Managing Heterogeneous Simulations Using Architecture-Driven Design

Nico Vansina[1] Bruno Loyer[2] Yosuke Ogata[3]

[1]Siemens PLM Software, Belgium, nico.vansina@siemens.com
[2]Siemens PLM Software, France, bruno.loyer@siemens.com
[3]Siemens PLM Software, Japan, yosuke.ogata@siemens.com

Abstract

This paper presents an architecture-driven approach to manage heterogeneous simulations. A European automotive OEM has requested Siemens PLM Software to use its tools and process knowledge to demonstrate the value and need for architecture-driven simulation. Siemens PLM Software proposed a project to demonstrate Simcenter System Synthesis[1] as a neutral framework for managing heterogeneous simulations. This includes three major capabilities:

- Integration of different subsystem models in the form of Simcenter Amesim[2] "supercomponents" and Functional Mock-up Units (FMUs) exported from Dymola[3].
- Plug-and-play configuration of subsystems regardless of their native software.
- Performant execution of heterogeneous simulation architectures with the numerical challenges of segregated strongly coupled systems

The focus of the project is on the process of model integration using Functional Mock-up Units (FMUs). An electrical vehicle case-study was selected to illustrate this process.

Keywords: Simcenter System Synthesis, Simcenter Amesim, FMI, Architecture-driven simulation, heterogeneous simulation

1 Introduction

System simulation is a proven method for anticipating the balancing of multiple performance attributes of a product. However, in the automotive industry today, a large diversity of vehicle architectures and technologies exists. This results in a huge number of variants for all subsystems. It becomes increasingly difficult to manage and analyze all possible configurations. An automotive example is depicted in Figure 1.

Additionally, subsystem models are implemented in different authoring tools. A framework is needed to integrate these subsystem models and assemble them into an executable system simulation (see Figure 2). This paper will focus on this topic of model integration using the FMI standard (Blochwitz *et al*, 2011).

Simcenter System Synthesis provides an architecture-driven approach to tackle this challenge.

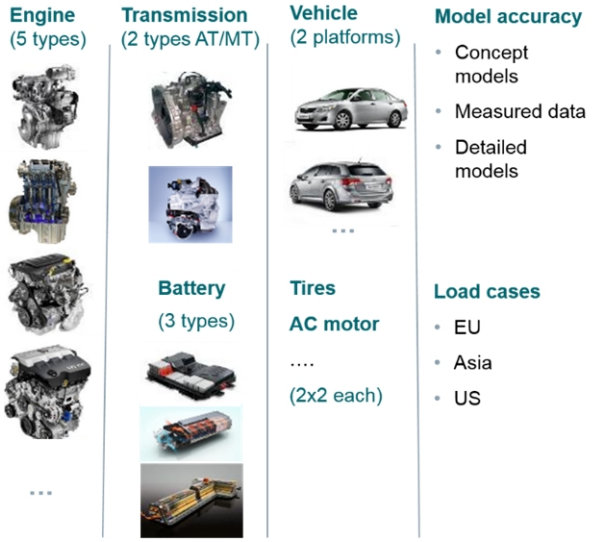

Figure 1. The challenge of dealing with many variants

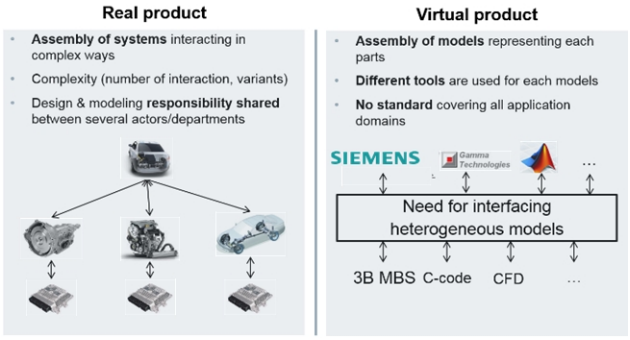

Figure 2. The challenge of integrating models from different authoring tools

[1] Simcenter System Synthesis is a configuration management, system integration and system architecture management tool developed by Siemens PLM Software

[2] Simcenter Amesim is a commercial simulation software for the modeling and analysis of multi-domain systems, developed by Siemens PLM Software

[3] Dymola is a commercial modeling and simulation environment based on the open Modelica modeling language, developed by Dassault Systèmes.

