Holistic simulation for integrated vehicle design

Nikolaos Kalantzis*, Tom Fletcher,
Antonios Pezouvanis and Kambiz Ebrahimi

Aeronautical and Automotive Engineering,
Stewart Miller Building Loughborough University,
Loughborough LE11 3TU, UK
Email: N.Kalantzis@lboro.ac.uk
Email: T.P.Fletcher@lboro.ac.uk
Email: a.pezouvanis@lboro.ac.uk
Email: k.ebrahimi@lboro.ac.uk
*Corresponding author

Abstract: A holistic vehicle simulation capability is necessary for front-loading component, subsystem, and controller design, for the early detection of component and subsystem design flaws, as well as for the model-based calibration of powertrain control modules. The current document explores the concept of holistic vehicle simulation by means of reviewing the current trends in automotive system design and available solutions in terms of model interfaces and neutral modelling environments. The review is followed by the presentation of a Simulink-based multi-disciplinary modelling environment (MME) developed by the authors to accommodate simulation work across the vehicle development cycle.

Keywords: co-simulation; heterogenous; simulation environment; model integration; automotive.

Reference to this paper should be made as follows: Kalantzis, N., Fletcher, T., Pezouvanis, A. and Ebrahimi, K. (2021) 'Holistic simulation for integrated vehicle design', *Int. J. Powertrains*, Vol. 10, No. 1, pp.27–53.

Biographical notes: Nikolaos Kalantzis holds a Doctorate from the University of Bradford in Micro-Cogeneration Systems and is a Research Associate in the Department of Aeronautical and Automotive Engineering of Loughborough University. His areas of expertise are distributed energy systems and automotive system engineering.

Tom Fletcher received his MEng in Automotive Engineering and PhD in Optimal Energy Management of Hydrogen Fuel Cells from Loughborough University, UK, in 2011 and 2017 respectively. He is currently a Research Associate in the Advanced Propulsion Group at Loughborough University in the Department of Aeronautical and Automotive Engineering. His research interests include powertrain systems integration and optimisation, thermal management and adaptive control of electrified vehicles.

Antonios Pezouvanis studied BEng in Mechanical and Automotive Engineering and obtained a PhD on the same subject in 2010. He is a Research Fellow in Advanced Propulsion at Loughborough University, UK. He has been involved in research and industrial collaborative research and development projects for over 10 years in the field of engine mapping and calibration, vehicle and powertrain mathematical modelling and testing, and other mechanical engineering applications.

Kambiz Ebrahimi is a Professor of Advanced Propulsion at Loughborough University. His research is in the area of systems dynamic and control with applications in powertrain design and testing.

This paper is a revised and expanded version of a paper entitled 'Modelling environment for holistic vehicle simulation' presented at 4th Biennial International Conference on Powertrain Modelling and Control (PMC 2018), Loughborough University, UK, 10–11 September 2018.

1 Introduction

Modern vehicles are highly complicated systems comprising of hardware, software, and mechanical components (Le Marrec et al., 1998) with vehicle electrification developing at a fast pace (Karvonen and Thiringer, 2015). Driving automation and electrification of modern vehicles has made vehicle development a very complicated process (Chen et al., 2018). With time, road vehicle complexity is increasing (Porlán, 2016; Maharun et al., 2013) while development cycles are becoming shorter and development budgets become tighter (Klein et al., 2017). The above, combined with the current trend in automotive companies of developing multiple offerings of a given vehicle model make for a multi-dimensional design space which requires a very high volume of test data (Mikelsons and Samlaus, 2017).

Computer aided engineering (CAE) involves the replacement of physical prototypes with mathematical representations of an engineering system with the intent of testing these mathematical models in the virtual world and extracting results that are relevant and applicable to the real world (Mikelsons and Samlaus, 2017). CAE enables the engineer to simulate the behaviour of a system before the 3D geometry of the components is available, and thus, it can be used from the early stages of the product design cycle (Khan et al., 2017). This allows the various research and development departments within an engineering entity to reduce the number of physical prototypes, and generate optimal component or subsystem designs and test them well before neighbouring components or subsystems are available (front loading) (Pedersen et al., 2015; Wu et al., 2014; Fitzgerald et al., 2010). Therefore, the implementation of CAE tools has the potential to reduce the cost and time of vehicle development while allowing for better designs (Wu et al., 2014). Due to the above, CAE is of a very high importance in automotive research and development (Aslan et al., 2015; Dyer et al., 2010) and has a wide spectrum of applications (Khan et al., 2017) which includes concept evaluation, combustion simulation for optimal engine design, simulation of the gear mesh forces, vehicle dynamics, development of embedded software functionality (Mikelsons and Samlaus, 2017), optimisation of design and control for minimum fuel consumption and emissions (Özener and Allouchery, 2017), virtual validation of software functionality, as well as the stimulation of tested hardware (HiL) (Mikelsons and Samlaus, 2017).

In the traditional design approach, also known as sequential system design (Kyllingstad et al., 2017), each component is developed separately as a complete unit by a specialised group (Le Marrec et al., 1998), each using a separate development approach (Fitzgerald et al., 2010; Reyneri et al., 2002). Once all components are available in physical form, they are integrated to the high-level system at the final developmental

stage. Even though the interaction between subsystems and the transient characteristics of the high level system plays an important role on high level system reliability and performance (Fitzgerald et al., 2010), the sequential system design approach does not take it into account (Wu et al., 2014), and therefore, it does not allow for an early detection of errors at a system level, and as a result, potential issues with subsystem incompatibility may appear during subsystem development (Fitzgerald et al., 2010), or even worse, when system design is finished (Fitzgerald et al., 2010). Late detection is expensive and difficult to correct (Kyllingstad et al., 2017) as even small design changes may require a redesign of the high level system from the ground up (Reyneri et al., 2002). Embedded systems are challenging to develop – computing and hardware components must be combined and match seamlessly, must be robust, and not collapse under fault conditions (Fitzgerald et al., 2010).

Addressing the dependability of subsystem performance on their interaction with the whole system is very important. The modern holistic/multi-disciplinary design approach involves the development of system components in parallel to save time (Fitzgerald et al., 2010). A complete virtual prototype of the system is developed before the creation of a physical prototype. This virtual prototype is detailed to incorporate system dynamics necessary for carrying out a certain development task (Kyllingstad et al., 2017). Using the virtual prototype, subsystems and control code are designed as part of the system rather than in isolation, even though no physical prototype of the system exists. As a result, subsystem incompatibilities are detected during early design stages and component design is optimised in synergy with the system (Fitzgerald et al., 2010; Kyllingstad et al., 2017; Reyneri et al., 2002; Casoli et al., 2014), thus facilitating globally optimal system designs (Khan et al., 2017; Kyllingstad et al., 2017). In addition, some design tasks (such as control strategy design and component testing) are rescheduled to earlier stages of the project (frontloading). Frontloading makes for more efficient product development and testing as it allows controller models and subsystems to be developed and tested in a wholistic virtual and test bench environments respectively rather than requiring the availability of a physical vehicle prototype (Le Marrec et al., 1998; Klein et al., 2017; Khan et al., 2017; Zhang et al., 2017). Faults can be artificially induced to controllers to test their behaviour offline (Fitzgerald et al., 2010). The communication system can also be modelled in detail in order to identify failures caused by communication errors (Pedersen et al., 2015). As a result, the design process of a controller is faster and cheaper with debugging and testing possible at an early design stage (Khan et al., 2017; Casoli et al., 2014), while powertrain components can be verified early on in the vehicle development cycle by coupling them to real-time capable numerical models (HiL) (Klein et al., 2017). In addition to the above, the use of holistic simulation has the potential to prevent damage to expensive prototypes and testing equipment (Pedersen et al., 2015) and for this reason, some researchers have developed engine test cell scheduling functionality on holistic virtual environments (Klein et al., 2017; Fletcher et al., 2018). Thus, the usefulness of multi-domain simulation is recognised within the automotive industry (Porlán, 2016).

In vehicle development, different domains are usually handled by different departments, each of which uses application specific software (Mikelsons and Samlaus, 2017; Pedersen et al., 2015; Aslan et al., 2015) with each simulation tool selected for being most suitable for a certain application (Khan et al., 2017). This results in a wide a variety of modelling and simulation environments being used across the vehicle

development cycle. While this multitude of used modelling and simulation environments may seem counterproductive, there are good reasons behind this software diversity.

- Using one software for all operations can be time consuming and, in many cases, impossible as the component libraries, numerical solvers, user interface, and optimisation tools of each software are tailor made for a given application. Building a component library from scratch can be very costly and running a component model with an unsuitable solver can result to an unstable simulation. It is therefore beneficial to use more than one environment in an integrated simulation, taking advantage of the specialised capabilities of each environment (Khan et al., 2017).
- Different CAE tools sharing the same area of focus may have a different approach towards modelling (effort to build a model vs. model fidelity, required data) (Khan et al., 2017), therefore, even two comparable software may have a place within the same organisation.
- Conventional single solver simulation is usually not sufficient for the system-level development of modern vehicles (Chen et al., 2018).

While the above makes sense considering the need for discipline-specific component libraries and numerical solvers, a heterogenous collection of subsystem models built in non-compatible environments is in most cases not useful for carrying out a holistic vehicle simulation. Multi-domain simulations may not just involve different modelling tools but also different time scales (Özener and Allouchery, 2017) and required numerical solvers (Casoli et al., 2014). The component models must be integrated into one high level/system model. Depending on the involved platforms, model integration may be difficult and require a lot of work (Aslan et al., 2015).

The aims of the current document are the review of the current model trends and methods in holistic modelling and simulation environments and model integration interfaces, as well as the development of a Simulink-based multi-disciplinary modelling environment (MME) to serve as a universal holistic vehicle simulation platform on a company-wide scale. Section 2 is comprised of a review of the model integration methods most commonly used in the automotive industry and a review of publications where co-simulation (CS) was used as a tool to design and test automotive software, hardware, and electromechanical components. Section 3 presents arguments in favour of adopting a generic holistic vehicle simulation environment and lists desirable characteristics such an environment must have followed by a brief description of neutral model integration environments. Section 4 presents the development and application of the Simulink-based MME. Section 5 describes the simulation setup and comments on the simulation results. Section 6 concludes on the outcome of the document.

