Software & Systems Modeling (2019) 18:1637–1653
https://doi.org/10.1007/s10270-017-0646-1

THEME SECTION PAPER

A model-based design approach for simulation and virtual prototyping of automotive control systems using port-Hamiltonian systems

Siyuan Dai[1] · Zhenkai Zhang[1] · Xenofon Koutsoukos[1]

Received: 1 March 2016 / Revised: 17 January 2017 / Accepted: 21 November 2017 / Published online: 8 December 2017

Abstract
Cyber–physical systems (CPS) such as automotive control systems consist of various interacting cyber and physical components. Heterogeneous domains, composition of multiple components, complex dynamics, and nonlinearities result in significant challenges for design, modeling, and simulation of CPS. Model-based design can be used to address such challenges, but it is very important to use physically accurate heterogeneous models that can be composed to represent the overall system behavior. Further, it is important to preserve the properties derived from analyses based on the mathematical models in the control system implementation in order to reduce costly testing and design changes late in the development cycle. This paper proposes a model-based design methodology for automotive control software using port-Hamiltonian systems (PHS). PHS are used to model the vehicle dynamics, speed and steering control systems, and the interactions between physical and cyber components. Passivity analysis is used to design the controllers and ensure system stability. More importantly, the proposed approach guarantees that passivity is preserved after time-discretization and quantization of the controllers. The models are then used for code generation and compilation, scheduling, and software deployment, ensuring that passivity is preserved by the control system implementation. We evaluate the methodology using an automotive control design case study implemented on a hardware-in-the-loop simulation platform and present simulation results to demonstrate its effectiveness.

Keywords Cyber–physical systems · Model-based design · Port-Hamiltonian systems · Passivity · Automotive control software

1 Introduction

Cyber–physical systems (CPS) are engineering systems characterized by complex integrations of physical and computational domains over communication networks [34]. Automotive control systems are examples of CPS in which various physical components are controlled by electronic control units (ECUs) communicating over networks. As the number of components increases in modern vehicles, challenges inevitably arise as a result of system complexity. Rigorous engineering methods are essential for designing control systems that can be integrated with the vehicle dynamics to achieve predictable and correct behavior [22]. Designing these systems requires the use of well-defined software tools and processes in order to model, simulate, analyze, and identify design flaws.

In our previous work, we developed a model-based design tool-chain for automotive control systems which integrates control design, software implementation, and deployment on a hardware-in-the-loop (HIL) simulation platform [12]. This tool-chain provides a means to design and implement control software on a realistic platform while focusing on the interactions that manifest during the integration stage of development. Although this model-based design tool-chain enables simulation and rapid prototyping of CPS, it does not address significant challenges caused by interactions between components, complex dynamics, and nonlinearities of the system models. Many methods rely on idea of “making it work,” and the control system implementation does not take full advantage of mathematical analyses using control

Communicated by Prof. J. Sztipanovits, M. Broy, and H. Daembkes.

✉ Siyuan Dai
siyuandai@gmail.com

Zhenkai Zhang
zhenkai.zhang@vanderbilt.edu

Xenofon Koutsoukos
xenofon.koutsoukos@vanderbilt.edu

[1] Institute for Software-Integrated Systems, Vanderbilt University, Nashville, TN, USA

theoretic methods. Such model-based design methods have worked well in the past; however, the increased complexity of modern automotive systems has made these methods implausible. Problems from the integration of components are often discovered late in the development cycle, leading to costly ramifications of the designs. As modern CPS become more complex, a model-based design framework is needed to effectively generate composable models which can be used not only for theoretic analysis, but also for software design and implementation while preserving the desirable system properties.

This paper presents a model-based design methodology based on port-Hamiltonian systems (PHS) for the modeling and design of CPS. The methodology begins with the physical system which is modeled as a PHS [11]. The system model is constructed by modeling both physical and cyber components (controllers) as PHS interacting through power ports modeled using Dirac structures. Passivity-based control methods are used to design the controllers and analyze system stability [26]. The modeling framework is applicable to complex systems that contain nonlinear and hybrid dynamics. In order to implement the controllers on a realistic platform, implementation effects such as discretization and quantization are analyzed to ensure that passivity is preserved. Code is generated from the discrete-time quantized PHS, which in turn, is analyzed for timing and scheduling, and finally deployed on the hardware platform. The proposed model-based design approach guarantees that system passivity and stability are preserved after time-discretization and quantization of the controllers. Further, the models are used for code generation and compilation, scheduling, and software deployment, ensuring that passivity is preserved by the control system implementation.

We present a case study where the model-based design methodology is applied to an automotive control system that integrates speed and steering control. The physical dynamics of a vehicle is decomposed into its longitudinal and lateral dynamics interacting through a Dirac structure and modeled as PHS. We model the speed and steering controllers also as PHS that interact with the vehicle dynamics and with each other to achieve autonomous driving. We compose the closed-loop system, and we provide conditions that guarantee stability based on the hybrid nonlinear models of the components. Next, we ensure that the control systems remain passive after discretization and quantization that is required for the control system implementation. The discrete-time quantized models are used for code generation using MATLAB/Simulink [23]. The generated code is deployed on a HIL simulation platform consisting of multiple ECUs communicating with a real-time simulation of the vehicle dynamics using CarSim [2]. The communication is realized using the time-triggered network TTEthernet [35]. Simulation results generated using the HIL platform show how the system behavior is affected by varying the sampling rate and the quantization levels and demonstrate the stability of the system under various driving conditions.

The rest of the paper is organized as follows: Sect. 2 presents the related work of this paper including domain-specific modeling languages, PHS theory, and model-based design. Section 3 describes the model-based design process starting from the modeling of physical systems to the implementation of the closed-loop system on a HIL simulation platform. Section 4 presents the physical system modeling framework using a domain-specific modeling language (DSML) based on PHS. Section 5 presents the control portion of the automotive system implemented using continuous-time PHS. Section 6 describes the control system implementation and includes the constraints imposed on the design by discretization and quantization as well as the processes of code generation and compilation from the Simulink models. In addition, this section presents the computation of the WCET and the scheduling of the control tasks. Section 7 presents the deployment of the control software on the HIL simulation platform and the simulation results that demonstrate control design and implementation. The paper is concluded in Sect. 8.

2 Related work

Component-based modeling of CPS is a challenging problem because of the inherent heterogeneity within CPS [33]. Components are implemented as well-defined models of computation, and abstract semantics are used to define interactions between them [8]. The intricacies of both acausal and causal modeling can be captured using a DSML [31]. Software tools typically formalize interactions between the physical and computational components through unidirectional physical signals and computational signals. Research in recent years has focused on using the exchange of energy to merge the abstractions between the computational and physical domains [22].

The theory of PHS is presented in detail in [11]. PHS are formulated through generalized bond graphs, which are domain-independent graphical representations based on energy exchange. Nodes in a bond graph represent physical phenomena, and edges (bonds) represent idealized energy connections. Different physical domains such as electrical, mechanical rotational, hydraulic, and thermodynamics are all described in the same way through a physical exchange of energy. Its major distinction from representations such as signal-flow graphs is that every arc (bond) represents a bidirectional exchange of energy, rather than a unidirectional flow of information. PHS have significant implications for passivity, which has been studied extensively for control design and analysis of nonlinear systems [20]. Background on the control of PHS is presented in detail in [25]. An important property of PHS is compositionality, where component PHS compose with each other through the interaction ports of their respective Dirac structures [42]. The result of composing two PHS is another PHS, which provides a compositional framework for modeling complex physical lumped-parameter systems [3].

Automotive Open System Architecture (AUTOSAR) is a worldwide development partnership with the goal of establishing an open-source and standardized software architecture for automotive ECUs. AUTOSAR contains details which describe standardized software modules, define how application software components interface with one another, and build a universal design approach using a standardized exchange format. These details allow AUTOSAR to help innovate automotive electronic systems to further improve the performance, safety, and efficiencies of modern vehicles. The scalability of different software and hardware components over the vehicle product life cycle is also facilitated by AUTOSAR. AUTOSAR strives to assist in the design and development of future vehicles and to improve the cost-efficiency of integrating new technologies [21].

Model-based design of CPS is an active research area where a large amount of work is being done to address the various challenges caused by the heterogeneity of the different layers of CPS [34]. We created a tool-chain to integrate the components of control design from MATLAB/Simulink with the aspects of software deployment such as scheduling, discretization, and quantization with a primary application to automotive control systems [12]. The tool-chain is designed using an embedded software design environment called Embedded Systems Modeling Language (ESMoL) [27] which enables a software development process unifying the control design stage of development with code generation and deployment. The tool-chain is evaluated over a hardware-in-the-loop experimental platform over a time-triggered communication network which guarantees that the whole process is reliable, predictable, and robust to disturbances [41].

The first step of the model-based design tool-chain is designing and modeling the controller for a particular function in MATLAB/Simulink and using simulations to test and verify the correctness of the system. The methodology we developed in this paper focuses on this step of the model-based design process by modeling both the controller and plant as PHS, and using passivity-based design methods to ensure correct behavior of the overall system. PHS provide an effective way of characterizing the interactions presented in the CPS and are also able to model nonlinearities and hybrid dynamics. The resulting control design model from this approach can then be imported into ESMoL and used for subsequent phases of the model-based design process. The remaining design steps consist of importing the control designs into ESMoL, specifying the logical software architecture, defining the hardware platform, deploying the model, specifying the timing behavior of the system, flattening the model for analysis, scheduling the tasks, and implementing on the platform for testing [12].

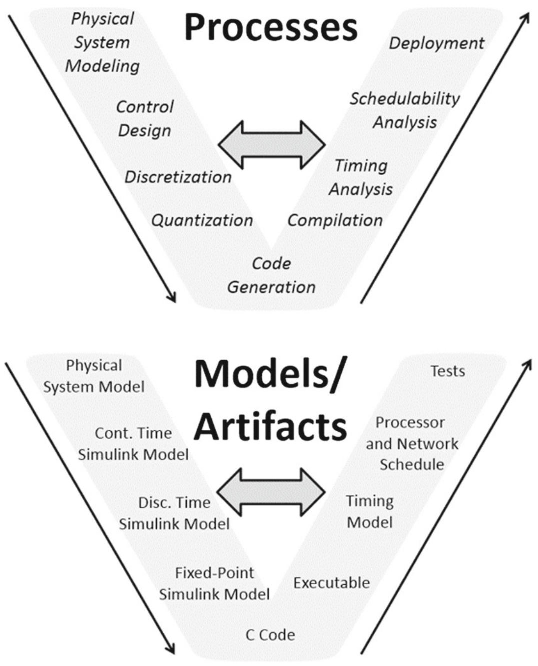

Fig. 1 Diagram of the processes and artifacts

In our previous work, we developed a PHS modeling framework which addresses the challenges of implementing components and interactions for model-based design [6]. The work presented in this paper is an extension of the work on the PHS modeling framework by focusing on an end-to-end methodology for the modeling and design of CPS. Finally, we have developed an approach for safety analysis of PHS and apply the approach for ensuring safety of automotive control systems [7].

