System Validation

Testing and validation

Author: Federico Galli

Author: Alessandro Luppi

Last revision on 11/07/2020

# Introduction

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.

The objective of this dissertation is to show SIL and PIL simulation results and evaluate them. The results obtained after the SIL validation process are analyzed through the Data Inspector tool of MATLAB, which allows the comparison with the signals obtained by the simulation of the model without any SIL block. The difference signal, between the designed controller running on host computer and the controller implemented as a PIL block in the block scheme and running on target device, is also shown as resulting from the PIL validation process.

This document reports:

* SIL test settings and parameters;
* SIL validation results;
* Embedded Code Generator settings;
* PIL test settings and parameters
* PIL validation results.

## System requirements

In order to perform the above-mentioned tests and simulations, the following software and hardware components are required:

* MATLAB R2019b and Simulink;
* Embedded Coder Support Package for STMicroelectronics Discovery Boards;
* 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).

# 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.

## Creation of a SIL Verification Harness for the Controller

In order to create a SIL test harness for the subsystem, signal logging must be enabled for the model; this can be done entering the following command at the MATLAB command prompt:

set\_param( bdroot,'SignalLogging','on','SignalLoggingName', ...

'SIL\_signals','SignalLoggingSaveFormat','Dataset' )

Then, signals wanted to be logged are selected; for the simulation of the developed controller for the car model, the required signals are the six inputs and the output of the controller: *<Accel>* (renamed *AccelPdl* due to a conflict with another signal of the model), *<WhlAng>*, *<xdot>*, *<ydot>*, *<r>* and *MotSpd* for the inputs, and *TrqCmd* for the output.

After simulation of the model, logged signals can be retrieved from the simulation output and imported into the workspace entering, at the command prompt:

out\_data = out.get('SIL\_signals');

control\_in1 = out\_data.get('AccelPdl');

control\_in2 = out\_data.get('<WhlAng>');

control\_in3 = out\_data.get('<xdot>');

control\_in4 = out\_data.get('<ydot>');

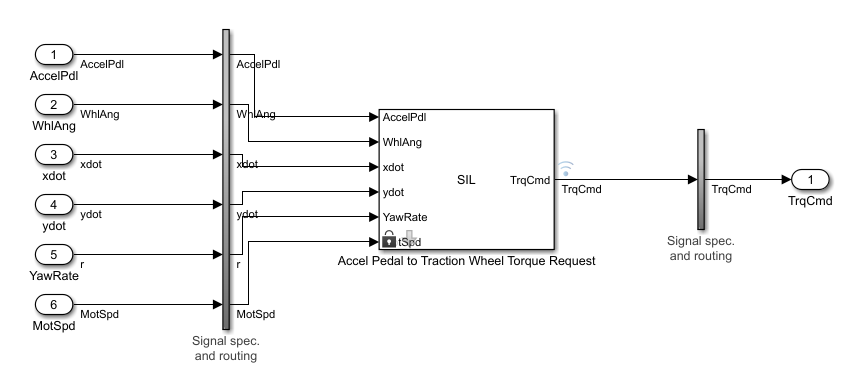
control\_in5 = out\_data.get('<r>');

control\_in6 = out\_data.get('MotSpd');

control\_out1 = out\_data.get('TrqCmd');

Software-in-the-loop test harness can therefore be created, aided by Matlab environment which provides means to easily create it. Test harness properties were set as follows:

* **Name**: SIL\_harness;
* **Sources and Sinks**: Inport and Outport;
* **Advanced Properties – Verification Mode**: Software-in-the-loop (SIL).

Resulting test harness shows a SIL block for SIL validation.

## Configuration and Simulation of SIL Verification Harness

Before starting the simulation of the car model including the SIL block, test harness must be configured to import the logged controller input values, by entering the six inputs in the **Data Import/Export** of the **Configuration Parameters** dialog box. The creation of a SIL test harness as described above created seven Simulink.SimulationData.Signal objects in the Matlab workspace, one per each signal; the Values field of those objects is the only field needed for simulation, in which time and data are logged.

Signal logging for the test harness is thus enabled entering, at the command prompt:

set\_param( 'SIL\_harness','SignalLogging','on','SignalLoggingName', ...