2 Model integration methods in literature

Most modern modelling environments can connect to other modelling environments (Wu et al., 2014; Sweafford et al., 2012). A model is sufficient for the purpose when it encapsulates a sufficient degree of detail (Fitzgerald et al., 2010). Co- simulation is a method of connecting different simulations running on different modelling and simulation environments and coordinate the execution of each simulation and the

communication between the connected simulations (Klein et al., 2017; Pedersen et al., 2015; Wu et al., 2014). This enables the engineers to connect different modelling and simulation environments, each dedicated to a specific domain or area of interest. The resulting integrated simulation combines the strengths (Khan et al., 2017; Zhang et al., 2017) and alleviates the weaknesses (Xie et al., 2011) of the individual environments as the engineers take advantage of the different capabilities of each environment to build as detailed component/subsystem models as possible (Mikelsons and Samlaus, 2017) for the least amount of effort and connect the subsystem simulations to an accurate system-level simulation. Combining CAE tools in such a manner facilitates the realisation of an interdisciplinary/holistic design approach (Le Marrec et al., 1998), which is a necessary condition for the parallel development of automotive subsystems.

CS incorporates the high-level dynamic behaviour of the system is for modern integrated design and allows for the development of control strategies as well as the evaluation of the design itself (Kyllingstad et al., 2017). For controller development, it is common to use real-time CS (Chen et al., 2018). Real-time CS is also used in HiL testing which allows for the connection of a physical component to a real-time simulation (Klein et al., 2017). For a real-time simulation, the modelled components are exported to FMI or S-Function (Mikelsons and Samlaus, 2017) and simulated on a real-time computer. Due to these reasons, CS is gaining popularity in the automotive (Chen et al., 2018).

2.1 Types of model integration interfaces

The need for a holistic vehicle simulation is currently being addressed by engineering software developers through the incorporation within their software of interfaces that allow for the connection of a model with models built in other engineering software via model import, model export or coupled simulation. Currently, a wide array of model integration options is available. In terms of the universality of model compatibility, model integration interfaces can be divided into two main categories:

- Proprietary – the interface is proprietary to the target software.

  Some CAE software includes interfaces that connect them to a specific target. In this case, the local model runs on its original platform and exchanges data with the global model running on another platform, usually via opening a virtual network to couple the two platforms (Maharun et al., 2013; Wu et al., 2014; Eckert et al., 2014). Most automotive simulation platforms support CS with Simulink in the role of the global but also the local model.

- Tool agnostic – the interface is supported by a wide array of commercial software.

  S-Function – Some commercial CAE software export models to MATLAB S-functions (Sweafford et al., 2012) which can be imported and simulated in Simulink, or imported to a wide array of real-time computers and simulated in real-time.

- Functional mockup interface (FMI). There are different subcategories of functional mockup units (FMU) classified according to the standard version and the location of the numerical solver.

Figure 1 Tree diagram of model integration methods most commonly used in automotive applications

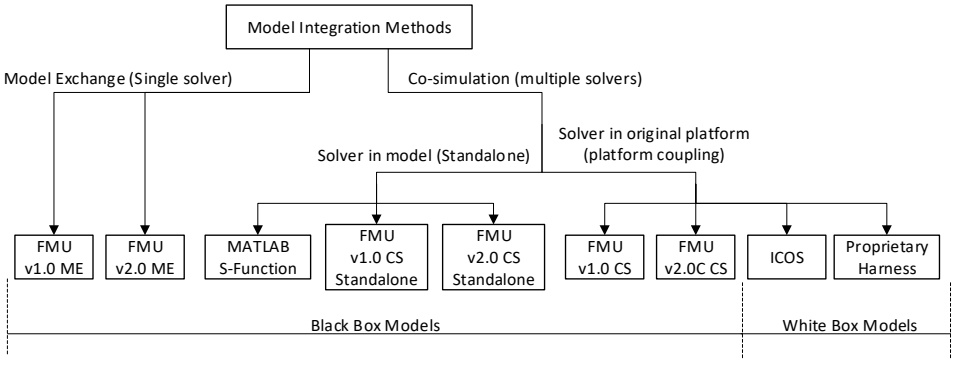
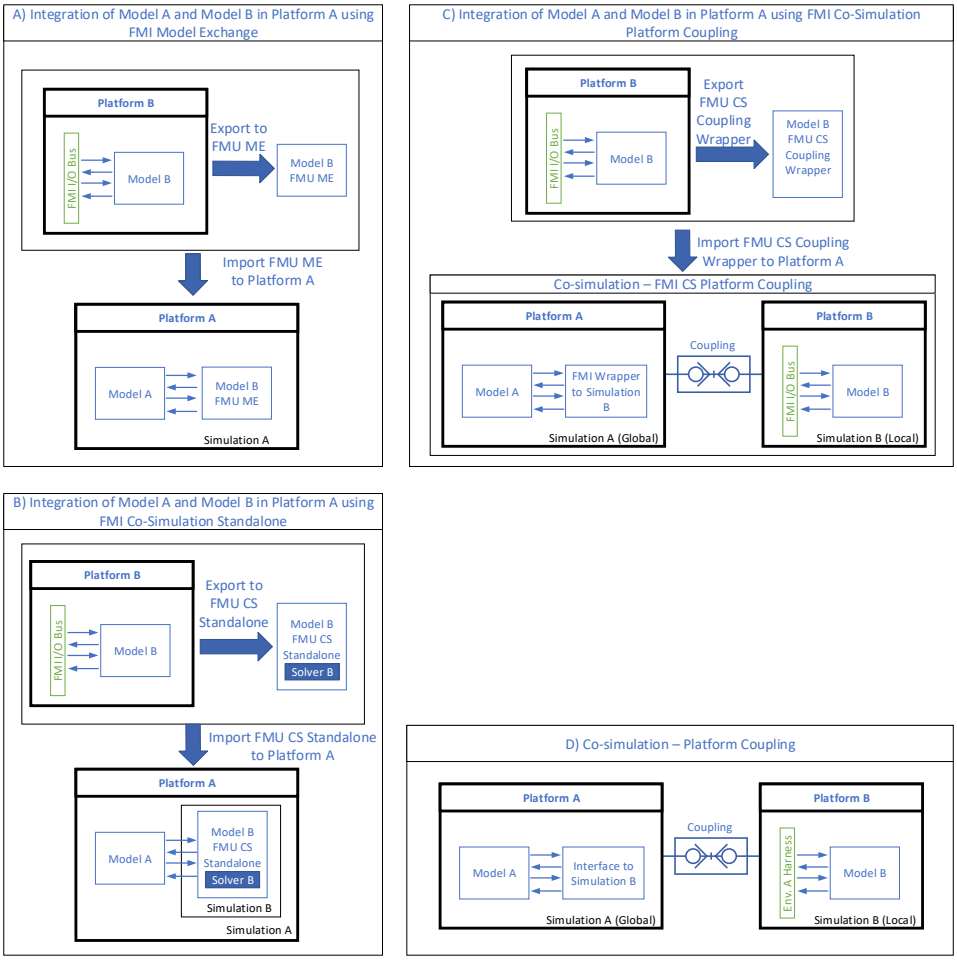


Figure 2 Diagrams of model FMU and dedicated harness model integration (see online version for colours)



The tree diagram of model integration methods most commonly encountered in automotive applications is shown in Figure 1. When the component models are integrated into a global simulation and are simulated by a single global solver, the integration method is described as model exchange (ME). When the component models are integrated into a global simulation but are simulated by one dedicated local solver per component, the integration method is described as CS. CS with solver coupling is capable of multi-resolution simulations. Multiple domain-specific local numerical solvers are integrated into one high-level multi-disciplinary simulation (Chen et al., 2018). The time step for each component model (local solver) is set to accommodate the different stiffness of each model (Casoli et al., 2014) as well as the sample rate of the controller models. CS is subdivided in to standalone versions, in which case the solver is integrated within the model file and local simulation runs on the target environment, and platform coupling versions in which case the local model is simulated on the original platform, and the two software platforms are coupled together and exchange data. In CS, data exchange between local models occur every macro-step of the global model. The local models are integrated under the local solver micro-step.

Figure 2 contains model integration diagrams of the different categories of FMU (A–C) as well as the proprietary platform coupling harness (D).

While the ME and CS is structurally different, the term CS is commonly used to describe model integration regardless of method. For the sake of simplicity, the term will be used to describe all forms of model integration unless stated otherwise.

Another way to distinguish between integration methods is to separate to white box where the structure of the exchanged model is visible to the model recipient, and to black box where the structure of the exchanged model is not accessible to the recipient. The latter category is very important for the protection of the intellectual property of the supplier (Blochwitz et al., 2011).

2.1.1 The FMI

One of the most common and promising methods of connecting models from different platforms together is through the export and the import of an FMI compliant model (Mikelsons and Samlaus, 2017; Pedersen et al., 2015; Aslan et al., 2015), and for this reason, it is of particular interest for this work. The interface has been developed by a group of engineering software developers and research groups (Aslan et al., 2015) known as the consortium of the MODELISAR project with the purpose of supporting the exchange of numerical models of subsystems (Blochwitz et al., 2011). Initially developed for vehicular embedded system design, FMI is starting to be used in other engineering sectors (Pedersen et al., 2015; Hallqvist et al., 2017; Gaaloul et al., 2013; Himmler et al., 2018). FMI is a popular software agnostic communication protocol between simulation environments that allows for models to be used in a wide array of software that support the protocol (Khan et al., 2017; Pedersen et al., 2015; Aslan et al., 2015; Bertsch et al., 2014). FMU is a model file [essentially an archive file (Pedersen et al., 2015)] that follows the FMI standard which comprises the model interface specs.

