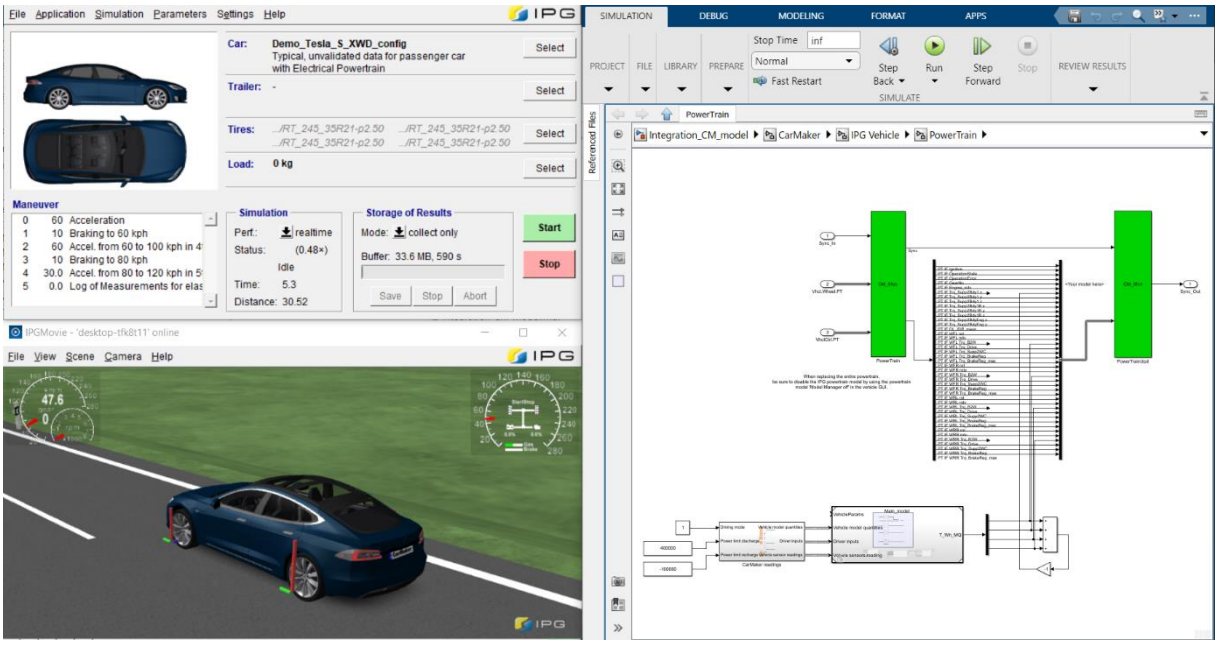
CHALMERS




# Propulsion development model for MiL/HiL/DiL purposes

**TME180 Automotive Engineering Project 2024**


ABNER ANKIT LAWRENCE
GABRIELE ARTESE
MIGUEL GAMEZ BERRAL
SANJANA HASSAN ANANDA KUMAR
SIMRAN JOY
SUMANTH PAI

*Department of Mechanics and Maritime Sciences*
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2024

# Propulsion development model for MiL/HiL/DiL purposes
## TME180 Automotive Engineering Project 2024



Supervisor: Bengt Jacobson, Department of Mechanics and Maritime Sciences
Supervisor: Albin Gröndahl, Zeekr Tech EU
Examiner: Alexey Vdovin, Department of Mechanics and Maritime Sciences

Studentarbeten – Mekanik och maritima vetenskaper (M2) – Projektarbete
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

ABSTRACT

Virtual simulations are increasingly critical for reducing vehicle development timelines, a priority in the automotive industry. Driving simulators and vehicle simulation software are essential tools that enable manufacturers to test and validate vehicle subsystems and overall performance prior to constructing physical prototypes. In the case of electric vehicles, early virtual verification of subsystems is particularly important, given the modular nature of the driveline platform. Modular simulation models further enhance this process by allowing companies to develop custom models for specific subsystems, which can be exchanged with subsystem suppliers and vehicle manufacturers, facilitating the creation of virtual prototypes and streamlining the design workflow.

This report details the implementation of a modular propulsion model specifically designed for electric vehicles, aimed at supporting feature verification across various simulation environments, including Model-in-the-Loop (MiL), Hardware-in-the-Loop (HiL), and Driver-in-the-Loop (DiL). The model's primary goal is to accelerate development cycles, reduce costs, and improve the reliability and performance of electric vehicle drivetrains. Key features such as torque vectoring, slip control, and torque distribution are incorporated, making the model compatible with diverse powertrain configurations, including rear-wheel drive (RWD) and all-wheel drive (AWD). Integration with CarMaker allows for realistic testing and validation of the model under various driving conditions.

The results indicate that the model effectively simulates vehicle dynamics and control, responding accurately to driver inputs, within its design domain. The model offers the opportunity to the customer to easily modify and calibrate each subsystem, which was the main objective of the project. However, some limitations and assumptions have to be considered, making future work on the model reasonable.

**Keywords:** Electric vehicles, propulsion model, powertrain, torque split, torque vectoring, slip control, Model-in-the-Loop (MiL), Hardware-in-the-Loop (HiL), CarMaker for Simulink

ABBREVIATIONS

| | |
|---|---|
| EV | Electric Vehicle |
| MiL | Model-in-the-Loop |
| HiL | Hardware-in-the-Loop |
| DiL | Driver-in-the-Loop |
| FWD | Front-Wheel Drive |
| RWD | Rear-Wheel Drive |
| AWD | All-Wheel Drive |
| ADAS | Advanced Driver Assistance Systems |
| CAE | Computer-Aided Engineering |
| PID | Proportional-Integral-Derivative |
| LuT | Look-up Table |
| smf | S-shaped membership function |
| GUI | Graphical User Interface |
| OPD | One Pedal Driving |
| CM | CarMaker |
| CM4SL | CarMaker for Simulink |
| XWD | Either RWD/FWD/AWD |

# Table of contents

# 1 Introduction

## 1.1 Background

More than 10 million electric cars were on the world's roads in 2020 [1]. Their rapid evolution is driven by continuous advancements in powertrain technologies and control systems. A well-designed propulsion architecture is the key to manage vehicle dynamics, energy efficiency and performance. Furthermore, modern vehicles incorporate a range of subsystems such as battery management, torque distribution and Advanced Driver Assistance Systems (ADAS), each contributing to overall vehicle behavior.

## 1.2 Goal

The project goal is to create an environment-agnostic propulsion development model for feature verification in a virtual environment. By creating a simulation model, the project aims to accelerate the development and test cycles, reducing costs and improving reliability and performance.

The model should be adaptable for different configurations, such as rear-wheel drive (RWD), all-wheel drive (AWD), and multi-motor setups, and integrate essential control features like torque vectoring and slip control. It should also support real-time simulation environments such as Driver-in-the-Loop (DiL), Hardware-in-the-Loop (HiL), and Model-in-the-Loop (MiL), making it usable for both offline computer-aided engineering (CAE) and real-time testing.

## 1.3 Methodology

Several modules can be combined in different ways to carry out different powertrain configurations. Parametrization, such as throttle map, driving modes and motor setup, are considered. One of the objectives is to make them easily adjustable via user input or external scripts. The main features requested by the customer are torque vectoring, axle torque split and slip control. The complete model will be then made compatible with CarMaker and other CAE tools, allowing the use on dynamic simulators.

All the design steps are supported by validation and testing phases. The model is challenged against known vehicle behavior benchmarks, ensuring it responds accurately to torque requests, traction limits, and vehicle stability requirements. Baseline test scenarios (e.g. high traction, low traction and aggressive driving) are so created.

In conclusion, this approach will provide a versatile, scalable Simulink model capable of handling modern EV propulsion needs while being flexible enough to support real-time testing and future expansions.

## 1.4 Project deliverables

The primary deliverable of this project is a Simulink model of a propulsion system featuring scalability and versatility to meet the desired specifications. Additionally, full integration with CarMaker is provided as part of the deliverables.

The project employed an iterative design and verification approach, producing multiple intermediate deliverables throughout its development process.

In particular:

- During the concept phase, we established a powertrain and controller strategy based on design choice. Multiple conceptual designs have been compared and the best one has been chosen based on parameters such as robustness, complexity, development time, etc.;

- During the development loop one, the powertrain and a basic version of the controller (C1.0) have been designed and verified in verification loop one;
- During the development loop two we built, on the verified basic version of the controller, a more complex and realistic model (C2.0) which we again integrated and verified with the powertrain. Controller C2.0 have tunability options which can be verified with SiL. (deliverable of verification loop two).

### 1.5 Limitations

This project focuses on evaluating the functionality of the propulsion system, with the brake system and other vehicle subsystems excluded from consideration. To account for these omitted components, complete vehicle simulations can be performed using software as CarMaker, which models all remaining subsystems.

Furthermore, the scope of the study is limited to conventional electric powertrains, characterized by the absence of mechanical connections between the axles, as typically found in combustion-engine all-wheel-drive systems. Advanced drivetrain features, such as clutches for motor decoupling or mechanical differential locks, will not be considered. This approach limits the model complexity, facilitating a more detailed exploration of control strategies for electric propulsion systems.

## 2 Theory

### 2.1 Torque split

In cars, torque split refers to how the engine's power is distributed between the front and rear axles. Active torque split involves dynamic adjustments made by the vehicle's drivetrain system, which continuously varies the torque distribution between the wheels based on factors like road conditions, speed, and driving style. This allows for optimal traction, stability, and low energy consumption. In contrast, passive torque split uses a fixed ratio of power distribution, often relying on mechanical components like a center differential or viscous coupling to deliver a pre-set torque balance without any active control or adjustment.

### 2.2 Torque vectoring

Torque vectoring is a technology used in modern vehicles to enhance handling and stability by independently controlling the distribution of torque to each wheel. It typically works by adjusting the power sent to individual wheels based on driving conditions or steering inputs, allowing for improved traction and responsiveness. For example, during cornering, more torque can be directed to the outside wheels to reduce understeering or to the inside wheels to minimize oversteering. While torque vectoring is commonly achieved using electronic differentials, clutches, or other advanced mechanisms, it can also be implemented by braking on one side of an axle with an open differential. Although this braking-based method is outside the scope of this project, it is worth mentioning as part of the general theoretical discussion on torque vectoring. Overall, these methods help optimize vehicle dynamics and performance by enhancing control and precision.

### 2.3 Slip control

Slip control based on driver input is a system that adjusts the level of wheel slip to optimize traction and stability according to the driver's actions, such as throttle, braking, and steering. By monitoring these inputs, the system can predict and manage the amount of slip (the difference between wheel rotation and vehicle speed) to prevent excessive wheel spin or loss of traction. For

example, if the driver continues to accelerate on low-grip surfaces, the system may allow more slip in the wheels but still prevent them from spinning excessively. This helps maintain control and ensures a safer driving experience, adjusting dynamically to both the driver's intentions and road conditions.

# 3 Modeling

## 3.1 Model overview

The model is divided into three main blocks: main control unit, torque management and powertrain. The main control unit is responsible for managing the total acceleration requested by the driver, accounting for the power availability. The torque management subsystem, instead, processes the overall torque demand (single signal) at the wheel, distributing it actively or passively among the wheels and arbitrating between features like torque vectoring and slip control. Finally, the powertrain block models the physical powertrain components across various configurations, including motors, transmission and driveshaft. A complete overview of the model is represented in Figure 1.

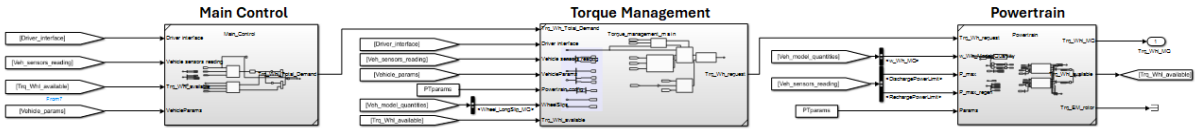



Figure 1: Model overview

The main challenge of the model is that several possibilities generated by powertrain architectures and togglable features (e.g. TV on/off, slip control on/off , driving mode) lead to several subsystems and non-trivial activating logics. Our solution is to use 'variant subsystems', enabling specific block activation based on pre-defined parameters and ensuring that only one configuration is run for the simulation.

Each subsystem is referenced in the project so that modifications and additional model implementation is possible both in the specific subsystem or in the complete model. The signals are transmitted as buses throughout the complete model, recalled when required through *From*/*Goto* blocks.

The model is organized as a MATLAB Simulink project, with automatic initialization through the use of background scripts. The final user is able to easily run simulations just modifying a list of parameters, representing the subsystem specifics, detailed in the Appendix.

## 3.2 Main control unit

The main control unit plays a central role in managing vehicle acceleration and power distribution by interpreting driver inputs depending on high-level functionalities such as driving modes. Its primary objective is to translate the driver's initial input, such as the position of the gas pedal, into a precise total torque demand at the wheel level, adjusting for the current state of the vehicle.

The following scheme represents the basic structure and working principle of the main control unit:

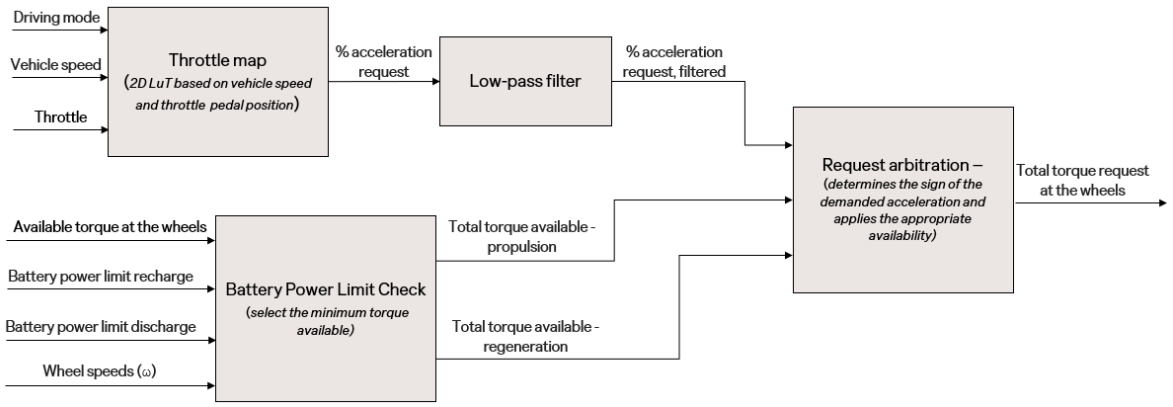



Figure 2: Main control unit structure

As well explained by Guzzella et al. [2], the torque request is generally determined using a look-up table that considers pedal position and wheel or vehicle speed. The principle guiding this approach is that full pedal depression (100%) corresponds to the maximum torque output achievable by the powertrain at the given speed, while zero pedal depression (coasting) emulates the sensation of "engine braking" experienced in conventional internal combustion engine (ICE) vehicles. Intermediate pedal positions are typically interpolated, either linearly or, as discussed in this context, non-linearly.

The main control unit is responsible for interpreting the driver's intent, specifically the requested level of acceleration as a percentage of the maximum available. To achieve this, various driving mode-dependent "maps" are employed. Each "map" is a two-dimensional look-up table that takes the gas pedal position and the vehicle's current speed as inputs and gives as output an interpolated percentage of the available acceleration or torque. This percentage is then multiplied by the total availability to calculate the actual total request at the wheel level. The result is a single signal transmitted to the torque management subsystem.

If the throttle map specifies demanded acceleration values across all speeds within certain regions, the limitations, such as decreasing traction force at higher speeds, are inherently addressed through the current torque availability. This design is clearly visible in the respective Unit Testing section, specifically in Fig. 36.

The torque availability in the system is determined comparing two inputs: the total torque available from the powertrain at the current instant (provided by the powertrain) and the battery limits, represented by constants for propulsion and regeneration. These latter values will be divided by the average wheel speed to compute the actual battery available torque.

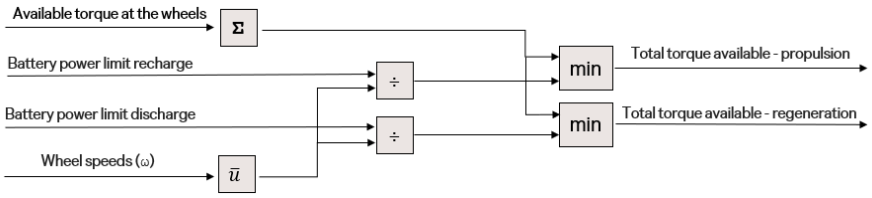



Figure 3: Battery power limit check structure

The system then identifies whether the requested acceleration is positive (propulsion) or negative (regeneration) through two saturation blocks. Depending on the sign, the appropriate limit—either the propulsion or regeneration torque limit—is applied through separate minimum

functions. These filtered outputs ensure that the torque availability aligns with the instantaneous battery and powertrain constraints.

Finally, the system consolidates these outputs into the actual total torque demand at the wheel using a summation block. This ensures that the appropriate torque is applied to meet the driver's request while respecting both powertrain and battery limitations.