'harness\_signals','SignalLoggingSaveFormat','Dataset' )

The output from the test harness is selected as the only signal to log. Simulation the SIL test harness can then be performed.

## Comparison between SIL Block and Model Controller outputs

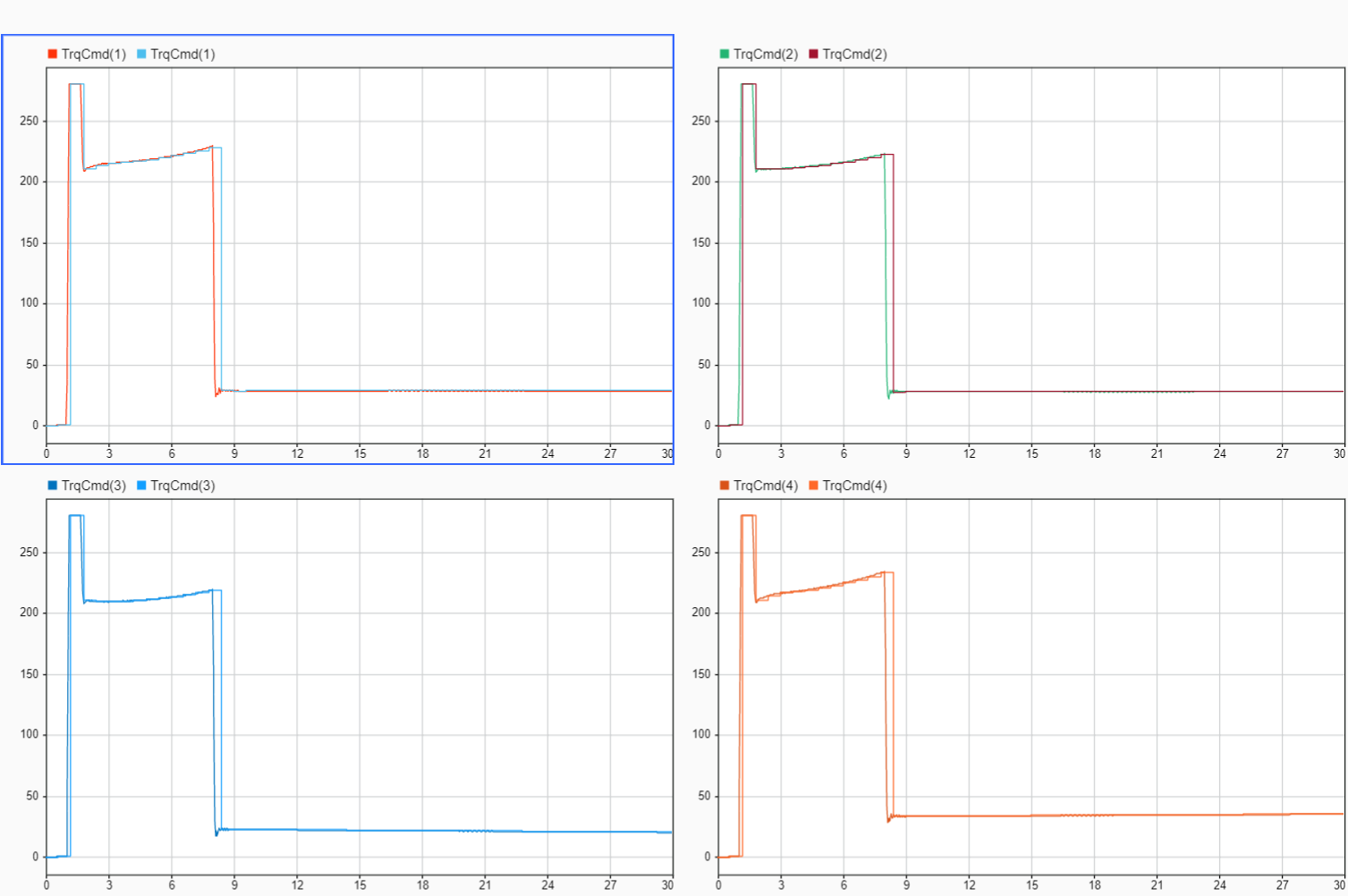
Last step is the comparison between the outputs of the verification harness and the controller subsystem.