There are currently two versions of FMI, and each version is subdivided to ME which runs on the solver of the host platform (global) (Aslan et al., 2015), and CS which uses its dedicated local solver in a global-local solver layout. The global model manages the simulation and local models solve respective problems. Local models are directly connected to the global model and indirectly connected with other local models via the

global model. The solutions within the local models are independent between one another and keep running even between communication points. Data exchange is discrete point (Aslan et al., 2015). FMU CS models are divided in two categories. The common CS version simulates the FMI on the original environment and exchanges data with the global solver via a virtual network. The standalone version is packed with its dedicated numerical solver which is executed in the target software. While the FMI standard has many advantages contributing to its popularity, there are some shortcomings under discussion in literature:

- The FMI standard does not specify a global simulator (Pedersen et al., 2015).
- FMI specs do not include high-level software approaches such as object-oriented development (Aslan et al., 2015).
- No vector and structure support. No means of modelling the communication between blocks. A virtual CAN must be modelled to validate timing, thus complicating ECU representation (Mikelsons and Samlaus, 2017).

2.2 Comparison of model integration interfaces

While there is currently a wide array of options available for integrating component models in a holistic system CS, each interface has certain advantages and disadvantages that may make it suitable for some but not applications. For this reason, it may be beneficial for different stages across the vehicle development cycle to use different integration interfaces the selection of which should be the outcome of careful consideration. The main characteristics of each interface are presented in tabulated form in Table 1.

FMI for ME runs on the solver of the host platform (Hallqvist et al., 2017; Blochwitz et al., 2011, 2012). As a result, no coupling or extrapolation errors are introduced. In addition, it does not require a solver to be packed with the model thus allowing for small file sizes and being capable to run on the host pc regardless of whether the original platform is installed (if the model is not licence protected).

MATLAB S-function runs on the solver of the local model. It can support discrete, continuous, and hybrid systems (Van Schijndel, 2014). It can be directly imported to MATLAB/Simulink target and is supported by real-time computers (Honek, 2010; He and Peng, 2010). Its main advantage is its very high computational efficiency when based on the C language (Siegele et al., 2014) and the small size of model files.

FMI for standalone CS uses the global-local solver architecture, thus allowing for different steps between the global and local models. Under this variant, the FMU is exported with its dedicated solver packed with the model. The local model is simulated in the host environment using the pre-packed local solver (Blochwitz et al., 2011). The main advantage of this FMI CS variant is the inherently higher simulation speed compared to the FMI CS for platform coupling variant due to the elimination of platform communication latency. Another advantage is the capability of the standalone version to be simulated without the need for an installation of the original platform. The main disadvantage of the standalone variant is the very large size of the model files which makes them less than ideal for sharing with other users.

FMI for platform coupling CS uses the global-local solver architecture, thus allowing for different steps between the global and local models but the local model is simulated

on the original platform and the two platforms are coupled together exchanging data. The target environment connects to an FMU wrapper rather than to the actual local model (Blochwitz et al., 2011). The main advantage of this method is the small size of the model file. The main disadvantages are the considerably slower simulation speed for simpler models compared to standalone FMI versions and the requirement for an installation of the environment of the local model on the host pc. This FMI variant is best used when the user has a full software installation on the host PC, model file size must be small, and/or the local model is computationally heavy.

Proprietary platform coupling harnesses are also supported by many vendors, but it is most common for these interfaces to connect the platform to MATLAB/Simulink due to its universality in control development applications. Simulation environments are usually equipped with one harness to use Simulink as the global, and another to use the Simulink as the local model. Other coupling interfaces that are relatively common connect an environment to another environment developed by the same company. Coupling harnesses to platforms developed by different companies are not common. Its main advantages over the competing interfaces are that it is usually the easiest method to connect two software with, it uses the dedicated solvers of each platform and allows for different global, local, as well as communication step sizes, it is white-box and therefore the content of each model is accessible for modification between simulations without the need for the user to manually recompile the model. The main disadvantage of this method it the relatively slow speed caused by delays in the communication of the two platforms (Zhang et al., 2017).

Table 1 Popular model integration standards and associated characteristics

Criteria	*Model integration interfaces*					
	FMU ME	*MATLAB S-function*	*FMU CS standalone*	*FMU CS platform coupling*	*ICOS*	*Proprietary platform coupling harness*
Support by software	Very high	Very high	Very high	Very high	High (virtual vehicle)	High
Simulation speed	Very high	Very high	Very high	Ranges from slow to high	Ranges from slow to high	Ranges from slow to high
Model configurability	Poor	Poor	Poor	Poor	Excellent	Excellent
Simplicity in procedure setup	Simple	Simple	Simple	Simple	Very simple	Very simple
Multiple solvers	No	Yes	Yes	Yes	Yes	Yes
Weak coupling	No					
Installation of original platform	No	No	No	Yes	Yes	Yes
Model access	Black box	Black box	Black box	Black box	White box	White box
Model file size	Small	Small	Very large	Very large	Small	Small

ICOS is a CS interface developed by VIRTUAL VEHICLE and supported by AVL Model.CONNECT CS tool. ICOS is capable of connecting a wide array of general purpose and automotive modelling and simulation environments such as Simulink,

Dymola, AMESim, and CarMaker, GT, and Simulation X with one another. Each ICOS interface has a specific target environment. It shares a platform coupling technology and has the advantages of supporting white model structure (making it suitable for early stages of model development), multi-solver structure with the added advantage of being capable of connecting to real-time systems, and being supported by a popular dedicated model integration tool (Innerwinkler et al., 2018). ICOS incorporates a number of techniques such as energy-preserving coupling methods to reduce the coupling error (Hübner, 2018). Since it is a platform coupling tool, simulation can be slower than other options where the local solvers run in the target platform.

2.2.1 Discussion on model integration interfaces and suitable applications

All presented interfaces have a high support by software vendors. The advantage of the FMI standard over the rest of the software is that it is supported by many environments for both export but also for import. In addition, it is supported by several real-time environments. The S-function is best used when a very fast black box model is to be ran in MATLAB/Simulink or when a real-time computer is the target environment. FMI for ME interface is best used when a model must be shared as black box and the imported model is compatible with the solver and time step of the host platform. FMI for standalone CS is best used when the local model is of a relatively low complexity and the CS speed is of top priority such as in the case of parameter optimisation and when the user has a solver licence but not a software licence for the local environment. Proprietary platform coupling harnesses are particularly useful during the initial developmental stages of an integrated simulation as it allows for quickly changing the models in one or more sides and observing the result of the change. ICOS interface is used when the AVL Model.CONNECT is used as the main model integration tool, and no export of the local model to FMI CS is possible, the co-model is under development and the white box model capability is useful, or when a connection to real-time system is necessary.

In terms of control systems design, it is a common practice for controller models to be developed in Simulink. When the controller of a cyber-physical system is developed on Simulink, it is a common practice to use the environment hosting the plant to act as the global modelling environment.

When a design optimisation is to be carried out, it is better to import the plant model on Simulink to take advantage of MATLAB optimisation and data processing libraries.

2.3 Application of model integration methods in the automotive industry

The following paragraphs of the current section comprise a brief categorised description of the automotive applications of model integration methods as encountered in the reviewed literature.

2.3.1 Automotive embedded control

Le Marrec et al. (1998) coded software in C language, modelled hardware in VHDL, and mechanical components in MATLAB, and all were integrated to a MATLAB global simulation using VCI interface for use in the functional validation of initial specification.

Li et al. (2015) integrated a Simulink model of an optimal slip ratio targeting ABS controller with a CarSim Vehicle dynamics model. The integrated model was used to compare the pressure compensating controller to a baseline controller.

Xie et al. (2011) developed a co-model of a light passenger car consisting of a vehicle model with an engine and a dual state CVT transmission model from AMESim, and MATLAB/Simulink Transmission Control Unit (TCU) model. The model was validated using a typical passenger car cycle to verify the CS model is operating as intended.

Reyneri et al. (2002) developed a plugin that partitions electromechanical systems and modelling hardware and software architecture in detail in respective sub models with the purpose of co-designing and testing software and hardware parts of embedded systems.

Maharun et al. (2013) developed a co-model of a Parallel Hybrid Electric Vehicle comprised of the energy management system (EMS) and fuzzy vehicle dynamics controller (VDC) model and the electrical components models in Simulink, and the vehicle model in ADAMS/Car with the purpose of evaluating EMS performance.

Wu et al. (2014) integrated a GT-Power engine model, AMESim torque converter, transmission, and vehicle dynamics models, and a Simulink controller model within a Simulink global model. The CS was used to prove that shift quality can be improved through the coordinated engine and gearbox control.

Mikelsons and Samlaus (2017) co-simulated a GT-Suite powertrain, a CarMaker vehicle dynamics model, and an ETAS EVE yaw rate controller model, all exported to FMI and integrated in AVL Model.CONNECT in order to carry out a functional validation of the FMI ECU model.

Zhao and Zhu (2018) imported a CarSim-generated S-Function model of a racing car into a global Simulink model of a fuzzy PID-based cruise controller and compared its performance to a baseline vehicle configuration.

Casoli et al. (2014) developed a co-model of engine and fluid power in Simulink and used it for the design of optimal engine and fluid circuit combination, as well as for the design of control for fuel efficient operation of mobile machinery. Fluid power circuit was modelled in AMESim and converted to an S-Function, then imported to Simulink and connected to a Simulink real- time engine model.

Li et al. (2014) setup a vehicle CS in order to test a Fuzzy ESP Control strategy with respect to vehicle handling stability. Multibody system dynamics were modelled in ADAMS/Car while the controller model was developed in Simulink.

He et al. (2017) setup a co-model of a novel integrated electro-hydraulic brake system Simulated in AMESim, a 15 DoF vehicle dynamics model simulated in AMESim, and a hierarchical nonlinear controller model simulated in Simulink with the purpose of comparing the vehicle stability under the proposed design and strategy to that under a baseline conventional ESP system.

Ibrahim et al. (2018) developed a CS framework between Matlab and SUMO and connected a Matlab multi-layer vehicle platooning control algorithm to a SUMO traffic behaviour model to evaluate the performance of the platooning control strategy.

Bours et al. (2013) setup a CS between of ADAS sensors and world simulation running in PreScan software and control algorithm and vehicle dynamics simulation running on Simulink using a dedicated CS interface with the purpose of evaluating the sensitivity of the modelled autonomous emergency braking (AEB) system via parameter variation.

