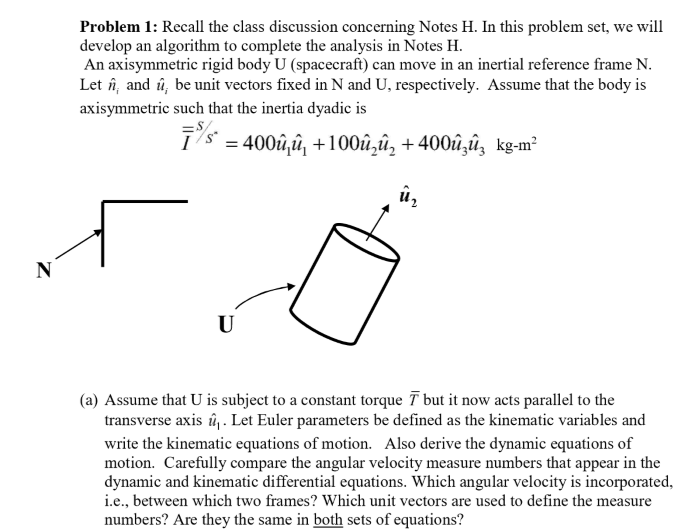
AAE 440: Spacecraft Attitude Dynamics

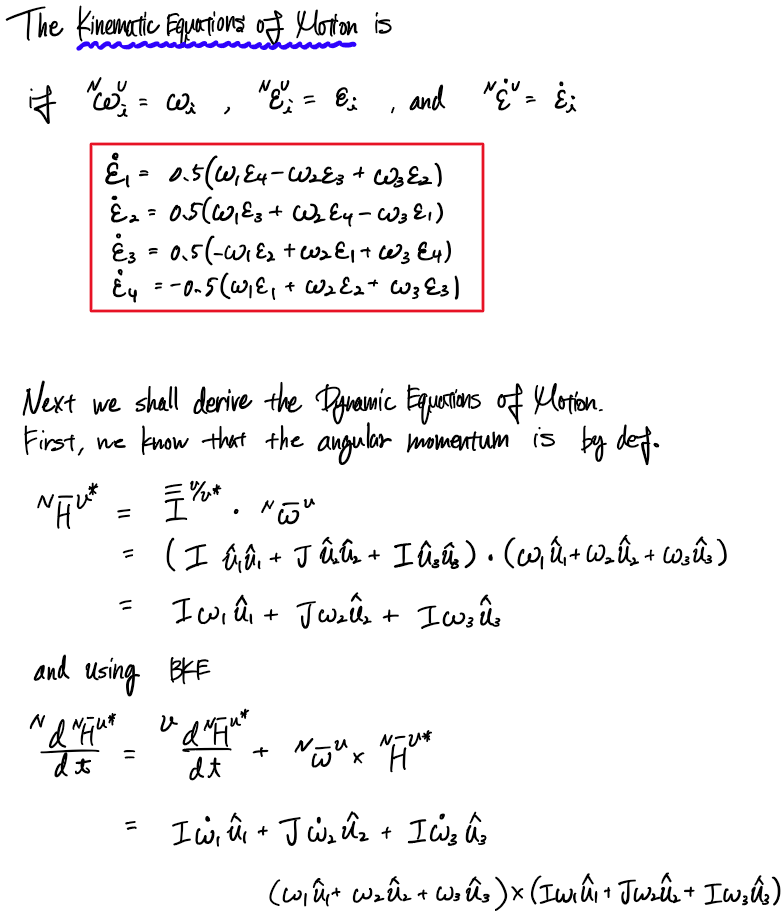
PS5\*: Dynamics and Kinematic DE Simulation

