CHALMERS
UNIVERSITY OF TECHNOLOGY

TME102 - Vehicle Motion and Control
Department - Mechanics and Maritime Sciences
Mobility Engineering, M.Sc.

CarMaker Project

Group 11:
Pietro Nardon
Vittorio Degiuli

September 22, 2024

Contents

1 Task 1: Normal force estimator

In this task it was required to develop a model-based estimator for the normal load acting on each wheel axle and then to compare the obtained results with the forces computed by CarMaker.

1.1 Normal force estimation

After creating the test track, as described in the project hand in, a model-based estimator for the normal loads, acting on each wheel axle, was created. The starting equation for the normal loads estimation is shown in (Eq. 1.1); only the front left wheel equations is presented for brevity.

$$\begin{aligned} F_{1z} = &\frac{l_r mg}{2(l_f+l_r)} - \frac{mh}{2(l_f+l_r)}a_x - \frac{m(hC_f(l_f+l_r)-h_{rc}(C_f l_f - C_r l_r))}{w(l_f+l_r)(C_f+C_r)}a_y \\ &+ \frac{C_f J_x}{w(C_f+C_r)}\ddot{\varphi}_x - \frac{h_{rc}J_z}{w(l_f+l_r)}\ddot{\varphi}_z + \frac{h_{rc}}{w(l_f+l_r)}M_{z,fx} \end{aligned} \tag{1.1}$$

The followiong assumptions were made in order to simplify the model:

- The tire cornering stiffness C_f and C_r were supposed to be equal, since the car mounts the same tires in both front and rear axles, and no big differences are expected between the two.
- $M_{z,fx}$ is supposed to be negligible, since the car doesn't have differential braking.

The obtained simplified equation, used to model the force, is shown in (Eq. 1.2).

$$F_{1z} = \frac{l_r mg}{2(l_f+l_r)} - \frac{mh}{2(l_f+l_r)}a_x - \frac{m(h(l_f+l_r)-h_{rc}(l_f-l_r))}{2w(l_f+l_r)}a_y + \frac{J_x}{2w}\ddot{\varphi}_x - \frac{h_{rc}J_z}{w(l_f+l_r)}\ddot{\varphi}_z \tag{1.2}$$

The variables a_x, a_y, $\ddot{\varphi}_x$ and $\ddot{\varphi}_z$ were taken directly from IMU sensor mounted on the car, considering the contribution of gravity. The parameters used in the model were found from CarMaker's model check and are shown in (Tab. 1)).

h	$0.678\,m$	w	$1.672\,m$
h_{RC}	$0.154\,m$	J_x	$1204.674\,kg/m^2$
l_f	$1.1933\,m$	J_z	$4029.903\,kg/m^2$
l_r	$1.7907\,m$	m	$2296.758\,kg$

Table 1: Model parameters

The distances from of the front and rear axles from the centre of gravity, were computed through a momentum balance, using the static equilibrium of the normal forces (Eq. 1.3), and are reported in (Tab. 1)).

$$\begin{cases} l_f = & \frac{wF_{Rz}}{F_{Fz}+F_{Rz}} \\ l_r = & \frac{wF_{Fz}}{F_{Fz}+F_{Rz}} \end{cases} \quad where \quad \begin{cases} F_{Fz} = & 13515.641\,N \\ F_{Rz} = & 9006.205\,N \end{cases} \tag{1.3}$$

The model was created inside Simulink as shown in (Fig. 1), where inside the MATLAB function colored in yellow, (Eq. 1.2) was inserted for each tire, as can be seen in (Code. A.1).

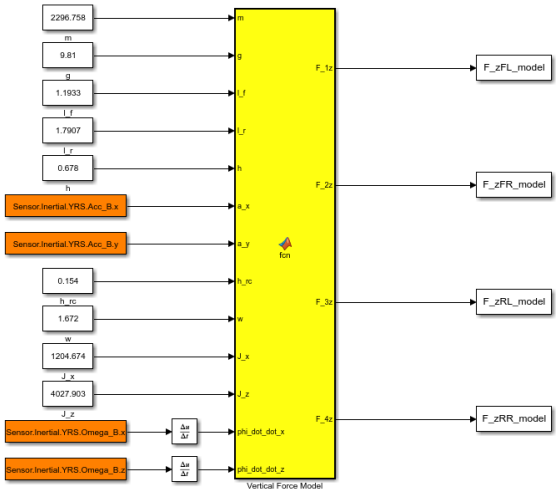


Figure 1: Normal load model inside Simulink

The results are plotted in (Fig. 2). Here it is clear that the modelled equation approximates well the normal forces computed directly by CarMaker. The main differences are visible in the transient regions, in particular at the moments in which the car is travelling along the upward and downward slopes and while is facing the road bump. In these situations the oscillations and values of the model are dumped down by the fact that in (Eq. 1.2) the pitch angle variations are not accounted for; as a result the model cannot perfectly approximate the load transfer that happens due to a change in the pitching angle.

