AAE 440: Spacecraft Attitude Dynamics

PS8\*

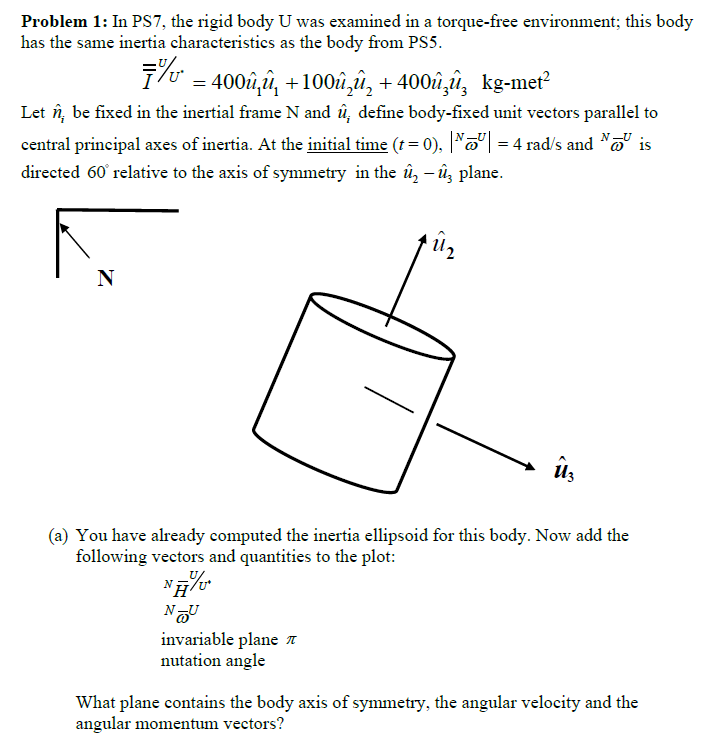
Dr. Howell

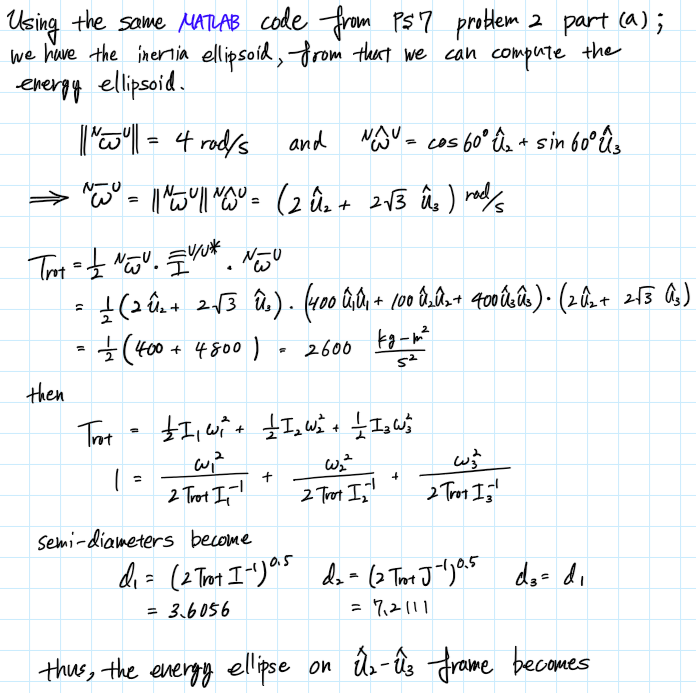
School of Aeronautical and Astronautical

