
Co-Simulation Platform for Diagnostic Development of a Controlled Chassis System

Siddharth D'Silva, Padma Sundaram and Joseph G. D'Ambrosio
Delphi Corporation

**Reprinted From: Safety Critical Systems
(SP-2029)**

ISBN 0-7680-1633-9



9 780768 016338

SAE *International*[™]

**2006 SAE World Congress
Detroit, Michigan
April 3-6, 2006**

The Engineering Meetings Board has approved this paper for publication. It has successfully completed SAE's peer review process under the supervision of the session organizer. This process requires a minimum of three (3) reviews by industry experts.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

For permission and licensing requests contact:

SAE Permissions
400 Commonwealth Drive
Warrendale, PA 15096-0001-USA
Email: permissions@sae.org
Tel: 724-772-4028
Fax: 724-776-3036



For multiple print copies contact:

SAE Customer Service
Tel: 877-606-7323 (inside USA and Canada)
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org

ISSN 0148-7191

Copyright © 2006 SAE International

Positions and opinions advanced in this paper are those of the author(s) and not necessarily those of SAE. The author is solely responsible for the content of the paper. A process is available by which discussions will be printed with the paper if it is published in SAE Transactions.

Persons wishing to submit papers to be considered for presentation or publication by SAE should send the manuscript or a 300 word abstract to Secretary, Engineering Meetings Board, SAE.

Printed in USA

Co-Simulation Platform for Diagnostic Development of a Controlled Chassis System

Siddharth D'Silva, Padma Sundaram and Joseph G. D'Ambrosio
Delphi Corporation

Copyright © 2006 SAE International

ABSTRACT

This paper discusses the development and application of a closed-loop co-simulation platform for a controlled chassis system. The platform is comprised of several software packages, including CarSim® (MSC Corporation), AmeSim® (ImaGine Software Corporation), MATLAB®/SIMULINK® (Mathworks Corporation). The platform provides the ability to quickly evaluate enhancements to existing algorithms and to evaluate new control or diagnostic concepts, making it a rapid medium for development, testing and validation. The co-simulation platform was configured with real vehicle calibration data and used to test the validity/limitations of a simple model-based sensor diagnostics strategy. Using this approach, it was possible to quickly check for performance issues and consider needed corrections or enhancements without incurring the time and cost burden associated with in-vehicle testing.

INTRODUCTION

There is a trend in the automotive industry towards an increasing number of safety-related electronic systems that directly contribute to active vehicle operation [1]. These applications will increase overall vehicle safety by assisting the driver with routine tasks and in critical situations. Advances in embedded system technology, microelectronics and control systems have enabled automotive manufacturers to design electronic systems that introduce new features in vehicles [2], improve their performance and increase safety. Antilock brake systems [3], electronic throttle control (ETC) [4], and active rollover protections [5-6] are examples of such technologies available today.

Electronic systems are increasingly being developed with coordinated sensors, actuators, microcomputers and information processing units for the engine, drive train, suspension, steering and brake systems in next generation automotive systems to achieve enhancements in vehicle comfort, feel, fuel efficiency, and safety. Such advanced systems can have several interacting software and hardware components including

electronic control units, hydraulic modules, and motor controllers etc., which in real time conditions often exhibit dynamic behavior. Although many of these systems help provide significant improvements in vehicle safety, unexpected interactions among the software, the hardware, and the environment may lead to situations of potential concern. The increasing demand for active safety systems in current and next generation vehicles motivates the consideration of robust diagnostic strategies. Depending on the functions to be implemented, such systems may need to rely on the integrity of the on-board inertial sensors to carry out their tasks. Traditionally, validation of inertial sensor diagnostic strategies has involved significant in-vehicle testing. As with all in-vehicle testing, significant time and resource costs are incurred, and additional costs associated with ensuring test-driver safety may be necessary.

With the increasing complexity of modern automotive systems, system modeling and simulation techniques are gaining increased significance to address a variety of system development issues including design robustness, safety and reliability. In this paper, we describe an integrated co-simulation model that is capable of simulating subsystem and vehicle behavior under different operating conditions. The co-simulation environment includes a vehicle model, a subsystem software model and a subsystem actuator model in the form of Dynamic Link Library (DLL) functions. Co-simulation of the subsystems was conducted with the vehicle in the loop to depict the real dynamic behavior of the vehicle under different environmental conditions. The above co-simulation platform was configured with real vehicle calibration data and used to test the validity/limitations of a simple model-based sensor diagnostics strategy. Using this approach, it was possible to quickly check for performance issues and consider needed corrections or enhancements without incurring the time and cost burden associated with in-vehicle testing. For the diagnostic development task described in this paper, the co-simulation platform provided a reliable and repeatable environment for chassis system product development. Our experience indicates that such an environment can potentially

shorten the system development cycle, reduce time and resources normally allotted for actual vehicle testing, and give the user a design environment that is safer, more cost efficient and robust.

THE SIMULATION MODEL

In this section, we provide an overview of the simulation model for the subsystem and the environment. The subsystem (see Figure 1) is a safety critical system consisting of a software component, which resides in an Electronic Control Unit (ECU), and an actuator component. The ECU receives input signals from several sensors in the vehicle including vehicle speed, vehicle attitude and driver intent. The software in the ECU is programmed to compute an actuator command based on the input sensor signals.