DOI 10.3384/ecp18148204

2 Use case description

2.1 Base line behavioral model

An electric vehicle use case is selected to illustrate the process of architecture-driven simulation. The Simcenter Amesim model depicted in Figure 3 serves as a baseline model for the project implementation. This model consists of the following subsystems:

- New European Driving cycle (NEDC) mission profile
- Driver
- Vehicle control unit (VCU)
- Electric battery (static model)
- Electric motor (static model)
- Transmission (fixed ratio)
- Vehicle (1D lateral model)

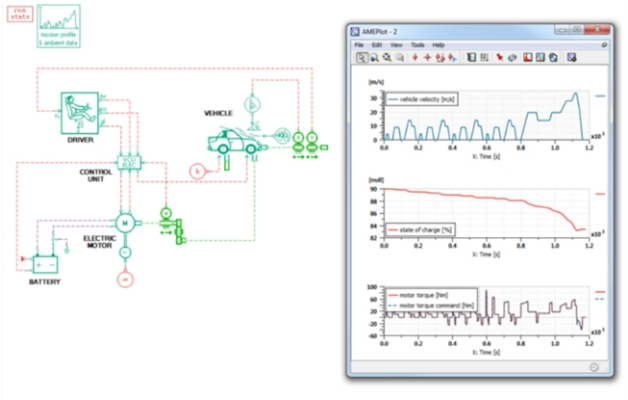

Figure 3. Baseline model for the electric vehicle use case in Simcenter Amesim

2.2 Dymola subsystem models

The transmission and electric motor subsystems are implemented as simple static behavioral models in Simcenter Amesim. These subsystem models will be replaced by alternative ones created by the European Automotive OEM in Dymola (see Figure 4 and Figure 5).

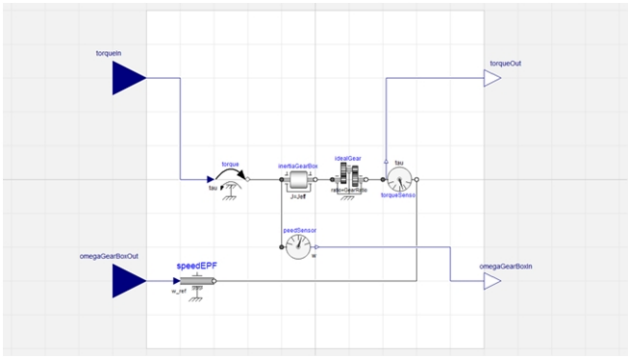

Figure 4. Transmission subsystem implemented in Dymola

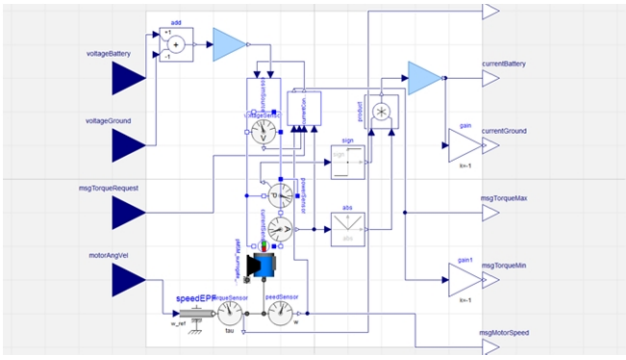

Figure 5. Electric motor subsystem implemented in Dymola

2.3 Integration of Dymola subsystem models through FMI standard

In order to integrate the Dymola subsystem models, they need to be exported as FMUs. Two different types of FMUs are tested to evaluate performance and accuracy:

- Slave co-simulation FMU compliant with the FMI 2.0 standard (Blochwitz *et al*, 2012)
- Model exchange FMU compliant with the FMI 1.0 standard (Blochwitz *et al*, 2011)

Simcenter System Synthesis is used as a framework for integrating the heterogeneous simulation models (Simcenter Amesim and FMUs originating from Dymola). This integration is done in 4 phases:

1. Integration of Simcenter Amesim baseline model
2. Replacing the transmission subsystem by the exported FMUs
3. Replacing the electric motor subsystem by the exported FMUs
4. Replacing both transmission and electric motor subsystems by the exported FMUs

3 Architecture-driven simulation framework

The workflow in Simcenter System Synthesis is broken down into 4 big steps:

1. Architecture and template definition
2. Model instrumentation
3. Model assembly creation
4. Simulation execution

Each of these steps will be discussed in a separate following subsection. The Simcenter baseline model is used to realize the architecture.

3.1 Architecture and template definition

In a first step, a tool-neutral architecture is defined. This architecture describes the layout of the system from a simulation standpoint. The electric vehicle architecture consists of the following subsystems: scenario definition, vehicle control unit (VCU), electric battery, electric motor, gearbox and vehicle. Afterwards, the connections between the subsystems are defined resulting in the architecture definition as depicted in Figure 6.

