AE353: Design Problem 01

J.Clements

February 8, 2019

1 Goal

The code provided in `DesignProblem01` simulates the rotational motion of a spacecraft. This spacecraft has actuators that can apply torque about two different axes. This spacecraft also has sensors—an inertial measurement unit (IMU)—to measure its angular velocity. The spacecraft starts with some random angular velocity. The goal is to achieve a 1 by 3 vector angular velocity of $\vec{\omega} = \begin{bmatrix} 3 & 0 & 0 \end{bmatrix}$ rad/s.

2 Model

The rotational motion of the spacecraft, if it is modeled as a single rigid body, is governed by the ordinary differential equations

$$\tau_1 = J_1 \dot{w}_1 - (J_2 - J_3) w_2 w_3 \tag{1}$$

$$0 = J_2 \dot{w}_2 - (J_3 - J_1) w_3 w_1 \tag{2}$$

$$\tau_3 = J_3 \dot{w}_3 - (J_1 - J_2) w_1 w_2, \tag{3}$$

where w_1, w_2, w_3 are the components of angular velocity, J_1, J_2, J_3 are the principle moments of inertia, and τ_1, τ_3 are the two different torques that can be applied.

3 Linearization to State Space Model

In order to determine the linear model, we first define the State, x, Input, u, and Output, y as well as the state. Our goal is to simplify our equations of motion to the following:

$$\dot{x} = Ax + Bu, \quad y = Cx + Du$$

$$x = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} rad/s, \;\; u = \begin{bmatrix} \tau_1 \\ \tau_3 \end{bmatrix} N - m, \;\; y = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} rad/s$$

Gathering all this information, we move to MatLab to utilize Jacobian() in order to find A, B, C, and D for the non-linearized model above:

```
syms j1 j2 j3 w1 w2 w3 wd1 wd2 wd3 t1 t3
x = [w1; w2; w3];
u = [t1; t3];
y = x;
eqn1 = t1 == j1*wd1 - (j2 - j3)*w2*w3;
eqn2 = 0 == j2*wd2 - (j3 - j1)*w3*w1;
eqn3 = t3 == j3*wd3 - (j1 - j2)*w1*w2;
xd = [solve(eqn1, wd1); solve(eqn2, wd2); solve(eqn3, wd3)];
%Jacobian to find values for A, B, C, D
A = jacobian(xd, x)
B = jacobian(xd, u)
C = jacobian(y, x)
D = jacobian(y, u)
```

$$C_1 = \frac{J_2 - J_3}{J_1} \quad C_2 = \frac{J_1 - J_3}{J_2} \quad C_3 = \frac{J_1 - J_2}{J_3}$$

$$A = \begin{bmatrix} 0 & (\omega_3 \times C_1) & (\omega_2 \times C_1) \\ -(\omega_3 \times C_2) & 0 & -(\omega_1 \times C_2) \\ (\omega_2 \times C_3) & (\omega_1 \times C_3) & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{J_1} & 0 \\ 0 & 0 \\ 0 & \frac{1}{J_3} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

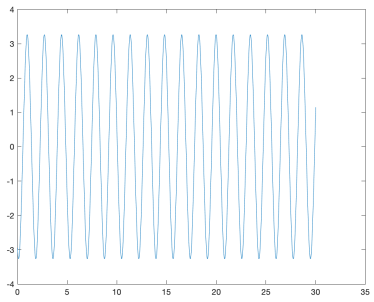
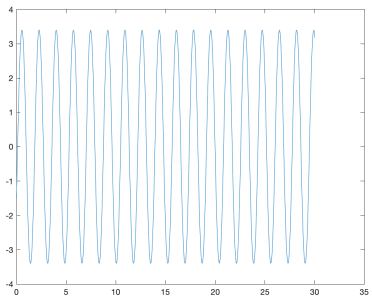
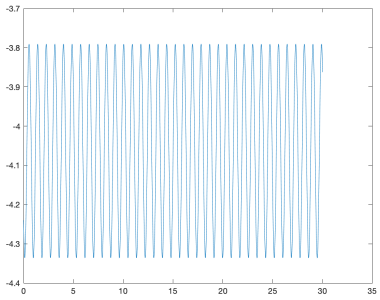
4 Test.m Configuration

First, we created our *Test.m*, a file which will be used in future reference to call the necessary functions to run the simulation.

```
clear, close all, clc
DesignProblem01('Controller','datafile', 'Data.mat','display',false)

load('Data.mat')
figure(1)
plot(processdata.t, processdata.w1)
saveas(gcf,['w1_statefeedback.png']);

figure(2)
plot(processdata.t, processdata.w2)
saveas(gcf,['w2_statefeedback.png']);

figure(3)
plot(processdata.t, processdata.w3)
saveas(gcf,['w3_statefeedback.png']);
```

5 Zero Input Analysis

For **Zero Input**, we set *actuators.tau1* and *actuators.tau3* $= 0$ in the *Controller.m* file. Due to the fact that there are no controllers providing input to the system and producing changes in angular velocity vectors, it may be hypothesized that the satellite will not converge nor be asymptotically stable. The resulting figures for zero input are posted on the next page:

As we can see from the graphs, the satellite does not converge, is not asymptotically stable, and continues oscillating for the duration of the simulation. Additionally, we found some values of which $F = eig(A - B * K) > 0$, therefore the system is not asymptotically stable.