In the test harness model, **Simulation Data Inspector** is opened and signals to be compared are imported. The procedure to do so is: click **Import** in the **Import** dialog box; set **Import from** to Base workspace, **Import to** to New Run; under **Data to import**, select **Signal Name** to import data from all sources.

SIL\_block\_out, output of the SIL block, and TrqCmd, output of the controller model, are 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.



# 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.

## 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 the configuration, **STM32F4-Discovery** is chosen as the Hardware board in **Hardware Implementation** > **Hardware board**. Then, in **Hardware board settings** > **Operating system/scheduler** > **Operating system options**, **Operating system** > **Baremetal** should be selected. Two options are available for the field, either **Baremetal** or **CMSIS-RTOS RTX**; both selections provide similar behaviors, but they make use of different scheduling techniques. During PIL test **Baremetal** has been chosen for efficiency and full visibility of the scheduler code, however **CMSIS-RTOS RTX** could be chosen to generate code compatible with legacy code depending on the ‘CMSIS-RTOS RTX’.

Finally, the **Build action** has to be changed to **Build, load and run**. Then the generated hex file can be automatically downloaded onto the connected STM32F4-Discovery board.

## 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 communication method can be set in **Configuration Parameters** > **Hardware Implementation** > **Target Hardware Resources** > **PIL** > **PIL communication interface.**

**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 your board by connecting a USB type A to Mini-B cable to the STM32F4-Discovery board.

As a consequence, a new serial/COM port is available for use on the host computer. The relative COM port number must be added in the Simulink model, under **Target hardware resources** in the Configuration Parameters dialog box.

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

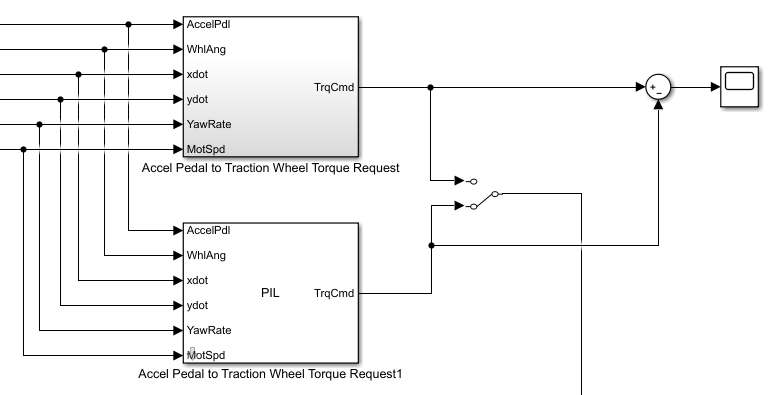
The model has to be configured for the ARMXXX target, in the **Hardware** section of the Configuration Parameters dialog box. Then, under **Code Generation** > **Verification** > **Create block**, **PIL** option must be selected. These settings allow to create a PIL block for the controller subsystem.

Thus, the controller block can be deployed to hardware, hitting **Deploy this subsystem to Hardware** and selecting **Build** option to generate code from the model.

