Table of Contents

Roshan Jaiswal-Ferri

```
%Section - 01
%Aero 421 HW5: 5/23/25
```

Workspace Prep

```
%warning off
format long     %Allows for more accurate decimals
close all;      %Clears all
clear all;      %Clears Workspace
clc;            %Clears Command Window
```

Problem 1

```
J = diag([1200,2000,2800]);

ts = 100; %seconds
zeta = 0.65;

wn = log(0.02*sqrt(1-zeta^2))/-zeta/ts;
beta = atan(sqrt(1-zeta^2)/zeta);
tr = (pi-beta)/wn/sqrt(1-zeta^2);

Kp = 2*J*eye(3)*wn^2;
Kd = J*eye(3)*2*zeta*wn;

disp('Kp =')
disp(num2str(Kp))
disp(' ')
disp('Kd =')
disp(num2str(Kd))
```

```
Kp =
9.95626            0            0
      0      16.5938            0
      0            0      23.2313

Kd =
100.4771            0            0
```

```
       0      167.4619             0
       0             0      234.4466
```

Setup

```
% Spacecraft Orbit Properties (given)
global mu
mu = 398600; % km^3/s^2
h = 53335.2; % km^2/s
e = 0; % none
Omega = 0*pi/180; % radians
inc = 98.43*pi/180; % radians
omega = 0*pi/180; % radians
nu = 0*pi/180; % radians

a = h^2/mu/(1 - e^2);
%orbital_period = 2*pi*sqrt(a^3/mu);

% Torque
T = [0;0;0];

% Set/Compute initial conditions
% intial orbital position and velocity
[r_ECI_0, v_ECI_0] = coes2rvd(a,e,rad2deg(inc),0,omega,nu,mu);

% Compute inital F_LVLH basis vectors in F_ECI components based on F_LVLH
% definition

rV = r_ECI_0; %Position Vector km
vV = v_ECI_0; %Vel Vector km/s

%Converting to F'LVLH

Zlvlh = -(rV/norm(rV));
Ylvlh = -(cross(rV,vV)/norm(cross(rV,vV)));
Xlvlh = cross(Ylvlh,Zlvlh);

%Creating Matrix with new vectors

Clvlh_eci = [Xlvlh, Ylvlh, Zlvlh]';
C_b_ECI_0 = Clvlh_eci;

% Initial Euler angles relating F_body and F_LVLH (given)
phi_0 = 0;
theta_0 = 0;
psi_0 = 0;

E_b_LVLH_0 = [phi_0; theta_0; psi_0];

% Initial Quaternion relating F_body and F_LVLH (given)
%q_b_LVLH_0 = [0; 0; 0; 1];

% Compute initial C_LVLH_ECI_0, C_b_LHVL_0, and C_b_ECI_0 rotaiton matrices
```

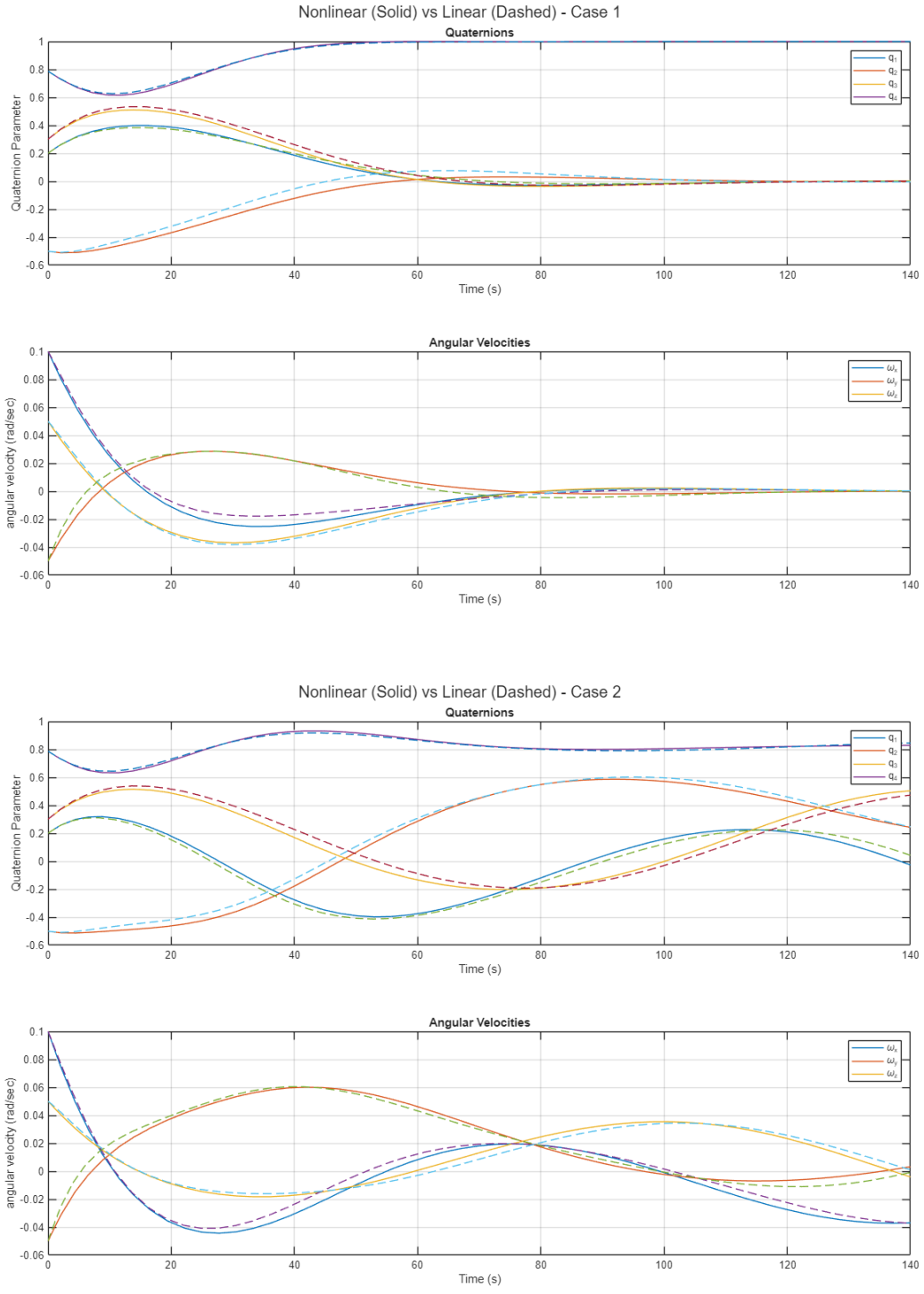
```
% Initial Euler angles relating body to ECI
% E_b_ECI_0 = C2EulerAngles(C_b_ECI_0);
%E_b_ECI_0 = rotm2eul(C_b_ECI_0);

% Initial quaternion relating body to E
%q_b_ECI_0 = -rotm2quat(C_b_ECI_0);
```