3 Model-based design process

Figure 1 shows a diagram of the flow of processes of the model-based design process and the corresponding flow of models/artifacts. The left branch of each diagram focuses on design steps using models, while the right branch of each diagram focuses on implementation steps on a hardware platform. The model and software artifact generated from each step of the process must be analyzed in order to validate correctness and the effect that each process step has on the overall system properties. For example, the discretization and quantization of a continuous-time controller affects the stability of the closed-loop system. Therefore, it is necessary to analyze (1) how close is the discrete-time quantized model to the original controller and (2) the stability of the system composed of the physical dynamics and the controller. Further, there are several dependencies between the left and right branches of the processes and models/artifacts. For example, schedulability analysis may impose constraints on the sampling rate that is feasible in the hardware platform, which in turn affects the discretization of the controller and its stability analysis.

The main goal of the proposed model-based design approach is to develop a framework that allows to analyze the effects of the design steps on system properties such as stability. We employ a modeling framework based on PHS using GME, a software developed at Vanderbilt University, which allows us to compose complex systems model from components [11]. We design the continuous-time controllers also as PHS using Simulink and analyze stability using passivity methods [26]. We generate discrete-time quantized controller models using Simulink's code generation tool that ensure passivity, and we use them to generate the control code. We estimate the worst-case execution time (WCET) of the control code running on an ECU, which is used to generate static schedules for the ECUs and the network. We select control parameters, sampling rate, and quantization levels so that passivity and system stability are guaranteed. Finally, we test the control design using a HIL simulation platform. The main advantage of the proposed method is that it guarantees passivity and stability through the design steps. The rest of the paper is organized based on the flow of the diagram in Fig. 1.

4 Physical system modeling

The use of models as an initial step in the development of complex systems is prevalent in all fields of science and engineering. A model is an abstraction of occurrences in the physical world. In engineering, models provide a way of representing systems using a specific set of mathematical constructs. The objective of physical system modeling is to represent the free-body diagram of a physical system as a PHS model implemented in Simulink. The Simulink model is then used for control design. This necessitates the development of a DSML which contains the rules and constraints of PHS.

4.1 Background on port-Hamiltonian systems

Physical systems are generally described as a set of differential and algebraic equations, since algebraic constraints between state variables are possible [30]. Though implicit equations create problems in simulation, they generally have no effect on mathematical formulations. In particular, PHS model both implicit and explicit systems through the Dirac structure [37]. An important component of PHS is the characterization of power. Power consists of a pair of conjugated variables, the effort e and the flow f, whose product is the instantaneous power. For example, in the electrical circuit domain, the effort is the voltage, while the flow is the current.

Definition 1 [11] A Dirac structure on $F \times F^*$ with efforts $e \in F$ and flows $f \in F^*$ is a subspace $D \subset F \times F^*$ if the following two conditions are true:

1. $\langle e, f \rangle = 0$, for all $(e, f) \in D$,
2. $\dim[D] = \dim[F]$.

There are various ways to represent Dirac structures, with each representation using different structural matrices. Regardless of the representation, Dirac structures establish the power-balancing equation which is fundamental to PHS:

$$\frac{\mathrm{d}}{\mathrm{d}t} H = e_R^{\mathsf{T}} f_R + e_C^{\mathsf{T}} f_C + e_I^{\mathsf{T}} f_I. \tag{1}$$

The function H in (1) denotes the Hamiltonian of the PHS. The Hamiltonian function represents the energy stored in the system. The flows of the energy storage are given by the rate $\dot{x}$ of the energy state variables x; the efforts are given by the co-energy variables $\frac{\partial H}{\partial x}$ [16]. The variables e_R and f_R are the effort and flow values for the energy dissipation of the system, which shows that a PHS extends beyond a conservative Hamiltonian system by including energy loss as well [11]. The variables e_C and f_C describe how the system interacts with controllers, while the variables e_I and f_I describe how the system interacts with the environment. A PHS with a Hamiltonian function H, energy storage ports S, resistive ports R, control ports C, interconnection ports I, and a Dirac structure D can be written in a formal model in (2),

$$\Sigma = (H, R, C, I, D), \tag{2}$$

which leads to the implicitly defined PHS dynamics:

$$\Sigma : \left(-x'(t), \frac{\partial H}{\partial x}, f_R(t), e_R(t), f_C(t), e_C(t), f_I(t), e_I(t) \right) \in D. \tag{3}$$

When there are no algebraic constraints on the state variables, representation (3) can be simplified to an input–state–output PHS [10]:

$$\Sigma : \begin{cases} \dot{x} = [J(x) - R(x)]\frac{\partial H}{\partial x} + G(x)u + K(x)d \\ y = G^{\mathsf{T}}(x)\frac{\partial H}{\partial x} \\ z = K^{\mathsf{T}}(x)\frac{\partial H}{\partial x}, \end{cases} \tag{4}$$

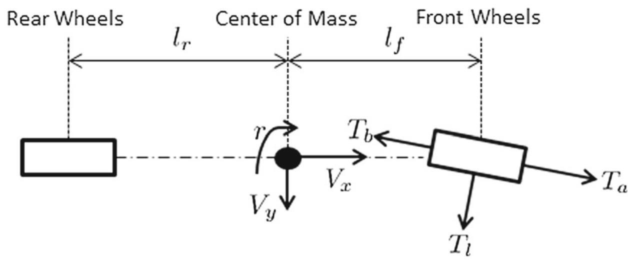

Fig. 2 Free-body diagram of the vehicle dynamics

and its Dirac structure is represented as the following skew-symmetric matrix:

$$D:\begin{bmatrix} -J(x) & -G_R(x) & -G(x) & -K(x) \\ G_R^{\mathsf{T}}(x) & 0 & 0 & 0 \\ G^{\mathsf{T}}(x) & 0 & 0 & 0 \\ K^{\mathsf{T}}(x) & 0 & 0 & 0 \end{bmatrix},$$

where the term $G_R(x)$ denotes the resistive structure of the system.

A PHS obeys the laws of conservation of energy. The canonical coordinate transform method is used extensively in classical mechanics for analyzing the dynamical equations of physical systems [13]. These transformations preserve the Hamiltonian structure of the system and preserves important system properties such as losslessness and passivity. Consider a generic PHS written in the form of (4). We then consider a time-invariant coordinate transformation defined by the expression $\overline{x} = \Phi(x)$ and apply it to (4). The dynamical equations transform into the following form

$$\begin{aligned}\dot{\overline{x}} &= \frac{\partial \Phi}{\partial x}^{\mathsf{T}} \dot{x} \\ &= \frac{\partial \Phi}{\partial x}^{\mathsf{T}} [J(x) - R(x)] \frac{\partial H}{\partial x} + \frac{\partial \Phi}{\partial x}^{\mathsf{T}} G(x)u \\ &= \frac{\partial \Phi}{\partial x}^{\mathsf{T}} [J(x) - R(x)] \frac{\partial \Phi}{\partial x} \frac{\partial H(\Phi^{-1}(\overline{x}))}{\partial \overline{x}} + \frac{\partial \Phi}{\partial x}^{\mathsf{T}} G(x)u,\end{aligned}$$

and the output equation transforms into the following form

$$y = G^{\mathsf{T}}(x) \frac{\partial \Phi}{\partial x} \frac{\partial H(\Phi^{-1}(\overline{x}))}{\partial \overline{x}}.$$

The new Hamiltonian function becomes $H(\Phi^{-1}(\overline{x}))$. The matrices $\frac{\partial \Phi}{\partial x}^{\mathsf{T}} J(x) \frac{\partial \Phi}{\partial x}$ and $\frac{\partial \Phi}{\partial x}^{\mathsf{T}} R(x) \frac{\partial \Phi}{\partial x}$ are skew-symmetric and positive symmetric, respectively, which means that the coordinate-transformed system is also a PHS [36]. Generally, the coordinate transform is used whenever two PHS are composed [14].

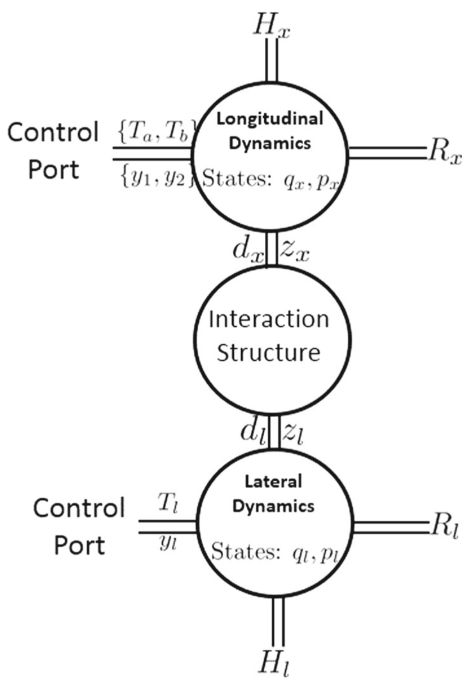

Fig. 3 PHS representation of the vehicle dynamics

4.2 Vehicle dynamics modeling

In this paper, we demonstrate the model-based design methodology using an automotive system as an example. Figure 2 shows a free-body diagram of the vehicle dynamics. The vehicle is front-wheel drive, resulting in the input forces from throttle and brake being applied to the front wheels as shown in the diagram. The longitudinal input force from the throttle, $T_a = C_a\theta_a$, is a function of the throttle valve angle θ_a and the experimental throttle constant C_a. The longitudinal input force from the brakes, $T_b = C_b P_b$, is a function of the braking pressure P_b and the experimental braking constant C_b. The lateral input force from the steering, $T_l = 2C_f\delta$, is a function of the steering angle δ and the cornering stiffness of the front wheels C_f. The longitudinal velocity, lateral velocity, and yaw rate are represented by V_x, V_y, and r, respectively. Interactions between the longitudinal and lateral dynamics can be derived by analysis of the free-body diagram [29].

