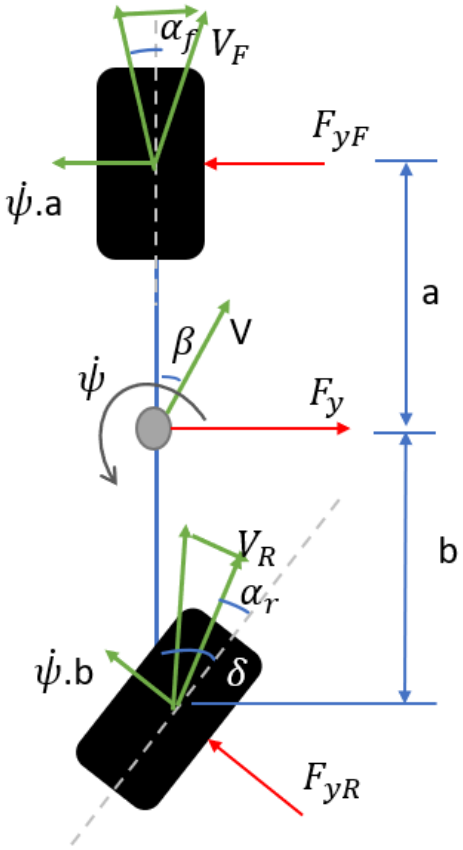
AuE-ME-4600/6600 (Fall 2022) – Graduate Student Project

Akshay Balachandran ; Rakshat Balu ; Rohit Ravikumar

PART I – MODEL DEVELOPMENT

Here, we will derive a rear-steered vehicle model with planar dynamics (longitudinal, lateral, and yaw motion). The derivation of this model follows the same steps discussed in class for the Bicycle Model but includes driving/braking force. Using the provided vehicle parameters and the reference model in Fig. (2), answer the following questions.

1. Assuming an adapted ISO coordinate system, provide the free body diagram of the combined system with appropriate sign conventions.



2. Derive the equations of motion (EOM) for the rear steered vehicle utilizing generic force terms (i.e., Fy, Fx, Fyf, etc.). Then, utilize the corresponding expressions for the forces.

$$F_{xB} = F_{aF} + F_{xR}cos\delta + F_{yR}sin\delta - F_y sin\beta$$
$$F_{yB} = F_y + F_{yR}cos\delta - F_{xR}sin\delta - F_y cos\beta$$
$$F_y sin\beta = F_{xf} + F_{xR}cos\delta + F_{yR}sin\delta$$
$$F_y cos\beta = F_{yf} + F_{yR}cos\delta - F_{xR}sin\delta$$
$$\Sigma Mcg = F_{yf}.a - F_{yR}cos\delta.b + F_{xR}sin\delta b$$
$$I_z\ddot{\psi} = F_{yf}.a - F_{yR}cos\delta.b + F_{xR}sin\delta b$$

3. Rewrite the EOMs in state-space representation. You may apply small angle approximations. What are the eigenvalues of the system? Provide a plot of eigenvalues vs CG position.

$$F_{xF} + F_{xR} + F_{xR}\delta = \beta F_y$$

$$F_{yF} + F_{yR} - F_{xR}\delta = F_y$$

Simplified lateral dynamics

$$\Sigma F = F_{yF} - F_y + F_{yR} = 0$$
$$\Sigma M = = -F_{yR}b + F_{yF}a$$
$$\Sigma F_x = F_{xF} + F_{xR} - \beta F_y = 0$$
$$F_{yR} = F_{yF}cos\delta - F_y cos\beta = 0$$
$$F_{yR} + F_{yF} = F_y$$
$$F_{yf} = C_f a_F$$
$$F_{yR} = C_r a_R$$

Eigen values of A matrix is: $\begin{bmatrix} -0.5636 & +3.9839i \\ -0.5636 & -3.9839i \end{bmatrix}$

4. Determine the steady-state characteristics (steering to instantaneous radius relationship as well as the vehicle slip angle relationship) a of rear steered vehicle through geometric analysis.

$$C_f a_F + C_r a_R - mv(-\dot{\beta} + \dot{\psi}) = 0$$

$$Iz\dot{\psi} - (C_f a_F)a + (C_r a_R)b = 0$$
$$C_f\left[\frac{\dot{\psi}a}{v} - \beta\right] + C_r\left[\delta + \frac{\dot{\psi}b}{v} - \beta\right] - mv[-\dot{\beta} + \dot{\psi}] = 0$$

$$I_z\ddot{\psi} - C_f\left[\frac{\dot{\psi}a}{v} - \beta\right]a + C_r\left[\delta + \frac{\dot{\psi}b}{v} - \beta\right]b = 0$$

Put in the standard form with control input δ on 1 becomes:

$$C_r\delta = mv[-\dot{\beta} + \dot{\psi}] + C_f\left[\frac{\dot{\psi}a}{v} + \beta\right] + C_R\left[-\frac{\dot{\psi}a}{v} + \beta\right]$$
$$\dot{\beta}(mv) = \dot{\beta}(C_f + C_R) - C_r\delta + \dot{\psi}\left[mv + \frac{aC_f}{v} - \frac{bC_R}{v}\right]$$
$$\dot{\beta} = -\delta\left(\frac{C_R}{mv}\right) + \beta\left(\frac{C_f + C_R}{mv}\right) + \dot{\psi}\left(1 + \frac{aC_f - bC_R}{mv^2}\right)$$
$$\ddot{\psi} = -\delta\left(\frac{bC_R}{I_z}\right) + \beta\left(\frac{bC_r + aC_f}{I_z}\right) + \dot{\psi}\left(\frac{b^2C_r + a^2C_f}{VI_z}\right)$$

$$\begin{bmatrix}\dot{\beta} \\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}\left(\frac{C_f + C_R}{mv}\right) & \left(\frac{aC_f - bC_R}{mv^2} + 1\right) \\ \left(\frac{bC_r + aC_f}{I_z}\right) & \left(\frac{b^2C_r + a^2C_f}{VI_z}\right)\end{bmatrix}X + \begin{bmatrix}-\left(\frac{C_R}{mv}\right) \\ -\left(\frac{bC_R}{I_z}\right)\end{bmatrix}\delta$$

5. Derive the static stability derivatives for the rear steered vehicle. Comment on how different they are when compared to a front steered vehicle.

$$a_F = -\frac{\dot{\psi}a}{v} - \beta$$
$$a_r = \delta + \frac{\dot{\psi}b}{v} - \beta$$
$$F_{fy} = C_f a_F$$
$$F_{yf} = -C_f\frac{\dot{\psi}a}{v} - C_f\beta$$

$$F_{ry} = C_r a_r$$

$$F_{yr} = C_r\delta + C_r\frac{\dot{\psi}b}{v} - C_r\beta$$
$$Y = F_{yf} + F_{yr}$$
$$Y = -C_f\frac{\dot{\psi}a}{v} - C_f\beta + C_r\delta + C_r\frac{\dot{\psi}b}{v} - C_r\beta$$
$$Y = (-C_f - C_r)\beta + (\frac{C_rb - C_fa}{v})\dot{\psi} + C_r\delta$$

Yaw coupling $Y_r = (\frac{C_rb - C_fa}{v})$
Damping in side slip $Y_\beta = (-C_f - C_r)$
Control force derivative $Y_\delta = C_r$
The total tire yaw moment:

$$N = aF_{yf} + bF_{yr}$$

$$N = -C_f\frac{\dot{\psi}a^2}{v} - aC_f\beta - bC_r\delta - C_r\frac{\dot{\psi}b^2}{v} + bC_r\beta$$
$$N = (-aC_f + bC_r)\beta + \left(\frac{-b^2C_r - a^2C_f}{v}\right)\dot{\psi} - bC_r\delta$$

Yaw damping derivative $N_r = (\frac{C_rb - C_fa}{v})$
Static stability derivative $N_\beta = (-C_f - C_r)$
Control moment derivative $N_\delta = C_r$

6. Derive the expressions for critical velocity and characteristic velocity of the rear steered vehicle.

$$\delta = \delta_A + a_F - a_r$$
$$\delta = \delta_A + -\frac{\dot{\psi}a}{v} - \delta - \frac{\dot{\psi}b}{v}$$

$$a_F - a_r = \frac{F_{yf}}{C_f} - \frac{F_{yr}}{C_r}$$
$$= \frac{bmv^2}{lRC_f} - \frac{amv^2}{lRC_r}$$
$$= \frac{mv^2}{lR}\left[\frac{b}{C_f} - \frac{a}{C_r}\right]$$
$$UG = \frac{m(C_rb - C_fa)}{l(C_fC_r)}$$
$$V_{crit} = \frac{l(C_fC_r)}{m(C_rb - C_fa)} = \sqrt{\frac{-l}{UG}}$$

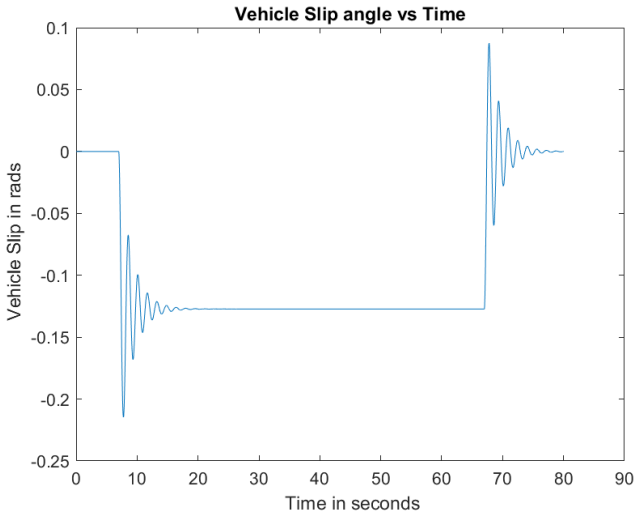
7. Comment on your observations.
We can see that $N_\beta, N_\delta, Y_\delta$ are changing. We can also see that the characteristic velocity is exactly the opposite sign as the critical velocity.

