University of Colorado - Boulder

ASEN 3801
Aerospace Vehicle Dynamics and Controls Lab | Fall 2023

ASEN 3801 Lab 2

Authors:
Hayden Gebhardt
Greg Kornguth
Sultan AlBinAli

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Contents

1 Problems

1.1 Problem 1

Description	ASEN 3728 Notation	ASEN 3700 Notation
Inerial position of the vehicle expressed in body coordinates.	p_B^E	${}^BP_{B/N}$
Inertial velocity of the vehicle expressed in body coordinates	V_B^E	${}^BV_{B/N}$
Inertial position of the target with body frame A	t_A^E	${}^Nt_{A/N}$
Rotation matrix from body frame to inertial frame	R_B^E	Q_B^N
Velocity of vehicle relative to target in inertial coordinates	v_B^t	${}^Nv_{T/N}$
Rotation matrix from frame A to B	R_A^B	Q_A^B

1.2 Problem 2

In this assignment, we are tasked with manipulating data collected in the Autonomous Systems Programming, Evaluation, and Networking (ASPEN) Lab, which contains a motion capture system. This system uses overhead cameras to track the three-dimensional position and orientation of objects in a 30 ft x 75 ft x 20 ft space. Two objects are being controlled in this environment: a Tello multirotor drone (referred to as the aerospace vehicle) and a pedestrian wearing a hard hat with motion capture markers (referred to as the target).

The aerospace vehicle, which is the Tello multirotor drone, is likely to exhibit controlled and dynamic motion. It performs various flight maneuvers, including hovering, ascending, descending, and translational motion (forward, backward, left, right. The motion includes smooth, continuous trajectories as well as abrupt changes in direction. Frame A is defined as the body frame of the aerospace vehicle, with its origin at the center of mass of the vehicle. This means that the orientation and position of Frame A will change as the drone moves.

The target is a pedestrian wearing a hard hat with motion capture markers. The target's motion is likely to be more varied and natural compared to the aerospace vehicle. The pedestrian walks and stands still at various points during data collection. Like the aerospace vehicle, Frame T is defined as the body frame of the target, with its origin at the center of mass of the target. As the pedestrian moves, the orientation and position of Frame T will change accordingly.

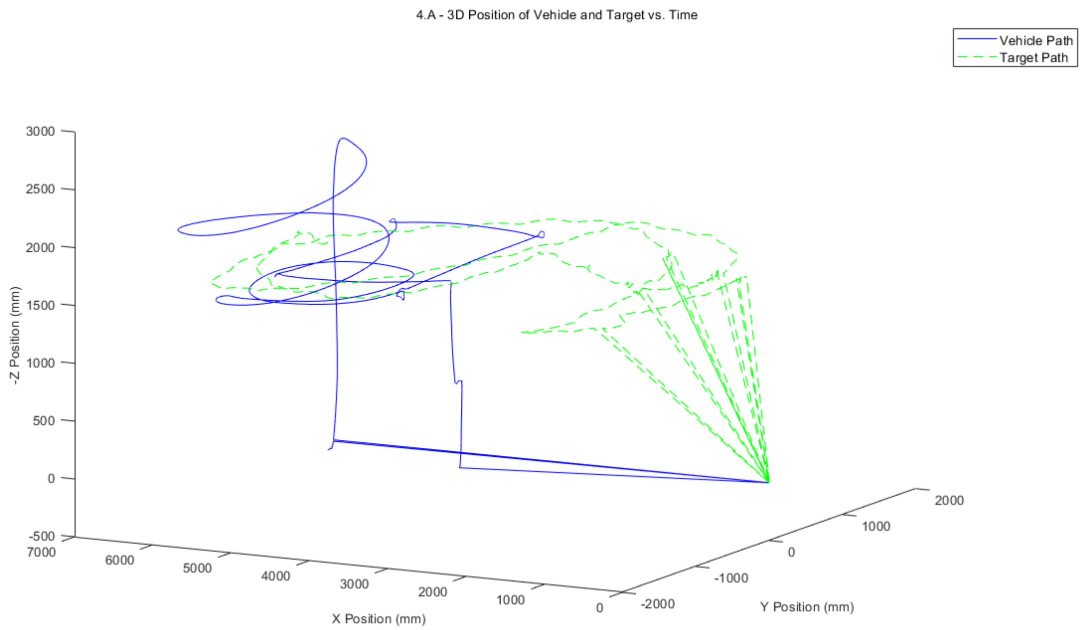
Overall, the motion of both the aerospace vehicle and the target will be characterized by their respective frames of reference (Frame A and Frame T), which will help in understanding their positions and orientations in the lab space.