Pedersen et al. (2017) used the FMI-based INTO-CPS tool chain to co-simulate multiple FMU CS models exported from different environments with the purpose of developing EGR Water Handling control strategy. The control strategy was developed in MATLAB and exported via the Modelon toolbox to FMU CS.

Domenici et al. (2018) setup a CS between a PVSio autonomous vehicle controller model and a Simulink vehicle kinematics model with the purpose of proving the concept of formal verification of autonomous vehicles via integrated simulation.

Chen et al. (2011) setup a CS between an MSC.ADAMS full vehicle model and a Simulink electric power steering (EPS) controller model and used the CS to develop an EPS boost curve and evaluate its performance with respect to vehicle control stability.

Chang and Choi (2007) co-simulated an ADAMS multi-body vehicle model with a Simulink-generated C code model of an EPS controller imported to ADAMS and used the CS for calibration and performance evaluation of the EPS.

Levesley et al. (2007) integrated an MSC.visualNastran multi-body quarter vehicle model into a Simulink global model containing tyre, damper, and controller models. They developed the co-model for the purpose of optimising the structure and control algorithm of semi-active suspension systems.

Zhang et al. (2010) integrated an ADAMS suspension multi-body dynamic model of 1/2 automotive suspension system within a global Simulink model containing an LQG-based active suspension controller via the dedicated coupling interface and used the co-model to investigate the potential of the LQG controller for handling and stability improvement.

Feng and Zhang (2012) integrated an ADAMS multi-body vehicle and front steering system model within a Simulink model containing the Fuzzy controller of an active steering system and used the co-model to show that the simulated controller improves vehicle stability.

Li and Xiao (2014) integrated a GT-Power compressed natural gas engine model within a global Simulink engine controller model via the dedicated coupling interface and used the co-model to develop an adaptive feed-forward engine control strategy.

Van Dong et al. (2019) co-simulated a CarSim vehicle model with a global Simulink model containing a Luenberger observer and used the model as a proof of the concept of using a Luenberger observer to estimate the states of the vehicle.

Ha et al. (2013) modelled an in-wheel electric vehicle in CarSim and integrated it into a global Simulink model of a VDC with the purpose of investigating the effect of VDC on vehicle behaviour.

2.3.2 Fuel consumption optimisation

Özener and Allouchery (2017) setup a CS between an IPG Truck Maker 3D bus and road model with an AVL Cruise drive train model and carried out offline fuel consumption and emission optimisations of speed profiles of busses. CS was established via a dedicated CS interface.

Eckert et al. (2014) setup a CS between a ADAMS vehicle multibody dynamics model and a Simulink longitudinal dynamics model including a lookup table based engine model. The authors also developed an optimiser that used the CS to produce optimal gear shifting strategy to minimise fuel consumption while maximising performance.

2.3.3 Combination of components and its effects on system performance – NVH studies

Khan et al. (2017) co-simulated an ADAMS multi-body dynamics 3D vehicle model with a powertrain controller developed in AMESim with the purpose of predicting vehicle Noise Vibration and Harshness induced by powertrain. The ADAMS vehicle model was converted to FMI and imported to AMESim.

Dyer et al. (2010) built a co-model of the US Army High Mobility Multi-Wheeled Vehicle comprising of an MSC.ADAMS vehicle model, FEAP (Finite Element Analysis Package) tyre models, and PSAT (Powertrain Systems Analysis Toolkit) powertrain model with the purposes of further developing tyre and contact models, and exploring model integration options.

Hareesha and Bharath (2015) co-simulated a virtual prototype of a double wishbone suspension system in ADAMS and a PID controlled force signal in Simulink with the purpose of investigating the dynamic behaviour of a double wishbone suspension system. The integration was carried out via the ADAMS/Control Module and Simulink served as the global simulator.

Fleissner and Eberhard (2009) built a co-model connecting a 17 DoF multi-body silo vehicle model running in Simpack, and a particle hydrodynamics model of the sloshing liquid cargo running in Pasimodo with the purpose of generating optimal tank designs under which sloshing liquid cargo has a minimal effect on vehicle stability. Simulink served as the global simulator and the connecting link between the two software. The connection between Simpack and Simulink was achieved via the proprietary coupling interface. Pasimodo and Simulink were connected via using the respective plugin interface by Pasimodo.

Han et al. (2017) setup a CS between a multi-body dynamics tracked vehicle model and road model running in RecurDyn, and a hydropneumatics suspension model running in AMESim with the purpose of investigating the effect of hydropneumatics parameter variation on ride safety.

Li et al. (2019) integrated a 11 DoF CarSim in-wheel motor drive electric vehicle (IWMD EV) model within a global Simulink model containing the motor and motor controller models using a platform coupling block with the purpose of studying the path tracking and self-driving performance of intelligent IWMD EVs.

Hong et al. (2009) setup an AMESim PHEV global and imported a Simulink Hybrid Control Unit model and used the co-model to investigate potential drivability issues during the engagement and disengagement of the ICE.

2.3.4 Combination of components and its effects on system performance – electric propulsion

Karvonen and Thiringer (2015) developed an Ansys Simplorer/Maxwell co-model of a drive system. The magnetic component of the electric machine is modelled in Ansys Maxwell, and all other components on Ansys Simplorer. The co-model was used to study the current and voltage harmonics induced on the DC bus by the switching events.

Peng et al. (2015) setup a CS comprising of a MATLAB/Simulink plug-in hybrid school bus model (powertrain, controllers, and EMS), and a CANoe controller area network (CAN) bus model and used the CS to investigate the performance and reliability of the CAN bus and the energy performance of the EMS.

2.3.5 Automotive test rig and test schedule development

Klein et al. (2017) developed an Engine in Loop (EiL) test rig as a tool to investigate the effects the variation of vehicle parameters on the actual engine. A Model in Loop (MiL) vehicle model CS was setup on dSpace VEOS comprising of a dSpace VSM vehicle dynamics model, a GT power FRM engine model, and a Simulation X automatic double clutch transmission (DCT) model exported to FMU CS at a fixed step. The vehicle was modelled in ASM Tool Suite. The TCU was modelled in Simulink. At the second stage, the GT engine model was replaced by a physical engine test bench which was connected to the CS. An interface that connects the non-rt simulation (MiL), the dSpace SCALEXIO and the test bench is developed by the authors.

Fletcher et al. (2018) developed an engine calibration validation tool using a co-model of a test cell and engine. The test cell controller was developed using Simulink StateFlow charts. The PCM SiL was developed on Simulink, while the GTDI engine was a Ricardo WAVE-RT model. The global model was running on Simulink and the engine model was integrated via the dedicated WAVE- RT Simulink harness.

Zhang et al. (2017) setup Simulink co-models of vehicle suspension durability rigs. The vehicle and the test rig mechanical components were modelled in ADAMS and the hydraulic and control elements modelled in Simulink. Remote Parameter Control RPC Pro software was used to control the CS. Connection between Simulink and ADAMS was established using the virtual server option.

2.3.6 Other

Khadr et al. (2013) setup a CS between an ADAMS scooter and road model connected to a Simulink PID controller-based driver model with the purpose of developing a methodology for the assessment of the dynamic behaviour of the scooter during manoeuvres.

Shojaei et al. (2017) integrated a Dymola-generated FMI CS thermal model (HVAC and engine, battery thermal management systems) into a global Simulink model containing WARPSTAR-based powertrain model and a controller model, and used the co-model to calculate the cooling loads for different duty cycles and ambient conditions.

Park et al. (2012) built an electric vehicle model in CarSim and integrated it in S-Function form into a global Simulink model containing battery, power electronics, and driver subsystems with the purpose of investigating the effects of inverter faults on vehicle performance.

From the above, it is observed that holistic simulation plays an instrumental role in both vehicle controller developments as it allows for controller software, hardware, and communication network to be developed and tested with a high-fidelity plant model under normal operating conditions and under fault conditions, and this improves controller reliability the performance. A holistic simulation allows for using well correlated vehicle models under realistic driving conditions both provided by the most suitable platform to identify vehicle driving patterns that optimise performance, fuel consumption, and emissions. Simulation of component interaction allows the engineers to verify component compatibility and avoid low performing combinations. Offline CS is useful for designing test rigs and testing schedules safely and for low cost, while real-time CS is used in testing physical components under realistic simulated conditions. In terms of used interfaces, the most popular options are the FMI standard and the platform

coupling CS, while in terms of CS environments for control development, Simulink is found to be the most popular option. Another encountered environment dedicated to model integration is AVL Model.CONNECT. In addition to commercial software, researchers have developed environments whose main functionality is to build and integrate FMU models to a CS.

3 The need for a generic holistic vehicle simulation environment

Section 2 compared the existing model integration methods and reviewed the usage of model integration methods in automotive literature. The current section will discuss the advantages of the concept of a general holistic vehicle simulation environment that supports all the major integration interfaces, to act as a standardised binder between the multiple heterogenous models.

While one may argue that the direct connection between two simulation platforms via a combination of FMU import and platform coupling interfaces may be the fastest route towards a holistic vehicle simulation, the fact that each department tends to use a modelling and simulation platform specific to its area of expertise means that under such arrangement, it is more likely than not for each department to setup their own holistic vehicle simulation with the global simulation running on their chosen software. Such practice may appear to save time in the short term, but combining many heterogenous models in a non-consistent manner can be the cause of data discrepancy and numerical inconsistencies (e.g., a CS integrated in one environment may not give identical numerical results to the same models integrated under another environment) and complicate the dissemination of information between departments (e.g., data files used by a given department may not be readily accessible by the software used by another department) (Badin et al., 2011), and in the long run, it may increase the model development and maintenance costs on a company-wide scale. In addition to the above, the design exploration/optimisation capabilities of each department are constrained by the capabilities of the design exploration module integrated within the selected global simulation platform.

From the above, one may acknowledge the potential for a universal modelling and simulation environment to serve on a company-wide scale as the global simulator of a holistic vehicle model/co-model. The modelling environment of a holistic vehicle simulation must possess the following characteristics:

- It must support the connection to a wide array of modelling and simulation platforms.
- It must be generic in nature and applicable on a company-wide scale across the vehicle design lifecycle without the need for any changes in the top-level architecture. Therefore, the data buses used for the communication between the main subsystems must support all signals that may possibly be needed by any given department within an automotive company.
- It must be modular and easily adaptable with the user being able to transition from a high-fidelity configuration utilising the maximum number of signals to a low-fidelity configuration utilising the minimum necessary number of signals and vice versa under the same general high-level layout and model setup procedure.