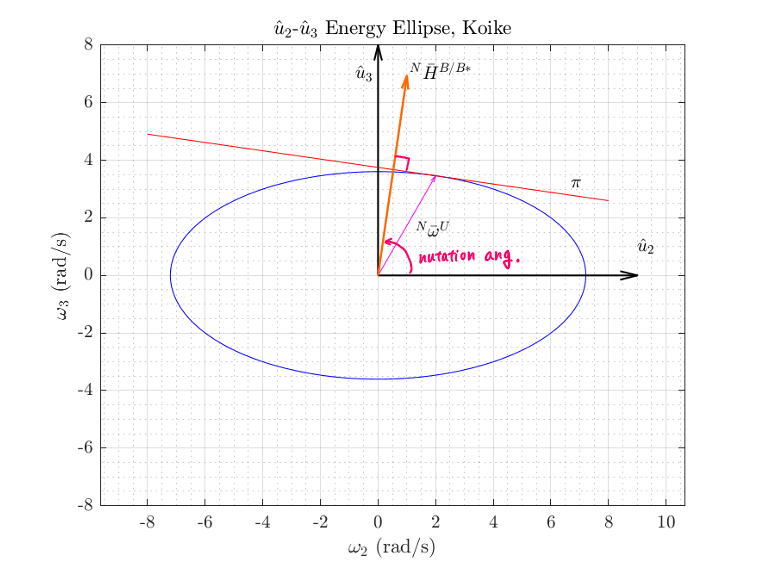
Purdue University

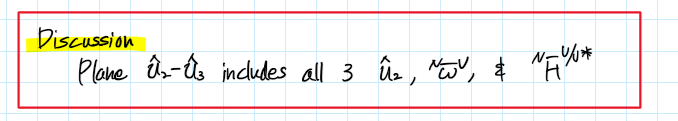
Tomoki Koike

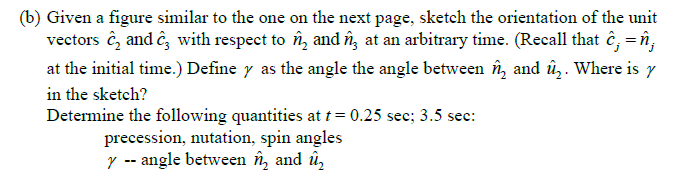
Friday April 3, 2020

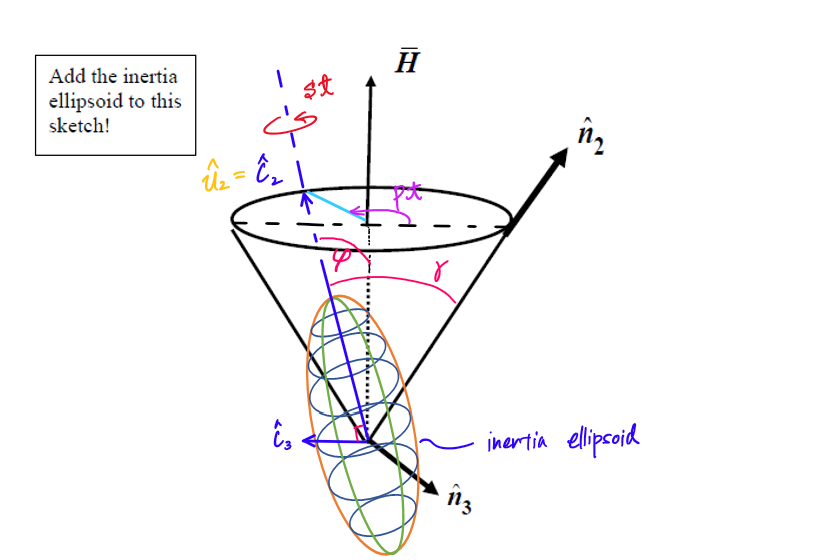


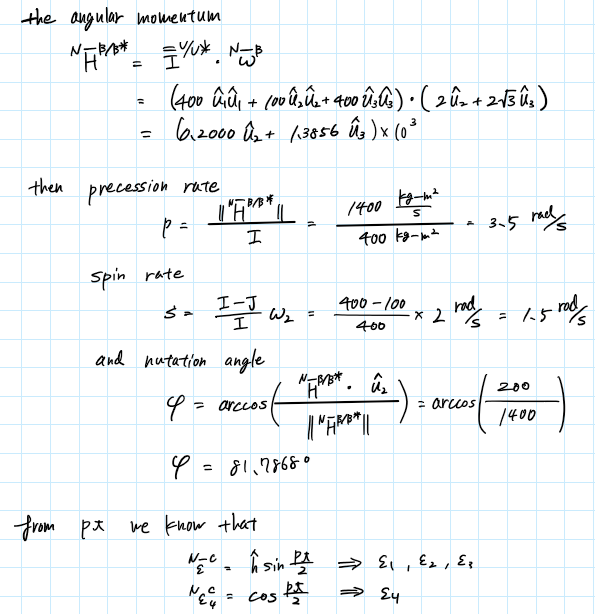


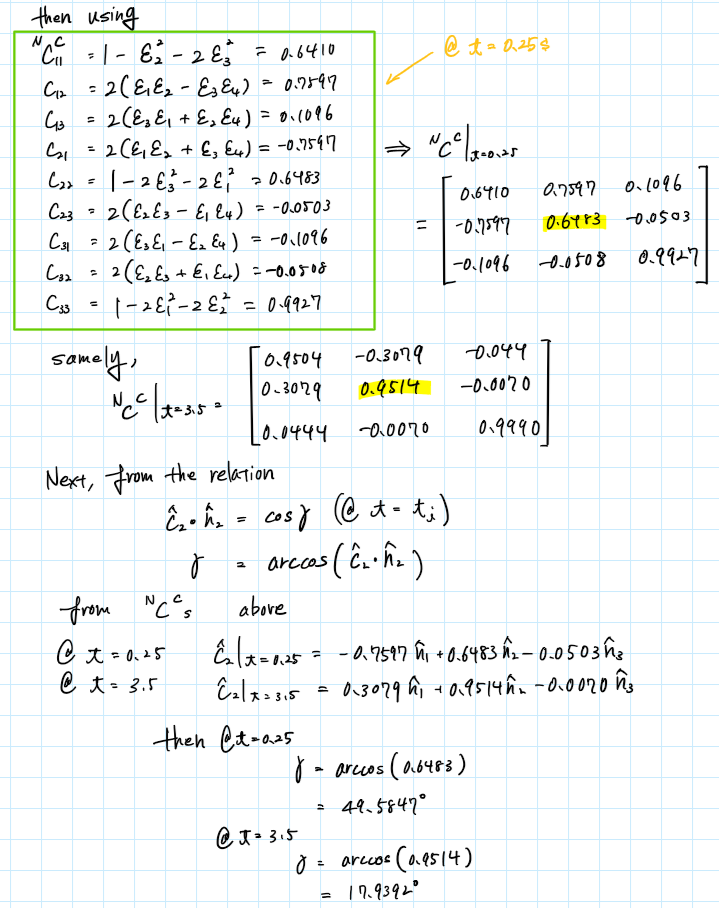


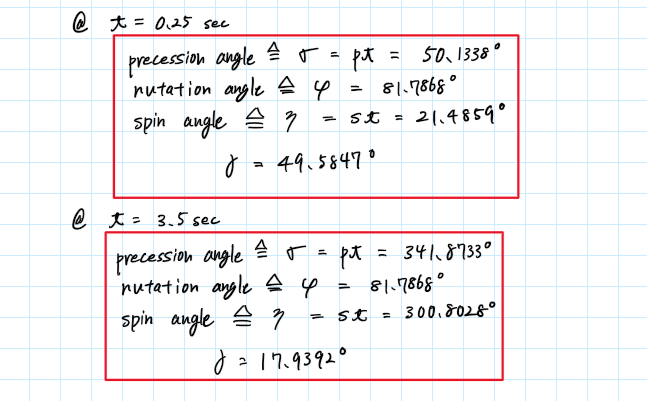


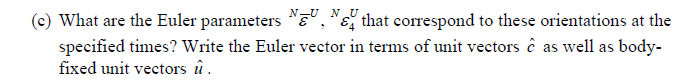


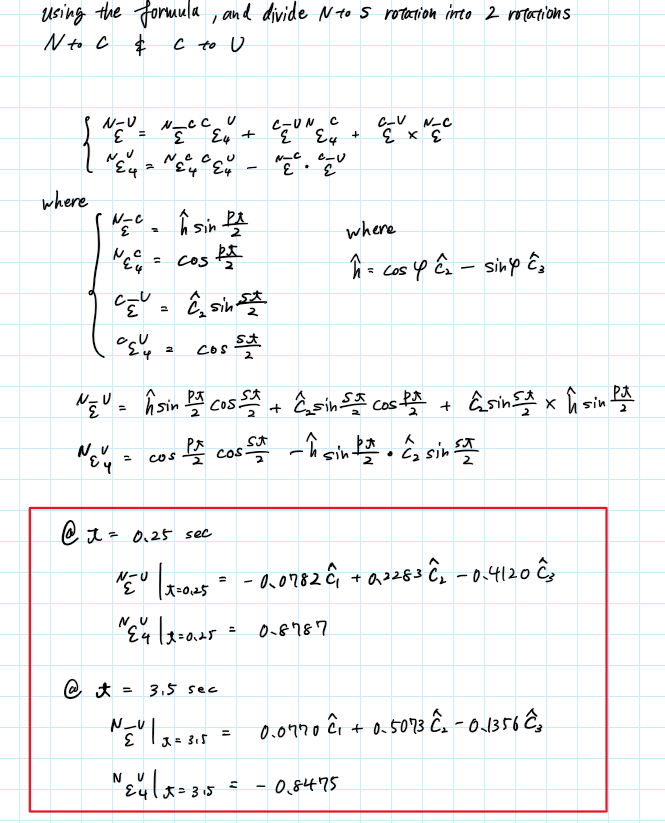


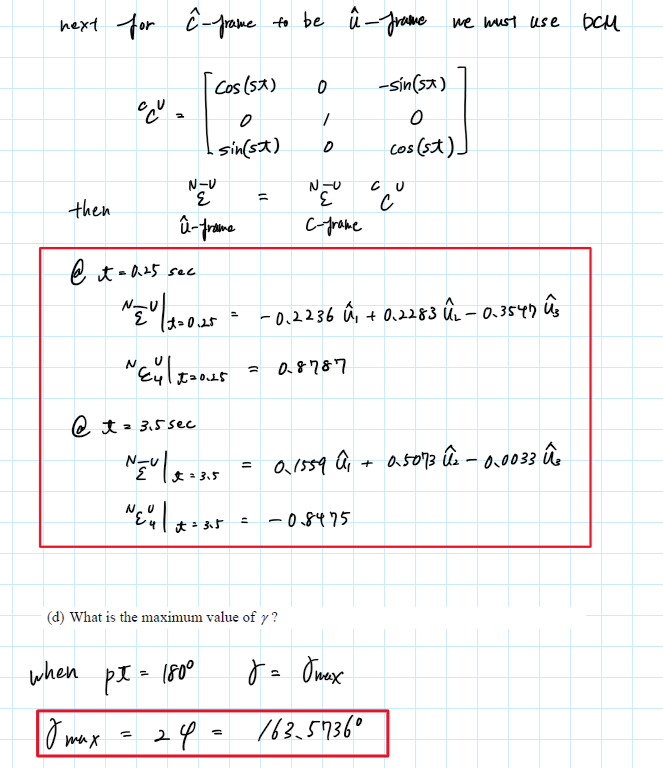


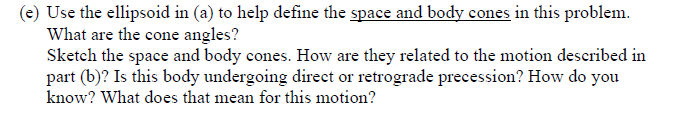