A PHS representation of the longitudinal dynamics, the lateral dynamics, and their interactions is shown in Fig. 3. The vehicle dynamics model is decomposed into a longitudinal dynamics component, a lateral dynamics component, and an interaction structure. The longitudinal dynamics contain two control ports (T_a, y_1) and (T_b, y_2) and an interaction port (d_x, z_x). The state variables are the longitudinal momentum p_x and the longitudinal displacement q_x. The outputs of the control ports y_1 and y_2 are V_x and $-V_x$, respectively. The x-component of the lateral force affecting longitudinal motion is represented by d_x, and its power-conjugate is represented by z_x. The lateral dynamics contain a control port (T_l, y_l) and an interaction port (d_l, z_l). The state variables are $q_l = \begin{bmatrix} q_y & q_r \end{bmatrix}^\mathsf{T}$ and $p_l = \begin{bmatrix} p_y & p_r \end{bmatrix}^\mathsf{T}$, where p_y is the lateral momentum, p_r is the angular momentum, q_y is the lateral displacement, and q_r is the angular displacement. The output of the control port y_l is $V_y + l_f r$. The y-component of the longitudinal force applied to the center of mass is represented by d_l, and its power-conjugate is represented by z_l.

4.2.1 Longitudinal dynamics

The longitudinal dynamics has the following Hamiltonian function:

$$H_x(q_x, p_x) = \frac{1}{2m} p_x^2 + U_x(q_x),$$

where m represents the mass of the vehicle and $U_x(q_x)$ represents the potential energy. The longitudinal dynamics is modeled in the form of (4):

$$\begin{cases} \begin{bmatrix} \dot{q}_x \\ \dot{p}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -R_x \end{bmatrix} \begin{bmatrix} \frac{\partial H_x}{\partial q_x} \\ \frac{\partial H_x}{\partial p_x} \end{bmatrix} + \begin{bmatrix} 0 \\ G_x \end{bmatrix} u_x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d_x \\ y_x \;\; = \begin{bmatrix} 0 & G_x^\mathsf{T} \end{bmatrix} \begin{bmatrix} \frac{\partial H_x}{\partial q_x} & \frac{\partial H_x}{\partial p_x} \end{bmatrix}^\mathsf{T} \\ z_x \;\; = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial H_x}{\partial q_x} & \frac{\partial H_x}{\partial p_x} \end{bmatrix}^\mathsf{T}, \end{cases} \tag{5}$$

where $u_x = \begin{bmatrix} T_a & T_b \end{bmatrix}^\mathsf{T}$, $G_x = \begin{bmatrix} 1 & -1 \end{bmatrix}$, $R_x = am + bp_x + \frac{cm^2}{p_x}$, a represents the tire rolling friction constant, b represents the air resistance constant, and c represents the static friction force.

4.2.2 Lateral dynamics

The lateral dynamics has the following Hamiltonian function:

$$H_l(q_y, q_r, p_y, p_r) = \frac{1}{2m} p_y^2 + \frac{1}{2I} p_r^2 + U_l(q_y, q_r),$$

where I represents the moment of inertia of the vehicle and $U_l(q_y, q_r)$ represents the potential energy. The lateral dynamics is modeled in the form of (4):

$$\begin{cases} \begin{bmatrix} \dot{q}_l \\ \dot{p}_l \end{bmatrix} = \begin{bmatrix} 0 & E \\ -E & -R_l \end{bmatrix} \begin{bmatrix} \frac{\partial H_l}{\partial q_l} \\ \frac{\partial H_l}{\partial p_l} \end{bmatrix} + \begin{bmatrix} 0 \\ G_l \end{bmatrix} T_l + \begin{bmatrix} 0 \\ K_l \end{bmatrix} d_l \\ y_l \;\; = \begin{bmatrix} 0 & G_l^\mathsf{T} \end{bmatrix} \begin{bmatrix} \frac{\partial H_l}{\partial q_l} & \frac{\partial H_l}{\partial p_l} \end{bmatrix}^\mathsf{T} \\ z_l \;\; = \begin{bmatrix} 0 & K_l^\mathsf{T} \end{bmatrix} \begin{bmatrix} \frac{\partial H_l}{\partial q_l} & \frac{\partial H_l}{\partial p_l} \end{bmatrix}^\mathsf{T}, \end{cases} \tag{6}$$

$$R_l = \begin{bmatrix} \frac{W_1}{V_x} & \frac{W_2}{V_x} \\ \frac{W_2}{V_x} & \frac{W_3}{V_x} \end{bmatrix},$$

where E is the identity matrix, $G_l = \begin{bmatrix} 1 & l_f \end{bmatrix}^\mathsf{T}$, and $K_l = \begin{bmatrix} 1 & 0 \end{bmatrix}^\mathsf{T}$. The parameter constants of R_l are $W_1 = 2C_f + 2C_r$, $W_2 = 2C_f l_f - 2C_r l_r$, and $W_3 = 2C_f l_f^2 + 2C_r l_r^2$, where C_r is the cornering stiffness of the rear wheels.

4.2.3 Vehicle dynamics interactions

Composition of the longitudinal and lateral dynamics through the interaction structure results in a nonlinear Dirac structure for the composed dynamics. The interaction between the longitudinal and lateral dynamics is a mapping of velocity to force, which indicates a gyrator relationship. The gyrator ratio must have units of kg/s which is represented by multiplying the mass of the vehicle with the yaw rate. The interaction structure is modeled as a Dirac structure modulated by the angular momentum p_r:

$$\begin{bmatrix} d_x \\ d_l \end{bmatrix} = \begin{bmatrix} 0 & -\frac{mp_r}{I} \\ -\frac{mp_r}{I} & 0 \end{bmatrix} \begin{bmatrix} z_x \\ z_l \end{bmatrix}. \tag{7}$$

The Hamiltonian function of the composed longitudinal and lateral dynamics is $H(q, p) = H_x + H_l$. Composition of (5) and (6) through (7) results in the following nonlinear PHS:

$$\begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & E \\ -E & -R \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{bmatrix} + \begin{bmatrix} 0 \\ G \end{bmatrix} u \\ y \;\; = \begin{bmatrix} 0 & G^\mathsf{T} \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial p} \\ \frac{\partial H}{\partial p} \end{bmatrix}, \end{cases} \tag{8}$$

$$R = \begin{bmatrix} R_x & \frac{mp_r}{I} & 0 \\ \frac{mp_r}{I} & \frac{mW_1}{p_x} & \frac{mW_2}{p_x} \\ 0 & \frac{mW_2}{p_x} & \frac{mW_3}{p_x} \end{bmatrix}, \quad G = \begin{bmatrix} G_x & 0 \\ 0 & G_l \end{bmatrix},$$

where $q = \begin{bmatrix} q_x & q_l \end{bmatrix}^\mathsf{T}$, $p = \begin{bmatrix} p_x & p_l \end{bmatrix}^\mathsf{T}$, $u = \begin{bmatrix} u_x & T_l \end{bmatrix}^\mathsf{T}$, and $y = \begin{bmatrix} y_x & y_l \end{bmatrix}^\mathsf{T}$. Interactions between the longitudinal and lateral dynamics are captured in R.

4.3 Port-Hamiltonian system modeling language

This section presents a domain-specific modeling language (DSML) based on PHS and uses principles of model-integrated computing [19]. In order to accurately model the interactions of different components in the systems, ports are used as interfaces between different elements and submodels [38]. Figure 4 depicts a PHS, where a set of ports (control, interaction, resistive, and storage) are interconnected through a Dirac structure [11]. PHSML is developed using Generic Modeling Environment (GME), based on the model-integrated computing tool suite developed at

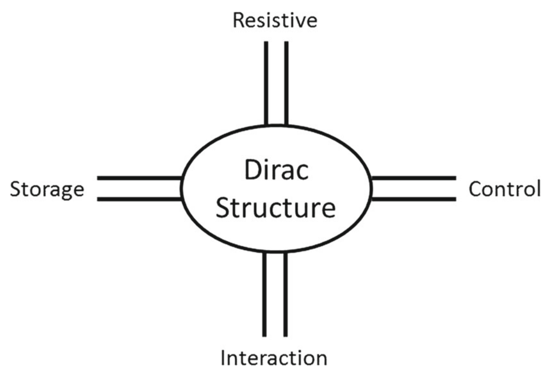

Fig. 4 Port-Hamiltonian system [11]

Institute for Software Integrated Systems at Vanderbilt University [15]. GME provides a meta-modeling environment to create a DSML and allows for the construction of domain models through a collection of objects and connectors defined for a specific domain. Hierarchical composition is supported through model elements.

PHSML encodes all modeling and connection rules specific for PHS. The model is developed in GME as a PHSML model. A model translator, i.e., interpreter, processes the structural information of the PHSML model and creates a PHS model characterized by Dirac structures and constituent equations of ports. The top-level hierarchy of PHSML is called component assembly that consists of components, which serve as the objects, and bonds, which serve as the connections. A component is defined as an object consisting of a Dirac structure, a set of internal ports, and a set of external ports. The most important modeling element to each component is the Dirac structure, which describes the power-conserving interconnection structure of each component. The ports (which are also denoted as atoms) have dynamics which relates their power-conjugate variables through constituent equations. The equation attribute allows the user to specify the constituent equations of the port with respect to the power-conjugate variables, effort and flow.

Once a model is created out of PHSML, an interpreter is used to generate the equations of the model. The first part of the interpreter is Algorithm 1 which transforms the PHS system into a set of acausal equations describing the Dirac structure.

Algorithm 1 Dirac Structure Generation

```
-for all Components
-  for all Power-conserving elements
-    Find E and F based on pattern;
-  while Interconnections left
-    Substitute Dirac structure equations to form combined Dirac structure;
```

However, acausal equations are not conducive to simulation, which is why we need to generate the causal equations, using Algorithms 2:

Algorithm 2 Equation Generation

```
-for all Components
-  Swap all source of flow columns;
-  Remove all resistive columns;
-  while There are still unassigned interactions
-    Find component with greatest interaction to constraint ratio;
-    Propagate interaction assignment;
-  Put back all resistive columns;
-  Determine which resistive columns to interchange;
-  Swap said resistive columns to obtain full ranked $F$;
-  $J \leftarrow -F^{-1}E$;
```

The most important goal of PHSML is to ensure that models are valid PHS. Constraints are implemented in the PHSML meta-model to ensure that certain interconnections between PHSML modeling elements, which will violate the underlying property of PHS, are not allowed. There are two types of constraints: (a) directional connections and connections between modeling types and (b) a model transformation software component defines a set of rules, which require a specific number of connections for each object. A valid PHS is required in order for Algorithms 1 and 2 to run and generate the equations of the model. Once the equations are generated, we can use MATLAB/Simulink to simulate the system. Additional information about PHSML can be found in [5].

