ASEN 3801-002 Lab 02

Group: #38
Names: Jackson Clark, Robert Kaplan, Eric Meyer, Jared Steffen
Date Modified: 09/24/2023

Problems:

1. Appendix A Table:

Description	ASEN 3728 Notation	ASEN 3700 Notation
The inertial position of the vehicle expressed in body coordinates	p_B^E	${}^{I}p_{B,\text{vehicle}}$
The inertial velocity of the vehicle expressed in body coordinates	v_B^E	${}^{I}v_{B,\text{vehicle}}$
Time vector aligned with aircraft A written and relative to Inertial Frame	t_A^E	${}^{I}t_{IA}$
Rotation matrix from body coordinate frame to inertial coordinate frame	R_B^E	Q_B^I
The velocity of the aerospace vehicle relative to the target, expressed in inertial coordinates	v_E^B	${}^{B}v_I$
Rotation matrix from coordinate frame A to coordinate frame B	R_A^B	Q_A^B

2. Qualitative description of ASPEN-Test flight
   a. Notation:
      i. T = Target
      ii. AV = Aerospace Vehicle
      iii. +i = X-axis (inertial)
      iv. +j = Y-axis (inertial)
      v. +k = Z-axis (inertial) = Down (Up = -k)
   b. Description of the target's (T) path:
      i. The target begins a couple meters away from the AV in the -j direction. The k position stays constant (~ -1.5 m) for the whole duration until the end, because the height of the target remains the same.

ii. The target moves in a rough rectangle while maintaining a constant distance from the AV’s starting point. The target begins with going in the +i direction, turning 90 degrees in the -k direction, until reaching the same starting position again.
iii. The target then moves in a more circular direction, in the same rotation (-k direction). This time, the target is a little closer to the AV’s starting point. This movement is done for a 1/2 circle.
iv. The target then moves in the -i direction, until reaching the same corner of the rectangular motion from ii. (the corner in the -i and +j direction)
v. Then the target travels in the +i / -j direction, until reaching the opposite corner of the same rectangle. There is a brief hesitation in the beginning of this movement.
vi. The target travels in the +j direction, and then turns 90 degrees in the -k direction. Then, it travels in the -i direction until it reaches the AV position.
vii. Finally, the target travels in the +k direction to pick up the AV.

c. Description of the aerospace vehicle's (AV) path:
   i. The AV starts on the ground and raises up to a point above the T.
   ii. Then it makes a square by going forward a certain distance, stopping, turning 90 degrees in the +k direction, and repeating until it is back where it started when it was at the point above the T.
   iii. Then starts moving in circles in the -k direction (meaning that it moves forward at a constant speed while turning at a constant rate). It starts by traveling in the -i direction. It travels 1 & 1/3rd of a circle, and then (without stopping) moves in the other direction.
   iv. It is now making a spiral in the +k direction and ascending direction (meaning that it is now making a circle just like before except it is moving the other direction and is traveling upwards at a constant speed). It travels 1 & 2/3rds spirals before stopping.
   v. It then descends (moves in the +k direction) until it reaches the ground)

3. MATLAB Functions Presented in Appendices A-E

4. Parts a-f listed below:
    a. Figure 1 shows the trajectories of both the target and the vehicle in the ASPEN Lab. The occasional return to the origin is due to erroneous data being replaced with a value of zero.

