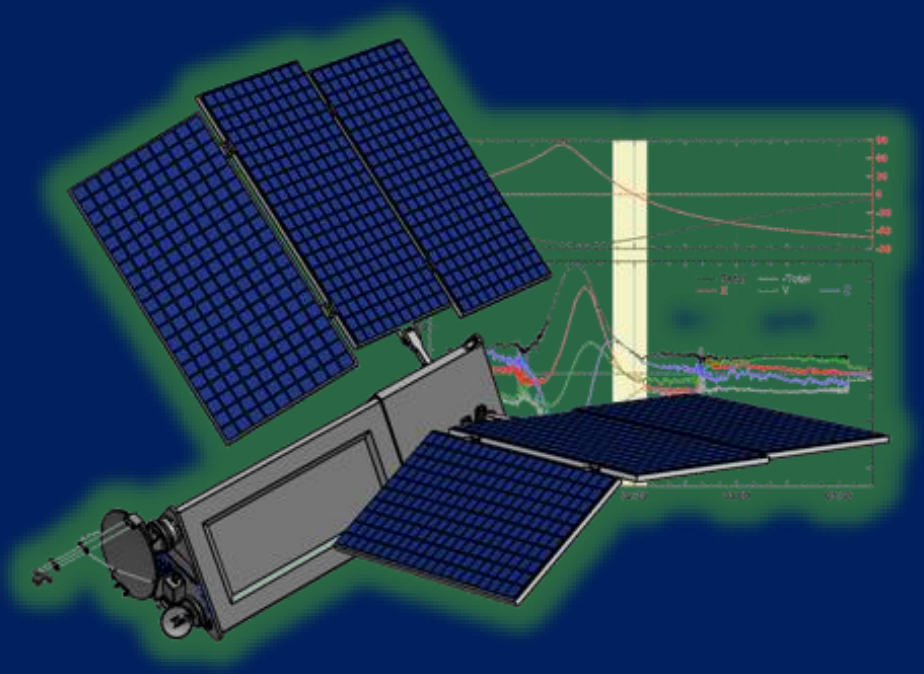


Attitude Control System Design
Bias Momentum Satellite

ARO4090-02 – Spacecraft Dynamics and Control

Garrett Parham | Bryan Phan | Burton Yale
CALIFORNIA POLYTECHNIC UNIVERSITY, POMONA DR. NAVID NAKHJIRI

CalPolyPomona

Executive Summary

Momentum biased spacecraft are one of the many ways to keep a spacecraft's attitude under control. Using the Earth's gravity field, it allows for the passive stabilization of objects in orbit. Under specific conditions, it reduces the need for a spacecraft to have active control methods. For the case of this design, the spacecraft's momentums are biased incorrectly, causing the spacecraft to pitch out of control. In an effort to bring the spacecraft motion back to reasonable limits, active control methods will be employed.

Using state and rate sensing controllers, the spacecrafts exponentially growing motion was brought under control within the pitch, roll, and yaw directions. As well the response was kept within the limits as required by the project.

Additional fidelity was added to the controllers in terms of actuator time constants. To simulate the lag produced by the activation time of actuators in systems like ACS motors, the spacecrafts response to the motion is not 100% on time. As a result, with the systems original gains, the spacecrafts motion was drastically slowed. By introducing new gains to the system, the motion returned to its previous response.

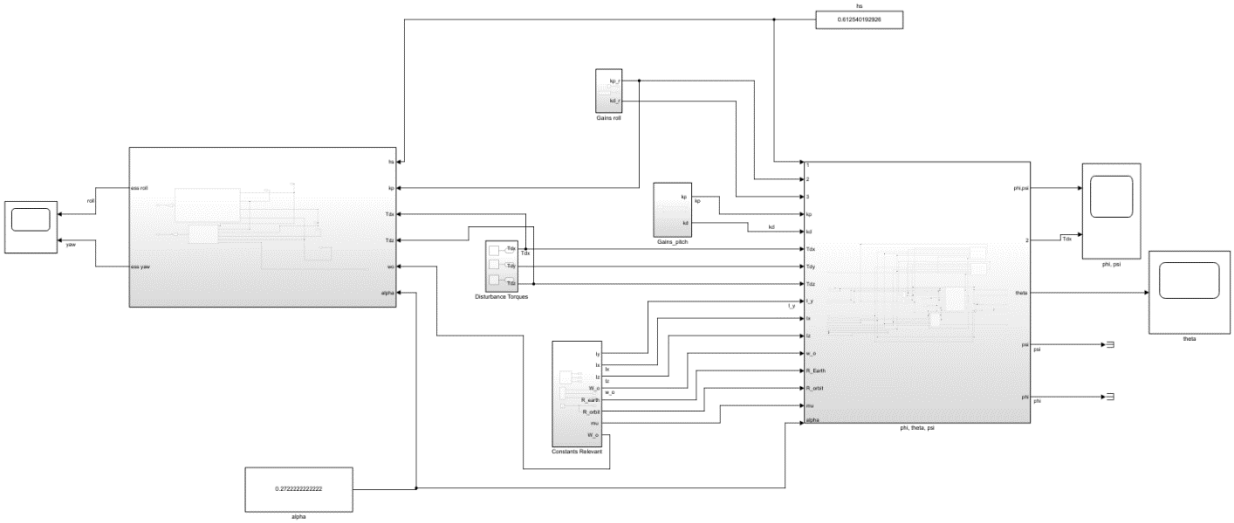
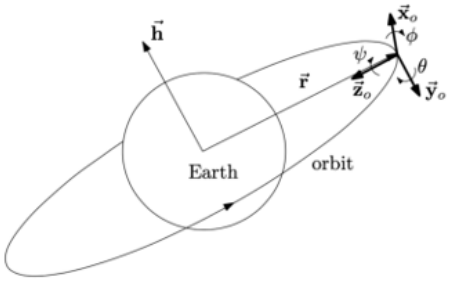


Figure 1.1-1: Simulink Model of the Spacecraft's Attitude Motion in Roll, Pitch, and Yaw

Methodology

This assessment involes the design of an attitude control system for a bias momentum satellite. The satellite is assumed to be orbiting about Earth with a momentum wheel alligned to the positive y-axis. The principal axes of the motion will be defined as x, y, and z. The z-axis is assumed to be alligned with the orbit radius vector, the x-axis is directed to be tangent from the orbit path, and the y-axis is a normal vector from the orbit plane. Roataions about the principle axes will be defined as ϕ when the roation is about the x-axis, θ when the rotation is about the y-axis, and ψ when the rotation is about the z-axis. Figure 1 below shows the denotation of the axes, rotations, and orbit parameters.



1.1-1: Spacecraft Orientation

Throughout the analysis, the direction of the center of earth from the spacecraft is known assuming that the satellite utilizes an Earth sensor. The Earth sensor is capable of sensing a rotation about the x and y-axes, but a rotation in yaw about the z-axis would not be sensed due to its allignment with the position vector of the satellite from Earth center of mass. The spacecraft is capable of being stabilized without using a yaw sensor due to the its bias momentum, where the roll and yaw are coupled. The single pitch wheel can be used to control the pitch angle by controlling its speed while the other two rotations can be controled by making small changes to the spin axis. The spacecraft and orbital parameters can be found on the next page, followed by the requirements for the control system.

Spacecraft and Orbital Parameters:

Spacecraft Principal Inertias

$I_x = 0.4$ kg·m^2

$I_y = 0.5$ kg·m^2

$I_z = 0.6$ kg·m^2

Orbit Altitude

H = 600 km

Maximum Disturbance Torques

$T_{dx,\,max} = 5 \times 10^{-6}$ N·m

$T_{dy,\,max} = 5 \times 10^{-6}$ N·m

$T_{dz,\,max} = 5 \times 10^{-6}$ N·m

Earth's Gravitational Constant

$\mu = 3.986 \times 10^5$ km^3/s^2

Earth's Radius

$R_e = 6378.1363$ km

Control Systems Requirement:

Maximum Allowable Steady-State Errors

$\phi_{ss} \leq 0.1$ °

$\theta_{ss} \leq 0.1$ °

$\psi_{ss} \leq 4.0$ °

Maximum Allowable Steady-State Errors

$\phi_{ss} \leq 0.1$ °

Damping Ratios of the Closed-Loop Poles for Roll & Yaw

Z = 0.7

Settling Time

$t_s \leq 200$ seconds

Maximum Percent Overshoot

$M_p \leq 30\%$

MATLAB Inputs:

```
clear; close all; clc;
% Setup
skew = @(a) [0 -a(3) a(2); a(3) 0 -a(1); -a(2) a(1) 0];


% Givens
    % Physical Constants
    inputs = struct('Ix', 0.4, ...
                    'Iy', 0.5, ...
                    'Iz', 0.6, ...
                    'Alt', 600, ...
                    'rE', 6378.1363, ...
                    'muE', 398600, ...
                    'hHat', [0; -1; 0], ...
                    'zeta', 0.7, ...
                    'TdMax', 5e-6);
    inputs.wo = sqrt(inputs.muE/(inputs.rE+inputs.Alt)^3);
    inputs.kg = 3*inputs.wo^2*(inputs.Ix - inputs.Iz);

    % Constraints
    const = struct('phiSS', 0.1 * pi/180, ...
                   'thSS', 0.1 * pi/180, ...
                   'psySS', 4 * pi/180, ...
                   'tSet', 200, ...
                   'Mp', 0.3);
```

Problem 1

1.1 Problem Statement

The first step is to determine whether the satellite is stable by just using the inherent stability provided the gravity gradient torque. The equations of motion were provided as:

$$[\boldsymbol{I}]\dot{\omega} + \dot{h_s}\hat{\boldsymbol{h}} + [\omega] \times \left([\boldsymbol{I}]\omega + h_s\hat{\boldsymbol{h}}\right) = \boldsymbol{T_c} + \boldsymbol{T_d} + \boldsymbol{T_g}$$

To linearize the equations, the angles and rates were assumed to have small values. Any product or powered term of either the rates or angles was considered to negate associated terms. The angular momentum about the principal axes due to the orbital motion of the spacecraft was also considered to have a much smaller value than the angular momentum of the reaction wheel. A substitution of $k_g = 3(I_x - I_z)\omega_0^2$ was used to simplify the equations. The controller was set with a value of null as a placeholder to be used later in further analysis. A small perturbation was added to the system to demonstrate the stability of the system

