2019-24-0079 Published 09 Sep 2019

A New Co-Simulation Approach for Tolerance Analysis on Vehicle Propulsion Subsystem

Claudio Mancuso, Domenico Cavaiuolo, and Giuseppe Corbo GM Global Propulsion Systems

Iakovos Papadimitriou Gamma Technologies LLC

Nicolas Brown Synopsys Inc

Citation: Mancuso, C., Cavaiuolo, D., Corbo, G., Papadimitriou, I. et al., "A New Co-Simulation Approach for Tolerance Analysis on Vehicle Propulsion Subsystem," SAE Technical Paper 2019-24-0079, 2019, doi:10.4271/2019-24-0079.

Abstract

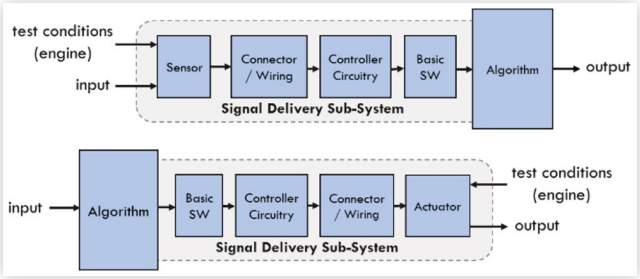
An increasing demand for reducing cost and time effort of the design process via improved CAE (Computer-Aided Engineer) tools and methods has characterized the automotive industry over the past two decades. One of the main challenges involves the effective simulation of a vehicle's propulsion system dealing with different physical domains: several examples have been proposed in the literature mainly based on co-simulation approach which involves a specific tool for each propulsion system part modeling. Nevertheless, these solutions are not fully suitable and effective to perform statistical analysis including all physical parameters. In this respect, this paper presents the definition and implementation of a new simulation methodology applied to a propulsion subsystem. The reported approach is based on the usage of Synopsis SABER as dominant tool for co-simulation: models of electronic circuitry, electro-mechanical components and control algorithm are implemented in SABER to perform tolerance analysis; in addition, a dynamic link with engine plant model developed in GT-SUITE environment has been established via a dedicated procedure. Moreover, a HPC Grid (High Performance Computing Grid) is used with the aim to execute simulations of long engine maneuvers as well as to parallelize jobs while applying Monte-Carlo methods. The overall approach is tested on the active thermal management subsystem of a General Motors internal combustion engine in order to evaluate the robustness of control algorithm against electro-mechanical part variation and software calibration settings.

Introduction

Nowadays, the development cycle of each software/hardware part and component of a vehicle propulsion system is inevitably assisted by a simulation phase which implies the usage of specific CAE tools and analysis: these allow engineers to verify system performance according to design requirements and predict any potential issue on the field with the advance to reduce hardware costs and compress development timing. Moreover, the continuous enhancements of simulator features and computational resources is opening to new breakthroughs and scenarios for the simulation of whole vehicle system and environment. Being one of the greatest automotive company in the world, General Motors has been put large effort, since many years, to create internal processes and dedicated infrastructures capable to fully sustain the virtualization of vehicle manufacturing and testing. For instance, *Bogden et al.* [5] introduced the SDSS (Signal-Delivery Sub-System) process which is intended to perform robust analysis on an engine control subsystem including electronics, algorithm and engine parts modeling. In [4] *Goodwin et al.* applied the process and method to study the impact of electronics and electro/mechanical parts variation on the operation of variable valve timing (VVT) subsystem, modelled in Synopsis SABER environment. In particular, they showed how to accomplish simulation of huge number of statistical samples via Monte-Carlo approach using distributed computing. As further improvement to the SDSS methodology, in [3] authors describe a procedure, referred as CAL-SIL, to include also the control algorithm and calibrations within same simulation environment. All described applications dealt with the simulation of engine subsystems which didn't require the usage of an engine plant model since the interaction of sensors/actuators with thermal, hydraulic and mechanical portion of the engine was obtained using fixed test conditions or experimental profiles (i.e. pressure, flow-rate, rpm, etc..), as reported in figure 1.