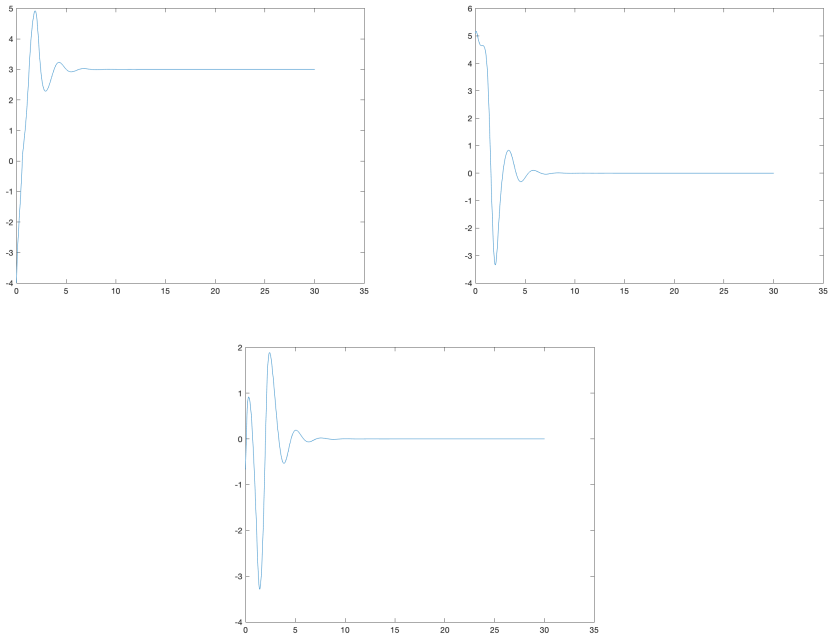
6 State Feedback Analysis

State feedback uses the linearized state space model to control the input, in this case τ_1 and τ_3, to obtain a desired angular velocity. If the controller is operating optimally, the initial position, $x(t_0)$, and initial velocity, $\dot{z}(t_0)$, will not change the final outcome but will affect the behavior of the process towards equilibrium. We modified the *Controller.m* code to add the linearized system. Since the controller observes the angular velocity from the sensors and makes slight adjustments in the x- and z-directions from gyroscopic wheels, we can hypothesize the satellite under the state feedback input will be asymptotically stable and converge. Note, the clip below will only show the additions to the Matlab code provided to us:

```
1 function [actuators,data] = runControlSystem(sensors, references, parameters, data)
2     w_e = [3 ; 0 ; 0];
3     constant1=(parameters.J2-parameters.J3)/parameters.J1;
4     constant2=(parameters.J1-parameters.J3)/parameters.J2;
```

```
    constant3=(parameters.J1-parameters.J2)/parameters.J3;
    K = [2, 1, 1; 1, 1, 1];
    A=[0 constant1*w_e(3) constant1*w_e(2);-constant2*w_e(3) 0 constant2*w_e(1);
        constant3*w_e(2) constant3*w_e(1) 0];
    B=[1/parameters.J1 0; 0 0; 0 1/parameters.J3];
    F = eig(A-B*K); %--- Eig() test to check asymptotic stability
    actuators.tau1 = -K(1,1)*(sensors.w1 - w_e(1)) - K(1,2)*(sensors.w2 - w_e(2)) -
        K(1,3)*(sensors.w3 - w_e(3));
    actuators.tau3 = -K(2,1)*(sensors.w1 - w_e(1)) - K(2,2)*(sensors.w2 - w_e(2)) -
        K(2,3)*(sensors.w3 - w_e(3));
end
```

By using the *Test.m* software to run the simulation, the following three figures are created and show promising results:



As we can see, the satellite converges on the desired points of angular velocity $\vec{\omega} = [3\ 0\ 0]$. In conclusion, we can see that the implementation of state space linearization results in an asymptotically stable system and helps the satellite converge to a desired velocity. Additionally, *DesignProblem01.m* code randomizes the initial position and angular velocities; the initial components affect how quickly the system converges and the magnitude of overshoot. For example, separate trials of running the same *Test.m* program reveals varying rise time, time-to-peak, overshoot, and settling times. Ultimately, state space model linearization succeeds in changing motion into an asymptotically stable system.