

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

<b>1</b>	<b>Problems</b>	<b>2</b>
1.1	Problem 1 . . . . .	2
1.2	Problem 2 . . . . .	2
1.3	Problem 3 . . . . .	2
1.4	Problem 4 . . . . .	3
1.4.1	Part A . . . . .	3
1.4.2	Part B . . . . .	3
1.4.3	Part C . . . . .	4
1.4.4	Part D . . . . .	5
1.4.5	Part E . . . . .	6
1.4.6	Part F . . . . .	6
<b>2</b>	<b>Appendix</b>	<b>7</b>
2.1	MATLAB Code . . . . .	7
2.1.1	main.m . . . . .	7
2.1.2	LoadASPENData.m . . . . .	11
2.1.3	EulerAngles321.m . . . . .	12
2.1.4	EulerAngles313.m . . . . .	12
2.1.5	RotationMatrix321.m . . . . .	13
2.1.6	RotationMatrix313.m . . . . .	13
2.1.7	ConvertASPENData.m . . . . .	14
2.2	Member Contributions . . . . .	16

# 1 Problems

## 1.1 Problem 1

Description	ASEN 3728 Notation	ASEN 3700 Notation
Inertial position of the vehicle expressed in body coordinates.	$p_B^E$	${}^B P_{B/N}$
Inertial velocity of the vehicle expressed in body coordinates	$V_B^E$	${}^B V_{B/N}$
Inertial position of the target with body frame A	$t_A^E$	${}^N t_{A/N}$
Rotation matrix from body frame to inertial frame	$R_B^E$	$Q_B^N$
Velocity of vehicle relative to target in inertial coordinates	$v_B^t$	${}^N v_{T/N}$
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.

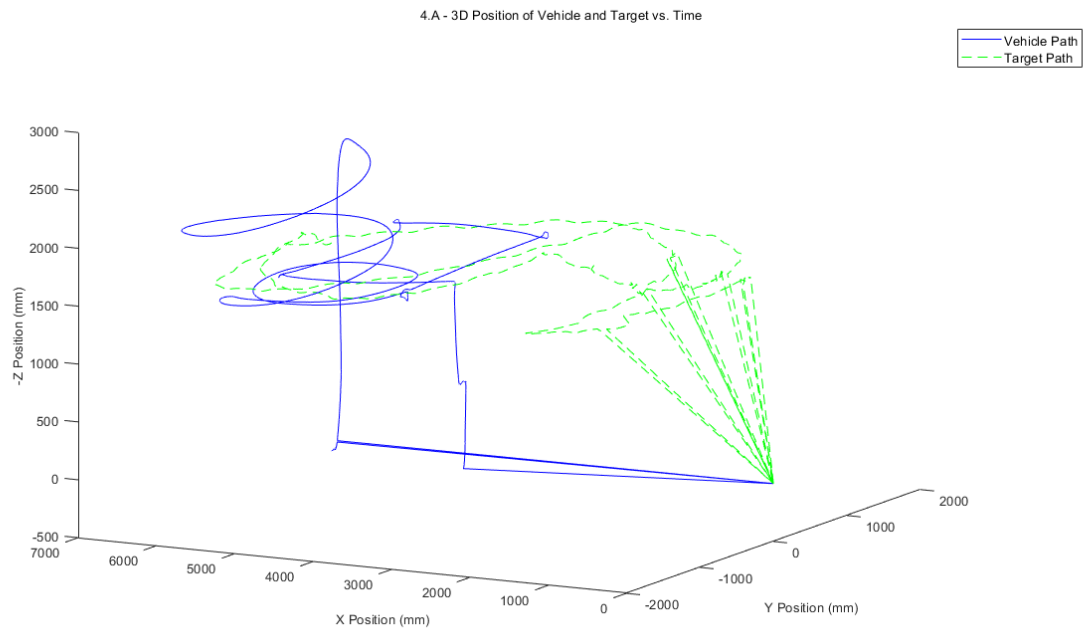
Overall, the motion of both the aerospace vehicle and the target will be characterized by their respective frames of reference (Frame A and Frame T), which will help in understanding their positions and orientations in the lab space.

