
Article:
Tristano, M., Lenzo, B., Xu, X. et al. (2024) Hardware-in-the-loop real-time implementation of a vehicle stability control through individual wheel torques. IEEE Transactions on Vehicular Technology, 73 (4). pp. 4683-4693. ISSN: 0018-9545

https://doi.org/10.1109/tvt.2024.3364151

Hardware-in-the-loop real-time implementation of a vehicle stability control through individual wheel torques

Mariagrazia Tristano, *Student Member, IEEE*, Basilio Lenzo, *Member, IEEE*, Xu Xu, Bart Forrier, Thomas D'hondt, Enrico Risaliti, Erik Wilhelm

***Abstract*—The enhancement of vehicle passenger safety is a central theme for car manufacturers. Since the introduction of Electronic Stability Control (ESC) in the late 90s, researchers in industry and academia have kept striving for continuously enhancing vehicle safety. However, despite significant efforts, the literature shows that a significant number of these endeavors have not advanced beyond theoretical formulations and software simulations. This paper presents the testing journey of an individual-wheel-torque-based vehicle stability controller through the major milestones of its development cycle. First, the controller is formulated based on specific vehicle dynamics design requirements. Then, an offline co-simulation is put in place to validate the controller along relevant maneuvers, with the controller running on Matlab-Simulink concurrently with the software Amesim running a 15-dof vehicle model of Siemens' SimRod battery electric vehicle. Next, a real-time co-simulation is achieved, running both the controller and the vehicle model on a real-time platform. Finally, an experimental hardware-in-the-loop setup is built, incorporating a dedicated Electronic Control Unit (ECU), and successfully tested.**

***Index Terms*—vehicle dynamics, hardware-in-the-loop, real-time, control, direct yaw moment, yaw rate, sideslip angle, vehicle stability, experiments.**

I. Introduction

Vehicle stability controllers allow to maintain driveability in safety-critical situations, preventing vehicle loss of control (e.g. spinning or drifting). A well-known example of vehicle stability control is the Electronic Stability Program (ESP), also referred to as Electronic Stability Control (ESC). ESC is mandatory in modern passenger cars and it has been shown to significantly contribute to reducing the number of traffic-related fatalities [1].

M. Tristano is with the Department of Engineering and Mathematics, Sheffield Hallam University, Sheffield, UK, e-mail: mariagrazia.tristano@student.shu.ac.uk.

B. Lenzo is with the Department of Industrial Engineering, University of Padova, Padua, Italy, e-mail: basilio.lenzo@unipd.it.

Xu Xu is with the Department of Computer Science, University of Sheffield, Sheffield, UK, e-mail: xu.xu@sheffield.ac.uk.

Bart Forrier, Thomas D'hondt and Enrico Risaliti are with the TSVT division of Siemens Digital Industries Software, Leuven, Belgium, e-mail: bart.forrier@siemens.com; thomas.dhondt@siemens.com; enrico.risaliti@siemens.com.

Erik Wilhelm is Head of Research at KYBURZ Switzerland AG, Freienstein, Switzerland, e-mail: erik.wilhelm@kyburz-switzerland.ch

Manuscript received April 2023; accepted February 2024.

The design and deployment of a vehicle control system require an important number of intermediate steps, where hardware is gradually integrated to replace simulation blocks. An overview of the steps to be followed to that end is given by the so-called V-cycle presented in [2], where the project cycle is split in a descending branch, where the user requirements are transformed in target performance (design phase), followed by an ascending one, containing the steps to get to full validation of the built implementation (verification phase). The main idea behind this approach is that the controller must be developed conjointly with an appropriate test bench (practically, a set of equations reproducing the complexity of the full-scale vehicle and simulating the vehicle response to the controller) to verify the effectiveness of the former at each stage. The original structure in [2] is adapted to a sequential workflow in Fig. 1, where the design and verification phases are highlighted above their corresponding set of activities. Once design requirements are defined, a controller is developed (e.g. in Matlab) and tested on a validated vehicle model. This phase is denoted as "Offline co-simulation", since it takes place offline (i.e. in a domain where the running time has no relevance) and both the controller and the vehicle model are running in co-simulation on separate software. The subsequent step is "Real time co-simulation", in which the individual building blocks containing the controller and the vehicle model are made software-independent and tested as interconnected black boxes on a high-performance PC, able to operate in real time. Then, the controller is turned into code to allow "Hardware-in-the-loop" testing, featuring the same high-performance PC interacting with a dedicated hardware running the controller code in real time. After the controller reliability has been extensively and thoroughly tested in a considerable number of meaningful driving scenarios, validation is possible on a full-scale vehicle.

Besides the well-known contributions from van Zanten, e.g. [3], many other literature efforts that looked into vehicle stability control, covering the V-cycle up to different stages, including sound work that successfully reached the experimental stage. For instance, [4] presents co-simulations to show the effectiveness of the design of a sliding mode controller integrating tire saturation effects by performing constrained torque allocation to affect a multi-actuated four-wheel electric vehicle with active steering capabilities. The optimised distribution of wheel torques is also at the foundation of [5], showing co-simulations of their control strategy

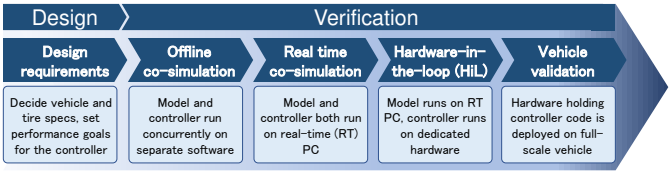


Fig. 1. Workflow for the validation of the presented vehicle control system.

that influences the vehicle yawing behaviour through a three-level controller. In [6], the benefits of the ESC are combined with those of the active front steering in order to achieve a more efficient tracking of the desired yawing behaviour; only simulations are presented. Active front steering is used also in [7], in coordination with direct yaw moment control. In [8], co-simulations show the effectiveness of a stability control algorithm operating on an electric vehicle equipped with four in-wheel motors, combined with regenerative braking. [9] proposes and experimentally validates a concurrent yaw rate and sideslip angle controller using a single-input single-output formulation. In [10] a comparison is made with a standard ESC formulation to show the benefits of the proposed controller.

These kinds of contributions exploit multi-motor actuation and/or active steering. While multi-motor electric cars, possibly even with active steering, are somewhat likely to be the future, the majority of vehicles are still powered by an internal combustion engine or a single electric motor (or two, one per axle). Instead, any vehicle is equipped with individual wheel braking capabilities. Given that traffic-related deaths are rising [11], the possibility of implementing effective individual-brake-based stability controllers is very interesting. [12] indeed developed an H_∞ controller braking one wheel at a time, even if only simulations are presented. [13] also looks into differential braking using a fuzzy logic controller, and shows simulations performed with a 7-DOF model with a Dugoff tire model. Recently, [14] has presented an individual brake-by-wire system for enhancing vehicle stability, proposing Matlab-CarSim co-simulations.

This paper is an extension of [15], where a yaw rate and sideslip controller using individual brakes was presented, with the validation workflow limited to the real time co-simulation stage. Within this paper, significant further details are provided and, most importantly, a further stage of the workflow is successfully achieved: Hardware-in-the-loop testing.

Section II describes the main components of the control framework. Sections III, IV and V describe respectively the offline co-simulation, real time co-simulation and Hardware-in-the-loop implementations. Performance results and their consistency throughout the testing progression are presented in Section VI. Conclusions and future steps are in Section VII.

