Simulation 4
ME – 565
Spring, 2024

By: Andres F. Hoyos

Consider a vehicle with the following parameters:

$W = 3000$ lbs,
$W_s = 2700$ lbs,
$x_1 = 3.5$ ft,
$x_2 = -4.5$ ft,
$h = -1.0$ ft,
$t = 6.0$ ft,
$I_z = 40000$ lbs*ft^2,
$I_x = 15000$ lbs*ft^2,
$c = 0.5$ ft,
$\frac{\partial L}{\partial \phi_f} = 40000$ lbs*ft,
$\frac{\partial L}{\partial \phi_r} = 5000$ lbs*ft,
$\frac{\partial L}{\partial \dot{\phi}_f} = 1000$ lbs*ft/sec,
$\frac{\partial L}{\partial \dot{\phi}_f} = 500$ lbs*ft/sec,
$p = d = 12$ in,
$\eta = 15:1$,
$\epsilon = 1$ (steering gearbox efficiency, NOT roll steer coefficient – let's call it 1),
$K_s = 10$ in*lbs/deg, and
$t_m = 3$ in.

Cornering coefficients:
For the linear part of the simulation use: $C_i = 140$ lbs/deg.

The following formula for nonlinear tire stiffness can be used where W is the weight on a given tire in pounds (lbs):

$C_i = \left(0.2 * W_i - 0.0000942 * W_i^2\right)$ lbs/deg.

Initial setup:

Before proceeding with the simulation, an initial set up is required. The first step will be to convert units since some units are incompatible. **Note:** ft are going to be used instead of in.

Unit conversions:

1. $I_z = \frac{I_z}{g} = \frac{40000}{32.174} = 1243.2$ lb*ft → lbm*ft.

2. $I_x = \frac{I_x}{g} = \frac{15000}{32.174} = 466.215$ lb*ft → lbm*ft.

3. $p = \frac{p}{12} = \frac{12}{12} = 1$ in → ft.

4. $d = \frac{d}{12} = \frac{12}{12} = 1$ in → ft.

5. $K_s = (K_s)(\frac{1}{12})(\frac{180}{\pi})$ in*lbs/deg → in*lbs/rad

6. $t_m = \frac{t_m}{12} = \frac{3}{12}$ in → ft.

Some necessary calculations:

Masses can be computed as

$m = \frac{W}{g} = 93.243$ lbm,

$m_s = \frac{W_s}{g} = 83.919$ lbm.

The cornering stiffnesses of each axle can be calculated from the given information. The cornering stiffnesses are for the tires, hence, to get the cornering stiffness of the axle, the left and right tire cornering stiffnesses for that axle should be added and the units can be converted to compatible ones with the notation used on this work:

$C_1 = (C_{1r} + C_{1l})(\frac{180}{\pi}) = (140 + 140)(\frac{180}{\pi}) = 16043$ lbs/rad,

$C_2 = (C_{2r} + C_{2l})(\frac{180}{\pi}) = (140 + 140)(\frac{180}{\pi}) = 16043$ lbs/rad.

A Matlab script was written to perform the initial setup (see Initial setup). This script loads the parameters into the workspace and performs all the calculations previously mentioned (Note**:** this code snippet should be executed before running the next scripts for part 1, the entire code can be found in):

Part 1: 2-DoF Vehicle (Linear + Non-Linear)

1. Develop a base 2-DoF linear state space yaw rate model and calculate the system eigen values when forward speed is 30 and 60 mph.

The 2-DoF Yaw Plane Model in state space form is given by

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}\frac{-C_1-C_2}{mu} & \frac{-x_1C_1-x_2C_2}{mu}-1\\ \frac{-x_1C_1-x_2C_2}{I_z} & \frac{-x_1^2C_1-x_2^2C_2}{I_zu}\end{bmatrix}\cdot\begin{bmatrix}\beta\\ r\end{bmatrix}+\begin{bmatrix}\frac{C_1}{mu}\\ \frac{x_1C_1}{I_z}\end{bmatrix}\cdot[\delta] \quad ,$$

where

$$A=\begin{bmatrix}\frac{-C_1-C_2}{mu} & \frac{-x_1C_1-x_2C_2}{mu}-1\\ \frac{-x_1C_1-x_2C_2}{I_z} & \frac{-x_1^2C_1-x_2^2C_2}{I_zu}\end{bmatrix} \quad \text{and} \quad B=\begin{bmatrix}\frac{C_1}{mu}\\ \frac{x_1C_1}{I_z}\end{bmatrix} \quad .$$

The eigenvalues of A matrix are the pole locations of the system. Hence, the eigenvalues for u=30 mph and u=60 mph can be calculated from A matrix. To solve the problem, the following Matlab script was written (see Code1Q1).

Running Code1Q1 generates the following results:

A30 =

```
   -7.8206   -0.9111
   12.9040   -9.5314
```

A60 =

```
   -3.9103   -0.9778
   12.9040   -4.7657
```

eg_30mph =

```
  -8.6760 + 3.3205i
```

-8.6760 - 3.3205i

eg_60mph =

-4.3380 + 3.5262i
-4.3380 – 3.5262i

Analysis P1Q1: Given these results, it can be seen that the **real part** of the eigenvalues have been halved after increasing the speed from 30 mph to 60 mph. It can be seen that the system became slower after performing this change. Moreover, it can be seen that as the speed increased from 30 mph to 60 mph, the **imaginary part** increased in magnitude which means that the transient response at 60 mph is going to have more oscillations than the transient response at 30 mph since imaginary poles produce oscillatory behaviors.

2. Using the steady-state yaw rate response, construct plots from 0 to 120 mph (i.e. 10 by 10) for various speeds, every 10 mph or so.

The steady state gain for the yaw rate in the yaw plane model is given by

$$K_{ss} = \frac{u}{l_2 + u^2 K_{understeer}} \quad \text{, with} \quad K_{understeer} = \frac{-m(x_1 C_1 + x_2 C_2)}{C_1 C_2 l_2} \quad .$$

The code shown in Code1Q2 was written in Matlab to solve this exercise:

Note: the code converts the speed units from mph to ft/sec and then converts back to mph to plot the results.

Code Code1Q2 generates the plot shown in Figure 1:

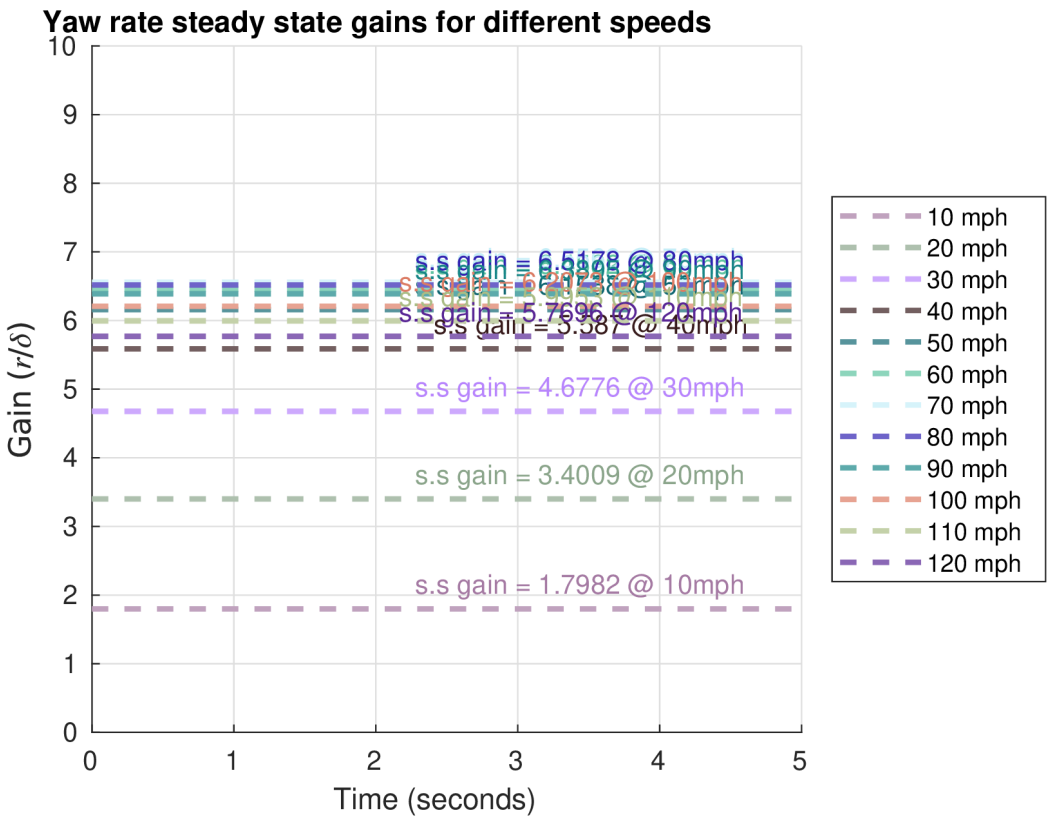


Figure 1: Shows the steady state gains of this vehicle for different speeds.

Analysis P1Q2: It can be seen from Figure 1 that before 70 mph, the steady-state gain increases, but after this value, the steady state gain decreases. A more in depth analysis will be to actually calculate the **characteristic speed** that will give information on exactly at which speed does the yaw rate steady state gain starts decreasing. The characteristic speed can be calculated as follows

$$u_{char} = \sqrt{\frac{x_1 - x_2}{K_{understeer}}} = \sqrt{\frac{8}{7.2652 * 10^{-4}}} = 104.94 \quad \text{ft/sec} = 71.55 \text{ mph, where}$$

$$K_{understeer} = \frac{-m(x_1 C_1 + x_2 C_2)}{C_1 C_2 l_2} = 7.2652 * 10^{-4} \quad .$$

This value is consistent with the experiment shown in Figure 1. Also, given that the **understeering coefficient** is positive, this means that the vehicle, given its configuration, has a tendency to understeer.

3. Using the 2-DoF equations of motion, simulate the vehicle response to a steering input. Determine the steering input required to result in a 400 ft radius turn. (hint: knowing the forward speed and radius of a circle, the required yaw rate can be calculated). Repeat the result at increments of 10 mph from 10 to 120 mph. Compare the results with the steady-state results from 2) above. We are using this step to validate your model.

To result in a turn radius of 400 ft, the steady state gain from steering wheel angle to yaw rate and the centripetal acceleration equation can be used to determine the required steering wheel angle δ as follows

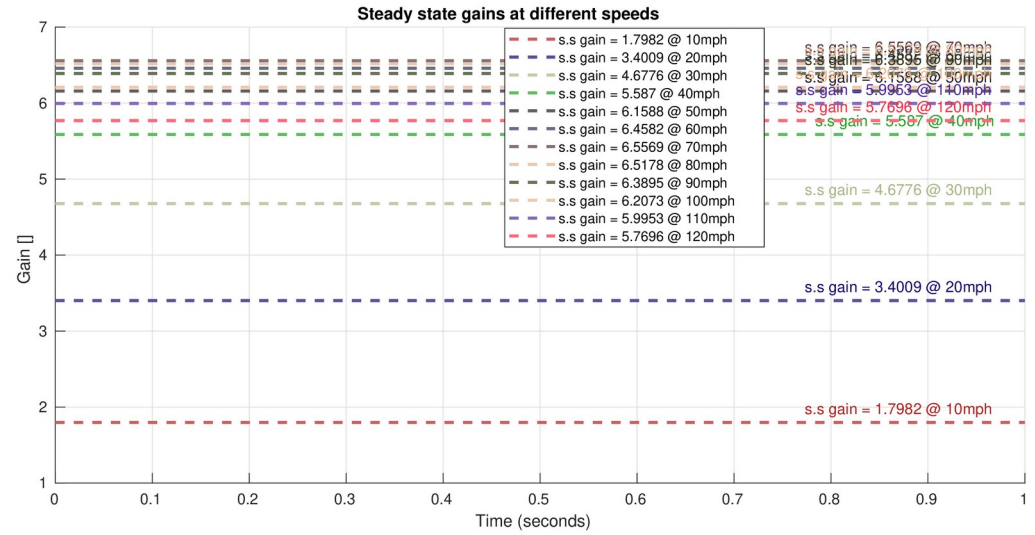
$$\frac{r_{ss}}{\delta} = \frac{(u/R)}{\delta} = \frac{u}{l_2 + u^2 K_{understeer}} \quad ,$$

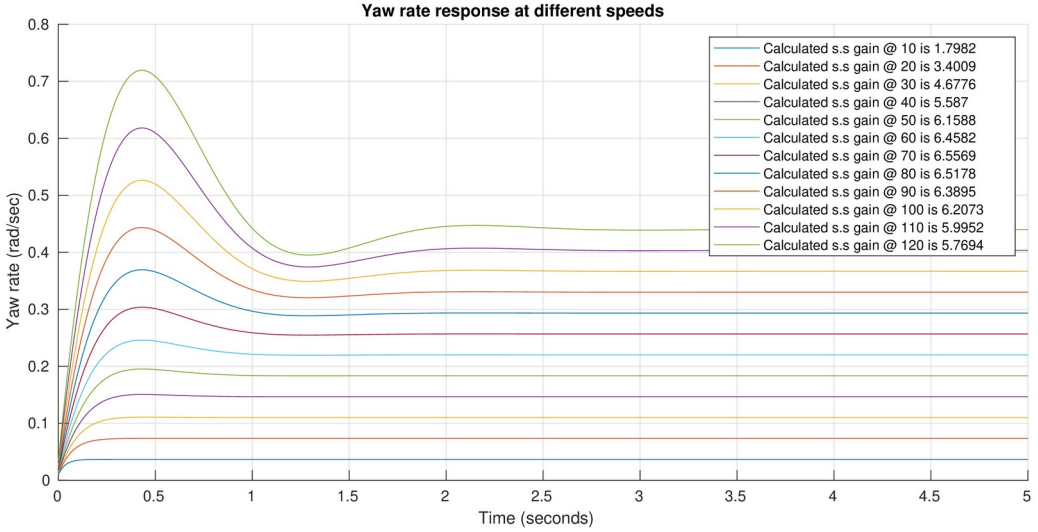
thus

$$\delta = \frac{l_2 + u^2 K_{understeer}}{R} \quad .$$

The code in Code1Q3 was written in Matlab to solve this exercise applying the formula just presented to calculate the steering input required to produce a 400 ft radius turn. Note that this code calls the function *simulate_bike_2dof* in Part 1 Code to perform the time-domain simulation of the system.

Code1Q3 generates the results shown in Figure 2:


Steady state gains at different speeds
s.s gain = 1.7982 @ 10mph
s.s gain = 3.4009 @ 20mph
s.s gain = 4.6776 @ 30mph
s.s gain = 5.587 @ 40mph
s.s gain = 6.1588 @ 50mph
s.s gain = 6.4582 @ 60mph
s.s gain = 6.5569 @ 70mph
s.s gain = 6.5178 @ 80mph
s.s gain = 6.3895 @ 90mph
s.s gain = 6.2073 @ 100mph
s.s gain = 5.9953 @ 110mph
s.s gain = 5.7696 @ 120mph
Gain []
Time (seconds)


Yaw rate response at different speeds
Calculated s.s gain @ 10 is 1.7982
Calculated s.s gain @ 20 is 3.4009
Calculated s.s gain @ 30 is 4.6776
Calculated s.s gain @ 40 is 5.587
Calculated s.s gain @ 50 is 6.1588
Calculated s.s gain @ 60 is 6.4582
Calculated s.s gain @ 70 is 6.5569
Calculated s.s gain @ 80 is 6.5178
Calculated s.s gain @ 90 is 6.3895
Calculated s.s gain @ 100 is 6.2073
Calculated s.s gain @ 110 is 5.9952
Calculated s.s gain @ 120 is 5.7694
Yaw rate (rad/sec)
Time (seconds)

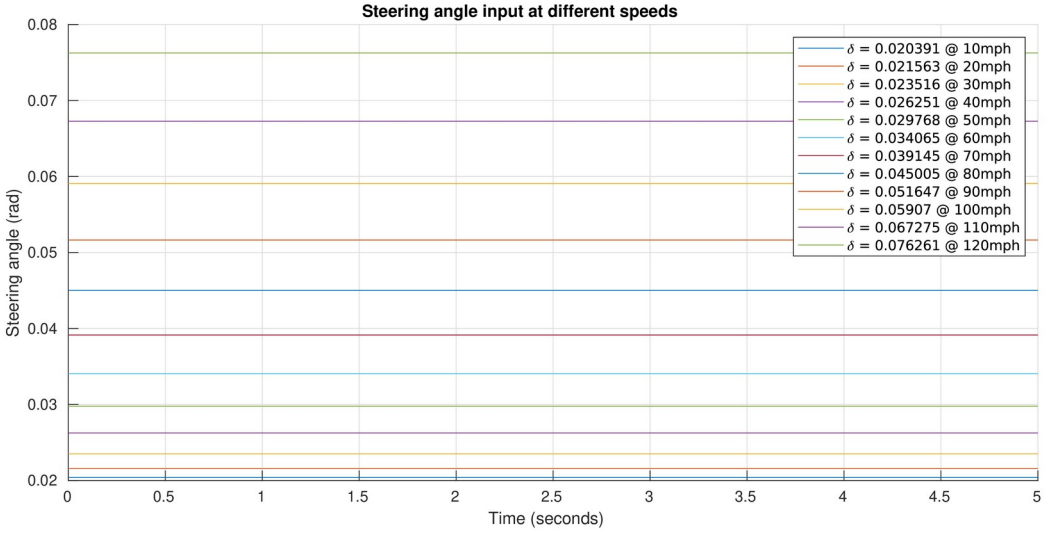

Steering angle input at different speeds
δ = 0.020391 @ 10mph
δ = 0.021563 @ 20mph
δ = 0.023516 @ 30mph
δ = 0.026251 @ 40mph
δ = 0.029768 @ 50mph
δ = 0.034065 @ 60mph
δ = 0.039145 @ 70mph
δ = 0.045005 @ 80mph
δ = 0.051647 @ 90mph
δ = 0.05907 @ 100mph
δ = 0.067275 @ 110mph
δ = 0.076261 @ 120mph
Steering angle (rad)
Time (seconds)

Figure 2: shows in the **top subplot** the steady state gains from steering wheel angle to yaw rate at different speeds, the **middle subplot** shows the time response of the yaw rate given the steering angle input signal shown in the **bottom plot**.

Analysis P1Q3: This code generates the plot shown in Figure 2, this figure has 3 subplots. The **top subplot** shows the steady state gains from steering wheel angle to yaw rate for different speeds. The **subplot in the middle** shows the time response of the yaw rate given the steering input in the **bottom subplot.**

Notice that the steady state gain values of the **middle subplot** on Figure 2 match with the steady state gains (plotted as horizontal lines) in the **top subplot**, this is a method of verification for the mathematical model proposed to solve this exercise. The steady state gain from question 1 gives information on the steady state gain, and this value is confirmed by the gain at which the yaw rate stabilized given the specified inputs.

Also, notice that as the speed increases, the steering wheel input signal increases, this makes sense as taking a sharper turn at high speeds will require higher controller effort.

4. Determine a particular time series of handwheel inputs. We define this input as a 0° handwheel input for 1 second, then a +45° handwheel input for 3 seconds, then a −45° handwheel input for 3 seconds, then back to 0° for 3 seconds. Practically we know that the maximum handwheel input a driver can perform is 2 *rev/sec*, so modify the steering input so that the handwheel velocity is consistent with this value with a 180° input amplitude. Plot the time history of the steering input .

The steering rate saturation in radians per second is:

$$sat = 2 * 2 * \pi = 4\pi \quad \text{rad/sec.}$$

Code1Q4 written in Matlab to solve this exercise. This code calls the function *slope_it_down* in Part 1 Code to saturate the steering angle rate with *sat* as required by the exercise.

This codes generates the steering signal shown in Figure 3 .

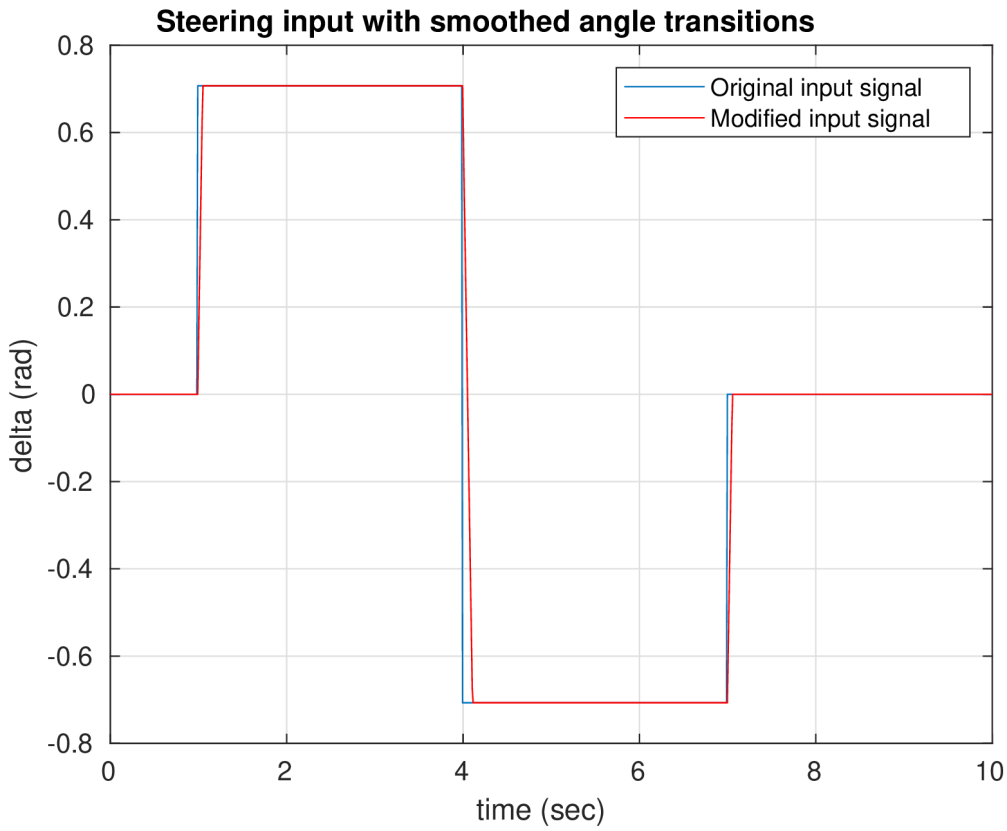


Figure 3: Steering input signal with steering angle rate saturation of 2 rev/sec. The plot compares the original input with no rate saturation (blue) and the modified input signal with rate saturation (red).

Analysis P1Q4: From Figure 3 it can be see that the steering angle now transitions from angle to angle with a ramp steady state. Physically this makes more sense as a human driver is not able to command a steering wheel with step signals.

