



THE FIRST JAPANESE MODELICA CONFERENCE

May 23–24, 2016, Tokyo

The First Japanese Modelica Conferences is sponsored by:



The first Japanese Modelica Conference is organized by Modelon and Modelica Association

Proceedings of the 1st Japanese Modelica Conference
Tokyo, Japan, May 23-24, 2016

Editors:

Dr. Yutaka Hirano and Dr. Johan Andreasson

Published by:

Modelica Association and Linköping University Electronic Press

ISBN: 978-91-7685-749-6

Series: Linköping Electronic Conference Proceedings, No 124

ISSN: 1650-3686

eISSN: 1650-3740

DOI: <http://dx.doi.org/10.3384/ecp16124>

Organized by:

Modelon K.K.

Embassy of Sweden Compound

1-10-3-901 Roppongi, Minato-ku, Tokyo 106-0032

Japan

in co-operation with:

Modelica Association

c/o PELAB, Linköpings Univ.

SE-581 83 Linköping

Sweden

Conference location:

Embassy of Sweden Compound

1-10-3-901 Roppongi, Minato-ku, Tokyo 106-0032

Japan

Copyright © Modelica Association, 2016

The 1st Japanese Modelica Conference, which takes place in Tokyo, is the first of hopefully many more to come. With this effort, we hope to create an arena in Japan and Asia for sharing knowledge and learning about the latest scientific and industrial progress related to Modelica and FMI (Functional Mockup Interface). Ultimately we also hope to see enough momentum to motivate an International Modelica Conference organized in the region. We are surprised but very glad to see the overwhelming interest to both join and contribute to the conference. Instead of originally planned 8 paper presentations and one key-note during one day, we have had to extend the scope and are now proud to present a conference with:

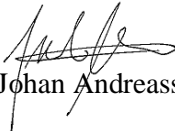

- 2 Keynote speeches
- 18 paper presentations
- A fully booked exhibition area featuring 8 exhibitors
- Concurrent interpretation between Japanese and English
- Great venue location in the heart of Tokyo at the Swedish Embassy
- A conference mingle dinner at the Ambassador's residence

According to Modelica Association standards, all papers are peer-reviewed and will be freely available for download.

We want to acknowledge the support we received from the conference board and program committee. Special thanks to our colleagues at Modelon K.K. for taking care of all the practical matters, and to the Swedish Embassy in Tokyo for hosting the event. Support from the conference sponsors is gratefully acknowledged. Last but not least, thanks to all authors, keynote speakers, and presenters for their contributions to this conference.

We wish all participants an enjoyable and inspiring conference!

Tokyo and Susono, April 30,

 
Johan Andreasson and Yutaka Hirano



Keynotes



Yasuhiro Harada

Mazda

What should Model Based Development aim for?

Vehicle systems have become vast and complex to attain a high level of functionality. It is expected to evolve at an even more accelerated rate. Model Based Development (MBD) has become essential to accomplish such a high level of manufacturing process in a short term. I will show what has been targeted and realized in MBD to date, by giving some examples of past developments, and will look into an ideal manufacturing process.



Scott A. Bortoff

Mitsubishi Electric Research Laboratory

Using Modelica Effectively in Industrial Research and Development

In this talk we will highlight some uses of Modelica in the context of industrial research and development. Beyond time-domain simulation, these uses include control system design and realization, computation of open-loop optimal control that is useful to establish fundamental limitations of performance, system inversion which is useful to compute unmeasured system inputs in real-time, state estimation and data assimilation.

We also want to point out some of the challenges in using large-scale models specifically in the context of product development. These include model calibration, which is often a chicken-and-egg problem, model reduction, incorporation of tabular data, and limited scalability of existing solvers. These highlight some interesting research opportunities for the academic community and are key enablers to even more effective use of Modelica in industry.

We conclude with some advice for new users and some personal experiences with introducing the technology into use at large companies.

Program Committee

General Chair

Dr. Johan Andreasson, Modelon K.K., Japan

Program Chair

Dr. Yutaka Hirano, Toyota Motor Corporation, Japan

Program Board

Dr. Yutaka Hirano, Toyota Motor Corporation, Japan

Prof. Peter Fritzson, Linköping University, Sweden

Prof. Martin Otter, DLR, Germany

Dr. Hilding Elmqvist, Mogram, Sweden

Dr. Michael Tiller, Xogeny Inc., USA

Program Committee

Dr. Johan Andreasson, Modelon K.K., Japan

Christian Bertsch, Robert Bosch GmbH, Stuttgart, Germany

Torsten Blochwitz, ITI GmbH, Dresden, Germany

Peter Bunus, ESI-Group, Sweden

Dr. Hilding Elmqvist, Mogram, Sweden

Prof. Peter Fritzson, Linköping University, Sweden

Dr. Rui Gao, Modelon K.K., Japan

Prof. Anton Haumer, Technical consultant, OTH Regensburg, Regensburg, Germany

Dr. Yutaka Hirano, Toyota Motor Corporation, Japan

Dr. Lee Johanson, ANSYS Inc., USA

Jochen Köhler, ZF AG, Friedrichshafen, Germany

Prof. Hidekazu Nishimura, Keio University, Japan

Prof. Shigeru Oho, Nippon Institute of Technology, Japan

Prof. Koichi Ohtomi, University of Tokyo, Japan

Prof. Martin Otter, DLR, Germany

Prof. Stefan-Alexander Schneider, Hochschule Kempten, Kempten, Germany

Dr. Martin Sjölund, Linköping University, Sweden

Dr. Michael Tiller, Xogeny Inc., USA

Dr. Hubertus Tummescheit, Modelon Inc., West Hartford, USA

Conference Organization Team:

Dr. Johan Andreasson, Modelon K.K., Japan

Dr. Yutaka Hirano, Toyota Motor Corporation, Japan

Tomohide Hirono, NewtonWorks Corporation, Japan

Masanori Kobayashi, IPG Automotive K.K., Japan

Eiji Nakamoto, ANSYS Japan K.K., Japan

Hideyuki Okabe, Dassault Systèmes K.K., Japan

Hiroshi Watanabe, CYBERNET SYSTEMS Co. Ltd., Japan

Noriko Yudahira, IPG Automotive K.K., Japan

Contents

Session 1: Automotive Applications

Dynamics Modelling of the Arc Spring for Powertrain NVH Prediction	9
Research of Model Matching Control of Torque Vectoring Differential Gear System	15
Simulation of Complete Systems at ZF using Modelica Standards	24

Session 2: Model Based Development

Virtual Vehicle Kinematics and Compliance Test Rig	29
Modelica-Association-Project “System Structure and Parameterization” – Early Insights	35
ADAS Virtual Prototyping using Modelica and Unity Co-simulation via OpenMETA	43

Session 3: Mechanical Applications

Active Elbow Joint Model	50
Modeling and simulation for leg-wheel mobile robots using Modelica	55
System-Level Design Trade Studies by Multi Objective Decision Analysis (MODA) utilizing Modelica	61

Session 4: Real Time Simulation, HILS

FMI for Co-Simulation of Embedded Control Software	70
Deployment of high-fidelity vehicle models for accurate real-time simulation	78
Validation of a Battery Management System based on AUTOSAR via FMI on a HiL platform	87

Session 5: Simulation Technologies

Chattering-Free Simulation of Hybrid Dynamical Systems with the Function Mock-Up Interface 2.0	95
Acceleration of FMU Co-Simulation On Multi-core Architectures	106
Rankine Cycles, Modeling and Control	113

Session 6: Thermal System Applications

Thermal Deformation Analysis Using Modelica	121
Validated Modelica Building Package for Energy Performance Simulation with Educational Purposes	129
Advances of Zero Flow Simulation of Air Conditioning Systems using Modelica	139

Dynamics Modeling of the Arc Spring for Powertrain NVH Prediction

Yoshihiro Yamakaji¹

¹Exedy Corporation, Japan, y-yamakaji@exedy.com

Abstract

The key value of Model Based Development is to realize capability of quick performance design and simulation at the early phase of development. In this paper, we modeled an arc spring type torsional damper, which has an impact on the torsional vibration characteristics of powertrain in a vehicle. To predict non-linearity of the arc springs, we took a discrete modeling approach using MODELICA and compared its simulation results with the physical test results. We also developed a user-friendly interface with FMIE (Modelon FMI Add-in for Excel) so that a non-expert of physical modeling can run performance design easily and precisely on their own.

Keywords: Powertrain, Arc Spring, Modelica, FMI, Nonlinear Vibration

1 Introduction

These days, as a car gets more complex than ever, Automotive Original Equipment Manufacturer (OEM, hereafter) are asking Parts Suppliers to deliver performance proposals at the very early phase of development. To do that, it is imperative for Parts Suppliers such as EXEDY to realize performance design with accurate and quick performance prediction.

Recently, Model Based Development (MBD, hereafter) has widely been spread out in Automotive Industry. MBD is an approach to model and simulate systems behaviors taking multiple physical domains into account even before starting the detailed design. To meet the OEM requirements described above, we applied MBD to the development of torsional damper products. The problem here is how to model the arc spring type torsional damper with a high non-linear characteristic. Therefore, we focus on modeling and validation of the arc spring component in a torsional damper.

In EXEDY, we chose some MBD tools capable of handling physical modeling, such as Dymola, for Prediction and Validation phases in V-process shown in Fig. 1. To accelerate its deployment, it is important to build simulation models for better performance and accuracy, and to establish workflow to utilize MBD tools efficiently.

The paper comprises following chapters. First, we explain the basic structure of the torsional damper. Second, we focus on the arc spring and illustrate its physical models. Third, we show Modelica implementation and the comparison with the physical tests. Then we present our interface program on top of Microsoft EXCEL by using FMIE which enables a non-expert of physical modeling to work on the performance design of arc springs based on Modelica.

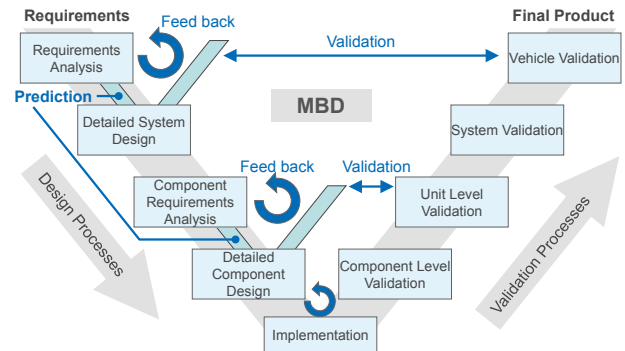


Figure 1. Model-Based-Development Process

2 Description of Target Systems

2.1 Functions of Launch Devices

Launch devices of a car must provide following four functionalities.

1. Transfer and cut off power
2. Smooth connection
3. Noise-proof and vibration-proof
4. Fuse of drivetrain

In this paper we focus on a torsional damper which plays a key role for the functionality 3 above, noise-proof and vibration-proof.

Torsional damper mitigates the torque fluctuation from a motor such as ICE (Internal Combustion Engine) and delivers only smoothed driving torque to downstream transmission (T/M, hereafter) (Fig. 2). It reduces the torsional vibration of drivetrain which leads to the elimination of gear noise and booming noise. Adoption of recent advanced environmental technology (such as fewer cylinders or turbo chargers) causes more torque fluctuation, which requires the torsional damper to be more effective.

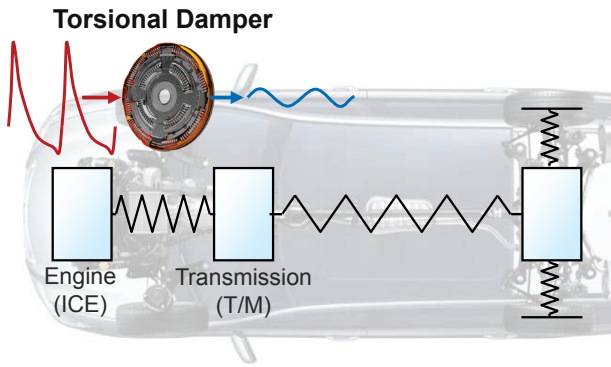


Figure 2. Function of Torsional Damper

2.2 Torsional Damper

Torsional dampers typically consist of multiple straight springs or arc springs. When torque input comes in from ICE side, the springs absorb and release torques repeatedly. Through this mechanism, torque fluctuation is rectified to reduce the drivetrain torsional vibration after T/M.

To optimize drivetrain torsional vibration, it is an effective means to optimize the eigenvalues of torsional vibration of drivetrain. Those eigenvalues are dominated by torsional stiffness of torsional dampers. By setting eigenvalues lower than the driving range (for instance, for Automatic Transmission, under the lock-up lower-limit rotation), silence characteristics during driving is assured (Fig. 3).

However, reducing torsional damper stiffness leads to a constraint on space, because torsional angle range has to be set wider. Also, there is a trade-off by excessive low stiffness, that is, for instance, low-frequency vibration on the vehicle. Therefore, it is necessary to determine design specifications optimally to satisfy all target performances from OEM.

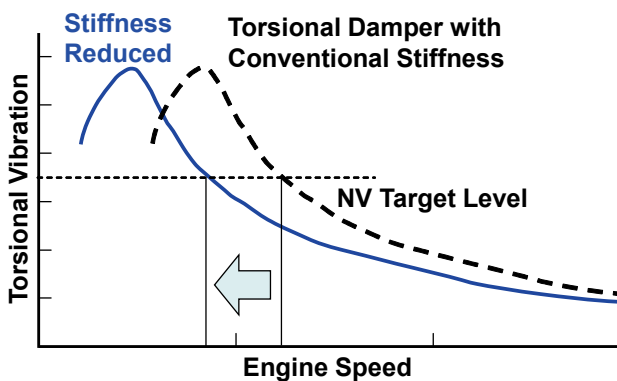


Figure 3. Powertrain Torsional Vibration Mitigation by Torsional Damper Stiffness Reduction

2.3 Characteristics of Arc Springs

Arc springs, which are included in the torsional damper, have non-linear damping characteristics. Fig. 4 shows the torque fluctuation against relative torsional angles when we apply certain amplitude of torque to the arc

spring. We see by those figures that the hysteresis curve resembles a leaf, which means equivalent stiffness and damping coefficients dynamically change depending on rotational speed or input torque amplitude. It is necessary to develop a highly accurate and predictive model to virtually reproduce such hysteresis curves.

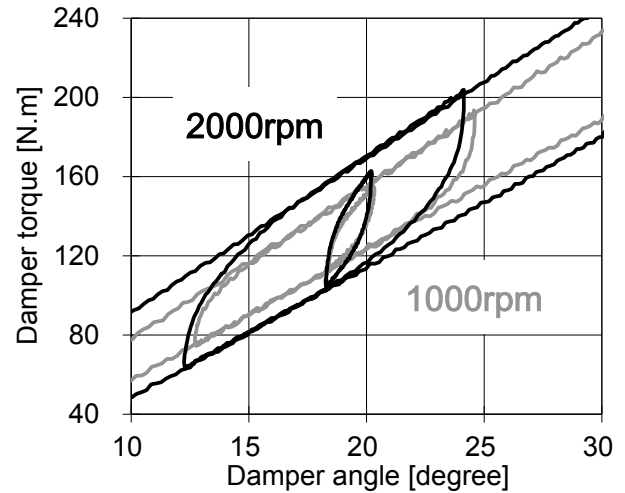


Figure 4. Dynamic Characteristics of Arc Springs

Torsional torque over the relative angle at 1000rpm and 2000rpm on Component Level Test(Experimental data)

3 Modeling Arc Springs

To understand the measured characteristics (hysteresis curve) of arc springs, we use n_{ELM} -number of linear spring elements k_n and mass elements m_n to discretize a continuous spring element (Fig. 5). Here, n is the element number counted from the input torque side, k_n is the discretized stiffness which is the overall spring ratio multiplied by n_{ELM} , m_n is the discretized spring mass which is the overall mass divided by n_{ELM} .

We also define that an arc spring is stored in a cylindrical container in which the inner diameter is the same size as the arc of the spring's outer diameter. One end of the arc spring is connected to the input element, and the end of the other side is coupled to the cylindrical container so as not to rotate relatively. The cylindrical container is connected to the output element. The angle limitation is not included in this modeling.

Considering that a torsional damper is rotating at some speed, centrifugal force is applied onto the mass element which is consequently pressed against the cylindrical container. If a mass element rotates relatively to the cylindrical container because of the input torque to the arc spring, friction torque T_{Fn} would be caused not only by the centrifugal load but also by the reaction force of the arc spring.

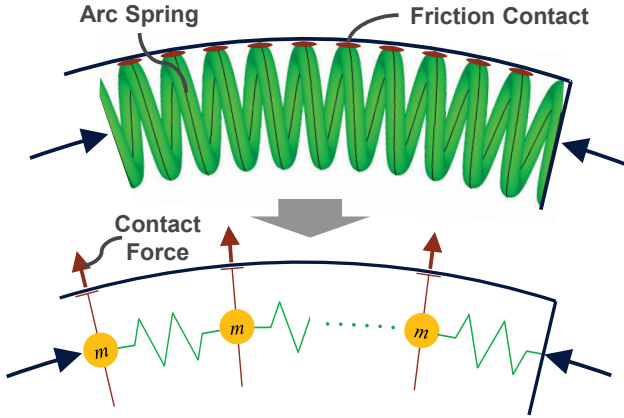


Figure 5. Discretized Model for Arc Springs

3.1 Mechanism Identification

Figure 6 shows the relation of torque and the relative torsional angle between input and output elements and the estimated mechanism, when torque T_{in} is added to a discretized model of a torsional damper. From the figures, we define the relation between T_{in} and output torque T_{out} in several steps.

1. $T_{in} < T_{F1}$:
 T_{in} is transmitted to the output side by the friction torque T_{F1} from a mass element m_1 . Torque is transmitted in a state where mass element m_1 and the output element are coupled, hence $T_{out} = T_{in}$.
2. $T_{in} > T_{F1}$ AND $T_{k1} < T_{F2}$
 T_{in} is transmitted by T_{F1} and friction torque T_{F2} of m_2 . Here m_1 slips relative to the output element, but m_2 is coupled to the output element. Input torque on m_2 is torsional torque on the spring element k_1 ; $T_{k1} = T_{in} - T_{F1}$. At this step, the torsional stiffness is composed of k_1 .
3. $T_{in} > T_{F1} + T_{F2}$ AND $T_{k2} < T_{F3}$
 m_1 and m_2 slip relative to the output element, then k_1 and k_2 work. At this step, the torsional stiffness is composed of the direct stiffness by k_1 and k_2 , so the slope represented the relation between torque and angle (= torsional torque) becomes smaller than condition 2.
4. For other elements, elements gradually move relative to the output elements by the relation between input torque and friction torque. When input torque becomes larger than all mass elements' friction torque, all spring elements will move. Every time the relative torsional direction is reversed caused by the fluctuation of the input torque, the motion is reset and restarts from the step 1.

By the mechanism identified above, we consider that the measured characteristics (a leaf shape hysteresis curve) appear.

Now we define the force equation based on the identified mechanism. The friction torque of one mass element is defined as the equation below:

$$T_{Fn} = \mu_D \cdot r_F \cdot F_{Rn}, \quad n = 1, \dots, n_{ELM} \quad (1)$$

Here, T_{Fn} is friction torque associated with each mass element, μ_D is a dynamic friction coefficient, r_F is a friction radius, F_{Rn} is a pressing force to the friction surface.

The pressing force to the friction surface is distributed as shown in Fig. 7, and defined per the equations below:

$$F_{Rn} = F_{Cn} + F_{SNn} + F_{SN(n-1)}$$

Where:

$$n = 2, \dots, n_{ELM}$$

$$F_{Cn} = m_n \cdot r_A \cdot \omega^2$$

$$F_{SNn} = F_{Sn} \cdot \sin \varphi_{Sn} \quad (2)$$

$$F_{Sn} = k_{Sn} \cdot r_A \cdot \sin \theta_{Sn}$$

Also:

$$F_{SN1} = \frac{T_{in}}{r_A} \sin 0 = 0$$

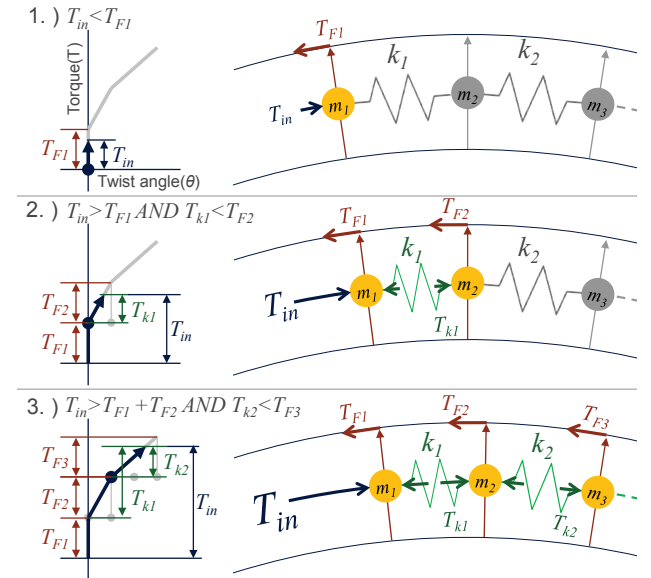


Figure 6. Momentary Behavior of Arc Springs

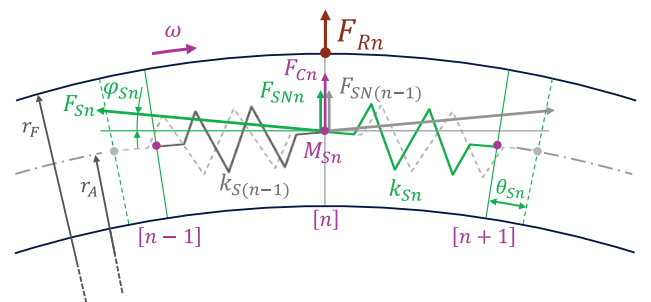


Figure 7. Modeling of Pressing Force on Mass Element

3.2 Study of Discretization Level

The arc spring is wound up dozens of times. Therefore it is important to determine the number of elements for discretization. If we take the actual number of turns for discretization, the simulation model will have several degrees of freedom only with an arc spring. This may lead to high accuracy but slower calculation speed. When the discretization number is too small, the problem comes into calculation accuracy and vice versa (because the detail behavior shown in 3.1 is not reproduced).

So we investigate how much T_F will be impacted depending on the change of n_{ELM} , with representative torsional damper design variables to equation (1) and (2). Fig. 8 shows prediction accuracy versus numbers of elements; here, when n_{ELM} is set as an actual winding number, prediction accuracy of T_F is defined as 100%. By the figure, prediction accuracy is expected more than 98%, when the number of elements is more than 6.