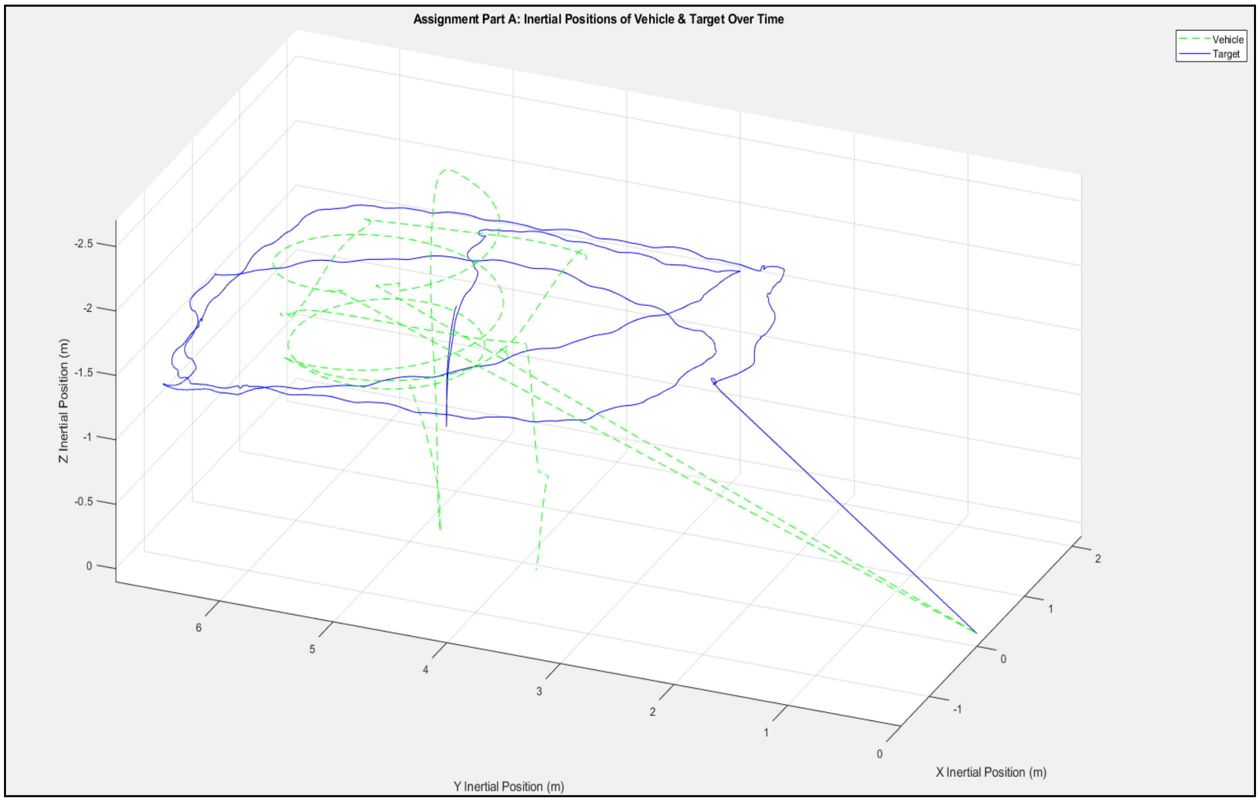

Figure 1

b. Figure 2 shows the X, Y, and Z components of both the target and vehicle over time during the ASPEN Lab demonstration. Figure 3 shows the 3-2-1 Euler Angles in degrees over time of both the target and the vehicle. These values were found through the RotationMatrix321 function in MATLAB.