Recently, the demand to extend all features developed for SDSS methodology on more complex vehicle and engine subsystems resulted in the need to include also modeling of engine plant within the simulation environment. Usually, models of internal combustion engines are developed in specific CAE simulators, such as GT-SUITE from Gamma Technologies, which allows to properly model, with high level of accuracy, the

FIGURE 1 Block-diagram of simulated subsystems with no engine plant model.



physical behavior of combustion chambers, heat flow and heat exchange with others plant sources. Whereby, the use of a co-simulation approach for the communication and the integration of different propulsion subsystem sub-models at system level is required. In literature there are a lot of examples of co-simulation setups between GT-SUITE and other tools for the engine/vehicle system analysis. For example, in [1] *Sweafford et al.* evaluate the benefits and drawbacks of co-simulation between high fidelity engine model implemented in GT-SUITE and engine control model (medium fidelity) in Simulink for two different scenarios, series and parallel co-simulation. Moreover, *Barasa et al.* in [2] present a similar case study for the Simulink "series" embedded approach to create a vehicle virtual platform for controls and calibration engineers. Nevertheless, it can be well recognized that the usage of Simulink "behavioral" models for electronic circuitry is not suitable to perform robustness analysis on control algorithm and calibrations taking into account all subsystem tolerances. Based on these considerations, the SABER environment described in [3], which consists of high-fidelity models of ECU electrical interfaces and engine EMS components plus an embedded model of control software represents a solid and effective simulation environment to be used in a conjunction with engine GT-SUITE model.

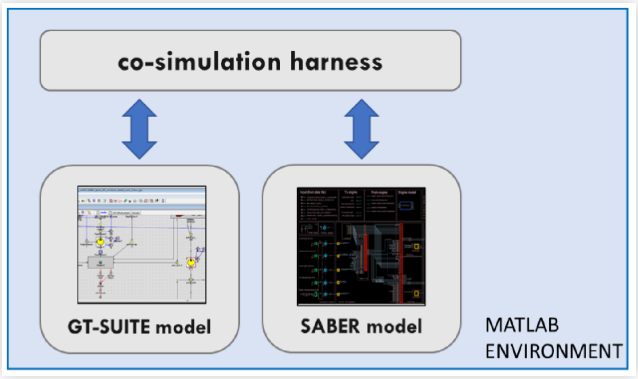
Aim of this paper is to describe the procedure put in place by General Motors in collaboration with Gamma Technologies and Synopsis to create a "direct" link between SABER and GT-SUITE, which excludes the need of a third co-simulation software. In particular, the engine model has been imported in SABER environment as a DLL. For this purpose, two different approaches have been explored: since the entire simulation schematic has been integrated in SABER environment, the simulator acts as dominant tool. Further details about this process will be provided in next sections. The main advantage of that approach is the possibility to submit jobs on a High-Performance Computing (HPC) Grid and parallelize simulation runs while performing statistical analysis (e.g. Monte Carlo).

In order to test and validate this new capability, the methodology has been applied to a practical case study made by a thermal management subsystem of a GM internal combustion engine.

Infrastructure Setup

Most of applications involving the simulation of SABER and GT-SUITE models relies on the use of Matlab/Simulink which

FIGURE 2 Simulink model for co-simulation SABER/GT-SUITE.



implements the communication bus and defines the time step for input and output data exchange between models. In fact, since both tools have the feature to natively communicate with Simulink, it's possible to simply create a basic co-simulation layout in Simulink environment by designing model schematic to be simulated and proper interfaces for input/output variables, as depicted in figure 2.

This solution may have some drawbacks when used in practical cases, especially when the purpose is to run statistical analysis:

- need to have available and dedicated license for each of the three software;
- for any change on input/output variables (data flowing on co-simulation harness), the update of native model and accordingly of Simulink schematic is needed;
- in case of Monte-Carlo analysis, a Matlab script should coordinate the selection of statistical parameter to be set within the native model;
- a complex infrastructure may be implemented for the management of parallel runs.

In order to overcome these problems, an alternative approach is proposed, based on the removal of Simulink and the use of SABER as dominant tool for co-simulation. Therefore, the simulation interface will be inside SABER with the advantage of exploiting all its features regarding the execution of parallel Monte-Carlo jobs, for instance, on a cluster of HPC machines. In particular, two different options are presented:

a. Embedded approach, with GT-SUITE engine model completely "transferred" to SABER;
b. Co-simulation approach with SABER acting as dominant tool;

The appendix A reports all the required tool to follow both approaches a) and b).

a. Embedded Approach

In the embedded approach:

- Saber manages the whole simulation environment
- The GT-SUITE differential equations are solved by the GT-SUITE solver kernel which is encapsulated inside a DLL.

These are the main advantages and drawbacks of this method:

- ✓ The GT-SUITE equations are solved by the GT Solver, although it's encapsulated into the DLL. For this reason, it doesn't require for the final user to have a full GT-SUITE installation.
- × No debugging info
- × No internal GT-SUITE variables can be plot
- × Difficulties to manage licenses in Linux system

The workflow to export a GT-SUITE model to SABER is detailed in Appendix B.

b. Co-Simulation Approach

In the co-simulation approach:

- Saber manages the whole simulation environment
- The GT-SUITE differential equations are solved by the GT-SUITE solver which communicates with SABER through TCP/IP

Also in this case there are a few positive aspects versus some cons:

- ✓ GT-SUITE simulation results stored for post-processing
- ✓ High degree of integration in several OS
- × Full GT-SUITE installation requiring Solver license.
- × Slightly slower simulation speed due to TCP/IT overhead

The workflow to implement this procedure is reported in Appendix C.