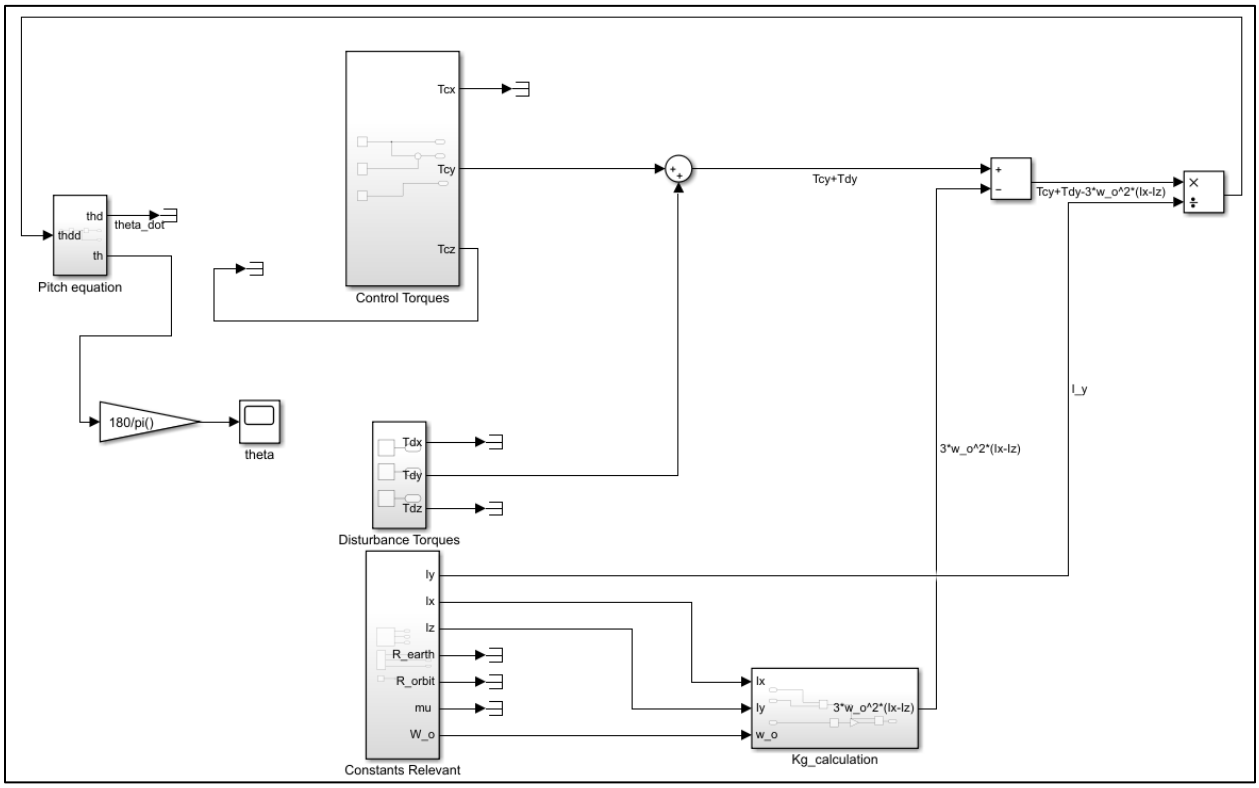
1.2 Solution Derivation

The equation of motion with the substitutions of the control system requirements and physical parameters is shown as $I_y\ddot{\theta} + 3\omega_0^2(I_x - I_z)\theta = T_{cy} + T_{dy}$ and is transformed into $I_y s^2\theta(s) + k_g\theta(s) = T_{cy} + T_{dy}$. The equation of motion, located in Appendix A, was solved using ode45 and the results were plotted to observe the behavior of the spacecraft after experiencing the perturbation.

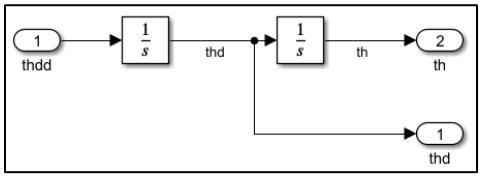
1.3 Le Code / Der Simulink

```
% Problem Gains
gains.p1.kp = 0;
gains.p1.kd = 0;


% Equations of Motion Inputs
eomOpts.p1.stepTime = 0;
eomOpts.p1.Td = 0;
eomOpts.p1.thc = 0;
eomOpts.p1.tSpan = [0 2*pi*sqrt((inputs.rE+inputs.Alt)^3/inputs.muE)];
eomOpts.p1.IC = [0.001; 0];


% Defining EoMs
EoM_ = @(t, x) EoMp(t, x, inputs, eomOpts.p1, gains.p1);



% Setup for ODE45
[t, Y] = ode45(EoM_, eomOpts.p1.tSpan, eomOpts.p1.IC);

    % ODE Cleanup
```

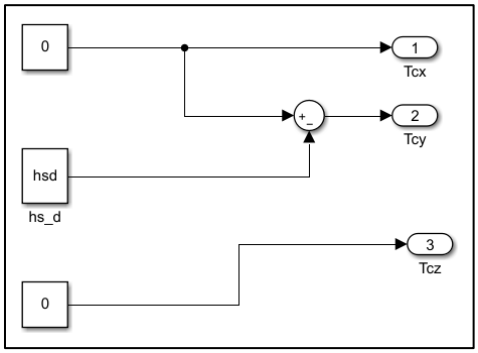
```
    t = t/3600;
    Y = Y.*(180/pi);


% Plotting
plot(t, Y(:,1), 'LineWidth', 1.5)
```



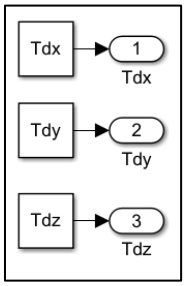
1.3-1: Simulink Model for Pitch Control



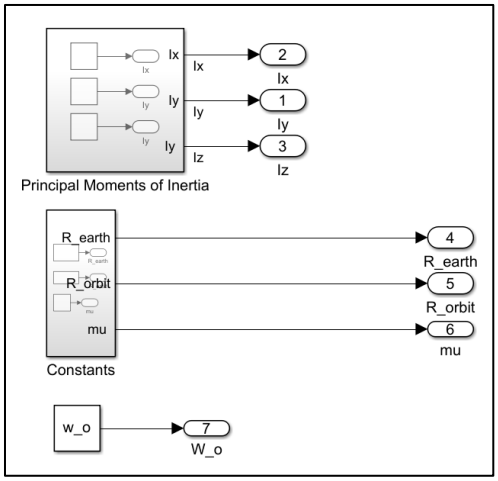
1.3-2: Pitch Equation Subsystem



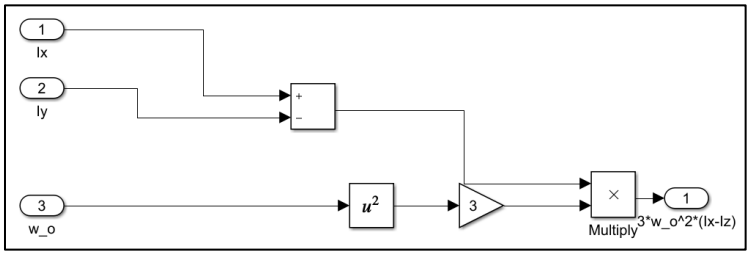
1.3-3: Control Torques Subsystem



1.3-4: Disturbance Torques Subsystem

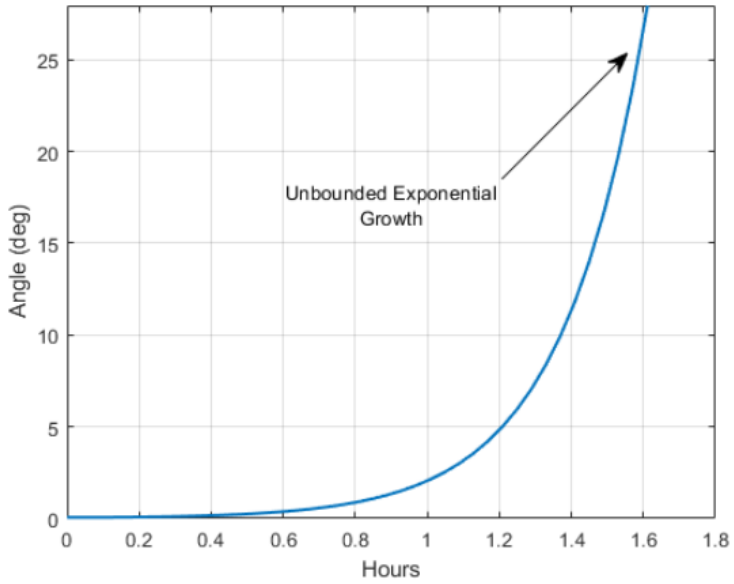


1.3-5: Constraints Relevant Subsystem



1.3-6: k_g Calculation Subsystem

1.4 Results



1.4-1: Pitch Position of the Satellite without a Control System

1.5 Discussion of Results

Following the stability condition for gravity gradient stabilized spacecraft, the vehicle must have an I_x greater than the value of I_z. Based on the response of the vehicle due to small initial conditions, and an I_z greater than the I_x, the spacecraft is not passively stabilized. Later sections will detail the performance of the spacecraft when various control methods are added.

Problem 2

2.1 Problem Statement

The next step is to determine the required gains to maintain stability and control for the pitch motion of the spacecraft. The control law for the pitch control is provided as the modified PD type:

$$T_{cy}(t) = k_p^p(\theta_d - \theta) - k_d^p\dot{\theta}$$

where the desired pitch angle is denoted as θ_d. Once the plant transfer function for the pitch angle is determined, the controller must be implemented into the control system outlined in part 1. Disturbance toques will be added into the system to observe their impact on the steady state error of the response. The pitch disturbance torque will be defined as:

$$T_{dy}(s) = \frac{T_{dy,max}}{s}$$

With the presence of the disturbance torque, the closed-loop transfer function from desired pitch angle θ_d to actual pitch angle θ. The pitch control gains were then designed to meet the pitch specification in the closed-loop control system.

2.2 Solution Derivation

The transfer function for the plant was based upon de Ruiter's solution, with the addition of the gravity gradient term, k_g.

To design the gains to meet the controller design requirements a visualization of the system response was generated and sliders for the gain values were used to vary the results. The response of the system was obtained by solving the differential equation of the pitch motion using ode45 and plotting the results.

The Simulink model was adjusted to reflect the changes necessary requirements within the control torques subsystem. A step input was inserted to represent the pitch disturbance torque.

