives

$$T_Z^K = (J_{kz}\ddot{\nu}_1) + [J_{ax}\dot{p}_A + (J_{az} - J_{ay})q_A r_A](-\sin\nu_2) + [J_{az}\dot{r}_A + (J_{ay} - J_{ax})p_A q_A](\cos\nu_2) \tag{48}$$

Substituting from Equations (39)-(44) into Equation (48) produces

$$\begin{aligned} T_Z^K = {} & J_{kz}\ddot{\nu}_1 - \sin\nu_2[J_{ax}(-\ddot{\nu}_1 \sin\nu_2 - \dot{\nu}_1\dot{\nu}_2 \cos\nu_2) + (J_{az} - J_{ay})(\dot{\nu}_2)(\dot{\nu}_1 \cos\nu_2)] \\ & + \cos\nu_2[J_{az}(\ddot{\nu}_1 \cos\nu_2 - \dot{\nu}_1\dot{\nu}_2 \sin\nu_2) + (J_{ay} - J_{ax})(-\dot{\nu}_1 \sin\nu_2)(\dot{\nu}_2)] \end{aligned} \tag{49}$$

and simplification yields

$$T_Z^K = \ddot{\nu}_1(J_{kz} + J_{ax}\sin^2\nu_2 + J_{az}\cos^2\nu_2) + \dot{\nu}_1\dot{\nu}_2(\sin 2\nu_2)(J_{ax} - J_{az}) \tag{50}$$

Equations (24) and (50) are the desired nonlinear pitch and yaw equations of motion. These equations are identical to Equations (6) and (14) in Yoon and Lundberg [1].

Optimization Procedure

Nonlinear optimization problems arise in many different application areas from engineering, economics, finance, statistics, management science, and medicine [6]. The goal is to minimize some nonlinear function, such as cost or energy, often subject to constraints on the variables. For purposes of this project, a root sum of squares of the error state was used. The root sum of squares, which forces a positive value as its output, is used to avoid the optimizer picking a solution with a large negative error. In the code developed for this project, the cost function is:

$$\sqrt{\left(\psi_C - \psi_F\right)^2 + \left(\theta_C - \theta_F\right)^2 + \left(\omega_{zC} - \omega_{zF}\right)^2 + \left(\omega_{yC} - \omega_{yF}\right)^2} \tag{51}$$

where ψ_C = commanded yaw angle, ψ_F = final yaw angle, θ_C = commanded pitch angle, θ_F = final pitch angle, ω_{zC} = commanded yaw angular rate, ω_{zF} = final yaw angular rate, ω_{yC} = commanded pitch angular rate, and ω_{yF} = final pitch angular rate. The cost function stems from trying to minimize the norm of the error between the commanded and final states of the system. The cost function is the simplest way to tell how close the code gets the user to the desired angles and rates. If that function is zero, then the telescope is at exactly the commanded angles and rates. If it is not zero (which is usually the case) then there is some error. The process continues by finding the coefficients of the seventh order polynomials iteratively until this cost function is lower than some pre-defined tolerance (i.e. telescope arrives sufficiently close to the commanded state).

The optimizer used in this thesis is Matlab's built-in FMINCON function that finds the minimum of constrained nonlinear multivariable functions. Fmincon allows the user to find the optimized coefficients of the seventh order polynomial torque, given certain constraint equations.

FMINCON attempts to solve problems of the form

$$\min F(X) \text{ subject to: } A*X \leq B,\ Aeq*X = Beq \text{ (linear constraints)}$$

$$C(X) \leq 0,\ Ceq(X) = 0 \quad \text{(nonlinear constraints)}$$

$$LB \leq X \leq UB \qquad \text{(bounds)}$$

X = FMINCON(FUN,X0,A,B) starts at X0 and finds a minimum X to the function FUN, subject to the linear inequalities $A*X \leq B$. FUN accepts input X and returns a scalar function value F evaluated at X. X0 may be a scalar, vector, or matrix. In the case of this thesis, X0 is a vector.

The inputs to the code are the initial states of the two-axis gimbal, the commanded state, constraints on positions, velocity, acceleration, and torque, a given time in which to complete the maneuver, and an integration time step. The outputs of the code are the coefficients of the seventh order input torque polynomials, and the final pitch and yaw states that were obtained using those polynomial coefficients.

Constraints were added in an attempt to model the physical limitations of a real-world system. A telescope may have limits on the angles it can slew. It may physically come upon a hard stop. A motor can only output so much torque, which is represented by the torque constraint. There could be an angular acceleration limit for structural reasons. The code can accept a constraint on angular velocity, although this can be redundant with an already constrained torque and angular acceleration. The designer needs to take these real world constraints on a telescope into account to determine feasible slewing maneuvers. The constraints present in this thesis are simple upper and lower bounds on all these variables.

Results

A 2 DOF nonlinear model was created based on the 1 DOF linear model in [5]. The initial and final conditions are a specified angle and angular velocity. To compare the 2 DOF model to the 1 DOF model, the yaw moment of inertia is set artificially high. As the moment of inertia in that axis gets arbitrarily large, no response will be obtained in the yaw channel, and it is an equivalent result to having a pure 1 DOF model. Consequently, the results that follow show the output in the yaw channel is close to zero.

To show that this model produced similar results to [5], the same inputs are used:

Initial state $\quad x_{t0} = \begin{bmatrix} 5 \\ -0.0042 \end{bmatrix}$

Final state $\quad x_{tf} = \begin{bmatrix} -5 \\ -0.0042 \end{bmatrix}$

with a time-step, $d_t = 0.0001$ (s), initial time $t_0 = 0$ (s), and final time $t_f = 5$ (s). In this example, the Matlab code rotates the telescope from 5 degrees to –5 degrees, while initial and final angular velocities are the same rate at which the Earth rotates. The 2 DOF nonlinear model results are shown in Figures 2-4.

Figure 2 shows the telescope going from the initial state of 5 degrees to the commanded final state of –5 degrees in five seconds. The pitch angle remains relatively constant early in the motion, but then decreases, beginning at approximately 2 seconds, and the curve is smooth

throughout the maneuver. This is as expected because the path is a polynomial. The system cannot go to the commanded orientation early and remain constant thereafter. It has to arrive smoothly at its commanded orientation at the exact specified time and cannot be discontinuous. Figure 2 shows that model working successfully.

Note that there is no motion in the yaw direction because of an artificially high yaw gimbal moment of inertia. That was added to force a 1 DOF comparison to [5]. O'Connor's final state was

$$x_{tf} = \begin{bmatrix} -5.000000384815846 \\ -0.004199326185375 \end{bmatrix}$$

and the nonlinear, 2 DOF model's final state was

$$x_{tf} = \begin{bmatrix} -5.000053735424057 \\ -0.004054072501210 \end{bmatrix}.$$

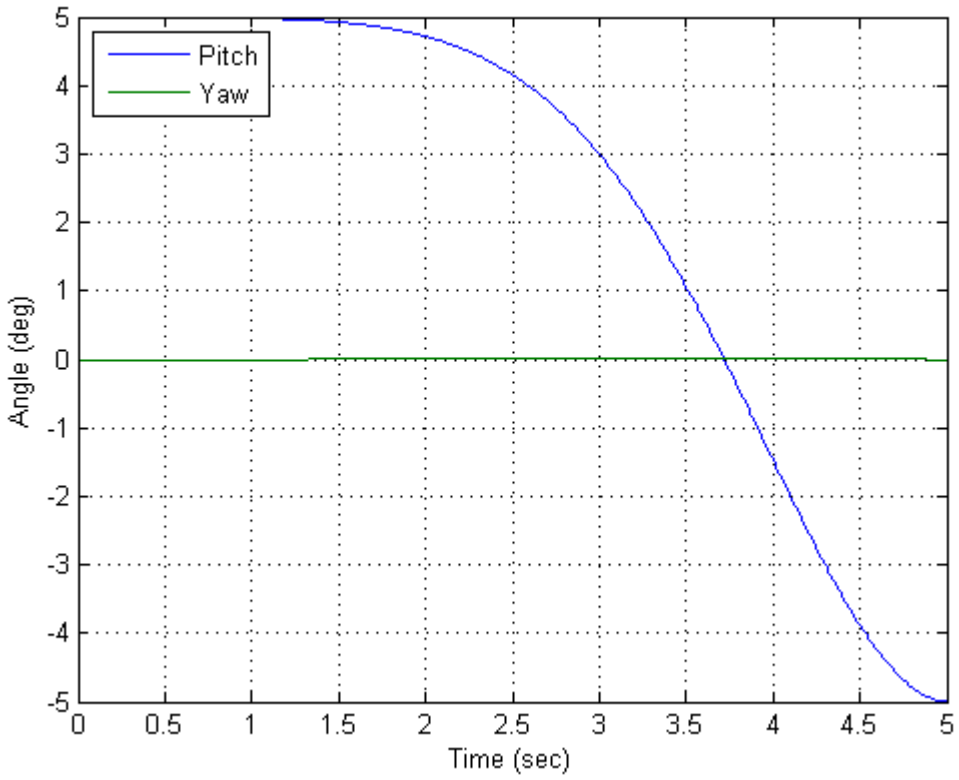

Figure 2: Angular Position vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities

Figure 3 shows the telescope speeding up and then slowing down to its commanded state. This result is expected because the commanded angle is negative, requiring the system to produce a negative angular velocity, and then the system begins to slow to reach its final state with no overshoot. There will not be overshoot at the end of the 5 second slewing period. In a real control system, this is impossible and some overshoot would be expected and may even be desirable because it is faster to command the system to its final state and then correct the overshoot after the maneuver. To recover from overshoot, another control system would be needed to make minor adjustments after this open-loop slew maneuver finished.