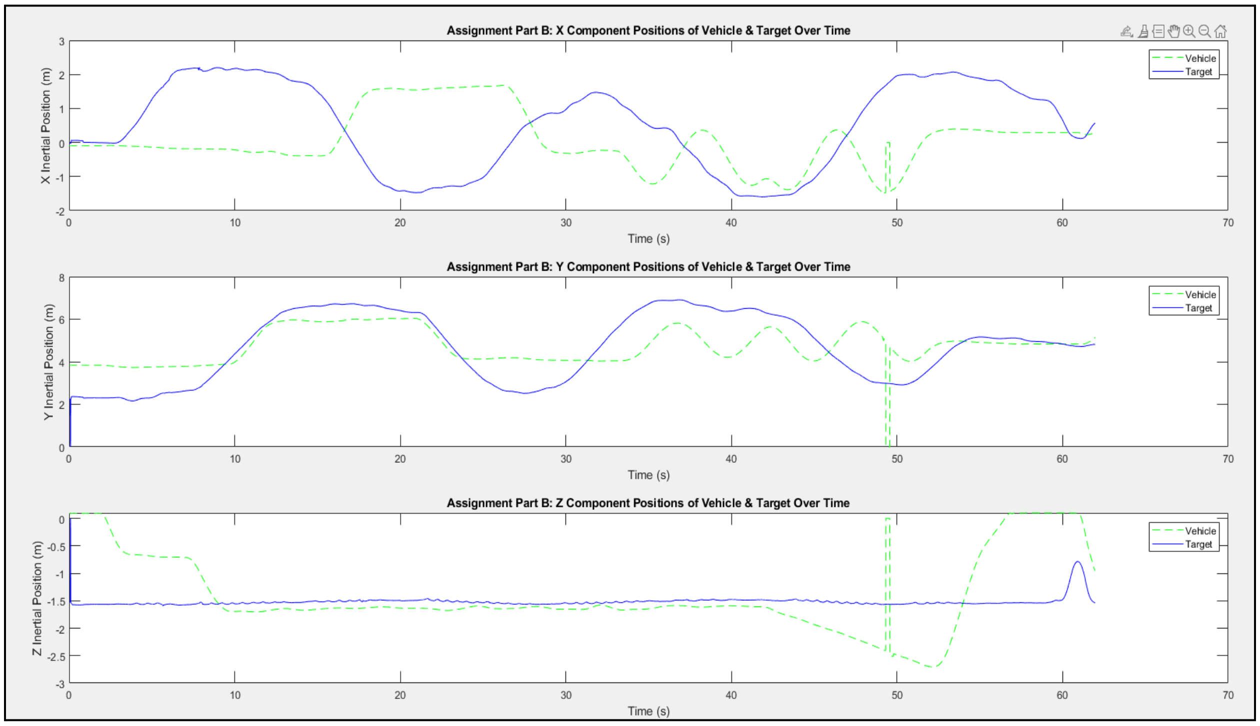

Figure 2

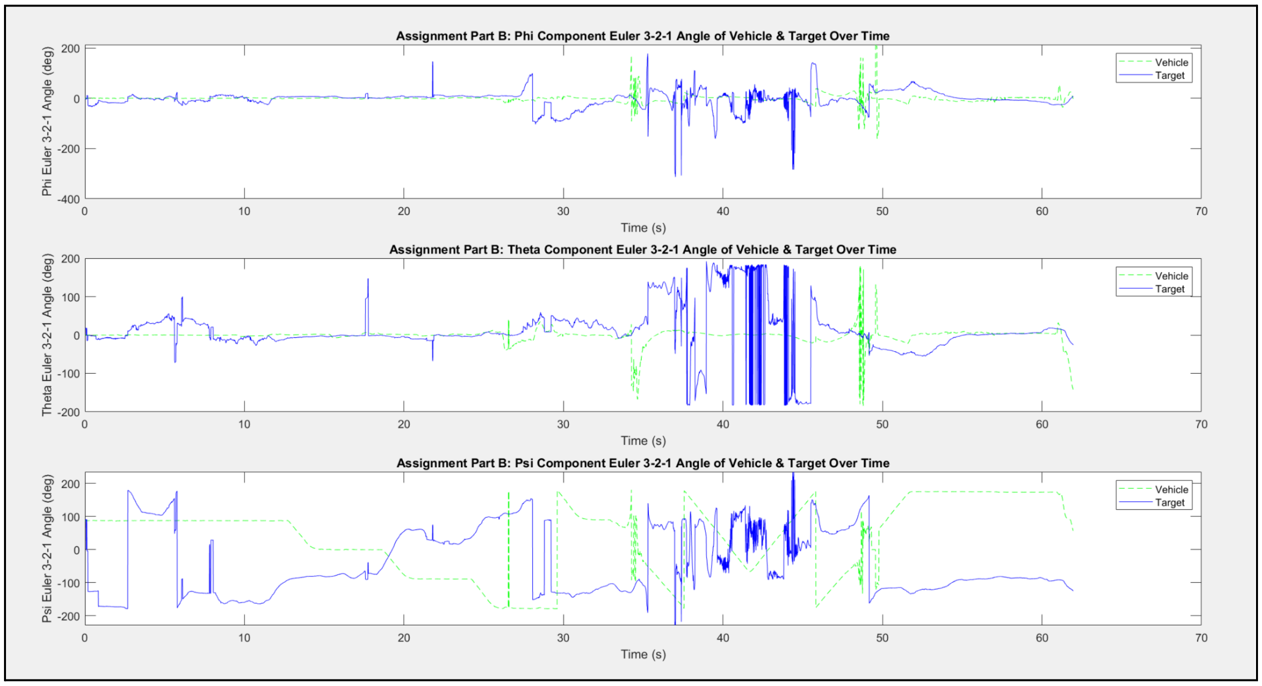

Figure 3

c. Figure 4 shows the 3-1-3 Euler Angles in degrees over time of both the target and the vehicle. These values were found through the RotationMatrix313 function in MATLAB.