## 1.3 Problem 3

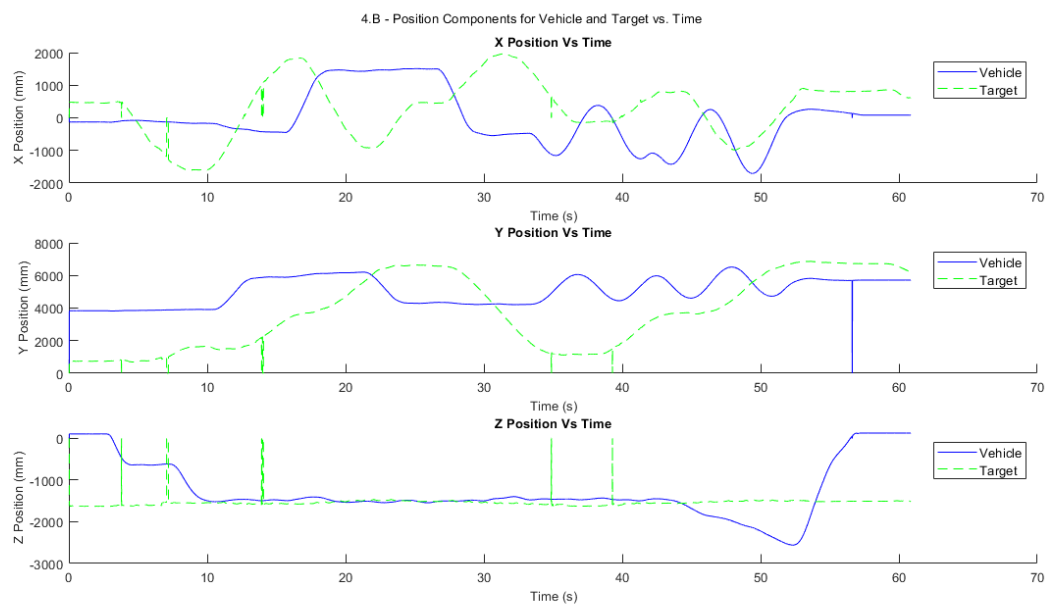
See appendix (2) or attached files for problem 3 functions.