The final Simulink model is here represented:

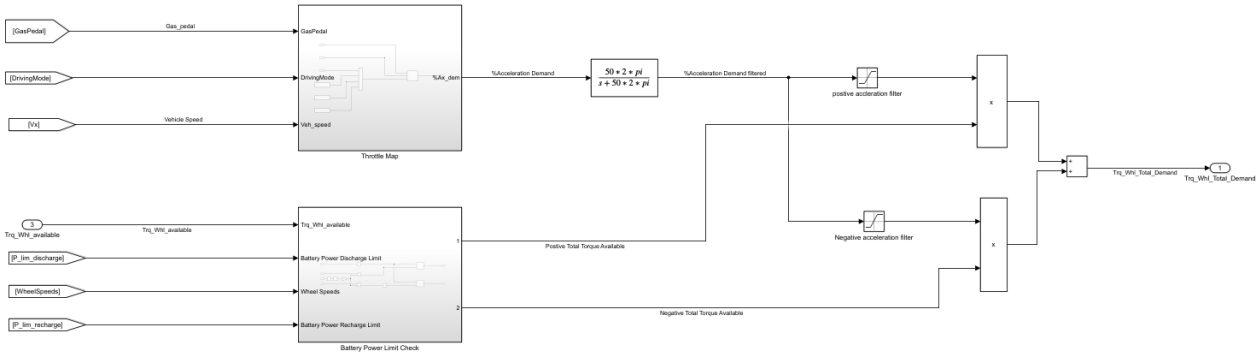



Figure 4: Main control unit - Simulink model

Several maps can be implemented and the selection is arbitrated through the driving mode parameter. Each map is associated with a unique scalar value, enabling efficient selection through a multi-port switch. In this application, three driving modes are considered: eco (soft), sport (aggressive) and custom (intermediate). The map creation process is fully parametrized, meaning that the maps for each mode are generated through systematic adjustment of key parameters.

As explained, the throttle maps are implemented as two-dimensional look-up tables, which can be visualized as three-dimensional surfaces. The axes of these surfaces correspond to the speed of the vehicle, the position of the gas pedal, and the percentage of available acceleration. Using acceleration instead of force as the output of the pedal map simplifies driver intent interpretation, aligns with natural expectations, and enhances system efficiency. Drivers intuitively associate pedal inputs with acceleration rather than force, making acceleration-based maps more intuitive and predictable. This approach eliminates dependency on vehicle mass, which can vary with passengers or cargo, simplifying tuning and ensuring consistent performance. Acceleration outputs also streamline torque distribution to the wheels, avoiding intermediate calculations and enhancing computational efficiency.

The maps are indeed designed to align closely with the driver's intent, ensuring intuitive and predictable responses to input. The demanded acceleration is modeled to vary between $-30\%$, representing regenerative braking, and $100\%$, indicating maximum propulsion. Understanding the nature of this surface requires analyzing its behavior across two primary factors: gas pedal position and vehicle speed.

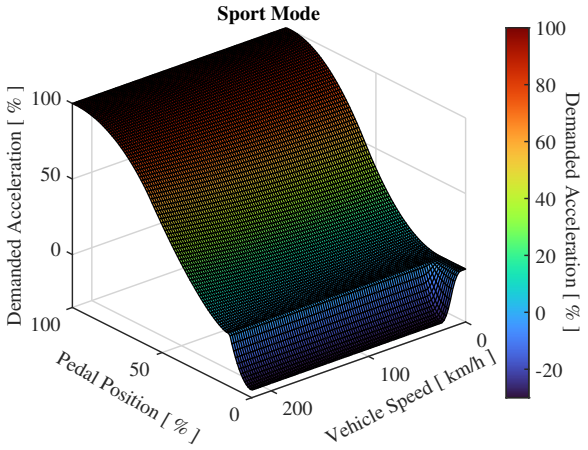


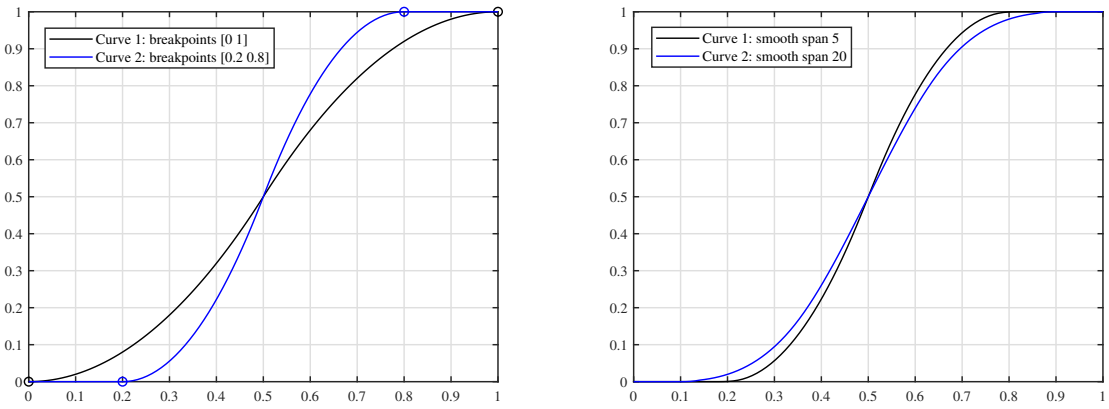

Figure 5: Map example - Sport mode

The relationship between the gas pedal position and the demanded acceleration is modeled using s-shaped functions. These functions are used to create a sensitive and precise response under typical driving conditions while ensuring stability and safety in extreme scenarios. At mid-range pedal positions, the response is intentionally more sensitive, allowing for fine control during tasks such as overtaking or gradual acceleration. Near the extremes —at very low or very high pedal inputs— the response is instead attenuated. This reduces fast acceleration changes at low speeds, which enhances handling during maneuvers like parking, and prevents excessive sensitivity at high speeds, where even minor adjustments could destabilize the vehicle.

The S-shaped functions are built using MATLAB's *smf* (S-shaped membership function) and smooth utilities, which together allow for a flexible three-parameter configuration. These parameters define the start and end breakpoints of the response curve (Figure 6a) and its overall smoothness span (Figure 6b). The smoothness span controls the level of curve smoothness; higher values produce smoother results but risk oversmoothing, potentially losing important details. Variations in the smoothness span may also result in slight changes to the breakpoint values.

The following two figures illustrate a general tuning example: the *smf* function is used to generate all the S-curves in the pedal map, with the axes varying according to the configuration parameters. As shown in Figure 7a and Figure 7b, the pedal map primarily includes three S-curves. Two of these represent the variation of demanded acceleration relative to pedal position in the propulsion and regeneration regions. The third curve models the transition between zero demand and full regeneration at low vehicle speeds.

Thus, the three S-curves in the pedal map can be flexibly manipulated using MATLAB's *smf* tool, as demonstrated in Figure 6a and Figure 6b.

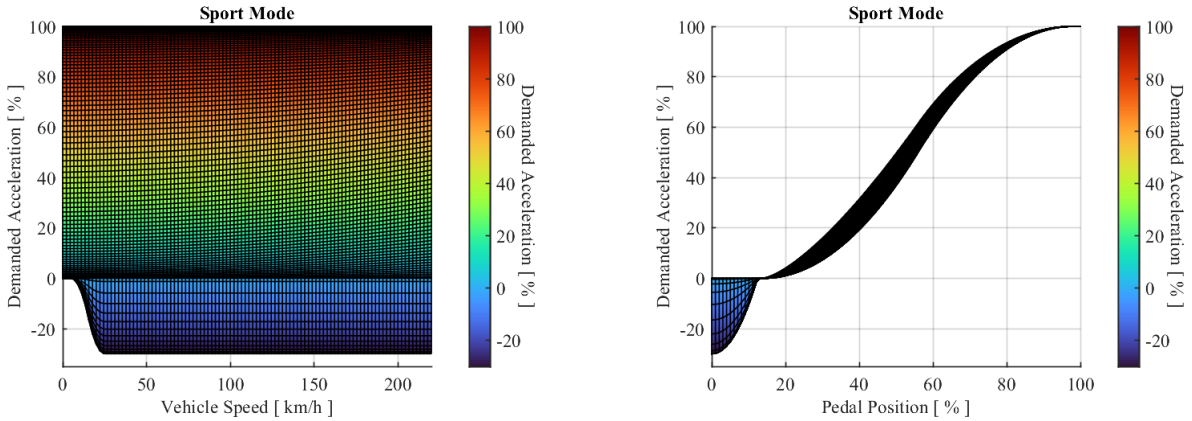



(a) Breakpoints variation

(b) Smoothness span variation

Figure 6: MATLAB smf parameters graphical example

The influence of vehicle speed on the demanded acceleration is integrated into the map to address specific challenges, particularly in the regenerative braking region. At low speeds, a dead zone, i.e. null demanded acceleration, is introduced up to a parametrized gas pedal position value - here 15%. This avoids unintended requests for negative torque, which could lead to reverse mode when low traction forces are required. As speed increases, a smooth ramp in regeneration levels begins. This transition is shaped using again an s-shaped function, with parameters defining its start and end points —here 5 and $25km/h$. At higher speeds, the regenerative demand stabilizes at a constant level. In the positive demand region (propulsion), the variation with vehicle speed is less pronounced but still significant. At higher speeds, the system slightly amplifies the acceleration demand, ensuring the driver perceives a responsive system despite potentially reduced power availability. This behavior is achieved through the interpolation of three distinct curves, ensuring that full acceleration is still achievable at maximum throttle. The curves are parameterized to control their placement and definition, allowing precise tuning of the acceleration response across the speed range. In this application, the three curves are still s-shaped functions with their already listed control parameters. These variations are observable in Figure 7a and Figure 7b.




(a) Demanded acceleration vs. vehicle speed

(b) Demanded acceleration vs. pedal position

Figure 7: Throttle map speed variation

By combining these two dimensions—pedal position and vehicle speed—the maps achieve a bal-

ance between performance, controllability, and safety. The modular parameter structure allows for customization across different driving modes, ensuring that the system adapts effectively to diverse operational requirements.

The described map is based on the concept of "one-pedal driving" (OPD), a mechanism that enhances driver control and feedback by allowing both positive and negative accelerations—corresponding to propulsion and braking torques—to be managed exclusively through the accelerator pedal. This approach provides a more intuitive driving experience, as the driver can modulate speed and deceleration seamlessly without needing to switch between the accelerator and brake pedals.

## 3.3 Torque management

The total torque request at the wheels is split to each wheel, according to the activated features, by the torque management subsystem. This block consists of the following subsystems:

- Torque split
- Torque vectoring
- Torque distribution
- Slip control
- Arbitration

Each block is designed for modularity so that it can be replaced without disrupting the functioning of other connected subsystems.

A general overview of the torque management organization is reported in Figure 8.

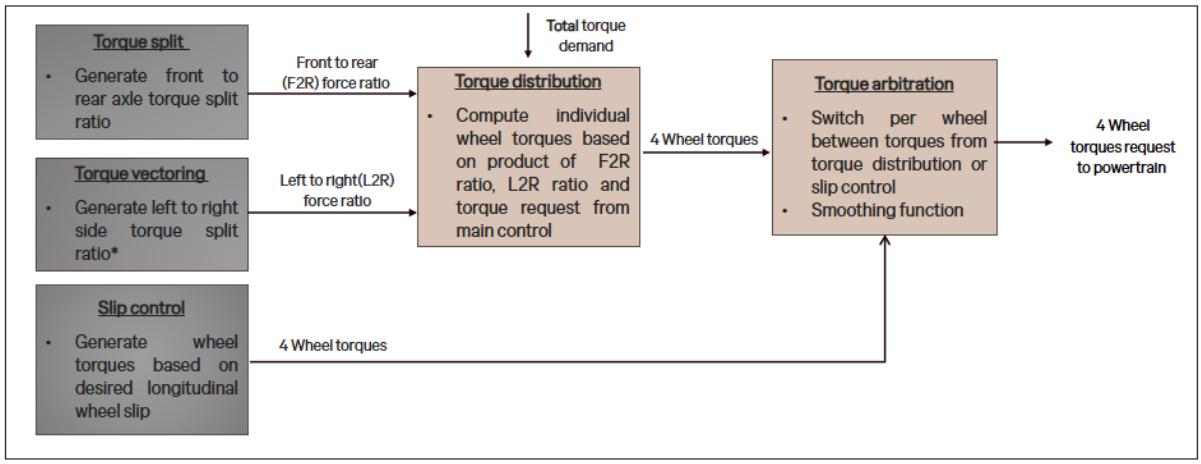



Figure 8: Torque management - block diagram

Torque or force ratios are used to enhance the flexibility of the structure, which depends on the total torque demand in a successive stage. In particular, torque vectoring ratios are computed using the vehicle coordinate system defined by ISO 8855-2011.

### 3.3.1 Torque split

The purpose of this subsystem is to distribute the torque between front and rear axles. The output from this block is the force ratio available at the front and rear axles.

The force ratio is computed through two types of torque split: active and passive. Passive torque split distributes torque according to a fixed ratio specified as a parameter by the user, using the

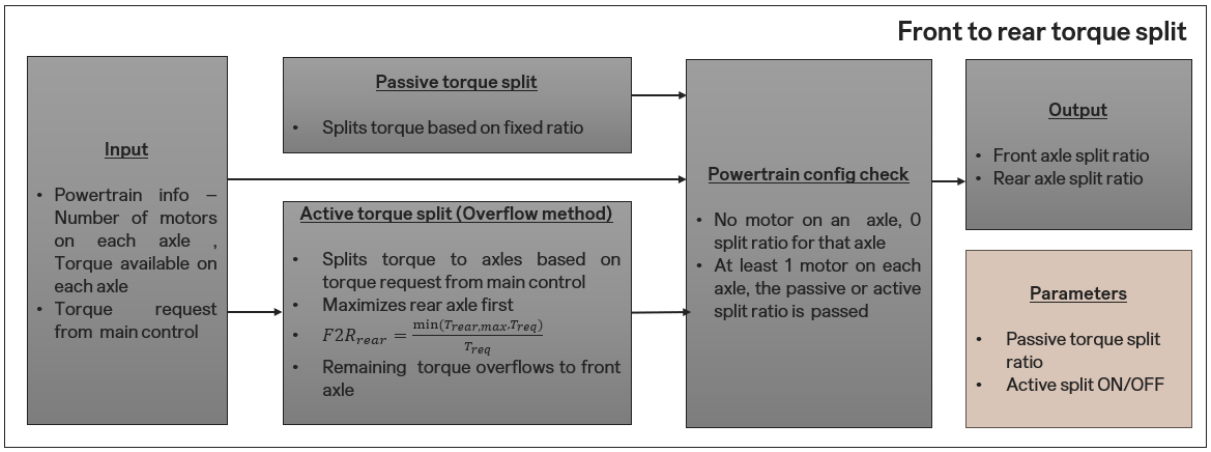



Figure 9: Torque split - block diagram

total torque demand at the wheels as input. In contrast, active torque split dynamically allocates torque using the overflow method, maximizing the force ratio at the rear axle (as defined by the formula below) before distributing the remaining torque to the front axle.

$$F2R_{rear} = \frac{min(T_{rear,max}, T_{req})}{T_{req}} \tag{1}$$

where $T_{rear,max}$ is the maximum torque available at the rear axle and $T_{req}$ is the driver total torque demand.

The user-defined parameter in this subsystem is passive torque split ratio.

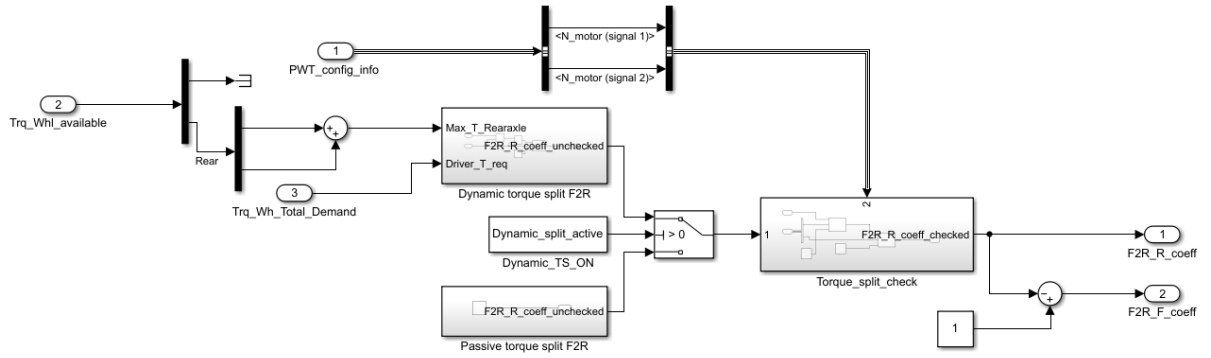



Figure 10: Torque split - Simulink model

The force ratios are further checked in the torque split check block, represented in Figure 11, which ensures that the front to rear split is obtained only if at least one motor is present on each axle.

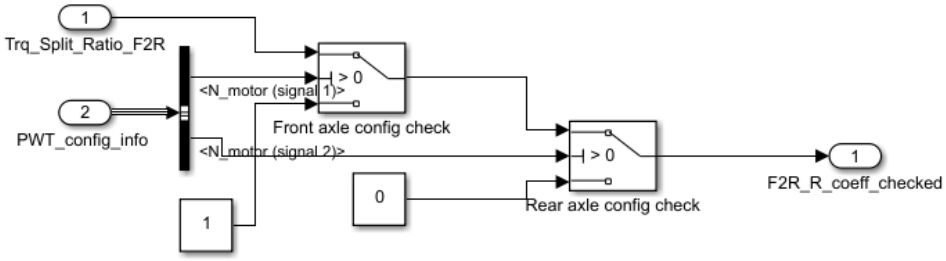



Figure 11: Torque split check - Simulink model

### 3.3.2 Torque vectoring

The purpose of this subsystem is to distribute torque at each of the wheels, for instance during cornering, only with the help of the propulsion system. The output of this subsystem is the left to right force ratios of front and rear axles.