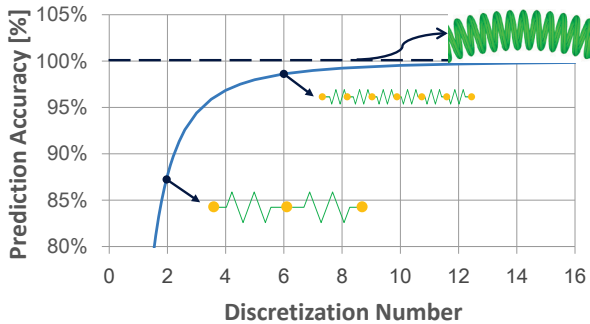


Figure 8. Relation between prediction accuracy and discretization level

When n_{ELM} is set as an actual winding number, prediction accuracy of T_F is defined as 100%.

4 Modeling & Verification

Modelica was chosen to implement the considered mechanism. The reason for this is that compared to other physical modeling tools, we get the following benefits:

- Straightforward description with equations
- Simple mixing of equations and physical models
- Reuse and extension of models due to expressiveness of the source code
- In-house built package can be integrated

4.1 Modeling Arc Spring Components

First, the spring and mass elements modeled using the SpringDamper and Inertia component are already available in the Mechanics.Rotational package included in the Modelica Standard Library (MSL). Internal variables are then bound to output signals so they can be used in the friction torque equations.

Next, the observer component computing the friction torque is created. Equation (1) is rewritten in Modelica code. The spring element information to be used as variables are retrieved from the MSL component outputs. Once computed, the friction torque is also exposed as an output.

When friction torque occurs, internal friction torque component is used. It generates the friction torque according to the output value computed by the observer component.

The arc spring component is based on these subcomponents. In order to validate the estimation accuracy depending on the variation of n_{ELM} , we create several arc spring modules based on different values for n_{ELM} .

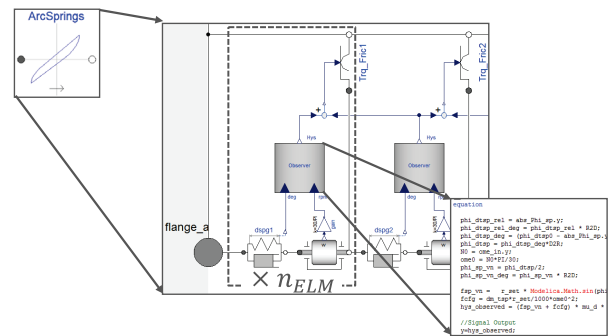


Figure 9. Arc spring component implementation in Modelica

The component is made of 4 subcomponents (SpringDamper, Inertia extended with extra outputs, Friction Torque Observer with the mechanism equations, and FrictionTorque).

4.2 Component Level Verification

In order to confirm the arc spring module correctness and precision, simulations reproducing unit test equivalent to Fig. 4 are run and we compared the obtained results.

On Fig. 10 and Fig. 11, results from a simulation run with $n_{ELM} = 8$ are compared with experimental measurements from Fig. 4. We can observe that the data is mostly matching, and that the input torque oscillations are varying according to a non-linear pattern.

Comparisons for $n_{ELM} = 1 \dots 10$ are shown on Fig. 12. When n_{ELM} is below 6, the non-linearity is not well-preserved and the precision loss observed on Fig. 8 is confirmed.

When comparing the number of generated equations and the overall computation time, it appears that time grows quadratically with the number. To keep reasonable simulation times it is important to have the discretization that would give us a good balance between accuracy and computation cost.

Confronting those results, we can validate the model we built.

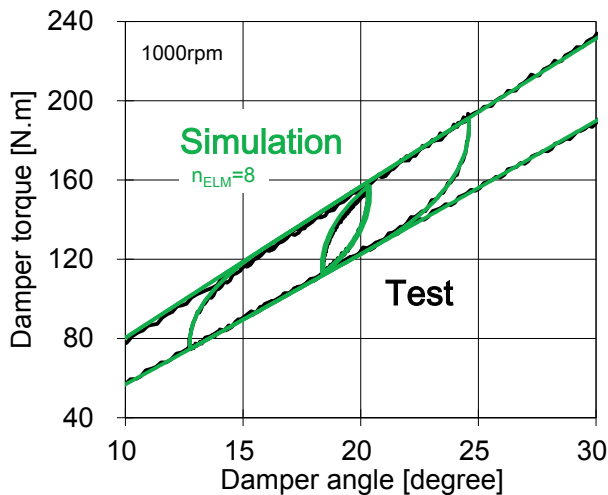


Figure 10. Arc spring model validation (1)
Simulated vs. experimental data for $n_{ELM} = 8$ at 1000rpm

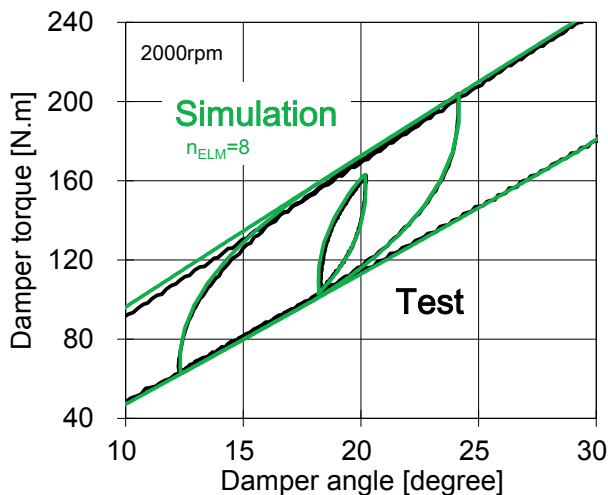
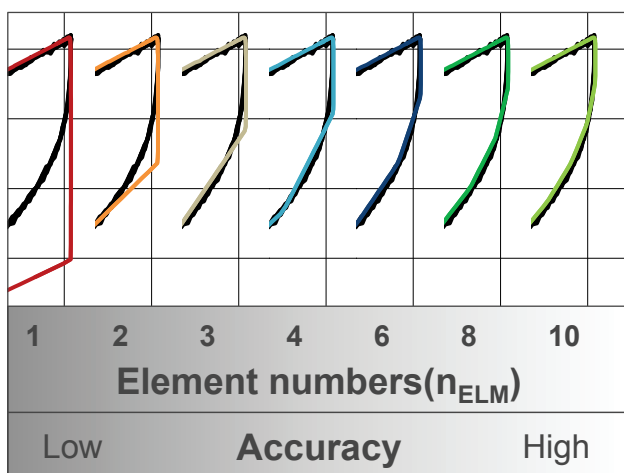
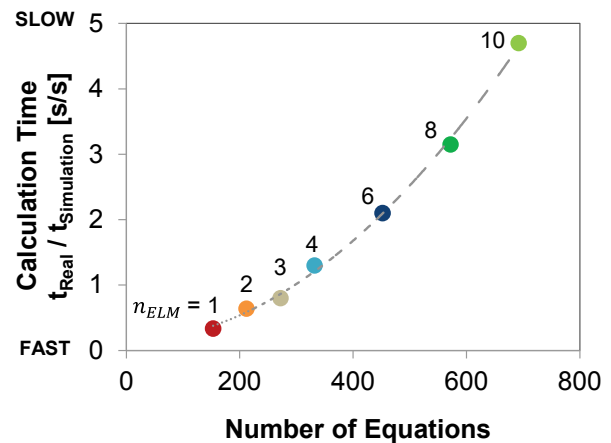


Figure 11. Arc spring model validation (2)
Simulated vs. experimental data for $n_{ELM} = 8$ at 2000rpm



(a) Simulated torsional characteristic (colored lines) compared by measurement data (black line)



(b) Relation between calculation time and number of equations

Figure 12. Comparison with each discretized level

5 Physical model deployment

To make highly accurate physical models accessible to a larger number of engineers, a consistent and easy-to-use interface is required. If the operation varies depending on physical modeling tools, it is ineffective in total because all the work must be done by physical modeling experts.

The capability of having a model run in black-box is also essential. Physical modeling tool seems easy to users and they might carelessly connect components in an unintended way. Behaviors and errors which are not intended by model developers must be avoided at all cost.

By using the Functional Mock-up Interface (referred as FMI from below) as a standard to connect models, we established a process which enables everyone to conduct performance prediction by physical models. Users may only interact with a generic Microsoft Excel interface using the Modelon FMI Add-in for Excel (FMIE). We choose Co-Simulation to export the arc spring model because the binary export license of Dymola is not needed.

FMIE can read FMI 1.0 models exported from MODELICA-based tools such as Dymola, can choose input or output variables, set up scenarios to be run, and execute the simulation, and return the results. However, going through these steps every time is inefficient. So we developed a VBA macro (Fig. 14) to partially automate the process. Once we provide design variables and simulation parameters and press a button, all the relevant simulations are performed and result graphics generated (Fig. 15). With this, anyone can simply run quality simulations only interacting with Excel, without any physical modeling tool even running in the background.

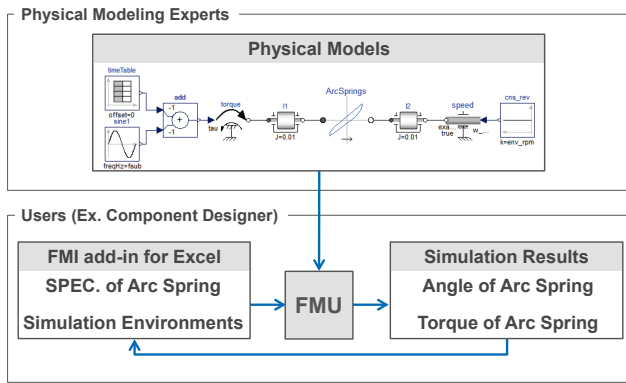


Figure 13. Fast Simulation Method using FMIE

		Simulation				
		Default	Case 1	Case 2	Case 3	Case 4
8	Expiry date					
9						
10	Settings					
11	Start time	0				
12	Stop time	20				
13	FMU	S:\dymola\fmux\torsionalTest.fmu				
14	Log level	Disable				
15	Enable	TRUE	TRUE	TRUE	TRUE	FALSE
16	Output points	2000				
17	Timeout	0				
18						
19	Indata					
20	Name	Variability	Type	Unit		
21	t1	parameter	Real	s	5	
22	t2	parameter	Real	s	10	
23	t3	parameter	Real	s	15	
24	t4	parameter	Real	s	15	
25	tmax	parameter	Real	N.m	511	
26	tmin	parameter	Real	N.m	-5	
27	tpre	parameter	Real	N.m	230	
28	tmax	parameter	Real	N.m	22	

Figure 14. Screenshot of FMIE embedding VBA

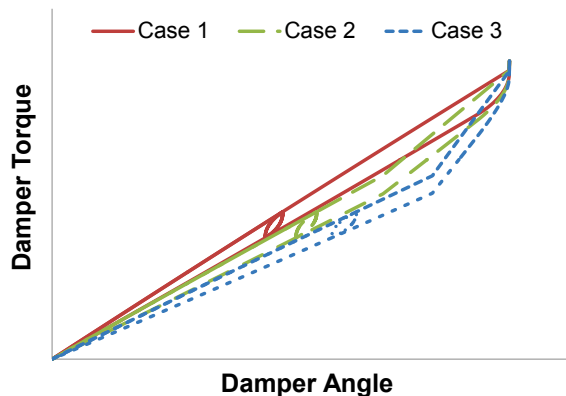


Figure 15. Example of prediction results using FMIE

6 Conclusion

- We developed a discretized model with MODELICA which represents non-linearity of arc springs
- We verified the model accuracy by comparing the dynamic simulation results with measured results of physical tests at component level
- We developed an intuitive interface to design arc spring performance easily and accurately

Those outcomes enable non-experts of physical modeling to run performance design easily with high accuracy. It will lead to the cost reduction of human resources and speed up product development.

Acknowledgements

We gratefully acknowledge the support of Mr. Okabe from Dassault Systèmes and Mr. Gao from Modelon to complete this paper with their technical advices and translation.

References

- Dr.-Ing. Albert Albers. Advanced Development of Dual Mass Flywheel, *LuK Symposium book*, 1994.
- Shinji Hounoki, Katsuyuki Kobayashi, Mitsuhiro Umeyama, Toshihiro Otake. Study of the Two-Mass Flywheel with the Torsional Damper. *Society of Automotive Engineers of Japan (JSAE) Proceedings*, 902138-1:157-160, 1990.
- Ulf Schaper, Oliver Sawodny, Tobias Mahl and Uti Blessing. Modeling and torque estimation of an automotive Dual Mass Flywheel. *2009 American Control Conference*, 978-1-4244-4524-0/09 (WeB16.6 1207-1212), Hyatt Regency Riverfront, St. Louis, MO, USA, June 10-12, 2009.
- Yasuo Shimizu, Nobutaka Tsujiuchi, Akihito Ito, Satoshi Yamamoto. Reduction of Low Frequency vibration at Acceleration/Deceleration by Optimization of Flywheel Damper for Passenger Car. *Society of Automotive Engineers of Japan (JSAE)*, 19-15A:454-459, 2015.
- Yamakaji Yoshihiro. Approaches to quick prediction for Powertrain NVH. *9th International CTI Symposium North America, Novi, USA*, 2015-05-20/21

Research of Model Matching Control of Torque Vectoring Differential Gear System

Yutaka Hirano¹

¹Toyota Motor Corporation, Japan, yutaka_hirano@mail.toyota.co.jp

Abstract

In this paper, model-based development of a control of torque vectoring differential (TVD) gear system is described. A new control logic was developed using model matching control to let the vehicle yaw rate and vehicle slip angle follow the desired dynamics. Simulation results using a single track model of vehicle dynamics are shown to prove the efficacy of the proposed control. Modelica was useful to express time-varying state space system such as the single track model of vehicle dynamics. Also full vehicle model considering all of the vehicle dynamics and drive train motion using Modelica clarified the characteristics of this method in actual driving cases.

Keywords: Model Based System Development, Vehicle Dynamics, Torque Vectoring, Model Matching Control

1 Introduction

To satisfy needs for future low-carbon mobility society, development of many new electric vehicles (EVs) is increasingly active in recent years. Additionally many new proposals about integrated electric power train which also has torque vectoring capability are presented. Authors had made an integrated model of the total vehicle system of such an EV using Modelica (Hirano, 2014) (Hirano, 2015).

In the paper (Hirano, 2014), the authors showed the capability of new construction of the new EV using new type of tire based on ‘Large and Narrow concept’ and torque vectoring differential (TVD) gear. For the model based development of the new EV, various kind of running resistance, vehicle dynamic performance and proper design of electric regeneration system were studied. In another previous research (Hirano, 2015), a multi-physics full vehicle model of the new EV is expanded to consider the detailed loss of motors and inverters. Also front and rear suspension model which has same 3D mechanical design as the real experimental vehicle was made and verified. By technical investigations using this full vehicle model, structure, specifications and control of the new EV system were researched about vehicle dynamics and energy consumption. However, the control logic of the TVD gear was only simple PI feedback control in the previous papers. In this paper, model based control of TVD gear system is developed using model matching control technique. Single track model of vehicle

dynamics is used to derive and verify the new control. At the same time, detailed design parameter of vehicle dynamics was obtained from the analysis of Modelica full vehicle model using detailed suspension model. Finally the developed controls were verified by using both the single track model and the full vehicle model.

2 Specification of Experimental EV

Table 1. Specifications of new experimental EV

	New EV	Conventional car
Vehicle Weight	750 kg	1240 kg
Yaw Moment Inertia	869 kgm ²	2104 kgm ²
Wheelbase	2.6 m	2.6 m
Front : Rear Weight Distribution	0.48 : 0.52	0.62 : 0.38
Height of CG	0.38 m	0.55 m
Aerodynamic Drag × Frontal Area	0.392 m ²	0.644 m ²
Tire RRC	5×10^{-3}	8.8×10^{-3}
Tire Normalized CP	16.1	20.4

The proposed experimental EV has specifications as shown in Table 1 (Hirano, 2015). Compared with a conventional small-class passenger car, the new EV has characteristics of lighter vehicle weight, smaller yaw moment of inertia, lower height of the center of gravity (CG) and lower rolling resistance coefficients (RRC) of tires. Because of these characteristics, this new EV is expected to have better handling and lower energy consumption than conventional vehicles. On the other hand, because of lighter weight and lower value of tire normalized CP (Cornering Power), this new EV seems more sensitive against external disturbances such as crosswind and road irregularity than the conventional cars. To cope with this problem, direct yaw moment control (DYC) was applied by using a new integrated transaxle unit for rear axle which has a main electric motor and also TVD gear unit with a control motor.

3 Vehicle Model for Controller Design

3.1 Single Track Vehicle Model

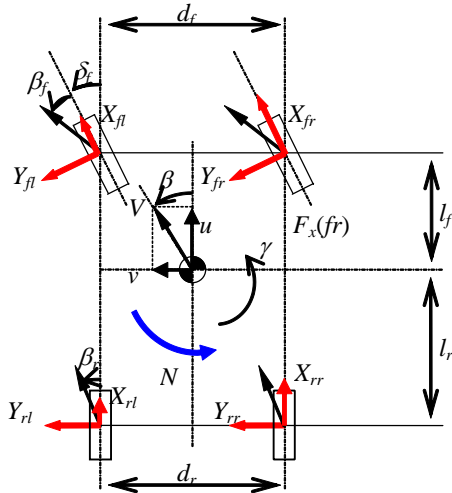


Figure 1. Expanded single track vehicle model

Figure 1 shows an expanded single track vehicle dynamics model to derive the control logic. The simplified equations of motion by this model become as follows.

$$M \frac{dV}{dt} = F \approx (X_{fr} + X_{fl}) \cos \delta_f + (X_{rr} + X_{rl}) \quad (1)$$

$$M \frac{d}{dt} (V \tan^{-1} \beta + V\gamma) \approx Y_{fl} + Y_{fr} + Y_{rl} + Y_{rr} \quad (2)$$

$$I_z \frac{d\gamma}{dt} \approx l_f (Y_{fl} + Y_{fr}) \cos \delta_f - l_r (Y_{rl} + Y_{rr}) + N \quad (3)$$

$$N = d_f (X_{fr} - X_{fl}) \cos \delta_f + d_r (X_{rr} - X_{rl}) \quad (4)$$

Here,

- β : Vehicle slip angle,
- γ : Vehicle yaw rate,
- M : Vehicle mass,
- V : Vehicle velocity,
- I_z : Vehicle yaw moment of inertia,
- $l_f(l_r)$: Distance from the CG to front (rear) axle, (CG: Center of Gravity)
- $d_f(d_r)$: Tread of front (rear) axle,
- X_{**} : Longitudinal force of each tire,
- Y_{**} : Lateral force of each tire,
- δ_f : Steering angle of front tire,
- F : Vehicle driving force,
- N : DYC moment by TVD.

3.2 Equation of Motion for Vehicle Dynamics

To derive the equations of motion for the target vehicle, equations (1) to (4) were further simplified. The lateral force at left and right tires were assumed to be equal and let $Y_{fl} = Y_{fr} = Y_f$, $Y_{rl} = Y_{rr} = Y_r$. Also we assume $\cos \delta_f \approx 1$ when front tire steering angle is not so big, and $\tan^{-1} \beta \approx \beta$ when β is small. Also by considering the TVD power unit is equipped only in

the rear axle, the equations of motion become as follows.

$$M \frac{dV}{dt} = F = (X_{rr} + X_{rl}) \quad (5)$$

$$MV \left(\frac{d\beta}{dt} + \gamma \right) = 2Y_f + 2Y_r \quad (6)$$

$$I_z \frac{d\gamma}{dt} = 2l_f Y_f - 2l_r Y_r + N \quad (7)$$

where

$$Y_f = -K_f \beta_f = -K_f \left(\beta + \frac{l_f}{V} \gamma - \delta_f \right) \quad (8)$$

$$Y_r = -K_r \beta_r = -K_r \left(\beta - \frac{l_r}{V} \gamma \right) \quad (9)$$

$$N = d_r (X_{rr} - X_{rl}) \quad (10)$$

Here, K_f and K_r are the equivalent cornering power of front and rear tire respectively. These values are calculated by using the full-vehicle model described in the section 5.1 to consider the effects of elasticity and friction of suspension and steering.

If driving force F and DYC moment N can be calculated by some control logic, then the target longitudinal forces of left and right rear wheel to be realized by TVD power unit become as follows from equation (5) and equation (10).

$$X_{rr} = \frac{1}{2} \left(F + \frac{N}{d_r} \right) \quad (11)$$

$$X_{rl} = \frac{1}{2} \left(F - \frac{N}{d_r} \right) \quad (12)$$

3.3 Longitudinal Driving Force Controller

Let us suppose the desired value of vehicle speed, vehicle yaw rate and vehicle slip angle as V_{ref} , γ_{ref} and β_{ref} respectively.

The desired vehicle driving force F can be calculated as below by PI feedback control and equation (5).

$$F = M \frac{dV_{ref}}{dt} + K_{PF} (V_{ref} - V) + K_{IF} \int (V_{ref} - V) dt \quad (13)$$

Here K_{PF} is a proportional feedback gain and K_{IF} is an integral feedback gain.

3.4 Model Matching Controller of Lateral Dynamics

3.4.1 Dynamic Model of Vehicle Lateral Dynamics

For the lateral dynamics, the state space form of the vehicle dynamics with TVD control becomes as follow from equation (6) and (7).

$$\frac{d}{dt} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} -\frac{2(K_f + K_r)}{MV} & -1 - \frac{2(l_f K_f - l_r K_r)}{MV^2} \\ \frac{2(l_f K_f - l_r K_r)}{I_z} & -\frac{2(l_f^2 K_f + l_r^2 K_r)}{I_z V} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} \frac{2K_f}{I_z} \\ \frac{MV}{2l_f K_f} \\ \frac{1}{I_z} \end{bmatrix} \frac{\delta_s}{G_s} + \begin{bmatrix} 0 \\ 1 \\ I_z \end{bmatrix} N \quad (14)$$

Here, $\delta_f = \delta_s/G_s$ (δ_s : steering wheel input angle, G_s : steering gear ratio).