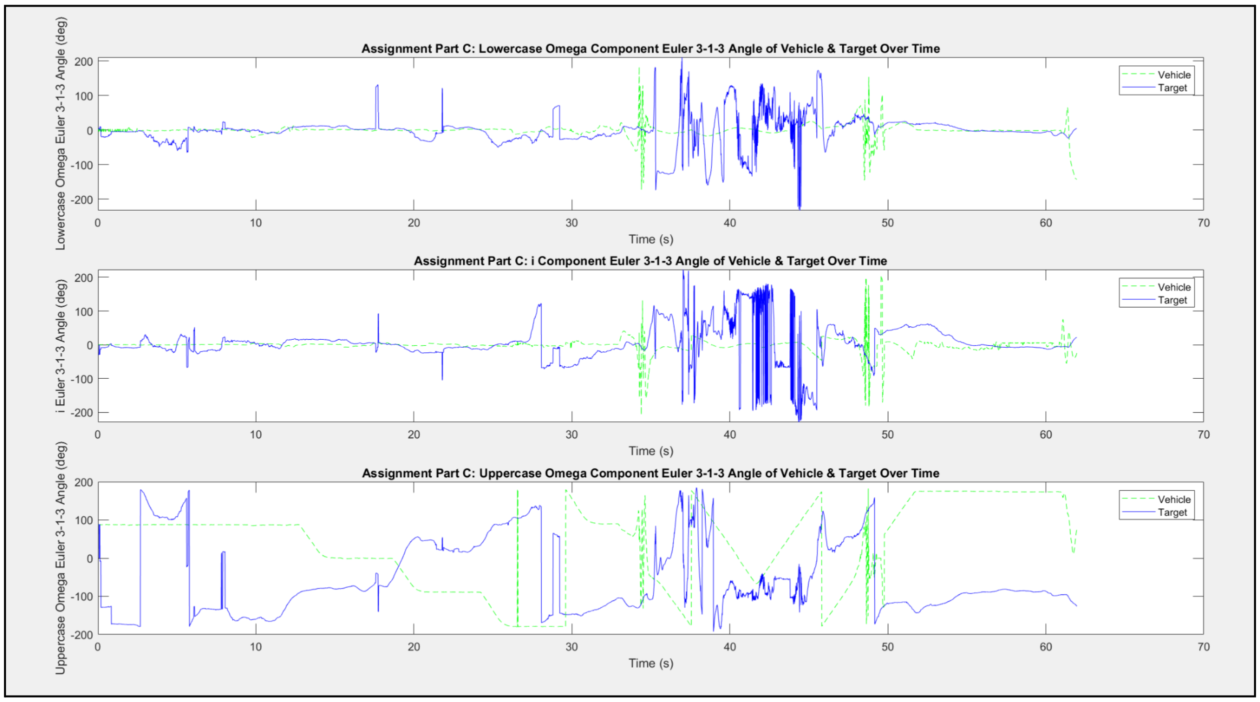

Figure 4

d. Figure 5 shows the X, Y, and Z components of the position of the target relative to the vehicle over time in inertial coordinates. The relative position vectors were found by subtracting the vehicle's position vector from the target's position vector.