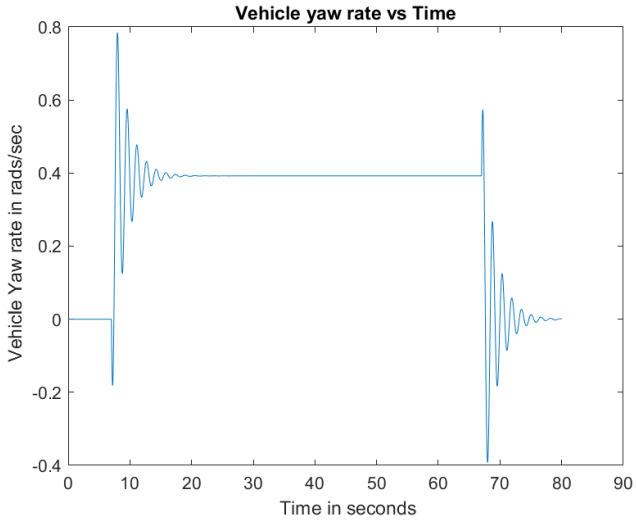
PART II – Model Simulation

Now that the system has been developed, let us test the system under different conditions.

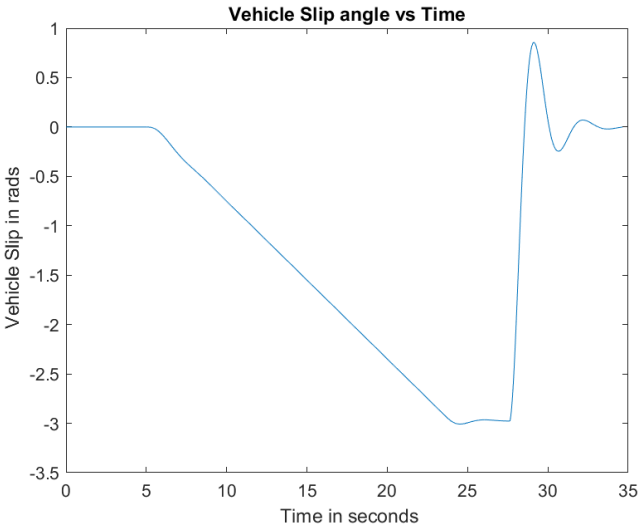
1. Using the input generated and conditions in the assignment 6, test your model for both maneuvers.
   a. Provide plots for each state for both cases.

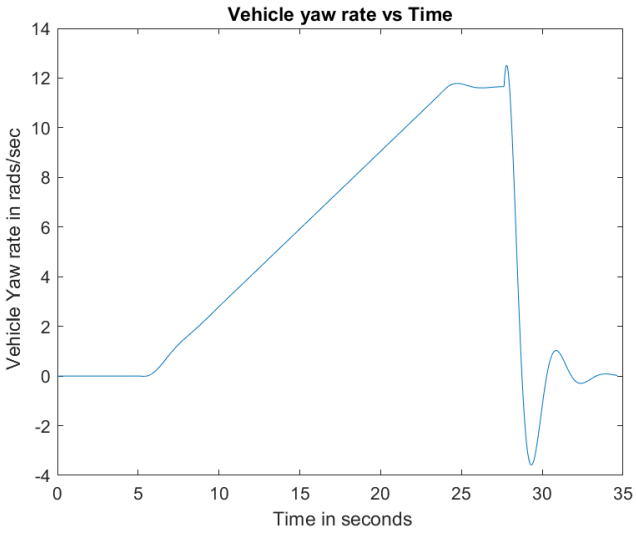
Maneveur 1 Step input:





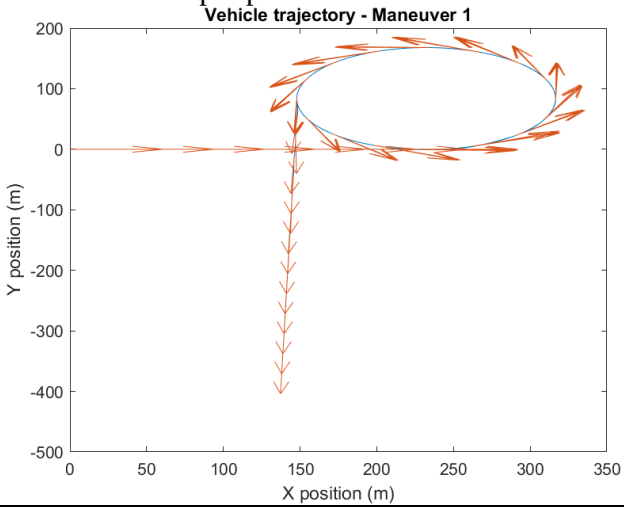
Maneuver 2 Fish hook:



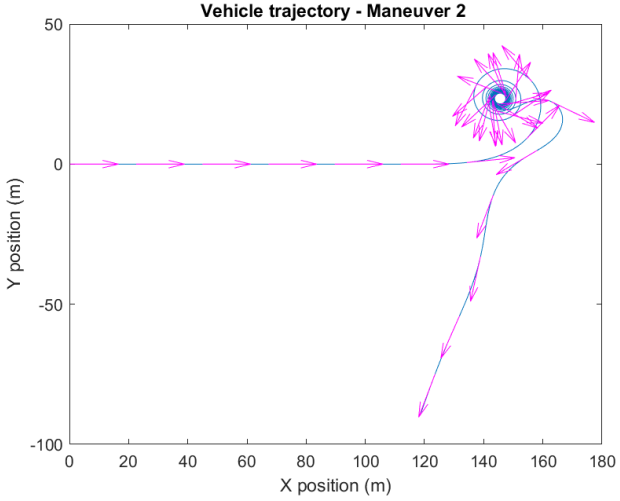


b. Provide plots of the trajectories of center of gravity of the vehicle in inertial coordinates. Also plot the velocity vector in inertial coordinates.

Maneuver 1 Step input:

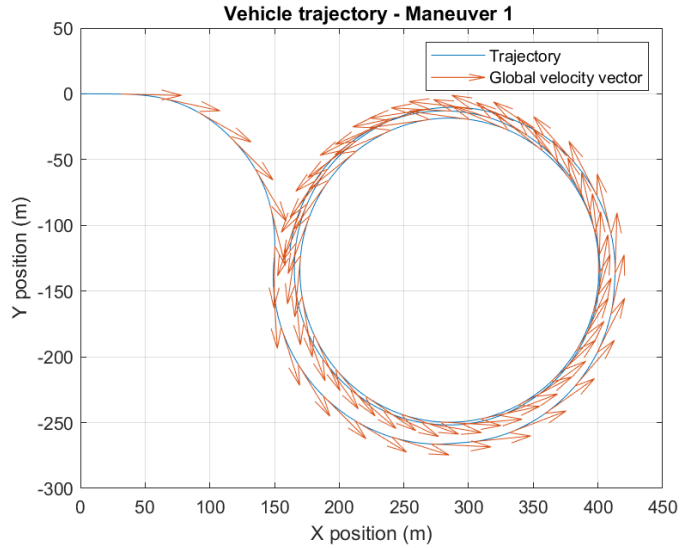


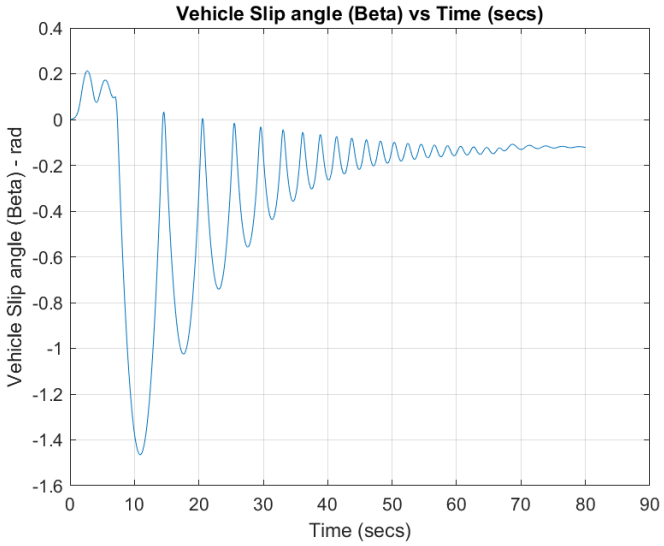
Maneuver 2 Fish hook:

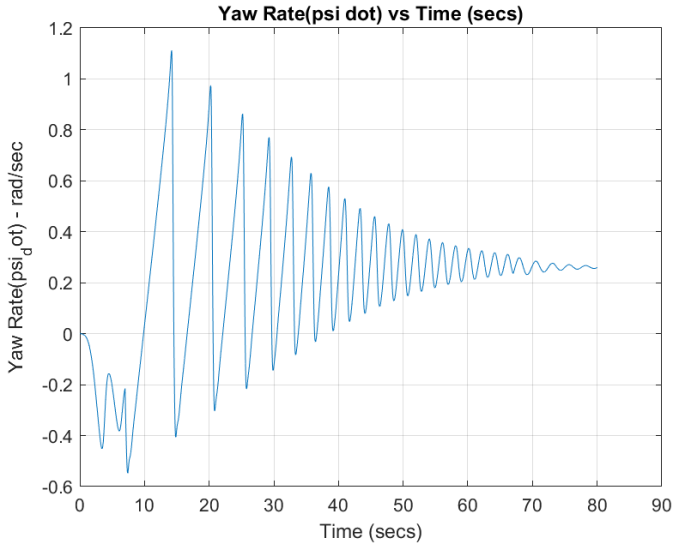


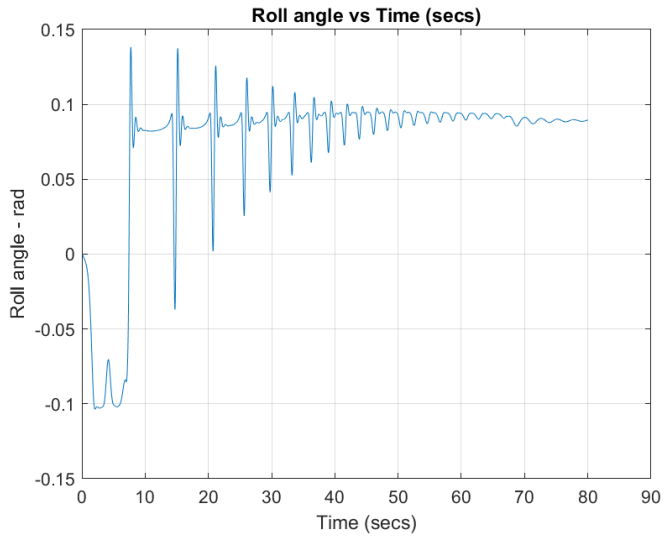
2. Now expand your model from Part I to include the non-linear tire model as well as load transfer with roll dynamics.