2.3 Le Code / Der Simulink

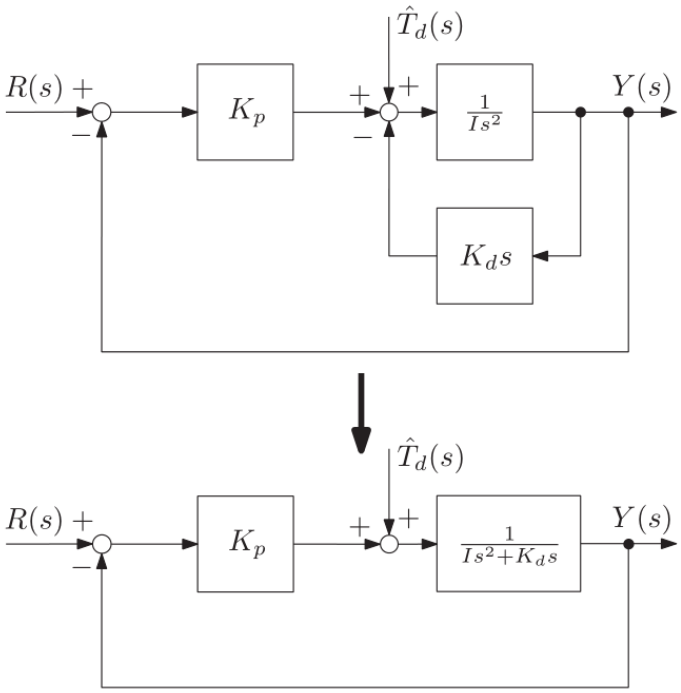
```
% Setting Gains
gains.p2.kp = 0.075;
gains.p2.kd = 0.145;


% Transfer Function Options
TFp.p2.M = inputs.TdMax;
TFp.p2.tau = 0;
TFp.p2.num = TFp.num(TFp.p2, gains.p2, sym("s"));
TFp.p2.den = TFp.den(TFp.p2, gains.p2, sym("s"));

% Equations of Motion Inputs
```

```
eomOpts.p2.stepTime = 0;
eomOpts.p2.Td = inputs.TdMax;
eomOpts.p2.thc = 0.1 * pi/180;
eomOpts.p2.tSpan = [0 200];
eomOpts.p2.IC = [0; 0];
clf
rlocusplot(tf(TFp.p2.num, TFp.p2.den))
title(sprintf('Root Locus w/o Actuator at k_p = %0.2f & k_d = %0.2f',
gains.p2.kp, gains.p2.kd))
grid on
axis equal


% Defining EoMs
EoM_ = @(t, x) EoMp(t, x, inputs, eomOpts.p2, gains.p2);

% Setup for ODE45
[t, Y] = ode45(EoM_, eomOpts.p2.tSpan, eomOpts.p2.IC); Y = Y*180/pi;


% Plotting
clf
plot(t, Y(:,1), 'LineWidth', 1.5, 'DisplayName', 'System Response')


    % Additional Plot Elements
    hold on
    % Commanded Value
    eomOpts.p2.thc = eomOpts.p2.thc*180/pi;
    plot(eomOpts.p2.tSpan, ones([1 2])*eomOpts.p2.thc, ...
                                'k', ...
                                'DisplayName', 'Commanded Value', ...
                                'PickableParts', "none")
    % Overshoot Threshold
    plot(eomOpts.p2.tSpan, ones([1 2])*eomOpts.p2.thc*(1+const.Mp), ...
                                        '--k', ...
                                        'DisplayName', sprintf('%0.0f%%
Overshoot Limit', const.Mp*100), ...
                                        'PickableParts', "none")
    % Finding settling time
    idx = find(abs(Y(:, 1) - Y(end, 1))/Y(end, 1) > 0.02, 1, 'last')+1;
    plot(t(idx), Y(idx, 1), 'xr', ...
                            'MarkerSize', 10, ...
                            'LineWidth', 2, ...
                            'DisplayName', sprintf('t_{s, 2%%} = %0.2f sec',
t(idx)), ...
                            'PickableParts', "none")

    % Printing Steady State Error
```

```
    plot(0, 0, 'o', 'MarkerSize', 0.001, 'DisplayName', sprintf('\\theta_{ss}
= %0.4f\\circ', abs(Y(end, 1)-eomOpts.p2.thc)))

    % Other Plot Elements
    title(sprintf('System Response at Gains: k_p^p = %0.2f & k_d^p = %0.3f',
gains.p2.kp, gains.p2.kd))
    legend('Location',"southeast")
    grid on; grid minor; xlabel('Time (sec)'); ylabel('Response Angle
(deg)');
    hold off;
```

2.4 Results

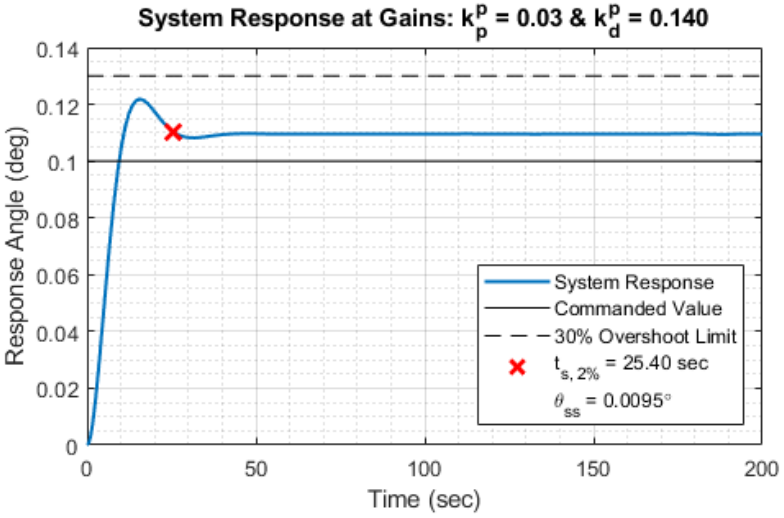
2.4.1 Part a



2.4-1: de Ruiter Block Diagram for PD Pitch Controller

$$G_p(s) = \frac{1}{Is^2 + k_g}$$

2.4.2 Part b



2.4-2: System Response at Designed Gain Values

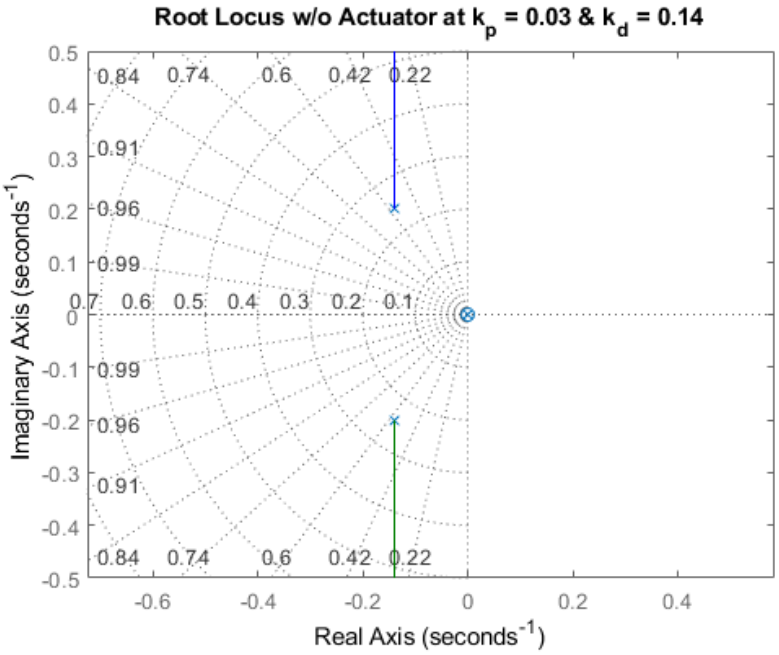


Figure 2.4-3: Root Locus of the Response

2.4.3 Part c

$$r(s) = T_{dy}(s) = \frac{T_{dy,max}}{s} \leftarrow \text{Step Disturbance}$$

$$e_{ss} = -\frac{T_{dy,max}}{k_p} :: \text{(de Ruiter) pg 318}$$

2.4.4 Part d

$$G_p(s) = \frac{1}{I_y \cdot s^2 + k_d \cdot s}$$

$$G_c(s) = k_p + \frac{T_{d,y}}{s}$$

$$T(s) = \frac{G_c(s)}{1 + G_p(s)G_c(s)}$$

$$T(s) = \frac{k_p + \frac{T_{d,y}}{s}}{1 + \left(k_p + \frac{T_{d,y}}{s}\right) \cdot \frac{1}{I \cdot s^2 + k_d \cdot s}} = \frac{T_{d,y}k_p + k_p s}{I_y s^3 + k_d s^2 + \left(k_g + k_p + T_{d,y}k_d\right)s + T_{d,y}k_p}$$

2.5 Discussion of Results

At the proportional gains of 0.3 and derivative gains of 0.14, the system's response was below the required 30% overshoot. As well, the time to reach steady state after the initial input to the desired pitch is shown to take 25.4 seconds to reach a 2% margin from the requested position. After all motion is damped out, the system levels off with a steady state error of 0.0095°, which is below the required 0.1°. To further the proof of the systems stability, the two dominant poles are on the LHS of the plot, while another pole and zero are close, but not crossing the imaginary axis.

Problem 3

3.1 Problem Statement

The roll and yaw control for the spacecraft can now be designed independently from the pitch control due to the coupled nature of the two rotations. The control law for roll and yaw is:

$$T_{cx} = -\left(k_p^{ry}\phi + k_d^{ry}\dot{\phi}\right)$$

$$T_{cz} = \alpha\left(k_p^{ry}\phi + k_d^{ry}\dot{\phi}\right)$$

Where $\alpha > 0$, $k_p^{ry} > 0$, and $k_d^{ry} > 0$ are the parameters which were designed for control of roll and yaw. The controllers were then implemented into the control system designed in part 1. The closed-loop equations of motion for roll and yaw were then transformed into the Laplace domain in matrix form to determine the transfer functions in terms of α, k_p^{ry}, and k_d^{ry}.

3.2 Solution Derivation

With the controller set up the disturbances were then made present in the equations of motion to determine steady state errors for the step disturbances and the upper boundary for the steady-state yaw. The bias momentum of the reaction wheel and the proportional gain were then designed to meet the design requirements of the controller given that the values are real and positive.

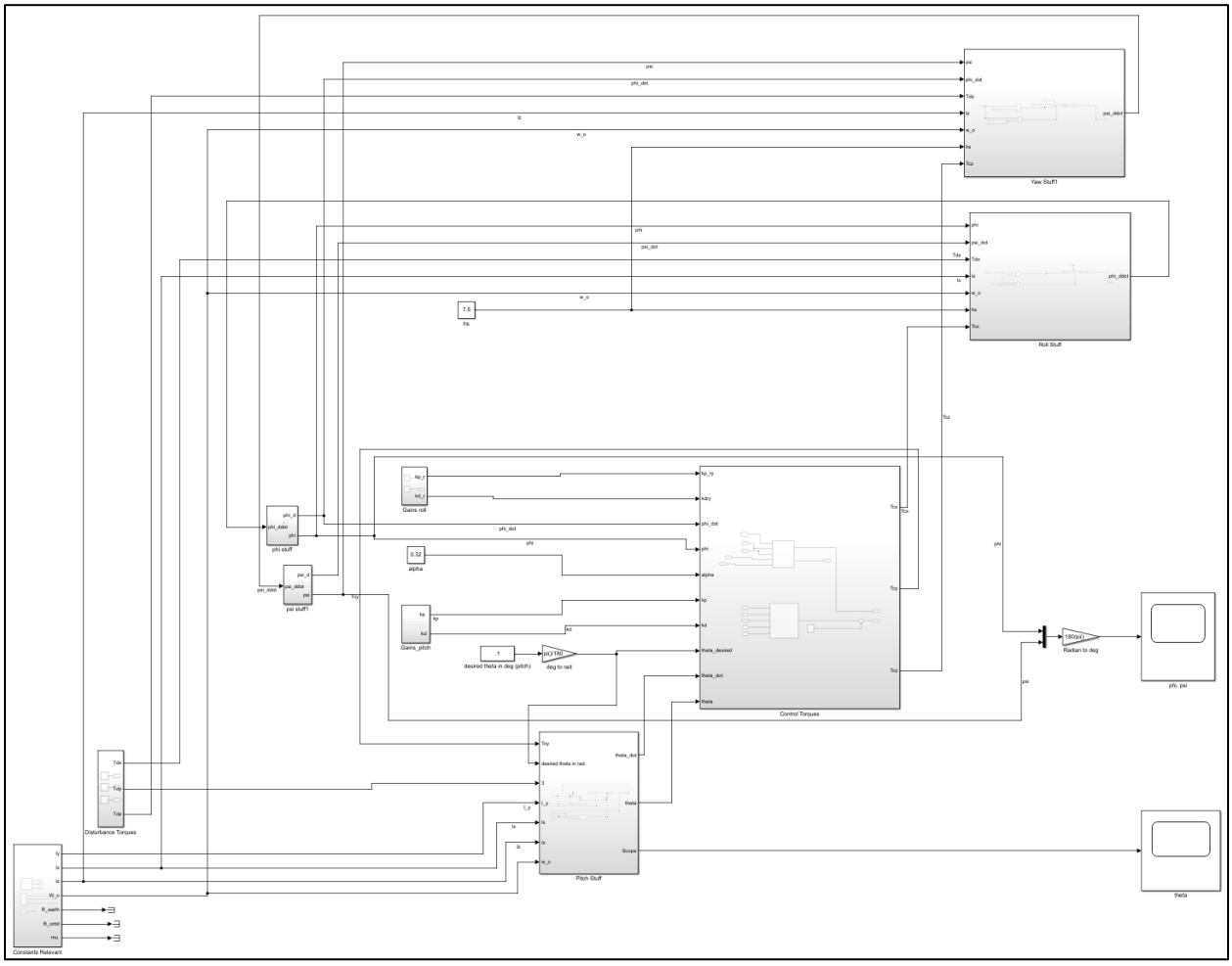
The final step of designing the system for the control of the yaw and roll motions was to design the derivative gain (k_d^{ry}) and α to meet the requirements of the system. The characteristic equation using the damping ratio and undamped natural frequency of the system was used to determine the derivative gain value. The characteristic equation can be found as