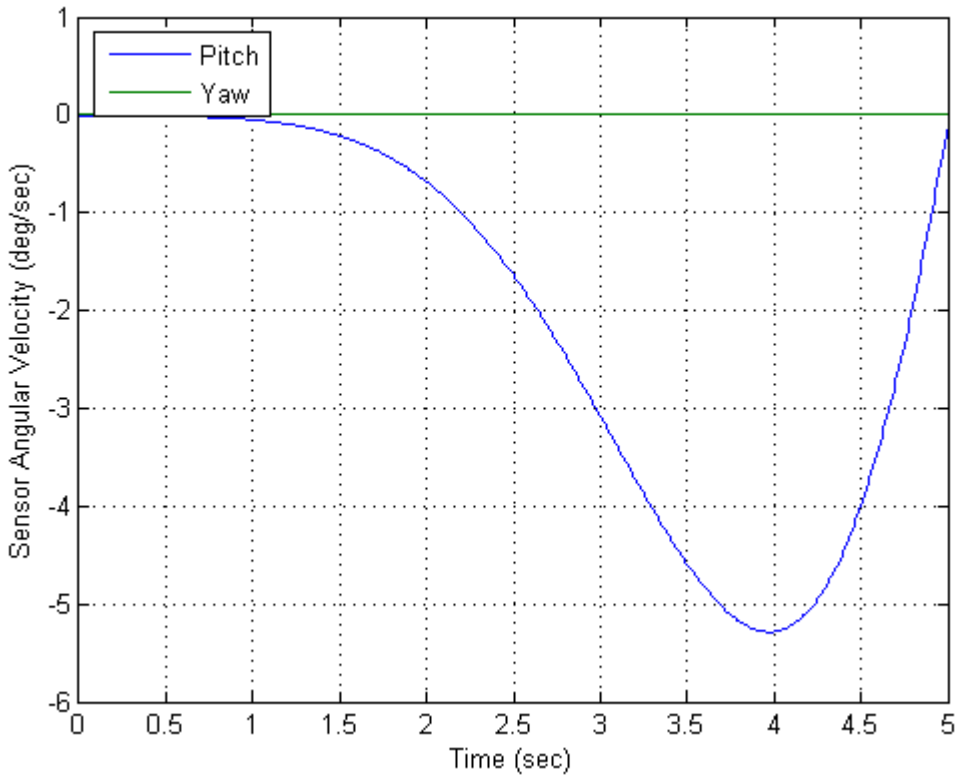

Figure 3: Angular Velocity vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities

Figure 4 depicts the output of the optimizer, which is the seventh order polynomial for gimbal torque. The gimbal is being commanded from a positive angle to a negative one. The torque also goes negative to cause the gimbal to undergo a negative angular acceleration, and then the torque switches direction to slow the gimbal as it approaches the commanded state.

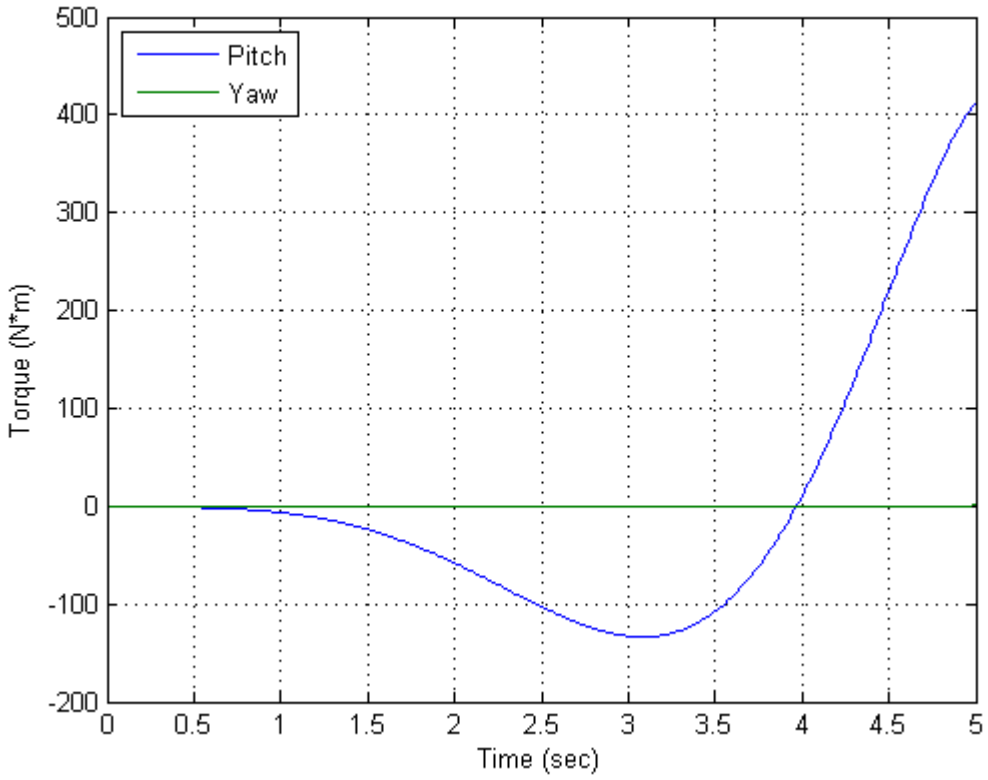

Figure 4: Simulated Input Torque vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities

It was noted (see figure 4) that the input torque the 2 DOF nonlinear model produced was approximately 400 N*m after 5 seconds, whereas the torque produced by O'Connor's model was around 1000 N*m. Further examination showed that O'Connor's model used full-state feedback, which was modeled by O'Connor as just a damping term that acts like viscous friction [5]. Normally there is more to full-state feedback than simply adding a viscous friction term to the equations of motion, but in O'Connor's case a damping term was sufficient to produce the desired system response. The absence of this damping term was causing the discrepancy in simulated input torques between O'Connor's model and the 2 DOF nonlinear model. The results of adding the full-state feedback term to the 2 DOF nonlinear model are shown in Figures 5-7.

Figure 5 shows the telescope going from the same initial state to the same commanded state as in the previous example in five seconds with the damping term added. There is a slight difference in the path of the pitch angle. The previous example without the damping term has the angle crossing zero degrees at approximately 3.75 seconds and Figure 5 shows the angle crossing zero degrees at approximately 3.85 seconds. This result is expected because the damping term (i.e. friction) slows the slewing maneuver. Figure 5 shows that the damping term has a small effect on the gimbal's pitch angle.