Maneuver 1 Step input:

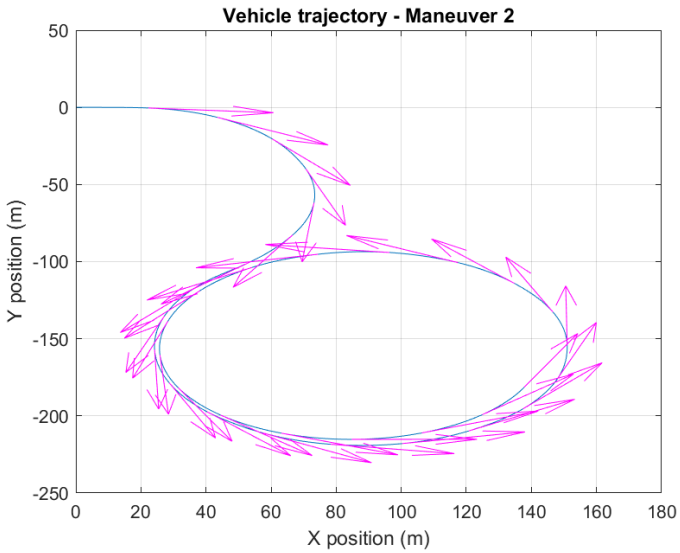


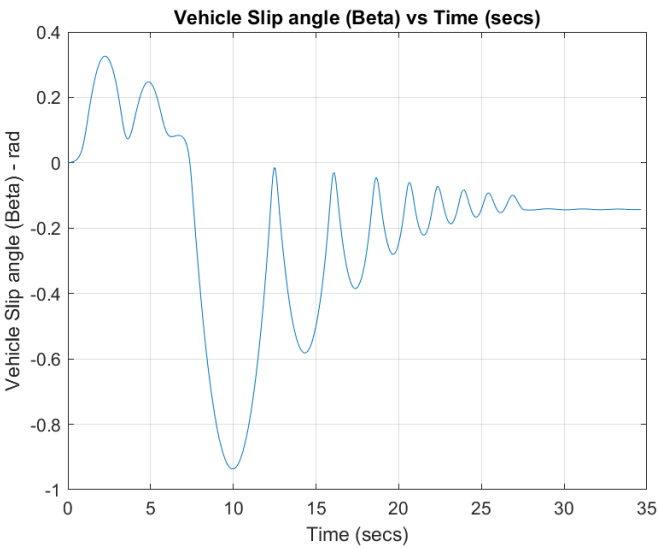


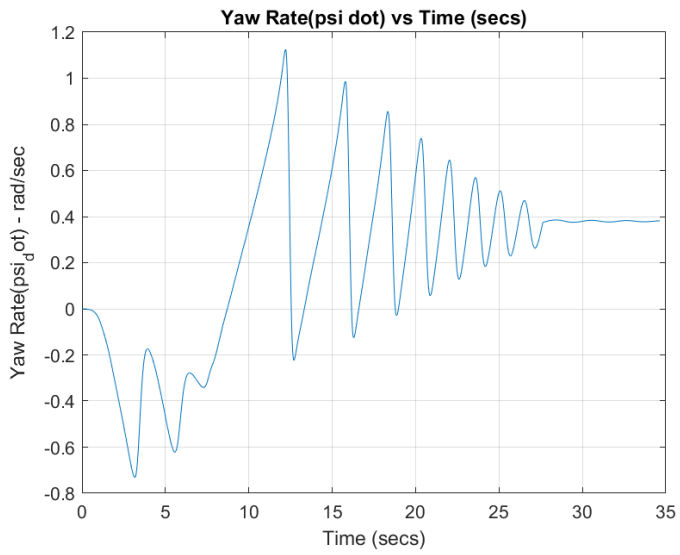


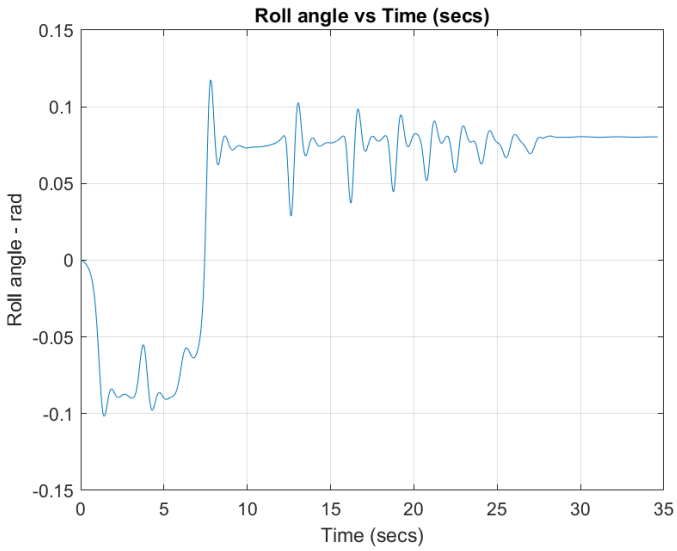


Maneuver 2 Fish hook:







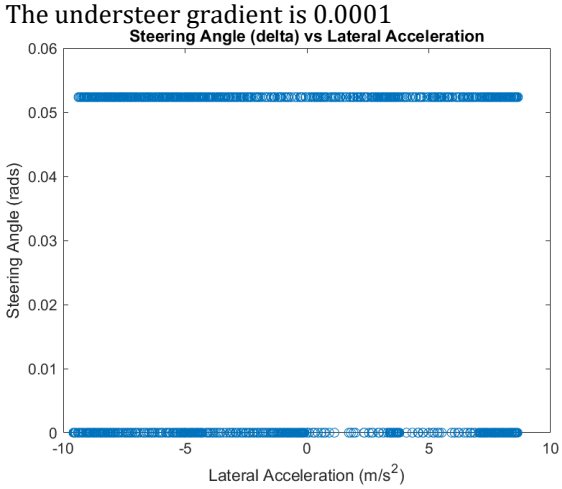


3. For each of the following vehicle conditions, Determine the understeer gradient

a. (Weight distribution, Front roll stiffness distribution) = (60 %,60 %)

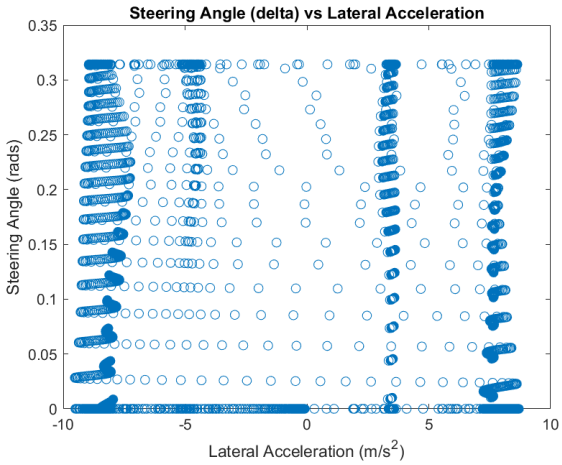
Maneuver 1 Step input:

The understeer gradient is 0.0001



Maneuver 2 Fish hook:

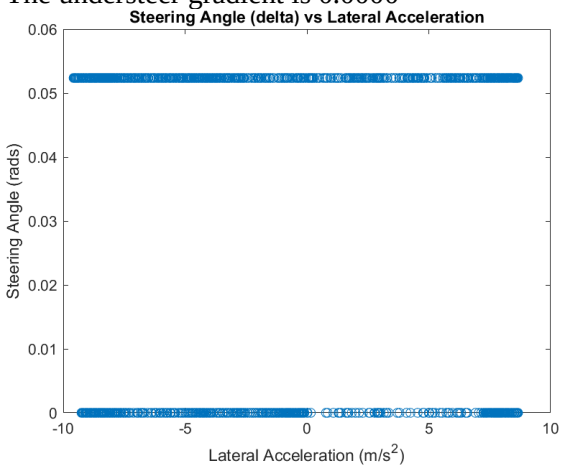
The understeer gradient is 0.0002



b. (55 %,46 %)

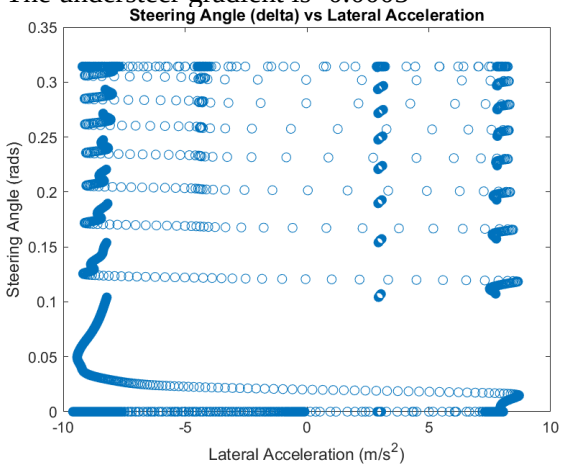
Maneuver 1 Step input:

The understeer gradient is 0.0000



Maneuver 2 Fish hook:

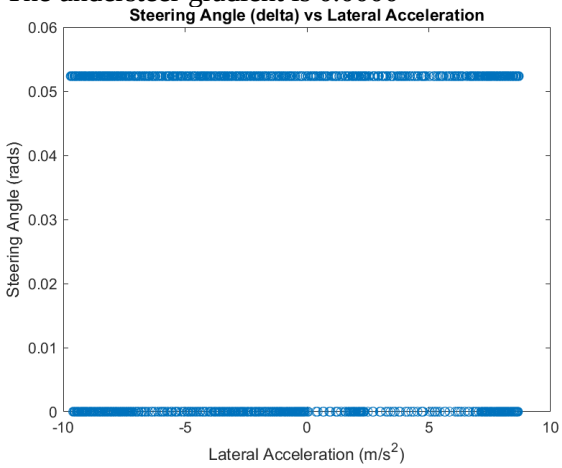
The understeer gradient is -0.0005



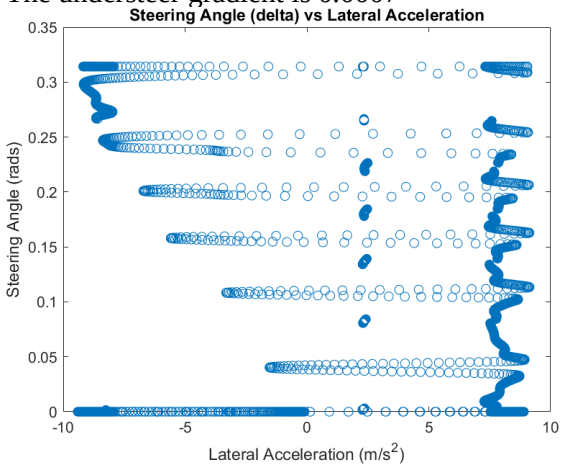
c. (50 %,60 %)

Maneuver 1 step input:

The understeer gradient is 0.0000



Maneuver 2 Fish hook:
The understeer gradient is 0.0007

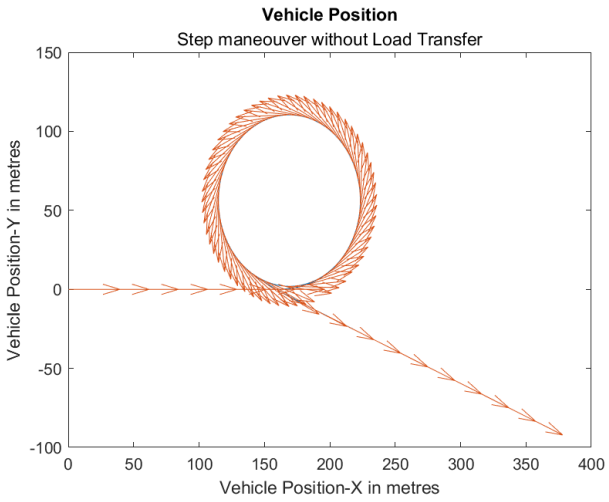


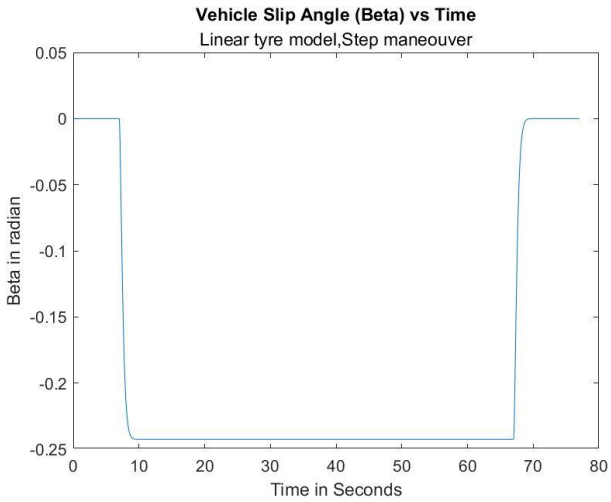
4. Comment on your observations

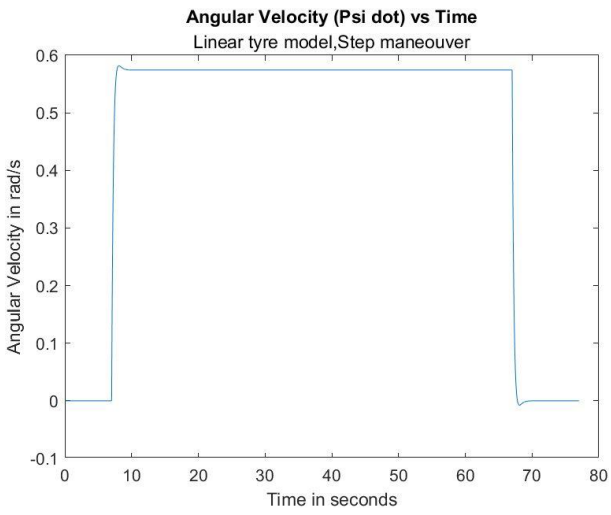
PART III – 2-axle steering
Section 1
Now that you had developed a vehicle model with rear steering input, it is time to combine front and rear axle steering to arrive at a 2-axle steering system. Develop a state-space representation of a 2-axle steering system. δf represents front steering angle and δr represents rear steering angle.
Input conditions:
• δf input: Step input and fishhook maneuver from assignment 6.
• δr input: Develop a strategy for rear steer angle such that vehicle is always at Neutral steer

1. Provide plots for states of the system and trajectories of center of gravity of the vehicle in inertial coordinates. Also plot the velocity vector in inertial coordinates.
Maneuver 1 Step Input:

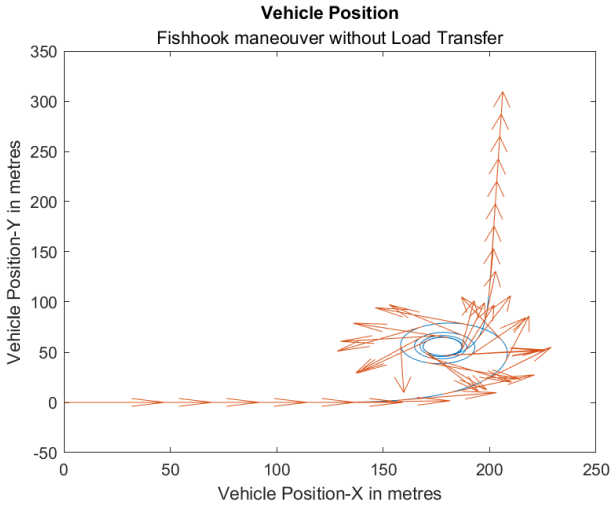
0.4266 and 0.6 roll, Velocity = 30mph

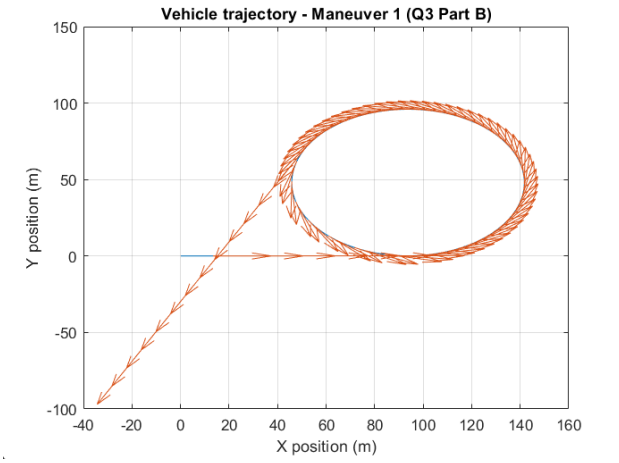


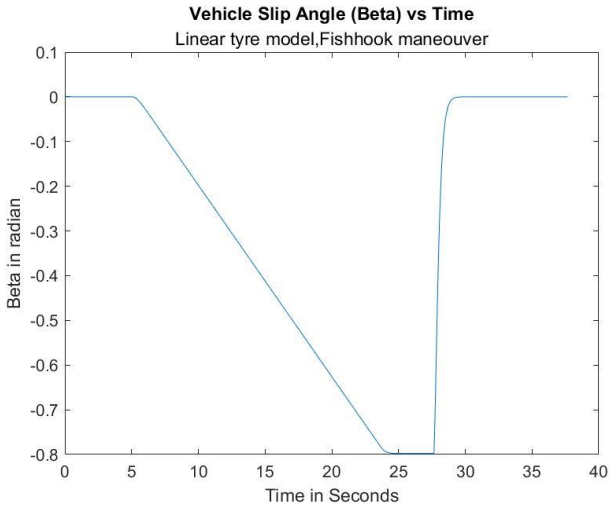


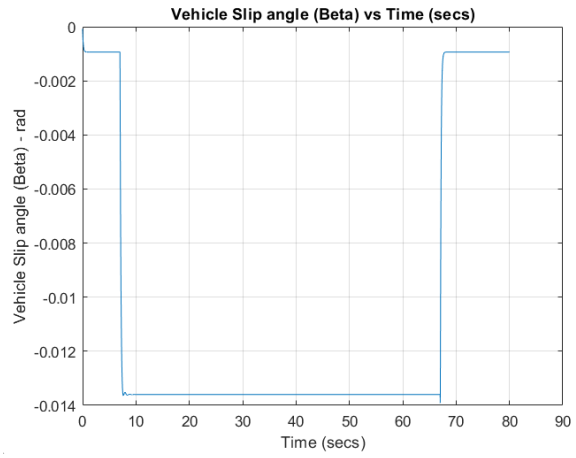


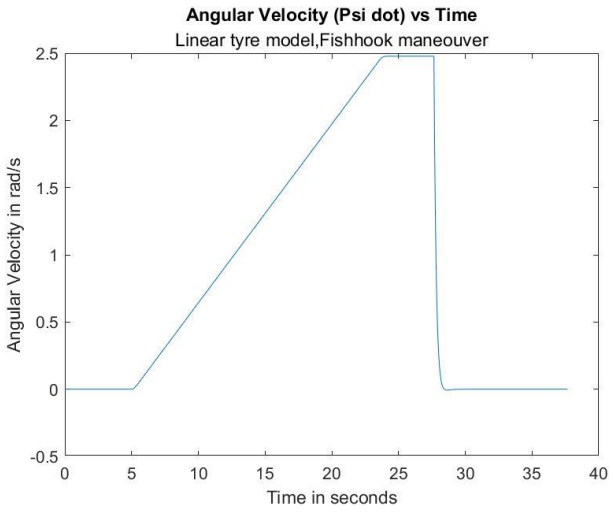
Maneuver 2 Fish hook:

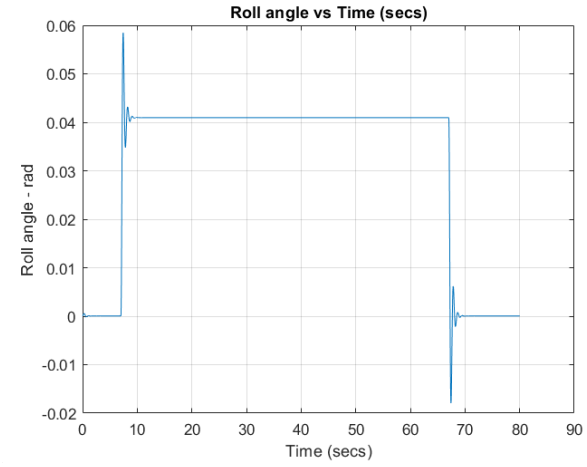




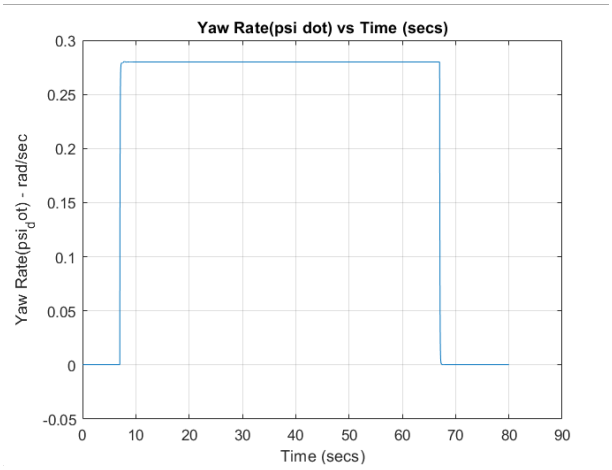


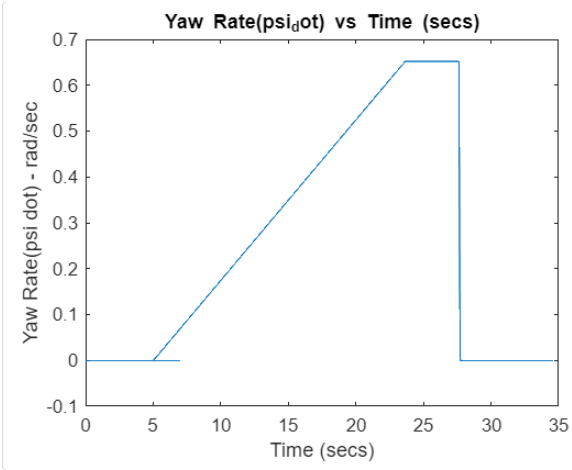






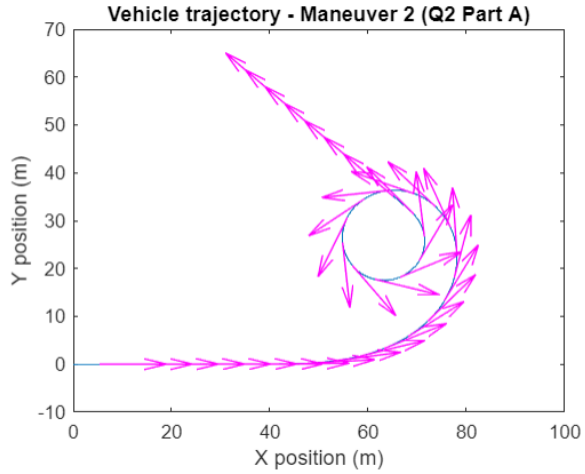
3. Now repeat part A with non-linear tire model and roll dynamics. Provide plots for states of the system and trajectories of center of gravity of the vehicle in inertial coordinates. Also plot the velocity vector in inertial coordinates.
   Maneuver 1 Step input:

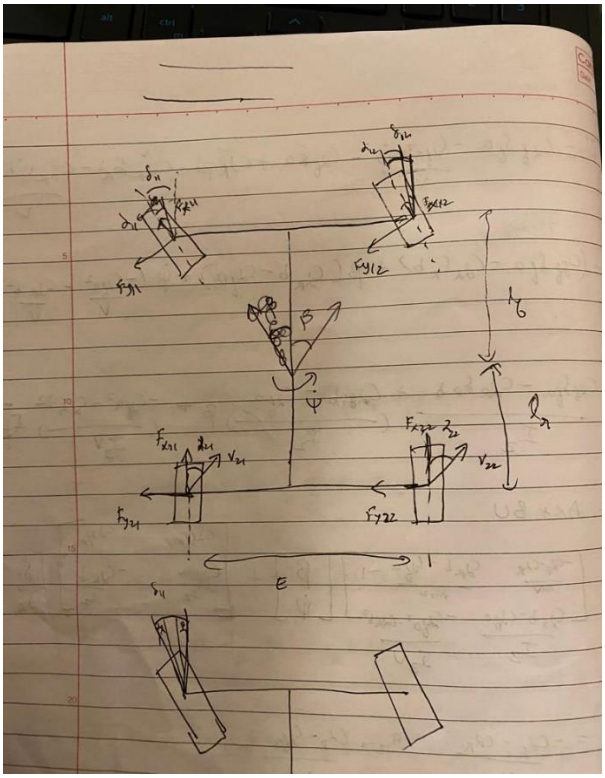


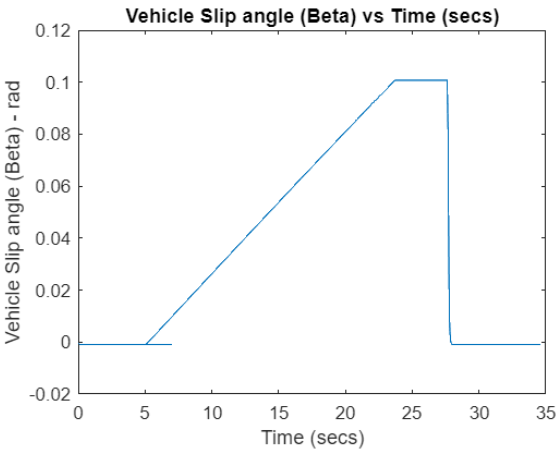


Section 2

Maneuver 2 Fish hook:

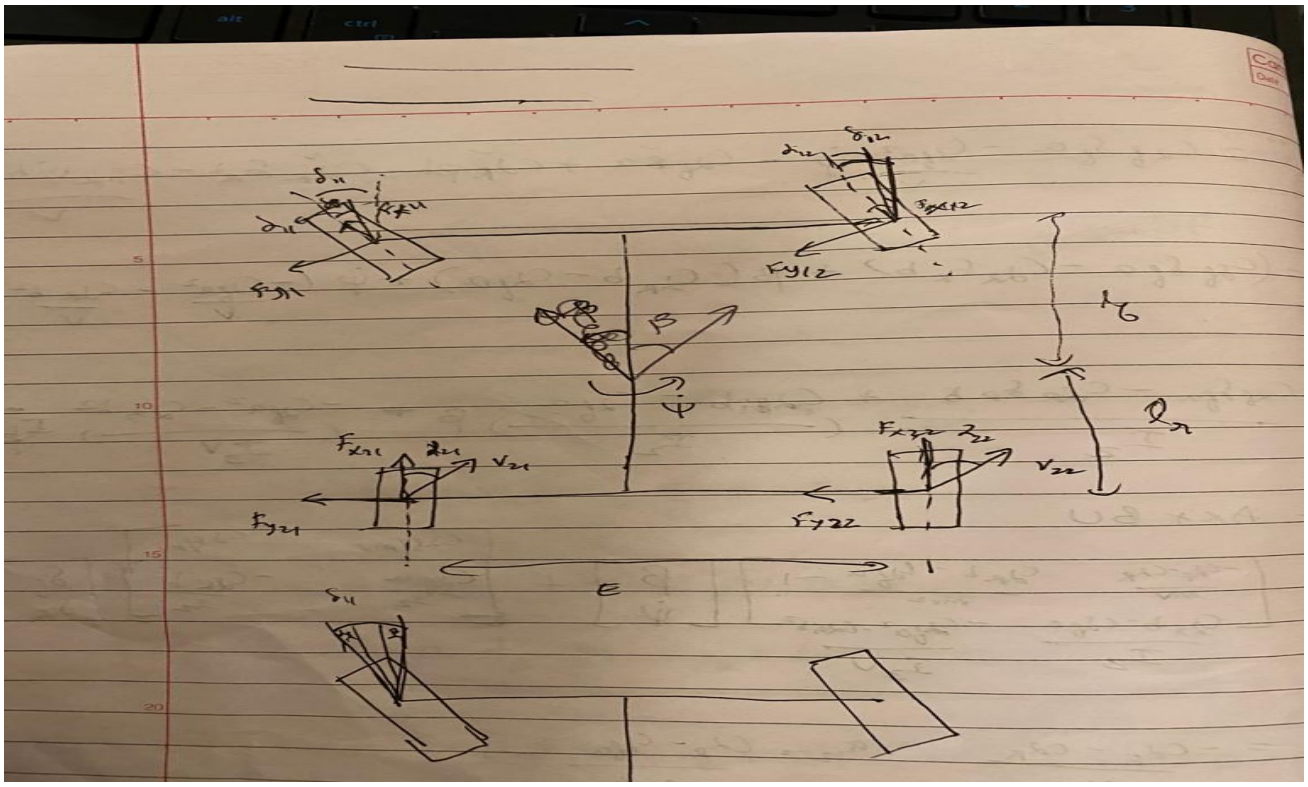




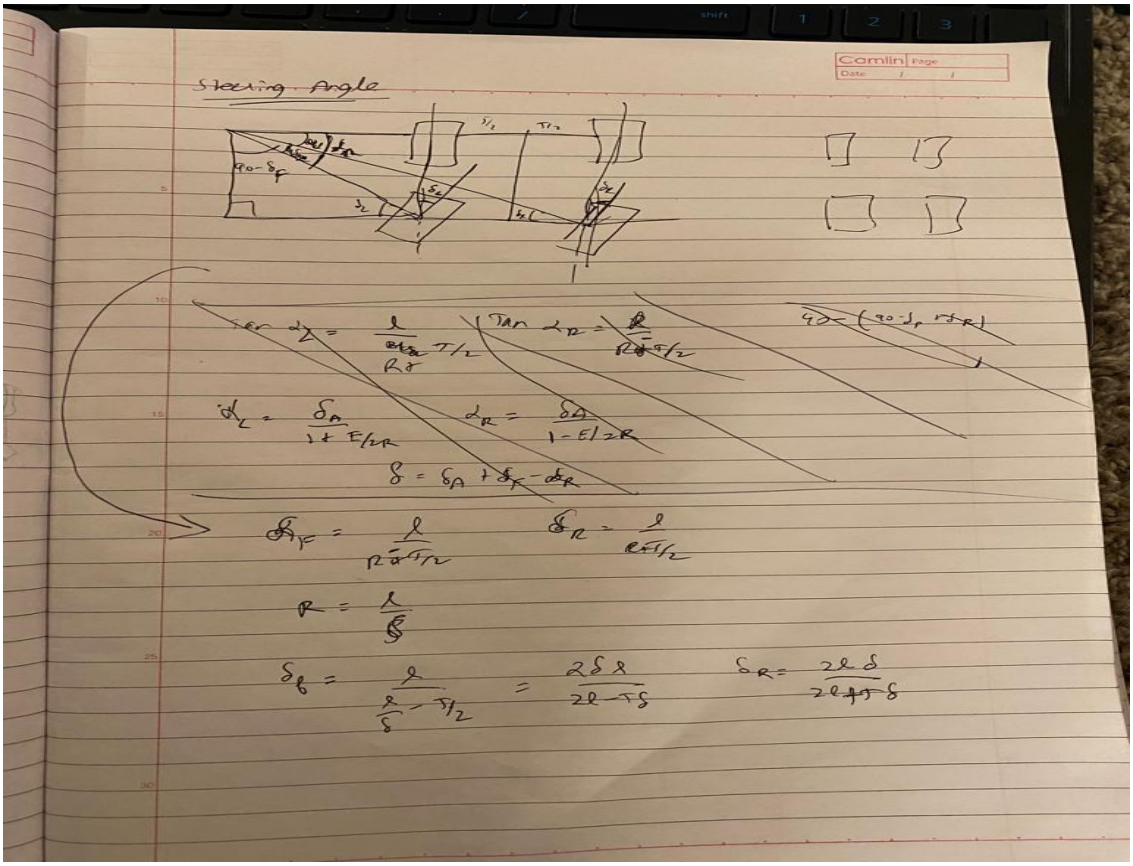


We can see that there is a difference in the trajectories for both maneuvers as expected (as compared to a non-neutral steer case). The exit point is different to both cases as the vehicle conditions are very different in both cases giving different responses in each case, due to always being in neutral steer. We can also see the difference in the states as the vehicle is being driven in different steering conditions. Rather than attain maximum slip angle and angular velocity immediately, we can see a gradual increase in state values towards saturation. We can also observe that there are no transients.

Section 2
Vehicle Model

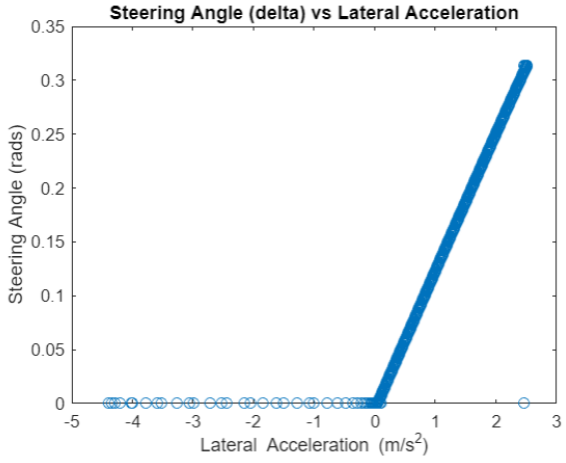


Vehicle Model Equations



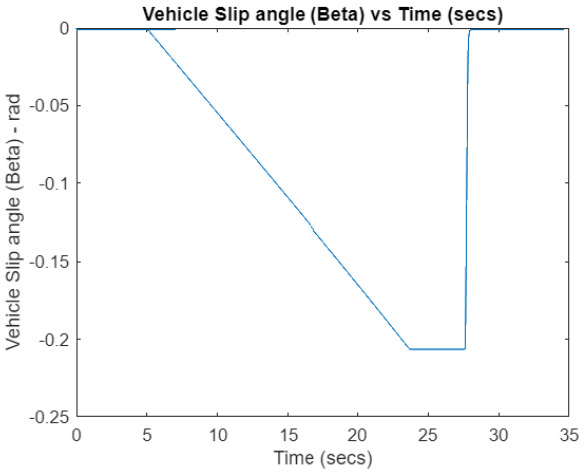
Section 3

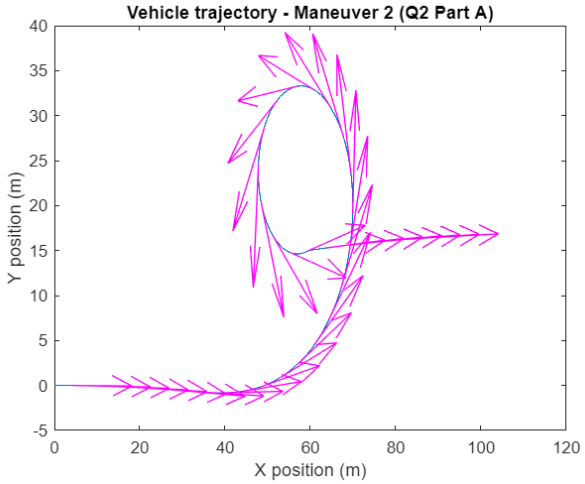
a) Fishhook Maneuver:

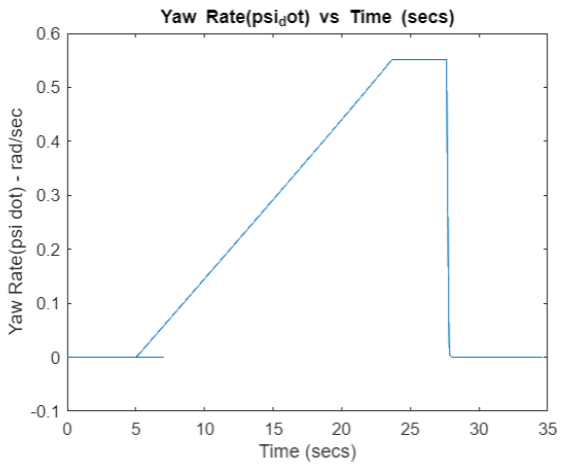


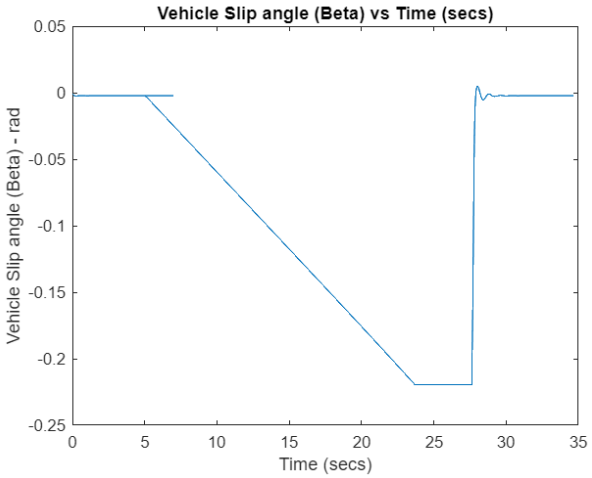

```
The Under steer Gradient for fishhook is -0.00107 rads^s/m
```

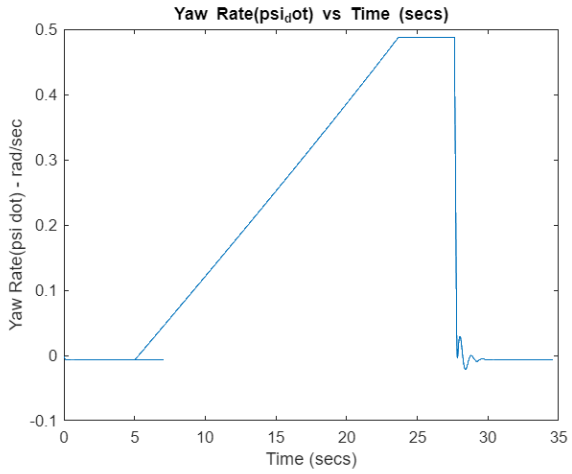
b) Toe out of 5 deg – just for representation