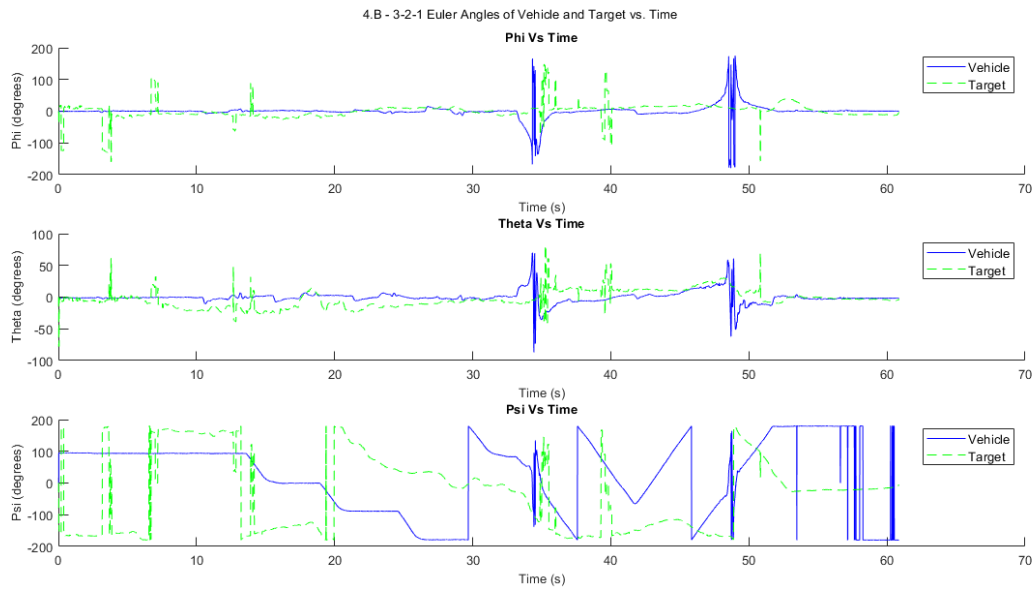
## 1.4 Problem 4

### 1.4.1 Part A

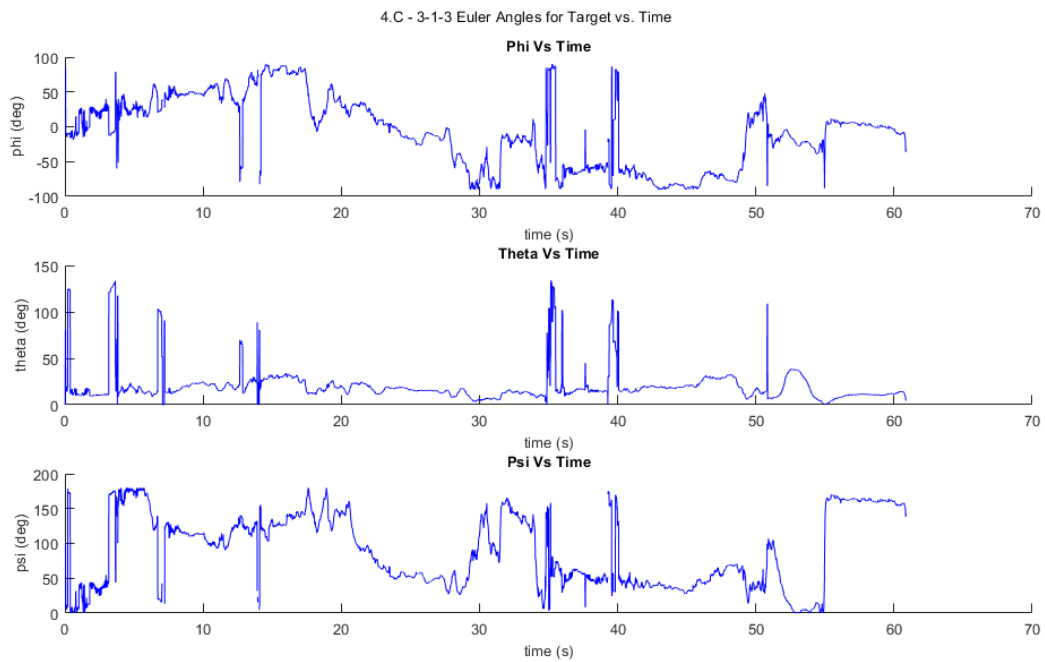


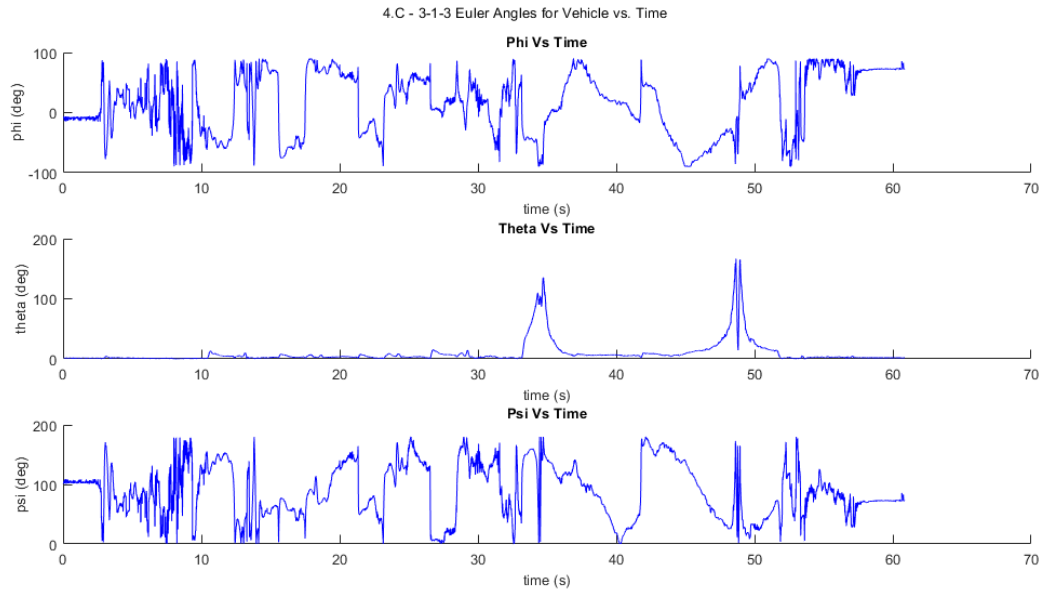
### 1.4.2 Part B



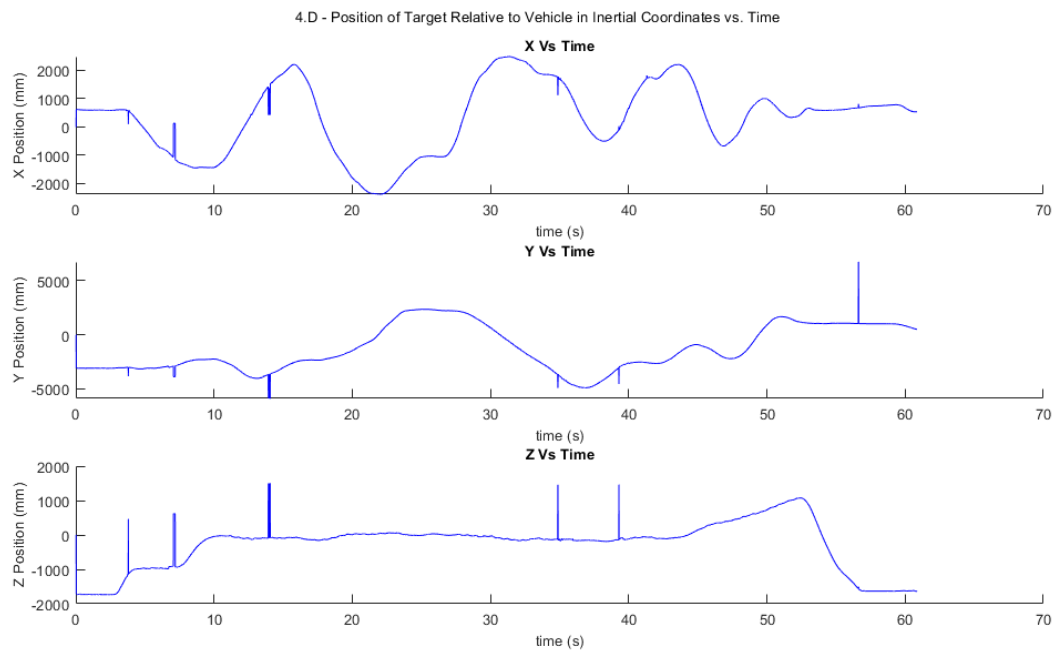


### 1.4.3 Part C

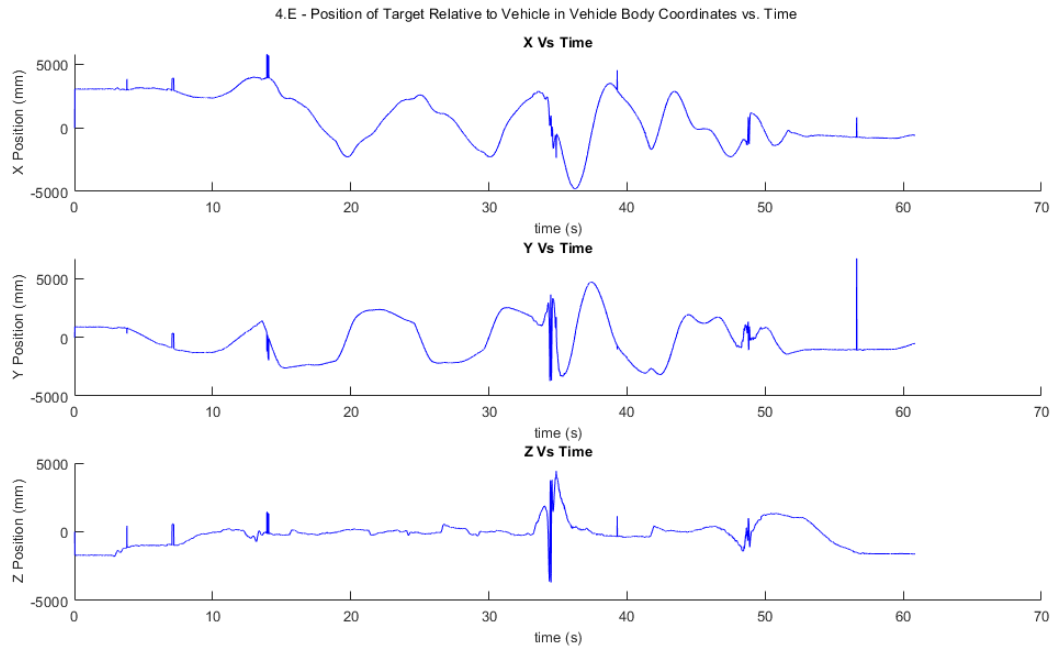




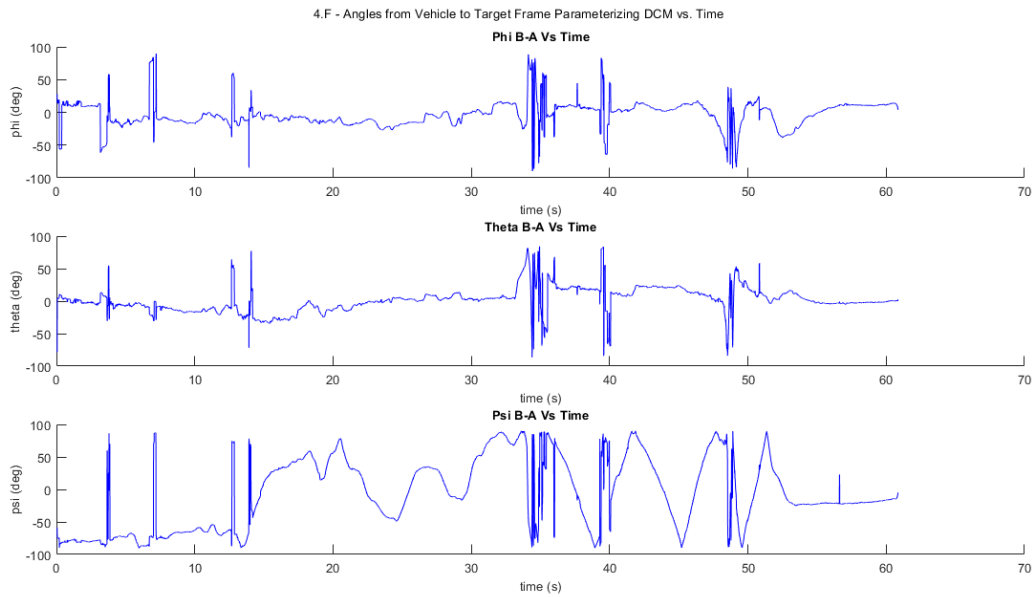
#### 1.4.4 Part D



### 1.4.5 Part E



### 1.4.6 Part F



## 2 Appendix

### 2.1 MATLAB Code

#### 2.1.1 main.m

```
1 % Hayden Gebhardt, Greg Kornguth, Sultan Albinali
2 % ASEN 3728 - Lab 2
3 % Updated: 09/29/2023
4
5 %% House Keeping
6
7 clc; clear; close all;
8
9 %% Main Code
10
11 % part A
12
13 [t_vec, av_pos_inert, av_att, tar_pos_inert, tar_att] = LoadASPENData("3801_Sec1_Test1.csv");
14
15 figure(1)
16 hold on
17 plot3(av_pos_inert(1,:), av_pos_inert(2,:), -av_pos_inert(3,:), "b") % Drone position plot
18 plot3(tar_pos_inert(1,:), tar_pos_inert(2,:), -tar_pos_inert(3,:), "--g") % Target position plot
19 view(3)
20 ylabel("X Position (mm)")
21 xlabel("Y Position (mm)")
22 zlabel("-Z Position (mm)")
23 legend("Vehicle Path", "Target Path")
24 sgtitle("4.A - 3D Position of Vehicle and Target vs. Time")