1.3 Problem 3

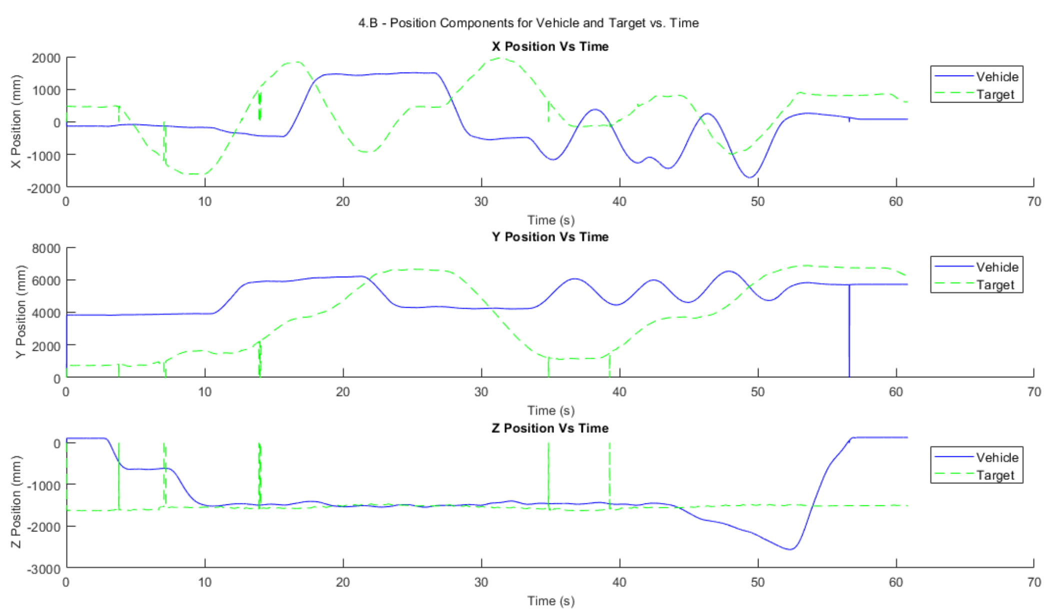
See appendix (2) or attached files for problem 3 functions.

