AAE440: Space Attitude Dynamics

HW4: Kinematical Differentials & Orientations

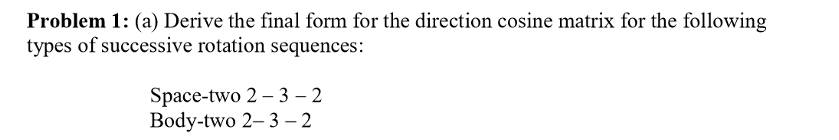
Dr. Howell

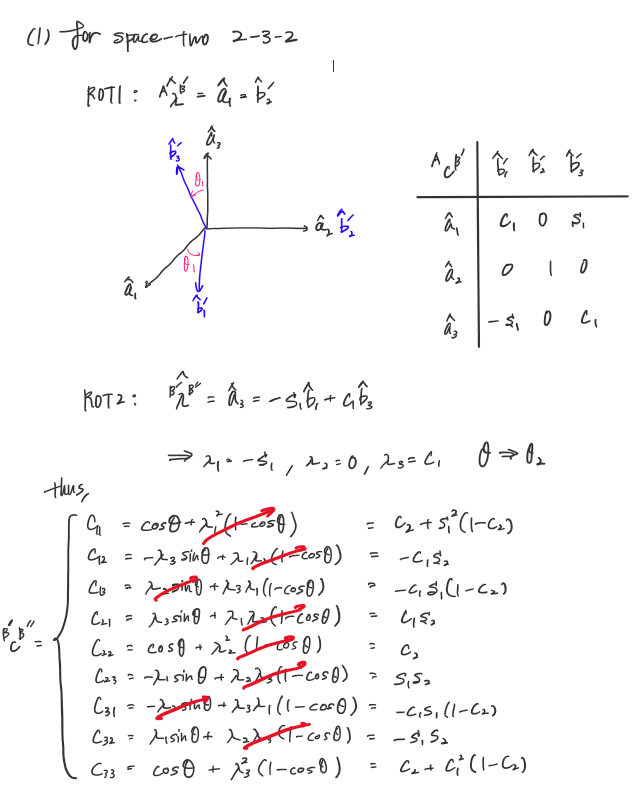
Tomoki Koike

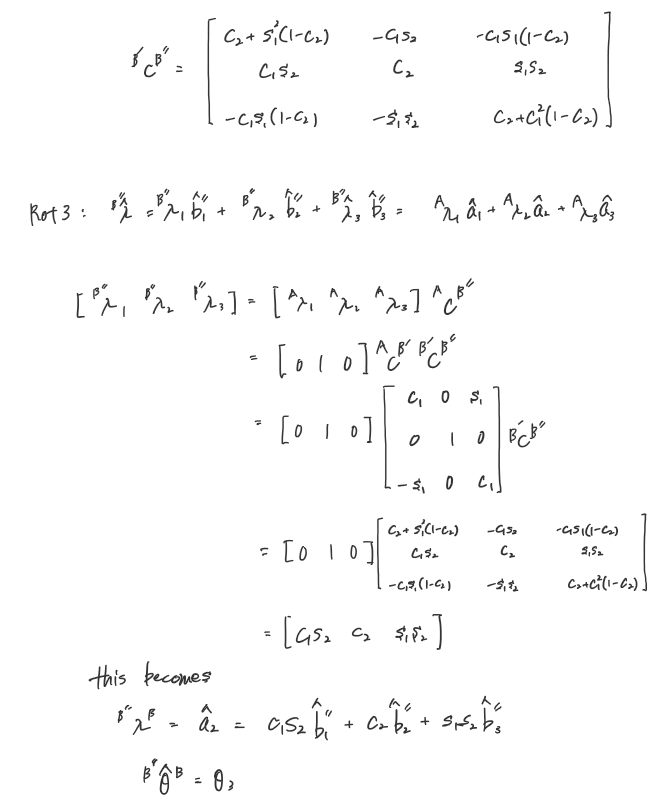
Friday February 14, 2020

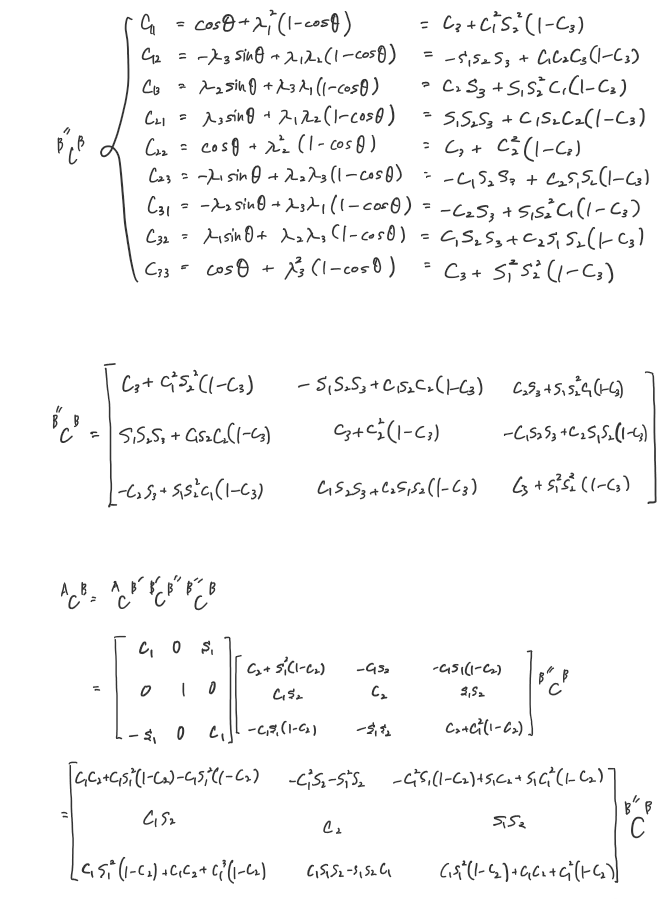
A close up of a tower

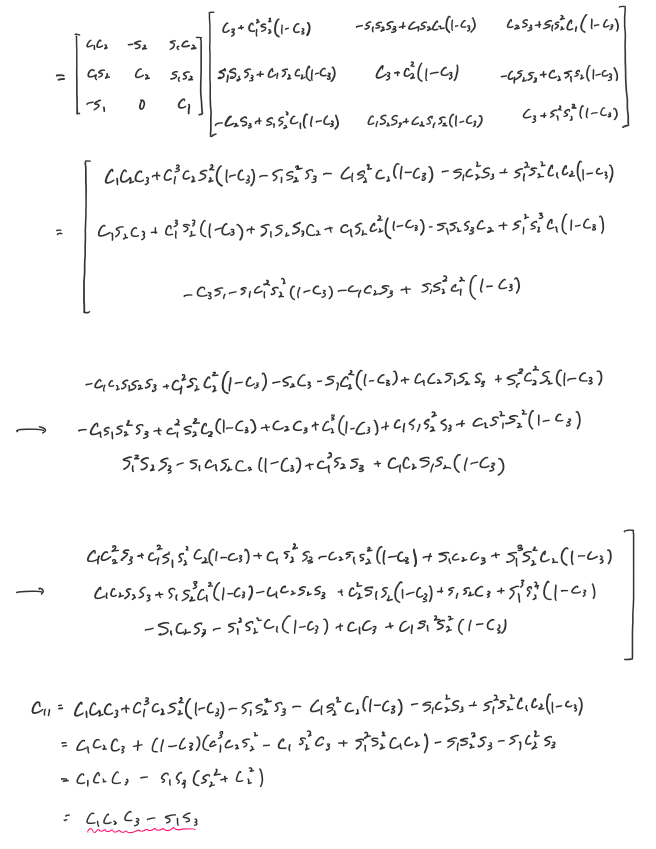