- It must be capable of carrying out the global vehicle simulation while running each subsystem model under different numerical solver configurations to cater for the special requirements of different energy domains, model fidelities, and numerical solvers.
- It must support complicated simulation control and data post-processing.
- It must feature a straightforward method of developing and integrating user code to carry out bespoke simulation control schemes, data post processing tasks, and data visualisation, as such would be highly advantageous.

A generic and at the same time modular holistic vehicle simulation environment will enable all departments to carry out design operations of different nature using the same top-level model structure. This will allow for consistency in design practices, reduction in training costs and an easier rotation of personnel between departments. In addition, it will prevent overlap in modelling work between departments as new configurations required by a team could easily be generated by adapting existing configurations already generated by another team. For example, the co-model used by team concerned with vehicle NVH may part of the local structure of a co-model used by the team concerned with vehicle drivability.

3.1 Neutral model integration environments

There is a wide array of software that can serve in the role of a holistic simulation environment. Some of the software are general modelling and simulation platforms with good connectivity to other software, and some are dedicated model integration platforms:

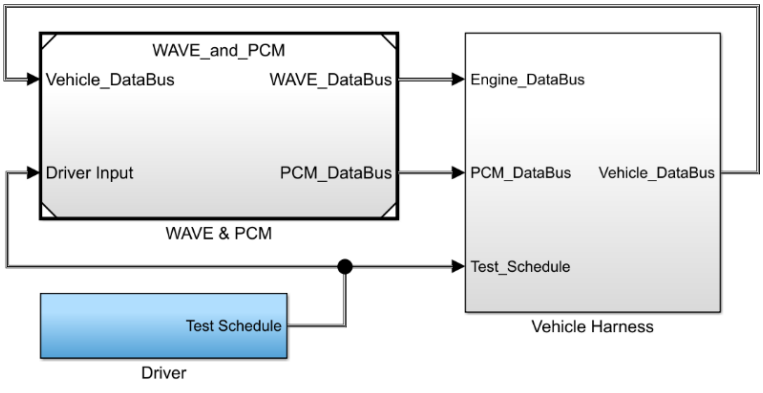
- MATLAB/Simulink – Most CAE software can connect to MATLAB/Simulink via a proprietary interface, export to S-function, or export to FMI (also supported by Simulink)
- OpenModelica supports the FMI standard and therefore has a neutral model integration capability
- AVL Model.CONNECT
  1 Dedicated CS middleware
  2 Allows for integrating models built on different environments into one high level simulation
  3 Supports model integration standards such as FMI and ICOS
  4 Connects to AVL Testbed.CONNECT to exchange data with and control the test bed emulators
- FEV xMOD
  1 Dedicated CS platform
  2 Allows for integrating models built on different environments into one high level simulation
  3 SiL, HiL capable

- VIAS MpCCI
  1 Dedicated CS platform
  2 Connects to FMI, MATLAB, Adams, SIMPACK, as well as multiple EM, FEM and CFD simulatiom platforms
- dSpace VEOS can simulate virtual ECUs and models. It supports S-function, AUTOSAR, as well as the FMI
- SystemC enables event-driven simulation of embedded systems. SCNSL extension to SystemC has been used to model the communication network (timing characteristics) the integration of FMU models is possible (Pedersen et al., 2015).

4 Simulink-based MME

Section 3 presented the advantages of using a universal model integration environment for automotive engineering applications, listed a number of characteristics that are highly desirable from such an environment, and briefly mentioned existing environments that could potentially serve in this role, with MATLAB/Simulink being among them. As discussed in Kalantzis et al. (2018), MATLAB/Simulink combines a number of characteristics that make it particularly suitable in serving as the backbone of a generic holistic vehicle simulation environment. The current section presents the MME, a holistic vehicle simulation environment developed by the authors in MATLAB/Simulink as an exercise in the combination of the characteristics listed in Section 3 into a MATLAB/Simulink-based holistic vehicle simulation tool.

Figure 3 Top-level Simulink block diagram of the MME (see online version for colours)

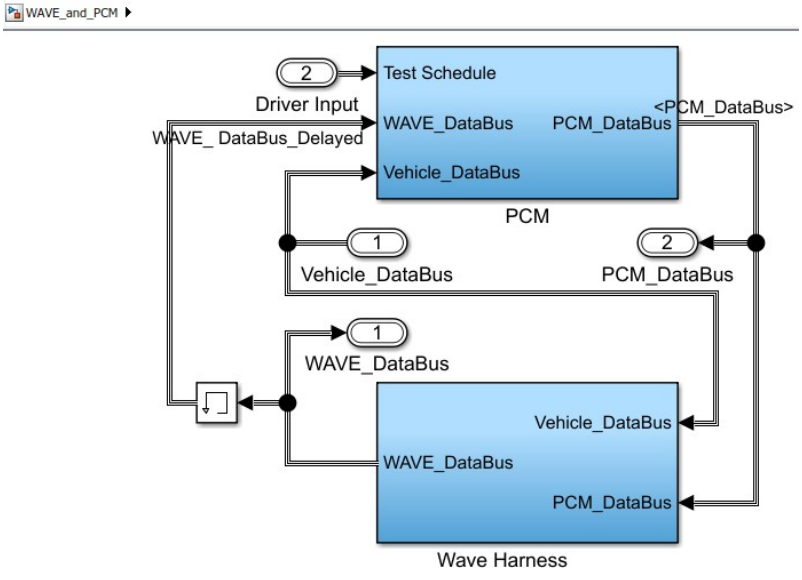


The developed MME can integrate the subsystem models built in several different heterogenous platforms, thus enabling for a holistic vehicle simulation. To ensure the applicability of the environment on a company-wide scale, the top-level Simulink block diagram and the involved data busses must be able to accommodate all possible uses of the MME. The top-level Simulink block diagram of the developed MME is shown in **Error! Reference source not found.**. As can be observed, the block diagram consists of three main blocks. The blocks communicate with each other using data busses. To allow for the reference to external Simulink local models, the data bus objects stored in the tool

library and are loaded when the global model is loaded. The input and output ports within local Simulink models destined to communicate with the global Simulink model via data busses have their data types set to the respective data bus. The 'WAVE & PCM' is a Simulink Model block within the MME block diagram referring to a separate Simulink model named 'WAVE_and_PCM'.

Referenced model 'WAVE_and_PCM' whose block diagram is shown in Figure 4 contains a virtual representation of the powertrain control module (SiL PCM), and the 'wave harness' block. The 'wave harness' block contains a Ricardo WAVE-RT Simulink interface block which calls a crank angle resolved, real-time capable gasoline turbo direct injection (GTDI) engine model. In addition to the WAVE-RT interface, the 'wave harness' block includes blocks and connections that convert the signals of the input data buses (PCM_DataBus and VehicleDataBus) to forms that are readily useable by Ricado WAVE engine models.

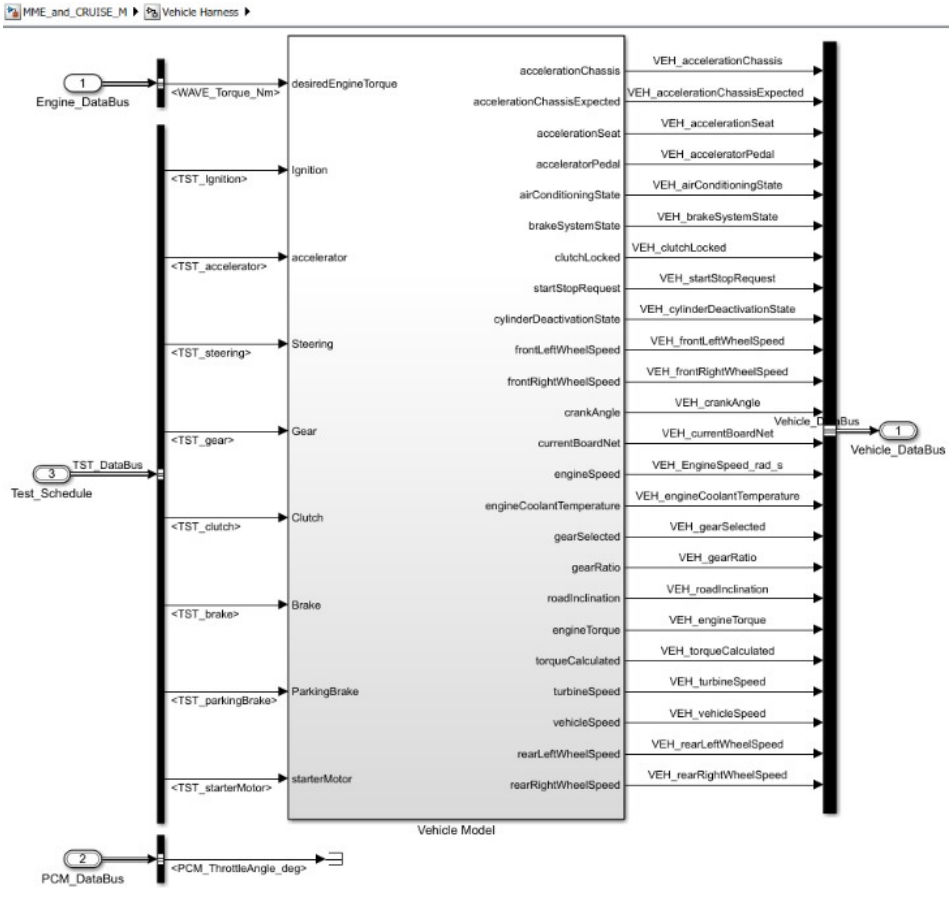
Figure 4 Block diagram of the 'WAVE_and_PCM' model (see online version for colours)



The 'vehicle harness' block whose content is shown in Figure 5 selects all signals that may be necessary for the operation of a vehicle model from the three data busses. This block has been included as a place holder for the case in which the engine model is integral to the vehicle model. The 'vehicle model' block contains the interface block to the vehicle model. An interface may be in the form of a third-party platform coupling/CS interface or a Simulink model referencing block that connects the global Simulink model to a local Simulink model containing and running the vehicle model under a different numerical solver configuration than that of the global holistic vehicle model. Alternatively, a Simulink or Simscape vehicle model may be added into the 'vehicle model' block and ran as part of the global vehicle model (provided the global solver is suitable). In terms of the vehicle model outputs, the 'vehicle harness' block combines all output signals of the 'vehicle model' block into the 'Vehicle_DataBus' bus.