4.4 Model validation

Validation of the analytical vehicle dynamics model is important for subsequent steps of the model-based design methodology. We use the CarSim S-function of a mid-size sedan for the HIL platform as the actual model [2]. Passivity indexes allow a way to characterize a system by determining its excess or shortage of passivity [40]. By selecting parameters so that the passivity indexes of the analytical models are similar to that of the CarSim model, we can conclude that the analytical models are reasonable approximations of the actual vehicle dynamics. The CarSim model has inherent bounds on its inputs [2]. The throttle angle valve (θ_f) has a lower bound of 0 and an upper bound of 1.5. The brake pressure (P_b) has a lower bound of 0 and upper bound of 10. The steering angle (δ) has a lower bound of -480 and an upper bound of 480 [2]. Using these CarSim variable bound values we mathematically determine that T_a has a lower bound of 0 N and an upper bound of 3104 N, T_b has a lower bound of 0 N and an upper bound of 3715 N, and T_l has a lower bound of -1200 N and an upper bound of 1200 N.

We experimentally selected values for the vehicle model (Table 1) so that the passivity indexes of the analytical model

Table 1 Table of vehicle parameter values

a	b	c	C_r	l_r	C_f	l_f
0.1	0.006	10	200	1.4	300	1.4

closely match that of the CarSim model by running both models through twenty scenarios and optimizing the passivity index values using the method in [17]. The CarSim model gives the mass ($m = 1650$) and the inertia ($I = 3234$). Using the techniques demonstrated in [39], we determine that the passivity indexes of the CarSim model (ν_c, ρ_c) are (181, 0.6). We determined that the passivity indexes of the analytical model (ν_a, ρ_a) are (177, 0.6), indicating that the analytical model is a reasonable approximation of the CarSim model.

5 Continuous-time control design

Given the model of a physical system implemented as a PHS, we can design controllers which enable the closed-loop system to behave correctly. Our objective is to design controllers modeled as PHS which interface with the physical system through designated power ports, thereby regulating the behavior of the closed-loop system. We use passivity analysis in order to ensure that the closed-loop system remains stable, is minimum-phased, and has a low relative degree [20]. The Simulink model generated from the controller PHS is used as the initial control design for the HIL platform.

A high-level system model of the vehicle dynamics interacting with the controllers (consisting of a speed control and steering control) is shown in Fig. 5. The controllers are implemented as PHS, and they interact with the vehicle dynamics through the power ports of T_a, T_b, and T_l, which were previously defined in the beginning of Sect. 4. Transformation of PHS into Simulink is a relatively simple procedure because the PHS equations are written in a format similar to state-space representation. State variables and subsequent computations are linked together through integrators and adders.

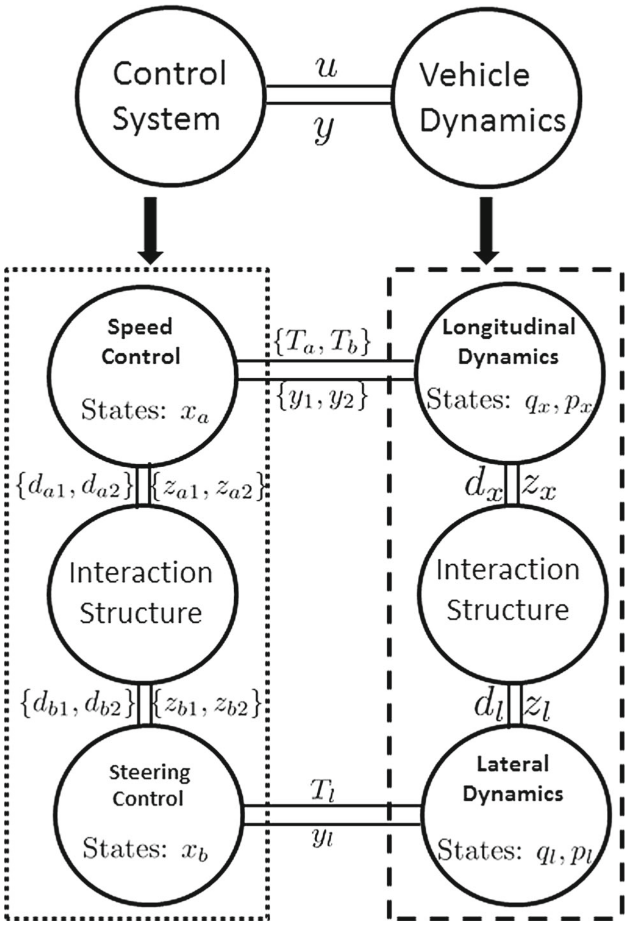

Fig. 5 PHS representation of the controllers interacting with vehicle dynamics

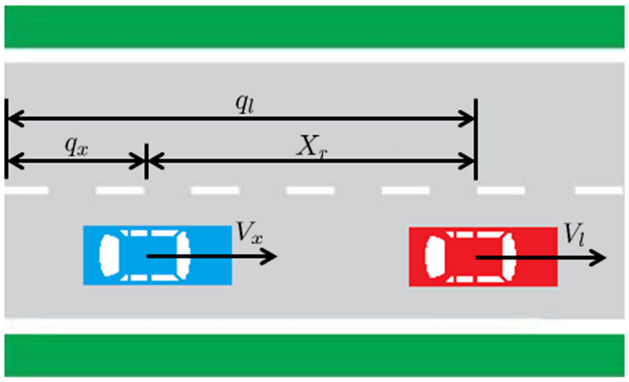

Fig. 6 Lead vehicle and host vehicle on a straight road

5.1 Automotive control design

The objective of the controllers is to maintain a desired speed V_d and lateral displacement q_d. The controller model consists of a speed control component, a steering control component, and an interaction structure. The speed control shares the two control ports with the longitudinal dynamics and contains two interaction ports (d_{a1}, z_{a1}) and (d_{a2}, z_{a2}). Its state variables $x_a = \begin{bmatrix} x_{at} & x_{ab} \end{bmatrix}^{\mathsf{T}}$ are derived using the desired speed, where $x_{at} = \int_{t_0}^{t}(V_x - V_d)\mathrm{d}\tau$ and $x_{ab} = \int_{t_0}^{t}(V_d - V_x)\mathrm{d}\tau$. The steering control shares the control port with the lateral dynamics and contains two interaction ports (d_{b1}, z_{b1}) and (d_{b2}, z_{b2}). Its state variable $x_b = q_y - q_d$ is derived using the desired lateral displacement.

5.1.1 Speed control

The control objective of the speed control system is to prevent the host car from colliding into the lead car by maintaining a safe distance between the vehicles and a desired speed. For simplicity we consider the case shown in Fig. 6 in which the vehicles are driving on a straight road, which allows us to disregard the lateral dynamics. We design the speed control to have the following Hamiltonian function:

$$H_a(x_a, s) = \frac{1}{2}\left(s_t k_{ti} x_{at}^2 + s_b k_{bi} x_{ab}^2\right),$$

where k_{ti} and k_{bi} are the gains of the Hamiltonian. The discrete variables $s = (s_t, s_b) \in \{0, 1\}$ are used to model the hybrid dynamics of throttling and braking. The switching dynamics are defined in (9), where h_+ and h_- are hysteresis constants introduced to prevent the system from rapidly alternating between accelerating and decelerating, and X_r and X_d are the relative distance between the two vehicles and the desired distance, respectively:

$$\begin{cases} (s_t, s_b) = (1, 0) & \text{for } V_d - y_1 \geq 0,\ X_r \geq h_+ X_d, \\ (s_t, s_b) = (0, 1) & \text{for } V_d - y_1 < 0,\ X_r < h_- X_d. \end{cases} \tag{9}$$

We design the speed control as an input–state–output PHS with direct-feed-through which is a modified version of (4) [38]:

$$\begin{cases} \dot{x} = [J(x) - R(x)]\frac{\partial H}{\partial x} + G(x)u + K_1(x)d_1 \\ y = G^{\mathsf{T}}(x)\frac{\partial H}{\partial x} + [M(x) + S(x)]u + K_2(x)d_2 \\ z = \begin{bmatrix} K_1(x)^{\mathsf{T}} & 0 \\ 0 & K_2(x)^{\mathsf{T}} \end{bmatrix}\begin{bmatrix} \frac{\partial H}{\partial x} \\ u \end{bmatrix}, \end{cases} \tag{10}$$

where $M(x) \in \mathbb{R}^{m\times m}$ is a skew-symmetric interconnection matrix and $S(x) \in \mathbb{R}^{m\times m}$ is a symmetric positive semi-definite damping matrix. We design the speed control in the form of (10) because the feed-through term helps to ensure zero steady-state error [11]:

$$\begin{cases} \dot{x}_a = -R_a\frac{\partial H_a}{\partial x_a} + G_a y_x + K_{a1}d_{a1} \\ u_x = G_a^{\mathsf{T}}\frac{\partial H_a}{\partial x_a} + S_a y_x + K_{a2}d_{a2} \\ \begin{bmatrix} z_{a1} \\ z_{a2} \end{bmatrix} = \begin{bmatrix} K_{a1}^{\mathsf{T}} & 0 \\ 0 & K_{a2}^{\mathsf{T}} \end{bmatrix}\begin{bmatrix} \frac{\partial H_a}{\partial x_a} \\ y_x \end{bmatrix}, \end{cases} \tag{11}$$

where $K_{a1} = [1\ 0]^{\mathsf{T}}$ and $K_{a2} = [1\ 0]^{\mathsf{T}}$. The parameter matrices are denoted as:

$$R_a = \begin{bmatrix} s_t k_t & 0 \\ 0 & s_b k_b \end{bmatrix},\quad G_a = \begin{bmatrix} s_t P & 0 \\ 0 & s_b \end{bmatrix},$$

$$S_a = \begin{bmatrix} s_t k_{td} & 0 \\ 0 & s_b k_{bd} \end{bmatrix},$$

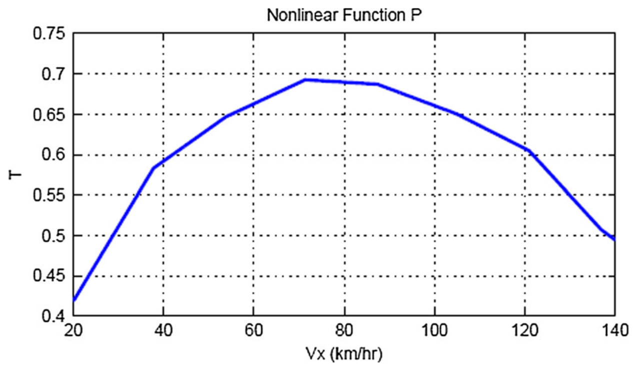

Fig. 7 Nonlinear function P

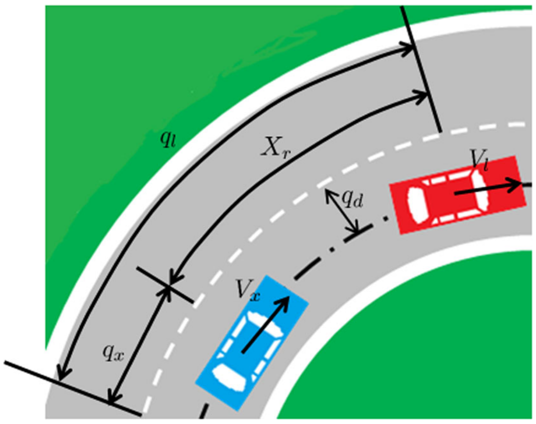

Fig. 8 Diagram of lead vehicle and host vehicle on a curved road

where k_t and k_{td} are throttle control gains. k_b and k_{bd} are brake control gains. P is derived from the inverse engine map for the vehicle and is a mapping of the ratio of the acceleration force to V_x (Fig. 7) [12].