However, even though the transitions have been smoothed out, this input signal is quite aggressive as the wheel are being suddenly turned to 45 deg and then from 45 deg to -45 deg and then to 0 deg; this is a signal that can potentially induce high lateral forces given the amplitude and the rapid rate of change, specially when the input signals transitions from a ramp to a constant as it produces a jump in steering rate and steering angular acceleration.

5. Use the steering input calculated in 4) as an input to your simulation. Plot the time history of the state variables drift angle and yaw rate r at 30 and 60 mph.

Code1Q5 was developed in Matlab to perform this simulation and generate the time histories.

This code generates the plot shown in Figure 4.

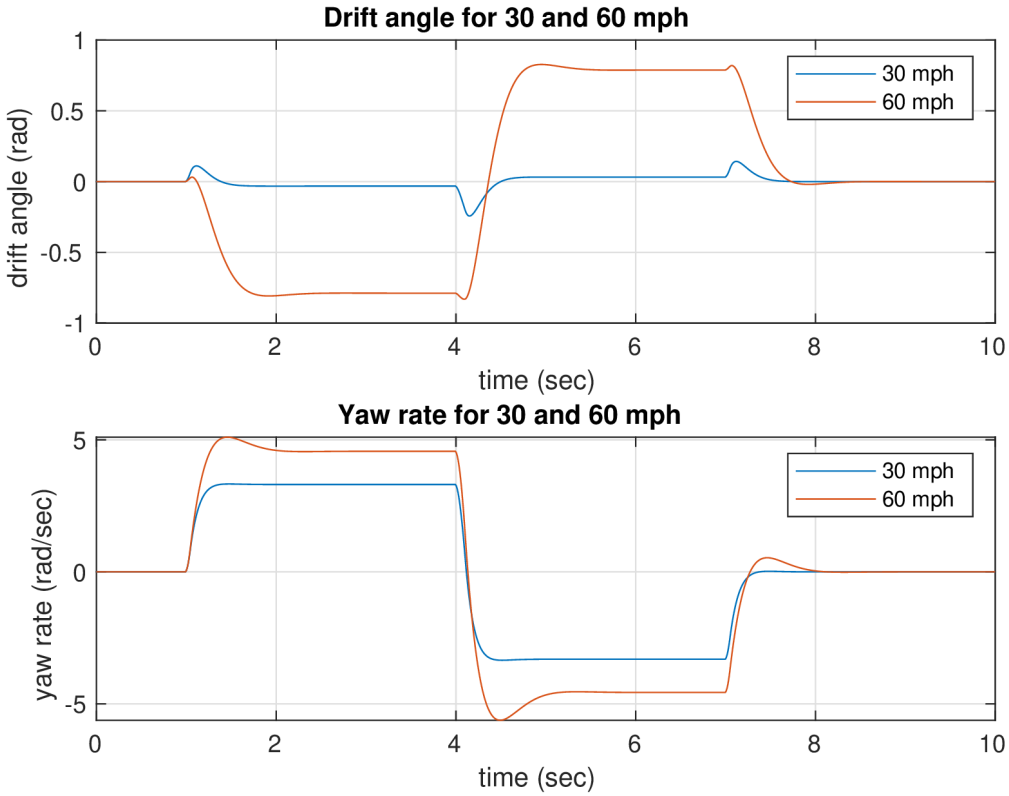


Figure 4: Yaw rate and drift angle time plots for 30 and 60 mph longitudinal speeds.

Analysis P1Q5: From the results shown in Figure 4, it can be supported again the statement done in part 1 – question 2 that the yaw rate gain increases with speed for speeds less than approximately 71 mph given that this is the characteristic speed of this understeering vehicle. Also, notice how the overshoot of the yaw rate response is bigger, this makes sense as the imaginary parts of the eigenvalues have higher absolute values.

Finally, it can be seen that the drift angle response decreases when increasing the speed from 30 mph to 60 mph.

6. Add nonlinear tires and assume equal roll moment front and rear, 60/40 biased to the front, and 40/60 biased to the rear. Compare your biased results to the results of 3)-5) with linear tires.

The front and rear axle cornering stiffnesses can be calculated as follows

$$C_1 = 2\left(0.2\,W_1 - 0.0000942\,W_1^2\right)\left(\frac{180}{\pi}\right) \quad \text{and} \quad C_2 = 2\left(0.2\,W_2 - 0.0000942\,W_2^2\right)\left(\frac{180}{\pi}\right) \ .$$

Comparison with 3:

Code1Q6 was used here with the input signals generated in Code1Q3 to generate the time responses for the **yaw rate** at different speeds and compare the non-linear vs the linear version of the tires. The same Matlab function *simulate_bike_2dof* was used to perform the time domain simulations and the weight bias was assumed constant. Figure 5 shows the yaw rate response for the steering input of **3)** (the steering angle required to produce a 400 ft radius turn) for different speeds for both the linear and non-linear tires proposed with a 50/50 weight bias; Figure 6 shows the same for a 60/40 configuration and Figure 7 for a 40/60 configuration.

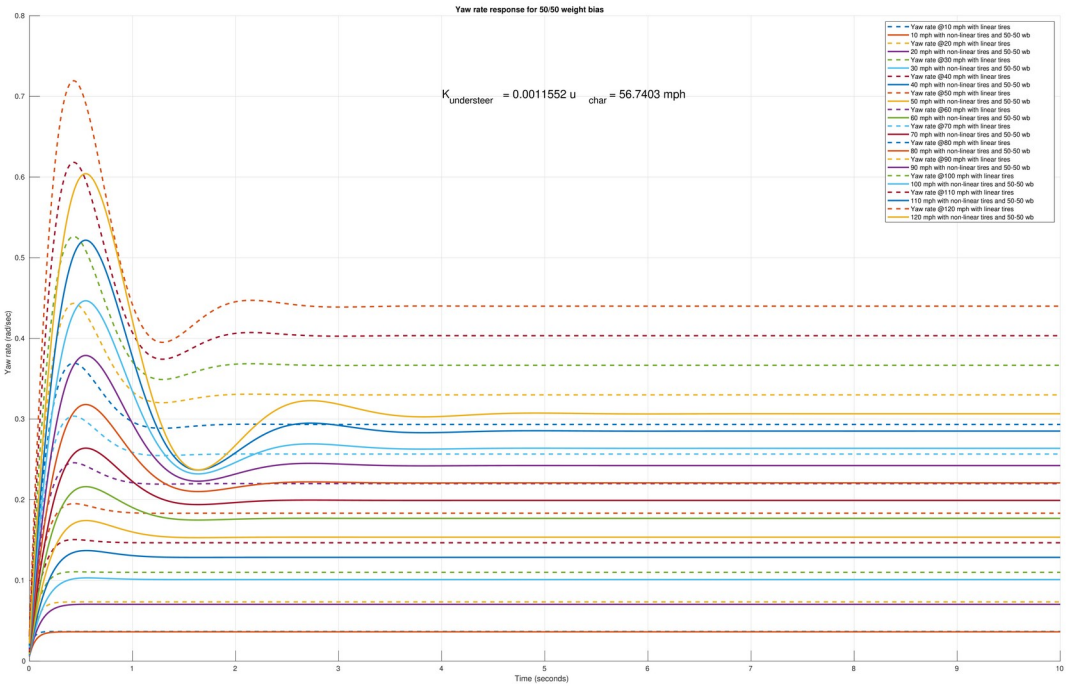


Figure 5: Shows the yaw rate response for different speeds for both the linear tires and the non-linear (quadratic) tires for a 50/50 weight bias configuration. The plot also shows the calculated understeering coefficient and the characteristic speed in the text at the top.

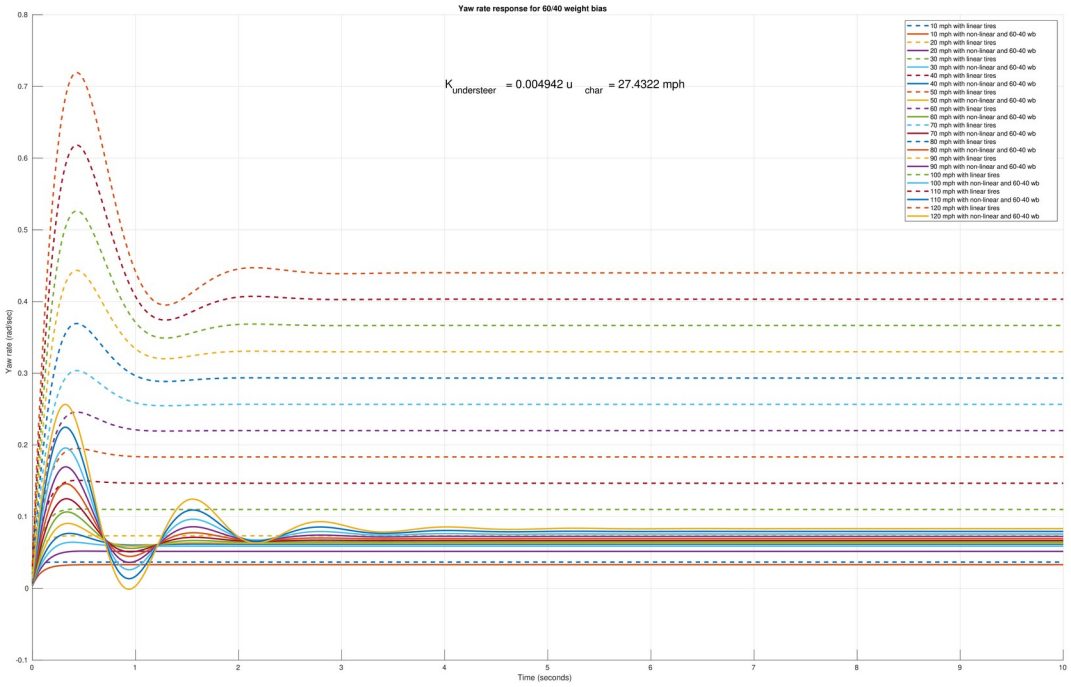


Figure 6: Shows the yaw rate response for different speeds for both the linear tires and the non-linear tires (quadratic) for a 60/40 weight bias configuration. The plot also shows the calculated understeering coefficient and the characteristic speed in the text at the top.

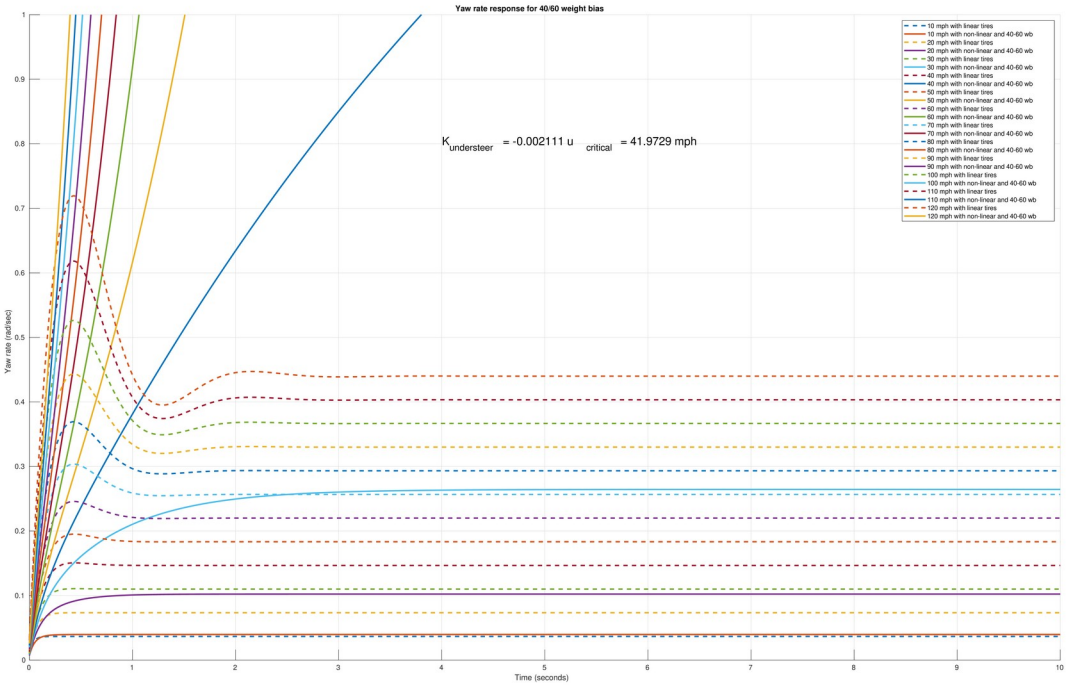


Figure 7: Shows the yaw rate response for different speeds for both the linear tires and the non-linear tires (quadratic) for a 40/60 weight bias configuration. The plot also shows the calculated understeering coefficient and the characteristic speed in the text at the top.

Comparison with 4-5:

Code1Q6B was written in Matlab to solve this exercise, the same steering signal input used in **4)** and **5)** is going to be used to excite the system. Running Code1Q6B generates the results shown in Figure 8, Figure 9 and Figure 10.

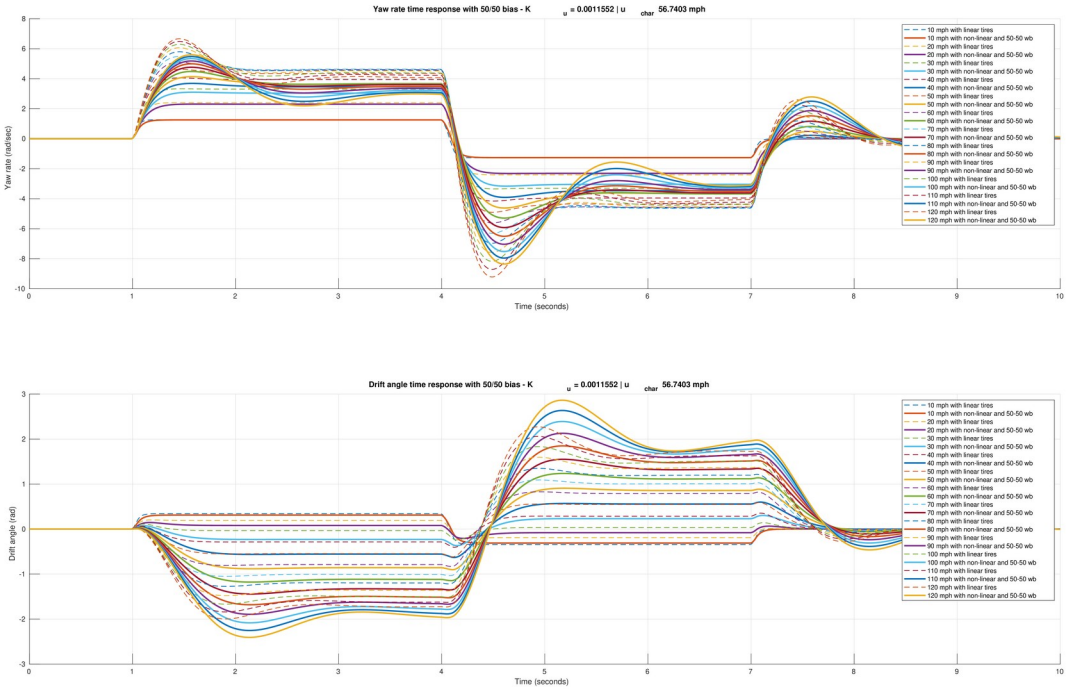
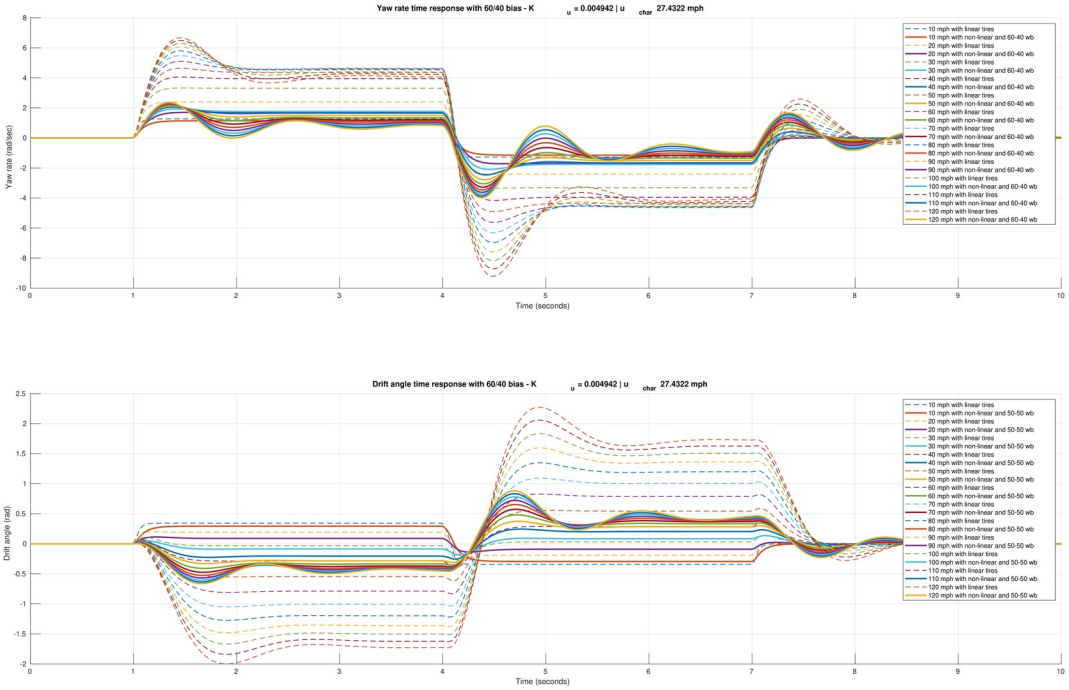


Figure 8: Shows the yaw rate and drift angle time responses for the input generated in **4)** for a 50/50 weight bias.



$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}\frac{-C_1-C_2}{mu} & \frac{-x_1C_1-x_2C_2}{mu}-1\\ \frac{-x_1C_1-x_2C_2}{I_z} & \frac{-x_1^2C_1-x_2^2C_2}{I_zu}\end{bmatrix}\cdot\begin{bmatrix}\beta\\ r\end{bmatrix}+\begin{bmatrix}\frac{C_1}{mu}\\ \frac{x_1C_1}{I_z}\end{bmatrix}\cdot[\delta]$$

Figure 9: Shows the yaw rate and drift angle time responses for the input generated in **4)** for a 60/40 weight bias.

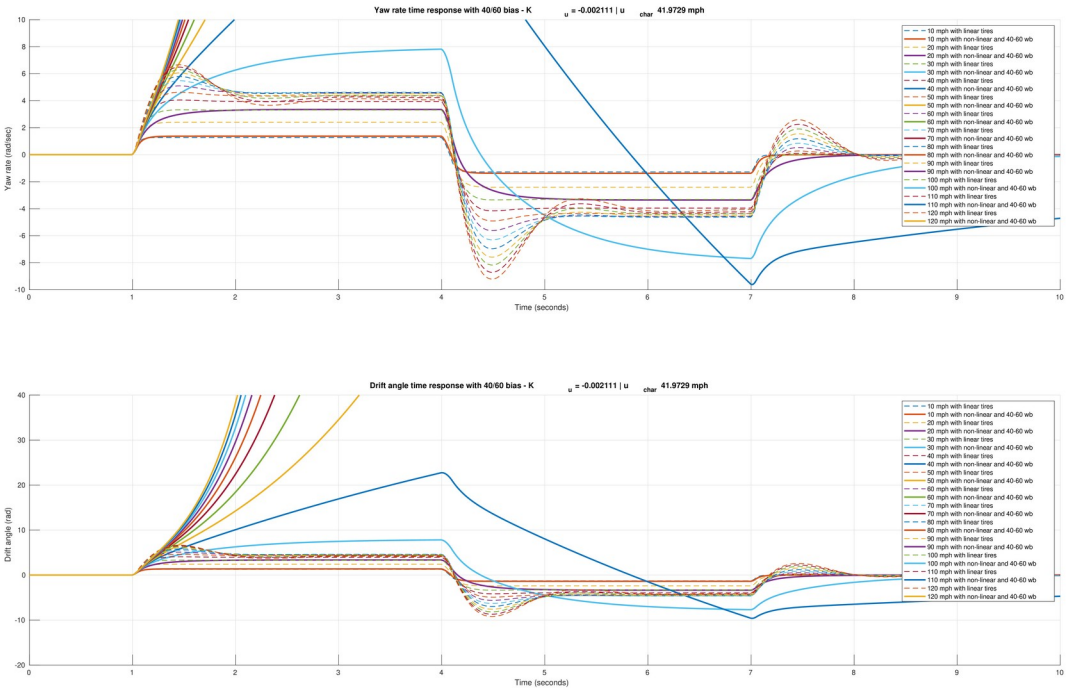


Figure 10: Shows the yaw rate and drift angle time responses for the input generated in **4)** for a 60/40 weight bias.

Analysis P1Q6: It can be seen from Figure 5, Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10, that the general pattern is that as speed increases, both the yaw rate and the drift angle tend to increase. Nevertheless, in the cases where the vehicle handling configurations induced understeer (K_understeer > 0.0), this increase tends to have less gain at high speeds, and the time responses of the states tend to have similar final values at speeds greater than the **characteristic speed.** On the other hand, the vehicle handling configurations that induced oversteer tend to have consistent increasing gains as speed increases as seen in Figure 7 and Figure 10.

50/50 Configuration Analysis:

Figure 5 and Figure 8 show time responses for different steering wheel angle inputs for the 50/50 configuration. It can be seen that the understeer coefficient is about 0.001 which means that the vehicle has tendency towards understeering with a characteristic speed of about 56.7 mph.

The understeering is evident as the gains tend to be lower as speed increases for Figure 5 and Figure 8. Also, notice that in these two figures, at speeds greater than the characteristic speed, the responses start presenting overshoot and oscillations, this is because of the presence of complex conjugate eigenvalues at speeds greater than the characteristic speed.

Overall, the 50/50 configuration presents the most tendency towards neutral steer compared to the other two configurations proposed for this work.

The effect of adding the non-linear tires on this configuration is a decrease in the steady state gain of the yaw rate compared to the linear counterpart. This is due to the fact that the linear tires have a constant slope that always increases with tire load while the non-linear tires have the addition of a squared term that makes the lateral forces decrease after a certain attack angle or load.

60/40 Configuration Analysis:

Figure 6 and Figure 9 show the 60/40 weight bias configuration time responses for the different steering wheel angle inputs. Notice that the understeering coefficient is now much bigger compared to the 50/50 configuration (Ku = 0.005); this is due to the fact that the front is supporting more load, this means that the lateral forces and attack angles will tend to be bigger at the front, making it easier for the front tires to loose grip.

The increased understeering behavior can be appreciated in Figure 6 and Figure 9 both show that the overall steady state gain after the characteristic speed is significantly less compared to the linear tire and the 50/50 configuration.