25 fontsize(figure(1), 10, "points")
26 hold off
27
28 % part B
29
30 figure(2)
31 subplot(3,1,1)
32 hold on
33 plot(t_vec, av_pos_inert(1, :), "b")
34 plot(t_vec, tar_pos_inert(1, :), "--g")
35 title("X Position Vs Time")
36 xlabel("Time (s)")
37 ylabel("X Position (mm)")
38 legend("Vehicle", "Target")
39 hold off
40 subplot(3,1,2)
41 hold on
42 plot(t_vec, av_pos_inert(2, :), "b")
43 plot(t_vec, tar_pos_inert(2, :), "--g")
44 title("Y Position Vs Time")
45 xlabel("Time (s)")
46 ylabel("Y Position (mm)")
47 legend("Vehicle", "Target")
48 hold off
49 subplot(3,1,3)
50 hold on
51 plot(t_vec, av_pos_inert(3, :), "b")
52 plot(t_vec, tar_pos_inert(3, :), "--g")
53 title("Z Position Vs Time")
```



```

54 xlabel("Time (s)")
55 ylabel("Z Position (mm)")
56 legend("Vehicle", "Target")
57
58 hold off
59 sgtitle("4.B - Position Components for Vehicle and Target vs. Time")
60 fontsize(figure(2), 10,"points")
61
62 figure(3)
63 subplot(3,1,1)
64 hold on
65 plot(t_vec, av_att(1,:).*180./pi, "b")
66 plot(t_vec, tar_att(1,:).*180./pi, "--g")
67 title("Phi Vs Time")
68 xlabel("Time (s)")
69 ylabel("Phi (degrees)")
70 legend("Vehicle", "Target")
71 hold off