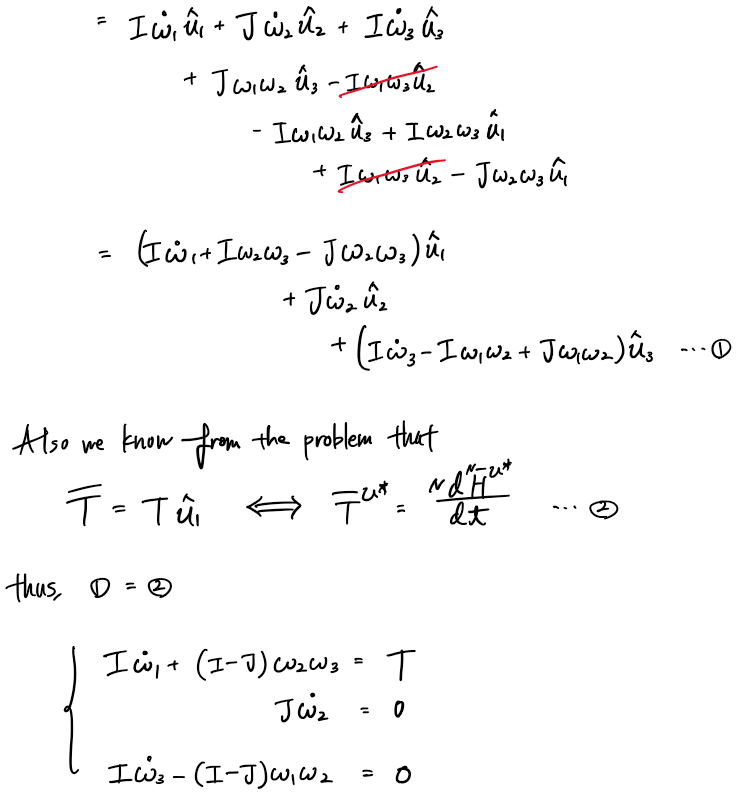
Dr. Howell

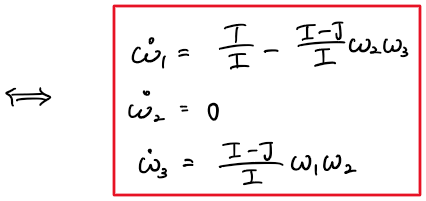
Tomoki Koike

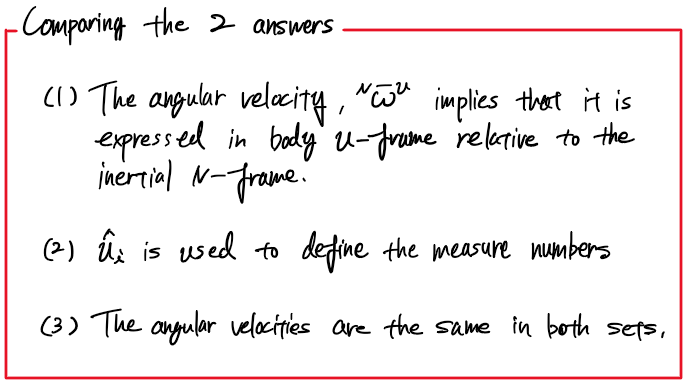
Monday March 2, 2020

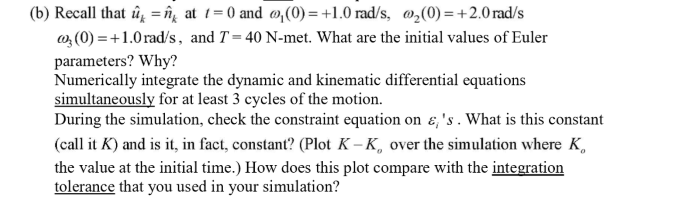


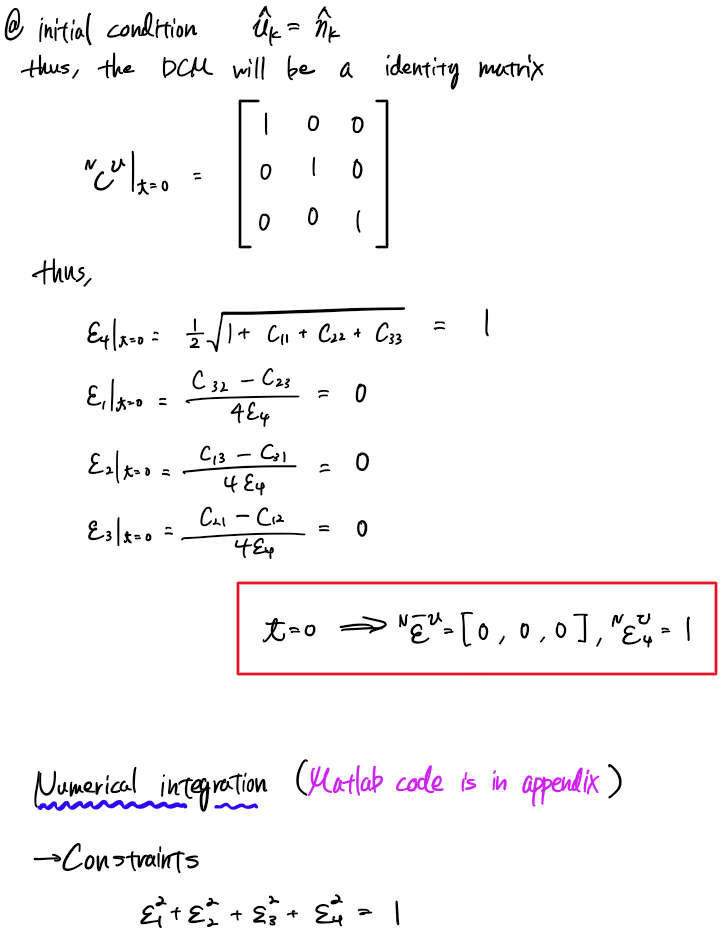


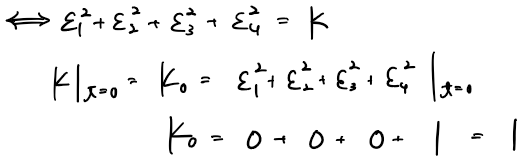


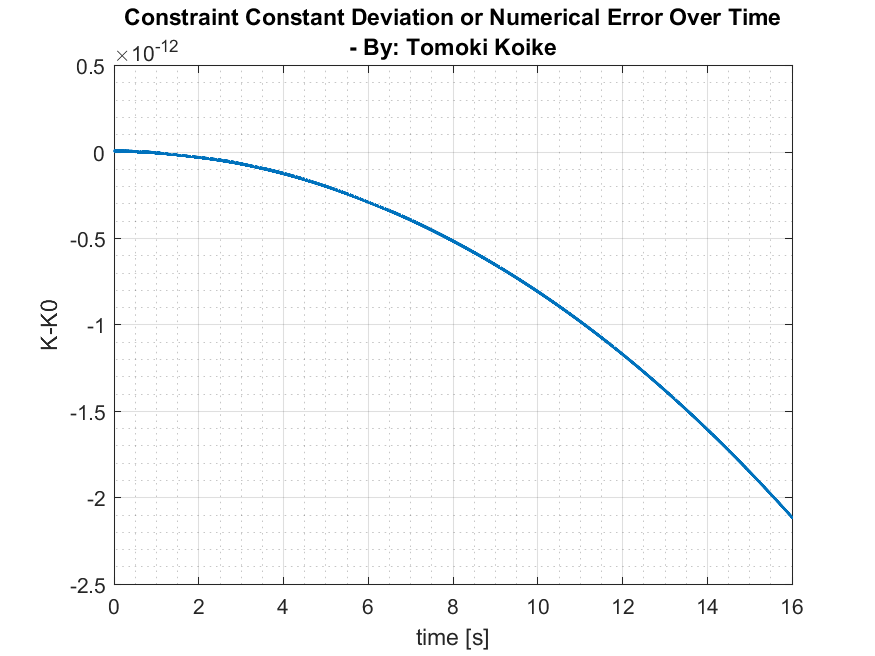


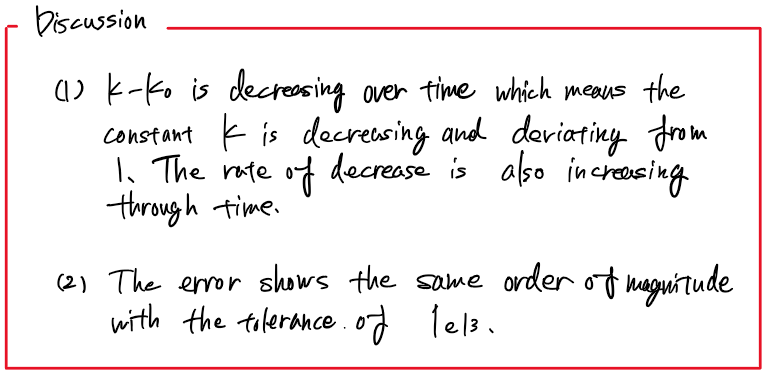


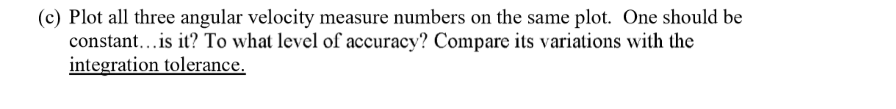