Active Thermal Management Subsystem - Case Study

As a case study for the application of implemented simulation setup, the Active Thermal Management (ATM) subsystem of General Motors 3-cylinder 1.5 L diesel engine is considered with the aim to investigate the behavior of control algorithm and calibrations against electrical and electro-magnetic components variation, during some specific vehicle maneuvers. A simplified block-diagram of the overall ATM subsystem is depicted in figure 3: the control strategy relies on information coming from a certain number of sensors (thermistors) mounted on engine head and coolant circuit, and determines the commands to be delivered to valve actuators to optimize the heat exchange between coolant and different parts of engine and vehicle systems, during the engine cycle.

Below the partition related to each portion of the subsystem.

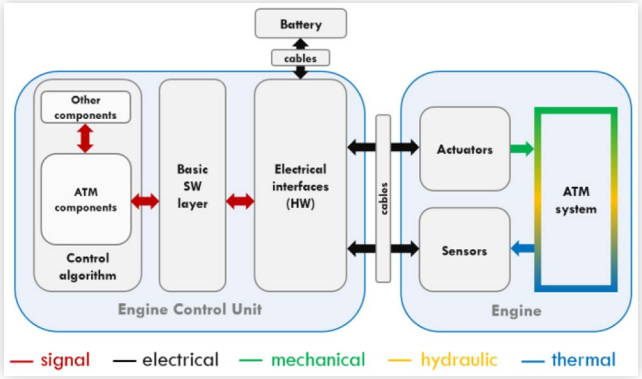
SABER MODELING:

Engine control unit (electrical interfaces + basic sw)
Battery
Wiring harness
Actuators (electro-mechanical portion / position sensor)
Sensors (thermistor)
Control algorithm

GT-SUITE MODELING

Combustion chambers
Coolers
Pipes
Radiator
Pumps
Valves chambers

FIGURE 3 Active Thermal Management (ATM) subsystem schematic block diagram.



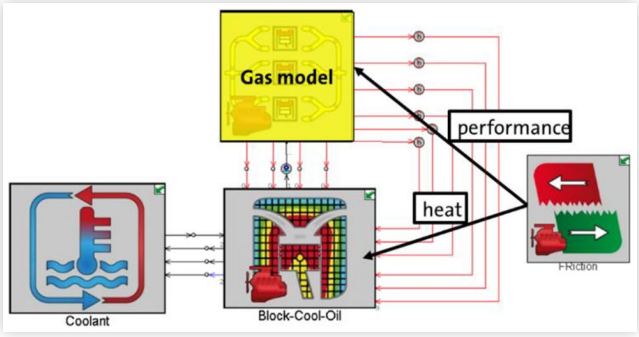
Since the analysis is mainly focused on control system, high-fidelity models of ECM electrical circuits and sensors/actuators components are required, as well as the actual control software and calibrations. On the contrary, an accurate modeling of all engine parts is not needed: therefore, a medium-fidelity engine model, described in next section, is used and then reported in SABER environment following the approach b) previously presented (it has been selected considering the tradeoff among positive and negative aspects).

GT-SUITE Modeling

For the engine part a 1D model was developed with the scope to simulate the engine response and inertia to the coolant system valves and actuators controls. This model is developed in GT-SUITE and it is a complete thermal model coupling hydraulic coolant system, oil system and combustion heat exchange on a map based model (fig. 4).

The model is reflecting the behavior of an engine in a dyno (normally used for initial calibration purpose) and it can simulate the thermal and hydraulic performances (flow rate, pressure and heat rejection) of the 3-cylinder 1.5 L diesel engine. Combustion chamber, base engine components, friction, heat exchange, coolers and radiator models and hydraulic maps or models of the actuators and pumps allow

FIGURE 4 Engine model block map.



the model to simulate different engine conditions, different calibrations and to test a lot of different controls maps.

The info about position of each actuator is requested as model input to simulate the reaction of the engine to the control command on a certain time step basis. As a feedback to actuator controls model, the engine model provides temperature and pressure in the engine points where sensors are located (as in a real engine). Also, other engine and calibration parameters are shared between the two models, as for example vehicle speed, engine rpm and fuel delivery request. The entire engine model has been validated in different conditions (steady-state and transient maneuvers) against experimental measurements performed on an engine running on a dyno bench: temperature, pressure and mass flow comparison showed a good accuracy that is less than 10% (fig. 5) and system time behavior is well caught in almost every condition.

