FMI implementation in LMS Virtual.Lab Motion and application to a vehicle dynamics case

Hunor Erdélyi, William Prescott, Stijn Donders, Jan Anthonis
LMS international
Interleuvenlaan 68 - 3001 Leuven - Belgium
hunor.erdelyi@lmsintl.com

Abstract

The aim of this paper is to present the implementation of the Modelisar Functional Mock-up Interface (FMI) in LMS Virtual.Lab Motion. This functionality enables co-simulation between multi-disciplinary subsystem models for a range of industrial applications. The validity of the methodology and industrial applicability of the implementation is demonstrated on an application case taken from automotive industry, with an Opposite Wheel Travel scenario using a half vehicle model in LMS Virtual.Lab Motion and an Air-spring FMU based on Modelica code.

Keywords: Functional Mock-up Interface (FMI); Modelica; Co-simulation; LMS Virtual.Lab Motion

1 Introduction

In complex systems such as in automotive and aerospace many different types of subsystems (e.g. mechanical, hydraulic or electric subsystems) interact with each other [1]. The simulation of such complex multidisciplinary systems is a new challenge in modern computer aided engineering.

A widely used technique to link together different multidisciplinary subsystems in a common simulation framework is what scientific literature refers to as *Co-Simulation*. In co-simulation, the overall system is split into different subsystems, which are treated by different optimized simulation tools, coupled by input and output variables, thus creating a coupling loop [2, 3].

The "Functional Mock-up Interface" (FMI) [4], developed within the framework of the ITEA2 Modelisar project [5], provides a standardized way for linking together different subsystems modeled in different simulation software. An instance of a model compiled for being linked with a 3rd party simulation environment is called a "Functional Mock-up Unit" (FMU).

Typically an FMU consists of the following main elements compressed into a single archive:

a) C-header files to interact with the equations of a model or to perform co-simulations with other simulators (model interface) and

b) XML schema files to inquire information about model and interface variables (model description file)

c) executable files

Two distinct standards have been defined within the framework of FMI: ***FMI for Model Exchange*** and ***FMI for Co-Simulation.*** The FMI for **Model Exchange** was developed to allow a modeling tool to generate C code or binary files from a model that can be integrated into another simulation environment [4]. The FMI for **Co-Simulation** defines an interface standard for the communication between a master and the individual simulation tools called slaves in a co-simulation environment. The data exchange is restricted to discrete communication points in time and the subsystems are solved independently between these communication points [4, 6].

FMI compatibility was implemented in LMS Virtual.Lab Motion [7], a multi-purpose simulation software, specially designed to simulate realistic motion and loads of mechanical system. LMS Virtual.Lab Motion can be used as a simulation platform into which one or several FMUs can be linked in order to perform simulations for analyzing complex multidisciplinary systems.

2 FMI Interface in LMS Virtual.Lab Motion

A schematic representation of linking an FMU into a simulation with LMS Virtual.Lab Motion is presented in Figure 1. To be able to establish the link between LMS Virtual.Lab Motion and an FMU, ***inputs*** and ***outputs*** have to be defined, which will represent the coupling data for the co-simulation.

The coupling data is exchanged at the level of ***Control Nodes***. A ***Control Input*** represents the signal which is transmitted from the mechanical model in LMS Virtual.Lab Motion to the FMU. Typically, Control Inputs are displacement, velocity or acceleration data. A ***Control Output*** is a signal received from an FMU that is applied to the mechanical model in LMS Virtual.Lab Motion (e.g. force or torque). ***Control Nodes*** are the nodes or connection points to which the above mentioned Control Inputs and Outputs are applied.

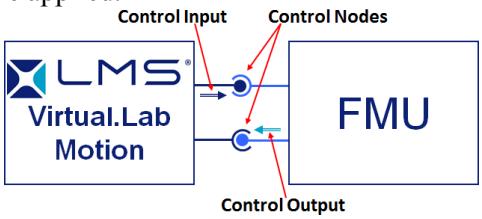

Figure 1: Schematic representation of the FMI interface in LMS Virtual.Lab Motion

For the two distinct standards, FMI for Model Exchange and FMI for Co-Simulation, the different approaches are described as follows.