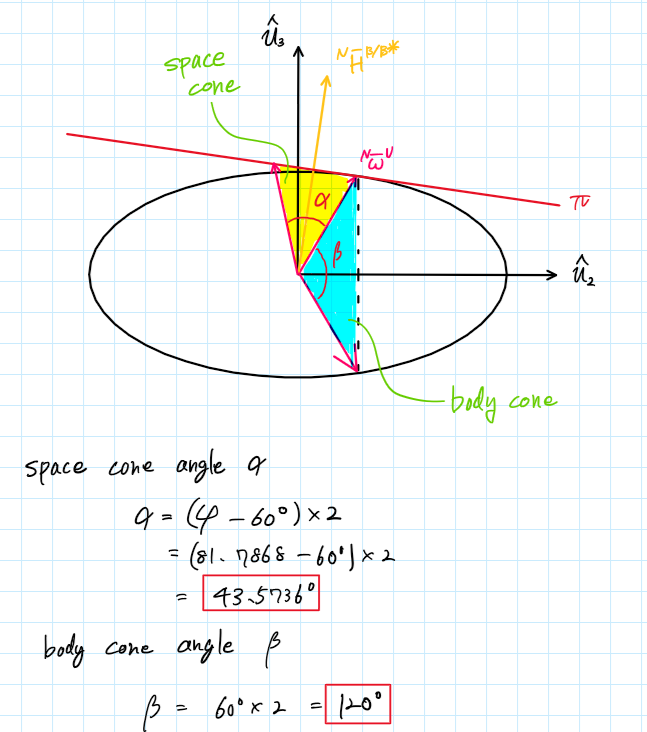


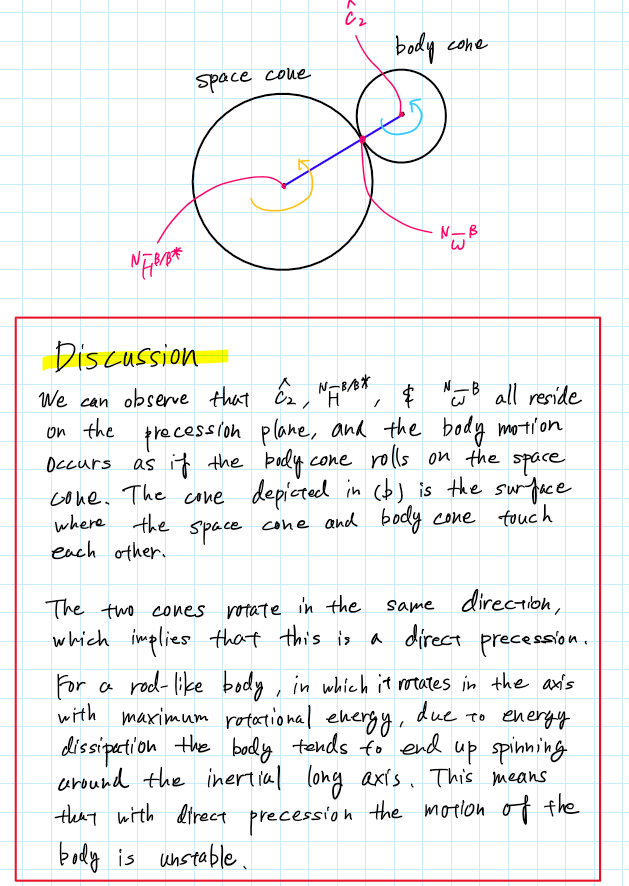


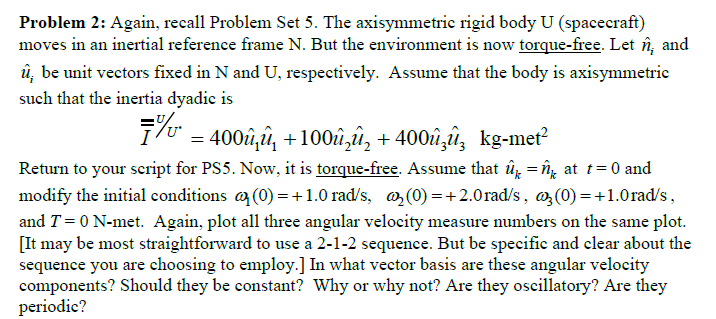




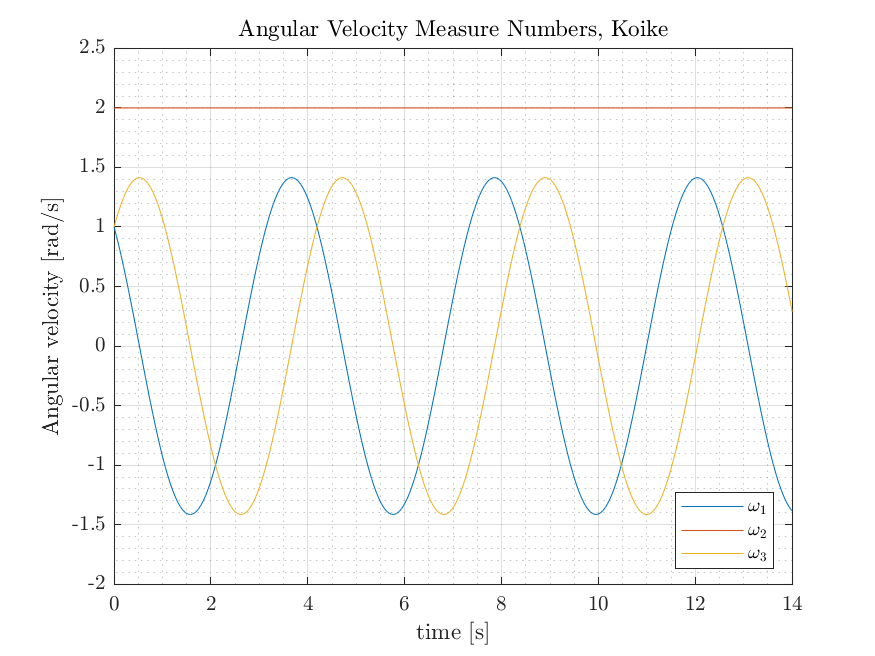


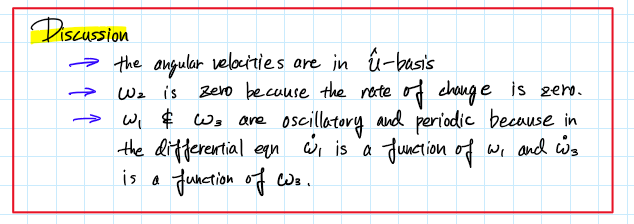


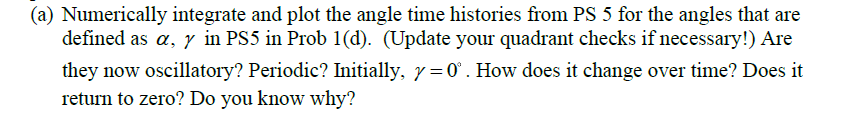


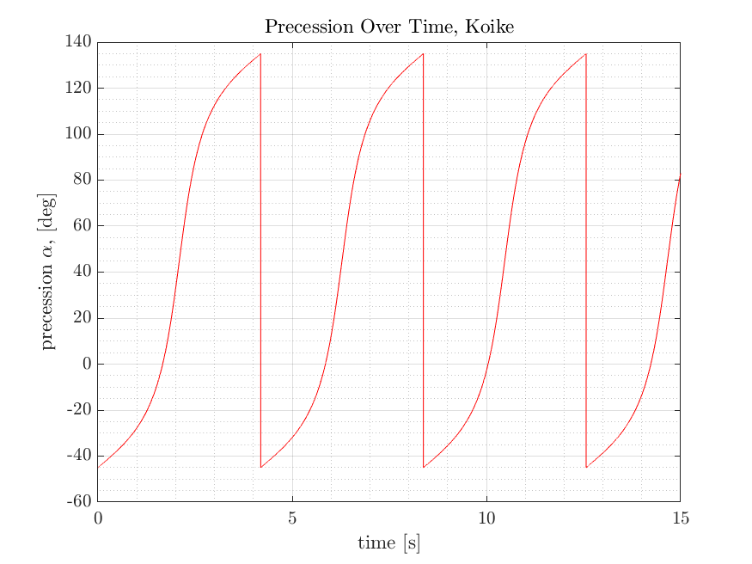
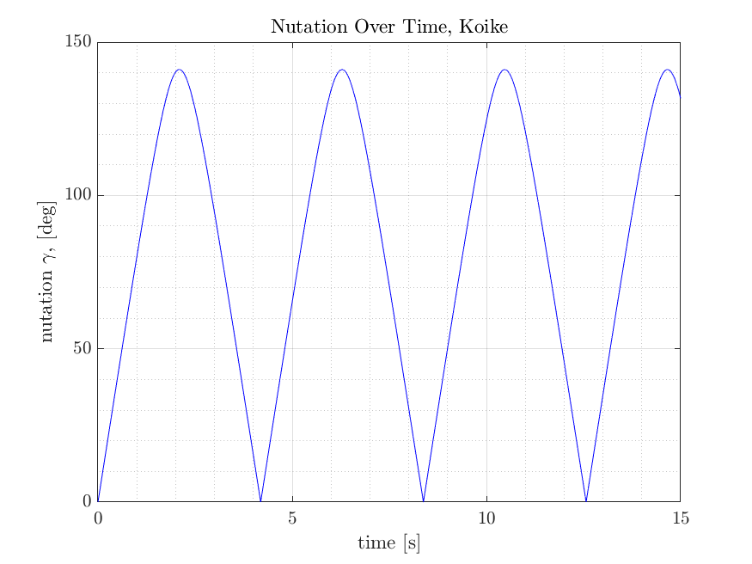


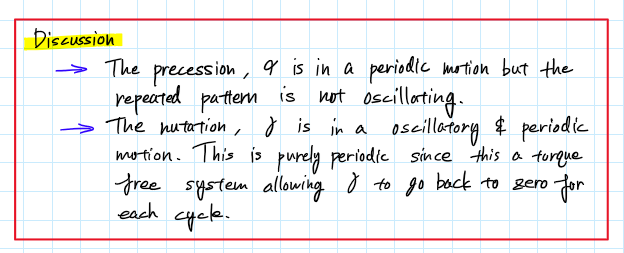


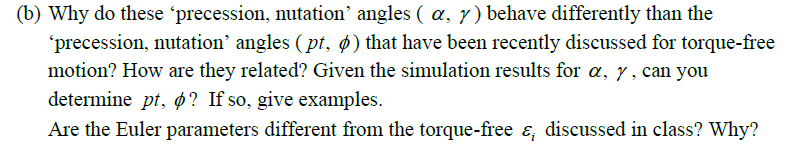


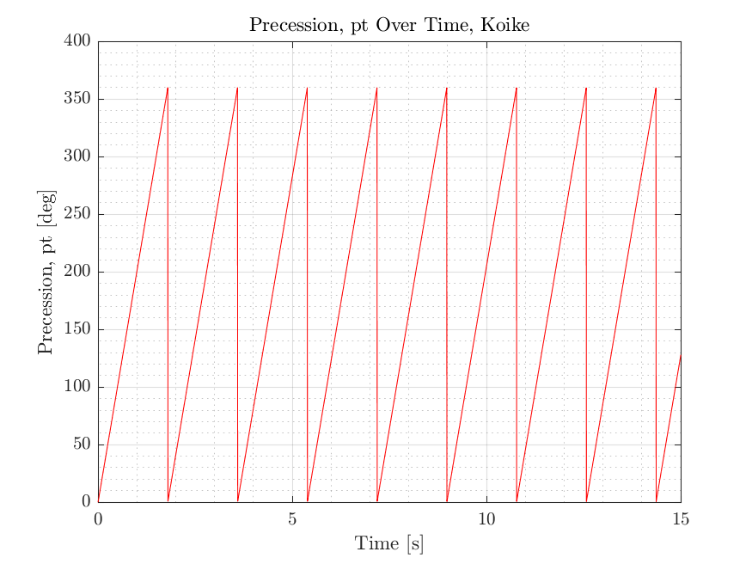
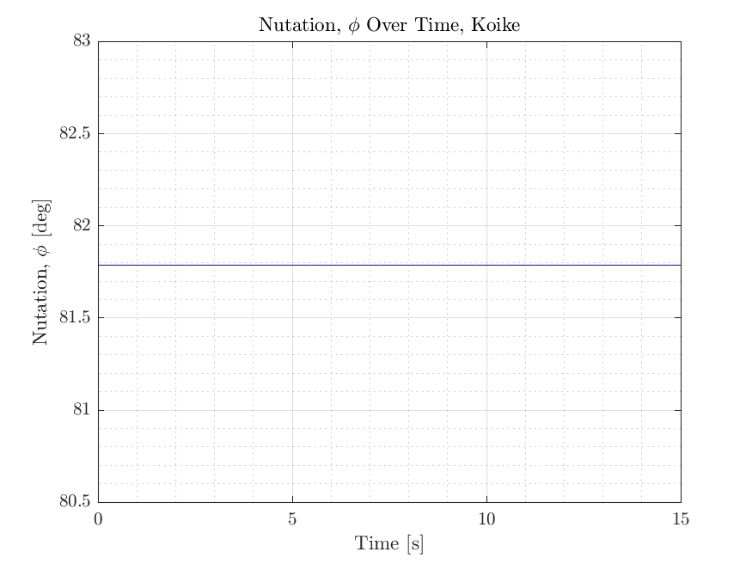