II. Control framework

The proposed control framework is shown in Fig. 2 and features four main components:

- **Vehicle model.** It acts as a test bench for the controller by mimicking the response of the validation vehicle. Its inputs are: i) steering angle; ii) pedal positions - mapped into a desired torque at each driven wheel; iii) additional torque demand from the Low-level controller.
- **Reference generator.** It makes use of the driver steering angle and estimated/measured vehicle states to generate a desired reference yaw rate.
- **High-level controller.** It uses the difference between the reference yaw rate and its actual value, provided as a feedback measurement, to compute - by means of, e.g., a Proportional-Integral (PI) controller - an appropriate direct yaw moment action M_z to implement the desired cornering behaviour on the vehicle.
- **Low-level controller.** It maps the desired M_z into additional torque demands ΔT_i at each vehicle side.

The following subsections dive deeper in the role of the aforementioned components.

A. Reference generator

The reference generator is in charge of establishing the desired yawing behaviour of the vehicle. When the driver inputs suggest a safe driving condition, the target reference yaw rate is a handling reference r_h, otherwise a stability reference r_s is also involved to help the driver regain control of the vehicle. Such inputs are combined in a unique reference yaw rate, defined as the weighted sum of r_h and r_s by means of a weight factor ρ:

$$r_\mathrm{ref} = \rho \cdot r_\mathrm{s} + (1-\rho) \cdot r_\mathrm{h} \tag{1}$$

The following subsections provide further insight on the individual quantities involved in Eq. (1).

1) The handling reference: The handling yaw rate reference r_h is defined starting from the desired vehicle cornering response, which in turn is computed based on the driver inputs, i.e. steering wheel angle and longitudinal velocity ($\delta_\mathrm{SW}, v_\mathrm{x}$ respectively). The desired cornering performance is described by three parameters: the linear understeer gradient K_lin, the limit acceleration for the linear range of operation a_y^* and the maximum achievable acceleration $a_\mathrm{y,max}$. The aforementioned parameters can be visualised in Fig. (3), where the cornering behaviour for a generic understeering vehicle is pictured.

The handling yaw rate itself is extracted at each time step from a two-dimensional lookup table, whose two entries are the steering angle δ and the longitudinal velocity v_x: details on how to build such lookup table are provided hereinafter.

The full ranges of longitudinal velocity (assuming its maximum value is 150kph) and lateral acceleration (whose maximum value is $a_\mathrm{y,max}$) are split in small intervals, and the corresponding yaw rate for each of their combinations is computed through Eq. (2), reporting the steady-state relationship between the lateral acceleration a_y, longitudinal velocity v_x and yaw rate r.

$$r = \frac{a_\mathrm{y}}{v_\mathrm{x}} \tag{2}$$

The relationship between yaw rate and velocity has now been defined. To transform the other table entry into steering angle, a linking relationship needs to be found between the

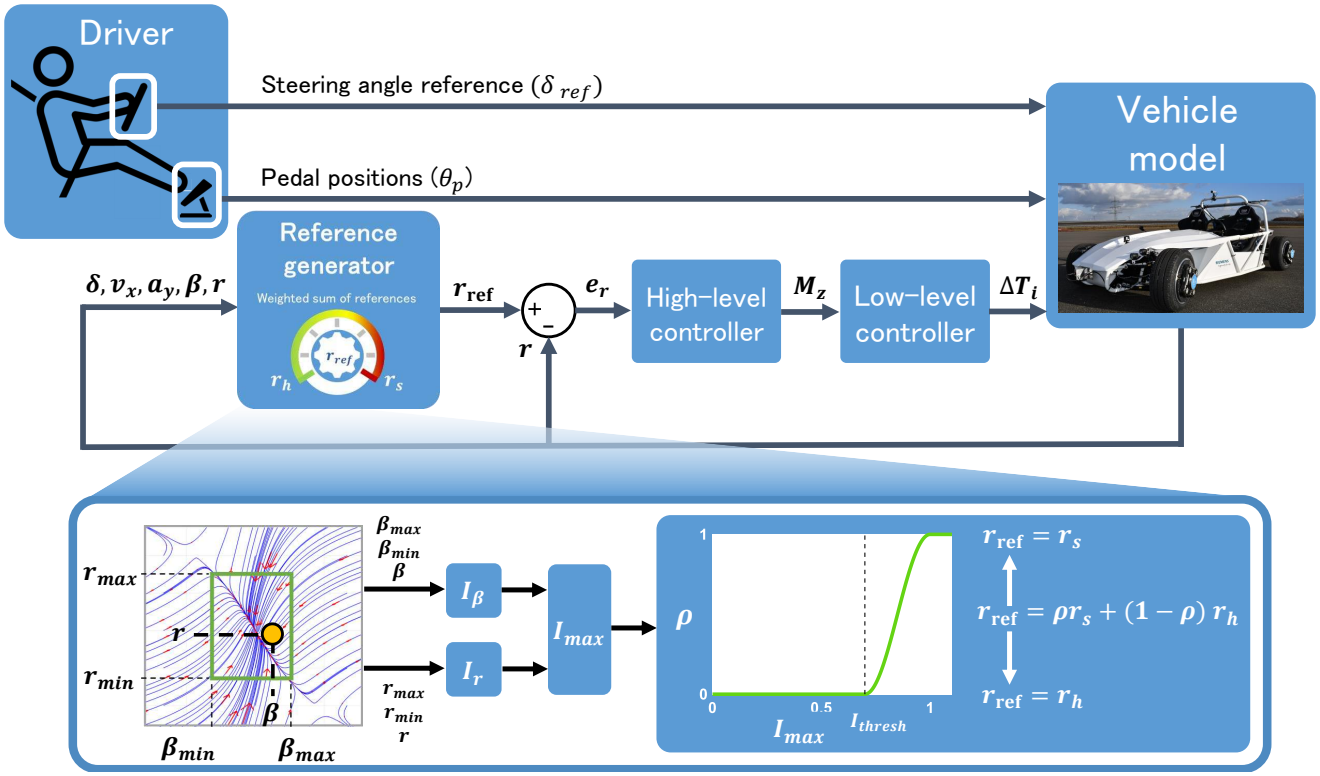


Fig. 2. Control scheme framework.

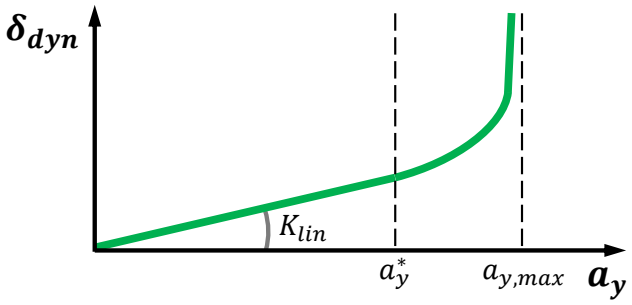


Fig. 3. Cornering response of a generic understeering vehicle, highlighting the parameters involved in the definition of the desired cornering response.

lateral acceleration and the steering angle. The green curve in Fig. 3 offers a parametrization between the lateral acceleration and the dynamic steering angle, formulated through the linear understeer gradient K_{lin}, the limit acceleration for the linear range of operation a_{y}^* and the maximum achievable acceleration $a_{\text{y,max}}$. The obtained dynamic steering angle is then summed to the kinematic one, which depends on the vehicle wheelbase l, the yaw rate r (as defined in Eq. 2) and the longitudinal velocity v_{x}: the kinematic steering angle δ_{kin} is defined in Eq. 3 and the full lookup table is obtained. More details can be found in [16,17].

$$\delta_{\text{kin}} = l\frac{r}{v_{\text{x}}} \tag{3}$$

The distinction between kinematic and dynamic steering angle is only relevant for the handling reference definition: in the remainder of the paper, the steering angle is regarded as a single quantity bearing both the aforementioned components.