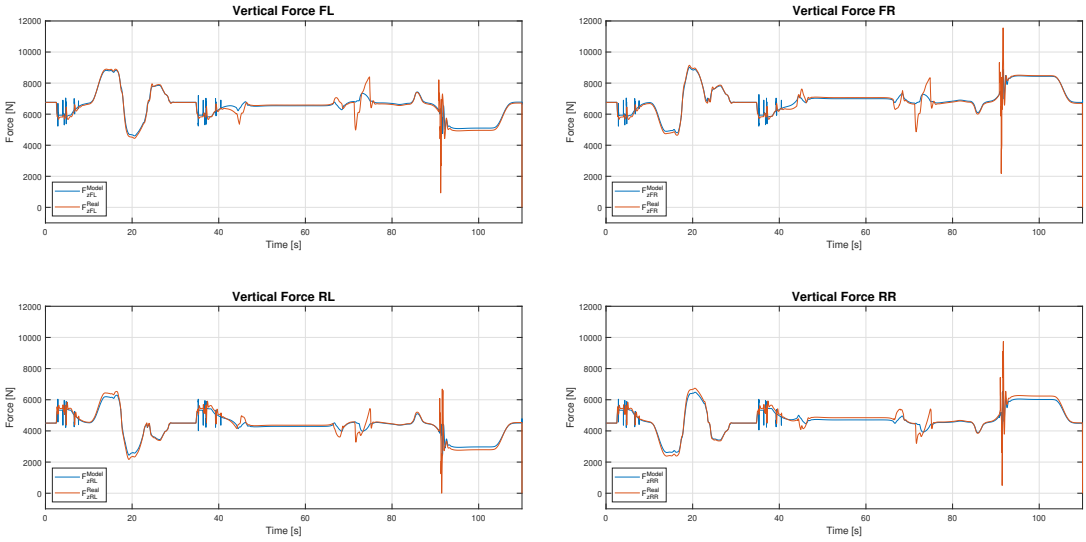


Figure 2: Modelled and real vertical forces

1.2 Warning function

A warning function was implemented in Simulink (Fig. 3) in order to check if any vertical wheel load falls below $\frac{1}{3}$ of the static case. The static vertical forces used as a limit were the one reported in (Eq. 1.3), while the dynamic wheel loads were the one modelled with (Eq. 1.2). The implemented warning function has been developed to have a value of 1, and then change to 0 if any vertical wheel load falls below the limit.

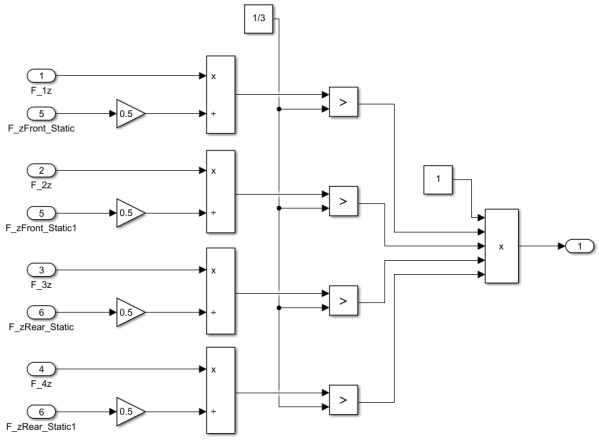


Figure 3: Warning function implementation in Simulink

The values of this function were plotted in (Fig. 4) where it can be seen that the warining is never triggered since its value and always remains equal to 1, meaning that during the entire simulation the vertical wheel loads never fall below the limit.

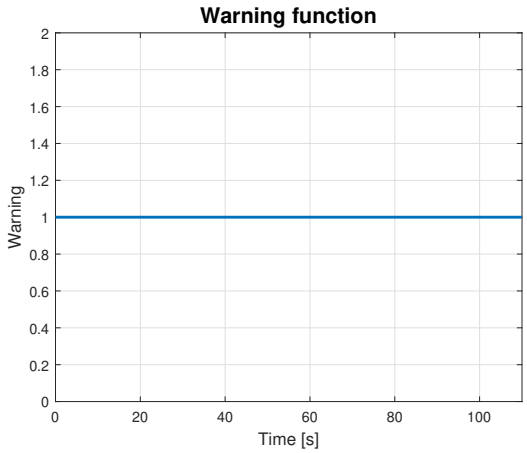


Figure 4: Warning function

Another proof of that can be found in (Fig. 5)), where all the dynamic wheel loads were plotted against their respective limits. Here is once again clear that no load fall below its limit.

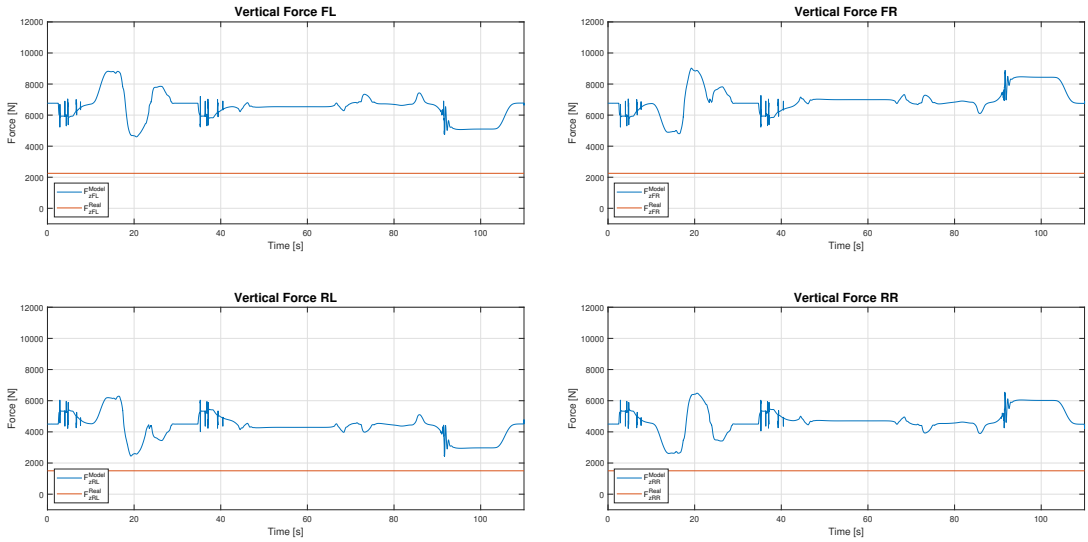


Figure 5: Modelled load forces with limits

1.3 RMS values

The RMS values for each wheel are presented in (Tab. 2). It can be seen that in general the rear wheels have a lower RMS value, meaning that the model is more precise in the rear wheels.

Vertical Force	RMS error
FL	292.1643
FR	282.4076
RL	269.1638
RR	263.4806

Table 2: RMS values

2 Task 2: Yaw rate frequency response

In this task the frequency response from steering angle to yaw rate was investigated comparing the results obtained from the CarMaker vehicle to the ones obtained analytically from the single-track model.

2.1 CarMaker frequency response function

The test was set up by creating a road with dimensions of 10x10km, to prevent the vehicle to end up out of the road. The manoeuvre carried out consisted in a steering characterized by the application of a sinus sweep of constant amplitude with increasing frequency at the steering wheel, while maintaining a constant speed.

To implement it in CarMaker, the steering from IPG driver was overwritten inside Simulink, by inserting a chirp block acting directly on the vehicle's steering wheel angle, while two discrete derivative blocks were used to define the vehicle's steering wheel angle derivatives, as shown in (Fig. 6). Inside the chirp block the initial frequency was set to $0\,Hz$, target time to $100\,s$ and frequency at target time to $20\,Hz$.