Now the matrix form of the state space system of equation (14) can be written as follows.

$$\dot{x} = Ax + Bu + E\delta_s \quad (15)$$

$$x = \begin{bmatrix} \beta \\ \gamma \end{bmatrix}, \quad u = N$$

$$A = \begin{bmatrix} -\frac{2(K_f + K_r)}{MV} & -1 - \frac{2(l_f K_f - l_r K_r)}{MV^2} \\ \frac{2(l_f K_f - l_r K_r)}{I_z} & -\frac{2(l_f^2 K_f + l_r^2 K_r)}{I_z V} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (16)$$

$$B = \begin{bmatrix} 0 \\ 1 \\ I_z \end{bmatrix} \quad (17)$$

$$E = \begin{bmatrix} \frac{2K_f}{G_s MV} \\ \frac{2l_f K_f}{G_s I_z} \end{bmatrix} \quad (18)$$

Please note that the elements of the matrix A of the equation (15) as shown in the equation (16) are dependent on the vehicle velocity V , namely time-varying variables.

3.4.2 Desired Dynamics Model for Lateral Motion

The desired dynamics of vehicle yaw rate and vehicle slip angle are assumed as the first order lag function of steering wheel input as below.

$$x_d = \begin{bmatrix} \beta_{ref} \\ \gamma_{ref} \end{bmatrix} = \begin{bmatrix} \frac{k_\beta}{1 + s\tau_\beta} G_{\beta 0} \\ \frac{k_\gamma}{1 + s\tau_\gamma} G_{\gamma 0} \end{bmatrix} \delta_s \quad (19)$$

Here, $G_{\beta 0}$ and $G_{\gamma 0}$ are steady state gain of slip angle and yaw rate respectively from the steering input. k_β and k_γ are gain of desired slip angle and desired yaw rate from the steady state gain of each state variables.

τ_β and τ_γ are time constant of desired slip angle and desired yaw rate as the first order lag function. Each state variables of slip angle and yaw rate at steady state can be calculated by solving the following equation

$$0 = Ax_0 + E\delta_s \quad (20)$$

and be obtained as follow.

$$\begin{aligned} x_0 &= -A^{-1}E\delta_s \\ &= -\frac{MI_z V^2}{4K_f K_r (l_f + l_r)^2 - 2MV^2 (l_f K_f - l_r K_r)} \\ &\quad \times \begin{bmatrix} -\frac{4K_f K_r l_r (l_f + l_r)}{MI_z V^2} + \frac{2l_f K_f}{I_z} \\ \frac{-4K_f K_r (l_f + l_r)}{MI_z V} \end{bmatrix} \frac{1}{G_s} \delta_s \end{aligned} \quad (21)$$

Thus, $G_{\beta 0}$ and $G_{\gamma 0}$ can be calculated as follows.

$$\begin{aligned} \begin{bmatrix} G_{\beta 0} \\ G_{\gamma 0} \end{bmatrix} &= -\frac{MI_z V^2}{4K_f K_r (l_f + l_r)^2 - 2MV^2 (l_f K_f - l_r K_r)} \\ &\quad \times \begin{bmatrix} -\frac{4K_f K_r l_r (l_f + l_r)}{MI_z V^2} + \frac{2l_f K_f}{I_z} \\ \frac{-4K_f K_r (l_f + l_r)}{MI_z V} \end{bmatrix} \frac{1}{G_s} \end{aligned} \quad (22)$$

The state space form of the desired dynamics can be written as below from the equation (19).

$$\dot{x}_d = A_d x_d + E_d \delta_s \quad (23)$$

Here,

$$A_d = \begin{bmatrix} -\frac{1}{\tau_\beta} & 0 \\ 0 & -\frac{1}{\tau_\gamma} \end{bmatrix} \text{ and } E_d = \begin{bmatrix} \frac{k_\beta}{\tau_\beta} G_{\beta 0} \\ \frac{k_\gamma}{\tau_\gamma} G_{\gamma 0} \end{bmatrix}$$

3.4.3 Model Matching Control of TVD

A state equation of the error between desired values and actual values of state variables can be obtained as below by subtracting equation (23) from equation (15).

$$\dot{e} = Ae + Bu + (A - A_d)x_d + (E - E_d)\delta_s \quad (24)$$

$$e = x - x_d$$

Let's assume the virtual control input U as below.

$$BU = Bu + (A - A_d)x_d + (E - E_d)\delta_s \quad (25)$$

Then the equation (24) can be transformed as below.

$$\dot{e} = Ae + BU \quad (26)$$

Now we can design the feedback control gain K as

$$U = -Ke \quad (27)$$

by using various linear control theories for the equation (26). Though, as mentioned above, the matrix A is

time-varying and dependent on vehicle velocity. To cope with this problem, an analytical solution using pole placement technique was used. By combining equation (26) and equation (27), the following equation is obtained.

$$\dot{e} = Ae - BKe = (A - BK)e \quad (28)$$

If we specify the pole of the dynamic system of the error e of equation (28) as p_1 and p_2 ($p_1, p_2 < 0$) and $K = [k_1, k_2]$, following equation can be derived.

$$|sI - (A - BK)| = (s - p_1)(s - p_2) \quad (29)$$

Here, s is the Laplace operator and I is the unit matrix. Above equation can be rewritten as follow.

$$\begin{aligned} & \left| s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ \frac{k_1}{I_z} & \frac{k_2}{I_z} \end{bmatrix} \right) \right| \\ &= s^2 - (a_{11} + a_{22} - \frac{k_2}{I_z})s + a_{11}(a_{22} - \frac{k_2}{I_z}) - a_{12}(a_{21} - \frac{k_1}{I_z}) \\ &= s^2 - (p_1 + p_2)s + p_1 p_2 \end{aligned}$$

Thus, the following simultaneous equation can be obtained.

$$\begin{cases} (a_{11} + a_{22} - \frac{k_2}{I_z}) = (p_1 + p_2) \\ a_{11}(a_{22} - \frac{k_2}{I_z}) - a_{12}(a_{21} - \frac{k_1}{I_z}) = p_1 p_2 \end{cases} \quad (30)$$

By solving the equation (30) analytically, we can obtain following solutions of k_1 and k_2 .

$$\begin{aligned} k_2 &= I_z (a_{11} + a_{22} - p_1 - p_2) \\ k_1 &= I_z \left\{ \frac{p_1 p_2 + a_{11}(a_{11} - p_1 - p_2)}{a_{12}} + a_{21} \right\} \end{aligned} \quad (31)$$

Please note that the above solution of $K = [k_1, k_2]$ is also dependent on vehicle velocity. (See equation (16).)

Finally, from the equation (25), the following solution of u ($= N$) can be calculated.

$$u = B^+ \{-BKe - (A - A_d)x_d - (E - E_d)\delta_s\} \quad (32)$$

Here, B^+ is a quasi-inverse matrix of B , and consequently $B^+ B = [0 \ I_z]$. ($B^+ B = 1$.) Finally we obtain the following solution of u .

$$u = -Ke - B^+(A - A_d)x_d - B^+(E - E_d)\delta_s \quad (33)$$

It is understood from equation (33) that the control input of the model matching controller consists of a feedback term of the error between desired value and actual value of state variables and also feedforward terms evoked from desired state variables and also steering input.

Figure 2 shows a plot of the feedback gain k_1 and k_2 by pole placement ($p_1 = -20$, $p_2 = -21$) according to the vehicle velocity.

Though we used analytical solution using pole placement in this paper, it is also possible to design the feedback gain K by gain scheduling method using other linear control techniques according to the change of vehicle velocity.

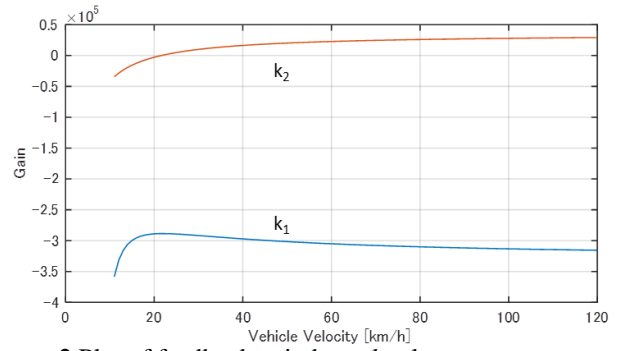


Figure 2 Plot of feedback gain by pole placement

4 Simulation Results by Single Track Vehicle Model

To confirm the validity of above mentioned model matching control, simulation test based on single track vehicle model was performed by using Modelica.

First of all, we should handle time-varying linear state space system such as that of equation (15) to (18). To cope with this problem, a new class of time-varying linear state space system was defined. To achieve this, the standard class of the state space system of Modelica Standard Library (MSL) was modified to release the constraint of variability of variables (i.e. by eliminating 'parameter' qualifier). The definition of the new class becomes as follow.

```

block StateSpace_Variable
...
extends Modelica.Blocks.Interfaces.MIMO(final nin=size(B, 2), final nout=size(C, 1));
Real A[:, size(A, 1)];
Real B[size(A, 1), :];
Real C[:, size(A, 1)];
Real D[size(C, 1), size(B, 2)] = zeros(size(C, 1), size(B, 2));
output Real x[size(A, 1)] (start=x_start)
"State vector";
equation
der(x) = A*x + B*u;
y = C*x + D*u;
end StateSpace_Variable;

model SingleTrackModel
...
Real c0 = 2*(kf+kr);
Real c1 = 2*(lf*kf-lr*kr);
Real c2 = 2*(lf*lf*kf+lr*lr*kr);
...
StateSpace_Variable Actual_x(
  A=A,
  B=B,
  C=identity(2));
StateSpace_Variable Desired_xd(
  A=Ad,
  B=Ed,
  C=identity(2));
...
equation
a11=-c0/m/v;
a12=-1-c1/m/v/v;

```

```

a21=-c1/iz;
a22=-c2/iz/v;
A={{a11, a12},
  {a21, a22}};
B={{cf*m/v, 0},
  {cf*lf/iz, 1/iz}};
Gb0=-m*iz*v*v/(cf*cr*1*1-m*v*v*c1)*(-
cf*cr*lr*1/m/iz/v/v + lf*cf/iz);
Gr0=-m*iz*v*v/(cf*cr*1*1-m*v*v*c1)*(-
cf*cr*1/m/iz/v);
Ad={{-1/t_b, 0},
  {0, -1/t_r}};
Ed={{k_b*Gb0/t_b},
  {k_r*Gr0/t_r}};
...
end SingleTrackModel;

```

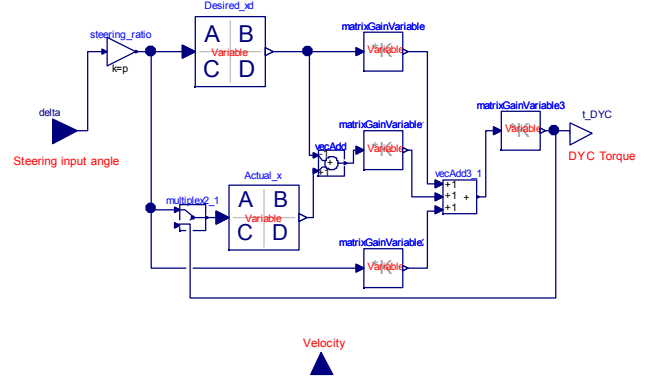


Figure 3. Modelica model of a single track model of vehicle and a controller

For comparison, the definition of the standard class of the state space system in MSL is as below.

```

block StateSpace "Linear state space system"
...
parameter Real A[:, size(A, 1)]=[1, 0; 0, 1];
parameter Real B[size(A, 1), :]=[1; 1];
parameter Real C[:, size(A, 1)]=[1, 1];
parameter Real D[size(C, 1), size(B, 2)]
=zeros(size(C, 1), size(B, 2));
...
equation
der(x) = A*x + B*u;
y = C*x + D*u;
...
end StateSpace;

```

Also a new class of time-varying matrix gain to express the feedback gain by the equation (31) can be made by similar way.

Figure 3 shows a diagram of an example of a single track vehicle model combined with the desired vehicle dynamics model and the model matching controller.

Figure 4 shows a plot of vehicle speed and steering angle input used in the simulation by single track model. The vehicle accelerates from 10km/h to 100km/h between time 1 sec to 10sec. The steering angle moves as 1Hz sinusoidal curve. For comparison, simple PI feedback of desired yaw rate and that of desired slip angle were also tested. The control law of both PI controllers became as follows respectively.

PI feedback of desired yaw rate:

$$N = K_{py}(\gamma_{ref} - \gamma) + K_{iy} \int (\gamma_{ref} - \gamma) dt \quad (34)$$

PI feedback of desired slip angle:

$$N = K_{p\beta}(\beta_{ref} - \beta) + K_{i\beta} \int (\beta_{ref} - \beta) dt \quad (35)$$

Desired dynamics was settled as $k_\beta = 0.3$, $k_\gamma = 1.0$. τ_β and τ_γ are settled as corresponding value of cut-off frequency of 1.3 Hz as shown in the equation (19).

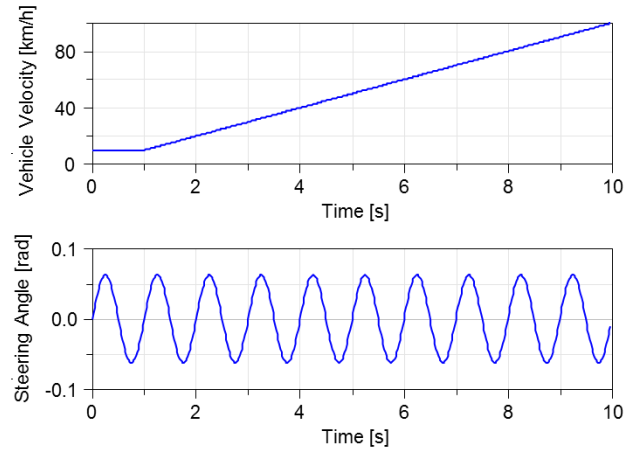


Figure 4. Plot of vehicle velocity and steering angle input

Figure 5 shows comparison of each control. The model matching control showed the best tracking performance of desired slip angle and desired yaw rate. Though, the control input N was bigger than other controls and also the tracking error of yaw rate was bigger especially at the low vehicle speed. Also, it was impossible to let both of the vehicle slip angle and the yaw rate to exactly track the desired value simultaneously. This is because that there are two independent state variables while there is only one control input.

Robustness of the model matching control (MMC) was also checked. **Figure 6** shows comparison of the simulation results of single track model when there are perturbation for the vehicle mass M and tire cornering power CP . For comparison, the result of yaw rate feedback control is also overlaid. MMC showed a good robustness against such parameter perturbations.

It is of course necessary to check the robustness of the control when parameter error of the plant and also other additional effects such as non-linearity and losses exist in the actual world. To do this, simulation tests using full vehicle model was also done as mentioned in the following section.

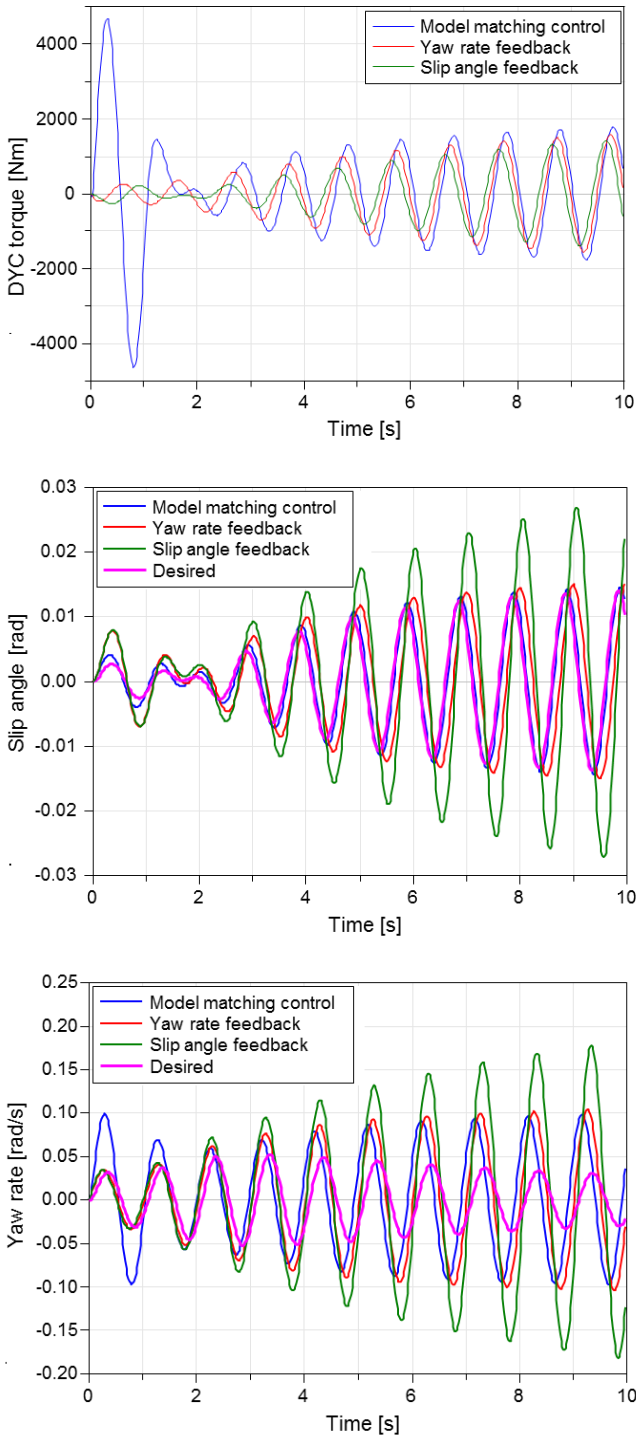


Figure 5. Simulation results by single track model

5 Simulation by Full Vehicle Model

5.1 Construction of the Full-Vehicle Model

The similar full vehicle model as previous research (Hirano, 2015) was used for full-vehicle simulation. The model was developed based on Vehicle Dynamics Library (Modelon, 2014) and was built as a full 3 dimensional (3D) multi-body-dynamic system (MBS) model. Component models of control systems such as TVD gearbox, electric motor and inverter were added with the full vehicle model. **Figure 7** shows the top

level of the model hierarchy of the full vehicle test model and also the power train model with the controller.

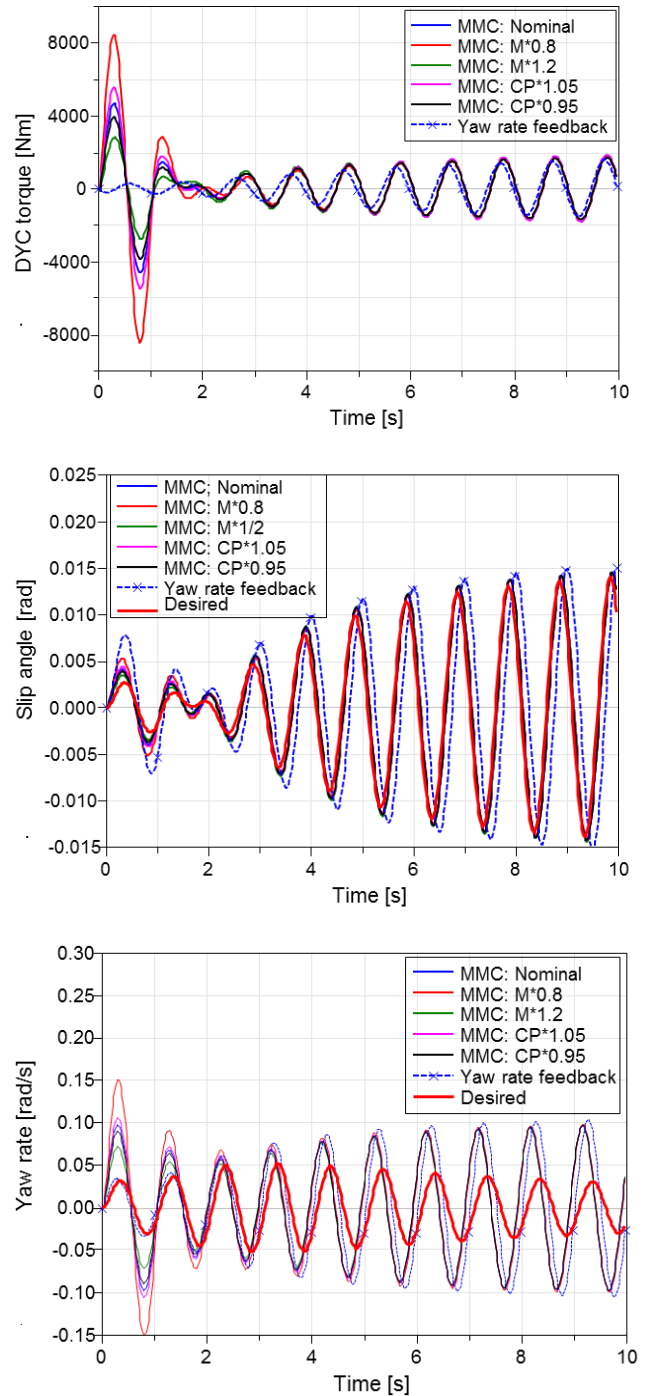


Figure 6. Robustness check by single track model

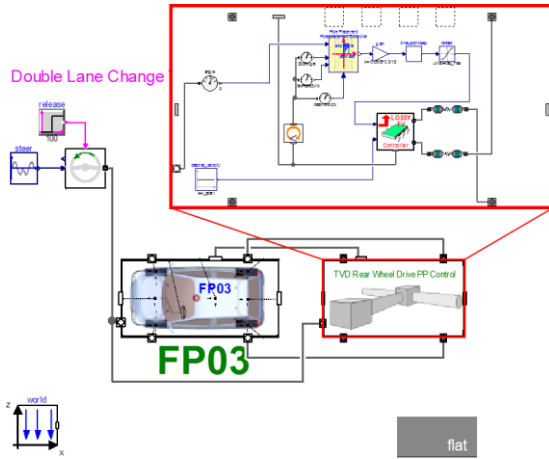


Figure 7. Structure of full vehicle test model

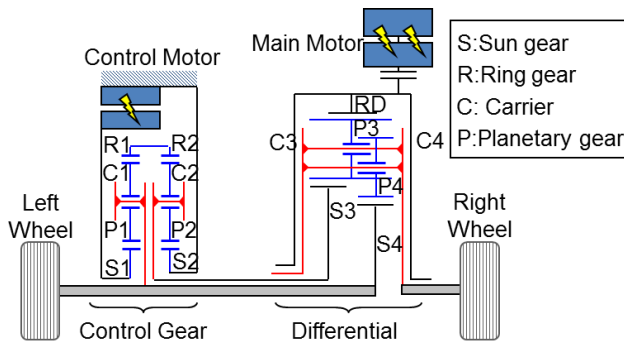


Figure 8. Torque vectoring differential (TVD) driveline

For the TVD gear train, a driveline structure referencing the MUTE project of the Technische Universität München (TUM) (Höhn et al., 2013) was selected. The TVD model was constructed using Power Train Library (DLR, 2013). **Figure 8** shows the configuration of the gear trains. Torque from the main motor is distributed equally to the left wheel and the right wheel through the differential gear. The torque distribution between the left wheel and the right wheel can be controlled by changing the torque input of the control motor.