2) The stability reference: The stability reference r_{s} is defined by Eq. (4), where the current lateral acceleration a_{y} is scaled by a factor $k_{\text{s}} < 1$ to ensure a sufficient safety margin.

$$r_{\text{s}} = k_s\frac{a_{\text{y}}}{v_x} \tag{4}$$

3) The weight factor: The weight factor ρ is computed at each time step as the result of a smoothed varying weight function, regulating the priority of either references depending on the stability condition of the vehicle, whose assessment comes from exploiting the concept of phase-plane stability.

Phase portraits are a widely-employed tool in vehicle dynamics (e.g. [18]–[22]): they display the evolution of relevant states in the form of trajectories. In the case at hand (Fig. 4) the yaw rate r is portrayed against sideslip angle β. The trajectory behaviour of a number of initial working points within reasonable boundaries is investigated: the points whose evolution settles within certain state boundaries constitute the stability region. Various strategies have been devised to properly encase the stability region. Some techniques rely on finding the so-called separatrix, which is the critical state trajectory that splits the stable initial conditions from the unstable ones on the phase plane ([23]). The stability region may also be identified as the plane strip bound within steady-state yaw rate conditions and constrained by tire saturation limits ([22]), which results in a segmented boundary shape. A more conservative approach is adopted by Guo in [24], where the risk of skidding in proximity of stability boundaries is

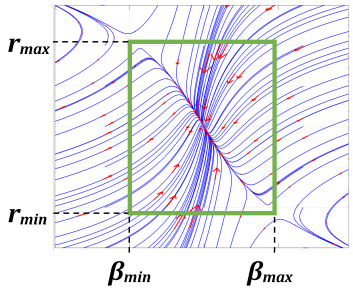


Fig. 4. Generic $\beta-r$ phase portrait featuring orientation vectors for individual trajectories, where the parallelogram in bold represents the stability region.

prevented by limiting the critical yaw rate and sideslip angle to even lower values: more specifically, constant maximum and minimum values are identified for both state variables. Similarly, in this work the boundaries for the yaw rate and the sideslip angle have both been chosen to be symmetrical with respect to the origin, hence the rectangular shape of the stability region in Fig. 4. More specifically, the sideslip angle limits β_{min} and β_{max} were chosen to coincide with the saddle point coordinates, while the yaw rate limits r_{min} and r_{max} are based on the friction constraints affecting the lateral acceleration, yielding Eq. (5). Further indications on how to retrieve such values are provided in [24].

$$r_{\text{max}} = \left| \frac{\mu g}{v_x} \right| \tag{5}$$

The vehicle working point coordinates (β, r) are estimated and measured (respectively) at each time step, and their closeness to the stability region boundaries is quantified in the two indexes I_β and I_r, defined respectively in Eq. (6) and Eq. (7).

$$I_\beta = 1 - \text{sign}((\beta_{\text{max}} - \beta)(\beta - \beta_{\text{min}})) \times \frac{\min(|\beta_{\text{max}} - \beta|, |\beta - \beta_{\text{min}}|)}{\frac{\beta_{\text{max}} - \beta_{\text{min}}}{2}} \tag{6}$$

$$I_{\text{r}} = 1 - \text{sign}((\text{r}_{\text{max}} - \text{r})(\text{r} - \text{r}_{\text{min}})) \times \frac{\min(|\text{r}_{\text{max}} - \text{r}|, |\text{r} - \text{r}_{\text{min}}|)}{\frac{\text{r}_{\text{max}} - \text{r}_{\text{min}}}{2}} \tag{7}$$

The maximum between such two indexes, i.e. the most critical one, is chosen (Eq. 8) and compared to a pre-set threshold value (I_{t}) to then compute ρ, using Eq. (9).

$$I_{\text{max}} = \max(I_\beta, I_{\text{r}}) \tag{8}$$

$$\rho(I_{\text{max}}) = \begin{cases} 0, & \text{if } 0 \leq I_{\text{max}} < I_{\text{t}} \\ \frac{1}{2}(1 - \cos\left(\pi \frac{I_{\text{max}} - I_{\text{t}}}{1 - I_{\text{t}}}\right)), & \text{if } I_{\text{t}} < I_{\text{max}} \leq 1 \end{cases} \tag{9}$$

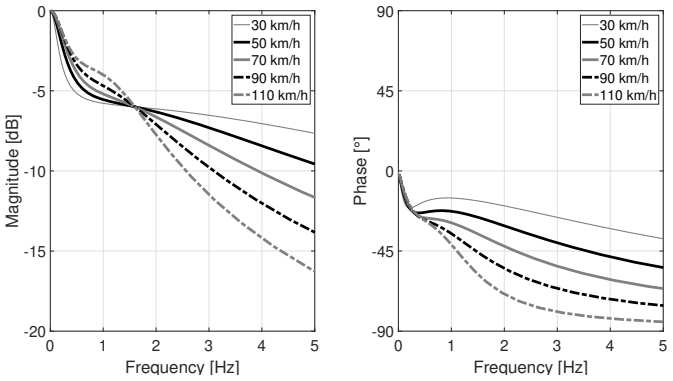


Fig. 5. Gain and phase closed-loop system Bode plots for different speeds and proportional gains K_p.

B. High-level controller and low-level controller

The high-level controller is a PI controller:

$$M_z = \text{K}_\text{p} e + \text{K}_\text{i} \int e \, \text{d}t \tag{10}$$

with constant integral gain K_i, and with a proportional gain K_p that is regulated through gain scheduling. Following the approach in [9], the underlying idea is to ensure a constant - regardless of speed - bandwidth of the closed loop transfer function $G_{rM_z}C/(1 + G_{rM_z}C)$, where G_{rM_z} is the transfer function of yaw moment to yaw rate and $C = K_p + K_i/s$ in which s is the Laplace operator. The target closed-loop bandwidth, herein defined as the frequency corresponding to a 6 dB gain drop, is set to 1.6 Hz. Table I reports the proportional gain values for a constant $\text{K}_\text{i} = 26000$ Nm/rad at different speeds, along with the bandwidth resulting from those parameters, while Fig. (5) depicts the gain and phase Bode plots of the obtained closed-loop system.

TABLE I
PROPORTIONAL GAIN AND CLOSED-LOOP BANDWIDTH FOR DIFFERENT SPEEDS.