Saber Modeling

The models included in Saber have been developed using library component model released with the Saber package and customized models written in MAST or VHDL-AMS language.

Some of them are:

- Electronic components (Resistors, Capacitors, FETs, BJTs, uC ADC, DC Motor driver IC)
- Thermistors
- Actuators (EMC filter, DC motor, Gear box, SENT positon sensor)

For the main parameters of each Saber model the tolerance and its statistical distribution has been defined.

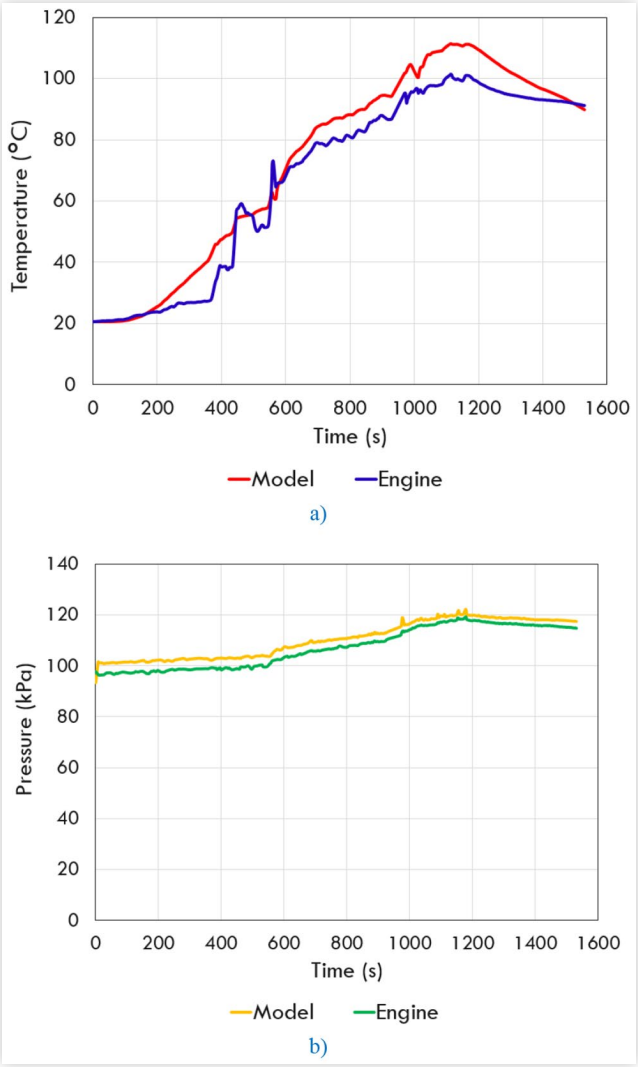
As an example, the Resistors tolerance is composed as follow:

$$Resistor\ tolerance = Nominal\ tolerance + Temp.coefficient + Long\ Term\ stability + Soldering + Lifetime\ drift$$

At the end of the development phase the Saber models are validated against experimental measurements.

Regarding the GM control software, it is developed in GM and for this reason an internal process has been created in order to export the C-code of the algorithm portion in a DLL compatible for Saber.

FIGURE 5 Model correlation and accuracy: a) coolant outlet temperature; b) coolant outlet pressure.



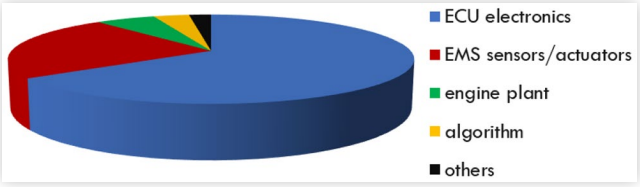
Subsystem Model Validation

Once the 'engine model' has been imported in SABER environment as a DLL and connected to the other system parts, the overall ATM subsystem model has been validated with respect experimental measurements acquired at dynamometer bench.

A nominal simulation of an engine maneuver has been performed on a Windows machine, providing the following simulation ratio K_{SIM} :

$$K_{sim} = \frac{T_{sim}}{T_{actual}} = \frac{206000\,sec}{1179\,sec} = 174 \quad (1)$$

FIGURE 6 Computational complexity comparison.



where T_{sim} is the overall simulation time and T_{actual} the actual duration of maneuver profile to be simulated. That value is strictly linked to the complexity of the entire SABER model, which is mainly affected by high-fidelity models of ECU electronics and EMS components compared to the small computational effort required to simulate the engine plant model (fig. 6).

The main reason why the engine GT-SUITE model requires a lower computation effort compared to the models developed in Saber (ECU electronics and EMS components) is because of the model accuracy. In fact the GT-SUITE model is a 1-D 'map based' behavioral model. While instead the Saber model are based on constitutive equations that govern the physical behavior of each component.