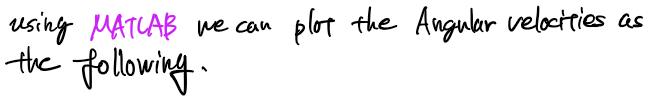


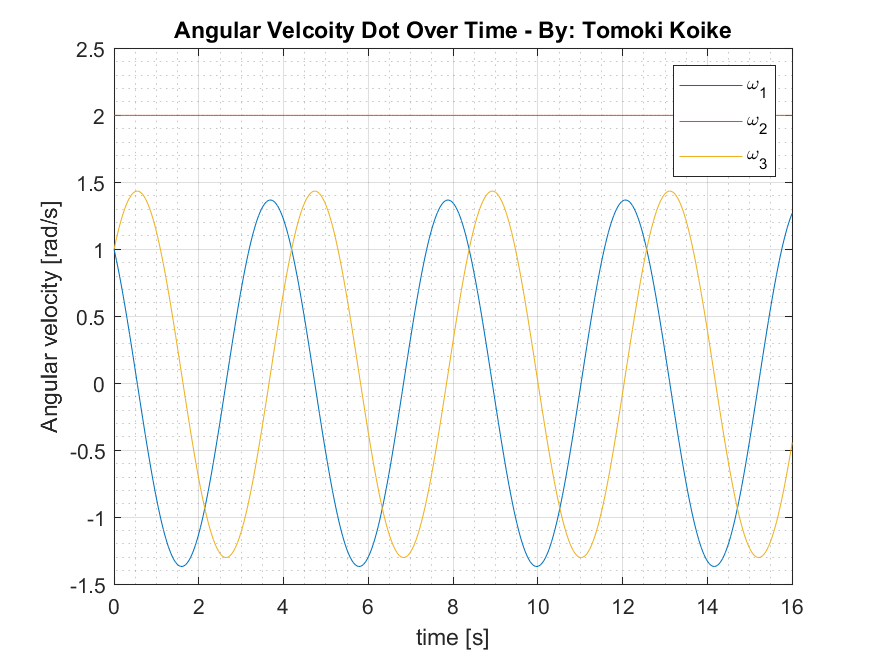


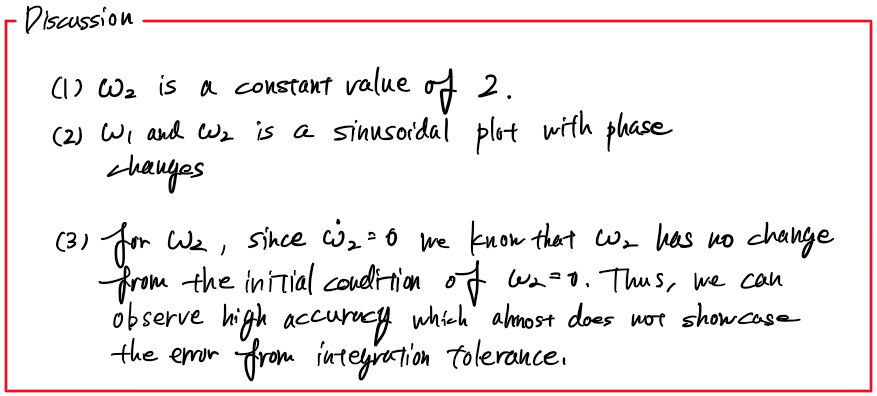


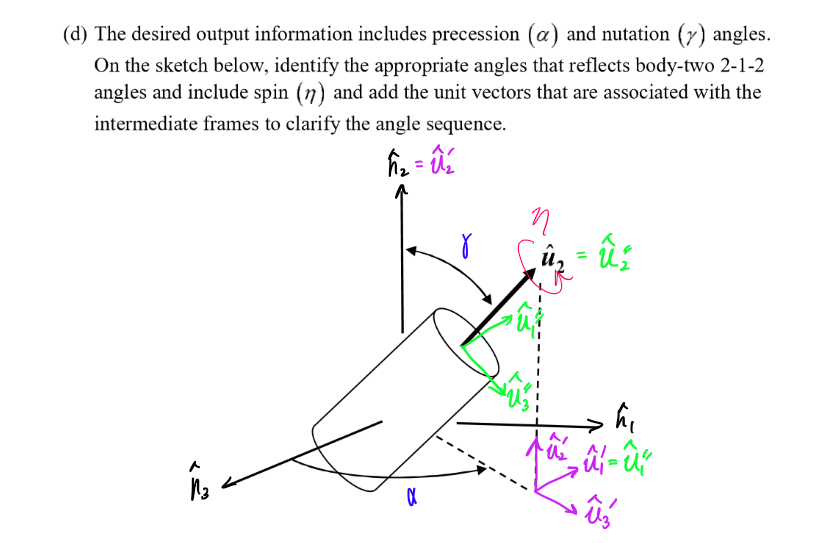




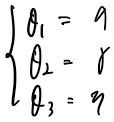


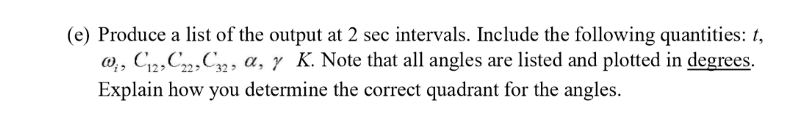


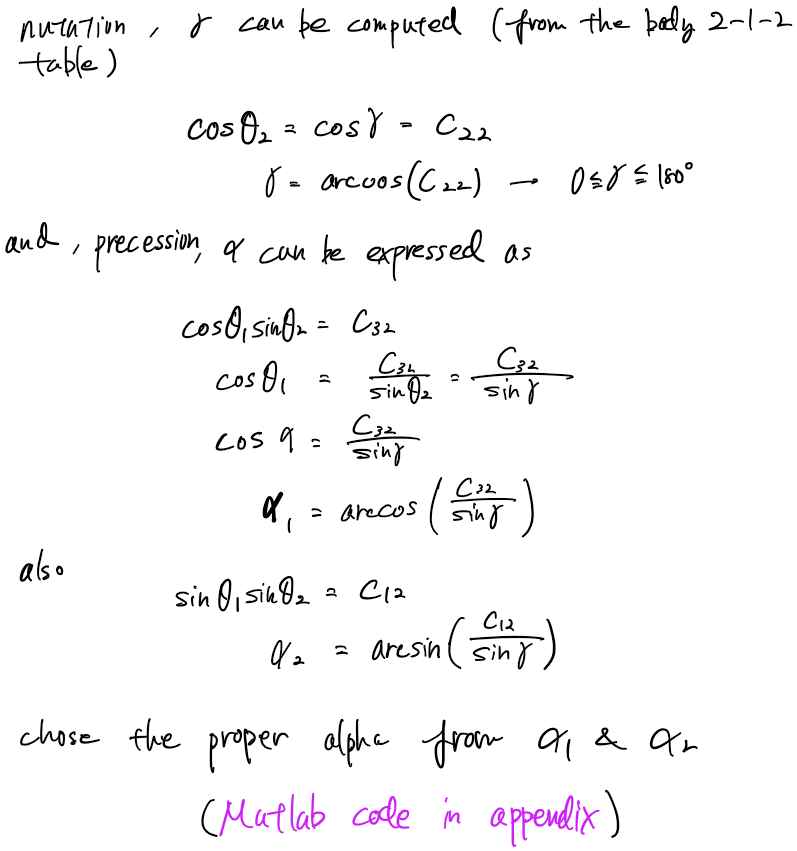


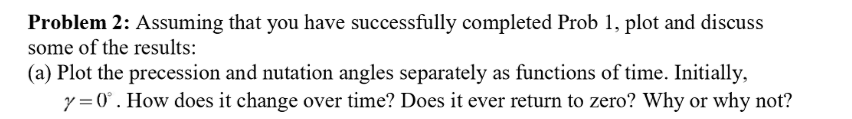


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |



