

Co-Simulation of Multiple Software Packages for Model Based Control Development and Full Vehicle System Evaluation

Trevor Sweafford and Hwan-Sik Yoon
Tennessee Technological Univ

Yanying Wang
General Motors Company

Anthony Will
General Motors

ABSTRACT

Recent advancements in simulation software and computational hardware make it realizable to simulate a full vehicle system comprised of multiple sub-models developed in different modeling languages. The so-called, co-simulation allows one to develop a control strategy and evaluate various aspects of a vehicle system, such as fuel efficiency and vehicle drivability, in a cost-effective manner. In order to study the feasibility of the synchronized parallel processing in co-simulation this paper presents two co-simulation frameworks for a complete vehicle system with multiple heterogeneous subsystem models. In the first approach, subsystem models are co-simulated in a serial configuration, and the same sub-models are co-simulated in a parallel configuration in the second approach. In order to demonstrate their capability, these two co-simulation methods are applied to a full vehicle system with two sub-models developed in different simulation environments: GT-POWER for the engine and MATLAB/Simulink for the driveline and the vehicle dynamics. The simulation results closely match experimental data, although there exist some discrepancies in their relative magnitudes. However, when compared together, the two co-simulation methods produce nearly the same results, while the parallel simulation could reduce the computation time by 25.7%.

CITATION: 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):2012, doi: 10.4271/2012-01-0951.

INTRODUCTION

Recently, computer simulation of automotive systems has received increasing attention in the automotive research community. This interest is motivated by the economic benefits offered by computer simulations, when compared to the conventional prototype development process. While the required cost for the development of prototype vehicles is still high, the ever-increasing computational power allows analysis of complex vehicle systems with a fraction of the prototype development cost. Thus, the so-called virtual prototyping is now becoming an important component in the vehicle development process.

The recent trend in virtual prototyping is to simulate all vehicle components together to obtain a comprehensive assessment of a vehicle. Currently, numerous third party vendors offer high-fidelity software packages for specific

types of systems. Most of these software packages are now equipped with a full suite of simulators covering all aspects of vehicle systems. Due to the development history, however, there still exists preference toward the simulation of specific components in a vehicle system. For example, GT-SUITE from Gamma Technologies is widely used for accurate engine simulation and AMESim from LMS International is regarded to provide high-fidelity transmission and driveline simulations. Although it is not generally possible to directly link system models developed in different software, most of the software packages now allow linkage to MATLAB/Simulink through Simulink's S-function block. Thus, Simulink can be used as a universal platform for the co-simulation of heterogeneous subsystem models developed in different languages.

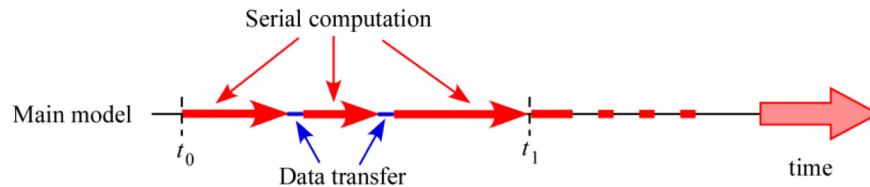


Figure 1. Co-simulation of a complex system with three sub-models

Co-simulation of heterogeneous sub-models has been studied recently for various systems. For example, [Kim et al. \(2008\)](#) reported co-simulation of ADAMS and MATLAB models to evaluate different algorithms for vehicle stability control of a four wheel drive hybrid electric vehicle. [Mancosu and Arosio \(2005\)](#) co-simulated system models developed in ADAMS and MATLAB/Simulink. In their study, a multi-body vehicle model was developed in ADAMS and subsystems were modeled in MATLAB/Simulink including electronic control systems, the braking system and the steering system. More recently, [Datar et al. \(2010\)](#) co-simulated a full vehicle system using a vehicle dynamic model developed in ADAMS and a powertrain model in PSAT. They examined the effect of vehicle dynamics on the fuel consumption emission, and vehicle control strategies, as well as the influence of the powertrain system on the vehicle dynamics.