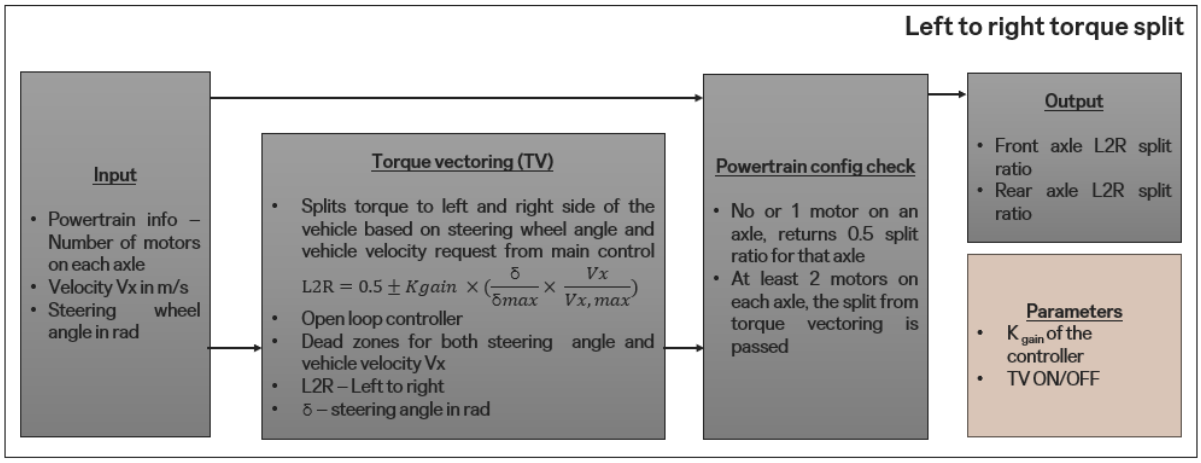



Figure 12: Torque vectoring - block diagram

The torque vectoring is based on the steering input and vehicle speeds. It is implemented using an open-loop controller which is linearly scaled with respect to both steering angle and vehicle speed.

$$L2R = 0.5 \pm K_{gain} * (\frac{\delta}{\delta_{max}} * \frac{Vx}{Vx_{max}}) \tag{2}$$

where L2R is the left-to-right ratio, K is the controller gain, $\delta$ is the steering angle, $\delta_{max}$ is the maximum steering angle, $Vx$ is the longitudinal vehicle speed, $Vx,max$ is the maximum vehicle speed.

The dead band region for steering wheel angle is between $-11deg$ to $+11deg$, and the saturation limits are $-70deg$ to $+70deg$.

The dead band region for vehicle speed is between 0 and $16km/h$.

The user-defined parameter in this subsystem is controller gain K.

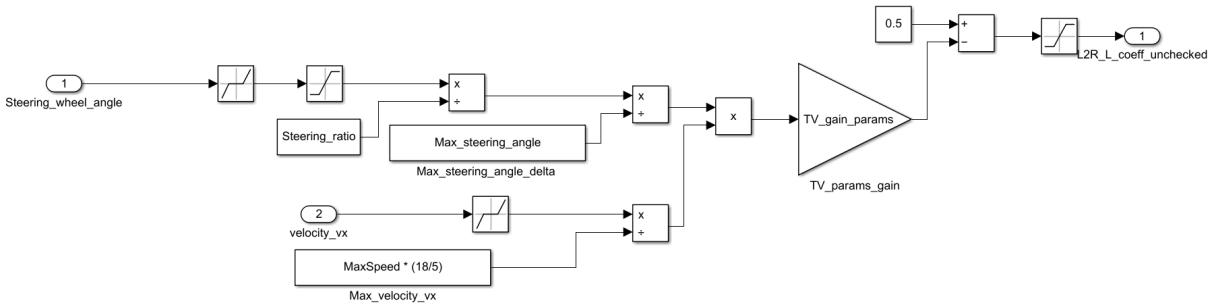



Figure 13: Torque vectoring - Simulink model

The left-to-right ratios are further checked in the torque vectoring check block, which ensures that these are obtained only when two motors on each axle are present.

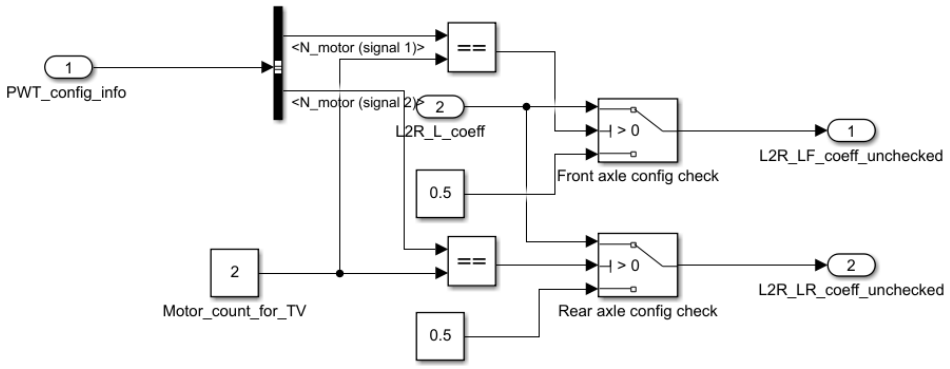



Figure 14: Torque vectoring check - Simulink model

#### 3.3.2.1 Torque distribution

Torque distribution block computes the torque required at each wheel as a product of front-to-rear force ratio, left-to-right force ratio, and driver torque demand.

### 3.3.3 Slip control

The main function of this subsystem is to regulate wheel slip in response to driver input, generating torque requests for each wheel to minimize slip.

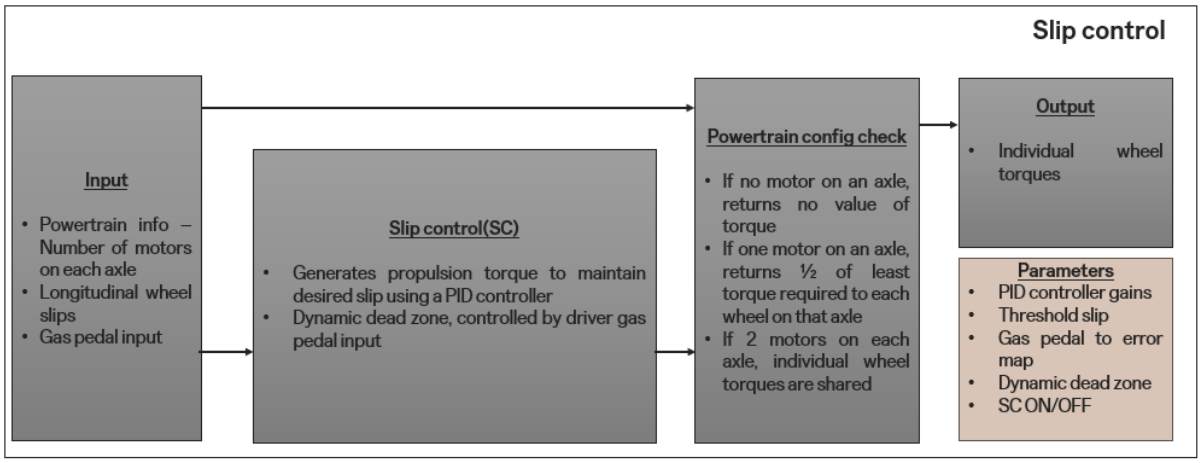



Figure 15: Slip control - block diagram

A user-defined slip threshold serves as the baseline for slip control, while the allowable slip error is dynamically determined based on the accelerator pedal input. In this case, a linear relationship between accelerator pedal input and slip error is employed, with the error limited at a maximum value of 0.06. A PID controller uses the slip error to calculate the wheel torques requested.

When wheel slip is detected, the slip controller provides the propulsion torque for the wheels; otherwise, it outputs a value of $-1$.

The user-defined parameters in this subsystem are PID controller gains, threshold slip, accelerator pedal to error map.

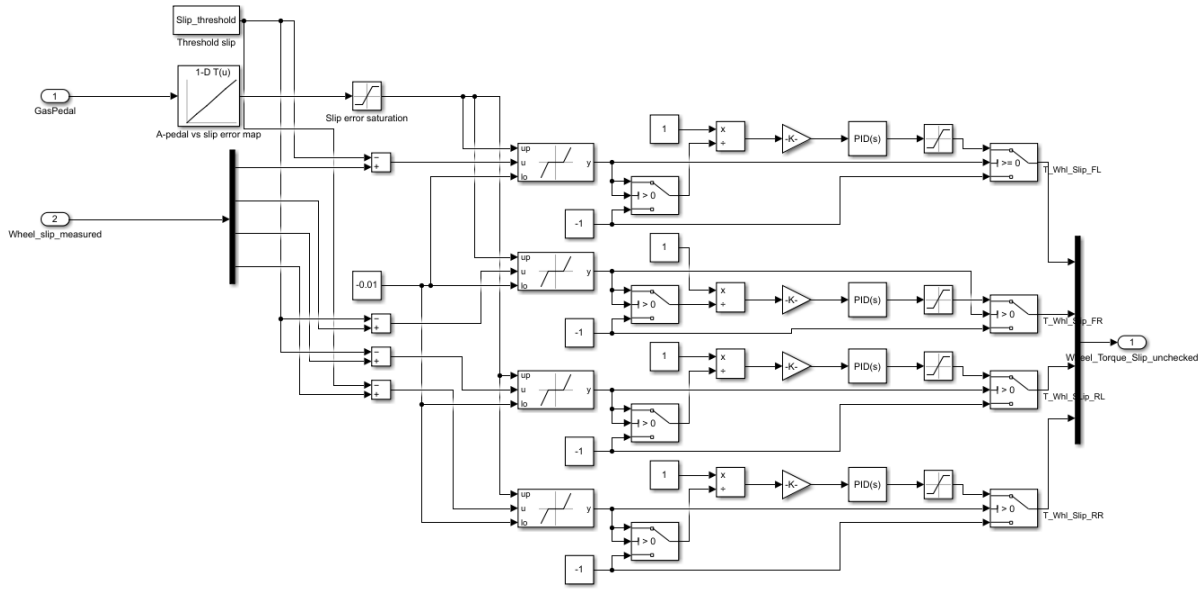



Figure 16: Slip control - Simulink model

The torque values are validated in the slip control check block. This block ensures that torque values are generated only when there is at least one active motor on the axle.

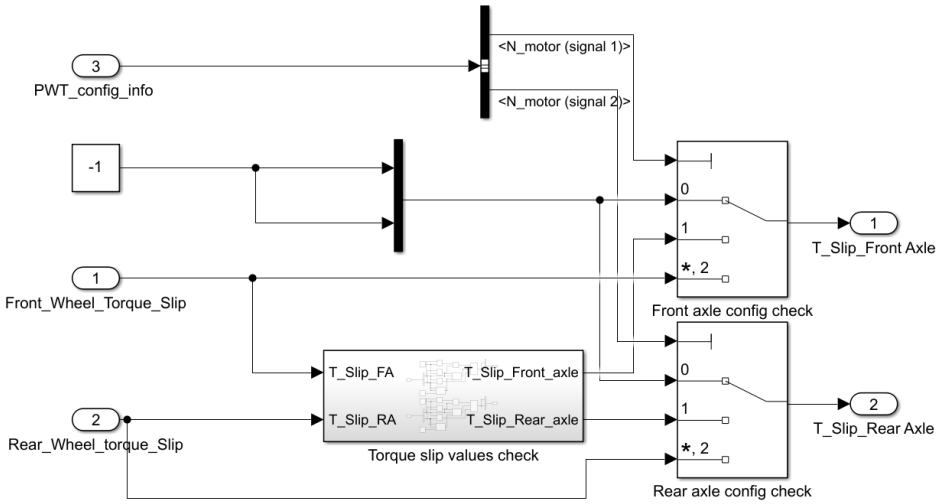



Figure 17: Slip control check - Simulink model

### 3.3.4 Arbitration

The arbitration block is a decision-making block. It outputs the torque values from the torque distribution block if there is no output from the slip controller; else it outputs the torque values from the torque distribution block.

A smoothing function prevents high-frequency output from the arbitration block.

$$\text{Smoothing function} = \frac{1}{0.1s + 1} \tag{3}$$

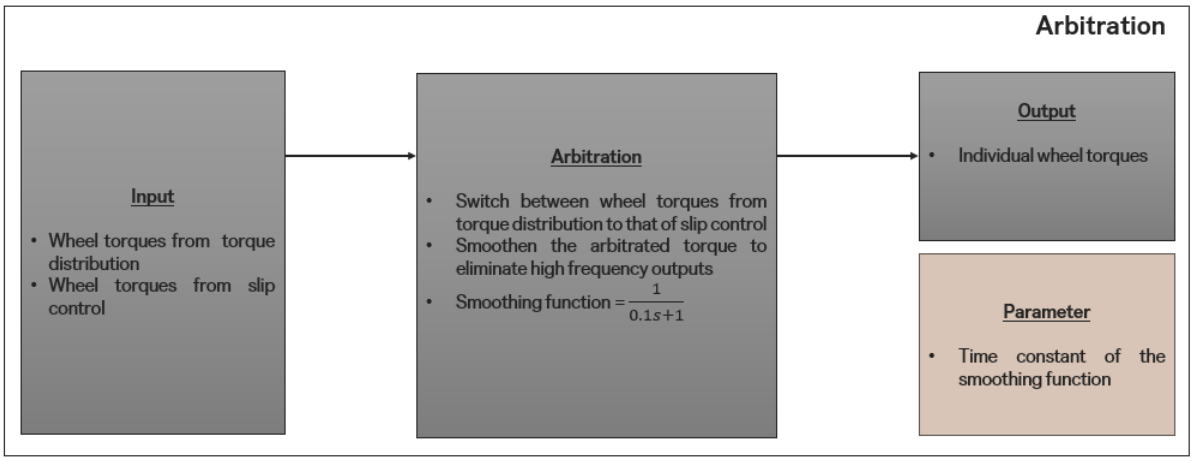



Figure 18: Arbitration block diagram

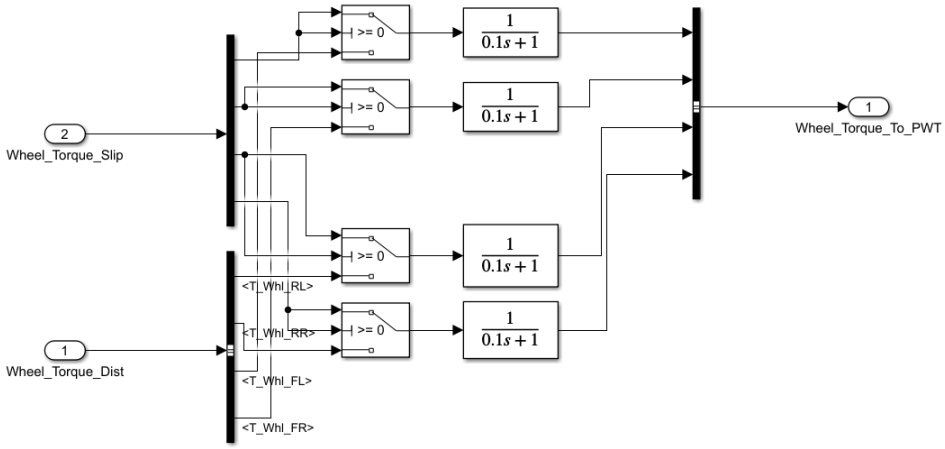



Figure 19: Arbitration block - Simulink model

The user-defined parameter in this subsystem is time constant of the smoothing function.

### 3.4 Powertrain

The Powertrain subsystem plays the role of modeling the mechanical components of the propulsion system to simulate the dynamic response of the powertrain to a given torque request. These components include all elements between the control signals and the torque applied to the wheels, essentially the motors and the driveline.

Moreover, the project focuses on simulating various powertrain architectures, requiring the model to be configurable for different numbers of motors, drivelines, and layouts. To address this, the model was divided axle-wise, as there is no mechanical connection between the axles in any conventional electric powertrain. As represented in Figure 20, each axle can be configured to have a single motor, two independent motors, or be non-powered, thereby supporting all the combinations considered.

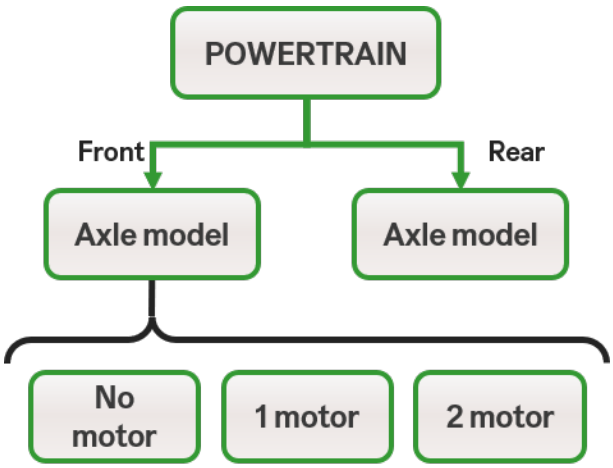



Figure 20: Powertrain Subsystem overview

It is important to note that a full vehicle model is also required for simulations (CarMaker in our case). The vehicle model manages the wheel dynamics by calculating the rotational acceleration

and velocity of the wheels, and takes as an input the torque applied by the powertrain, which is computed in our model.

The implementation of the model was carried out by creating small sub-models in Simulink subsystems, which are referenced in the Axle Model subsystem, which is also referenced twice in the Powertrain Model. The Axle model is a Variant-type subsystem, enabling the automation of the model configuration based on the powertrain architecture selected by the user in the parameters file. This results in a highly modular system, where each individual sub-model (such as the Motor model) is fully parameterized with the characteristics defined as a data structure in the parameters file. These sub-models can be instantiated in several larger models and easily updated or replaced with a more advanced version.

The simplified diagram in Figure 21 illustrates the process of converting the torque request signal into the torque output at the wheel, which includes the driveline model to compute the dynamic response of the applied torque.