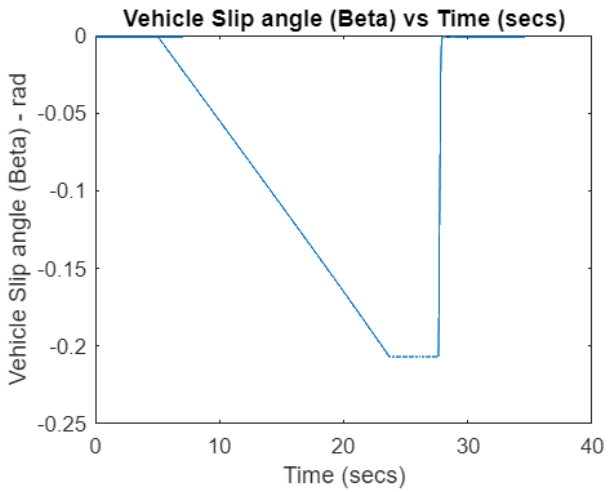


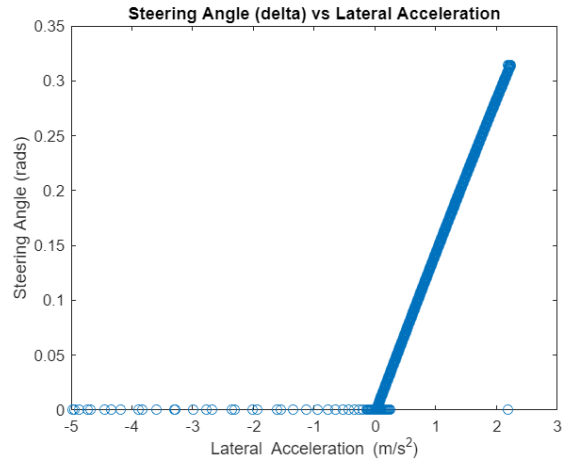




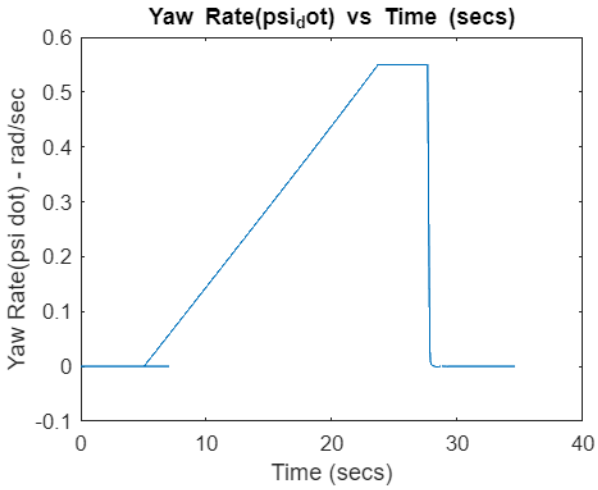




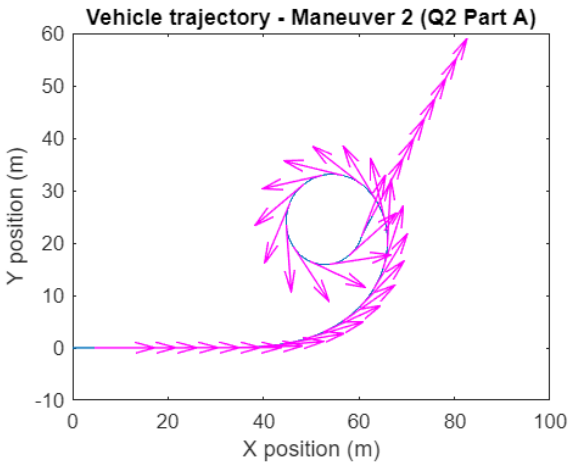


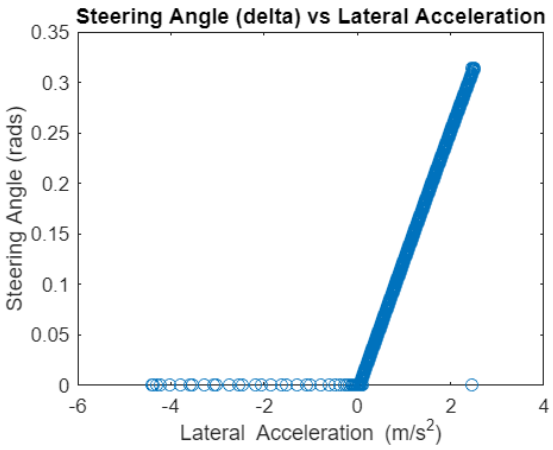


The Under steer Gradient for fishhook is **0.01674 rads^s/m**



Toe of 0.5 deg





The Under steer Gradient for fishhook is **-0.00064 rads^s/m**

i) *Now to visualize a difference between 0 toe angle and 0.5 toe angle, we must plot a trajectory with a toe angle of 5 degreed, as the 0.5 degree to and the 0-degree toe model had no difference visually. From the Delta vs ay plot of 0 toe angle we get UG equal to -0.0017 rad s^2/m, and for the one with toe angle of 0.5 deg the UG is -0.00064 rad s^2/m and it further increases as we increase the toe to 5 deg. The UG value for this case is 0.01674 rad s^2/m.*

This behavior is expected as with a toe in angle the straight-line stability increases which increases the under-steering tendency of the tire.

Appendix:

Linear Model

STEP INPUT:

```
clear
clc


Cf_3 = 1500*(180/pi); % N/rad
Cr_3 = 1146*(180/pi);% N/rad
t_3 = 1.8; % m
m_3 = 3750*0.453592; % kg
Iz = 2150; % kg-m^2
L_1A = 2.6; % m
a_3 = (1-0.42660)*L_1A; % m
b_3 = L_1A - a_3; % m
str_gear_rat = 15;
h_3 = 2.4*0.3048;
v_3 = 74*0.44704; % m/s
a11 = ((Cf_3+Cr_3)/(m_3*v_3));
a12 = ((-(Cr_3*b_3)+(Cf_3*a_3))/(m_3*v_3^2)) + 1;
b1 = -Cr_3/(m_3*v_3);
a21 = ((Cr_3*b_3)-(Cf_3*a_3))/Iz;
a22 = (-(Cr_3*b_3^2)-(Cf_3*a_3^2))/(Iz*v_3);
b2 = -(Cr_3*b_3)/Iz;
A = [a11 a12; a21 a22];
eig(A)
B = [b1;b2];
% 1st manouver
% Euler method is used
% for the 1st 7 seconds, 0deg steering input
t1 = 0:0.01:7;
X = [0;0]; % X is initiated as 0
dt = 0.01;
px = 0; % declaring position = 0 initially
py = 0;
psi = 0;
for i = 1:length(t1)
    delta_3 = 0;
    % State Space equation - to find Xdot
```

```
    Xdot = A*X + B.*delta_3; % new xdot value every step
    % Euler forward equation
    X = X + Xdot*(dt); % new x value = old x value + xdot * dt(time step)
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3*cos(X(1,:))*cos(psi) + v_3*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3*cos(X(1,:))*sin(psi) - v_3*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    betadot_list1(i) = Xdot(1,:);
    Beta_list1(i) = X(1,:); % stores all X value for plot (only 1st row)
    psi_list1(i) = X(2,:); % stores all X value for plot (only 2nd row)
    Vxglb_list1(i) = Vxglb;
    Vyglb_list1(i) = Vyglb;
    px_list1(i) = px;
    py_list1(i) = py;
    p_resultant_list1(i) = p_resultant;
    psi_array1(i) = psi;
end
% for 7.01 - 67.01 seconds, 45deg steering input
t = [7.01:0.01:67.01];
for i = 1:length(t)
    delta_3 = (45/15)*(pi/180); % taking into account steering ratio
    Xdot = A*X + B*delta_3;
    X = X + Xdot*(dt);
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3*cos(X(1,:))*cos(psi) + v_3*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3*cos(X(1,:))*sin(psi) - v_3*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    betadot_list2(i) = Xdot(1,:);
    Beta_list2(i) = X(1,:);
    psi_list2(i) = X(2,:);
    Vxglb_list2(i) = Vxglb;
    Vyglb_list2(i) = Vyglb;
    px_list2(i) = px;
    py_list2(i) = py;
    p_resultant_list2(i) = p_resultant;
    psi_array2(i) = psi;
end
% for 67.02 - 80.02 seconds, 0 deg steering input
t2 = [67.02:0.01:80.02];
```

```
for i = 1:length(t2)
    delta_3 = 0;
    Xdot = A*X + B*delta_3;
    X = X + Xdot*(dt);
    % finding psi
    psi = psi + X(2,:)*(dt);
     % local to global
    Vxglb = v_3*cos(X(1,:))*cos(psi) + v_3*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3*cos(X(1,:))*sin(psi) - v_3*(sin(X(1,:)))*cos(psi);
    Vxglb_list(i) = Vxglb;
    Vyglb_list(i) = Vyglb;
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    betadot_list3(i) = Xdot(1,:);
    Beta_list3(i) = X(1,:);
    psi_list3(i) = X(2,:);
    Vxglb_list3(i) = Vxglb;
    Vyglb_list3(i) = Vyglb;
    px_list3(i) = px;
    py_list3(i) = py;
    p_resultant_list3(i) = p_resultant;
    psi_array3(i) = psi;
end
Beta_list = [Beta_list1,Beta_list2,Beta_list3];
psi_list = [psi_list1,psi_list2,psi_list3];
time = [t1,t,t2];
Vx_global_list = [Vxglb_list1 Vxglb_list2 Vxglb_list3];
Vy_global_list = [Vyglb_list1 Vyglb_list2 Vyglb_list3];
px_list = [px_list1 px_list2 px_list3];
py_list = [py_list1 py_list2 py_list3];
p_resultant_list = [p_resultant_list1 p_resultant_list2 p_resultant_list3];
psi_array = [psi_array1 psi_array2 psi_array3];
Vx_quiver = Vx_global_list(1:100:end);
Vy_quiver = Vy_global_list(1:100:end);
px_quiver = px_list(1:100:end);
py_quiver = py_list(1:100:end);


figure(1)
plot(px_list,py_list)
hold on
quiver(px_quiver,py_quiver,Vx_quiver,Vy_quiver);
title('Vehicle trajectory - Maneuver 1 (Q1 Part A)')
xlabel('X position (m)')
ylabel('Y position (m)')
hold off
figure(2)
plot(time,Beta_list)
```

```
title("Vehicle Slip angle vs Time")
xlabel(" Time in seconds")
ylabel(" Vehicle Slip in rads")
figure(3)
plot(time,psi_list)
title("Vehicle yaw rate vs Time")
xlabel(" Time in seconds")
ylabel(" Vehicle Yaw rate in rads/sec")
```

