Drive-by-wire electronic differential with ESP control capabilities

Author: Davide Capuzzo

Author: Federico Galli

Author: Alessandro Luppi

Last revision on 12/07/2020

1 Introduction

The goal of this dissertation is to introduce the design procedures, describe the development of units and systems, analyze the results of the designed drive-by-wire electronic differential with ESP control capabilities.

In the following, there will be introduced:

- The documents accompanying this dissertation;
- The organization of the team and the project;
- The requirements driving the design of the project;
- The development and testing of units and systems;
- The integration and validation procedures.

The designed system is a control system composed of a virtual differential and an ESP controller. The differential is virtual as it is not physically implemented by means of mechanical components, rather it is emulated by an elaboration unit. The aim of the ESP controller is to increase the stability of the vehicle.

The control system is though to be implemented on an all-wheel drive electric car having four on-wheels electric motors driving the four wheels of the vehicle.

The developed control system is not ready yet to be implemented on a vehicle as other validation steps are required, since it constitutes a potentially safety-critical system.

1.1 System requirements

The design has all been performed on Matlab/Simulink environment.

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

- MATLAB R2019b and Simulink;
- Powertrain Blockset;
- Vehicle Dynamics Blockset.

To complete the PIL validation procedure, an additional software package is also required:

- Embedded Coder Support Package for STMicroelectronics Discovery Boards.

Finally, the generated code has been deployed on a target board, requiring the following hardware:

- STMicroelectronics STM32F4-Discovery board;
- USB type A to Mini-B cable;
- USB TTL-232 cable – TTL-232R 3.3V (serial communication for STM32F4-Discovery board).

2 Related documents

All the development and validation files, together with the related reports detailing the design procedures, are stored on a repository on GitHub, which can be found at the url:

https://github.com/meltinglab/electronic-differential

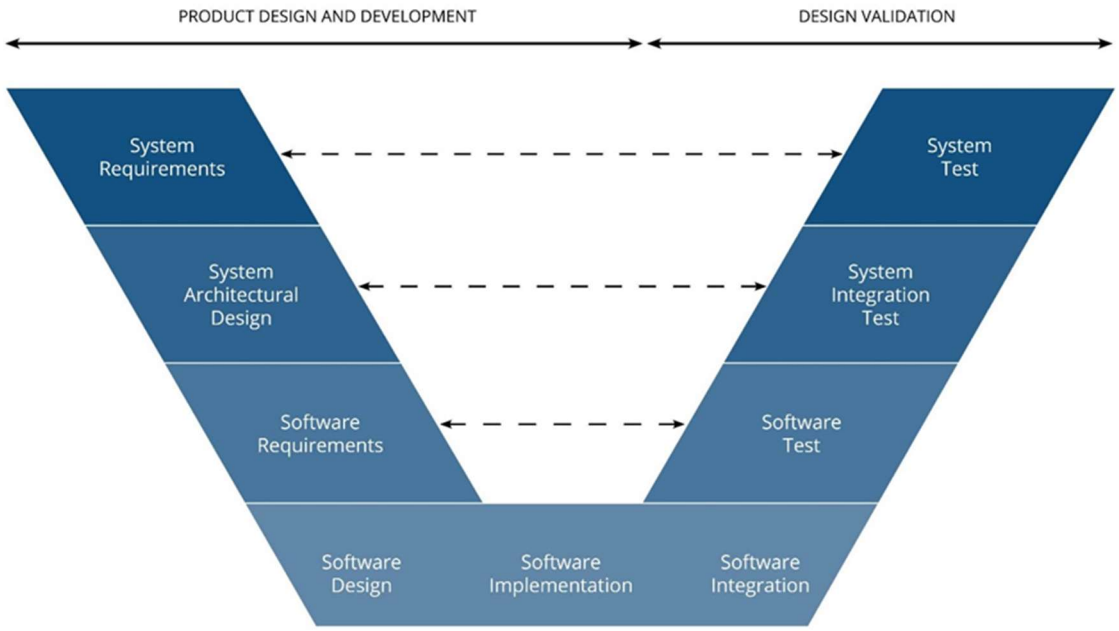
Specifically, for more detailed documentation on the development and validation of the systems, some reference documents can be found under the *reports* directory:

- Car Model: introduces the Simulink model of the car, with which the developed control system is supposed to interact with.
- Electronic Stability Program: describes the development and testing of the ESP controller unit.
- Virtual Differential: describes the development and testing of the virtual differential controller unit.
- Integration: introduces the integration procedure and illustrates some tests performed on the integrated system.
- Validation SIL and PIL: describes the method used to validate the developed control system according to SIL and PIL validation processes.

3 Project and team organization

3.1 Project organization

In software development, the V-model represents a development process that may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development process and its associated phase of testing and validation. The horizontal and vertical axes represent time, or project completeness, (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.



The approach followed throughout all the development of the controller system described in this dissertation is well depicted by the above picture.

The development process starts from System Requirements phase, where a definition of the high-level system behavior is carried out, including all the requirements needed to obtain a high accuracy controller for the intended application. The successive step is the System Architectural Design, which comprehends the definition of the individual software modules for the controller, the development of the car model where to integrate the controller, the

design of the verification process, and the deployment of the final validation phase on target hardware. Thus, Software Requirements definition is possible: a characterization of the structure of the ESP controller and the virtual differential system, with the definition of input and expected output values, as well as the physical equations (the physical model, indeed) that lie behind the behavior of the controllers, responsible for the desired performance, has been accomplished.

The two branches of the V-model are linked through a design phase in which developers must take care of Software design, its implementation and then its integration. In the project, Software Design has been represented by actual implementation on Matlab/Simulink of the software modules defined in the previous steps using both hand-written Matlab code and Simulink blocks. Then, Automatic Code Generation played a significant role in Software Implementation phase. Hence, the code was ready to be implemented on the target hardware, which has been properly configured.

Finally, verification and validation phases have been carried out. Firstly, controllers have been tested individually, by giving them specific input vectors and defining particular case scenarios in order to analyze their response and check whether their behaviors were correct or not (what can be called unit testing, preliminary Software Test run in Matlab environment on host computer). After that, the behavior of the controllers integrated with the car system has been checked; several tests have been performed on the integrated system (integration testing, System Integration Test still run on host computer), pushing the controller to work in different situations and with different parameters (making the car following several trajectories). Finally the last kind of testing, for the aim of the project and the resources available, has been accomplished: Software-In-the-Loop (SIL), with controller running on the host computer, and Processor-In-the-Loop (PIL), for which controllers are implemented on a target board, the STM32F4-Discovery. These two simulation tests highlighted the correspondence between the C code of the controller, generated by Matlab Embedded Code Generator (the automatic code generation tool), and the controller model.

Further tests could be possible such as on-field testing with a real car, and they should be performed before implementing the safety-critical system on a marketable vehicle, but they are out of scope for the purpose of this project.

3.2 Team organization

The project required some design steps, which have been distributed among team members to effectively reach the requested results. Some checkpoints allowed the team members to check the obtained results, meet design deadlines and organize remaining work. During the whole development, communication among team members was fundamental for the continuation of the project.

The design steps are summarized below, each near to the team member assigned to it. Checkpoints are also illustrated.

- Team: analysis and choices on project requirements and specifications
    - First meet point
- Davide: development of car model on Simulink environment
- Alessandro: design of virtual differential subsystem
- Federico: design of ESP controller subsystem
    - Second meet point
- Davide: unit testing on virtual differential subsystem
- Alessandro: unit testing on ESP controller subsystem
- Federico: check on vehicle model
    - Third meet point
- Davide, aided by Alessandro: integration of the systems
- Federico, aided by Alessandro: preparation to code generation and validation
    - Fourth meet point
- Team: validation by means of MiL, SiL, PiL procedures
    - End of the project

All the steps, introduced in chronological orders, have been supported by reports constituting the documentation.

4 Objective of the design

The developed system is intended to work on an electric sedan vehicle of medium dimensions, with 4 independent driving wheels, having one electric motor each.

The aim of the control system is to substitute the mechanical components transmitting power from a central engine to the wheels on traditional vehicles, directly controlling the wheels rotation by means of control signals produced by an electronic control unit. The control system shall also properly counteract the effects of slipping due to lack of friction on critical condition roads (e.g. icy road, wet asphalt, etc.).

The system, that combines the actions of both a mechanical differential and an ESP controller, must provide the correct wheel speed to each of the four wheels in order to safely follow the speed and trajectory intended by the driver and avoid lateral skidding despite the road/conditions inefficiencies. The overall task of the controller is thus to enhance the performance of the vehicle in normal driving conditions and increase the stability in critical and harsh case scenarios.

The objective of the control system under development is therefore to produce control signals which allow, when integrated with the rest of the vehicle, to drive the wheels and move the vehicle as requested by the driver. The system shall have a negligible loss of performance with respect to its mechanical counterpart regardless of the driving scenario, rather, it should increase the performance of the vehicle under various road and driving conditions. Finally, it is expected to increase the stability of the vehicle when turning and skidding.

5 Virtual differential

The virtual differential is a subsystem of the complete all-wheel-drive electric vehicle under development. Its aim is to produce a reference wheel speed, for each of the four wheels, which will be tracked by the electric motor driving the wheel, with the eventual addition of correction factors for enhanced vehicle stability. The introduced differential model is called "virtual" as there not exist a mechanical component transferring power from a single engine to the wheels; rather, the virtual differential will be implemented by an electronic control unit actively controlling actuators which will drive the wheels' rotation.

A summary of the development and unit testing procedures is introduced below. For further details, see the related document.

5.1 Model-based design

The design of the virtual differential subsystem has been based on the Ackerman steering geometry assuming ideal vehicle and road conditions.

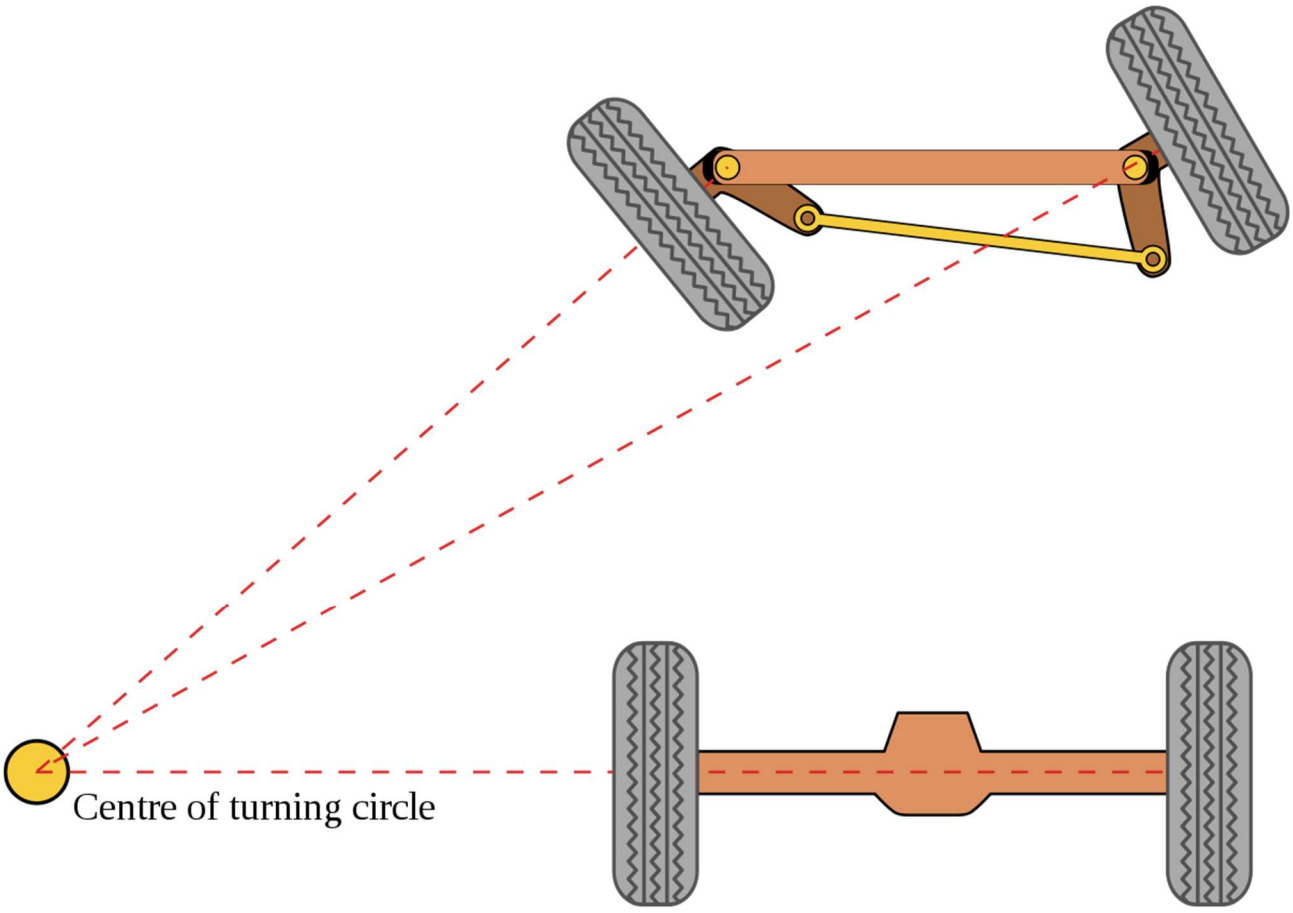
The assumptions for the design are:

- Ideal vehicle and road conditions, leading to the non-slip condition; road is dry asphalt, and vehicle moves on the road without any lateral skidding; eventual skidding will be addressed by other systems.
- Front and rear axles have equal length, and wheels have the same dimension (radius); these parameters are known, as well as the distance between front and rear axles.
- Throttle pedal position reflects the reference vehicle speed, and no other signal influence the requested vehicle speed; action of brake pedal will be addressed by other systems.
- There are sensors providing the instantaneous angular position of the two front wheels.

These assumptions conform with the other developed systems for the all-wheel-drive electric vehicle. The integration of the virtual differential subsystem with other vehicle's systems introduce further complexity and eliminates some of the idealized assumptions.

The Ackerman steering model describes the reference steering geometry on which most of the vehicles rely on, as it introduces a situation in which none of the wheels of the vehicles is subject to longitudinal slip under ideal driving conditions. Variations from the model allow to reach different levels of performance. It is assumed that the all-wheel-drive electric vehicle under development implements a steering mechanism which conforms with the Ackerman steering model.