Description automatically generated

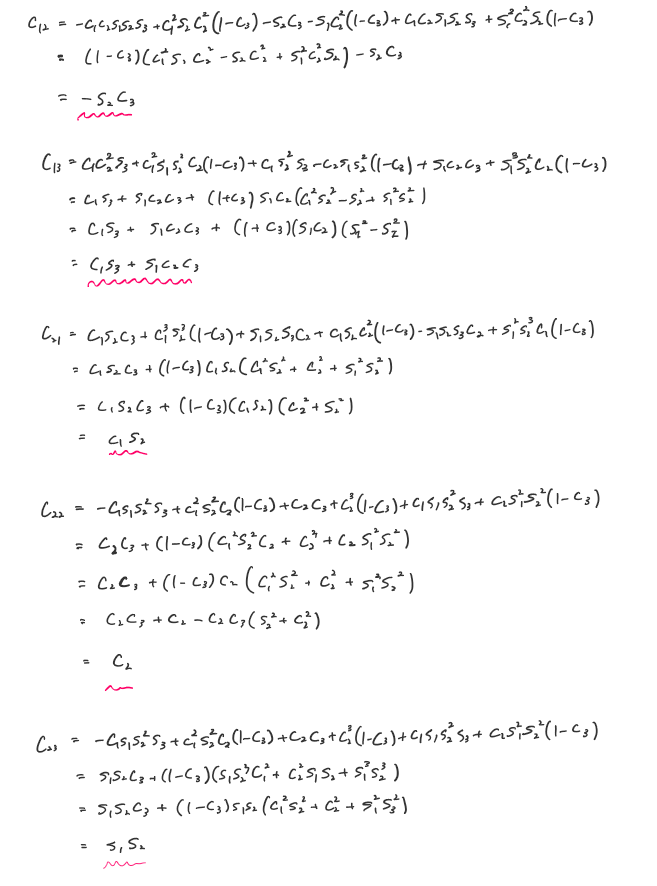


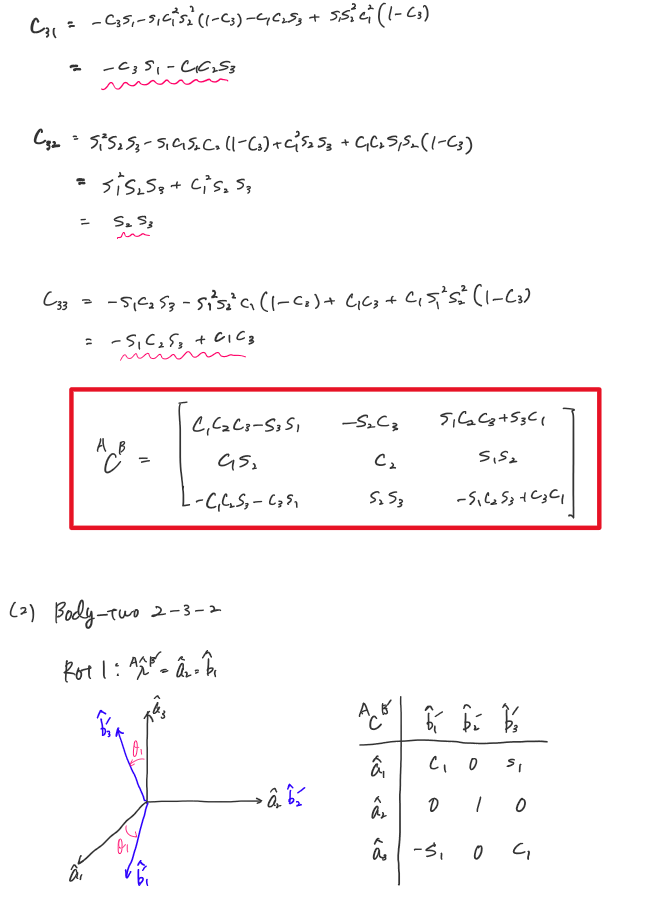


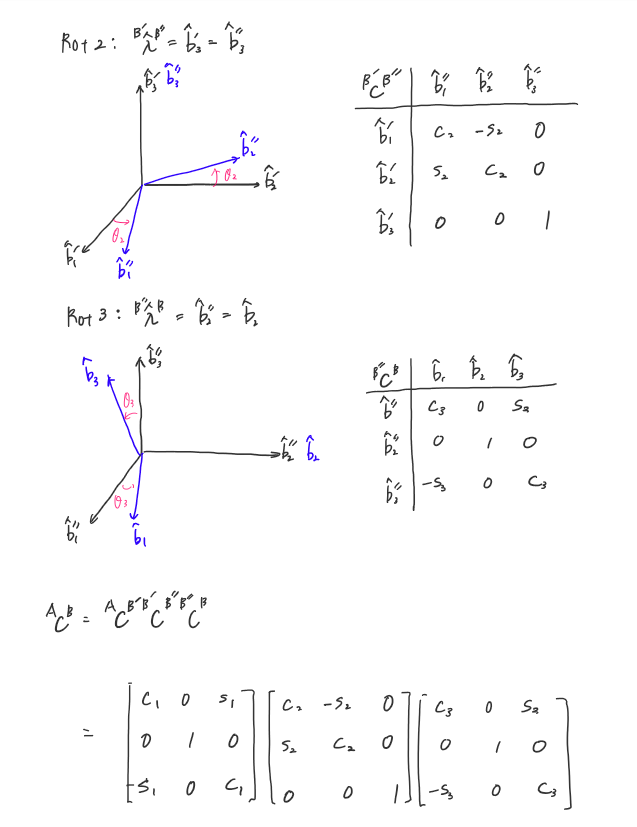


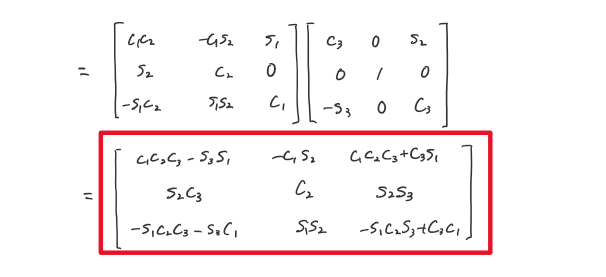


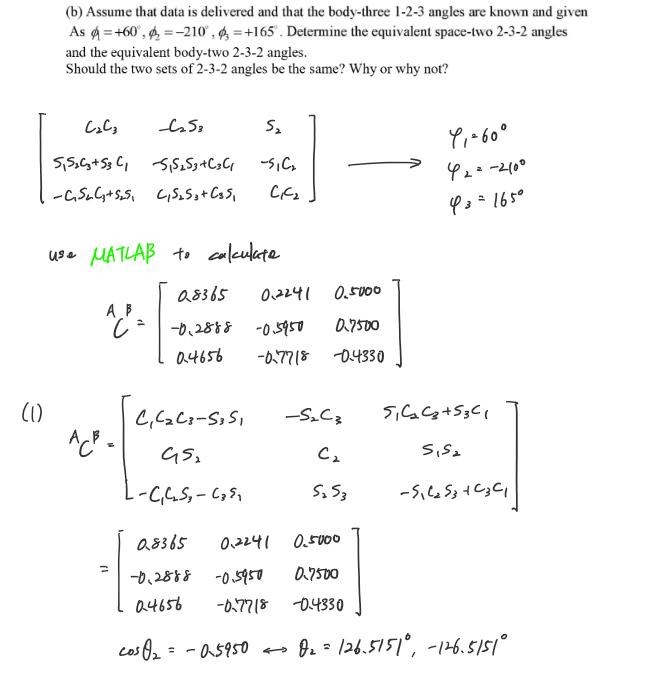


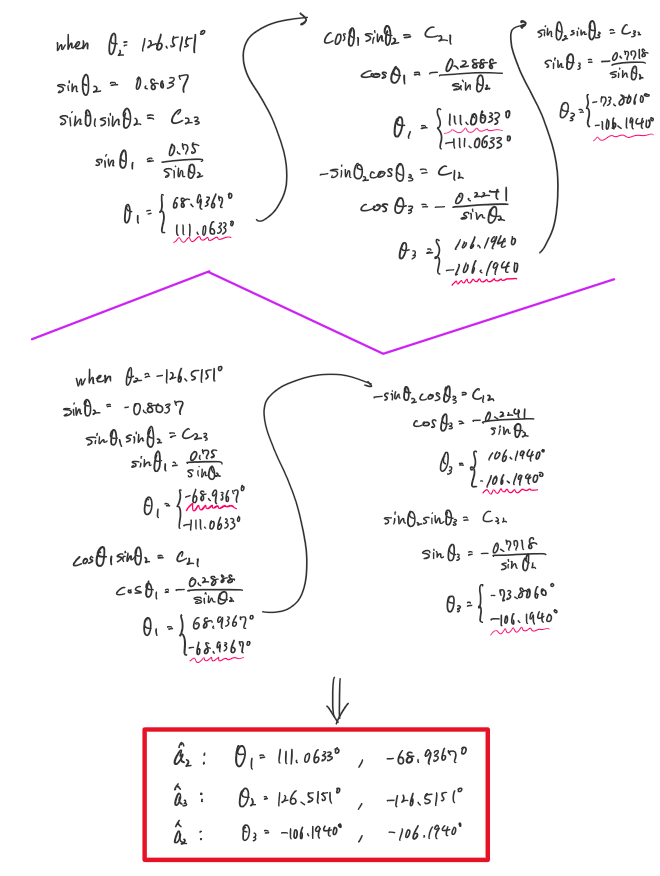


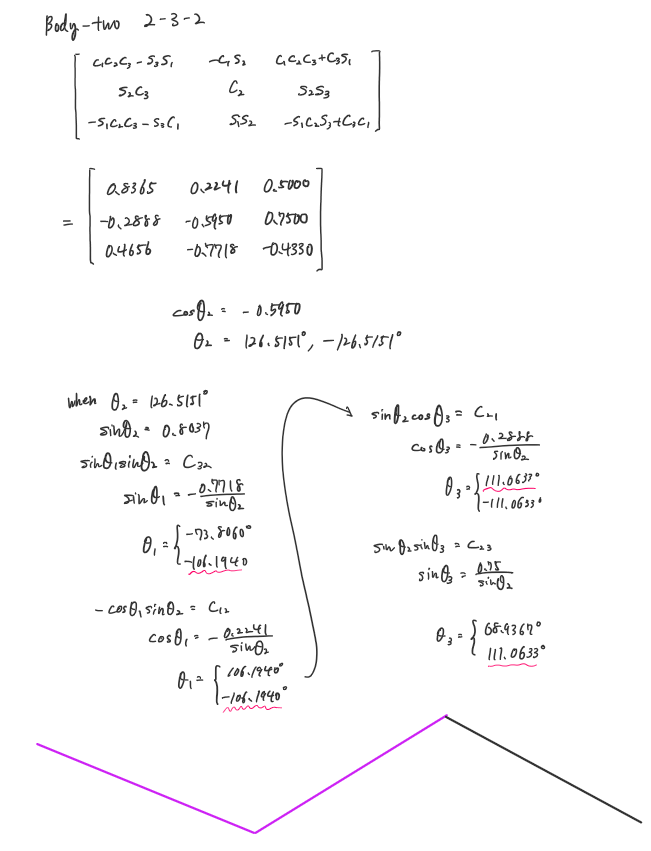


