Car model

Model-based design

Development: Davide Capuzzo

Last revision on 08/07/2020

# Introduction

This document aims at introducing the model of an all-wheel drive electric vehicle. The purpose of this model is to allow the verification and validation of developed subsystems. Indeed, having a global model of the car is necessary to test the various systems, designed following a model-based design approach.

In order to obtain and validate a full model, an ideal environment is necessary. Simulink provides suggestions on how to organize the simulation environment. Mathworks also provides a complete car model with simulation of some maneuvers which stress the components of the car, having the goal of finding some boundary conditions and testing, in a virtual environment, how a new developed subsystem integrates with the other components of the vehicle. Another advantage of the Matlab/Simulink development environment is the presence of a block simulating the driver behavior, that enables more realistic simulations of the complete system.

## System requirements

In order to interface with the model, the following software modules are required:

* MATLAB R2019b and Simulink
* Powertrain Blockset
* Vehicle Dynamics Blockset

# Design requirements

The goal of the all-wheel drive electric vehicle model is to create a simulating environment for the design and testing of a newly developed virtual differential system and ESP control system.

The vehicle is an electric sedan with a central battery pack. The are also four in-wheel motors that provide the necessary torque to drive the car. The virtual differential will have to provide the right torque to each wheel in order to perform as good as a mechanical active differential. Such torque will also be influenced by the ESP control system, which must provide control signals to increase the global stability of the vehicle while cornering.

The model of the vehicle must allow the verification and validation of the designed controllers and subsystems. It must perform as much as possible, during simulation, like a real-world sedan vehicle. Indeed, the model has the ultimate goal of reducing development and testing time of safety-critical systems which will be used on-board.

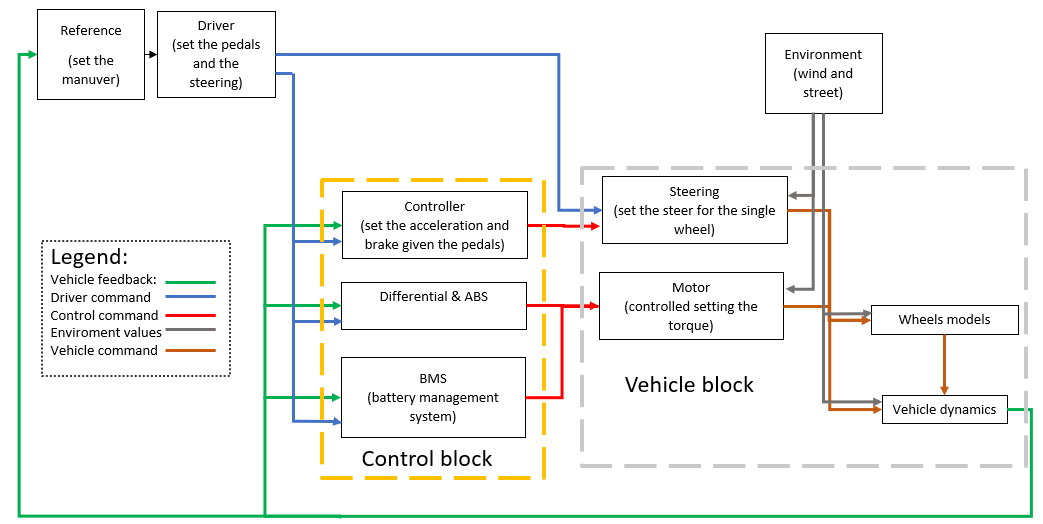
The model’s parameters must be comparable with related real-world vehicle’s parameters.

The behavior of a real driver shall be simulated as well, to validate the systems when interacting with final consumers.

# Model-based design

Matlab/Simulink provide a simulated environment where simulated vehicle is driven, and a block emulating the behavior of the driver.

The availability of a Simulink block scheme modelling a complete car allowed to reduce design time and validation of the model, as an already validated model is provided.

The model’s blocks and subsystems are organized as depicted below.

* The refence generator block is a block that generates the reference environment for the simulation of the car model; in this block it is possible to set the kind of the maneuver and related conditions.
* The driver command is a block that, given as input the feedback of vehicle dynamics and the reference of the maneuver, provides as output the ratio of the throttle and brake pedals and the steering angle, simulating the behavior of a driver in a predictive way.
* In the control block there are all the electronic controls of the vehicle; in particular, given as input the command of the driver and the vehicle’s feedback, it manages the batteries and controls the motors and brakes. In this block, torque is set for ABS and ESP control.
* In the environmental block, all the parameters to simulate the interaction between the car and the environment are present.
* In the passenger vehicle block, there is the complete model of a standard electric sedan car. Inside it is split in two modules, interacting with each other:
  + The mechanical block, containing the models of the steering, the brake, the motors; and the batteries;
  + The body block, modelling the chassis and the interaction with road.