3D MBS model of suspension, steering and body were installed to calculate vehicle dynamics characteristics. Suspension model was constructed as an assembled model of each suspension linkage, joints and force elements such as spring, damper and bushing. Non-linear tire model based on ‘Magic Formula’ model (Pacejka02) was used to calculate combined lateral force and longitudinal force of each tire. Steering model considered the characteristics of viscous friction of steering gear box and steering shaft as well as steering shaft stiffness. By these detailed models, it became possible to analyze the effects of steering angle change and camber angle change caused by vehicle roll, side force and tire aligning torque.

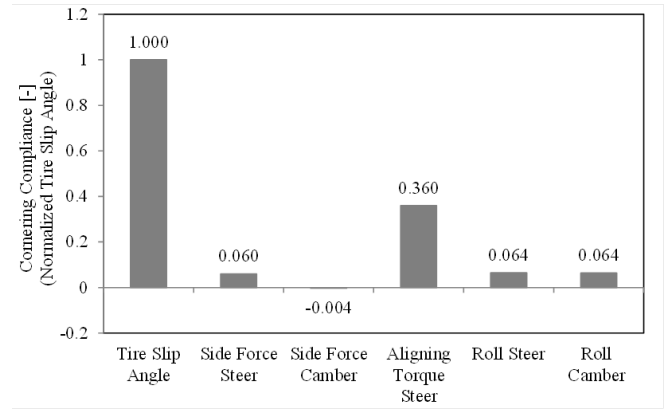


Figure 9. Effect of suspension characteristics to cornering compliance coefficient. (Normalized by the effect of tire slip angle.)

Figure 9 shows an analysis result about the effect of suspension characteristics to cornering compliance coefficient for an example of front double wish-born suspension. The coefficients are normalized by the effect of tire slip angle change. The equivalent cornering power coefficients were calculated by following equation.

$$\varepsilon_{f,r} = \frac{1}{C_{f,r} + \left\{ W_{f,r} \left(\frac{\partial \delta}{\partial F} \right)_{f,r} + W_{f,r} \left(\frac{\partial \gamma}{\partial F} \right)_{f,r} \frac{C_{s_{f,r}}}{C_{f,r}} + W_{f,r} \left(\frac{\partial \delta}{\partial M} \right)_{f,r} + \phi \left(\frac{\partial \delta}{\partial \phi} \right)_{f,r} + \phi \left(\frac{\partial \gamma}{\partial \phi} \right)_{f,r} \frac{C_{s_{f,r}}}{C_{f,r}} \right\}}$$

Here, C_f and C_r are the cornering power of tire itself. The terms in the curly brace of the denominator of the above equation indicates each effect shown in **Figure 9** respectively. Those are the effects by side force steer, side force camber, aligning torque steer, roll steer and roll camber respectively. Finally, the equivalent cornering power coefficients of front tires and rear tires were calculated as ε_f and ε_r respectively. These values are used to calculate the equivalent cornering power of each wheel shown in the equation (8) and equation (9) as bellows.

$$K_f = \varepsilon_f C_f$$

$$K_r = \varepsilon_r C_r$$

5.2 Results of Full Vehicle Simulation

Figure 10 shows the results of a double lane change test by the full vehicle model. Steering angle was given as a series of sinusoidal curves at a constant vehicle velocity of 100[km/h]. The model matching control showed better performance of tracking desired slip angle than the yaw rate feedback control. On the other hand, the yaw rate feedback control showed better performance for tracking the desired yaw rate, though this result can be expected naturally. Additionally, it became clear that the result of the vehicle motion by the model matching control was smoother than that by the PI yaw rate feedback control of desired yaw rate. The reason of this is assumed that feedforward part of model matching control works to improve the response. On the other hand, PI feedback control of the desired slip angle became unstable.

Figure 11 shows the result of full vehicle model simulation for the side wind test. Here, side wind of 20[m/s] blows while Time=2 [s] to 3.5 [s]. The vehicle runs at 120[km/h] and the steering wheel angle is kept to zero. Here, the similar result as the side wind test was obtained. The model matching control was good at tracking performance of the desired slip angle, and the PI feedback control of the desired yaw rate was good at tracking performance of the desired yaw rate. Also it is indicated that the control ability against steady deviation for the model matching controller is not enough. This indicates the necessity of modifying the model matching controller to introduce first order servo control by considering the integral of the error. Anyway both controls showed good performance of vehicle stabilization against the side wind than when no control was applied.

6 Conclusions

Model matching control of TVD was researched by using both linear single track model of vehicle dynamics and multi-physics large-scale full vehicle model. The following conclusions were obtained.

- (i) Proposed model matching control showed a good performance especially for the tracking of the desired slip angle.
- (ii) On the other hand, simple PI feedback control of desired yaw rate was good at tracking the desired yaw rate than the model matching control.
- (iii) Improving the model matching controller to realize servo control of steady error deviation is necessary for future work.

Also for future work, the effect of drive shaft stiffness for TVD control should be investigated. More sophisticated control of tire slip and drive train oscillation should be researched also satisfying the requirement for the vehicle dynamics performance.

References

- DLR, PowerTrain Library Users Guide (Version 2.1.0), 2013
- Y. Hirano, S. Inoue and J. Ota, Model-based Development of Future Small EVs using Modelica, *Proceedings of Modelica Conference 2014*, 2014.
- Y. Hirano, S. Inoue and J. Ota, Model Based Development of Future Small Electric Vehicle by Modelica, *Proceedings of Modelica Conference 2015*, 2015.
- B. Höhn et al., Torque Vectoring Driveline for Electric Vehicle, *Proceedings of the FISITA 2012 World Automotive Congress*, Vol. 191, pp. 585-593, 2013.
- Modelon, A.B., Vehicle Dynamics library Users Guide (Version 1.8), 2014.

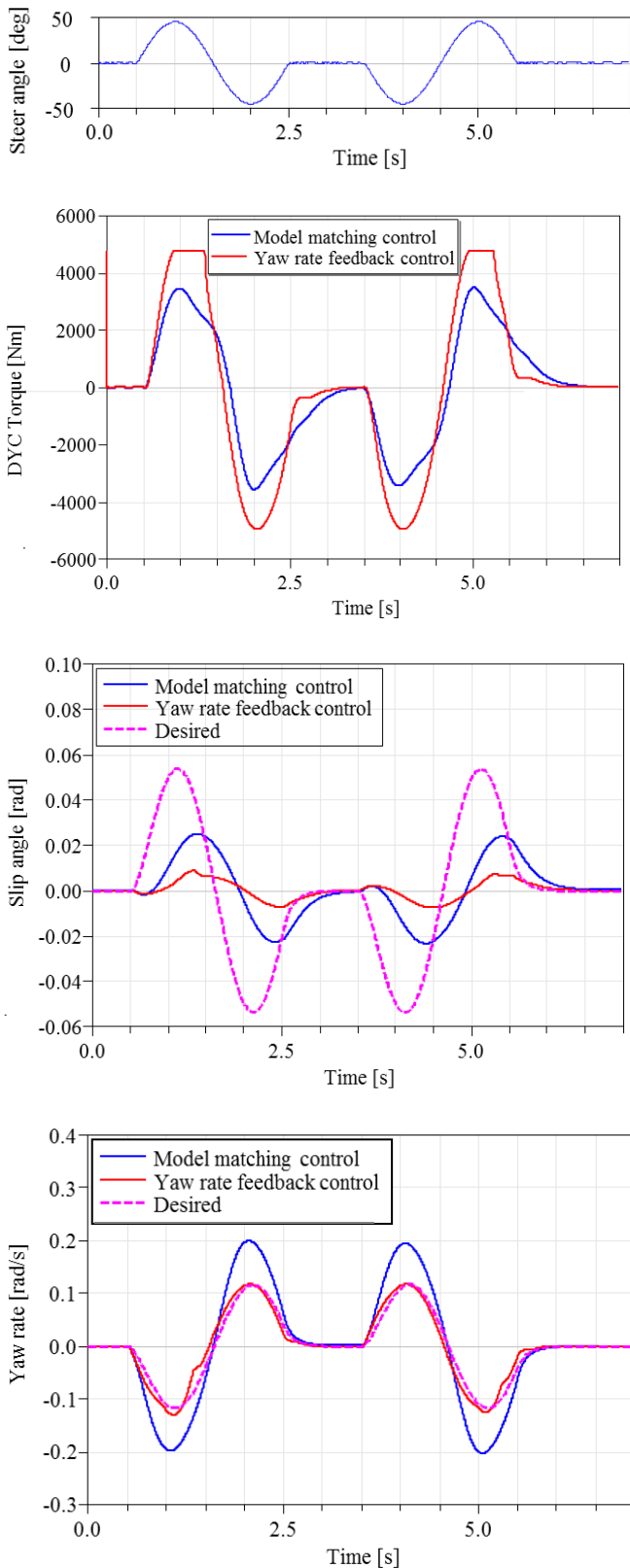


Figure 10. Simulation result of double lane change test by full vehicle model

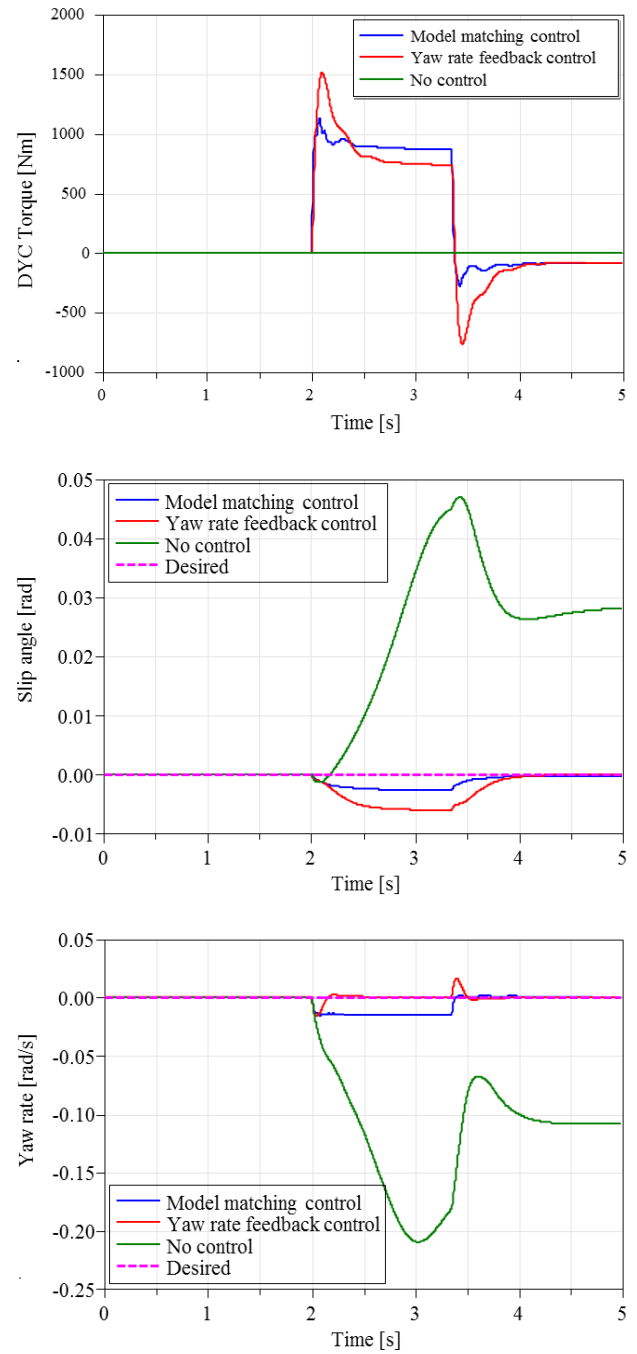


Figure 11. Simulation result of side wind test by full vehicle model

Simulation of Complete Systems at ZF using Modelica Standards

Jochen Köhler¹ Julian King² Michael Kübler³

¹R&D, ZF Friedrichshafen AG, Germany, jochen.koehler@zf.com

²R&D, ZF Friedrichshafen AG, Germany, julian.king@zf.com

³R&D, ZF Friedrichshafen AG, Germany, michael.kuebler@zf.com

Abstract

In this paper we describe how ZF is using existing and upcoming Modelica standards for simulating a variety of systems in automotive industry. In particular, Modelica is employed for driveline modeling. The FMI standard is used to transport models over tool boundaries. The novel SSP standard will contribute towards interconnecting FMUs and defining complete system architectures.

Keywords: Longitudinal vehicle dynamics, FMI, SSP, Parameterization, system architecture, autonomous driving

1 Introduction

ZF is a major automotive supplier, with an extensive product portfolio ranging from driveline and chassis technologies to active and passive safety systems for autonomous driving. In this context, detailed simulation models mimicking vehicle dynamics and the involved actuators/sensors are vital for developing reliable control software as well as for supplementing time- and cost-intensive test bed measurements. Simultaneously, in order to optimize the interplay between these components, the demand for holistic simulation approaches is steadily increasing. Complete system analysis is becoming recognized as an important factor for fast and robust system engineering activities as it allows for directly examining the global system behavior and for identifying relevant feedback loops. Modelica is especially suited to this task due to its inherent multi-physics capabilities. Furthermore, the associated Functional Mock-up Interface (FMI) standard facilitates an efficient integration and coupling of specialized sub-models (possibly developed in other tools), thereby going beyond an isolated analysis of single components.

2 Modeling drivelines with Modelica

Facing an enormous variety of driveline concepts and continuously decreasing innovation cycle times, simulation has become a backbone for developing gearbox control software at ZF. In particular, Modelica was introduced in our company over ten years ago and

nowadays represents a standard approach for modeling a wide range of distinct transmission and related actuator concepts. Building on the freely available Modelica Standard library, more than ten context-specific ZF Modelica libraries have been created so far. These in-house libraries are made accessible throughout the company and preserve corporate modeling know-how in several highly relevant areas, such as

- Transmission and driveline components (Köhler, 2005)
- Extensions for hybrid and electrical powertrains (HEV/PEV/EV), including complete hybrid driving strategies (Köhler *et al.*, 2006)
- Model parameterization tools and export templates to other platforms (Kellner, 2006)
- Combustion engine dynamics and exhaust after-treatment (Kuberczyk, Köhler, 2013)
- General mechatronic solutions, hydraulic and pneumatic actuators (based on Modelica Fluid) (King *et al.*, 2014)

The above-mentioned central libraries are consequently re-used within distinct modeling projects, thereby generating thoroughly tested component models characterized by a high degree of reliability and robustness. Also, many component models are available in varying levels of detail and function, from highly resolved descriptions encompassing all relevant physical effects to simplified formulations optimized for real-time use (King *et al.*, 2014).

A major advantage of the object-oriented modeling approach underlying Modelica is that it allows for an easy exchange of such component variants without having to modify other parts of the global model, thereby offering the possibility to quickly switch between different model configurations. Furthermore, Modelica models directly reflect the physical structure of the system under scrutiny. This fact enables a rapid transfer of system knowledge into model equations. On the other hand, also some shortcomings have to be mentioned. Due to the symbolic simplification of the system equations by the Modelica front-end tool, efficient debugging is difficult and the transparency of

the generated run-time C-Code is limited. This hampers the development of complex functions as compared to scripting languages. Moreover, to the authors' experience, Modelica is not yet widely known among young engineers. For all these reasons, further efforts are necessary to improve the interaction between Modelica models and models/methods originating from other tools. The Functional Mock-up Interface (FMI) offers great promise in this regard.

3 Using FMI to modularize system components

Modelica models can be exported to a wide range of simulation tools – usually via C-Code. At ZF this process has been extended to support even tools developed internally. To enhance the flexibility for the usage of simulation tools and to reduce the effort for maintaining this process there is a need to use established standards.

FMI satisfies this need since there is a broad list of tools supporting FMI. We also enriched internally developed tools to support FMI, which are used for system simulations, e.g. in case of CO2 analysis or software development and test. They are able to handle FMI 1.0 and 2.0 both in model-exchange and co-simulation mode. The verification of this FMI interface is done by applying the FMI cross-check rules.

In the past mostly monolithic models have been used as FMU, but recently the demand for a modular setup is increasing, internally and with customers of ZF. This is motivated by two requirements on virtual product development: efficiency and quality. Both requirements can be fulfilled if models, once built up by a modeling expert reflecting all needed physical effects, are reused and exchanged amongst different departments or even companies.

A crucial point here is a feasible definition of the simulation architecture for all relevant use cases. Therefore one needs to think about a proper definition of the interface signals, in order to

- enable an easy integration of existing models
- replace models of different levels of details
- regard existing solutions (e.g. within other simulation tools) inside the company and with customers.

The usability must not be neglected; therefore our aim is to decompose the overall system to smaller, but still reasonable sub-systems. These sub-systems mostly represent the components developed by the collaborating partners, e.g. in the case of analyzing longitudinal dynamics models of a combustion engine, a transmission system and vehicle dynamics, as shown in Figure 1.

When cutting tightly coupled systems to modules also numerical issues such as algebraic loops or stiff

systems must be considered. FMI for model-exchange might be an option for such a simulation setup, but FMI for co-simulation is in focus because it enables the comparability of the simulation results and shows a feasible performance in most tools. Figure 1 shows an example of a modular setup used for software development and test. The focus of the modular setup currently is on physical or simplified logical models but will also be extended on virtual ECUs.

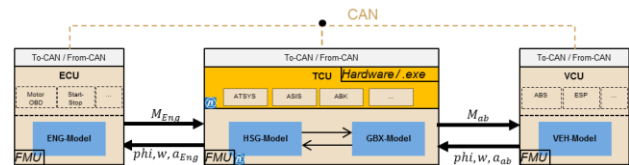


Figure 1. Structure of a driveline

This definition of interfaces needs discussions and it is good to take all parties into account. Therefore parts of those discussions are also handled in a cross-company approach, e.g. the “Smart Systems Engineering” project of the ProSTEP iViP Association (cf., <http://www.prostep.org/en/projects/smart-systems-engineering.html>).

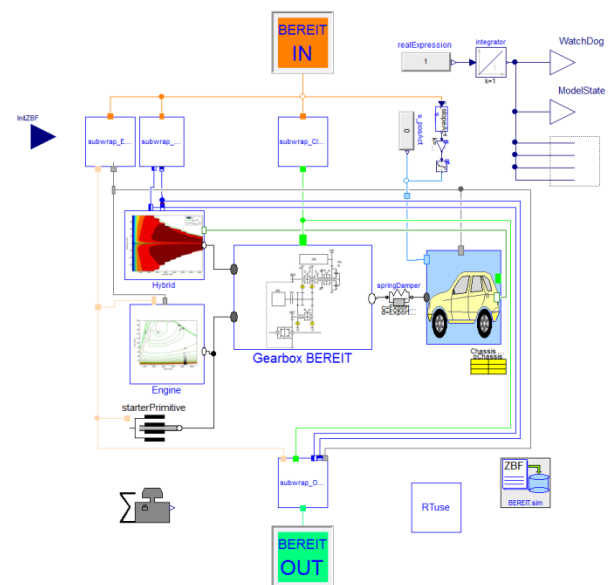


Figure 2. Model structure of the BEREIT range extender transmission concept in Dymola

The use of FMI to exchange behavioral simulation models has become an official standard in ZF. A prototypic example is the joint research project “BEREIT” (Bezahlbare Elektrische Reichweite durch Modularität und Skalierbarkeit – Affordable Electric Range by Modularity and Scalability), supported by the German Federal Ministry of Economics and Technology (cf., <http://pt-em.de/de/1508.php>). The goal here was to develop gearbox control software for a modular range extender concept for electric vehicles, featuring three possible operation modes: a purely

electric mode, a hybrid mode with load point shifting, and an “EDA” mode, which allows for charging the battery while the vehicle is in standstill or driving at low speeds (Roske *et. al.*, 2006). In order to minimize transmission losses, only dog clutches (which are synchronized by the electric motor) are used as switching elements for shifting gears.

Figure 2 shows the top-level model in Modelica/Dymola, including blocks for loading ZBF parameterization data as explained below (see chapter 5), as well as two connectors containing all input and output signals for communicating with the control software. This minimal interface remains the same for all model configurations, while the driveline modules may be freely exchanged. Typically, either the whole model or the gearbox part is then compiled as a co-simulation FMU to be imported into the in-house SiL-platform SOFTCAR (see below), which has been enabled to handle both FMI 1.0 and 2.0.

In Modelica an extensive use of expandable connectors is used to exchange measured signals e.g. between controller and plant model. Unfortunately this approach analog to a CAN-Bus in real vehicles currently cannot be obtained when switching to FMI. Therefore wrapper models and mapping functionalities are needed for the seamless integration of those FMUs, which will be addressed by the Modelica SSP project, as mentioned below.

Establishing FMI as the all-in-one solution for model exchange in ZF is still on-going since all the processes need to be mature enough and all issues which arise with the use of FMI need to be solved.

One major drawback so far is the lack of proper protection of the intellectual property within the exported FMU. To overcome this we established a post-procedural encryption with the following options:

- selection of signals and parameters to hide
- definition of a period of validity
- checksum usage to guard against manipulation

Another main issue is the consistent parameterization of a system simulation setup by a bunch of modular sub systems, which motivated the SSP (“system structure and parameterization”) project.

4 Using “SSP” to specify system architectures

4.1 Motivation

Same as for the products itself that ZF offers, each simulation model has to be provided to several customers. The goal is to reuse the same simulation model of a certain product as a FMU. The challenge here is the adoptions that are needed for any customer, because there is no standard way to couple things especially in the area of personal cars. One could think

of including the “kernel” FMU into a wrapper with all the signal modifications but this means quite an overhead, because usually these modifications are quite simple (linear manipulation or mapping tables for discrete values).

In Addition to that it has to be taken into account that also the receiver of the simulation model (e.g. the customer but also another department in ZF) has to integrate this FMU into his complete system. Assuming that the complete system is built from several FMUs as component representation the main additional information that defines the system architecture is the connection of all FMUs and a possible hierarchical arrangement. This requirement isn’t focused by the FMI standard.