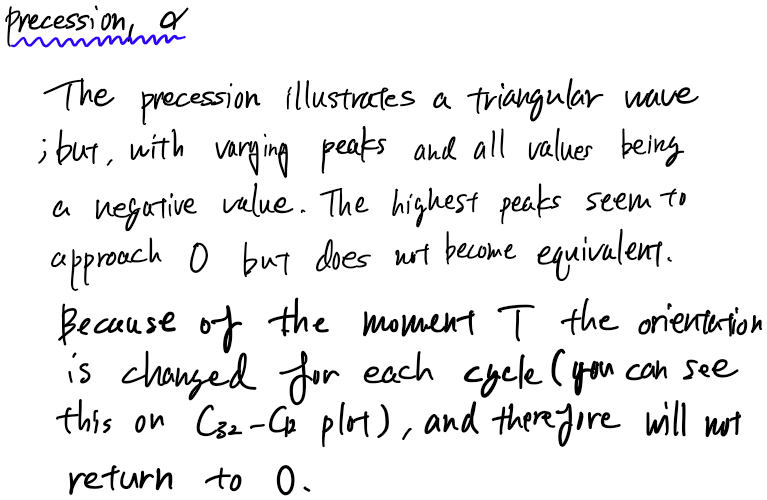






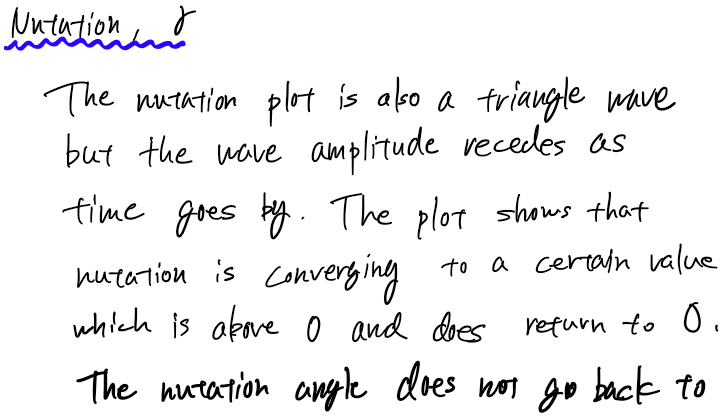
A screenshot of a cell phone

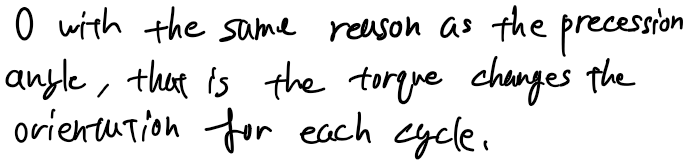
Description automatically generated

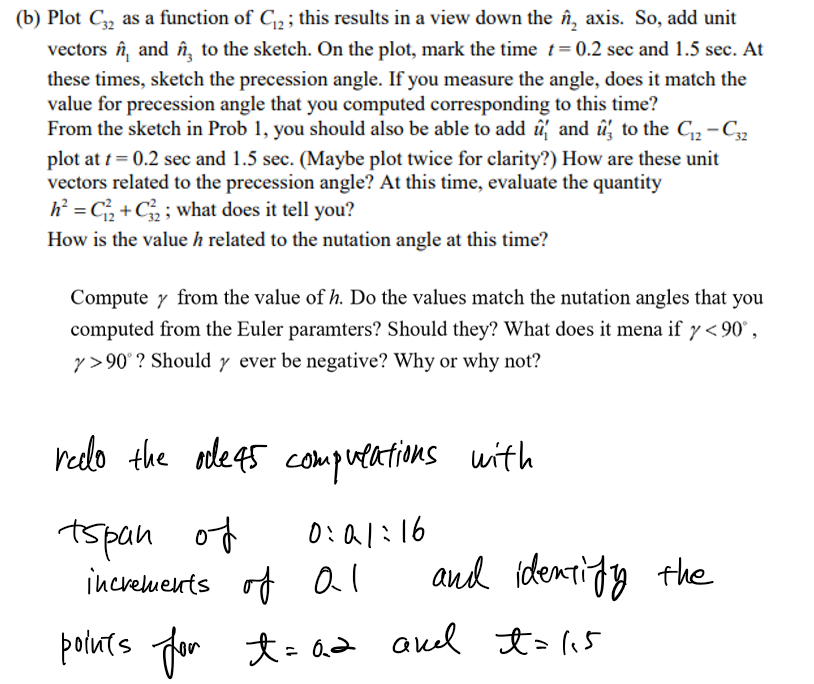


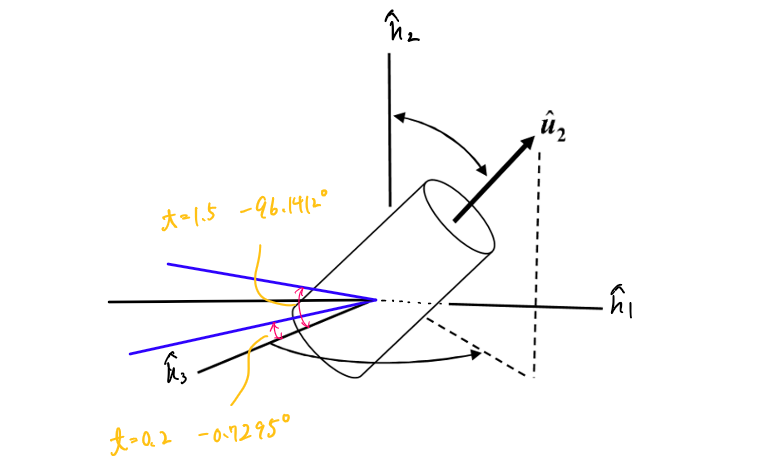
A close up of a logo

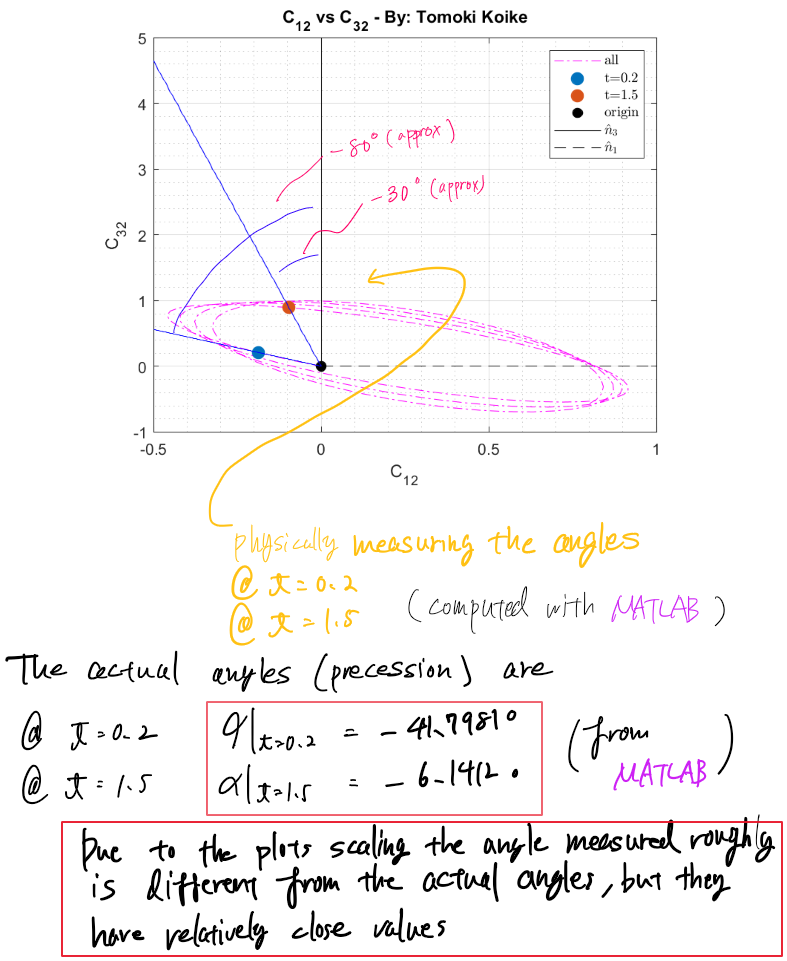
Description automatically generated

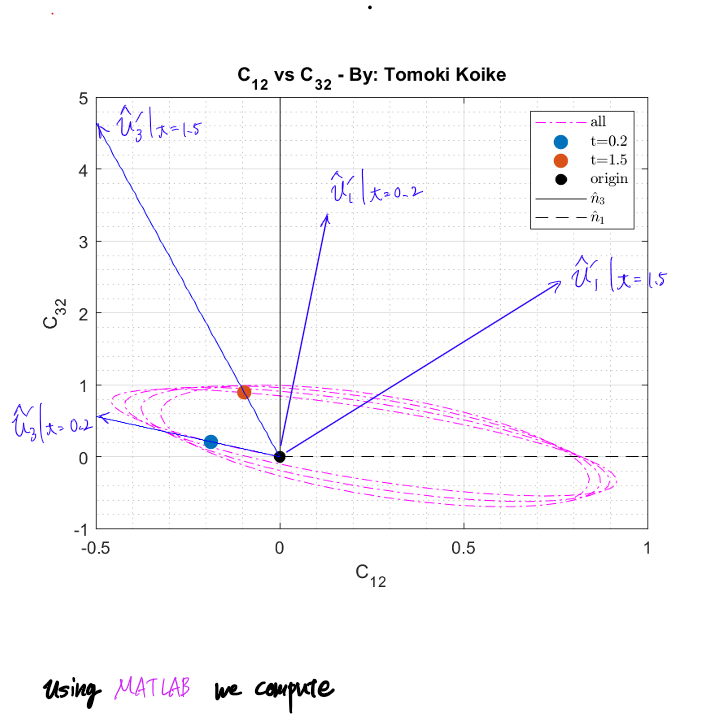