Figure 1 — Subsystem Layout

In order to build a simulation model that depicts a real time system we must simulate the actual dynamic conditions that the subsystem is exposed to in the vehicle. As such, the simulation environment should be comprised of the following:

1. A model that closely represents the actual vehicle the subsystem belongs to
2. A model that captures the functionality of the software logic and is able to closely imitate the software behavior, and
3. A model that emulates the actuator dynamics.

Figure 2 represents the high level layout of the simulation setup consisting of the three system blocks mentioned above.

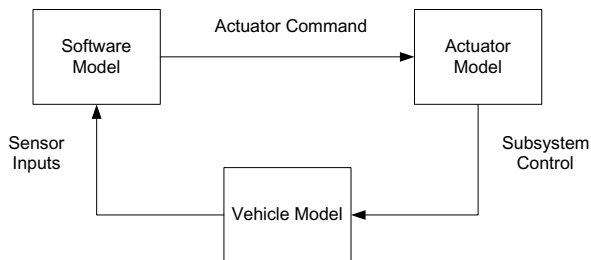


Figure 2 — Simulation Platform

Building a co-simulation environment requires that the different blocks of the simulation model co-exist and work together. There are several simulation and math solver tools available and the choice of the right tools for this task is significant. It is important that the

modeling and control tools, solvers, and the related program libraries have compatible interfaces and can integrate with each other seamlessly and with minimal effort from the model developers. For the simulation setup being described, three different tools were used from three different vendors:

1. MATLAB[®], Simulink[®] tools from *Mathworks Corporation*,
2. CarSim[®] simulation tool from *Mechanical Simulation Corporation (MSC)*, and
3. AmeSim[®] Simulation tool from *IMAGINE Corporation*.

As described next, Simulink[®] facilitates co-simulation among the different simulation tools. Simulink[®] models the subsystem control algorithm, CarSim[®] models the vehicle environment, and AmeSim[®] models the actuator system (see Figure 3).

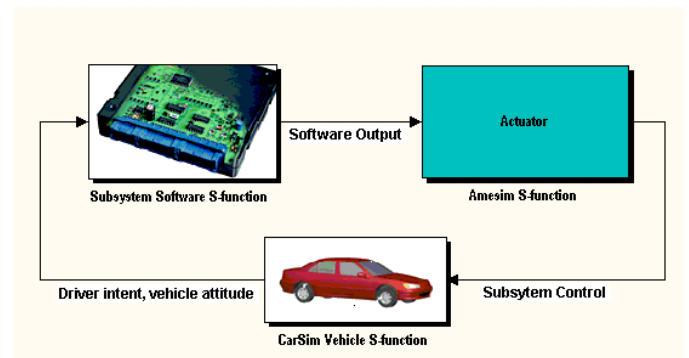


Figure 3 —Co-Simulation Model in Simulink[®]

MATLAB[®]/Simulink[®] was chosen as a common shell for building the simulation model because of its ability to support and interface seamlessly with the different DLLs provided from other tools. Simulink[®] provides a common platform that can integrate all the three system blocks of the simulation model into a single simulation environment. Simulink uses block diagrams to model dynamic systems and provides an overall modeling environment that provides a co-simulation capability that is compatible with many simulation programs. It provides the environment where other simulation models from third party applications can execute within the Simulink shell as a S-Function. A S-Function is a system function that has been compiled to run in the Simulink[®] environment. On a Microsoft Windows[®] platform, these system functions are compiled into dynamically linked libraries (DLLs). The DLLs do not need their original files in the working directory at runtime. Both CarSim[®] and AmeSim[®] provide a DLL to represent the vehicle model and the actuator model respectively. The DLL can be included in the Simulink[®] environment in the form of an S-Function [7].

CarSim[®] can be used for simulating and animating the behavior of four-wheel vehicles. It provides dynamic vehicle simulation with open and closed loop inputs, both from the driver and the environment. The user may define a three-dimensional environment in which the vehicle travels, as well as various parameters of the vehicle. CarSim[®] can provide graphical outputs of specified parameters, a data file, or an animation of the vehicle in its environment.

AmeSim[®] is advanced modeling simulation software which supports simulation based on library elements. It offers an environment for simulating the dynamics of the actuator. It enables the user to build an actuator model by plugging in the physical, mechanical, hydraulic and/or electrical characteristics. With these three software packages running in a co-simulation environment, it was possible to perform closed-loop modeling of the vehicle control system. All individual blocks of the simulation model were validated against real test data. In the following sections, details of the simulation model are discussed.

VEHICLE MODEL

The vehicle model was developed using CarSim[®] produced by Mechanical Simulation Corporation (MSC). CarSim[®] provides several default vehicle models within its standard library. The intent was to develop a vehicle model that accurately behaves like the real test vehicle that was being used for the subsystem development. The test vehicle data including the lateral and longitudinal tire characteristics, suspension characteristics, brake and steering system parameters was acquired by testing the development vehicle on different maneuvers at different speeds. Other vehicle measurements gathered included sprung and un-sprung vehicle mass parameters, vehicle linear measurements and aerodynamics characteristics. Starting with a default CarSim[®] vehicle model and using the real test data of the vehicle, a vehicle model was built in the CarSim[®] environment that closely matched the actual test vehicle. CarSim[®] simulation software provides the necessary math model equations to simulate the vehicle dynamics.

For closed loop vehicle simulation, a driver model, or a path follower model is necessary. Typically steering models require a driver response time to be associated for the closed loop steering system control. This driver response time typically is a factor of the natural human reaction time and the driver perception time. For the subsystem simulation model, vehicle testing with average drivers was conducted to measure the average driver response time. This time factor was used in the simulation model to simulate the driver steering delay time.