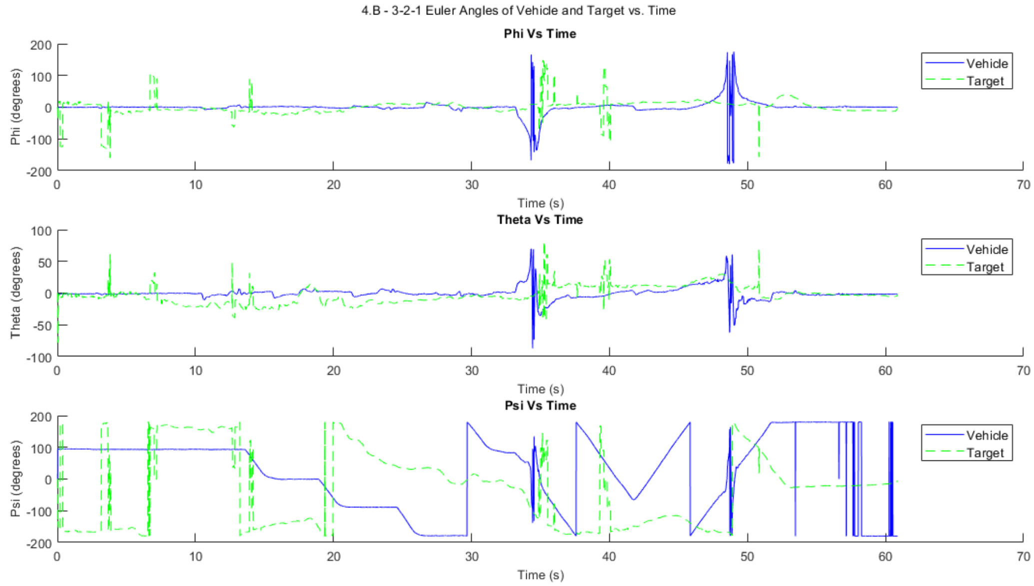
1.4 Problem 4

1.4.1 Part A



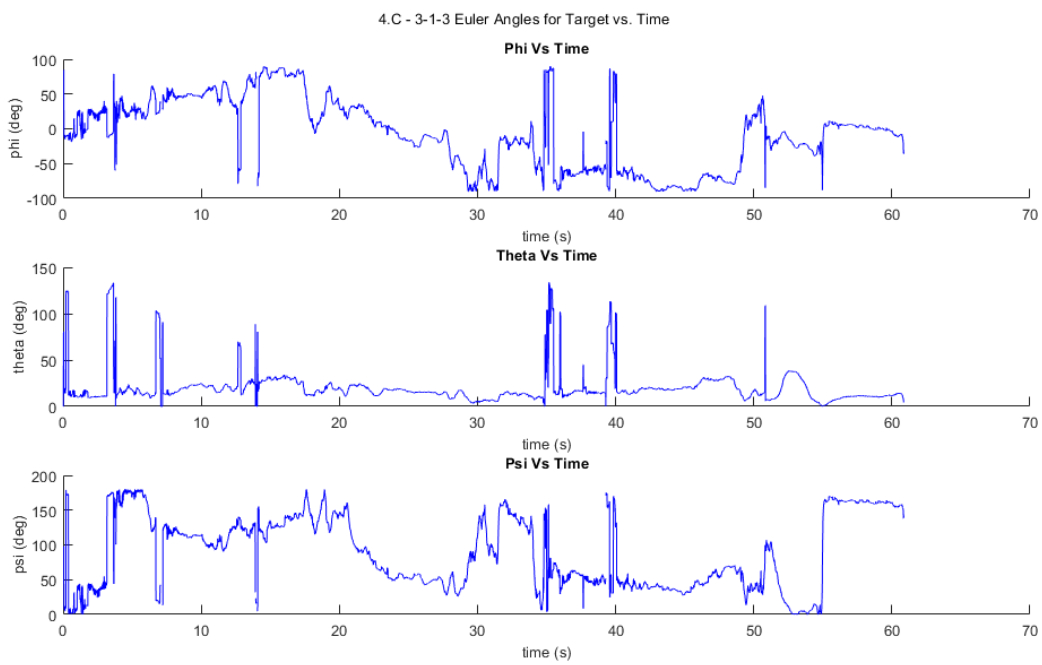
1.4.2 Part B



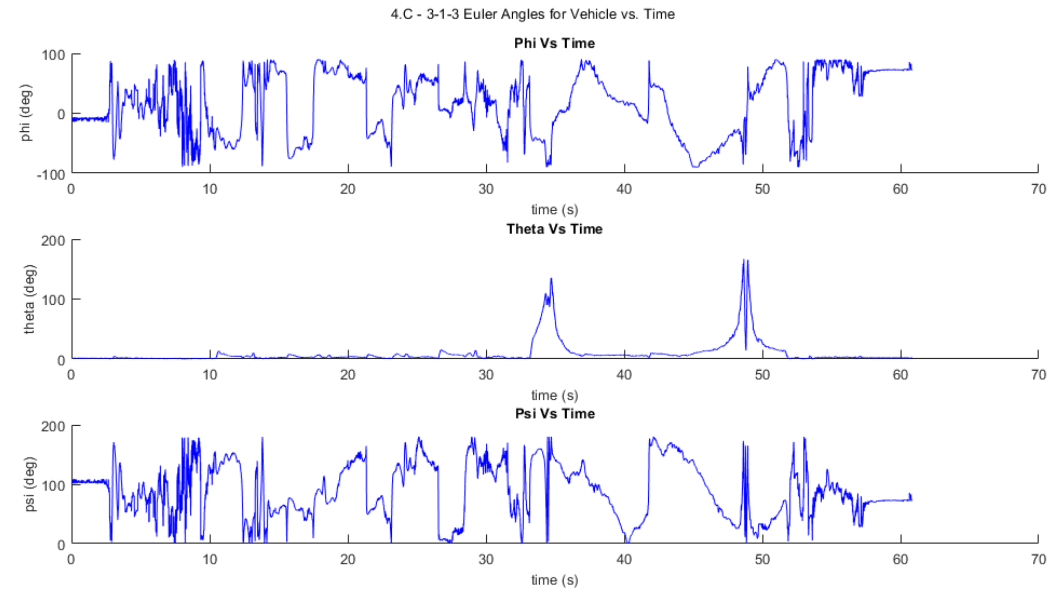


4.B - 3-2-1 Euler Angles of Vehicle and Target vs. Time
Phi Vs Time
Phi (degrees)
Time (s)
Vehicle
Target
Theta Vs Time
Theta (degrees)
Time (s)
Vehicle
Target
Psi Vs Time
Psi (degrees)
Time (s)
Vehicle
Target