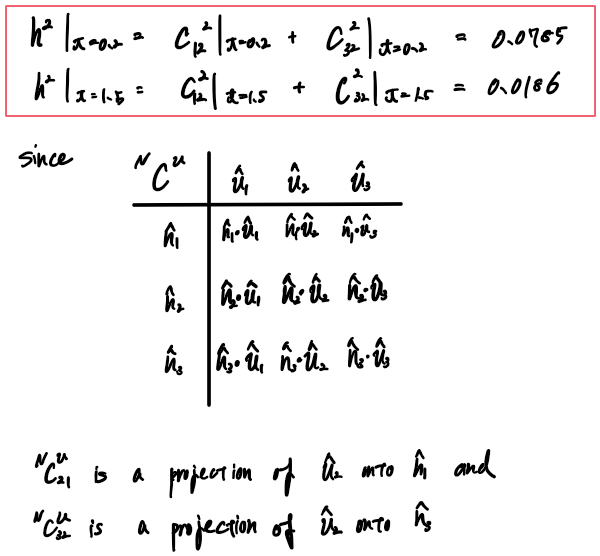


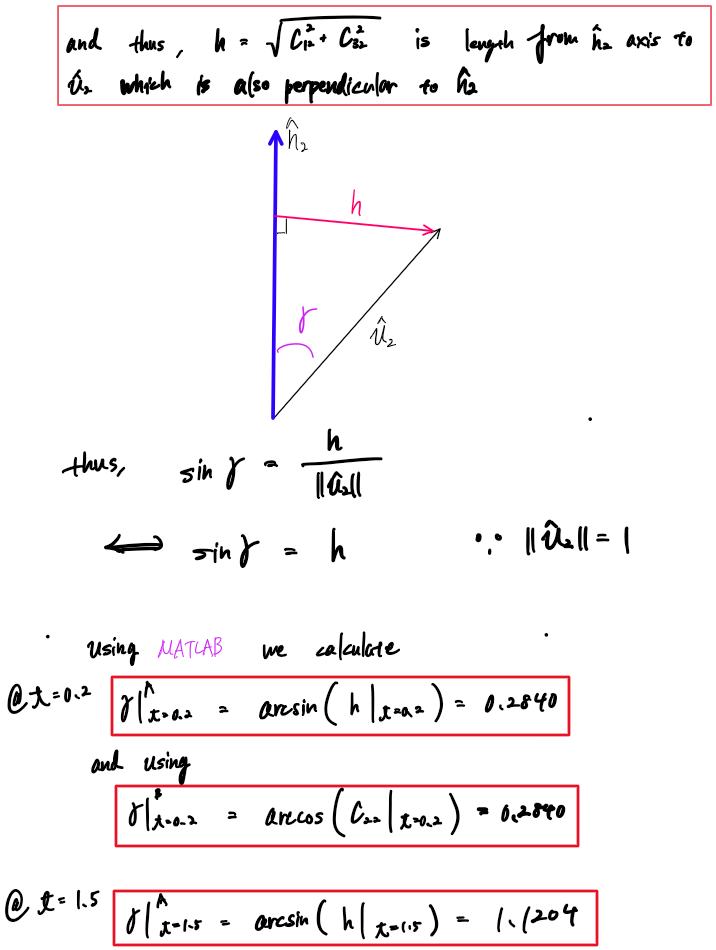


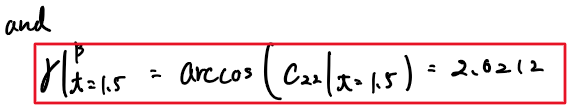


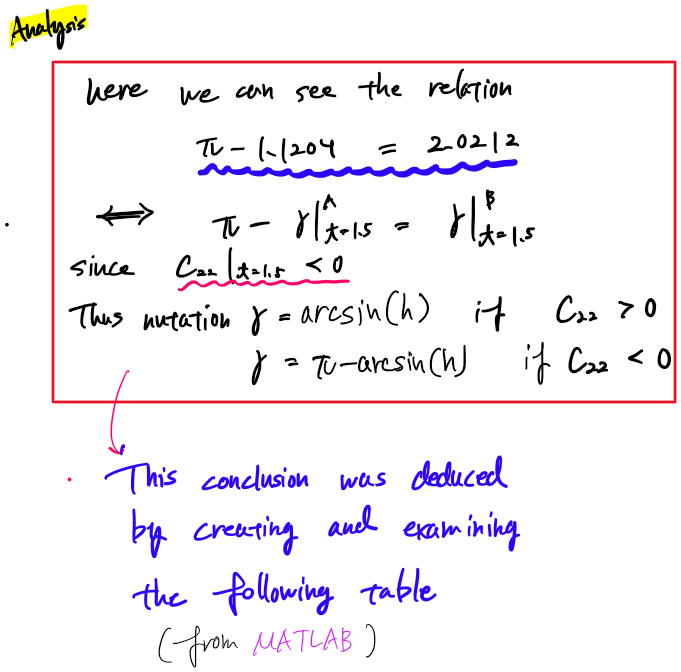


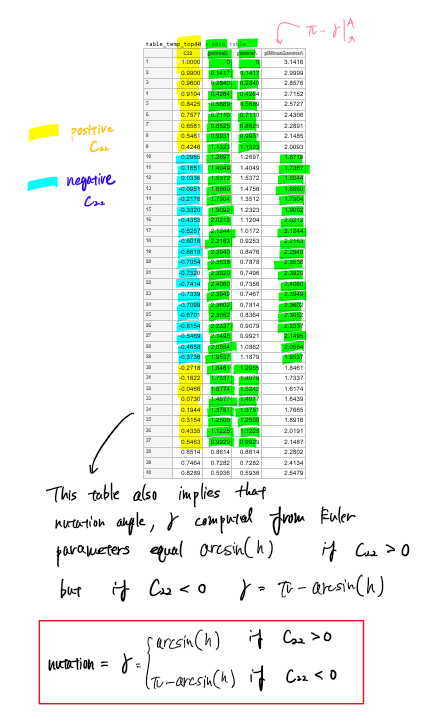


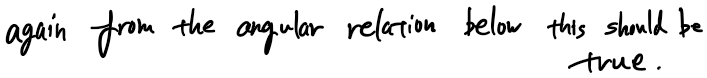


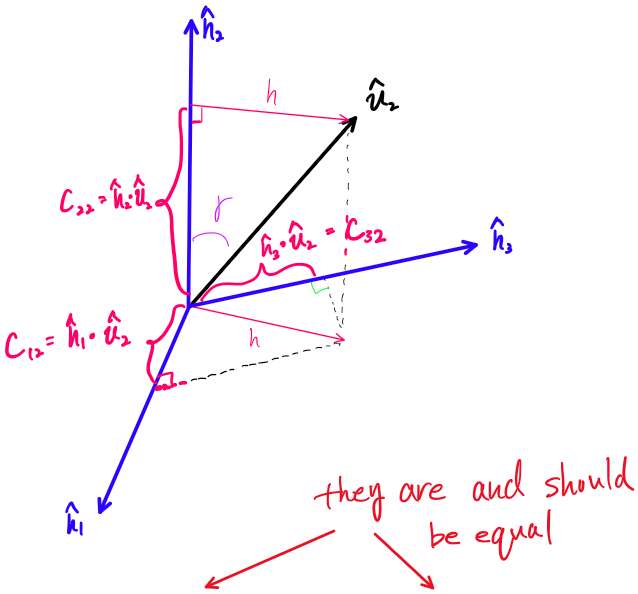


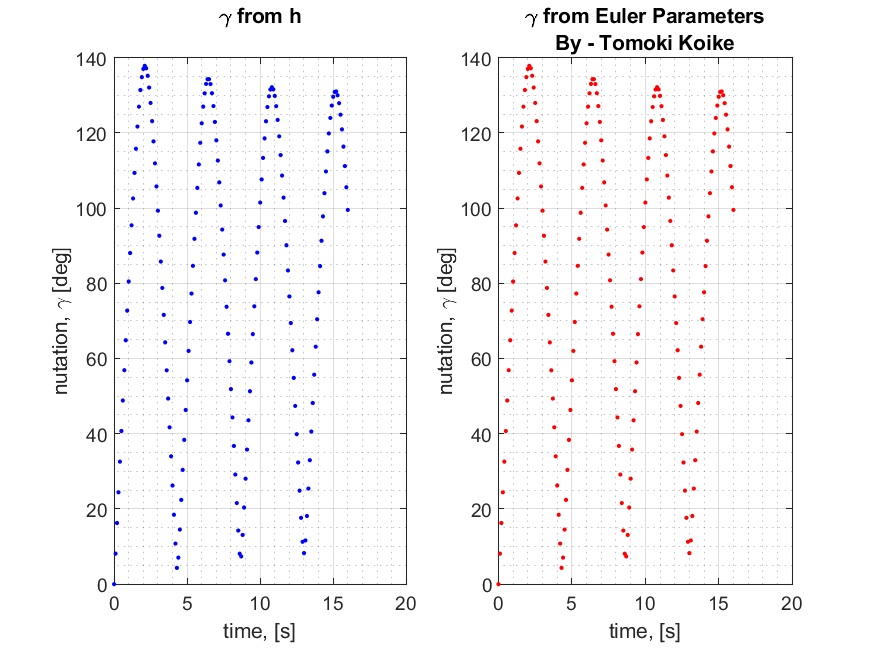


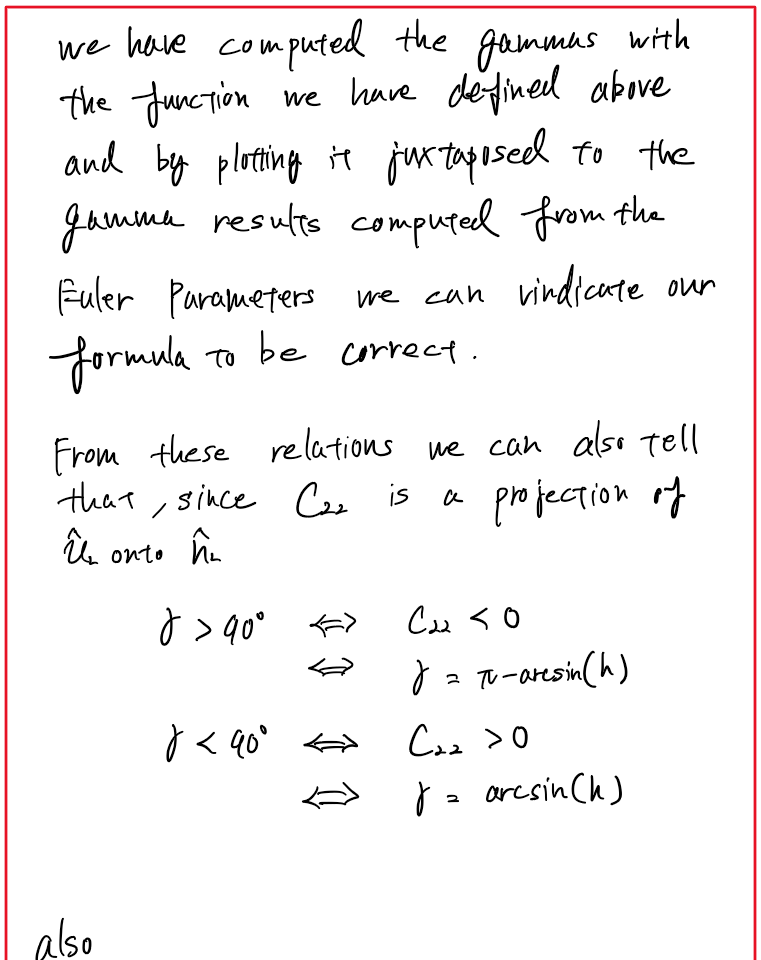
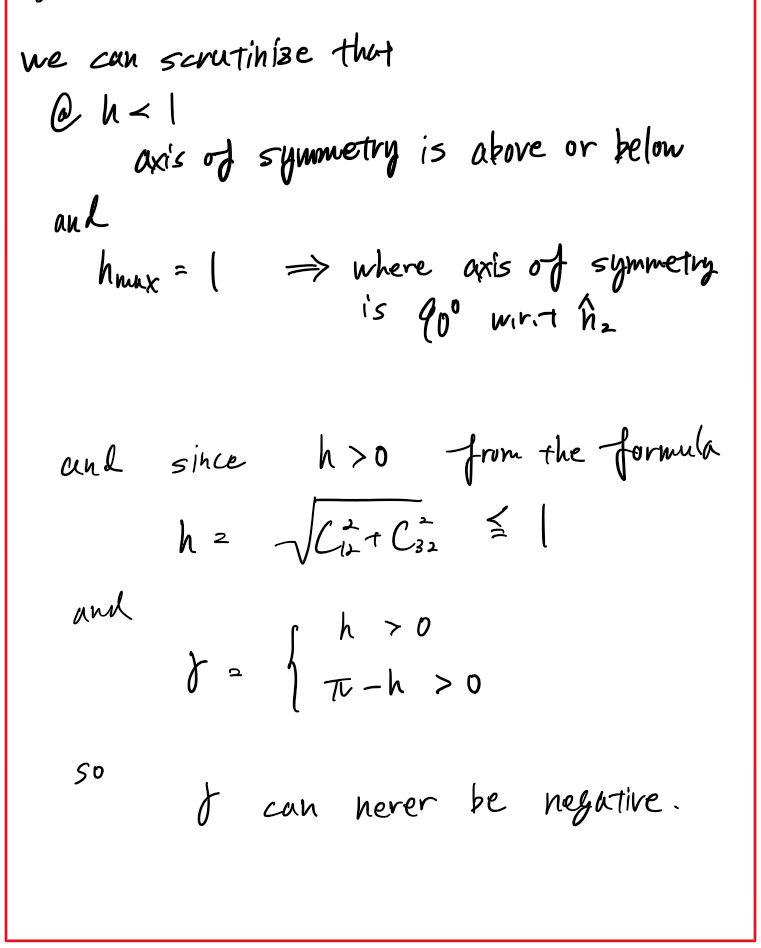