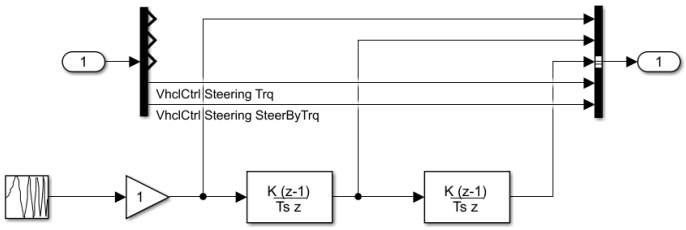
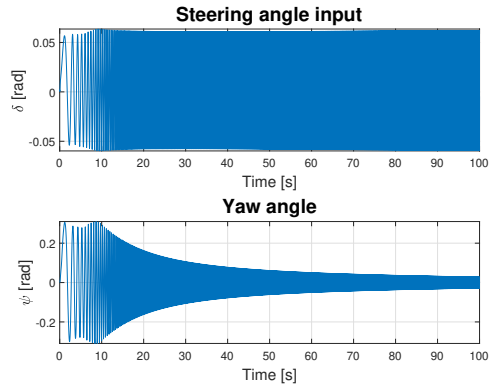


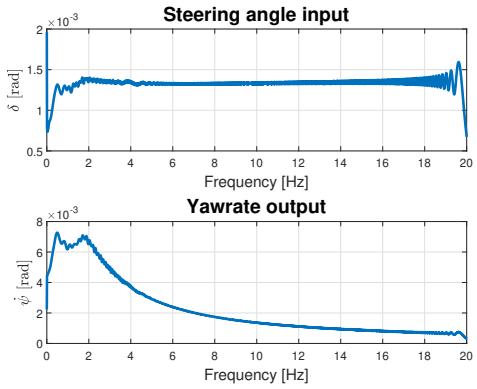
Figure 6: IGP driver overwrite

Regarding longitudinal control, the simulation was set by imposing to the vehicle a certain starting velocity, which had to be kept constant throughout the whole simulation time of $100\,s$; three different simulations at different speeds were conduced, respectively at $80km/h$, $110km/h$ and $130km/h$. The obtained time series results representing the road-wheel steering angle and the vehicle yaw rate at $80km/h$ are reported in (Fig. 7a).

The CarMaker computed yaw rate and steering angle time series were extracted and a discrete Fourier transform of both signals was carried out in MATLAB using a rectangular window with the aim of converting the signals in frequency domain obtaining the relative spectrum (Fig. 7b). Afterwards the frequency response function was computed as the ratio between the yaw rate spectrum and the steering angle spectrum, a mobile window average was carried out to reduce noise (Code. A.3), and the result was plotted in dB in a Bode diagram (Fig. 9).



(a) Signals in time domain



(b) Signals in frequency domain

Figure 7: Road-wheel steering angle and yawrate signals

2.2 Single-track model frequency response function

In order to implement correctly the single-track model was used the axles cornering stiffness were needed. Those were computed as shown in (Code. A.2), where the cornering stiffness of each tyre was found after testing the car inside CarMaker. The test consisted in a turn, at a constant speed of 40 km/h, were the steering wheel angle was increased linearly. This was done by substituting the chirp block, present in (Fig. 6), with a ramp block, where the slope was set to $0.08\,rad/s$. After running the simulation, the cornering stiffness of each tire was computed by dividing the lateral force by the slip angle, in (Eq. 2.1).

$$C_{ij} = \frac{F_y^{ij}}{\alpha_{ij}} \quad where \quad \begin{cases} i & = front, rear \\ j & = left, right \end{cases} \tag{2.1}$$

The tire cornering stiffness were converted into an equivalent axle cornering stiffness by summing the corresponding left and right wheel of each axle. The resulting cornering stiffness are shown in (Tab. 3).

	Left	Right	**Axle**
Front	$1.2119 \cdot 10^5 \frac{N}{rad}$	$1.2641 \cdot 10^5 \frac{N}{rad}$	$\mathbf{2.4759 \cdot 10^5 \frac{N}{rad}}$
Rear	$9.0248 \cdot 10^4 \frac{N}{rad}$	$9.5825 \cdot 10^4 \frac{N}{rad}$	$\mathbf{1.8607 \cdot 10^5 \frac{N}{rad}}$

Table 3: Cornering stiffness

After finding the axle cornering stiffness it was finally possible to plug them in inside the single-track model. The transfer function of the single-track model is presented in (Eq. 2.2), from which only the first line was selected in order to investigate the steering's influence on the yaw rate.

$$G = \frac{X}{\Delta} = (s\underline{I} - \underline{A})^{-1}\underline{B} = \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} \frac{-C_f - C_r}{mv_x} & \frac{C_r l_r - C_f l_f}{mv_x} - v_x \\ \frac{C_r l_r - C_f l_f}{J_z v_x} & -\frac{C_r l_r^2 + C_f l_f^2}{J_z v_x} \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} \frac{C_f}{m} \\ \frac{C_f l_f}{J_z} \end{bmatrix} \tag{2.2}$$

Using this transfer function, together with the data collected in (Tab. 1), the Bode plots, at different speeds, could be drawn in (Fig. 8). For each velocity used, the system always had a pair of complex conjugated stable poles about $1Hz$, with negative real part.

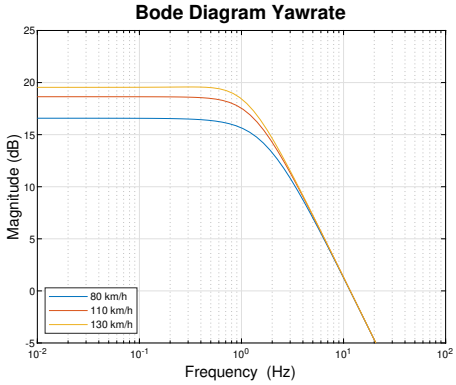


Figure 8: Bicycle model Bode plot

2.3 Comparison

A comparison between the CarMaker model and the single-track model was carried out, running the simulations at different vehicle speeds of 80, 110 and 130 km/h. In (Fig. 9), (Fig. 10), (Fig. 11) the transfer function obtained with CarMaker signals are plotted from $1Hz$ to $20Hz$.