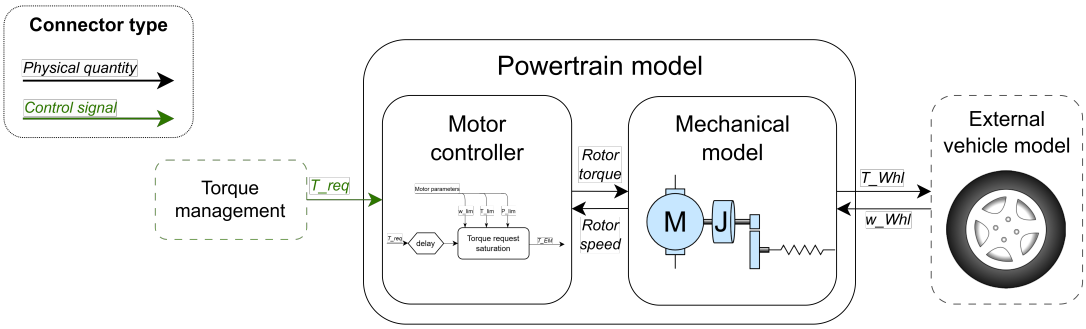



Figure 21: Powertrain model functionality, showing only one side of a 2-motor axle.

### 3.4.1 Axle model

The Axle model consists of a configurable system that can be set up with a single motor, two independent motors, or as non-powered. The model includes the electric motors and the driveline, which is simplified to a transmission and a driveshaft. Inertia and stiffness are modeled to compute the dynamic response of the torque applied by the motor through the driveline.

The two-motor setup features independent motors and drivelines for each wheel, as shown in Figure 22, allowing different torques to be applied to each wheel. In the single-motor configuration, a differential splits the torque equally between both wheels, as illustrated in Figure **??**. No dynamics are modeled in the differential, allowing both driveshafts to be represented as a single shaft with twice the stiffness.

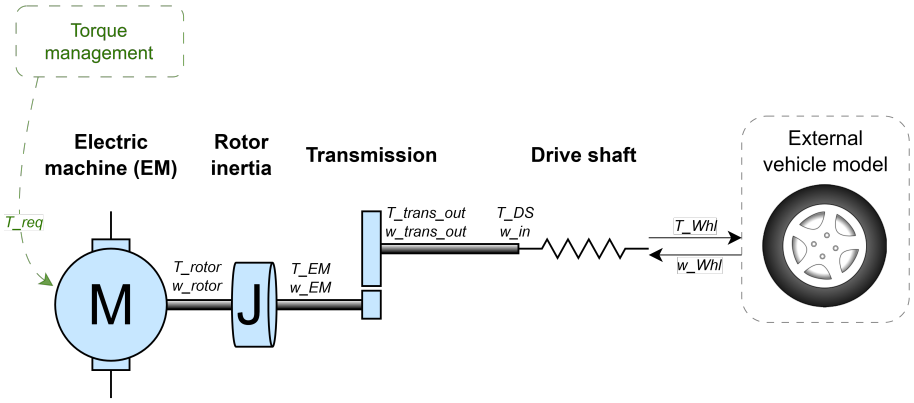



Figure 22: Axle model with two motor (one per wheel)

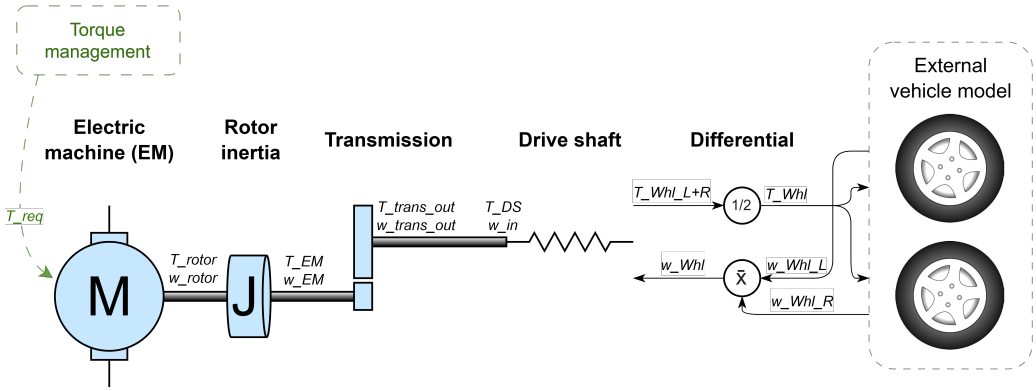



Figure 23: Axle with one motor and differential

The axle model is implemented in a Simulink Variant subsystem, which can automatically switch between configurations based on the architecture selected in the parameters file. The signals are uniform across the different variants, enhancing modularity and interchangeability.

Each variant corresponds to one of the configurations of the Axle model, which are implemented by connecting subsystem references of the sub-models (Motor, Transmission and Drive shaft). The Figure 24 shows the implementation of the two motor configuration, that consists of two copies of the motor and driveline system.

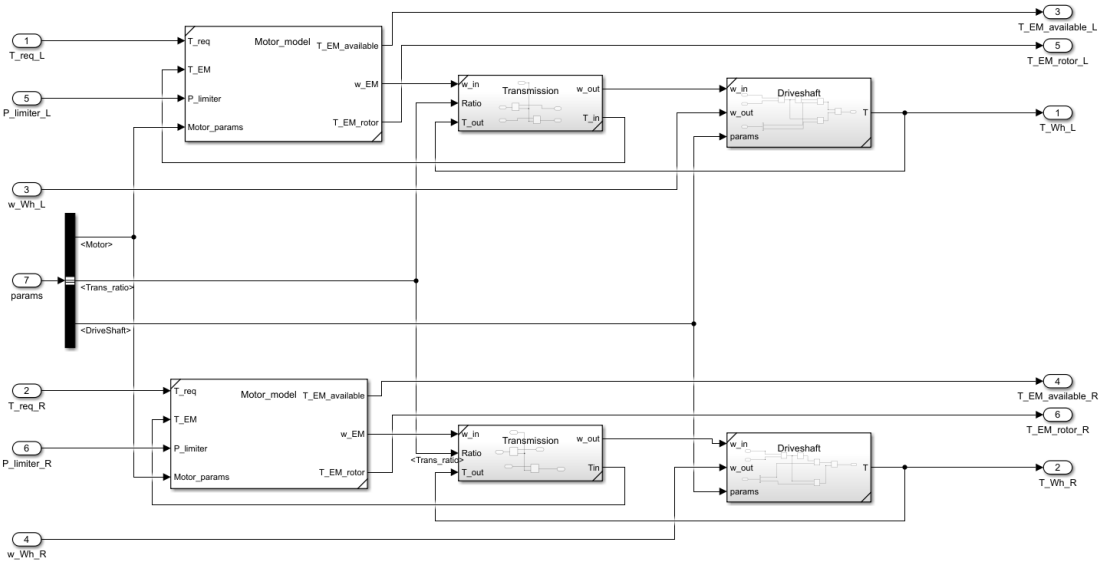



Figure 24: Two-motor variant of the Axle model, implemented in a Simulink Variant subsystem

### 3.4.2 Modular sub-models

#### 3.4.2.1 Motor

The objective of the Motor model is to simulate the torque applied to the electric motor to the driveline, which ultimately affects the wheel and tire forces. Therefore, the model emphasizes the mechanical response while simplifying other phenomena, such as power losses, which are relatively low in electric motors. It also assumes that the motor's capabilities are symmetric for both motor and generator operation.

The model consist of a rotor characterized by the moment of inertia, modeled as a flywheel, and a controller that applies torque to that rotor based on the torque request input. The controller limits the applied torque according to the motor capabilities specified in the parameters file. As shown in Figure 25, electrical phenomena is not directly modeled; however, a first-order delay defined by a time constant is included. Additionally, an external power limit signal is incorporated, which can serve as a battery power limit signal.

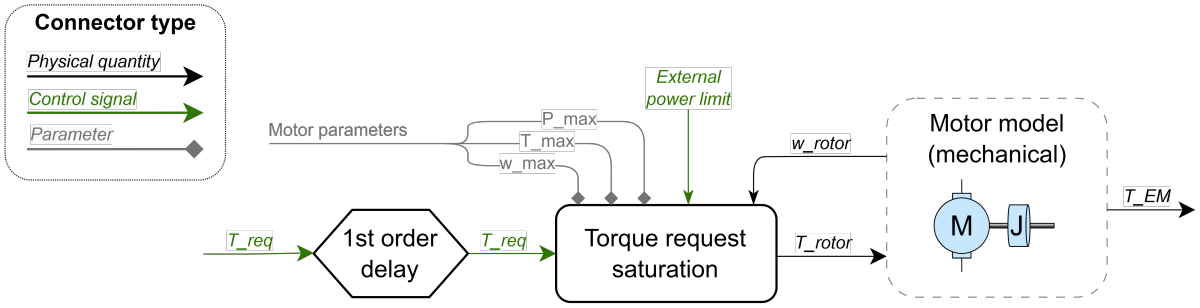



Figure 25: Motor controller diagram

The motor controller subsystem is presented in Figure 26. All the parameters are defined externally through input signals, enabling the block to be referenced across multiple models.

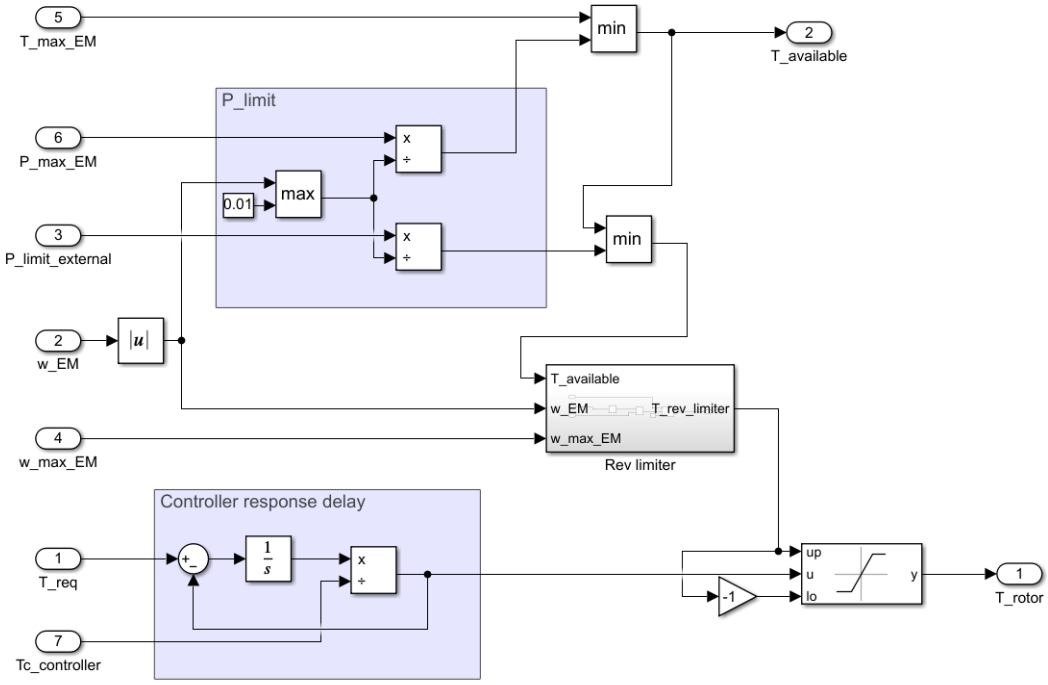



Figure 26: Motor controller subsystem in Simulink

The rotational inertia is modeled mathematically according to the Equation 4, which is implemented with Simulink standard blocks (Figure 27).

$$\omega = \int \frac{T_{total}}{J} dt = \int \frac{T_{EM} - T_{react}}{J} dt \tag{4}$$

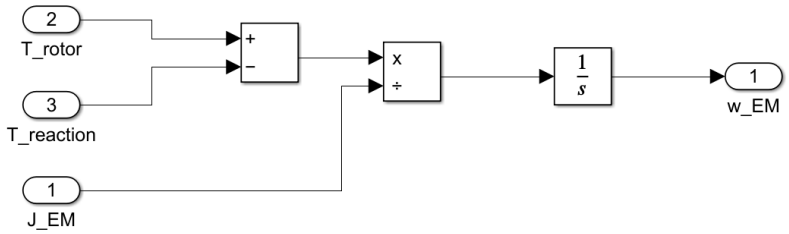



Figure 27: Rotor model in Simulink

#### 3.4.2.2 Transmission

The transmission subsystem applies the corresponding transmission ratio to both torque and angular velocity, following the relations in the Equation 5. Friction or any losses are not modeled.

$$\begin{aligned} \omega_{out} &= \frac{\omega_{in}}{Ratio} \\ T_{in} &= \frac{T_{out}}{Ratio} \end{aligned} \tag{5}$$

The implementation in Simulink is presented in Figure 28. It is worth noting that the torque from the output shaft is an input signal to the subsystem, since the torque value is derived from the drive shaft spring-damper system.

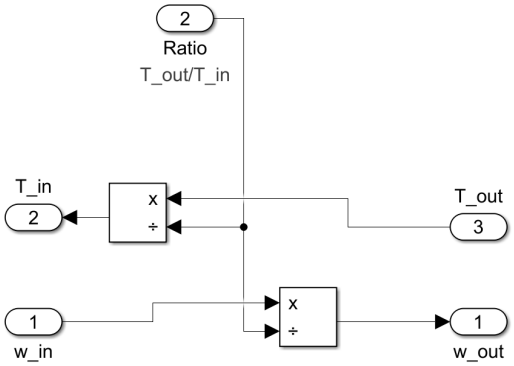



Figure 28: Transmission subsystem implemented in Simulink

#### 3.4.2.3 Drive shaft

The drive shaft connects the transmission, which is directly connected to the motor rotor, to the wheel. Essentially it couples two "flywheels" —each subjected to external torques— with a specified angular stiffness and damping. The mechanical response of this torsional spring-damper system is described by the Equation 6.

$$\begin{aligned} \dot{\phi} &= \omega_{in} - \omega_{out} \\ T &= k * \int \dot{\phi} dt + \dot{\phi} * b \end{aligned} \tag{6}$$

where

$\omega$ : Angular velocity (rad/s)

$T$ : Torque (Nm)

$\phi$ : Relative angle between drive shaft ends (rad)

$k$ : Stiffness (Nm/rad)

$b$ : Damping coefficient (Nm*s/rad)

The subsystem is implemented in Simulink and parametrized to take the stiffness and damping values as inputs, as shown in Figure 29. This design ensures modularity, allowing the block to be directly referenced across several models.

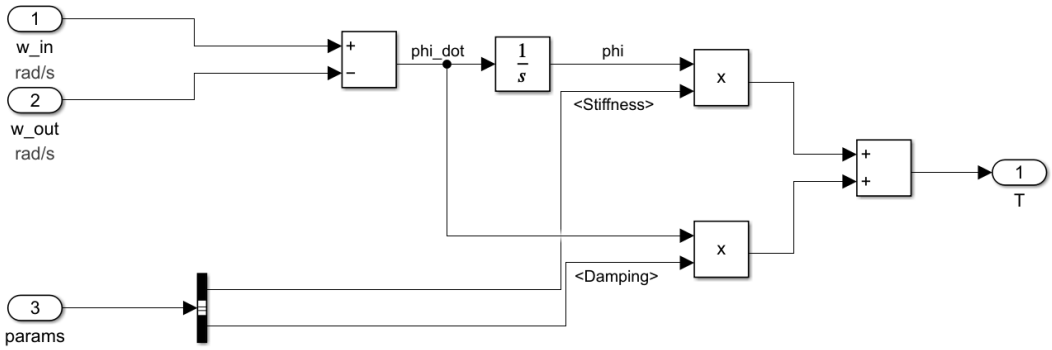



Figure 29: Drive shaft subsystem implemented in Simulink

### 3.4.3 Power limit arbitration

The purpose of this feature is to limit the total battery power available.

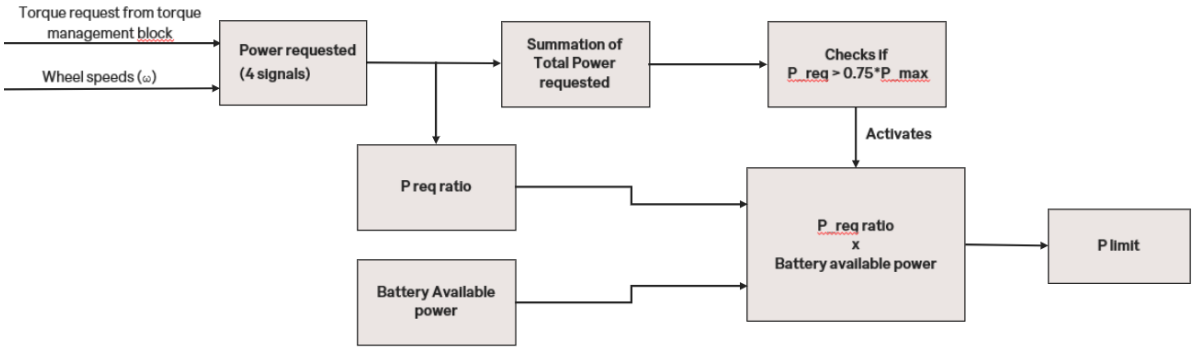



Figure 30: Power limit block

The block diagram above shows the power limiting case for positive power. It behaves the same for regeneration power, beside the fact that the regeneration power has negative gain.

The requested power is derived from the torque request, which comes from the Torque Management block, and the angular wheel speeds for all the wheels. For each individual power request, we calculate the power request ratios and then sum them to determine the total requested power. If the total requested power exceeds 75% of the maximum available battery power, the power is limited by the product of the power request ratio and the available battery power.