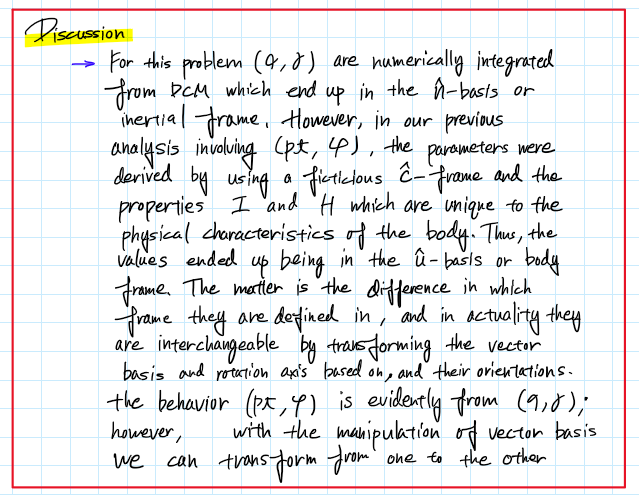


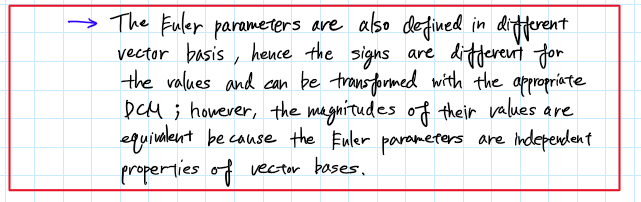


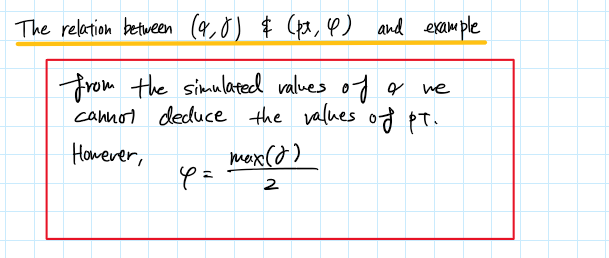


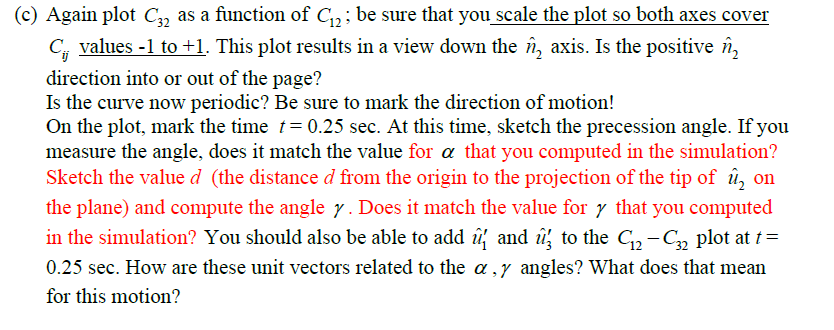


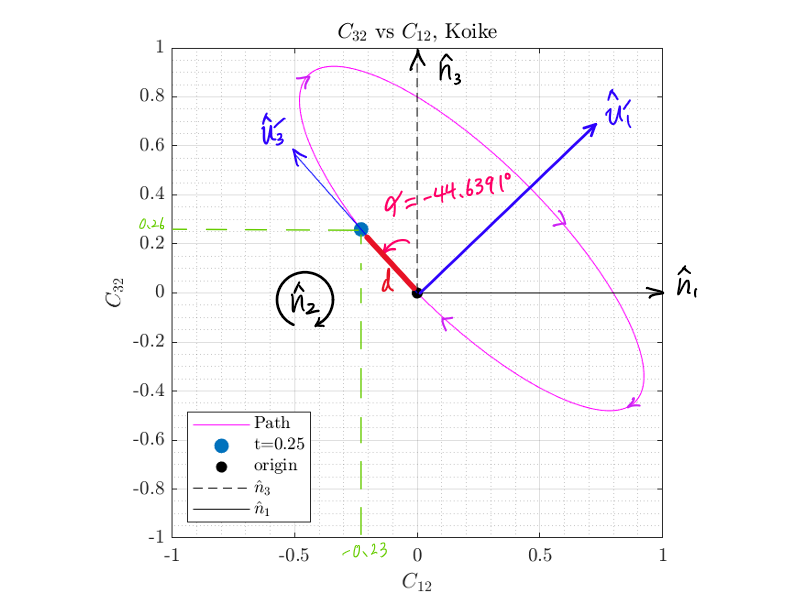


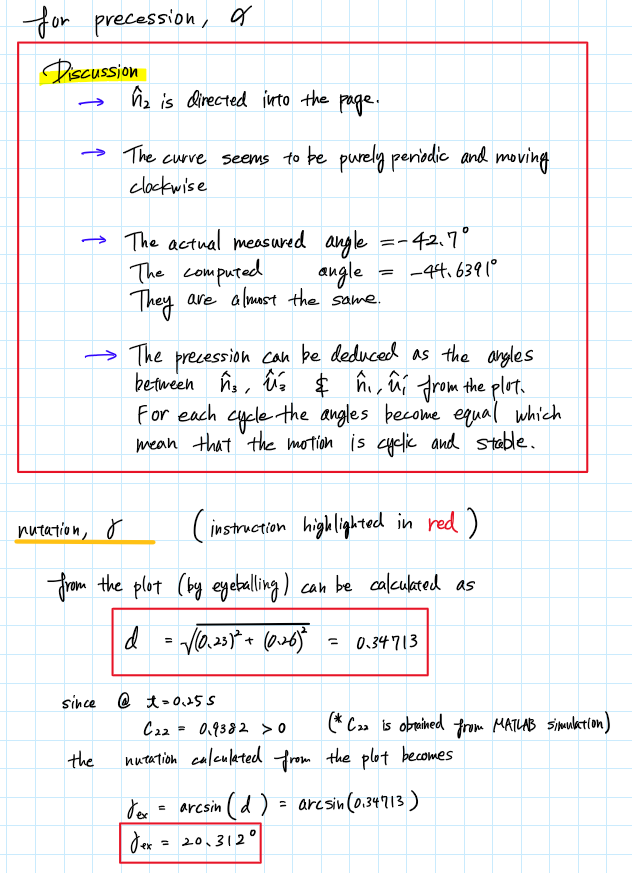


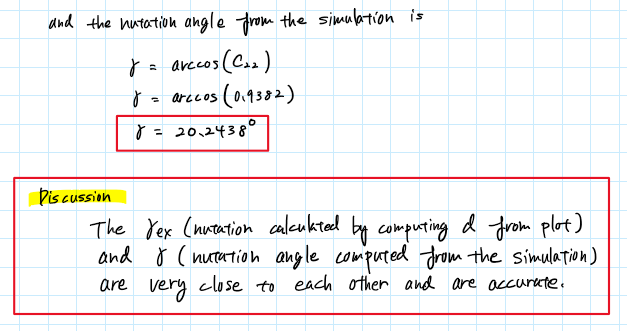


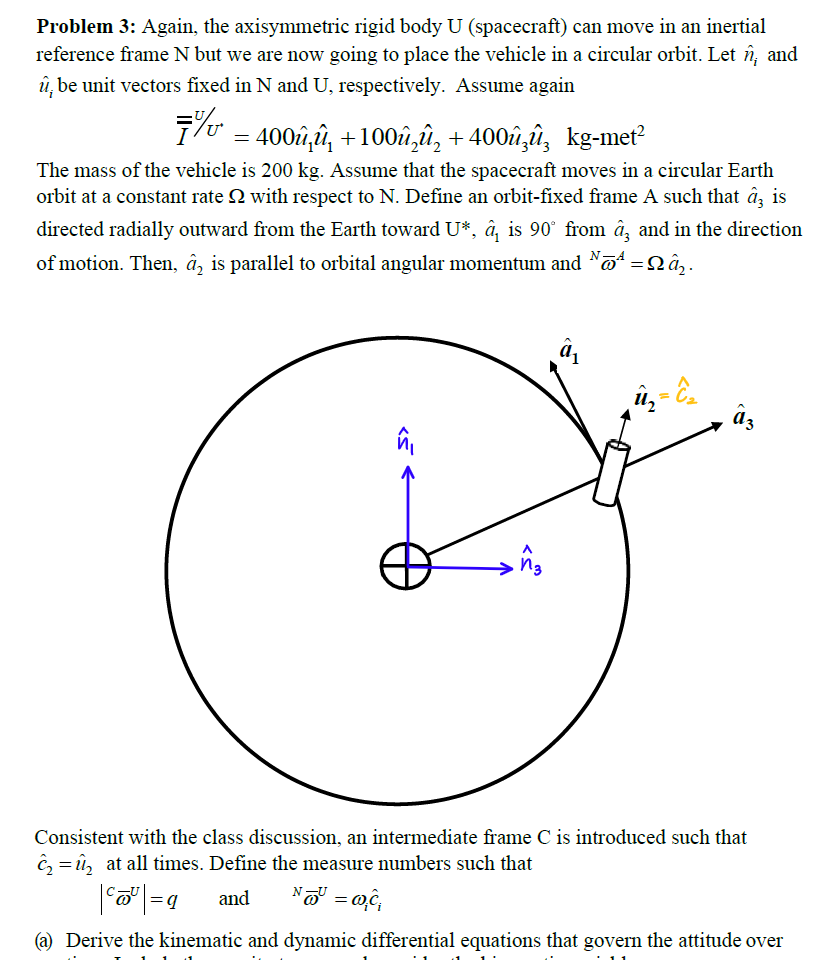
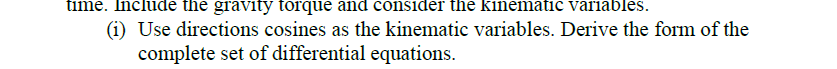