Speed (km/h)	K_p (Nms/rad)	Bandwidth (Hz)
30	15058	1.60
50	9080	1.60
70	6279	1.60
90	4549	1.60
110	3271	1.60

The low-level controller is essentially a torque allocator, that translates the computed direct yaw moment into an individual braking torque effort to be applied on the wheels. Intuitively, a positive (anti-clockwise) direct yaw moment can be achieved by assigning braking torque to the left-side wheels. Conversely, a negative (clockwise) direct yaw moment is accomplished by allocating braking torque on the right-side wheels. Therefore:

$$\Delta T_i = \begin{cases} 2\dfrac{M_z R_w}{t_w}, & \text{i = left} \quad \text{if} \quad M_z \geq 0 \\ -2\dfrac{M_z R_w}{t_w}, & \text{i = right} \quad \text{if} \quad M_z < 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where R_w represents the wheel radius, t_w is the vehicle track (front and rear are assumed to be equal) and ΔT_i is the additional torque effort needed on the left or right side. ΔT_i is equally split between front and rear wheel of each side because in normal driving conditions that allows a greater achievable yaw moment than would be possible by choosing either only front or only rear. This is overruled in case of wheel torque saturation, when the torque demand beyond the limit is transferred to the other wheel on the same side until saturation is reached as well.

C. Vehicle model

The presented control strategy underwent some preliminary mild tests on a single-track model in MATLAB-Simulink. However the model linearity makes it inaccurate especially when approaching the vehicle stability limits. For the sake of reliability, a 15 degree-of-freedom (DoF) model defined on Simcenter Amesim [25] and based on the SimRod battery electric vehicle [26] (Fig. 6) was adopted. The number of degrees of freedom comes from: i) the vehicle chassis, a rigid body with spatial motion, hence 6 DoF; ii) four wheels, each with a rolling DoF and a vertical DoF, 8 DoF overall; steering system, 1 DoF. Measurements of interest are made available as well as delays associated with their retrieval. Actuators have an ideal behavior, i.e. without delay/rate limitation and with an infinite bandwidth. A Dugoff tire model is employed, whose longitudinal and lateral force expressions account for adherence and load conditions and any related phenomena. The longitudinal and lateral force expressions of the Dugoff model for a generic tire are reported in Eq. (12), where C_λ and C_α are respectively the longitudinal and lateral tire tread stiffness, s_s and s_l are respectively the longitudinal and side (lateral) slip, F_z is the vertical load, $\mu_{\max}$ is the maximum friction coefficient between the longitudinal and lateral one (defined as the ratio between the longitudinal and lateral force, respectively, over the vertical load), v_x and v_y are respectively the longitudinal and lateral velocity, ω is the angular speed of the wheel and R_w is the effective rolling radius.

$$F_x = C_\lambda \frac{s_l}{1+s_l} f(\xi), \;\; F_y = C_\alpha \frac{\tan(s_s)}{1+s_l} f(\xi)$$

$$\text{where} \quad f(\xi) = \begin{cases} (2-\xi)\xi, & \xi < 1 \\ 1, & \xi \geq 1 \end{cases}$$

$$\text{with} \quad \xi = \frac{\mu_{max} F_z (1+s_l)}{2\sqrt{(C_\lambda s_l)^2 + (C_\alpha \tan(s_s))^2}} \tag{12}$$

$$\text{where in turn} \quad s_s = \begin{cases} \dfrac{v_y}{\omega R_w}, & \text{if driving} \\ \dfrac{v_y}{v_x}, & \text{if braking} \end{cases}$$

$$\text{and} \quad s_l = \begin{cases} \dfrac{\omega R_w - v_x}{\omega R_w}, & \text{if driving} \\ \dfrac{\omega R_w - v_x}{v_x}, & \text{if braking} \end{cases}$$

The main vehicle model and tire model parameters are in Table II, where the acronym CoG stands for Center of Gravity.

TABLE II
VEHICLE AND TIRE PARAMETERS.

Parameter	Description	Value	Unit
m	Total vehicle mass	860	kg
l	Vehicle wheelbase	2.335	m
a_1	Distance of front axle to CoG	1.171	m
a_2	Distance of front axle to CoG	1.164	m
t_w	Track width	1.428	m
h_{CoG}	Height of CoG from ground	0.1	m
J_z	Yaw inertia	700	kg · m ^{2}
C_λ	Tire longitudinal stiffness	37500	N/rad
$C_{\alpha,f}$	Tire longitudinal stiffness (front)	37816	N/rad
$C_{\alpha,r}$	Tire longitudinal stiffness (rear)	52140	N/rad
μ_{max}	Maximum friction coefficient	1	-
R_w	Effective rolling radius	0.302	m

The Amesim model is essentially a state-space description of the actual vehicle, whose states are retrieved by applying an integration algorithm performed by a solver. To ensure the numerical stability of the model and secure in turn a smooth and reliable simulation, a linearization analysis must be executed. The linearization analysis studies the second-order model dynamics and yields the response of the model when subject to a given source of excitation. Eigenvalues are then computed, making it possible to draw conclusions about the model stability. Since the model will operate in discrete time, another important result of the analysis is the suggested sampling time to guarantee a robust transition from the continuous to the discrete time domain. In the case at hand, the examined framework is the 15 degree-of-freedom Amesim model using a second order Runge-Kutta solver algorithm. Results suggest that the sampling time should not fall below 5.67 ms, value above which one of the computed poles crosses the boundaries of the stability region. This leads to the safe choice of 1 ms as the model sampling time.

Fig. 6. SimRod electric vehicle.

III. Offline co-simulation

As stated in Section I, at each testing stage the controller needs an appropriate test bench to determine its performance. The first testing instance occurs entirely on separate simulation software, namely Simcenter Amesim, running the vehicle model, and MATLAB-Simulink, carrying the controller. Nonetheless, both environments are not operating independently one from the other: Simcenter Amesim can communicate with MATLAB-Simulink and viceversa through co-simulation interfaces.

The individual braking torques provided by the controller black-box in Amesim are summed to the ones already generated within Amesim, hence the braking action does not override the regular braking actions performed by the driver, but rather overlaps to their effort.

The terminology "co-simulation" implies that both software packages are run concurrently on a unique simulation, so the simulation run parameters need to match. Following the linearization analysis described in Section II-C, a fixed-step solver is chosen (hence the need for a time step) featuring an order 2 Runge-Kutta integration algorithm: the run parameters are summarized in Table III.

TABLE III
Run parameters for co-simulation.

Parameter	Description	Value	Unit
t_s	Simulation start time	0	s
t_f	Simulation end time	7	s
Δt	Integration time step	1	ms

Upon completion of the offline testing phase, it is worth noticing that the simulation time thus far has been affected by the speed of the employed CPU: the co-simulation at this stage has been characterized as "offline", meaning that the completion time has no significance. While progressing with the testing, the controller will ultimately only impact the full-scale vehicle in the desired way if it is capable of real time operation: using more computationally capable hardware is then a mandatory requirement. This necessity can be fulfilled through gradual hardware integration in the framework, which at this stage can be seen as an interaction between three main components (Fig. 7):

- **Driver:** provides the steering wheel angle and the longitudinal velocity to the vehicle.