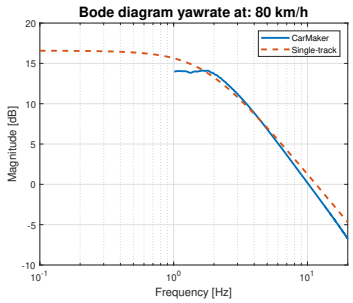


Figure 9: Yawrate Bode 80 $\frac{km}{h}$

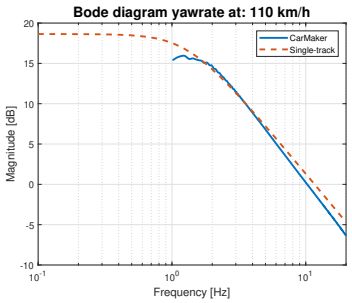


Figure 10: Yawrate Bode 110 $\frac{km}{h}$

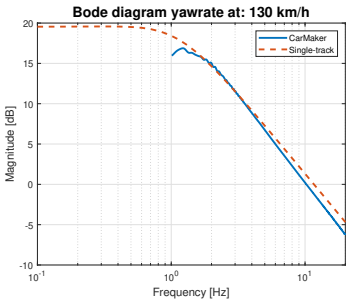


Figure 11: Yawrate Bode 130 $\frac{km}{h}$

By looking at the obtained results it can be stated that, at each considered speed, starting from $1Hz$, the analytical transfer function shows a similar trend compared to the transfer function computed using the CarMaker signals, meaning that the single track model represent a good approximation of the real car behavior. The small differences that can be appreciated are due to the simplifications introduced in the single track model that is not accounting for the dynamic effects present on the real vehicle. Moreover it can be seen that a pole is present in correspondence of $1Hz$ meaning that at higher frequency the vehicle capability to respond to rapid steering change is damped; this indicate that the vehicle is able to handle normal driving conditions and it remains stable even when subjected to rapid steering inputs.

3 Task 3: Speed controller

This task focus on the implementation of a self developed speed controller in the CarMaker model.

3.1 Cruise controller

To implement the cruise controller a straight long road was set up in CarMaker, then the required speed profile was built in Simulink and used as a reference for the controller (Fig. 12).

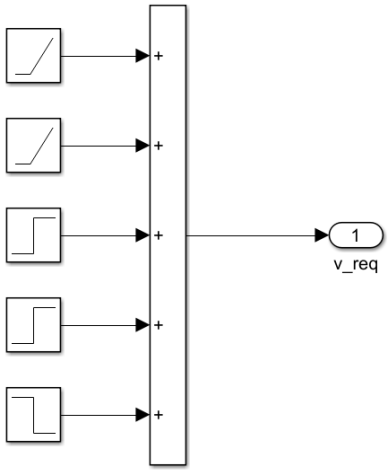


Figure 12: Speed profile generation

A proportional controller was then created in Simulink (Fig. 13), taking as input the difference between the required velocity and the output velocity computed by CarMaker, giving as outputs the gas and brake pedal signals that were used as input for CarMaker instead of the ones coming from the IPG driver.

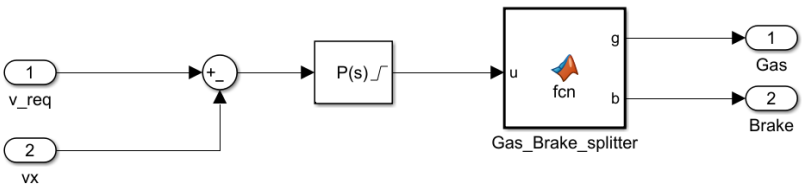


Figure 13: Cruise controller

The adopted proportional gain value of $k_p = 1.6$, is the result of a trial and error tuning aimed at obtaining a car velocity profile showing a stable behaviour and similar to the required one; the obtained result is shown in (Fig. 14).

By looking at (Fig. 14) it can be seen that the controller is able to make the car match the required velocity almost for each time instant without introducing instability; the differences that can be appreciated between the two profiles are due to the limited performances of the car that can not match a step behavior as the one used as reference.

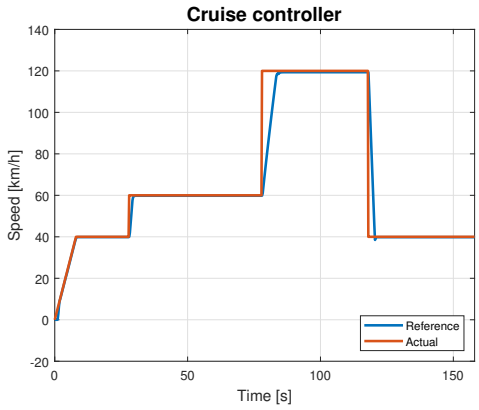


Figure 14: Cruise controller speed profile

3.2 Curve speed limitation

Starting from the previously developed controller, a speed limitation feature which limits the lateral acceleration within $2 m/s^2$ had to be implemented; to do so, the vehicle was ran on the test track used in (Sec. 1.1), considering as requested velocity $70 km/h$. To keep the lateral acceleration in the desired range, going from $-2m/s^2$ to $2m/s^2$, a function to limit the controller reference velocity was implemented starting from the simplified single track kinematic model of the car. The lateral acceleration has been defined as a function of the steering angle, of the velocity and of the wheelbase of the car (Eq. 3.1), then the maximum allowed velocity for a given lateral acceleration and small steering angle has been obtained (Eq. 3.2), this formula was used to decide the vehicle behaviour when cornering.

$$a_y = v\frac{v}{L}tan(\delta) \tag{3.1}$$

$$v_{max} = \sqrt{\frac{|a_{y,max}|L}{\delta}} \tag{3.2}$$

A MATLAB function computing the controller reference velocity as the minimum between the required velocity and the maximum allowed velocity has been implemented in the controller as shown in (Fig. 15). The functions "v_max_limiter" and "Gas_Brake_splitter" present in (Fig. 15) are shown in (Code. A.5) and (Code. A.6).