72 subplot(3,1,2)
73 hold on
74 plot(t_vec, av_att(2,:).*180./pi, "b")
75 plot(t_vec, tar_att(2,:).*180./pi, "--g")
76 title("Theta Vs Time")
77 xlabel("Time (s)")
78 ylabel("Theta (degrees)")
79 legend("Vehicle", "Target")
80 hold off
81 subplot(3,1,3)
82 hold on
83 plot(t_vec, av_att(3,:).*180./pi, "b")
84 plot(t_vec, tar_att(3,:).*180./pi, "--g")
85 title("Psi Vs Time")
86 xlabel("Time (s)")
87 ylabel("Psi (degrees)")
88 legend("Vehicle", "Target")
89 hold off
90 sgtitle("4.B - 3-2-1 Euler Angles of Vehicle and Target vs. Time")
91 fontsize(figure(3), 10,"points")
92
93 % part C
94
95 av_att313 = zeros(3,length(av_att));
96 for i = 1:length(av_att)
97
98     R321_rel = RotationMatrix321(av_att(:,i)); % creating 321 rotation matrix for each vehicle
99     attitude
100     av_att313(:,i) = EulerAngles313(R321_rel); % calculate 313 euler angles from each 321 matrix
101 end
102
103 av_att313 = av_att313(:, 4:end); % pruning NaNs from dataset
104 t_vec2 = t_vec(4:end, :);
105
106 figure(4)
107 subplot(3,1,1)
108 hold on
109 plot(t_vec2, av_att313(1,:).*180./pi, "b")
110 title("Phi Vs Time")
111 xlabel("time (s)")
112 ylabel("phi (deg)")