That choice has been dictated by the trade-off among accuracy and computational speed.

As concerns the accuracy of the results, the plot reported below shows the good agreement between measured and simulated valve positions (fig. 7a) and coolant outlet/inlet temperatures (fig. 7b) during a certain time interval of the overall engine cycle.

Simulation Results and Performance on HPC

The validated SABER model of the ATM subsystem has been then configured with input profiles acquired on vehicle tests during four emission driving cycles in order to run predictive Monte-Carlo simulations and detect any potential unexpected behavior of control system on a certain number of samples.

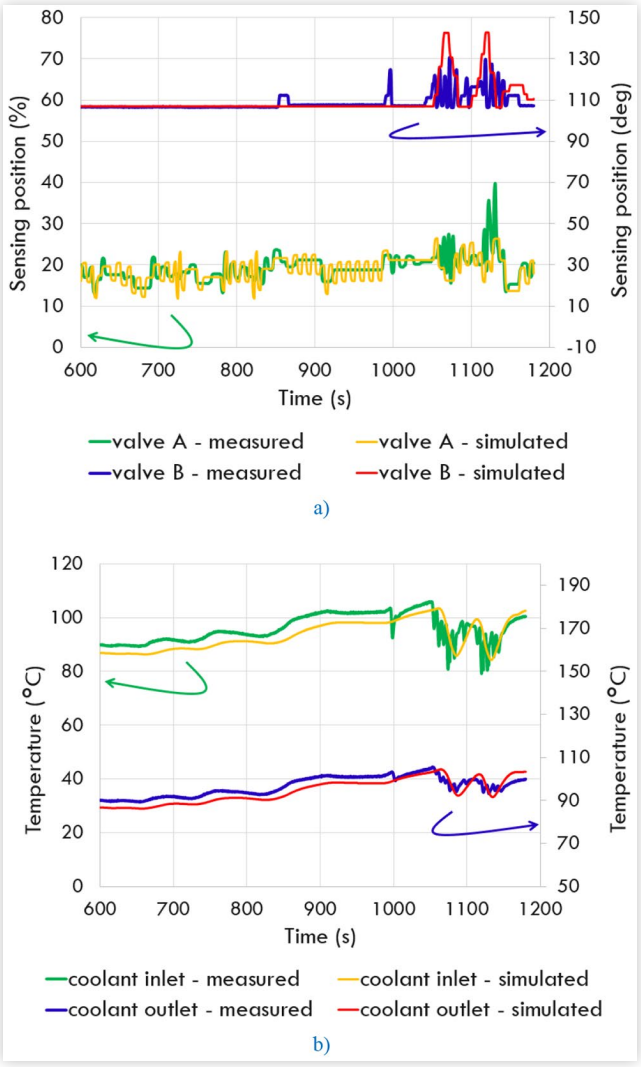
The availability of a cluster of machines running in Linux environment on an High-Performance Computing(HPC) Grid made it possible to execute parallel jobs for the single driving cycle but also to run all driving cycles at the same time: this results in a drastic reduction of the final computational time and ensure that process execution is not interrupted due to machine shutdown, as it may happen when running on a single workstation.

Note that simulation ratio, in case of parallel simulations, for each driving cycle, is calculate as follows:

$$K_{sim_{parcycle}} = \frac{T_{sim_{parcycle}}}{N_r * T_{actual_{cycle}}}, \quad (2)$$

where $T_{sim\_par\_cycle}$ is the total time needed to perform N_r Monte-Carlo runs in parallel and T_{actual_cycle} the actual duration of driving cycle to be simulated. This parameter depends, of course, on characteristics of each simulation (convergence, iterations), on machines available on the cluster and also on resource available on the single machine.

FIGURE 7 ATM SABER subsystem model validation: measured and simulated a) valve positions and b) coolant temperatures.



However, this result can be simply compared with the simulation ratio of a series execution of Monte-Carlo runs: this may be the case when a standard co-simulation approach is used and a structure to effectively manage statistical parallel simulation cannot be implemented:

$$K_{sim\_ser\_cycle} = \frac{\sum_{i=0}^{N_r} t_{sim\_cycle}}{N_r * t_{actual\_cycle}} \quad (3)$$