In order to study the feasibility of the parallelization of a co-simulation process, this paper presents two different co-simulation configurations to simulate a full vehicle system for the FTP driving cycle. For the engine subsystem, a high-fidelity GT-POWER model for a commercial production vehicle is applied and for the remaining parts of the vehicle, a medium fidelity model developed in MATLAB/Simulink is utilized. In the first co-simulation method, the GT-POWER engine model is embedded in the Simulink vehicle model and the combined heterogeneous system is run for the full FTP driving cycle in a serial manner. In the second method, the two sub-models are run concurrently in a synchronized parallel manner to reduce the computation time. Two different co-simulation configurations are addressed in section 1 and the system models used in this study are detailed in section 2. Sections 3 and 4 present results from serial co-simulation and parallel co-simulation, respectively. Finally, a summary and a conclusion are presented in section 5.

CO-SIMULATION OF COMPLEX SYSTEMS

SERIAL CO-SIMULATION

A physical system with multiple subsystems evolves incrementally in time, with each subsystem interacting with others. This time domain evolution occurs simultaneously in all parts of the system by exchanging mass, force or energy between different components. In a co-simulation framework, this type of system can be simulated in a serial manner where

each sub-model is simulated one after another as shown in [Figure 1](#). Note that there exists a time slot for data transfer between adjacent subsystem models. However, this data transfer occurs asynchronously after the completion of each sub-model, and does not require global synchronization. Two distinct simulation parameters related to time can be defined. The simulation time is the time period over which a user desires to simulate a system. On the contrary, the computation time is the actual amount of real time that has elapsed as the simulation progresses on a specific computer. In this serial configuration, the overall computation time depends on the number of sub-models and the time it takes to advance each sub-model by one step size.

PARALLEL CO-SIMULATION

The concept of parallel co-simulation promotes the idea that multiple sub-models can be simulated concurrently in a synchronized manner. The nature of the physical models used for the simulation is such that each sub-model requires data produced by other sub-models in order to advance to the next time step, implying that data exchange must occur synchronously. The frequency of the data exchange is governed by a simulation time step, which is user-defined. During this time step, each sub-model is processed and the results are transmitted to appropriate models. This process is continued until the simulation runs to completion. The actual time required to complete one simulation time step is limited by the sub-model that runs slowest. The overall computation time is calculated by adding all the computation time for the longest-running model within each time step and the time for data exchange.

In parallel simulation, multi-sub-model simulations can be performed by evaluating each sub-model with the inputs from other sub-models calculated at the previous time step. Thus, during each time step, the output of a certain sub-model can be calculated independently of the other sub-models, allowing it to be done in a parallel manner. This algorithm is depicted in [Figure 2](#). For this illustration, it is assumed that a given system model is partitioned into three sub-models that are running on three different CPU cores. Depending on the complexity of each sub-model, the actual computation time may vary for different sub-models. In the figure, it is shown that the computation time for sub-model C is the longest between time, t_0 and t_1 . In this time interval, sub-models A and B finish their calculations earlier than sub-model C. For the synchronized parallel simulation to work, A and B should wait until C finishes its task. When C finishes, all the sub-

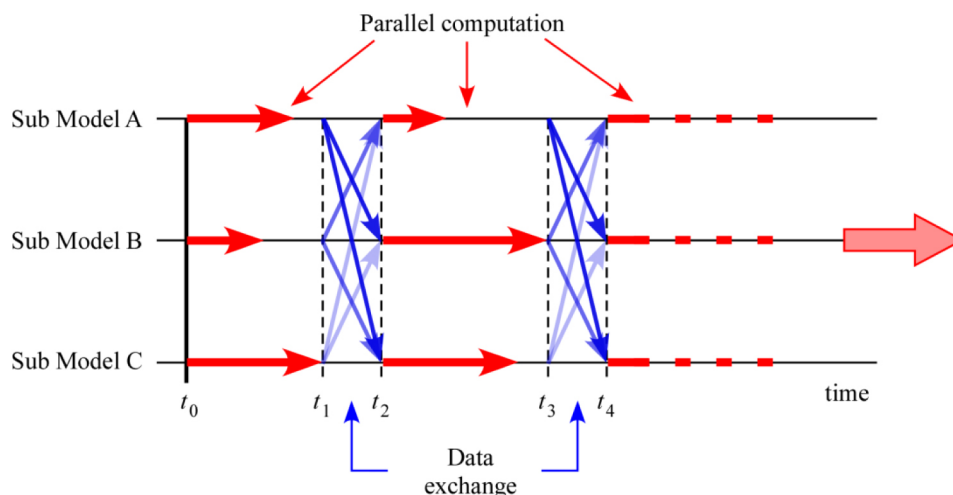


Figure 2. Synchronized parallel computation of three different sub-models

models exchange their computation results, which will serve as initial conditions for the next time step. Then, another set of computations will begin at t_2 . Between t_2 and t_3 , the computation of sub-model B takes the longest, and the other sub-models should wait until B finishes its task. After this, they will exchange the computed data again. This synchronized parallel process will continue until the predefined stop time has been reached. In this configuration, creating a balance in the computational load will result in a faster execution time. This balanced load distribution can be achieved by revising the model partitioning strategy in a way that distributes the computational load evenly across different CPU cores.