In case of linking to an FMU for Model Exchange the state equations of both the FMU and LMS Virtual.Lab Motion are solved by the Motion solver.

The LMS Virtual.Lab Motion Solver uses a set of Differential-Algebraic equations (DAE) of motion in Netwon-Euler format [7].

$$M\dot{v} + \Phi_q^T \lambda = Q_a(q, v) \quad (1)$$

$$\Phi(q) = 0 \quad (2)$$

Here, q is the vector of generalized position coordinates, v denotes the vector of generalized coordinate velocities, M is the mass matrix, Q_a is the vector of applied forces, $\Phi(q)$ denotes the vector joint constraint equations and λ stands for the vector of LaGrange multipliers. A maximal set of coordinates are considered first and then the extra degrees of freedom are removed by applying a set of joint constraint equations.

When linking an FMU for Model Exchange to LMS Virtual.Lab Motion a set of control forces is applied on the mechanism bodies representing the contribution of the FMU. In turn sensors feed position, velocity and acceleration data back to the FMU. Usually, the FMU forces are the product of state equations. This means that the Motion solver must integrate a set of differential equations from the FMU.

Representing the FMU state equations by g and the state variable by χ, the coupled equations of motion become:

$$M\dot{v} + \Phi_q^T \lambda = Q_a(q, v, \chi) \quad (3)$$

$$\Phi(q) = 0 \quad (4)$$

$$g(q, v, \chi, \dot{\chi}) = 0 \quad (5)$$

In case of linking to an FMU for Co-Simulation, each simulation package runs its own solver, which is in turn synchronized with the other solver. Each solver is running and communicating with the other solver at discrete intervals in time. The same equations (3-5) are solved in the co-simulation mode as in the case of model exchange, but separately. In this situation the LMS Virtual.Lab Motion solver is the master. The Motion solver solves its own set of state equations from the current time (t^i) to the time at the next communication interval (t^{i+1}). Equation (5) now becomes equation (6) where the FMU variable inputs (q, v) are still at the last sample time.

$$g(q^i, v^i, \chi, \dot{\chi}) = 0 \quad (6)$$

Once the LMS Virtual.Lab Motion solver has finished integrating to the next communication interval the FMU solver is called and told to integrate to the current time. The FMU solver now uses the LMS Virtual.Lab Motion inputs at the last communication interval to move forward to the next communication interval.

$$M\dot{v} + \Phi_q^T \lambda = Q_a(q, v, \chi^i) \quad (7)$$

$$\Phi(q) = 0 \quad (8)$$

For both cases described above, a fixed communication interval has been used.

In the following paragraphs, the implementation of the FMI standard into LMS Virtual.Lab Motion will be demonstrated with a simple air-spring FMU.

3 Application case description and results

For demonstrating the implementation of the FMI interface and industrial applicability, an application case is presented from automotive industry, with an Opposite Wheel Travel scenario using a half vehicle model in LMS Virtual.Lab Motion and an Air-spring FMU based on Modelica code.

3.1 Development of a Modelica FMU of an air-spring

An air-spring can be approximated as a volume of air, enclosed either in a cylinder fitted with a piston or in a flexible bellows, as shown in Figure 2. The air is compressed to a predetermined pressure under the static load of the vehicle. Subsequent motion of the piston either increases or decreases the pressure and consequently increases or decreases the force acting on the piston.

For simplicity, the air-spring is modeled with an isothermal process, considering a closed system and

DOI 10.3384/ecp12076759

ideal gas. The chamber of the gas is considered as rigid, thus neglecting the elasticity of the bellow.

The diameter of the piston is variable as highlighted in Figure 2.

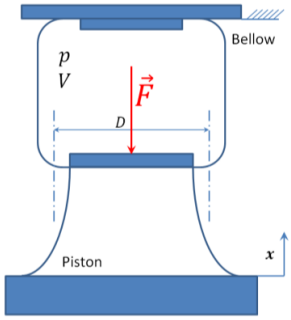

Figure 2: Schematic representation of an air-spring (p is the pressure and V is the volume of the gas, D represents the piston diameter and F the piston force, x is the piston displacement)

For an ideal gas at constant temperature, the Boyle-Mariotte law is valid (9):

$$pV = nRT = constant \tag{9}$$

Where, p denotes the pressure of the system, V denotes the volume of the gas, n is the number of moles of gas present, R is the ideal gas constant and T denotes the temperature of the system.