5.1.2 Steering control

In this section we consider the control objectives of a vehicle with both speed and steering control. In addition to maintaining a safe distance between it and the lead vehicle, the host car must also maintain a reasonable lateral acceleration as to not spin off the road (see Fig. 8). This is a result of interactions between the lateral and longitudinal dynamics, characterized in (7). We consider the case shown in Fig. 8 in which the lead vehicle and host vehicle are driving on a curved road. We design the steering control to have the following Hamiltonian function:

$$H_b(x_b) = \frac{1}{2}k_{si}x_b^2,$$

where k_{si} is the gain of the Hamiltonian. We design the steering control in the form of (10):

$$\begin{cases} \dot{x}_b &= y_l + d_{b1} \\ T_l &= \frac{\partial H_b}{\partial x_b} + k_{sd} y_l + d_{b2} \\ \begin{bmatrix} z_{b1} \\ z_{b2} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial H_b}{\partial x_b} \\ y_l \end{bmatrix}, \end{cases} \tag{12}$$

where k_{sd} is the steering control gain.

5.1.3 Controller interactions

It can be seen from (8) that the inputs to the longitudinal dynamics (T_a and T_b) affect the lateral dynamics. Similarly, the input to the lateral dynamics (T_l) affects the longitudinal dynamics. This can create problems such that at high speeds, actions by the speed control may interfere with the objective of the steering control. In order to alleviate this problem, we introduce an interaction structure so that the state variables and outputs of the speed control are affected by the state variable of the steering control, and vice versa. Similar to (7), the interaction structure of the control system is represented with the following Dirac structure:

$$\begin{bmatrix} d_{a1} \\ d_{a2} \\ d_{b1} \\ d_{b2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & J_c & 0 \\ 0 & 0 & 0 & M_c \\ -J_c^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & -M_c^{\mathsf{T}} & 0 & 0 \end{bmatrix} \begin{bmatrix} z_{a1} \\ z_{a2} \\ z_{b1} \\ z_{b2} \end{bmatrix}. \tag{13}$$

The parameters J_c and M_c define how the speed control and the steering control interact. The Hamiltonian function of the composed control system is denoted as $H_c = H_a + H_b$. Composition of (11) and (12) through (13) results in the following PHS representation:

$$\begin{cases} \dot{x} = \begin{bmatrix} -R_a & J_s \\ -J_s^{\mathsf{T}} & 0 \end{bmatrix} \frac{\partial H_c}{\partial x} + \begin{bmatrix} G_a & 0 \\ 0 & 1 \end{bmatrix} y \\ u = \begin{bmatrix} G_a^{\mathsf{T}} & 0 \\ 0 & 1 \end{bmatrix} \frac{\partial H_c}{\partial x} + \begin{bmatrix} S_a & M_s \\ -M_s^{\mathsf{T}} & k_{sd} \end{bmatrix} y, \end{cases} \tag{14}$$

where $J_s = \begin{bmatrix} J_c & 0 \end{bmatrix}^{\mathsf{T}}$, $M_s = \begin{bmatrix} M_c & 0 \end{bmatrix}^{\mathsf{T}}$, and $x = \begin{bmatrix} x_{at} & x_{ab} & x_b \end{bmatrix}^{\mathsf{T}}$.

5.2 Passivity analysis

The passivity property of a system is defined in terms of its inputs and outputs. For a passive system, the energy stored in the system is equal to the external energy coming into the system subtracting the energy leaving the system and the energy dissipated in the system [20]. Regardless of the representation, Dirac structures establish the power-balancing equation which is fundamental to PHS.

$$\int_{t_0}^{t} e_I^{\mathsf{T}} f_I \mathrm{d}\tau + \int_{t_0}^{t} e_C^{\mathsf{T}} f_C \mathrm{d}\tau = H(x) - \int_{t_0}^{t} e_R^{\mathsf{T}} f_R \mathrm{d}\tau. \tag{15}$$

Equation (15) implies that the component is passive with respect to a supply energy of $\int_{t_0}^{t} e_I^{\mathsf{T}} f_I \mathrm{d}\tau + \int_{t_0}^{t} e_C^{\mathsf{T}} f_C \mathrm{d}\tau$ and dissipated energy of $\int_{t_0}^{t} e_R^{\mathsf{T}} f_R \mathrm{d}\tau$ as long as the storage function is positive definite.

Theorem 1 *The closed-loop system (14) is passive with respect to inputs y, outputs u, and Hamiltonian function* $H_c = H_a + H_b$ *if* $k_{ti}, k_{bi}, k_t, k_{td}, k_b, k_{bd}, k_{si}, k_{sd} \geq 0$. *Additionally, (14) will asymptotically stabilize the velocity* V_x *and the lateral position* q_y *to the desired velocity* V_d *and lateral position* q_d*, respectively.*

Proof Passivity of the composed control system is proven using the energy-balancing equation:

$$\frac{\mathrm{d}H_c}{\mathrm{d}t} = \frac{\partial H_c}{\partial x}^{\mathsf{T}} \begin{bmatrix} -R_a & J_c \\ -J_c^{\mathsf{T}} & J_l \end{bmatrix} \frac{\partial H_c}{\partial x} + \frac{\partial H_c}{\partial x}^{\mathsf{T}} \begin{bmatrix} G_a & 0 \\ 0 & G_l \end{bmatrix} y$$
$$\leq u^{\mathsf{T}} y - y^{\mathsf{T}} \begin{bmatrix} S_a & 0 \\ 0^{\mathsf{T}} & S_l \end{bmatrix} y.$$

Passivity of the system is shown by the inequality $\frac{\mathrm{d}H_c}{\mathrm{d}t} \leq u^{\mathsf{T}} y$. Asymptotic stability of the closed-loop system is shown by combining (8) with (14). The PHS representation of the closed-loop system is:

$$\begin{cases} \begin{bmatrix} \dot{q} \\ \dot{p} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -\tilde{M} & \tilde{M} & 0 \\ -\tilde{M}^{\mathsf{T}} & \tilde{J} & \tilde{K} \\ 0 & -\tilde{K}^{\mathsf{T}} & -\tilde{Q} \end{bmatrix} \begin{bmatrix} \frac{\partial \tilde{H}}{\partial q} \\ \frac{\partial \tilde{H}}{\partial p} \\ \frac{\partial \tilde{H}}{\partial x} \end{bmatrix}, \end{cases}$$

where $\tilde{M}$, $\tilde{J}$, $\tilde{K}$, and $\tilde{Q}$ are defined as:

$$\tilde{M} = \mathrm{diag}\left(\frac{1}{m}, \frac{1}{m}, \frac{1}{I}\right), \quad \tilde{J} = \begin{bmatrix} 0 & \frac{M_c}{m} & 0 \\ -\frac{M_c}{m}^{\mathsf{T}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\tilde{K} = \begin{bmatrix} s_t k_{ti} P & 0 & 0 \\ 0 & s_b k_{bi} & 0 \\ 0 & 0 & \frac{k_{si}}{l_f} \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} s_t k_t & -\frac{J_c}{m} & 0 \\ \frac{J_c}{m}^{\mathsf{T}} & s_b k_b & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

with a modified Hamiltonian function, $\tilde{H}(q, p, z)$:

$$\tilde{H} = \frac{s_t k_{td}}{2m}(m V_d - p_x)^2 + \frac{s_b k_{bd}}{2m}(p_x - m V_d)^2$$
$$+ \frac{1}{2m} p_y^2 + \frac{1}{2I} p_r^2 + \frac{k_{sd}}{2}(q_y - q_d)^2$$
$$+ \frac{s_t k_{ti}}{2} x_{at}^2 + \frac{s_b k_{bi}}{2} x_{ab}^2 + \frac{k_{si}}{2} x_b^2,$$

which we can use to verify that $\dot{\bar{H}}$ is always less than or equal to zero and that $p_x = mV_d$ and $q_y = q_d$ at the boundary. □

The proposed approach has established a compositional framework for modeling systems as interacting PHS. Further, passivity analysis is used to select the control parameters and prove stability of the closed-loop system. Next, our goal is to implement the continuous-time control design. First, we discretize and quantize the controllers while ensuring that passivity and, therefore, system stability are preserved. Then, we generate and deploy the control software, and we schedule the execution to ensure that passivity is preserved in the control system implementation.

6 Control system implementation

6.1 Discretization

Although the continuous-time control system models are useful for early stages of design and validation, they cannot be directly used in the HIL simulation platform. Given a continuous-time controller PHS, discretization is required to generate the C code for deployment on the ECUs in the platform. The objective of discretization is to convert the controllers from continuous-time PHS to discrete-time PHS. Prior to discretization, the continuous-time PHS is represented using block diagrams in a continuous-time Simulink model. Transformation of a continuous-time Simulink model into a discrete-time Simulink model is a procedure involving bilinear transformations, up-samplers, and down-samplers. State variables and subsequent computations inside the controllers are linked together through delays and adders. In this step we discretize the PHS controllers using sampling periods of 10, 30, and 50 ms. We employ discrete-time passivity in order to ensure that the closed-loop system will remain passive.