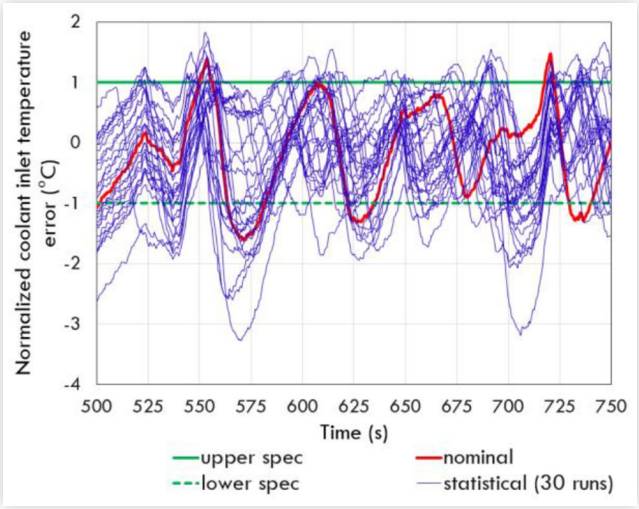
In case of driving cycle #1 the series simulation ratio would be in the order of 5000: in fact, considering that the time for the single run is almost equal to 3days, the total simulation time would be 90days!

The very short time requested to effectively simulate a certain number of ATM control subsystem samples with high accuracy, allowed to analyze the behavior of some system variables in critical conditions. For instance, figure 8 reports the behavior, during a portion of a cycle, of the error between engine the coolant inlet temperature measured by ECM (blue lines) and the target defined by the control algorithm (red line), normalized with respect to the error functional specification:

TABLE 1 Results of Monte-Carlo analysis obtained in the HPC Grid.

		Single run		Multiple runs (parallel)			
Driving cycle (#)	Duration (Tact) (sec)	Simulation time (days)	Simulation ratio (#)	Totalruns (#)	Parallel jobs (#)	Simulation time (days)	Simulation ratio (#)
1	1535	2d 14h	147	30	15	5d 19h	22
2	2355	3d 19h	140	30	20	8d 2h	15
3	820	1d 8h	143	30	10	3d 13h	37
4	699	1d 5h	151	30	10	3d 5h	39

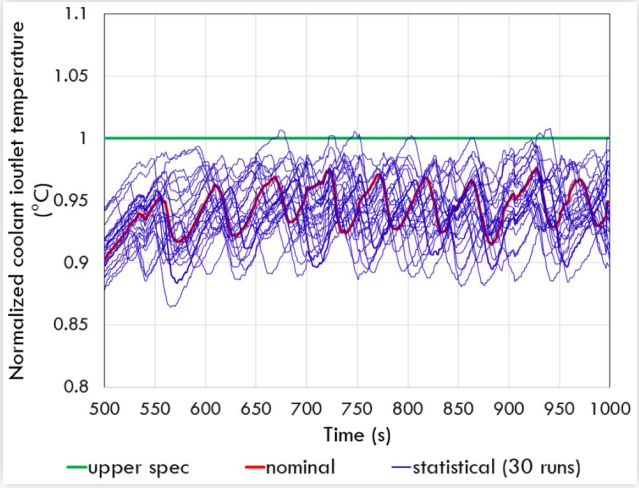
FIGURE 8 Coolant inlet temperature error VS target value normalized with respect to functional specification: statistical and nominal.



results of Monte-Carlo simulations highlight that, in some cases, the error may be larger than expected, so refinement of software calibrations may be needed.

Furthermore, the behavior of ECM measured coolant outlet temperature which is expected not to go over a certain target set by algorithm, is confirmed by statistical simulations, as depicted in figure 9: the coolant outlet temperature,

FIGURE 9 Coolant outlet temperature normalized with respect to maximum target value: statistical and nominal.



normalized with respect to algorithm set maximum limit, always lies below this threshold.

A Pareto analysis has been conducted in order to separate the major causes of the above plots variation. It has been seen that the electrical circuitry demanded to battery voltage monitoring, temperature sensor monitoring and valve position sensing are the main source of temperature error variability.

Conclusion

The new simulation methodology presented in this paper permits CAE engineers to use a unique simulation tool (SABER) to collect multi-domain models (e.g. electrical, fluid, thermal) like for example controller electrical interfaces, engine thermal models, electrification components, SW and calibrations.

This choice has been dictated by the ability to use HPC to perform statistical simulations on a significant population of subsystems. This enable the development of a robust design in an early phase of a development gaining significant advantages in terms of time and cost.

GM is now thinking to the second phase of this project. In fact, recently SABER became compatible with the FMI standard and GT-SUITE already permits to export a model in FMI.