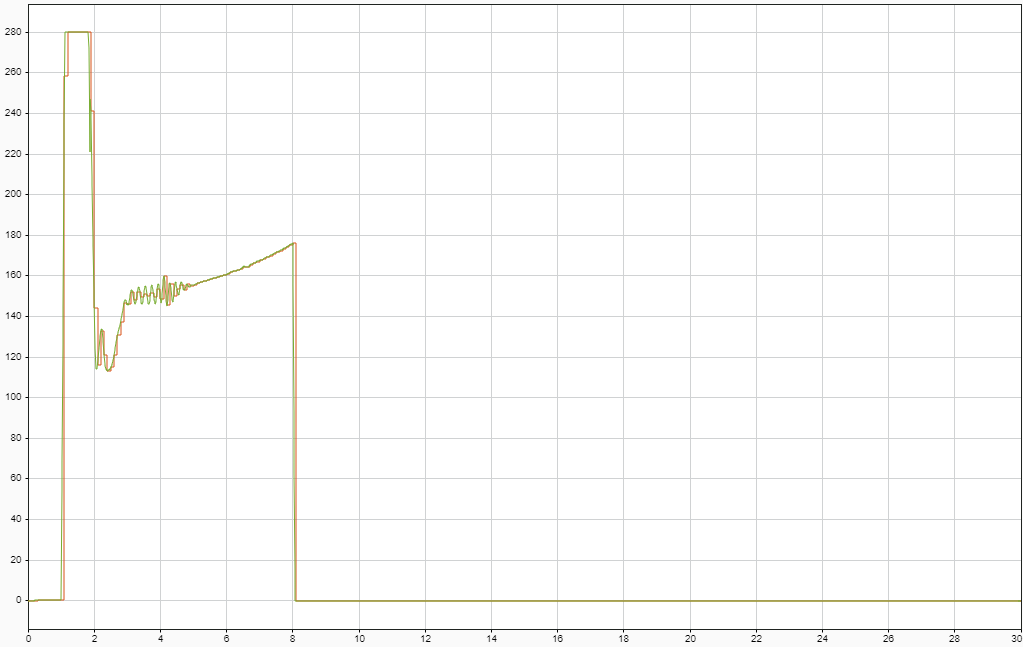
Matlab/Simulink environment automatically generates a PIL block that can be placed into the car system Simulink model. The simulation can finally be started, with the PIL block allowing the communication with the board on which the controller is running.

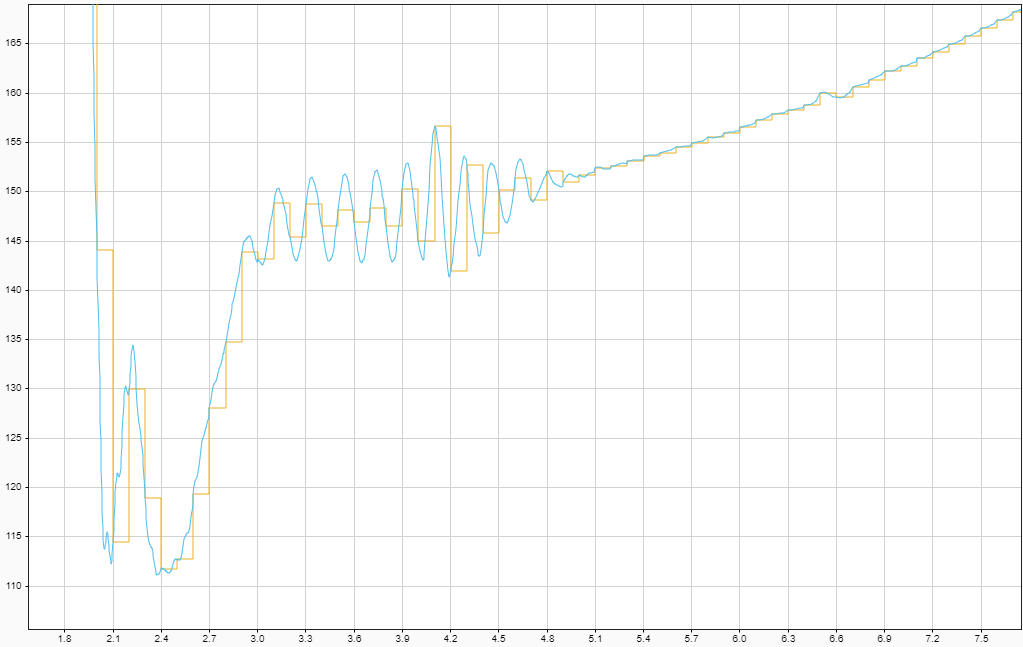
In case of STM32F4-Discovery, starting the PIL simulation, a new OpenOCD session is launched to download the code onto the external target. If ST-LINK is selected as the PIL communication interface, OpenOCD also performs host-to-target communication during PIL simulations.

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.