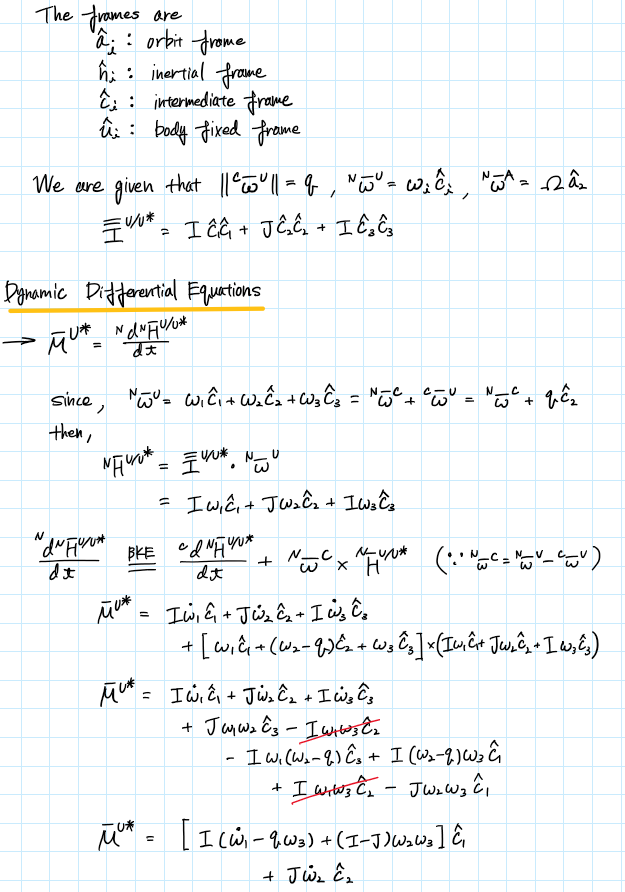


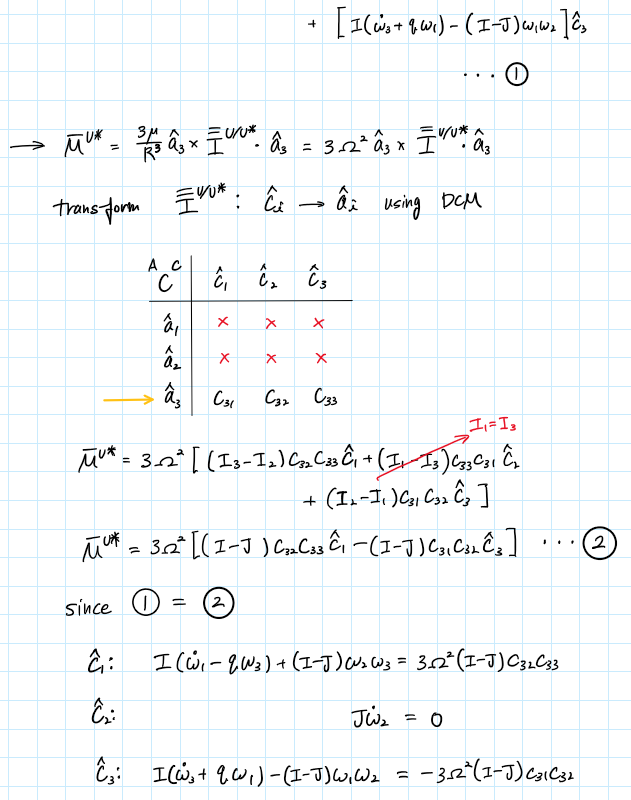


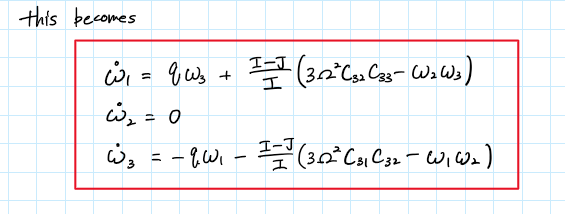


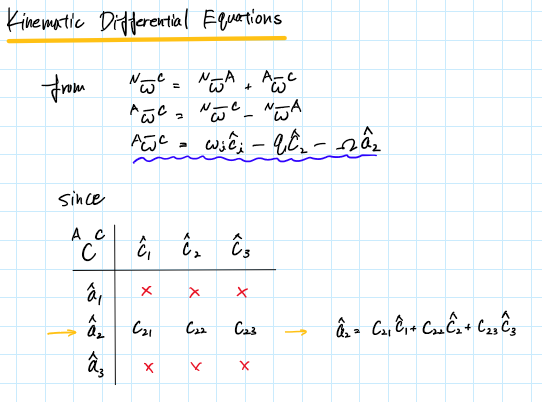


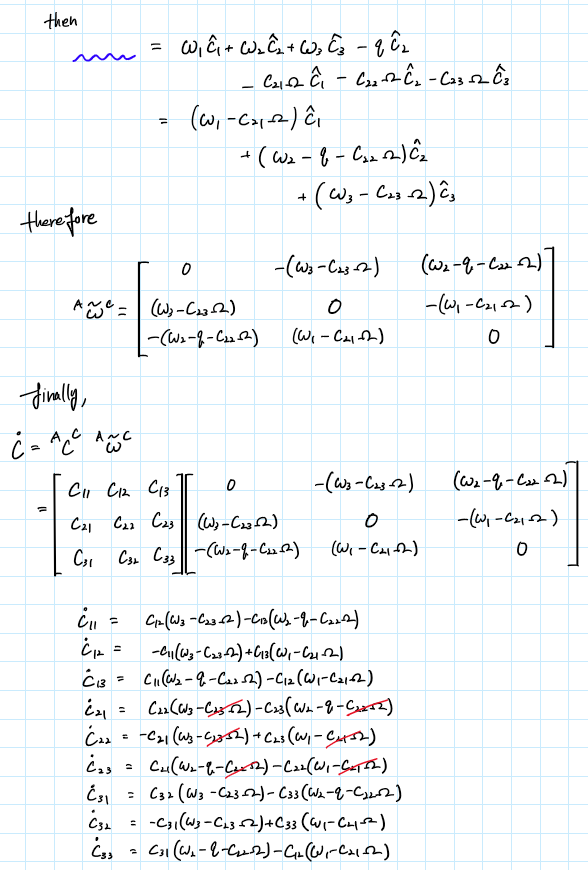
  


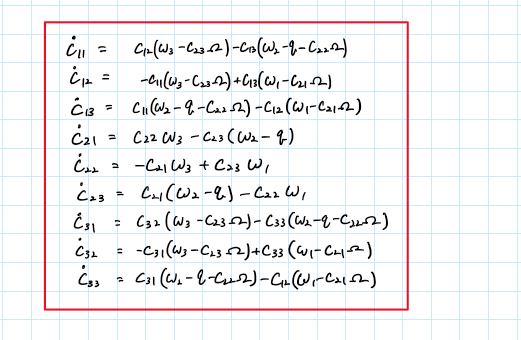




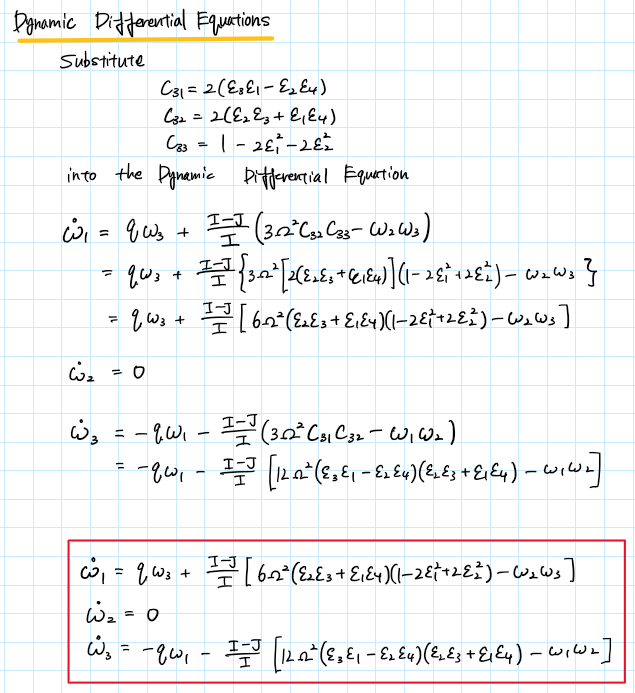


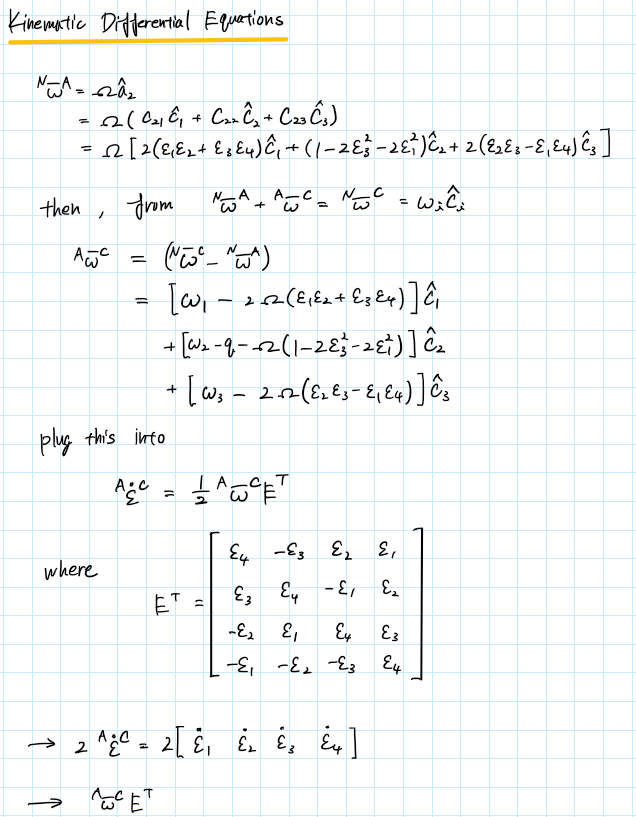


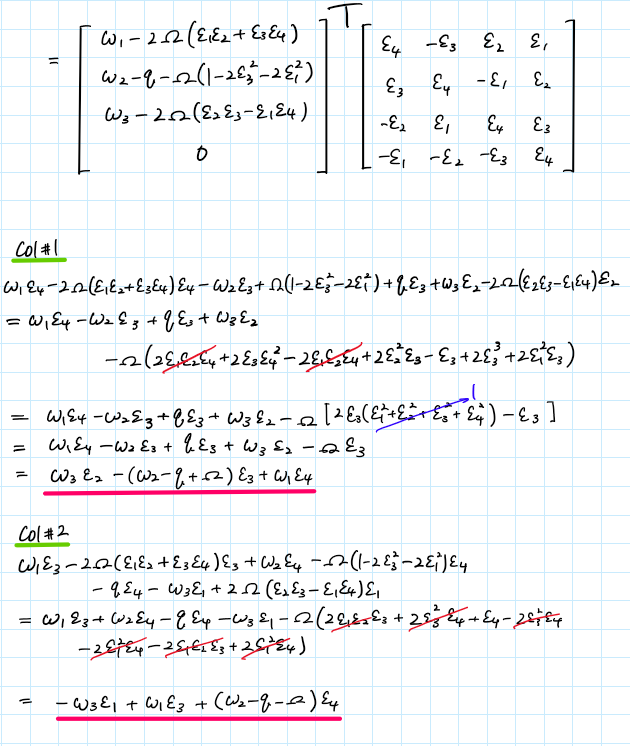


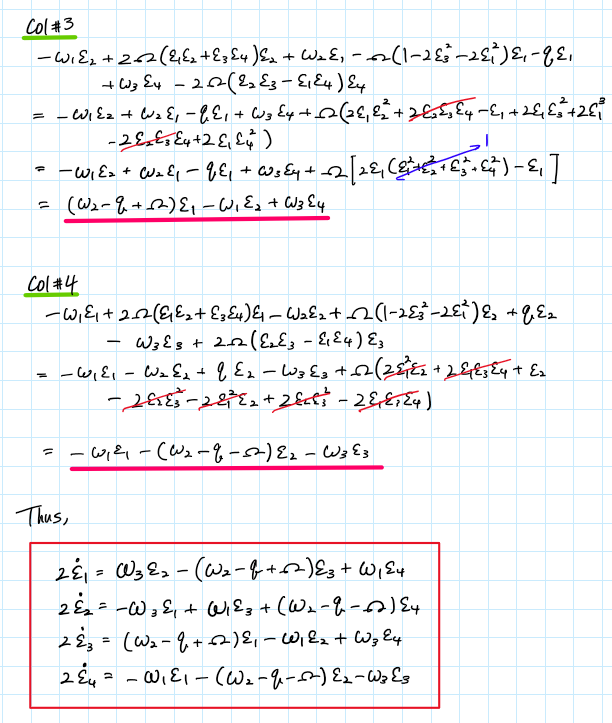


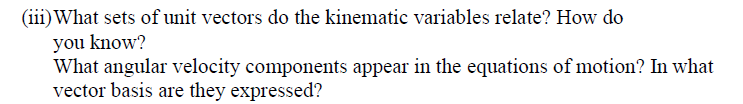


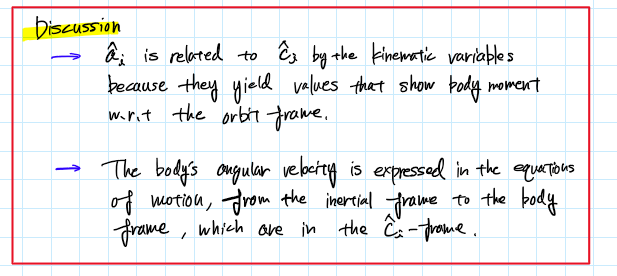


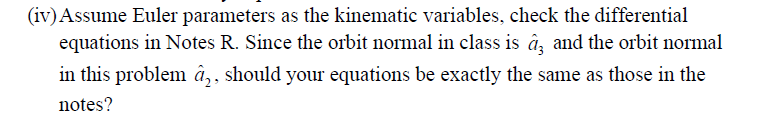


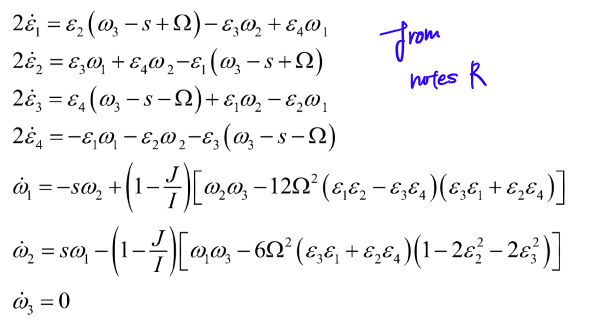


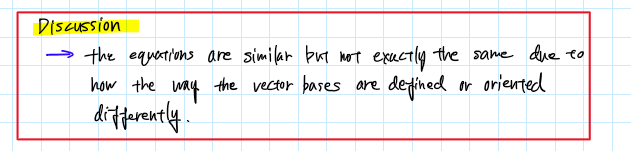


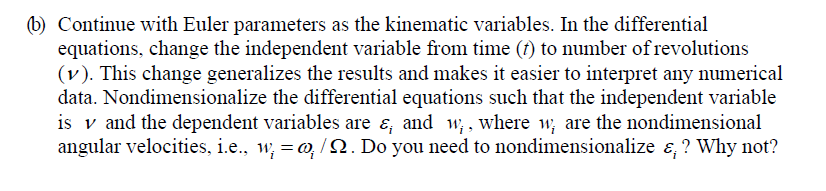


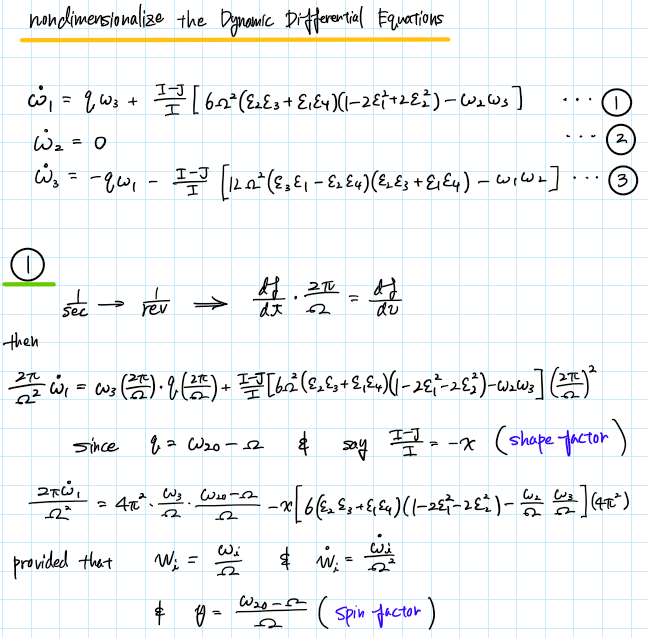


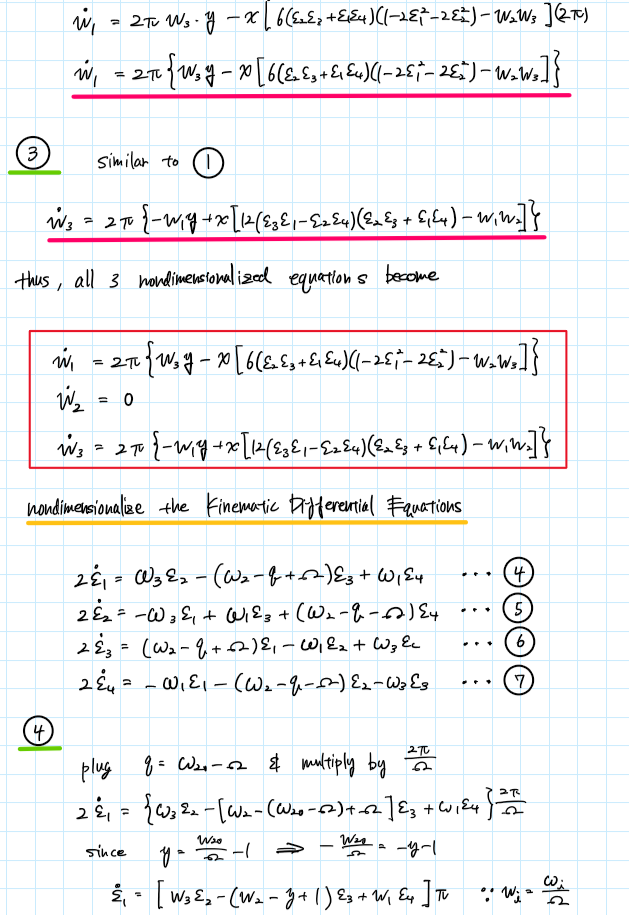


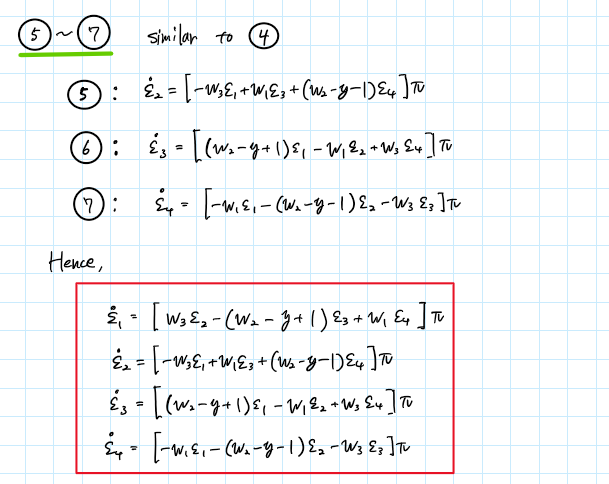












Appendix

## **AAE440 HW8 PROBLEM 1 MATLAB**

clear all; close all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Spring\AAE440\MATLAB\outputs\HW8';

set(groot, 'defaulttextinterpreter',"latex");

set(groot, 'defaultAxesTickLabelInterpreter',"latex");

set(groot, 'defaultLegendInterpreter',"latex");

% Arrow drawing function

drawArrow = @(x,y,varargin) quiver( x(1),y(1),x(2)-x(1),y(2)-y(1),0, varargin{:} );

#### (a)

% Given properties

I\_body = [400 0 0; 0 100 0; 0 0 400]; % [kg-m2]

I = I\_body(1,1); J = I\_body(2,2);

w\_NU\_mag = 4; % magnitude of angular velocity [rad/s]

w\_NU\_hat = [0 cosd(60) sind(60)]; % angle of w\_NU relative to u\_2 [deg]

w\_NU = w\_NU\_mag\*w\_NU\_hat;

% Kinetic rotational energy

Trot = 0.5\*w\_NU\*I\_body\*w\_NU.'

% Semi-diameters of energy ellipsoid

d1 = sqrt(2\*Trot\*I^(-1))

d2 = sqrt(2\*Trot\*J^(-1))

d3 = d1

% Plotting the inertial ellipsoid

theta = 0:0.01:2\*pi;

u\_str = ["$\hat{u}\_1$","$\hat{u}\_2$","$\hat{u}\_3$"];

% u2-u3

fig1 = figure("Renderer","painters");

plot(d2\*cos(theta), d3\*sin(theta), 'b')

title('$\hat{u}\_2$-$\hat{u}\_3$ Energy Ellipse, Koike')