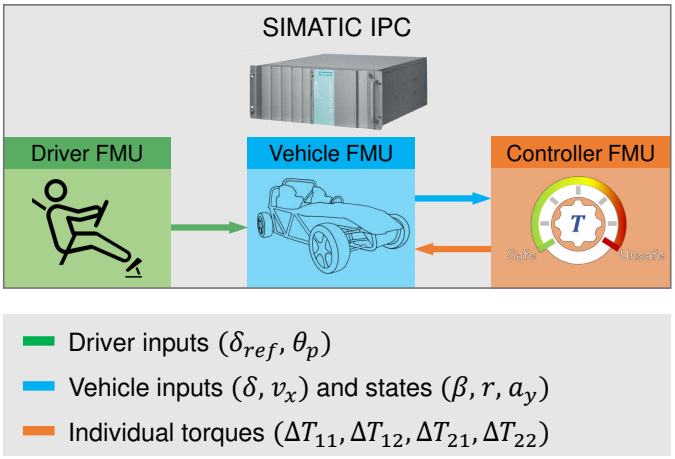


Fig. 7. Overview of real-time co-simulation signal exchange, happening within FMUs operating on the RT platform.

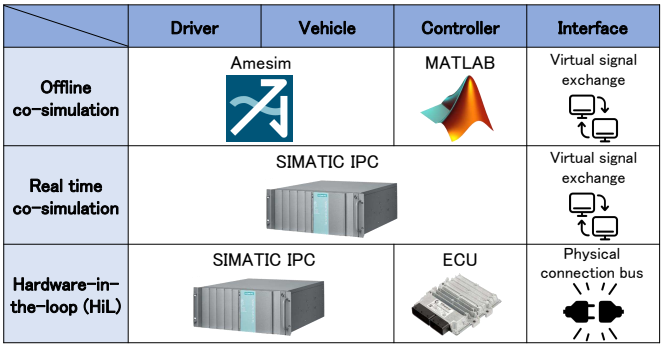


Fig. 8. Location map of the components for each test instance.

- **Vehicle model:** reacts to the inputs provided by the driver model mimicking the reaction of the full-scale vehicle to present driver inputs.
- **Controller:** receives the information coming from the vehicle model and accordingly generates (if need be) the braking torques to stabilise it.

To successfully keep track of where each component is located in all testing instances, the hardware integration flow is mapped in Fig. 8.

IV. Real time co-simulation

The first introduced hardware component is a Real-Time (RT) platform, displayed in Fig. 9. As suggested by its name, the computational abilities of this unit make it a suitable candidate for real-time evaluations of the proposed algorithm, and it also allows real-time communication with the dedicated hardware on which the controller will be deployed further on, namely the Electronic Control Unit (ECU).

Fig. 9 features an overall description of the platform components. More specifically:

- **I/O modules** provide simple digital I/O with the ECU and allow wheel speed sensor emulation, a useful tool for further testing steps.
- **SIMATIC IPC** is a high-performance computer, with a real-time operating system, thus acting as real time platform.

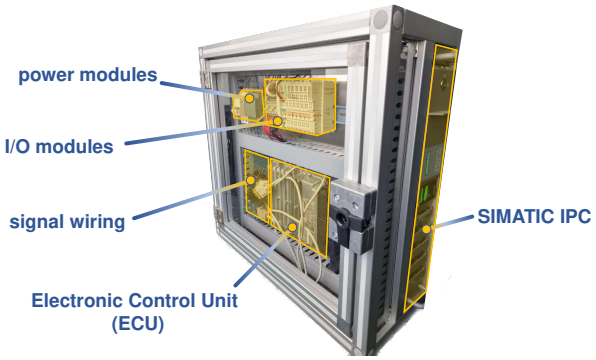


Fig. 9. Real-time test platform.

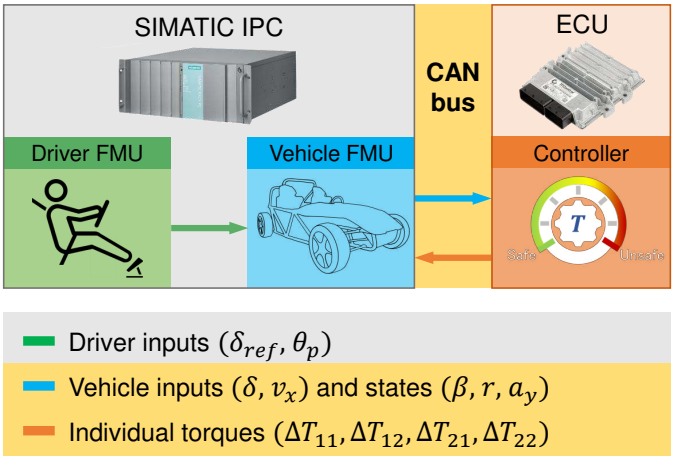


Fig. 10. Overview of Hardware-in-the-loop signal exchange. Communication between Driver FMU and Vehicle FMU still happens within the RT platform, while the Vehicle FMU and the controller deployed on the ECU exchange signals via CAN bus.

The ECU is also shown in place on the platform in Fig. 9. However, a sensible initial approach is to run the entire framework on the RT platform: this can be referred to as "real time co-simulation" and constitutes a middle-ground testing strategy while transitioning from the offline domain to Hardware-in-the-loop.

The RT platform [27] supports the Functional Mock-up Interface (FMI) standard (Version 2.0) [28] for co-simulation, hence it can accommodate models defined in various simulation tools: this makes the testing universal, rather than software-dependent. The individual building blocks of the present implementation, discussed in Section III, become Functional Mock-up Units (FMUs), black boxes whose interface input and output signals are known. Connections are performed among interface signals, e.g. expected torque inputs from the vehicle model and output torques generated from the ECU, so that all components are correctly interfaced.

The co-simulation instance is finally called from a simple user interface: a real-time co-simulation master handles the the time step definition for the simulation instance and sets the timing of the task to be performed by the individual framework components (driver model, vehicle model and controller), which consequently classify as slaves.

V. Hardware-in-the-loop

The hardware progression path for the vehicle model and driver model has now reached its final stage: from this point forward, the test bench for the controller is going to remain as is. On the other hand, the controller itself needs to advance to its final testing stage by being moved to a dedicated hardware, namely the ECU. Specifically, HYDAC's TTC-580 was selected as target hardware [29]. The controller block scheme from Simulink is then be turned into code and flashed on the ECU: as indicated by the last table row entry in Fig. 8, this testing instance will have two interfaced FMUs (driver and vehicle model) running on the real-time platform while the controller runs on the ECU. In an initial phase, due to the uncertain computational load required by the controller, the controller time step is set to 10 ms.

So far, the signals that have been handled in co-simulation were all travelling on the same component. However, the addition of the ECU needs a communication protocol for correct interfacing of the two modules. The CAN bus is then designed to translate the analog signals coming from the RT platform to suitable grouped digital signals (messages) to be read by the ECU, as well as writing the digital outputs to be translated for the RT platform.

A visual overview of the process is given in Fig. 10, specifying the transmitted and received signals between the two units. The RT platform and the ECU are communicating via CAN: message transmission is regulated by the CAN interface, a dedicated piece of software, and performed through a physical cable, the CAN connector. The connection also occurs virtually by stating the appropriate connection target for the ECU in a dedicated software.

Finally, the controller block scheme is converted to source code and flashed on the target device, i.e. the ECU.

Other than the signals themselves, calling a simulation instance will also yield insightful information on how long it takes for each slave to complete their tasks at every time step: this is labelled as execution time. The sampling time constitutes a rigorous upper bound for the completion of a single task: if the latter takes longer than the sampling time to finish, that instance is said to be an overrun. The absence of overruns is hence definite proof that the controller is able to operate in real time. The ability to check both performance and timing of signals to ensure real time capabilities highlights the value of executing Hardware-in-the-loop testing.