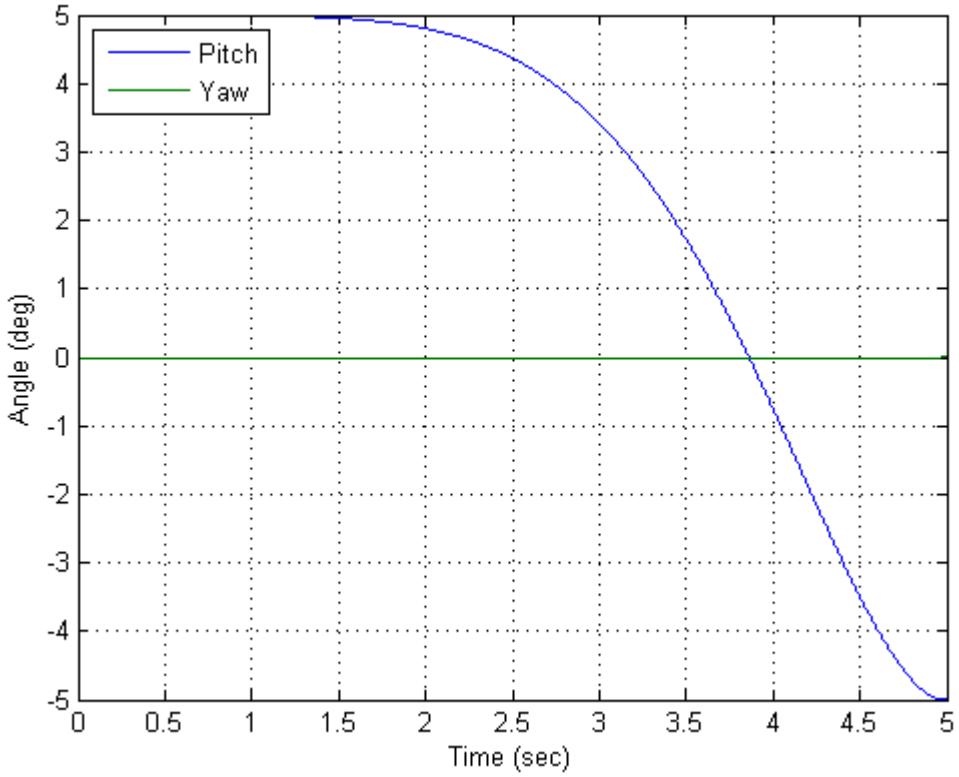

Figure 5: Angular Position vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with full state feedback

Figure 6 shows the telescope speeding up and then slowing down to its commanded state with the damping term added. The angular velocity ranges from 0 to approximately –6.5 deg/s with the minimum coming at a time of approximately 4.3 seconds. The commanded values are likely to

appear in a typical-use case study on a spacecraft. The angular rates calculated are small enough to be appropriate for a spacecraft to experience in operations.

As with the pitch angle, there is a slight difference in the angular velocity curve from the example without damping added. In the previous example, the minimum angular velocity is reached at approximately 4 seconds, whereas in Figure 6, the minimum angular velocity occurs at approximately 4.3 seconds. Additionally there is a difference in the value of the minimum angular velocity. The damping term causes an increase in the magnitude of the angular velocity. This is expected because the system has to overcome friction in order to get to the commanded state at 5 seconds. The system has to use greater angular velocity to accomplish this. Figure 6 shows a more pronounced effect of the damping term on the model than with the angle.

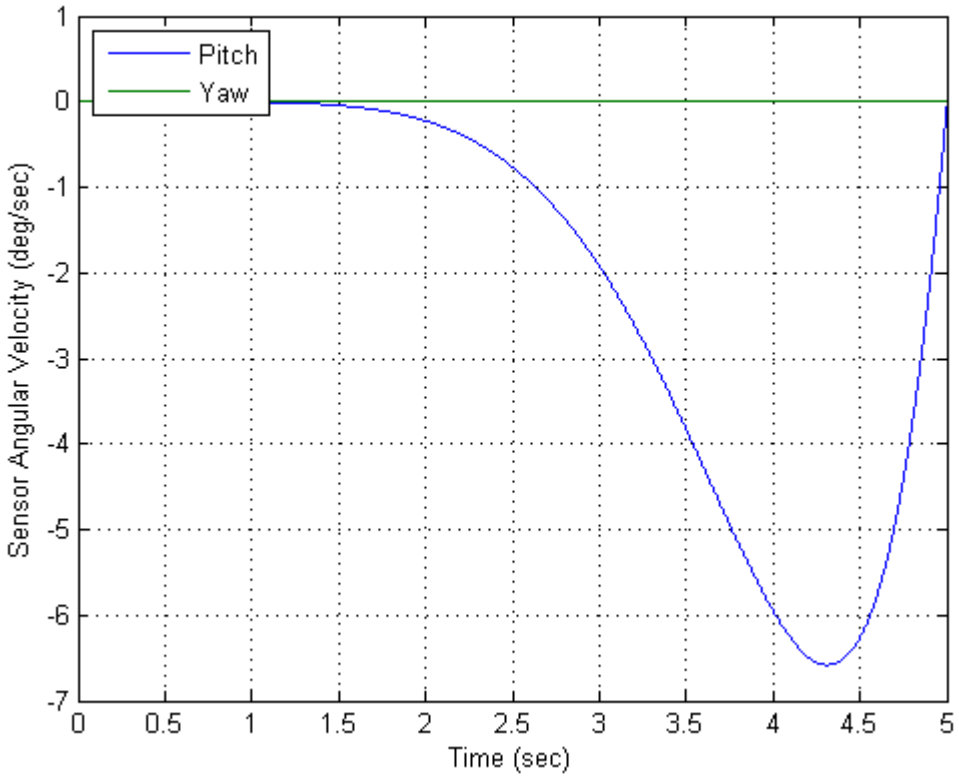

Figure 6: Sensor Angular Velocity vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with full state feedback

Figure 7 shows the output of the optimizer that is the seventh order polynomial for gimbal torque with the damping term added. The 2 DOF nonlinear model now has similar torque to the 1 DOF linear model in [5]. The gimbal is once more being commanded from a positive angle to a negative one. Correspondingly, the torque also goes negative to cause the gimbal to experience a negative angular acceleration, and then the torque switches direction to slow the gimbal as it approaches the commanded state.

With the damping term added, the torque is now greater in both directions. In the previous example without the damping term, the minimum torque was approximately –150 N*m and the maximum torque reached was approximately 400 N*m. With the damping added, the minimum torque reached was –650 N*m and the maximum was approximately 1000 N*m. These numbers now closely match the results obtained with the model in [5]. The resulting larger torque is expected because the system needs it to overcome the torque due to friction.

The close agreement between the results in [5] and the results shown in Figures 5-7 provide good evidence that the code developed for this thesis is accurate. To illustrate that the 2 DOF nonlinear model is a higher fidelity model than the model in [5], the input torque (Tau_max) was next constrained to 700 N*m, while the other inputs remain unchanged. The results are shown in Figures 8-10.

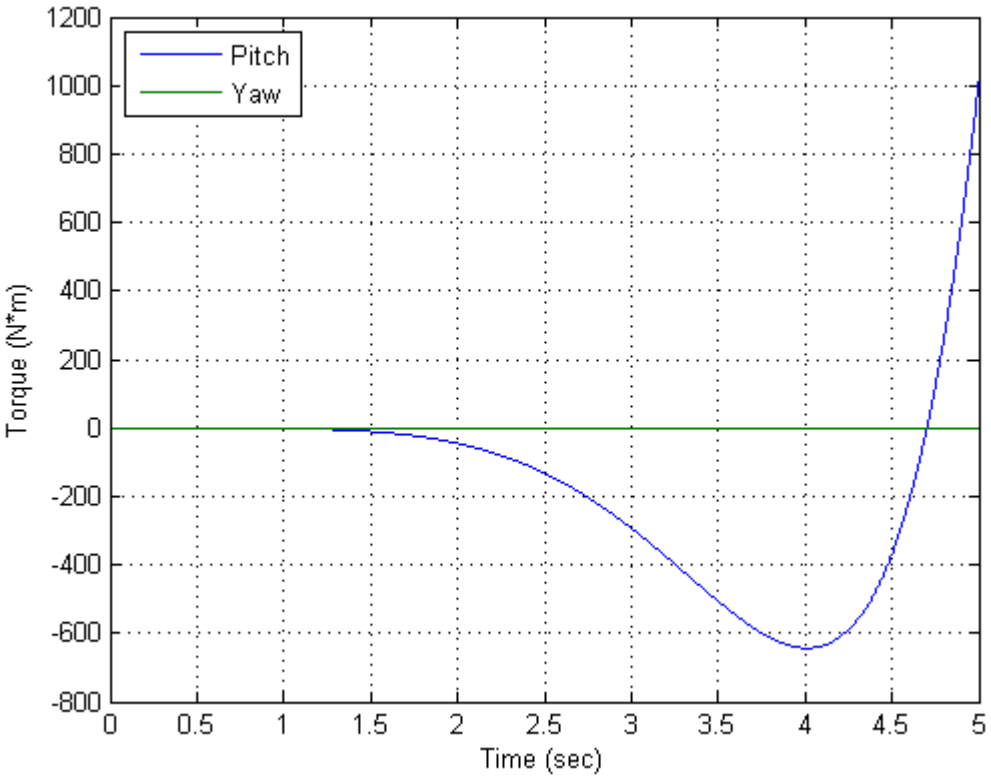

Figure 7: Torque vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with full state feedback

Figure 8 shows the telescope going from the initial state used previously to the same commanded state in five seconds with a constraint placed on the input torque. The time that the angle departs from 5 degrees is occurring sooner (at 1 second vs. 1.75 seconds in Figure 5) and the time the telescope crosses zero degrees is occurring slightly sooner (at approximately 3.8 seconds vs. just after 4 seconds in Figure 5). This result is expected because with less actuator capability the only way to achieve the commanded state in the same amount of time is to start the maneuver sooner. Figure 8 shows that the effect of the constrained torque on the pitch angle is minor.