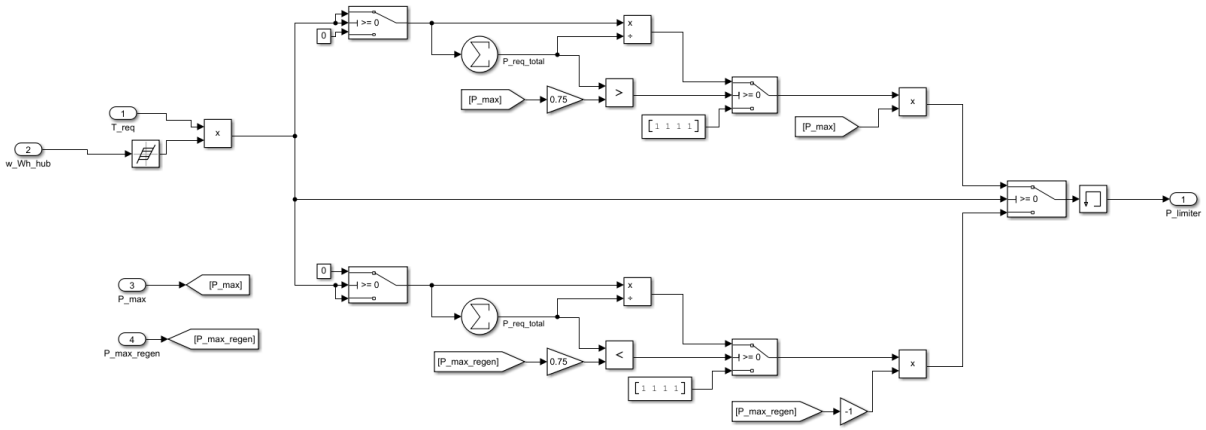
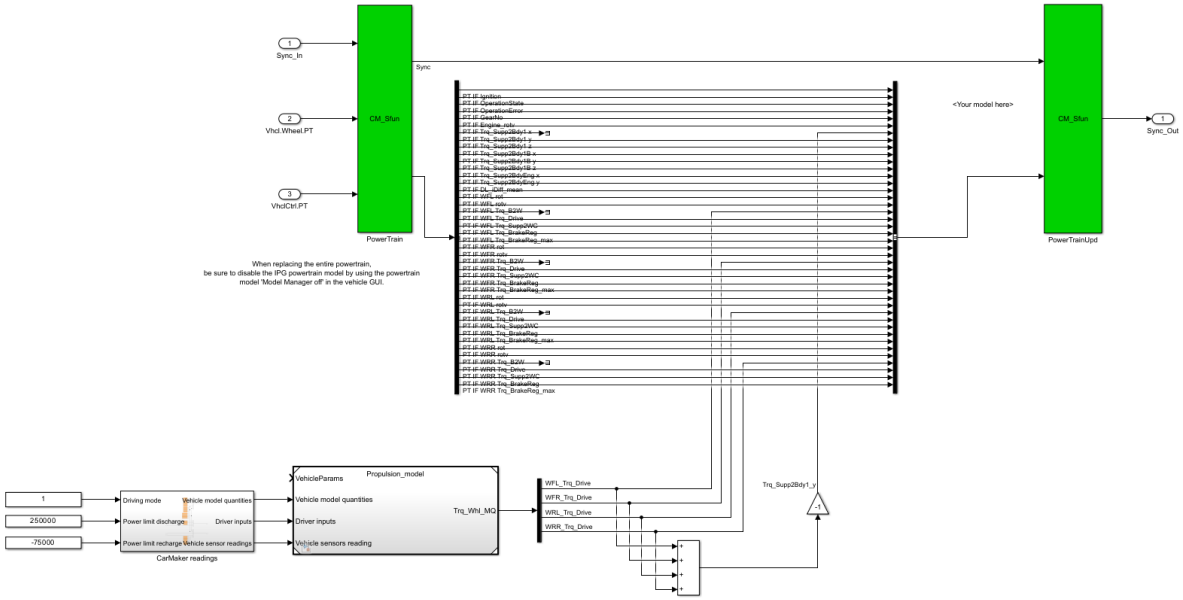


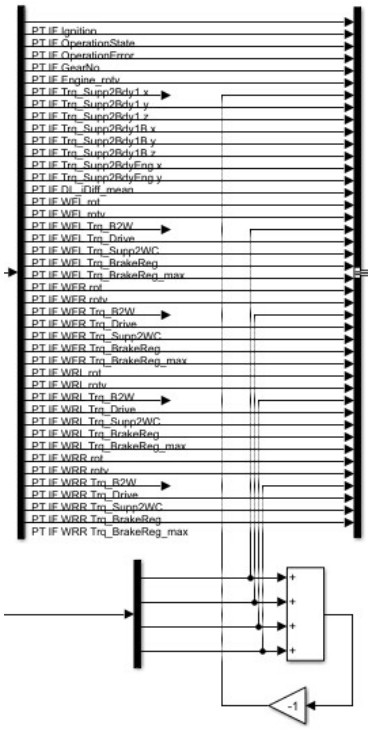

Figure 31: Power limit arbitration- Simulink block

# 4 CarMaker integration

Once the model has been built, a proper integration with a simulation software has to be done in order to allow for virtual simulations. CarMaker for Simulink (CM4SL) is a tool that tests realistic models of the vehicle, many other features and other environmental conditions. We integrate our custom model with CarMaker to test its performance in a realistic vehicle and road conditions - save time and resources by reducing physical testing. The torque coming out of the CarMaker's powertrain are replaced with torques coming out of our model and also the torque at the chassis would be the reaction torque.




(a) CarMaker integration




(b) Overridden signals

Figure 32: CarMaker integration

In the figure it is possible to identify:

- PT.W<pos>.Trq Drive as the drive torque of the wheel;
- PT.Trq Supp2Bdy1.y as the supplied torque to wheel frame, which can also be referred as the reaction torque from the wheels

Once the signals are correctly substituted, the propulsion model has to be considered as OpenXWD inside the simulation environment. OpenXWD is used as it would consider the wheel inertias along with our custom model.


Figure 33: OpenXWD Custom model

# 5 Simulation results and discussion

In our proposed architecture, users are encouraged to perform unit testing as an important step before progressing to a complete vehicle simulation. Unit testing focuses on verifying the functionality of individual components or modules in isolation to ensure their proper operation. This step helps identify and address potential issues at an early stage, reducing the likelihood of errors propagating into the full simulation where they can be more difficult to resolve.

Unit testing is particularly useful for validating each module against expected behaviors and requirements. It ensures that key components, such as algorithms, subsystem models, or control logic, are functioning correctly. Additionally, it provides an opportunity to test how individual components respond to different conditions, including edge cases and rare scenarios, which is valuable for ensuring reliability.

## 5.1 Unit test results

The following sections present the unit testing results for each subsystem. It is important to highlight that proper unit testing of the main control unit is feasible, even when it depends on outputs from the powertrain. However, the presented results are more significant in demonstrating the functionality of this subsystem, highlighting the influences of the inputs.

### 5.1.1 Main control unit

The primary objective of these simulations is to show the effects of the throttle map, while maintaining a constant driving mode, on the final output of the main control unit. The same maneuver is performed under two different set of battery power limits, illustrating how this external parameter influences the total torque request at the wheel.

The maneuver is created using user-defined gas pedal input in the CarMaker GUI. It is a sequence of:

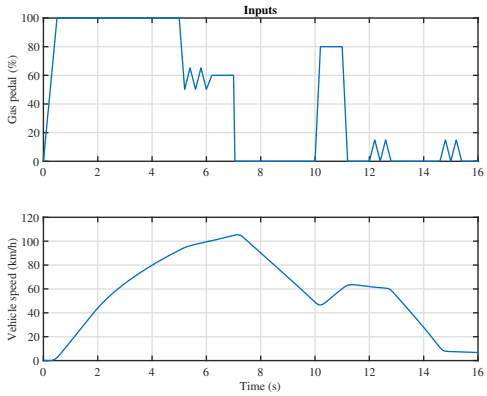
- Full throttle up to $5s$;
- Rapid drop to 50% with small oscillations (15%);
- Rapid drop to 0%, braking to reduce speed;
- Sudden press to 80% to increase speed;
- Rapid release to 0%;
- Small, brief presses (15%).

Both simulations are conducted using one motor per axle (AWD), where each motor has a power of $100kW$. The simulations were performed on a flat straight road.

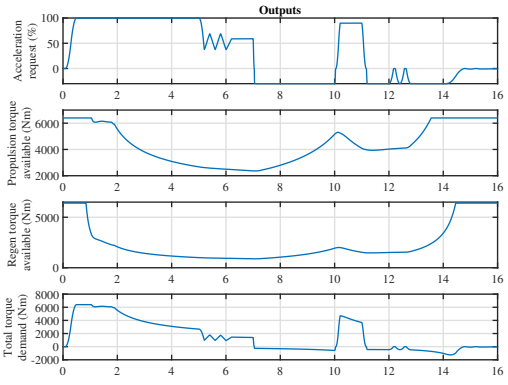
The battery power limits for the simulation are set to:

- $250kW$ (discharging);
- $-75kW$ (charging);

The results of the above maneuver are presented in Figure 34a and Figure 34b.




(a) Inputs (gas pedal & vehicle speed) vs time




(b) Outputs (acceleration request, propulsion & regen. Torque, total torque demand) vs time

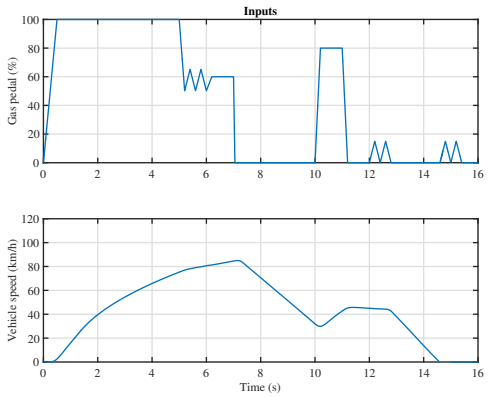
Figure 34: Main control unit test results with [250 kW; -75 kW] battery power limits

From the first full-throttle acceleration, the rapid increase in the acceleration request from 0 to 100% can be observed, closely following the s-curve described in Sec. 3.2. As the speed increases, the total torque demand decreases due to motor torque availability limits, with some oscillations caused by wheel slipping. Oscillations in the gas pedal position outside the dead zone result in smoothed oscillations in the final demand. A rapid release of the accelerator pedal causes the map to enter the regeneration region, as the speed exceeds the ramp's upper limit (here $25km/h$). At this stage, the percentage deceleration requested is multiplied by the available regeneration torque, which increases as the speed decreases. Oscillations within the

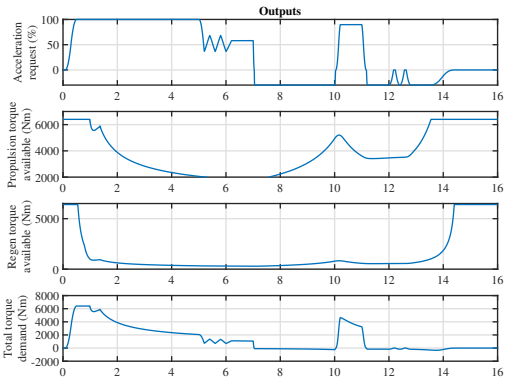
regeneration region demonstrate the effective operation of the OPD, causing the final demanded torque to alternate between full regeneration and null values. As the speed drops below the ramp's lower limit, the dead zone is reached, and small oscillations in the gas pedal position within its boundaries correspond to a null final demand.
The same maneuver is now simulated with lower battery power limits:

- $125kW$ (discharging);
- $-20kW$ (charging);

The results of the maneuver with lower battery limits are presented in Figure 35a and Figure 35b.




(a) Inputs (gas pedal & vehicle speed) vs time




(b) Outputs (acceleration request, propulsion & regen. torque, total torque demand) vs time

Figure 35: Main control unit test results with [125 kW; -20 kW] battery power limits

Comparing Figure 34 and Figure 35 we see that the vehicle accelerates slower due to the lesser propulsion torque available, and it is also evident in vehicle speed plot that the vehicle goes slower with lower battery power limits.

The impact of the gas pedal input on the total torque demand at the wheel, as output by the main control unit, is shown in Figure 36. This plot depicts the traction diagram obtained under identical road and vehicle conditions, with a constant gas pedal input throughout the simulation. As the gas pedal input increases, the total torque demand rises accordingly, up to the point where wheel slip occurs, resulting in oscillations. The torque demand aligns with motor and battery availability across different speeds.

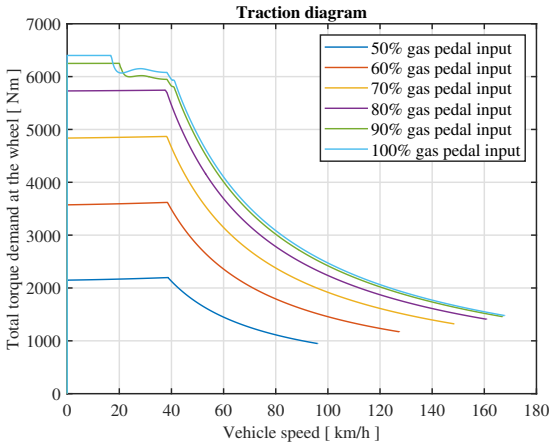



Figure 36: Traction diagram for increasing gas pedal input

### 5.1.2 Torque management

Unit test of torque management block is performed to verify each sub-feature, i.e. torque split, torque vectoring, and slip control. Here, the inputs and relevant user-defined parameters are modified to only observe if the outputs are as expected. It is worth noticing that unit testing does not imply any controller performance evaluation.

**Torque split**

To unit test the passive split feature, user-defined constant ratio of front to rear axle split is set as 40 : 60 and 'Active split ON/OFF' is set to OFF. It can be noted in Figure 37 that the output is constant with time.

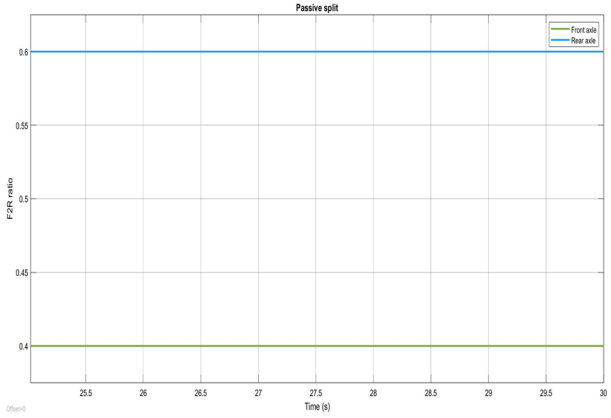



Figure 37: Passive split

To unit test active split feature, user-defined 'Active split ON/OFF' is set to ON. The output is now varying with time, as it can be observed in 38

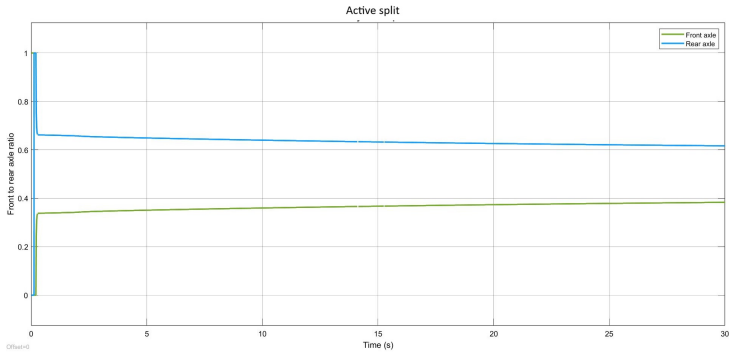



Figure 38: Active split

### Torque vectoring

To unit test the torque vectoring feature, two cases are considered:

- Constant longitudinal vehicle speed with varying steering wheel angle;
- Constant steering wheel angle with varying vehicle speed.

The outputs from these simulations help visualize the dead bands defined for both steering wheel angle and longitudinal vehicle speed. Dead bands are characterized by zero torque vectoring.

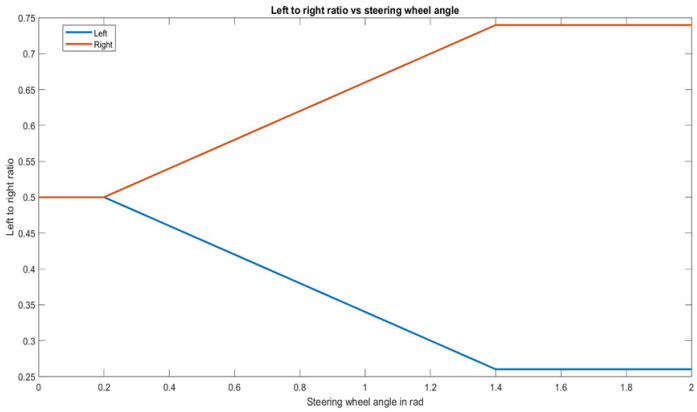



Figure 39: L2R ratio vs steering wheel angle, constant velocity – 30kmph, left turn

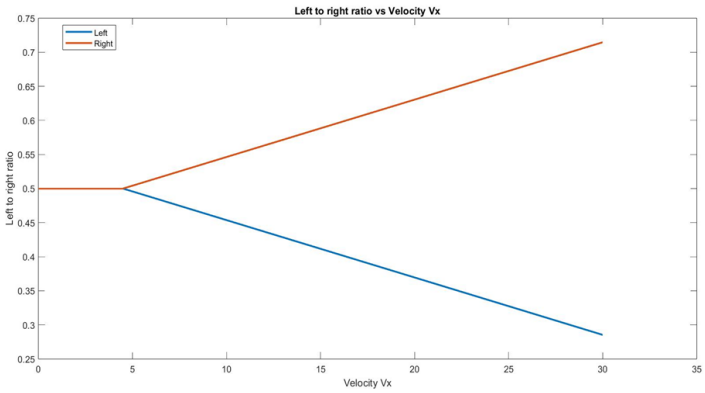



Figure 40: L2R ratio vs vehicle speed Vx, constant steering wheel angle – 30deg, left turn

### Slip control

To unit test the slip control feature, a step input of wheel slip is provided. The slip controller torque output is a step output, as expected. However, when there is no wheel slip, the slip controller gives '−1' as the output value.