As a consequence a new Modelica Association project called “System Structure and Parameterization” (abbreviation SSP) was initiated after the Modelica Conference 2014 in Lund. One main goal of this project is to define a tool independent format to be able to specify the structure of a simulated system. So one has to define this structure only once in any (authoring) tool and can transfer it to any (integration) tool to include the system components and simulate the system. This is especially interesting when a system is simulated on different platforms like MIL, SIL or HIL.

First results of the project were presented 2015 at the Modelica Conference in Paris. There is a first draft for defining system structures. A XML schema was developed therefore. Three prototypes of tools were presented that are able to read and write such files.

ZF is developing its own software for integration of physical models of driveline with ECU software code called SOFTCAR in parallel to commercial ones. On top of that another tool is being developed that generates complete simulation models for the simulation platforms SOFTCAR and dSpace® out of selected FMUs, several (CAN-)bus specifications and parameterization definitions. For handling all these system architectures and parameterization data, the SSP approach will be used.

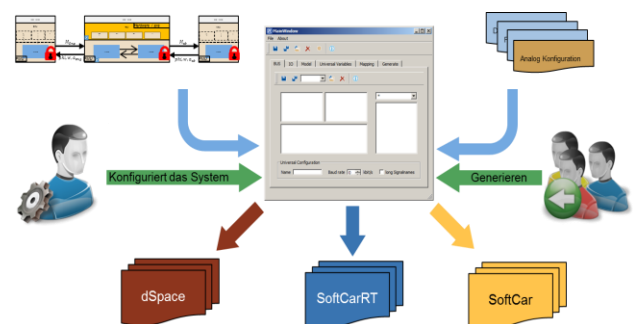


Figure 3 Sketch of mapping tool

The element in the SSP XML schema called “signal dictionary” is here very useful for connecting inputs and outputs of FMUs not directly but with a separate instance where you can define project independent

signal names that can be reused in many projects. Signal dictionaries can also be thought as “special” version of FMUs without any dynamics. Another useful feature is that the signal dictionaries can be used easily for visualization of these signals.

4.2 SSP for simulation of autonomous vehicles

Usage of SSP is also intended in ZF for developing reusable system architecture for ADAS (advanced driver assistance systems) and autonomous vehicles. In these systems there’s a lot of feedback and interdependencies between all involved components. So the need of being able to simulate not only the isolated components but also the complete system is evident for a fast development process.

The system architecture is quite complex and the variety of combining components is very high. ZF offers a lot of components needed for autonomous driving:

- Sensors
 - Radar
 - Cameras
- Actuators
 - Steering
 - Braking
 - Transmission
 - Electric Machines for traction
- (Active) chassis components
- High performance ECUs
- Functions for autonomous driving

There’s a little gap to end up with a complete virtual system by adding the missing parts.

- Engine
- Car body
- Driver
- Environment

The first three parts can be modeled quite simple for the purposes here. Having a model of the environment is very important and this is probably provided by a tool vendor.

4.3 Requirements

The system architecture to be defined has to fulfill the following requirements:

- It must fit to real systems.
- Defined modules have to correspond to real components.
- Variations of modules can be exchanged – also by real components on HiL-platforms.
- It must be possible to include also modules from OEM / other suppliers.
- The numeric coupling of modules must be robust for efficient and stable simulations.

- It must be possible to both simulate the complete systems and fragments of it.
- It must be usable on a variety of platforms (MiL, SiL, HiL).

Having this system architecture also allows extracting single components as empty template. You can give this template to somebody else for implementing. When finished, this new component model can be easily integrated in the complete system.

Of course this work just starts now. But it gives a good overview of the opportunities by using the SSP approach.

5 Using SSP for parameterization

The project “System Structure and Parameterization” also focuses to parameterization of components. In the FMI standard, the dynamics of a component and its parameterization is not separated. We experienced that this approach can be inconvenient sometimes. Another problem of this entity is our requirement to use the ZF internal standard for parameterization called “ZBF”. This is a simple format of ASCII files to specify simulation parameters separately from simulation models and tools. Already before using FMI we used it to be able to parameterize models just during the initialization.

The advantages are:

- A Model can be provided with multiple parameter sets without the need to rebuild it.
- The parameter sets can be either readable or encrypted.
- Parameter definitions can be used for multiple components without duplication (“single source” pattern).
- Parameters can be hidden to the model user if wanted.
- Parameters can be modified by the model user even though the model is a black box, if wanted (Intellectual property issues)

SSP can be used to have all these features without being forced to give up the internal standard because there will be a functionality to implement your own adapters to the SSP API. The part of parameterization of SSP is under development at the moment. As soon as there is a first version, ZF will implement its own adapter. So it’s possible to take benefit of the other coming features of SSP:

- The possibility to handle parameters within any authoring tools that support SSP seamlessly.
- Use same parameterization approach also for entire system models with many components (either FMUs or proprietary models)
- Enrich parameterization data by meta data
- Handle IP issues for parameterization

- Efficient handling of complex parameters (lookup tables etc.)

SSP brings its own data format as a XML definition. But with the possibility to implement your own adapter it's also possible to transfer proprietary formats to the SSP standard format.

6 Outlook

For ZF it is important to contribute to Modelica projects like FMI and SSP to be sure to get powerful standards and reliable tools supporting them in order to do the jobs that have to be done. The number of simulation tasks will grow rapidly in future, so it's inevitable to be efficient in this context.

Also the cooperation of industrial users on these projects benefits everybody; industry gets tools that fit their needs and tool vendors can offer more attractive software.

Especially the work on SSP is not finished yet. Some effort has to be made to bring the upcoming standard to a mature status so it can be used in daily business. We try hard to make this happen quickly by evaluating the results very early to get experience with it. Another important point is the close cooperation with the FMI project group.

Acknowledgements

Thanks to the Modelica Design group for being open to suggestions and requirements of industry.

The great engagement of SSP project members helps a lot to create and establish the new standard.

References

- M. Kellner and M. Neumann and A. Banerjee and P. Doshi. Parameterization of Modelica Models on PC and Real Time Platforms, *Proceedings of the 5th Modelica Conference 2006*, pp. 267-274, Vienna, Austria, 2006
- J. King and J. Köhler and M. Kübler. Multi-platform simulation models for the development of transmission control software in commercial vehicles. *Simulation and testing for automotive electronics V*, pp. 56-65, 2014
- J. Köhler and A. Banerjee. Usage of Modelica for transmission simulation in ZF. *Proceedings of the 4th Modelica Conference 2005*, pp. 587-592, Hamburg, Germany
- J. Köhler and T. Mauz and J. Schnur. Systematische Entwicklung von Simulationsmodellen und Fahrstrategien für hybride Antriebe. *VDI Berichte*, pp. 473-500, 2006
- J. Köhler and R. Kuberczyk. Simulationsverfahren zur Auslegung eines PHEV-Antriebsstrangs. *VDI-Tagung Plug-In-Hybride*, 2013
- R. Kuberczyk and J. Köhler and S. Blattner. Benchmark of Saving Potentials of Diesel-Hybrid Vehicles. *22. Aachener Kolloquium*, 2013
- M. Roske and F.-D. Speck and S. Kersch. Das elektrodynamische Anfahrerelement - ein Hybridantrieb mit

erweiterter Anfahrunktionalität. *VDI-Tagung Getriebe in Fahrzeugen*, 2006.

Virtual Vehicle Kinematics and Compliance Test Rig

Peter Sundström¹ Maria Henningsson¹ Xabier Carrera Akutain² Yutaka Hirano³ Alejandro Ocariz²
Hiroo Iida³ Naoki Aikawa³ Johan Andreasson⁴

¹Modelon AB, Sweden, {peter.sundstrom,maria.henningsson}@modelon.com

²Toyota Motor Europe, Belgium, {Alejandro.Ocariz,Xabier.Carrera.Akutain}@toyota-europe.com

³Toyota Motor Corporation, Japan, {hiroo_iida,naoki_aikawa,yutaka_hirano}@mail.toyota.co.jp
⁴Modelon KK, Japan, johan.andreasson@modelon.com

Abstract

This paper presents a virtual kinematics & compliance (K&C) test rig, also known as a Suspension Parameter Measurement Machine, SPMM. The focus is to explain the requirements and implementation of the rig model is built to be a virtual equivalent to the physical test rig, capable of reading the same input and producing the same output.

The virtual test rig is implemented as a Modelica model that is plug compatible with any vehicle model using the standard interface from the Modelica Vehicle Dynamics Library. The operation of the virtual test rig is done from a scripting environment that executes a co-simulation FMU.

An example test case is also shown where results from a virtual test is compared with the corresponding run on the physical test rig.

Keywords: virtual testing, kinematics and compliance (K&C), FMI, vehicle dynamics

1 Introduction

A kinematics and compliance (K&C) test rig, Figure 1, is one of the most common way to benchmark and fingerprint cars today. A model which can reproduce the excitations and generate output in exactly the same fashion as an established physical rig presents several advantages:

The most obvious one is the time reduction in the execution of the complex tasks of the K&C test rig, measurable in terms of orders of magnitude. Carefully considering the testing efficiency, the level of confidence and relative importance of each, the K&C test rig owner can choose to allocate the testing resources more selectively. For instance, a batch of lower priority tests could be skipped and replaced with simulation, provided the data available and/or measured from the fundamental tests is enough to generate a high fidelity base vehicle model.

Most of the OEMs are pursuing a strategy of creating multiple derivatives of the same vehicle platform which in turn generate a need for extensive testing. Again, provided a reliable base model, the K&C simulation of

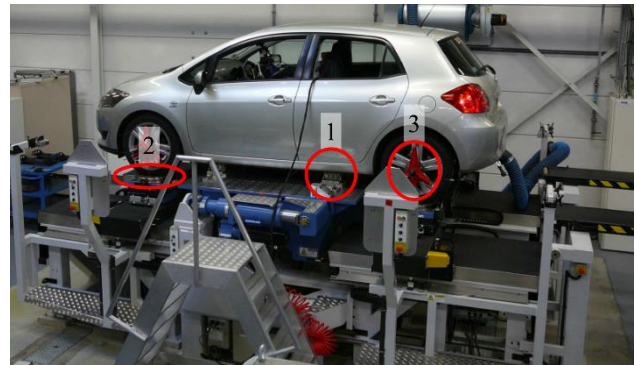


Figure 1. Kinematics and Compliance (K&C) rig in operation at Toyota. The vehicle body is clamped (1) so that the rig can induce roll, pitch and bounce motion of the body. Under each wheel, there is a pad (2) that can move and rotate to induce tire forces and moments. On each wheel, sensors (3) that measure wheel center motion are attached.

these derivatives present a major time and cost saving in terms of vehicle availability and testing resources can be significant.

Very connected to this, the utilization of the virtual environment makes as well possible to perform system identification in order to achieve higher fidelity of the model to study, especially important if there's a significant amount of unknown parameters on the initial model. A more precise parametrization can be used for the optimization of the base vehicle model and all its derivatives according to the desired elasto-kinematic targets. Similarly, quick and inexpensive virtual sensitivity analysis of the model can help to quantify the robustness of the base performance and to study potential performance improvements instead of replacing chassis components with limited criteria.

The ownership of complete K&C test rigs is restricted to big or specialized corporations. On top of the reasons exposed above, sharing a common simulation rig can enhance the cooperation significantly between different industry or academia partners for every effort on chassis optimization. In this joint effort the authors have targeted a full replication of the excitations and

workflow of an existing K&C test rig with the highest fidelity possible, in order to make it a useful tool in the vehicle development cycles. Additionally, to make it deployable throughout the organization, the following requirements are identified:

1. The operation of the virtual test rig should mimic the real one, especially it should be compatible with the data formats used
2. It should be possible to operate the virtual test rig without knowing the model and/or how to operate the modeling environment.

Figure 2 gives an overview of the developed toolchain and its connection to the physical test rig and related tools. The core of the toolchain is a Modelica model that is compiled as a co-simulation FMU that is able to pull parameter data from a data base (Andreasson, J. et. al., 2016). This is then executed from the deployment environment according to the test specification. The input to and the output from the virtual test rig are compatible with the corresponding real test rig signals. This allow data to be exchanged at (1) and (2). Additionally, there is a tuner that can be used to fit parameters to measurement (or other simulation) data.

The deployment environment is MATLAB with FMI support (FMI Toolbox for MATLAB, 2016) and it is chosen since the platform is widely available for the target engineers to meet requirement 2 listed above. However, the deployment could just as well be carried out in for example Python (Andersson C., 2013), (Nilsson, T., 2013) and (pyFMI, 2016) or MS Excel (FMI Add-in for Excel, 2016).

2 Vehicle Test Rig Model

Testing a vehicle in a test rig as opposed to on a test track is beneficial for a number of reasons. The rig can accurately reproduce load cases while iterating on suspension setups or part changes which is critical when evaluating how these changes affect vehicle performance. Furthermore, the rig can produce load cases which are difficult, dangerous or even impossible to achieve in driving scenarios.

Evaluating the kinematics and compliance of vehicle suspensions gives good insight into how the wheel and tire will move relative to the chassis and thus also what forces will be generated for different vehicle states. Essentially, the K&C curves for a chassis is the fingerprint of the suspension which can readily be compared between different setups or competitor vehicles to predict performance.

The results from a K&C test are usually expressed in the form of curves or gradients. The curves describe how for example the toe angle changes with wheel travel, known as bump steer. These curves give vehicle dynamics engineers a good idea of the performance of the vehicle and are important in defining and evaluating requirements.

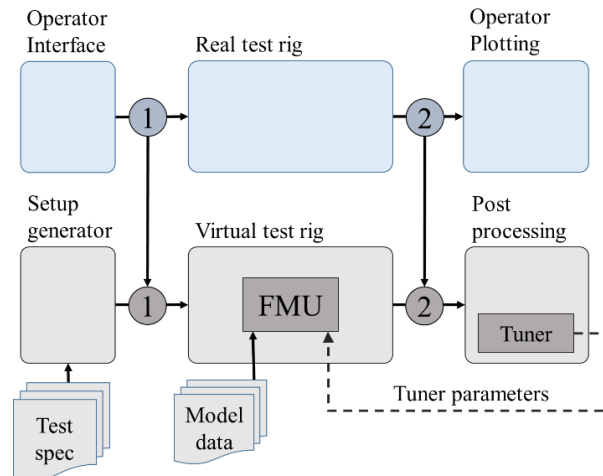


Figure 2. Toolchain overview. The real test rig receives information on execution (1) from the operator interface and then output measurement results (2) that is used for plotting. The virtual test rig is designed to be able to read and write compatible data.

The main independent quantity for K&C curves is normally vertical wheel travel. For a non-steered, independent suspension, the kinematics can be completely defined as a function of wheel travel. For steered suspensions, steering wheel angle or steering rack displacement is normally used as the second independent quantity. Suspension where there is a coupling between left and right sides (dependent or semi-dependent suspensions) need the wheel travel of the opposite wheel as an additional independent quantity.

Toe/steer and camber/inclination angles are key dependent quantities to study as they greatly affect the tire force generation. The change in lateral and longitudinal position are also important as this gives additional information about the momentary center of rotation of the suspension.

The compliance properties of the suspension are key to vehicle handling since these affect how the wheel angles and position changes with load, e.g. during cornering or braking. While typically measured in rigs capable of higher frequency excitation than described here, compliance also has large effect on how noise and vibration is transmitted through the chassis. Details of suspension kinematics and compliance effect on vehicle performance can be found in standard literature, see e.g. (Bastow, 2004) or (Milliken, 1995).

2.1 Physical Test Rig

The virtual test rig is based on an off-the-shelf physical test rig capable of exciting the vehicle chassis as well as the in plane motion of the wheels (Anthony Best Dynamics Ltd, 2014).

Before a test is performed, the vehicle is driven onto the test rig and the body is clamped to the rig table. The

table is able to move vertically as well as roll and pitch the vehicle body relative to the ground plane. The wheels of the vehicle are positioned on wheel pads which can move in the ground plane and rotate around the vertical axis to generate planar forces and torque at ground level. By changing the wheel clamping method, forces and moments can also be applied in all directions at the wheel hubs.

The rig can perform many different test cases including the basic roll, pitch and heave movements of the chassis as well as force sweeps at the tires and/or hubs at varying ground heights.

2.2 Virtual Test Rig

The virtual test rig is designed to mimic the functionality of the physical test rig and to allow them to be operated in a consistent way by sharing parameterization for configuration as well as data formats. This has some important implications on the test rig model implementation. Figure 3 shows a screenshot of a vehicle model on the virtual test rig. The large yellow plate corresponds to the table to which the body is clamped and the four circular pads are used to actuate the wheels.

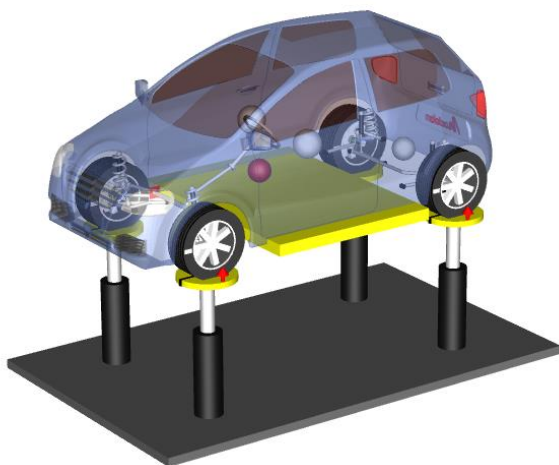


Figure 3. Animation screenshot of a vehicle in the virtual test rig. The vehicle is equipped with an independent steerable front suspension of MacPherson type and a dependent rear suspension of twist beam type.

Since the model is deployed as FMUs in a non-Modelica environment, care is taken to design the virtual test rig so that it does not have to be recompiled for the different operating modes. Additionally, since typical test procedure contain a sequence of events, it must be possible to chain these events together while maintaining states and other properties of the FMU. Figure 4 shows the diagram layer of the rig model including the tested vehicle model. Note the inputs, both real and Boolean signals which are used to control the rig. The test rig model is built using the Vehicle Dynamics Library (Andreasson et. al., 2006) and is

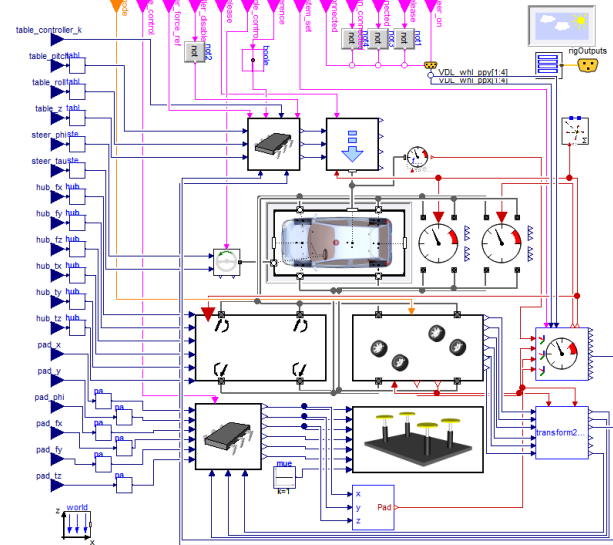


Figure 4. Diagram layer of the complete rig model.

compatible with the corresponding interface and template structure.

Each test starts with an initialization phase. Usually, the table is set to be released during this phase to let the vehicle settle. A flag *coord_system_set* is switched to true at an appropriate time, usually when the vehicle has settled, which then locks the vehicle coordinate system relative to the chassis and the ground coordinate system relative to ground. References for vertical load control and wheel travel are also stored for use in the test procedure. While the *coord_system_set* flag is set to false these coordinate systems will float in the ground plane to be centered w.r.t. to the wheelbase and track width of the vehicle. The coordinate systems are used for output signals and need to be stored after the vehicle is settled to mimic the physical procedure.

The table can then move to generate bounce, roll and pitch motion of the body relative to ground. Each of these degrees-of-freedom can be prescribed by an input signal or be controlled to maintain a fixed level of vertical load or wheel travel.

To allow these different modes to be contained in one FMU, the model is equipped with inputs that allow for each degree-of-freedom to be set independently from the others. By having these as inputs, the modes can be switched during a simulation that makes it possible to conveniently describe complex sequences as described further in Section 3.

Controllers are built to handle several different modes by using appropriate scaling parameters so a single set of controller parameters can be used for both force and position control. The control error for each wheel (force or travel) is multiplied by a coefficient contained in a 4x3 matrix which controls how the control error for each wheel affects each of the 3 table

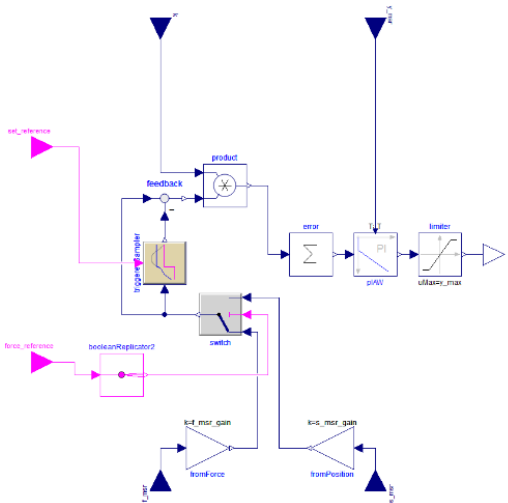


Figure 5. Controller block for one degree of freedom of the chassis table.

degrees-of-freedom. Figure 5 shows the controller block for a single degree of freedom of the chassis table. The bottom inputs are vertical load and wheel travel vectors for the wheels, each multiplied by scaling parameters. The Boolean input on the top left triggers storing of the control reference on its positive flank. The bottom left Boolean input switches between force and position control. On the top, the left input is k , a vector corresponding to one column of the matrix mentioned above.

By setting the control parameters individually for each degree-of-freedom, it is for example possible to have a prescribed roll angle applied by the table while adjusting table bounce and pitch motion to maintain a constant total vertical load on each axle. Table 1 shows how the settings required to achieve this. The roll column has only zeroes as this is prescribed motion. The bounce column states that if load is too high, the chassis should be lifted. The pitch column state that the pitch angle should be increased if the front load is too low and/or the rear load is too high and vice versa.

The wheel pads can also be either force or position controlled. In order for the pads to be able to generate

force in the tires, a tire model capable of windup in all in-plane directions is required. Typically, state-of-the-art single contact point tire models do not have this capability. Therefore, a dedicated stand-still tire model is provided which acts as a linear spring-damper in the six degrees of freedom between the wheel and the pad. The model has different operating modes depending on how forces should be applied. For example, all forces and torques except pure vertical force can be disabled.