References

1. Sweafford, T., Yoon, H., Wang, Y., and Will, A., "Co-Simulation of Multiple Software Packages for Model Based Control Development and Full Vehicle System Evaluation," *SAE Int. J. Passeng. Cars - Mech. Syst.* 5(1):702-714, 2012, doi:10.4271/2012-01-0951.
2. Barasa, P., Tian, Y., Hardes, S., Owlia, S. et al., "Virtual Engine, Controls, and Calibration Development in Automated Co-Simulation Environment," SAE Technical Paper 2016-01-0090, 2016, doi:10.4271/2016-01-0090.
3. Goodwin, W., Mancuso, C., and Brown, N., "Software Test and Calibration Using Virtual Manufacturing," SAE Technical Paper 2017-01-0536, 2017, doi:10.4271/2017-01-0536.
4. Goodwin, W., Bhatti, A., and Jensen, M., "Designing Automotive Subsystems Using Virtual Manufacturing and Distributed Computing," SAE Technical Paper 2008-01-0288, 2008, doi:10.4271/2008-01-0288.
5. Bogden, D., Grimes, M., Michaels, L., and Amann, R., "Robust Electronic Control System Design Requires Signal Delivery Analysis," SAE Technical Paper 2004-01-0892, 2004, doi:10.4271/2004-01-0892.

Acknowledgments

The authors would like to thank GM engineers: Fabio Autieri, Alfonso Renella, Marcello Rimondi, Francesco Mercuri, Luca Scavone, William Debs, Vincenzo Gesa, Alessandro Grosso, Tim Greger, Dong Yi, William Goodwin for the contribution to this work.

Appendix A - Tool Requirements

- Matlab
- Simulink
- Stateflow
- Simulink Coder
- Stateflow Coder
- Saber
- C/C++ Compiler
- GTSuite

Appendix B - Embedded Approach Workflow

1. In the GT-SUITE model add a Simulink Harness and define the list of inputs and outputs
2. Define the Initial Output values
3. Locate the tab Tools → Export as FMU or MEX (A Simulink model and an S-function *.mex is created at the end of this step)
4. Open the Simulink model → Add Mux and Demux in order to attach Input and Output ports to the S-function.

Now the user shall run the Simulink model to check that it's bug-free and that results are according to the expectation.

Below the remaining steps which can be applied to export a generic Simulink model to Saber.

5. Open Simulink Coder (a.k.a Real Time Workshop)
   a. Setup the Solver tab
   b. Setup the Code Gen tab
6. Build the model (Ctrl+B)

The step #6 involves the usage of Simulink Coder (a.k.a Simulink Coder). This is the official way provided by Mathworks to generate code from Matlab/Simulink models to be used in another computational environment. In this particular case the Simulink Coder shall create:

- Saber MAST template (wrapper to call the DLL using a foreign routine)
- DLL(Dynamic-link library) in Windows OS
- SO (Shared object) in Linux OS
- Saber symbol

Actually, there's an intermediate step which is transparent from the user. In fact, before compiling the final DLL, the Simulink model is converted in C-code. From C-code the TLC (target language compiler) converts to be compatible with Saber.

In figure B1 and B2 it's possible to see the graphical view of the workflow to setup and run a GT-Saber model.