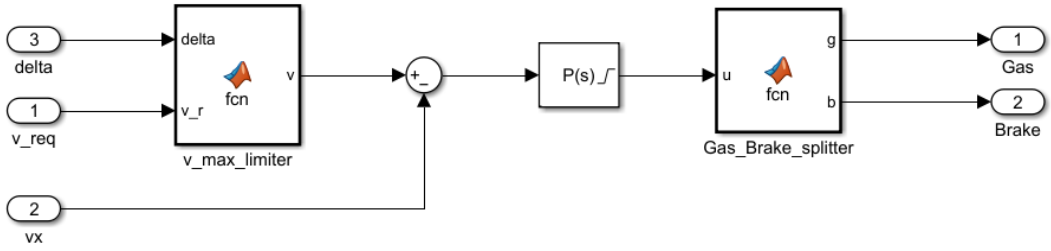
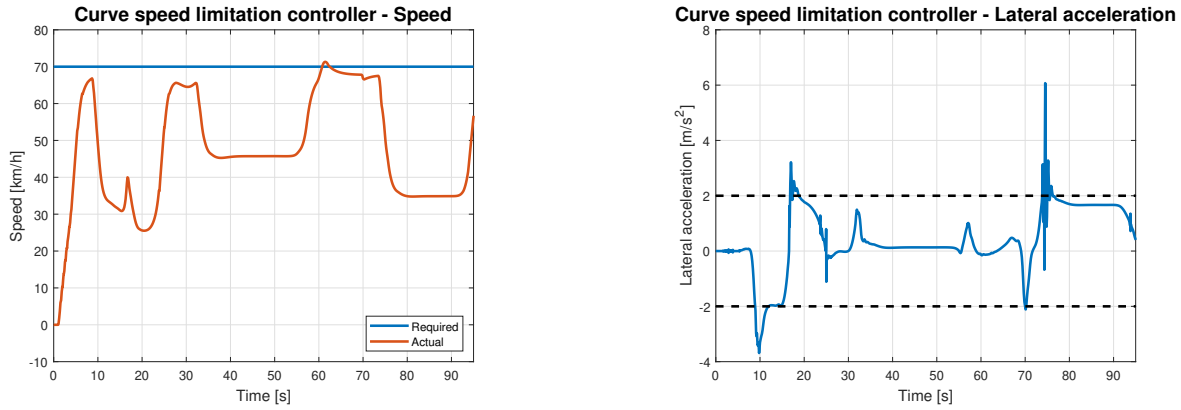


Figure 15: Curve speed limitation controller

To keep the lateral acceleration below the threshold of $2m/s^2$ the maximum acceleration value adopted in the limiting function was $1.8m/s^2$; this assumption was made to try to compensate for the huge simplifications introduced with the kinematic model formula that is not accounting for the dynamic phenomena going on in the real vehicle. The adopted proportional gain value of $k_p = 0.2$, is the result of a trial and error tuning aimed at obtaining a lateral acceleration profile limited between $-2m/s^2$ and $2m/s^2$ while introducing limited oscillations in the car behaviour.

The obtained results, both in terms of longitudinal speed and IMU measured lateral acceleration with gravity, are reported in (Fig. 16). It can be seen that the lateral acceleration overcame significantly the threshold of $2m/s^2$ only approaching the first turn, while in the other regions of the track only instantaneous peaks violate the limit; this is due to the fact that the adopted maximum allowed velocity estimator implemented in the proportional controller is not accurate enough to predict exactly the lateral acceleration that the car will face in tight turns as the one considered. In particular, the last peak overcoming the threshold is due to the presence of the road bump, which cannot be predicted in advanced by the implemented controller.



(a) Speed

(b) Lateral acceleration

Figure 16: Curve speed limitation results

4 Task 4: Handling diagram

A test was carried out to generate the handling diagram of the given car. Out of all the possible methods, it was chosen to run the car at high speed, in neutral, with a slowly increasing steering wheel angle. This choice allows to keep an almost constant speed throughout the test, slowly decreasing due to the rolling and aerodynamic resistances, without the application of torques on the wheels, preventing traction slip.

4.1 ICE vehicle setup

To implement this test in CarMaker the vehicle started with a speed of $200\,km/h$ and no steering. Regarding longitudinal dynamics, manual selection of pedal and gear was chosen inside CarMaker, and all parameters were set to 0, in this way the vehicle was put in neutral and no longitudinal acceleration nor braking was applied. The lateral dynamics was imposed through Simulink in a similar fashion to (Sec. 2.2), in this case a ramp with a slope of $0.05\,rad/s$ was set as steering wheel input.

4.2 Electric vehicle setup

To implement in CarMaker the new $50/50$ weight distribution, due to the different power-train architecture, it was necessary to change the vehicle's parameters. It was chosen to change the CoG position of the unladen mass of the vehicle, in the x-direction. After several trials varying the CoG position it was possible to obtain an equal force distribution on the axles; the results are shown in (Tab. 4). The simulation was ran using the same manoeuvre, but with the new weight distribution.

	Unladen mass CoG	Total mass CoG	Vertical Force Front	Vertical Force Rear
ICE	$2.916\,m$	$2.865\,m$	$1377.748\,N$	$919.012\,N$
Electric	$2.578\,m$	$2.567\,m$	$1148.380\,N$	$1148.380\,N$

Table 4: Weight distribution change

4.3 Handling diagram and tuning

The handling diagram of each vehicle was plotted using as y-axis the normalized lateral acceleration, taken from the IMU without considering gravity, while on the x-axis the side slip balance $-\delta + \frac{L}{R}$ was shown (Code. A.7). As can be seen in (Fig. 17), both vehicle are in the understeering region, which is where they are usually designed to work in, since it's safer. Already at a first glance it can be noted that the electric vehicle shows less understeer compared to the ICE version, being closer to the right side of the graph. This was expected since the weight distribution was moved from 60-40% to 50-50%, and moving the CoG towards the back means reducing the understeer. Moreover the maximum limit for lateral acceleration has increased since, when moving the CoG backwards, the front and rear tires will support loads in a more balanced way, generating lateral forces in a more efficient way and delaying the loss of grip at the front wheels.

The goal of the tuning is to make the electric configuration more understeering with the aim of behaving like the ICE one; in particular the main goal was to reach the same understeering gradient and have a similar handling diagram. The tuning was conduced with a trial and error approach, varying several parameters related to the suspensions architecture affecting the side force steering and roll steering of the vehicle, with the aim of making the rear axle stiffer and the front axle less stiff compared to the original ICE version in order to obtain a more understeering vehicle.