$$\Delta = (s^2 + 2\zeta\omega_{n1}s + \omega_{n1}^2)(s^2 + 2\zeta\omega_{n2}s + \omega_{n2}^2).$$

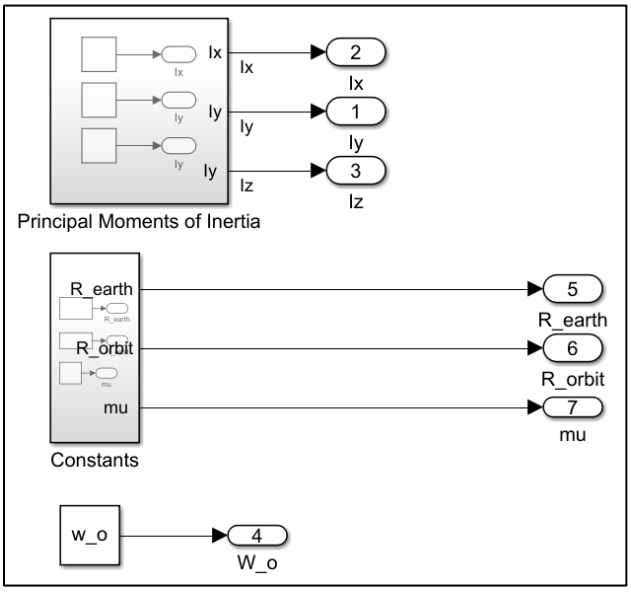
3.3 Le Code / Der Simulink

```
% Input Gains
hs = 7.577497*units.h;
kpry = 0.1559333;
kdry = 10.421095018;
alpha = 0.32407316;

% EoM Constants Setup
tspan = [0 2500];
tStep = 500;

    % Removing Units
    Td_ = sepUnits(Tdmax);
    thc_ = sepUnits(unitConvert(thc, u.rad));
    Ix_ = sepUnits(Ix);
    Iy_ = sepUnits(Iy);
    Iz_ = sepUnits(Iz);
    wo_ = sepUnits(wo);
```

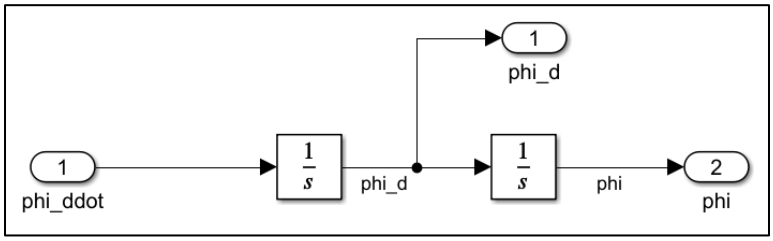
```
    hs_ = sepUnits(hs);


% Defining EoM
EoM = @(t, x) EoMf(t, x, tStep, Td_, ...
                           thc_, Ix_, Iy_, Iz_, wo_, hs_, ...
                           kpp, kdp, kpry, kdry, alpha);


% Running ODE
[t, Y] = ode45(EoM, tspan, [0; 0; 0; 0; 0; 0]); Y = Y.*(180/pi);


% Plotting
clf
plot(t, Y(:, 1), 'LineWidth', 1.5, 'DisplayName', 'Roll Response'); hold on
plot(t, Y(:, 3), 'LineWidth', 1.5, 'DisplayName', 'Pitch Response')
plot(t, Y(:, 5), 'LineWidth', 1.5, 'DisplayName', 'Yaw Response')


    % Plotting Commanded Theta
    plot(tspan, thc_*ones([1 2]), '-k', 'PickableParts', "none", 'DisplayName',"Commanded
\theta")

    % Additional Plot Elements
    title(sprintf('System Response at Gains: k_p^{ry} = %0.2f & k_d^{ry} = %0.3f', kpry, kdry))
    legend('Location',"best")
    grid on; grid minor; xlabel('Time (sec)'); ylabel('Response Angle (deg)');
    hold off
```



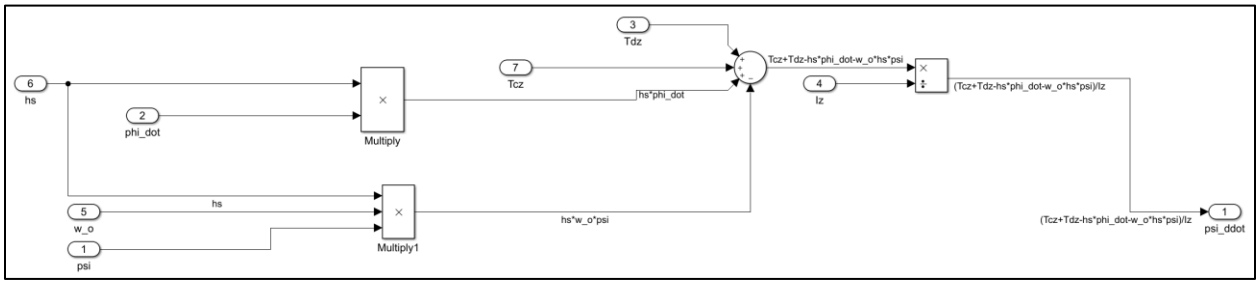
3.3-1: Simulink Model for Roll and Yaw Control