According to Ackerman steering geometry, at any steering angle the four vehicle's wheels rotates around the same center of turning. It means that, during any turn, the lines perpendicular to the wheels' axis meet at a single point, corresponding to the center of turning around which the vehicle is rotating. Assuming a front-steering vehicle, the center of turning lies on the line connecting the two rear wheels, corresponding, inside the vehicle, with the rear axle.



Once the geometry of the vehicle is known, the speeds of the four wheels can be computed from the knowledge of the steering angle and the (reference) speed of the vehicle.

5.2 Usage

The virtual differential model may be used whenever the ideal speeds of the four wheels of a vehicle have to be determined. The speeds are ideal as they assume no-slip condition and a vehicle implementing a steering mechanism compliant with Ackerman steering geometry.

In order to use the virtual differential model, some variable parameters, usually provided by in-vehicle sensors or electronic control units, and some fixed parameters, known from car geometry, must be available. These are described in more details below.

5.2.1 System interfaces and parameters

The virtual differential model requires 2 input signals:

- **WhlAng** – 2x1 or 1x2 vector (rad)
    - First vector element is the wheel angle of left wheel, second vector element is the wheel angle of the right wheel. The two elements represent the wheel angles of the two front wheels. A wheel angle is the angle between the longitudinal axis of the wheel, perpendicular to the axis of rotation, and the longitudinal axis of the vehicle. The two wheel angles must satisfy Ackerman geometry for the vehicle in use.
- **RefSpd** – scalar (m/s)
    - Reference speed at which vehicle is supposed to go. In case of constant vehicle speed, reference speed equals current speed. Reference speed may be provided by reading of throttle pedal position or by an ADAS feature requesting it, such as Adaptive Cruise Control or Traction Control features. Negative values refer to vehicle moving backward.

The virtual differential model provides 1 output signal:

- **WhlSpd** – 4x1 vector (rad/s)

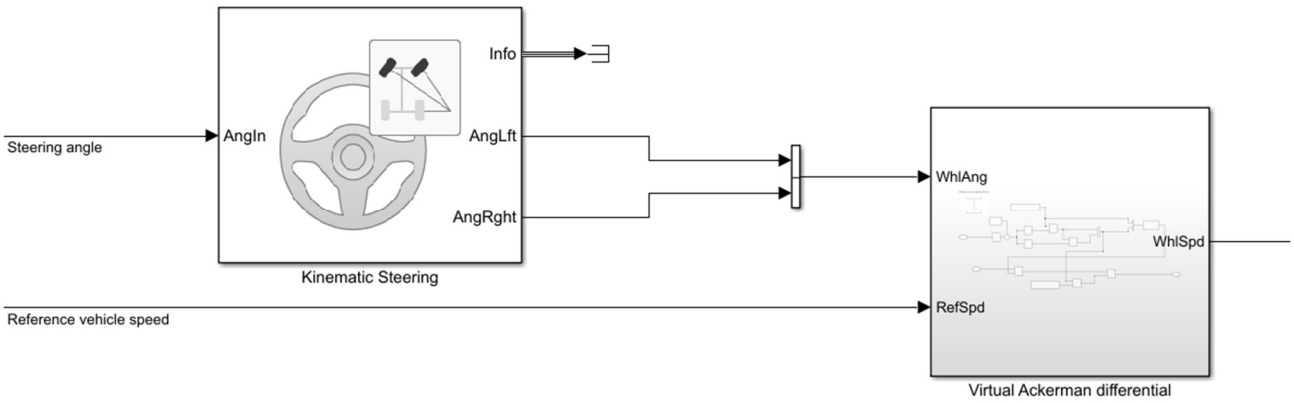
- Each element reports the wheel angular speed of a wheel of the vehicle. Wheels' speeds are, in order: front-left, front-right, rear-left, rear-right. Negative values refer to wheels rotating in reverse direction, that is when vehicle is backward moving.

The virtual differential model requires two vehicle parameters:

- **Wheel base** – scalar (m)
  - It represents the distance between front and rear axles of the vehicle.
- **Wheel radius** – scalar (m)
  - It represents the radius of the wheels, assumed to be equal for all the four wheels. It is the distance between wheel's center of rotation and a planar ground neglecting elastic effects of wheel.

5.2.2 How to use

To use the virtual differential model, input signals and parameters must be provided as explained above. Here an example of use is shown.



The two wheel angles must be provided as a vector at first input port. They must comply with the Ackerman steering geometry, and one way to ensure it is to use a Kinematic Steering block with steering type set to "Ackerman". In this way, the steering angle, from steering wheel, is only required to provide the two wheel angles, which must be concatenated in a single vector.

On the other hand, the reference speed may come from reading of the throttle pedal position, and it must represent the speed at which vehicle is required to move. Note that no braking force is considered at this stage, any braking action must be added consequently.

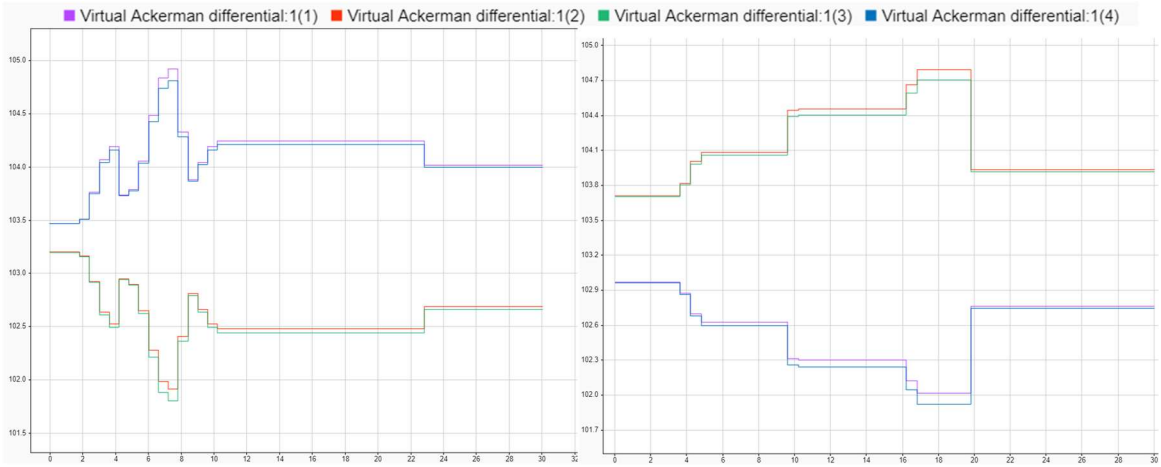
The wheels' speeds provided by the virtual differential model are provided as a vectorial signal, and they may be used as reference speeds in on-wheel motors, to achieve the requested vehicle speed while maintaining the benefits of a traditional (i.e. mechanical) differential system.

5.3 Unit testing

In order to have a complete test of the unit, normal behavior and boundary conditions have been checked:

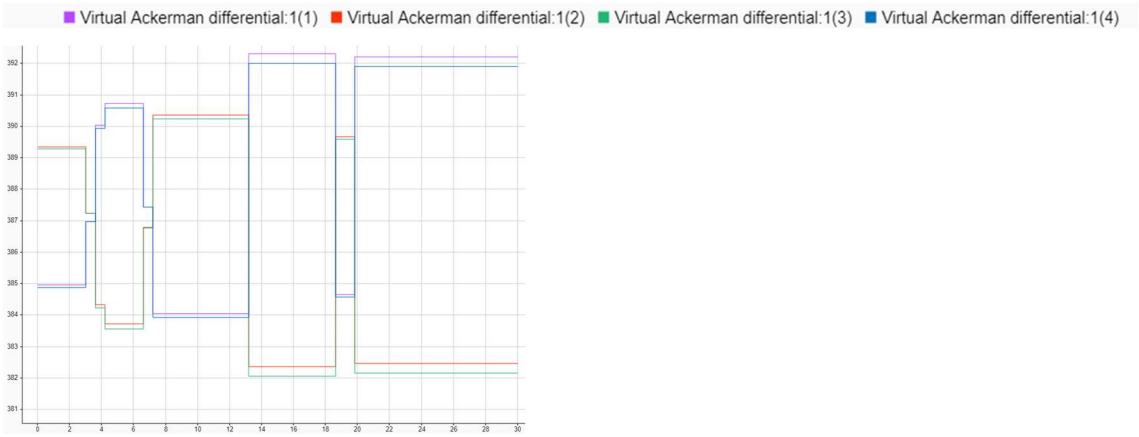
- Normal conditions:

  In normal conditions, if the vehicle is steering to right (left image) the left wheels rotate faster than the right wheels, and vice versa (right image). It is possible to notice that the rear wheels rotate slower than the front wheels due to the smaller steering radius.



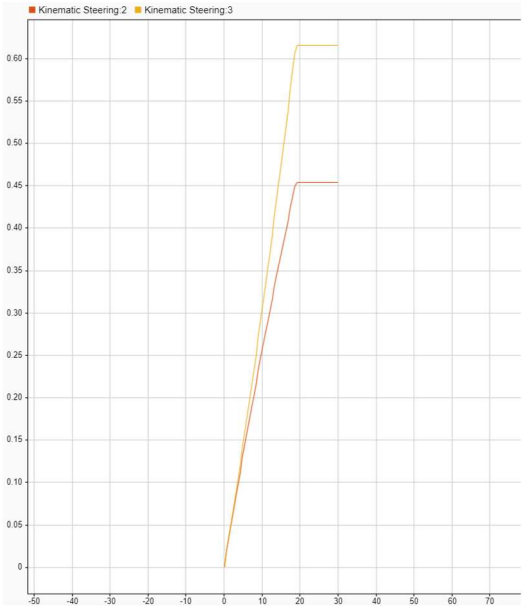
- Reference speed going to + ∞:

  If the reference speed goes to +∞, the system limits the max speed at 120 m/s. This is the expected behavior reported in the documentation.



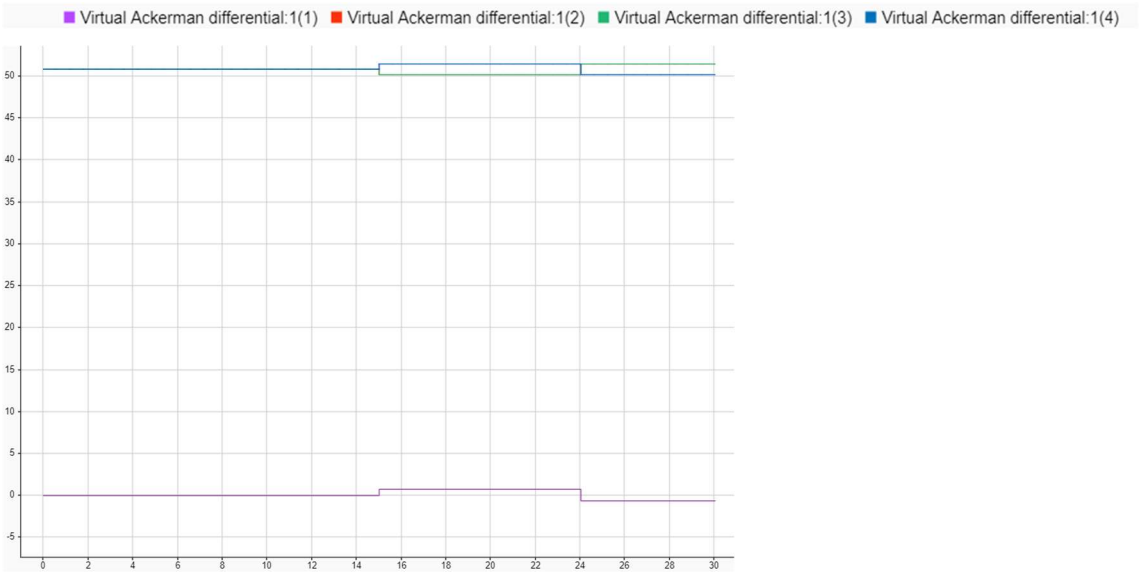
- Steering angle above 75°.

  Setting a ramp as input, it is possible to notice that after 75° the system saturates, and the output remains constant.



- Steering angle changes as a step.

  If the steering angle changes as a step, the virtual differential block follows it without delays, as expected due to the absence of dynamic elements. In the image, the bottom signal represents the input signal applied during evaluation.



- Using steering angles for the two front wheels that do not respect Ackerman geometry, the virtual differential still provides outputs, even if they have wrong values. This is expected, as respecting Ackerman geometry is a requirement for the inputs.

The behavior of the unit has proven to be the one expected.

6 ESP controller

The electronic stability program (ESP) is a subsystem of the complete all-wheel-drive electric vehicle under development. Its aim is to produce a wheel speed variation, for each of the four wheels, which will be added to the wheel speed given by the virtual differential. The ESP controller will be implemented by an electronic control unit actively controlling actuators which will drive the wheels' rotation.

A summary of the development and unit testing procedures is introduced below. For further details, see the related document.

6.1 Model-based design

The design of the ESP subsystem has been based on the vehicle dynamics: it has been obtained from the combination of the kinematics equations, the inertial accelerations and the rotative dynamics of the vehicle with the expression of the forces (aerodynamics and wheel forces) acting on it.

The complete vehicle system could be expressed as a generic nonlinear system; a linearization allows to obtain a linear system having state x represented by slip angle β and yaw rate ω_z, disturbance w represented by steering angle δ_w, and control u represented by the four wheel speed variations $d\omega_{1,2,3,4}$. Thus, linearized matrices of the system are computed, so that a state space representation of the relevant planar vehicle dynamics can be written.

The system has been designed by means of optimal control. An optimization function has been defined through the two matrices Q and R, which carry information about the maximum allowable error on the state and the maximum control effort, respectively. The gain matrix Klqr is obtain as the optimal gain of a linear quadratic regulator. Matlab design environment offers a *lqr* function to compute the optimal gain, which takes as input the linearized system matrices A and B, and the optimization function matrices Q and R.

The assumptions for the design are:

- Front and rear axles have equal length, and wheels have the same dimension (radius); these parameters are known, as well as the distance between front and rear axles.
- There are sensors providing the instantaneous angular position of the two front wheels.
- Linearization is performed around a straight trajectory, so linearization point is

  $x_0 = [\beta_0; \omega_{z0}] = [0; 0]$, $\delta_{Wi0} = 0$, $d\omega_{i0} = 0$.
- Total slip λ_{Ti} is kept close to 0 so that it is possible to linearize the friction coefficient μ.

These assumptions conform with the other developed systems for the all-wheel-drive electric vehicle. The integration of the ESP subsystem with other vehicle's systems highlights its efficacy even in absence of the above-mentioned idealized assumptions.