FISH HOOK:

```
Cf_3 = 1500*(180/pi); % N/rad
Cr_3 = 1146*(180/pi);% N/rad
t_3 = 1.7; % m
m_3 = 3750*0.453592; % kg
Iz = 2150; % kg-m^2
L_1A = 2.6; % m
a_3 = (1-0.4266)*L_1A; % m
b_3 = L_1A - a_3; % m
str_gear_rat = 15 ;
h_3 = 2.4*0.3048;
v_3_2B = 50*0.44704; % m/s
a11_2B = ((Cf_3+Cr_3)/(m_3*v_3_2B));
a12_2B = ((-(Cr_3*b_3)+(Cf_3*a_3))/(m_3*v_3_2B^2)) + 1;
b1_2B = -Cr_3/(m_3*v_3_2B);
a21_2B = ((Cr_3*b_3)-(Cf_3*a_3))/Iz;
a22_2B = (-(Cr_3*b_3^2)-(Cf_3*a_3^2))/(Iz*v_3_2B);
b2_2B = -(Cr_3*b_3)/Iz;
A_2B = [a11_2B a12_2B; a21_2B a22_2B];
B_2B = [b1_2B;b2_2B];
% Fish-hook manouver
t1_2 = 0:0.01:5;
X = [0;0]; % X is initiated as 0
dt = 0.01;
psi = 0;
px = 0;
py = 0;
for i = 1:length(t1_2)
    delta_3 = 0;
    % State Space equation - to find Xdot
    Xdot = A_2B*X + B_2B.*delta_3; % new xdot value every step
    % Euler forward equation
    X = X + Xdot*(dt); % new x value = old x value + xdot * dt(time step)
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3_2B*cos(X(1,:))*cos(psi) + v_3_2B*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3_2B*cos(X(1,:))*sin(psi) - v_3_2B*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
```

```
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    Beta_list1_1(i) = X(1,:); % stores all X value for plot (only 1st row)
    psi_list1_1(i) = X(2,:); % stores all X value for plot (only 2nd row)
    Vxglb_list1_1(i) = Vxglb;
    Vyglb_list1_1(i) = Vyglb;
    px_list1_1(i) = px;
    py_list1_1(i) = py;
    p_resultant_list1(i) = p_resultant;
end
% part2 increment steering input - time is autocalculated based on the
% formula mentioned
delta_3_1 =[0:0.145:270];
t_2 = [5:0.01:(5+(270/14.5))];
 % taking into account steering ratio
for i = 1:length(t_2)
    Xdot = A_2B*X + B_2B*delta_3_1(i)*(pi/180)*(1/15);
    X = X + Xdot*(dt);
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3_2B*cos(X(1,:))*cos(psi) + v_3_2B*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3_2B*cos(X(1,:))*sin(psi) - v_3_2B*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    Beta_list1_2(i) = X(1,:); % stores all X value for plot (only 1st row)
    psi_list1_2(i) = X(2,:); % stores all X value for plot (only 2nd row)
    Vxglb_list1_2(i) = Vxglb;
    Vyglb_list1_2(i) = Vyglb;
    px_list1_2(i) = px;
    py_list1_2(i) = py;
    p_resultant_list1(i) = p_resultant;
end
% for 67.02 - 80.02 seconds, fixed deg steering input
delta_3_2 = (270)*(pi/180);
t2_2 = [t_2(length(t_2))+0.01:0.01:t_2(length(t_2))+0.01+4];
for i = 1:length(t2_2)
    Xdot = A_2B*X + B_2B*delta_3_2*(1/15);
    X = X + Xdot*(dt);
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3_2B*cos(X(1,:))*cos(psi) + v_3_2B*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3_2B*cos(X(1,:))*sin(psi) - v_3_2B*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
```

```
    % Storing values
    Beta_list1_3(i) = X(1,:); % stores all X value for plot (only 1st row)
    psi_list1_3(i) = X(2,:); % stores all X value for plot (only 2nd row)
    Vxglb_list1_3(i) = Vxglb;
    Vyglb_list1_3(i) = Vyglb;
    px_list1_3(i) = px;
    py_list1_3(i) = py;
    p_resultant_list1(i) = p_resultant;
end
% for 67.02 - 80.02 seconds, 0 deg steering input
delta_3_3 = 0;
t2_3 = [t_2(length(t_2))+0.01+4.01:0.01:t_2(length(t_2))+0.01+4.01+7];
for i = 1:length(t2_3)
    delta_3 = 0;
    Xdot = A_2B*X + B_2B*delta_3_3;
    X = X + Xdot*(dt);
    % finding psi
    psi = psi + X(2,:)*(dt);
    % local to global
    Vxglb = v_3_2B*cos(X(1,:))*cos(psi) + v_3_2B*(sin(X(1,:)))*sin(psi);
    Vyglb = v_3_2B*cos(X(1,:))*sin(psi) - v_3_2B*(sin(X(1,:)))*cos(psi);
    % position euler
    px = px + Vxglb*(dt);
    py = py + Vyglb*(dt);
    p_resultant = sqrt(px^2 + py^2);
    % Storing values
    Beta_list1_4(i) = X(1,:); % stores all X value for plot (only 1st row)
    psi_list1_4(i) = X(2,:); % stores all X value for plot (only 2nd row)
    Vxglb_list1_4(i) = Vxglb;
    Vyglb_list1_4(i) = Vyglb;
    px_list1_4(i) = px;
    py_list1_4(i) = py;
    p_resultant_list1(i) = p_resultant;
end
Beta_list_2_3 = [Beta_list1_1,Beta_list1_2,Beta_list1_3,Beta_list1_4];
psi_list_2_3 = [psi_list1_1,psi_list1_2,psi_list1_3,psi_list1_4];
time_2_3 = [t1_2,t_2,t2_2,t2_3];
Vx_global_list_fsh = [Vxglb_list1_1 Vxglb_list1_2 Vxglb_list1_3 Vxglb_list1_4];
Vy_global_list_fsh = [Vyglb_list1_1 Vyglb_list1_2 Vyglb_list1_3 Vyglb_list1_4];
px_list_fsh = [px_list1_1 px_list1_2 px_list1_3 px_list1_4];
py_list_fsh = [py_list1_1 py_list1_2 py_list1_3 py_list1_4];
% p_resultant_list = [p_resultant_list1 p_resultant_list2 p_resultant_list3];
Vx_quiver_fsh = Vx_global_list_fsh(1:100:end);
Vy_quiver_fsh = Vy_global_list_fsh(1:100:end);
px_quiver_fsh = px_list_fsh(1:100:end);
py_quiver_fsh = py_list_fsh(1:100:end);


figure(4)
plot(px_list_fsh,py_list_fsh)
hold on
```

```
quiver(px_quiver_fsh,py_quiver_fsh,Vx_quiver_fsh,Vy_quiver_fsh,0.5,'m')
hold on
title('Vehicle trajectory - Maneuver 2 (Q1 Part A)')
xlabel('X position (m)')
ylabel('Y position (m)')
hold off
figure(5)
plot(time_2_3,Beta_list_2_3)
title("Vehicle Slip angle vs Time")
xlabel(" Time in seconds")
ylabel(" Vehicle Slip in rads")


figure(6)
plot(time_2_3,psi_list_2_3)
title("Vehicle yaw rate vs Time")
xlabel(" Time in seconds")
ylabel(" Vehicle Yaw rate in rads/sec")
```

NON-LINEAR MODEL

```
for i = 1:length(t2_2)

    del_rr = -2*(L_1A*delta_3_2)/((2*L_1A)+(t_3*delta_3_2));
    del_rl = -2*(L_1A*delta_3_2)/((2*L_1A)-(t_3*delta_3_2));
    ay = v_3*(X(2,:) - Xdot(1,:));


    % adding roll
    phi_double_dot = (1/Izz_roll)*((m_3*ay*Hcr) + ...
        (m_3*9.81*Hcr*(phi_roll)) - (Ksr_tot*(phi_roll)) - ...
        (Br*phi_dot));
    phi_dot = phi_dot + phi_double_dot*dt;
    phi_roll = phi_roll + phi_dot*dt;


    % delta Fz
    del_Fz_F = (1*(Ksr_tot*roll_front*phi_roll/t_3)) +
((1*m_3*ay*b_3*Hr_f)/(L_1A*t_3));
    del_Fz_R = (1*(Ksr_tot*(1-roll_front)*phi_roll/t_3)) +
((1*m_3*ay*a_3*Hr_r)/(L_1A*t_3));

    Fz_FL = (wd_front*m_3*0.5*9.81) - del_Fz_F;
    Fz_FR = (wd_front*m_3*0.5*9.81) + del_Fz_F;
    Fz_RL = ((1-wd_front)*m_3*0.5*9.81) -  del_Fz_R;
    Fz_RR = ((1-wd_front)*m_3*0.5*9.81) +  del_Fz_R;


    alpha_rear_r = (del_rr + (X(2,:)*((b_3)/(v_3))) - X(1,:));
    alpha_rear_l = (del_rl + (X(2,:)*((b_3)/(v_3))) - X(1,:));
    alpha_front_l = -(X(2,:)*((a_3)/(v_3))) - X(1,:) + (0.5*pi/180);
```

```
alpha_front_r = -(X(2,:)*((a_3)/(v_3))) - X(1,:)- (0.5*pi/180);



 Fy_fl = nonlintire(alpha_front_l,(Fz_FL),v_3);
Fy_fr = nonlintire(alpha_front_r,(Fz_FR),v_3);
Fy_rl = nonlintire(alpha_rear_l,(Fz_RL),v_3);
Fy_rr = nonlintire(alpha_rear_r,(Fz_RR),v_3);


Fy_f = -(Fy_fl + Fy_fr);
Fy_r = -(Fy_rl + Fy_rr);


Xdot_new_beta = ((Fy_f + Fy_r)/(m_3*v_3)) - X(2,:);
Xdot_new_psidot = ((Fy_f*a_3)-(Fy_r*b_3))/Iz;
% new Xdot for next iteration
Xdot = [Xdot_new_beta ; Xdot_new_psidot];
X = X + Xdot.*(dt); % new x value = old x value + xdot * dt(time step)

% finding psi
psi = psi + X(2,:)*(dt);

% local to global
Vxglb = v_3*cos(X(1,:))*cos(psi) + v_3*(sin(X(1,:)))*sin(psi);
Vyglb = v_3*cos(X(1,:))*sin(psi) - v_3*(sin(X(1,:)))*cos(psi);


% position euler
px = px + Vxglb*(dt);
py = py + Vyglb*(dt);
p_resultant = sqrt(px^2 + py^2);


% Storing values
Beta_list1_3(i) = X(1,:); % stores all X value for plot (only 1st row)
psi_list1_3(i) = X(2,:); % stores all X value for plot (only 2nd row)
Vxglb_list1_3(i) = Vxglb;
Vyglb_list1_3(i) = Vyglb;
px_list1_3(i) = px;
py_list1_3(i) = py;
p_resultant_list1(i) = p_resultant;
ay_list1_3(i) = ay;
```