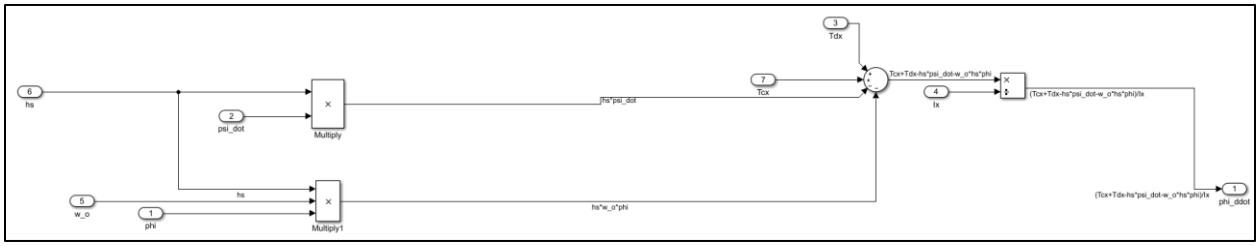
3.3-2: Constants Relevant Subsystem



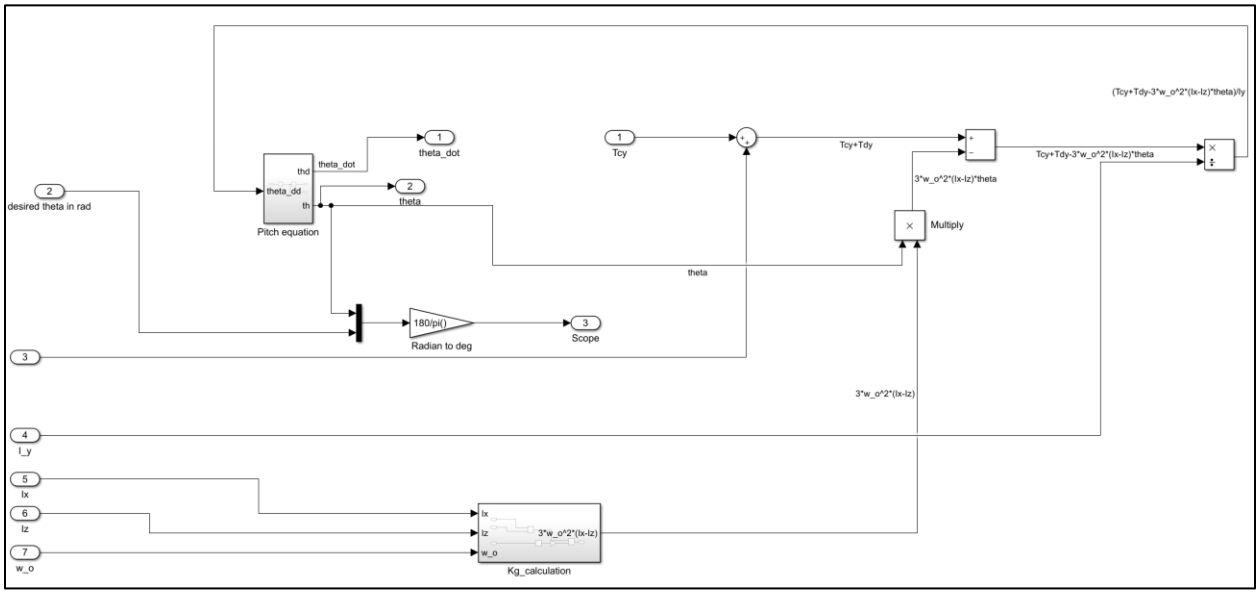
3.3-3: Phi Relations Subsystem



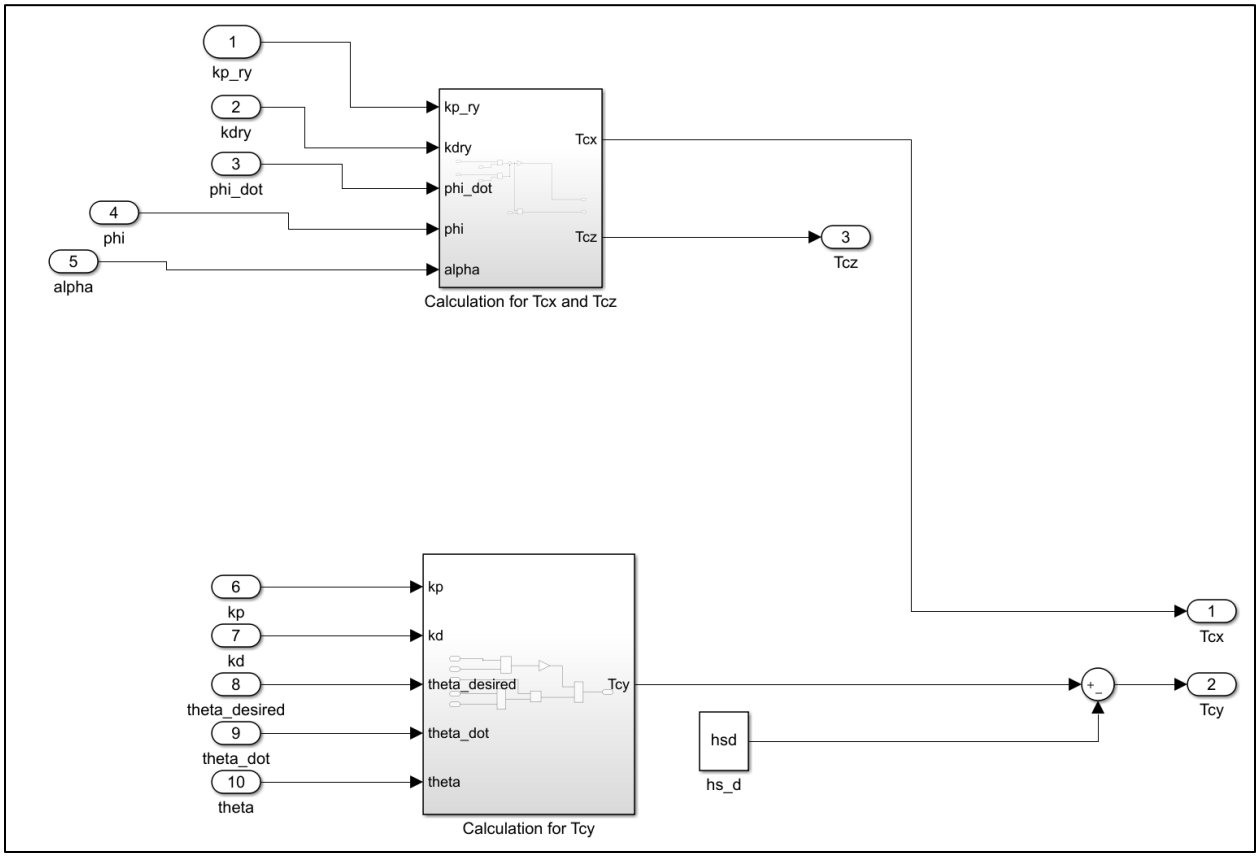
3.3-4: Yaw Relations Subsystem



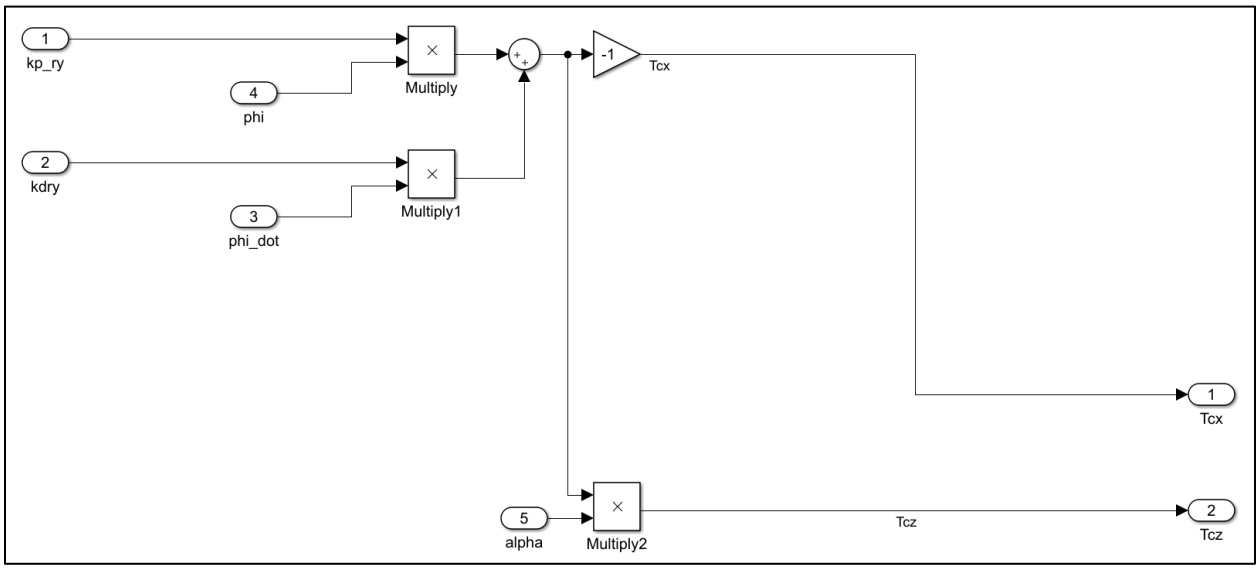
3.3-5: Roll Relations Subsystem



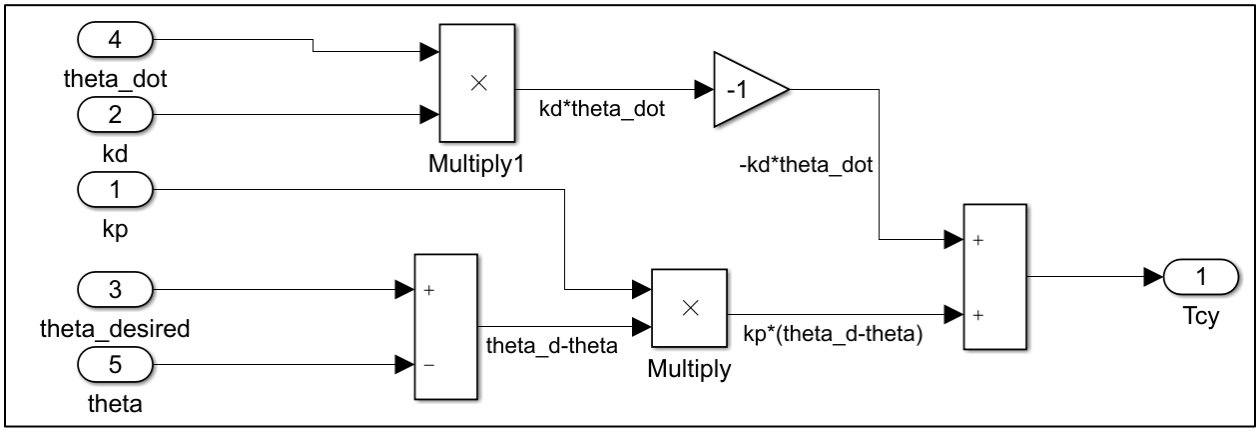
3.3-6: Roll Relations Subsystem



3.3-7: Control Torques Subsystem



3.3-8: Calculation for T_{cx} and T_{cz} Subsystem



3.3-9: Calculation for T_{cy} Subsystem

3.4 Results

$$I_x\ddot{\phi} + \dot{\psi}h_s + \phi\omega_o h_s + \left(k_p^{\mathrm{ry}}\phi + k_d^{\mathrm{ry}}\dot{\phi}\right) = T_{d,x}$$

$$I_z\ddot{\psi} - \dot{\phi}h_s + \psi\omega_o h_s - \alpha\left(k_p^{\mathrm{ry}}\phi + k_d^{\mathrm{ry}}\dot{\phi}\right) = T_{d,z}$$

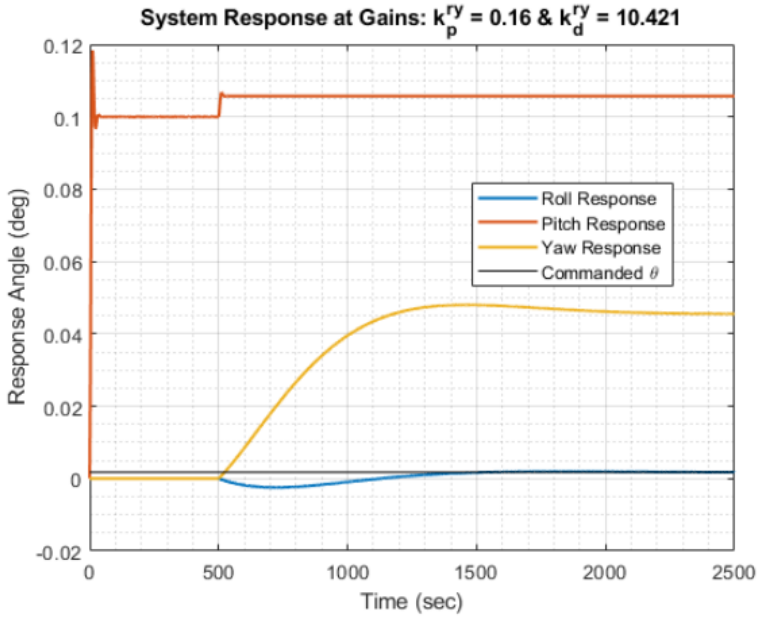
↓ Laplace Transform

$$\underbrace{\begin{bmatrix} I_x s^2 + k_d^{\mathrm{ry}} s + \omega_o h_s + k_p^{\mathrm{ry}} & sh_s \\ -s\left(+\alpha k_d^{\mathrm{ry}} + h_s\right) - \alpha k_p^{\mathrm{ry}} & I_z s^2 + \omega_o h_s \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \phi(s) \\ \psi(s) \end{bmatrix} = \begin{bmatrix} T_{d,x} \\ T_{d,z} \end{bmatrix}$$

$$\begin{aligned} \Delta = \det(A) = {} & s^4(I_x I_z) + \\ & s^3\left(I_z k_d^{\mathrm{ry}}\right) + \\ & s^2 h_s\left[\alpha k_d^{\mathrm{ry}} + (I_x + I_z)\omega_o + h_x\right] + I_z k_p^{\mathrm{ry}} + h_s^2 + \\ & s^1\left(\alpha k_p^{\mathrm{ry}} h_s + \omega_o k_d^{\mathrm{ry}} h_s\right) + \\ & s^0\left[(\omega_o h_s)^2 + \omega_o h_s k_p^{\mathrm{ry}}\right. \end{aligned}$$

3.4.1 Part a

The control law portrayed was inserted as a subsystem into the “Control Torques” subsystem. The α term was assumed to be a constant, and the gains for roll and yaw motion were set as positive values. The overall Simulink assembly ran successfully.



The subsystems discussed are portrayed in figures 3.37 through 3.39.

3.4.2 Part b

Part B required the system to be converted to the frequency domain to determine the transfer function. The transfer functions in roll and yaw were determined using symbolic math in MATLAB Live.

3.4.3 Part c

Part C is also represented in the equation above by taking the inverse of the A to make the equations be output over input.