VEHICLE MODEL

A functional block diagram of a simulated vehicle system is shown in Figure 3. The system is developed in MATLAB/Simulink with an S-function block connected to an external engine model developed in GT-POWER. The GT-POWER engine model is a four-cylinder model that takes 11 input and produces 10 output variables. The input variables include engine speed, intake/exhaust cam phasing, spark timing, ambient temperature, ambient pressure, and air-fuel ratio. The output variables include air flow rate at the throttle, intake/exhaust manifold pressure, exhaust manifold temperature, and indicated torque. While the GT-POWER engine model is a high-fidelity 1D physical model, the remaining parts of the vehicle are modeled with lower fidelity. The driver logic receives the desired vehicle speed profile as an input and compares it with the actual vehicle speed to produce the accelerator pedal position as a percentage value, and braking force actuation signals. These signals are applied to the vehicle model that consists of four functional blocks: engine, transmission, driveline, and chassis. The transmission block simulates all gear ratios but does not include the gear shift dynamics, resulting in reduced accuracy in transient response. The driveline model converts the transmission input torque

and speed into the output torque and speed based on the applied gear ratio. Finally, basic longitudinal motion of the vehicle is modeled in the chassis component, with tires simplified as rigid bodies, and the output vehicle speed is fed back to the driver logic.

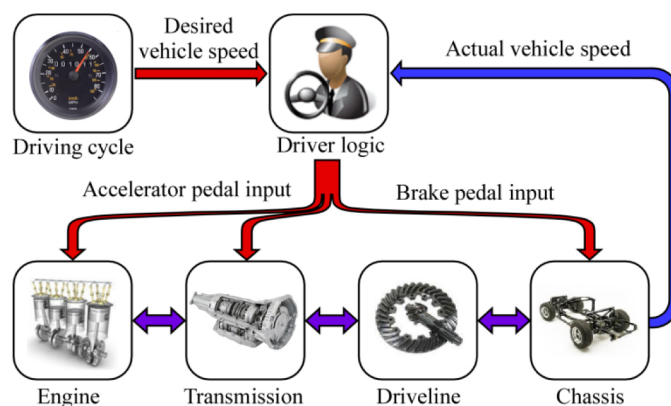


Figure 3. Structure of the vehicle simulation model

Co-simulation of the system requires a dominant simulator that controls the progress of the simulation through a user-defined time step. In the serial co-simulation, the Simulink vehicle model serves this purpose. Since the GT-POWER engine model is nested within the Simulink model as an S-function block, its progression and data exchange are controlled by the Simulink model. This configuration is shown in Figure 4. Since there is no model partitioning and the overall system runs as a single entity, the computational load is not efficiently distributed across multiple CPU cores. Therefore, this type of configuration may require a large computation time, especially when the system is comprised of multiple sub-models that are computationally demanding.

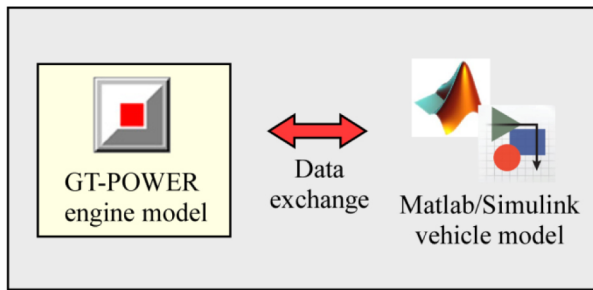


Figure 4. Configuration of serial co-simulation process

To explore the opportunities that might exist to achieve a faster computation time, a commercial software package that allows parallel co-simulation is tested. This program serves as an interface between different models written in various simulation languages including MATLAB/Simulink and GT-POWER. The co-simulation method used by the program requires a partitioning strategy that divides a large complex system into smaller sub-models based on the model functions or computational load. The configuration of the parallel co-simulation is depicted in Figure 5, where two sub-models exchange simulation data through a master control block in a synchronized manner. The master control block manages simulation progress as well as the synchronized data exchange among the attached sub-model blocks. As explained in the previous section, the parallel configuration

allows faster execution of co-simulation by assigning multiple sub-models to different CPU cores.