VI. Results

Results may be divided into: i) controller performance; ii) consistency of said performance throughout the various testing instances; iii) required execution time to achieve them. All three aspects are featured in the following subsections.

A. Controller performance

To assess the performance of the proposed controller in improving the vehicle lateral stability, a lane change manoeuvre is executed in both mild and challenging driving

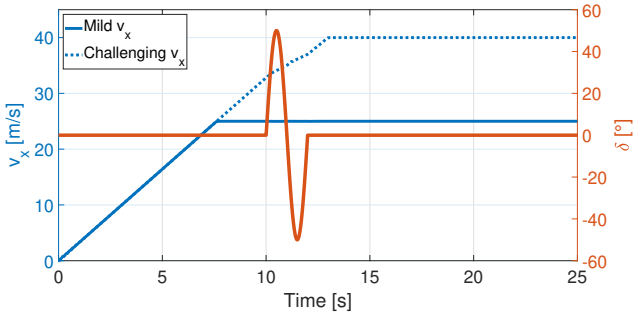


Fig. 11. Single lane change manoeuvre inputs for both presented scenarios.

conditions. The mild scenario features a sinusoidal steering input applied at a constant speed of 25 m/s, while in the challenging one the sinusoidal steering input occurs while the vehicle is traveling at 33 m/s and is experiencing a 3 m/s^2 longitudinal acceleration. The amplitude of the sinusoidal steering input is of 50° and its frequency is 0.5 Hz in both scenarios. Fig. 11 shows the speed and steering angle time histories. A more comprehensive description of the results for the mild and challenging scenarios follows.

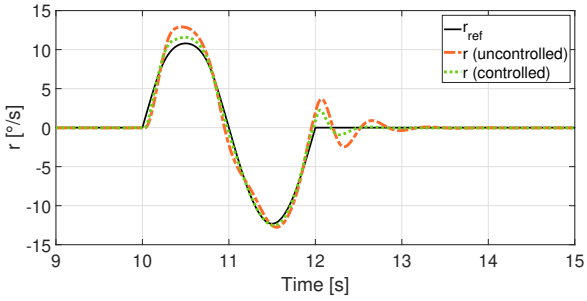
1) Mild scenario: Offline co-simulation results for the mild scenario are provided in terms of yaw rate and sideslip angle, respectively in Fig. 12a and Fig. 12b, for the uncontrolled vehicle and the controlled one, i.e. with the controller set to "off" and "on" state respectively. Fig. 12a also depicts the reference yaw rate, which gives a visual indication on how close are the expected and actual yaw rate behaviour.

As further proof of the mild nature of the manoeuvre, Fig. 12c portrays the indexes I_β and I_r alongside the weight factor ρ: the indexes never exceed the threshold values (red dashed lines) which never prompts ρ to change from 0. This indicates a safe scenario.

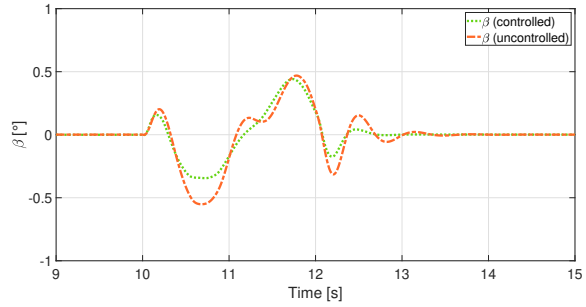
2) Challenging scenario: Offline co-simulation results for the challenging scenario are reported in Fig. 13a and Fig. 13b, showing respectively the behaviour of the yaw rate and sideslip angle. While the uncontrolled vehicle clearly shows instability and loss of control, the controlled vehicle is capable of safely completing the manoeuvre. As for the mild scenario, additional evidence of the proper triggering of the controller upon detection of instability is given by the stability index of the yaw rate (Fig. 13c), exceeding the allowed threshold and prompting the variable weight factor ρ to increase, hence prioritizing the stability reference r_s.

B. Results consistency through testing

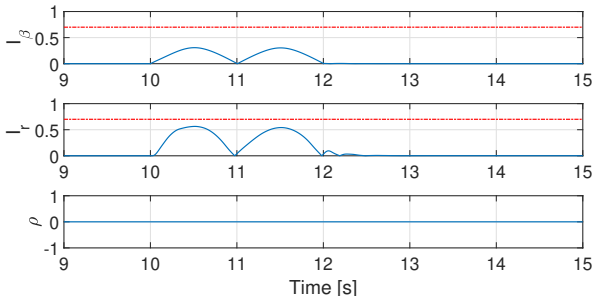
Thus far, the presented outcomes have all been referred to offline co-simulation testing. The same results can be faithfully reproduced both in the real time co-simulation and Hardware-in-the-loop test cases. Fig. 14 shows the overlapped sideslip angle for the challenging scenario in all three testing instances, where the signals are observably very close. Furthermore, focusing on a narrow time interval, the timing of the three signals can be appreciated: the offline and real time co-simulation sideslip angle signals are only around 0.5 ms apart, a timing difference mostly due to the presence of the



(a) Yaw rate comparison for the mild scenario, in the offline co-simulation testing instance.



(b) Sideslip angle comparison for the mild scenario, in the offline co-simulation testing instance.



(c) Stability indexes and weight factor for the mild scenario.

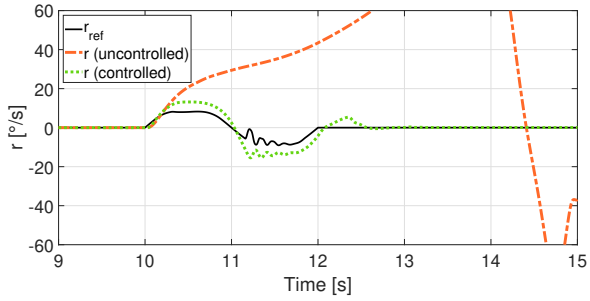
Fig. 12. Mild scenario results.

co-simulation master, running all co-simulation slaves and handling data exchange and logging. The Hardware-in-the-loop signal is approximately 10 ms apart from the offline co-simulation one: the reason can be easily identified in the controller sampling time which, as stated in Section V, is indeed set to 10 ms.

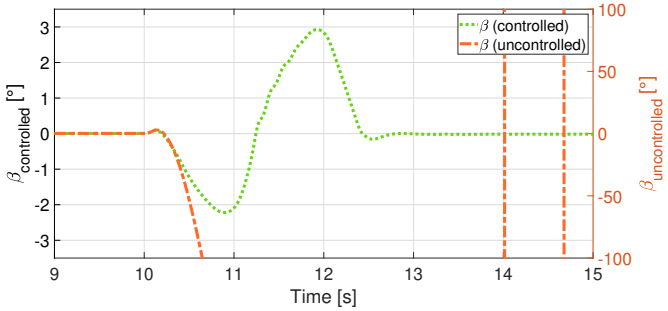
C. Execution time

Studying the execution time for both the master and the slaves is not only a testimony to the real time capabilities of the algorithm, but it can also establish whether there is sensible margin for improvement in the control strategy: provided there is a sufficient time margin in every single time instance, the high-level controller may accommodate a more computationally complex controller than the PI and provide an even better performance.