The vehicle simulation model developed in CarSim[®] was validated against real test data during steady state and transient maneuvers on surfaces with different environment conditions including surface friction. The

validation process showed that the behavior and the simulation model of the vehicle were very close to the actual vehicle data in all the aspects of our tests. Figure 4 compares the lateral acceleration, yaw rate and roll angles of the actual vehicle and the co-simulation for a given steering maneuver and vehicle speed. The vehicle model in the co-simulation is tuned until the necessary accuracy is achieved.

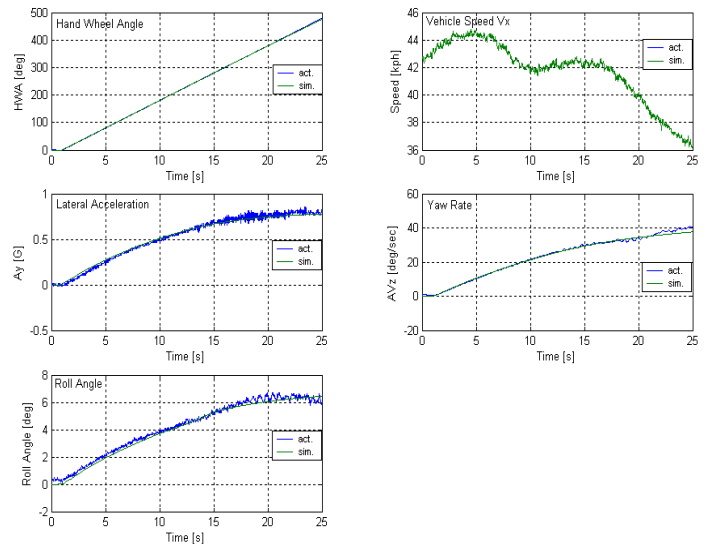


Figure 4 — Validation of Key Vehicle Dynamic Signals in the Co-Simulation

The appropriate vehicle configuration DLL from CarSim[®] is used in the simulation model in Simulink by associating it with a user defined S-Function block. In this paper, the vehicle model in Simulink is referred to as the *vehicle S-function*.

SUBSYSTEM SOFTWARE MODEL

The software for the subsystem algorithm is programmed in the C-language. In order for the simulation to capture accurate behavior of the subsystem software in real time, it is essential that the same software be modeled in the simulation setup as well. Simulink[®] supports the conversion of software into a custom S-function. Simulink[®] provides the necessary templates to create a S-function. These templates were used to program the input and output interfaces and set the scheduler function for the software. Also since the software is in fixed point format, the inputs to the software from Simulink[®] to the software model must be converted to appropriate fixed point data types and similarly the output from the software model has been converted from fixed point to floating point engineering units. The software component is then compiled and built using the MATLAB[®] mex[®] function library. This build process within the MATLAB[®] application produces a DLL for the software component. The MATLAB[®]-produced DLL is then used in the simulation model by

associating it with a user defined S-function. In this paper, this S-Function is called the subsystem software S-function.

ACTUATOR MODEL

The actuator can be a hydraulic or an electronic system. There are several modeling tools to model the actuator system. In our simulation setup AmeSim[®] was used to model the actuator system. AmeSim[®] is Advanced Modeling Environment Simulation software, which allows for simulating actuator dynamics including electrical motor, hydraulic systems through use of several of their libraries. Developing a model of the actuator requires the knowledge of several parameters including electrical and mechanical, hydraulic characteristics, and physical dimensions of the actuator.

AmeSim[®] generates C-files for the actuator model that is built and creates a DLL for the actuator model. The DLL is then used in the simulation model in Simulink by associating it with a user defined S-Function block similar to the software S-function creation mentioned in the last subsection. In this paper, this actuator simulation model is referred to as the actuator S-function.

VALIDATION OF THE CLOSED-LOOP SIMULATION

The three S-functions are connected in Simulink[®] as shown in Figure 3 to provide a closed loop simulation model of the target subsystem. To be useful for analysis, this simulation model must provide an accurate representation of the behavior of the target system. The closed-loop simulation was validated for correct operation by comparing simulation outputs with real data from the test vehicle. Shown in the figure 5 are the comparisons of outputs of a subsystem and the actuator for a given steering maneuver and vehicle speed. Once again the comparisons indicate a very high fidelity co-simulation.