There are also inputs available to change the suspension behavior. The stabilizer and tie rods can be individually disconnected, power steering can be switch on or off and the steering rack motion can be locked.

Table 1. Table controller table for roll test with constant axle load.

	Roll	Pitch	Bounce
Left front	0	-1	1
Right front	0	-1	1
Left rear	0	1	1
Right rear	0	1	1

3 MATLAB Environment

A user interface based on the MATLAB scripting language has also been developed. This interface uses functions from the FMI Toolbox for MATLAB/Simulink to import a co-simulation FMU.

A set of standard test setups is stored in an excel spreadsheet. This spreadsheet mimics the one used for parameterizing tests on the physical test rig. There is a column for each parameter that needs to be set in the rig model and each test is defined in one row. To run a specific test, the specification for that test is read from the corresponding row in the spreadsheet based on a unique test number. The test specification is then loaded into a test object in the MATLAB environment which is sent as an argument when running the test using the test rig. The test specification can be modified after it is read from the spreadsheet by changing variables in the test

Test no	Sequence	Test Name	cycle time	amplitude	description	amplitude code
0	Set-up	Set-up (vehicle initial condition)	10			0
1.1	1.Lateral Compliance	Lateral Compliance in phase	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
1.2		Lateral Compliance anti phase	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
1.3		Lateral compliance left wheel input	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
1.4		Lateral compliance right wheel input	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
2.1	2.Longitudinal Com	Longitudinal Compliance in phase	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
2.2		Longitudinal Compliance anti phase	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
2.3		Longitudinal Compliance left wheel input	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
2.4		Longitudinal Compliance right wheel input	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11
3.1	3.Aligning Torque C	Aligning Torque Compliance in phase	45	<i>st</i>	<i>static tyre vertical</i>	STVL -11

Figure 6. Excerpt from the spreadsheet containing test specifications. Each column corresponds to a parameter setting for the rig model, each row corresponds to a test.

object. shows a small section of the test specification spreadsheet.

When comparing results from the physical rig to simulation results for K&C correlation work, the measurement files coming from the physical test rig can be read in and presented on the same format as is produced by the virtual test rig functions. This makes it easy to compare virtual test runs to the corresponding tests performed on the physical rig.

A typical set of commands to run a rig test from Matlab looks like this:

```
test_rig = VirtualKCTestRig('FMUfile.fmu');
test_setup = CreateKCTest('3.3.1');
result = test_rig.run(test_setup);
plot(result.time,result.get_channel(57));
result.save_xml('resultfile.xml');
```

The first command, `VirtualKCTestRig()`, initializes the test rig object, `test_rig`, by loading the FMU. `CreateKCTest()` is then used to pull the relevant test parameters from the excel parameter sheet and store it in `test_setup`. The `run()` command of the test rig object is then called with the test setup object as an argument to run the test. The results can then be plotted directly or stored in an xml file compatible with the format of the real test rigs.

4 Application Example

An example of a test run in the virtual rig is shown here. The vehicle used for the example has an elasto-kinematic McPherson front suspension and a twist beam rear suspension with bushing mounts. During the work with the virtual rig, a new twist beam suspension model was also developed. The following plots focus on the kinematics of this rear suspension model. Parameterization is based on hard point data and other known quantities as far as possible, but some are estimated.

The test shown is a roll test with constant axle load. Figure 7 shows the roll angle measured from the physical test rig as well as the simulated roll angle in the virtual test rig. As the roll motion is directly prescribed for the test, the two signals match very well.

The vertical motion of the chassis table is used to maintain a constant total vertical load on the four wheels during the test. Figure 8 shows the measured and simulated values for the total vertical load.

The kinematics of the rear suspension for the roll test are illustrated by the following plots. Magnitudes of the signals are hidden for confidentiality reasons. Figure 9 shows the bounce motions of the two rear wheels plotted against their respective longitudinal displacement. Figure 10 shows the corresponding lateral displacement. Wheel angle, toe and camber, correlation is shown in Figures 11 and 12. The hysteresis and asymmetric behavior in the real suspension that can be seen in the

measurements is not accounted for in the model which limits the accuracy that can be achieved.

Compared to the real test rig, the virtual version provides some important advantages. The user is free to change parameters in quick iterations. Depending on complexity level and length of test sequence, the execution of a complete cycle on a standard laptop normally ranges between 2 and 10s. This allow for rapid execution of DOE, and also allow for parameter tuning either to reach desired characteristics or to match with measurement data from the real test rig.

Additionally, the test rig can be used to excite any vehicle that is compatible with the standard interfaces of the Vehicle Dynamics Library, which allow the user to conveniently change vehicle configuration and model fidelity level so that the K&C behavior can be predicted at any time during the design process.

5 Conclusions

With the newly developed test rig model, it is possible to run physical and virtual test rig experiments in parallel. This simplifies correlation work since equal circumstances are ensured in both environments. Also, it facilitates moving certain tests completely to the virtual rig since specifications and output formats are equivalent. Finally, the virtual version allows for rapid iterations due to the fact that simulation and setup time is significantly less then real-time, the virtual representation allow for changes in the model that is very time consuming/expensive/impossible on real prototypes, and that simulations can be distributed onto several machines with little effort.

References

- Andersson, C. (2013), A Software Framework for Implementation and Evaluation of co-Simulation Algorithms, ISBN 978-91-7473-671-7.
- Andreasson, J. et. al (2006), The Vehicle Dynamics Library – Overview and Applications, in Proceedings of 5th International Modelica Conference, Vienna, September 4-5.
- Andreasson, J. et. al (2016), Deployment of high-fidelity vehicle models for accurate real-time simulation, in Proceedings of First Japanese Modelica Conference, Tokyo, May 23-24.
- Anthony Best Dynamics Ltd. (2014). SPMM 5000 - Outline Specification - SP20016 issue 2. Wiltshire, UK.
- Bastow, D. H. (2004). Car Suspensions and Handling, 4th edition. SAE International.
- FMI add-in for Excel (2016) www.modelon.com/products/fmi-add-in-for-excel
- Milliken, W. M. (1995). Race Car Vehicle Dynamics. SAE International. ISBN 978-1-56091-526-3

Nilsson, T. (2013), A simulation environment for coupled systems of discontinuous ODE:s. Chapter 4, PyFMI2.0, ISSN: 1654-6229
 pyFMI (2016) www.pyfmi.org

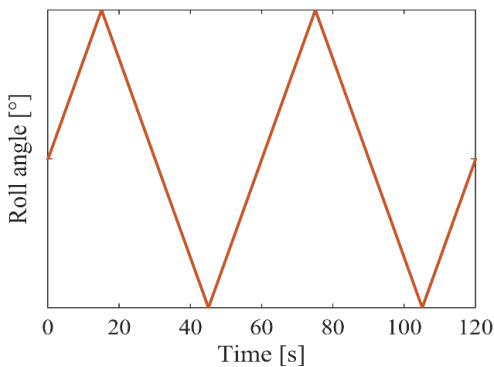


Figure 7. Roll angle, blue curve is from physical test rig, red curve is from simulation.

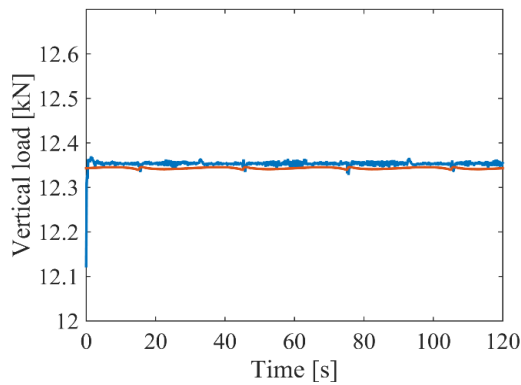


Figure 8. Total vertical force during test, blue curve is from physical test rig, red curve is from simulation.

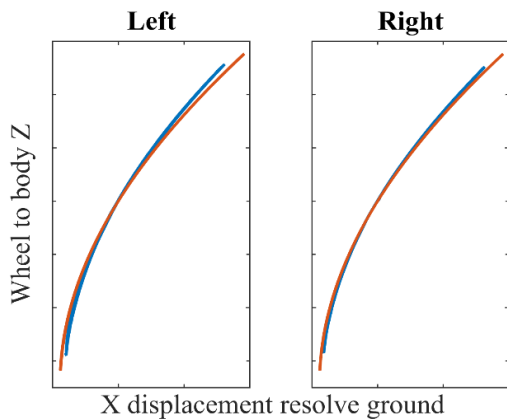


Figure 9. Vertical bounce at the two rear wheels plotted vs the corresponding longitudinal displacement, blue curve is from physical test rig, red curve is from simulation.

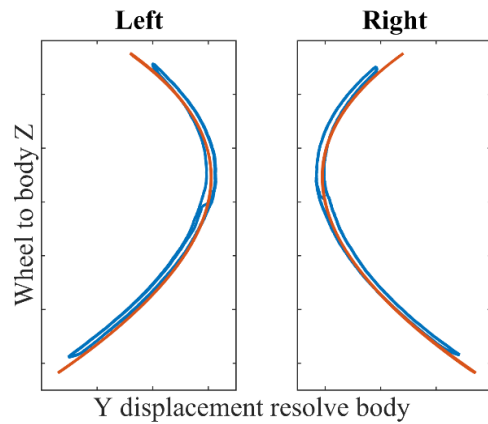


Figure 10. Vertical bounce at the two rear wheels plotted vs the corresponding lateral displacement, blue curve is from physical test rig, red curve is from simulation.

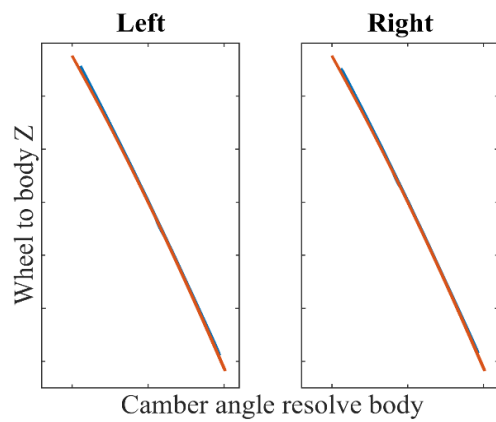


Figure 11. Camber angle at the two rear wheels plotted on the x axis with bounce motion on the y axis, blue curve is from physical test rig, red curve is from simulation.

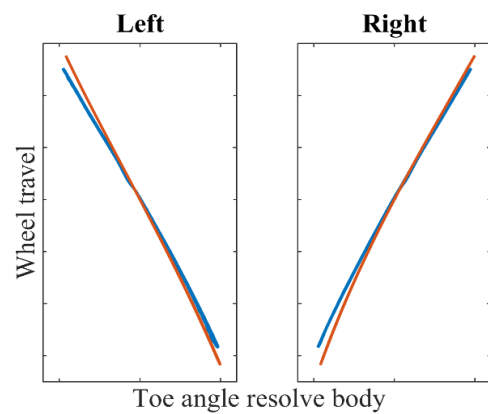


Figure 12. Toe angle at the two rear wheels plotted on the x axis with bounce motion on the y axis, blue curve is from physical test rig, red curve is from simulation.

Modelica-Association-Project “System Structure and Parameterization” – Early Insights

Jochen Köhler¹ Hans-Martin Heinkel² Pierre Mai³ Jürgen Krasser⁴ Markus Deppe⁵ Mikio Nagasawa⁶

¹ZF Friedrichshafen AG, Germany, jochen.koehler@zf.com

²Robert Bosch GmbH, Germany, Hans-Martin.Heinkel@de.bosch.com

³PMSF IT Consulting, Germany, pmai@pmsf.eu

⁴AVL List GmbH, Austria, juergen.krasser@avl.com

⁵dSPACE, Germany, MDeppe@dspace.de

⁶CYBERNET SYSTEMS Co., Ltd., Japan, mikio-n@cybernet.co.jp

Abstract

Starting with the motivation to invent the new standard SSP (“System Structure and Parameterization”) within the Modelica Association and the need to have one more standard beyond the mature Modelica language and the already well established Functional Mockup Interface (FMI) proposed in Modelica Association (Blochwitz *et al.*, 2011), the main use-cases are presented where SSP can help. As SSP relies on XML, the schemas and in consequence the main features for defining system structures and parameterization of models are described. The need to be able to transport complex networks of FMUs between different simulation platforms like MIL, SIL and HIL is emphasized as a motivator for SSP.

A variety of prototypes are shown that support the early version of SSP. This gives a good impression how the standard can be used for quite different tasks and proofs, that system structures can be exchanged between them seamlessly.

Finally the next steps for the ongoing development of SSP are outlined.

Keywords: FMI, System Structure, Parameterization, Collaboration, Standardization

1 Introduction

It’s still a very big challenge for different kinds of industry areas to build up simulation models for behavioral simulation of complex systems consisting of multiple domains. In the engineering process we are used to separate a system into its components and do all the necessary simulations for one component in a tool that fits best to the specific problem. Good results can be achieved in this way if the dynamic behavior of one component has no large impact on the other parts of the system. But as the systems to be developed become more complex and the interaction between all components becomes more important or is even essential for the product value the simulation of the

connected parts is inevitable. Figure 1 shows a typical example from the automotive industry, where component models have to be combined for overall simulation in different environments.

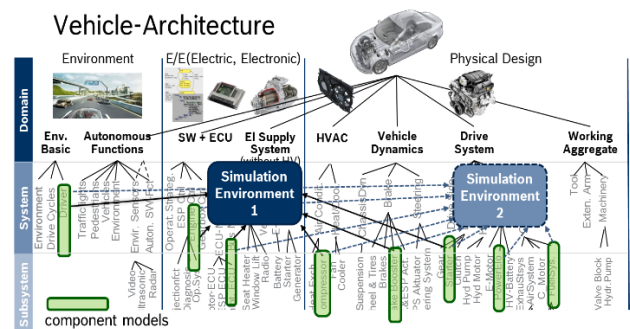


Figure 1. System simulation in automotive industry

The attempt to model all physical domains within one tool could be a potential solution, e.g. modeling languages like Modelica can handle this quite well.

One large benefit here is the possibility that during the translation process of the complete system a lot of mathematical simplification mechanisms can be used to optimize the mathematical problem and make it easier to solve the DAE during simulation with one single solver. However due to the complexity of the language and the fact, that other simulation tools are quite more established in certain domains it is very hard to enforce this approach in a company, or for collaborative development across companies.

The second best approach came with FMI. The standardized Functional-Mockup-Interface gives the possibility to export an FMU (Functional mockup unit) of a component from the authoring tool that was used to build it and integrate it in another environment to simulate it. Of course in this integration environment other component FMUs can be integrated as well to connect them all into a complete virtual system. But once again this can be done only in a proprietary way

for this single integration tool and the built-up system structure cannot be transferred to another environment. However, this is a common use case when the modeled system has to be used in different targets like MIL, SIL or HIL.

Another issue when simulating complex systems is parameterization. FMUs can be parameterized in an isolated way but there is no convenient way to handle these parameters e.g. exchanging complete parameter sets or handling any associated intellectual property concerns. To parameterize complete systems a “global” instance has to exist to handle all the parameters that are not part of a component or have to be used in several components at the same time. These features are quite relevant when models of components are interchanged between different departments in one organization or - even more important - between different companies.

These should not be considered as disadvantages of the FMI approach because FMI does not have these issues in its scope.

These thoughts were presented first at the Modelica Design Meeting in 2014 in Lund by BMW, Bosch and ZF and there was a commitment to instantiate a new Modelica Association project called “System Structure and Parameterization” to develop mechanisms and standards to enhance the existent FMI standard. It is important to emphasize that this new standard is developed in close cooperation with the FMI project group to secure a perfect fit of both standards.

In the last months there was quite good progress starting with the definition of different use-cases to describe various scenarios handling system structures and parameters. Derived from that a number of XML schemas have been developed to describe both system structures and the parameterization of complete systems in a standardized way that can be used independently of specific tools.

First evaluation of this could be shown at the Modelica Conference 2015 in France where three different tools were presented that were able to read in the same XML representation of a simulation model and handle it in their specific ways.

The next sections give detailed insights into the use cases, the actual status of the XML schemas and a short presentation of the already existing tool prototypes that make use of this upcoming standard.

2 Use Cases

In the following chapter the basic use cases and the principal solutions by the SSP-project are described

2.1 Parameter Exchange

In future, updates and effective variant handling for models will be done predominantly by parameter sets. To do this effectively in a heterogeneous environment, we need a tool independent standard. Figure 2 and

Figure 3 show the use cases and possible solution for a single model and for a structure of models

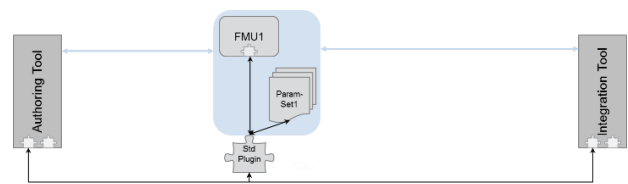


Figure 2. Exchange of one FMU/model with multiple different parameter sets

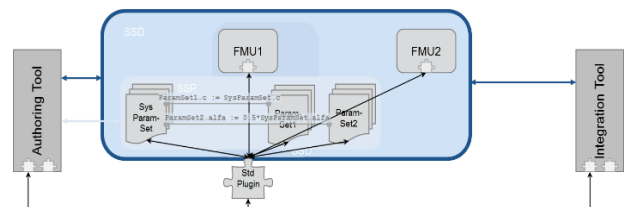


Figure 3. Describing parameter sets for system architecture

2.2 Model Structure

As shown in Figure 1 the multiple use of (sub-) structures of models in heterogeneous environments get more important. For the seamless and tool independent usage of networks of components, we need a standardized format for the connection structure, which also support basic mathematical manipulation of signals (for manual unit conversion or mapping of discrete signals). Figure 4 shows the approach of the SSP project.

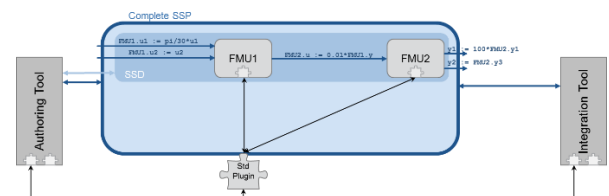


Figure 4. System architectures with signal modifications

2.3 Model Structure and Parametrization

Use case 2.3 is the combination of use case 2.1 and 2.2 (Figure 5). The structure and the according parameter sets have to be handled in a tool independent standard.

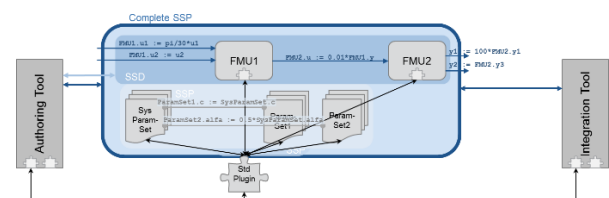


Figure 5. System architectures with signal adoption layer and parameter sets

3 XML Schemas

The file formats defined for the MAP-SSP project are intended to provide a minimal interchange format between different tools, not as a replacement for tool-specific formats, and are focused on the exchange of information on systems needed for their execution or integration into other systems. The interchange of architectural information between architecture tools (e.g. SysML-based tools, for which XMI already provides an interchange format) is out of focus of the current efforts,

The formats try to duplicate as little information as possible from any referenced component formats, like FMUs, and try to be agnostic as to the detailed semantics of the connections being described, submitting to the semantics definitions of the relevant standards for e.g. FMUs for actual connection semantics. In this way the format should be useful for many different purposes and should potentially be compatible with currently envisaged FMI standard developments, like e.g. structured ports.

By defining both a basic system structure file format (SSD) and a format for packaging the SSD and its related resources, including referenced SSDs/SSPs and FMUs into an easily transportable archive (SSP), the proposal tries to offer flexibility in the way system structure is being exchanged in different contexts, e.g. within companies using PLM systems or between companies in a customer/supplier context.

Currently XML schemas have been defined for the description of the system structure itself (System Structure Definition – SSD, file extension .ssd), of parameter sets (System Structure Parameter Values – SSV) and their mapping to system/component parameters (System Structure Parameter Mapping – SSM). Additionally the System Structure Package format (SSP, file extension .ssp) is defined, which constitutes a ZIP-archive that packages together a set of system structure definitions and any referenced parameter sets, mappings, components and sub-systems into one easily handled and transferable unit.

A SSP must contain at least one SSD file, but can contain multiple such files at top-level, which give the ability to package multiple variants of a system into one SSP, allowing the importing user/tool the selection of which variant to process. This enables the efficient exchange of systems/sub-systems with varying system topology, e.g. for vehicle models with different propulsion systems and architectures, while being able to reuse commonly shared resources like sub-systems, FMUs, or parameter sets.

The SSD file defines the structure of a system: Its external interface (if any), i.e. the system input, output and parameter connectors as exposed to the outside, and the internal structure, including instantiated components, like FMUs or referenced external systems, subsystems, as well as connections between

components and between components and the external interface.

For each component any referenced inputs, outputs and parameters are specified as connectors as well. Connections between connectors that are physical quantities will perform unit conversions by default. Connections can also apply linear transformations (for continuous quantities) or mapping transformations (for discrete quantities) in order to adjust values between components as needed.

The system description also assigns parameter sets (SSV) to components or complete (sub-)systems, either with a natural 1:1 mapping or by specifying explicit parameter mappings in the SSM format. See Figure 6 for a simplified overview of the data model behind the XML schema and Figure 11 for a simple example file.

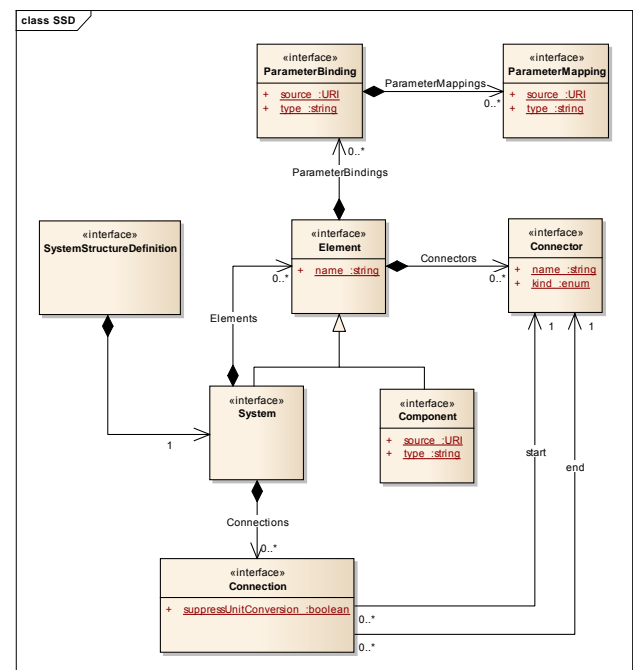


Figure 6. Simplified Class Diagram for SSD Schema.