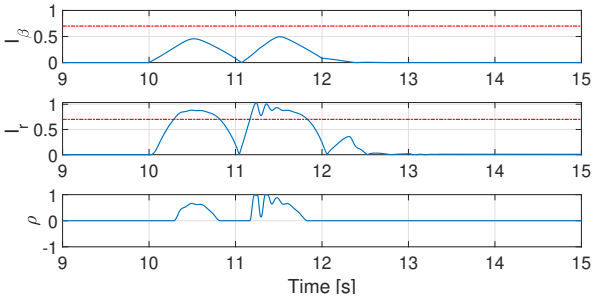
To provide further visual insight, Fig. 15, Fig. 16 and Fig. 17 show the task execution times as fractions of the total available



(a) Yaw rate comparison for the challenging scenario, in the offline co-simulation testing instance.



(b) Sideslip angle comparison for the challenging scenario, in the offline co-simulation testing instance.



(c) Stability indexes and weight factor for the challenging scenario.

Fig. 13. Challenging scenario results.

task execution time (i.e. the sampling time) for the challenging controlled driving scenario, in three cases:

- Fig. 15 shows the average execution times of the individual slaves (except the ECU) and that of the co-simulation master.
- Fig. 16 depicts the worst-case execution times, meaning the highest ones for every slave (except the ECU) and for the master.
- Fig. 17 shows the execution time worst-case scenarios where the highest master execution time is combined with the highest slave execution times, bearing in mind that the master execution time is the time needed to run all co-simulation slaves (except the ECU) and to handle data exchange and logging.

It can be inferred, particularly from Fig. 17, that there is some margin of task time still available to use. Conducting an analogous analysis for the ECU execution time and receiving similarly-encouraging results, would mean that a more computationally-complex controller could possibly be

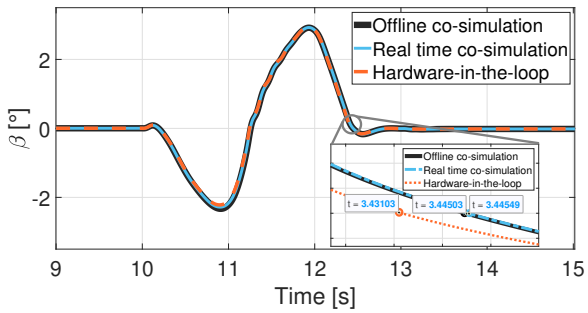


Fig. 14. Sideslip angle comparison for the challenging scenario, throughout development stages of the controller.

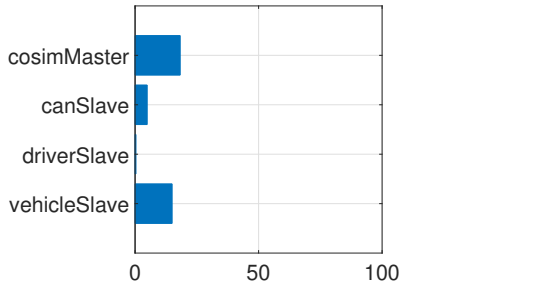


Fig. 15. Average execution times as fraction of total task time.

adopted, potentially able to yield an even better performance than the current best one. Examples of increasingly more complex control strategies are Model Predictive Control (MPC) or Sliding mode. Changing the controller implies starting the development procedure from the design stage, hence making sure that every step yields a satisfactory and appropriate result before diving into the evaluation of execution times.

As far as communication delays associated with the CAN protocol are concerned, they are deemed negligible [30] also considering the limited number of signals dealt with (in Figure 10, those with yellow background).

VII. Conclusions

This paper followed the testing journey of a vehicle stability controller through the major milestones of a development cycle, successfully completing all stages up to and including

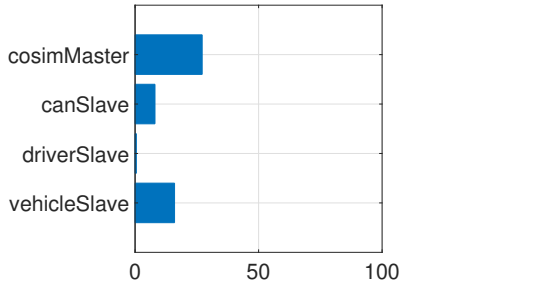


Fig. 16. Worst-case (maximum) execution times as fraction of total task time.

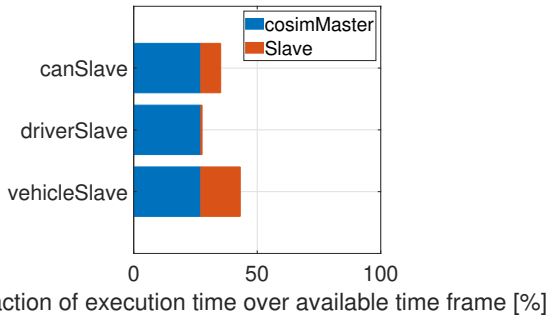


Fig. 17. Worst-case (maximum) slave execution times summed to the worst-case master time as fraction of total task time.

Hardware-in-the-loop testing. At first, a co-simulation between MATLAB-Simulink and Simcenter Amesim was developed, to ensure a reliable assessment of the developed control strategy (offline co-simulation). Next, a validation on a target platform was set up to progress towards specific hardware targets, allowing real-time assessments (real time co-simulation). Finally, the testing platform was connected to a dedicated hardware running the controller (Hardware-in-the-loop testing).

Performance was discussed in terms of achievable control action when compared to the uncontrolled vehicle, as well as looking at execution time. Studying the latter clearly forges a way forward on this work, particularly suggesting the retrieval of ECU execution time to encourage exploration of different controller options if there is still task time availability. Moreover, the potential transition to a new control strategy - or a full-scale vehicle implementation - would require a new communication delay assessment, to keep ensuring an effective and reliable control action.

REFERENCES

[1] Webb, C. N. Estimating lives saved by electronic stability control, 2011-2015 (No. DOT HS 812 391) (2017).

[2] Forsberg, K. & Mooz, H. The relationship of system engineering to the project cycle. *INCOSE International Symposium*. **1**, 57-65 (1991)

[3] Van Zanten, A. Bosch ESP systems: 5 years of experience. *SAE Transactions*. pp. 428-436 (2000)

[4] Nah, J. & Yim, S. Optimization of control allocation with ESC, AFS, ARS and TVD in integrated chassis control. *Journal Of Mechanical Science And Technology*. **33** pp. 2941-2948 (2019)

[5] Chen, B. & Kuo, C. Electronic stability control for electric vehicle with four in-wheel motors. *International Journal Of Automotive Technology*. **15**, 573-580 (2014)

[6] Guo, J., Chu, L., Liu, H., Shang, M. & Fang, Y. Integrated control of active front steering and electronic stability program. *2010 2nd International Conference On Advanced Computer Control*. **4** pp. 449-453 (2010)

[7] Ahmadian, N., Khosravi, A., & Sarhadi, P. (2022). Driver assistant yaw stability control via integration of AFS and DYC. Vehicle system dynamics, 60(5), 1742-1762 (2022).

[8] Zhai, L., Sun, T. & Wang, J. Electronic stability control based on motor driving and braking torque distribution for a four in-wheel motor drive electric vehicle. *IEEE Transactions On Vehicular Technology*. **65**, 4726-4739 (2016)

[9] Lenzo, B., Zanchetta, M., Sorniotti, A., Gruber, P. & De Nijs, W. Yaw rate and sideslip angle control through single input single output direct yaw moment control. *IEEE Transactions On Control Systems Technology*. **29**, 124-139 (2020)

[10] Lenzo, B., Sorniotti, A., Gruber, P., Sannen, K. On the experimental analysis of single input single output control of yaw rate and sideslip angle. *International Journal of Automotive Technology*. **18**, 799-811 (2017)

[11] World Health Organization. (2018). Global status report on road safety 2018: Summary (No. WHO/NMH/NVI/18.20).