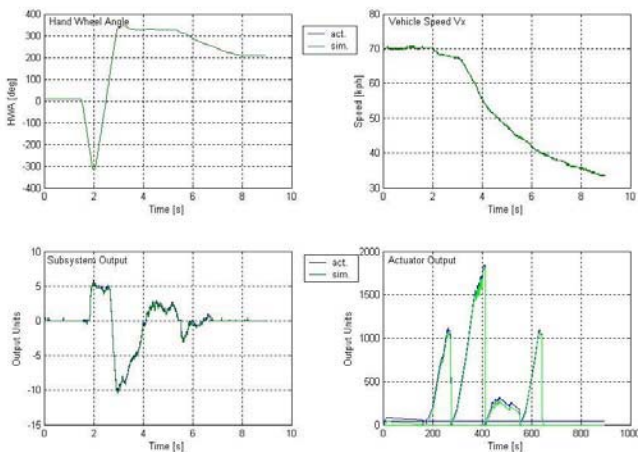


Figure 5 — Validation Results of the Co-Simulation Model

THE PROPOSED DIAGNOSTIC STRATEGY

Vehicle dynamics control can be classified as safety critical because it assists the control functions of certain critical vehicle subsystems like brakes, steering and suspension. Since a vehicle dynamics controller takes critical sensor signals as its inputs and sends control commands to the individual connected subsystems, the validity of the sensor signals becomes critical. This entails that the sensors be reliable. Depending on the criticality of the sensed signal, additional sensors may or may not be needed to achieve the required dependability at the system level. Critical sensors may require some form of redundancy either in the form of a built-in-self-test (BIST) and/or internal replication. It is desirable to have either a fail silent sensor or be able to achieve the fail silent property by replicating the sensor. In the latter case, the values of the sensors are collected and analyzed by an intelligent unit that makes a decision of which sensor value to use in further calculations. Again, the required degree of replication is dependent on the criticality of the sensor.

Redundancy can be achieved through physically redundant sensors; however, physical redundancy increases size, packaging and bill of materials. Under some conditions redundancy of sensors may be required to satisfy the safety requirements of high integrity applications. Under other conditions, it may be possible to satisfy the system safety requirements by other means like analytical redundancy. Traditionally, knowledge based diagnostics techniques have been widely used. These techniques depend on the designer's knowledge and understanding of the system such as operating range of sensor values, the expected rate of change in sensor outputs etc. In some cases, these techniques may not be always sufficient. Self-checking sensors, in production today may not be robust enough to satisfy the safety requirements of critical systems. This motivates the need for some other intelligent solution.

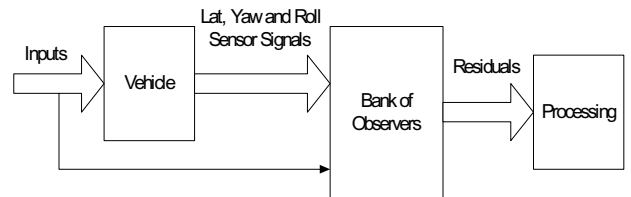


Figure 6 – The Proposed Diagnostic Strategy

Analytical redundancy can be achieved by modeling small portions of the vehicle dynamics and estimating some measurable states (sensor outputs), based on other, directly measured signals. This leads to a bank of observers (estimators) that can estimate the outputs of certain sensors by using information from other sensors, which can be used to monitor the health of the important sensors. In the subsections to follow, a model-based approach is provided to achieve this analytical

redundancy. An appropriate linear model is chosen that relates the sensor signals in the desired operating conditions. The diagnostic capability is completely dependent on the model selected and, thus, much care was taken to develop a model that accurately emulates the actual vehicle behavior in most operating conditions. Next, observer design is used to develop a bank of estimators based on the selected model and the sensor outputs. Figure 6 represents the model-based strategy proposed for sensor diagnostics. The strategy is geared towards monitoring the health of the lateral acceleration, yaw rate and roll rate sensors. The subsections that follow explain the construction of the diagnostic unit and outline the diagnostic principle

SELECTED VEHICLE MODEL

Modeling of vehicle dynamics is a varied and a widely researched topic. In the past, researchers used the laws of physics and vehicle dynamics to model small portions of the vehicle [8]. Depending on the application, these models can vary from complex and highly nonlinear to simple and linear. Recently, there has been increased effort on vehicle stability control [9, 10], especially roll and yaw. Complex seven [11], eight, nine and even 104 [12] degree-of-freedom (DOF) nonlinear models have been used to model different vehicle ride and handling aspects, but these models are too complex for vehicle electronics. Simpler, yet accurate models are needed for vehicle control and diagnostics systems. On the other hand, the commonly used bicycle model [13], though simple enough, does not offer the complexity needed for advanced stability control and diagnostics. In this paper the design is based on a linear three DOF vehicle handling model. With lateral displacement, yaw and roll angles as the chosen DOF's, the model dynamically relates the lateral acceleration, yaw and roll rates of the vehicle. This model provides a good description of vehicle handling that is valid in more than 90% of driving conditions.

The equations summarizing the dynamics of a linear 3-DOF vehicle-handling model are:

$$M(\dot{v}_y + v_x \dot{\psi}) + M_s h_s \ddot{\phi} = Y_v v_y + Y_r \dot{\psi} + Y_\phi \phi + Y_\delta \delta \quad (1)$$

$$I_{zzo} \ddot{\psi} + I_{xzzo} \ddot{\phi} = N_v v_y + N_r \dot{\psi} + N_\phi \phi + N_\delta \delta \quad (2)$$

$$L_\phi \phi + L_p \dot{\phi} = M_s h_s (\dot{v}_y + v_x \dot{\psi}) + I_{xxso} \ddot{\phi} + I_{xzso} \ddot{\psi} \quad (3)$$

where, v_y denotes the vehicle lateral velocity, $\dot{\psi}$ the yaw rate, $\dot{\phi}$ the roll rate, ϕ the roll angle and δ the steering input at the road wheel. Also,