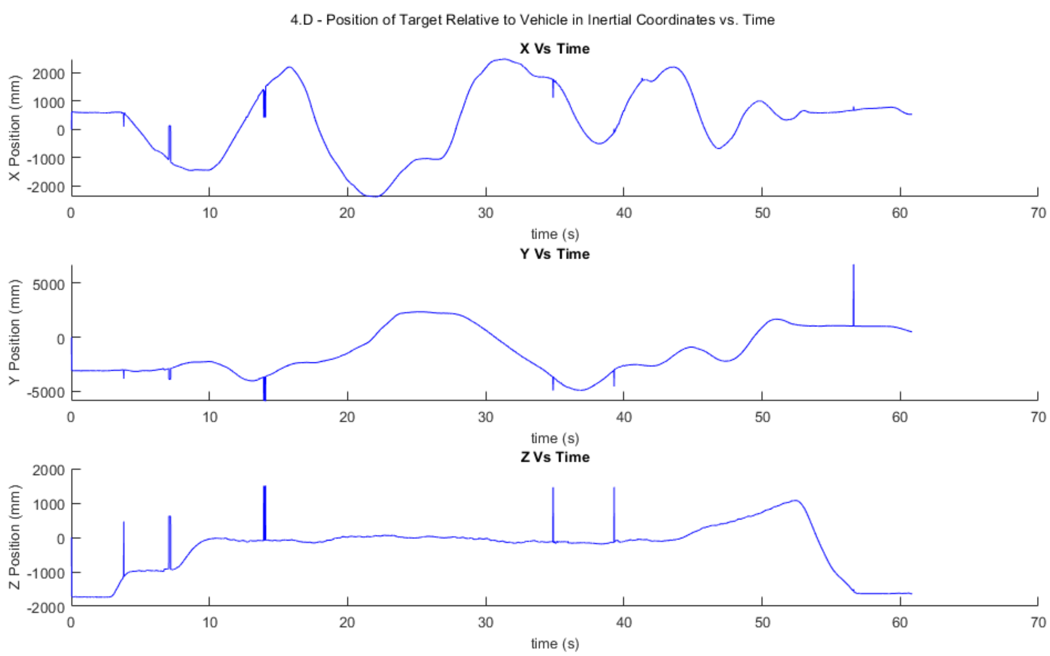
1.4.3 Part C



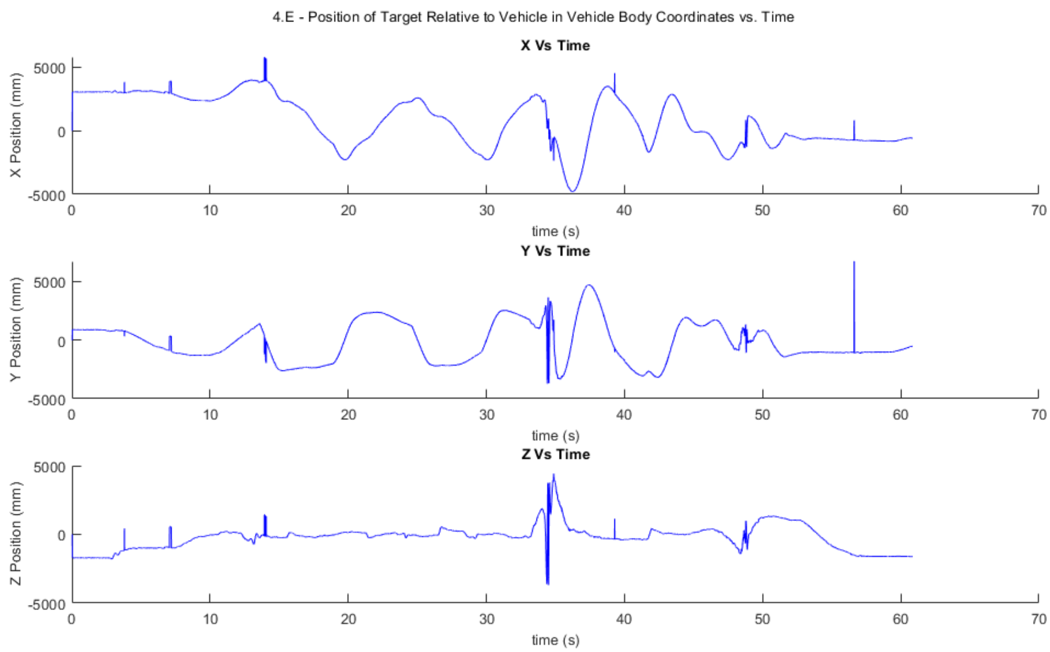
4.C - 3-1-3 Euler Angles for Target vs. Time
Phi Vs Time
phi (deg)
time (s)
Theta Vs Time
theta (deg)
time (s)
Psi Vs Time
psi (deg)
time (s)


4.C - 3-1-3 Euler Angles for Vehicle vs. Time
Phi Vs Time
phi (deg)
time (s)
Theta Vs Time
theta (deg)
time (s)
Psi Vs Time
psi (deg)
time (s)