```

```

112 hold off
113 subplot(3,1,2)
114 hold on
115 plot(t_vec2, av_att313(2,:).*180./pi,"b")
116 title("Theta Vs Time")
117 xlabel("time (s)")
118 ylabel("theta (deg)")
119 hold off
120 subplot(3,1,3)
121 hold on
122 plot(t_vec2, av_att313(3,:).*180./pi,"b")
123 title("Psi Vs Time")
124 xlabel("time (s)")
125 ylabel("psi (deg)")
126 hold off
127 sgttitle("4.C - 3-1-3 Euler Angles for Vehicle vs. Time")
128 fontsize(figure(4), 10,"points")
129
130 tar_att313 = zeros(3,length(tar_att));
131 for i = 1:length(tar_att)
132
133     R321_rel = RotationMatrix321(tar_att(:,i)); % creating 321 rotation matrix for each target
134     attitude
135     tar_att313(:,i) = EulerAngles313(R321_rel); % calculate 313 euler angles from each 321 matrix
136 end
137
138 figure(5)
139 subplot(3,1,1)
140 hold on
141 plot(t_vec, tar_att313(1,:).*180./pi,"b")
142 title("Phi Vs Time")
143 xlabel("time (s)")
144 ylabel("phi (deg)")
145 hold off
146 subplot(3,1,2)
147 hold on
148 plot(t_vec, tar_att313(2,:).*180./pi,"b")
149 title("Theta Vs Time")
150 xlabel("time (s)")
151 ylabel("theta (deg)")
152 hold off
153 subplot(3,1,3)
154 hold on
155 plot(t_vec, tar_att313(3,:).*180./pi,"b")
156 title("Psi Vs Time")
157 xlabel("time (s)")
158 ylabel("psi (deg)")
159 hold off
160 sgttitle("4.C - 3-1-3 Euler Angles for Target vs. Time")
161 fontsize(figure(5), 10,"points")
162
163 % part D
164 tar_rel_av = tar_pos_inert - av_pos_inert; % calculate inertial position of target relative to
165     vehicle
166
167 figure(6)
168 subplot(3,1,1)
169 hold on

```

```

169 plot(t_vec, tar_rel_av(1,:), "b")
170 title("X Vs Time")
171 xlabel("time (s)")
172 ylabel("X Position (mm)")
173 hold off
174 subplot(3,1,2)
175 hold on
176 plot(t_vec, tar_rel_av(2, :), "b")
177 title("Y Vs Time")
178 xlabel("time (s)")
179 ylabel("Y Position (mm)")
180 hold off
181 subplot(3,1,3)
182 hold on
183 plot(t_vec, tar_rel_av(3, :), "b")
184 title("Z Vs Time")
185 xlabel("time (s)")
186 ylabel("Z Position (mm)")
187 hold off
188 sgtitle("4.D - Position of Target Relative to Vehicle in Inertial Coordinates vs. Time")
189 fontsize(fg(6), 10, "points")
190
191 % part E
192
193 tar_rel_av_body = zeros(3, length(av_att));
194 for i = 1:length(av_att)
195
196     R321_rel = RotationMatrix321(av_att(:, i)); % creating 321 rotation matrix for each vehicle
197     attitude
198
199     % multiply by DCM to get from inertial position of target relative
200     % to vehicle to body coordinates
201     tar_rel_av_body(:, i) = R321_rel \ tar_rel_av(:, i);
202 end
203
204 figure(7)
205 subplot(3,1,1)
206 hold on
207 plot(t_vec, tar_rel_av_body(1,:), "b")
208 title("X Vs Time")
209 xlabel("time (s)")
210 ylabel("X Position (mm)")
211 hold off
212 subplot(3,1,2)
213 hold on
214 plot(t_vec, tar_rel_av_body(2, :), "b")
215 title("Y Vs Time")
216 xlabel("time (s)")
217 ylabel("Y Position (mm)")
218 hold off
219 subplot(3,1,3)
220 hold on
221 plot(t_vec, tar_rel_av_body(3, :), "b")
222 title("Z Vs Time")
223 xlabel("time (s)")
224 ylabel("Z Position (mm)")
225 hold off
226 sgtitle("4.E - Position of Target Relative to Vehicle in Vehicle Body Coordinates vs. Time")
227 fontsize(fg(7), 10, "points")