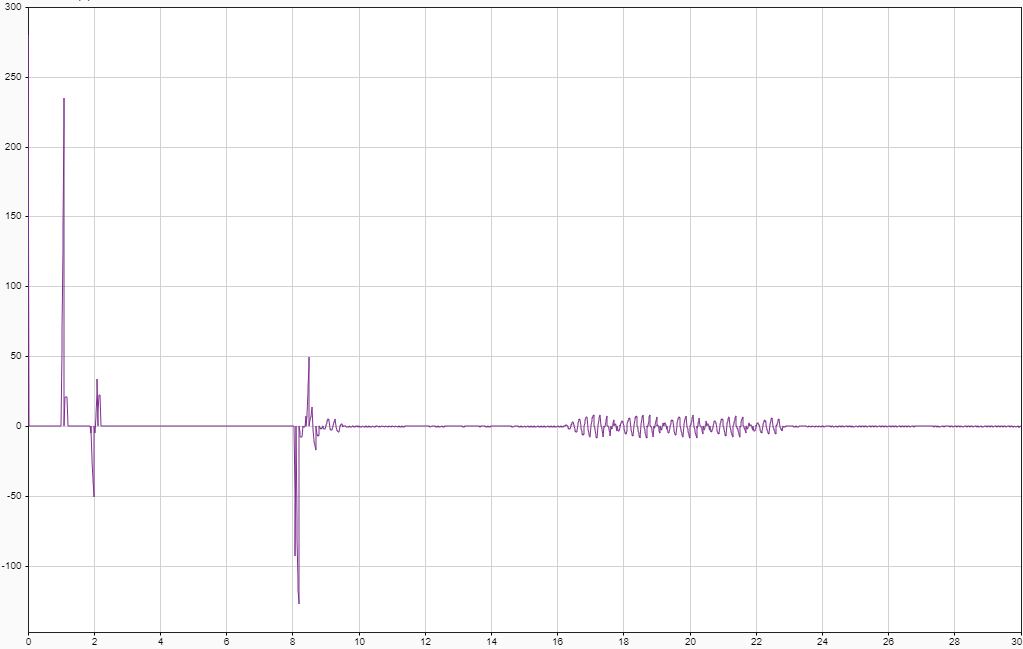


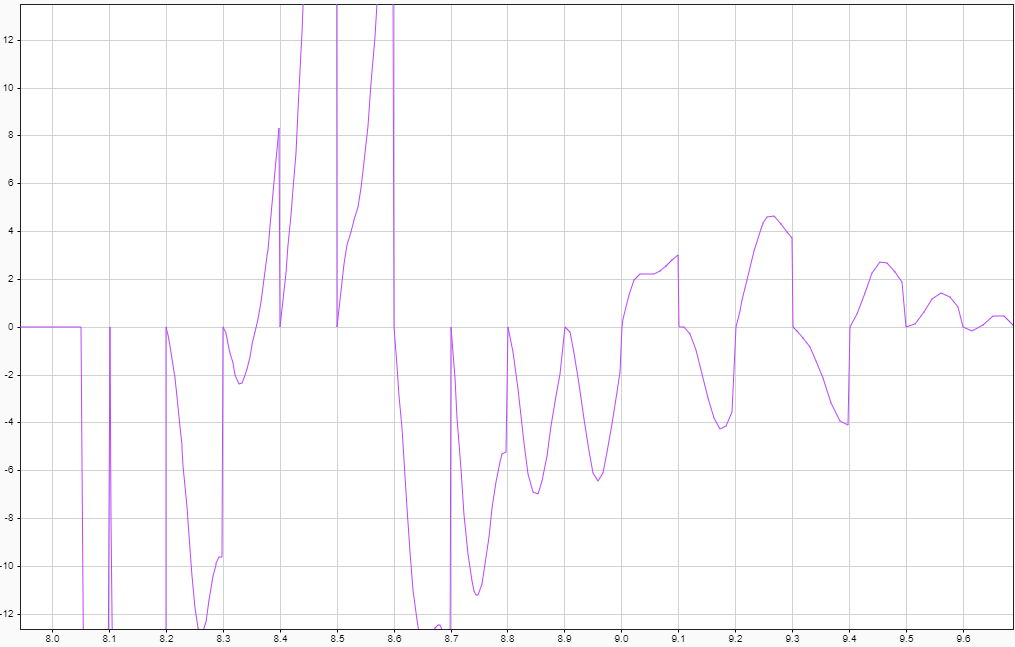


This sampling and holding technique leads to a error difference, between the two output signals, which has large spikes when requested torque changes very quickly, usually due to a nervous driver suddenly pushing or releasing the throttle pedal. These spikes are, however, limited in time, as they vanish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