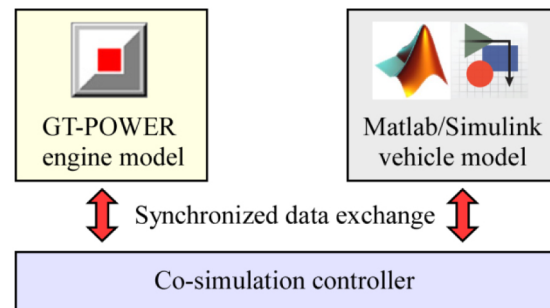


Figure 5. Configuration of parallel co-simulation process

SERIAL CO-SIMULATION RESULTS

Simulation of the full vehicle model was conducted for the full FTP cycle (1400 seconds) using the serial co-simulation method. The goal of the simulation is to examine the feasibility of virtual experiments to capture important performance variables without conducting any actual experiment with a physical prototype vehicle. The simulation took about 22 hours and 50 minutes (1370 minutes) using a workstation with a Quad Core Intel Xeon W3565 processor. A screenshot of the serial co-simulation is shown in Figure 6,

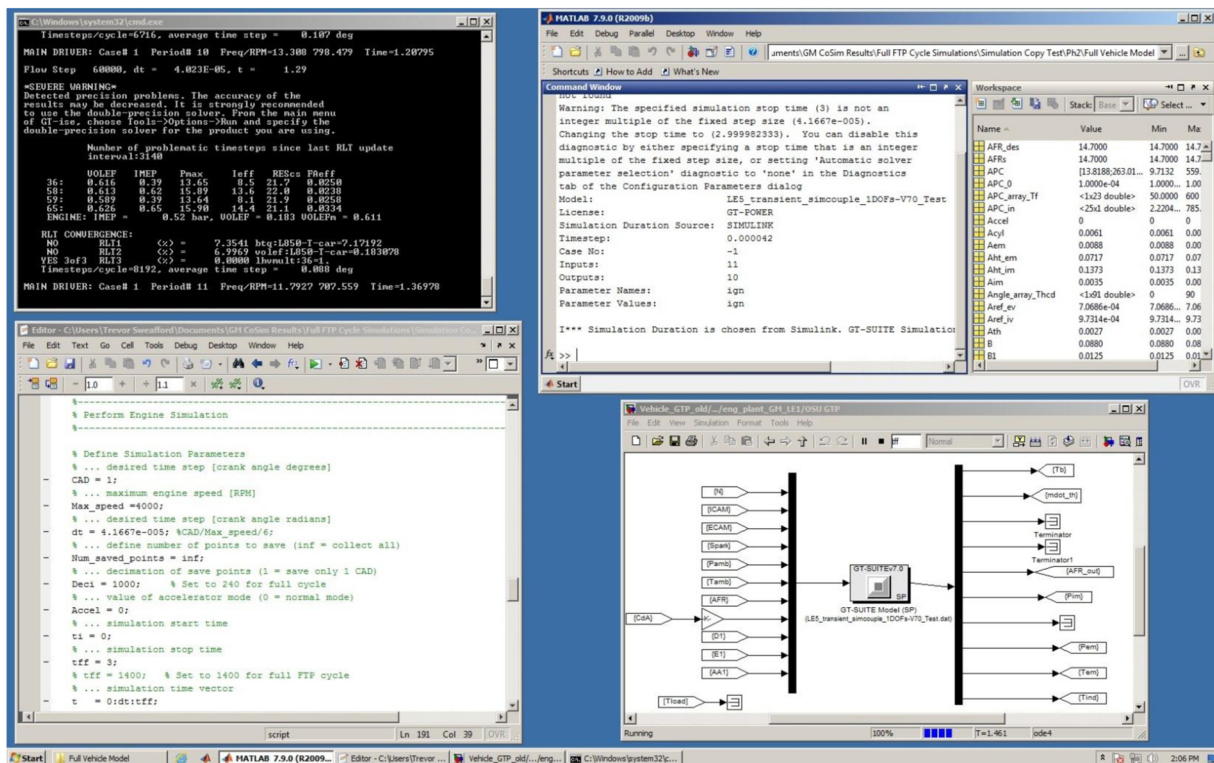


Figure 6. Screenshot of serial co-simulation of the vehicle model