The effect of adding non-linear tires is a significant decrease in the steady-state gain, even more than the previous configuration.

40/60 Configuration Analysis:

Figure 7 and Figure 10 show the 40/60 weight bias configuration which has a tendency to oversteer given that the understeering coefficient K_u is negative. This system can actually go unstable given that the negative understeering coefficient can produce positive eigenvalues, this happens after the **critical speed** that for this case is around 42 mph. In both figures, it can be seen that at 40 mph (very close to the critical speed), the system becomes very slow and of very high gain, this is due to the fact that it is very close to becoming unstable. After the critical speed, the time responses tend to infinity.

The effect of adding non-linear tires is a significant increase in the steady state gains of the system given its new tendency to oversteer with this configuration.

7. Comment and interpret your analysis.

- Analysis P1Q1 presents an analysis specific to Part 1 – Question 1.

- Analysis P1Q2 presents an analysis specific to Part 1 – Question 2.

- Analysis P1Q3 presents an analysis specific to Part 1 – Question 3.

- Analysis P1Q4 presents an analysis specific to Part 1 – Question 4.

- Analysis P1Q5 presents an analysis specific to Part 1 – Question 5.

- Analysis P1Q6 presents an analysis specific to Part 1 – Question 6.

Overall takeaways:

1. As speed increases, the steady state gain between steering angle and yaw rate tends to increase. This gain will tend to decrease after the characteristic speed if the vehicle has tendency to understeer, and it will tend to infinity after the critical speed if the vehicle has tendency to oversteer (negative K_u). This is evidenced in Figure 1, Figure 2, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10.

2. The effect of adding the non-linear tires given by $C_i = 2(0.2\,W_i - 0.0000942\,W_i^2)(\frac{180}{\pi})$ lbs/rad is that the lateral force will not always increase with vertical load and/or attack angle as the tire will have a linear region close to zero, but then it will decrease its lateral force, this produces overall lower steady state gains in the **yaw rate** and **drift angle** time plots. This is more realistic as it considers non-linearities of the tire behavior that the linear model falls short for operating points far from 0 degrees of attack angle. This can be seen in Figure 5, Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10.

3. Also, it can be seen in the plots when the vehicle tends to understeer that the overshooting and oscillatory behaviors of the **yaw rate** and **drift angle** responses tend to increase after the characteristic speed. For the oversteering vehicles, there is no overshoot and no oscillations as the system will not have complex conjugate roots, but it can actually go unstable as seen in Figure 7 and Figure 10.

Part 2: 3-DoF Vehicle (Linear)

Click here: Part 2-3 Code to see the entire code for both parts 2 and 3 of this simulation project.

Initial setup:

As before, an initial setup Matlab section has to be executed before running the rest of the code for this part. The initial setup code can be found in Code2P1 and it loads the parameters, unit conversions and some very simple initial calculations to solve the exercises.

1. Repeat the steps in (Part 1: 2-DoF Vehicle) with a three degree of freedom model allowing sprung mass roll. Using the steady-state yaw rate response, construct plots from 0 to 120 mph (i.e. increases of 10 mph) for various speeds without roll steer and with a rear roll steer coefficient of $r = -0.03$.

The steady state gain from steering wheel angle δ to yaw rate r is given by

$$\frac{r}{\delta_1}=\frac{u}{l_2+u^2 K_{u,\phi}} \quad \text{, where:}$$

$$l_2=x_1-x_2 \quad ,$$

$$K_{u,\phi}=K_u+\frac{-C_b m+\left(C_b\left(C_{\phi,1}+C_{\phi,2}\right)-C_a\left(x_1 C_{\phi,1}+x_2 C_{\phi,2}\right)\right)\left(\frac{m_s h}{K_\phi}\right)}{C_1 C_2\left(x_1-x_2\right)} \quad \text{, and}$$

$$K_u=\frac{-m\left(x_1 C_1+x_2 C_2\right)}{C_1 C_2 l_2} \quad .$$

This steady state gain is calculated for different speeds for a configuration without roll steer and a configuration with a rear roll steer coefficient of -0.03 (ε_1=0 and ε_2=-0.03) and visualized. The Matlab code in Code2Q1 generates the output shown in Figure 11.

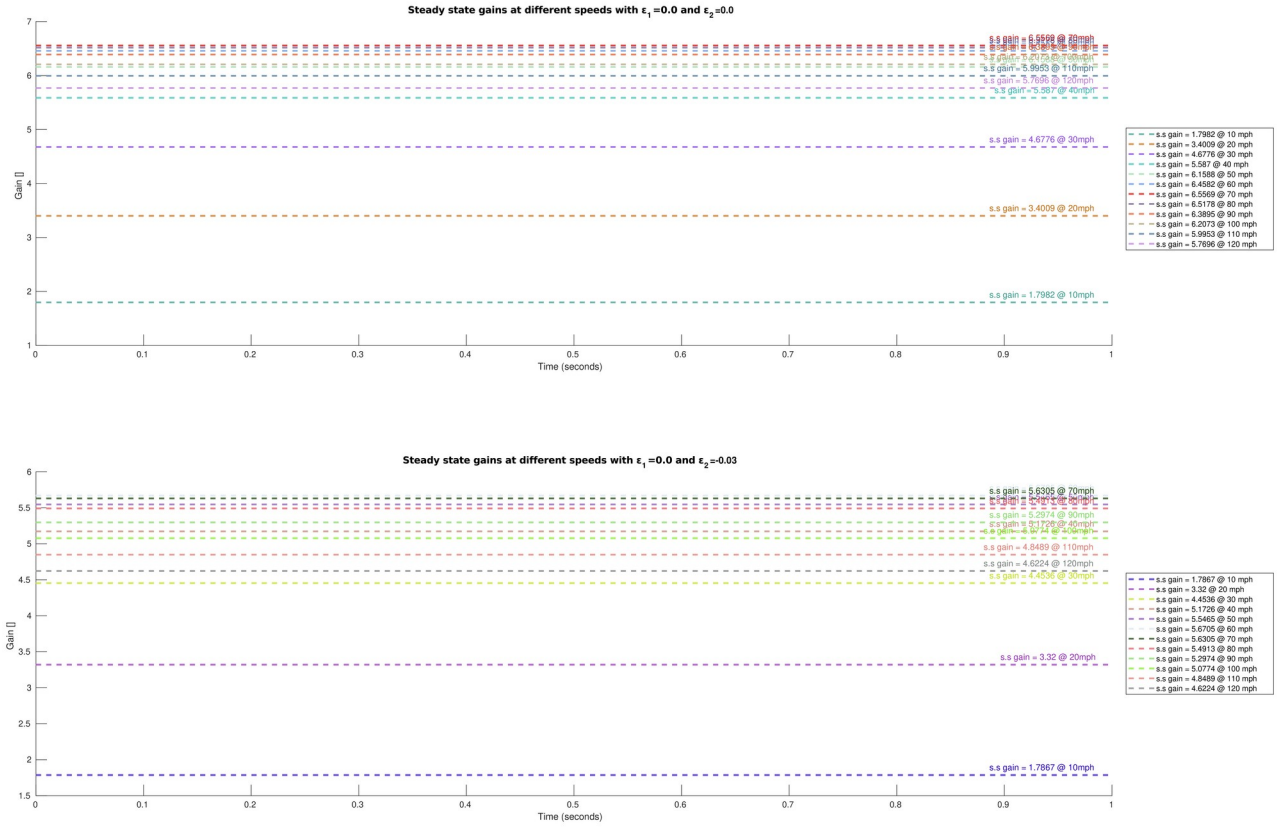


Figure 11: Shows the steady state responses for different speeds. The top plot corresponds to a configuration with no roll steer and the bottom one has ε_1=0 and ε_2=-0.03.

Analysis P2Q1: From Figure 11, it can be observed that adding a negative roll steer at the rear will produce more tendency to understeer on the vehicle as the steady-state gains are going to be lower compared to the configuration with no roll steer. This makes sense as a negative roll steer in the rear will produce less attack angles at the rear, producing less lateral force.

2. Using the 3-DoF equations of motion, simulate the vehicle response to a steering input. Determine the steering input required to result in a 400 ft radius turn (hint: knowing the forward speed and radius of a circle, the required yaw rate can be calculated). Repeat the result at increments of 10 mph from 10 to 120 mph. Compare the results with the steady-state results from 1) above. As before, we are using this step to validate your simulation model.

The equations of motion for a 3 DOF vehicle with roll and only front steering are:

1. $$m\dot{v}+m_s(-\ddot{\phi}h)=(\frac{-C_a}{u})v+(\frac{-C_b}{u}-mu)r+(C_{\phi,1}+C_{\phi,2})\phi+C_1\delta_1$$

2. $$I\dot{r}-m_s hc\ddot{\phi}=(\frac{-C_b}{u})v+(\frac{-C_c}{u})r+[x_1C_{\phi,1}+x_2C_{\phi,2}]\phi+x_1C_1$$
3. $$I_{x,\phi}\ddot{\phi}-m_s h\dot{v}-m_s hc\dot{r}=m_s hur-D_\phi\dot{\phi}-K_\phi\phi$$

These 3 equations can be written in matrix form as

$$\begin{bmatrix}\dot{v}\\ \dot{r}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}m & 0 & -m_s h\\ 0 & I_z & -m_s hc\\ -m_s h & -m_s hc & I_{x,\phi}\end{bmatrix}^{-1}\cdot\left(\begin{bmatrix}\frac{-C_a}{u} & \frac{-C_b}{u}-mu & 0 & C_{\phi,1}+C_{\phi,2}\\ \frac{-C_b}{u} & \frac{-C_c}{u} & 0 & x_1C_{\phi,1}+C_{\phi,2}\\ 0 & m_s hu & -D_\phi & -K_\phi\end{bmatrix}\cdot\begin{bmatrix}v\\ r\\ \phi\end{bmatrix}+\begin{bmatrix}C_1\\ x_2C_2\\ 0\end{bmatrix}\cdot[\delta]\right)\ ;$$

or in a simpler form,

$$\dot{z}(t)=M^{-1}(A\cdot z(t)+B\cdot\delta(t))\ .$$

The system can be discretized using Euler's method by applying

$$z_k=z_{k-1}+(dt\cdot M^{-1}(Az_{k-1}+Bu_{k-1}))\ .$$

Notice also that the roll angle can be added to this system as a 4th state by simply adding a discrete integral of the 3rd state.

Matlab code was written to solve this exercise (see Code2Q2_1 and Code2Q2_2). This code generates Figure 12 with the configuration without roll steer and Figure 13 with the configuration with a rear roll steer of -0.03.

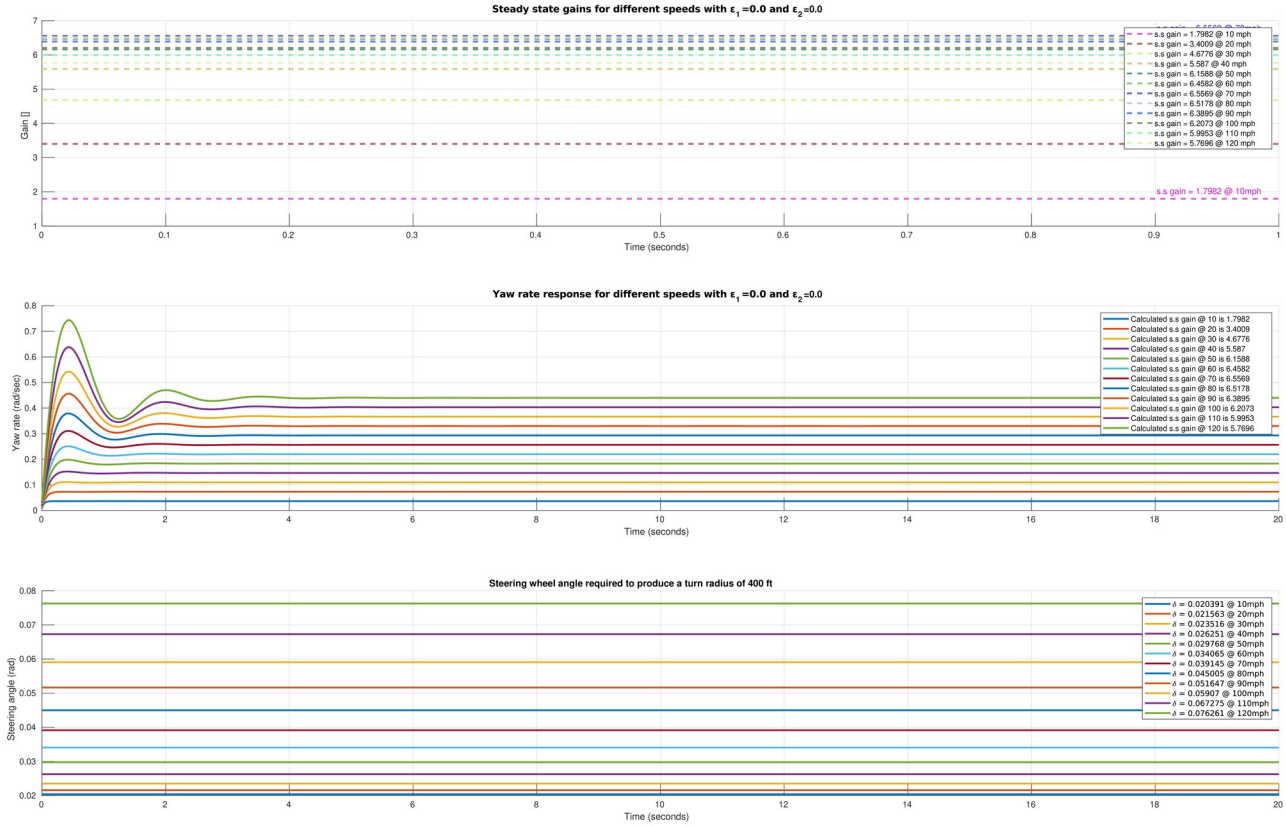


Figure 12
Shows the steady state gains, the yaw rate time response and the steering angle needed to produce a 400 ft turn radius with ε_1=0 and ε_2=0.

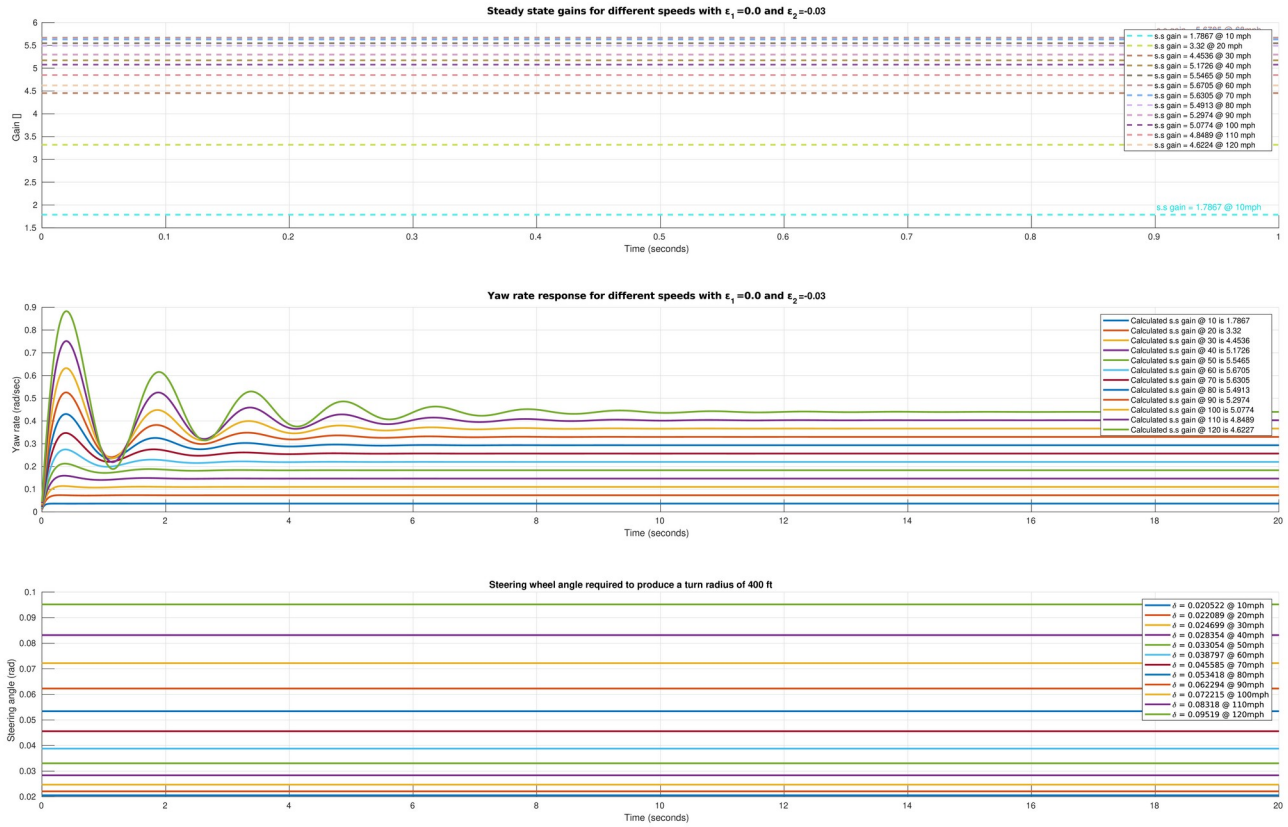


Figure 13: Shows the steady state gains, the yaw rate time response and the steering angle needed to produce a 400 ft turn radius with ε_1=0 and ε_2=-0.03.

Analysis P2Q2: The plots shown in Figure 12 and Figure 13 reinforce Analysis P2Q1 as it shows that the steady state gain is lower with ε_2=-0.03 since it induces more understeer. Moreover, Figure 13 shows a more oscillatory transient response compared to Figure 12, this is due to the fact that the configuration with more tendency to understeer will have higher complex conjugate absolute values which are a direct cause of oscillations in the transient response; nevertheless, these oscillations will only occur after the characteristic speed.

3. Use the steering input calculated in 4) from Part 1 as an input to your simulation. Repeat results for the following combinations of roll steer coefficients:

Configuration	ε_1	ε_2
1	0.04	0.04
2	0.04	0.0
3	0.04	-0.04
4	0.0	0.04
5	0.0	0.0
6	0.0	-0.04
7	-0.04	0.04
8	-0.04	0.0
9	-0.04	-0.04

Present significant simulation and steady state yaw rate response results as appropiate.

The code Code2Q3 was written in Matlab to be able to simulate the 3 degrees of freedom bicycle model with roll for different front and rear roll steer values ε_1 and ε_2 to analyze the different configurations proposed. This code contain the necessary functions to run the simulation for the different proposed configurations.

Configuration #1: ε_1=0.04 and ε_2=0.04

Code Code2Q3_1 was written in Matlab to call the function *test_model* with the required configuration.

Notice that the code also calculates an average for the steady state gains for the different velocities the simulations is tested at; this is going to be useful later at the end of this chapter to create a table summarizing the results which will help making good conclusions on what are the effects of varying ε_1 and ε_2.

This code generates the plot shown in Figure 14.

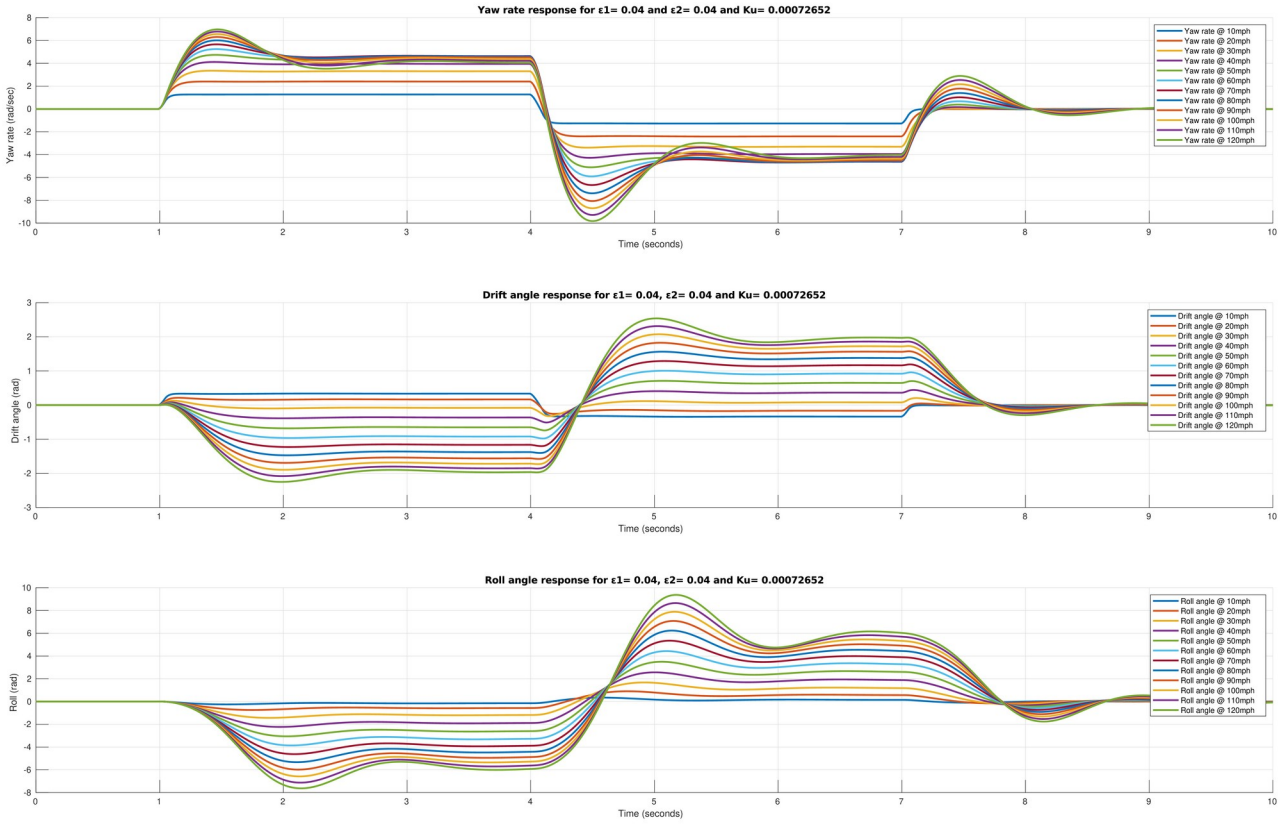


Figure 14: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.04 and ε_2=0.04.

Configuration #2: ε_1=0.04 and ε_2=0.0

Code2Q3_2 was written in Matlab to call the function *test_model* with the required configuration:

This code generates the plot shown in Figure 15.