Considering the air-spring modeled as an isothermal process, the pressure p of the system will be variable as a function of the volume V. Furthermore, the volume V depends on the displacement and diameter of the piston of the air-spring.

The diameter of the piston is defined as a function of its displacement x (10):

$$D = D_0\left(\frac{\tanh(k_1 x)}{k_2} + 1\right) \tag{10}$$

For the present case the piston diameter varies following the curve shown in Figure 3. Parameters k_1 and k_2 are used for tuning the shape of the curve.

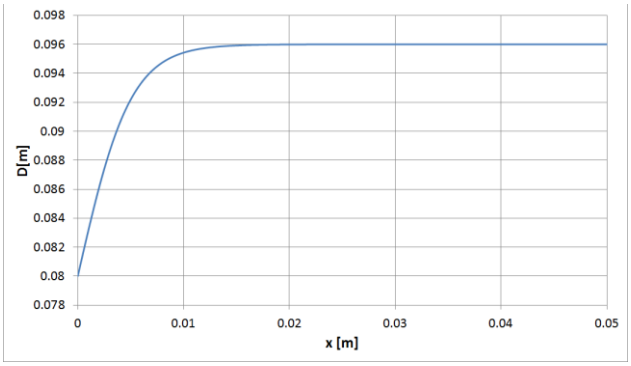

Figure 3: Piston diameter as a function of piston displacement

The volume of the system is defined as a function of the initial volume $V0$, the piston area A, and displacement x (11):

$$V = V_0 - \frac{Ax}{2} \tag{11}$$

Where the piston area A is defined as follows (12):

$$A = \pi\left(\frac{D}{2}\right)^2 \tag{12}$$

The pressure acting on the piston can be defined based on the ideal gas law (13):

$$p = \frac{nRT}{V} \tag{13}$$

Where n is the number of moles of gas present in the chamber of the air-spring and can be determined as follows (14):

$$n = \frac{p_0 V_0}{RT} \tag{14}$$

In the above equation (14) p_0 denotes the initial pressure of the air-spring system. For a displacement of 0.05 m the pressure evolution of the air-spring is presented in Figure 4 below.

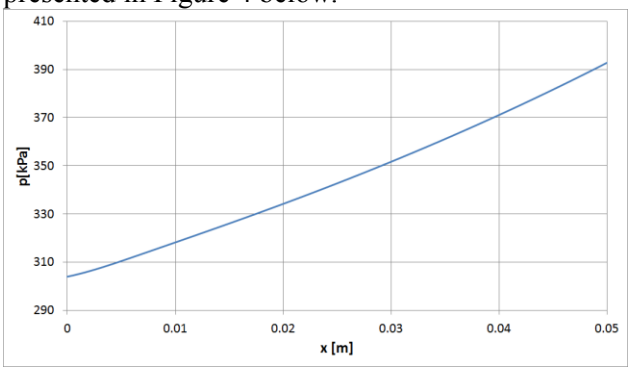

Figure 4: Pressure of the system as a function of piston displacement

The force acting on the piston is defined as a function of the piston area and the pressure in the air-spring system (15):

$$F = pA \tag{15}$$

Considering a displacement of 0.05 m, the evolution of the force acting on the piston is presented in Figure 5.