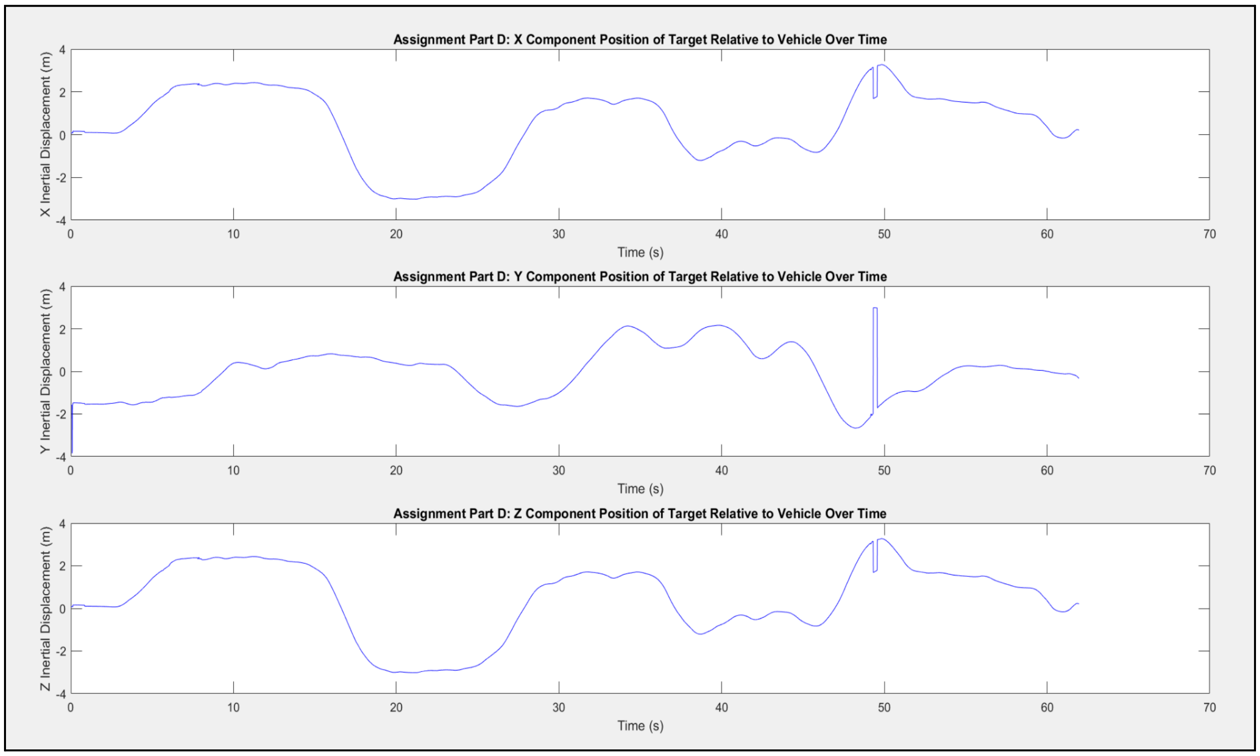

Figure 5

e. Figure 6 shows the X, Y, and Z components of the position of the target relative to the vehicle over time in body coordinates. The relative position vector from Figure 5 was put through the RotationMatrix321 function in MATLAB to rotate it from inertial to body coordinates.