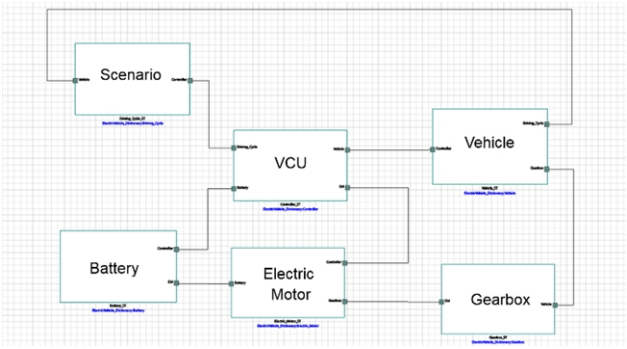

Figure 6. Architecture definition

In a second step, a template is created for each of the subsystems. The template is an interface contract specifying input and output ports between subsystems, parameters and variables. In Figure 7 the electric motor simulation template is depicted. The interface contract between the electric motor and gearbox is specified as rotary speed and torque. Similar interfaces are defined for all subsystems.

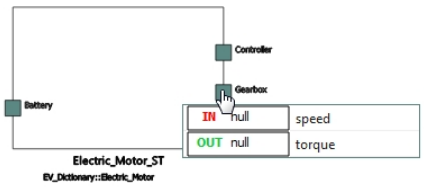

Figure 7. Electric motor simulation template

Defining architecture and templates will increase control and collaboration. The template acts as a target for the subsystem designer ensuring integration in the overall system. The architecture is the framework for integrating models developed in different departments and created in different tools.

3.2 Model instrumentation

In a next step, instrumented models are created. They are a combination of a behavioral model and a simulation template. The instrumentation process consists in mapping ports, parameters and variables between template and behavioral model. Figure 8 shows the port mapping of gearbox simulation template and the behavioral model implemented in Simcenter Amesim. Parameters and variables are mapped to the exposures of the template in a similar way.

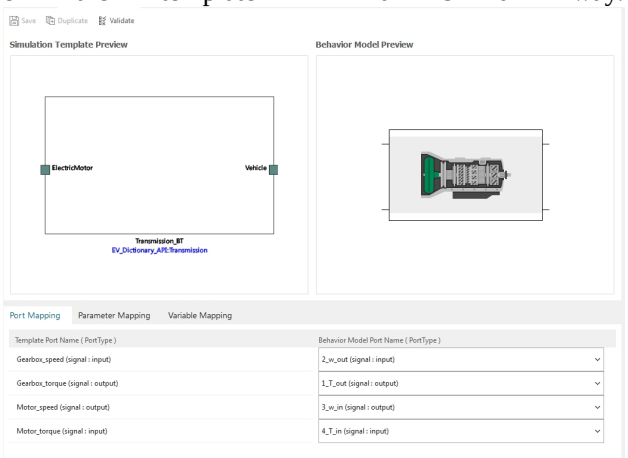

Figure 8. Instrumentation of gearbox simulation template with Simcenter Amesim behavioral model

Instrumentation increases the modularity and reusability of models. They don't need to be redeveloped, but rather can be reused in future projects.

DOI
10.3384/ecp18148204

3.3 Model assembly creation

Afterwards, a model assembly can be created. For each template an instrumented model is selected. This connection is “plug-and-play” thanks to the interface contract. The simulation template filters out the compliant instrumented models that can be selected. Figure 9 shows that the gearbox simulation template can be realized by one of the two variant FMU instrumented models (Slave co-simulation FMU compliant with the FMI 2.0 standard and model exchange FMU compliant with the FMI 1.0 standard).

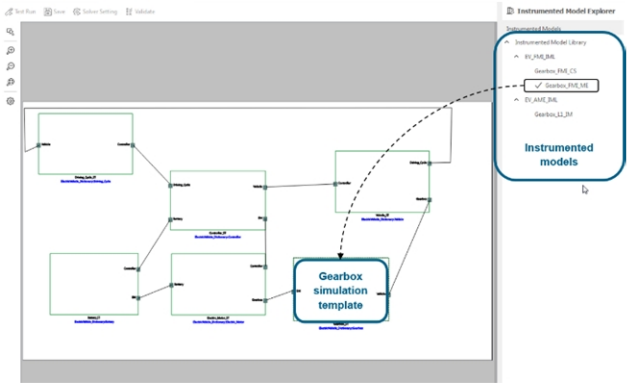

Figure 9. Model assembly creation. The gearbox simulation template filters out the compliant instrumented models that can be selected.

Subsystem models become plug-and-play and are directly integrated. There is no need any more for complex integrations like co-simulation setups, importing and exporting results.

3.4 Simulation execution

The study is launched from Simcenter System Synthesis (Figure 10). In the background the models are composed and the heterogeneous simulations are started in Simcenter Amesim. When the simulation is complete, the results can be plotted by selecting the variables of interest. This process could be extended to manage and execute all possible scenarios and load cases.