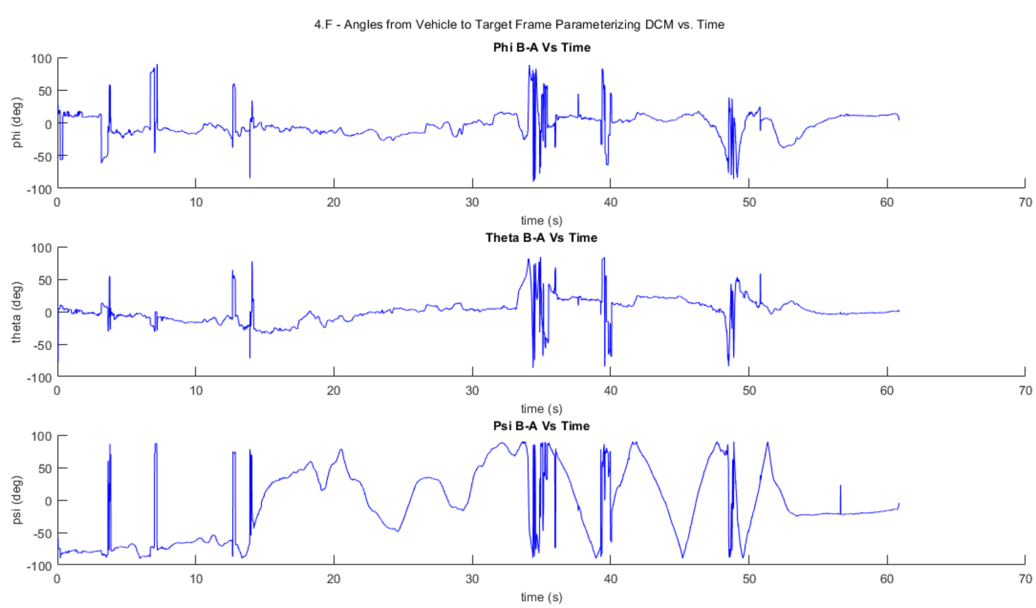
1.4.4 Part D


4.D - Position of Target Relative to Vehicle in Inertial Coordinates vs. Time
X Vs Time
X Position (mm)
time (s)
Y Vs Time
Y Position (mm)
time (s)
Z Vs Time
Z Position (mm)
time (s)