In order to support exchange of system structure between tools that offer a graphic view of a system, optional geometric information for systems, components, connectors and connections is supported.

The SSV format defines a parameter set, consisting of a set of parameter definitions, including parameter values and related meta-information (like data type, physical unit), as necessary to aid in the exchange of parameter sets and their use in parametrizing systems and components. A core set of meta-information is likely to be included in the final standard with extension mechanisms to support the exchange of user-specific meta-information as needed. Parameter sets in the SSV format can be contained directly in the relevant parameter binding element of the SSD file or referenced as an external .ssv file.

The SSM format, as illustrated in Figure 7, defines a mapping between the parameters in a parameter set and

the parameters of a component or system (potentially including subsystems and components) by mapping the names of parameters between the two namespaces and optionally providing further transformations on parameter values, like mapping of enumerations or linear transformations of continuous values. Like the SSV format this mapping can be contained within an SSD file or referenced as an external .ssm file.

All formats offer easy extensibility for specific tool or user needs through optional tool- or usage-specific annotations on all modeling elements. This also allows the simple addition of layered standards on top of the current formats.

References between SSD files and related resources, like components, parameter sets or parameter mappings are implemented as (relative) URIs. This allows the integration of these formats into larger resource management systems like PLM systems, so that e.g. SSD and/or referenced parameter sets or component FMUs can be located via HTTPS or PLM-specific URI schemes, in addition to the default file-based access mechanisms.

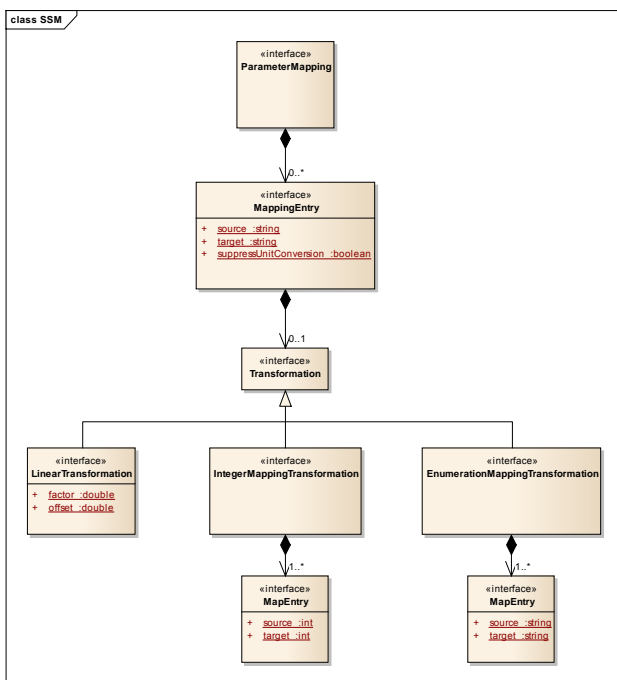


Figure 7. Simplified Class Diagram for SSM Schema.

Currently design discussions are on-going on the support of additional connection constructs, like signal dictionaries or bus-like connections, which aid in the maintenance of component and system interconnections with many, frequently changing signals, like those employed by bus communication mechanisms between controller models. It is expected that the initial release of the SSP standard will include such mechanisms.

While the work in the SSP project was initially focused on FMU-based systems, the SSD and

SSV/SSM formats are intentionally also suitable for describing systems containing other component types, like models or controller code, if relevant definitions are implemented for these component types.

4 Hardware-in-the-Loop Simulation

In order to cope with the growing complexity of modern electronic control units (ECUs), Model-Based Design (MBD) is used throughout the embedded software development process. The result is an increasing number of models designed for various purposes. During the MBD process different methods are applied to test the software of an ECU. In early stages PC-based model-in-the-loop (MIL) and software-in-the-loop (SIL) simulations are commonly used to validate the software. Additionally hardware-in-the-loop (HIL) simulation based testing is applied as the tried-and-tested method for function, component, integration and network tests of an entire system. HIL simulation is an integral part of the development process of many OEMs and suppliers across different industries. Due to the inclusion of real hardware in the test setup, HIL test systems have special requirements that do not allow the same free choice of simulation methods as for MIL/SIL use cases. Usually specialized HIL simulation systems with optimized hardware and real-time operating systems (e.g., QNX, Linux-RT) are necessary. These systems have to meet real-time requirements and handle the system dynamics and timing of the ECU computation timing loops. Common HIL applications typically require hard real-time, fixed-step solvers with sampling times of 1ms or less.

FMUs for co-simulation are a good basis for the tool- and platform-independent exchange of simulation models in HIL environments. The lean co-simulation interface reduces possible compatibility issues in a tool chain that includes various FMI supporting tools. Moreover, it systematically separates the FMU functions from the tool functions. This separation enables efficient FMU internal implementations of e.g. tunable parameter support and internal multi-rate subsampling. These co-simulation FMUs can transport verified combinations of solver and model code. Additionally the communication point concept can separate internal solver steps from external communication steps. FMUs may include ANSI-C source code, which is important for platform-independent reuse, but often conflicts with modelers' and tool vendors' interest in protecting their IP. Exported FMUs therefore often only contain precompiled binaries and are consequently limited to specific pre-selected target platforms.

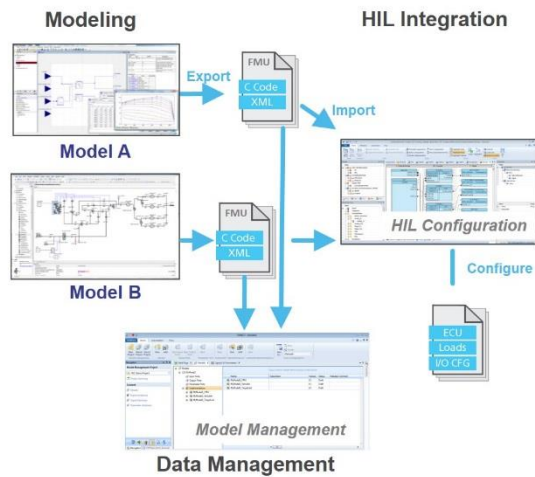


Figure 8. Integration of FMUs for HIL Testing.

Today single FMUs are imported into HIL configuration tools to integrate them with other FMUs, Simulink-based models, Virtual-ECUs or real ECUs (Figure 8). HIL simulation tests with real-time capable FMUs can rely on the full functionality of a tool chain including test automation and visualization. Additionally newer Model-Based Design Test and Data Management environments provide capabilities for managing model compositions, handling variants of models and systems-under-test and managing the parameter and signal interfaces of the different model systems. These functions are necessary in order to optimize the usage of models and associated data assets throughout the lifecycle of development and validation. Especially for managing complex simulation scenarios resulting from the use of models from various modeling environments such capability is crucial.

Environment models are often developed and specially designed for certain use cases. However, there is an increasing desire to reuse these models to provide proven, consistent solutions for the validation of controller models in different projects and development stages (e.g., for virtual validation and HIL simulations). A reuse of models increases productivity and saves time by eliminating the duplication of design efforts. The environment models that are exchanged – e.g., based on FMI – need to be suitable for all the intended MIL/SIL/HIL simulation use cases. Modular design of models and particularly clear identification of interfaces pertaining to real or simulated components, are necessary to allow an exchange of simulated components by a real hardware component at various points in the development and testing processes. In co-simulation scenarios, a model structure should be chosen that separates the overall model into weakly coupled model parts that can be computed concurrently and are insensitive to input delays due to co-simulation effects.

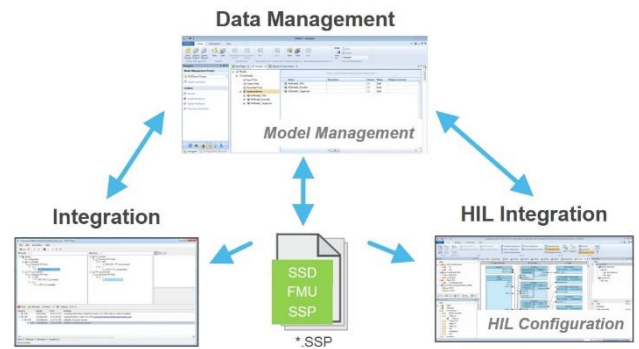


Figure 9. Potential ways to exchange the System Structure Description.

Once a reasonable model structure is designed there is no standardized way so far to exchange it among tools from different vendors especially if no overall integration model exists. The SSP approach allows to share a standardized system structure description between data management, integration and configuration tools for SIL, MIL and HIL scenarios (Figure 9). Hence, the SSP interchange format helps to improve the consistent simulation and interchange of complex models in the MBD process.

5 Prototypes

The specifications of SSD are investigated with some prototype tools assuming various co-simulation environments.

5.1 Integration Tool

Model.CONNECT™, a product of AVL List GmbH, is a tool to set up and execute system simulation models which are composed of subsystem and component models from multiple model authoring environments. Models can be integrated based on standardized interfaces (FMI) as well as based on specific interfaces to a wide range of well-known simulation software.

In order to validate the SSP specification, we implemented a prototype plug-in for the tool that supports the export and import of system configurations.

During the prototype development we particularly explored the capabilities of SSP to pack variants of system configurations into one archive. We found that the very basic variant support in SSP could be mapped to/from the sophisticated variant management capabilities in the tool, which is designed to describe both different configurations of the system under investigation as well as different testing scenarios and testing environments. It is important to mention, that the variant handling in SSP is deliberately and by design not expressive enough to support loss-less export-import roundtrips (e.g. Model.CONNECT™ → SSP → Model.CONNECT™) with respect to variant management. We plan to address this in future by

enriching the export/import plug-in based on tool-specific annotations in the SSP.

This prototype SSP export/import plug-in was also used to test the specifications of SSP with regards to the graphical representation (as 2D block model) of system configurations. Also here we did not encounter any major issues mapping between SSP and the tool-specific geometry handling. This is again a result of SSP design principle to keep the specification as simple as possible: SSP allows to transfer component and connector positions as well as connection waypoints. Thus, SSP allows to re-construct the main aspects of a layout, but it makes no attempt towards a pixel-by-pixel identical rendering of systems in the exporting and importing systems.

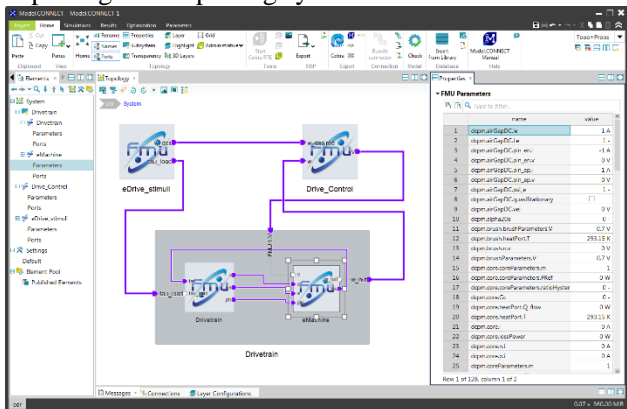


Figure 10 eDrive example in Model.CONNECT™. The subsystem “Drivetrain” is displayed in transparent mode to see its internal structure.

Future work on the plug-in will be focusing on the support of parameter values and mappings. Workflow-wise, we will explore using SSP to transfer system information from SysML-based MBSE tools to Model.CONNECT™.

5.2 Co-Simulation Browser

FMI has become a common model exchange and co-simulation standard. However the master-level runtime verification and validation of the virtual system made of many slave models are still difficult for FMI users. The integration of multi-domain expertise is required to analyze the multi-FMU complexity and the large scale virtual system simulation results. In order to facilitate the system-level FMU user collaboration, a light weight FMI/SSP Co-Simulation browser is prototyped. This co-simulation browser includes the simulation player and the parser of SSP defined System Structure XML such as in Figure 11 .