Discretization does not affect the high-level system model of the vehicle dynamics interacting with the controllers. However, passivity is a property that degrades under discretization [1,24]. Intuitively, the larger the sampling period, the greater the degradation. Figure 9 shows a visual representation of the parameter space for the control system being progressively restricted as the system has continuous-time passivity then discrete-time passivity imposed on itself as constraints. The standard discretization relates the discrete-time input $u_d(k)$ to the continuous-time input $u(t)$ using zero-order holds in that the continuous time t is bounded by kt_s and $(k+1)t_s$, where k is a nonnegative integer and t_s is the sampling period. The discrete-time output $y_d(k)$ relates to the continuous-time output $y(t)$ by a sampler $y_d(k) = y(kt_s)$. A crucial information here is that even if the original continuous-time system is a passive PHS, its

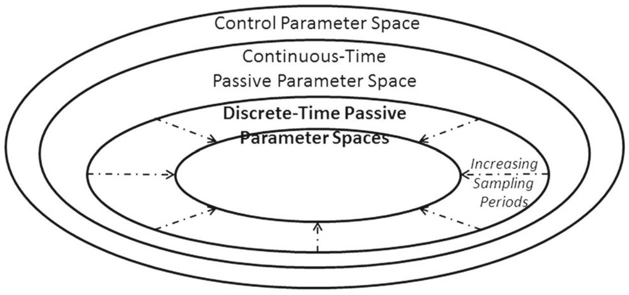
Control Parameter Space
Continuous-Time Passive Parameter Space
Discrete-Time Passive Parameter Spaces
Increasing Sampling Periods

Fig. 9 Constraints on the control parameter space

discretization is not necessarily passive [32]. To circumvent this problem, a different discretization approach is developed in [4] in which the discrete-time output is modified as

$$y_d(k) = \frac{1}{t_s} \int_{kt_s}^{(k+1)t_s} y(t)\mathrm{d}t.$$

This discretization approach guarantees that the resulting discrete-time system is passive and thus able to be analyzed using passivity-based control methods. However, the approach has a shortcoming in that it requires a future output value of $y(t)$ at $(k+1)t_s$ which may not be possible to obtain if the system is highly nonlinear. Given this fact, we discretize (14) using the standard method and choose a sampling period so that the system satisfies the discrete-time passivity inequality:

$$t_s \sum_{k=0}^{N} u_d(k)^\mathsf{T} y_d(k) \ge \mu_d t_s \sum_{k=0}^{N} \left\| u_d(k) \right\|^2 + \rho_d t_s \sum_{k=0}^{N} \left\| y_d(k) \right\|^2, \tag{16}$$

where N is a positive integer, μ_d is a real number, and ρ_d is a real number. In order to guarantee that the inequality in (16) is satisfied, we have to ensure that the sampling period is chosen so that the discrete-time passivity indexes are larger than zero given $\mu_d = \mu - t_s\gamma - t_s\gamma\left|\rho\right| - t_s^2\gamma^2\left|\rho\right|$ and $\rho_d = \rho - t_s\gamma\left|\rho\right|$ [4]. Using the experimental passivity index methods and the experimental data of the controllers from Sect. 4, we compute the passivity indexes and finite L^2 gain of the controllers as $\mu = 0.8$, $\rho = 5.6$, and $\gamma = 2$. We find that the discretized system will be passive given a sampling period smaller than $t_s \approx 55$ ms.

6.2 Quantization

Quantization is the mathematical process in which a large set of input values is rounded and truncated down to a smaller set and is needed as a result of hardware limitations on the ECUs, leading to the fact that floating-point data cannot be processed. The objective of quantization is to convert the floating-point discrete-time PHS into fixed-point discrete-time PHS. This procedure involves using the MATLAB data command fixedt() and data conversion blocks between the CarSim S-functions and controllers. State variables and other variables are rounded and truncated into fixed-point data values.

The ECUs that we use to implement the control system require either 16- or 32-bit fixed-point data types in its operation and computation, which necessitates a concern for passive quantization. In Simulink the quantization process is done using MATLAB's Fixed-Point Toolbox, in which the word lengths for all data are set as fixdt(1, 16, 8) and fixdt(1, 32, 16) for 16- and 32-bit data types, respectively [23]. Simulink's quantizer is a uniform mid-tread quantizer and is considered to be a passive quantizer, which is a concept introduced in [43] where the input v and output u mappings are bounded by two lines of slopes a and b, $av^2 \leq uv \leq bv^2$. However, even though the quantizer is passive it does not necessarily mean that the quantized system is passive. In order to ensure passivity for the quantized system we implemented the transformation block M from [43] using the values of $m_{11} = 2, m_{12} = -0.36, m_{21} = 0$, and $m_{22} = 1$, which are computed using the passivity indexes of the controllers.

6.3 Code generation and compilation

The objective of the code generation is to convert the discretized and quantized Simulink models into C code that will be executed in the HIL platform. This procedure involves the Simulink Coder (previously called Real-Time Workshop) which automatically generates the necessary C code. In our case, the control tasks execute in the kernel space of RT-Linux (under which floating-point computation is not supported), and each variable is of integral type. According to the available bits in a word and the value ranges, this can be achieved using Q format [18]. For example, in the case of a 32-bit word size, the data type of all signals in the Simulink model is set as fixdt(1, 32, 16), which means that there are 1 sign bit, 15 integer bits, and 16 fractional bits in this Q format. The Simulink Coder generates code with proper computation according to the chosen fixdt.

The code generated from the Simulink models is in C, which is compiled in order for deployment on the platform. The objective of the compilation is to deploy the control software C code onto the ECUs. In this procedure, we use gcc-4.2.4 together with the TTEthernet configuration file generated by the TTTech tool-chain to compile the generated C code, which is linked with the provided TTEthernet driver to become a kernel module. This kernel module does the computation and drives the Ethernet port of the ECU in a time-triggered fashion, namely that there is a static schedule table specifying when to compute and communicate based on a synchronized global time.

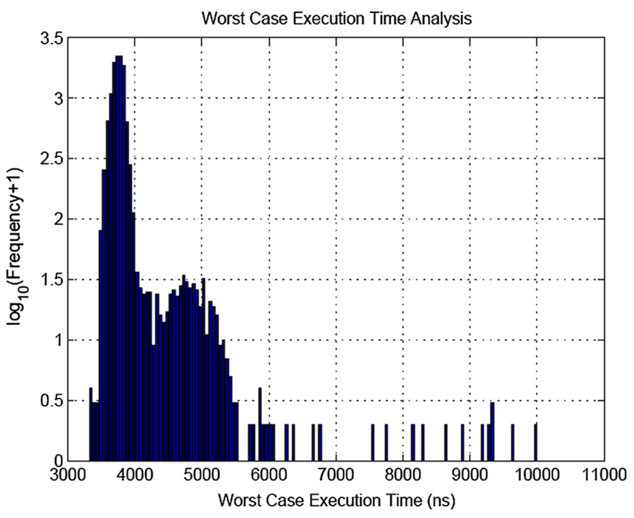

Fig. 10 Histogram of worst-case execution time

6.4 Timing analysis

The objective of timing analysis is to compute the WCET of the control tasks. In our case, the procedure involves empirically measuring the execution times of the designed control task (i.e., recording the time difference between each start and end of the execution of the task) through multiple runs under multiple circumstances. In our experiment, there are 12,000 measurements recorded, and we can plot them to observe the distribution, as shown in Fig. 10. (In order to show the frequencies of the execution times clearly, we use $\log_{10}$(number of times + 1) on the y-axis.)

From the figure, we can observe that the majority of the execution times fall in the range of 3500–4100 ns. However, there are several times in which the execution times become bigger than 8000 ns. According to our measurements, we can obtain the observed best-case execution time (BCET) as 3317 ns and the observed WCET as 10,005 ns. Since the measurement-based WCET estimation approach cannot achieve an exhaustive analysis, it is common in industry that the observed WCET is augmented by an ad hoc "scale factor." In our case, we set the scale factor to 3, so that the WCET of the control task is 30,015 ns.

6.5 Scheduling

The underlying execution model of our test bed platform is a time-triggered architecture (TTA) which maintains a synchronized global time base and requires a static schedule table for both computation and communication. The objective of scheduling is to generate a feasible static schedule for

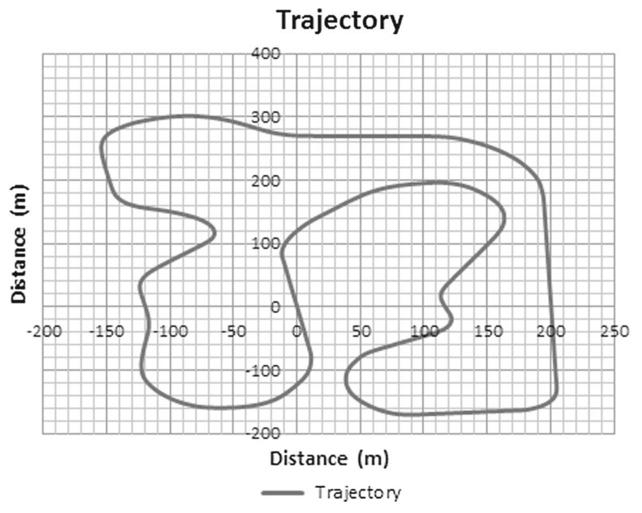

Fig. 11 Trajectory of the test track

the control tasks and messages. In general, there are many scheduling approaches for TTA combining computation and communication [28]. In our case, we have one control task whose WCET is almost negligible, thus needing only two time-triggered messages. (One is for sensing, and the other is for actuation.) The measured WCET reveals that the computation is almost negligible (only about 30 μs) compared to the sampling period (which is on the millisecond level); therefore, the schedulability problem becomes trivial in our experiments.

As mentioned above, the static schedule table is located in the TTEthernet configuration file generated by the TTTech tool-chain [35]. The main input to the tool-chain is a Python script which specifies the topology of the system as well as the hyper-period, the synchronization, and each message configuration. According to the input script, the tool-chain can generate corresponding configuration files in C for the ECUs (and binary files for other components in the system). The generated C files are compiled into a kernel module as mentioned above. The static schedule table in a generated C file only specifies how the messages are transmitted and received in a time-triggered fashion. Therefore, we need to manually add control tasks into the table with the appropriate time offsets according to the created feasible schedule. In this case, a control task becomes a kernel-space application task that is invoked periodically by the timer interrupt service routine with respect to the schedule.

7 Testing using a HIL simulation platform

The objective of the testing is to verify the correctness of the design and to record simulation results that demonstrate its effectiveness. Our HIL simulation of the closed-loop system consists of two minutes of running time in which the host vehicle follows a lead vehicle on a road with a trajectory as shown in Fig. 11. This trajectory is comprehensive because it contains many curves and straight segments which will test the effectiveness of the control design. Information from the trajectory is encoded into the vehicle model provided by CarSim, which is then given to the controllers via the communication network.