$$\begin{aligned} Y_v &= -\frac{C_{\alpha_f} + C_{\alpha_r}}{v_x} \\ Y_r &= \frac{l_r C_{\alpha_r} - l_f C_{\alpha_f}}{v_x} \\ Y_\phi &= C_{\alpha_r} \frac{\partial \delta_r}{\partial \phi} + C_{\gamma_f} \frac{\partial \gamma_f}{\partial \phi} \\ Y_\delta &= C_{\alpha_f} \\ N_v &= \frac{l_r C_{\alpha_r} - l_f C_{\alpha_f}}{v_x} \\ N_r &= -\frac{l_f^2 C_{\alpha_{fr}} - l_r^2 C_{\alpha_r}}{v_x} \\ N_\phi &= l_f C_{\gamma_f} \frac{\partial \gamma_f}{\partial \phi} - l_r C_{\alpha_r} \frac{\partial \delta_r}{\partial \phi} \\ N_\delta &= l_f C_{\alpha_f} \\ L_\phi &= M_s h_s g - k_s \\ L_p &= -c_s \end{aligned}$$

where all the above vehicle parameters are defined in Table 1 and g is the acceleration due to gravity. For convenience, we write equations 1, 2, 3 in matrix form as

$$\underbrace{\begin{bmatrix} M & 0 & M_s h_s & 0 \\ 0 & I_{zzo} & I_{xzzo} & 0 \\ M_s h_s & I_{xzzo} & I_{xxso} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_P \underbrace{\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \\ \ddot{\phi} \\ \dot{\phi} \end{bmatrix}}_x + \underbrace{\begin{bmatrix} -Y_v & M v_x - Y_r & 0 & -Y_\phi \\ -N_v & -N_r & 0 & -N_\phi \\ 0 & M_s h_s v_x & -L_p & -L_\phi \\ 0 & 0 & -1 & 0 \end{bmatrix}}_Q \underbrace{\begin{bmatrix} v_y \\ \dot{\psi} \\ \dot{\phi} \\ \phi \end{bmatrix}}_x = \underbrace{\begin{bmatrix} Y_\delta \\ N_\delta \\ 0 \\ 0 \end{bmatrix}}_R \delta \quad (4)$$

Equation 4 has the form $P\dot{x} + Qx = R\delta$ which can be written in the convenient state space form

$$\dot{x} = \underbrace{(-P^{-1}Q)}_A x + \underbrace{(P^{-1}R)}_B u$$

where the control signal $u = \delta$ and the matrix P , also known and the inertia matrix, is invertible.

Table 1 – Vehicle Parameter Definitions

M_s	Vehicle sprung mass
M	Total vehicle mass
h_s	Height of center of gravity
l_f	Distance of CG to front axle
l_r	Distance of CG to rear axle
I_{xxso}	Sprung mass moment of inertia about roll axis
I_{xzso}	Sprung mass product of inertia about roll axis
I_{zzo}	Total vehicle moment of inertia about z-axis
v_x	Vehicle longitudinal velocity
k_s	Overall suspension stiffness
c_s	Overall suspension damping
C_{α_f}	Front cornering stiffness
C_{α_r}	Rear cornering stiffness
C_{γ_f}	Camber thrust stiffness
$\frac{\partial \gamma_f}{\partial \phi}$	Degree incline change per degree roll
$\frac{\partial \delta_r}{\partial \phi}$	Degree rear steer per degree roll

All the parameters defined in Table 1 can be obtained for any vehicle and so the numeric state space for any vehicle can be developed. Simulation results clearly indicate the validity of the above model in most operating scenarios. Since the model is linear by design, it fails to capture nonlinear operating scenarios. Thus the model is no longer valid when in a skid/spin maneuver. In this paper, we assume that the vehicle stays in the linear region of operation. Lateral acceleration, yaw rate and roll rate can be viewed as the outputs of the model and represented as

$$y_1 = a_y = C_1 x + D_1 \delta$$

$$y_2 = \dot{\psi} = C_2 x$$

$$y_3 = \dot{\phi} = C_3 x$$

where the matrices C_1, C_2, C_3 and D_1 are known. The pairs $(A, C_1), (A, C_2), (A, C_3)$ are observable, which suggests that full state estimators can be constructed from each of the sensed outputs. This is critical to the design of the diagnostic strategy.

OBSERVER CONSTRUCTION

Consider the system

$$\dot{x} = Ax + Bu, \quad y = Cx + Du$$

An observer [14] basically tries to estimate the state vector x from knowledge of input u and output y . The observer continually compares its estimate (as measured via the output equation) with the true plant to create an error signal. This error signal is then used to drive the observer towards asymptotic convergence with the plant. Since the observer's output should track those of the plant, and difference in the outputs, drive the observer, it is also possible to think of the observer as having the same basic structure of the plant, with additional input $y - \hat{y}$. The analysis is as follows, Let

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + L(y - \hat{y}) \\ &= A\hat{x} + Bu + L(y - C\hat{x}) \\ &= (A - LC)\hat{x} + (B - LD)u + Ly \end{aligned}$$

where, L is a matrix of scalar gains that has to be determined. In order to enforce convergence of the observer states \hat{x} to those of the vehicle, we assume the existence of a state error vector $e := x - \hat{x}$. Thus

$$\begin{aligned} \dot{e} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu - (A - LC)\hat{x} - (B - LD)u - L(Cx + Du) \\ &= A(x - \hat{x}) - LC(x - \hat{x}) \\ &= (A - LC)e. \end{aligned}$$

Since the desire was to achieve asymptotic convergence of the states \hat{x} to those of the vehicle, it is required that

$$\lim_{t \rightarrow \infty} e = 0$$

To guarantee asymptotic convergence of states, the matrix L must be chosen wisely. More specifically, L should be chosen such that the eigenvalues of $(A - LC)$ are in the open-left half of the complex plane. Note that the eigenvalues of the observer error system can be arbitrarily placed in the left half complex plane if the pair (A, C) is observable. We use the principal above to construct a lateral acceleration, yaw rate and roll rate based observer where the outputs of each observer are estimates of the other two sensed signals.