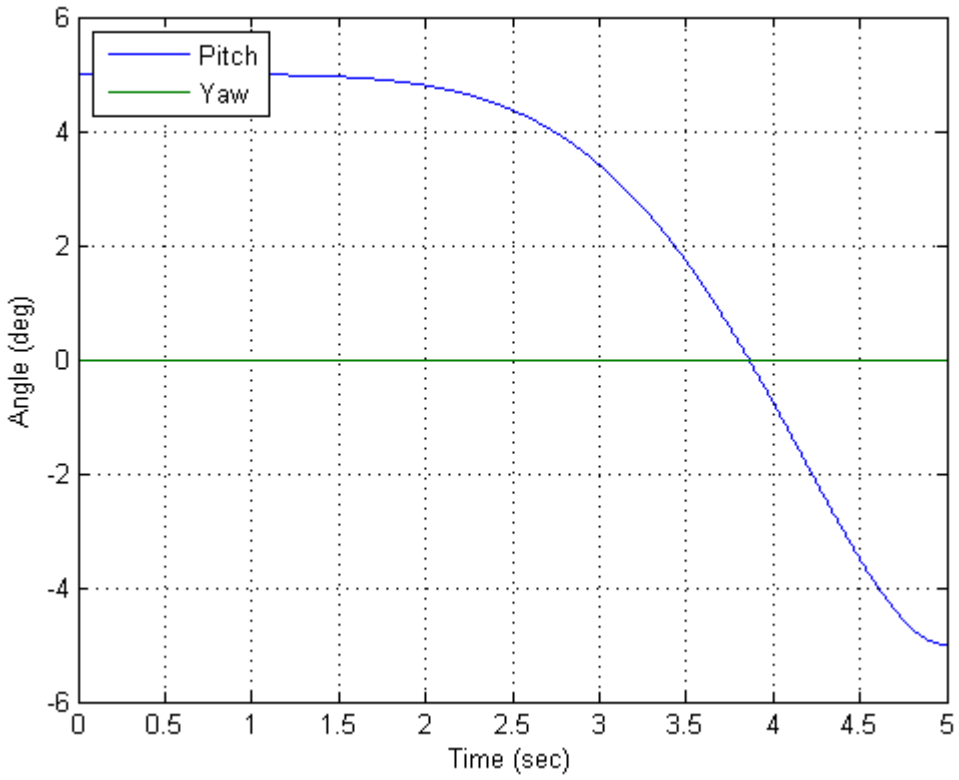

Figure 8: Angular Position vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with constrained Tau_max

Figure 9 shows the telescope speeding up and then slowing down to its commanded state with a constraint placed on the input torque. The angular velocity departs from zero degrees/sec sooner and reaches minimum sooner than in the previous examples (Figure 6). This earlier response is similar to that observed in the pitch angle of Figure 8 vs. its previous response seen in Figure 5. Additionally, the magnitude of the angular velocity is decreased due to decreased available torque (see –5.5 degrees/sec vs. –6.5 degrees/sec in Figure 6). This result is expected because decreased available torque does not permit the system to move as fast as when the torque is unconstrained. Figure 9 shows a more noticeable effect on the model than Figure 8.

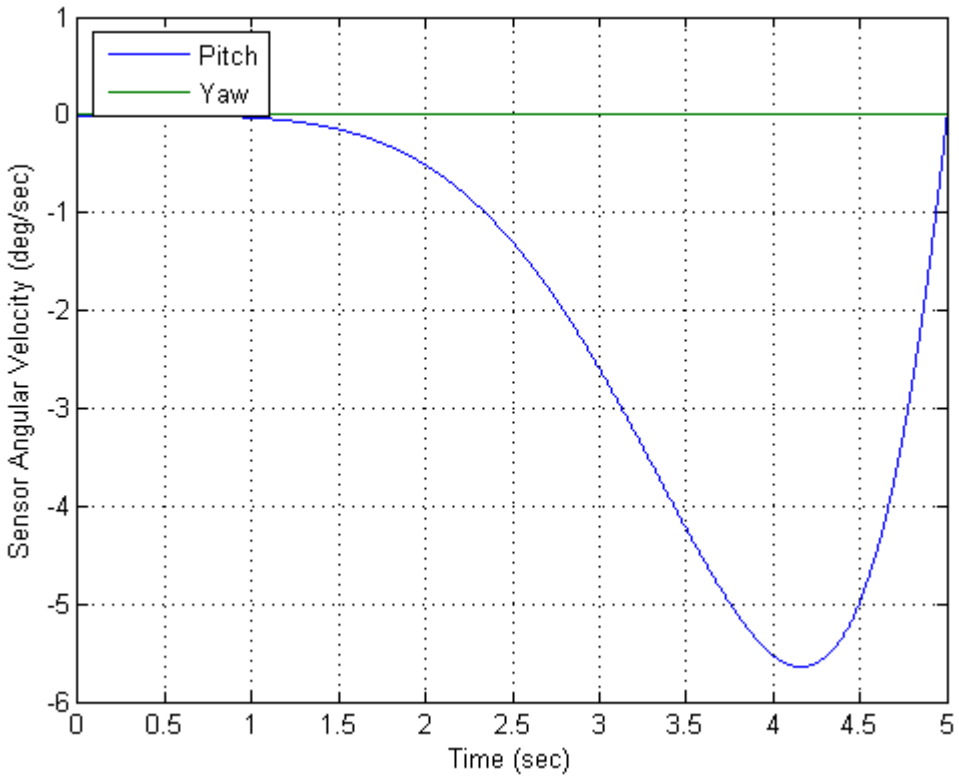

Figure 9: Sensor Angular Velocity vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with constrained Tau_max

Figure 10 shows that an input torque below the constrained Tau_max (700 N*m) is found, and that the system begins pushing approximately 0.5 seconds sooner than in Figure 7. This example illustrates that the 2 DOF nonlinear model is of higher fidelity than the one put forth in [5] because the linear 1 DOF model does not handle constraints. The Least Squares Method presented in [5] permits arbitrarily large input torques (which may not be practical or physically possible) and provides no way to limit orientation or angular velocity in the 1 DOF linear model. There are typically limitations on angular acceleration for structural reasons. Orientation might also need to be constrained if there is a hard stop on the gimbals.

A typical telescope needs to be able to point with two degrees of freedom (azimuth and elevation) so a 1 DOF model is not as useful as a 2 DOF model in the case of pointing a telescope in space. Unfortunately 2 DOF models are nonlinear by the nature of the nonlinearity of the angular momentum equation except for certain trivial cases like perfectly spherical bodies. Additionally, constraints on a system are a form of nonlinearity, and any real world system is going to have constraints. The model presented in [5] may not be useful for real world spacecraft applications. The 2 DOF nonlinear Matlab optimizer will inform the user if it cannot find a solution for the given constraints.