3.4.4 Part d

Under the assumption that α>0 and that the step disturbances for each section were at maximum disturbance (5E-6, as determined by the problem statement), the system produced the following steady state error. The problem was completed using symbolic math in MATLAB Live.

$$Assume\ \alpha < 1;\ k_p \gg \omega_0 h_s$$

$$\psi_{ss} = \frac{\alpha k_p T_{dx}}{\omega_o h_s(\omega_o h_s + k_p)} + \frac{T_{dz}}{\omega_o h_s}; \qquad k_p + \omega_o h_s \approx k_p \quad if \quad k_p \gg \omega_o h_s$$

$$= \frac{\alpha \cancel{k_p} T_{dx}}{\omega_o h_s \cancel{k_p}} + \frac{T_{dz}}{\omega_o h_s} = \frac{\alpha T_{dx}}{\omega_o h_s} + \frac{T_{dz}}{\omega_o h_s}; \qquad T_{dx} \ll 1; \qquad assuming \quad (small\ value) \times (small\ value) \approx 0$$

Attitude Control System Design – Bias Momentum Satellite

$|\psi_{ss}| \leq \frac{|T_{dx}|+|T_{dz}|}{\omega_o |h_s|}$; T_{dx} & T_{dz} must remain positive

TF =

$$\begin{pmatrix} \frac{\text{Tdx}\ (\text{Iz}\ s^2 + \text{hs wo})}{\sigma_1} - \frac{\text{Tdz hs}\ s}{\sigma_1} \\ \frac{\text{Tdx}\ (\alpha\,\text{kp} + \text{hs}\ s + \alpha\,\text{kd}\ s)}{\sigma_1} + \frac{\text{Tdz}\ (\text{Ix}\ s^2 + \text{kd}\ s + \text{kp} + \text{hs wo})}{\sigma_1} \end{pmatrix}$$

where

$$\sigma_1 = \text{hs}^2\, s^2 + \text{hs}^2\, \text{wo}^2 + \text{hs kp wo} + \text{Ix Iz}\, s^4 + \text{Iz kd}\, s^3 + \text{Iz kp}\, s^2 + \alpha\,\text{hs kp}\, s + \text{hs kd}\, s\, \text{wo} + \text{Ix hs}\, s^2\, \text{wo} + \text{Iz hs}\, s^2\, \text{wo} + \alpha\,\text{hs kd}\, s^2$$

TF_roll =

$$\frac{\text{Tdx}\ (\text{Iz}\ s^2 + \text{hs wo})}{\sigma_1} - \frac{\text{Tdz hs}\ s}{\sigma_1}$$

where

$$\sigma_1 = \text{hs}^2\, s^2 + \text{hs}^2\, \text{wo}^2 + \text{hs kp wo} + \text{Ix Iz}\, s^4 + \text{Iz kd}\, s^3 + \text{Iz kp}\, s^2 + \alpha\,\text{hs kp}\, s + \text{hs kd}\, s\, \text{wo} + \text{Ix hs}\, s^2\, \text{wo} + \text{Iz hs}\, s^2\, \text{wo} + \alpha\,\text{hs kd}\, s^2$$

TF_yaw =

$$\frac{\text{Tdx}\ (\alpha\,\text{kp} + \text{hs}\ s + \alpha\,\text{kd}\ s)}{\sigma_1} + \frac{\text{Tdz}\ (\text{Ix}\ s^2 + \text{kd}\ s + \text{kp} + \text{hs wo})}{\sigma_1}$$

where

$$\sigma_1 = \text{hs}^2\, s^2 + \text{hs}^2\, \text{wo}^2 + \text{hs kp wo} + \text{Ix Iz}\, s^4 + \text{Iz kd}\, s^3 + \text{Iz kp}\, s^2 + \alpha\,\text{hs kp}\, s + \text{hs kd}\, s\, \text{wo} + \text{Ix hs}\, s^2\, \text{wo} + \text{Iz hs}\, s^2\, \text{wo} + \alpha\,\text{hs kd}\, s^2$$

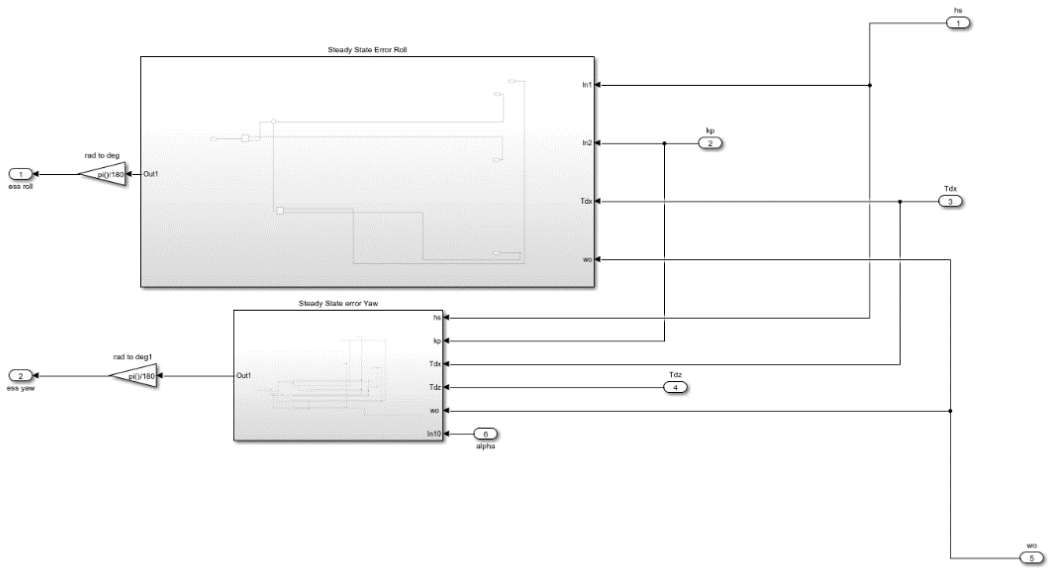
TF_roll_ess =

$$\frac{\text{Tdx}}{\text{kp} + \text{hs wo}}$$

TF_yaw_ess =

$$\frac{\text{Tdz kp} + \text{Tdx}\ \alpha\,\text{kp} + \text{Tdz hs wo}}{\text{hs wo}\ (\text{kp} + \text{hs wo})}$$

The steady state error for roll and yaw were found by modeling the roll yaw equations found in Part C in Simulink.

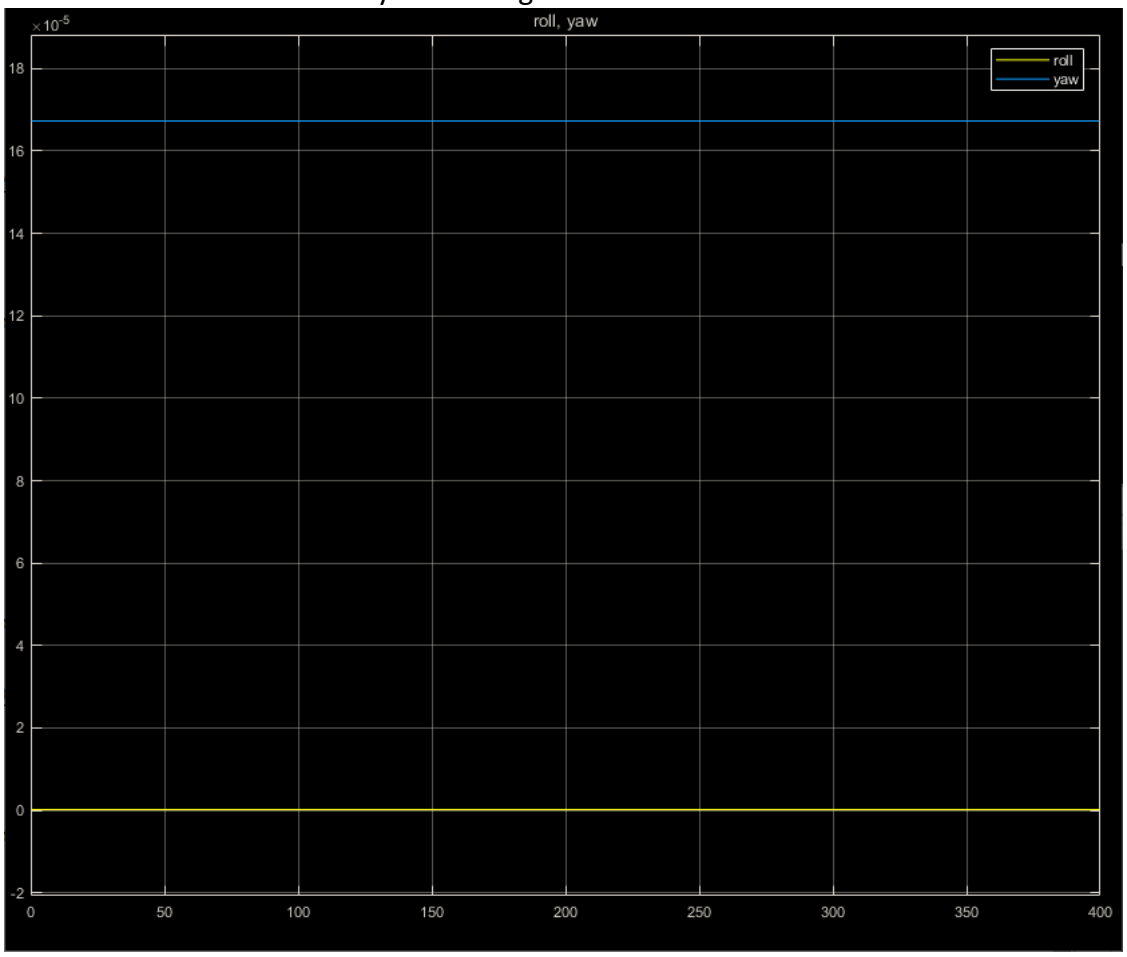


3.4.5 Part e

Under the assumption that $\alpha < 1$ and that $K_p >> w_o h_s$, one can conclude that $K_p + w_o h_s$ is equivalent to K_p, as the proportional can still be approximated to itself, assuming "$w_o h_s$" is relatively small. As such, the proportional gain cancels out within the system of equations. Utilizing the idea that $T_{dx} * \alpha$ is still a small value, one can assume that T_{dx} remains the same. As such, the upper bound can be seen below.

3.4.6 Part f

Using the Simulink model, one can determine steady state error as portrayed in Part D. Holding the conditions that the proportional gains and angular momentum are positive values, the system was shown to have a smaller segment of error which is concerted to degrees. The result also satisfies the condition for a larger time step, which is not displayed below.



3.4.7 Part g

The derivative gain K_d and α were considered as symbolic variables in MATLAB Live when determining their values to satisfy the desired transient performance in roll and yaw.

3.5 Discussion of Results

The response of the system after the step disturbance was added remains stable for both roll and yaw using a proportional gain of 0.16 and a derivative gain of 10.42. The settling time of the roll motion is approximately 800 seconds and the settling time of the yaw motion is approximately 1500 seconds.

Problem 4

4.1 Problem Statement

The next step was to design the actuator controller to be used for pitch control. The actuator dynamics have been defined as $T_{cy}(s) = \frac{1}{Ts+1} U_c(s)$ where T is the actuator time constant, $U_c(s)$ is the control torque commanded by the control law determined in the pitch controller design, and $T_{cy}(s)$ is the pitch torque applied to the spacecraft by the actuator. The goal is to implement a controller for the actuator into the pitch control system which was designed earlier. Once actuator is represented in the system, the characteristic equation for the closed-loop system can be determined to obtain the time-constant to keep the system asymptotically stable. A root locus will then be generated to observe the behavior of the system and check for stability.