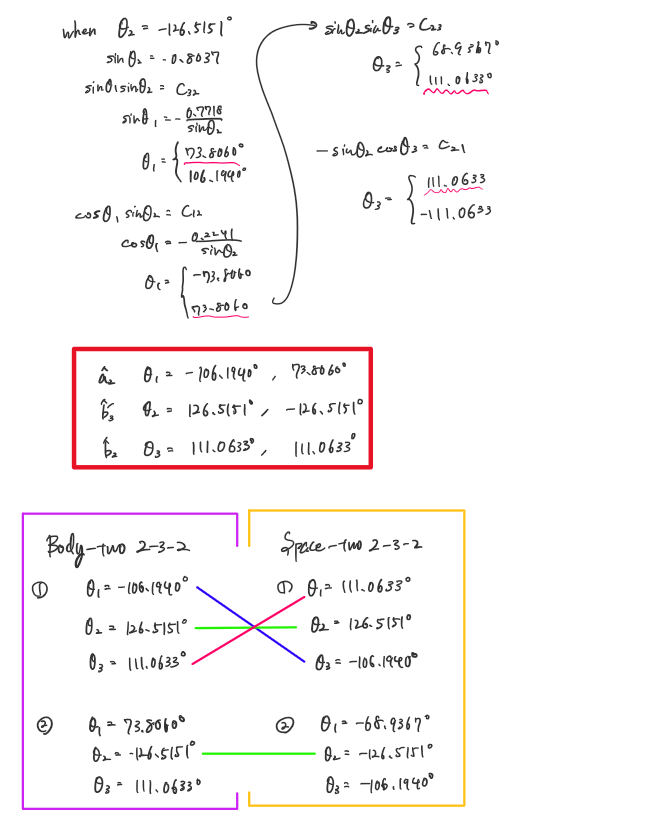


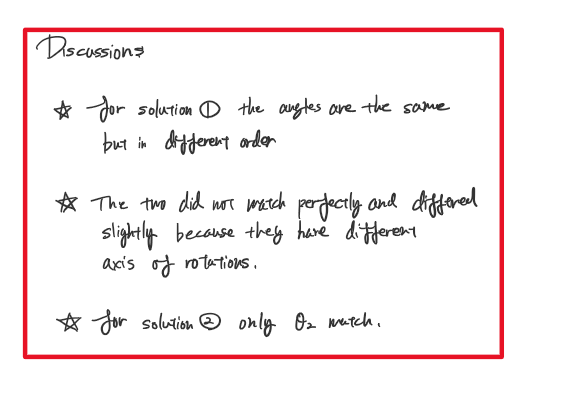


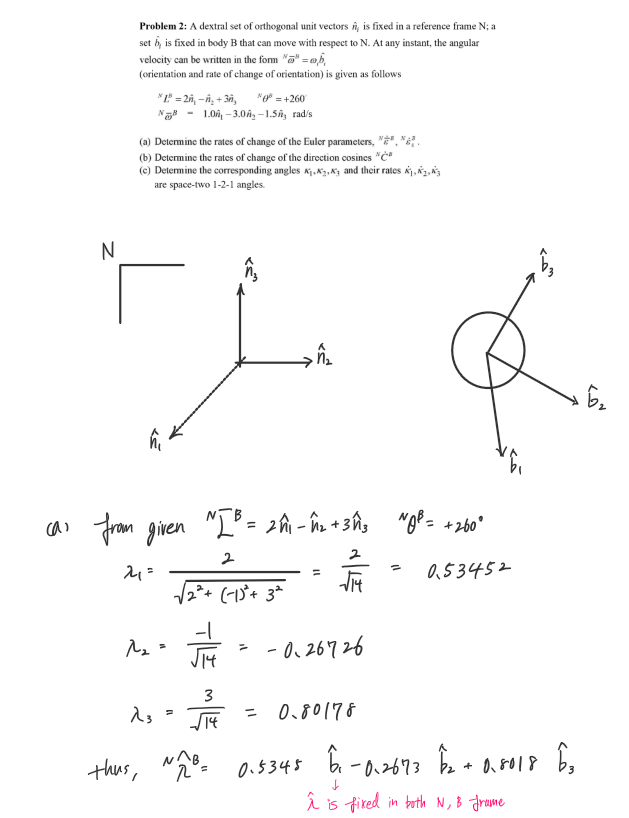


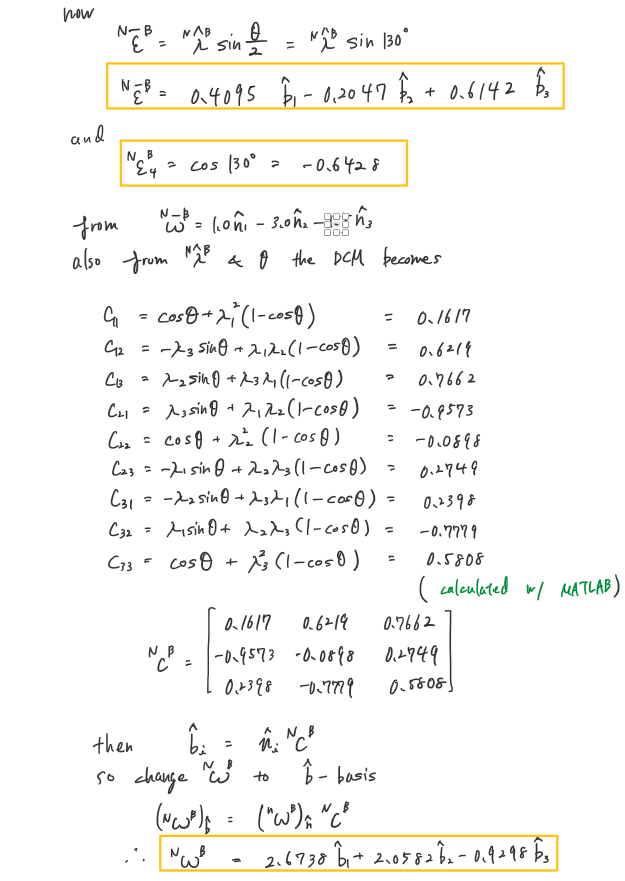


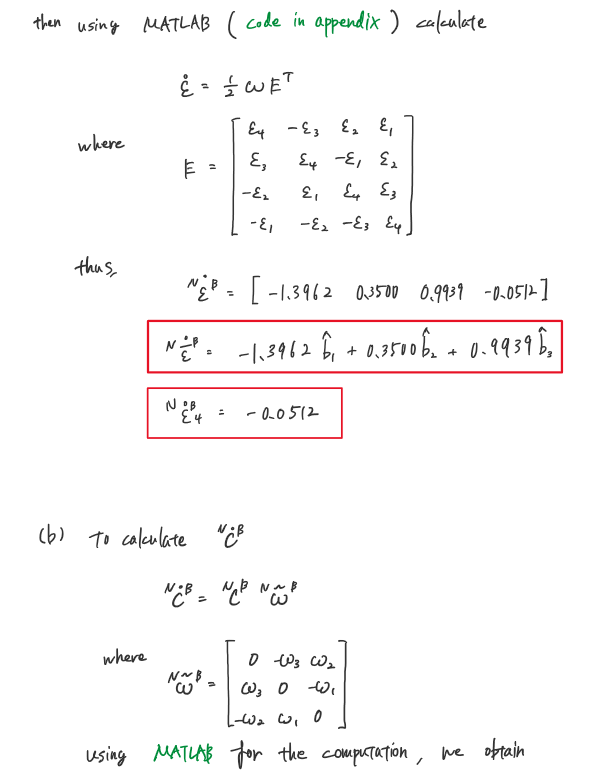


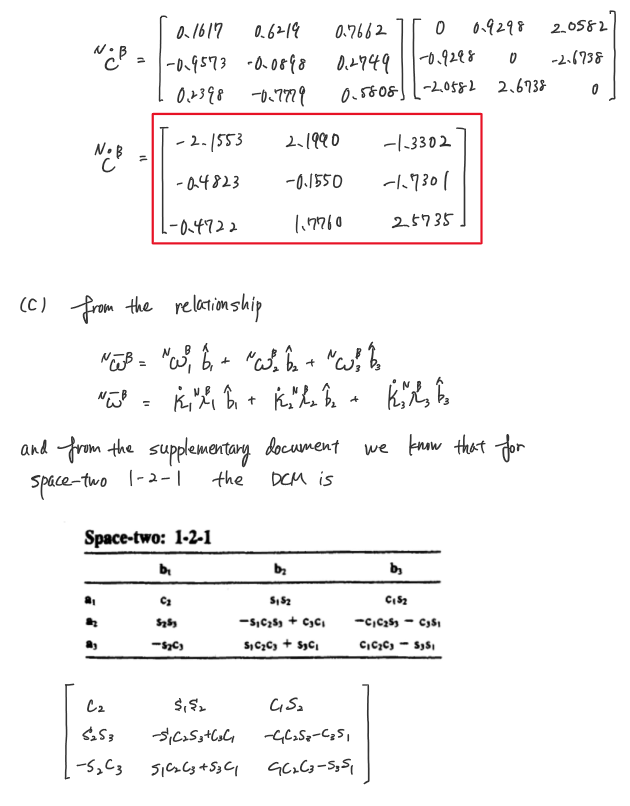


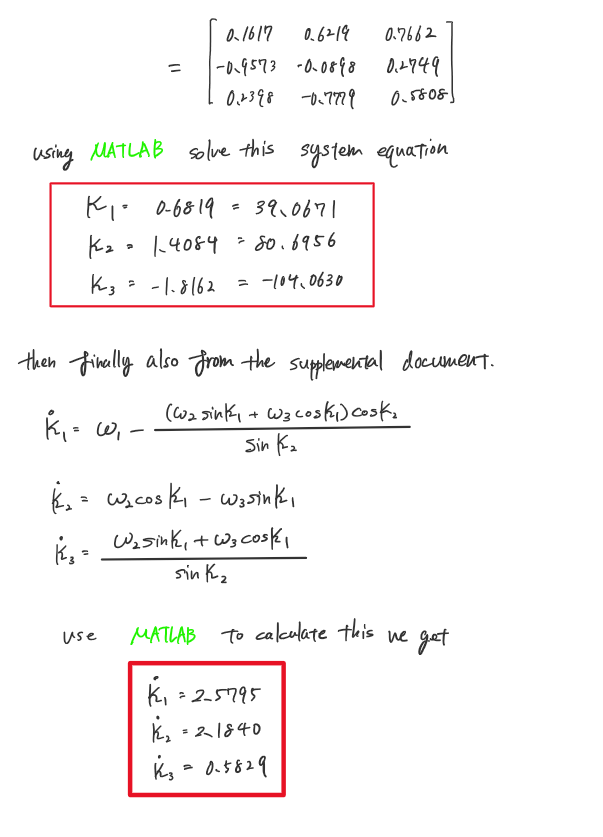


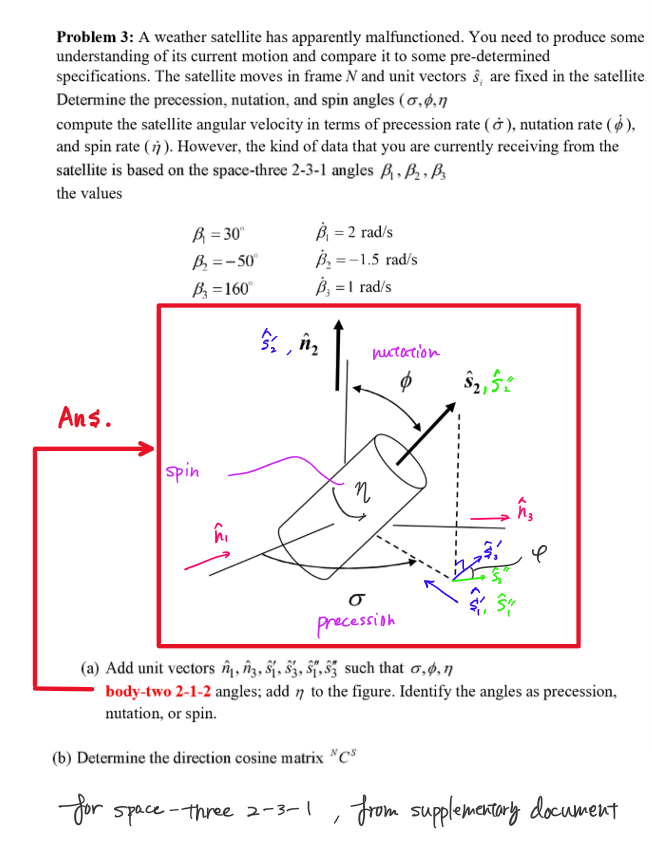