FIGURE B1 Approach a): Embedded approach workflow

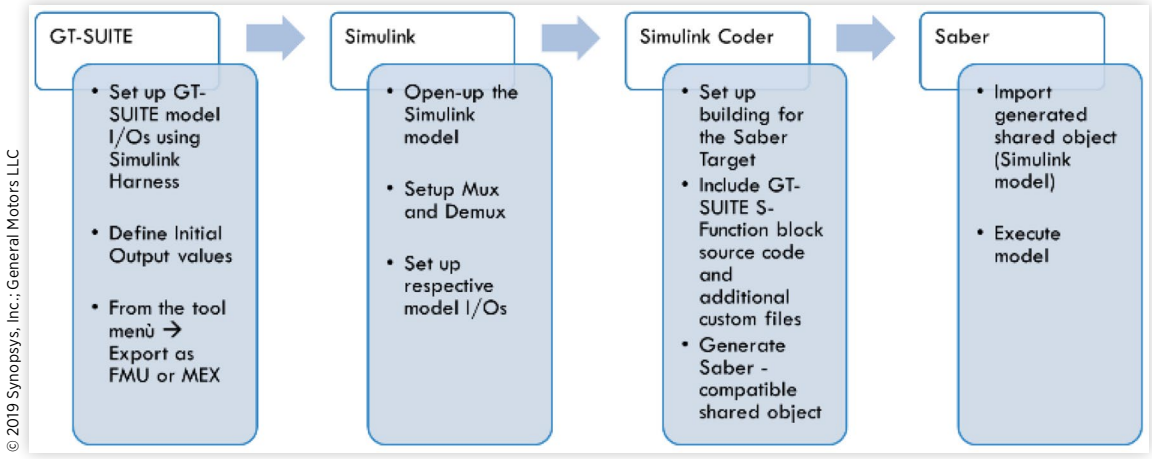


FIGURE B2 Approach a): Embedded approach simulation sequence

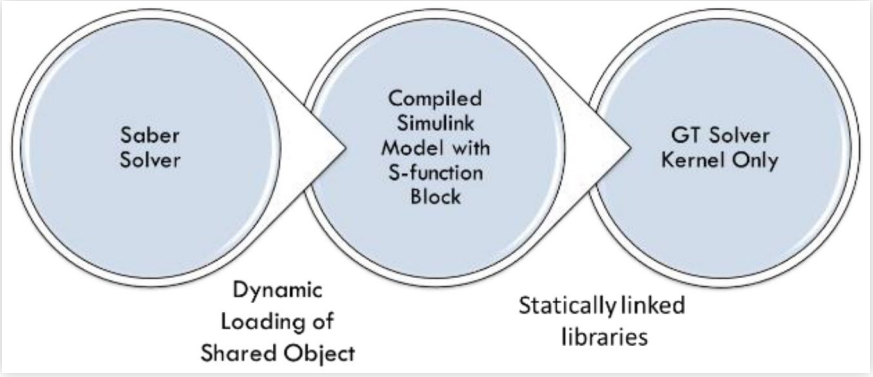


FIGURE B3 Approach b): Co-simulation approach workflow

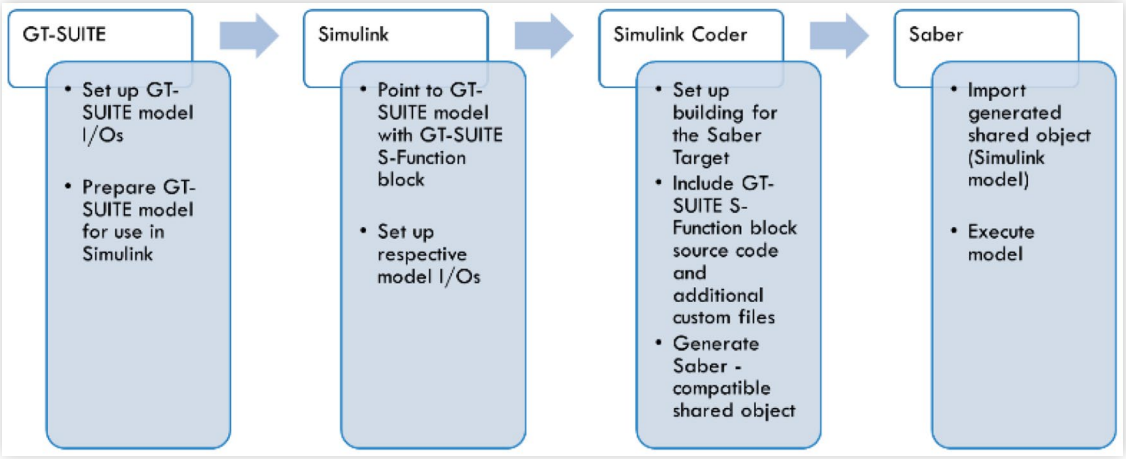
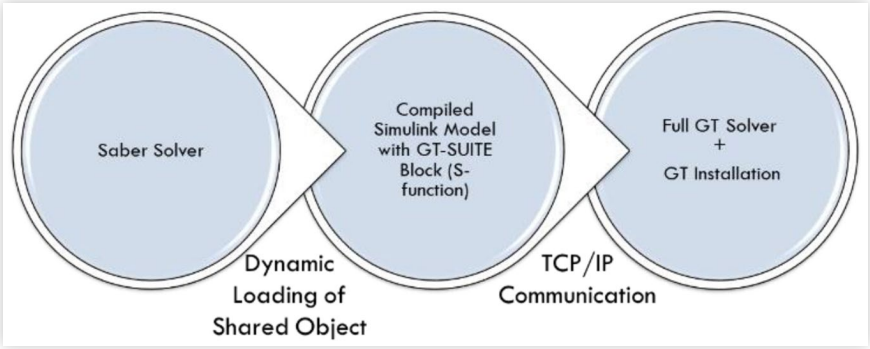


FIGURE B4 Approach b): Co-simulation approach simulation sequence



Appendix C - Co-Simulation Approach Workflow

Workflow to export a GTpower model to Saber:

1. In the GTpower model add a Simulink Harness and define the list of inputs and outputs
2. Define the Initial Output values
3. Execute the model
4. From the GTPower working folder copy the <modelname.dat>
5. Open Simulink and locate the Simulink library browser. Here select the GT-SUITE library and add the symbol called GT-SUITE S-Function into the Simulink schematic
6. Open Simulink Coder (a.k.a Real Time Workshop)
   a. Setup the Solver tab
   b. Setup the Code Gen tab
   c. Build the model (Ctrl+B)

In figure B3 and B4 it's possible to see the graphical view of the workflow to setup and run a GT-Saber model.

ISSN 0148-7191