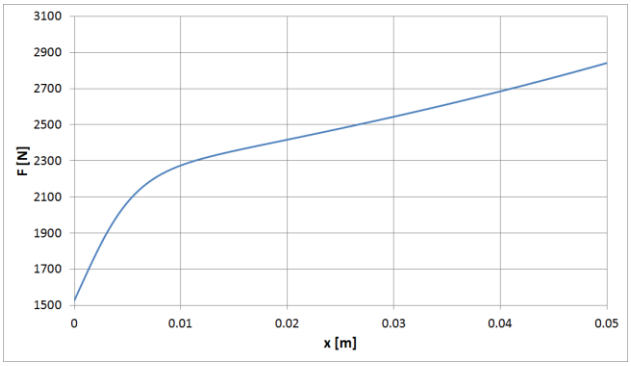

Figure 5: Piston force as a function of piston displacement

Based on the thermodynamic relations described above, the air-spring system was translated into Modelica code.

```
class Airspring
  parameter Real R = 8.3144621;
  parameter Real V0 = 0.0008;
  parameter Real T = 293.15;
  parameter Real p0 = 303975.0;
  parameter Real D0 = 0.08;
  parameter Real k1 = 200.0;
  parameter Real k2 = 5.0;
  Real V;
  Real n;
  Real A;
  Real p;
  Real D;
  input Real x;
  output Real F;
equation
  D = (D0 * tanh(k1 * x)) / k2 + D0;
  A = 3.14 * (D / 2.0) ^ 2.0;
  p = (n * (R * T)) / V;
  n = (p0 * V0) / (T * R);
  V = V0 + ((-A) * x) / 2.0;
  F = p * A;
end Airspring;
```

The pre-defined parameters of the Modelica code of the air-spring are the following:

R = 8.3144621[J/mol K]	ideal gas constant
V0 = 0.0008[m^3]	initial chamber volume
T = 293.15[K]	gas temperature
p0 = 303975[Pa]	initial gas pressure
D0 = 0.08[m]	initial piston diameter
k1 = 200	parameter 1
k2 = 5	parameter 2

The input to the Modelica air-spring model is the displacement of the piston *x* and the output of the model is the force *F* acting on the piston.

An FMU for Model Exchange of the Modelica air-spring was generated with the specified IN and OUT ports, using OpenModelica 1.8.0 based on the FMI standard V1.0. This FMU was linked into a dynamic simulation with LMS Virtual.Lab Motion.

3.2 LMS Virtual.Lab Motion vehicle dynamics simulation with a Modelica air-spring FMU

In LMS Virtual.Lab Motion a front suspension of a vehicle was modeled (as shown in Figure 6). An Opposite Wheel travel scenario was implemented, which is one of the typical scenarios considered in vehicle suspension design for analyzing relevant suspension parameters and forces in the connecting elements.

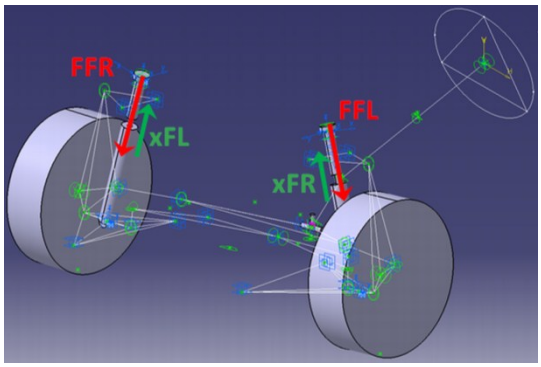

Figure 6: Vehicle front suspension in LMS Virtual.Lab Motion (air-spring FMU inputs are highlighted in green and outputs in red)

In an opposite wheel-travel analysis the left and right wheels are moved vertically on an equal but opposite path to simulate body roll. The left and right wheels move 180° out of phase with respect to each other along a specified bounce and rebound travel. For the present case, the wheel travel distance of 0.05m was considered with a cycle time of 1 s.

Two instances of the Modelica Air-spring FMU for Model Exchange were linked into the LMS Virtual.Lab Motion suspension model for the left and right side. The air-spring FMUs were linked to the upper and lower part of the damper units on the left and right side of the suspension.

Corresponding to the Modelica air-spring model the input to the air-spring FMU was the relative displacement of the lower damper part with respect to the upper part. In Figure 6, highlighted with green, *xFL* and *xFR* represent the relative displacement of the Front Left and Front Right dampers respectively.

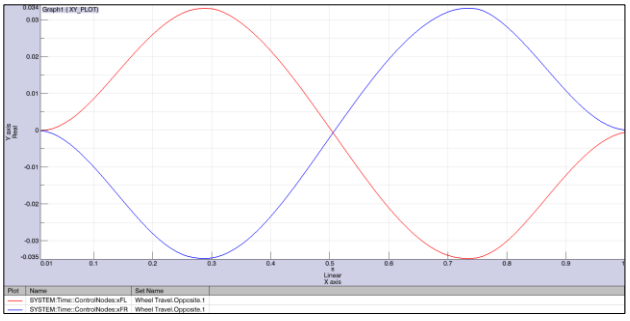

Figure 7: Air-spring FMU input signals (*xFL* in red and *xFR* in blue)

The evolutions of the FMU input signals for the left and right air-springs are presented in Figure 7.