6.2 Usage

The ESP control model may be used whenever the ideal speed variations of the four wheels of a vehicle have to be determined in order to address the lack of performance of the vehicle due to slippery condition of the road. The speeds are ideal as they are calculated on a basis of specific assumptions and a vehicle implementing a steering mechanism compliant with Ackerman steering geometry.

In order to use the ESP control model, some variable parameters, usually provided by in-vehicle sensors or electronic control units, and some fixed parameters, known from car geometry, must be available. These are described in more details below.

6.2.1 System interfaces and parameters

The ESP control model requires 3 input signals:

- **SlipAng** – scalar (rad)
  - Slip angle is a scalar because we are assuming that each wheel has the same slip angle β. The slip angle is the angle between the direction in which a wheel is pointing and the direction in which it is actually traveling.

- **YawRate**– scalar (rad/s)
    - Yaw rate ω_z of the vehicle is a scalar quantity. The yaw rate is the angular velocity of its rotation around the z-axis of the body reference frame.
- **CrnSpd** – scalar (m/s)
    - Current speed the vehicle is travelling at. It is supposed to be constant and equal to the reference speed for the purpose of this system. Negative values refer to vehicle moving backward.

The virtual differential model provides 1 output signal:

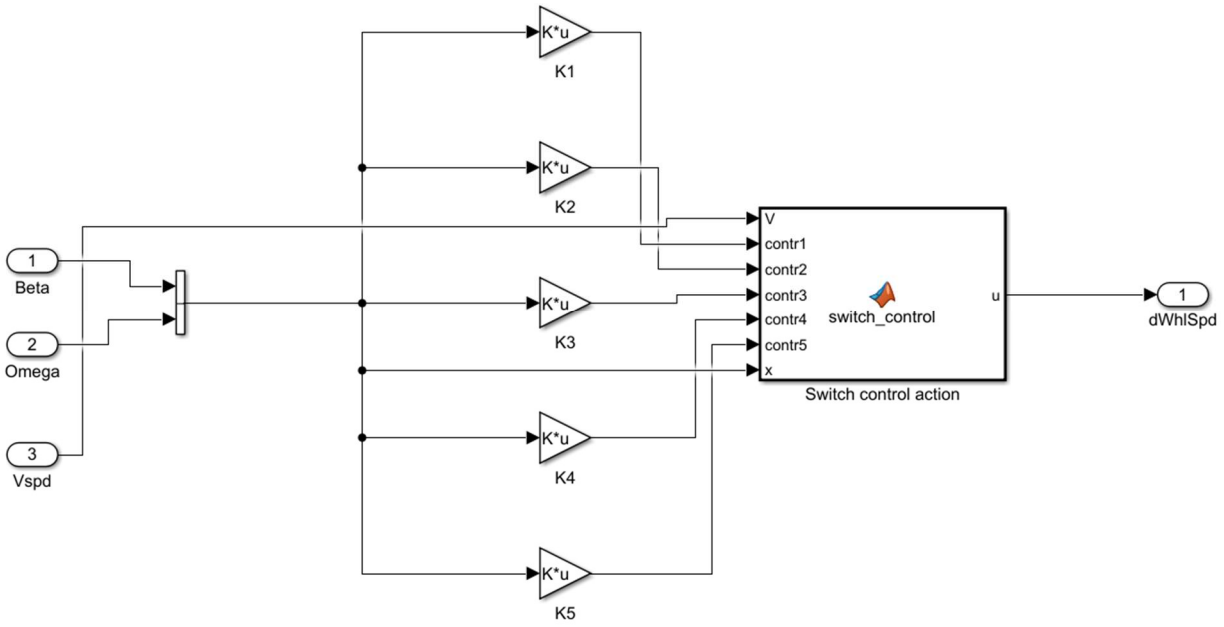
- **dWhlSpd** – 4x1 vector (rad/s)
    - Each element reports the wheel angular speed variation of a wheel of the vehicle. Wheels' speed variations are, in order: front-left, front-right, rear-left, rear-right. Negative values refer to wheels rotating in reverse direction, that is when vehicle is backward moving

The ESP control model requires vehicle parameters:

- **Wheel base** – scalar (m)
    - It represents the distance between front and rear axles of the vehicle.
- **Wheel distance** – scalar (m)
    - It represents the distance between wheel's center of rotation and the center of the front/rear axle, assumed to be equal for all the four wheels.
- **Wheel radius** – scalar (m)
    - It represents the radius of the wheels, assumed to be equal for all the four wheels. It is the distance between wheel's center of rotation and a planar ground neglecting elastic effects of wheel.
- **Vehicle mass** – scalar (kg)
    - It represents the mass of the vehicles.
- **Aerodynamic coefficients** – 3x1 vector (-)
    - It consists of the cross-surface S of the vehicle, and the two drag coefficients C_{x0} and C_{yb}. They are used to compute the aerodynamic force the vehicle is subject to.

6.2.2 How to use

To use the ESP control model, input signals and parameters must be provided as explained above. Here an example of use is shown.



Five-scenario control actions are precomputed and logged onto the system as $Klqr_{1,2,3,4,5}$: these values are calculated at five current-velocity values (50,70,90,110 and 130 km/h). Then, based on the actual current velocity of the vehicle (*Vspd* in the schematic), the *switch_control* block selects the most suitable control action to the situation (among the possible five options it takes as inputs, $contr_{1,2,3,4,5}$). Moreover, the block performs also a check on the two state variables *beta* and *omega*, slip angle and yaw rate of the vehicle: if the two variables are out of range (described above, in the list of input variables of the subsystem), then the ESP control model does nothing; otherwise, it chooses the best control action and gives as output the wheel-speed variations of the four wheels to counteract the slippery conditions of the road.

6.3 Unit testing

In order to have a complete test of the unit, normal behavior and boundary conditions have been checked:

- Behavior of the unit when states are at equilibrium.
  In perfect equilibrium conditions, the control action is negligible and does not provide any useful torque to rotate the vehicle.
- Testing of the control action with small and big tracking errors.
  When the system is not at equilibrium, meaning that either slip angle or yaw rate are not the ones expected during a normal turn, so car is slipping, the control takes sufficient actions to move the vehicle back to a stable condition.
- Analysis of symmetric control action.
  A positive error on one state is compensated with the same control action applied to counteract a negative error of the same entity on the same state, though different in sign.
- Stress test with particular angles.
  The controller supposes a linear system, therefore critical state errors of $\frac{\pi}{2}$ and multiples does not constitute a problem for the controller, which provides a coherent control action.
- Stress test with high errors.
  The controller correctly rejects too high errors on both slip angle and yaw rate states. Slip angle is correctly bounded to $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ range.

After minor adjustments, the behavior of the unit has proven to be the one expected.

7 Car model

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.

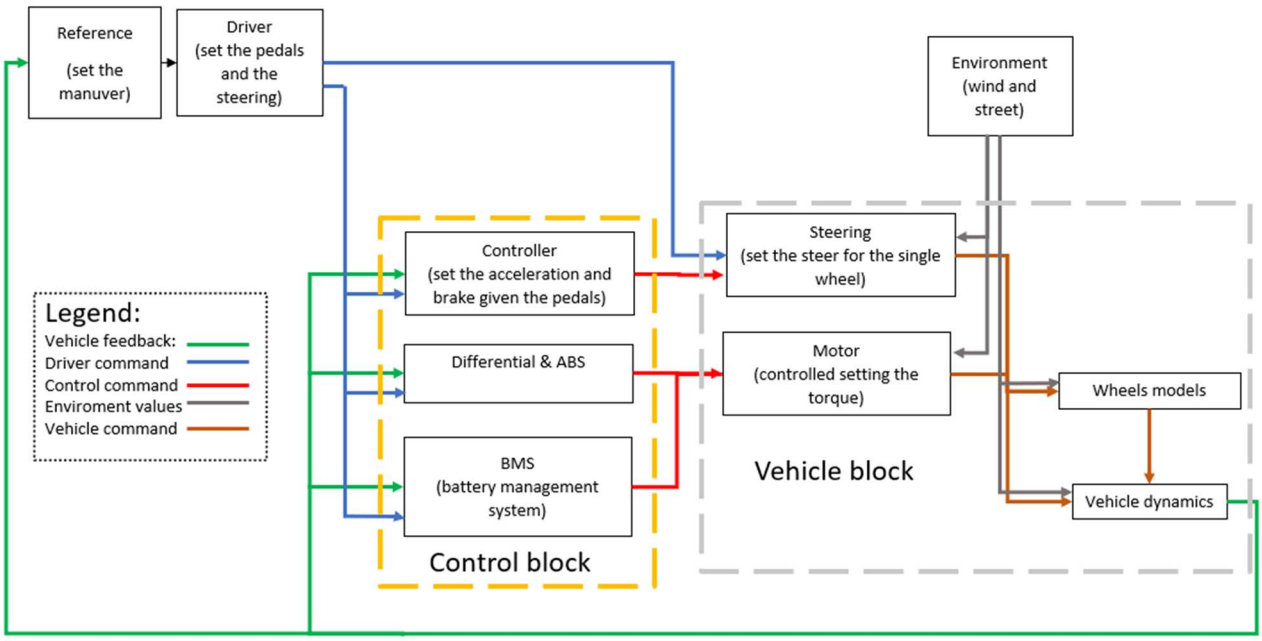
A summary of the design of the model is introduced below. For further details, see the related document.

7.1 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.

7.2 Usage

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.

7.2.1 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.

For a list of all the parameters which may be set, see the related document.

8 System integration

After the development of a virtual differential system and of an ESP control, for an all-wheel drive electric vehicle, it is fundamental to integrate the two, to study the combined effect when on a vehicle. The integrated system also requires integration with the complete vehicle model, devised appositely to support testing and validation of the developed system.

The integration requires special care, as it has to opportunely anticipate the behavior of the system which will eventually be implemented on car, to reduce development and testing costs and fasten up the individuation and correction of errors, also increasing the safety of real-world testing. The units integrated in the system shall be extensively tested, so that errors resulting from the interconnection can be more quickly asserted and solved.

A summary of the integration process is introduced below. For further details, see the related document.

8.1 Input and output signals

The developed units, and the surrounding blocks necessary for a correct integration, require some signals to be retrieved from the vehicle model. On a real car, these signals would either be retrieved from on-board sensors, eventually following some elaboration procedures, or from signals produced by other ECUs.

The required input signals, used for subsystems integration, are:

- **AccelPdl**: percentage of throttle pedal position, set by the driver and provided in the scheme by the Predictive Driver; it is required to determine the wanted reference speed.
- **WhlAng**: absolute rotation around vertical axis of each wheel, provided by the steering column; it is required to determine wheels' speeds in ideal driving conditions.
- **Xdot**: longitudinal (along X axis) speed of the vehicle in vehicle refence frame, typically provided by on-board sensors, provided in the scheme by vehicle feedback loop; it is required to determine ESP control for increased stability.

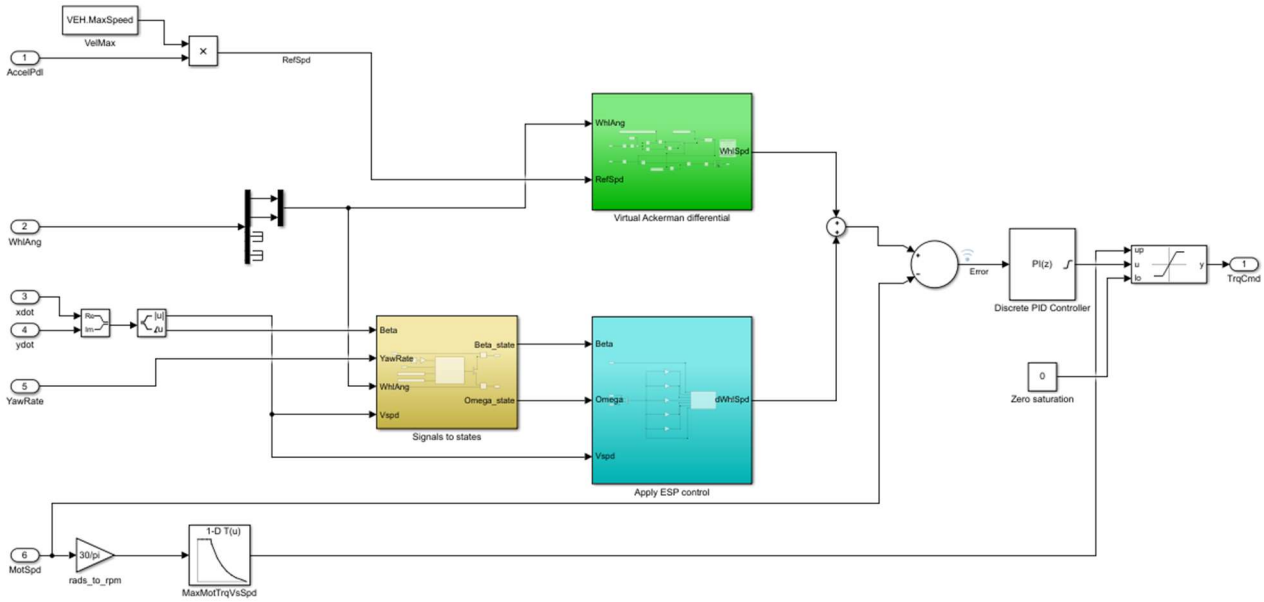
- **Ydot**: lateral (along Y axis) speed of the vehicle in vehicle refence frame, typically provided by on-board sensors, provided in the scheme by vehicle feedback loop; it is required to determine ESP control for increased stability.
- **YawRate**: angular speed of the vehicle along the vertical axis, typically provided by on-board sensors, provided in the scheme by vehicle feedback loop; it is required to determine ESP control for increased stability.
- **MotSpd**: speed of each electric motor, provided by the electric machines; it is required to determine the control action of the on-wheel motors, as well as determining the maximum deliverable torque.

The integrated system provides some output signals:

- **TrqCmd**: torque requested to each electric motor, as an application of the control action; on a real vehicle, it is a command passed over in-vehicle network.

8.2 Integration of subsystems

The scheme of the integrated subsystem is shown below. In figure, green block is the virtual differential, orange block is the translation of vehicle's status to state space variables, blue block is ESP controller.



Virtual differential block

Virtual differential block requires front wheels' angles and reference vehicle's speed as input.

- Wheels' angles are directly provided from the steering mechanism, which computes them starting from the steering angle on the steering wheel controlled by driver.
- Reference speed is computed from the throttle pedal position. The throttle pedal opening (in percentage) is multiplied by maximum vehicle speed to provide the reference speed as a fraction of the maximum vehicle speed. Therefore, throttle pedal linearly relates to the desired vehicle speed.