To demonstrate the modularity and prove the concept of the developed MME, three vehicle models have been built in AVL CRUISE M, Ricardo IGNITE, and LMS AMESIM to serve as local vehicle models to the developed MME. The three models have been built to resemble each other as closely as possible in both structure and parameterisation to exhibit as a close of a behaviour as possible.

Figure 5 Content of the 'vehicle harness' block



The first configuration of the tested MME involves the connection of the MME to an AVL CRUISE M vehicle model. The content of the 'vehicle model' Simulink block connects the main MME to an external (referenced) local Simulink model which in turn contains a MATLAB S-function vehicle model exported by AVL CRUISE M. The reason for running the CRUISE M generated S-function as a referenced local model is the fact that the MME global simulation must run under a variable step solver while the S-function vehicle model must be simulated under a fixed step solver of a 10^{-5} sec time step. Unused I/O ports serve as placeholders for when they are needed. The non-connection of some of the output ports does not prevent the simulation of the holistic vehicle model.

The second configuration of the tested MME vehicle model involves the connection of the MME to a vehicle model built in Ricardo IGNITE. The connection of the global MME model to the local IGNITE model is achieved through the addition of two IGNITE CS interface blocks within the 'vehicle model' block with one interface providing the time step of the CS, and the other interface exchanging data with the IGNITE vehicle model. In this case, the IGNITE vehicle model is simulated in IGNITE using the solver and simulation time step that have automatically been selected by IGNITE. This configuration makes use of only two vehicle model outputs compared to the ten outputs used in the above configuration with the unconnected I/O ports block serving as placeholders.

The third configuration of the tested MME involves the CS of the MME with an LMS AMESIM vehicle model. The connection of the MME with the AMESIM vehicle model is achieved through the addition of an AMESIM CS interface Simulink block within the 'vehicle model' block. The local vehicle model is simulated within LMS AMESIM (under a solver configuration automatically selected by AMESIM) and exchanges data with the MME via the AMESIM CS interface Simulink block. In this case, eight of the output ports of the 'vehicle model' block are connected to the vehicle model, while the rest of the blocks are left unconnected, thus showcasing once again the modularity and flexibility of the developed MME.

5 Simulation and comparison of results

Following the description of the developed modelling environment and the three MME configurations, this section validates and demonstrates the modularity of the developed MME concept by means of simulating the three configurations of the MME described above and comparing the generated simulation results. For this purpose, a constant desired engine load input is applied to the holistic vehicle model. Constant blocks within the 'driver' block provide the 'WAVE_and_PCM' local model with the desired engine load, and the vehicle local model with desired gear, brake signal, and clutch signal through the 'TST_DataBus'.

The simulation of the holistic vehicle model starts with an initial vehicle speed of 20 km/h and stops at 30 seconds. The clutch is kept engaged, and the brakes are kept disengaged throughout the simulation. The desired engine load is constant throughout the simulation.

The engine speed traces of the MME simulated with CRUISE M-generated S-function, IGNITE, and AMESIM vehicle models are plotted in Figure 6. It can be observed that all third-party vehicle models have been successfully integrated within the MME and communicate with the WAVE-RT engine model, as well as the Simulink based PCM. The engine speed traces of all tested configurations exhibit shapes that behave in a similar manner and are closely located with one another. All traces exhibit an oscillatory behaviour of a higher amplitude within the 14" to 18", and 24" to 27" intervals.

As expected, the vehicle speed traces plotted in Figure 7 follow the trends exhibited by the respective engine speed traces with all vehicle speed traces being located at a very close proximity with one another. The oscillatory behaviour exhibited by the engine speed traces is also present in the respective vehicle speed traces of Figure 7.

Figure 6 Engine speed vs. time plot of the simulated MME-vehicle combinations (see online version for colours)

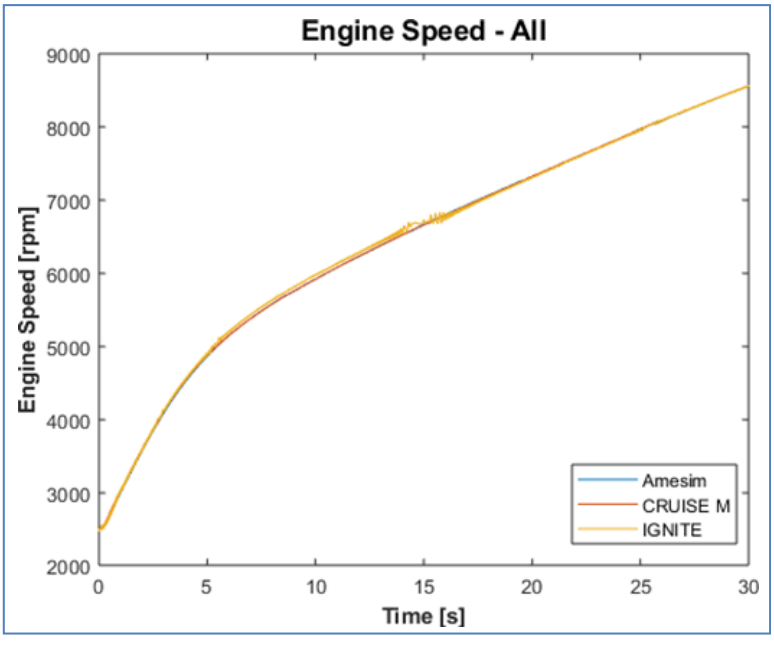


Figure 7 Vehicle speed vs. time plot of the simulated MME-vehicle combinations (see online version for colours)

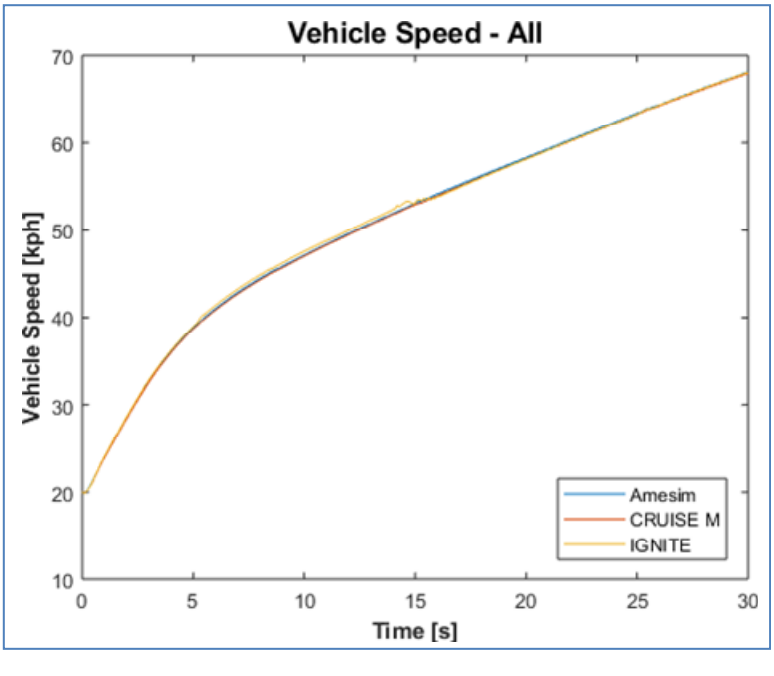


Figure 8 Detailed vehicle speed comparison (see online version for colours)

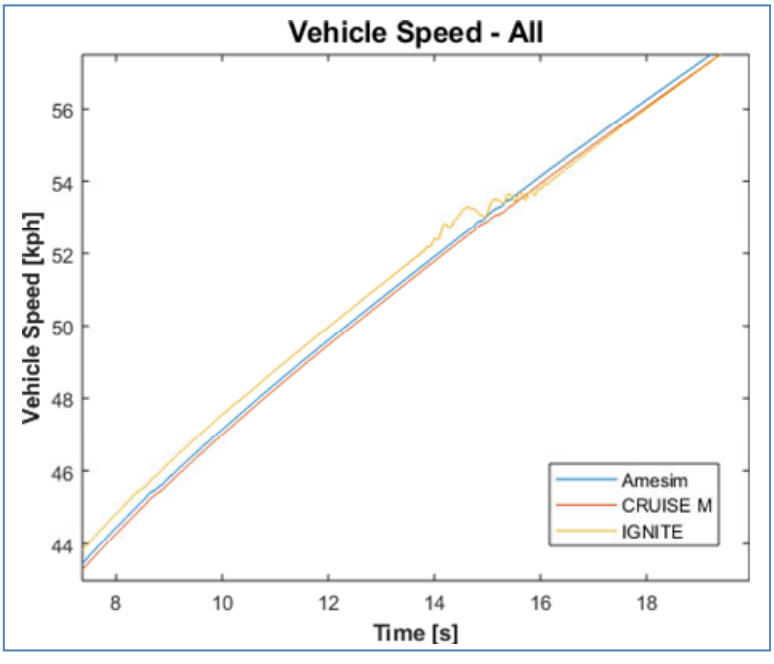
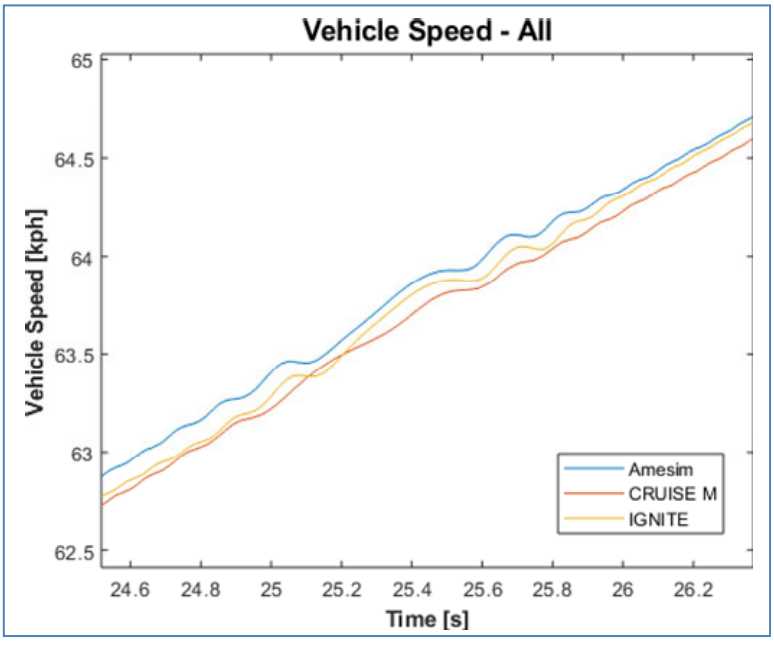