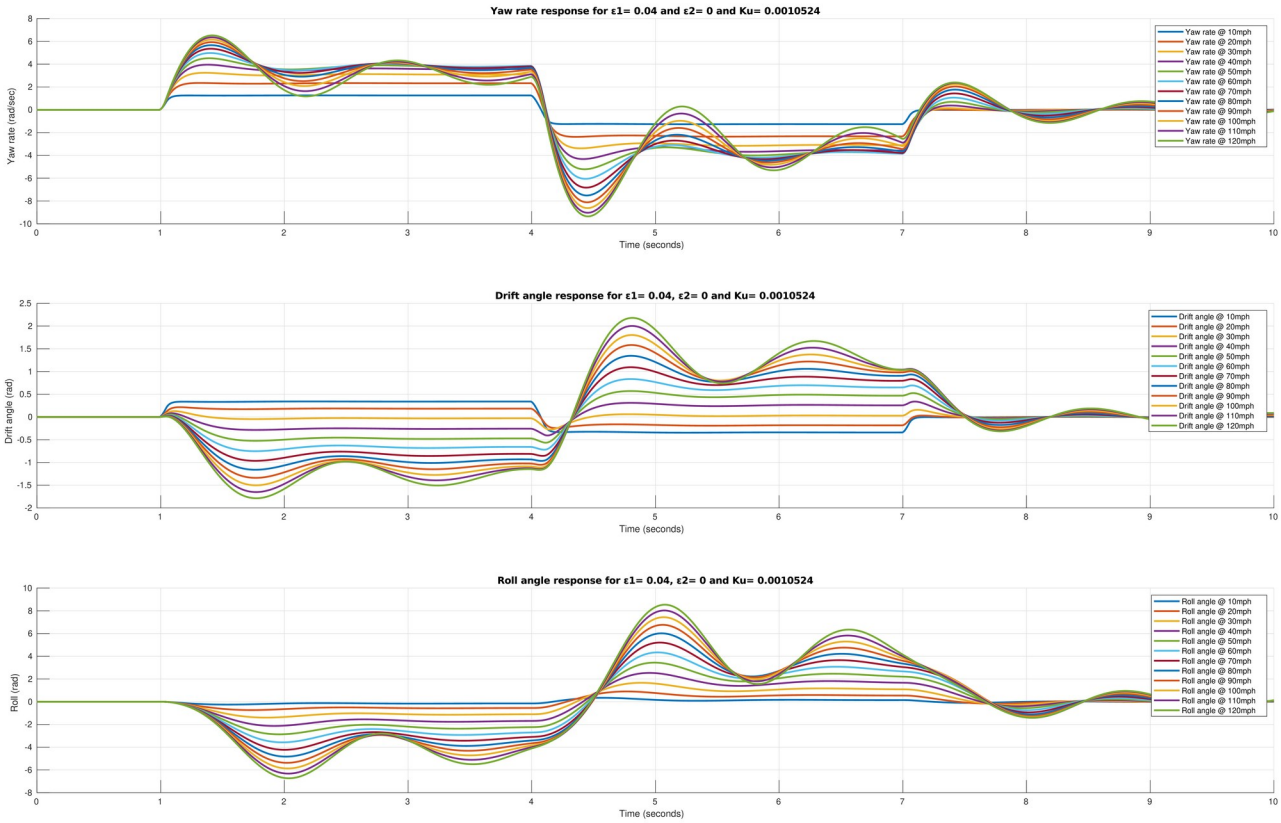


Figure 15: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.04 and ε_2=0.0

Configuration #3: ε_1=0.04 and ε_2=-0.04

Code2Q3_3 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 16.

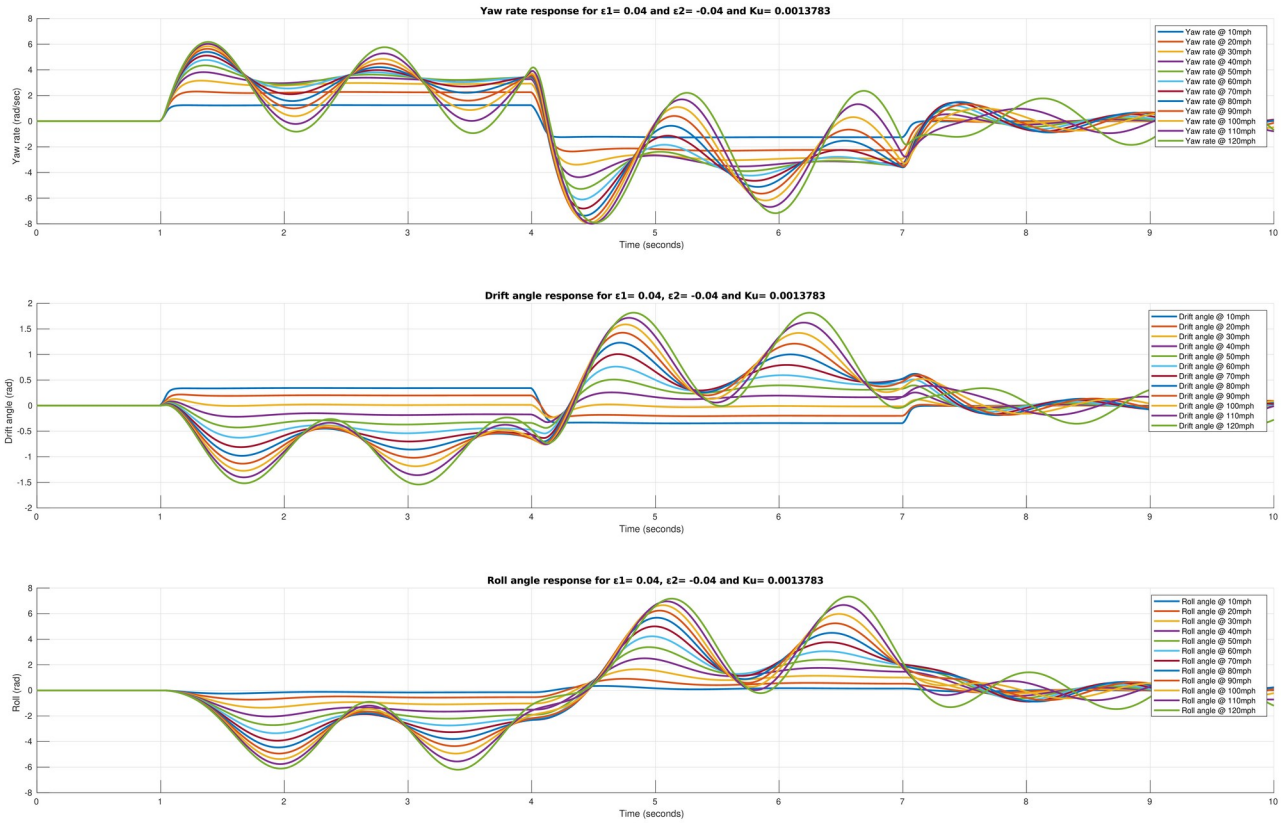


Figure 16: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.04 and ε_2=-0.04

Configuration #4: ε_1=0.0 and ε_2=0.04

Code2Q3_4 was written in Matlab to call the function *test_model* with the required configuration.

This code generates the plot shown in Figure 17.

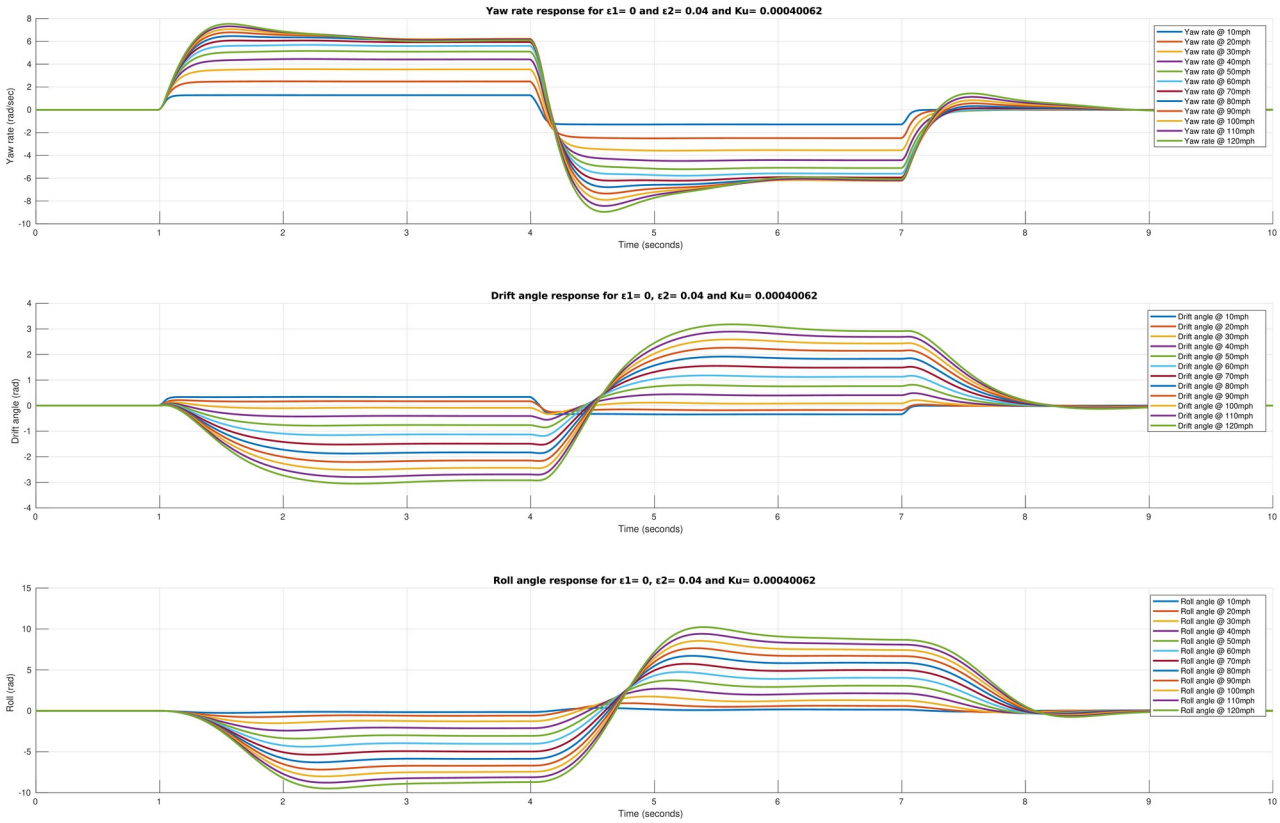


Figure 17: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.0 and ε_2=0.04

Configuration #5: ε_1=0.0 and ε_2=0.0

Code2Q3_5 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 18.

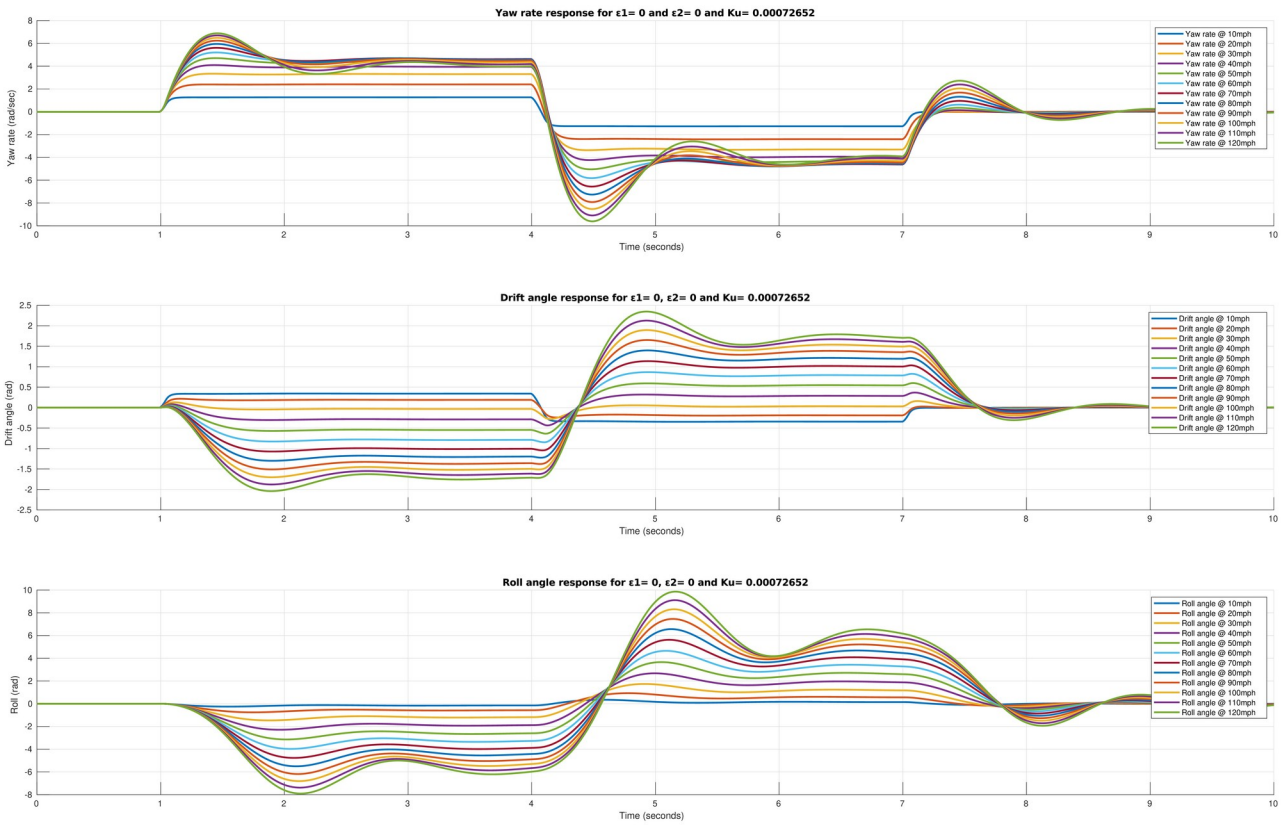


Figure 18: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.0 and ε_2=0.0

Configuration #6: ε_1=0.0 and ε_2=-0.04

Code2Q3_6 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 19.

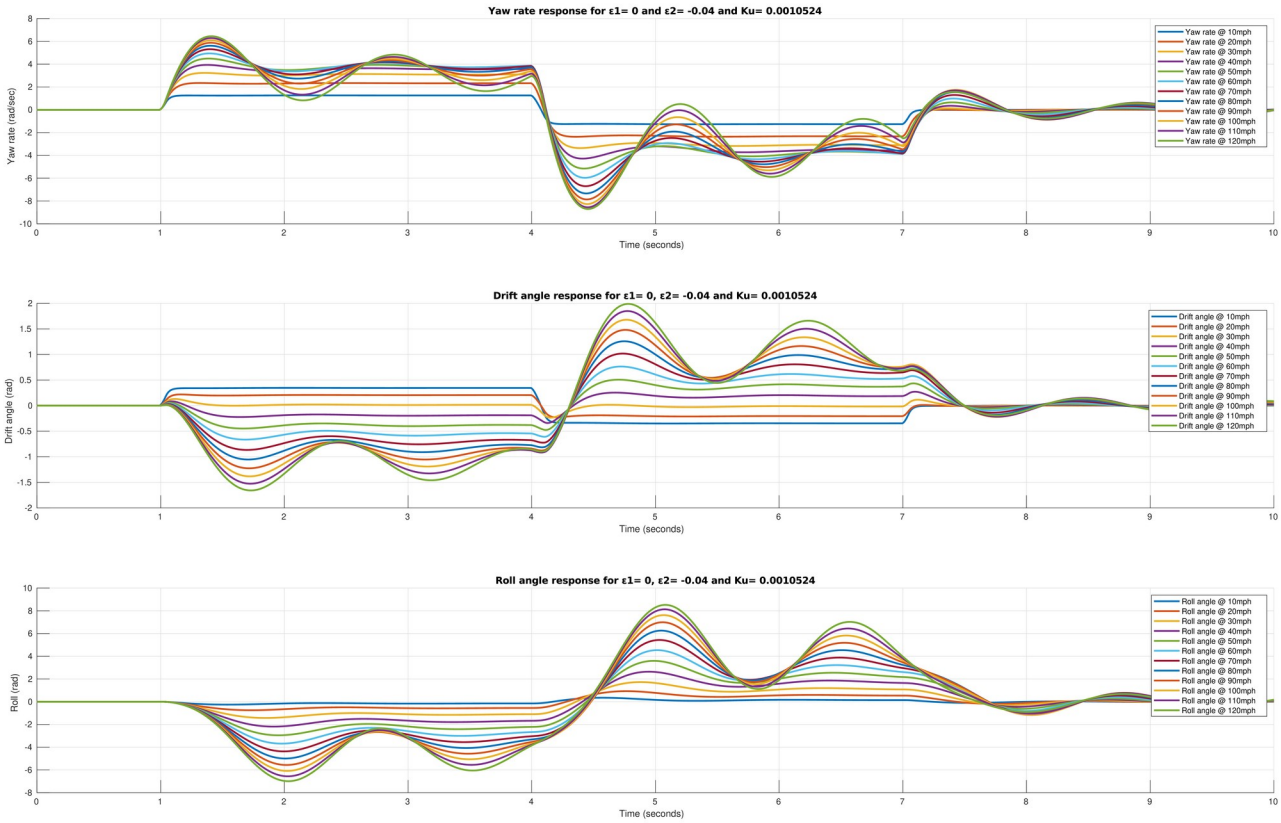


Figure 19: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=0.0 and ε_2=-0.04

Configuration #7: ε_1=-0.04 and ε_2=0.04

Code2Q3_7 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 20.

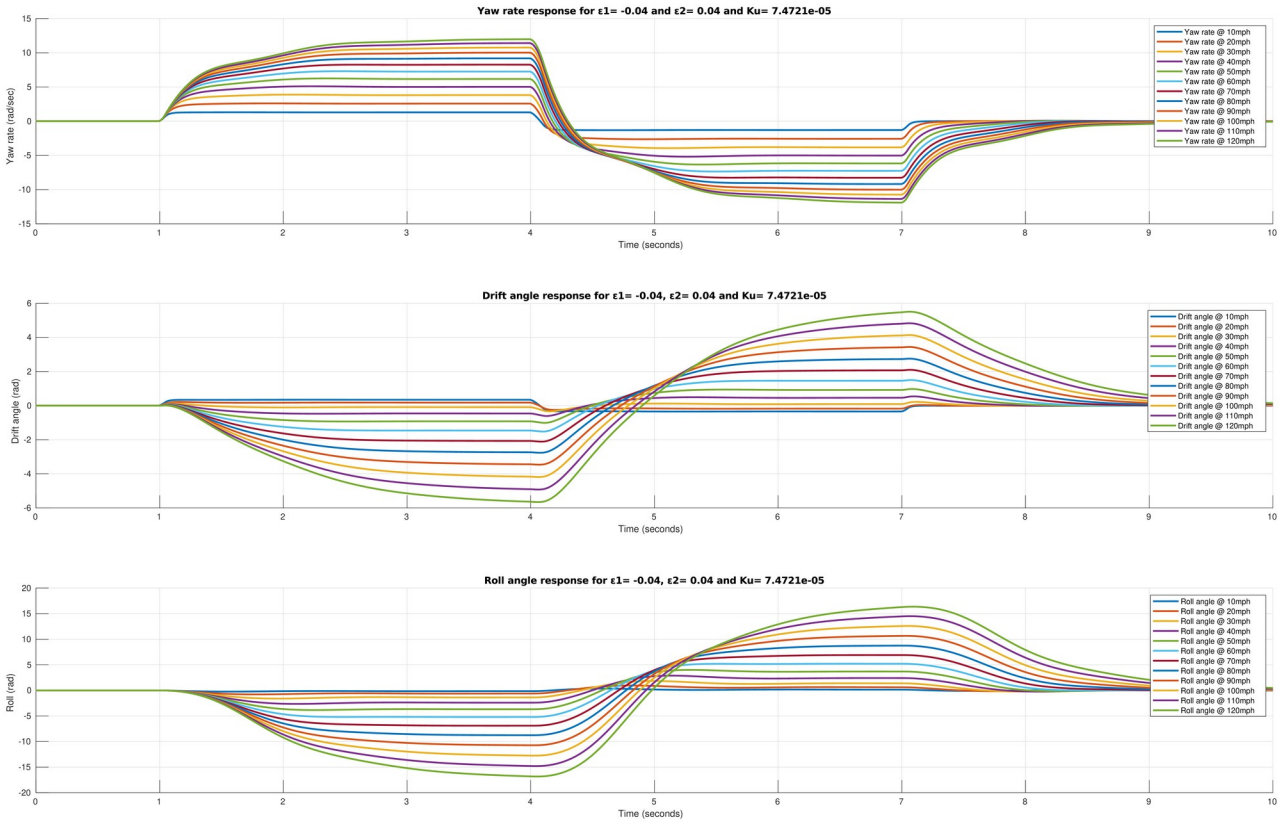


Figure 20: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=-0.04 and ε_2=0.04

Configuration #8: ε_1=-0.04 and ε_2=0.0

Code2Q3_8 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 21.

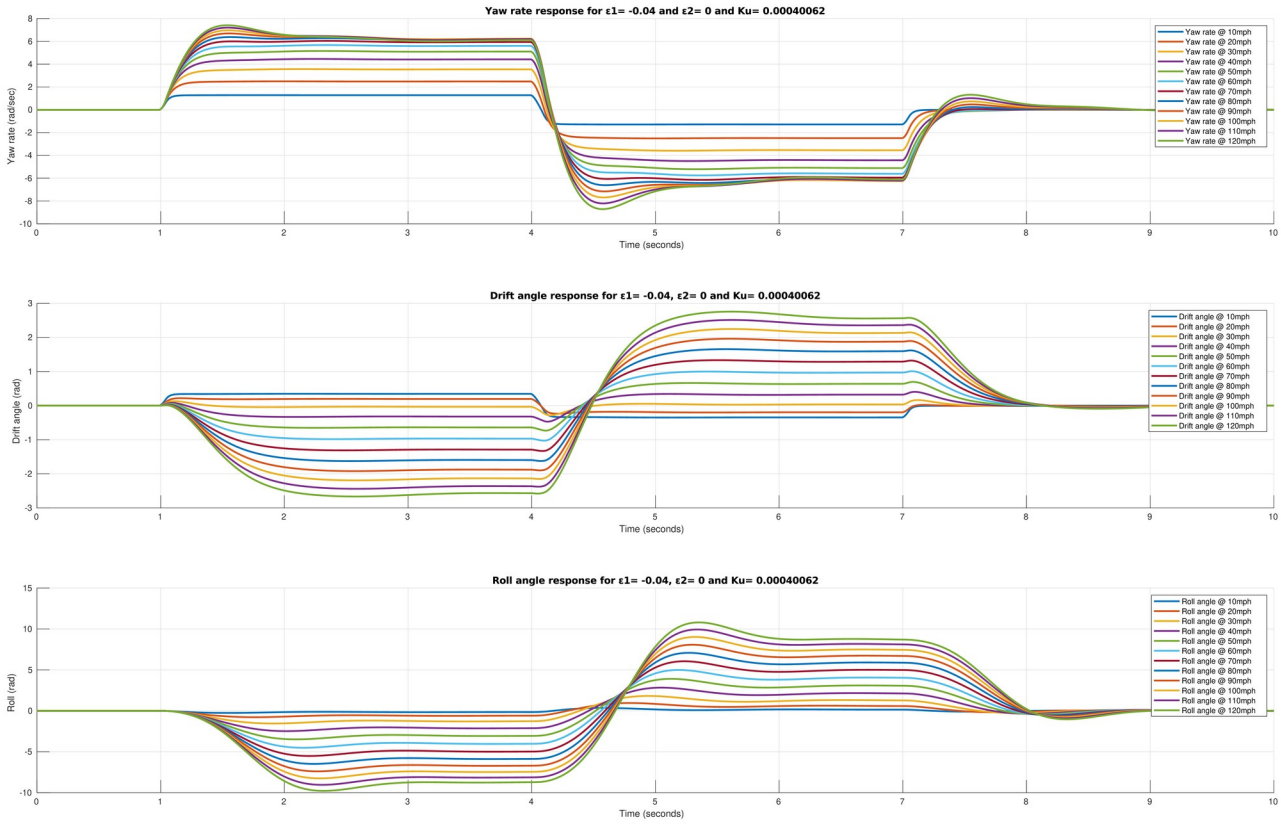


Figure 21: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=-0.04 and ε_2=0.0

Configuration #9: ε_1=-0.04 and ε_2=-0.04

Code2Q3_9 was written in Matlab to call the function *test_model* with the required configuration. This code generates the plot shown in Figure 22.