1.4.5 Part E



1.4.6 Part F



2 Appendix

2.1 MATLAB Code

2.1.1 main.m

```
% Hayden Gebhardt, Greg Kornguth, Sultan Albinali
% ASEN 3728 - Lab 2
% Updated: 09/29/2023

%% House Keeping

clc; clear; close all;

%% Main Code

% part A

[t_vec, av_pos_inert, av_att, tar_pos_inert, tar_att] = LoadASPENData("3801_Sec1_Test1.csv");

figure(1)
hold on
plot3(av_pos_inert(1,:), av_pos_inert(2,:), -av_pos_inert(3,:), "b") % Drone position plot
plot3(tar_pos_inert(1,:), tar_pos_inert(2,:), -tar_pos_inert(3,:), "--g") % Target position plot
view(3)
ylabel("X Position (mm)")
xlabel("Y Position (mm)")
zlabel("-Z Position (mm)")
legend("Vehicle Path", "Target Path")
sgtitle("4.A - 3D Position of Vehicle and Target vs. Time")
fontsize(figure(1), 10,"points")
hold off

% part B

figure(2)
subplot(3,1,1)
hold on
plot(t_vec, av_pos_inert(1, :),"b")
plot(t_vec, tar_pos_inert(1, :),"--g")
title("X Position Vs Time")
xlabel("Time (s)")
ylabel("X Position (mm)")
legend("Vehicle", "Target")
hold off
subplot(3,1,2)
hold on
plot(t_vec, av_pos_inert(2, :),"b")
plot(t_vec, tar_pos_inert(2, :),"--g")
title("Y Position Vs Time")
xlabel("Time (s)")
ylabel("Y Position (mm)")
legend("Vehicle", "Target")
hold off
subplot(3,1,3)
hold on
plot(t_vec, av_pos_inert(3, :),"b")
plot(t_vec, tar_pos_inert(3, :),"--g")
title("Z Position Vs Time")
```

```
xlabel("Time (s)")
ylabel("Z Position (mm)")
legend("Vehicle", "Target")

hold off
sgtitle("4.B - Position Components for Vehicle and Target vs. Time")
fontsize(figure(2), 10,"points")

figure(3)
subplot(3,1,1)
hold on
plot(t_vec, av_att(1,:).*180./pi, "b")
plot(t_vec, tar_att(1,:).*180./pi, "--g")
title("Phi Vs Time")
xlabel("Time (s)")
ylabel("Phi (degrees)")
legend("Vehicle", "Target")
hold off
subplot(3,1,2)
hold on
plot(t_vec, av_att(2,:).*180./pi,"b")
plot(t_vec, tar_att(2,:).*180./pi,"--g")
title("Theta Vs Time")
xlabel("Time (s)")
ylabel("Theta (degrees)")
legend("Vehicle", "Target")
hold off
subplot(3,1,3)
hold on
plot(t_vec, av_att(3,:).*180./pi,"b")
plot(t_vec, tar_att(3,:).*180./pi,"--g")
title("Psi Vs Time")
xlabel("Time (s)")
ylabel("Psi (degrees)")
legend("Vehicle", "Target")
hold off
sgtitle("4.B - 3-2-1 Euler Angles of Vehicle and Target vs. Time")
fontsize(figure(3), 10,"points")

% part C

av_att313 = zeros(3,length(av_att));
for i = 1:length(av_att)

    R321_rel = RotationMatrix321(av_att(:,i)); % creating 321 rotation matrix for each vehicle attitude
    av_att313(:,i) = EulerAngles313(R321_rel); % calculate 313 euler angles from each 321 matrix
end

av_att313 = av_att313(:, 4:end); % pruning NaNs from dataset
t_vec2 = t_vec(4:end, :);

figure(4)
subplot(3,1,1)
hold on
plot(t_vec2, av_att313(1,:).*180./pi,"b")
title("Phi Vs Time")
xlabel("time (s)")
ylabel("phi (deg)")
```

```
hold off
subplot(3,1,2)
hold on
plot(t_vec2, av_att313(2,:).*180./pi,"b")
title("Theta Vs Time")
xlabel("time (s)")
ylabel("theta (deg)")
hold off
subplot(3,1,3)
hold on
plot(t_vec2, av_att313(3,:).*180./pi,"b")
title("Psi Vs Time")
xlabel("time (s)")
ylabel("psi (deg)")
hold off
sgtitle("4.C - 3-1-3 Euler Angles for Vehicle vs. Time")
fontsize(figure(4), 10,"points")

tar_att313 = zeros(3,length(tar_att));
for i = 1:length(tar_att)

    R321_rel = RotationMatrix321(tar_att(:,i)); % creating 321 rotation matrix for each target attitude
    tar_att313(:,i) = EulerAngles313(R321_rel); % calculate 313 euler angles from each 321 matrix
end

figure(5)
subplot(3,1,1)
hold on
plot(t_vec, tar_att313(1,:).*180./pi,"b")
title("Phi Vs Time")
xlabel("time (s)")
ylabel("phi (deg)")
hold off
subplot(3,1,2)
hold on
plot(t_vec, tar_att313(2,:).*180./pi,"b")
title("Theta Vs Time")
xlabel("time (s)")
ylabel("theta (deg)")
hold off
subplot(3,1,3)
hold on
plot(t_vec, tar_att313(3,:).*180./pi,"b")
title("Psi Vs Time")
xlabel("time (s)")
ylabel("psi (deg)")
hold off
sgtitle("4.C - 3-1-3 Euler Angles for Target vs. Time")
fontsize(figure(5), 10,"points")

% part D

tar_rel_av = tar_pos_inert - av_pos_inert; % calculate inertial position of target relative to vehicle

figure(6)
subplot(3,1,1)
hold on
```

```
plot(t_vec, tar_rel_av(1,:),"b")
title("X Vs Time")
xlabel("time (s)")
ylabel("X Position (mm)")
hold off
subplot(3,1,2)
hold on
plot(t_vec, tar_rel_av(2, :),"b")
title("Y Vs Time")
xlabel("time (s)")
ylabel("Y Position (mm)")
hold off
subplot(3,1,3)
hold on
plot(t_vec, tar_rel_av(3, :),"b")
title("Z Vs Time")
xlabel("time (s)")
ylabel("Z Position (mm)")
hold off
sgtitle("4.D - Position of Target Relative to Vehicle in Inertial Coordinates vs. Time")
fontsize(figure(6), 10,"points")

% part E

tar_rel_av_body = zeros(3,length(av_att));
for i = 1:length(av_att)

    R321_rel = RotationMatrix321(av_att(:, i)); % creating 321 rotation matrix for each vehicle attitude

    % multiply by DCM to get from inertial position of target relative
    % to vehicle to body coordinates
    tar_rel_av_body(:, i) = R321_rel\tar_rel_av(:, i);
end

figure(7)
subplot(3,1,1)
hold on
plot(t_vec, tar_rel_av_body(1,:),"b")
title("X Vs Time")
xlabel("time (s)")
ylabel("X Position (mm)")
hold off
subplot(3,1,2)
hold on
plot(t_vec, tar_rel_av_body(2, :),"b")
title("Y Vs Time")
xlabel("time (s)")
ylabel("Y Position (mm)")
hold off
subplot(3,1,3)
hold on
plot(t_vec, tar_rel_av_body(3, :),"b")
title("Z Vs Time")
xlabel("time (s)")
ylabel("Z Position (mm)")
hold off
sgtitle("4.E - Position of Target Relative to Vehicle in Vehicle Body Coordinates vs. Time")
fontsize(figure(7), 10,"points")
```

```
% part F

av_to_tar = zeros(3,length(av_att));
for i = 1:length(av_att)

    R321_BtoN = RotationMatrix321(av_att(:, i)); % calculate 321 rotation matrix from vehicle
        frame B to intermediate N frame
    R321_NtoA = inv(RotationMatrix321(tar_att(:, i))); % calculate 321 rotation matrix from N to
        target frame A

    R321_BtoA = R321_NtoA * R321_BtoN; % calculate full DCM from vehicle frame to target frame

    EulerAngles321_BtoA(:, i) = EulerAngles321(R321_BtoA); % calculate resulting 321 euler angles
        for each data point
end

EulerAngles321_BtoA = EulerAngles321_BtoA(:, 4:end); % pruning NaNs from dataset

figure(8)
subplot(3,1,1)
hold on
plot(t_vec2, EulerAngles321_BtoA(1, :).*180./pi,"b")
title("Phi B-A Vs Time")
xlabel("time (s)")
ylabel("phi (deg)")
hold off
subplot(3,1,2)
hold on
plot(t_vec2, EulerAngles321_BtoA(2, :).*180./pi,"b")
title("Theta B-A Vs Time")
xlabel("time (s)")
ylabel("theta (deg)")
hold off
subplot(3,1,3)
hold on
plot(t_vec2, EulerAngles321_BtoA(3, :).*180./pi,"b")
title("Psi B-A Vs Time")
xlabel("time (s)")
ylabel("psi (deg)")
hold off
sgtitle("4.F - Angles from Vehicle to Target Frame Parameterizing DCM vs. Time")
fontsize(figure(8), 10,"points")
```