4.2 Solution Derivation

4.2.1 Part a

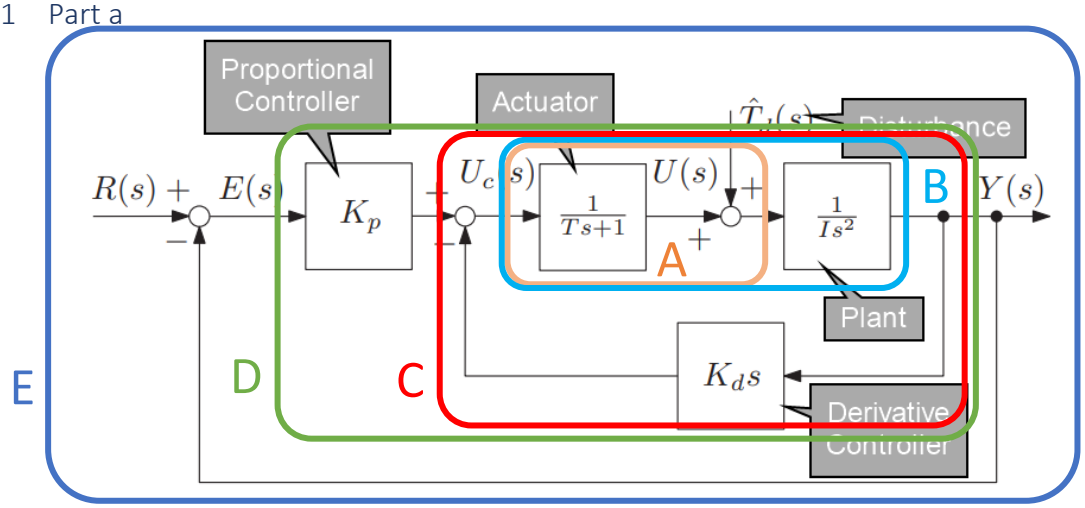


Figure 4.2-1: Block Diagram from de Ruiter pg 323 & Their Associated Functions

$$A = \frac{\mathrm{M_d}}{\mathrm{s}} + \frac{1}{\mathrm{T\,s} + 1}$$

$$B = \frac{\frac{M_d}{s} + \frac{1}{T\,s+1}}{I_y\,s^2 + k_g}$$

$$C = \frac{\frac{M_d}{s} + \frac{1}{T\,s+1}}{\left(\frac{k_d\,s\left(\frac{M_d}{s} + \frac{1}{T\,s+1}\right)}{I_y\,s^2 + k_g} + 1\right)\left(I_y\,s^2 + k_g\right)}$$

$$\boldsymbol{D} = \frac{k_p\left(\frac{M_d}{s} + \frac{1}{T\,s+1}\right)}{\left(\frac{k_d\,s\left(\frac{M_d}{s} + \frac{1}{T\,s+1}\right)}{I_y\,s^2 + k_g} + 1\right)\left(I_y\,s^2 + k_g\right)}$$

$$\boldsymbol{E} = \frac{s\left(k_p + M_d\,T\,k_p\right) + M_d\,k_p}{\mathrm{I_y}Ts^4 + \mathrm{I_y}s^3 + \left(k_d + T\,k_g + M_d\,T\,k_d\right)s^2 + \left(k_g + k_p + M_d\,k_d + M_d\,T\,k_p\right)s + M_d\,k_p}$$

4.2.2 Part b

$$\Delta = den(E)$$
$$= \mathrm{I_y}Ts^4 + \mathrm{I_y}s^3 + \left(k_d + T\,k_g + M_d\,T\,k_d\right)s^2 + \left(k_g + k_p + M_d\,k_d + M_d\,T\,k_p\right)s + M_d\,k_p = 0$$
$$\downarrow for\ M_d = 0$$
$$\Delta = I_yTs^3 + I_ys^2 + k_ds + k_p + k_g = 0$$

4.2.3 Part c

$$\downarrow Routh - Hurwitz\ Criteroin$$

$$\begin{array}{c|cc} s^3 & b_0 & b_2 \\ s^2 & b_1 & b_3 \\ s^1 & c_1 & \\ s^0 & d_1 & \end{array} \rightarrow \begin{array}{c|cc} s^3 & I_yT & k_gT + k_d \\ s^2 & I_y & k_g + k_p \\ s^1 & c_1 & \\ s^0 & d_1 & \end{array}$$

$$c_1 = \frac{I_y\left(k_gT + k_d\right) - I_yT(k_g + k_p)}{I_y}$$
$$= k_gT + k_d - Tk_g - Tk_p$$
$$T < \frac{k_d}{k_p}$$

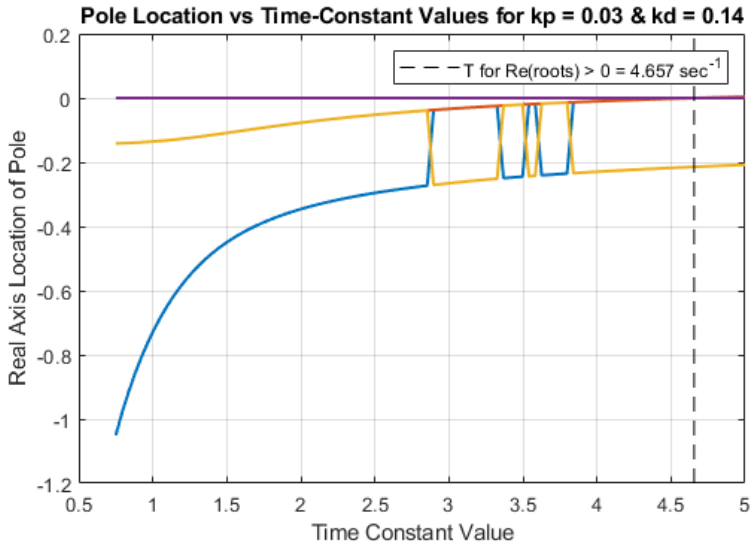
4.3 Le Code / Der Simulink

```
rootVals = double.empty(0,4);
% Transfer Function Options
TFp.p4.M = inputs.TdMax;


for T = linspace(0.75, 5, 100)
    TFp.p4.tau = T;
    charEqCoeff = TFp.den(TFp.p4, gains.p2, sym("s"));
    temp = real(roots(charEqCoeff))';
```

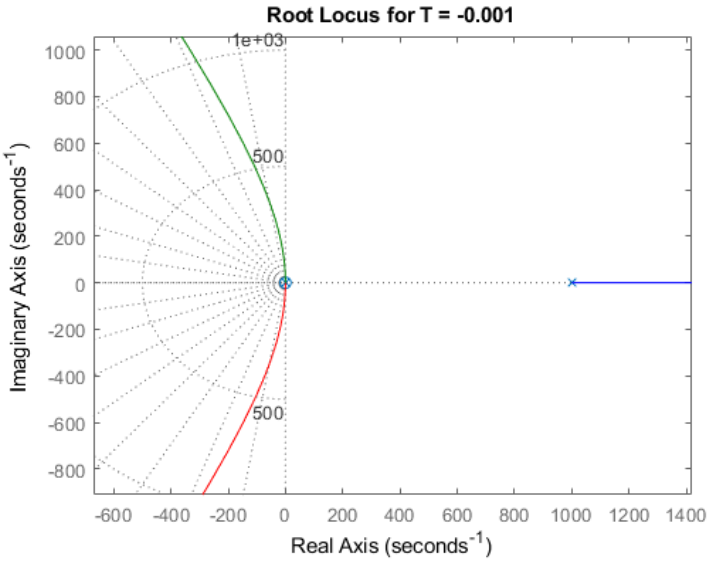
```
    rootVals = cat(1, rootVals, temp);
end
T = linspace(0.75, 5, 100);
clf
plot(T, rootVals, 'LineWidth', 1.5, 'HandleVisibility',"off")
grid on
hold on
idx = find(any(rootVals > 0, 2), 1, "first")-1;
plot([T(idx) T(idx)], ylim, '--k', 'DisplayName', sprintf('T for Re(roots) >
0 = %.3f sec^{-1}', T(idx)))
hold off
ylabel('Real Axis Location of Pole')
xlabel('Time Constant Value')
title(sprintf('Pole Location vs Time-Constant Values for kp = %.2f & kd =
%.2f', gains.p2.kp, gains.p2.kd))
legend


% Part d
TFp.p4.tau = -0.001;
%{
For any 1/T ≤ 0 (i.e.: negative T) there is no stability possible as
one pole is always on the RHS of the graph
%}
rlocus(tf(TFp.num(TFp.p4, gains.p2, sym("s")), TFp.den(TFp.p4, gains.p2,
sym("s"))))
title(sprintf('Root Locus for T = %.3f', TFp.p4.tau))
grid on
```

4.4 Results

4.4.1 Part c



4.4-1: Pole Location vs. Time-Constant Values for k_p = 0.03 & k_d = 0.14



4.4-2: Root Locus for T = -0.001

4.5 Discussion of Results

By using the Routh-Hurwitz Criteria, an expression for the stability of the system was found. For asymptotic stability, the time constant, T, must be less than the ratio of the derivative gain over the proportional gain. With the associated gains from problem 2, it was found that the time constant must be less than a value of 4.667. This would proven by plotting the movement of the poles versus different values of the time constant in 4.4-1. It can be noted that the jagged pattern from the plot is due to the method by which MATLAB calculated the poles. Between each iteration, the order of the poles within the variable is not saved.

In addition, the fourth pole, in purple, appears to stay constant throughout the iteration. Its value fluctuates between -5.0001e-6 and -5.0000e-6, never crossing to the RHS plot.

For part d, it was found that any negative value for T was associated with an unstable system, as seen in 4.4-2.

Problem 5

5.1 Problem Statement

The addition of the actuator dynamics into the control system slowed the response of the system when compared to the speed of the response without the inclusion of the actuator. The next step in the design of the control system is to adjust the controller to return to the previous response before the inclusion of the actuator.

5.2 Solution Derivation

To match the original poles of the actuator-less system in problem 2, with a proportional gain of 0.03 and a derivative gain of 0.14, new system gains must be selected. While varying the new gains for the system, the distance between the dominants nodes were measured and the gains with the shortest distance was selected.