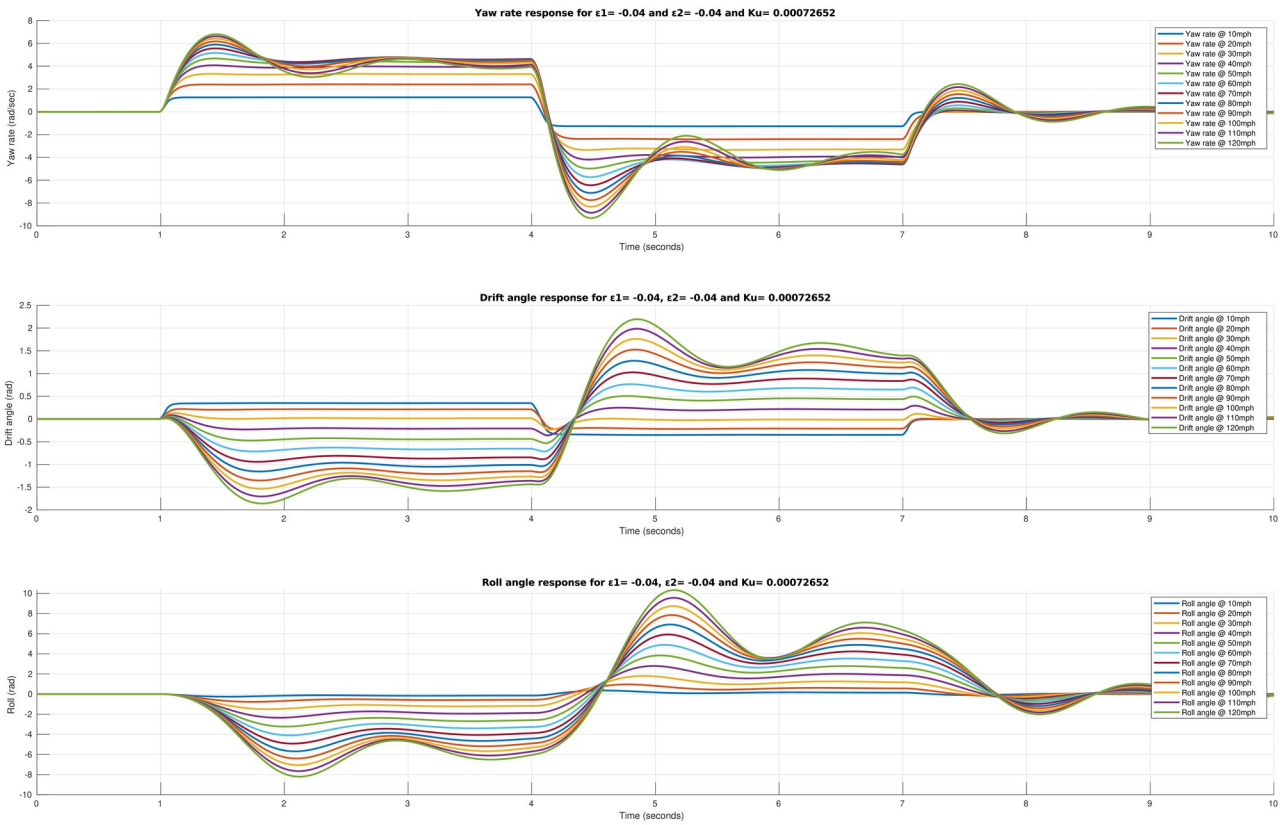


Figure 22: Shows the time response for the states yaw rate, drift angle and roll angle for different speeds with ε_1=-0.04 and ε_2=-0.04

From Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, , Figure 21 and Figure 22 the following table can be constructed summarizing relevant results:

Configuration	ε_1	ε_2	K_u	Steady-state gain	Characteristic speed (mph)
1	0.04	0.04	0.00073	5.4598	71.4
2	0.04	0.0	0.00105	4.5511	59.5
3	0.04	-0.04	0.00138	3.9362	51.9
4	0.0	0.04	0.00040	6.9869	94.4
5	0.0	0.0	0.00073	5.4598	71.4
6	0.0	-0.04	0.00105	4.5511	59.5
7	-0.04	0.04	7.4721e-5	10.3652	223.1
8	-0.04	0.0	0.00040	6.9869	96.4
9	-0.04	-0.04	0.00073	5.4598	71.4

Table 2: Shows a summary of the results of varying ε_1 and ε_2, this is useful to analyze how varying these parameters affect important handling characteristics such as the understeering coefficient, the steady-state gain and the characteristic speed.

Analysis P2Q3: From Table 2, it can be seen that all the 9 configurations induce understeer in the vehicle, but some configurations induce more understeering than others. The closest configuration we have to **neutral steer** is **configuration 7,** this is why this configuration has the highest steady-state gain; this configuration shows how significant is the effect of the roll steer coefficients ε_1 and ε_2 as its gain is significantly (almost double roughly) higher compared to the other configurations.

Moreover, **configuration 7** presents less oscillations in the time responses of the state variables compared to the others as the characteristic speed is much higher (around 223 mph) which means that the system will have only real eigenvalues at speeds equal or less than 223.1 mph.

From Table 2 it can also be concluded that the effect of decreasing ε_2 and leaving ε_1 is an increase in the understeering coefficient, this means that this is adding more understeering behavior to the vehicle; in **configurations 1-3, 4-6 and 7-9** this increase in the understeering coefficient K_u can be appreciated. The effect of decreasing ε_1 is a decrease in the understeering coefficient K_u, this means that this induces more oversteering to the vehicle handling characteristics.

4. Comment on and interpret your results.

1. Analysis P2Q1 presents the analysis specific to Part 2 - Question 1.

2. Analysis P2Q2 presents the analysis specific to Part 2 - Question 2.

3. Analysis P2Q3 presents the analysis specific to Part 2 - Question 3.

Part 3: 3-DoF Vehicle (Non-Linear)

The complete code for this part can be found in the appendix in Code3.

1. Return to the baseline case of $\varepsilon_1=0$; $\varepsilon_2=-0.03$. Add the nonlinear tire model. You will have to calculate the tire loads at each time step, knowing that the moments from roll stiffness and roll damping on the sprung mass are reacted by vertical forces through the contact patch of the tires, and add to or subtract from the static tire loads. Compare and interpret your results above with the corresponding results from (Part 2: 3-DoF Vehicle (Linear)).

The roll stiffness and roll damping moments are reacted at the contact patch, this physical behavior can be expressed by $M_{roll}=-K_\phi\phi-D_\phi\dot{\phi}$; a 50/50 static weight bias is assumed, hence, the axle weights are given by $W_1=\frac{W}{2}$ and $W_2=\frac{W}{2}$.

The dynamic loading at each corner will be given by the equations

$$W_{1r}=\frac{W_1}{2}-\frac{M_{roll}}{4h}=\frac{W_1}{2}-\left(\frac{1}{4h}\right)\left(-K_\phi\phi-D_\phi\dot{\phi}\right) \quad , \quad W_{1l}=\frac{W_1}{2}+\frac{M_{roll}}{4h}=\frac{W_1}{2}+\left(\frac{1}{4h}\right)\left(-K_\phi\phi-D_\phi\dot{\phi}\right) \quad ,$$

$$W_{2r}=\frac{W_2}{2}-\frac{M_{roll}}{4h}=\frac{W_2}{2}-\left(\frac{1}{4h}\right)\left(-K_\phi\phi-D_\phi\dot{\phi}\right) \quad \text{and} \quad W_{2l}=\frac{W_2}{2}+\frac{M_{roll}}{4h}=\frac{W_2}{2}+\left(\frac{1}{4h}\right)\left(-K_\phi\phi-D_\phi\dot{\phi}\right) \quad .$$

The function *simulate_bike_3dof* was modified to add weight transfer effects, a new function was created called *simulate_non_linear_bike_3dof*; this should result in a more realistic simulation as roll dynamics are very dependent on weight transfer as it produces moments around the roll axis of the car.

Code3Q1 was developed in Matlab to solve this exercise. This code generates the plots shown in Figure 23 and Figure 24 in which the states and weight transfer time responses are shown respectively.

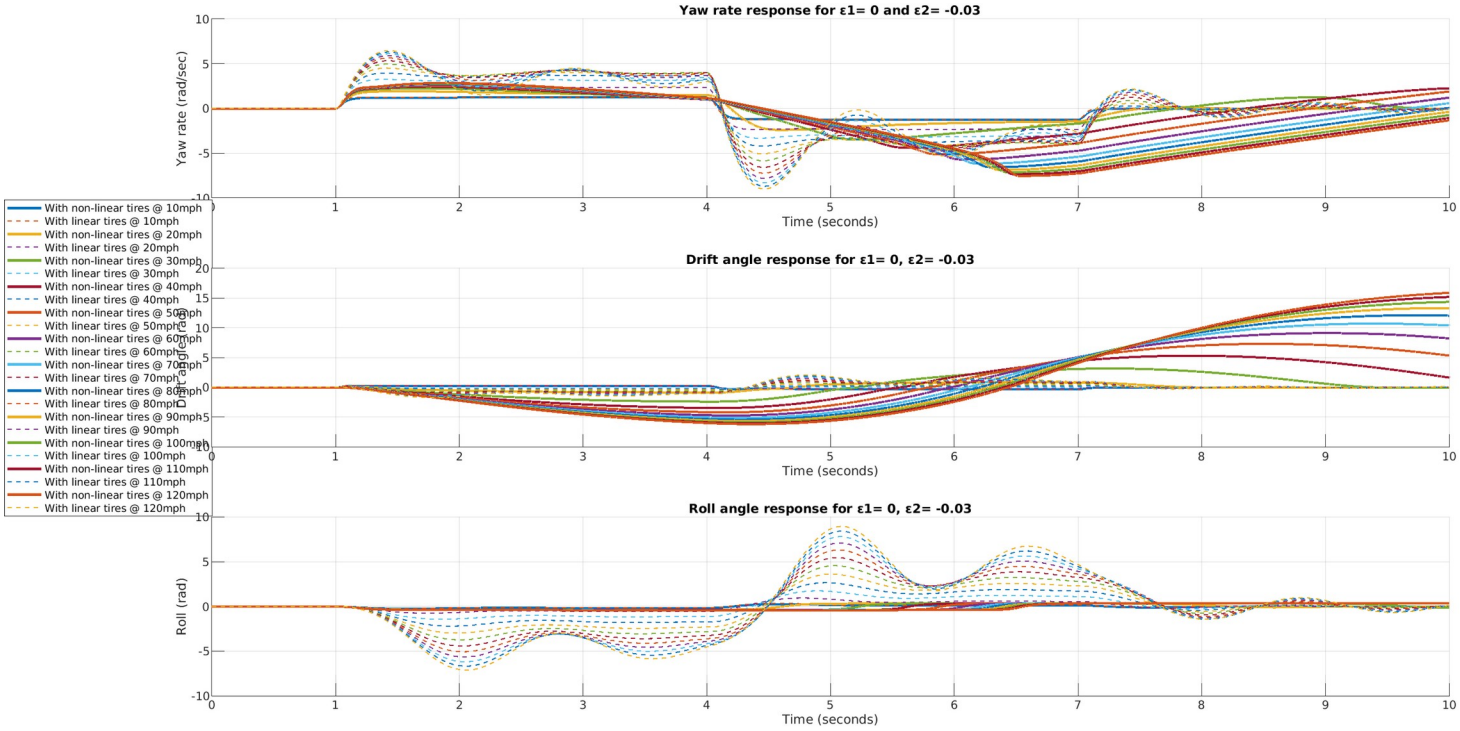


Figure 23: Shows the time responses for the yaw rate, drift angle and roll angle.

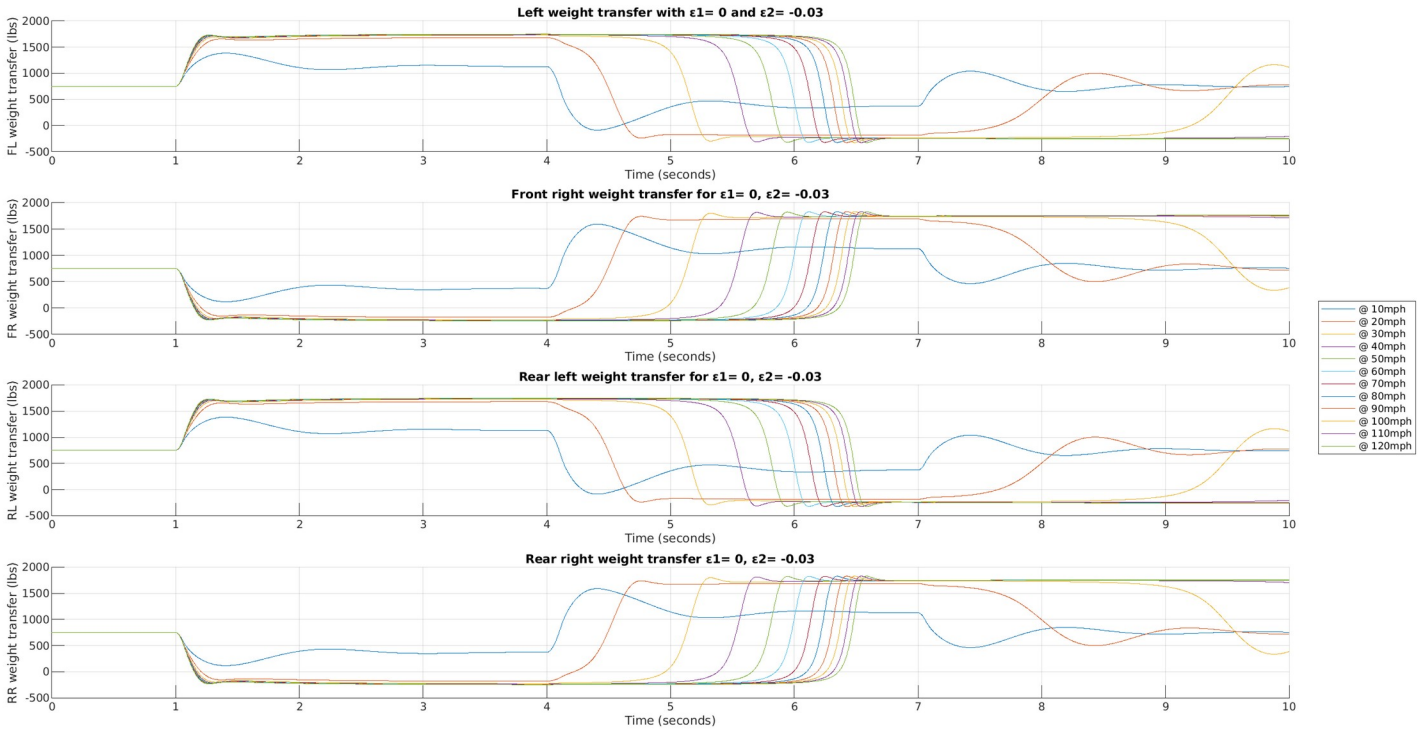


Figure 24: Shows the weight transfer time responses for the front right, front left, rear right and rear left tires.

Analysis P3Q1: Given Figure 23 and Figure 24, it can be seen that weight transfer has significant effects on vehicle handling. Notice that the steering input used here (see Figure 3) is very aggressive as the steering wheels go from 0 to 45 degrees and then from 45 degrees to -45 degrees very rapidly, this causes the roll angles for the linear tires in Figure 23 to be very high, values that don't make physical sense.

Nevertheless, the non-linear version can saturate the roll angle as the limits of the weight transfer behavior is when the vehicle tips to either side; this produces a more realistic behavior as vehicle roll is directly related to the roll moments the car is experiencing in the current moment and these roll moments are produced mostly by weight transfer. This is why the roll angle has more realistic final values on the non-linear version.

Notice that on Figure 24 the weight transfer starts at 750 lbs for the 4 tires, this makes sense as the weight bias was assumed to be 50/50, thus the weight is evenly distributed among 4 tires. Notice that the weight transfer functions have a positive and negative saturation values (where the curves flatten) indicating the limit for the weight transfer, which will also produce limits on the roll angle.

The system is now excited with lower steering angle signals to provide a better view of the states operating at points that don't roll the car that much. The signal in Figure 3 is scaled by 0.05, and the plots are generated again as seen in Figure 26. From these plots, it can be seen that the roll angles tend to be less overall compared to the linear tires version which supports the conclusions already made on Figure 23 and Figure 24.

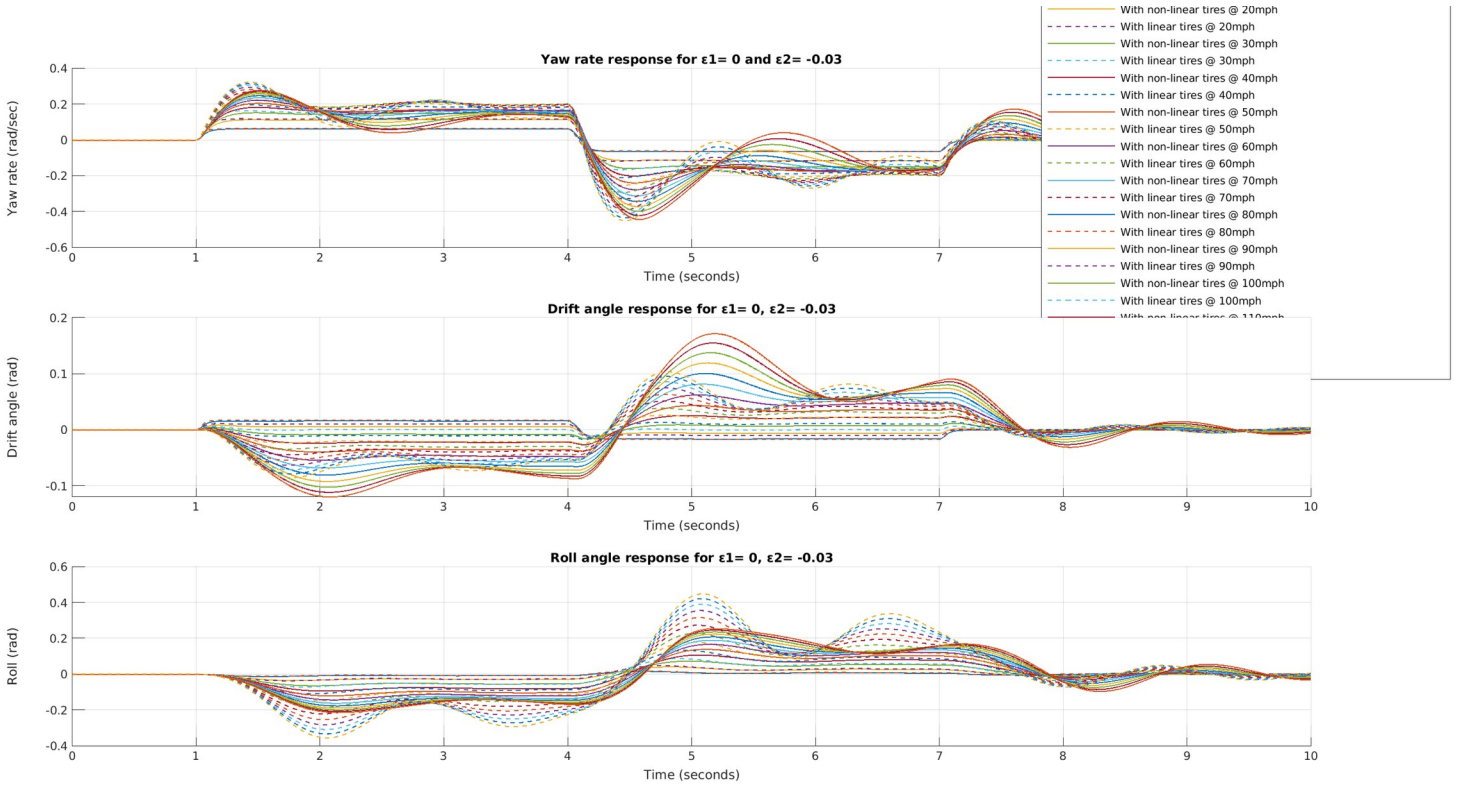


Figure 25: Shows the states time responses for a scaled version of the input shown in Figure 3 by a factor of 0.05. This allows a more detailed comparison of the linear and non-linear model.