2.1.2 LoadASPENData.m

```
function [t_vec, av_pos_inert, av_att, tar_pos_inert, tar_att] = LoadASPENData(filename)

% ---------------------------------------------------------------------- %
% Inputs:      filename = name of data file for processing
%
% Outputs:        t_vec = time vector scaled to data refresh rate (s)
%          av_pos_inert = inertial position of vehicle (mm)
%                av_att = attitude angles of vehicle (rad)
%         tar_pos_inert = inertial position of target (mm)
%               tar_att = attitude angles of target (rad)
%
% Methodology: Read in data file and parse columns for each output listed
```

```
%              above. Data file used for this lab is 100 Hz, but change the
%              denominator in t_vec to the refresh rate of whatever data is
%              being used. Raw data is then passed into ConvertASPENData so
%              that the correct outputs can be produced, and NaNs can be
%              removed.
% ---------------------------------------------------------------------- %

    data = readmatrix(filename);
    pos_av_aspen = data(:, 12:14)';
    att_av_aspen = data(:, 9:11)';
    pos_tar_aspen = data(:, 6:8)';
    att_tar_aspen = data(:, 3:5)';
    t_vec = 0:1:length(data) - 1;
    t_vec = (t_vec/100)';
    [av_pos_inert, av_att, tar_pos_inert, tar_att] = ConvertASPENData(pos_av_aspen, att_av_aspen,
        pos_tar_aspen, att_tar_aspen);

end
```

2.1.3 EulerAngles321.m

```
function attitude = EulerAngles321(RotMat)

% ---------------------------------------------------------------------- %
% Inputs:    RotMat = 3x3 input rotation matrix
%
% Outputs: attitude = attitude representation for one data frame
%                   = [phi, theta, psi] (rad)
%
% Methodology: Calculate the 3-2-1 euler angles for corresponding rotation
%              matrix.
% ---------------------------------------------------------------------- %

    theta = -asin(RotMat(1, 3));

    psi = asin(RotMat(1, 2) / cos(theta));

    phi = asin(RotMat(2, 3) / cos(theta));

    attitude = [phi, theta, psi]';

end
```

2.1.4 EulerAngles313.m

```
function attitude = EulerAngles313(RotMat)

% ---------------------------------------------------------------------- %
% Inputs:    RotMat = 3x3 input rotation matrix
%
% Outputs: attitude = attitude representation for one data frame
%                   = [omega, i, Omega] (rad)
%
% Methodology: Calculate the 3-1-3 euler angles for corresponding rotation
%              matrix.
% ---------------------------------------------------------------------- %

    i = acos(RotMat(3, 3));
```

```

    Omega = acos(RotMat(2, 3) / sin(i));

    omega = asin(RotMat(3, 1) / sin(i));

    attitude = [omega, i, Omega]';

end
```

2.1.5 RotationMatrix321.m

```
function R321 = RotationMatrix321(attitude)

% ---------------------------------------------------------------------- %
% Inputs: attitude = attitude representation for one data frame
%                  = [phi, theta, psi] (rad)
%
% Outputs:    R321 = 3-2-1 rotation matrix comprised of three angle DCMs
%
% Methodology: Calculate the 3-2-1 rotation matrix for corresponding euler
%             angles.
% ---------------------------------------------------------------------- %

    phi = attitude(1);
    theta = attitude(2);
    psi = attitude(3);

    R1 = [1, 0, 0;
          0, cos(phi), sin(phi);
          0, -sin(phi), cos(phi)];

    R2 = [cos(theta), 0, -sin(theta);
          0, 1, 0;
          sin(theta), 0, cos(theta)];

    R3 = [cos(psi), sin(psi), 0;
          -sin(psi), cos(psi), 0;
          0, 0, 1];

    R321 = R1 * R2 * R3;

end
```

2.1.6 RotationMatrix313.m

```
function R313 = RotationMatrix313(attitude)

% ---------------------------------------------------------------------- %
% Inputs: attitude = attitude representation for one data frame
%                  = [omega, i, Omega] (rad)
%
% Outputs:    R313 = 3-1-3 rotation matrix comprised of three angle DCMs
%
% Methodology: Calculate the 3-1-3 rotation matrix for corresponding euler
%             angles.
% ---------------------------------------------------------------------- %

    omega = attitude(1);
```

```
    i = attitude(2);
    Omega = attitude(3);

    R1 = [1, 0, 0;
          0, cos(i), sin(i);
          0, -sin(i), cos(i)];

    R3_phi = [cos(omega), sin(omega), 0;
          -sin(omega), cos(omega), 0;
          0, 0, 1];

    R3_psi = [cos(Omega), sin(Omega), 0;
          -sin(Omega), cos(Omega), 0;
          0, 0, 1];

    R313 = R3_psi * R1 * R3_phi;

end
```