xlabel('$\omega\_2$ (rad/s)')

ylabel('$\omega\_3$ (rad/s)')

hold on

drawArrow([0 9], [0 0],'k','linewidth',1); text(9,1,u\_str(2),"Interpreter","Latex");

drawArrow([0 0], [0 8],'k','linewidth',1); text(-0.8,7,u\_str(3),"Interpreter","Latex");

% Angular velocity

drawArrow([0 w\_NU(2)], [0 w\_NU(3)],'color','#FD07EA');

text(1.3,1.5,'${}^N\bar{\omega}^U$','Interpreter','Latex');

% Invariable plane PI

[a, b] = line\_tangent2ellipse(w\_NU(2),w\_NU(3),d2,d3);

x = -8:0.1:8;

y = a\*x + b;

plot(x,y,'-r'); text(6.7,3.2,'$\pi$','Interpreter',"latex");

% H\_body

drawArrow([0 1.0],[0 1.0\*(-1/a)],'color','#FF6800','linewidth',1.2)

text(1.1,7,'${}^N\bar{H}^{B/B\*}$')

hold off

xlim([-9 9]); ylim([-8 8]);

grid on; grid minor; box on; axis equal;

saveas(fig1, fullfile(fdir,'P1-a-u2\_u3\_EN-ellipse.png'));

### (b)

% Angular momentum

H\_NU = I\_body\*w\_NU.';

H\_NU\_mag = norm(H\_NU);

% Computing p, s, and phi

p = H\_NU\_mag/I;

s = (I - J)/I\*w\_NU(2);

phi = acos(H\_NU(2)/H\_NU\_mag)

phi\_deg = rad2deg(phi)

% Computing the precession, nutation, and spin angles

% @ t = 0.25

t = 0.25;

sigma = p\*t

sigma\_deg = rad2deg(sigma)

eta = s\*t

eta\_deg = rad2deg(eta)

% @ t = 3.5

t = 3.5;

sigma = mod(p\*t,2\*pi)

sigma\_deg = rad2deg(sigma)

eta = mod(s\*t,2\*pi)

eta\_deg = rad2deg(eta)

h\_hat\_U = H\_NU/H\_NU\_mag;

h\_hat\_C = [0 cos(phi) -sin(phi)];

% gamma

syms t1

e\_NC = h\_hat\_C\*sin(p\*t1/2);

e4\_NC = cos(p\*t1/2);

% DCM

% @ t= 0.25

e\_NC\_025 = double(subs(e\_NC,t1,0.25));

e4\_NC\_025 = double(subs(e4\_NC,t1,0.25));

C\_NC\_025 = DCM\_from\_EulerPara([e\_NC\_025 e4\_NC\_025])

% @ t= 3.5

e\_NC\_35 = double(subs(e\_NC,t1,3.5));

e4\_NC\_35 = double(subs(e4\_NC,t1,3.5));

C\_NC\_35 = DCM\_from\_EulerPara([e\_NC\_35 e4\_NC\_35])

gamma\_025 = acosd(C\_NC\_025(2,2))

gamma\_35 = acosd(C\_NC\_35(2,2))

### (c)

% Euler parameters

c2\_hat = [0 1 0];

syms t

e\_NC = h\_hat\_C\*sin(p\*t/2);