The PID values of the controller are tuned after integrating with the vehicle model.

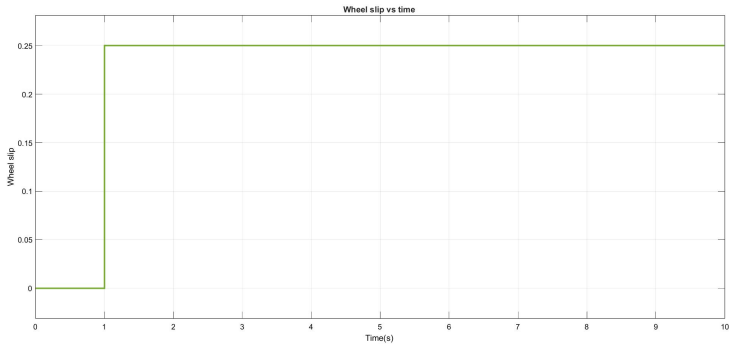



Figure 41: Wheel slip - step input

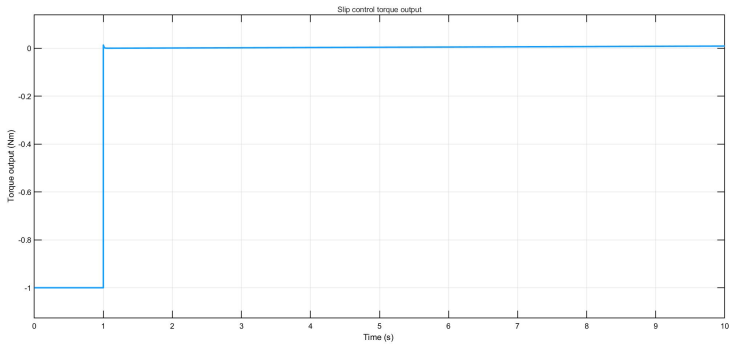



Figure 42: Slip control - torque output

## 5.2 Complete vehicle test results

Testing of the complete vehicle was carried out to test torque split, torque vectoring, and slip control with these cases:

- Torque split:
  - Straight line acceleration with passive / active torque split for AWD (1 motor each axle);
  - Straight line acceleration with passive / active torque split for AWD (2 motor each axle) & (Tri motor);
- Torque vectoring:
  - Cornering at low speed with torque vectoring on/off;
  - Cornering at high speed with torque vectoring on/off;
- Slip control:
  - Straight split mu (=0.1) road for an AWD configuration with slip control on/off;
  - Straight split mu road for 1 motor on each axle configuration with slip control on/off;
  - 15% Gradient split $\mu$ (=0.1) road for an AWD configuration with slip control on/off.

### 5.2.1 Torque split results

The objective of this simulation is to evaluate the effects of different torque split configurations (passive vs. active). A ramp input on the gas pedal was simulated on a straight, flat road. The maneuver was tested under two scenarios: a passive split with a $40:60$ front-to-rear bias and an active split configuration.

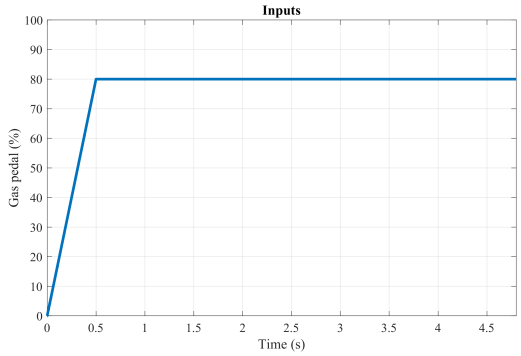
Three all-wheel-drive (AWD) powertrain configurations were tested:

1. AWD with one motor on each axle;
2. AWD with two motors on each axle;
3. AWD trimotor with two motors on the rear axle and one on the front axle.

In all configurations, the motors were rated at $100kW$ each.

The results of the simulation are presented in Sections 5.2.1.2 and 5.2.1.2.

#### 5.2.1.1 Straight line acceleration with passive / active torque split for AWD (1 motor each axle)

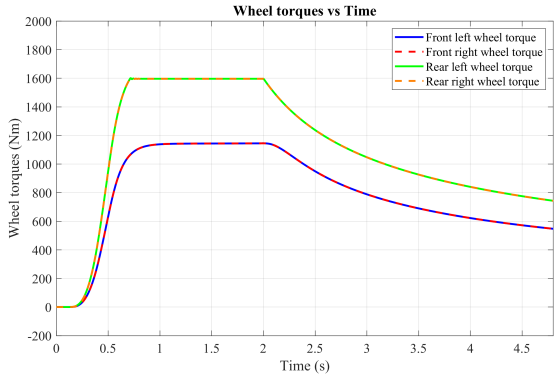



(a) Gas pedal input

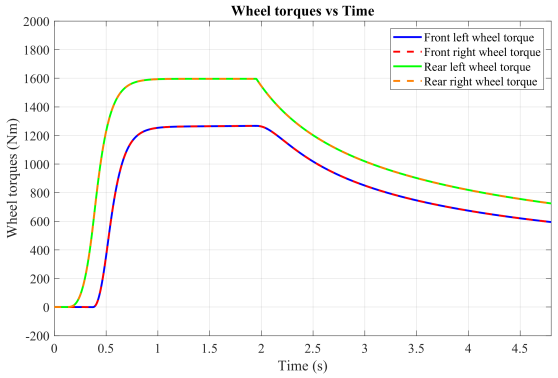
(b) Road

Figure 43: Setup for torque split test

Figure 44a illustrates the operation of the passive split configuration. In this setup, the total torque demand is divided with a fixed bias of 40 : 60, and both torques are supplied simultaneously from the start. In contrast, the active split simulation (Figure 44b) provides a greater torque bias to the rear axle at the start. Once the torque on the rear axle reaches its maximum capacity, the front axle begins to deliver torque. This behavior is depicted in Figure 45a, where, at approximately 0.3s, the total torque demand exceeds the maximum torque available on the rear axle, leading to an increase in the front axle bias (Figure 45b).

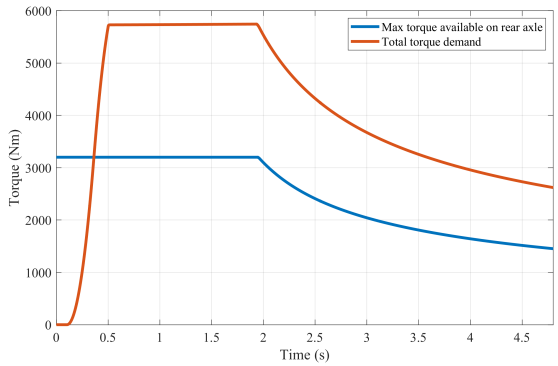



(a) Propulsion torque at the wheels for a passive 40:60 front-to-rear torque split configuration

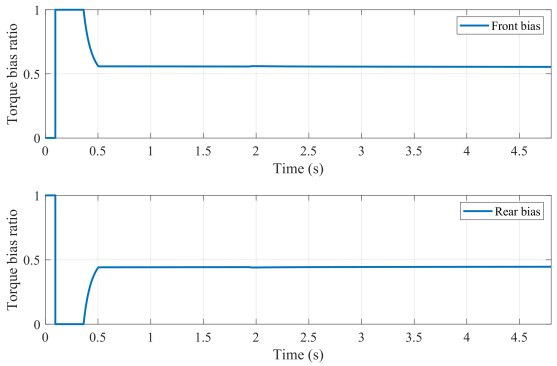



(b) Propulsion torque at the wheels with active torque split configuration

Figure 44: Torque split comparison (passive vs active)




(a) Variation of total torque demand and rear axle torque availability over time

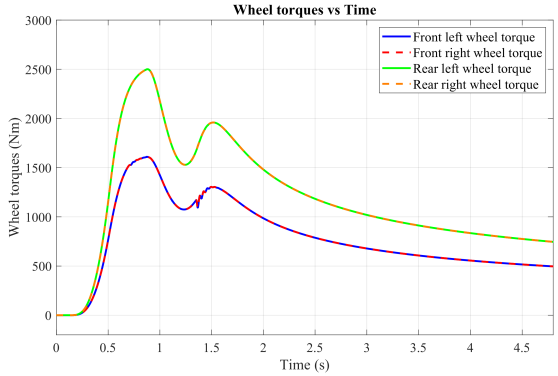



(b) Dynamic variation of the front-to-rear torque split ratio over time

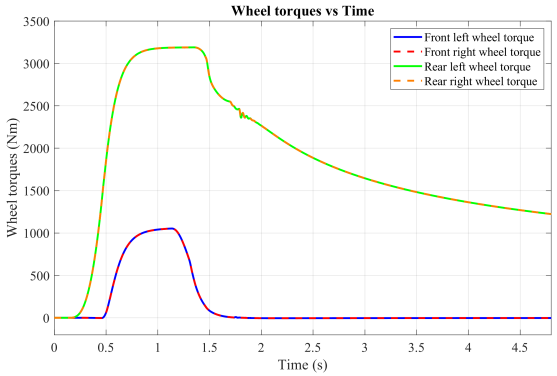
Figure 45: Variation of torque split ratio vs time

#### 5.2.1.2 Straight line acceleration with passive / active torque split for AWD (2 motor each axle) & (tri motor)

The same test was done with a ramp input of gas pedal up to 65% for AWD (2 motors each axle ) presented in Figure 46 & with a ramp input of gas pedal up to 80% for trimotor setup shown in Figure 47. Same behaviour was seen as explained in previous section.

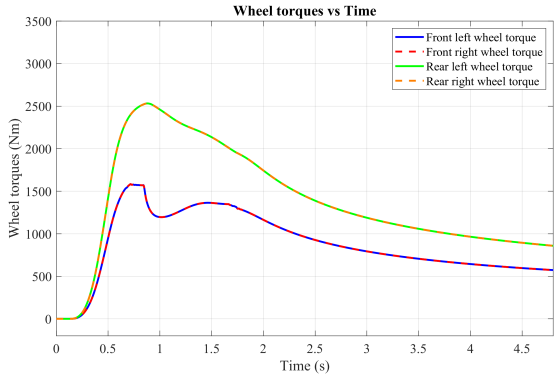



(a) Propulsion torque at the wheels for a passive 40 : 60 front-to-rear torque split configuration

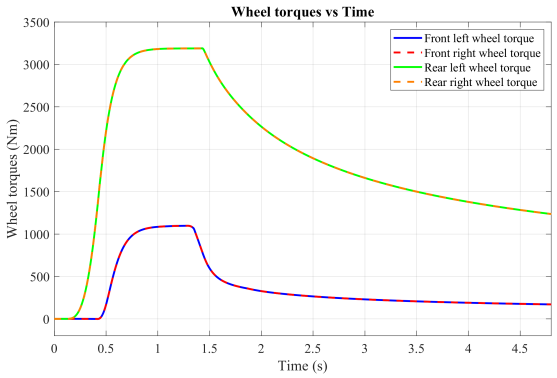



(b) Propulsion torque at the wheels with active torque split configuration

Figure 46: Torque split comparison (passive vs active) for AWD 2 motor each




(a) Propulsion torque at the wheels for a passive 40:60 front-to-rear torque split configuration




(b) Propulsion torque at the wheels with active torque split configuration

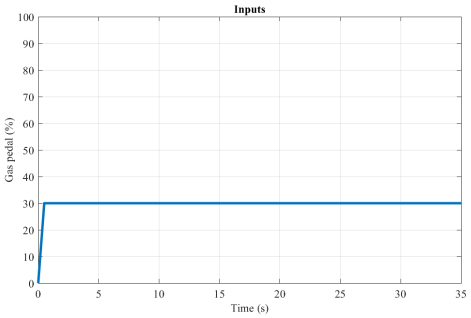
Figure 47: Torque split comparison (passive vs active) for AWD tri-motor each

### 5.2.2 Torque vectoring results

The objective of this simulation is to evaluate the effects of steering wheel angle based torque vectoring. A ramp input on the gas pedal was simulated on a curved, flat road. The powertrain configuration consists of an all-wheel-drive (AWD) system with two motors on each axle, each rated at $100kW$. The maneuver was tested under two scenarios: with & without torque vectoring. This test was done at low speeds & high speeds. The results of the simulation at slow speeds are presented in Figures 49 - 51.

#### 5.2.2.1 Cornering at slow speed

The maneuver involves a constant gas pedal input of 30%, achieved within $0.2s$. This input is applied to accelerate the vehicle to a target speed of $60km/h$, which is considered within the low-speed range. The simulation setup is illustrated in Figure 48.

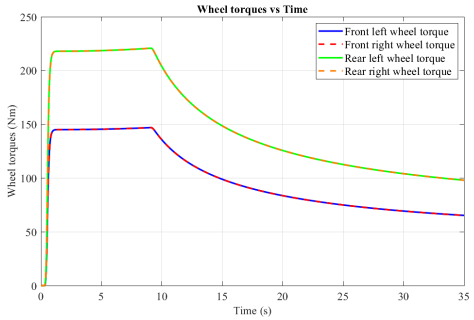



(a) Gas pedal input

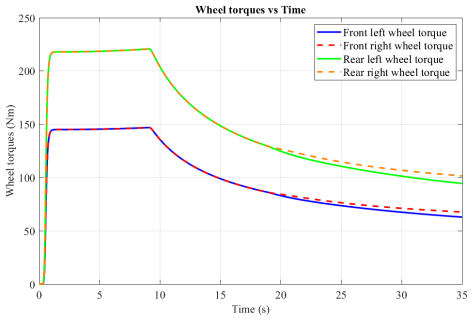
(b) Road

Figure 48: Setup for torque vectoring test

The torque vectoring action, shown in Figure 49b, involves an active distribution of torque between the wheels on the same axle. The controller activates only when the steering wheel angle exceeds the predefined deadband (Figure 50a). Beyond this threshold, the left-to-right torque bias decreases progressively as the steering wheel angle increases over time.

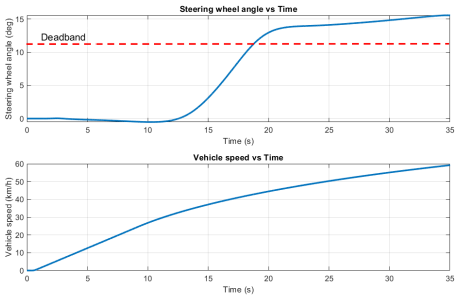



(a) Wheel torques over time without torque vectoring

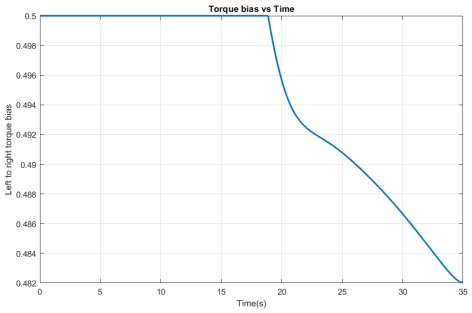



(b) Wheel torques over time with torque vectoring

Figure 49: Torque vectoring effect on wheel torques over time



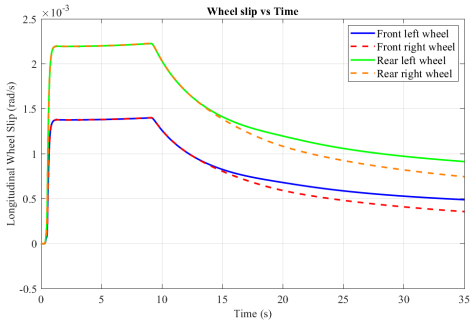

(a) Steering wheel angle & vehicle speed over time



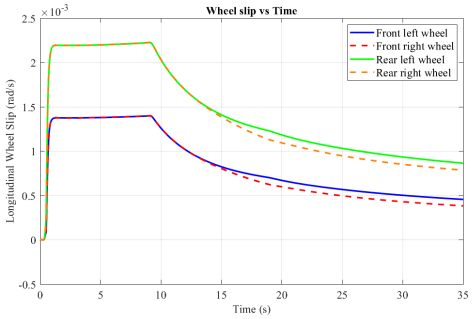

(b) Left to right torque bias over time

Figure 50: Vehicle states' effect on the left to right torque bias

In Figure 51 it is possible to observe a reduction in the wheel slip over time. This side effect is achieved thanks to the torque reduction at the wheel.




(a) Wheel slip over time without torque vectoring

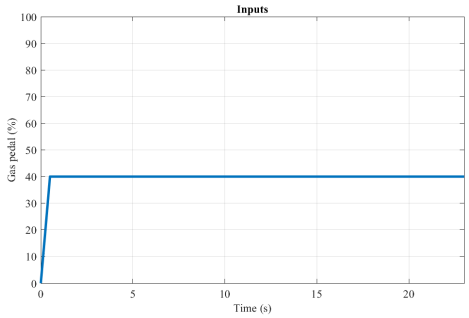



(b) Wheel slip over time with torque vectoring

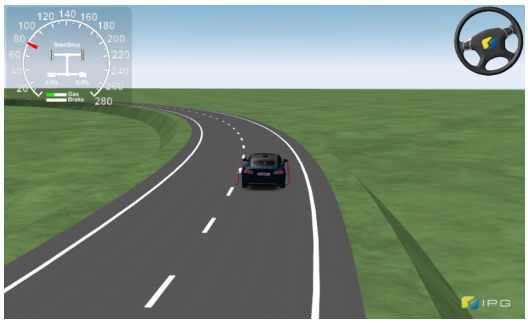
Figure 51: Torque vectoring effect on wheel slip over time

#### 5.2.2.2 Cornering at high speed

This second simulation is conducted under the same road and vehicle conditions, but with the gas pedal input increased to 40%. This variation enables testing the controller at higher speeds, simulating a more extensive utilization of the system.