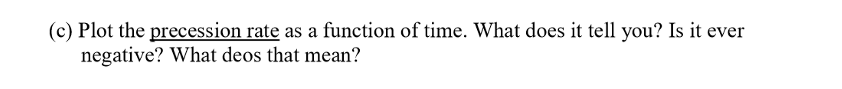


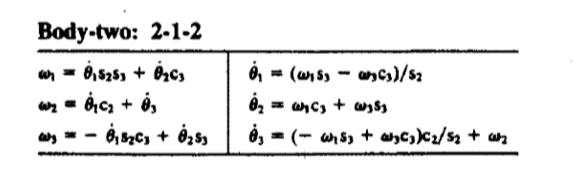


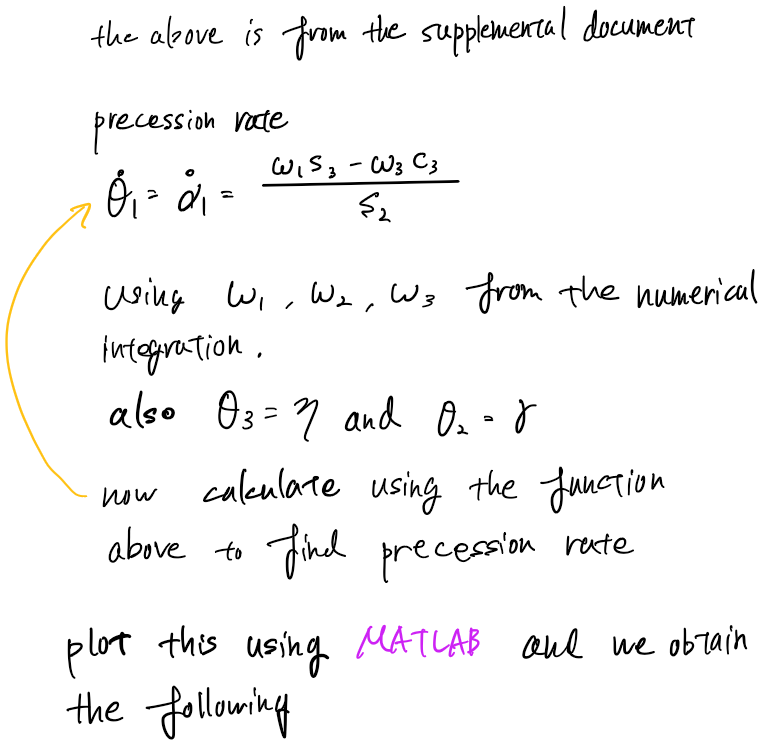


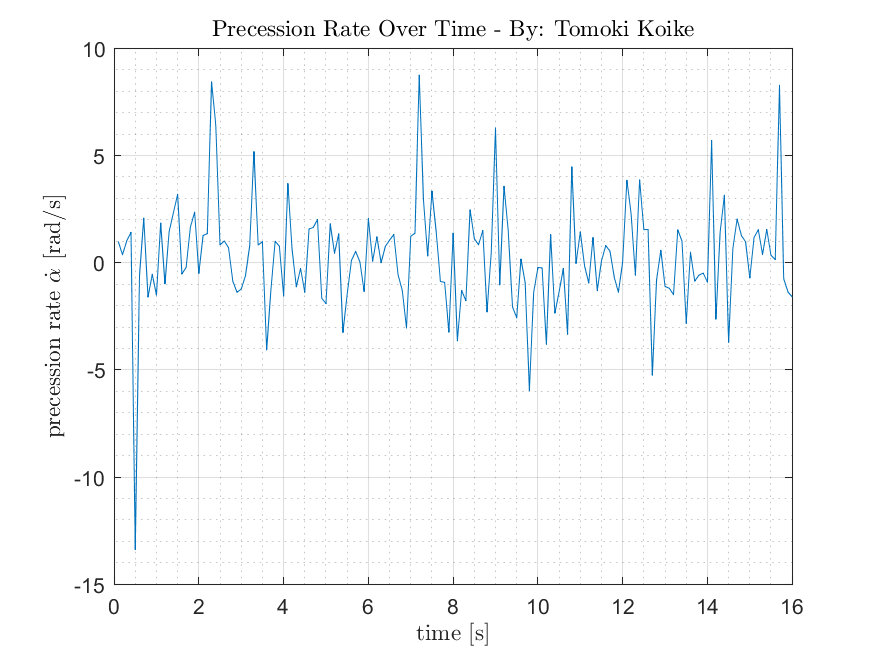


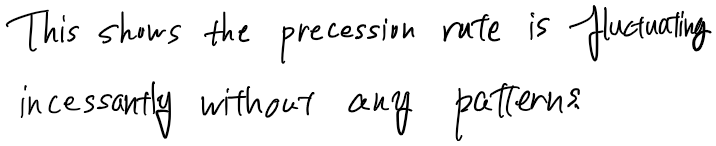
  


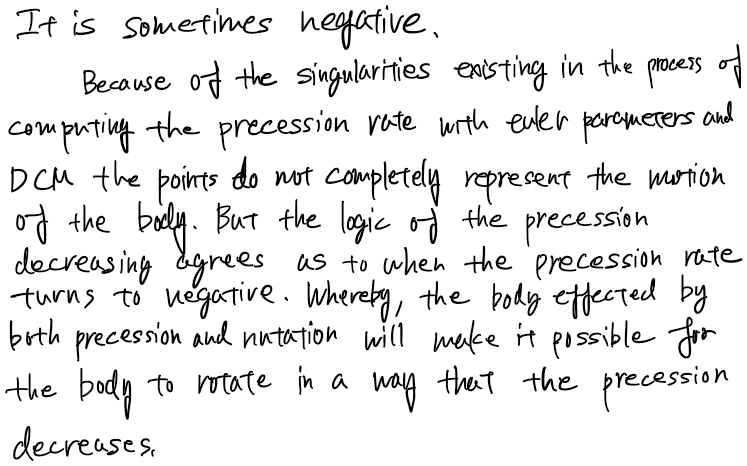


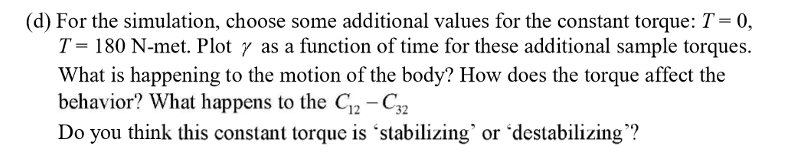


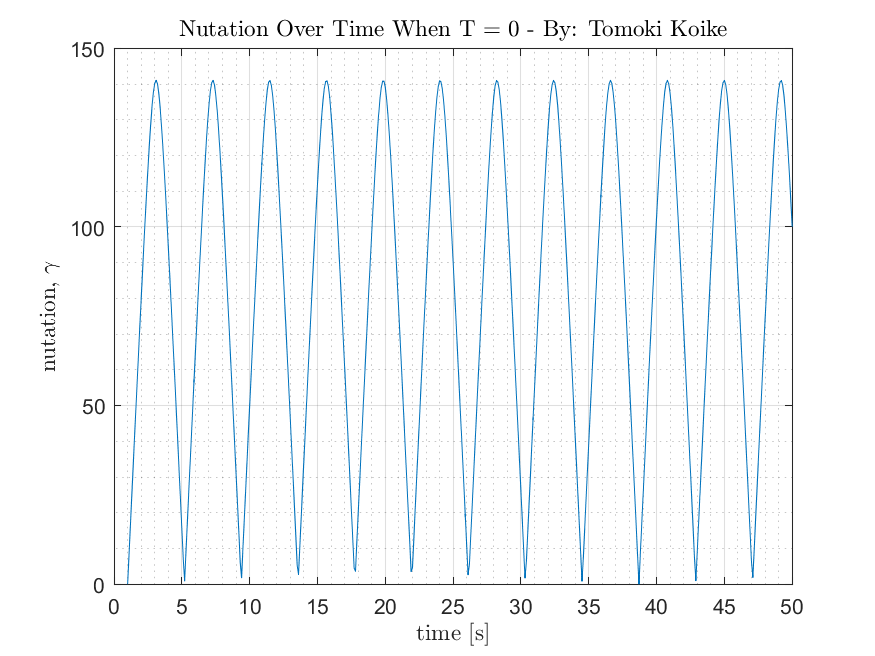


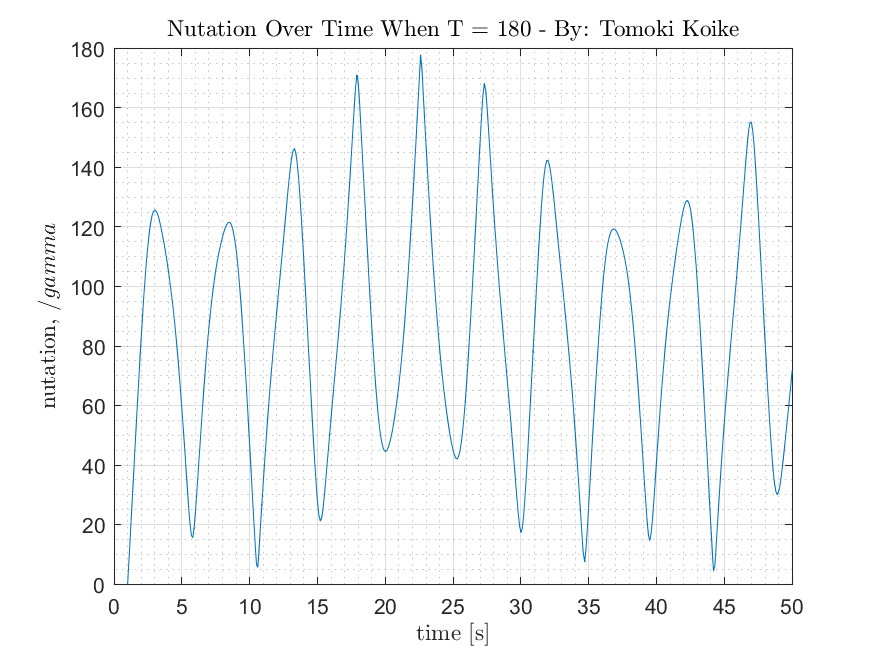


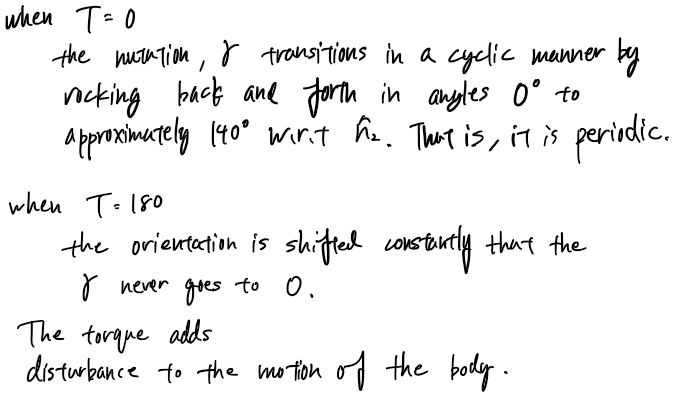


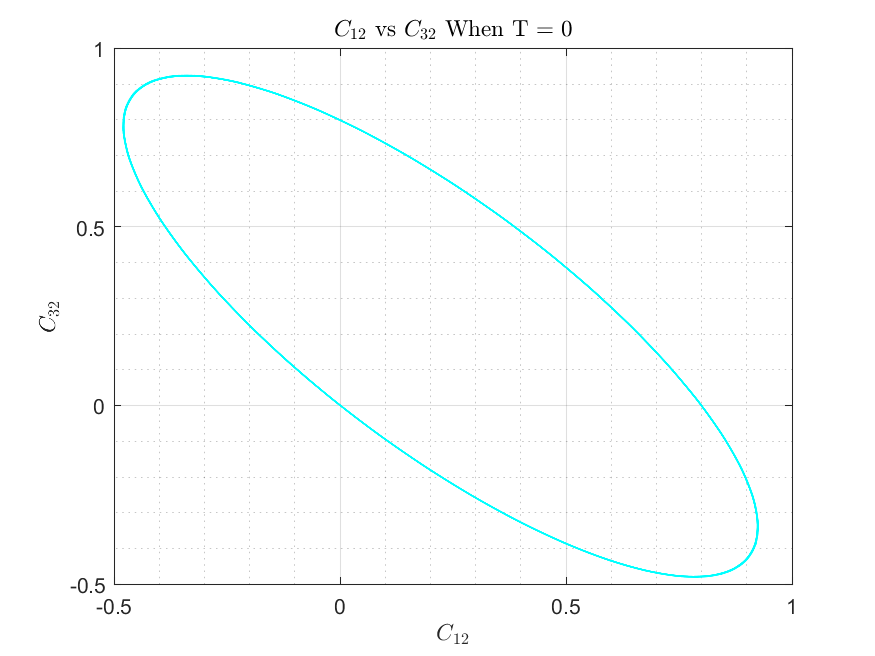


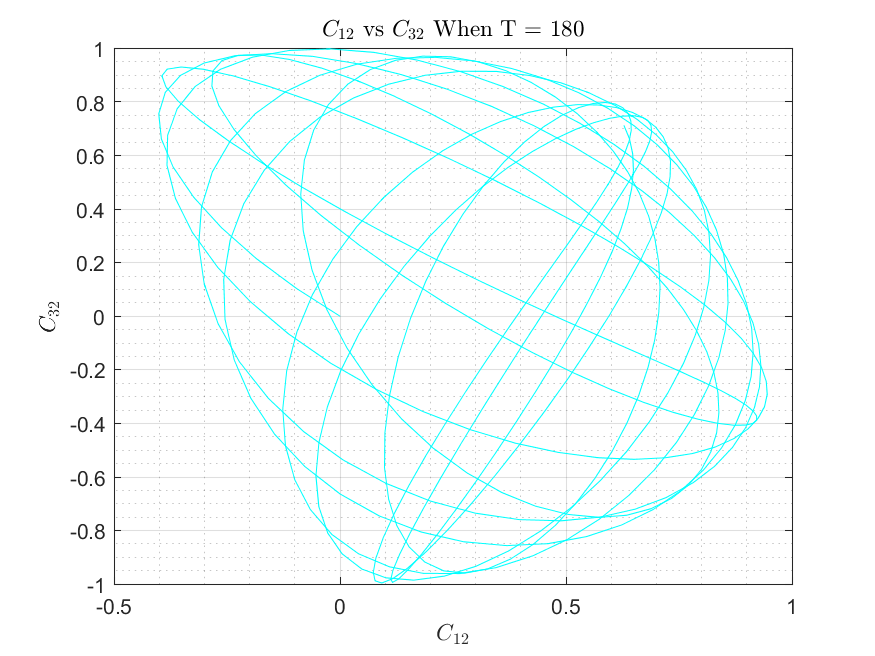




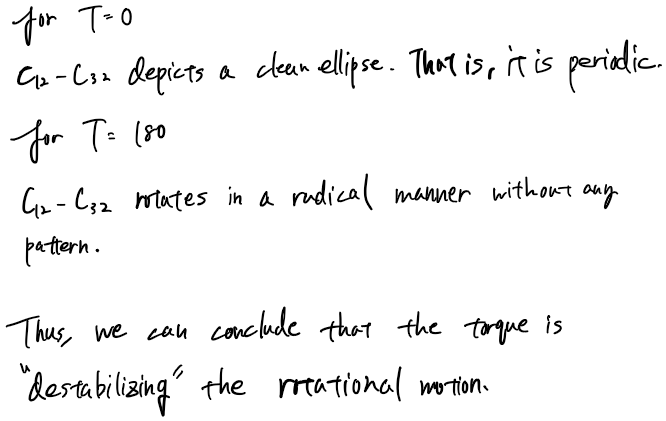


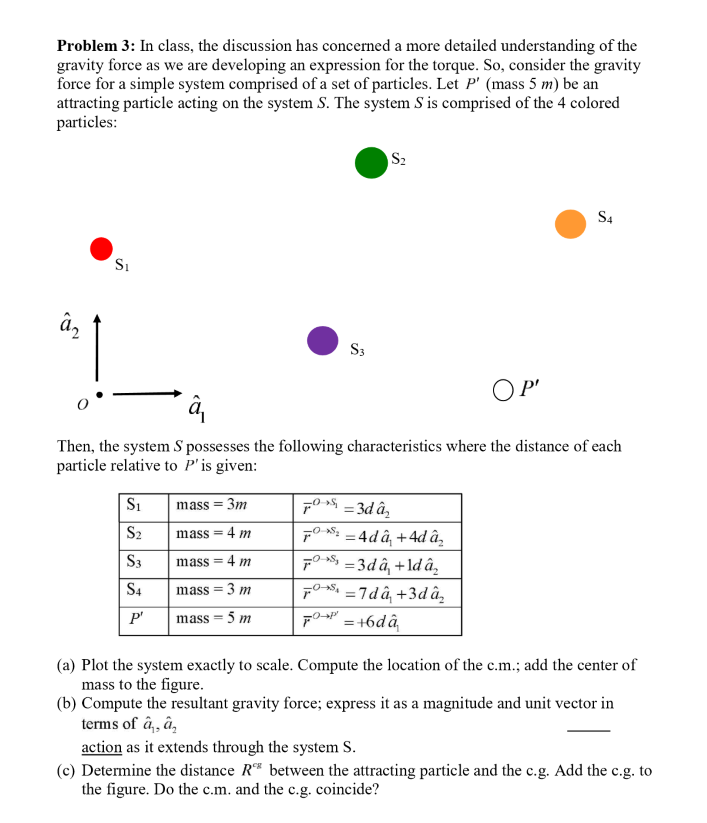


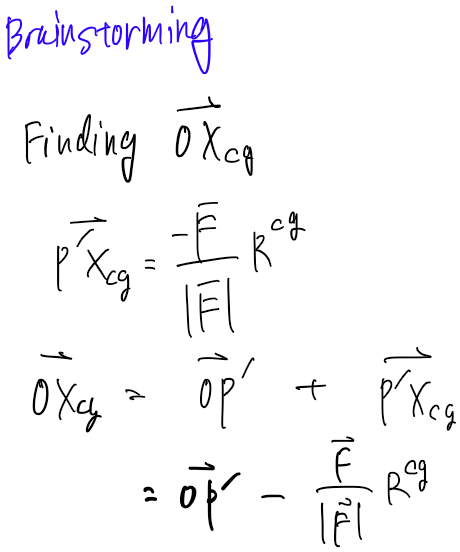


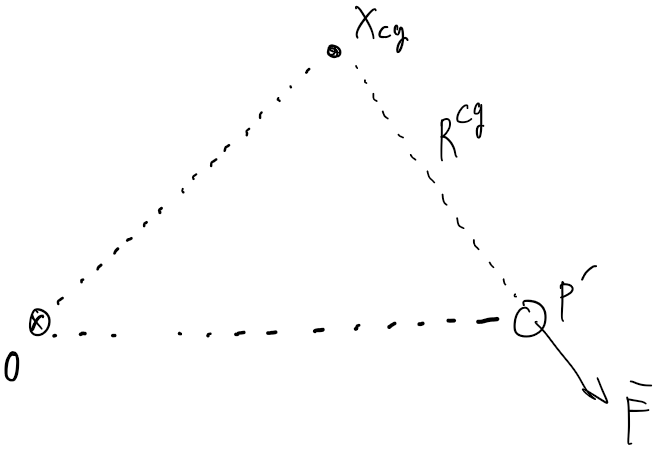


-



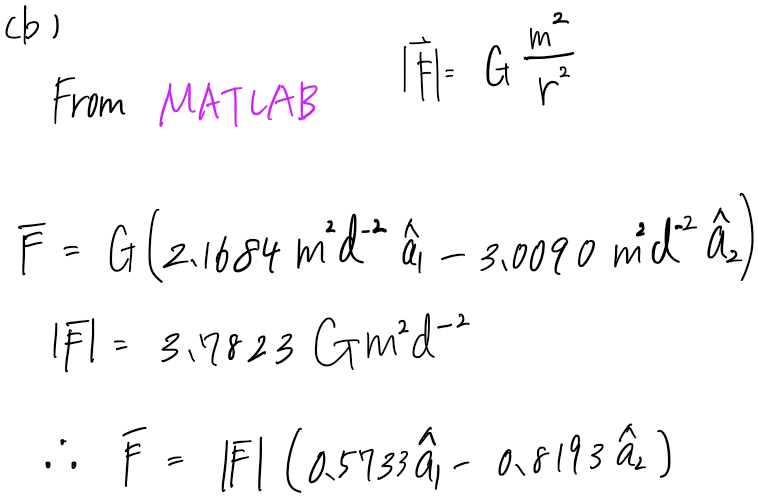


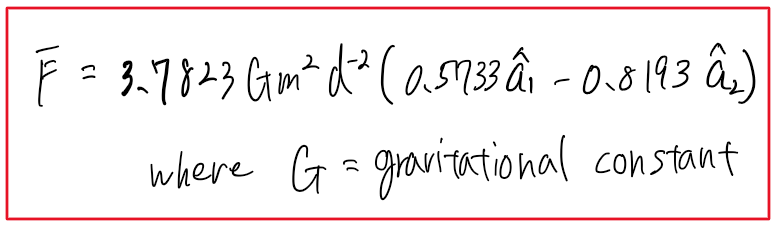


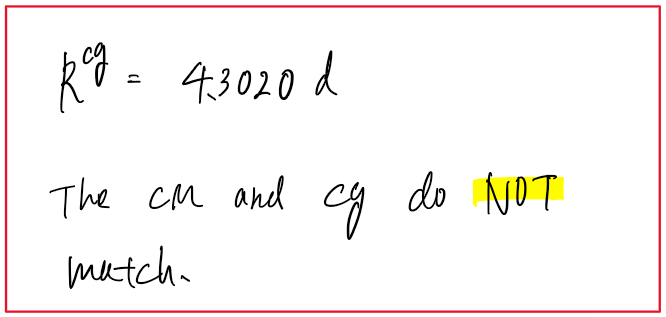


A close up of a person

Description automatically generated







Appendix

**AAE440 HW5 MATLAB CODE**

**Problem 1**

**<b>**

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Spring\AAE440\MATLAB\outputs\HW5';

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

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

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

% Constants

T = 40; % Torque [N-m]

I\_cm = [400 0 0; 0 100 0; 0 0 400]; % Inertia Dyadic [kg-m2]

I = 400;

J = 100;

% Initial Conditions

w0 = [1 2 1]; % Initial angular velocities [rad/s]

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

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

% Numerical integrations dynamic and kinematic EOMs

tspan = [0 16]; % Integration time

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

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

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

ws = res(:,1:3); % angular velocity dot

es = res(:,4:7); % Euler parameter dot

K\_minus\_K0 = res(:,8); % K-K0

C\_mats = res(:,9:end); % DCM values

% Plotting K-K0 against time

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

plot(t, K\_minus\_K0)

ylabel('$K-K0$', "Interpreter","latex")

xlabel('$time$ [s]', "Interpreter","latex")

title({'Constraint Constant Deviation or Numerical Error Over Time',['- By:' ...

' Tomoki Koike']},"Interpreter", "latex")

grid on

grid minor

box on

saveas(fig1, fullfile(fdir, 'constraint\_constant.png'));

**<c>**

% Plotting the angular velocity

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

plot(t, ws)

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

xlabel('time [s]')

title({'Angular Velcoity Dot Over Time - By: Tomoki Koike'})

ylim([-1.5, 2.5])

legend('$\omega\_1$', '$\omega\_2$', '$\omega\_3$')

grid on

grid minor

box on

saveas(fig2, fullfile(fdir, 'angular\_velocity.png'))

**<d>**

% Define a new time span with a 2 second increment

tspan2 = 0:0.2:50;

% conduct ode45 for differential equation

[t2, res2] = ode45(@(t,y) EOM(t,y,I,J,T), tspan2, y0, option);

% Assign C12, C22, and C32

C12s = res2(:,10);

C22s = res2(:,13);

C32s = res2(:,16);

% Calculating the precession and nutation using loop

alphas = zeros([length(t2),1]);

gammas = zeros([length(t2),1]);

for n = 1:length(t2)

% calculating and verfiying gamma

gammas(n) = acos(C22s(n));

if gammas(n) < 0 || gammas(n) > pi

gammas(n) = -gammas(n);

end

% calculating and verfying the alpha

alpha1 = asin(C12s(n)/sin(gammas(n)));

alpha2 = acos(C32s(n)/sin(gammas(n)));

if alpha1 == alpha2 || alpha1 == -alpha2

alphas(n) = alpha1;

elseif pi-alpha1 == alpha2 || -pi-alpha1 == alpha2