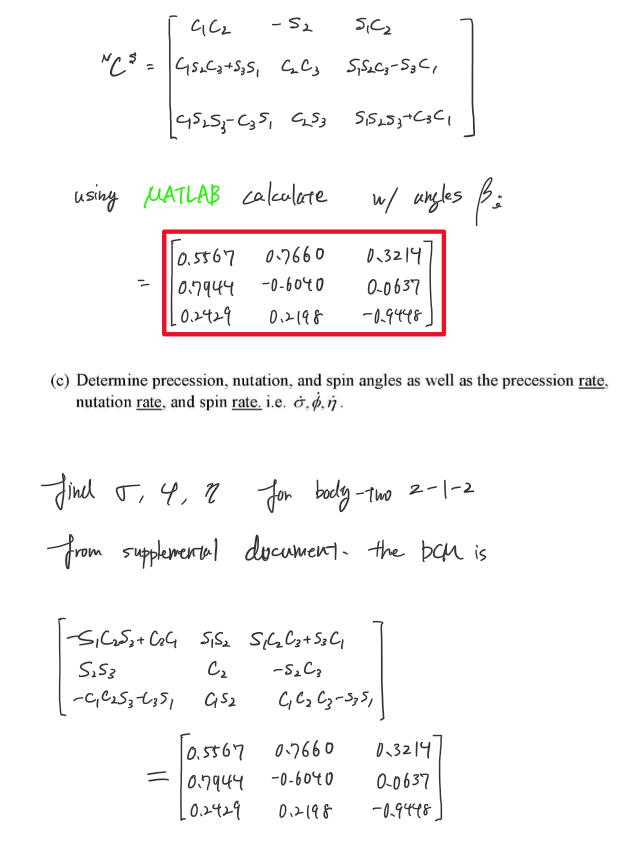


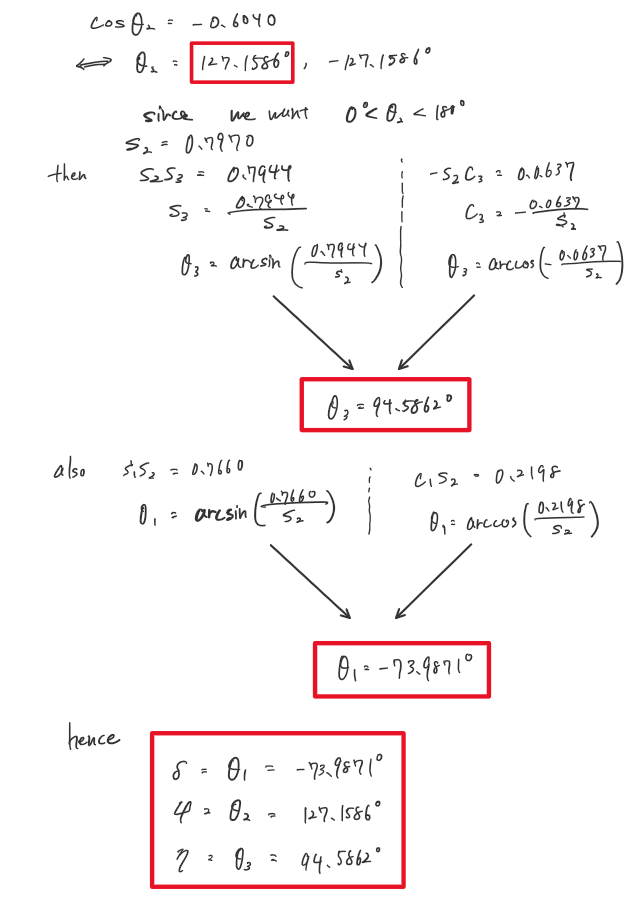


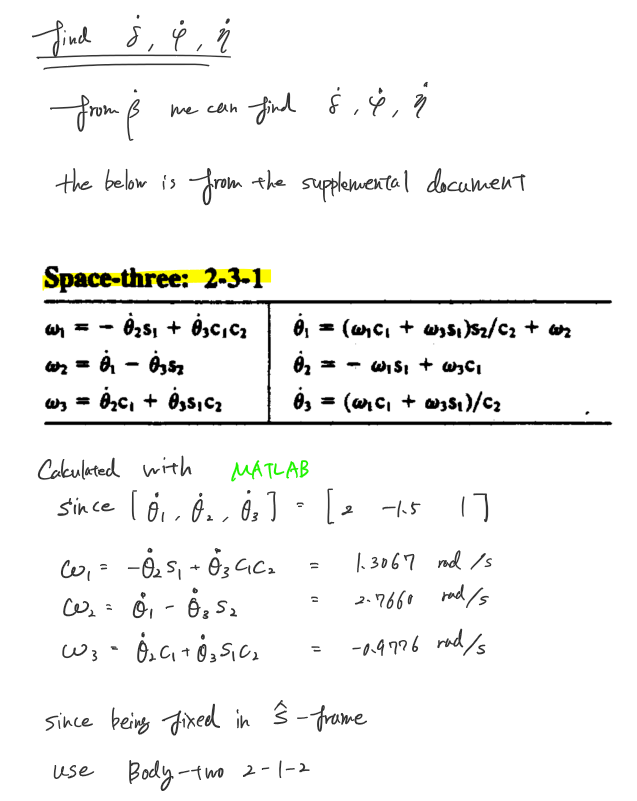


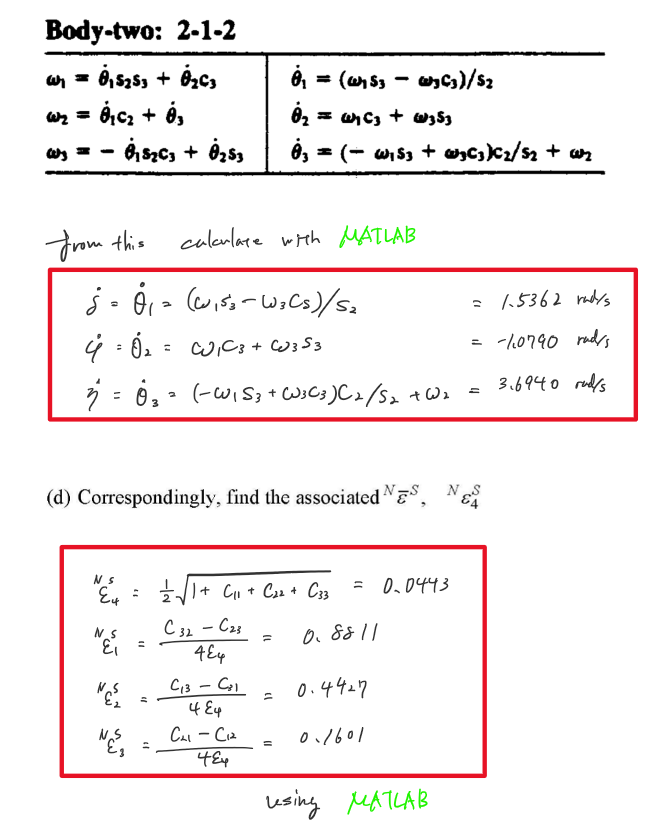


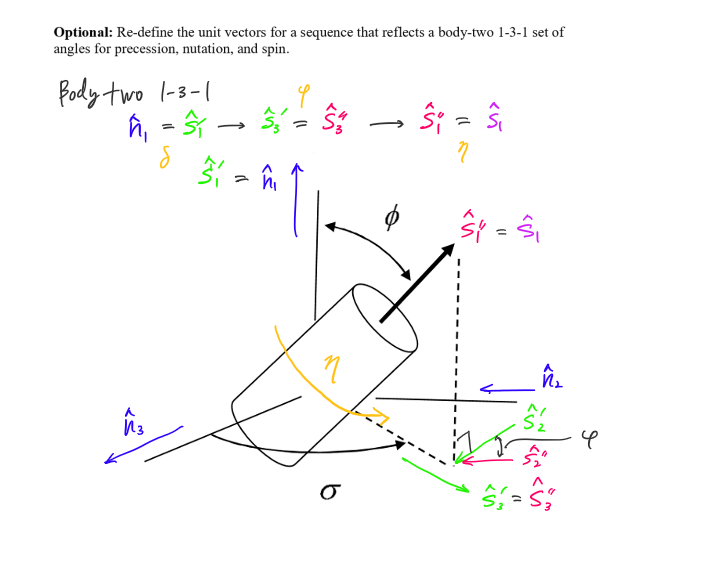












Appendix

## **AAE440 HW4 MATLAB CODE problem 1**

clear all; close all; clc;

% Defining body-three 1-2-3 system