Virtual differential block provides the required wheels' speeds in ideal driving conditions.

- The wheels' speeds are computed according to ideal Ackerman steering geometry model and assumes ideal driving conditions. They are used to determine the torques required to on-wheels motors.

Signals to states block

The signals-to-states block determines the vehicle's states as required by the ESP control system. Indeed, ESP control system applies optimal control on delta variables, computed as the difference between vehicle's state variables and states at equilibrium condition.

The block requires some inputs for the correct computation of the states.

- Beta, current side-slip angle, angle between vehicle's speed vector and vehicle's longitudinal axis. It is provided by on-vehicle sensors.
- Yaw rate, current angular speed of the vehicle with respect to vertical axis passing through vehicle's center of gravity. It is provided by on-vehicle sensors.
- Wheels' angles are directly provided from the steering mechanism, which computes them starting from the steering angle on the steering wheel controlled by driver. They are required for the computation of the equilibrium condition.
- Current vehicle speed is directly provided by sensors. It is required for the computation of the equilibrium condition.

The block returns delta states as required by the ESP control system.

- Delta beta, computed as the difference between side-slip angle, from sensors, and the same angle at equilibrium according to current driving conditions.
- Delta yaw rate, computed as the difference between yaw rate, from sensors, and the same angular speed at equilibrium according to current driving conditions.

ESP control block

The ESP control block applies an optimal control strategy, based on the current state of the vehicle, to increase global stability of the car. This system helps in avoiding skidding when cornering or due to bad road conditions (i.e. ice on road).

The ESP control block requires the knowledge of the state of the vehicle.

- Beta is current side-slip angle, compared to current equilibrium condition. It is a state for the ESP control; therefore, it is multiplied by a gain to compute the control action.
- Omega is current yaw rate, compared to current equilibrium condition. It is a state for the ESP control; therefore, it is multiplied by a gain to compute the control action.
- Current vehicle speed is required to choose the proper gain to apply optimal control. It thus determines the gain matrix that will be multiplied by state vector.

The ESP control block provides the required wheels' speeds difference due to non-ideal driving condition.

- Delta wheels' speeds are determined, applying optimal control, as the angular speeds which should be added/subtracted to current angular speeds to compensate move a skidding vehicle back to a stable driving condition. They are used to determine the torques required to on-wheels motors.

Wheels' torques controller

The on-wheels motors are torque-controlled. An outer control system is necessary to provide the control torque based on required wheels' speeds.

A feedback control system is used to enhance the stability. A discrete PID is used to compute the requested torque.

The input for the controller is the difference between required and current wheels' speeds (input vector with four elements). The output from the controller is the required torque to electric motors (output vector with four elements).

The design requirements for the controller are:

- Fast reaction to variations, to quickly actuate controls required by ESP for increased stability and safety;
- Small admissible overshoot and low ringing, to avoid unwanted accelerations/decelerations of the vehicle;
- Saturation on the torque according to maximum torque deliverable by the motors;
- Only positive torque, as braking torque is eventually handled by a regeneration system when braking;
- No requirements on settling time, as control system is ultimately in feedback with human driver who can adjust speed as wished.

For the above reasons, the PID has been designed having:

- High proportional gain, to fast react to variations in requested speed;
- Low integral gain, to have zero steady state error while reducing overshoots;
- No derivative gain, to reduce ringing;
- Saturation having maximum motor's torque as upper limit, with anti-windup method;
- Saturation having zero torque as lower limit, with anti-windup method;
- Dynamic saturation to saturate torque according to current maximum torque delivered by the electric motor, with no anti-windup method since integral gain is small.

8.3 Testing of the integrated system

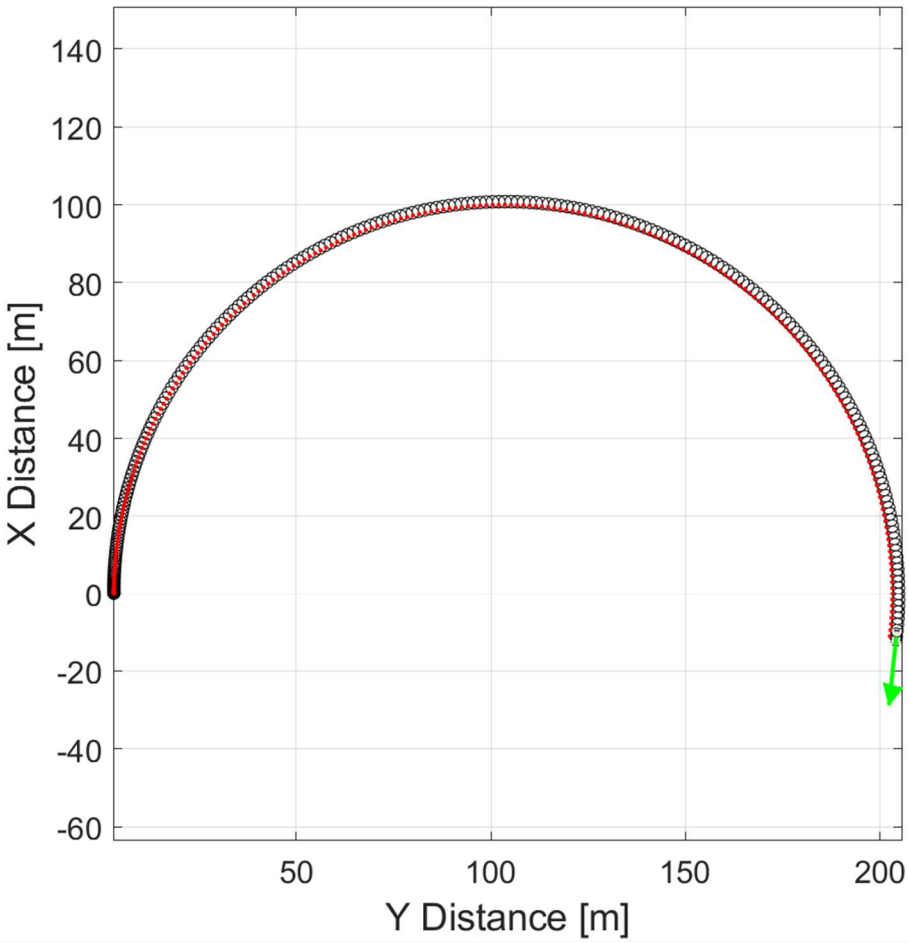
Extensive testing has been performed to both asses the intended functionality of the subsystem and validate the correct integration of the subsystem with the car model.

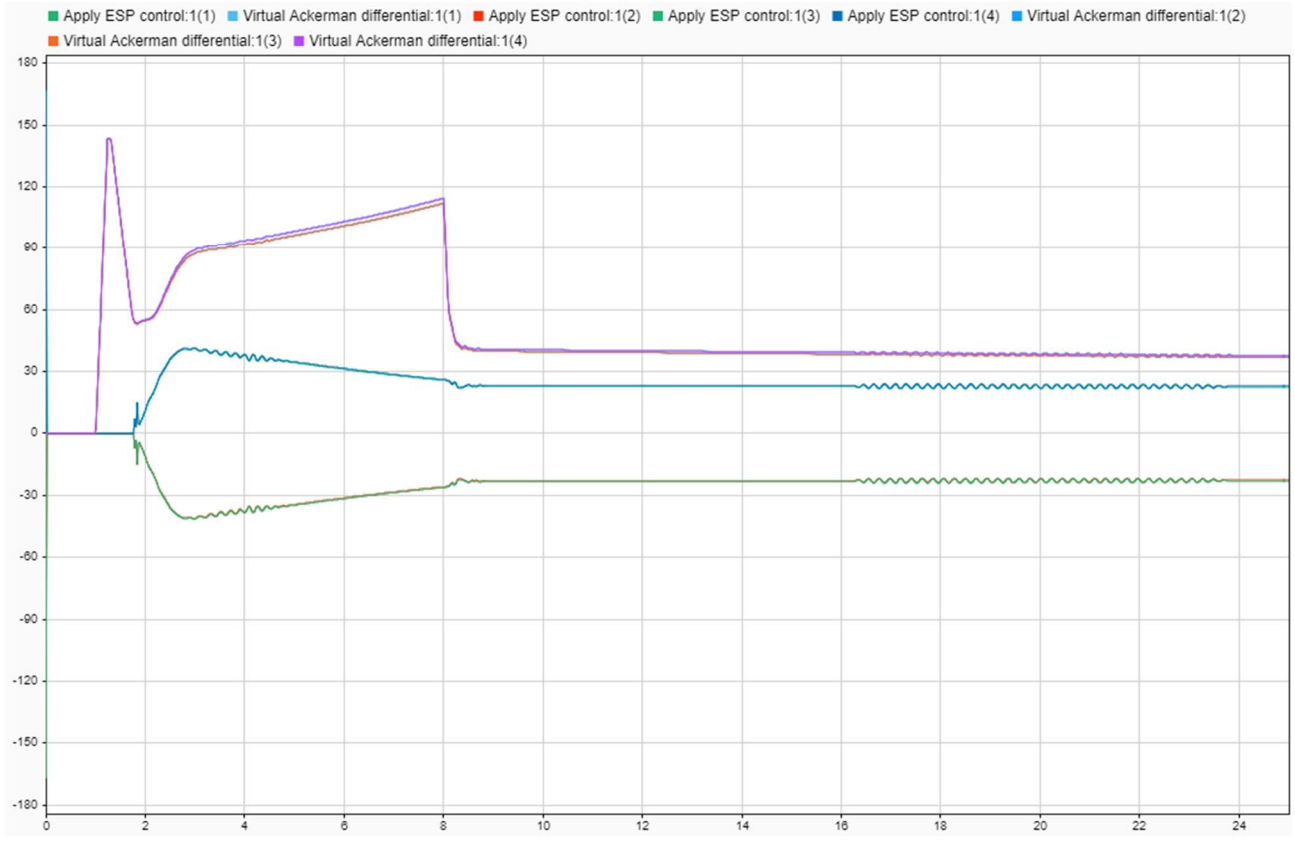
In order to test the integrated subsystem under various working conditions, several simulative scenarios for the car model have been defined, some of which are introduced below. To validate the integrated system, the behavior of the developed car model, implementing the designed controller subsystem is compared to the original vehicle model as provided by Mathworks, implementing a mechanical differential and no ESP controller.

In the below images, when car position is shown, black circles indicate car position in time while red dots determine the reference trajectory, with abscissa and ordinate being the (X, Y) coordinates in inertial reference frame, (0, 0) being the position of the vehicle at the beginning of the simulation. On the other hand, when signals in time are shown, the magnitude of the signal against the time is reported. Some signals may be not visible as completely overlapped by other signals; units of measure comply with SI units and derived (e.g. m for positions, s for time, m/s for speed, rad/s for angular velocities, Nm for torques). Finally, when the behavior of the developed car model is compared with the behavior of the original car model, the latter is reported before the former. It is worth mentioning that the behavior of the car model is strongly influenced by the driver, set to be a predictive driver (acting to follow reference trajectory) during all the tests.

The testing scenarios and results are reported:

- Test of the integrated subsystem while turning.
  The first series of tests had the aim of analyzing the behavior of the implemented controller subsystem during normal driving conditions, with the goal of verifying the correct integration as well as the functioning of the controller block. The results show, as expected, the ability of the system to accelerate the vehicle as requested by the driver and support the car while maneuvering. Movements of the vehicle during a 25 s time span and output signals of virtual differential (*Virtual Ackerman differential* signals) and ESP controller (*Apply ESP control* signals) subsystems are shown.