alphas(n) = alpha2;

else

alphas(n) = -alpha2;

end

gammas(n) = rad2deg(gammas(n));

alphas(n) = rad2deg(alphas(n));

end

% Define K

K = res2(:,8) + 1;

res\_array = [t2 res2(:,1:3) C12s C22s C32s alphas gammas K];

res\_table = array2table(res\_array, "VariableNames",{'time','omega1',['' ...

'omega2'],'omega3','C12','C22','C32','alpha', 'gamma','K'});

writetable(res\_table, fullfile(fdir, 'output\_table.xlsx'));

**Problem 2**

**<a>**

% Plotting the precession and nutation individually as a function of time

% Precession

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

plot(t2, alphas,'g')

title({'Precession Over Time - By: Tomoki Koike'})

xlabel('time [s]')

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

grid on

grid minor

box on

saveas(fig3, fullfile(fdir, 'precession\_vs\_time.png'));

% Nutation

fig4 = figure("Renderer","painters");

plot(t2, gammas,'r')

title({'Nutation Over Time - By: Tomoki Koike'})

xlabel('time [s]')

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

grid on

grid minor

box on

saveas(fig4, fullfile(fdir, 'nutation\_vs\_time.png'));

**<b>**

% Do the steps in the last half of problem 1 with smaller increments of

% time span

tspan3 = 0:0.1:16;

% conduct ode45 for differential equation

[t3, res3] = ode45(@(t,y) EOM(t,y,I,J,T), tspan3, y0, option);

C\_new = res3(:,9:17);

% Assign C12, C22, and C32

C12s\_new = res3(:,10);

C22s\_new = res3(:,13);

C32s\_new = res3(:,16);

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

idx\_t0p2 = find(t3==0.2);

idx\_t1p5 = find(t3==1.5);

C12\_t0p2 = C12s\_new(idx\_t0p2);

C22\_t0p2 = C22s\_new(idx\_t0p2);

C32\_t0p2 = C32s\_new(idx\_t0p2);

C12\_t1p5 = C12s\_new(idx\_t1p5);

C22\_t1p5 = C22s\_new(idx\_t1p5);

C32\_t1p5 = C32s\_new(idx\_t1p5);

% Assigning a temporary DCM with corresponding times

C\_temp = res3([idx\_t0p2 idx\_t1p5], 9:end);

% calculating and verfiying gamma

[alphas\_temp, gammas\_temp, etas\_temps] = ang\_calc\_body212(C\_temp);

% Plots with the specific times t = 0.2 and 1.5

fig5 = figure("Renderer","painters");

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

title('$C\_{12}$ vs $C\_{32}$ - By: Tomoki Koike')

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

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

hold on

plot(C12\_t0p2, C32\_t0p2, '.','MarkerSize',26)

plot(C12\_t1p5,C32\_t1p5,'.','MarkerSize',26)

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

plot([0 0],[0 5],'-k')

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

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

plot(d,d.\*(C32\_t0p2/C12\_t0p2),'-b')

plot(d,d.\*(C32\_t1p5/C12\_t1p5),'-b')

hold off

legend('all','t=0.2','t=1.5','origin','$\hat{n}\_3$','$\hat{n}\_1$')

grid on

grid minor

box on

saveas(fig5, fullfile(fdir, 'C12\_vs\_C32.png'));

% Calcluating h

% @ t = 0.2

h\_t0p2 = sqrt(C12\_t0p2^2 + C32\_t0p2^2);

gamma\_est\_t0p2 = asin(h\_t0p2);

% @ t = 1.5

h\_t1p5 = sqrt(C12\_t1p5^2 + C32\_t1p5^2);

gamma\_est\_t1p5 = asin(h\_t1p5);

% Analysis

array\_temp = [C22s\_new, acos(C22s\_new), asin(sqrt(C12s\_new.^2+C32s\_new.^2)),...

pi-asin(sqrt(C12s\_new.^2+C32s\_new.^2))];

table\_temp = array2table(array\_temp, "VariableNames",{'C22', 'gammaB', 'gammaA', 'piMinusGammaA'});

table\_temp\_top40 = table\_temp(1:40,:);

% Actually calculating gamma with h

gammas\_h = calc\_gamma\_with\_h(C12s\_new, C22s\_new, C32s\_new);

gammas\_h = rad2deg(gammas\_h);