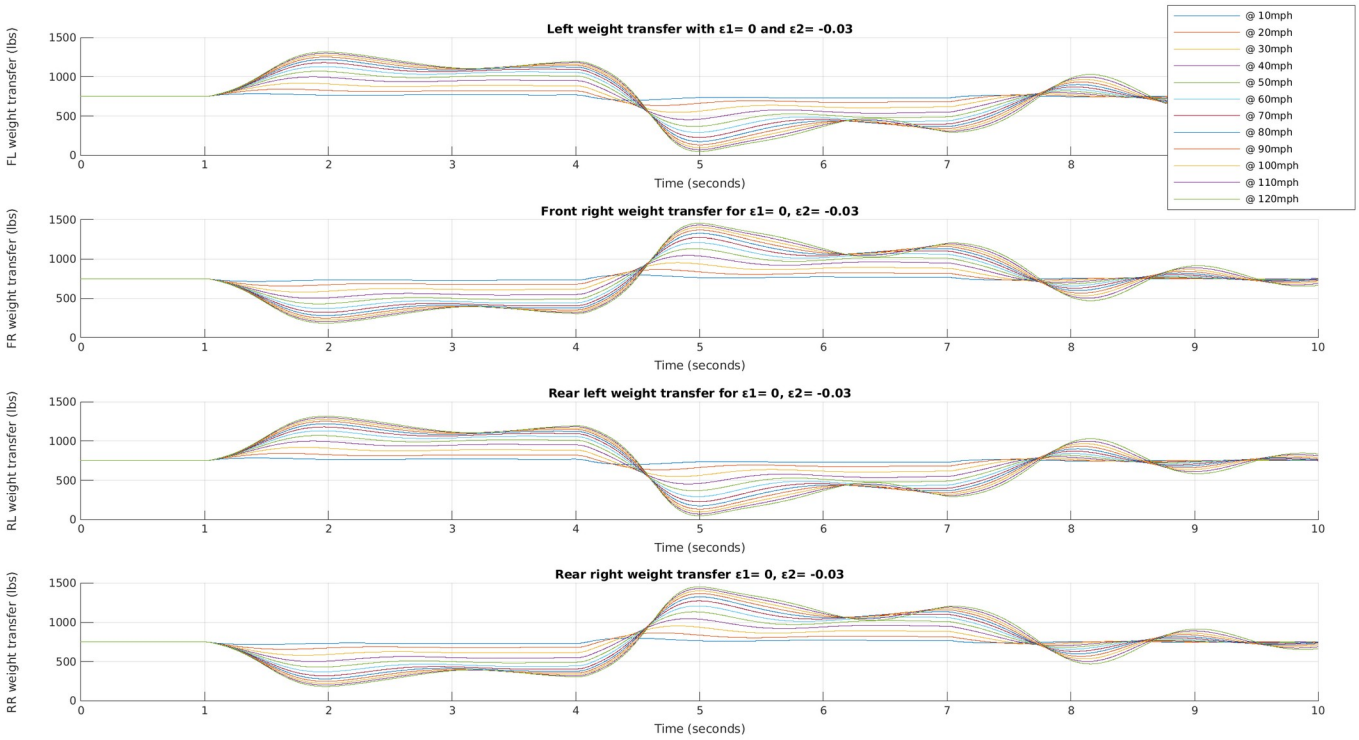


Figure 26: Shows the time response for the weight transfers of each tire using a steering input equal to a 0.05 scale version of the steering angle input signal shown in Figure 3.

2. Use your model to comment on the effect of roll stiffness and roll damping on vehicle handling. That is, vary the parameters and describe the effect. Present significant results and comment on them.

The simulation functions *simulate_non_linear_bike_3dof* and *simulate_bike_3dof* are used for this analysis. First, the damping coefficient is fixed at its nominal value and the stiffness is varied, then, the stiffness coefficient is fixed at its nominal value and the stiffness is varied; a speed of 60 mph was selected for the analysis and the data is plotted in comparison plots useful to draw conclusions from it. Code3Q2 solves this exercise and generates the plots shown in Figure 27 and Figure 28.

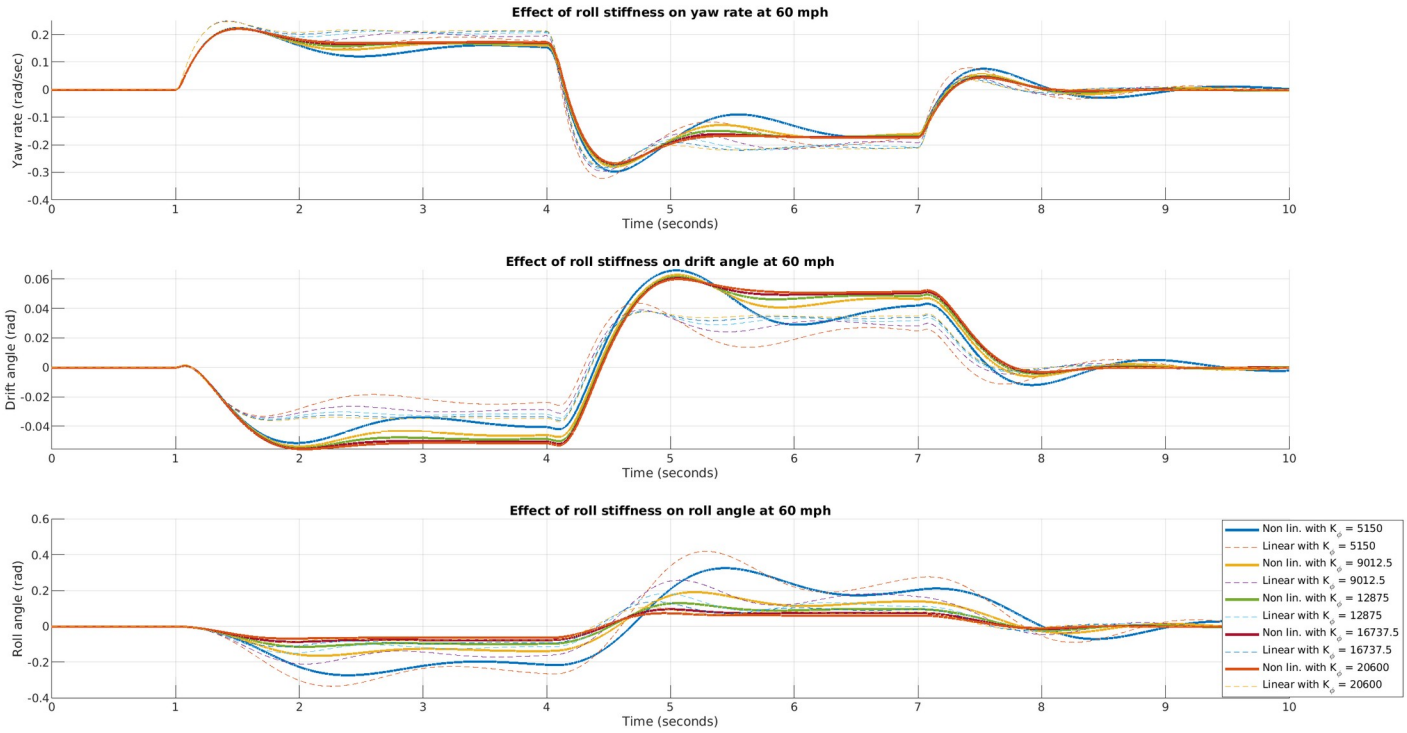


Figure 27: Shows the time responses of the yaw rate, drift angle and roll angle states for different K_{φ} at 60mph.

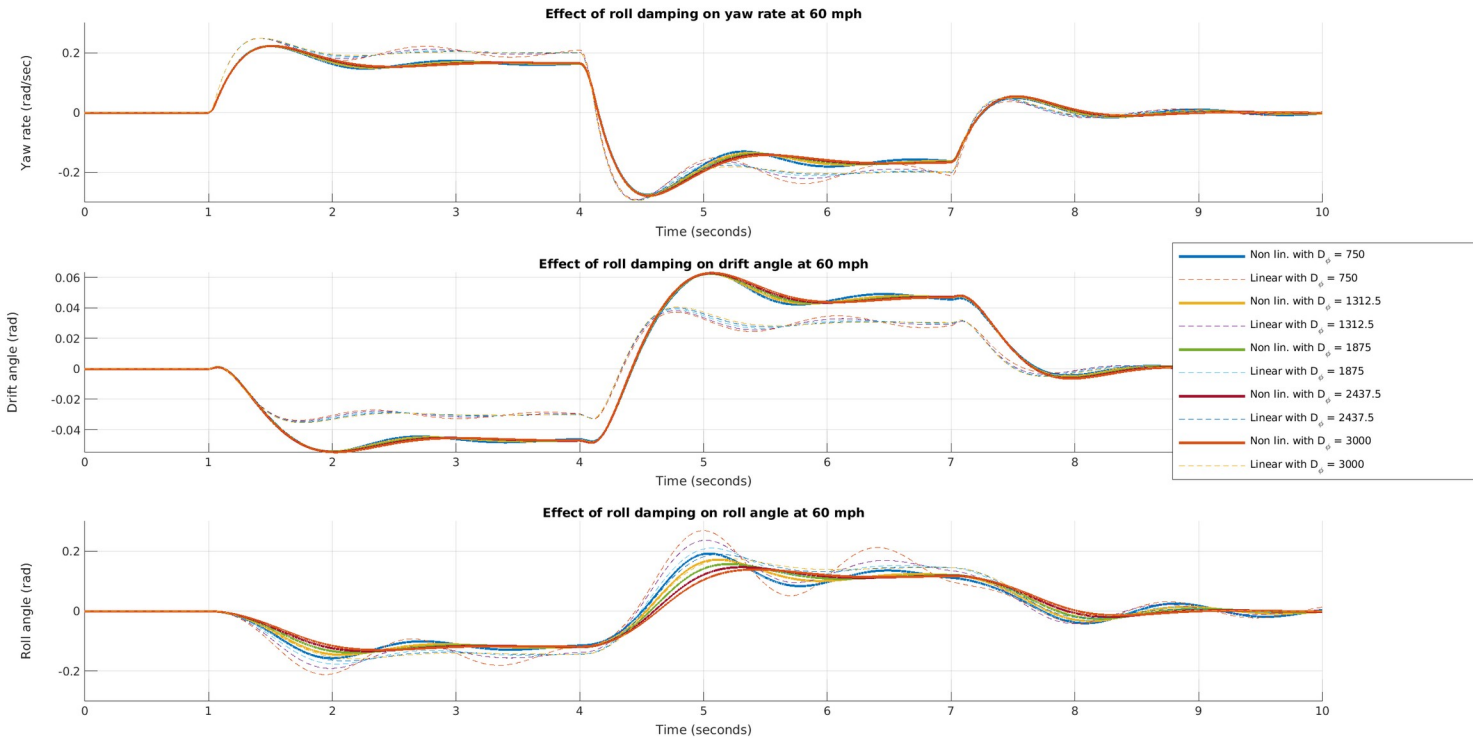


Figure 28: Shows the time responses for the yaw rate, drift angle and roll angle for varying roll damping coefficients D_{φ}.

Analysis P3Q2:

On Figure 27 it can be seen that increasing K_φ in both the linear model and the non-linear model tend to reduce the oscillations and make the settling times of the states faster, especially in the roll mode as a stiffer roll will require more moment to produce movement. Overall, increasing K_φ seems to be a good alternative to minimize roll movement.

On Figure 28 it is possible to observe that increasing the damping coefficient of roll D_φ decreases the oscillations but makes the settling times slower for the roll mode. This is the expected behavior of a damper as a trade-off exists when selecting a linear damper given between system's bandwidth and oscillations induced by the damper.

In conclusion, both K_φ and D_φ can be used as important physical parameters that affect the roll mode and given that they affect the roll mode they also affect the handling characteristics given the coupling of the system. Reducing roll movements of the vehicle through K_φ and D_φ can be useful to maintain stability and grip.

Part 4: Summary and Conclusions

In summary, of this

5. APPENDIX:

Part 1 Code:

```
%% Initial setup:
%% Load some initial values:
% Conversion factors:
deg2rad = pi / 180;
rad2deg = 180 / pi;
in2ft = 1 / 12;
ft2in = 12;
mph2ftps = 5280 / 3600;
ftps2mph = 3600 / 5280;
% Bicycle model parameters:
W = 3000; % lbs
Ws = 2700; % lbs
g = 32.174; % ft/sec^2
x1 = 3.5; % ft
x2 = -4.5; % ft = -1.0; % ft
h = -1.0; % ft
track_width = 6.0; % ft
Iz = 40000 / g; % lbs*ft^2
Ix = 15000 / g; % lbs*ft^2
c = 0.5; % ft
dl_phi_f = 8000; % lbs*ft
dl_phi_r = 5000; % lbs*ft
dl_dphi_f = 1000; % lbs*ft
```

```
dl_dphi_r = 500; % lbs*ft
p = 12*in2ft; % ft
d = 12*in2ft; % ft
gear_ratio = 15; % []
efficiency = 1.0; % Steering gearbox efficiency
Ks = 10*in2ft*rad2deg; % (in*lbs/deg)*(ft/in)*(deg/rad) -> ft*lbs / rad
tm = 3*in2ft; % in*(ft/in)-> ft - Pneumatic trail
% Masses:
m = W / g;
ms = Ws / g;
% u = 30; % ft/sec
C1 = 2*140*180/pi; %*rad2deg; % lbs/deg * (deg/rad) -> lbs / rad
C2 = 2*140*180/pi; % *rad2deg; % lbs/deg * (deg/rad) -> lbs / rad
l2 = x1 - x2; % Wheelbase
dt = 0.01;
t_initial = 0;
t_final = 5;
%%  Code1Q1:
%% 1. Develop a base 2-DoF linear state space yaw rate model and calculate the system eigen values
% when forward speed is 30 and 60 mph
u30 = 30*mph2ftps;
u60 = 60*mph2ftps;
A30 = [
(-C1 - C2)/(m*u30), (((-x1*C1 - x2*C2)/(m*u30^2)) - 1);
(-x1*C1 - x2*C2)/(Iz), (-x1*x1*C1 - x2*x2*C2)/(Iz*u30);
]
A60 = [
(-C1 - C2)/(m*u60), ((-x1*C1 - x2*C2)/(m*u60^2)) - 1;
(-x1*C1 - x2*C2)/(Iz), (-x1*x1*C1 - x2*x2*C2)/(Iz*u60);
]
eg_30mph = eig(A30)
eg_60mph = eig(A60)
%% Code1Q2:
%% 2. Using the steady-state yaw rate response, construct plots from 0 to 120 mph (i.e. r/    ) for various
% speeds, every 10 mph or so.
speeds = linspace(10, 120, 12)*mph2ftps;
delta2r_gain_arr = zeros(1, length(speeds));
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure;
hold on;
grid on;
title('Yaw rate steady state gains for different speeds')
xlabel('Time (seconds)');
ylabel('Gain (r/   )');
xlim([0, 5]);
ylim([0, 10.0]);
K_understeer = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
u_char = sqrt((x1-x2)/K_understeer);
legend_arr = cell(1, length(speeds));
for i = 1:length(speeds)
u = speeds(i);
delta2r_gain_arr(i) = u / (l2 + u*u*K_understeer);
% plot each gain in an xy plot:
yline(delta2r_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_gain_arr(i)) ' @ ' num2str(u*ftps2mph) 'mph'], 'Color', rand(1,3), 'LineWidth',
2);
% plot(t, delta2r_gain_arr(i)*ones(length(t)), '--', 'LineWidth', 2);
legend_arr{i} = [num2str(u*ftps2mph) ' mph'];
hold on;
end
legend(legend_arr{:}, 'Location', 'eastoutside');
hold off;
%% Code1Q3:
%% 3. Using the 2-DoF equations of motion, simulate the vehicle response to a steering input.
% Determine the steering input required to result in a 400 ft radius turn. (hint: knowing the forward
% speed and radius of a circle, the required yaw rate can be calculated). Repeat the result at
```

```
% increments of 10 mph from 10 to 120 mph. Compare the results with the steady-state results from
% 2) above. We are using this step to validate your model.
radius = 400; % ft
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
% Time domain simulation:
t_initial = 0; % sec
t_final = 5; % sec
dt = 0.01; % sec
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure;
subplot(3,1,1);
title('Steady state gains at different speeds');
xlabel('Time (seconds)');
ylabel('Gain []');
hold on;
grid on;
subplot(3,1,2);
title('Yaw rate response at different speeds');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
hold on;
subplot(3,1,3);
title('Steering angle input at different speeds');
xlabel('Time (seconds)');
ylabel('Steering angle (rad)');
hold on;
legend_gains_arr = cell(1, length(speeds));
legend_yaw_rate_arr = cell(1, length(speeds));
legend_steering_angle_arr = cell(1, length(speeds));
for i = 1:length(speeds)
u = speeds(i);
subplot(3,1,1);
yline(delta2r_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_gain_arr(i)) ' @ ' num2str(u*ftps2mph) 'mph'], 'Color', rand(1,3), 'LineWidth',
2);
% plot(t, delta2r_gain_arr(i)*ones(length(t)), '--', 'LineWidth', 2);
legend_gains_arr{i} = ['s.s gain = ' num2str(delta2r_gain_arr(i)) ' @ ' num2str(u*ftps2mph) 'mph'];
subplot(3,1,2);
delta2 = (1 / radius)*(l2 + (u*u*K_understeer))*ones(1, length(t));
states_arr2 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2);
plot(t, states_arr2(2,:));
legend_yaw_rate_arr{i} = ['Calculated s.s gain @ ' num2str(u*ftps2mph) ' is ' num2str(states_arr2(2, length(t))/delta2(length(t)))];
subplot(3,1,3);
plot(t, delta2);
legend_steering_angle_arr{i} = ['    = ' num2str(delta2(1)) ' @ ' num2str(u*ftps2mph) 'mph'];
end
subplot(3,1,1);
legend(legend_gains_arr);
grid on;
subplot(3,1,2);
legend(legend_yaw_rate_arr);
grid on;
subplot(3,1,3);
legend(legend_steering_angle_arr);
grid on;
hold off;
%% Code1Q4:
%% 4. Determine a particular time series of handwheel inputs. We define this input as a 0° handwheel
% input for 1 second, then a +45° handwheel input for 3 seconds, then a −45° handwheel input for 3
% seconds, then back to 0° for 3 seconds. Practically we know that the maximum handwheel input a
% driver can perform is 2 rev/sec, so modify the steering input so that the handwheel velocity is
% consistent with this value with a 180° input amplitude. Plot the time history of the steering input    .
% Time array:
t_initial = 0;
t_final = 10;
```

```
t = linspace(t_initial, t_final, (t_final - t_initial)/dt);
% Generate the specified handwheel input:
delta = zeros(1, length(t));
% Input of 0° for 1 second:
delta(1, 1:1/dt) = 0;
% Input of +45° for 3 seconds:
delta(1, 1/dt:4/dt) = 0.707;
% Input of -45° for 3 seconds:
delta(1, 4/dt:7/dt) = -0.707;
% Input of 0° for 3 seconds:
delta(1, 7/dt:length(t)) = 0;
% Plot input:
figure;
plot(t, delta);
grid on;
title('Steering input with no rate saturation');
xlabel('time (sec)');
ylabel('delta (rad)');
% Modify the steering input:
slope = 2*(2*pi); % rad/sec
% Smooth step applied at 1 second:
left_bound = 0;
right_bound = 0.707;
at = 1; % /dt;
delta_mod = slope_it_down(t, dt, at, delta, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Smooth step applied at 4 seconds:
left_bound = 0.707;
right_bound = -0.707;
at = 4; % /dt;
delta_mod = slope_it_down(t, dt, at, delta_mod, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Smooth step applied at 7 seconds:
left_bound = -0.707;
right_bound = 0;
at = 7; % /dt;
delta_mod = slope_it_down(t, dt, at, delta_mod, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Plot input:
figure;
plot(t, delta);
hold on;
plot(t, delta_mod, 'r');
grid on;
title('Steering input with smoothed angle transitions');
xlabel('time (sec)');
ylabel('delta (rad)');
legend('Original input signal', 'Modified input signal');
%% Code1Q5:
%% 5. Use the steering input calculated in 4) as an input to your simulation. Plot the time history of the
% state variables drift angle     and yaw rate  r  at 30 and 60 mph.
% Speeds of interest:
u30 = 30*mph2ftps;
u60 = 60*mph2ftps;
% Time domain simulation:
states_arr30 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u30, delta_mod);
states_arr60 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u60, delta_mod);
% Plot the states:
figure;
subplot(2,1,1);
plot(t, states_arr30(1,:));
xlabel('time (sec)');
ylabel('drift angle (rad)');
grid on;
% title('yaw rate for 30 mph');
hold on;
plot(t, states_arr60(1,:));
xlabel('time (sec)');
```

```
ylabel('drift angle (rad)');
title('Drift angle for 30 and 60 mph');;
legend('30 mph', '60 mph');
subplot(2,1,2);
plot(t, states_arr30(2,:));
xlabel('time (sec)');
ylabel('yaw rate (rad/sec)');
grid on;
hold on;
plot(t, states_arr60(2,:));
title('Yaw rate for 30 and 60 mph');
legend('30 mph', '60 mph');
%% Code1Q6:
%% 6. Add nonlinear tires and assume equal roll moment front and rear, 60/40 biased to the front, and
% 40/60 biased to the rear. Compare your biased results to the results of 3)-5) with linear tires.
%% Compare with 3):
close all;
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
% Time domain simulation:
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure(1); % 50-50 weight bias
hold on;
figure(2); % 60-40 weight bias
hold on;
figure(3);
hold on;
legends_50_50 = {};
legends_60_40 = {};
legends_40_60 = {};
for i = 1:length(speeds)
u = speeds(i);
delta2 = (1 / radius)*(l2 + (u*u*K_understeer))*ones(1, length(t)); % (u / radius)*ones(1, length(t)); % Update delta to accomplish 400 ft
radius
% Linear tire:
C1 = 2*140*180/pi;
C2 = 2*140*180/pi;
states_arr_lin = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2);
% 50 / 50 weight bias:
figure(1);
W1 = W / 2; W2 = W / 2;
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_50_50 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2);
plot(t, states_arr_lin(2,:), '--', 'linewidth', 2);
legends_50_50{end+1} = ['Yaw rate @' num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_50_50(2,:), 'linewidth', 2);
legends_50_50{end+1} = [num2str(u*ftps2mph) ' mph with non-linear tires and 50-50 wb'];
K_understeer_50_50 = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
u_char_50_50 = sqrt(l2/K_understeer_50_50)*ftps2mph;
% 60 / 40 weight bias:
figure(2);
W1 = 0.6*W; W2 = 0.4*W;
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_60_40 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2);
plot(t, states_arr_lin(2,:), '--', 'linewidth', 2);
legends_60_40{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_60_40(2,:), 'linewidth', 2);
legends_60_40{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 60-40 wb'];
K_understeer_60_40 = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
u_char_60_40 = sqrt(l2/K_understeer_60_40)*ftps2mph;
% 40 / 60 weight bias:
figure(3);
W1 = 0.4*W; W2 = 0.6*W;
```

```
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_40_60 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2);
plot(t, states_arr_lin(2,:), '--', 'linewidth', 2);
legends_40_60{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_40_60(2,:), 'linewidth', 2);
legends_40_60{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 40-60 wb'];
K_understeer_40_60 = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
u_critical_40_60 = sqrt(l2/-K_understeer_40_60)*ftps2mph;
end
figure(1);
legend(legends_50_50{:});
% Add text to plot with understeering coefficient and characteristic speed:
text(4.0, 0.7, ['K_{understeer} = ' num2str(K_understeer_50_50) ' u_{char}= ' num2str(u_char_50_50) ' mph'], 'FontSize', 16);
title('Yaw rate response for 50/50 weight bias');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
grid on;
hold off;
figure(2);
legend(legends_60_40{:});
% Add text to plot with understeering coefficient and characteristic speed:
text(4.0, 0.7, ['K_{understeer} = ' num2str(K_understeer_60_40) ' u_{char} = ' num2str(u_char_60_40) ' mph'], 'FontSize', 16);
title('Yaw rate response for 60/40 weight bias');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
grid on;
hold off;
figure(3);
ylim([0, 1.0])
legend(legends_40_60{:});
% Add text to plot with understeering coefficient and critical speed:
text(4.0, 0.8, ['K_{understeer} = ' num2str(K_understeer_40_60) ' u_{critical} = ' num2str(u_critical_40_60) ' mph'], 'FontSize', 16);
title('Yaw rate response for 40/60 weight bias');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
grid on;
hold off;
% Create a table where the leftmost column is the speed, then understeer coefficient, then characteristic speed:
table = [["50/50 bias", "60/40 bias", "40/60 bias"]', [K_understeer_50_50, K_understeer_60_40, K_understeer_40_60]', [u_char_50_50,
u_char_60_40, u_critical_40_60]'];

%% Code1Q6B:
%% Compare with 4-5:
close all;
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
% Time domain simulation:
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure(1); % 50-50 weight bias
subplot(2,1,1);
hold on;
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)')
title('Yaw rate time response 50/50 bias');
subplot(2,1,2);
hold on;
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
title('Drift angle time response 50/50 bias');
figure(2); % 60-40 weight bias
subplot(2,1,1);
hold on;
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
```

```
title('Yaw rate time response 60/40 bias');
subplot(2,1,2);
hold on;
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
title('Drift angle time response 60/40 bias');
figure(3);
subplot(2,1,1);
hold on;
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
% ylim([0.0, 2.0]);
title('Yaw rate time response 40/60 bias');
subplot(2,1,2);
hold on;
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
% ylim([-20.0, 20.0]);
title('Drift angle time response 40/60 bias');
legends_50_50_r = {};
legends_60_40_r = {};
legends_40_60_r = {};
legends_50_50_beta = {};
legends_60_40_beta = {};
legends_40_60_beta = {};
line_width = 1;
for i = 1:length(speeds)
u = speeds(i);
% Linear tire:
C1 = 2*140*180/pi;
C2 = 2*140*180/pi
states_arr_lin = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod);
% Non-linear tires:
% 50 / 50 weight bias:
figure(1);
W1 = W / 2; W2 = W / 2;
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_50_50 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod);
subplot(2,1,1);
plot(t, states_arr_lin(2,:), '--', 'linewidth', line_width);
legends_50_50_r{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_50_50(2,:), 'linewidth', line_width*2);
legends_50_50_r{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 50-50 wb'];
% Understeering coefficient:
K_understeer_50_50_nlar = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
% Characteristic speed:
u_char_50_50 = sqrt(l2/K_understeer_50_50_nlar)*ftps2mph;
subplot(2,1,2);
plot(t, states_arr_lin(1,:), '--', 'linewidth', line_width);
legends_50_50_beta{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_50_50(1,:), 'linewidth', line_width*2);
legends_50_50_beta{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 50-50 wb'];
% 60 / 40 weight bias:
figure(2);
W1 = 0.6*W; W2 = 0.4*W;
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_60_40 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod);
subplot(2,1,1);
plot(t, states_arr_lin(2,:), '--', 'linewidth', line_width);
legends_60_40_r{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_60_40(2,:), 'linewidth', line_width*2);
legends_60_40_r{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 60-40 wb'];
% Understeering coefficient:
```

```
K_understeer_60_40_nlar = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
% Characteristic speed:
u_char_60_40 = sqrt(l2/K_understeer_60_40_nlar)*ftps2mph;
subplot(2,1,2);
plot(t, states_arr_lin(1,:), '--', 'linewidth', line_width);
legends_60_40_beta{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_60_40(1,:), 'linewidth', line_width*2);
legends_60_40_beta{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 50-50 wb'];
% 40 / 60 weight bias:
figure(3);
W1 = 0.4*W; W2 = 0.6*W;
C1 = 2*(0.2*W1 - 0.0000942*(W1^2))*180/pi; % lbs/rad
C2 = 2*(0.2*W2 - 0.0000942*(W2^2))*180/pi; % lbs/rad
states_arr_40_60 = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod);
subplot(2,1,1);
plot(t, states_arr_lin(2,:), '--', 'linewidth', line_width);
legends_40_60_r{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_40_60(2,:), 'linewidth', line_width*2);
legends_40_60_r{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 40-60 wb'];
% Understeering coefficient:
K_understeer_40_60_nlar = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
% Characteristic speed:
u_critical_40_60 = sqrt(l2/-K_understeer_40_60_nlar)*ftps2mph;
subplot(2,1,2);
plot(t, states_arr_lin(2,:), '--', 'linewidth', line_width);
legends_40_60_beta{end+1} = [num2str(u*ftps2mph) ' mph with linear tires'];
plot(t, states_arr_40_60(2,:), 'linewidth', line_width*2);
legends_40_60_beta{end+1} = [num2str(u*ftps2mph) ' mph with non-linear and 40-60 wb'];
end
figure(1);
subplot(2,1,1);
title(['Yaw rate time response with 50/50 bias - K_u = ' num2str(K_understeer_50_50_nlar) ' | u_{char} ' num2str(u_char_50_50) ' mph']);
legend(legends_50_50_r{:});
grid on;
subplot(2,1,2);
title(['Drift angle time response with 50/50 bias - K_u = ' num2str(K_understeer_50_50_nlar) ' | u_{char} ' num2str(u_char_50_50) '
mph']);
legend(legends_50_50_beta{:});
grid on;
hold off;
figure(2);
subplot(2,1,1);
title(['Yaw rate time response with 60/40 bias - K_u = ' num2str(K_understeer_60_40_nlar) ' | u_{char} ' num2str(u_char_60_40) ' mph']);
legend(legends_60_40_r{:});
grid on;
subplot(2,1,2);
title(['Drift angle time response with 60/40 bias - K_u = ' num2str(K_understeer_60_40_nlar) ' | u_{char} ' num2str(u_char_60_40) '
mph']);
legend(legends_60_40_beta{:});
grid on;
hold off;
figure(3);
subplot(2,1,1);
title(['Yaw rate time response with 40/60 bias - K_u = ' num2str(K_understeer_40_60_nlar) ' | u_{char} ' num2str(u_critical_40_60) '
mph']);
legend(legends_40_60_r{:});
ylim([-10, 10]);
grid on;
subplot(2,1,2);
title(['Drift angle time response with 40/60 bias - K_u = ' num2str(K_understeer_40_60_nlar) ' | u_{char} ' num2str(u_critical_40_60) '
mph']);
legend(legends_40_60_beta{:});
ylim([-20, 40]);
grid on;
hold off;
```

```
%% Try out transfer functions:
u = speeds(5);
K_understeer = -m*(x1*C1 + x2*C2)/(C1*C2*l2);
G_yaw_rate = tf([((x1*m*u*u)/(l2*C2)), u], [(m*u*u*Iz)/(l2*C1*C2), (u/l2)*((m*x1*x1+Iz)/(C2)), l2 + u*u*K_understeer]);
step(G_yaw_rate)
%% Run time domain simulation:
t_initial = 0;
t_final = 25;
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
% Generate a step input for the steering angle delta:
delta = zeros(1, length(t));
delta(1, length(t)/4:length(t)) = 0.1;
states_arr = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta);
% Time plots for each state:
figure
plot(t, states_arr(1,:));
figure
plot(t, states_arr(2,:));
function states_arr = simulate_bike_2dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta)
% Bicycle model using attack angle conventions:
% Attack angle is the negative of the slip angle
% The sign is embedded in the x1 and x2 distances
A = [
(-C1 - C2)/(m*u), ((-x1*C1 - x2*C2)/(m*u^2)) - 1;
(-x1*C1 - x2*C2)/(Iz), (-(x1^2)*C1 - (x2^2)*C2)/(Iz*u);
];
B = [
(C1)/(m*u);
(x1*C1)/Iz;
];
% Discretize using Euler:
A_dis = eye(2) + dt * A;
B_dis = dt * B;
% Initial conditions:
states = [0; 0];
states_arr = zeros(2,length(t));
% simulation loop:
for i = 1:length(t)
states = A_dis * states + B_dis * delta(i);
states_arr(:, i) = states;
end
end
function delta = slope_it_down(t, dt, at, delta, left_bound, right_bound, slope)
time_to_reach = (right_bound - left_bound) / slope; % time units
delta(1, at/dt:(at + time_to_reach)/dt) = left_bound + slope * (t(at/dt:(at + time_to_reach)/dt) - t(at/dt)); % * sign(right_bound -
left_bound);
end
```