This procedure involves the implementation of the architecture shown in Fig. 12. The physical dynamics modeled in CarSim is deployed as a real-time (RT)-Target so that it acts as a real vehicle. The RT-Target is also integrated with a TTTech

Fig. 12 HIL simulator architecture

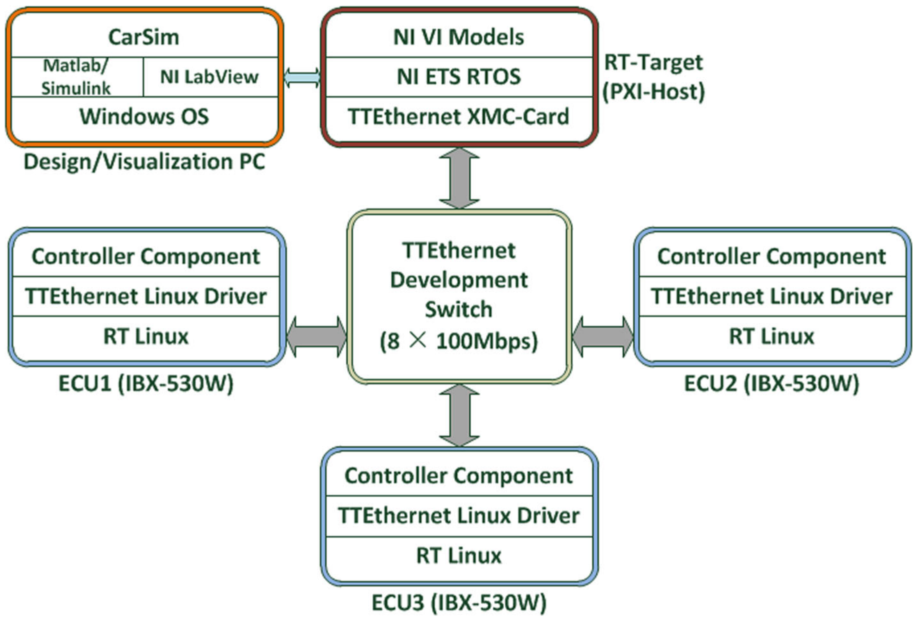

Table 2 Table of controller gains

k_{ti}	k_{bi}	k_t	k_{td}	k_b	k_{bd}	k_{si}	k_{sd}	J_c	M_c
0.05	0.01	0.1	0.02	0.2	0.02	40	15	0.2	− 0.5

PCIe-XMC card which enables the seamless integration and communication with the ECUs on a time-triggered network. The HIL simulator has three ECUs which are connected to an 8-port 100 Mbps TTEthernet development switch from TTTech [35]. Each ECU is an IBX-530W box with an Intel Atom processor running a RT-Linux operating system and is integrated with a TTEthernet Linux driver, which is a software-based implementation of the TTEthernet protocol in order to enable communication with the other systems in a TTEthernet network. The automotive control software is distributed over the ECUs, and the tasks execute in the kernel space of RT-Linux which can utilize the synchronized time base off of the TTEthernet communication. The controllers are deployed on ECU1 and communicate with the RT-Target via the TTEthernet network which provides a synchronized time base for computation and communication.

Using the constraints provided by passivity, we select control parameters so that the overall system remains passive. Table 2 shows the gain values of the controllers, which are computed using power-shaping stabilization. Power-shaping stabilization is a technique which originated from the control of passive RLC circuits; the method shapes the energy of the system by equating the storage function with the difference between the total energy in the system and the energy provided by the controller [9]. We experimentally verified that the power-shaped closed-loop system retains passivity given the sampling periods of 10, 30, and 50 ms using (16).

In order to ensure that the system remains passive during quantization, we must implement a transformation matrix M, in addition to using a passive quantization scheme. Fortunately, the quantization scheme which Simulink uses is a passive quantization. The transformation matrix M ensures the passivity of a quantized system by offsetting the effects of quantization on the inputs and outputs of the system [43]. The choice of M is not unique, which indicates that there is more than one valid transformation; as long as the matrix is invertible and satisfies the equations, it can be used to preserve passivity. Using the transformation matrix M, we are able to guarantee that the control system retains passivity given a quantization of 16 bits or 32 bits.

By ensuring that the system remains passive given discretization and quantization, we validate the control design using the HIL simulation results to show that the vehicle behaves correctly given changes in lead vehicle speed, road curvature, and slope of the road. The lead vehicle starts at a speed of 60 km/h and runs for forty seconds, before speeding up linearly to 85 km/h for twenty seconds. The lead vehicle

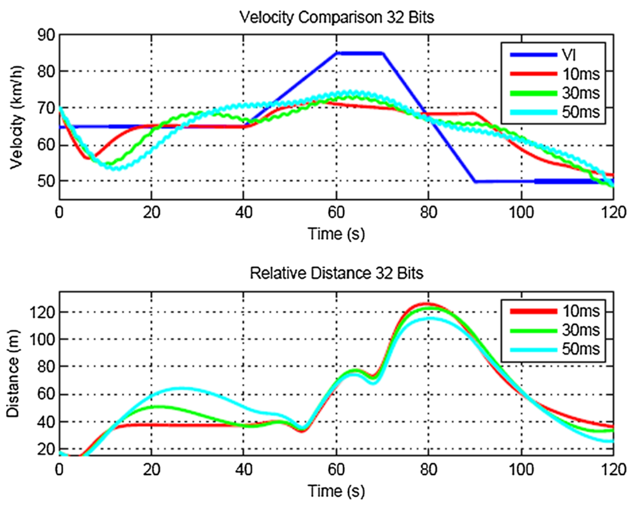

Fig. 13 Vehicle velocities and relative distances at 32-bit quantization

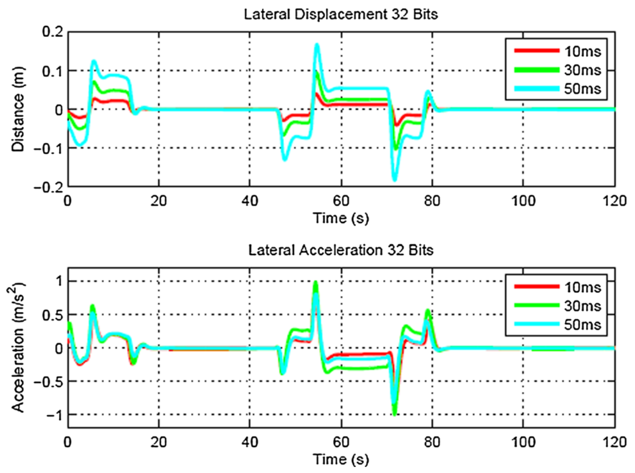

Fig. 14 Vehicle lateral accelerations and displacements at 32-bit quantization

maintains 85 km/h for ten seconds, before linearly slowing down to 50 km/h. Given the six combinations of quantization and discretization, we ran six HIL simulations in which the discrete-time controllers are implemented using a combination of sampling periods of 10, 30, and 50 ms with a quantization of 16 bits or 32 bits. After generating the C code from the six controllers, we computed the worst-case execution time to being 12 µs.

Figure 13 shows the velocity of the lead vehicle and the host vehicle under various sampling periods on the top subplot and the relative distance between the two vehicles under various sampling periods on the bottom subplot for the case of 32-bit quantization. Figure 14 shows the lateral displacement under various sampling periods on the top subplot and the lateral acceleration under various sampling periods on the bottom subplot for the case of 32-bit quantization. The simulation results show that despite keeping the objectives of speed and steering control, there is a

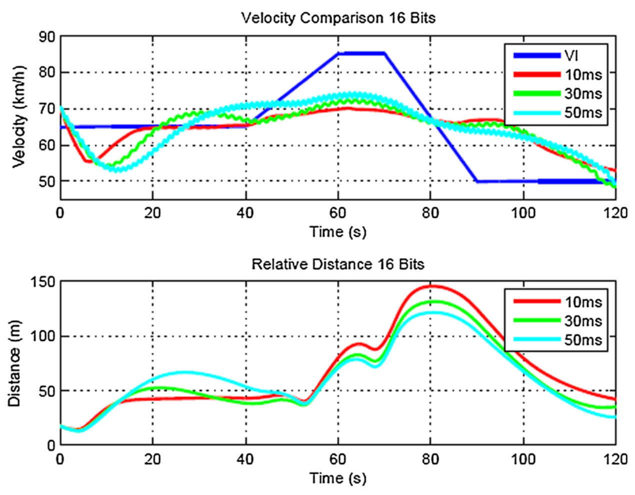

Fig. 15 Vehicle velocities and relative distances at 16-bit quantization

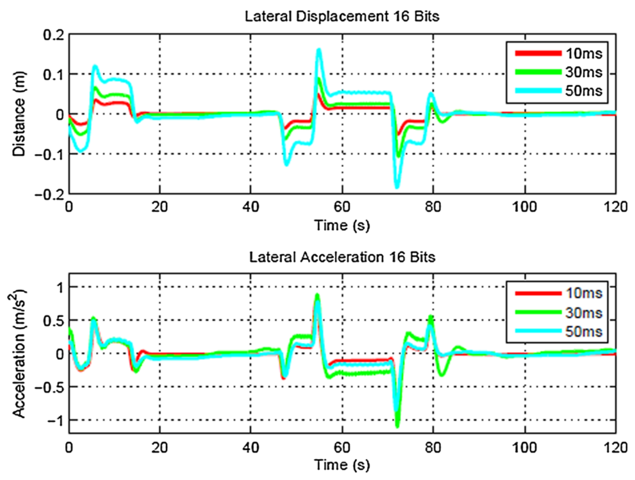

Fig. 16 Vehicle lateral accelerations and displacements at 16-bit quantization

noticeable difference between the different sampling periods. As the sampling period increases the controller reacts more slowly to the behavior of the lead vehicle. Passivity also degrades with greater sampling periods. We experimentally computed the passivity indexes for the 10-ms case as (0.7, 5.6), the 30-ms case as (0.4, 5.4), and the 50-ms case as (0.1, 5.4).

Figure 15 shows the velocity of the lead vehicle and the host vehicle under various sampling periods on the top subplot and the relative distance between the two vehicles under various sampling periods on the bottom subplot for the case of 16-bit quantization. Figure 16 shows the lateral displacement under various sampling periods on the top subplot and the lateral acceleration under various sampling periods on the bottom subplot for the case of 16-bit quantization. We experimentally computed the passivity indexes for the 10-ms case as (0.7, 5.5), the 30-ms case as (0.3, 5.3), and the 50-ms case as (0.1, 5.2). The simulation results show that

Table 3 Table of simulation scenarios

Scenario	Time (s)	V_l (km/h)	Slope (°)	Turns
1	0–40	65	0	3
2	40–52	65–77	0	1
3	52–60	77–85	– 15	0
4	60–70	85	– 15	1
5	70–90	85–50	– 15	1
6	90–94	50	– 15	0
7	94–103	50	0	1
8	103–120	50	15	1

the transformation block managed to prevent any significant degradation in passivity. Table 3 shows the various scenarios that appear during the simulation.