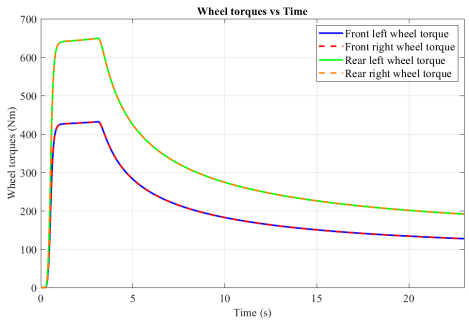



(a) Gas pedal input

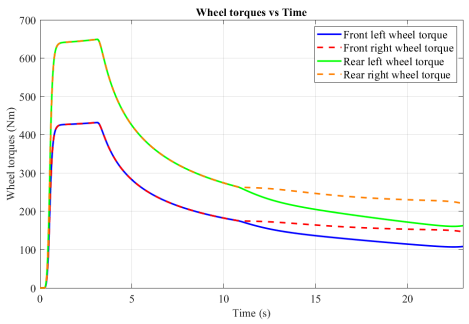


(b) Road

Figure 52: Setup for high speed torque vectoring test

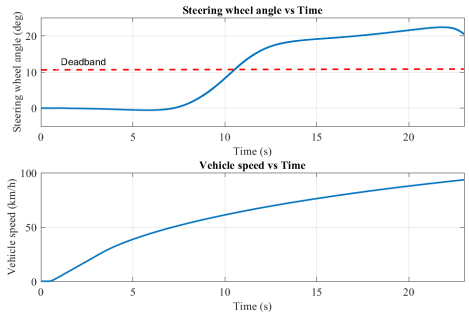



(a) Wheel torques over time without torque vectoring

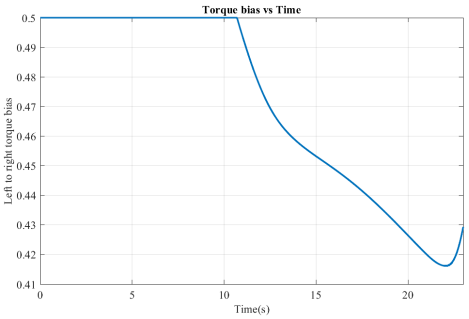



(b) Wheel torques over time with torque vectoring

Figure 53: Torque vectoring effect on wheel torques over time



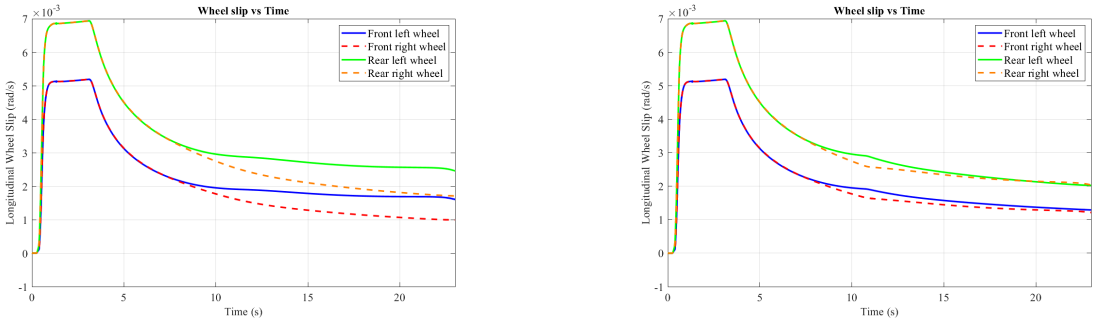

(a) Steering wheel angle & vehicle speed over time




(b) Left to right torque bias over time

Figure 54: Vehicle states' effect on the left to right torque bias

Also the wheel slips reduction is increased with respect to the previous case.



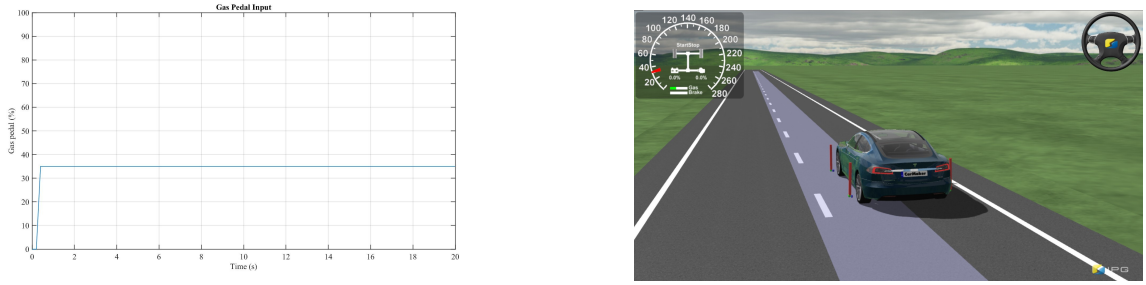

(a) Wheel slip over time without torque vectoring (b) Wheel slip over time with torque vectoring

Figure 55: Torque vectoring effect on wheel slip over time at high speed

### 5.2.3 Slip control results

#### 5.2.3.1 Straight road with split $\mu$ surface with AWD configuration

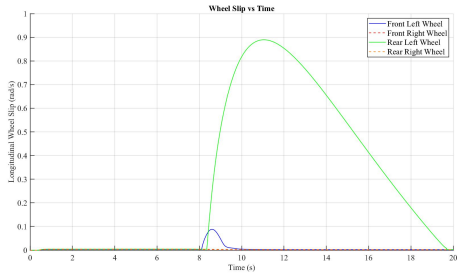
The maneuver was tested with an All Wheel Drive (AWD) configuration or two motors at each axle on a straight flat road with split $\mu$ surface having a coefficient of friction as 0.1. It involves a constant gas pedal input of 35%, achieved in $0.2s$ with $60 : 40$ passive torque split. The simulation setup is illustrated in Figure 56.



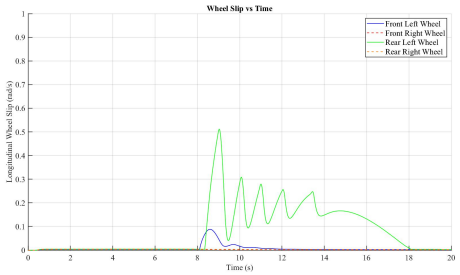

(a) Gas pedal input (b) Road

Figure 56: Setup for slip control testing on a straight road with split $\mu$ surface and AWD configuration

In Figure 57(a), it is observed that the slip control is off and the rear left wheel is slipping as soon as it enters the split $\mu$ surface or ice patch having a slip value of 0.9 around $11s$ and settles as the car leaves the split $\mu$ surface. Whereas in Figure 57(b), slip control is on and the controller tries to limit the torque so that it does not slip and the slip value is reduced to 0.3 at $11s$.



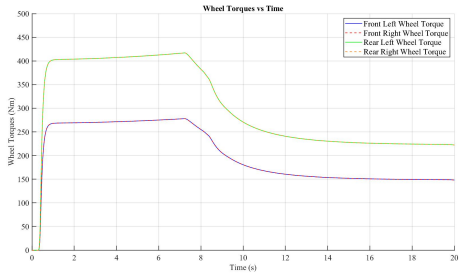

(a) Wheel slip over time without slip control



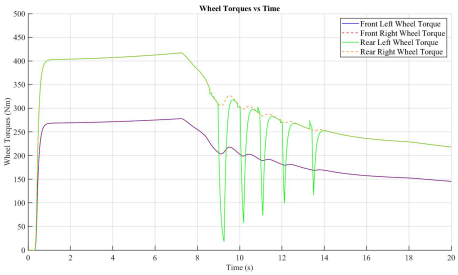

(b) Wheel slip over time with slip control

Figure 57: Effect of slip control on wheel slip over time

In Figure 58(a), it reflects the torque at all the wheels when slip controller is off while in Figure 58(b) where the wheel torque of the rear left wheel is trying to be limited by the slip controller so that the vehicle follows the path and does not slip.

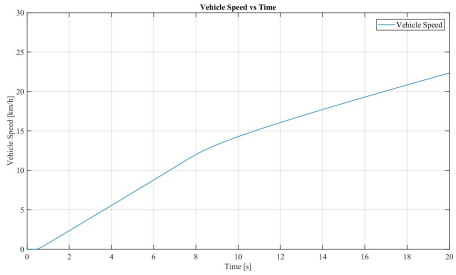



(a) Wheel torques over time without slip control

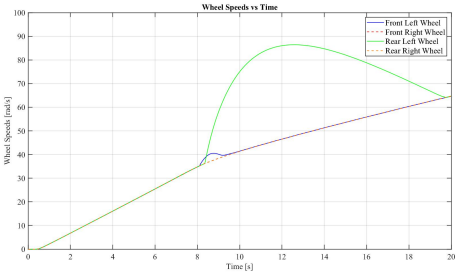



(b) Wheel torques over time with slip control

Figure 58: Effect of slip control on wheel torques over time



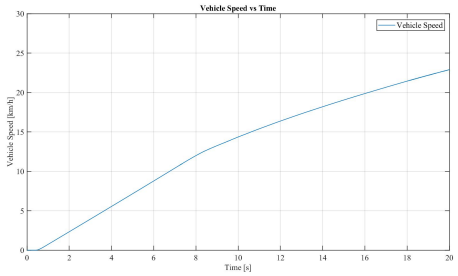

(a) Vehicle speed over time without Slip Control



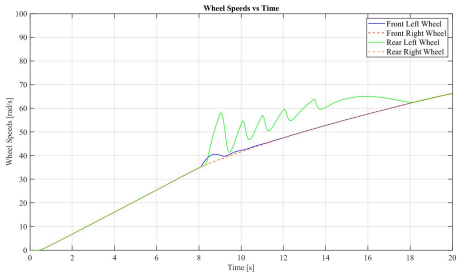

(b) Wheel speeds over time without Slip Control

Figure 59: Vehicle speed and wheel speeds over time without slip control

As seen in Figure 59(a), it shows the speed of the vehicle without slip control which is around $18km/h$ at $12s$ and in Figure 59(b), the wheel speeds are represented in rad/sec and the rear left wheel that is slipping is around $90rad/s$.




(a) Vehicle speed over time with Slip Control



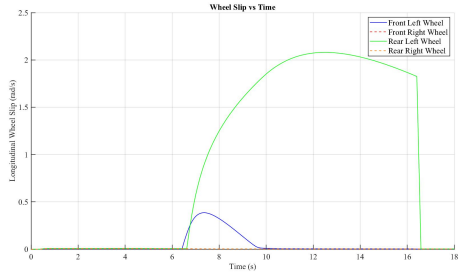

(b) Wheel speeds over time with Slip Control

Figure 60: Effect of slip control on vehicle speed and wheel speeds over time

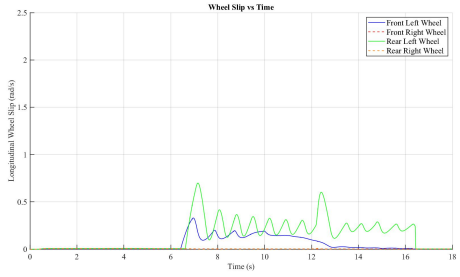
In contrast, as observed in Figure 60(a) and Figure 60(b), it is observed that when the slip control is on, the wheel speed of the rear left wheel is limited to $60 rad/s$ in order to control the vehicle from slipping.

#### 5.2.3.2 Straight road with split $\mu$ surface with 1 motor on each axle configuration

In this case, the conditions are similar to the last one except the configuration which is 1 motor on each axle with an input of 50% gas pedal input and same road with split $\mu$ surface.



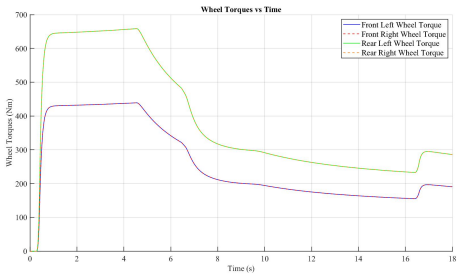

(a) Wheel slip over time without slip control



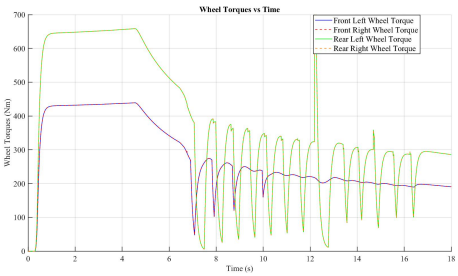

(b) Wheel slip over time with slip control

Figure 61: Effect of slip control on wheel slip over time

Figure 61(a) shows the case when slip control is off and figure 61(b) shows the case when slip controller is on, doing the same thing which is limiting the longitudinal slip and torque as seen in Figure 60(a) and (b) on the slipping wheel, i.e. rear left wheel.




(a) Wheel torques over time without slip control

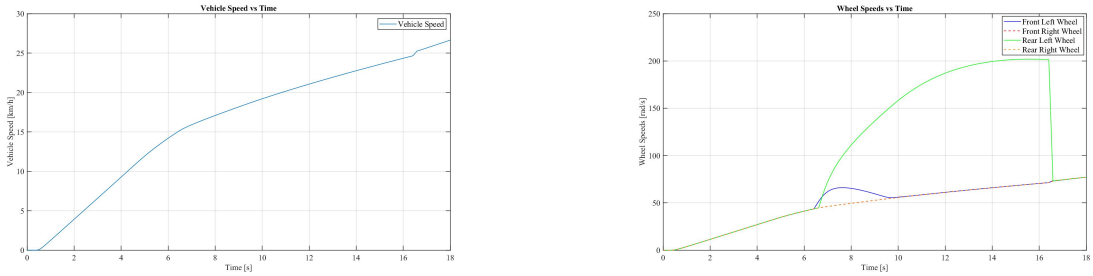



(b) Wheel torques over time with slip control

Figure 62: Effect of slip control on wheel torques over time

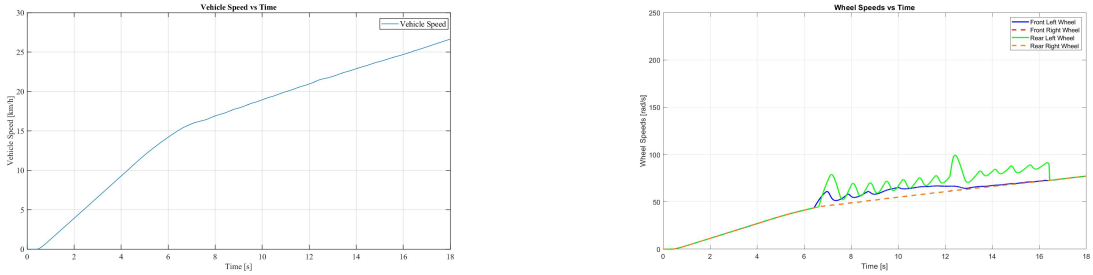
Figure 63 and 64 represent the effect of vehicle speed and wheel speed with slip controller on or

off which is similar to that in the previous test case.




(a) Vehicle speed over time without slip control (b) Wheel speeds over time without slip control

Figure 63: Vehicle speed and wheel speeds over time without slip control

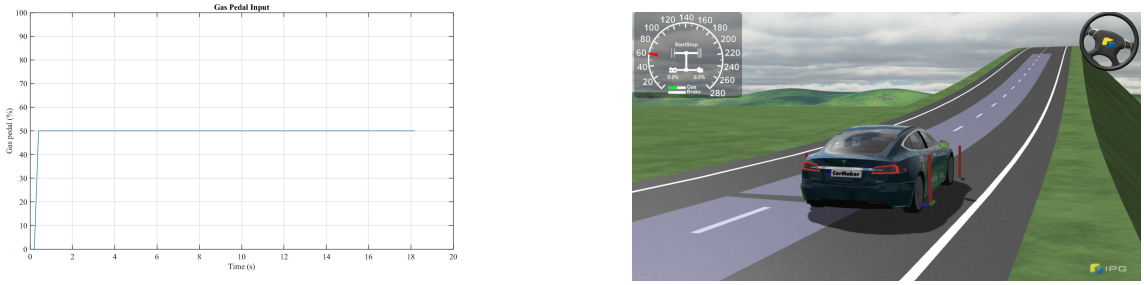



(a) Vehicle speed over time with slip control (b) Wheel speeds over time with slip control

Figure 64: Effect of slip control on vehicle speed and wheel speeds over time

#### 5.2.3.3 15% gradient road with split $\mu$ surface and AWD configuration

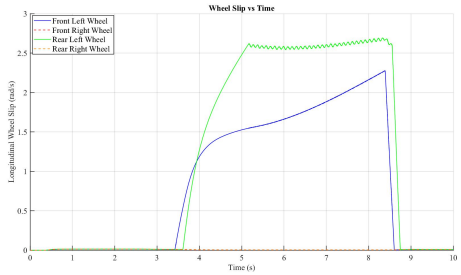
The maneuver was tested with an All Wheel Drive (AWD) configuration or two motors at each axle on a 15% gradient road with split $\mu$ surface having a coefficient of friction as 0.1. It involves a constant gas pedal input of 50%, achieved in $0.2s$ with $60:40$ passive torque split. The simulation setup is illustrated in Figure 65.



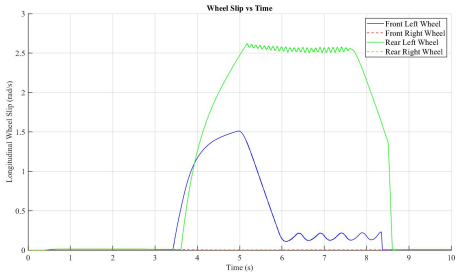

(a) Gas pedal input (b) Road

Figure 65: Setup for slip control testing on a 15% gradient road

In Figure 66(a) it can be seen illustrated that how the wheel slip reacts when the slip controller is off. The moment front left wheel is on the split $\mu$ surface or ice it starts to loose traction thus increasing slip and torque. The slip has a maximum value of 2.3 because of which it is not able to obtain the required torque to climb up the gradient. Post activating the slip controller as seen in Figure 66(b), it limits the slip to a maximum of 1.5 then it ´starts to dip even further once the vehicle is able to up the incline with traction.