The side force steering of the vehicle was altered modifying the compliance parameters present in CarMaker (Tab. 5); in particular the compliance was increased on the front axle and reduced in the rear axle making them respectively more and less stiff than the originals.

		Compliance					
		Front			Rear		
		dtx $[m/N]$	drx $[rad/N]$	drz $[rad/N]$	dtx $[m/N]$	drx $[rad/N]$	drz $[rad/N]$
Original	F_x	$2.29 \cdot 10^{-6}$	0	$4.84 \cdot 10^{-7}$	$1.21 \cdot 10^{-6}$	0	$-8.92 \cdot 10^{-7}$
	F_y	0	$3.22 \cdot 10^{-6}$	$-4.98 \cdot 10^{-7}$	0	$3.05 \cdot 10^{-6}$	$6.43 \cdot 10^{-7}$
	F_z	0	0	0	0	0	0
Tuned	F_x	$3 \cdot 10^{-6}$	0	$6 \cdot 10^{-7}$	$1.2 \cdot 10^{-6}$	0	$-6 \cdot 10^{-7}$
	F_y	0	$4 \cdot 10^{-6}$	$-6 \cdot 10^{-7}$	0	$3 \cdot 10^{-6}$	$8 \cdot 10^{-7}$
	F_z	0	0	0	0	0	0

Table 5: Electric vehicle compliance tuning

On the other hand the roll steering of the vehicle was modified varying the values of kinematics parameters present in CarMaker (Tab. 6); the rate of change of displacement in each direction per unit of compression were reduced on the front axle and increased on the rear axle obtaining a vehicle stiffer in the rear part.

		Kinematics	
		Front	Rear
		Compr. $[m/m]$	Compr. $[m/m]$
Original	Transl. x	0.01	-0.14
	Transl. y	0.058	0.04
	Transl. z	1.0	1.0
Tuned	Transl. x	0.0075	-0.145
	Transl. y	0.054	0.042
	Transl. z	0.9	1.1

Table 6: Electric vehicle kinematic tuning

The obtained results for the ICE version, the original electric version and the modified electric version were plotted together in (Fig. 17).

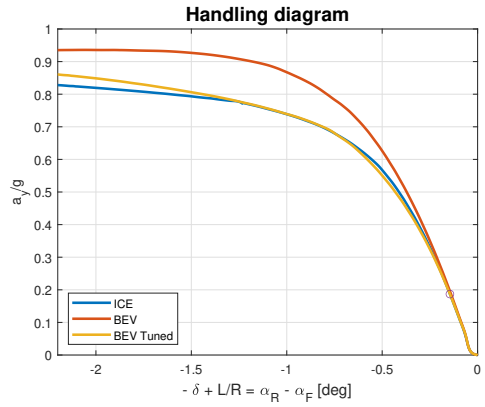


Figure 17: Handling diagram

In (Fig. 17) it can be seen that the modified electric version handling diagram match the behaviour of the ICE version up to a value of $-1.5\,^{\circ}$ of side slip balance; moreover the understeering gradient of the two configuration is almost equal as shown in (Tab. 7).

	ICE	BEV	BEV tuned
k_{us}	-1.290	-1.349	-1.289

Table 7: Understeering gradient

A MATLAB code

In this paragraph the MATLAB code used for calculations is presented.

The codes used in Task 1 and 3 are not shown since inside them are present only plots used to create the figures used in report.

A.1 Task 1: Vertical force estimator

```
function [F_1z, F_2z, F_3z, F_4z] = fcn(m, g, l_f, l_r, h, a_x,
   a_y, h_rc, w, J_x, J_z, phi_dot_dot_x, phi_dot_dot_z)


F_1z = (l_r*m*g)/(2*(l_f+l_r)) - (m*h*a_x)/(2*(l_f+l_r)) - m*a_y*(
   h*(l_f + l_r) - h_rc*(l_f - l_r))/(2*w*(l_r+l_f)) +  J_x*
   phi_dot_dot_x/(2*w) -  J_z*h_rc*phi_dot_dot_z/(w*(l_f + l_r));
F_2z = (l_r*m*g)/(2*(l_f+l_r)) - (m*h*a_x)/(2*(l_f+l_r)) + m*a_y*(
   h*(l_f + l_r) - h_rc*(l_f - l_r))/(2*w*(l_r+l_f)) -  J_x*
   phi_dot_dot_x/(2*w) +  J_z*h_rc*phi_dot_dot_z/(w*(l_f + l_r));
F_3z = (l_f*m*g)/(2*(l_f+l_r)) + (m*h*a_x)/(2*(l_f+l_r)) - m*a_y*(
   h*(l_f + l_r) + h_rc*(l_f - l_r))/(2*w*(l_r+l_f)) +  J_x*
   phi_dot_dot_x/(2*w) +  J_z*h_rc*phi_dot_dot_z/(w*(l_f + l_r));
F_4z = (l_f*m*g)/(2*(l_f+l_r)) + (m*h*a_x)/(2*(l_f+l_r)) + m*a_y*(
   h*(l_f + l_r) + h_rc*(l_f - l_r))/(2*w*(l_r+l_f)) -  J_x*
   phi_dot_dot_x/(2*w) -  J_z*h_rc*phi_dot_dot_z/(w*(l_f + l_r));
```