Problem 2: Defining Useful Stuff

```
% Initial body rates of spacecraft (given)
w_b_ECI_0 = [0.1; -0.05; 0.05];

e0 = [0.2;-0.5;0.3];
n0 = sqrt(1-e0'*e0);

q_b_ECI_0 = [e0;n0];
q0 = [n0, e0'];
E_b_ECI_0 = quat2eul(q0);

tspan = 140; %orbital_period;

qc1 = [0;0;0;1];

e02 = [-0.2;0.5;0.2];
n02 = sqrt(1-e02'*e02);
qc2 = [e02;n02];

for i = 1:2
    if i == 1
        q_C = qc1;
    else
        q_C = qc2;
    end

    output(i) = sim("ADCS_Design_RJF_NonLinear.slx");
end

out = output(1);
out2 = output(2);

for i = 1:2
    if i == 1
        q_C = qc1;
    else
        q_C = qc2;
    end

    outputL(i) = sim("ADCS_Design_RJF_Linear.slx");
end

outL = outputL(1);
out2L = outputL(2);
```

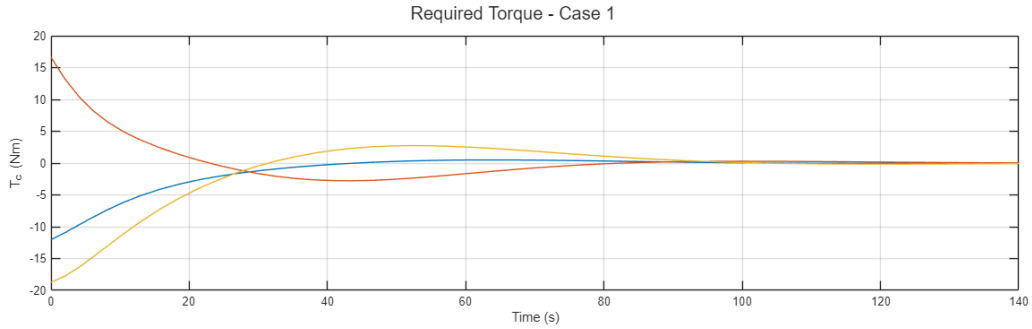
Plot Results

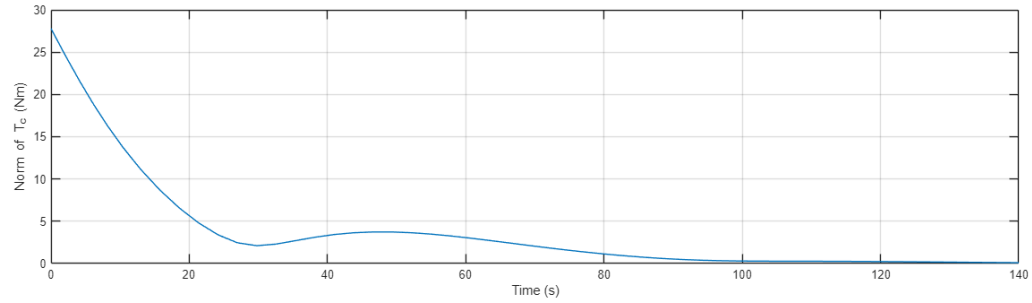
```
figure('Name','Case 1')
subplot(2,1,1)
plot(out.tout, out.q_b_ECI(:,2:5))
hold on
plot(outL.tout, outL.q_b_ECI(:,2:5),'--')
title('Quaternions')
ylabel('Quaternion Parameter')
legend('q_1','q_2','q_3','q_4')
grid on
xlabel('Time (s)')

subplot(2,1,2)
plot(out.w_b_ECI(:,1), out.w_b_ECI(:,2:4))
hold on
plot(outL.w_b_ECI(:,1), outL.w_b_ECI(:,2:4),'--')
title('Angular Velocities')
ylabel('angular velocity (rad/sec)')
legend('\omega_x','\omega_y','\omega_z')
grid on
xlabel('Time (s)')

sgtitle('Nonlinear (Solid) vs Linear (Dashed) - Case 1')