RESIDUAL GENERATION

Residuals are generated by comparing the difference between sensor outputs and those of the observers to a threshold. They take values 0 or 1 based on the signal difference. For yaw rate sensor diagnostics, the lat-based yaw estimate, roll-based yaw estimate and the yaw rate sensor value are used to generate residuals Y_1 and Y_2 . Similarly we generate residuals L_1, L_2 for lateral

acceleration and residuals R_1 , R_2 for roll rate sensor diagnostics.

FAULT SIGNATURE TABLE

In the event of a yaw rate sensor fault, the sensor value differs from the estimated yaw rate outputs of the lateral acceleration-based and roll rate-based observers. Also, the yaw rate-based observer would provide faulty estimates of the lateral acceleration and roll rate outputs. However, the non-faulty roll rate and lateral acceleration signals would provide the correct estimates of each other through their respective observers.

Table 2 – Fault Signature Table

$Y_1 + Y_2$	$R_1 + R_2$	$L_1 + L_2$	Faulty Sensor
2	X	X	Yaw Rate
X	2	X	Roll Rate
X	X	2	Lateral Acceleration

Thus while $Y_1 + Y_2 = 2$, $R_1 + R_2 = L_1 + L_2 = 0$ or 1 and this would prove that the fault lies in the yaw rate observer. Similarly, faults in the other two sensors can be isolated. Table 2 provides the signatures to identify the faulty sensor where 'X' denotes a 0 or 1.

DIAGNOSTIC SIMULATION RESULTS

We use the above developed co-simulation platform for the verification of the proposed sensor diagnostic strategy.

MODEL VERIFICATION

The proposed model-based diagnostic strategy relies heavily on the fidelity of the selected vehicle model. This is because any disparities in the vehicle sensor signals and model may result in residuals that erroneously flag a fault. Hence the model must differ from the actual signals within certain tolerance limits. The selected model provides an accurate description of the vehicle in almost 90% of the driving conditions. Most of the time drivers are going straight or making steady state turns. In such scenarios the model closely matches the signals from the vehicle. Transient maneuvers like a sudden double lane change might introduce dynamics that cannot be captured by the vehicle. Figure 7 compares the responses of the vehicle and the 3-DOF model in a double lane change maneuver at a speed of 85 km/h. The slight disparity in signal values is because a 3-DOF model cannot encompass all the subtle dynamics of the actual vehicle. This difference is within vehicle tolerance limits and does not affect the diagnostic capability of the model. However if the same maneuver is done at 100 km/h, the disparities widen and may not be acceptable as will be shown in a later section. Once a model with reasonable fidelity is developed, it can be leveraged in the design of observers that help with diagnostics.

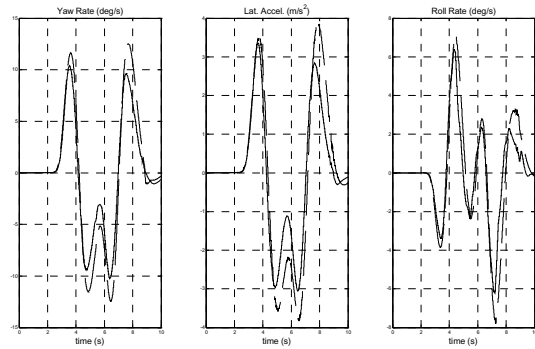


Figure 7 – Comparison of Sensor and Vehicle Model Signals (dotted) for a double lane change at 85 km/h

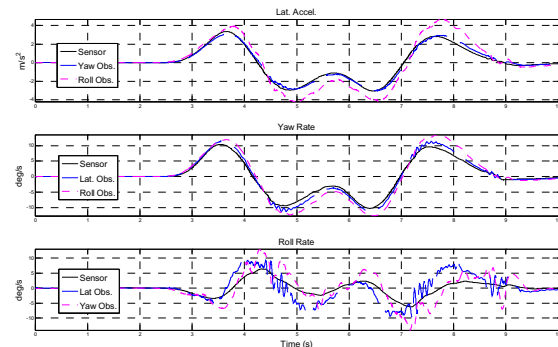


Figure 8 – Comparison of Vehicle Sensor and Observer's Estimated Outputs in the Absence of Faults

For the same maneuver as in figure 7, figure 8, shows the sensor output from the co-simulation (solid) and compares it to the analytical estimates (dotted and dashed) based on the observers when no faults have occurred.

Thus for each sensor output from the co-simulation platform, there are two additional analytically generated signals from the observer bank. This can be interpreted as a type of sensor triplication. Once again, we mention that the disparity seen is a result of using a very simple analytical model. However, for the purpose of diagnostics we compare signal differences to a threshold and the disparity seen here is much lower than that threshold.

YAW RATE SENSOR FAULT SCENARIO

Figure 9 and 10 summarize what happens when a fault is introduced in the yaw rate sensor and how the fault is identified in the software.