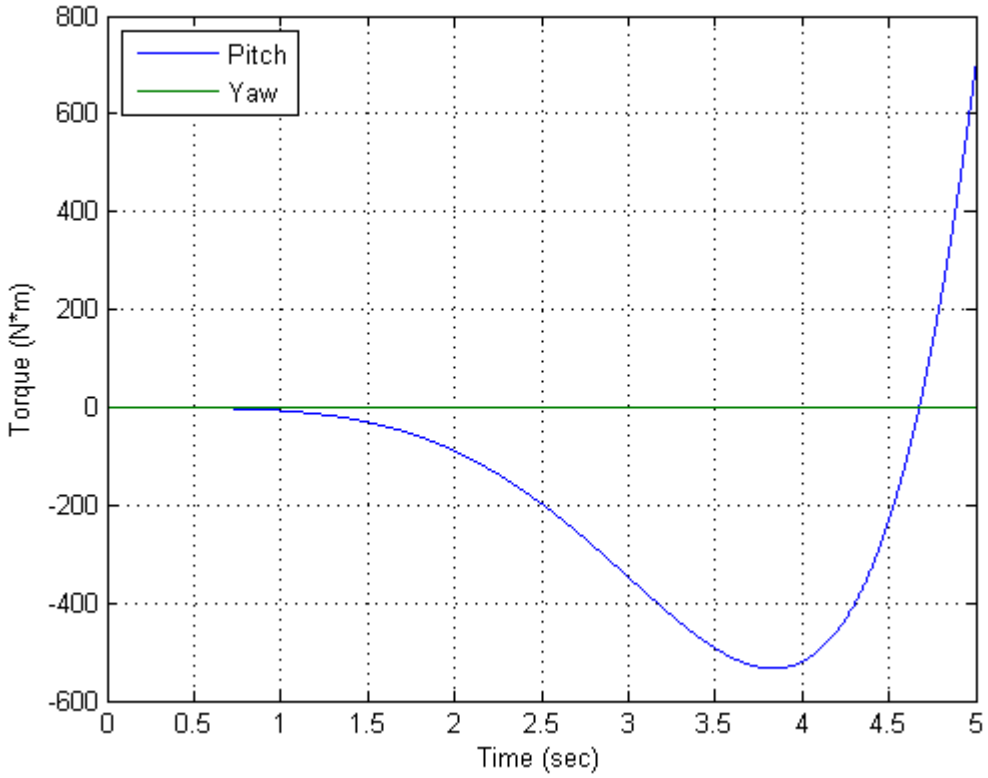

Figure 10: Torque vs. Time for slew rates of –5 to 5 degrees and earth-rate velocities with constrained Tau_max

2 DOF example using full constrained optimized solution

The final case is a 2 DOF example that would show a fully 2 DOF nonlinear, constrained, optimized solution. The data presented are not necessarily representative of an actual spacecraft slew maneuver. The initial state for pitch is 5 degrees at a rate of 0 degrees/second; yaw is –5 degrees at a rate of 0 degrees/second. The commanded state for pitch is –5 degrees at a rate of 0 degrees/second and yaw is 5 degrees at a rate of 0 degrees/second, with a time-step, $d_t = 0.001$ (s), initial time $t_0 = 0$ (s), and final time $t_f = 5$ (s). The results are shown in Figures 11-14.

Figure 11 shows both yaw and pitch channels of the telescope going from the initial state to the commanded state in five seconds. Since the telescope's yaw and pitch commands were in opposite directions from one another, their plot paths are nearly the negatives of each other. It is important to note that since the moments of inertia in pitch and yaw are different from each other (pitch is 2400 kg*m^2 and yaw is 2000 kg*m^2) that these two plots are not exact negatives of each other. The pitch result is exactly as in Figure 8 since the pitch gimbal is the inner gimbal and is not dependent on yaw motion. The yaw result is expected since the telescope moves smoothly from –5 degrees to 5 degrees and is nearly the opposite of the pitch curve. Figure 11 shows the model working successfully in two dimensions.

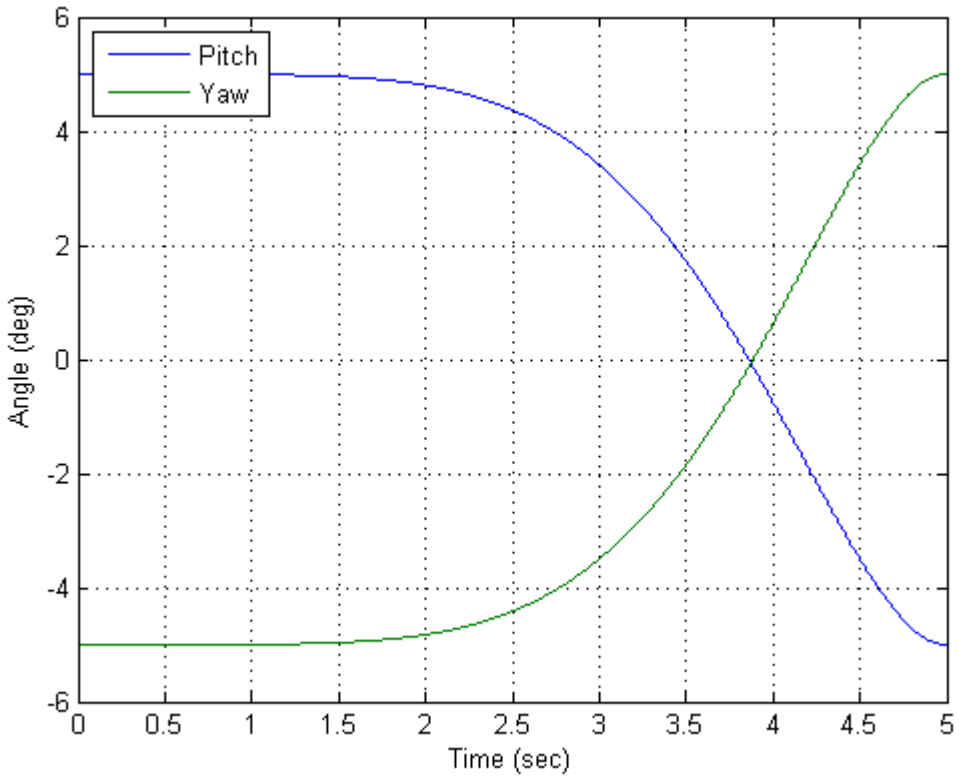

Figure 11: Yaw and Pitch Angular Position vs. Time for the fully constrained 2 DOF example

Figure 12 is an enlarged section of Figure 11 where the pitch and yaw curves cross the x-axis. Figure 12 shows that the pitch and yaw curves cross the x-axis at different points, indicating they are two different and distinct curves, not merely the same curve reflected about the x-axis.

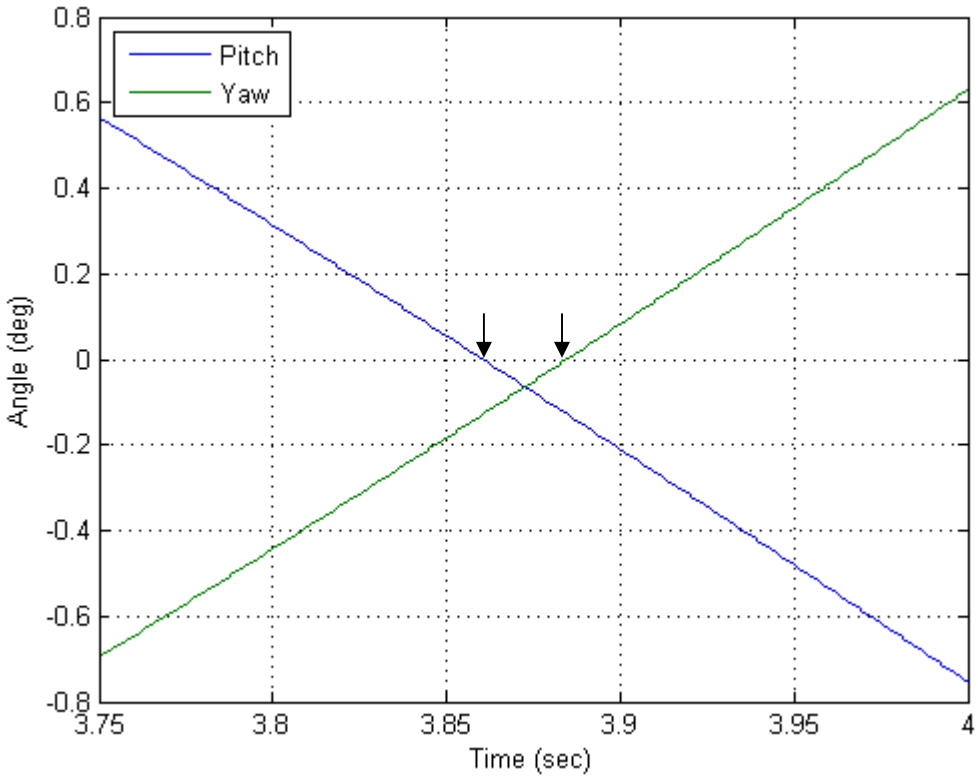

Figure 12: Detailed view of Yaw and Pitch Angular Position vs. Time for the fully constrained 2 DOF example

Figure 13 shows the pitch and yaw rates increasing and then decreasing to their commanded states. As in Figure 11, yaw and pitch rates are nearly reflections of each other due to comparable moments of inertia in both pitch and yaw axes. The pitch result is identical to Figure 9, as none of the conditions have changed. The yaw result is expected because of the similar moments of inertia, initial conditions, and final commanded conditions of pitch. Figure 13 further shows the model working successfully in two dimensions.