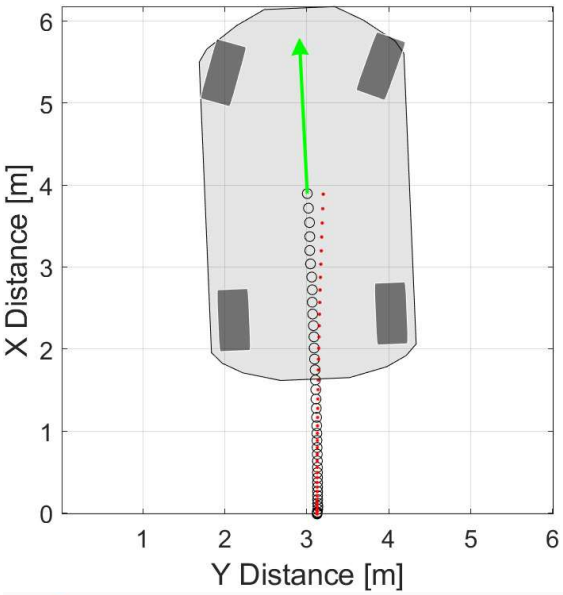


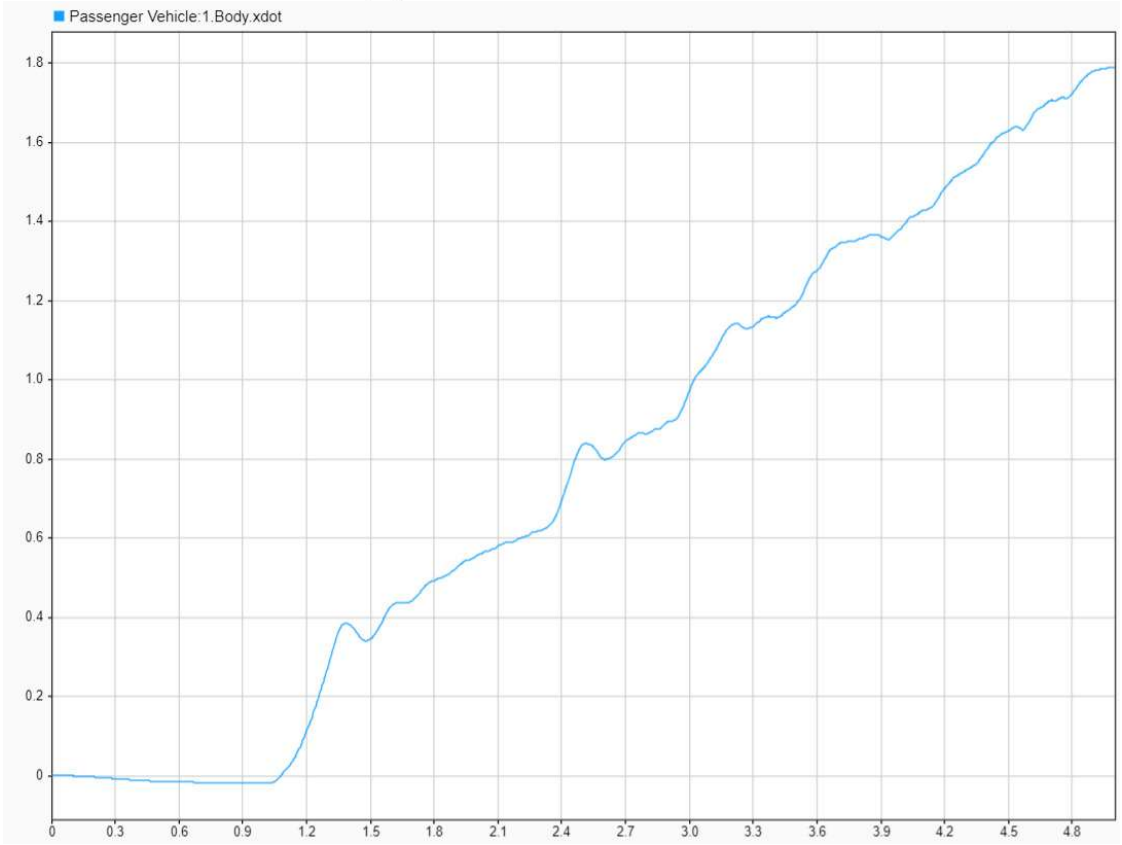


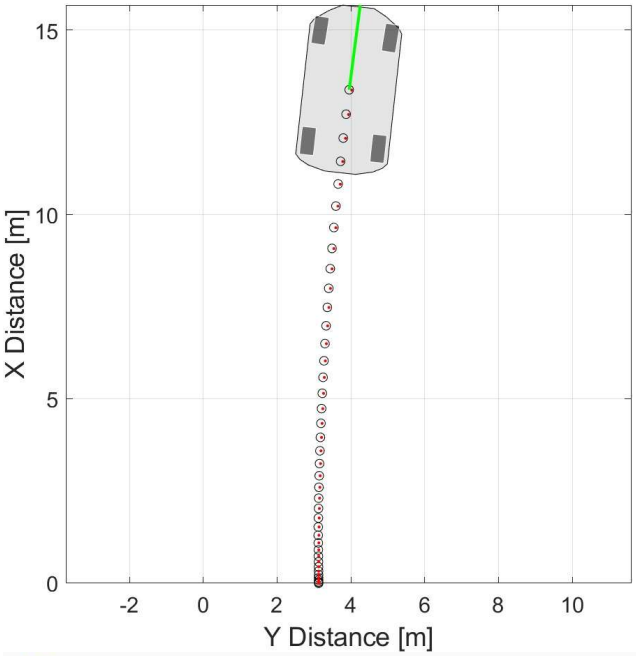
- Ability of the controller subsystem to compensate for slippery road conditions.

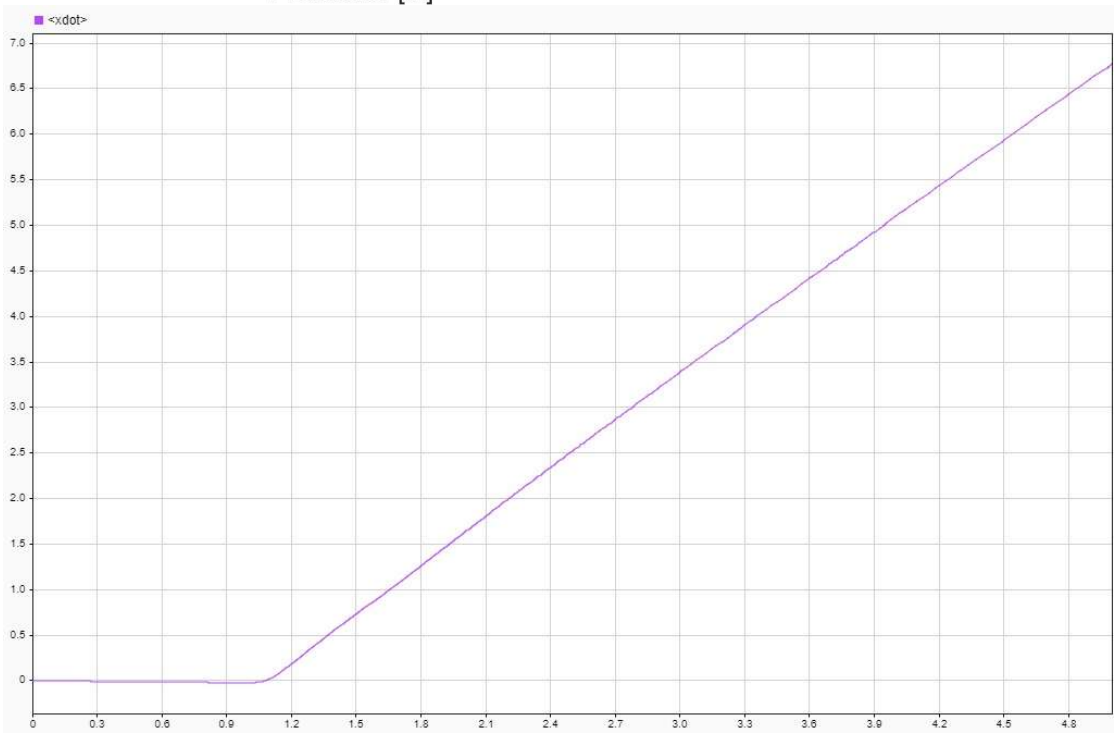
  One advantage of the virtual differential subsystem is its ability to distribute the torque

on the wheels regardless of the conditions of the road, enforcing the working principle of a typical limited-slip differential system. The reported images show the movement and the speed (*xdot*) of the vehicle in a 5 s time span when two out of four wheels are on icy road, for both the original vehicle with a classical differential and the developed car model implementing virtual differential and ESP control. The developed system allows the vehicle to more easily move forward and accelerate in case of ice on the road.





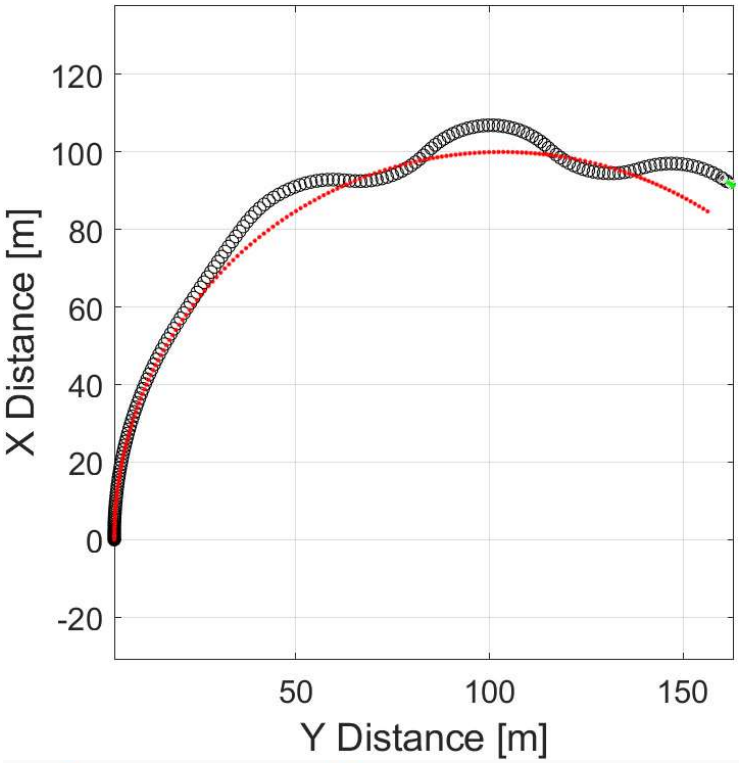


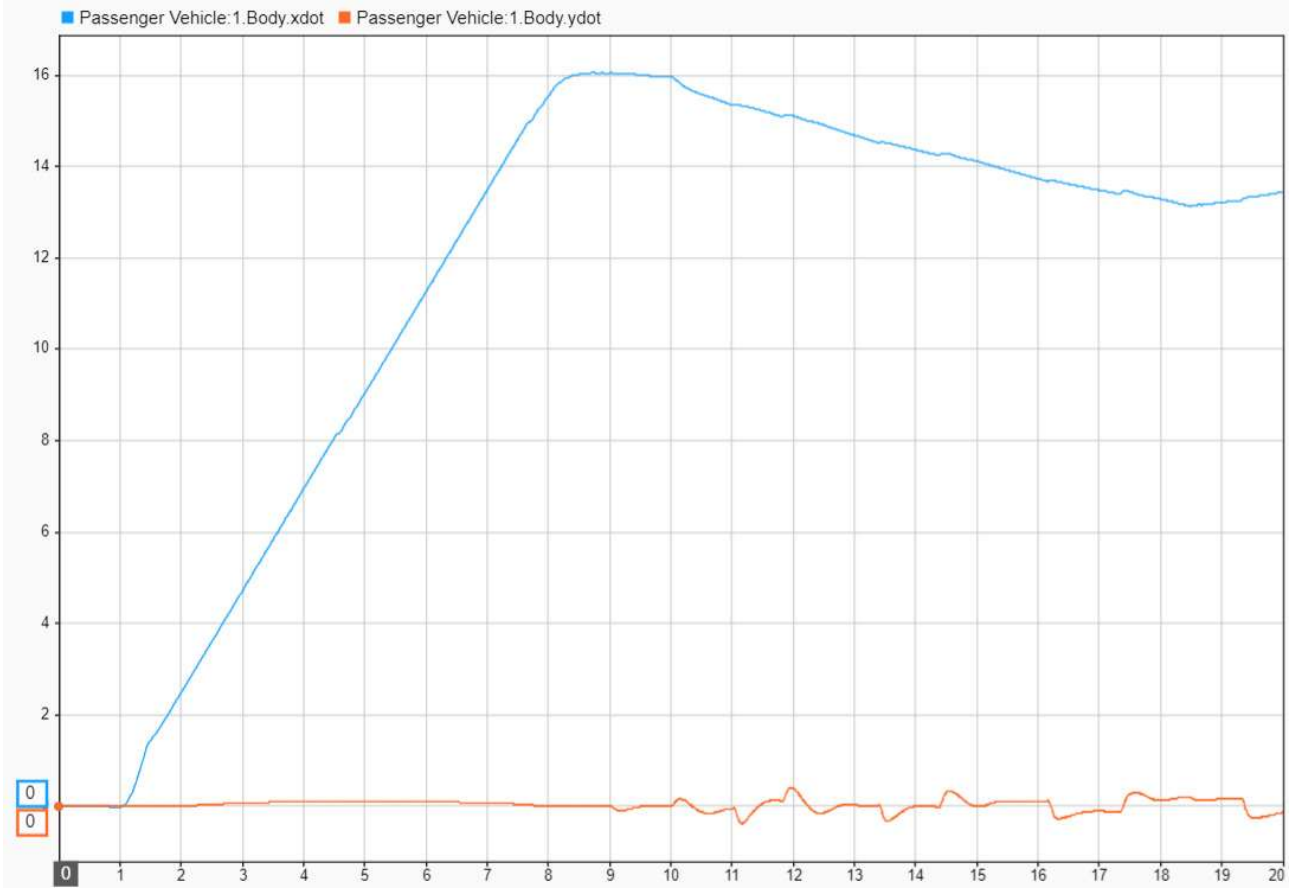


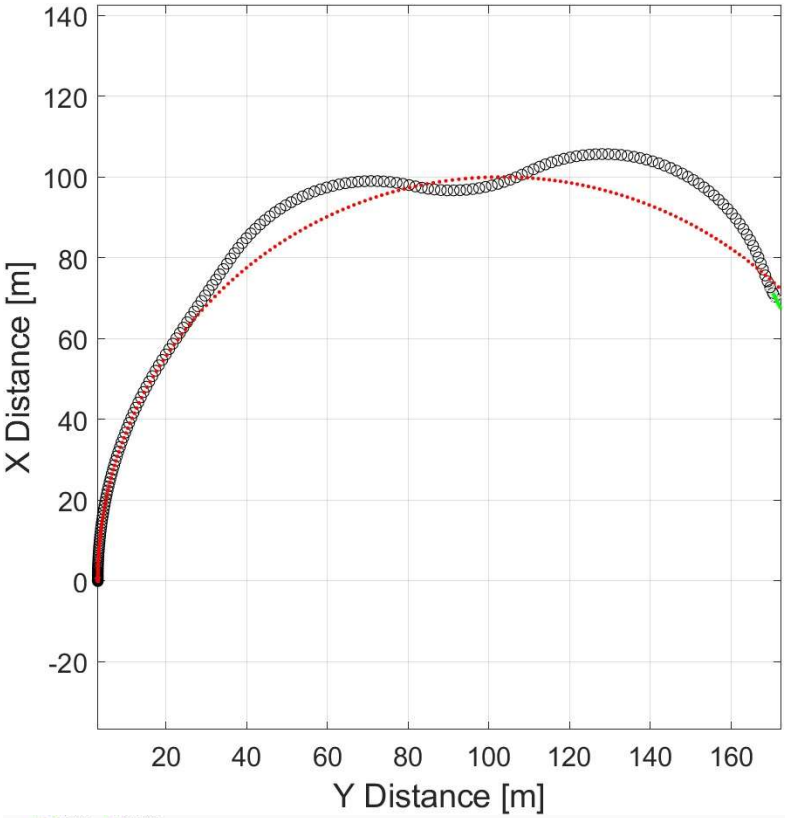
- Ability of the ESP controller to stabilize the vehicle when ice is found on road.