2.1.7 ConvertASPENData.m

```
% Eric W. Frew
% ASEN 3801
% ConvertASPENData
% Created: 9/21/23

function [pos_av_class, att_av_class, pos_tar_class, att_tar_class] =
    ConvertASPENData(pos_av_aspen, att_av_aspen, pos_tar_aspen, att_tar_aspen)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Inputs:  pos_av_aspen = 3 x n array of position vectors in coordinate
%                         system of the ASPEN motion capture system
%          att_av_aspen = 3 x n array of attitude vectors using the 'helical angles'
%                         provided by the ASPEN motion capture system
%          pos_tar_aspen = 3 x n array of position vectors in coordinate
%                         system of the ASPEN motion capture system
%          att_tar_aspen = 3 x n array of attitude vectors using the 'helical angles'
%                         provided by the ASPEN motion capture system
%
%
% Outputs: pos_av_class, att_av_class, pos_tar_class, att_tar_class
%
%         pos_av_class = 3 x n array of position vectors in 'class' coordinate
%                         system with same x-axis as ASPEN and y-axis and
%                         z-axis flipped so z is down
%          att_av_class = 3 x n array of attitude vectors using NASA Standard
%                         321 Euler angles
%          pos_tar_class = 3 x n array of position vectors in 'class' coordinate
%                         system with same x-axis as ASPEN and y-axis and
%                         z-axis flipped so z is down
%          att_tar_class = 3 x n array of attitude vectors using NASA Standard
%                         321 Euler angles
%
%
%
% Methodology: Converts the data provided by the ASPEN lab motion capture system into
% the convention used for ASEN 3801:
%   The inertial frame has down as positive z
```

```
37 %   The body frame has x out front, y to the right, z down
38 %   The angles are the NASA Standard 321 Euler angles
39 %
40 % Position data is converted by changing the signs of the y- and z-axes
41 %
42 % Attitude data is convereted by calculating the Direction Cosine Matrix
43 % from the ASPEN helical angles and then determining the Euler angles from
44 % the DCM. The conversion equations were taken from:
45 %   https://academicflight.com/articles/kinematics/rotation-formalisms/euler-angles/
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47 
48 
49 [rows, len] = size(pos_av_aspen);
50 
51 % This matrix converts from the ASPEN frame to the "class inertial" frame
52 % with Z down. Because the inverse of the matrix is the matrix I dont need
53 % an inverse or transpose in lines 72 or 89
54 P = [1 0 0;
55     0 -1 0;
56     0 0 -1];
57 
58 for i=1:len
59 
60     %%% Aerospace Vehicle data
61     % Input arrays have NaN if there was no data in the original file
62     % Replace with all zeros if that is the case
63     if(isnan(pos_av_aspen(:,i)))
64         pos_av_class(:,i) = zeros(3,1);
65         att_av_class(:,i) = zeros(3,1);
66 
67     else
68         pos_av_class(:,i) = [pos_av_aspen(1,i); -pos_av_aspen(2,i); -pos_av_aspen(3,i)];
69 
70         xfm_av =
             makehgtform('xrotate',att_av_aspen(1,i),'yrotate',att_av_aspen(2,i),'zrotate',att_av_aspen(3,i));
71         DCM_av = P*xfm_av(1:3,1:3)'*P;
72         att_av_class(1,i) = atan2(DCM_av(2,3), DCM_av(3,3));
73         att_av_class(2,i) = -asin(DCM_av(1,3));
74         att_av_class(3,i) = atan2(DCM_av(1,2), DCM_av(1,1));
75     end
76 
77 
78     %%% Target data
79     % Input arrays have NaN if there was no data in the original file
80     % Replace with all zeros if that is the case
81     if(isnan(pos_tar_aspen(:,i)))
82         pos_tar_class(:,i) = zeros(3,1);
83         att_tar_class(:,i) = zeros(3,1);
84     else
85         pos_tar_class(:,i) = [pos_tar_aspen(1,i); -pos_tar_aspen(2,i); -pos_tar_aspen(3,i)];
86 
87         xfm_tar =
             makehgtform('xrotate',att_tar_aspen(1,i),'yrotate',att_tar_aspen(2,i),'zrotate',att_tar_aspen(3,i));
88         DCM_tar = P*xfm_tar(1:3,1:3)'*P;
89         att_tar_class(1,i) = atan2(DCM_tar(2,3), DCM_tar(3,3));
90         att_tar_class(2,i) = -asin(DCM_tar(1,3));
91         att_tar_class(3,i) = atan2(DCM_tar(1,2), DCM_tar(1,1));
92     end
93 
```

```
end
```

2.2 Member Contributions

Lab 2 9/28/2023

Name	Plan	Model	Experiment	Results	Report	Code	ACK
Sultan AlBinAli	2	2	2	2	2	2	X
Hayden Gebhardt	2	2	2	2	2	2	X
Greg Kornguth	2	2	2	2	2	2	X