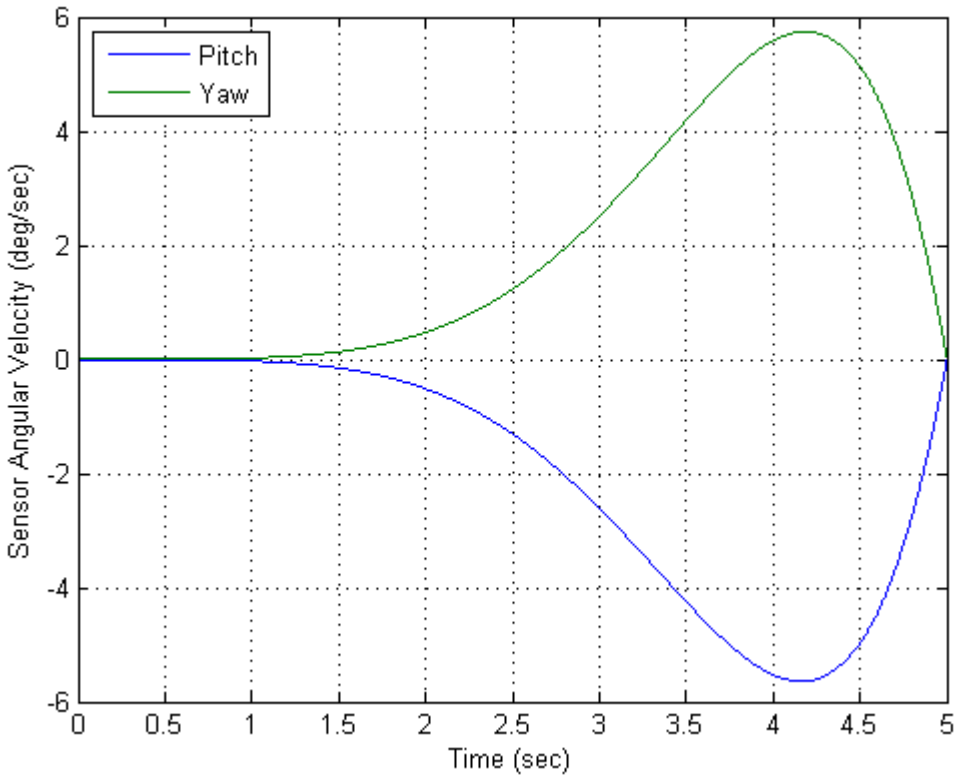

Figure 13: Yaw and Pitch Sensor Angular Velocity vs. Time for the fully constrained 2 DOF example

Figure 14 shows the pitch and yaw seventh order polynomial input torques both constrained to 700 N*m. As in Figure 13, the yaw and pitch curves are nearly reflections of each other due to their very similar moments of inertia in both axes. However in Figure 14, the differences are more apparent with the pitch torque going to a minimum of –550 N*m and the yaw going to a maximum of approximately 450 N*m. Additionally, the pitch torque goes to a maximum of 700 N*m and the yaw torque only reaches a minimum of just over –600 N*m at 5 seconds.

The pitch result is identical to Figure 10 as none of the conditions have changed. The lower yaw torque result compared to pitch is expected because of the lower yaw moment of inertia compared to pitch. Figure 14 shows the nonlinear constrained 2 DOF model working successfully.

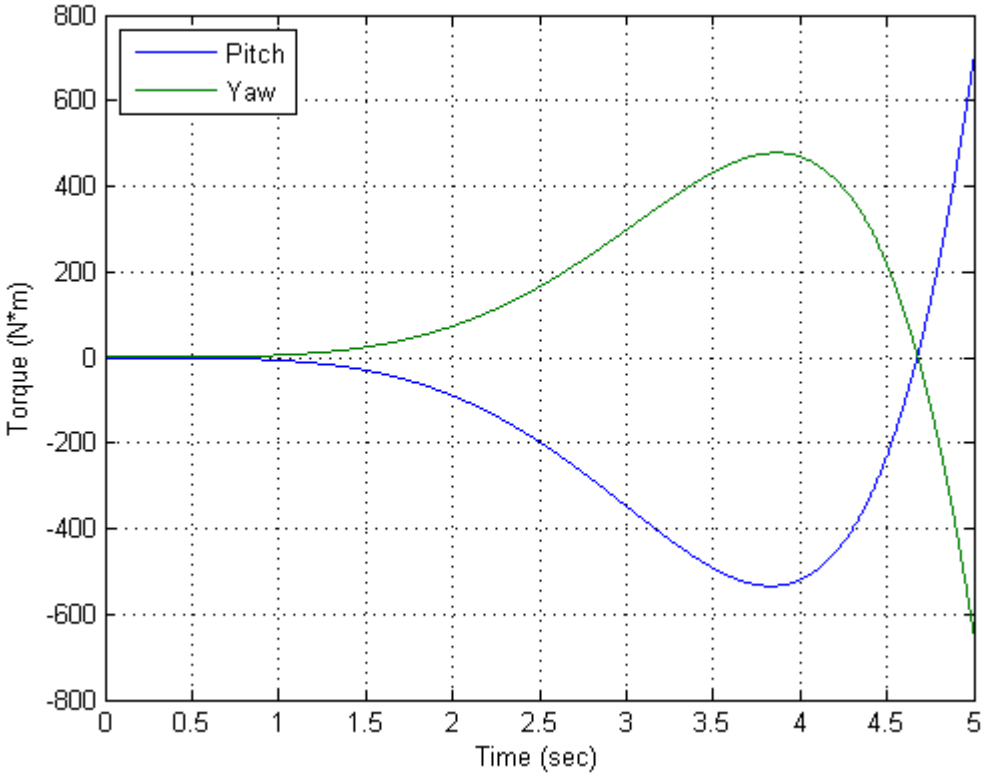

Figure 14: Yaw and Pitch Torque vs. Time for the fully constrained 2 DOF example

Conclusion

The goal of the thesis was to develop a technique to optimally slew a telescope in 2 DOF using seventh order polynomial commands, subject to constraints, and taking into account the full nonlinear equations of motion of the two-axis gimbal system in which the telescope is mounted. Matlab's FMINCON was used to do the optimization, and the results were compared to a previously validated 1 DOF model for validation of the nonlinear, 2 DOF model. Results for a fully constrained 2 DOF slew maneuver were also shown. This thesis demonstrated that seventh order polynomial torques can be used to accurately slew a telescope in space using nonlinear equations of motion.

There are some limitations to the code developed for this thesis. The Matlab optimizer, the inner workings of which are proprietary to Mathworks, can sometimes take several minutes to several hours to run and converge to an answer. Matlab can also be unpredictable as to which problems it takes a longer amount of time to solve. Without knowing the inner workings of Matlab's optimizer, it is difficult to predict which problems Matlab will solve quickly and which ones it will take much longer to solve. This could be a problem for onboard operations if they are time-sensitive. Perhaps this time could be improved upon by developing an optimizer tailored to this particular problem.

Also, if this control system were employed on an actual telescope, there would be no Matlab onboard because Matlab is not designed for the types of computers that are onboard satellites and telescopes. Those computers do not have the same processing power as a desktop computer. In

that case one would have to code an optimizer specifically designed to run on the satellite's computer.

Another change to the optimizer would be to let it select the time required to perform the slew maneuver in addition to the seventh order polynomials that it finds. Currently the user specifies the time as an input (5 seconds for the results shown). Perhaps an actual telescope user would want to slew as fast as possible and 5 seconds would be too slow, so the code could be extended to also try to minimize that slew time. Selecting a cost function for that problem however would require additional study.

In theory this model should be useful for slewing a telescope in space; however actual testing would be necessary to validate the model and assess how sensitive it is to disturbances or errors in predicted vs. actual moments of inertia and torque. Further work could be done on improving the optimizer so that it converges to a solution more rapidly and consistently. Additionally, it should be noted that this is an open loop control system and that a closed loop system could be developed to improve accuracy and robustness in the presence of modeling errors and disturbances.

Bibliography

1.) Yoon, Sungpil and John. B. Lundberg, "Equations of Motion for a Two-Axes Gimbal System," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 37, No. 3, July 2001, pp. 1083 – 1091.

2.) Wongkamchang, Prasatporn, and Viboon Sangveraphunsiri. "Control of Inertial Stabilization Systems Using Robust Inverse Dynamic Control and Adaptive Control." *Thammasat International Journal of Science and Technology* 13.2 (2008): 20-32. Thammasat University, Thailand. Web. <http://www.tijsat.tu.ac.th/>.