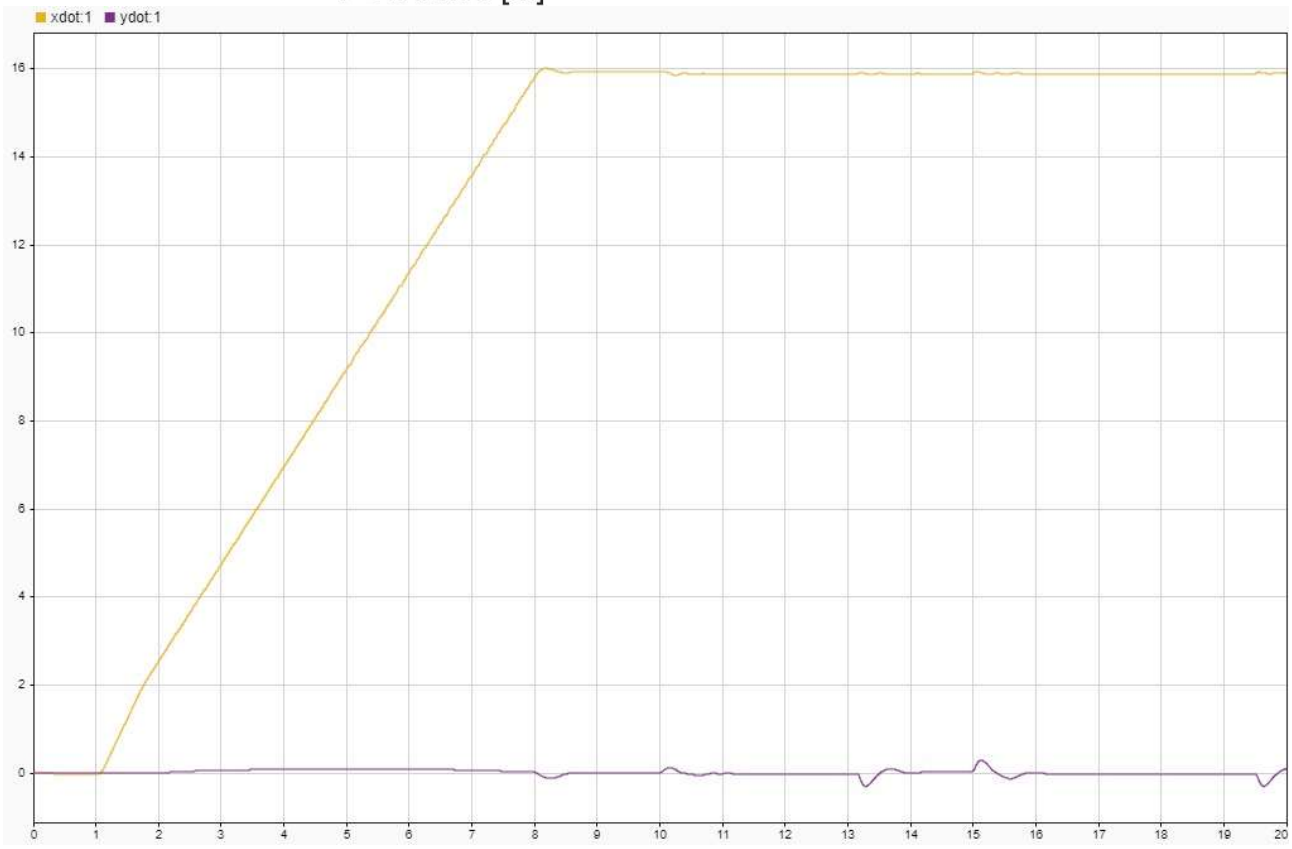
  When ice is present when turning on a road, the vehicle tends to travel out of its original reference trajectory. The aim of the ESP control, in these cases, is to stabilize the vehicle, to avoid it starting to spin causing the driver to lose control. This action usually enlarges the turning radius of the car, but it allows the driver to re-gain control of the vehicle. The reported images show the movement of the vehicle for the original and the developed car model. In the former case, turning radius is reduced but oscillations are more frequent, causing a greater difficulty, for the driver, to re-gain control of the vehicle. The action of the ESP control also reduces lateral speed (*ydot*),

and thus skidding, of the vehicle. It is worth mentioning that the nervous control by the predictive driver, forced to follow the reference trajectory, increase the instability of the system in both cases.





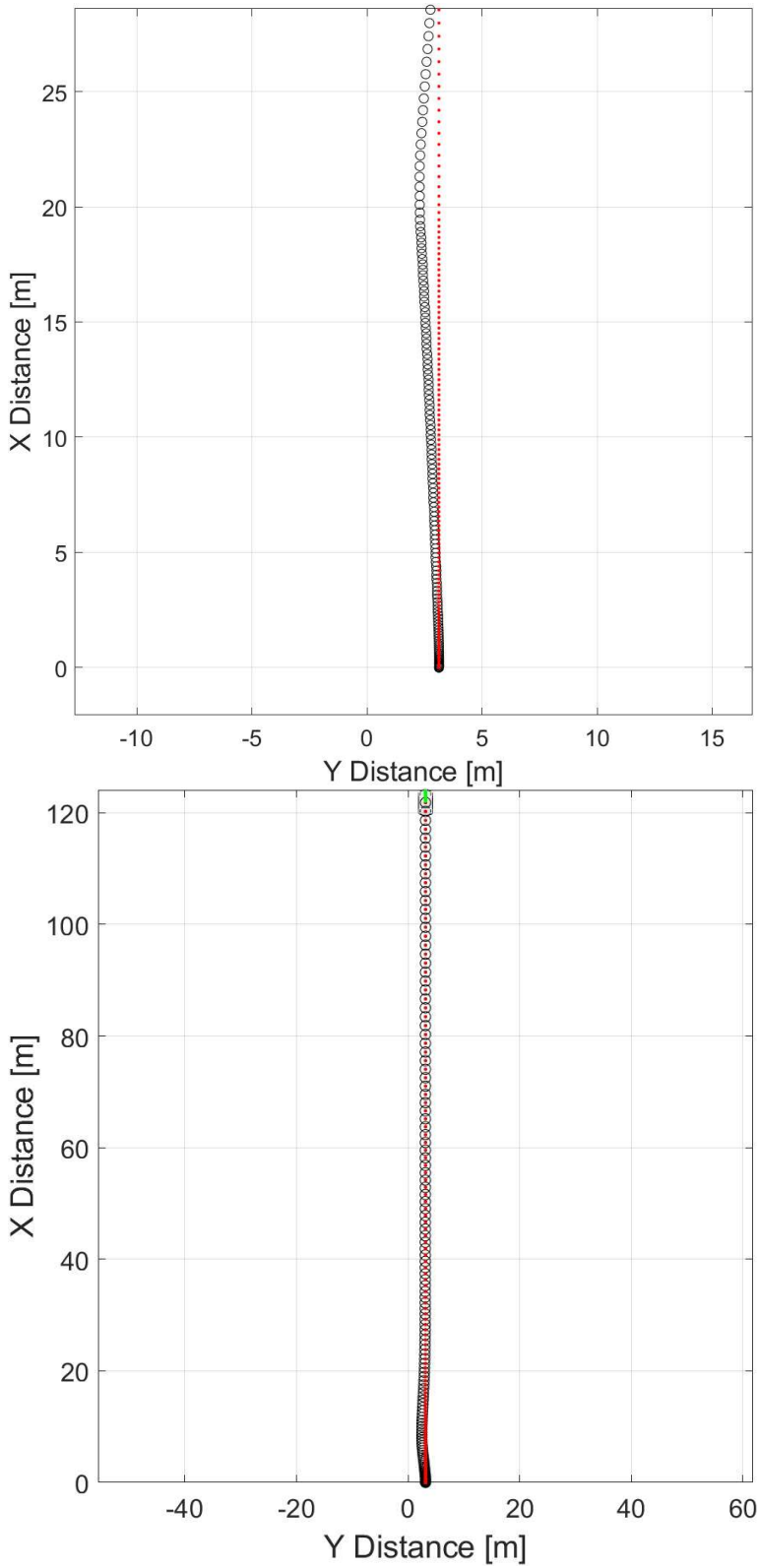




- Straight trajectory with ice on road.

  The combined action of the virtual differential subsystem and the ESP control allows the vehicle to both accelerate on icy roads and maintain a straight trajectory if required. The reported images show position (X signal) of the vehicle when ice is

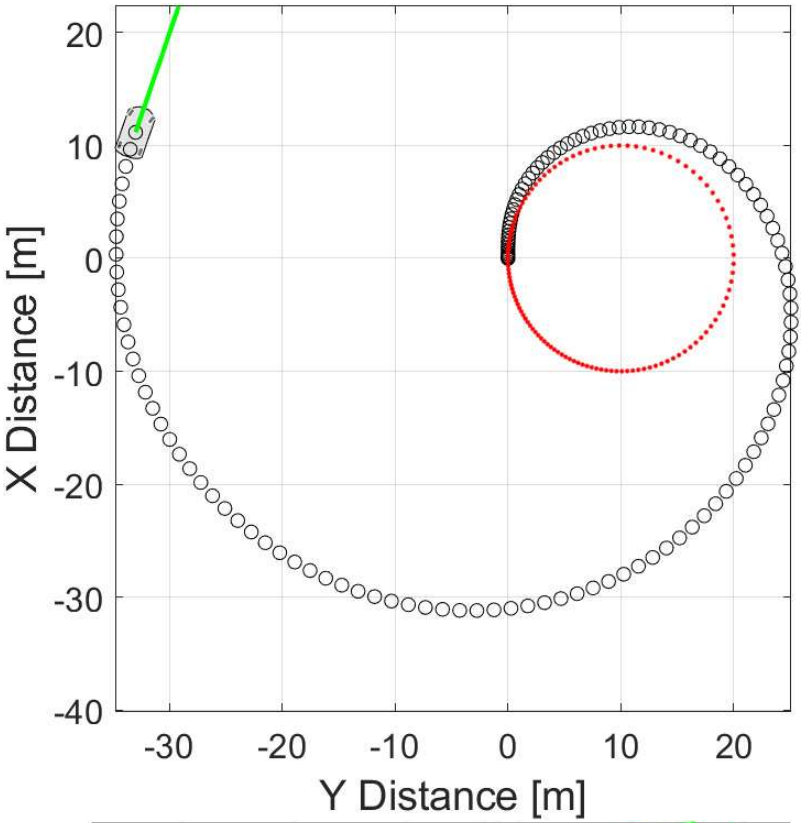
present on half of the road, meaning two skidding wheels and two wheels with high friction coefficient; behaviors for both original and developed car model are shown.

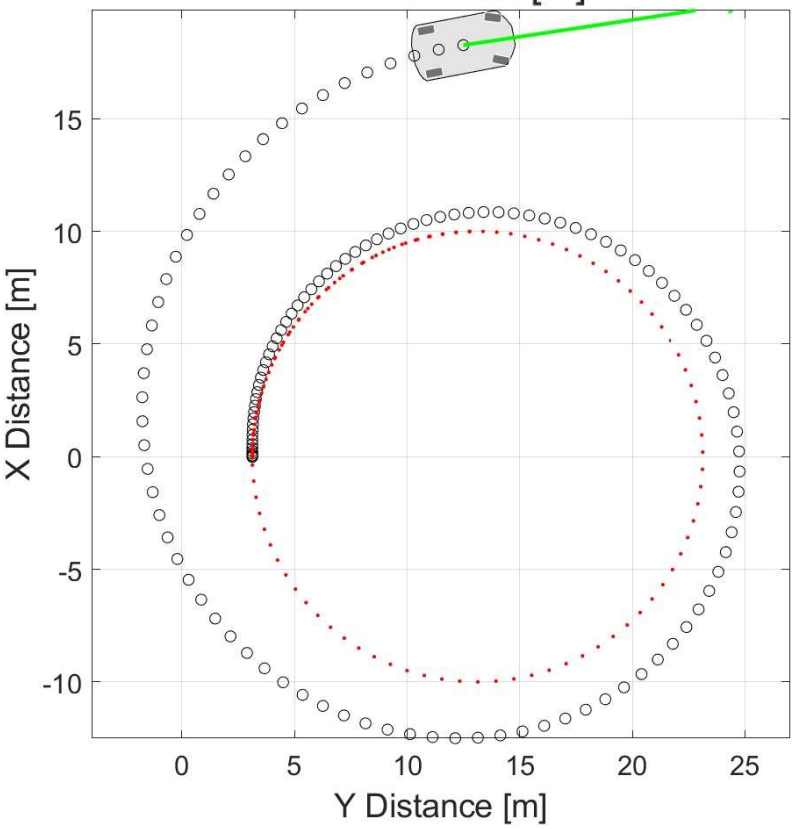


- Cornering with small turn radius.

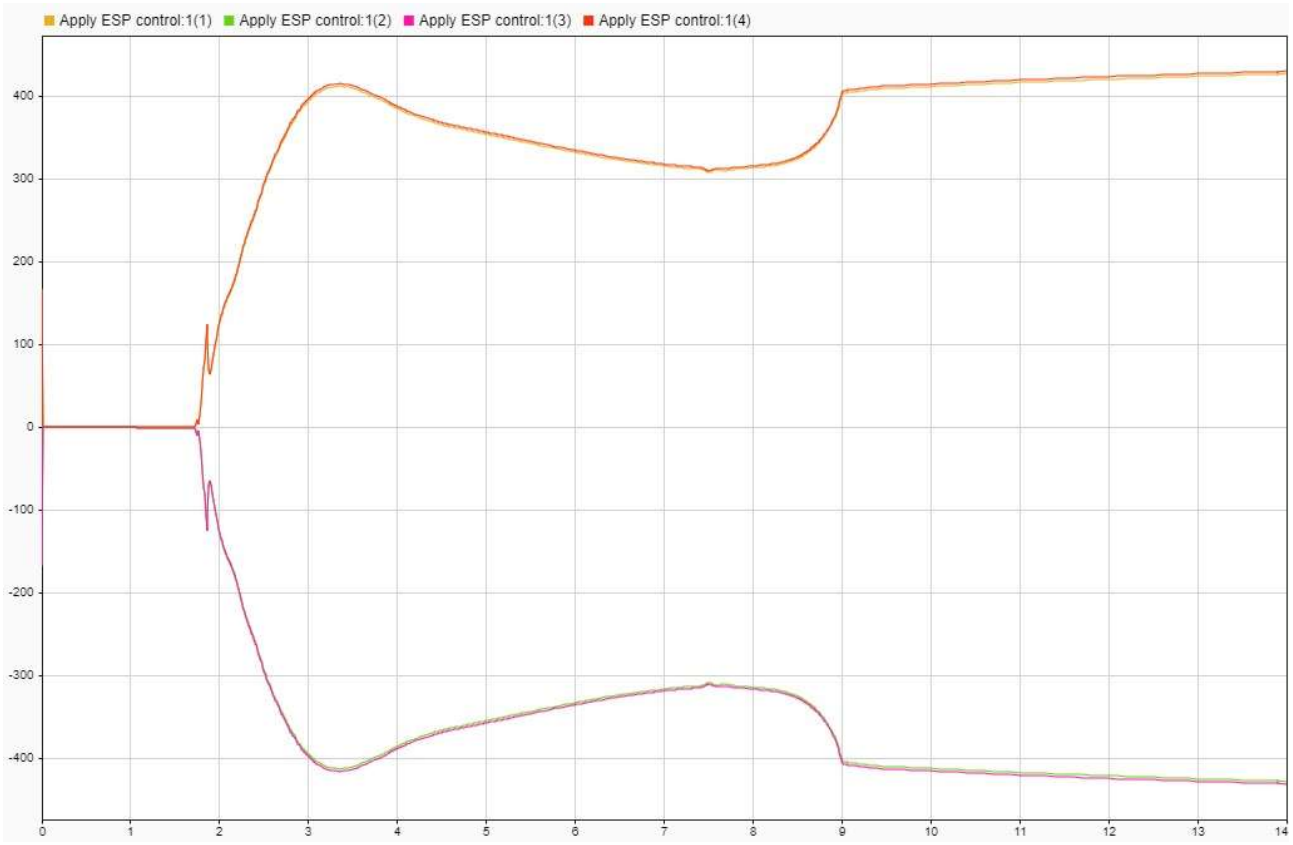
  Another proven advantage of the developed controller system is its aid provided when

cornering. The combined action of the virtual differential system and the ESP controller allows the driver to follow unconventional trajectories having a too small turn radius for the current vehicle speed. The reported images show the vehicle position in for both original and developed car models.