figure('Name','Case 2')
subplot(2,1,1)
plot(out2.tout, out2.q_b_ECI(:,2:5))
hold on
plot(out2L.tout, out2L.q_b_ECI(:,2:5),'--')
title('Quaternions')
ylabel('Quaternion Parameter')
legend('q_1','q_2','q_3','q_4')
grid on
xlabel('Time (s)')

subplot(2,1,2)
plot(out2.w_b_ECI(:,1), out2.w_b_ECI(:,2:4))
hold on
plot(out2L.w_b_ECI(:,1), out2L.w_b_ECI(:,2:4),'--')
title('Angular Velocities')
ylabel('angular velocity (rad/sec)')
legend('\omega_x','\omega_y','\omega_z')
grid on
xlabel('Time (s)')

sgtitle('Nonlinear (Solid) vs Linear (Dashed) - Case 2')
```

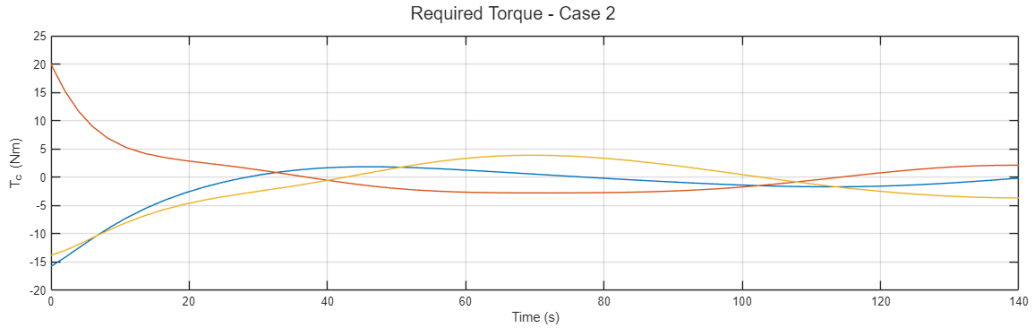


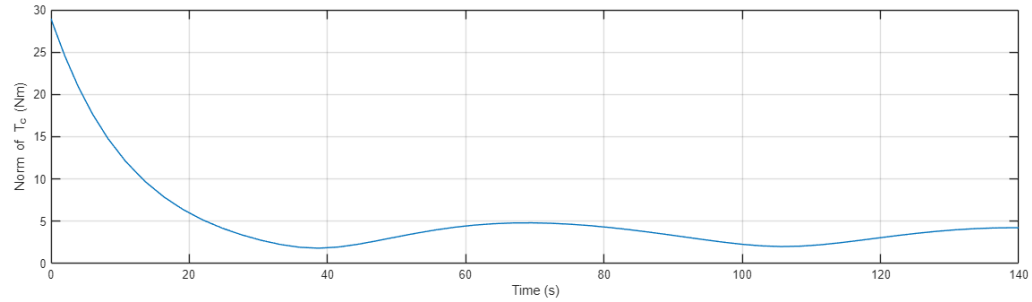

```
Tc = squeeze(outL.T.signals.values);
Tnorm = vecnorm(Tc);

figure('Name','Required Torque - Case 1')
subplot(2,1,1)
```

```
plot(outL.tout,Tc)
xlabel('Time (s)')
ylabel('T_c (Nm)')
grid on

subplot(2,1,2)
plot(outL.tout,Tnorm)
xlabel('Time (s)')
ylabel('Norm of T_c (Nm)')
grid on

sgtitle('Required Torque - Case 1')


Tc2 = squeeze(out2L.T.signals.values);
Tnorm2 = vecnorm(Tc2);

figure('Name','Required Torque - Case 2')
subplot(2,1,1)
plot(out2L.tout,Tc2)
xlabel('Time (s)')
ylabel('T_c (Nm)')
grid on

subplot(2,1,2)
plot(out2L.tout,Tnorm2)
xlabel('Time (s)')
ylabel('Norm of T_c (Nm)')
grid on

sgtitle('Required Torque - Case 2')
```









Published with MATLAB® R2024b