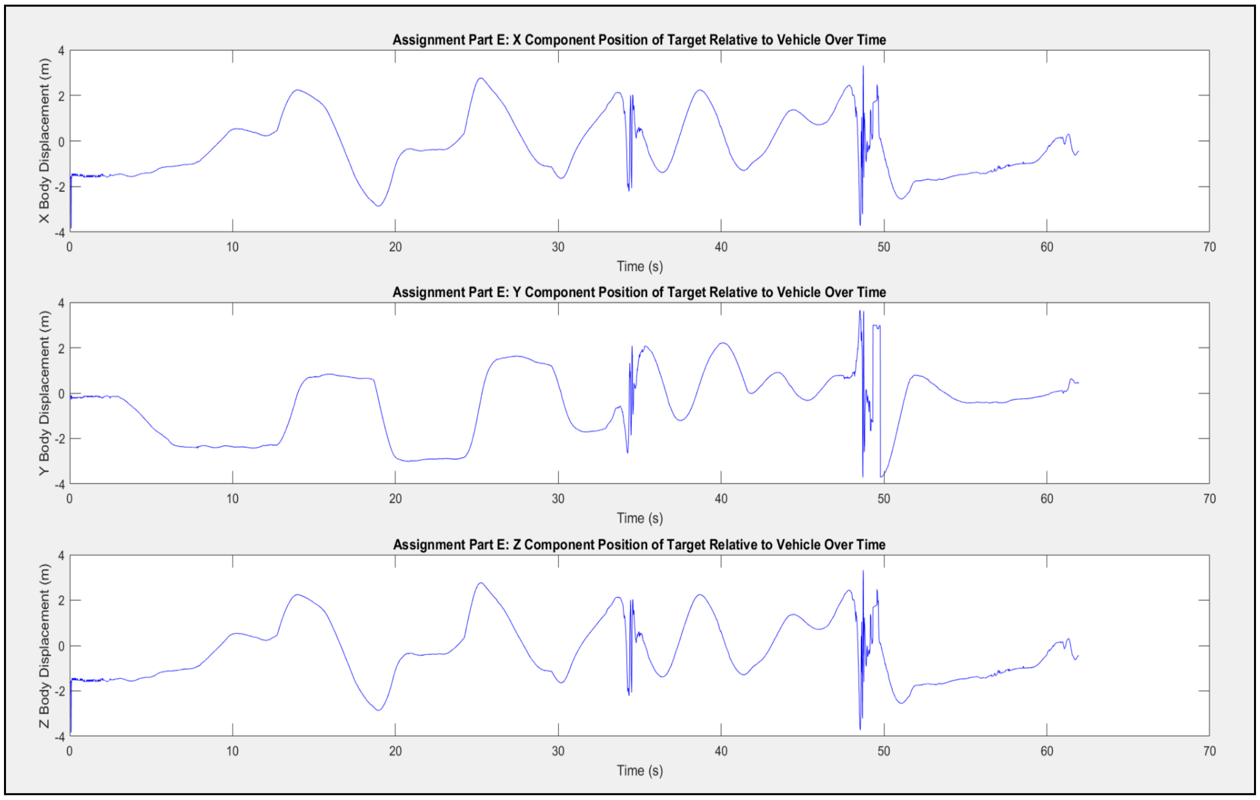

Figure 6

f. Figure 7 shows the 3-2-1 Euler Angles of the vehicle relative to the target over time. These angles were found by putting both the target and the vehicle's position vectors into the RotationMatrix321 function in MATLAB, then multiplying the target's rotation matrix by the vehicle's rotation matrix transpose. The new rotation matrix was then run through the EulerAngles321 function in MATLAB.