Figure 9 Detailed vehicle speed comparison (2) (see online version for colours)



The closer inspection of the traces in the plots of Figure 8 reveals the effects the CS and the interaction between separate models have on the results generated by the holistic vehicle model. Because of the use of the engine speed (calculated within the vehicle model) as a feedback signal to the engine and PCM models, small differences in the communication of signals between the different configurations cause small differences in calculated acceleration. In addition to the above, small differences in the architecture of the models as well as in the solvers employed by the individual platforms introduce additional error in the calculated values.

Figure 9 focuses on the time interval between 24.6" and 26.2" during which, all three vehicle speed traces exhibit an oscillatory behaviour. The oscillatory nature of the torque generated by the crank angle resolved WAVE-RT engine model excites the driveline and drift in and out of phase with the driveline natural frequencies. Another observation on the oscillations of vehicle speed traces is that while they depict the nature of the system correctly, their high magnitude highlights the need for the selected stiffness values for the driveline shaft components used for this study to be replaced by the calibrated values. Despite this, the vehicle models still served well the purpose of demonstrating the advantages offered by the developed MME by being capable of integrating distinct solvers dedicated to different physical domains into a holistic vehicle simulation.

While differences in calculated engine speed and vehicle speed values have been observed between the three tested MME configurations, these can be characterised as small.

The capability of the developed MME of integrating subsystem models of varying capabilities built on many different platforms in one holistic vehicle simulation allows for building numerous holistic vehicle model configurations. The model fidelity and chosen modelling environments of the constituent subsystem of these configurations depends on the subject of the study, as well as the position of the study within the product development cycle.

A team working on a very early stage of the vehicle design lifecycle may make use of the MME to integrate a simple mean value or response surface engine model and a rigid driveline vehicle model to formulate the set of requirements/specifications for the design.

In later stages of the vehicle design lifecycle, engineering teams may use the MME to integrate high-fidelity subsystem models to carry out detailed design of the subsystem under investigation.

6 Conclusions

The aims of the current document were the review of the current model trends and methods in holistic modelling and simulation environments and model integration interfaces, as well as the proof of the concept of a Simulink-based MME in the role of a universal holistic vehicle simulation platform.

The reviewed literature showed that model integration is used in numerous applications across the vehicle development cycle with embedded controller design being the most common application. During embedded controller design, it is a common practice for controller models to be developed in Simulink. Other areas of focus include NVH studies, fuel consumption studies, subsystem compatibility studies, automotive subsystem design (e.g., thermal management systems), and test bench development. The most encountered integration method involves the use of proprietary platform coupling

interfaces due to being the simplest and most straightforward to implement and well suited to the model developmental phase due their white box structure. Another highly popular interface is the FMI which is gradually emerging as the dominant tool-agnostic model integration interface. Finally, MATLAB S-function has a strong presence in model integration applications to Simulink target.

The comparison between the encountered interfaces, showed that each family of interfaces characterised by a distinctive set of advantages and disadvantages that make them best suited for a particular type of application. The S-function is best used when a very fast black box model is to be ran in MATLAB/Simulink or when a real-time computer is the target environment. FMI for ME interface is best used when a model must be shared as black box and the imported model is compatible with the solver and time step of the host platform. FMI for standalone CS is best used when the local model is of a relatively low complexity and the CS speed is of top priority such as in the case of parameter optimisation and when the user has a solver licence but not a software licence for the local environment. Proprietary platform coupling harnesses are particularly useful during the initial developmental stages of an integrated simulation as it allows for quickly changing the models in one or more sides and observing the result of the change. ICOS interface is essentially a platform coupling interface and it is used when supported by one tool for integration to a target tool and the co-model is under development hence the usefulness of the white box model capability is useful. Another application of ICOS is when a connection to real-time system is necessary. Hence it is concluded that there is not an all-around best option, but rather there is a place for all reviewed interfaces.

A holistic vehicle simulation environment that supports all popular model integration interfaces can streamline the vehicle development process. The authors built and used the Simulink-based MME to explore the potential of MATLAB/Simulink as the backbone of such application. Three modelling combinations of the same vehicle were successfully executed giving numerical results in close proximity between modelling combinations thus proving the feasibility of the concept.

References

Aslan, M., Oğuztüzün, H., Durak, U. and Taylan, K. (2015) 'MOKA : an object-oriented framework for FMI', *47th Summer Comput. Simul. Conf. 2015*, *SummerSim '15*, pp.1–8.

Badin, J., Monticolo, D., Chamoret, D. and Gomes, S. (2011) 'Knowledge configuration management for product design and numerical simulation', *ICED 11 – 18th Int. Conf. Eng. Des. - Impacting Soc. Through Eng. Des.*, August, Vol. 6, pp.161–172.

Bertsch, C., Ahle, E. and Schulmeister, U. (2014) 'The functional mockup interface – seen from an industrial perspective', *Proc. 10th Int. Model. Conf.*, Lund, Sweden, Vol. 96, pp.27–33, 10–12 March.

Blochwitz, T. et al. (2011) 'The functional mockup interface for tool independent exchange of simulation models', in *Proceedings of the 8th International Modelica Conference*, pp.105–114.

Blockwitz, T. et al. (2012) 'Functional mockup interface 2.0: the standard for tool independent exchange of simulation models', *Proc. 9th Int. Model. Conf.*, Munich, Ger., Vol. 76, pp.173–184, 3–5 September.

Bours, R., Rauf, K. and Kietlinski, K. (2013) 'A method for developing AEB systems based on integration of virtual and experimental tools', *23rd Int. Tech. Conf. Enhanc. Saf. Veh.*, pp.1–8.

Casoli, P., Gambarotta, A., Pompini, N. and Riccò, L. (2014) 'Development and application of co-simulation and 'control - oriented' modeling in the improvement of performance and energy saving of mobile machinery', *Energy Procedia*, Vol. 45, pp.849–858.

Chen, A., Dai, X. and Hu, S. (2011) 'Co-simulation study on EPS system based on ADAMS and MATLAB', *Proc. – 3rd Int. Conf. Meas. Technol. Mechatronics Autom. ICMTMA 2011*, Vol. 1, pp.791–794.

Chen, W., Klomp, M. and Ran, S. (2018) 'Real-time co-simulation method study for vehicle steering and chassis system', *IFAC-PapersOnLine*, Vol. 51, No. 9, pp.273–278.

Domenici, A., Fagiolini, A. and Palmieri, M. (2018) 'Integrated simulation and formal verification of a simple autonomous vehicle', *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 10729 LNCS, pp.300–314, ISBN 978-3-319-74781-1 [online] https://doi.org/10.1007/978-3-319-74781-1_21.

Dyer, A., Pagerit, S., Datar, M., Mehr, D. and Negrut, D. (2010) 'A co-simulation environment for virtual prototyping of ground vehicles', *SAE Tech. Pap. Ser.*, Vol. 1, pp.1–9.

Eckert, J.J., Santiciolli, F.M., Costa, S., Corrêa, F.C., Dionísio, H.J. and Dedini, F.G. (2014) 'Vehicle gear shifting co - simulation to optimize performance and fuel consumption in the Brazilian standard urban driving cycle', in *XXII Simpósio Internacional de Engenharia Automotiva*, Vol. 1.

Feng, Y. and Zhang, Q. (2012) 'Multi-body dynamical modeling and co-simulation of active front steering vehicle', in *Proceedings of The 2nd International Conference on Computer Application and System Modeling*, pp.371–375.

Fitzgerald, J. et al. (2010) 'Collaborative modelling and co-simulation in the development of dependable embedded systems', *Integrated Formal Methods – IFM*, pp.12–26.

Fleissner, F. and Eberhard, P. (2009) 'A co-simulation approach for the 3D dynamic simulation of vehicles considering sloshing in cargo and fuel tanks', *Pamm*, Vol. 9, No. 1, pp.133–134.

Fletcher, T., Kalantzis, N., Cary, M., Lygoe, B., Pezouvanis, A. and Ebrahimi, K. (2018) 'Automated model based engine calibration procedure using co-simulation', in *4th Biennial International Conference on Powertrain Modelling and Control (PMC 2018)*, p.118.

Gaaloul, S., Dang, H.A., Kashif, A., Delinchant, B. and Wurtz, F. (2013) 'A new co-simulation architecture for mixing dynamic building simulation and agent oriented approach for users behaviour modelling', in *Proceedings of BS 2013: 13th Conference of the International Building Performance Simulation Association*, pp.3225–3233.

Ha, H., Kim, J., Chung, S. and Lee, J. (2013) 'Advanced VDC simulations of in-wheel electric vehicle using carsim and simulink', *World Electr. Veh. J.*, Vol. 6, No. 1, pp.95–99.

Hallqvist, R., Braun, R. and Krus, P. (2017) 'Early insights on FMI-based co-simulation of aircraft vehicle systems', *Proc. 15th Scand. Int. Conf. Fluid Power, 15th Scand. Int. Conf. Fluid Power, Fluid Power Digit. Age, SICFP'17*, 7–9 June, Linköping, Sweden, Vol. 144, pp.262–270.