Part 2-3 Code:

```
%% Code2P1:
%% Load some initial values:
close all;
clear;
clc;
% Conversion factors:
deg2rad = pi / 180;
rad2deg = 180 / pi;
in2ft = 1 / 12;
ft2in = 12;
mph2ftps = 5280 / 3600;
ftps2mph = 3600 / 5280;
% Bicycle model parameters:
```

```
W = 3000; % lbs
Ws = 2700; % lbs
g = 32.174; % ft/sec^2
x1 = 3.5; % ft
x2 = -4.5; % ft = -1.0; % ft
h = -1.0; % ft
track_width = 6.0; % ft
Iz = 40000 / g; % lbs*ft^2
Ix = 15000 / g; % lbs*ft^2
c = 0.5; % ft
dl_phi_f = 8000; % lbs*ft
dl_phi_r = 5000; % lbs*ft
dl_dphi_f = 1000; % lbs*ft
dl_dphi_r = 500; % lbs*ft
% Masses:
m = W / g;
ms = Ws / g;
% Roll stiffness:
K_phi = (dl_phi_f + dl_phi_r + ms*g*h); % lb*ft / rad
% Roll damping:
D_phi = (dl_dphi_f + dl_dphi_r); % lb*ft/sec / (rad/sec)
p = 12*in2ft; % ft
d = 12*in2ft; % ft
gear_ratio = 15; % []
efficiency = 1.0; % Steering gearbox efficiency
Ks = 10*in2ft*rad2deg; % (in*lbs/deg)*(ft/in)*(deg/rad) -> ft*lbs / rad
tm = 3*in2ft; % in*(ft/in)-> ft - Pneumatic trail
% u = 30; % ft/sec
C1 = 2*140*180/pi; %*rad2deg; % lbs/deg * (deg/rad) -> lbs / rad
C2 = 2*140*180/pi; % *rad2deg; % lbs/deg * (deg/rad) -> lbs / rad
l2 = x1 - x2; % Wheelbase
%% Code2Q1:
%% 1. Repeat the steps in (Part 1: 2-DoF Vehicle) with a three degree of freedom model allowing sprung
% mass roll. Using the steady-state yaw rate response, construct plots from 0 to 120 mph (i.e.      ) for
% various speeds without roll steer and with a rear roll steer coefficient of    r = −0.03.
% Calculate the steady state gains for different speeds:
speeds = linspace(10, 120, 12)*mph2ftps;
delta2r_w_roll_steer_gain_arr = zeros(1, length(speeds));
delta2r_wout_roll_steer_gain_arr = zeros(1, length(speeds));
dt = 0.01;
t_initial = 0;
t_final = 5;
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure;
hold on;
grid on;
xlabel('Time (seconds)');
ylabel('Gain (r/   )');
xlim([t_initial, t_final]);
ylim([0, 6.0]);
legend_w_roll_steer_arr = cell(1, length(speeds));
legend_wout_roll_steer_arr = cell(1, length(speeds));
for i = 1:length(speeds)
u = speeds(i);
% Without roll steer:
eps1 = 0;
eps2 = 0.0;
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
delta2r_wout_roll_steer_gain_arr(i) = (u*C1*(Cb-x1*Ca)) /(Cb*Cb - Ca*Cc + Cb*m*u*u + (x1*Ca*C_phi1 + x2*Ca*C_phi2 - (C_phi1
+ C_phi2)*Cb)*(ms*h*u*u / K_phi));
% plot each gain in an xy plot:
subplot(2,1,1);
```

```
yline(delta2r_wout_roll_steer_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph)
'mph'], 'Color', rand(1,3), 'LineWidth', 2);
legend_wout_roll_steer_arr{i} = ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph) ' mph'];
hold on;
% With rear roll steer coefficient of -0.03:
eps1 = 0;
eps2 = -0.03;
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
delta2r_w_roll_steer_gain_arr(i) = (u*C1*(Cb-x1*Ca)) /(Cb*Cb - Ca*Cc + Cb*m*u*u + (x1*Ca*C_phi1 + x2*Ca*C_phi2 - (C_phi1 +
C_phi2)*Cb)*(ms*h*u*u / K_phi));
% plot each gain in an xy plot:
subplot(2,1,2);
yline(delta2r_w_roll_steer_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_w_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph) 'mph'],
'Color', rand(1,3), 'LineWidth', 2);
legend_w_roll_steer_arr{i} = ['s.s gain = ' num2str(delta2r_w_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph) ' mph'];
hold on;
end
subplot(2,1,1);
xlabel('Time (seconds)');
ylabel('Gain []');
title('Steady state gains at different speeds with ε_1=0.0 and ε_2=0.0');
legend(legend_wout_roll_steer_arr{:}, 'Location', 'eastoutside');
subplot(2,1,2);
xlabel('Time (seconds)');
ylabel('Gain []');
title('Steady state gains at different speeds with ε_1=0.0 and ε_2=-0.03');
legend(legend_w_roll_steer_arr{:}, 'Location', 'eastoutside');
hold off;
%% Code2Q2:
%% 2. Using the 3-DoF equations of motion, simulate the vehicle response to a steering input.
% Determine the steering input required to result in a 400 ft radius turn (hint: knowing the forward
% speed and radius of a circle, the required yaw rate can be calculated). Repeat the result at
% increments of 10 mph from 10 to 120 mph. Compare the results with the steady-state results from
% 1) above. As before, we are using this step to validate your simulation model.
close all;
radius = 400; % ft
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
% Time domain simulation:
t_initial = 0; % sec
t_final = 20; % sec
dt = 0.01; % sec
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
figure(1);
subplot(3,1,1);
xlabel('Time (seconds)');
ylabel('Gain []');
title('Steady state gains for different speeds with ε_1=0.0 and ε_2=0.0');
hold on;
grid on;
subplot(3,1,2);
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
title('Yaw rate response for different speeds with ε_1=0.0 and ε_2=0.0');
hold on;
subplot(3,1,3);
xlabel('Time (seconds)');
ylabel('Steering angle (rad)');
title('Steering wheel angle required to produce a turn radius of 400 ft');
hold on;
figure(2);
subplot(3,1,1);
```

```
xlabel('Time (seconds)');
ylabel('Gain []');
title('Steady state gains for different speeds with ε_1=0.0 and ε_2=-0.03');
hold on;
grid on;
subplot(3,1,2);
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
title('Yaw rate response for different speeds with ε_1=0.0 and ε_2=-0.03');
hold on;
subplot(3,1,3);
xlabel('Time (seconds)');
ylabel('Steering angle (rad)');
title('Steering wheel angle required to produce a turn radius of 400 ft');
hold on;
legend_gains_wout_roll_steer_arr = cell(1, length(speeds));
legend_yaw_rate_wout_roll_steer_arr = cell(1, length(speeds));
legend_steering_angle_wout_roll_steer_arr = cell(1, length(speeds));
legend_gains_w_roll_steer_arr = cell(1, length(speeds));
legend_yaw_rate_w_roll_steer_arr = cell(1, length(speeds));
legend_steering_angle_w_roll_steer_arr = cell(1, length(speeds));
line_width = 2.0;
for i = 1:length(speeds)
u = speeds(i);
% Code2Q2_1:
% Configuration 1: eps1 = 0 and eps2 = 0:
eps1 = 0;
eps2 = 0;
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
K_understeer_wout_roll = (-m*(Cb)/(C1*C2*l2));
K_understeer_roll_effect = (ms*h/K_phi)*(Cb*(C_phi1 + C_phi2) - Ca*(x1*C_phi1 + x2*C_phi2))/(C1*C2*l2);
K_understeer_total_wout_roll_steer = K_understeer_wout_roll + K_understeer_roll_effect;
delta2r_wout_roll_steer_gain_arr(i) = (u*C1*(Cb-x1*Ca)) /(Cb*Cb - Ca*Cc + Cb*m*u*u + (x1*Ca*C_phi1 + x2*Ca*C_phi2 - (C_phi1
+ C_phi2)*Cb)*(ms*h*u*u / K_phi));
% plot each gain in an xy plot:
figure(1);
subplot(3,1,1);
yline(delta2r_wout_roll_steer_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph)
'mph'], 'Color', rand(1,3), 'LineWidth', 2);
legend_gains_wout_roll_steer_arr{i} = ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph) ' mph'];
hold on;
subplot(3,1,2);
delta2 = (1 / radius)*(l2 + (u*u*K_understeer_total_wout_roll_steer))*ones(1, length(t));
% Simulate:
states_arr2 = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2, ms, h, K_phi, D_phi, eps1, eps2, Ix, c);
plot(t, states_arr2(2,:), LineWidth=line_width);
legend_yaw_rate_wout_roll_steer_arr{i} = ['Calculated s.s gain @ ' num2str(u*ftps2mph) ' is ' num2str(states_arr2(2,
length(t))/delta2(length(t)))];
subplot(3,1,3);
plot(t, delta2, LineWidth=line_width);
legend_steering_angle_wout_roll_steer_arr{i} = ['    = ' num2str(delta2(1)) ' @ ' num2str(u*ftps2mph) 'mph'];
% Code2Q2_2:
% Configuration 2: eps1 = 0 and eps2 = -0.03:
eps1 = 0;
eps2 = -0.03;
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
K_understeer_wout_roll = (-m*(Cb)/(C1*C2*l2));
K_understeer_roll_effect = (ms*h/K_phi)*(Cb*(C_phi1 + C_phi2) - Ca*(x1*C_phi1 + x2*C_phi2))/(C1*C2*l2);
K_understeer_total_wout_roll_steer = K_understeer_wout_roll + K_understeer_roll_effect;
```

```
delta2r_wout_roll_steer_gain_arr(i) = (u*C1*(Cb-x1*Ca)) /(Cb*Cb - Ca*Cc + Cb*m*u*u + (x1*Ca*C_phi1 + x2*Ca*C_phi2 - (C_phi1
+ C_phi2)*Cb)*(ms*h*u*u / K_phi));
% plot each gain in an xy plot:
figure(2);
subplot(3,1,1);
yline(delta2r_wout_roll_steer_gain_arr(i),'--', ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph)
'mph'], 'Color', rand(1,3), 'LineWidth', 2);
legend_w_roll_steer_arr{i} = ['s.s gain = ' num2str(delta2r_wout_roll_steer_gain_arr(i)) ' @ ' num2str(u*ftps2mph) ' mph'];
hold on;
subplot(3,1,2);
delta2 = (1 / radius)*(l2 + (u*u*K_understeer_total_wout_roll_steer))*ones(1, length(t));
% Simulate:
states_arr2 = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta2, ms, h, K_phi, D_phi, eps1, eps2, Ix, c);
plot(t, states_arr2(2,:), LineWidth=line_width);
legend_yaw_rate_w_roll_steer_arr{i} = ['Calculated s.s gain @ ' num2str(u*ftps2mph) ' is ' num2str(states_arr2(2,
length(t))/delta2(length(t)))];
subplot(3,1,3);
plot(t, delta2, LineWidth=line_width);
legend_steering_angle_w_roll_steer_arr{i} = ['    = ' num2str(delta2(1)) ' @ ' num2str(u*ftps2mph) 'mph'];
end
figure(1);
subplot(3,1,1);
hold off;
legend(legend_wout_roll_steer_arr);
grid on;
subplot(3,1,2);
hold off;
legend(legend_yaw_rate_wout_roll_steer_arr);
grid on;
subplot(3,1,3);
hold off;
legend(legend_steering_angle_wout_roll_steer_arr);
grid on;
figure(2);
subplot(3,1,1);
hold off;
legend(legend_w_roll_steer_arr);
grid on;
subplot(3,1,2);
hold off;
legend(legend_yaw_rate_w_roll_steer_arr);
grid on;
subplot(3,1,3);
hold off;
legend(legend_steering_angle_w_roll_steer_arr);
grid on;
%% Code2Q3:
%% Code2Q3_1:
%% Configuration 1:
eps1 = 0.04; eps2 = 0.04;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config1 = mean(steady_state_gains_arr)
%% Code2Q3_2:
%% Configuration 2:
eps1 = 0.04; eps2 = 0.0;
% Time array:
t_initial = 0;
```

```
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config2 = mean(steady_state_gains_arr)
%% Code2Q3_3:
%% Configuration 3:
eps1 = 0.04; eps2 = -0.04;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config3 = mean(steady_state_gains_arr)
%% Code2Q3_4:
%% Configuration 4:
eps1 = 0; eps2 = 0.04;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config4 = mean(steady_state_gains_arr)
%% Code2Q3_5:
%% Configuration 5:
eps1 = 0; eps2 = 0;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config5 = mean(steady_state_gains_arr)
%% Code2Q3_6:
%% Configuration 6:
eps1 = 0; eps2 = -0.04;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config6 = mean(steady_state_gains_arr)
%% Code2Q3_7:
%% Configuration 7:
eps1 = -0.04; eps2 = 0.04;
% Time array:
t_initial = 0;
```

```
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config7 = mean(steady_state_gains_arr)
%% Code2Q3_8:
%% Configuration 8:
eps1 = -0.04; eps2 = 0;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config8 = mean(steady_state_gains_arr)
%% Code2Q3_9:
%% Configuration 9:
eps1 = -0.04; eps2 = -0.04;
desired_radius = 400;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.01;
% Define speeds of interest:
speeds = linspace(10, 120, 12);
speeds = speeds*mph2ftps;
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod);
gains_avg_config9 = mean(steady_state_gains_arr)
%% Code3:
%% Part 3:
%% Code3Q1:
% 1. Return to the baseline case of    f = 0;    r = −0.03. Add the nonlinear tire model. You will have to
% calculate the tire loads at each time step, knowing that the moments from roll stiffness and roll
% damping on the sprung mass are reacted by vertical forces through the contact patch of the tires,
% and add to or subtract from the static tire loads. Compare and interpret your results above with
% the corresponding results from (Part 2: 3-DoF Vehicle (Linear)).
close all;
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.002;
% Generate input signal with time array:
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
% Define speeds of interest:
speeds = linspace(10, 120, 12); % linspace(10, 120, 12);
speeds = speeds*mph2ftps;
eps1 = 0; eps2 = -0.03;
test_nonlinear_model(dt, t, m, x1, x2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta_mod*0.05, W, C1, C2)
%% Test non-linear model:
% Generate a step test input for the steering angle delta:
delta_test = zeros(1, length(t));
delta_test(1, length(t)/4:length(t)) = 0.1;
% eps1 = 0.0; eps2 = 0.0;
% Simulate:
[W1l_arr, W1r_arr, W2l_arr, W2r_arr, states_arr] = simulate_non_linear_bike_3dof(dt, t, m, x1, x2, Iz, u, delta_test, ms, h, K_phi, D_phi,
eps1, eps2, Ix, c, W);
% Calculate the understeering coefficient:
```

```
K_understeer_handling_effect = (-m*(Cb)/(C1*C2*l2));
K_understeer_roll_effect = (ms*h/K_phi)*(Cb*(C_phi1 + C_phi2) - Ca*(x1*C_phi1 + x2*C_phi2))/(C1*C2*l2);
K_understeer_total = K_understeer_handling_effect + K_understeer_roll_effect;
% Plot the results:
figure;
subplot(4,1,1);
plot(t, states_arr(2,:));
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
title('Yaw rate response for ε1= 0 and ε2= -0.03');
grid on;
subplot(4,1,2);
delta_test = zeros(1, length(t));
plot(t, states_arr(1,:) / u);
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
title('Drift angle response for ε1= 0, ε2= -0.03');
grid on;
subplot(4,1,3);
plot(t, states_arr(4,:));
xlabel('Time (seconds)');
ylabel('Roll (rad)');
title('Roll angle response for ε1= 0, ε2= -0.03');
grid on;
subplot(4,1,4);
plot(t, delta_mod);
xlabel('Time (seconds)');
ylabel('Steering angle (rad)');
title('Steering angle input');
grid on;
% test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi,
% eps1, eps2, Ix, c, delta_mod);

%% Code3Q2:
%% 2. Use your model to comment on the effect of roll stiffness and roll damping on vehicle handling.
% That is, vary the parameters and describe the effect. Present significant results and comment on
% them.
% Time array:
t_initial = 0;
t_final = 10;
dt = 0.002;
% Generate input signal with time array:
[t, delta_mod] = generate_input_signal(dt, t_initial, t_final);
delta_mod = delta_mod*0.05; % Scale the input signal
K_phi_arr = linspace(K_phi*0.5, 2*K_phi, 5);
D_phi_arr = linspace(D_phi*0.5, 2*D_phi, 5);
% Roll steers:
eps1 = 0; eps2 = -0.03;
close all;
figure(1);
subplot(3,1,1);
hold on;
grid on;
title('Effect of roll stiffness on yaw rate at 60 mph');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
subplot(3,1,2);
hold on;
grid on;
title('Effect of roll stiffness on drift angle at 60 mph');
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
subplot(3,1,3);
hold on;
grid on;
```

```
title('Effect of roll stiffness on roll angle at 60 mph');
xlabel('Time (seconds)');
ylabel('Roll angle (rad)');
legend_non_lin_yaw_rate_arr = {};
legend_non_lin_drift_angle_arr = {};
legend_non_lin_roll_angle_arr = {};
for i = 1:length(K_phi_arr) % We leave the damping constant at its nominal value and vary the stiffness
u = 60*mph2ftps;
[W1l_arr_non_lin, W1r_arr_non_lin, W2l_arr_non_lin, W2r_arr_non_lin, states_arr_non_lin] = simulate_non_linear_bike_3dof(dt, t, m,
x1, x2, Iz, u, delta_mod, ms, h, K_phi_arr(i), D_phi, eps1, eps2, Ix, c, W);