3.) Quigley, Morgan, Michael A. Goodrich, Stephen Griffiths, Andrew Eldredge, and Randal W. Beard. "Target Acquisition, Localization, and Surveillance using a Fixed-Wing Mini-UAV and Gimbaled Camera." Proceedings of the IEEE International Conference on Robotics and Automation, April 2005, Barcelona,
Spain, pp. 2600–2605

4.) Lee, DongBin, Vilas K. Chitrakaran, Timothy C. Burg, Darren M. Dawson, Bin Xian , and Enver Tatlicioglu. "Integrated Control of a Remotely Operated Quadrotor UAV and Camera Unit by Fly-The-Camera Perspective." *Control and Robotics (CRB) Technical Report*, Jan. 2009, 01–39

5.) O'Connor, Cory. *Finding the Input Polynomial Coefficients Satisfying the Time-domain Analytic Solution to a Linear Dynamic System Using the Least Squares Method*. Thesis. California Polytechnic State University, 2006. Print.

6.) Ruszczyński, Andrzej P. *Nonlinear Optimization*. Princeton, NJ: Princeton UP, 2006. Print.

7.) Peronto, Andy. "Re: Full State Feedback Term Added to Code." Message to the author. 27 Jan. 2012. E-mail.

Appendix A

The Matlab code developed for this thesis follows below. It consists of four functions: optimizePolynomial.m, myfun.m, mycon.m, and propagate.m.

```
function optimizePolynomial

% Start with an initial guess at a vector of polynomial coefficients.
% X is the vector of 7th order polynomial torque coefficients.
% There are two input torques, one for pitch and one for yaw, so there
% needs to be 2 7th order polynomials.
% Let the pitch torque, Ty, be:
% Ty = a + b*t + c*t^2 + d*t^3 + e*t^4 + f*t^5 + g*t^6 + h*t^7
% Let the yaw torque, Tz, be:
% Tz = i + j*t + k*t^2 + l*t^3 + m*t^4 + n*t^5 + o*t^6 + p*t^7
% Thus, X = [a b c d e f g h i j k l m n o p]
X = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';

% Set some parameters for the simulation.
% Define the state of the slewing motion as:
% x = [pitch; yaw; pitchdot; yawdot] or
% x = [nu2; nu1; nu2dot; nu1dot*cos(nu2)]
% The commanded final state is xc.
% The initial state in degrees and degrees per second:
% 1-D example: xi = [5 0 -360/60/60/24 0]';
xi = [5 -5 0 0]';
% And the commanded state in degrees and degrees per second:
xc = [-5 5 0 0]';
```

```
% Convert to radians and radians per second

xi = deg2rad(xi);

xc = deg2rad(xc);


% Define the initial and final times, and the time step of the
simulation in seconds.

t0 = 0;

tf = 5;

dt = 0.001;


% Define the moments of inertia in kg*m^2.

Jax = 100;

Jay = 2400;

Jaz = 100;

Jkx = 0; % Drops out of equations

Jky = 0; % Drops out of equations

% 1-D example: Jkz = 1000000000; arbitrarily large yaw

Jkz = 2000;


% Define viscous friction coefficients in 1/s

cy = 2;

cz = 2;


% Define constraints.

% Maximum torque in N*m.

Tau_max = 700;

% Max angle displacement in degrees.

Theta_max = deg2rad(60);

% Max angular velocity in degrees/second.

Omega_max = deg2rad(7);
```

```
% Max angular acceleration in degrees/second^2.

Alpha_max = deg2rad(30);


% Call Matlab's fmincon optimizer to minimize the cost function.

options = optimset('fmincon');

options.Algorithm = 'active-set';

options.Display = 'iter';

options.MaxFunEvals = 10000;

options.MaxIter = 10000;

options.TolFun = 1e-7;

options.TolX = 1e-7;

[Xopt,Fval] = fmincon(@(X)
myfun(X,xi,xc,t0,tf,dt,Jax,Jay,Jaz,Jkx,Jky,Jkz,cy,cz),X,[],[],[],[],[],
[],@(X)
mycon(X,xi,Tau_max,Theta_max,Omega_max,Alpha_max,t0,tf,dt,Jax,Jay,Jaz,J
kx,Jky,Jkz,cy,cz),options);


% Print out the results.

disp('The minimized function value is:');

disp(Fval);

disp('The input that minimizes the function is:');

disp(Xopt);


% Make some plots.

% Declare vectors.

t = (t0:dt:tf);

nTimes = length(t);

nu1 = zeros(nTimes,1);

nu2 = zeros(nTimes,1);

nu1dot = zeros(nTimes,1);

nu2dot = zeros(nTimes,1);

qa = zeros(nTimes,1);
```

```
ra = zeros(nTimes,1);

nu1dotdot = zeros(nTimes,1);

nu2dotdot = zeros(nTimes,1);

qadot = zeros(nTimes,1);

radot = zeros(nTimes,1);


% Set the torques based off of the output from the optimizer.

Ty = Xopt(1) + Xopt(2)*t + Xopt(3)*t.^2 + Xopt(4)*t.^3 + Xopt(5)*t.^4 +
Xopt(6)*t.^5 + Xopt(7)*t.^6 + Xopt(8)*t.^7;

Tz = Xopt(9) + Xopt(10)*t + Xopt(11)*t.^2 + Xopt(12)*t.^3 +
Xopt(13)*t.^4 + Xopt(14)*t.^5 + Xopt(15)*t.^6 + Xopt(16)*t.^7;


% Initialize the first data point.

x = xi;


% With this input torque, propagate everything from t = t0 to t = tf.

% This is the same code as propagate.m except now it's storing all of
the

% state information into vectors.

for i=1:nTimes

  % x = [pitch; yaw; pitchdot; yawdot]

  nu2(i) = x(1);

  nu1(i) = x(2);

  nu2dot(i) = x(3);

  nu1dot(i) = x(4)/cos(x(1));

  qa(i) = x(3);

  ra(i) = x(4);


  % Calculate the gimbal angular accelerations, nu1dotdot and
nu2dotdot,

  % and sensor angular accelerations, qadot and radot, from that paper
as
```

```
  % given in symbolic_project_equations.m.

  qadot(i) = 1/Jay*(Ty(i)+(Jax-
Jaz)*nu1dot(i)^2*sin(nu2(i))*cos(nu2(i))) - cy*nu2dot(i);

  nu2dotdot(i) = qadot(i);

  radot(i) = 1/(Jkz+Jax*sin(nu2(i))^2+Jaz*cos(nu2(i))^2)*(Tz(i)-
nu2dot(i)*(Jax-Jaz)*nu1dot(i)*sin(2*nu2(i)))*cos(nu2(i)) -
nu1dot(i)*nu2dot(i)*sin(nu2(i)) - cz*nu1dot(i);

  nu1dotdot(i) = 1/(Jkz+Jax*sin(nu2(i))^2+Jaz*cos(nu2(i))^2)*(Tz(i)-
nu2dot(i)*(Jax-Jaz)*nu1dot(i)*sin(2*nu2(i))) - cz*nu1dot(i);



  % Update the pointing states.

  x = x + [.5*nu2dotdot(i)*dt^2 + nu2dot(i)*dt; .5*nu1dotdot(i)*dt^2 +
nu1dot(i)*dt; qadot(i)*dt; radot(i)*dt];

end



% Plot the torques.

figure;

plot(t,Ty,t,Tz);

xlabel('Time (sec)');

ylabel('Torque (N*m)');

legend('Pitch','Yaw','Location','NorthWest');

grid on;



% Plot the angles.

figure;

plot(t,nu2*180/pi,t,nu1*180/pi);

xlabel('Time (sec)');

ylabel('Angle (deg)');

legend('Pitch','Yaw','Location','NorthWest');

grid on;



% Plot the gimbal angular velocities.
```

```
figure;
plot(t,nu2dot*180/pi,t,nu1dot*180/pi);
xlabel('Time (sec)');
ylabel('Gimbal Angular Velocity (deg/sec)');
legend('Pitch','Yaw','Location','NorthWest');
grid on;

% Plot the sensor angular velocities.
figure;
plot(t,qa*180/pi,t,ra*180/pi);
xlabel('Time (sec)');
ylabel('Sensor Angular Velocity (deg/sec)');