theta\_d = [60 -210 165];

theta = deg2rad(theta\_d); % convert the thetas to radians

% Function that calculates DCM in Body System from angles and rotational axes

C\_body\_123 = DCM\_Body(1, 2, 3, theta(1), theta(2), theta(3));

function C\_body = DCM\_Body(Rot1, Rot2, Rot3, theta1, theta2, theta3)

Rot = [Rot1, Rot2, Rot3];

theta = [theta1, theta2, theta3]; % radians

C\_body = zeros([3,3]);

C = zeros([3,3,3]);

for i=1:3

if Rot(i) == 1

% DCM for rotation about axis 1

C(:,:,i) = [ 1 0 0;

0 cos(theta(i)) -sin(theta(i));

0 sin(theta(i)) cos(theta(i))];

elseif Rot(i) == 2

% DCM for rotation about axis 2

C(:,:,i) = [ cos(theta(i)) 0 sin(theta(i));

0 1 0;

-sin(theta(i)) 0 cos(theta(i))];

else

% DCM for rotation about axis 3

C(:,:,i) = [ cos(theta(i)) -sin(theta(i)) 0;

sin(theta(i)) cos(theta(i)) 0;

0 0 1];

end

end

C\_body = C(:,:,1)\*C(:,:,2)\*C(:,:,3);

end

## **AAE440 HW4 MATLAB CODE problem 2**

clear all; close all; clc;

#### <a>

% Given

L\_NB\_vec = [2 -1 3]; % n-hat

theta = 260/180\*pi;