% The actual gammas from the Euler parameters

[alpha\_eulers, gammas\_eulers, etas\_eulers] = ang\_calc\_body212(C\_new);

fig6 = figure("Renderer","painters");

subplot(1,2,1)

plot(t3, gammas\_h,'.b')

title({'$\gamma$ from h',''})

xlabel('time, [s]')

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

grid on

grid minor

box on

subplot(1,2,2)

plot(t3, gammas\_eulers, '.r')

title({'$\gamma$ from Euler Parameters','By - Tomoki Koike'})

xlabel('time, [s]')

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

grid on

grid minor

box on

saveas(fig6, fullfile(fdir, 'gammas\_with\_h\_and\_euler.png'))

**<c>**

% Do the steps in the last half of problem 1 with smaller increments of

% time span

tspan\_c = 0:0.1:16;

% conduct ode45 for differential equation

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

% Assign C12, C22, and C32

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

w1 = res\_c(:,1);

w2 = res\_c(:,2);

w3 = res\_c(:,3);

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

% Calculating precession rate

alpha\_dot = (w1.\*sin(eta\_c) - w3.\*cos(eta\_c)) ./ sin(gamma\_c);

% Plotting

fig7 = figure('Renderer',"painters");

plot(t\_c, alpha\_dot,'-b')

title('Precession Rate Over Time - By: Tomoki Koike',"Interpreter","latex")

xlabel('time [s]',"Interpreter","latex")

ylabel('precession rate $\dot{\alpha}$ [rad/s]',"Interpreter","latex")

grid on

grid minor

box on

saveas(fig7, fullfile(fdir,'precession\_rate.png'));

**<d>**

% Constants

tspan\_d = 1:0.1:50;

T\_d1 = 0; % Torque [N-m]

[t\_d1, res\_d1] = ode45(@(t,y) EOM(t,y,I,J,T\_d1), tspan\_d, y0, option);

C\_d1 = res\_d1(:,9:end);

[alpha\_d1, gamma\_d1, eta\_d1] = ang\_calc\_body212(C\_d1);

T\_d2 = 180;

[t\_d2, res\_d2] = ode45(@(t,y) EOM(t,y,I,J,T\_d2), tspan\_d, y0, option);

C\_d2 = res\_d2(:,9:end);

[alpha\_d2, gamma\_d2, eta\_d2] = ang\_calc\_body212(C\_d2);

% Plotting

fig8 = figure("Renderer","painters");

plot(t\_d1, gamma\_d1)

xlabel('time [s]', "Interpreter","latex")

ylabel('nutation, $\gamma$', "Interpreter","latex")

title('Nutation Over Time When T = 0 - By: Tomoki Koike',"Interpreter","latex")

grid on

grid minor

box on

saveas(fig8, fullfile(fdir,'gamma\_t\_equal\_0.png'));

fig9 = figure("Renderer","painters");

plot(t\_d2, gamma\_d2)

xlabel('time [s]',"Interpreter","latex")

ylabel('nutation, $/gamma$',"Interpreter","latex")

title('Nutation Over Time When T = 180 - By: Tomoki Koike',"Interpreter","latex")

grid on

grid minor

box on

saveas(fig9, fullfile(fdir,'gamma\_t\_equal\_180.png'));

fig10 = figure("Renderer","painters");

plot(C\_d1(:,2), C\_d1(:,8), '-c')

title('$C\_{12}$ vs $C\_{32}$ When T = 0',"Interpreter","latex")

xlabel('$C\_{12}$',"Interpreter","latex")

ylabel('$C\_{32}$',"Interpreter","latex")

grid on

grid minor

box on

saveas(fig10, fullfile(fdir,'C12\_C32\_T0.png'))

fig11 = figure('Renderer',"painters");

plot(C\_d2(:,2), C\_d2(:,8), '-c')

title('$C\_{12}$ vs $C\_{32}$ When T = 180',"Interpreter","latex")

xlabel('$C\_{12}$',"Interpreter","latex")

ylabel('$C\_{32}$',"Interpreter","latex")

grid on

grid minor

box on

saveas(fig11, fullfile(fdir,'C12\_C32\_T180.png'))

**Functions**

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

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

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

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

for n = 1:length(alphas)

% calculating and verfiying gamma

gammas(n) = acos(C22s(n));

if gammas(n) < 0 || gammas(n) > pi

gammas(n) = -gammas(n);

end

% calculating and verfying the alpha

alpha1 = asin(C12s(n)/sin(gammas(n)));

alpha2 = acos(C32s(n)/sin(gammas(n)));

if alpha1 == alpha2 || alpha1 == -alpha2

alphas(n) = alpha1;

elseif pi-alpha1 == alpha2 || -pi-alpha1 == alpha2

alphas(n) = alpha2;

else

alphas(n) = -alpha2;

end

eta1 = asin(C21s(n)/sin(gammas(n)));

eta2 = acos(-C23s(n)/sin(gammas(n)));

if eta1 == eta2 || eta1 == -eta2

etas(n) = eta1;

elseif pi-eta1 == eta2 || -pi-eta1 == eta2

etas(n) = eta2;

else

etas(n) = -eta2;

end

gammas(n) = rad2deg(gammas(n));

alphas(n) = rad2deg(alphas(n));

etas(n) = rad2deg(etas(n));

end

end

function ang = eval\_cos(theta)

if theta < 0 || theta > pi

ang = -theta;

else

ang = theta;

end

end

function ang = eval\_sin(theta)

if -pi/2 <= theta && theta <= pi/2

ang = theta;

else

ang = -theta;

end

end

function gamma = calc\_gamma\_with\_h(C12, C22, C32)

h = asin(sqrt(C12.^2 + C32.^2));

gamma = zeros([length(h), 1]);

for x = 1:length(h)

if C22(x) > 0

gamma(x) = h(x);

elseif C22(x) < 0

gamma(x) = pi - h(x);

end

end

end

## **Problem 3**

clear all; close all; clc;

fdir = 'C:\Users\Tomo\Desktop\studies\2020-Spring\AAE440\MATLAB\outputs\HW5';

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)

% Plotting the system

% Postion vectors for each S(i) and P

d = 1;

origin = [0; 0];

S1 = [d\*0 d\*3];

S2 = [d\*4 d\*4];

S3 = [d\*3 d\*1];

S4 = [d\*7 d\*3];

P = [d\*6 d\*0];

S\_x = [S1(1); S2(1); S3(1); S4(1)]; % all x positions

S\_y = [S1(2); S2(2); S3(2); S4(2)]; % all x positions

name\_str = ["$0$","$\hat{a}\_1$","$\hat{a}\_2$","$S\_1$","$S\_2$",...

"$S\_3$","$S\_4$", "$P^\prime$"];

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

hold on; grid on; box on; axis equal;

ylim([-1, 5]); xlim([-1, 8]);

plot(origin(1),origin(2), '.','MarkerSize', 20,'Color',[0 0 0])

plot(S1(1),S1(2), '.','MarkerSize', 60,'Color',[0.9 0 0]);

plot(S2(1),S2(2), '.','MarkerSize', 80,'Color',[ 0 0.4 0]);

plot(S3(1),S3(2), '.','MarkerSize', 80,'Color',[0.5 0.1 1]);

plot(S4(1),S4(2), '.','MarkerSize', 60,'Color',[ 1 0.6 0]);

plot(P(1),P(2), 'ko','MarkerSize',20);

text(-0.3, -0.2, name\_str(1),"Interpreter","latex")

text(0.2, 2.6, name\_str(4),"Interpreter","latex")

text(4.4, 4.1, name\_str(5),"Interpreter","latex")

text(3.5, 0.8, name\_str(6),"Interpreter","latex")

text(7.4, 3.1, name\_str(7),"Interpreter","latex")

text(6.3, 0, name\_str(8), "FontSize",15, "Interpreter","latex")

% a1\_hat axis

x1 = [0.2 1];

y1 = [0 0];

drawArrow(x1,y1,'k', 'linewidth',2); text(1.1,-0.2, name\_str(2),"Interpreter","latex");

% a2\_hat axis

x2 = [0 0];

y2 = [0.2 1];

drawArrow(x2,y2,'k', 'linewidth',2); text(-0.2,1.2, name\_str(3),"Interpreter","latex");

% Masses for each S(i) and P

m1 = 3;

m2 = 4;

m3 = 4;

m4 = 3;

mP = 5;

m\_S = [m1 m2 m3 m4];

m\_tot = sum(m\_S);

% Computing the CM of the system

x\_cm = dot(S\_x,m\_S)/m\_tot;

y\_cm = dot(S\_y,m\_S)/m\_tot;

% Plotting the CM

plot(x\_cm, y\_cm, '.k','MarkerSize', 32); text(2.8, 2.5, '$CM$',"Interpreter","latex");

% Re-defining positions of S(i) in terms of P' (attracting body)

S1\_P = S1 - P;

S2\_P = S2 - P;

S3\_P = S3 - P;

S4\_P = S4 - P;

S\_P\_all = [S1\_P; S2\_P; S3\_P; S4\_P];

% Computing the Line of action

% \*\*Non-dimensionalized so disregard gravitational constant G

N = length(S\_P\_all(:,1));

dim = length(S\_P\_all(1,:));

F\_i = zeros([N, dim]);

for i = 1:N

F\_i(i,:) = -mP.\*m\_S(i).\*S\_P\_all(i,:).\*norm(S\_P\_all(i,:)).^-3;

end

F = sum(F\_i);

F\_mag = norm(F);

F\_unit = F/F\_mag;

% Computing the CG

% \*\*Non-dimensionalized so disregard gravitational constant G

R\_cg = sqrt(mP\*(m\_tot)/norm(F));

% R\_cg vector is essentially G'-P' (from CG to P') vector

% To make this into a postion vector wrt the origin we do the following vector manipulation

P\_Xcg = -F\*R\_cg/norm(F);

O\_Xcg = P + P\_Xcg;

% Plotting the Line of action

x3 = [6 O\_Xcg(1)];

y3 = [0 O\_Xcg(2)];

drawArrow(x3,y3, 'linewidth',2,'Color',[0 0 1]);

text(5, 2, '$Line$ $of$ $action$','Color','b',"Interpreter","latex");

% Plotting the CG

plot(O\_Xcg(1), O\_Xcg(2), 'k.','MarkerSize', 32);

text(3, 3.3, '$CG$', "Interpreter","latex");

title('Problem 3 Simulation Final Answer - By: Tomoki Koike')

saveas(fig1, fullfile(fdir,'hw5\_p3\_gravity\_system.png'));

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