5.3 Le Code / Der Simülink

```
% Setting Gains
gains.p5.kp = 0.001;
gains.p5.kd = 0.115;


% Transfer Function Options
TFp.p5.M = inputs.TdMax;
TFp.p5.tau = 5;
TFp.p5.num = TFp.num(TFp.p5, gains.p5, sym("s"));
TFp.p5.den = TFp.den(TFp.p5, gains.p5, sym("s"));




% Plotting Root Locus vs Problem 2
TF = [tf(TFp.p2.num, TFp.p2.den) tf(TFp.p5.num, TFp.p5.den)];
poles.p2.comp = pole(TF(1));
poles.p2.real = unique(real(poles.p2.comp));
poles.p2.imag = unique(abs(imag(poles.p2.comp)));
poles.p5.comp = pole(TF(2));
poles.p5.real = unique(real(poles.p5.comp));
poles.p5.imag = unique(abs(imag(poles.p5.comp)));
rlocusplot(TF(1), 'r', TF(2), 'b')
legend('System Response w/o Actuator', 'System Response w/ Actuator',
'Location', 'NorthWest')
title(sprintf('Root Locus T = %1.0f, k_p = %0.3f, k_d = %0.3f', TFp.p5.tau,
gains.p5.kp, gains.p5.kd))
grid on
```

```
axis equal
```

5.4 Results

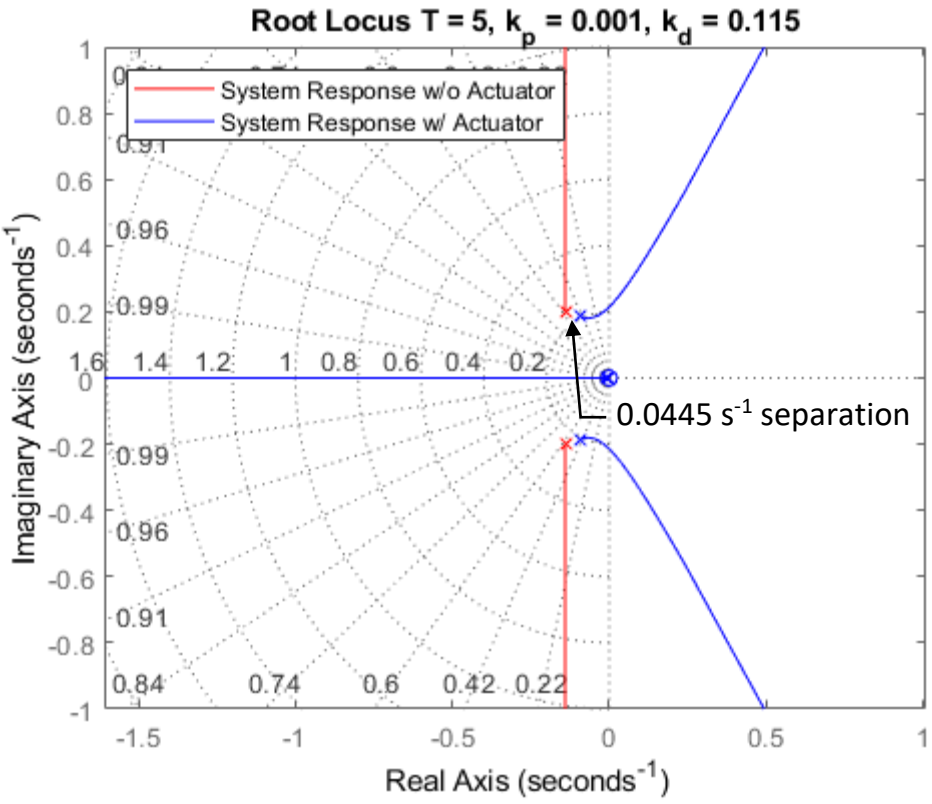


Figure 5.4-1: Plot of Root Locus from Problem 2 w/ New Root Locus with Actuator Dynamics

5.5 Discussion of Results

For a time constant of 5, the gains associated with the shortest reasonable distance, 0.0445 s^{-1}, were a proportional gain of 0.001 and derivative gain of 0.115. This is 30 times decrease original proportional gain and 82.1% of the original derivative gain.

A smaller distance, 0.04 s^{-1}, was found for the poles, but it was at a proportional gain that approached 0.

Conclusion

The bias momentum satellite was initially unstable in the provided conditions. State and rate sensing controllers were used to restrain the motion of the spacecraft from growing exponentially. Design of the controller was performed separately for pitch control from yaw and roll control due to the decoupled nature of the motions.

Without the inclusion of the actuator, the proportional gain of the system was 0.03 and the derivative gain was 0.14. As a result, the systems response settled after 25.4 seconds, with a steady-state error of 0.0095°, and a maximum overshoot under the 30% required.

As for the roll and yaw portion of the controller, the proportional gain was 0.518 and the derivative gain was 9.68. The associated response kept the spacecrafts motion from growing exponentially, even without a yaw sensing component. The steady-state error for the roll in this case was approaching 0, while the error in yaw was on the order of 1.7e-4 degrees.

Control of the actuator for the pitch controller considered for the control laws of the system. A proportional gain of 0.001 and derivative gain of 0.115 were used to return the response time of the system to a state similar to the response before the integration of the actuator controller. The control system designed for the spacecraft meets all design requirements of the controller and ensure that the spacecraft retains a stable motion.

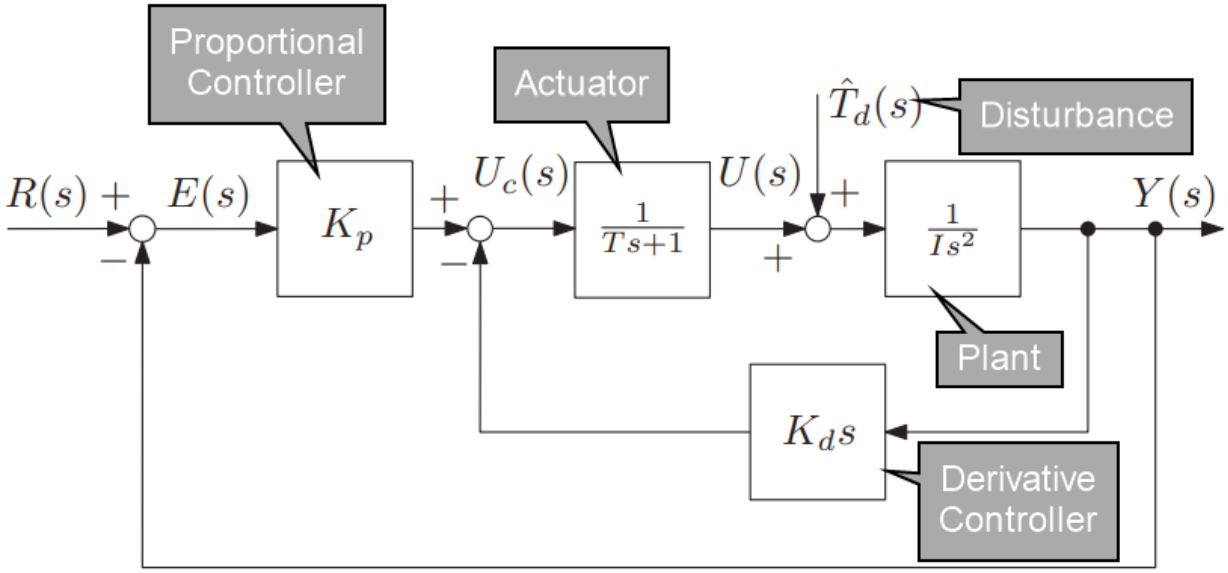
References

[1] C. D. J. F. A. Ruiter, Spacecraft Dynamics and Control - An Introduction, Chichester, West Sussex: John Wiley & Sons, Ltd., 2013.

[2] "TenTech LLC, Solutions and Services," TenTech LLC, 2011-2020. [Online]. Available: https://www.tentechllc.com/success-stories.html. [Accessed 8 May 2020].

Appendix A MATLAB Codes

Pitch Transfer Function Calculation




```
% Setting SymboliC Variables
I = sym("I_y"); M = sym("M_d"); kp = sym("k_p"); kd = sym("k_d"); kg =
sym("k_g");
syms T s


% Turning off certain values
%M = 0;
%T = 0;


% Creating Block Diagram Blocks
plnt = 1/(I*s^2+kg);
act = 1/(T*s+1);
prop = kp;
deriv = kd*s;
distur = M/s;


% Combining System Blocks
A = (act + distur)                  % Adding Actuator & Distrubance
B = A*plnt                          % Mutiplying above with plant
C = simplify(B/(1 + B*deriv))       % Derivative Controller Feedback
D = simplify(C*prop)                % Multiplying Proportional Gain
E = D/(1 + D);                      % Full System Feedback
E = subs(collect(E), I, I)
```

```
% Creating Callable Functions
[n, d] = numden(E);
n = matlabFunction(n);   % (M, T, kp, s)
d = matlabFunction(d);   % (M, T, kd, kp, s)


    % Modifying Functions
    TFp.num = @(opts, gains, s) fliplr(double(coeffs(n(opts.M, opts.tau,
gains.kp, s))));
    TFp.den = @(opts, gains, s) fliplr(double(coeffs(d(opts.M, opts.tau,
gains.kd, inputs.kg, gains.kp, s))));
```

Pitch Equation of Motion

$$Y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \xrightarrow{\frac{d}{dt}} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{I_y}(T_{cy} + T_{dy}) - 3\omega_o^2(I_x - I_z)\theta \end{bmatrix}$$

```
function Y = EoMp(t, x, inputs, eomOpts, gains)
    % Converting Inputs
    th = x(1);
    thd = x(2);

        % System Constants
        Ix = inputs.Ix;
        Iy = inputs.Iy;
        Iz = inputs.Iz;
        kg = inputs.kg;

        % EoM Options
        tStep = eomOpts.stepTime;
        Td = eomOpts.Td;
        thc = eomOpts.thc;

        % System Gains & Commanded Value
        kp = gains.kp;
        kd = gains.kd;

    % Getting External Torques
    Tcy = (kp*(thc - th) - kd*thd);

    if t < tStep
        Tdy = 0;
    else
        Tdy = Td;
    end
```

```
    % Defining EoM
    Y = [thd;
         (1/Iy)*(Tcy + Tdy - kg*th)];
end
```

Roll, Pitch, Yaw Equations of Motion

$$Y = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} \xrightarrow{\frac{d}{dt}} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \frac{1}{I_x}\left(T_{cx} + T_{dx} - h_s\dot{\psi} - \omega_o h_s \phi\right) \\ \dot{\theta} \\ \frac{1}{I_y}\left(T_{cy} + T_{dy}\right) - 3\omega_o^2(I_x - I_z)\theta \\ \dot{\psi} \\ \frac{1}{I_z}\left(T_{cz} + T_{dz} + h_s\dot{\phi} - \omega_o h_s \psi\right) \end{bmatrix}$$

```
function Y = EoMf(t, x, tStep, Td, thc, Ix, Iy, Iz, wo, hs, kpp, kdp, kpr, kdr,
alpha)
    % Converting Inputs
    ph  = x(1);
    phd = x(2);
    th  = x(3);
    thd = x(4);
    ps  = x(5);
    psd = x(6);

    % Calculating Control Torques
    Tcx = -(kpr*ph + kdr*phd);
    Tcy = (kpp*(thc - th) - kdp*thd);
    Tcz = alpha*(kpr*ph + kdr*phd);

    % Checking for Step Inptus
    if t < tStep
        TdVal = 0;
    else
        TdVal = Td;
    end

    % Defining EoM
    Y = [phd;
         (1/Ix)*(Tcx + TdVal - hs*psd - wo*hs*ph);
         thd;
         (1/Iy)*(Tcy + TdVal - 3*wo^2*(Ix - Iz)*th);
         psd;
         (1/Iz)*(Tcz + TdVal + hs*phd - wo*hs*ps)];
end
```