```

```

227
228 % part F
229
230 av_to_tar = zeros(3,length(av_att));
231 for i = 1:length(av_att)
232
233     R321_BtoN = RotationMatrix321(av_att(:, i)); % calculate 321 rotation matrix from vehicle
                frame B to intermediate N frame
234     R321_NtoA = inv(RotationMatrix321(tar_att(:, i))); % calculate 321 rotation matrix from N to
                target frame A
235
236     R321_BtoA = R321_NtoA * R321_BtoN; % calculate full DCM from vehicle frame to target frame
237
238     EulerAngles321_BtoA(:, i) = EulerAngles321(R321_BtoA); % calculate resulting 321 euler angles
                for each data point
239 end
240
241 EulerAngles321_BtoA = EulerAngles321_BtoA(:, 4:end); % pruning NaNs from dataset
242
243 figure(8)
244 subplot(3,1,1)
245 hold on
246 plot(t_vec2, EulerAngles321_BtoA(1, :).*180./pi,"b")
247 title("Phi B-A Vs Time")
248 xlabel("time (s)")
249 ylabel("phi (deg)")
250 hold off
251 subplot(3,1,2)
252 hold on
253 plot(t_vec2, EulerAngles321_BtoA(2, :).*180./pi,"b")
254 title("Theta B-A Vs Time")
255 xlabel("time (s)")
256 ylabel("theta (deg)")
257 hold off
258 subplot(3,1,3)
259 hold on
260 plot(t_vec2, EulerAngles321_BtoA(3, :).*180./pi,"b")
261 title("Psi B-A Vs Time")
262 xlabel("time (s)")
263 ylabel("psi (deg)")
264 hold off
265 sgtitle("4.F - Angles from Vehicle to Target Frame Parameterizing DCM vs. Time")
266 fontsize(figure(8), 10,"points")

```

### 2.1.2 LoadASPENData.m

```

1 function [t_vec, av_pos_inert, av_att, tar_pos_inert, tar_att] = LoadASPENData(filename)
2
3 % ----- %
4 % Inputs:      filename = name of data file for processing
5 %
6 % Outputs:      t_vec = time vector scaled to data refresh rate (s)
7 %              av_pos_inert = inertial position of vehicle (mm)
8 %              av_att = attitude angles of vehicle (rad)
9 %              tar_pos_inert = inertial position of target (mm)
10 %             tar_att = attitude angles of target (rad)
11 %
12 % Methodology: Read in data file and parse columns for each output listed

```

```

13 %         above. Data file used for this lab is 100 Hz, but change the
14 %         denominator in t_vec to the refresh rate of whatever data is
15 %         being used. Raw data is then passed into ConvertASPENData so
16 %         that the correct outputs can be produced, and NaNs can be
17 %         removed.
18 % ----- %
19
20 data = readmatrix(filename);
21 pos_av_aspen = data(:, 12:14)';
22 att_av_aspen = data(:, 9:11)';
23 pos_tar_aspen = data(:, 6:8)';
24 att_tar_aspen = data(:, 3:5)';
25 t_vec = 0:1:length(data) - 1;
26 t_vec = (t_vec/100)';
27 [av_pos_inert, av_att, tar_pos_inert, tar_att] = ConvertASPENData(pos_av_aspen, att_av_aspen,
28     pos_tar_aspen, att_tar_aspen);
29 end

```

### 2.1.3 EulerAngles321.m

```

1 function attitude = EulerAngles321(RotMat)
2
3 % ----- %
4 % Inputs:   RotMat = 3x3 input rotation matrix
5 %
6 % Outputs: attitude = attitude representation for one data frame
7 %             = [phi, theta, psi] (rad)
8 %
9 % Methodology: Calculate the 3-2-1 euler angles for corresponding rotation
10 %             matrix.
11 % ----- %
12
13 theta = -asin(RotMat(1, 3));
14
15 psi = asin(RotMat(1, 2) / cos(theta));
16
17 phi = asin(RotMat(2, 3) / cos(theta));
18
19 attitude = [phi, theta, psi]';
20
21 end

```

### 2.1.4 EulerAngles313.m

```

1 function attitude = EulerAngles313(RotMat)
2
3 % ----- %
4 % Inputs:   RotMat = 3x3 input rotation matrix
5 %
6 % Outputs: attitude = attitude representation for one data frame
7 %             = [omega, i, Omega] (rad)
8 %
9 % Methodology: Calculate the 3-1-3 euler angles for corresponding rotation
10 %             matrix.
11 % ----- %
12
13 i = acos(RotMat(3, 3));

```

```

14
15     Omega = acos(RotMat(2, 3) / sin(i));
16
17     omega = asin(RotMat(3, 1) / sin(i));
18
19     attitude = [omega, i, Omega]';
20
21 end