The output of the FMU air-spring was the force on the piston of the air-spring, applied between the upper and lower damper part. Highlighted in red in Figure 6, for the left and right air-springs are the FMU output forces denoted with FFL and FFR respectively.

DOI 10.3384/ecp12076759

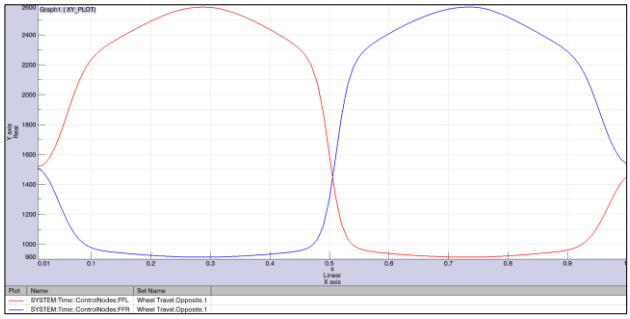

Figure 8: Air-spring FMU output signals (*FFL* in red and *FFR* in blue)

Figure 8 presents the evolutions of the FMU output signals. The nonlinear behavior of the air-spring forces is clearly visible.

3.3 Validation of the presented air-spring FMU with LMS Imagine.Lab AMESim

To validate the FMI implementation in LMS Virtual.Lab Motion, the results obtained with the FMU for Model Exchange have been compared to the results obtained with LMS Imagine.Lab AMESim.

LMS Imagine.Lab AMESim is a 1D simulation suite to model and analyze multi-domain, intelligent systems and predict their multi-disciplinary performance [8].

For the purpose of validation, the air-spring model has been replicated in LMS Imagine.Lab AMESim using the same equations (10–15). The AMESim model of the air-spring has been coupled with the LMS Virtual.Lab Motion model using a Model exchange approach, but instead of using the FMI standard, an internally developed interface was adopted.

Consequently, the set of control forces from the LMS Imagine.Lab AMESim air-spring have been applied on the LMS Virtual.Lab Motion mechanism, which have been solved together by the Virtual.Lab Motion solver. To be able to correctly compare results, the same communication time interval of *0.001s* has been used for both cases.

Figure 9 presents the comparison of the different air-spring forces obtained with the FMU for Model Exchange with the LMS Imagine.Lab AMESim model. In this figure the front left air-spring force (*FFL*) is presented in red and the front right air-spring force (*FFR*) in blue. The FMU forces are depicted with continuous lines while the LMS Imagine.Lab AMESim forces are presented with dashed lines.

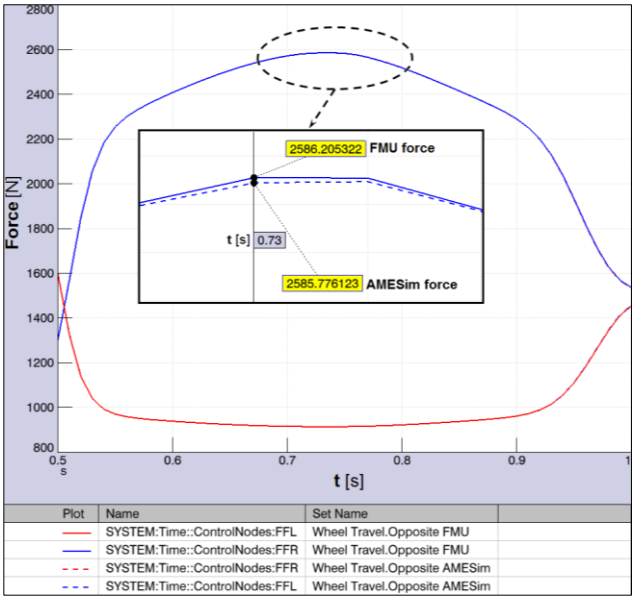

Figure 9: Comparison of Air-spring forces: *FFL* in red and *FFR* in blue; FMU signal in continuous line, AMESim signal in dashed line

As it can be noticed in Figure 9 the FMU forces and the AMESim forces follow very closely each other. In the central region of the figure, a close-up is presented at *t=0.73 s.*

The difference between the signals is 0.429 N, which expressed in percentage, is approximately 0.016% and as such can be considered negligible.