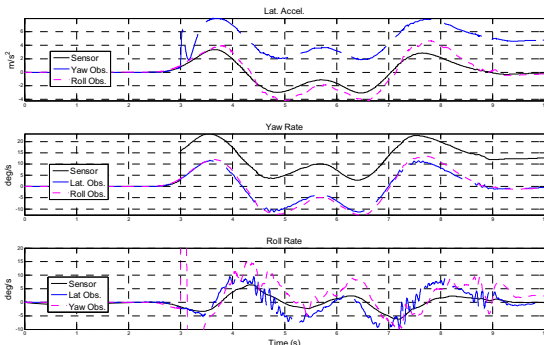


Figure 9 – Comparison of Vehicle Sensor and Observer's Estimated Outputs during a Yaw Rate Sensor Fault of 12 deg/s

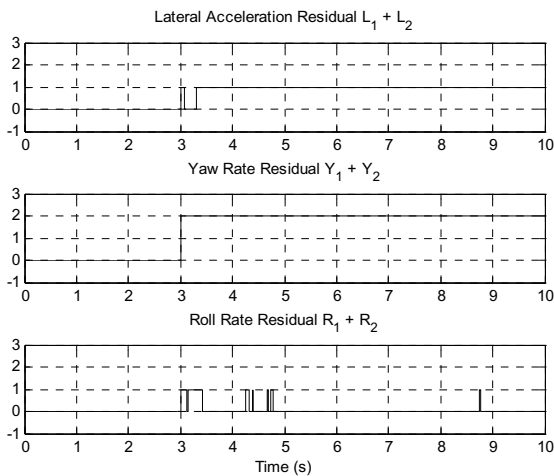


Figure 10 – Yaw Rate Sensor Fault Signature

A yaw rate fault of 12 deg/s was injected at $t = 3$ seconds through the co-simulation platform. The residual ($Y_1 + Y_2$) turns to 2 at 3.01 seconds, and the signature indicates the presence of a yaw rate sensor fault immediately after the fault injection. Similarly, injected faults in the other two sensors are immediately detected and identified based on the signature they match from Table 2

FALSE ALARM SCENARIO

When the diagnostic software erroneously flags the presence of a fault, the condition is known as a false alarm. It is imperative that any diagnostic strategy not only detect faults but also have a very low probability of false alarm. This is because a false alarm may lead to an unwanted termination of the safety critical system. The above strategy is prone to false alarms in situations when the selected model differs significantly from the actual vehicle operation. This may happen during a skid or spin situation, which leads to a sustained nonlinear vehicle operation.

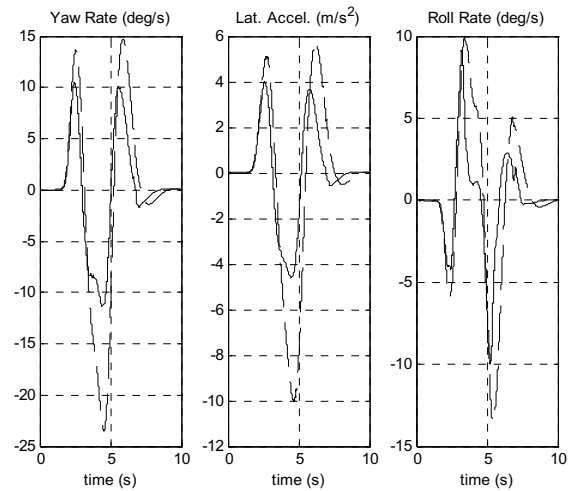


Figure 11 – Comparison of Sensor and Vehicle Model Signals (dotted) for a double lane change at 100 km/h

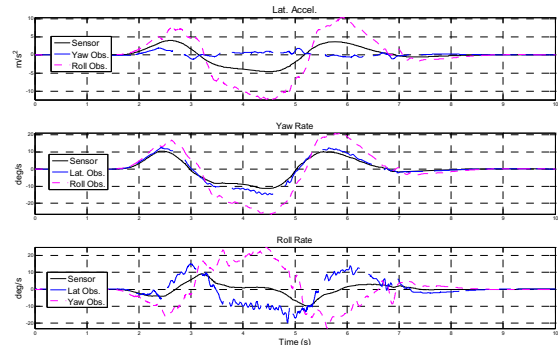


Figure 12 – Comparison of Vehicle Sensor and Observer's Estimated Outputs in the Absence of Faults

Figure 11 compares the responses of the vehicle and the 3-DOF model for a double lane change at a speed of 100 km/h. The maneuver generates nonlinear dynamics that cannot be properly captured by the simple linear 3-DOF model and results in significant difference. This disparity propagates through the observer bank and leads to low fidelity estimates of the sensor signals. This is clearly seen in figure 12. When these estimates are fed into the residual generation unit the signal differences exceed the design threshold leading to residuals signatures that may be misleading. The corresponding residuals plotted in figure 13 cannot be properly interpreted by the diagnostic software. Thus some additional design provisions must be made to accommodate such events and prevent false alarms.

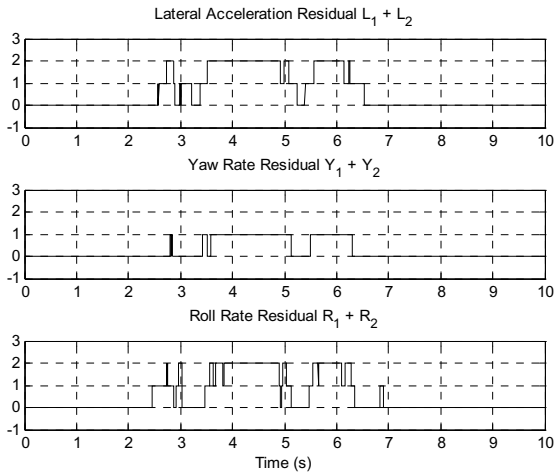


Figure 13 – Residual Signatures during a False Alarm

CONCLUSION

This paper discussed the development of a simulation model for system diagnostics development and analysis. Simulation-based engineering tasks, such as those for robustness studies, diagnostics and system safety analysis, require a simulation model of high fidelity and precision. Setting up such a simulation environment may require significant effort and time investment. Developing the environment requires a collective team effort, including the system safety engineer, the algorithm development engineer, the software engineer and the vehicle test engineer. However, such an initial investment can be justified when the simulation model can be confidently used for multiple system development tasks, such as:

- Controls algorithm development,
- Robustness analysis,
- System safety analysis,
- Diagnostic development,
- Design improvement analysis, and
- Verification and validation.