```

### 2.1.5 RotationMatrix321.m

```

1 function R321 = RotationMatrix321(attitude)
2
3 % ----- %
4 % Inputs: attitude = attitude representation for one data frame
5 %           = [phi, theta, psi] (rad)
6 %
7 % Outputs:   R321 = 3-2-1 rotation matrix comprised of three angle DCMs
8 %
9 % Methodology: Calculate the 3-2-1 rotation matrix for corresponding euler
10 %              angles.
11 % ----- %
12
13     phi = attitude(1);
14     theta = attitude(2);
15     psi = attitude(3);
16
17     R1 = [1, 0, 0;
18           0, cos(phi), sin(phi);
19           0, -sin(phi), cos(phi)];
20
21     R2 = [cos(theta), 0, -sin(theta);
22           0, 1, 0;
23           sin(theta), 0, cos(theta)];
24
25     R3 = [cos(psi), sin(psi), 0;
26           -sin(psi), cos(psi), 0;
27           0, 0, 1];
28
29     R321 = R1 * R2 * R3;
30
31 end

```

### 2.1.6 RotationMatrix313.m

```

1 function R313 = RotationMatrix313(attitude)
2
3 % ----- %
4 % Inputs: attitude = attitude representation for one data frame
5 %           = [omega, i, Omega] (rad)
6 %
7 % Outputs:   R313 = 3-1-3 rotation matrix comprised of three angle DCMs
8 %
9 % Methodology: Calculate the 3-1-3 rotation matrix for corresponding euler
10 %              angles.
11 % ----- %
12
13     omega = attitude(1);

```

```

14     i = attitude(2);
15     Omega = attitude(3);
16
17     R1 = [1, 0, 0;
18           0, cos(i), sin(i);
19           0, -sin(i), cos(i)];
20
21     R3_phi = [cos(omega), sin(omega), 0;
22              -sin(omega), cos(omega), 0;
23              0, 0, 1];
24
25     R3_psi = [cos(Omega), sin(Omega), 0;
26              -sin(Omega), cos(Omega), 0;
27              0, 0, 1];
28
29     R313 = R3_psi * R1 * R3_phi;
30
31 end

```

### 2.1.7 ConvertASPENData.m

```

1 % Eric W. Frew
2 % ASEN 3801
3 % ConvertASPENData
4 % Created: 9/21/23
5
6 function [pos_av_class, att_av_class, pos_tar_class, att_tar_class] =
7     ConvertASPENData(pos_av_aspen, att_av_aspen, pos_tar_aspen, att_tar_aspen)
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %
10 % Inputs: pos_av_aspen = 3 x n array of position vectors in coordinate
11 %           system of the ASPEN motion capture system
12 %           att_av_aspen = 3 x n array of attitude vectors using the 'helical angles'
13 %           provided by the ASPEN motion capture system
14 %           pos_tar_aspen = 3 x n array of position vectors in coordinate
15 %           system of the ASPEN motion capture system
16 %           att_tar_aspen = 3 x n array of attitude vectors using the 'helical angles'
17 %           provided by the ASPEN motion capture system
18 %
19 % Outputs: pos_av_class, att_av_class, pos_tar_class, att_tar_class
20 %
21 %           pos_av_class = 3 x n array of position vectors in 'class' coordinate
22 %           system with same x-axis as ASPEN and y-axis and
23 %           z-axis flipped so z is down
24 %           att_av_class = 3 x n array of attitude vectors using NASA Standard
25 %           321 Euler angles
26 %           pos_tar_class = 3 x n array of position vectors in 'class' coordinate
27 %           system with same x-axis as ASPEN and y-axis and
28 %           z-axis flipped so z is down
29 %           att_tar_class = 3 x n array of attitude vectors using NASA Standard
30 %           321 Euler angles
31 %
32 %
33 %
34 % Methodology: Converts the data provided by the ASPEN lab motion capture system into
35 % the convention used for ASEN 3801:
36 % 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 converted 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 =
71             makehgtform('xrotate',att_av_aspen(1,i),'yrotate',att_av_aspen(2,i),'zrotate',att_av_aspen(3,i));
72         DCM_av = P*xfm_av(1:3,1:3)*P;
73         att_av_class(1,i) = atan2(DCM_av(2,3), DCM_av(3,3));
74         att_av_class(2,i) = -asin(DCM_av(1,3));
75         att_av_class(3,i) = atan2(DCM_av(1,2), DCM_av(1,1));
76     end
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 =
88             makehgtform('xrotate',att_tar_aspen(1,i),'yrotate',att_tar_aspen(2,i),'zrotate',att_tar_aspen(3,i));
89         DCM_tar = P*xfm_tar(1:3,1:3)*P;
90         att_tar_class(1,i) = atan2(DCM_tar(2,3), DCM_tar(3,3));
91         att_tar_class(2,i) = -asin(DCM_tar(1,3));
92         att_tar_class(3,i) = atan2(DCM_tar(1,2), DCM_tar(1,1));
93     end

```



94 || [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