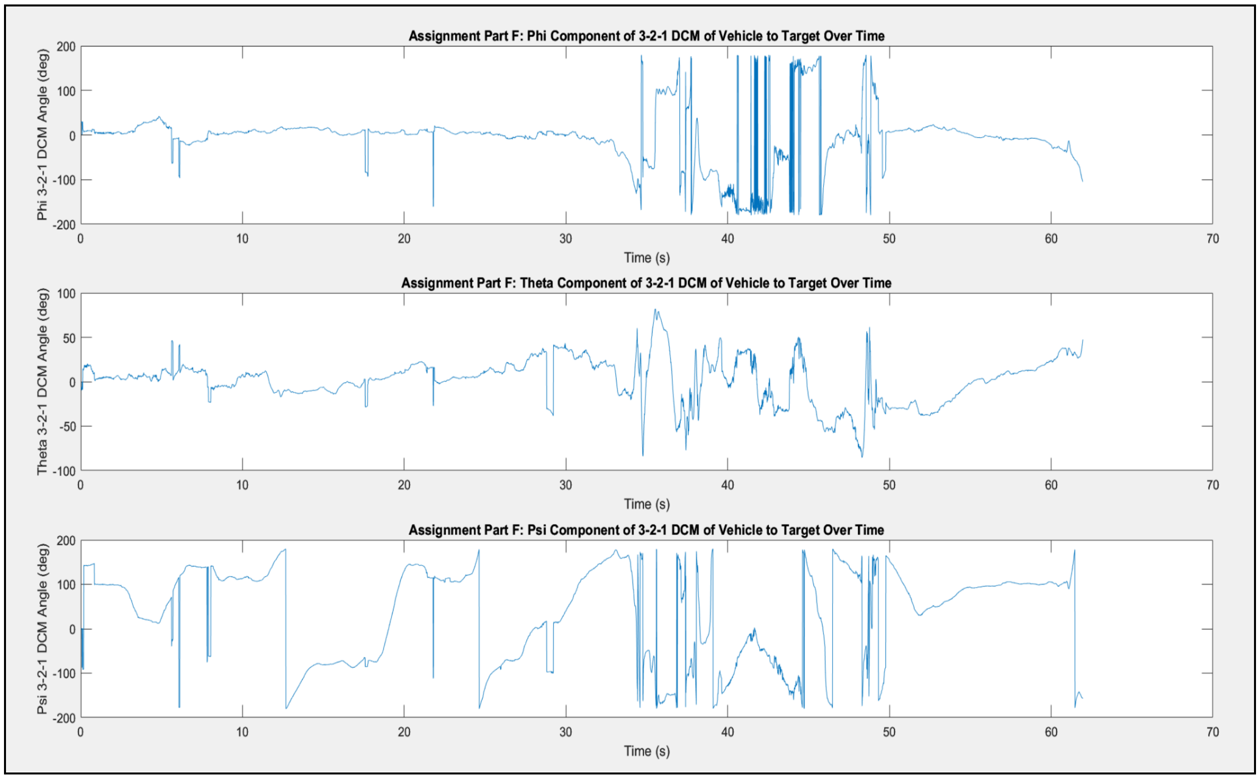

Figure 7

Appendix A
LoadASPENData Function

```
function [t_vec, av_pos_inert, av_att, tar_pos_inert, tar_att] =
 LoadASPENData(filename)
%   LOADASPENDATA | The purpose of this function is to organize/categorize
%   the input data given in the input file to individualized object
%   position/attitude vectors
%   Inputs: A .csv file containing one column for each category with
%   subsequent rows below each indicating values for an incrementing snapshot
 in time
%   Outputs: Multiple vectors/matrices formatted such that the physical
%   values for each snapshot in time are per columm and each component of
%   position/angle values is per row. Ex. Time has only one component,
%   incrementing through every snapshot in time, therefore its size is:
%   1 x length(time snapshots)

    data = readmatrix(filename); % store the excel-like file data in a matrix

    FR = 100; % Frame Rate Frequency [Hz]
    t_vec = data(:, 1)'./FR; % Seconds = Frames/Hz [Seconds]

    av_pos_inert = [data(:, 12)'; data(:, 13)'; data(:, 14)'] * 10^-3; % Data
 formating into new vector + unit converison [mm -> m]
    av_att = [data(:, 9)'; data(:, 10)'; data(:, 11)']; % Data formating into
 new vector

    tar_pos_inert = [data(:, 6)'; data(:, 7)'; data(:, 8)'] * 10^-3; % Data
 formating into new vector + unit converison [mm -> m]
    tar_att = [data(:, 3)'; data(:, 4)'; data(:, 5)']; % Data formating into
 new vector

    % Pass the formatted data into a conversion function that substitutes
    % NaN values to 0 as well as the measured angles into more
    % lecture-conventional angles that will be applicable to formulas. Then
    % return said values.
    [av_pos_inert, av_att, tar_pos_inert, tar_att] =
 ConvertASPENData(av_pos_inert, av_att, tar_pos_inert, tar_att);
end
```

Published with MATLAB® R2022b

Appendix B
RotationMatrix321 Function

```
function [R321] = RotationMatrix321(attitude)
%   ROTATIONMATRIX321 | This function is used to determine the euler 3-2-1
 rotation matrix
%   given a set of inputted euler angles
%   Inputs = 3 x 1 vector of phi-theta-psi angles
%   Output = Euler 3-2-1 Rotation Matrix

    phi = attitude(1); % Euler Angle phi
    theta = attitude(2); % Euler Angle theta
    psi = attitude(3); % Euler Angle psi

    % [Inertial to Body Frame] Plugging into formula
    R321 = [ cosd(phi) .* cosd(psi), sind(phi) .* sind(theta) .* cosd(psi) -
 cosd(phi) .* sind(psi), cosd(phi) .* sind(theta) .* cosd(psi) + sind(phi) .*
 sind(psi);
             cosd(theta) .* sind(psi), sind(phi) .* sind(theta) .* sind(psi) +
 cosd(phi) .* cosd(psi), cosd(phi) .* sind(theta) .* sind(psi) - sind(phi) .*
 cosd(psi);
             -sind(theta), sind(phi) .* cosd(theta), cosd(phi) .*
 cosd(theta)]';

end
```