e4\_NC = cos(p\*t/2);

e\_CU = c2\_hat\*sin(s\*t/2);

e4\_CU = cos(s\*t/2);

e\_NU = e\_NC\*e4\_CU + e\_CU\*e4\_NC + cross(e\_CU,e\_NC);

e4\_NU = e4\_NC\*e4\_CU - dot(e\_NC,e\_CU);

% C-frame

% @ t = 0.25

e\_NU\_025\_C = double(subs(e\_NU,t,0.25))

e4\_NU\_025 = double(subs(e4\_NU,t,0.25))

% @ t = 3.5

e\_NU\_35\_C = double(subs(e\_NU,t,3.5))

e4\_NU\_35 = double(subs(e4\_NU,t,3.5))

% U-frame

syms t2

C\_CU = [cos(s\*t2) 0 -sin(s\*t2);

0 1 0;

sin(s\*t2) 0 cos(s\*t2)];

% @ t = 0.25

e\_NU\_025\_U = double(e\_NU\_025\_C\*subs(C\_CU,t2,0.25))

C\_NU\_025\_1 = double(C\_NC\_025\*subs(C\_CU,t2,0.25))

C\_NU\_025\_2C = DCM\_from\_EulerPara([e\_NU\_025\_C e4\_NU\_025])

C\_NU\_025\_2U = DCM\_from\_EulerPara([e\_NU\_025\_U e4\_NU\_025])

% @ t = 3.5

e\_NU\_35\_U = double(e\_NU\_35\_C\*subs(C\_CU,t2,3.5))

### (d)

gamma\_max = 2\*phi\_deg;

### (e)

alpha = 2\*(phi\_deg - 60)

### ADDITIONAL (FOR PROBLEM 2)

tspan = 0:0.005:15;

fig2 = figure("Renderer","painters")

plot(tspan,rad2deg(mod(p\*tspan,2\*pi)),'r')

title("Precession, pt Over Time, Koike")

ylabel('Precession, pt [deg]')

xlabel('Time [s]')

grid on; grid minor; box on;

saveas(fig2,fullfile(fdir,"pt\_precession.png"))

fig3 = figure("Renderer","painters")

plot(tspan,rad2deg(phi).\*ones(size(tspan)),'b')

title("Nutation, $\phi$ Over Time, Koike")

ylabel('Nutation, $\phi$ [deg]')

xlabel('Time [s]')

grid on; grid minor; box on;

saveas(fig3,fullfile(fdir,"phi\_nutation.png"))

### FUNCTION

function [slope, y\_intercept] = line\_tangent2ellipse(x1,y1,a,b)

slope = -x1/y1\*b^2/a^2;

y\_intercept = y1 - x1\*slope;

end

## **AAE440 HW8 PROBLEM 2 MATLAB**

clear all; close all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Spring\AAE440\MATLAB\outputs\HW8';

set(groot, 'defaulttextinterpreter',"latex");

set(groot, 'defaultAxesTickLabelInterpreter',"latex");

set(groot, 'defaultLegendInterpreter',"latex");

% Defining System Properties

T = 0; % Torque [N m]