Han, S., Chao, Z. and Liu, X. (2017) 'Research on the effects of hydropneumatic parameters on tracked vehicle ride safety based on cosimulation', *Shock Vib.*, Vol. 2017, Article ID 1256536 | https://doi.org/10.1155/2017/1256536.

Hareesha, P.B.Y. and Bharath, M.S. (2015) 'Flexible body integration and co-simulation of double wishbone suspension system', *Int. Res. J. Eng. Technol.*, Vol. 2, No. 7, pp.1107–1113.

He, T. and Peng, L. (2010) 'Application of neuron adaptive PID on DSPACE in double loop DC motor control system', *2010 Int. Conf. Comput. Control Ind. Eng. CCIE 2010*, Vol. 2, pp.257–260.

He, X., Yang, K., Ji, X., Liu, Y. and Deng, W. (2017) 'Research on vehicle stability control strategy based on integrated-electro-hydraulic brake system', in *WCXTM 17: SAE World Congress Experience*.

Himmler, A., Pillekeit, A., Loyer, B., Mouvand, S. and Dezobry, V. (2018) 'Joint application of FMI- and FPGA-based models for real-time simulation of aircraft systems', *AIAA Model. Simul. Technol. Conf. 2018*, No. 209959.

Honek, M., Csambál, J., Wojnar, S., Kopačka, M., Šimončič P. and Lauko, M. (2010) 'Rapid control prototyping system dspace used for control of combustion engine processes', *Technical Computing Bratislava Conference*.

Hong, J., Kim, S. and Min, B. (2009) *Drivability Development based on CoSimulation of AMESim Vehicle Model and Simulink HCU Model for Parallel Hybrid Electric Vehicle*, SAE Tech. Pap, USA.

Hübner, K. (2018) *Architecture and Interface Specification of the Co-Simulation Environment*, INFRAMIX, European Union.

Ibrahim, A., Math, C.B., Goswami, D., Basten, T. and Li, H. (2018) 'Co-simulation framework for control, communication and traffic for vehicle platoons', *Proc. – 21st Euromicro Conf. Digit. Syst. Des., DSD 2018*, pp.352–356.

Innerwinkler, J.Z.P., Stettinger, G., Weissnegger, R., Derse, C. and Aydemir, E. (2018) *Modular Co-Simulation Architecture Plan*, TrustVehicle, European Union.

Jang, B.C. and Choi, G. (2007) 'Co-simulation and simulation integration for a full vehicle dynamic system', *Math. Comput. Model. Dyn. Syst.*, Vol. 13, No. 3, pp.237–250.

Kalantzis, N., Fletcher, T., Pezouvanis, A., Ebrahimi, K., Cary, M. and Lygoe, B. (2018) 'Modelling environment for holistic vehicle simulation', in *4th Biennial International Conference on Powertrain Modelling and Control (PMC 2018)*, p.120.

Karvonen, A. and Thiringer, T. (2015) 'Co-simulation and harmonic analysis of a hybrid vehicle traction system', *2015 IEEE Veh. Power Propuls. Conf. VPPC 2015 – Proc*.

Khadr, A., Houidi, A. and Romdhane, L. (2013) 'Development of co-simulation environment with ADAMS/Simulink to study maneuvers of a scooter', in Haddar, M., Romdhane, L., Louati, J. and Ben Amara, A. (Eds.): *Design and Modeling of Mechanical Systems. Lecture Notes in Mechanical Engineering*, pp.37–43, Springer, Berlin, Heidelberg.

Khan, I.M., Datar, M., Sun, W., Festag, G., Bin Juang, T. and Remisoski, N. (2017) 'Multibody dynamics cosimulation for vehicle NVH response predictions', *SAE Int. J. Veh. Dyn. Stability, NVH*, Vol. 1, No. 2, pp.131–136.

Klein, S. et al. (2017) 'Engine in the loop: closed loop test bench control with real-time simulation', *SAE Int. J. Commer. Veh.*, Vol. 10, No. 1, pp.2017-01–0219.

Kyllingstad, L.T., Rindarøy, M. and Eirik, D. (2017) 'Distributed co-simulation of maritime systems and operations', *CoRR*, Vol. 141, No. 1..

Le Marrec, P. et al. (1998) 'Hardware, software and mechanical cosimulation for automotive applications', in *Ninth International Workshop on Rapid System Prototyping*, Cat. No. 98TB100237.

Levesley, M.C., Ramli, R., Stembridge, N. and Crolla, D.A. (2007) 'Multi-body co-simulation of semi-active suspension systems', *Proc. Inst. Mech. Eng. Part K J. Multi-body Dyn.*, Vol. 221, No. 1, pp.99–115.

Li, G., Wang, T., Zhang, R., Gu, F. and Shen, J. (2015) 'An improved optimal slip ratio prediction considering tyre inflation pressure changes', *J. Control Sci. Eng.*, Vol. 2015, D 512024 | https://doi.org/10.1155/2015/512024.

Li, K. and Xiao, B. (2014) 'Research on adaptive feedforward control strategy of engine based on GT-power and simulink simulation', *26th Chinese Control Decis. Conf. CCDC 2014*, pp.398–401.

Li, S., Zhao, L. and Yang, C. (2014) 'Co-simulation study for fuzzy ESP control strategy on vehicle', *Open Mech. Eng. J.*, Vol. 8, pp.682–688.

Li, Y., Deng, H., Xu, X. and Wang, W. (2019) 'Modelling and testing of in-wheel motor drive intelligent electric vehicles based on co-simulation with Carsim/Simulink', *IET Intell. Transp. Syst.*, Vol. 13, No. 1, pp.115–123.

Maharun, M., Bin Baharom, M. and Mohd, M.S. (2013) 'Modelling and control of 4WD parallel split hybrid electric vehicle converted from a conventional vehicle', *World J. Model. Simul.*, Vol. 9, No. 1, pp.47–58.

Mikelsons, L. and Samlaus, R. (2017) 'Towards Virtual Validation of ECU Software using FMI', in *Proceedings of the 12th International Modelica Conference*, 15–17 May, pp.307–311.

Özener, O. and Allouchery, L. (2017) 'Istanbul metrobus line fuel consumption optimization via 3d road model by using AVL cruise & IPG truck maker co-simulation', *Int. J. Adv. Automot. Technol.*, Vol. 1, No. 2, pp.100–104.

Park, G., Son, B., Kum, D., Lee, S. and Kwak, S. (2012) 'Dynamic modeling and simulation for battery electric vehicles under inverter fault conditions', *Appl. Mech. Mater.*, Vols. 110–116, pp.3007–3015.

Pedersen, N. et al. (2015) 'Co-simulation of distributed engine control system and network model using FMI & SCNSL', *IFAC-PapersOnLine*, Vol. 48, No. 16, pp.261–266.

Pedersen, N., Lausdahl, K., Sanchez, E.V., Larsen, P.G. and Madsen, J. (2017) 'Distributed co-simulation of embedded control software with exhaust gas recirculation water handling system using INTO-CPS', *SIMULTECH 2017 - Proc. 7th Int. Conf. Simul. Model. Methodol. Technol. Appl., no. Simultech*, pp.73–82.

Peng, J., Fan, H., He, H. and Pan, D. (2015) 'A rule-based energy management strategy for a plug-in hybrid school bus based on a controller area network bus', *Energies*, Vol. 8, No. 6, pp.5122–5142.

Porlán, J.V. (2016) *Co-Simulation Methodologies for Hybrid and Electric Vehicle Dynamics*, Universidad Politécnica De Madrid, Madrid, Spain.

Reyneri, F.R.L.M., Bellei, E., Bussolino, E. and Mari, L. (2002) 'Codesign and cosimulation of automotive systems based on Matlab/Simulink', in *Seminario Anual de Automática, Electrónica Industrial e Instrumentación*, May, SAAEI.

Shojaei, S., McGordon, A., Robinson, S., Marco, J. and Jennings, P. (2017) 'Developing a model for analysis of the cooling loads of a hybrid electric vehicle by using co-simulations of verified submodels', *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, Vol. 232, No. 6, p.095440701770709.

Siegele, D., Ochs, F. and Feist, W. (2014) *Modelling and Simulation of Façade Integrated Active Components with MATLAB/SIMULINK*, Institute of Construction and Material Science, Unit for Energy Efficient Buildings, Passive House Institute, Darmstadt, Germany, pp.198–205.

Sweafford, T., Yoon, H-S., Wang, Y. and Will, A. (2012) 'Co-simulation of multiple software packages for model based control development and full vehicle system evaluation', *SAE Int. J. Passeng. Cars - Mech. Syst.*, Vol. 5, No. 1, pp.702–714.

Van Dong, N., Thai, P.Q., Duc, P.M. and Thuan, N.V. (2019) 'Estimation of vehicle dynamics states using Luenberger observer', *Int. J. Mech. Eng. Robot. Res.*, Vol. 8, No. 3, pp.430–436.

Van Schijndel, A.W.M.J. (2014) 'A review of the application of SimuLink S-functions to multi domain modelling and building simulation', *J. Build. Perform. Simul.*, Vol. 7, No. 3, pp.165–178.

Wu, T., Han, K., Pei, L. and Zhao, C. (2014) 'Co-simulation study of coordinated engine control focusing on tracked vehicle shift quality', *J. Autom. Control Eng.*, Vol. 2, No. 2, pp.160–165.

Xie, F., Wang, J. and Wanga, Y. (2011) 'Modelling and co-simulation based on AMESim and Simulink for light passenger car with dual state CVT', in *Procedia Engineering (2011)*, Vol. 16, pp.363–368.

Zhang, B. et al. (2017) *Component Tests based on Vehicle Modeling and Virtual Testing*, SAE Tech. Pap. 2017-01-0384, USA.

Zhang, Y., Xia, M.M., Qin, J.Y. and Zhang, H. (2010) 'Research on Co-simulation using ADAMS and MATLAB for automobile active suspension system', in *ICCASM 2010 – 2010 International Conference on Computer Application and System Modeling, Proceedings*, Vol. 14, pp.366–370.

Zhao, S. and Zhu, L. (2018) 'Cruise control system based on joint simulation of CarSim and Simulink', *OALib*, Vol. 5, No. 7, pp.1–8.