The action of the ESP controller is also shown.



9 System validation

After having obtained an integrated system of the four-wheel drive electric vehicle, system tests can be performed, according to the validation branch of the V-model.

First of all, software-in-the-loop (SIL) is performed, during which the controller is simulated in Matlab environment on host computer. Next, processor-in-the-loop (PIL) validation is accomplished, with the controller being implemented on a physical board, chosen to be the STM32F4-Discovery. In order to perform this latter test, the equivalent C code of the controller developed in Matlab has been automatically generated (specifically for the hardware at our disposal in the PIL simulation) through the Matlab Embedded Code Generator tool, which also provides a documentation for the generated C code.

A summary of the validation process is introduced below. For further details and for a description of the steps for validation, see the related document.

9.1 Software-In-the-Loop (SIL) test

SIL test of the controller allowed a subsystem validation, to ensure the output of software-in-the-loop (SIL) code matched that of the model subsystem. In order to perform it, a SIL verification harness has required to be created, then simulation results have been collected, and the results have been compared using the simulation data inspector. The procedure applied for processor-in-the-loop (PIL) validation was quite similar, with the only difference being the necessary setting of the hardware parameters for the simulation, as detailed in the following.

With SIL simulation, the behavior of production source code on host computer is validated. Embedded Coder tool allowed to create a test harness in SIL or PIL mode for model validation. The SIL or PIL block results can be compared with the model results, and metrics, including execution time and code coverage, can be collected.

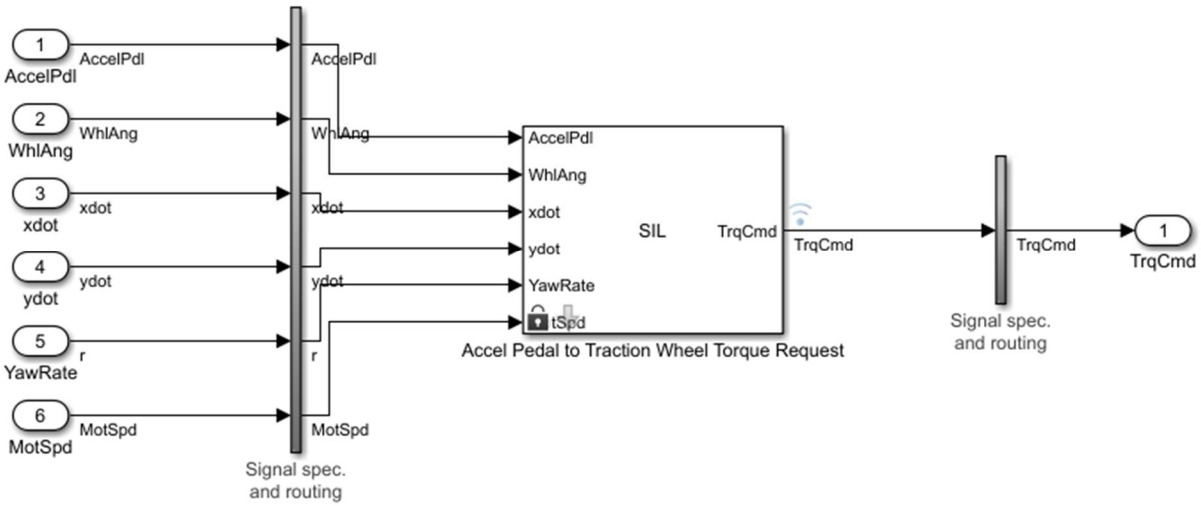
Using the test harness to perform SIL and PIL verification, it is possible to:

- Manage the harness to interface with the car model. Generating the test harness also generates the SIL block. The test harness is associated with the component under validation;
- Use built-in tools for these test-design-test workflows: checking the SIL or PIL block equivalence, updating the SIL or PIL block to the latest model design;
- View and compare logged data and signals using the Test Manager and Simulation Data Inspector.

9.1.1 Creation of a SIL Verification Harness for the Controller

Software-in-the-loop test harness can be created, aided by Matlab environment which provides means to easily create it. For a step-by-step procedure on the creation of the SIL test harness, see the related document.

Resulting test harness shows a SIL block for SIL validation.



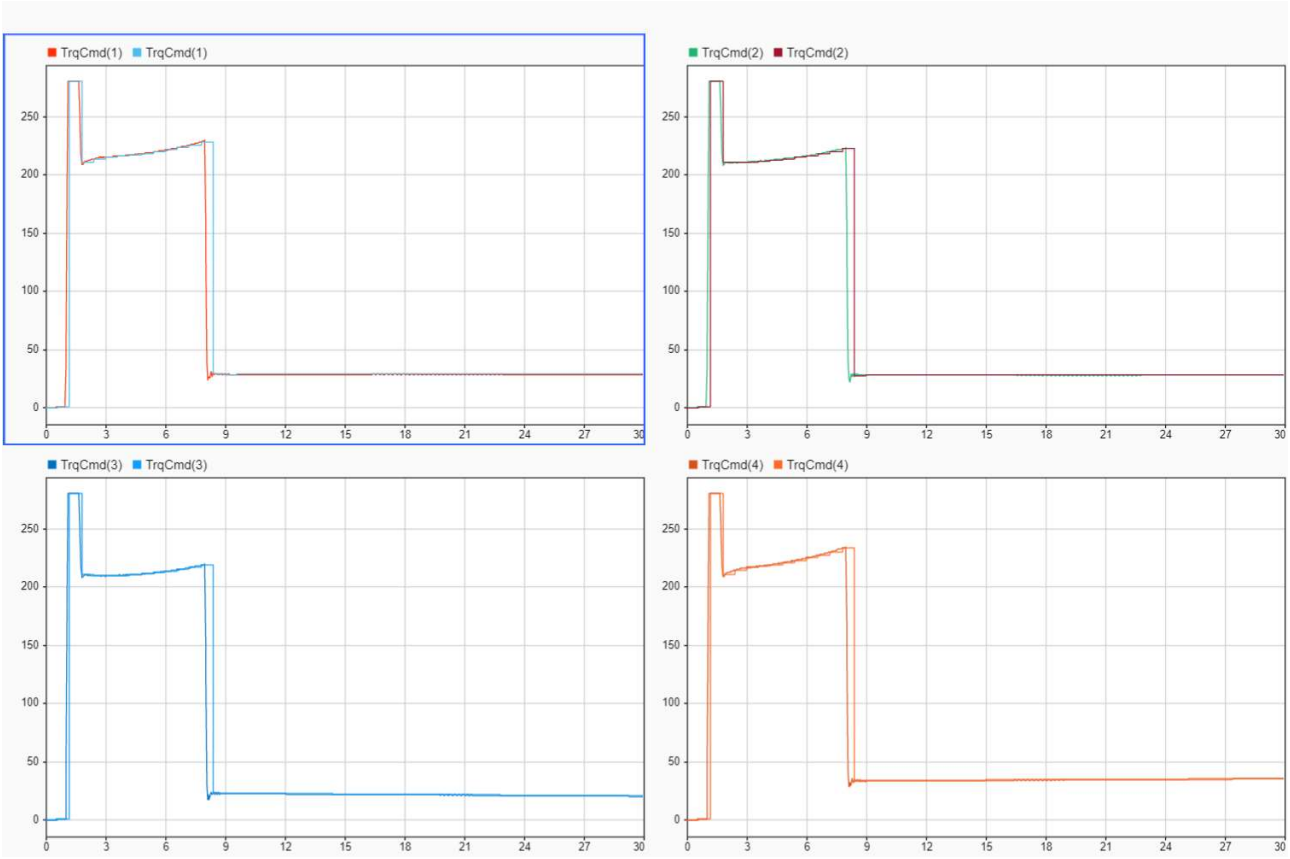
9.1.2 Comparison between SIL Block and Model Controller outputs

`SIL_block_out`, output of the SIL block, and `TrqCmd`, output of the controller model, are the signals selected to be shown and compared on the data inspector window. Signal differences can also be plotted to perform a more detailed analysis for validation.

From the graphs, is can be noticed how the two signals are almost the same. The only difference consists in a SIL output signal being the sampled version of the controller model

output, with a zero-order causal hold of 0.6 s, introducing some delays and imperfections in the output of the SIL block.

The simulation and the SIL block signals, for each of the four wheels, are reported below.



9.2 Processor-In-the-Loop (PIL) test

During the PIL simulation, the generated code of the controller runs on the STM32F4-Discovery board, while the vehicle model still runs on the host computer. The results of the PIL validation are available on Matlab/Simulink environment to verify the numerical equivalence of the results obtained with a simulated controller and a controller implemented by code on target board. The PIL validation process is a crucial part of the development cycle since it ensures that the behavior of the deployed code matches the designed model.

Embedded Coder Support Package for STMicroelectronics Discovery boards allows to create and run Simulink models on the STM32F4-Discovery board. The support package includes a

library of Simulink blocks for the configuration and access STMicroelectronics board's peripherals and communication interfaces.

9.2.1 Configuration of the model for automatic load and run of the generated code on STM32F4-Discovery Board

To run the controller block on the STMicroelectronics STM32F4-Discovery board, the model must firstly be configurated. For a step-by-step procedure on the configuration of the test, see the related document.

9.2.2 Choice of a communication interface for PIL simulation

The STM32F4-Discovery board supports two different communication interfaces for PIL validation: ST-LINK and serial. The ST-LINK communication interface does not require any additional cable or hardware besides a USB type A to Mini-B cable used to connect the STM32F4-Discovery board to the host computer. The serial communication interface requires a USB TTL-232 cable. Running a PIL simulation using the serial communication interface is much faster than using ST-LINK. Therefore, Serial (USART) interface has been selected for the PIL validation procedure.

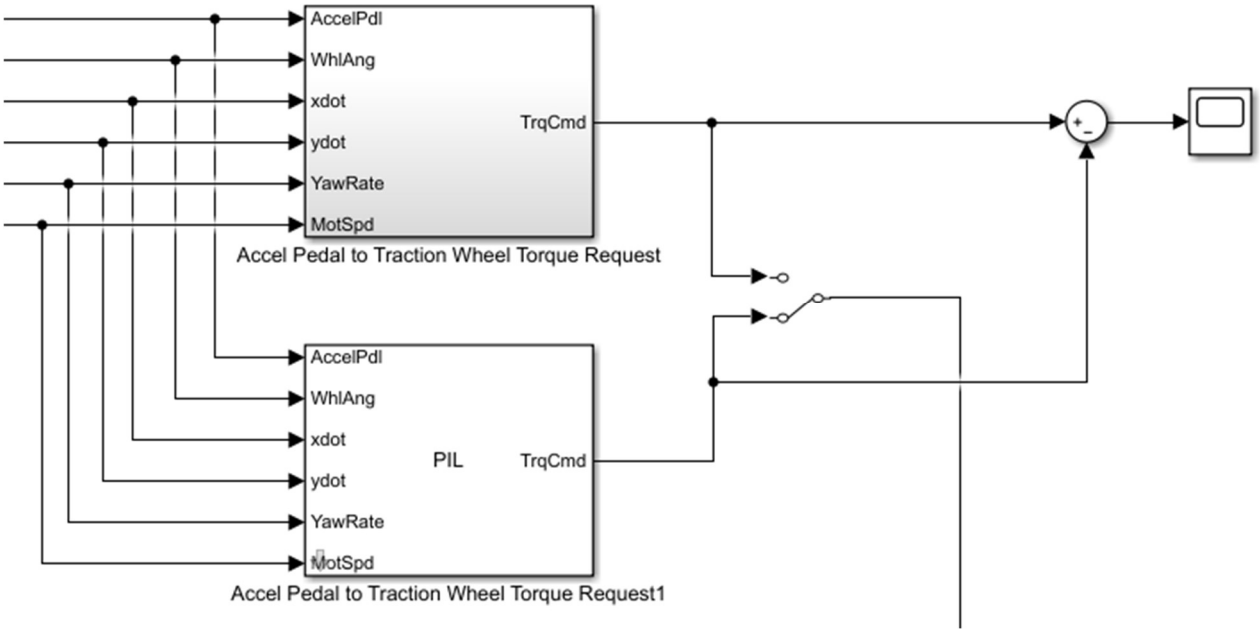
The hardware setup is summarized below:

- Connect ground pin of the USB TTL-232 cable to one of the **GND** pins on the STM32F4-Discovery board;
- Connect the RX pin of the USB TTL-232 cable to **PA2** pin on the STM32F4-Discovery board;
- Connect the TX pin of the USB TTL-232 cable to **PA3** pin on the STM32F4-Discovery board;
- Connect the USB side of the USB TTL-232 cable to host computer;
- Power on the board by connecting a USB type A to Mini-B cable to the STM32F4-Discovery board.