I\_cm = [400 0 0;

0 100 0;

0 0 400]; % Inertia Dyadic [kg m2]

I = 400;

J = 100;

% Given Initial Conditions

w0 = [1 2 1]; % Initial AngVel [rad s-1]

e0 = [0 0 0 1]; % Initial Euler Parameters

C0 = [1 0 0 0 1 0 0 0 1]; % Initial DCM

% Numerically integrating dynamic and kinematic EOMs

tspan = [0 14]; % Integration time

y0 = [w0 e0 0 C0]; % Initial conditions

opt = odeset('RelTol', 1e-13, 'AbsTol', 1e-13); % Integration Tolerance

[t1, res1] = ode45(@(t,y) EOM(t,y,I,J,T), tspan, y0, opt);

% Plotting three angular velocity measure numbers over time

fig1 = figure("Renderer","painters");

plot(t1, res1(:,1:3))

ylabel('Angular velocity [rad/s]')

xlabel('time [s]')

title({'Angular Velocity Measure Numbers, Koike'})

axis([tspan -2 2.5])

legend('$\omega\_1$', '$\omega\_2$', '$\omega\_3$','Location',"best")

grid on; grid minor; box on;

saveas(fig1, fullfile(fdir, 'angVel\_measure\_nums.png'));

### (a)

% Plotting precession and nutation angles

tspan\_a = 0:0.001:15;

[t\_a, res\_a] = ode45(@(t,y) EOM(t,y,I,J,T), tspan\_a, y0, opt);

% Assigning computed C12, C22, and C32 to a variable

C\_a = res\_a(:,9:end);

[alpha\_a, gamma\_a] = ang\_calc\_body212(C\_a);

fig2 = figure(2);

plot(t\_a, alpha\_a, 'r')

xlabel('time [s]')

ylabel('precession $\alpha$, [deg]')

title('Precession Over Time, Koike')

grid on; grid minor; box on;

saveas(fig2, fullfile(fdir,'alpha.png'));

fig3 = figure(3);

plot(t\_a, gamma\_a, 'b')

xlabel('time [s]')

ylabel('nutation $\gamma$, [deg]')

title('Nutation Over Time, Koike')

grid on; grid minor; box on

saveas(fig3, fullfile(fdir,'gamma.png'));

### (c)

% Integration with smaller time step

tspan\_c = 0:0.05:15;

[t\_c, res\_c] = ode45(@(t,y) EOM(t,y,I,J,T), tspan\_c, y0, opt);

% C\_new = res\_c(:,9:17);

% Assigning computed C12, C22, and C32 to a variable

C11s\_c = res\_c(:,9);

C12s\_c = res\_c(:,10);

C13s\_c = res\_c(:,11);

C21s\_c = res\_c(:,12);

C22s\_c = res\_c(:,13);

C23s\_c = res\_c(:,14);

C31s\_c = res\_c(:,15);

C32s\_c = res\_c(:,16);

C33s\_c = res\_c(:,17);

% Finding the index when t=0.2 and t=1.5 and corresponding C12 C22 C32

index\_t0p25 = find(t\_c==0.25);

C11\_t025 = C11s\_c(index\_t0p25);

C12\_t025 = C12s\_c(index\_t0p25);

C13\_t025 = C13s\_c(index\_t0p25);

C21\_t025 = C21s\_c(index\_t0p25);

C22\_t025 = C22s\_c(index\_t0p25);

C23\_t025 = C23s\_c(index\_t0p25);

C31\_t025 = C31s\_c(index\_t0p25);

C32\_t025 = C32s\_c(index\_t0p25);

C33\_t025 = C33s\_c(index\_t0p25);

C\_025 = [C11\_t025 C12\_t025 C13\_t025;

C21\_t025 C22\_t025 C23\_t025;

C31\_t025 C32\_t025 C33\_t025]

% Computing gamma

gamma = acosd(C22\_t025)

% Plotting at t = 0.2 and 1.5

fig4 = figure(4);

plot(C12s\_c, C32s\_c,'-m','MarkerSize',15)

title('$C\_{32}$ vs $C\_{12}$, Koike')

xlabel('$C\_{12}$')

ylabel('$C\_{32}$')

hold on

plot(C12\_t025, C32\_t025, '.','MarkerSize',26)

plot(0,0,'.k','MarkerSize',20)

plot([0 0],[0 1],'--k')

plot([0 1],[0 0],'-k')

d = linspace(0,-0.5,100);

plot(d,d.\*(C32\_t025/C12\_t025),'-b')

hold off

legend('Path','t=0.25','origin','$\hat{n}\_3$','$\hat{n}\_1$',"location",'southwest')

grid on; grid minor; axis equal; box on;

xlim([-1 1]); ylim([-1 1]);

saveas(fig4, fullfile(fdir, 'C12\_vs\_C32.png'));

### FUNCTION

function [alphas, gammas] = ang\_calc\_body212(DCM)

%{

This function calculates the precession, nutation, and spin angle

from the provided DCM

%}

% DCM is 1 by 9 matrix with each column being C\_ij

C12s = DCM(:,2);

C21s = DCM(:,4);

C22s = DCM(:,5);

C23s = DCM(:,6);

C32s = DCM(:,8);

% Preallocating alpha and gamma arrays

alphas = zeros([length(C12s),1]);

gammas = zeros([length(C12s),1]);

% For loop to construct alpha and gamma arrays interatively

for i = 1:length(alphas)

% calculating gamma

gammas(i) = acos(C22s(i));

% calculating and verfying alpha

alpha1 = round([acos(C32s(i)/sin(gammas(i))), ...

-acos(C32s(i)/sin(gammas(i))), ...

-acos(C32s(i)/sin(gammas(i)))+2\*pi],4);

alpha2 = round([asin(C12s(i)/sin(gammas(i))), ...

pi-asin(C12s(i)/sin(gammas(i))), ...

pi-asin(C12s(i)/sin(gammas(i)))],4);

if i == 1

alphas(i) = deg2rad(-44.9829164957209);

else

alphas(i) = intersect(alpha1, alpha2);

end

gammas(i) = rad2deg(gammas(i));

alphas(i) = rad2deg(alphas(i));

end

end

function dwdt = EOM(t,y,I,J,T)

%{

inputs: 1) t: time lapse

2) y: angular velocities, euler parameters, initial

euler constraint constant, DCM

3) I: moment of inertia about the non-rotating axis

4) J: moment of inertia about the rotating axis

5) T: torque

outputs: 1) dwdt: differential y

%}

dwdt = zeros(17,1);

% Dynamics EOMs

dwdt(1) = T/I - (I-J)/I\*y(3)\*y(2);

dwdt(2) = 0;

dwdt(3) = (I-J)/I\*y(1)\*y(2);

% Kinematic EOM of angular velocities and Euler parameters

dedt1 = 0.5\*( y(1)\*y(7)-y(2)\*y(6)+y(3)\*y(5));

dedt2 = 0.5\*( y(1)\*y(6)+y(2)\*y(7)-y(3)\*y(4));

dedt3 = 0.5\*(-y(1)\*y(5)+y(2)\*y(4)+y(3)\*y(7));

dedt4 = -0.5\*( y(1)\*y(4)+y(2)\*y(5)+y(3)\*y(6));

dwdt(4) = dedt1;

dwdt(5) = dedt2;

dwdt(6) = dedt3;

dwdt(7) = dedt4;

dwdt(8) = y(4)^2 + y(5)^2 + y(6)^2 + y(7)^2 - 1; % Euler Constraint

e = [y(4) y(5) y(6) y(7)];

C = DCM\_from\_EulerPara(e); % DCM

% Kinematic EOM of angular velocities and direction cosines

dwdt(9) = C(1,2)\*y(3)-C(1,3)\*y(2);

dwdt(10) = C(1,3)\*y(1)-C(1,1)\*y(3);

dwdt(11) = C(1,1)\*y(2)-C(1,2)\*y(1);

dwdt(12) = C(2,2)\*y(3)-C(2,3)\*y(2);

dwdt(13) = C(2,3)\*y(1)-C(2,1)\*y(3);

dwdt(14) = C(2,1)\*y(2)-C(2,2)\*y(1);

dwdt(15) = C(3,2)\*y(3)-C(3,3)\*y(2);

dwdt(16) = C(3,3)\*y(1)-C(3,1)\*y(3);

dwdt(17) = C(3,1)\*y(2)-C(3,2)\*y(1);

end

function C\_mat = DCM\_from\_EulerPara(epsilons)

% Euler Parameters

epsilon1 = epsilons(1);

epsilon2 = epsilons(2);

epsilon3 = epsilons(3);

epsilon4 = epsilons(4);

% Calculating DCM from Euler Parameters

C11 = 1 - 2\*epsilon2^2 - 2\*epsilon3^2;

C12 = 2\*(epsilon1\*epsilon2 - epsilon3\*epsilon4);

C13 = 2\*(epsilon3\*epsilon1 + epsilon2\*epsilon4);

C21 = 2\*(epsilon1\*epsilon2 + epsilon3\*epsilon4);

C22 = 1 - 2\*epsilon3^2 - 2\*epsilon1^2;

C23 = 2\*(epsilon2\*epsilon3 - epsilon1\*epsilon4);

C31 = 2\*(epsilon3\*epsilon1 - epsilon2\*epsilon4);

C32 = 2\*(epsilon2\*epsilon3 + epsilon1\*epsilon4);

C33 = 1 - 2\*epsilon1^2 - 2\*epsilon2^2;

C\_mat = [C11 C12 C13; C21 C22 C23; C31 C32 C33];

end