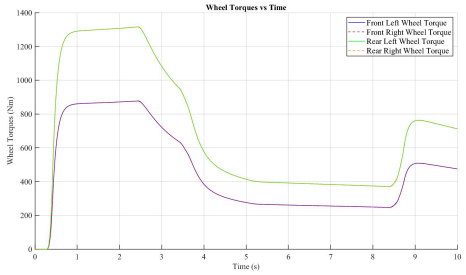

(a) Wheel slip over time without slip control



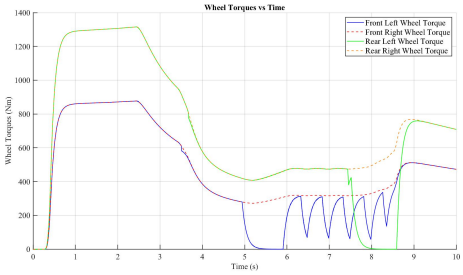

(b) Wheel slip over time with slip control

Figure 66: Effect of slip control on wheel slip over time

Same effect can be observed on the effect of torque as wheel wherein on the activation of the controller, it tries to limit the torque on the front left wheel and some amount of torque at the rear left wheel.



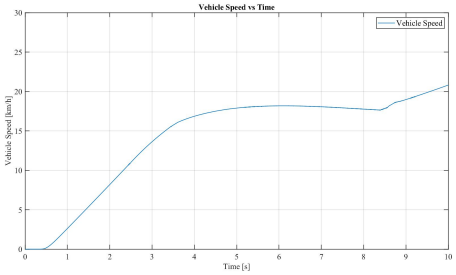

(a) Wheel torques over time without slip control



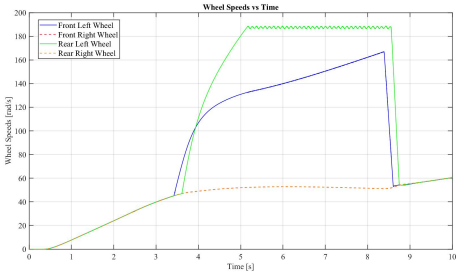

(b) Wheel torques over time with slip control

Figure 67: Effect of slip control on wheel torques over time

Figures 68 and 69 show the difference between the speed of the wheels and the speed of the vehicle when turning on and off the slip controller.

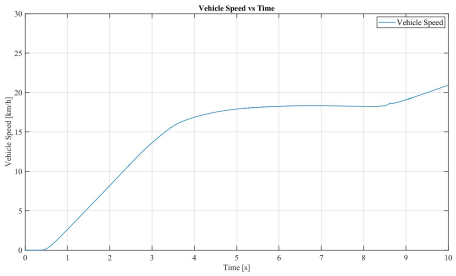



(a) Vehicle speed over time without slip control

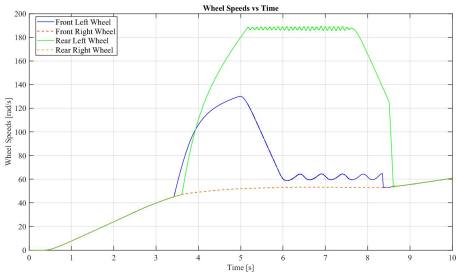



(b) Wheel speeds over time without slip control

Figure 68: Vehicle speed and wheel speeds over time without slip control




(a) Vehicle speed over time with slip control




(b) Wheel speeds over time with slip control

Figure 69: Effect of slip control on vehicle speed and wheel speeds over time

# 6 Conclusion

The primary goal of this project was to design and implement a modular propulsion model customized for electric vehicles, ensuring adaptability across Model-in-the-Loop (MiL), Hardware-in-the-Loop (HiL), and Driver-in-the-Loop (DiL) environments. The resulting model consists of three core modules, each parameterized through a background MATLAB (.m) script. These modules support a variety of powertrain configurations and include advanced features such as torque split, torque vectoring, and slip control. Additionally, the model allows users to seamlessly integrate different control algorithms, offering flexibility and ease of customization.

The final propulsion model has been successfully integrated with CarMaker, where it has undergone extensive validation and testing. The results confirm the real-time capability, configurability and tunability of the model, making it suitable for a variety of driving modes and propulsion system features. This robust and versatile framework provides a solid foundation for further developments in electric vehicle simulation and control.

The tool's independence (agnostic) is ensured by using open and well-organized MATLAB, making it easy to transition to another similar tool. Additionally, the clear and simple interfacing with MATLAB makes CarMaker easily replaceable by other similar tools.

# 7 Future work

Although the model has demonstrated strong performance and adaptability, there are several opportunities for further enhancement to better represent real-world conditions and expand its functionality. Future work may include:

- **Comprehensive DiL Verification:** Validate the propulsion model through detailed Driver-in-the-Loop (DiL) simulations, ensuring accurate representation of driver interactions and system responses in real-world scenarios;
- **ADAS Integration:** Extend the model's capabilities to support Advanced Driver Assistance Systems (ADAS) features, such as Adaptive Cruise Control (ACC), for improved safety and driver comfort;
- **Dynamic Battery Power Testing:** Implement dynamic testing of external battery power limits to ensure accurate response under varying load conditions and to better reflect real-world battery behavior;
- **Driveline Power Losses:** Incorporate the modeling of power losses within the driveline to enhance fidelity, accounting for energy losses due to friction, heat, and inefficiencies in the transmission system;
- **Friction Brake Interaction:** Since brake modeling falls outside the scope of this project, the control structure does not account for interactions between the brakes and the propulsion model. Such interactions could significantly impact the overall system architecture.

By addressing these areas, the propulsion model can be refined to achieve greater accuracy and functionality, supporting the ongoing advancement of electric vehicle technology and simulation frameworks.

# 8 Appendix

The background MATLAB script (parameters.m) is organized as follows:

**Vehicle parameters**

| Variable Name | Unit | Description |
|---|---|---|
| `EXTERNAL_VEH_PARAMS_FLAG` | [*bool*] | When active, the model will read the parameters from the model input |
| `VehicleParams.WheelRadius` | [m] | User-defined wheel radius |
| `VehicleParams.Wheelbase` | [m] | User-defined wheelbase |
| `VehicleParams.TrackWidth` | [m] | User-defined track width |
| `VehicleParams.Mass` | [kg] | User-defined vehicle mass |
| `VehicleParams.H_CoG` | [m] | User-defined center of gravity height |
| `VehicleParams.Mass_F2R` | [/] | User-defined ratio of mass over the front axle |
| `VehicleParams.J_z` | $[\mathrm{kg} \cdot \mathrm{m}^2]$ | User-defined yaw inertia |
| `Steering_ratio` | [/] | Steering ratio |
| `Max_steering_angle` | [rad] | Maximum steering angle |

Table 1: Vehicle parameters

**Powertrain subsystem**

| Variable Name | Unit | Description |
|---|---|---|
| `PTparams.FrontAxle.N_motor` | [\] | Front axle number of motors |
| `PTparams.FrontAxle.Motor.P_max_EM` | [W] | Maximum EM power - Front axle |
| `PTparams.FrontAxle.Motor.T_max_EM` | [N · m] | Maximum EM torque - Front axle |
| `PTparams.FrontAxle.Motor.w_max_EM` | [rad/s] | Maximum EM angular speed - Front axle |
| `PTparams.FrontAxle.Motor.J_EM` | [kg · m$^2$] | EM rotational inertia - Front axle |
| `PTparams.FrontAxle.Motor.Tc_controller` | [s] | Time constant - Front axle controller response delay |
| `PTparams.FrontAxle.Driveshaft.Stiffness` | [N · m/rad] | Driveshaft stiffness - Front axle |
| `PTparams.FrontAxle.Driveshaft.Damping` | [N · m · s/rad] | Driveshaft damping - Front axle |
| `PTparams.FrontAxle.Trans_ratio` | [\] | Ratio between the EM and wheel angular speeds |
| `PTparams.RearAxle.N_motor` | [\] | Rear axle number of motors |
| `PTparams.RearAxle.Motor.P_max_EM` | [W] | Maximum EM power - Rear axle |
| `PTparams.RearAxle.Motor.T_max_EM` | [N · m] | Maximum EM torque - Rear axle |
| `PTparams.RearAxle.Motor.w_max_EM` | [rad/s] | Maximum EM angular speed - Rear axle |
| `PTparams.RearAxle.Motor.J_EM` | [kg · m$^2$] | EM rotational inertia - Rear axle |
| `PTparams.RearAxle.Motor.Tc_controller` | [s] | Time constant - Rear axle controller response delay |
| `PTparams.RearAxle.Driveshaft.Stiffness` | [N · m/rad] | Driveshaft stiffness - Rear axle |
| `PTparams.RearAxle.Driveshaft.Damping` | [N · m · s/rad] | Driveshaft damping - Rear axle |
| `PTparams.RearAxle.Trans_ratio` | [\] | Ratio between the EM and wheel angular speeds |

Table 2: Powertrain subsystem parameters

**Torque management subsystem**

| Variable Name | Unit | Description |
|---|---|---|
| `SC_active` | [*bool*] | It describes the slip control activation |
| `Slip_threshold` | [\] | Slip ratio threshold |
| `SC_p` | [\] | Slip Control proportional gain |
| `SC_i` | [\] | Slip Control integral gain |
| `SC_d` | [\] | Slip Control derivative gain |
| `TV_active` | [*bool*] | It describes the torque vectoring activation |
| `TV_gain_params` | [\] | Torque vectoring controller gain |
| `T_split_F2R_passive` | [\] | Passive (constant) torque split ratio on the front axle |
| `Dynamic_split_active` | [\] | Dynamic torque split activation |

Table 3: Torque management subsystem parameters

**Main control unit**

| Variable Name | Unit | Description |
|---|---|---|
| `MaxSpeed` | [km/h] | Maximum vehicle speed allowed inside the throttle map. Values higher than the maximum speed result in an extrapolated value. |
| `PedalDeadBandLimit` | [\] | Accelerator pedal position value (0 to 1) that results in a non-positive acceleration demand. |
| `MeshSteps` | [\] | Throttle map mesh interpolation points per axis. |
| `Reg.bp1` | [\] | Value 0 to 1 describing the lower breakpoint of the s-curve (Negative acceleration demand - Regeneration). |
| `Reg.bp2` | [\] | Value 0 to 1 describing the upper breakpoint of the s-curve (Negative acceleration demand - Regeneration). |
| `Reg.SmoothSpan` | [\] | Smoothness span of the s-curve (Negative acceleration demand - Regeneration). |
| `Reg.SpeedDeadBandLowerLimit` | [km/h] | Lower limit of the transition region between null acceleration demand and full regeneration (lower smf breakpoint). |
| `Reg.SpeedDeadBandUpperLimit` | [km/h] | Upper limit of the transition region between null acceleration demand and full regeneration (lower smf breakpoint). |

Table 4: Main control unit parameters - Part 1

| Variable Name | Unit | Description |
|---|---|---|
| Prop.Ecobp1 | [\] | Value 0 to 1 describing the lower breakpoint of the s-curve (Eco positive acceleration demand - Propulsion). |
| Prop.Ecobp2 | [\] | Value 0 to 1 describing the upper breakpoint of the s-curve (Eco positive acceleration demand - Propulsion). |
| Prop.EcoSmoothSpan | [\] | Smoothness span of the s-curve (Eco positive acceleration demand - Propulsion). |
| Prop.EcoPowerCurve2 | [\] | Second eco curve creation: this value represents the value to which each point of the base curve is raised. |
| Prop.EcoPosCurve2 | [km/h] | Second eco curve: position over the vehicle speed axis. |
| Prop.EcoPowerCurve3 | [\] | Third eco curve creation: this value represents the value to which each point of the base curve is raised. |
| Prop.EcoPosCurve3 | [km/h] | Third eco curve: position over the vehicle speed axis. |
| Prop.Sportbp1 | [\] | Value 0 to 1 describing the lower breakpoint of the s-curve (Sport positive acceleration demand - Propulsion). |
| Prop.Sportbp2 | [\] | Value 0 to 1 describing the upper breakpoint of the s-curve (Sport positive acceleration demand - Propulsion). |
| Prop.SportSmoothSpan | [\] | Smoothness span of the s-curve (Sport positive acceleration demand - Propulsion). |
| Prop.SportPowerCurve2 | [\] | Second sport curve creation: this value represents the value to which each point of the base curve is raised. |
| Prop.SportPosCurve2 | [km/h] | Second sport curve: position over the vehicle speed axis. |
| Prop.SportPowerCurve3 | [\] | Third sport curve creation: this value represents the value to which each point of the base curve is raised. |
| Prop.SportPosCurve3 | [km/h] | Third sport curve: position over the vehicle speed axis. |

Table 5: Main control unit parameters - Part 2

| Variable Name | Unit | Description |
|---|---|---|
| `Prop.Custombp1` | [\] | Value 0 to 1 describing the lower breakpoint of the s-curve (Custom positive acceleration demand - Propulsion). |
| `Prop.Custombp2` | [\] | Value 0 to 1 describing the upper breakpoint of the s-curve (Custom positive acceleration demand - Propulsion). |
| `Prop.CustomSmoothSpan` | [\] | Smoothness span of the s-curve (Custom positive acceleration demand - Propulsion). |
| `Prop.CustomPowerCurve2` | [\] | Second custom curve creation: this value represents the value to which each point of the base curve is raised. |
| `Prop.CustomPosCurve2` | [km/h] | Second custom curve: position over the vehicle speed axis. |
| `Prop.CustomPowerCurve3` | [\] | Third custom curve creation: this value represents the value to which each point of the base curve is raised. |
| `Prop.CustomPosCurve3` | [km/h] | Third custom curve: position over the vehicle speed axis. |

Table 6: Main control unit parameters - Part 3

# 9 Bibliography

[1] https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets

[2] Guzzella L, Sciarretta A, Springer-Verlag Gmb H. Vehicle Propulsion Systems Introduction to Modeling and Optimization. Berlin: Springer Berlin, 2015.

[3] Beacock, Benjamin. "Control System and Simulation Design for an All-Wheel-Drive Formula SAE Car Using a Neural Network Estimated Slip Angle Velocity," n.d.

[4] Blakqori, Albijon, and Mille Kotur. "Virtual Verification Framework for Vehicle Motion Systems," n.d.

[5] Efremov, Denis, Tomáš Haniš, and Martin Klaučo. "Vehicle and Wheels Stability Defined Using Driving Envelope Protection Algorithm." IEEE Transactions on Intelligent Transportation Systems 25, no. 9 (September 2024): 11304–16. https://doi.org/10.1109/TITS.2024.3362064.

[6] Guglielmi, Paolo, and Domenico Barile. "ELECTRIC POWERTRAIN CONTROL: AN ADAPTABLE SIMULINK MODEL OF TORQUE PATH AND VALIDATION BY MODEL IN THE LOOP," n.d.

[7] Jain, Chirag Vasanth. "Modelling of a Rear Axle Torque Vectoring Dual Clutch Driveline for Battery Electric Vehicles with Verification in a Virtual Environment," n.d.

[8] Kiyakli, Ahmet Onur, and Hamit Solmaz. "Modeling of an Electric Vehicle with MATLAB/Simulink." International Journal of Automotive Science and Technology 2, no. 4 (December 31, 2018): 9–15. https://doi.org/10.30939/ijastech..475477. Sjoblom, Jonas, Bengt Jacobson, Christian Tsobanoglou, Ganesh Jayanna Kundur, Mohammad Siam Siraj, Nitesh Anil Kumar Nikkam, Simon Tsobanoglou, and Varun Ramakrishnan Bharadwaj. "Model Exchange for Virtual SIL/MIL Verification of Passenger Cars with Electric Axles," n.d.

[9] The Math Works, Inc. "The Math Works Inc." SIMULATION 57, no. 4 (October 1991): 240–240. https://doi.org/10.1177/003754979105700407.

[10] Tomar, Vedant, A Chitra, Daki Krishnachaitanya, N S Raghavendra Rao, Indragandhi V, and W Raziasultana. "Design of Powertrain Model for an Electric Vehicle Using MATLAB/Simulink." In 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), 1–7. Kuala Lumpur, Malaysia: IEEE, 2021. https://doi.org/10.1109/i-PACT52855.2021.9696518.

[11] Wang, Lidong. "Vehicle Dynamics Development Process With Offline and Driver-in-the-Loop (DIL) Simulation," n.d. Wu, Guang, and Zuomin Dong. "Design, Analysis and Modeling of a Novel Hybrid Powertrain System Based on Hybridized Automated Manual Transmission." Mechanical Systems and Signal Processing 93 (September 2017): 688–705. https://doi.org/10.1016/j.ymssp.2016.12.029.

[12] Accurate Technologies, Inc. Pi-Innovo EV Strategy. https://www.accuratetechnologies.com/PDFs/Pi-Innovo_EV_Strategy.pdf.

[13] Mangia, A., Lenzo, B. & Sabbioni, E. An integrated torque-vectoring control framework for electric vehicles featuring multiple handling and energy-efficiency modes selectable by the driver. Meccanica 56, 991–1010 (2021). https://doi.org/10.1007/s11012-021-01317-3

[14] Knauder, Bernhard & Savitski, Dzmitry & Theunissen, Johan & De Novellis, Leonardo. (2014). Electric Torque Vectoring for Electric Vehicles. ATZelektronik worldwide. 9. 50-55. 10.1365/s38314-014-0268-0.