For further details on the reference vehicle model, refer to Matlab documentation:

* [Constant Radius Reference Application](https://www.mathworks.com/help/vdynblks/ug/constant-radius-maneuver-reference-application.html)
* [EV Reference Application](https://www.mathworks.com/help/autoblks/ug/electric-vehicle-reference-application.html)

# Usage

## System interfaces

The model presents plenty of internal signals that may be used by subsystems. Specifically, it is possible to access to every variable describing the state of the vehicle.

The model provides all the signals required by the developed subsystems to operate as intended. Refer to specific subsystem report to see the list of signals required for correct operation.

## System parameters

The complete all-wheel drive electric vehicle model, with four in-wheel electric motors, presents many settable parameters, comprising those managing the various kinds of maneuver.

A list of all the parameters, which may be set, is reported below.

More information are available onCommonVariables.m file

**Vehicle parameters**

Forward location of tire, [m]: *VEH.FrontAxlePositionfromCG*

Rearward location of tire, [m]: *VEH.RearAxlePositionfromCG*

Vehicle mass, [kg]: *VEH.Mass*

Track hardpoint coordinates relative to axle center, [m]: *VEH.TrackWidth*

Vertical distance from center of mass to axle plane, [m]: *VEH.HeightCG*

Initial position in inertial frame [Xeo,Yeo,Zeo], [m]: *VEH.InitialLongPosition*, *VEH.InitialLatPosition*, *VEH.InitialVertPosition*

Initial velocity in body axes [xdot\_o,ydot\_o,zdot\_o], [m/s]: *VEH.InitialLongVel*, *VEH.InitialLatVel*, *VEH.InitialVertVel*

Initial Euler orientation [roll, pitch, yaw], [rad]: *VEH.InitialRollAngle*, *VEH.InitialPitchAngle*, *VEH.InitialYawAngle*

Initial body rotation rates [p,q,r], [rad/s]: *VEH.InitialRollRate*, *VEH.InitialPitchRate*, *VEH.InitialYawRate*

Chassis inertia tensor, [kg\*m^2]: *VEH.RollMomentInertia*, *VEH.PitchMomentInertia*, *VEH.YawMomentInertia*

Longitudinal drag area, [m^2]: *VEH.FrontalArea*

Wheel radius, [m]: *VEH.WheelRadius*

Max speed, [m/s]: *VEH.MaxSpeed*

**Suspensions**

Axle and wheels lumped principal moment of inertia about longitudinal axis, [kg\*m^2]: *AxlIxx*

Axle and wheels lumped mass, [kg]: *AxlM*

Wheel and axle interface compliance constant, [N/m]: *KzWhlAxl*

Wheel and axle interface compliance preload, [N]: *F0zWhlAxl*

Wheel and axle interface damping constant, [Ns/m]: *CzWhlAxl*

Suspension spring constant, [N/m]: *Kz*

Suspension spring preload, [N]: F0z

Suspension shock damping constant, [Ns/m]: *Cz*

Suspension maximum height, [m]: *Hmax*

Vehicle Adapter[m]: *h*

**Litium ion battery pack**

Rated capacity at nominal temperature, [Ah]: *BattChargeMax*

Open circuit voltage table data, [V]: *Em*

Open circuit voltage breakpoints 1, [V]: *CapLUTBp*

Internal resistance table data, [Ohms]: *RInt*

Battery temperature breakpoints 1, [K]: *BattTempBp*

Battery capacity breakpoints 2[Ah]: *CapSOCBp*

Number of cells in series: *Ns*

Number of cells in parallel: *Np*

Initial battery capacity, [Ah]: *BattCapInit*

**Motors**

Torque control time constant, [s]: *Tc*

Maximum torque, [Nm]: *torque\_max*

Maximum power, [W]: *power\_max*

**Brake control**

Converts Brake Pedal Position to Brake Pressure Request[N]: *BrkPrsMax*

**Throttle control**

Accelerator Pedal Position to Torque Command: *MotSpd* [rad/s], *MotTrq* [N/m]