Published with MATLAB® R2022b

Appendix C
RotationMatrix313 Function

```
function [R313] = RotationMatrix313(attitude)
%   ROTATIONMATRIX313 | This function is used to determine the euler 3-1-3
 rotation matrix
%   given a set of inputted euler angles
%   Inputs = 3 x 1 vector of LowercaseOmega-i-UppercaseOmega angles
%   Output = Euler 3-1-3 Rotation Matrix

    LMGA = attitude(1); % Euler Angle Lowercase Omega
    i = attitude(2); % Euler Angle i
    UMGA = attitude(3); % Euler Angle Uppercase Omega

    % [Inertial to Body Frame] Plugging into formula
    R313 = [ cosd(UMGA) .* cosd(LMGA) - sind(UMGA) .* sind(LMGA) .* cosd(i),
 -sind(UMGA) .* cosd(LMGA) - cosd(UMGA) .* sind(LMGA) .* cosd(i), sind(i) .*
 sind(LMGA);
             cosd(UMGA) .* sind(LMGA) + sind(UMGA) .* cosd(i) .* cosd(LMGA),
 -sind(UMGA) .* sind(LMGA) + cosd(UMGA) .* cosd(i) .* cosd(LMGA), -sind(i) .*
 cosd(LMGA);
             sind(UMGA) .* sind(i), cosd(UMGA) .* sind(i), cosd(i)]';

end
```

Published with MATLAB® R2022b

Appendix D
EulerAngles321 Function

```
function [attitude] = EulerAngles321(RotMat)
%   EULERANGLES321 | This function is used to extract the euler rotation
%   angles present in a 3-2-1 euler rotation matrix
%   Input: Euler 3-2-1 Rotation Matrix
%   Output: 3 x 1 vector of phi-theta-psi euler angles

    % Calulating psi, theta, and phi based on the Euler 3-2-1 formula by
 manipulating together certain position values in the inputted rotation matrix

    psi = atan2d(RotMat(1,2),RotMat(1,1));
    theta = asind(RotMat(1,3));
    phi = atan2d(RotMat(2,3),RotMat(3,3));

    attitude = [phi; theta; psi]; % Output a vector of the 3-2-1 euler angles

end
```

Published with MATLAB® R2022b

Appendix E
EulerAngles313 Function

```
function [attitude] = EulerAngles313(RotMat)
%   EULERANGLES321 | This function is used to extract the euler rotation
%   angles present in a 3-2-1 euler rotation matrix
%   Input: Euler 3-2-1 Rotation Matrix
%   Output: 3 x 1 vector of phi-theta-psi euler angles

    % Calulating psi, theta, and phi based on the Euler 3-1-3 formula by
 manipulating together certain position values in the inputted rotation matrix

    LMGA = atan2d(RotMat(1,3),RotMat(2,3));
    i = acosd(RotMat(3,3));
    UMGA = atan2d(RotMat(3,1),-RotMat(3,2));

    attitude = [LMGA; i; UMGA]; % Output a vector of the 3-1-3 euler angles

end
```

Published with MATLAB® R2022b

Appendix F
Team Participation Table

-	**Plan**	**Model**	**Experiment**	**Results**	**Report**	**Code**	**ACK**
Name	**-**	**-**	**-**	**-**	**-**	**-**	**-**
Jackson Clark	1	1	1	1	1	2	X
Jared Steffen	1	1	1	1	2	1	X
Eric Meyer	1	1	2	1	1	1	X
Robert Kaplan	1	1	1	2	1	1	X