One significant advantage of simulation-based vehicle analysis is the ability to perform desired vehicle tests at various limit-handling conditions. Performing such activities in a real vehicle may involve risk to the driver, and can increase costs due to the needed additional vehicle safety devices. In these cases, substituting simulation in place of in-vehicle testing can greatly reduce associated risks and costs.

Selection of the appropriate tools for development of a co-simulation platform can be a challenging task. The different simulation and modeling tools must be able to interface and integrate with each other seamlessly without requiring significant integrating effort from development engineers.

This paper presented how a relatively simple vehicle model can be used to diagnose vehicle level signals of

roll rate, yaw rate and lateral acceleration. By setting appropriate thresholds, the diagnostics can be tuned to minimize false alarms, and to maximize the fault detection capability. Although the above presented model based diagnostic strategy shows promising results in the linear region it has certain inherent limitations under nonlinear dynamic conditions. Additional techniques that can detect nonlinear vehicle dynamic conditions are necessary to complement the analytical algorithm so that false triggering of the diagnostics can be avoided when the vehicle is in the nonlinear region.

ACKNOWLEDGEMENTS

The authors wish to thank Aleksander Hac, Edward Bedner, Chester Gryczan, and Andrew Mooradian of Delphi Corporation for their support in the development and validation of the simulation model. The authors also thank Laci Jalics and Song You of Delphi Corporation for their help on some aspects of the paper.

REFERENCES

1. A. Manzone, A. Pincetti, D. De Constantini,, "Fault Tolerant Automotive Systems: An Overview", *Proc. IEEE International On-Line Testing Workshop*, Italy, 2001.
2. E. Touloupis, J.A. Flint and D. Ward, "Safety-Critical Architectures for Automotive Applications", *Electronic Systems and Control Division Research 2003*, Department of Electronic and Electrical Engineering, Loughborough University, UK.
3. A.F. Williams and J.K. Wells, "Driver Experience with Antilock Brake Systems", *Accident Analysis and Prevention*, Vol. 26, pg 807-811, Dec. 1994.
4. H.M. Streib and H. Bischof, "Electronic Throttle Control: An Effective System for Improved Emissions, Fuel Economy and Driveability", *SAE Technical Paper Series*, no. 960338, in R.K. Jurgen, Ed., *Electronic Engine Control Technologies*, Warrendale, PA, SAE:PT-73, 1998.
5. J. Ackerman, T. Bunte, and D. Odenthal, "Advantages of Active Steering for Vehicle Dynamics Control", *Proceedings of 32nd ISATA, Automotive Mechatronics Design and Engineering*, Vienna, pp. 263-270, 1999.
6. A. Hac, "Influence of Active Chassis Systems on Vehicle Propensity to Maneuver-Induced Rollovers", *SAE Technical Paper Series no. 2002010967*, SAE 2002 World Congress, Detroit, Michigan.
7. "Writing S-Functions", Version 4, *The Mathworks Digest*, April 2003.
<http://www.mathworks.es/access/helpdeskr12p1/hel/pdf/doc/simulink/sfunctions.pdf>

8. T.D. Gillespie, "Fundamentals of Vehicle Dynamics", SAE Publications, Warrendale, PA, 1992.
9. B.A. Guvenc, L. Guvenc, E. Ozturk and T. Yigit, "Model Regulator based Individual Wheel Braking Control," *Proceedings of 2003 IEEE Conference on Control Applications*, vol.1, June 2003.
10. M. Durali and A.R. Kassaiezadeh, "Design and Software based Modeling of Anti-Roll System," *Automotive and Transportation Technology Conference*, France, 2002.
11. F. Momiyama, "7 DOF Vehicle Model for Understanding Vehicle Fluctuation during Straight Running," *SAE 2003-01-3412*, Nov. 2003.
12. M. Durali and A.R. Kassaiezadeh, "A Neural Network Approximation of Nonlinear Car Model using Adams Simulation Results," *SAE 2001-01-3324*, Oct. 2001.
13. J. P. Wideberg, "Dynamic Effect of the Non-Rigid Modified Bicycle Model," *Journal of Automobile Engineering*, Vol. 216, No. D9, March 2001.
14. John S. Bay, "Fundamentals of Linear State Space Systems," *McGraw-Hill*, August 1998.

CONTACT

Siddharth D'Silva, Research Engineer, Delphi Corporation, 51786 Shelby Pkwy, Shelby Twp., MI 48315 Phone: (586) 323 9449, Fax: (586) 323 9797, Email: siddharth.d.silva@delphi.com

Padma Sundaram, Senior System Safety Engineer, Delphi Corporation, 12501 E.Grand River, Brighton, MI 49116 Phone: 810-494-2453, Email: padma.sundaram@delphi.com