The simulation results indicate that the system is able to keep to the objectives of speed control and steering control despite changes in the lead vehicle speed, slopes, and turns encountered by the host vehicle. By using a passive PHS control framework, we are able to design the system to safely avoid collisions and navigate roads by moving energy between the steering control and the speed control depending on the curvature of the road. For the design and implementation of controllers onto ECUs, an important objective is ensuring that the control tasks satisfy both the execution time and performance. Finding the appropriate sampling period is important, because as the sampling rate increases, the system performance decreases. From the results of this paper, we can conclude that passivity is a good way to ensure the performance of the controller by minimizing the passivity loss from discretization.

8 Conclusion

The proposed model-based design methodology in this paper addresses the complexity challenges presented by the interactions of components in CPS. The main advantage of the proposed approach is that it guarantees that system passivity and stability are preserved after the control design and implementation steps. The methodology creates an end-to-end process in which physical systems are modeled, controllers are designed and implemented, and the closed-loop system is tested using a HIL simulation platform. The methodology handles the interactions and facilitates the integration of passive components to form a passive closed-loop system. Passivity-based control is used ensure stable behavior of the closed-loop system. We apply the model-based design methodology to an automotive system to model the interactions between the longitudinal and lateral dynamics before integrating the vehicle with a speed and steering control. The closed-loop system is implemented on a HIL simulation platform, and the results are recorded and show the effectiveness of the methodology.

Acknowledgements The authors would like to acknowledge the partial support by the National Science Foundation under awards CNS-1035655 and CNS-1739328.

References

1. Byrnes, C., Lin, W.: Losslessness, feedback equivalence, and the global stabilization of discrete-time nonlinear systems. IEEE Trans. Autom. Control **39**(1), 83–98 (1994)
2. CarSim: Mechanical: Simulation. Michigan, Ann Arbor (2013)
3. Cervera, J., van der Schaft, A., Baños, A.: Interconnection of port-Hamiltonian systems and composition of Dirac structures. Automatica **43**(2), 212–225 (2007)
4. Costa-Castello, R., Fossas, E.: On preserving passivity in sampled-data linear systems. In: Proceedings of the 2006 American Control Conference, Minneapolis, USA, pp. 4373–4378 (2006)
5. Dai, S.: Compositional modeling and design of cyber-physical systems using port-Hamiltonian systems. Doctoral dissertation, Vanderbilt University. Retrieved from http://etd.library.vanderbilt.edu/available/etd-08042016-123306/unrestricted/Dai.pdf (2016)
6. Dai, S., Koutsoukos, X.: Model-based automotive control design using port-Hamiltonian systems. In: International Conference on Complex Systems Engineering (ICCSE 2015), University of Connecticut, November 9–10 (2015)
7. Dai, S., Koutsoukos, X.: Safety analysis of automotive control systems using multi-modal port-Hamiltonian systems. In: 19th ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2016), Vienna, Austria, April 12–14 (2016)
8. Derler, P., et al.: Modeling cyber-physical systems. Proc. IEEE **100**(1), 13–28 (2012)
9. Dirksz, D., Scherpen, J.: Port-Hamiltonian and power-based integral type control of a manipulator system. In: 18th IFAC World Congress, Italy, Milan (2011)
10. Donaire, A., Junco, S.: Derivation of input-state-output port-Hamiltonian systems from bond graphs. Simul. Model. Pract. Theory **17**, 137–151 (2009)
11. Duindam, V., Macchelli, A., Stramigioli, S., Bruyninckx, H.: Modeling and Control of Complex Physical Systems: The Port-Hamiltonian Approach. Springer, Berlin (2009)
12. Eyisi, E., Zhang, Z., Koutsoukos, X., Porter, J., Karsai, G., Sztipanovits, J.: Model-based control design and integration of cyber-physical system: an adaptive cruise control case study. J. Control Sci. Eng. **2013**, 678016 (2013). https://doi.org/10.1155/2013/678016
13. Fujimoto, K., Sugie, T.: Canonical transformation and stabilization of generalized Hamiltonian systems. Syst. Control Lett. **42**, 217–227 (2001)
14. Fujimoto, K., Sugie, T.: Freedom in coordinate transformation for exact linearization and its application to transient behavior improvement. In: Proceedings of the 35th IEEE Conference on Decision and Control, pp. 84–89 (1996)
15. Generic Modeling Environment. http://repo.isis.vanderbilt.edu/downloads?tool=GME
16. Golo, G., van der Schaft, A., Beedveld, P., Mascheke, B.: Hamiltonian formulation of bond graphs. In: Johansson, R., Rantzer, A. (eds.) Nonlinear and Hybrid Systems in Automotive Control, pp. 351–372. Springer, London (2003)
17. Hooke, R., Jeeves, T.: Direct search solution of numerical and statistical problems. J. Assoc. Comput. Mach. **7**, 212–229 (1969)
18. Jiyang, K.K., Kum, K., Kang, J., Sung, W.: A floating-point to fixed-point C converter for fixed-point digital signal processors. In: Second SUIF Compiler Workshop (1997)
19. Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. Proc. IEEE **91**, 1 (2003)
20. Khalil, H.K.: Nonlinear Systems, 3rd edn. Prentice Hall, Upper Saddle River (2002). ISBN 0-13-067389-7
21. Kirschke-Biller, F.: AUTOSAR—a global standard. In: 4th AUTOSAR Open Conference, Paris, France (June 11, 2012)
22. Lee, L.: Cyber-physical systems: design challenges. In: International Symposium on Object, Component, and Service-Oriented Real-Time Distributed Computing (ISORC), Orlando. FL, USA (2008)
23. MATLAB, Version 7.10.0 (R2010a). The Mathworks Inc., Natick (2010)
24. Oishi, Y.: Passivity degradation under the discretization with the zero-order hold and the ideal sampler. In: 49th IEEE Conference on Decision and Control, December 15–17, Atlanta, GA, USA (2010)
25. Ortega, R., Garcia-Canseco, E.: Interconnection and damping assignment passivity-based control: a survey. Eur. J. Control **10**, 432–450 (2004)
26. Ortega, R., Jiang, Z.P., Hill, D.J.: Passivity-based control of nonlinear systems: a tutorial. In: Proceedings of the American Control Conference, Albuquerque, NM (1997)
27. Porter, J., Hemingway, G., Nine, H., et al.: The ESMoL language and tools for high-confidence distributed control systems design—part 1: language, framework, and analysis. Technical report ISIS-10-109, Vanderbilt University (2010)
28. Porter, J., Karsai, G., Sztipanovits, J.: Towards a time-triggered schedule calculation tool to support model-based embedded software design. In: Proceedings of the Seventh ACM International Conference on Embedded Software, Grenoble, France, October 12–16, pp. 167–176 (2009)
29. Rajamani, R.: Vehicle Dynamics and Control. Springer, Berlin (2006). ISBN:978-0-387-26396-0
30. Sakai, S., Stramigioli, S.: Port-Hamiltonian approaches to motion generation for mechanical systems. In: IEEE International Conference on Robotics and Automation, Rome, Italy (2007)
31. Simko, G., Levendovsky, T., Maroti, M., Sztipanovits, J.: Towards a Theory of Cyber-Physical Systems Modeling. CyPhy, Philadelphia (2013)
32. Stramigioli, S., Secchi, C., van der Schaft, A.J., Fantuzzi, C.: Sampled data systems passivity and discrete port-Hamiltonian systems. IEEE Trans. Robot. **21**(4), 574–587 (2005)
33. Sztipanovits, J.: Composition of cyber-physical systems. In: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Washington, DC, USA (2007)
34. Sztipanovits, J., et al.: Toward a science of cyber-physical system integration. Proc. IEEE **100**(1), 29–44 (2012)
35. TTEthernet. http://www.tttech.com/en/products/ttethernet/
36. van der Schaft, A.J., Maschke, B.M.J.: Mathematical modeling of constrained Hamiltonian systems. In: Proceedings of the Third IFAC Symposium on Nonlinear Control Systems (1995)
37. van der Schaft, A.: Port-Hamiltonian systems: an introductory survey. In: Proceedings of the International Congress of Mathematicians (2006)
38. van der Schaft, A.: Port-Hamiltonian systems: network modeling and control of nonlinear physical systems. In: Advanced Dynamics and Control of Structures, pp. 127–167, Springer, Vienna (2004). ISBN: 978-3-7091-2774-2
39. Wu, P., McCourt, M., Antsaklis, P.J.: Experimentally determining passivity indices: theory and simulation. Technical report of the ISIS Group (2013)

40. Yu, H., Antsaklis, P.J.: A passivity measure of systems in cascade based on passivity indices. In: 49th IEEE Conference on Decision and Control (2010)
41. Zhang, Z., Eyisi, E., Koutsoukos, X., Porter, J., Karsai, G., Sztipanovits, J.: Co-simulation framework for design of time-triggered cyber physical systems. In: Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems, Philadelphia, PA, USA (2013)
42. Zhao, J., Hill, D.J.: Dissipativity theory for switched systems. IEEE Trans. Autom. Control **53**(4), 941–953 (2008)
43. Zhu, F., Yu, H., McCourt, M.J., Antsaklis, P.J.: Passivity and stability of switched systems under quantization. In: Conference on Hybrid Systems Computation and Control, April 17–19, Beijing, China (2012)

Siyuan Dai received his Ph.D. degree in electrical engineering from Vanderbilt University and B.S. degree in electrical engineering from the University of Notre Dame. Previously, he was a researcher at Toyota InfoTechnology Center in Mountain View, CA, the USA. He is currently employed as a researcher at Autoneum, an automotive tier 1 supplier. His research work is in the areas of autonomous vehicles and electric vehicles.

Zhenkai Zhang received his Ph.D. degree in computer science from Vanderbilt University, M.S. degree from University of Chinese Academy of Sciences, and B.S. degree from Beijing Institute of Technology. He is a Research Scientist with the Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN, the USA. His research work is in the areas of cyber-physical systems, real-time systems, and security.

Xenofon Koutsoukos received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, the USA, in 2000. He is a Professor with the Department of Electrical Engineering and Computer Science and a Senior Research Scientist with the Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN, the USA. He was a Member of Research Staff at the Xerox Palo Alto Research Center (PARC) (2000–2002), working in the embedded collaborative computing area. His research work is in the area of cyber-physical systems with emphasis on formal methods, data-driven methods, distributed algorithms, security and resilience, diagnosis and fault tolerance, and adaptive resource management. He has published more than 250 journal and conference papers, and he is co-inventor of four US patents. Prof. Koutsoukos was the recipient of the NSF Career Award in 2004, the Excellence in Teaching Award in 2009 from the Vanderbilt University School of Engineering, and the 2011 NASA Aeronautics Research Mission Directorate (ARMD) Associate Administrator (AA) Award in Technology and Innovation. He was named a Fellow of the IEEE for the class of 2018.