states_arr_lin = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod, ms, h, K_phi_arr(i), D_phi, eps1, eps2, Ix, c);
% Plot the states:
figure(1);
subplot(3,1,1);
plot(t, states_arr_non_lin(2,:), 'LineWidth', 2);
legend_non_lin_yaw_rate_arr{end+1} = ['Non lin. with K_{\phi} = ' num2str(K_phi_arr(i))];
plot(t, states_arr_lin(2,:), '--');
legend_non_lin_yaw_rate_arr{end+1} = ['Linear with K_{\phi} = ' num2str(K_phi_arr(i))];
subplot(3,1,2);
plot(t, states_arr_non_lin(1,:) / u, 'LineWidth', 2); % Normalize by longitudinal speed to get drift angle.
plot(t, states_arr_lin(1,:) / u, '--'); % Normalize by longitudinal speed to get drift angle.
% legend_non_lin_drift_angle_arr{end+1} = ;
subplot(3,1,3);
plot(t, states_arr_non_lin(4,:), 'LineWidth', 2);
plot(t, states_arr_lin(4,:), '--');
end
subplot(3,1,3)
legend(legend_non_lin_yaw_rate_arr);
%% Do the same as before, but keeping K_phi at its nominal value and varying D_phi:
close all;
subplot(3,1,1);
hold on;
grid on;
title('Effect of roll damping on yaw rate at 60 mph');
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
subplot(3,1,2);
hold on;
grid on;
title('Effect of roll damping on drift angle at 60 mph');
xlabel('Time (seconds)');
ylabel('Drift angle (rad)');
subplot(3,1,3);
hold on;
grid on;
title('Effect of roll damping on roll angle at 60 mph');
xlabel('Time (seconds)');
ylabel('Roll angle (rad)');
legend_non_lin_yaw_rate_arr = {};
legend_non_lin_drift_angle_arr = {};
legend_non_lin_roll_angle_arr = {};
for i = 1:length(D_phi_arr) % We leave the damping constant at its nominal value and vary the stiffness
u = 60*mph2ftps;
[W1l_arr_non_lin, W1r_arr_non_lin, W2l_arr_non_lin, W2r_arr_non_lin, states_arr_non_lin] = simulate_non_linear_bike_3dof(dt, t, m,
x1, x2, Iz, u, delta_mod, ms, h, K_phi, D_phi_arr(i), eps1, eps2, Ix, c, W);
states_arr_lin = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_mod, ms, h, K_phi, D_phi_arr(i), eps1, eps2, Ix, c);
% Plot the states:
figure(1);
subplot(3,1,1);
plot(t, states_arr_non_lin(2,:), 'LineWidth', 2);
legend_non_lin_yaw_rate_arr{end+1} = ['Non lin. with D_{\phi} = ' num2str(D_phi_arr(i))];
plot(t, states_arr_lin(2,:), '--');
legend_non_lin_yaw_rate_arr{end+1} = ['Linear with D_{\phi} = ' num2str(D_phi_arr(i))];
subplot(3,1,2);
plot(t, states_arr_non_lin(1,:) / u, 'LineWidth', 2); % Normalize by longitudinal speed to get drift angle.
```

```
plot(t, states_arr_lin(1,:) / u, '--');
% legend_non_lin_drift_angle_arr{end+1} = ;
subplot(3,1,3);
plot(t, states_arr_non_lin(4,:), 'LineWidth', 2);
plot(t, states_arr_lin(4,:), '--');
end
subplot(3,1,3);
legend(legend_non_lin_yaw_rate_arr);
%% Test linear model here:
t_initial = 0;
t_final = 5;
t = linspace(t_initial, t_final, (t_final - t_initial) / dt);
% Generate a step input for the steering angle delta:
delta_test = zeros(1, length(t));
delta_test(1, length(t)/4:length(t)) = 0.01;
states_arr = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta_test, ms, h, K_phi, D_phi, eps1, eps2, Ix, c);
% Time plots for each state:
figure;
plot(t, states_arr(1,:));
figure;
plot(t, states_arr(2,:));
%% Build Yaw Plane Model with roll:
function states_arr = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta, ms, h, K_phi, D_phi, eps1, eps2, Ix, c)
% Bicycle model using attack angle conventions + roll as a 3rd degree of freedom:
% Attack angle is the negative of the slip angle
% The sign is embedded in the x1 and x2 distances
% Ordinary bicycle model:
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
% Roll steer effects:
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
% Parallel axis theorem:
Ix_steiner = Ix + ms*h*h;
A = [
-Ca/u, (-Cb/u) - m*u, 0, C_phi1 + C_phi2;
-Cb/u, -Cc/u, 0, x1*C_phi1 + x2*C_phi2;
0, ms*h*u, -D_phi, -K_phi;
];
B = [
C1;
x1*C1;
0;
];
inertia_matrix = [
m, 0, -ms*h;
0, Iz, -ms*h*c;
-ms*h, -ms*h*c, Ix_steiner;
];
% Discretize using Euler:
% Initial conditions:
states = [0; 0; 0; 0];
states_arr = zeros(length(states),length(t));
% simulation loop:
for i = 1:length(t)
states(4) = states(4) + dt*states(3); % Roll = Roll_prev + dt*(dRoll/dt)
states(1:3) = (states(1:3) + inv(inertia_matrix)*dt*(A*states + B*delta(i))); % inv(inertia_matrix)*(A_dis * states + B_dis * delta(i));
states_arr(:, i) = states;
end
end
function [W1l_arr, W1r_arr, W2l_arr, W2r_arr, states_arr] = simulate_non_linear_bike_3dof(dt, t, m, x1, x2, Iz, u, delta, ms, h, K_phi,
D_phi, eps1, eps2, Ix, c, W)
% Bicycle model using attack angle conventions + roll as a 3rd degree of freedom:
% Attack angle is the negative of the slip angle
% The sign is embedded in the x1 and x2 distances
```

```
% Parallel axis theorem:
Ix_steiner = Ix + ms*h*h;
inertia_matrix = [
m, 0, -ms*h;
0, Iz, -ms*h*c;
-ms*h, -ms*h*c, Ix_steiner;
];
% Initial conditions:
states = [0; 0; 0; 0];
states_arr = zeros(length(states),length(t));
% Weight arrays:
W1r_arr = zeros(1, length(t));
W1l_arr = zeros(1, length(t));
W2r_arr = zeros(1, length(t));
W2l_arr = zeros(1, length(t));
% simulation loop:[W1l_arr, W1r_arr, W2l_arr, W2r_arr, states_arr]
for i = 1:length(t)
W1 = W/2; W2 = W/2; % Assuming 50/50 weight bias:
% Weight transfer:
W1r = (W1/2) + (1/(4*h))*(-K_phi*states(4) - D_phi*states(3)); %*0.5; % + ms*h*u*states(2)
W1l = (W1/2) - (1/(4*h))*(-K_phi*states(4) - D_phi*states(3)); %*0.5;
W2r = (W2/2) + (1/(4*h))*(-K_phi*states(4) - D_phi*states(3)); % *0.5;
W2l = (W2/2) - (1/(4*h))*(-K_phi*states(4) - D_phi*states(3)); % *0.5;
% Store the weights to analyze weight transfer over time:
W1r_arr(1,i) = W1r;
W1l_arr(1,i) = W1l;
W2r_arr(1,i) = W2r;
W2l_arr(1,i) = W2l;
% Non-linear tires:
C1r = 0.2*W1r - 0.0000942*W1r*W1r;
C1l = 0.2*W1l - 0.0000942*W1l*W1l;
C1 = (C1r + C1l)*(180/pi); % Convert to lbs/rad
C2r = 0.2*W2r - 0.0000942*W2r*W2r;
C2l = 0.2*W2l - 0.0000942*W2l*W2l;
C2 = (C2r + C2l)*(180/pi); % Convert to lbs/rad
% Roll steer effects:
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
% Bicycle model with roll:
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
A = [
-Ca/u, -Cb/u - m*u, 0, C_phi1 + C_phi2;
-Cb/u, -Cc/u, 0, x1*C_phi1 + x2*C_phi2;
0, ms*h*u, -D_phi, -K_phi;
];
B = [
C1;
x1*C1;
0;
];
states(1:3) = (states(1:3) + inv(inertia_matrix)*dt*(A*states + B*delta(i))); % inv(inertia_matrix)*(A_dis * states + B_dis * delta(i));
states(4) = states(4) + dt*states(3);
states_arr(:, i) = states;
end
end
function steady_state_gains_arr = test_model(dt, t, m, x1, x2, C1, C2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta)
% Conversion factors:
deg2rad = pi / 180;
rad2deg = 180 / pi;
in2ft = 1 / 12;
ft2in = 12;
mph2ftps = 5280 / 3600;
ftps2mph = 3600 / 5280;
% Ordinary bicycle model setup:
```

```
Ca = C1 + C2;
Cb = x1*C1 + x2*C2;
Cc = x1*x1*C1 + x2*x2*C2;
l2 = x1 - x2;
C_phi1 = C1*eps1; C_phi2 = C2*eps2;
% Calculate our understeering coefficient:
K_understeer_wout_roll = (-m*(Cb)/(C1*C2*l2));
K_roll_effect = (ms*h/K_phi)*(Cb*(C_phi1 + C_phi2) - Ca*(x1*C_phi1 + x2*C_phi2))/(C1*C2*l2);
K_understeer = K_understeer_wout_roll + K_roll_effect;
figure;
subplot(3,1,1);
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
title(['Yaw rate response for ε1= ' num2str(eps1) ' and ε2= ' num2str(eps2) ' and Ku= ' num2str(K_understeer)]);
hold on;
subplot(3,1,2);
xlabel('Time (seconds)');
ylabel('Drift angle (rad)')
title(['Drift angle response for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2) ' and Ku= ' num2str(K_understeer)]);
hold on;
grid on;
subplot(3,1,3);
xlabel('Time (seconds)');
ylabel('Roll (rad)');
title(['Roll angle response for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2) ' and Ku= ' num2str(K_understeer)]);
hold on;
legend_yaw_rate_arr = cell(1, length(speeds));
legend_drift_angle_arr = cell(1, length(speeds));
legend_roll_angle_arr = cell(1, length(speeds));
steady_state_gains_arr = zeros(1, length(speeds));
% Simulate for different speeds:
for i = 1:length(speeds)
u = speeds(i);
% Simulate:
states_arr = simulate_bike_3dof(dt, t, m, x1, x2, C1, C2, Iz, u, delta, ms, h, K_phi, D_phi, eps1, eps2, Ix, c);
steady_state_gains_arr(i) = (u*C1*(Cb-x1*Ca)) /(Cb*Cb - Ca*Cc + Cb*m*u*u + (x1*Ca*C_phi1 + x2*Ca*C_phi2 - (C_phi1 +
C_phi2)*Cb)*(ms*h*u*u / K_phi));
subplot(3,1,1);
plot(t, states_arr(2,:), LineWidth=2.0);
legend_yaw_rate_arr{i} = ['Yaw rate @ ' num2str(u*ftps2mph) 'mph'];
subplot(3,1,2);
plot(t, states_arr(1,:) / u, LineWidth=2.0);
legend_drift_angle_arr{i} = ['Drift angle @ ' num2str(u*ftps2mph) 'mph'];
subplot(3,1,3);
plot(t, states_arr(4,:), LineWidth=2.0);
legend_roll_angle_arr{i} = ['Roll angle @ ' num2str(u*ftps2mph) 'mph'];
end
subplot(3,1,1);
legend(legend_yaw_rate_arr);
grid on;
subplot(3,1,2);
legend(legend_drift_angle_arr);
grid on;
subplot(3,1,3);
legend(legend_roll_angle_arr);
grid on;
hold off;
end
function test_nonlinear_model(dt, t, m, x1, x2, Iz, speeds, ms, h, K_phi, D_phi, eps1, eps2, Ix, c, delta, W, C1_lin, C2_lin)
% Conversion factors:
deg2rad = pi / 180;
rad2deg = 180 / pi;
in2ft = 1 / 12;
ft2in = 12;
mph2ftps = 5280 / 3600;
```

```
ftps2mph = 3600 / 5280;
figure(1);
subplot(3,1,1);
xlabel('Time (seconds)');
ylabel('Yaw rate (rad/sec)');
title(['Yaw rate response for ε1= ' num2str(eps1) ' and ε2= ' num2str(eps2)]);
hold on;
subplot(3,1,2);
xlabel('Time (seconds)');
ylabel('Drift angle (rad)')
title(['Drift angle response for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2)]);
hold on;
grid on;
subplot(3,1,3);
xlabel('Time (seconds)');
ylabel('Roll (rad)');
title(['Roll angle response for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2)]);
hold on;
figure(2);
subplot(4,1,1);
xlabel('Time (seconds)');
ylabel('FL weight transfer (lbs)');
title(['Left weight transfer with ε1= ' num2str(eps1) ' and ε2= ' num2str(eps2)]);
hold on;
subplot(4,1,2);
xlabel('Time (seconds)');
ylabel('FR weight transfer (lbs)')
title(['Front right weight transfer for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2)]);
hold on;
grid on;
subplot(4,1,3);
xlabel('Time (seconds)');
ylabel('RL weight transfer (lbs)');
title(['Rear left weight transfer for ε1= ' num2str(eps1) ', ε2= ' num2str(eps2)]);
hold on;
subplot(4,1,4);
xlabel('Time (seconds)');
ylabel('RR weight transfer (lbs)')
title(['Rear right weight transfer ε1= ' num2str(eps1) ', ε2= ' num2str(eps2)]);
hold on;
grid on;
legend_yaw_rate_arr = {}; % cell(1, 2*length(speeds));
legend_drift_angle_arr = {}; % cell(1, 2*length(speeds));
legend_roll_angle_arr = {}; % cell(1, 2*length(speeds));
legend_W1l_arr = cell(1, length(speeds));
legend_W1r_arr = cell(1, length(speeds));
legend_W2l_arr = cell(1, length(speeds));
legend_W2r_arr = cell(1, length(speeds));
% Simulate for different speeds:
for i = 1:length(speeds)
u = speeds(i);
% Simulate:
[W1l_arr, W1r_arr, W2l_arr, W2r_arr, states_non_linear_arr] = simulate_non_linear_bike_3dof(dt, t, m, x1, x2, Iz, u, delta, ms, h, K_phi,
D_phi, eps1, eps2, Ix, c, W);
states_linear_arr = simulate_bike_3dof(dt, t, m, x1, x2, C1_lin, C2_lin, Iz, u, delta, ms, h, K_phi, D_phi, eps1, eps2, Ix, c);
% States time response plots:
figure(1);
subplot(3,1,1);
plot(t, states_non_linear_arr(2,:), LineWidth=1);
legend_yaw_rate_arr{end+1} = ['With non-linear tires @ ' num2str(u*ftps2mph) 'mph'];
plot(t, states_linear_arr(2,:), '--', LineWidth=1);
legend_yaw_rate_arr{end+1} = ['With linear tires @ ' num2str(u*ftps2mph) 'mph'];
subplot(3,1,2);
plot(t, states_non_linear_arr(1,:) / u, LineWidth=1); % Normalize by longitudinal speed to get the drift angle
legend_drift_angle_arr{end+1} = ['Drift angle w nlar tires @ ' num2str(u*ftps2mph) 'mph'];
```

```
plot(t, states_linear_arr(1,:) / u, '--', LineWidth=1);
legend_drift_angle_arr{end+1} = ['Drift angle w linear tires @ ' num2str(u*ftps2mph) 'mph'];
subplot(3,1,3);
plot(t, states_non_linear_arr(4,:), LineWidth=1);
legend_roll_angle_arr{end+1} = ['Roll angle w weight transfer @ ' num2str(u*ftps2mph) 'mph'];
plot(t, states_linear_arr(4,:), '--', LineWidth=1);
legend_roll_angle_arr{end+1} = ['Roll angle w linear tires @ ' num2str(u*ftps2mph) 'mph'];
% Weight transfer time response plots:
figure(2);
subplot(4,1,1);
plot(t, W1l_arr);
legend_W1l_arr{i} = ['@ ' num2str(u*ftps2mph) 'mph'];
subplot(4,1,2);
plot(t, W1r_arr);
legend_W1r_arr{i} = ['@ ' num2str(u*ftps2mph) 'mph'];
subplot(4,1,3);
plot(t, W2l_arr);
legend_W2l_arr{i} = ['@ ' num2str(u*ftps2mph) 'mph'];
subplot(4,1,4);
plot(t, W2r_arr);
legend_W2r_arr{i} = ['@ ' num2str(u*ftps2mph) 'mph'];
end
% First figure:
figure(1);
subplot(3,1,1);
legend(legend_yaw_rate_arr);
grid on;
hold off;
subplot(3,1,2);
% legend(legend_drift_angle_arr);
grid on;
hold off;
subplot(3,1,3);
% legend(legend_roll_angle_arr);
grid on;
hold off;
% Second figure:
figure(2);
subplot(4,1,1);
legend(legend_W1l_arr);
grid on;
hold off;
subplot(4,1,2);
% legend(legend_W1r_arr);
grid on;
hold off;
subplot(4,1,3);
% legend(legend_W2l_arr);
grid on;
hold off;
subplot(4,1,4);
% legend(legend_W2r_arr);
grid on;
hold off;
end
function [t, delta_mod] = generate_input_signal(dt, t_initial, t_final)
% Generate the input:
t = linspace(t_initial, t_final, (t_final - t_initial)/dt);
% Generate the specified handwheel input:
delta = zeros(1, length(t));
% Input of 0° for 1 second:
delta(1, 1:1/dt) = 0;
% Input of +45° for 3 seconds:
delta(1, 1/dt:4/dt) = 0.707;
% Input of -45° for 3 seconds:
```

```
delta(1, 4/dt:7/dt) = -0.707;
% Input of 0° for 3 seconds:
delta(1, 7/dt:length(t)) = 0;
% Plot input:
figure;
plot(t, delta);
grid on;
title('steering input with no rate saturation');
xlabel('time (sec)');
ylabel('delta (rad)');
% Modify the steering input:
slope = 2*(2*pi); % rad/sec
% Smooth step applied at 1 second:
left_bound = 0;
right_bound = 0.707;
at = 1; % /dt;
delta_mod = slope_it_down(t, dt, at, delta, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Smooth step applied at 4 seconds:
left_bound = 0.707;
right_bound = -0.707;
at = 4; % /dt;
delta_mod = slope_it_down(t, dt, at, delta_mod, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Smooth step applied at 7 seconds:
left_bound = -0.707;
right_bound = 0;
at = 7; % /dt;
delta_mod = slope_it_down(t, dt, at, delta_mod, left_bound, right_bound, sign(right_bound - left_bound)*slope);
% Plot input:
figure;
plot(t, delta);
hold on;
plot(t, delta_mod, 'r');
grid on;
title('steering input with smoothed angle transitions');
xlabel('time (sec)');
ylabel('delta (rad)');
legend('Original input signal', 'Modified input signal');
end
function delta = slope_it_down(t, dt, at, delta, left_bound, right_bound, slope)
time_to_reach = (right_bound - left_bound) / slope; % time units
delta(1, at/dt:(at + time_to_reach)/dt) = left_bound + slope * (t(at/dt:(at + time_to_reach)/dt) - t(at/dt)); % * sign(right_bound -
left_bound);
end
```