legend('Pitch','Yaw','Location','NorthWest');
grid on;

% Plot the gimbal angular accelerations.
figure;
plot(t,nu2dotdot*180/pi,t,nu1dotdot*180/pi);
xlabel('Time (sec)');
ylabel('Gimbal Angular Acceleration (deg/sec^2)');
legend('Pitch','Yaw','Location','NorthWest');
grid on;

% Plot the sensor angular accelerations.
figure;
plot(t,qadot*180/pi,t,radot*180/pi);
xlabel('Time (sec)');
ylabel('Sensor Angular Acceleration (deg/sec^2)');
legend('Pitch','Yaw','Location','NorthWest');
grid on;
```

```
% Print out some more results.
disp('The initial state was:');
disp(xi*180/pi);
disp('The commanded state was:');
disp(xc*180/pi);
disp('The final state was:');
disp(x*180/pi);



function F = myfun(X,xi,xc,t0,tf,dt,Jax,Jay,Jaz,Jkx,Jky,Jkz,cy,cz)

% X is the vector of 7th order polynomial torque coefficients.
% There are two input torques, one for pitch and one for yaw, so there
% needs to be 2 7th order polynomials.
% Let the pitch torque, Ty, be:
% Ty = a + b*t + c*t^2 + d*t^3 + e*t^4 + f*t^5 + g*t^6 + h*t^7
% Let the yaw torque, Tz, be:
% Tz = i + j*t + k*t^2 + l*t^3 + m*t^4 + n*t^5 + o*t^6 + p*t^7
% Thus, X = [a b c d e f g h i j k l m n o p]

% F is the norm of the difference between some commanded final state
and
% the actual final state of the slewing motion.
% Define the state of the slewing motion as:
% x = [yaw; pitch; yawdot; pitchdot]
% The commanded final state is xc.
% The initial time is t0 and the final time is tf.
% dt is the time step of the simulation in seconds.
```

```
% Propogate the pointing state of the system forward to the final time.
xf = propagate(X,xi,Jax,Jay,Jaz,Jkx,Jky,Jkz,cy,cz,t0,tf,dt);


% Return F.
F = norm(xf-xc);




function [C,Ceq] =
mycon(X,xi,Tau_max,Theta_max,Omega_max,Alpha_max,t0,tf,dt,Jax,Jay,Jaz,J
kx,Jky,Jkz,cy,cz)


% X is the vector of 7th order polynomial torque coefficients.
% There are two input torques, one for pitch and one for yaw, so there
% needs to be 2 7th order polynomials.
% Let the pitch torque, Ty, be:
% Ty = a + b*t + c*t^2 + d*t^3 + e*t^4 + f*t^5 + g*t^6 + h*t^7
% Let the yaw torque, Tz, be:
% Tz = i + j*t + k*t^2 + l*t^3 + m*t^4 + n*t^5 + o*t^6 + p*t^7
% Thus, X = [a b c d e f g h i j k l m n o p]


% The initial time is t0 and the final time is tf.
% dt is the time step of the simulation in seconds.


% C is the nonlinear constraint function of X that must be less than or
% equal to zero.
% Ceq is the nonlinear equality constraint Ceq(X) = 0.


% Get the vector of times.
t = (t0:dt:tf);
```

```
nTimes = length(t);

nu1 = zeros(1,nTimes);

nu2 = zeros(1,nTimes);

nu1dot = zeros(1,nTimes);

nu2dot = zeros(1,nTimes);

nu1dotdot = zeros(1,nTimes);

nu2dotdot = zeros(1,nTimes);


% Compute the pitch and yaw torques.

Ty = X(1) + X(2)*t + X(3)*t.^2 + X(4)*t.^3 + X(5)*t.^4 + X(6)*t.^5 +
X(7)*t.^6 + X(8)*t.^7;

Tz = X(9) + X(10)*t + X(11)*t.^2 + X(12)*t.^3 + X(13)*t.^4 + X(14)*t.^5
+ X(15)*t.^6 + X(16)*t.^7;


% Initialize the first data point.

x = xi;


% With this input torque, propagate everything from t = t0 to t = tf.

% This is the same code as propagate.m except now it's storing all of
the

% state information into vectors.

for i=1:nTimes

  % x = [pitch; yaw; pitchdot; yawdot]

  nu2(i) = x(1);

  nu1(i) = x(2);

  nu2dot(i) = x(3);

  nu1dot(i) = x(4)/cos(x(1));


  % Calculate the gimbal angular accelerations, nu1dotdot and
nu2dotdot,

  % and sensor angular accelerations, qadot and radot, from that paper
as
```

```
  % given in symbolic_project_equations.m.

  qadot = 1/Jay*(Ty(i)+(Jax-Jaz)*nu1dot(i)^2*sin(nu2(i))*cos(nu2(i))) -
cy*nu2dot(i);

  nu2dotdot(i) = qadot;

  radot = 1/(Jkz+Jax*sin(nu2(i))^2+Jaz*cos(nu2(i))^2)*(Tz(i)-
nu2dot(i)*(Jax-Jaz)*nu1dot(i)*sin(2*nu2(i)))*cos(nu2(i)) -
nu1dot(i)*nu2dot(i)*sin(nu2(i)) - cz*nu1dot(i);

  nu1dotdot(i) = 1/(Jkz+Jax*sin(nu2(i))^2+Jaz*cos(nu2(i))^2)*(Tz(i)-
nu2dot(i)*(Jax-Jaz)*nu1dot(i)*sin(2*nu2(i))) - cz*nu1dot(i);



  % Update the pointing states.

  x = x + [.5*nu2dotdot(i)*dt^2 + nu2dot(i)*dt; .5*nu1dotdot(i)*dt^2 +
nu1dot(i)*dt; qadot*dt; radot*dt];

end



% Return C.

C = [Ty-Tau_max;Tz-Tau_max;-Ty-Tau_max;-Tz-Tau_max;nu2-Theta_max;nu1-
Theta_max;-nu2-Theta_max;-nu1-Theta_max;nu2dot-Omega_max;nu1dot-
Omega_max;-nu2dot-Omega_max;-nu1dot-Omega_max;nu2dotdot-
Alpha_max;nu1dotdot-Alpha_max;-nu2dotdot-Alpha_max;-nu1dotdot-
Alpha_max]';



% Return Ceq.

Ceq = [];







function xf = propagate(X,x,Jax,Jay,Jaz,Jkx,Jky,Jkz,cy,cz,t0,tf,dt)

% This propogates the slewing motion of the telescope from an initial
time

% and initial pointing state to a final time and final pointing state
by

% marching the equations of motion forward in time.



% Begin the simulation.
```

```
time = t0;

for t=t0:dt:tf

  % x = [pitch; yaw; pitchdot; yawdot]

  nu2 = x(1);

  nu1 = x(2);

  nu2dot = x(3);

  nu1dot = x(4)/cos(x(1));

  % Define the input torques from X where X = [a b c d e f g h i j k l
m n o p]

  % Let the pitch torque, Ty, be:

  % a + b*t + c*t^2 + d*t^3 + e*t^4 + f*t^5 + g*t^6 + h*t^7

  % Let the yaw torque, Tz, be:

  % i + j*t + k*t^2 + l*t^3 + m*t^4 + n*t^5 + o*t^6 + p*t^7

  Ty = X(1) + X(2)*time + X(3)*time^2 + X(4)*time^3 + X(5)*time^4 +
X(6)*time^5 + X(7)*time^6 + X(8)*time^7;

  Tz = X(9) + X(10)*time + X(11)*time^2 + X(12)*time^3 + X(13)*time^4 +
X(14)*time^5 + X(15)*time^6 + X(16)*time^7;



  % Calculate the gimbal angular accelerations, nu1dotdot and
nu2dotdot,

  % and sensor angular accelerations, qadot and radot, from that paper
as

  % given in symbolic_project_equations.m.

  qadot = 1/Jay*(Ty+(Jax-Jaz)*nu1dot^2*sin(nu2)*cos(nu2)) - cy*nu2dot;

  nu2dotdot = qadot;

  radot = 1/(Jkz+Jax*sin(nu2)^2+Jaz*cos(nu2)^2)*(Tz-nu2dot*(Jax-
Jaz)*nu1dot*sin(2*nu2))*cos(nu2) - nu1dot*nu2dot*sin(nu2) - cz*nu1dot;

  nu1dotdot = 1/(Jkz+Jax*sin(nu2)^2+Jaz*cos(nu2)^2)*(Tz-nu2dot*(Jax-
Jaz)*nu1dot*sin(2*nu2)) - cz*nu1dot;



  % Update the pointing states.

  x = x + [.5*nu2dotdot*dt^2 + nu2dot*dt; .5*nu1dotdot*dt^2 +
nu1dot*dt; qadot*dt; radot*dt];
```

```
  % Update the time.
  time = time + dt;
end

% Set the final state to xf.
xf = x;
```