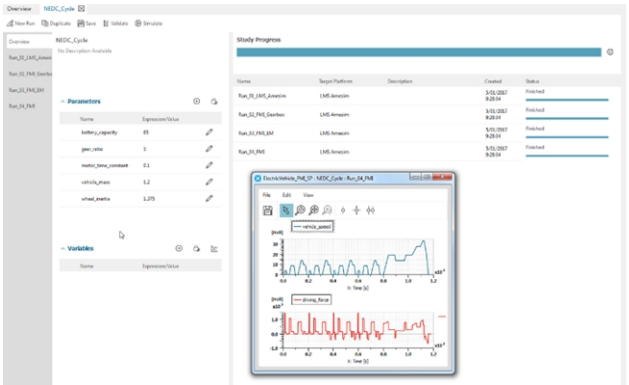

Figure 10. Heterogeneous simulation execution

4 Heterogeneous model assembly with gearbox FMU

In the last section, the framework for heterogeneous simulation was established using the Simcenter Amesim baseline model (Figure 3). In this section the gearbox subsystem will be replaced by a Dymola model exported as FMU (Figure 4). All other subsystems are implemented as Simcenter Amesim behavioral models. The end state is visualized graphically in Figure 11.

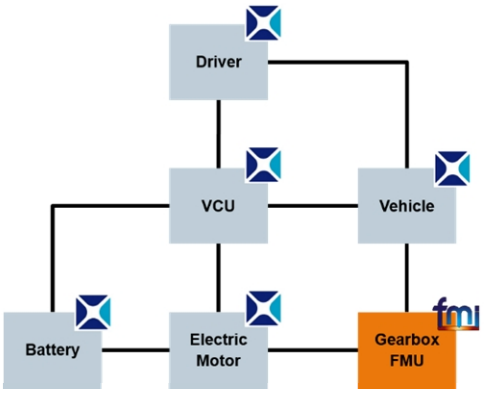

Figure 11. Heterogeneous model assembly after integration of gearbox FMU

4.1 Model instrumentation

The gearbox simulation template is instrumented with different FMUs (Figure 12). One for co-simulation (CS) and one for model exchange (ME).

Two model assemblies are added for the FMI gearbox configurations (see Figure 9). The study consists of 2 additional simulation runs to analyze the impact of replacing the gearbox subsystem with a functional mock-up unit

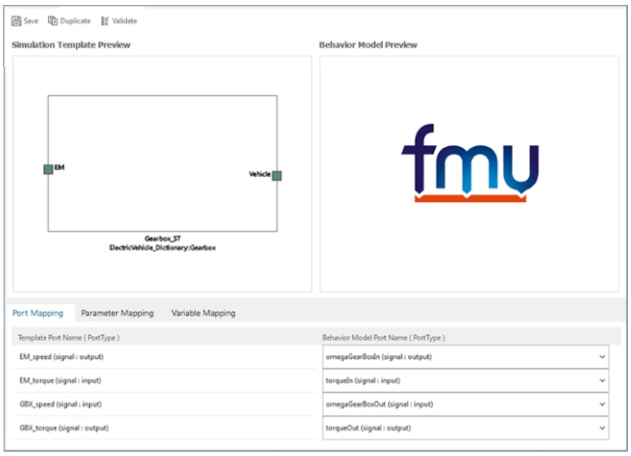

Figure 12. Instrumentation of gearbox simulation template and FMU behavioral model

4.2 Simulation results

4.2.1 Accuracy

Overall there is a good correlation between the results for both model exchange and co-simulation FMUs as can be seen in Figure 13 for the gearbox output torque.

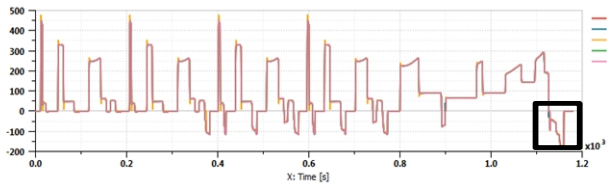

Figure 13. Gearbox output torque for full NEDC Cycle

A Small difference in gearbox torque output can be noticed when looking into a smaller region (Figure 14). The deviation between Simcenter Amesim model and FMUs can be explained by the fact that the model content is slightly different. The Simcenter Amesim gearbox subsystem doesn't include an inertia where both FMUs have an inertia of 0.1 kg.m².