% Calculate lambda

lambda\_NB = L\_NB\_vec/sqrt(sum(L\_NB\_vec.^2)); % b-hat

% Calculate epsilons

e\_NB\_vec = lambda\_NB\*sin(theta/2); % b-hat

e\_NB\_4 = cos(theta/2);

% Angular velocity

omega\_NB\_n = [1 -3 -1.5]; % n-hat

% Find the DCM

C\_NB = DCM\_from\_EulerAxisAng(lambda\_NB,theta);

% Change basis of omega from n-hat to b-hat

omega\_NB\_b = omega\_NB\_n\*C\_NB;

e\_NB\_dot = EulerParaDot\_from\_AngVel([e\_NB\_vec, e\_NB\_4], [omega\_NB\_b, 0]);

#### <b>

% Calculating the rate of change of DCM

C\_dot = DCM\_Dot\_from\_PoissonKineEq(C\_NB, omega\_NB\_b);

#### <c>

% Angles

kappa2 = acos(C\_NB(1,1)); % Neglecting the second solution (Class Convention)

kappa1 = asin(C\_NB(1,2)/sin(kappa2));

kappa1 = acos(C\_NB(1,3)/sin(kappa2)); % Check if it satisfies condition

kappa3 = -(pi+asin(C\_NB(2,1)/sin(kappa2)));

kappa3 = -acos(-C\_NB(3,1)/sin(kappa2));

% Angles in degrees

kappa1\_d = rad2deg(kappa1);

kappa2\_d = rad2deg(kappa2);

kappa3\_d = rad2deg(kappa3);

% Angular velocities

omega = omega\_NB\_b;

kappa1\_dot = omega(1) - (omega(2)\*sin(kappa1) + omega(3)\*cos(kappa1))\*cos(kappa2)/sin(kappa2);

kappa2\_dot = omega(2)\*cos(kappa1) - omega(3)\*sin(kappa1);

kappa3\_dot = (omega(2)\*sin(kappa1) + omega(3)\*cos(kappa1))/sin(kappa2);

function C\_mat = DCM\_from\_EulerAxisAng(lambdas, theta)

% Euler Axis

lambda1 = lambdas(1);

lambda2 = lambdas(2);

lambda3 = lambdas(3);

% Calculating DCM from Euler Axis and Euler Angle

C11 = cos(theta) + lambda1^2\*(1-cos(theta));

C12 = -lambda3\*sin(theta) + lambda1\*lambda2\*(1-cos(theta));

C13 = lambda2\*sin(theta) + lambda3\*lambda1\*(1-cos(theta));

C21 = lambda3\*sin(theta) + lambda1\*lambda2\*(1-cos(theta));

C22 = cos(theta) + lambda2^2\*(1-cos(theta));

C23 = -lambda1\*sin(theta) + lambda2\*lambda3\*(1-cos(theta));

C31 = -lambda2\*sin(theta) + lambda3\*lambda1\*(1-cos(theta));

C32 = lambda1\*sin(theta) + lambda2\*lambda3\*(1-cos(theta));

C33 = cos(theta) + lambda3^2\*(1-cos(theta));

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

end

function e\_dot = EulerParaDot\_from\_AngVel(epsilons, omega)

%{

epsilons: 1x4 row vector with 3 vector elements and the 4th being a

scalar element

omega: 1x4 row vector with 4th element equal to 0

%}

e1 = epsilons(1);

e2 = epsilons(2);

e3 = epsilons(3);

e4 = epsilons(4);

% E matrix

E = [e4 -e3 e2 e1; e3 e4 -e1 e2; -e2 e1 e4 e3; -e1 -e2 -e3 e4];

% Output

e\_dot = 0.5\*omega\*E.';

end

function C\_dot = DCM\_Dot\_from\_PoissonKineEq(C\_mat, omegas)

omega1 = omegas(1);

omega2 = omegas(2);

omega3 = omegas(3);

omega\_wave = [0 -omega3 omega2; omega3 0 -omega1; -omega2 omega1 0];

C\_dot = C\_mat\*omega\_wave;

end

## **AAE440 HW4 MATLAB CODE problem 3**

clear all; close all; clc;

% Angles and angular velocities

Beta = deg2rad([30 -50 160]); % rad

Beta\_dot = [2 -1.5 1]; % rad s-1

% Creating DCM for space three 2-3-1 (N\_C\_S)

C\_NS\_space\_231 = DCM\_Space(2, 3, 1, Beta(1), Beta(2), Beta(3));

delta = 1.291318577654501; % Precession

phi = 2.219335441889180; % Nutation

eta = 1.650841195002049; % Spin

theta1 = delta;

theta2 = phi;

theta3 = eta;

% Calculating omega

omega1 = -Beta\_dot(2)\*sin(Beta(1)) + Beta\_dot(3)\*cos(Beta(1))\*cos(Beta(2));

omega2 = Beta\_dot(1) - Beta\_dot(3)\*sin(Beta(2));

omega3 = Beta\_dot(2)\*cos(Beta(1)) + Beta\_dot(3)\*sin(Beta(1))\*cos(Beta(2));

% Rates

delta\_dot = (omega1\*sin(theta3) - omega3\*cos(theta3))/sin(theta2);

phi\_dot = omega1\*cos(theta3) + omega3\*sin(theta3);

eta\_dot = (-omega1\*sin(theta3) + omega3\*cos(theta3))\*cos(theta2)/sin(theta2) + omega2;

% Calcuate the euler parameters

epsilons = EulerPara\_from\_DCM(C\_NS\_space\_231);

function epsilons = EulerPara\_from\_DCM(C\_mat)

epsilon4 = 0.5\*sqrt(1+C\_mat(1,1)+C\_mat(2,2)+C\_mat(3,3));

epsilon1 = (C\_mat(3,2)-C\_mat(2,3))/4/epsilon4;

epsilon2 = (C\_mat(1,3)-C\_mat(3,1))/4/epsilon4;

epsilon3 = (C\_mat(2,1)-C\_mat(1,2))/4/epsilon4;

epsilons = [epsilon1 epsilon2 epsilon3 epsilon4];

end