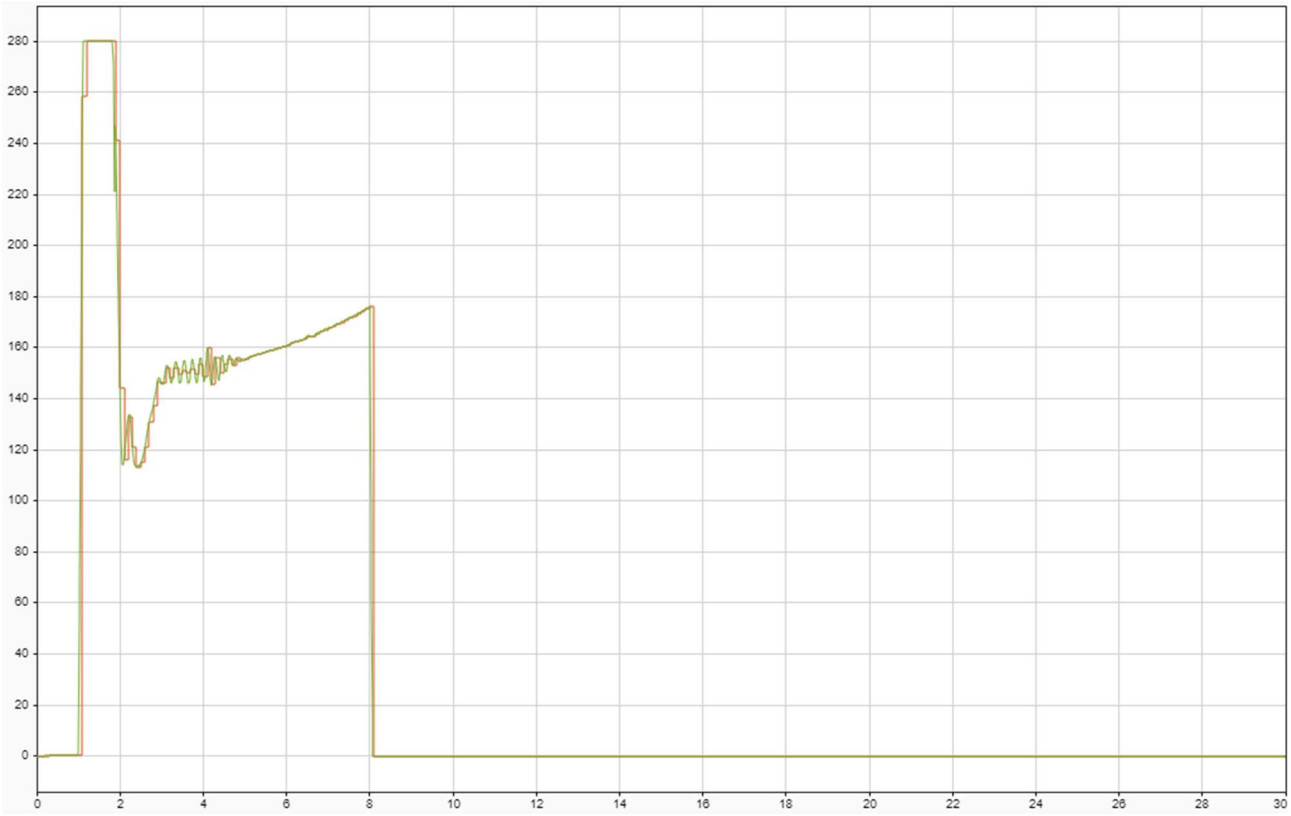
9.2.3 Verification of the Generated Code for the Controller using a PIL block

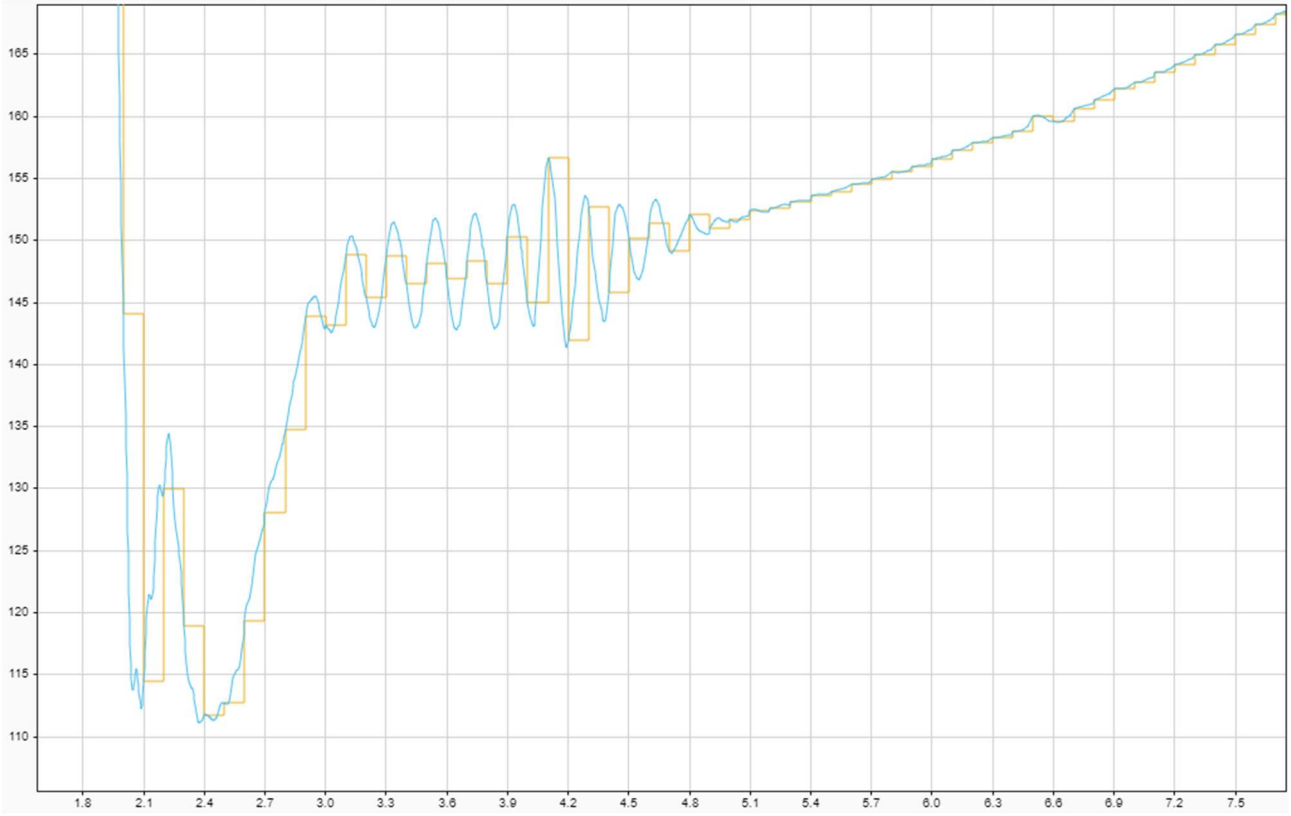
A PIL block for the controller subsystem has been created for the purpose of PIL validation. Matlab/Simulink environment automatically generates a PIL block that can be placed into the car system Simulink model. The simulation can finally be started, with the PIL block allowing the communication with the board on which the controller is running. In case of STM32F4-Discovery, starting the PIL simulation, a new OpenOCD session is launched to download the code onto the external target.

When the simulation has completed, the analysis of the differences between the output signal from simulated controller on host computer and the output from PIL block, coming from the controller running on the board, can be performed. Some additional blocks help in this analysis procedure.



From the comparison between the controller simulated on the host computer and the controller simulated on the target board, a very similar behavior of the output signals is noticed. In particular, the torque command coming from the controller on target board is the sampled version of the simulated one, with a zero-order causal hold of 0.1 s. The pictures below show the comparison between the torque commands requested in time on a single wheel by the two kinds of controllers.



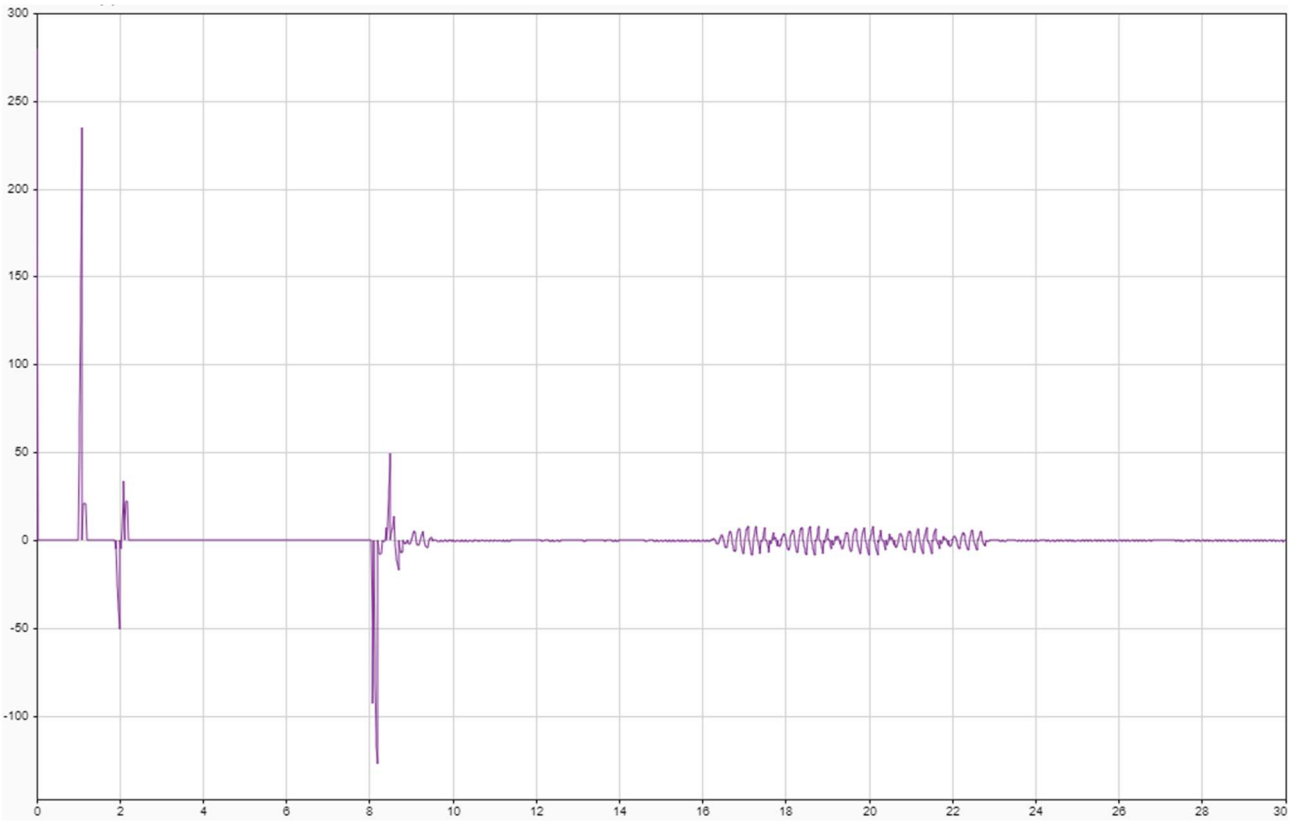


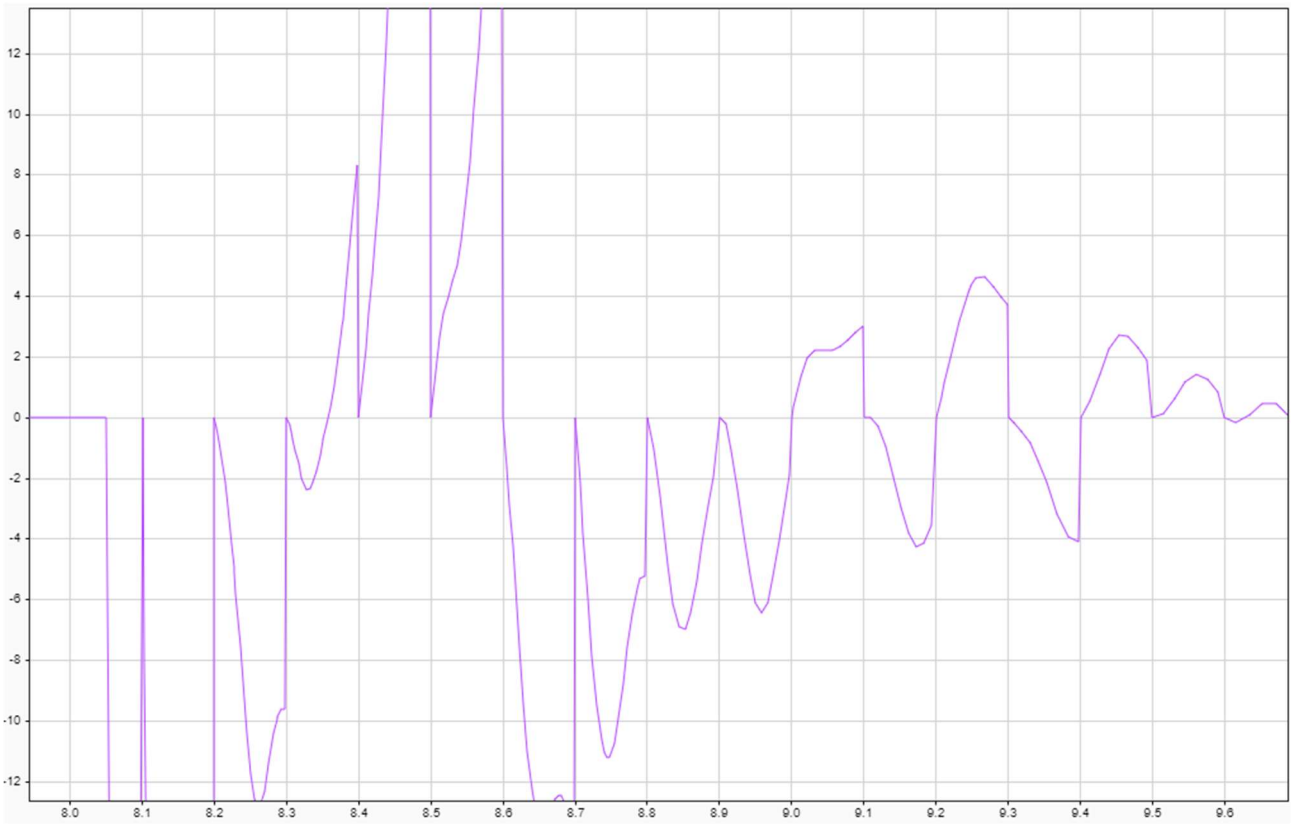
This sampling and holding technique leads to a error difference, between the two output signals, which has large spikes when requested torque changes very quickly, usually due to a nervous driver suddenly pushing or releasing the throttle pedal. These spikes are, however,

limited in time, as they vanish every 0.1 s, when a new sample of the control action is performed.

This error introduces non-negligible delays within the control system, reducing the robustness of the control system itself. It is worth mentioning, nevertheless, that the PIL simulation usually introduces delay due to the communication between target board and host computer. Moreover, although the delays, the system remains stable, suggesting that the final controller implemented on a real target board, for which delays will be lower, will remain stable too.

The images below report the error occurring between the output signals of the controller simulated on the host computer and on the target board.





The vehicle still operates as intended regardless of the reported errors