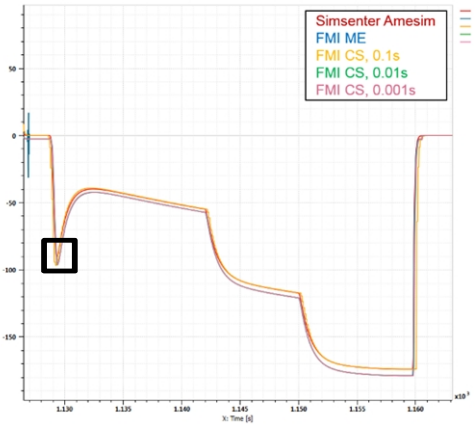

Figure 14. Gearbox output torque (zoomed in to the black region in the right left corner of in Figure 13)

The small deviation between FMUs is due to the co-simulation delay. Co-simulation discretizes the system and introduces a delay. The inputs are held constant throughout a co-simulation step. This results in discrete output which is clearly visible in the case of a co-simulation step of 100ms in Figure 15.

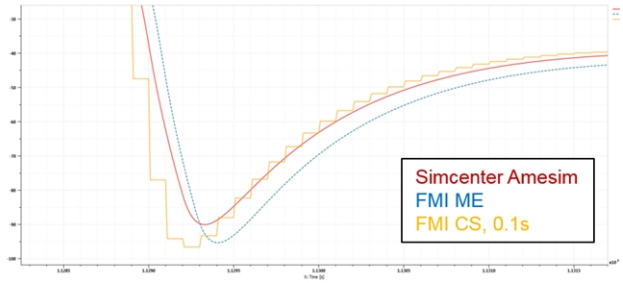

Figure 15. Co-simulation discretizes the system and introduces a delay (detail of the black area on the left in Figure 14)

4.2.2 Performance

The CPU time is summarized in Figure 16 for different gearbox implementations. The model exchange FMU runs as fast as the native Simcenter Amesim model. Co-simulation performance is dependent on the co-simulation step: The simulation is run for 3 different time steps: 0.1, 0.01 and 0.001s.

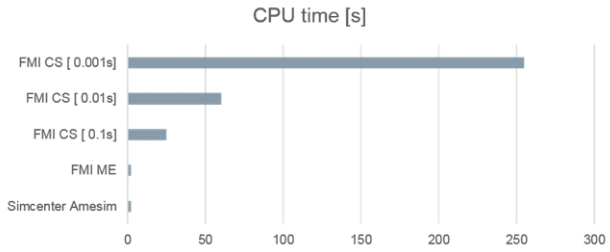

Figure 16. CPU time Performance

4.2.3 Discussion

Both model exchange and co-simulation options are possible for integrating the gearbox subsystem as FMU. In the case of co-simulation, the co-simulation time step has an important effect on accuracy and performance. A co-simulation time step of 10ms is chosen as a good compromise. In this case, an NEDC cycle simulation scenario of 20 minutes is run in 60 seconds on a standard laptop, resulting in a speedup factor of 20 compared to the wall clock time. Note that in cases where co-simulation is mandatory, typically when no unique suitable solver can be found for model exchange, then some advanced co-simulation techniques might be used for dealing with strongly coupled systems, as shown in (Viel, 2014). Version 2.0 of the FMI standard paves the way towards a wider use of these promising techniques but today, classic zero-order hold co-simulation is still well-established and remains the most frequent use case. This situation is mainly due to the limited number of tools complying with the required FMI optional capabilities (e.g. provide directional derivatives).

DOI
10.3384/ecp18148204

5 Heterogeneous model assembly with electric motor FMU

In this section the electric motor subsystem will be replaced by a Dymola model exported as an FMU (Figure 5). All other subsystems are still implemented as Simcenter Amesim behavioral models. The end state is visualized graphically in Figure 17.

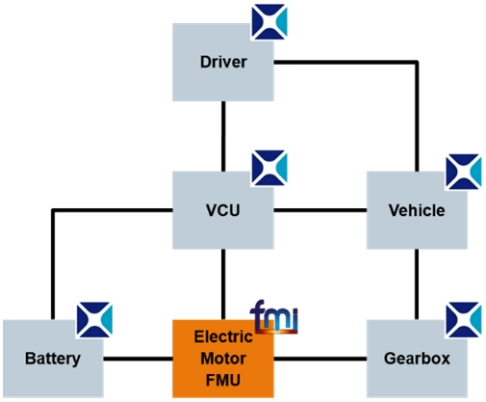

Figure 17. Heterogeneous model assembly after integration of electric motor FMU

5.1 Model instrumentation

The electric motor simulation template is instrumented with different FMUs (Figure 18) similar to the gearbox in the previous section. Two model assemblies are added for both electric motor configurations (CS and ME FMUs).

Figure 18. Instrumentation of electric motor simulation template and FMU behavioral model

5.2 Simulation results

5.2.1 Accuracy

Figures Figure 19 and Figure 20 show a detail of the electric motor variables within a region of transient conditions. We can conclude that the model exchange FMU performs well in general. For co-simulation the accuracy of the results is dependent on the co-simulation step. A time step of 100ms is close to the stability limits resulting in oscillating behavior. Overshoots go up to 20% in transient regions. A time step of 10ms doesn't show this behavior and closely matches the Simcenter Amesim native model and model exchange FMU.