A.2 Task 2: Cornering stiffness and bicycle model

```
clc

close all

mod = 'generic';


%% Data

F_zF = 13515.641;
F_zR = 9006.205;
w = 2.984;
m = 2296.758;
J_z = 4027.903;
Time = F_zFL_model.Time;

l_f = w*F_zR/(F_zF + F_zR);
```

```
l_r = w*F_zF/(F_zF + F_zR);


%% Cornering stiffness

FyFL = FyFL - FyFL(15000);
FyFR = FyFR - FyFR(15000);
FyRL = FyRL - FyRL(15000);
FyRR = FyRR - FyRR(15000);


slipAng_FL = abs(slipAng_FL - slipAng_FL(15000));
slipAng_FR = abs(slipAng_FR - slipAng_FR(15000));
slipAng_RL = abs(slipAng_RL - slipAng_RL(15000));
slipAng_RR = abs(slipAng_RR - slipAng_RR(15000));

C_FL = (FyFL(23000)-FyFL(21000))/(slipAng_FL(23000)-slipAng_FL
   (21000));
C_FR = (FyFR(23000)-FyFR(21000))/(slipAng_FR(23000)-slipAng_FR
   (21000));
C_RL = (FyRL(23000)-FyRL(21000))/(slipAng_RL(23000)-slipAng_RL
   (21000));
C_RR = (FyRR(23000)-FyRR(21000))/(slipAng_RR(23000)-slipAng_RR
   (21000));

C_F = sum([C_FL, C_FR])
C_R = sum([C_RL, C_RR])

%% Plots

figure
nexttile
plot(rad2deg(slipAng_FL(20000:end -5)), FyFL(20000:end -5), '
   LineWidth', 2); hold on
plot(rad2deg(slipAng_FL([21000, 23000])), FyFL([21000, 23000]),'
   LineWidth', 2);
title('FL'); xlabel('Angle [deg]'); ylabel('Lateral Force [N]');
   grid on
nexttile
plot(rad2deg(slipAng_FR(20000:end -5)), FyFR(20000:end -5), '
   LineWidth', 2); hold on
plot(rad2deg(slipAng_FR([21000, 23000])), FyFR([21000, 23000]),'
   LineWidth', 2);
title('FR'); xlabel('Angle [deg]'); ylabel('Lateral Force [N]');
   grid on
nexttile
```

```
plot(rad2deg(slipAng_RL(20000:end -5)), FyRL(20000:end -5), '
   LineWidth', 2); hold on
plot(rad2deg(slipAng_RL([21000, 23000])), FyRL([21000, 23000]),'
   LineWidth', 2);
title('RL'); xlabel('Angle [deg]'); ylabel('Lateral Force [N]');
   grid on
nexttile
plot(rad2deg(slipAng_RR(20000:end -5)), FyRR(20000:end -5), '
   LineWidth', 2); hold on
plot(rad2deg(slipAng_RR([21000, 23000])), FyRR([21000, 23000]),'
   LineWidth', 2);
title('RR'); xlabel('Angle [deg]'); ylabel('Lateral Force [N]');
   grid on
sgtitle('Slip versus Lateral Force');


figure
nexttile
plot(Time, rad2deg(slipAng_FL), 'LineWidth', 2);
title('FL'); ylabel('Angle [deg]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, rad2deg(slipAng_FR), 'LineWidth', 2);
title('FR'); ylabel('Angle [deg]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, rad2deg(slipAng_RL), 'LineWidth', 2);
title('RL'); ylabel('Angle [deg]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, rad2deg(slipAng_RR), 'LineWidth', 2);
title('RR'); ylabel('Angle [deg]'); xlabel('Time [s]'); grid on
sgtitle('Slip Angle');


figure
nexttile
plot(Time, FyFL, 'LineWidth', 2);
title('FL'); ylabel('Force [N]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, FyFR, 'LineWidth', 2);
title('FR'); ylabel('Force [N]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, FyRL, 'LineWidth', 2);
title('RL'); ylabel('Force [N]'); xlabel('Time [s]'); grid on
nexttile
plot(Time, FyRR, 'LineWidth', 2);
title('RR'); ylabel('Force [N]'); xlabel('Time [s]'); grid on
sgtitle('Lateral Force');
```

```
%% Bicyle model

v_x_CM = mean(vx);
figure
for v_x = [80, 110, 130]/3.6
    a11 = (C_F+C_R)/abs(v_x);
    a12 = (C_F*l_f-C_R*l_r)/abs(v_x) + m*v_x;
    a21 = (C_F*l_f-C_R*l_r)/abs(v_x);
    a22 = (C_F*l_f^2 + C_R*l_r^2)/abs(v_x);
    A = -([m 0;0 J_z])\[a11 a12;a21 a22];
    B = ([m 0;0 J_z])\[C_F;C_F*l_f];
    C = [1 0;0 1];
    D = [0;0];
    sys = ss(A,B,C,D);
    [num,den] = ss2tf(A,B,C,D);
    s = tf('s');
    sys_yawrate = tf(num(2,:),den);
    sys_yawAng = sys_yawrate/s;

    % Plots
    opts = bodeoptions;
    opts.PhaseVisible = 'off';
    opts.FreqUnits = 'Hz';
    opts.XLabel.FontSize = 12;
    opts.YLabel.FontSize = 12;
    opts.YLim={[-5 25]};
    bodeplot(sys_yawrate,opts); grid on; hold on
    title('Bode Diagram Yawrate', 'FontSize', 15, 'FontWeight', '
       bold');
end
legend('80 km/h', '110 km/h', '130 km/h', 'Location', 'southwest')
```