[12] Park, J. H. (2001). H_∞ direct yaw-moment control with brakes for robust performance and stability of vehicles. JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing, 44(2), 404-413.

[13] Zhao, C., Xiang, W., Richardson, P. (2006, July). Vehicle lateral control and yaw stability control through differential braking. In 2006 IEEE international symposium on industrial electronics (Vol. 1, pp. 384-389). IEEE.

[14] Zhou, J., Di, Y., Miao, X. (2023). Single-Wheel Failure Stability Control for Vehicle Equipped with Brake-by-Wire System. World Electric Vehicle Journal, 14(7), 177.

[15] Tristano, M., Lenzo, B., Xu, X., Forrier, B., D'hondt, T., Risaliti, E. & Wilhelm, E. Real-time implementation of yaw rate and sideslip control through individual wheel torques. *2022 IEEE Vehicle Power And Propulsion Conference (VPPC)*. pp. 1-6 (2022)

[16] De Novellis, L., Sorniotti, A. & Gruber, P. Driving modes for designing the cornering response of fully electric vehicles with multiple motors. *Mechanical Systems And Signal Processing*. **64** pp. 1-15 (2015)

[17] Mangia, A., Lenzo, B., & Sabbioni, E. (2021). An integrated torque-vectoring control framework for electric vehicles featuring multiple handling and energy-efficiency modes selectable by the driver. Meccanica, 56(5), 991-1010.

[18] Zhang, L., Ding, H., Guo, K., Zhang, J., Pan, W., & Jiang, Z. (2019). Cooperative chassis control system of electric vehicles for agility and stability improvements. IET Intelligent Transport Systems, 13(1), 134–140.

[19] Ono, E., Hosoe, S., Tuan, H. D., & Doi, S. (1998). Bifurcation in vehicle dynamics and robust front wheel steering control. IEEE Transactions on Control Systems Technology, 6(3), 412–420.

[20] Selby, M. Intelligent vehicle motion control. (University of Leeds,2003)

[21] Hao, Z., Xian-sheng, L., Shu-ming, S., Hong-fei, L., Rachel, G., & Li, L. (2011). Phase plane analysis for vehicle handling and stability. International Journal of Computational Intelligence Systems, 4(6), 1179–1186.

[22] Bobier-Tiu, C. G., Beal, C. E., Kegelman, J. C., Hindiyeh, R. Y., & Gerdes, J. C. (2019). Vehicle control synthesis using phase portraits of planar dynamics. Vehicle System Dynamics, 57(9), 1318–1337.

[23] Klomp, M. Graphical Methods for Road Vehicle System Dynamics Analysis. *The IAVSD International Symposium On Dynamics Of Vehicles On Roads And Tracks*. pp. 827-835 (2021)

[24] Guo, N., Zhang, X., Zou, Y., Lenzo, B., Du, G. & Zhang, T. A supervisory control strategy of distributed drive electric vehicles for coordinating handling, lateral stability, and energy efficiency. *IEEE Transactions On Transportation Electrification*. **7**, 2488-2504 (2021)

[25] Lugo, L., Bartolozzi, M., Vandermeulen, W., Geluk, T., & Dom, S. Test-driven full vehicle modelling for ADAS algorithm development (No. 2021-26-0033). SAE Technical Paper. (2021)

[26] https://blogs.sw.siemens.com/simcenter/simrod-experience-model-based-system-testing/

[27] Dhondt, T., Mollet, Y., Joos, A. J., Cecconi, L., Sarrazin, M., Gyselinck, J. Scalable Electric-motor-in-the-Loop Testing for Vehicle Powertrains. In ICINCO (pp. 594-603). (2020)

[28] https://fmi-standard.org/

[29] https://www.hydac.com/shop/en-gb/electronic-control-technology/controllers

[30] Klehmet, U., Herpel, T., Hielscher, K. S., German, R. Delay bounds for CAN communication in automotive applications. In 14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems (pp. 1-15). VDE. (2008)

Mariagrazia Tristano received her M.Sc. degree in Mechanical Engineering from Politecnico di Torino, Turin, Italy, in 2020. She is currently a Ph.D. student at Sheffield Hallam University, Sheffield, UK, in partnership with Siemens Digital Industries Software in Leuven, Belgium. Her research interests include vehicle dynamics, torque vectoring and state estimation.

Basilio Lenzo is a tenure-track Assistant Professor with the Department of Industrial Engineering, University of Padova, Padua, Italy. Before this appointment, he was Senior Lecturer in Automotive Engineering at Sheffield Hallam University, UK. He was also a Visiting Researcher with the Ecole Normale Superieure Cachan, France, the University of Delaware, USA, Columbia University, USA, the University of Naples, Italy, the German Aerospace Center DLR, Germany, Politecnico di Torino, Italy, Stanford University, USA. He was a two-time TEDx Speaker. His research interests include vehicle dynamics, torque vectoring, state estimation, control, and robotics.

Xu Xu received an MSc in Control Systems Engineering and subsequently a PhD covering the areas of nonlinear dynamical systems, cellular automata, optimal control and sliding mode control engineering at the University of Sheffield. She is currently a reader in Control Systems Engineering and Nonlinear Systems Modelling at Sheffield Hallam University. Xu actively researches fluid dynamics simulation methods for blood flow modelling. As a control engineer, she is also interested in nonlinear control techniques and related control engineering applications. Her current research interest further extends to theoretical studies of cellular automata as nonlinear dynamical systems and the dynamical behaviours existing in cellular systems.

Bart Forrier received the M.Sc. degree in mechanical engineering from KU Leuven, Leuven, Belgium, in 2011. After graduation, he first joined LMS Intl., Leuven, Belgium, and later worked in the Noise & Vibration Research Group at KU Leuven. In 2018 he obtained his Ph.D. in engineering sciences. Since then, he is a researcher at SISW NV, and a voluntary researcher at KU Leuven. His main research activities are in model-based system testing and virtual sensing, with a focus on mechatronic powertrain applications.

Thomas D'hondt received his M.Sc. degree in electromechanical engineering from Brussels Faculty of Engineering, Belgium, in 2016. Since then, he is a research engineer at Siemens Digital Industries Software. His main research activities focus on model-based system testing, with a focus on electric powertrain applications and automated vehicles.

Enrico Risaliti received his M.Sc. degree in mechanical engineering from Università degli Studi di Firenze, Florence, Italy, in 2014. He received his Ph.D. in Mechanical Engineering in 2019 from KU Leuven, Leuven, Belgium. His main research interests are mechanical system modelling and testing, vehicle dynamics and virtual sensing.

Erik Wilhelm Erik Wilhelm received the B.S. and M.S. degrees from the University of Waterloo, Canada, in 2007, the Dr.Sci. from ETH Zurich, Switzerland, in 2011, and the Ph.D. degree from MIT. His research interests include powertrain design, energy storage and conversion, optimal and robust control, applied machine learning, transportation systems, and pervasive sensing.