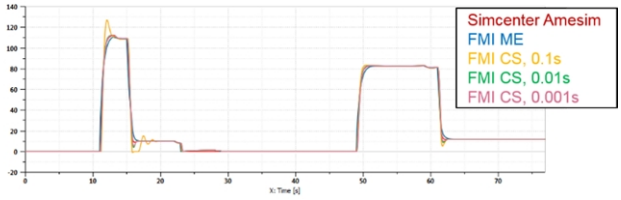

Figure 19. Detail of electric motor torque in a transient region

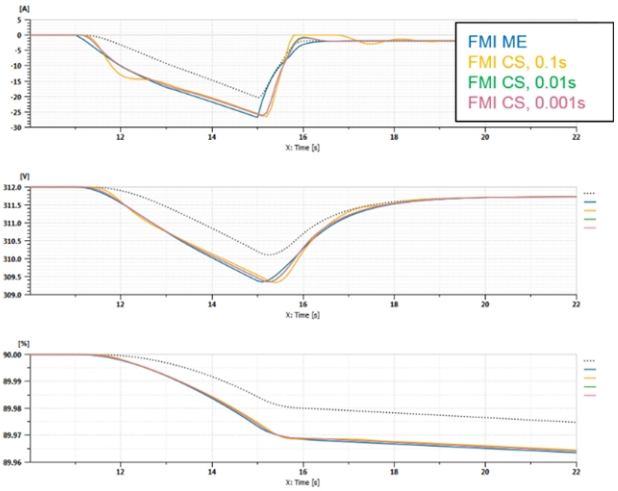

Figure 20. Detail of electric behavior in a transient region: a. current [A]. b. Voltage [V] and c. State of charge [%]

5.2.2 Performance

The CPU time is summarized in Figure 21 for different electric motor implementations. Since a small co-simulation time step is needed to get accurate results and the separate subsystems are low frequency models, we can conclude that the high-frequency dynamics originates from the coupling itself. The electric motor and gearbox are strongly coupled systems.

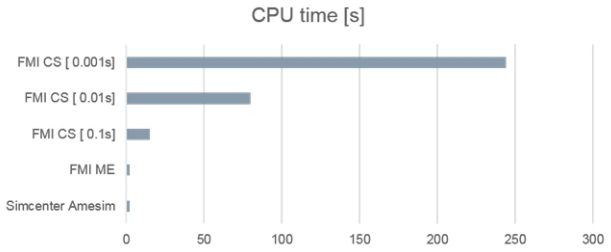

Figure 21. CPU time Performance

The CPU time increases strongly with the decrease of the co-simulation step and can be explained by the increasing number of function evaluations (Figure 22).

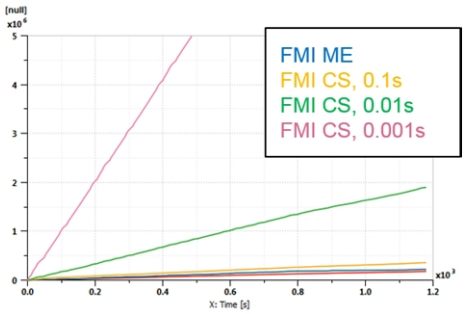

Figure 22. Number of function evaluations over time

Within one macrostep, several microsteps are taken by each individual solver as depicted in Figure 23. If the macrostep (co-simulation step) gets very small, this necessarily increases the number of microsteps. This results in a high number of function evaluations or solver calls and eventually a high CPU time.

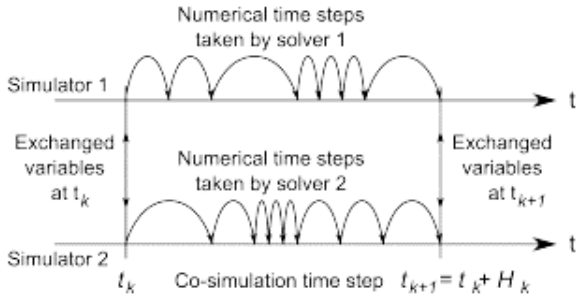

Figure 23. Within one macrostep (co-simulation step), several microsteps are taken by each individual solver.

5.2.3 Discussion

Both model exchange and co-simulation options are possible for integrating the electric motor subsystem as FMU. Model exchange can be very performant for strongly coupled systems whereas co-simulation is interesting for decoupling different dynamics. When using co-simulation in this example, a time step of 10ms is a needed to balance accuracy and CPU time. In this case, an NEDC cycle simulation scenario of 20 minutes is run in 80 seconds on a standard laptop, resulting in a speedup factor of 15.