A.3 Task 2: CarMaker FRF

```
clc
% clear all
close all

t = F_zFL_model.Time;
steering = SteerAng_FL;
yaw = yawRate;

%% Time signal

dt = t(6)-t(5);
fsamp = 1/dt;
N = length(t);

figure ()
subplot(2,1,1);
plot(t,steering);
xlabel('Time [s]', 'FontSize', 12);
ylabel('$\delta$ [rad]', 'FontSize', 12, 'Interpreter', 'latex')
grid on
axis tight
title('Steering angle input', 'FontSize', 15);

subplot(2,1,2);
plot(t,yaw);
xlabel('Time [s]', 'FontSize', 12);
ylabel('$\dot{\psi}$ [rad]', 'FontSize', 12, 'Interpreter', 'latex
    ')
grid on
axis tight
title('Yawrate', 'FontSize', 15)

%% Frequency signal

[S, freq] = fft_n(rectwin(N).*steering,fsamp);
[Y, freq] = fft_n(rectwin(N).*yaw,fsamp);

FRF=Y./S;
FRF = movmean(FRF, 25);

figure(2)
subplot(2,1,1)
plot(freq,abs(S), 'LineWidth', 2)
ylabel('$\delta$ [rad]', 'FontSize', 12, 'Interpreter', 'latex')
```

```
xlabel('Frequency [Hz]', 'FontSize', 12)
xlim([0 20])
title('Steering angle input', 'FontSize', 15)
grid on

subplot(2,1,2)
plot(freq,abs(Y), 'LineWidth', 2);
ylabel('$\dot{\psi}$ [rad]', 'FontSize', 12, 'Interpreter', 'latex
    ')
xlabel('Frequency [Hz]', 'FontSize', 12)
xlim([0 20])
title('Yawrate output', 'FontSize', 15)
grid on

subplot(3,1,3)
semilogy(freq,abs(FRF))
title('FRF=Y/S')
xlim([0 20])
xlabel('Frequency [Hz]')
ylabel('angle/angle')
grid on


figure(3)
semilogx(freq(102:end),20*log10(abs(FRF(102:end))), 'LineWidth',
    2)
title('Bode diagram yawrate', 'FontSize', 15)
xlim([0.1 20])
xlabel('Frequency [Hz]', 'FontSize', 12)
ylabel('Magnitude [dB]', 'FontSize', 12)
grid on; hold on


%% Single track model

% Data
C_F = 2.4759e5;
C_R = 1.8607e5;
F_zF = 13515.641;
F_zR = 9006.205;
w = 2.984;
m = 2296.758;
J_z = 4027.903;
Time = F_zFL_model.Time;

l_f = w*F_zR/(F_zF + F_zR);
```

```
l_r = w*F_zF/(F_zF + F_zR);



v_x = 130/3.6% [80, 110, 130]/3.6
a11 = (C_F+C_R)/abs(v_x);
a12 = (C_F*l_f-C_R*l_r)/abs(v_x) + m*v_x;
a21 = (C_F*l_f-C_R*l_r)/abs(v_x);
a22 = (C_F*l_f^2 + C_R*l_r^2)/abs(v_x);
A = -([m 0;0 J_z])\[a11 a12;a21 a22];
B = ([m 0;0 J_z])\[C_F;C_F*l_f]; %only steering as input
C = [1 0;0 1]; %out1=vy, out2=yawrate
D = [0;0];
sys = ss(A,B,C,D);
[num,den] = ss2tf(A,B,C,D);
s = tf('s');
sys_yawrate = tf(num(2,:),den);
sys_yawAng = sys_yawrate/s;%divide by s to get yawangle

% Plot
figure
opts = bodeoptions;
opts.PhaseVisible = 'off';
opts.FreqUnits = 'Hz';
opts.XLabel.FontSize = 12;
opts.YLabel.FontSize = 12;
opts.YLim={[-5 25]};
[bode_ampl, ~,  bode_freq]  = bode(sys_yawrate);
bode_ampl = squeeze(bode_ampl);
bode_freq = squeeze(bode_freq);
bodeplot(sys_yawrate,opts);grid on; hold on
title('Bode Diagram Yawrate', 'FontSize', 15, 'FontWeight', 'bold'
   );


figure(3)
semilogx(bode_freq/(2*pi), 20*log10(bode_ampl),'--', 'LineWidth',
   2)
title(['Bode diagram yawrate at: ', num2str(v_x*3.6), ' km/h'], '
   FontSize', 15)
legend('CarMaker','Single-track')
```

A.4 Task 2: Normalized fast Fourier transform

```
%% This function does the normalisation of the output of the fft
   function
% Input:
%   - data: input data matrix with size [r,c], with r samples and
   c signals
%   - fsamp: sampling frequency
% Output:
%   - norm_sp: normalized spectrum (positive frequencies)
%   - freq_vec: frequency vector

function [norm_sp, freq_vec]=fft_n(data,fsamp)

dim=size(data);

if dim(2)>dim(1)
    data=data';
end

N=length(data);
df=fsamp/N;

if (N/2)==(floor(N/2))

    freq_vec=[0:df:(N/2*df)]';
    NF=length(freq_vec);
    sp=fft(data,[],1);
    norm_sp(1,:)=sp(1,:)/N;
    norm_sp(2:N/2,:)=sp(2:N/2,:)/(N/2);
    norm_sp(N/2+1,:)=sp(N/2+1,:)/N;

else

    freq_vec=[0:df:((N-1)/2)*df]';
    NF=length(freq_vec);
    sp=fft(data,[],1);
    norm_sp(1,:)=sp(1,:)/N;
    norm_sp(2:(N+1)/2,:)=sp(2:(N+1)/2,:)/(N/2);

end
```

A.5 Task 3: v max limiter

```
function v = fcn(delta,v_r)

l_r = 1.7907;
l_f = 1.1933;
L = l_r + l_f;

a_max = 1.8;

v_max = sqrt(a_max*L/abs(delta));

if v_r>v_max
    v = v_max;
else
    v = v_r;
end
```

A.6 Task 3: Gas Brake splitter

```
function [g,b] = fcn(u)
if u>=0
    g = u;
    b = 0;
else
    b = -u;
    g = 0;
end
```

A.7 Task 4: Handling diagram

```
clc

close all


%% neutral & ramp steering

g = 9.81;

l_r = 1.7907;   %m
l_f = 1.1933;   %m
L = l_r + l_f;

% SteerAng = (SteerAng_FL + SteerAng_FR)/2;
% save('nrs_BEV_Try',"Acc_y_IMU","SteerAng","vx")

clear('SteerAng', 'Acc_y_IMU', 'vx')

%% Plots

% ICE
load('nrs_ICE');

R = vx.^2./Acc_y_IMU;
yy = Acc_y_IMU/g;
xx = (-SteerAng + L./R)*180/pi;
ku = yy(2000)/xx(2000)
figure(8)
plot(xx(1:25000), yy(1:25000), 'LineWidth', 2); hold on; grid on
xlim([-2.2 0])

% BEV
load('nrs_BEV');

yy = Acc_y_IMU/g;
R = vx.^2./Acc_y_IMU;
xx = (-SteerAng + L./R)*180/pi;
ku = yy(2000)/xx(2000)
ku_l = xx*ku;
figure(8)
plot(xx(1:25000), yy(1:25000), 'LineWidth', 2);

% BEV New
load('nrs_BEV_new');
```

```
yy = Acc_y_IMU/g;
R = vx.^2./Acc_y_IMU;
xx = (-SteerAng + L./R)*180/pi;
ku = yy(2000)/xx(2000)
figure(8)
plot(xx(1:25000), yy(1:25000), 'LineWidth', 2); title('Handling
   diagram', 'FontSize', 15);
plot(xx(2000),yy(2000), 'o');
xlabel('- \delta + L/R = \alpha_R - \alpha_F [deg]', 'FontSize',
   12);
ylabel('a_y/g'); legend('ICE', 'BEV', 'BEV Tuned', 'Location', '
   southwest');
```