4 Conclusions

The Modelisar FMI standard provides a vendor-neutral interface that allows the exchange of simulation models between different tools and platforms and enables their use in multidisciplinary simulations.

This paper presents the implementation of the Modelisar Functional Mock-up Interface (FMI) in LMS Virtual.Lab Motion. This functionality is demonstrated with an Opposite Wheel Travel scenario using a half vehicle model in LMS Virtual.Lab Motion and an Air-spring FMU for Model Exchange compiled from Modelica code.

Linking together different FMUs and an LMS Virtual.Lab Motion model in a co-simulation environment brings several benefits. However, both Co-simulation and Model Exchange type of simulation have their benefits and drawbacks.

In a Model Exchange type of simulation, in addition to the set of multibody equations of motion, a set of control forces from the FMU are applied on the mechanism, which are solved together by the Virtual.Lab Motion solver. Usually, the FMU forces

are the product of state equations. In a Model Exchange type of simulation the main benefits are: good numerical stability and use of the full capability of the solver (variable step sizes, iterative methods…). The drawback is that this approach may be inefficient and time consuming if large differences in stiffness exist between the subsystems and the systems are loosely coupled.

In case of Co-simulation, the coupling data is exchanged between the Virtual.Lab Motion solver and the FMU at each communication interval, consequently, the co-simulation approach is less stable. In the case of Co-simulation, the main benefits are: problem-specific solvers can be used for integrating different subsystems and hence it may be more time efficient for loosely coupled systems (solvers may use different integration step sizes). On the downside, this approach is less stable as the Model Exchange type. The main reason for this instability is the approximation of the coupling variables between two consecutive communication time steps. However, by choosing the communication step size carefully a stable simulation can be achieved.

As a result it is suggested to use the model exchange approach for tightly coupled systems, while the co-simulation approach may be more efficient in loosely coupled problems.

Acknowledgements

We gratefully acknowledge IWT Vlaanderen and ITEA2 for their support of the R&D projects IWT-080067 (ITEA2-07006) “MODELISAR” (From System Modeling to S/W running on the Vehicle), and we furthermore acknowledge IWT Vlaanderen for supporting the R&D project IWT-090408 "CHASING". In addition, we gratefully acknowledge the European Commission for their support of the Marie Curie IAPP project 285808 “INTERACTIVE” (Innovative Concept Modelling Techniques for Multi-Attribute Optimization of Active Vehicles, http://www.fp7interactive.eu/).

References

[1] Anthonis J., Gubitosa M., Donders S., Gallo M., Mas P., Van der Auweraer H. Multi-Disciplinary Optimization of an Active Suspension System in the Vehicle Concept Design Stage, 14th Belgian-French-German Conference on Optimization, Leuven, Belgium, September, 2009.

[2] Busch M. and Schweizer B. Numerical Stability and Accuracy of Different Co-Simulation Techniques: Analytical Investigations Based on a 2-DOF Test Model, 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland, May 25–27, 2010.

[3] Schierz, T., Arnold, M. MODELISAR: Innovative numerische Methoden bei der Kopplung von multidisziplinären Simulationsprogrammen, in A. Brenke (ed.): Tagungsband ASIM-Konferenz STS/GMMS 2011, Krefeld, 24.02.-25.02.11, Shaker Aachen, 2011.

[4] Functional Mock-up Interface: http://www.functional-mockup-interface.org/

[5] IWT Vlaanderen, IWT-080067 (ITEA 07006) “MODELISAR” (From System Modeling to S/W running on the Vehicle), http://www.modelisar.com

[6] Bastian J., Clauß C., Wolf S., Schneider P. Master for Co-Simulation Using FMI, Proceedings 8th Modelica Conference, Dresden, Germany, March 20-22, 2011.

[7] LMS International, LMS Virtual.Lab Rev. 11, http://www.lmsintl.com/virtuallab, May 2011.

[8] LMS International, LMS Imagine.Lab AMESim Rev. 11 http://www.lmsintl.com/LMS-Imagine-Lab-AMESim, May 2011.

 DOI 10.3384/ecp12076759