6 Heterogeneous model assembly with gearbox and electric motor FMU

Finally, both the electric motor and gearbox subsystems will be replaced by a Dymola model exported as an FMU. The end state is visualized graphically in Figure 24.

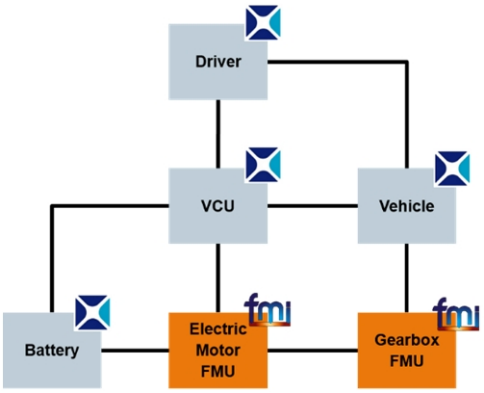

Figure 24. Heterogeneous model assembly after integration of both the gearbox and electric motor FMUs

6.1 Simulation results

6.1.1 Accuracy

Figure 25 compares the electric motor torque for the combination of ME and CS FMUs.

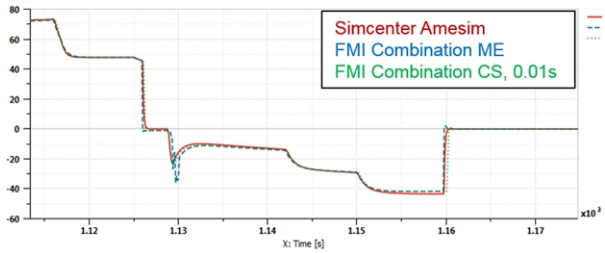

Figure 25. Detail of electric motor torque in a transient region

As shown in (Ogata *et al*, 2014), solver settings play an important role when integrating multiple FMUs for model exchange. The torque output is depicted for different solver tolerances in Figure 26. To remove spikes in the results the mixed tolerance needs to be tightened from 1e-05 to 1e-07.

Co-simulation tends to decouple the different subsystems because it reduces the coupling to the minimal set of relevant variables. Boundary conditions are updated only at discrete predefined rendez-vous points.

DOI
10.3384/ecp18148204

Model exchange implies the solver settings (including tolerance) are applied to the full system. Thus the most computationally demanding subsystem imposes these settings. With co-simulation each subsystem uses its own solver, when going for model exchange this modularity is lost. To ensure convergence for the full model a restrictive tolerance is required.

In practice, co-simulation requires no solver tuning, assuming each tool manages their native subsystems correctly but co-simulation time steps need to be adjusted to get the best compromise between accuracy and CPU efficiency. With model exchange you need to adapt the solver to manage heterogeneous models coming from different tools, which can be challenging when dealing with numerous subsystems and/or when some subsystems require very specific "exotic" solving methods. Generally speaking, exporting strongly solver-dependent models as FMUs for model exchange should be avoided or done with proper documentation about required solving methods.

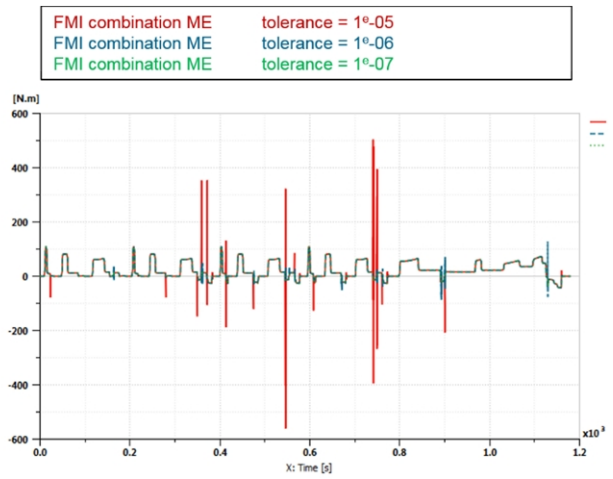

Figure 26. Effect of solver settings (tolerance) on the simulation results

6.1.2 Performance

The CPU time is summarized in Figure 27 for different combined setups. For both co-simulation FMUs a co-simulation time step of 10ms is chosen. CPU time increases by 40% when adding the gearbox as co-simulation FMU next to the electric motor.

Solver settings have an important impact when integrating FMUs for model exchange. To ensure convergence for the full model a restrictive tolerance is required leading to CPU times similar or even higher than the co-simulation case.

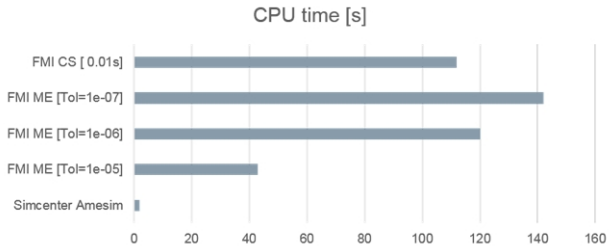

Figure 27. CPU time Performance

6.1.3 Discussion

Integration of both gearbox and electric motor subsystems were presented in this section using model exchange and co-simulation. Model exchange implies solver settings are applied to the full system. In order to get accurate results some expertise is therefore needed to tune the solver to ensure convergence for the full model. In the present study the solver tolerance had to be optimized to reduce inaccurate overshoots in the transient regions. A tolerance of 1e-07 was selected, leading to a CPU time slightly higher than the co-simulation case. In this use case where several FMUs are combined, co-simulation is the best choice for:

- Ease of use (no model solver tuning expertise needed)
- Comparable accuracy for lower CPU time.

When choosing a co-simulation time step of 10ms for both systems, the NEDC cycle simulation scenario of 20 minutes is run in just under 2 minutes on a standard laptop, resulting in a speedup factor of 11.

7 Conclusion

In this paper, Simcenter System Synthesis was presented as a framework for managing heterogeneous simulations. The integration of different subsystem models was performed in the form of Simcenter Amesim "supercomponents" and co-simulation or model exchange FMUs exported from Dymola. This framework offers configuration management of subsystems regardless of their native software as depicted in Figure 28. Such a neutrality is critical for OEMs who have to leverage all the capabilities of individual tools within a unique heterogeneous simulation and architecture-management platform.

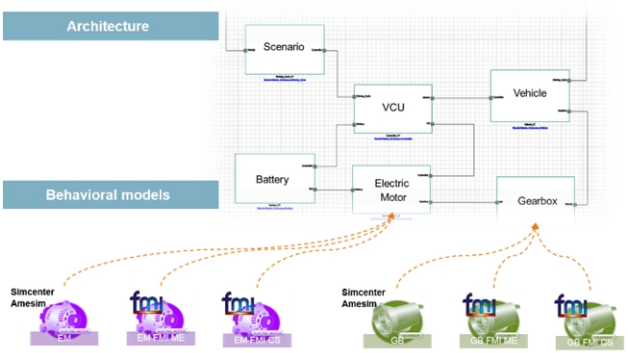

Figure 28. Simcenter System Synthesis as a framework for managing heterogeneous simulations

In the present case, the execution of heterogeneous simulation architectures with the numerical challenges of segregated strongly coupled systems was done in a performant way and the factors influencing this performance were documented. All these simulations were initiated from Simcenter System Synthesis and in the background composed and run in Simcenter Amesim. This provides a transparent way of running and comparing the different configurations regardless of their native software implementation.

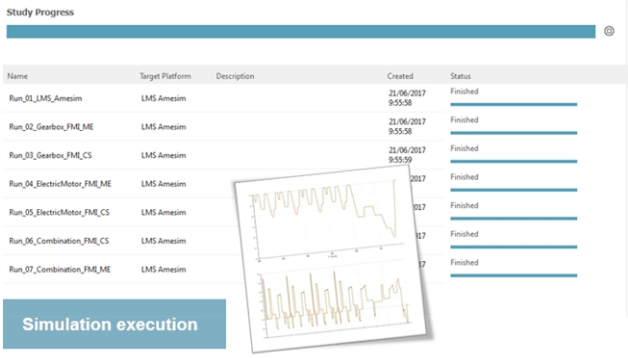

Figure 29. Simulation execution from Simcenter System Synthesis

References

Ogata Y., Loyer B., Viel A.: New trends and methods for the co-simulation of strongly coupled systems using the Functional Mock-up Interface 2.0, JSAE Annual Congress, Yokohama, May 23, 2014.

Viel A.: Implementing stabilized co-simulation of strongly coupled systems using the Functional Mock-up Interface 2.0, 10th International Modelica Conference, Lund, March 2014.

Blochwitz T., Otter M., Arnold M., Bausch C., Clauß C., Elmqvist H., Junghanns A., Mauss J., Neumerkel D., Olsson H., Peetz J.-V., Wolf S.: The Functional Mock-up Interface for Tool independent Exchange of Simulation Models, 8th International Modelica Conference, Dresden, Germany, 2011.

Blochwitz T., Otter M., Åkesson J., Arnold M., Clauß C., Elmqvist H., Friedrich M., Junghanns A., Mauss J., Neumerkel D., Olsson H., Viel A.: Functional Mock-up Interface 2.0: The Standard for Tool independent Exchange of Simulation Models, 9th International Modelica Conference, Munich, Germany, 2012.

DOI
10.3384/ecp18148204