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <ssd:StructureDescription xmlns:ssd="https://www.maf.net/od/Std/ssd/StructureDescriptionDraft"
3 xmlns:sxi="https://www.w3.org/2001/XMLSchema-instance" version="Draft1010721" xsi:schemaLocation="https://www.maf.net/od/Std/ssd/
4 <ssd:System name="eDrive" description="eDrive ssi with geometry">
5 <ssd:SystemGeometry xs="0.0" y="0.0" width="0.0" height="0.0" canvasWidth="2.76" canvasHeight="2.41"/>
6 <ssd:Connectors>
7 <ssd:Connector name="ExternalOut" causality="output"/>
8 </ssd:Connectors>
9 <ssd:System name="Drivetrain" description="">
10 <ssd:SystemGeometry xs="0.899" y="1.18" width="2.26" height="1.28" canvasWidth="2.07" canvasHeight="0.88"/>
11 </ssd:System>
12 <ssd:Connector name="FMU_1_V" causality="input"/>
13 <ssd:Connector name="w_out" causality="output"/>
14 </ssd:Connectors>
15 <ssd:Component name="Drivetrain" type="application/x-fmi-shared-library" source="resource/Drive_Drivetrain.fmu">
16 <ssd:ComponentGeometry xs="0.0" y="0.0" width="0.888" height="0.888"/>
17 </ssd:Component>
18 <ssd:Connector name="tau_load" causality="input"/>
19 <ssd:Connector name="w_desired" causality="input"/>
20 <ssd:Connector name="w" causality="input"/>
21 <ssd:Connector name="phi" causality="input"/>
22 <ssd:Connector name="w_out" causality="output"/>
23 <ssd:Connector name="tau_load" causality="output"/>
24 </ssd:Connectors>
25 <ssd:Component name="eMachine" type="application/x-fmi-shared-library" source="resource/Drive_eMachine.fmu">
26 <ssd:ComponentGeometry xs="1.45" y="0.0372" width="0.62" height="0.62"/>
27 </ssd:Component>
28 <ssd:Connector name="y" causality="input"/>
29 <ssd:Connector name="y" causality="input"/>
30 <ssd:Connector name="phi" causality="input"/>
31 <ssd:Connector name="w" causality="input"/>
32 <ssd:Connector name="w" causality="input"/>
33 <ssd:Connector name="w" causality="input"/>
34 <ssd:Connector name="w" causality="input"/>
35 <ssd:Connector name="w" causality="input"/>
36 <ssd:Connector name="w" causality="input"/>
37 <ssd:Connector name="w" causality="input"/>
38 <ssd:Connector name="w" causality="input"/>
39 <ssd:Connector name="w" causality="input"/>
40 <ssd:Connector name="w" causality="input"/>
41 <ssd:Connector name="w" causality="input"/>
42 <ssd:Connector name="w" causality="input"/>
43 <ssd:Connector name="w" causality="input"/>
44 <ssd:Connector name="w" causality="input"/>
45 <ssd:Connector name="w" causality="input"/>
46 <ssd:Connector name="w" causality="input"/>
47 <ssd:Connector name="w" causality="input"/>
48 <ssd:Connector name="w" causality="input"/>
49 <ssd:Connector name="w" causality="input"/>
50 <ssd:Connector name="w" causality="input"/>
51 <ssd:Connector name="w" causality="input"/>
52 <ssd:Connector name="w" causality="input"/>
53 <ssd:Connector name="w" causality="input"/>
54 <ssd:Connector name="w" causality="input"/>
55 <ssd:Connector name="w" causality="input"/>
56 <ssd:Connector name="w" causality="input"/>
57 <ssd:Connector name="w" causality="input"/>
58 <ssd:Connector name="w" causality="input"/>
59 <ssd:Connector name="w" causality="input"/>
60 <ssd:Connector name="w" causality="input"/>
61 <ssd:Connector name="w" causality="input"/>
62 <ssd:Connector name="w" causality="input"/>
63 <ssd:Connector name="w" causality="input"/>
64 <ssd:Connector name="w" causality="input"/>
65 <ssd:Connector name="w" causality="input"/>
66 <ssd:Connector name="w" causality="input"/>
67 <ssd:Connector name="w" causality="input"/>
68 <ssd:Connector name="w" causality="input"/>
69 <ssd:Connector name="w" causality="input"/>
70 <ssd:Connector name="w" causality="input"/>
71 <ssd:Connector name="w" causality="input"/>
72 <ssd:Connector name="w" causality="input"/>
73 <ssd:Connector name="w" causality="input"/>
74 <ssd:Connector name="w" causality="input"/>
75 <ssd:Connector name="w" causality="input"/>
76 <ssd:Connector name="w" causality="input"/>
77 <ssd:Connector name="w" causality="input"/>
78 <ssd:Connector name="w" causality="input"/>
79 <ssd:Connector name="w" causality="input"/>
80 <ssd:Connector name="w" causality="input"/>
81 <ssd:Connector name="w" causality="input"/>
82 <ssd:Connector name="w" causality="input"/>
83 <ssd:Connector name="w" causality="input"/>
84 <ssd:Connector name="w" causality="input"/>
85 <ssd:Connector name="w" causality="input"/>
86 <ssd:Connector name="w" causality="input"/>
87 <ssd:Connector name="w" causality="input"/>
88 <ssd:Connector name="w" causality="input"/>
89 <ssd:Connector name="w" causality="input"/>
90 <ssd:Connector name="w" causality="input"/>
91 <ssd:Connector name="w" causality="input"/>
92 <ssd:Connector name="w" causality="input"/>
93 </ssd:StructureDescription>

```

Figure 11. Example of eDrive.ssd.

This light-weight co-simulation player is easy to extend for the pursuit of ‘X-In-the-Loop’ methodology. The ‘X’ stands for ‘model’, ‘software’, ‘hardware’, and ‘human’. By connecting various abstract models and devices, our FMI/SSP virtual system would be widely expanded. With the flash-based integration of co-simulation browser, users can easily access mobile simulations and visualizations of FMI models in the cloud environment. The FMI co-simulation slaves would be executed on network distributed servers. The control of co-simulation master can be included through the brand-new smart devices with some intuitive multi-touch operations.

The co-simulation browser was applied to check the project example of eDrive.ssd test case. The XML parser reads the FMU connections and interprets the FMI-compliant simulation parameters. The layout of FMU slaves is automatically adjusted in a circular configuration to show the complex connections as compact as possible (see r.h.s in Figure 12). Before starting the system-level simulation, the co-simulation browser can run the unit tests of each FMU with manually added test I/O functions to fit into the FMI input ports.

The system parameter dataset/database could also be distributed on the network servers. There is a process integration tool Optimus® that can export a parameter database wrapped as a portable FMU. For example, the parameters compiled as ResponseSurfaceModel.fmu is

easily imported to our co-simulation test bed, namely, ‘FMI/SSP Stage’.

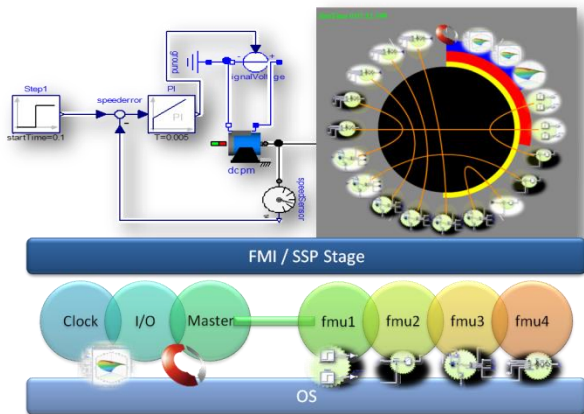


Figure 12. Connection UI on FMI/SSP Stage.

The FMI/SSP stage master manages the simulation context which contains time integration scheme, simulation clock, event handling, and parameter input/output functionalities. In a master-slave co-simulation, the master algorithms can define the quality and performance of the co-simulation. The clock functions setup the master timestep sizes to optimize the simulation efficiency. The way how to share such simulation parameters would be discussed further in the ongoing SSP project.

When we complete the definition of System Structures and System Parameters on this SSP test bed, we can extend the co-simulation environment, to that of ‘Co-Optimization’. A FMI co-optimization case with the sequential tool chain process is reported (Batteh *et al.*, 2015). The Co-Optimization stands for the paradigm such that every model is gathering as FMU modules on the virtual system stage of SSP. The simulation result could be reflected onto the system parameter set such as Response Surface Model to refine the next co-simulation trial in the optimization or calibration cycle. The SSP-based environment will enhance the co-optimization paradigm and speed up the parameter exploration in virtual systems.

5.3 FMI Bench

FMI Bench is a product of PMSF IT Consulting that provides a workbench for manipulating and integrating FMUs into assembled systems which can then be exported as new complex FMUs for use in other simulations or complete executable simulations for stand-alone use.

As part of the work on SSP a prototypical implementation of the SSP drafts has been undertaken, allowing the importation and exportation of complete SSP packages from FMI Bench.

Special consideration was placed on the ability of SSP to describe systems with external interfaces that

would allow exportation as complex FMUs so that systems packaged as SSPs could be re-exported as complex FMUs for use as subsystems in other simulation systems while still making use of the FMI Bench features, such as automatic multi-threading of complex FMUs or remote FMU execution.

The experiences with the SSP drafts showed that this usage was indeed possible, as seen in Figure 13, showing both the imported eDrive example SSP in the upper window and the generated native FMI Bench project, which allows direct code-generation, in the lower window.

Future work is intended to track the progress of the SSP project work in the areas of parameters and complex communication primitives, while integrating SSP/SSD support into the core product.

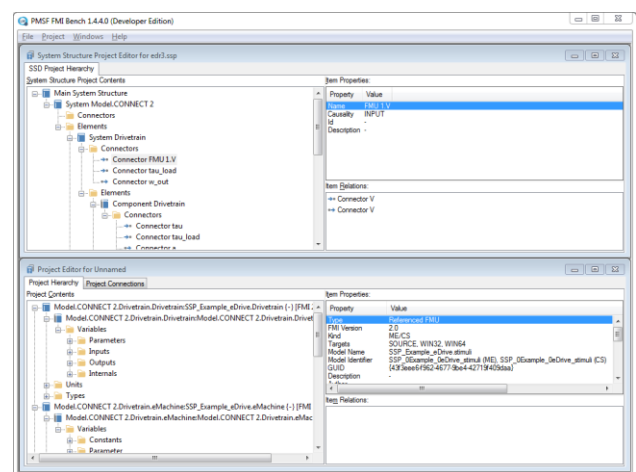


Figure 13. eDrive example in the FMI Bench SSP prototype showing imported SSP and derived native FMI Bench project.

6 Outlook

As shown SSP is a valid approach to make the work with FMUs and their parameterization easier especially when complex systems with several components have to be simulated and interchanged. The way to define system structures is derived from daily work in industry so it can be easily adapted to existing working processes. The close cooperation with the FMI project group guarantees, that both standards work well together, even if SSP is not solely restricted to working with FMUs as components.

In the first stage the project group concentrated on defining system structures. That work is currently mature enough to enable first practical evaluations of it. Parameterization is the second stage to go into in more detail in 2016. The overall goal is to have a first version of the standard rather early to be able to get experience quickly by evaluating it with running prototypes which are developed in parallel. Therefore it is very appreciated if many tool vendors and key users contribute to the project. If you are interested in more information or if you want be get involved in our work,

feel free to contact us: map-ssp@modelica.org
[Maybe add contact information to SSP working group
for vendors/users to join].

References

- Blochwitz T., Otter M., Arnold M., Bausch C., Elmqvist H., Junghanns A., Mauß J., Monteiro M., Neidhold T., Neumerkel D., Olßon H., Peetz J.-V., Wolf S., Clauß C. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *Proceedings of the 8th International Modelica Conference*, pp.105-114, Dresden, Germany, 2011 doi:10.3384/ecp11063105.
- Batteh J., Gohl J., Pitchaikani A., Duggan A., Fateh N. Automated Deployment of Modelica Models in Excel via Functional Mockup Interface and Integration with modeFRONTIER. *Proceedings of the 11th International Modelica Conference*, pp.171-180, Versailles, France, 2015 doi:10.3384/ecp15118171.

ADAS Virtual Prototyping using Modelica and Unity Co-simulation via OpenMETA

Masahiro Yamaura¹ Nikos Arechiga¹ Shinichi Shiraishi¹

Scott Eisele² Joseph Hite² Sandeep Neema² Jason Scott² Theodore Bapty²

¹Toyota InfoTechnology Center, U.S.A., Inc., U.S.A., {myamaura, narechiga, sshiraishi}@us.toyota-itc.com

²Institute for Software-Integrated Systems, Vanderbilt University, U.S.A., {eiseles, jhite7, sandeep, jscott, bapty}@isis.vanderbilt.edu

Abstract

Automotive control systems, such as modern Advanced Driver Assistance Systems (ADAS), are becoming more complex and prevalent in the automotive industry. Therefore, a highly-efficient design and evaluation methodology for automotive control system development is required. In this paper, we propose a closed-loop simulation framework that improves ADAS design and evaluation. The proposed simulation framework consists of four tools: Dymola, Simulink, OpenMETA and Unity 3D game engine. Dymola simulates vehicle dynamics models written in Modelica. Simulink is used for vehicle control software modeling. OpenMETA provides horizontal integration between design tools. OpenMETA also has the capability to improve design efficiency through the use of PET (Parametric Exploration Tool) and DSE (Design Space Exploration) tools. Unity provides the key functionality to enable interactive, or closed-loop ADAS simulation, which contains sensor models for ADAS, road environment models and provides visualization.

Keywords: ADAS, Efficient Design, Game Engine, Modelica, Simulink

1 Introduction

The number of installations of Advanced Driver Assistance Systems (ADAS) is rapidly growing in the automotive industry. In the case of Toyota cars, Toyota Safety Sense, which is a type of ADAS package, will be available in most passenger cars released by Toyota Motor Corporation in Japan, North America, and Europe by the end of 2017 (Toyota Motor Corporation, 2014). This emerging market of ADAS poses difficult system design problems. That is, we cannot use a traditional development methodology that considers only a target vehicle. We need to derive a new methodology which allows us to take its environment into account, e.g., road, other vehicles, pedestrians, etc. With the announcement that the majority of cars will contain an ADAS, it is apparent that the design space of future cars will be vast. Moreover, the complexity of

these systems is also increasing along with their extended features, e.g., communication with other vehicles, cooperation with a navigation system, etc.

The above problems imply that a highly-efficient design and evaluation methodology for ADAS development is required. Van Waterschoot and van der Voort have recognized this same need for efficient design when looking at ADAS as a human factors problem (van Waterschoot *et al*, 2009). Simulation-based verification and validation can be a key technology in such a methodology as shown by Gruyer *et al*. (Gruyer *et al*, 2011). More precisely, closed-loop simulation including vehicle dynamics and road environments is essential.

Our work addresses the need for closed-loop simulation by using Simulink to model software components and Modelica to model physics components. Simulink is currently the state-of-the-art tool for developing and analyzing automotive software models. Modelica is well suited to describe and simulate physics which includes vehicle dynamics. However, the task of describing complex conditions around the vehicle, such as traffic events, pedestrian activity and weather activity is complex and results in simulations not amenable to interactive simulation. Our work uses Unity to model complex environmental conditions. Unity is a video game development tool which is well suited to describe complex road situations. Our proposed framework consists of a co-simulation-based solution for ADAS development challenges by using OpenMETA to integrate Simulink, Modelica and Unity, and provide some features which aid in the design of complex systems.

Generally, game engines provide sophisticated virtual reality environments, and can be used to allow users to collaborate. These game engine advantages support valuable features in ADAS development such as gamified and crowd-sourced vehicle testing, and virtual dealership, which are described below.

Section 2 provides a background on existing design tools. Section 3 describes our tool framework. Section 4 describes our case study and Section 5 presents our conclusions and possible directions for future work.

2 Background

2.1 Existing Tools

Modelica is a multi-physics, multi-domain, acausal modeling language. Dymola, developed by Dassault Systèmes, is a powerful tool as an editor and simulator of Modelica models (Dassault Systèmes, 2015).

Simulink is a graphical modeling tool produced by MathWorks. It provides a graphical modeling editor, a customizable set of block libraries, and solvers for simulations (Mathworks, 2015). Designers can edit models by deploying blocks from the libraries and adding causal connections between blocks. Simulink is widely used in the automotive industry for vehicle control system software design, such as ADAS systems, engine control systems, transmission control systems, etc. This is the reason why we selected Simulink for control system software modeling.

The OpenMETA toolchain was developed by Vanderbilt University in conjunction with the Adaptive Vehicle Make program of DARPA (Sztipanovits *et al*, 2014; Sztipanovits *et al*, 2015). OpenMETA is a tool infrastructure with the goal of enabling development of cyber-physical systems. This is accomplished by providing horizontal integration between external software tools. A model in OpenMETA references component models which exist in external tools such as Dymola, Creo, or ADAMS. In order to interface with an external tool, an interpreter is created for OpenMETA which transforms the model into a format the external tool can use. Typically OpenMETA is setting parameters which are then used to provide inputs into a detailed model that exists in the external tool. Once the interface is in place, then any parametric changes made to the model in OpenMETA will also appear in the external tool. Additionally, if the internals of any tool specific model are changed, as long as the interface remains, the models will function as they

would if the model had been generated entirely in the external tool.

In the case of Simulink integration, a Simulink model is wrapped as a C library which is referenced in OpenMETA. The representation of this library in OpenMETA includes the interfaces exposed in the original Simulink model.

Additionally, OpenMETA has other features for highly-efficient design, such as the Parametric Exploration Tool (PET) and Design Space Exploration (DSE). Automotive control software generally has many parameters that should be calibrated in the development phase. PET enables a designer to explore the interactions between parameters in an automated fashion and then displays the results in a way which allows the designer to make tradeoffs and select the parameter set which is most suited to the design criteria. Since OpenMETA has this feature, there is interest in applying OpenMETA to ADAS development. In this paper, we focus more on utilizing the PET in OpenMETA to calibrate system parameters with the simulation models.

A growing challenge with the design of complex systems is that each system may be designed using a variety of architectures and each architecture is comprised of components which also have variations as shown in Figure 1. This information is represented in OpenMETA through the use of design spaces. A design space is a set of design containers which contain a family of components which share a common interface. Alternate design architectures are similarly represented except rather than alternative components there are alternative architectures where the architecture is in turn made up of components. This results in an explosion in the number of configurations that could be considered. OpenMETA provides a tool, which is called DSE, to list all of the candidate design configurations and simulate those that meet the static

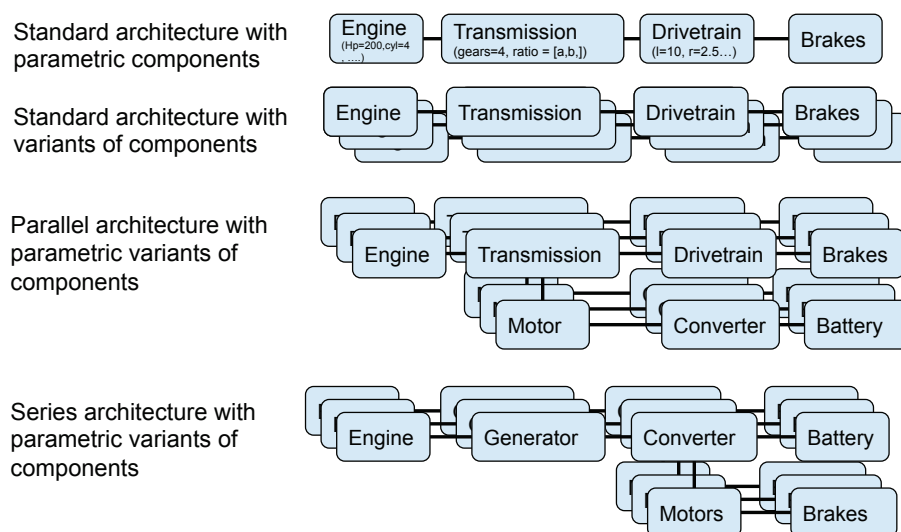


Figure 1. An example of multiple architectures with component alternatives.

constraints which are specified by the designer. OpenMETA also provides a visualization tool that allows a designer to compare the designs which meet system requirements from DSE simulation results.

Although existing tools mentioned above allow us to build a vehicle model with cyber components and physical components, the proposed integration with Unity allows us to provide a far richer, virtual reality testing environment, including traffic, pedestrians, sensor models and effects of weather and the environment, such as those due to fog, heavy rain, and icy road conditions. These effects are important, because they can potentially compromise the correct functionality of ADAS systems.

2.2 Unity

Instead of commercial tools of ADAS simulation, e.g., PreScan, CarMaker, etc. we use Unity (Unity Technologies, 2015), which is a 3D game engine for video game development, for road environment modeling, and sensor modeling. The primary reason that we selected a game engine was to leverage the capability to involve users from all over the world in our ADAS evaluation tests. This concept is referred to as “gamified and crowd-sourced virtual testing”. Generally, many situations would need to be considered in order to evaluate an ADAS implementation such as driver input, other vehicle behaviors, road geometry and so on. By using a game environment populated by human and virtual users, the ADAS software can be tested more extensively than with traditional static test scenarios.

In addition to providing a multi-user platform, Unity has a user friendly GUI editor, 3D physics engine, animation engine, 3D model import, and scripting in C# or JavaScript. These features help a simulation designer model cities which contain road models and

other dynamic objects, such as vehicles, pedestrians, motorcycles, and bicycles. Although other 3D game engines are available, Unity was selected because of its large asset library, multiplatform support, and large community support.. Many assets are available through the Unity Asset Store which accelerates development and would not be available in other tools. For example, road editors, vehicle physics, and car traffic simulators are available as ready-to-use assets with Unity.

3 Methodology

3.1 Simulation Architecture

We integrated the four tools for the simulation framework: Dymola, Simulink, OpenMETA and Unity. The architecture is shown in Figure 2. OpenMETA integrates the Dymola and Simulink models. The Dymola model has some vehicle physical components, including the engine, transmission, driveshaft, and differential from Vehicle Dynamics Library. The Vehicle Dynamics Library is developed by Modelon (Modelon, 2014). Other components, such as wheels, are modeled in Unity. The interfaces between Unity and Dymola are wheel torques and wheel rotation speeds. The Dymola simulation sends wheel torques to the Unity simulation, and the Unity simulation sends wheel rotation speeds to Dymola. These interfaces are implemented by UDP socket communication. The Modelica Device Drivers (Bernhard et al, 2015) library provides UDP communication blocks used in Dymola.

The Unity model also has road environments, other vehicle models, and sensor models for ADAS systems. The sensor data in Unity is also sent to the Simulink controller via UDP. The list of UDP interfaces is shown in Table 1. In the Unity model, some assets from Unity Asset Store were used for modeling. For example, EasyRoad3D Pro is used for road building,

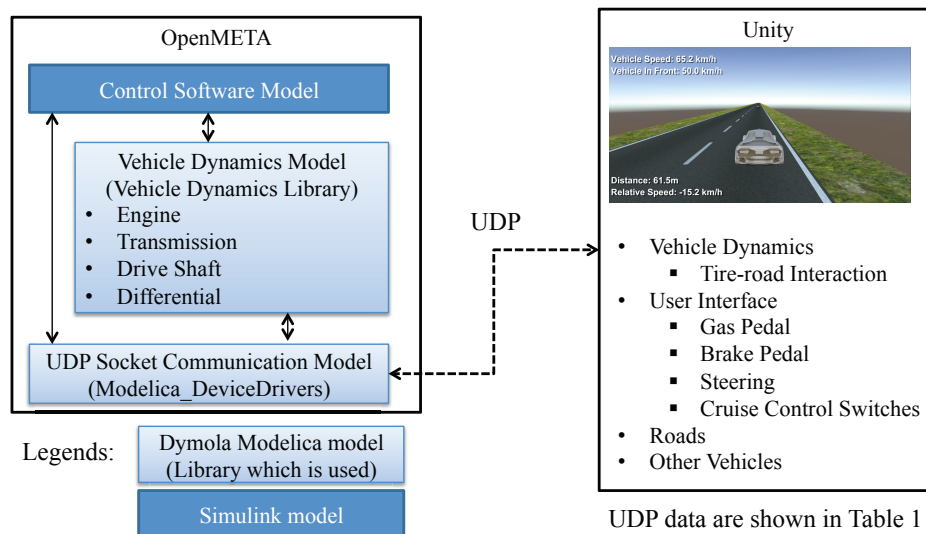


Figure 2. Simulation architecture

and the Urban Construction Pack for building city models.

Table 1. UDP Communication data list

Direction	Data
Dymola → Unity	Wheel torques
	Vehicle status (ADAS status, etc.)
Unity → Dymola	Wheel rotation speeds
	Sensor data (Millimeter wave radar, etc.)
	User operation (Pedals, steering, etc.)

4 Case Study

In this paper, we show simulation results with PET in OpenMETA to illustrate the advantages of this simulation framework. The PET is a highly-efficient methodology for calibrating control software parameters.

As the first case study for this simulation toolchain with PET, we decided to model the Adaptive Cruise Control (ACC) system. The ACC is one of the ADAS systems. This system helps mitigate driver fatigue by assisting accelerator operations. Toyota’s ACC system has 2 modes: constant speed control mode and vehicle-to-vehicle distance control mode. Constant speed control mode is the same as a conventional cruise control system. While this mode is active the system works to maintain a target velocity. Vehicle-to-vehicle distance mode works with sensors, such as millimeter wave radar sensor that detects the presence of lead vehicles. Upon detecting a vehicle, the ACC adjusts the speed in order to maintain a safe following distance. The control flow is shown in Figure 3. The driver can choose the following distance: Long, Middle or Short. Actual distances are determined based on the velocity of the vehicle.

For the ACC case study in this paper, we defined following scenario as shown in Figure 4. There is an ACC installed in host vehicle A, which has initial speed of v_0 and a lead vehicle B, which has constant speed v_{front} . Vehicle A is initialized with the ACC active and velocity set point v_{set} . The initial distance between two vehicles was set so that vehicle A accelerates to the speed v_{set} . After some time vehicle A senses vehicle B and transitions to vehicle following mode. Vehicle A decelerates to maintain the distance between two vehicles d_{set} . In this paper, we used values in Table 2.

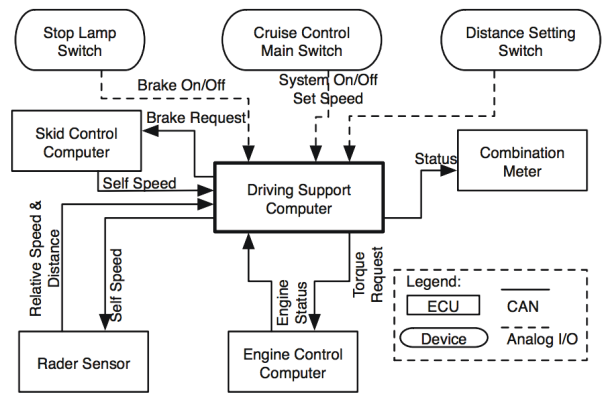


Figure 3. Adaptive cruise control diagram (Shiraishi *et al*, 2011)

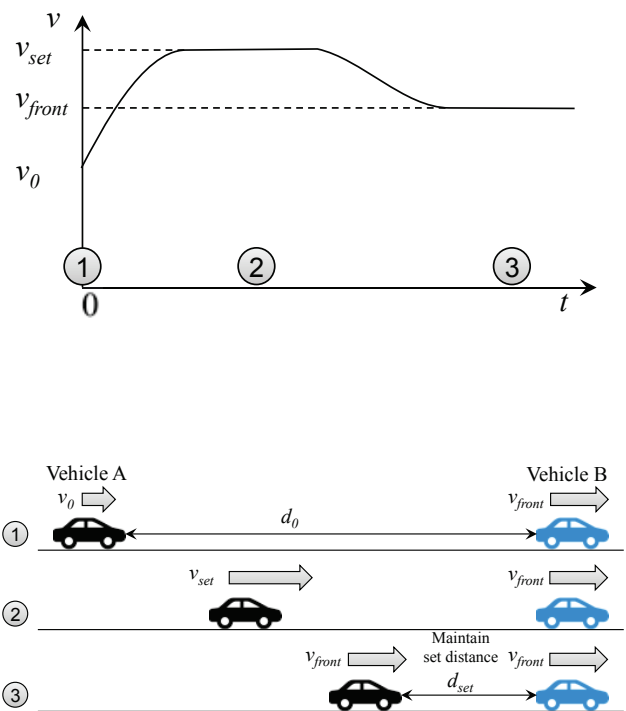


Figure 4. Case study of ACC system

Table 2. Parameters used in the simulation

Parameters	Values
v_0	0 km/h
v_{set}	70 km/h
v_{front}	50 km/h
d_0	150 m
d_{set}	40m

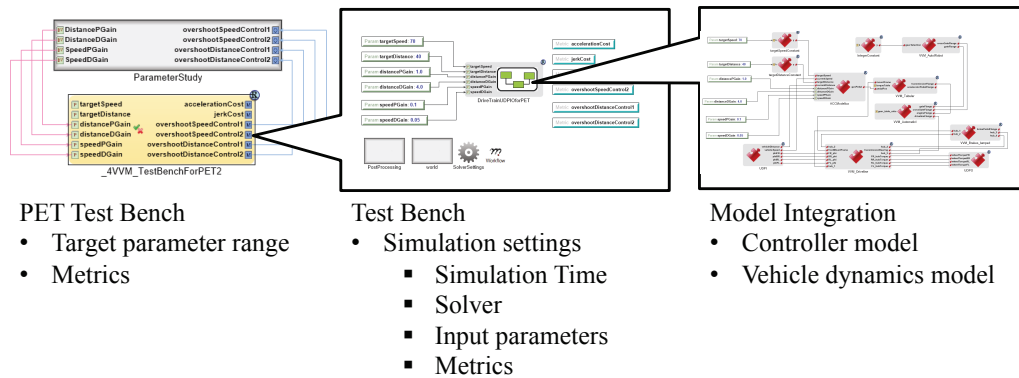


Figure 5. OpenMETA models for PET

4.1 Controller Model

The controller¹ developed for this experiment is bimodal. The first mode utilizes vehicle-to-vehicle distance as the desired value and the second mode uses a velocity set point as the desired value. Both controllers were implemented using PD controllers. The controller behavior is to maintain a safe following distance if there is a lead vehicle. Otherwise, it keeps vehicle speed to the set speed. The PD gains are calibrated by using PET which is discussed below. The ACC is always activated during the simulations so that the simulation does not need any driver inputs.

4.2 Unity Model

For the ACC case study, we added a long straight road, two vehicles and a millimeter wave radar sensor model. The sensor model obtains a distance between two vehicles and their relative speed. These signals are communicated to Dymola via UDP and are used as inputs to the controller. The sensor model was built by using “Ray” class in Unity scripting C# API which is often used in shooting games.

4.3 Parametric Exploration Tool and Test Bench

To run the simulation from OpenMETA with PET, designers have to set up some Dymola parameters, such as simulation time, solver, etc. in the OpenMETA Test Bench as shown in Figure 5. Additionally, metrics, which are used for evaluations of the models, need to be described in the Test Bench. The metrics in the ACC case study are velocity and gap distance overshoot. Settling time and rise time are also major metrics of this kind of system, but have not been included in this case study.

Next step toward parameter design is building a PET test bench. PD gains, which are speed control P, D gains and distance control P, D gains, are target parameters to be calibrated. In the PET test bench,

designers need to assign parameters, their ranges, and testbench outputs or metrics. The PET test bench is also shown in Figure 5.

4.4 Simulation Results

We ran the PET of the ACC case study with parameters which are shown in Table 2. The result of the PET simulation results can be visualized as a “Constraint Plot” in the OpenMETA dashboard, which is shown in Figure 6. The horizontal axis and vertical axis of this plot are the PD gains mentioned above. The plots show boundaries, which represent which combinations of parameters that meet the overshoot requirements. Thresholds are adjustable in the dashboard. The threshold values in Figure 6 are $overshoot < 0$.

A designer can find PD gains by clicking on a point in a plot. We picked gains which are close to the boundary which meets both overshoot requirements. The graphs in Figure 7 are Dymola simulation results using gains selected by examining the constraint plot shown in Figure 6. The upper graph in Figure 7 represents distance between two vehicles; the lower graph in Figure 7 represents the velocity of vehicle A. The red line represents the target value and the blue line represents the current value. This plot shows that there is no overshoot in either graph, demonstrating that the PET was useful in selecting design parameters.

¹ The ACC controller model in this paper is *not* a real ACC model

5 Conclusions

This paper described the methodology of integrating OpenMETA and Unity. The foundation of an integrated simulation framework, which includes PET for ADAS evaluation, has been built. As a result, designers may calibrate control software parameters more efficiently. Next steps include developing high quality and multi-fidelity models which would allow

for greater flexibility in the design process and developing additional case studies including the addition of driving scenarios such as: curves, intersections, etc. Additionally, it is also planned to consider driver-in-the-loop simulation which incorporate user input from Unity clients. These simulations would allow for some interesting applications. One such application is developing a

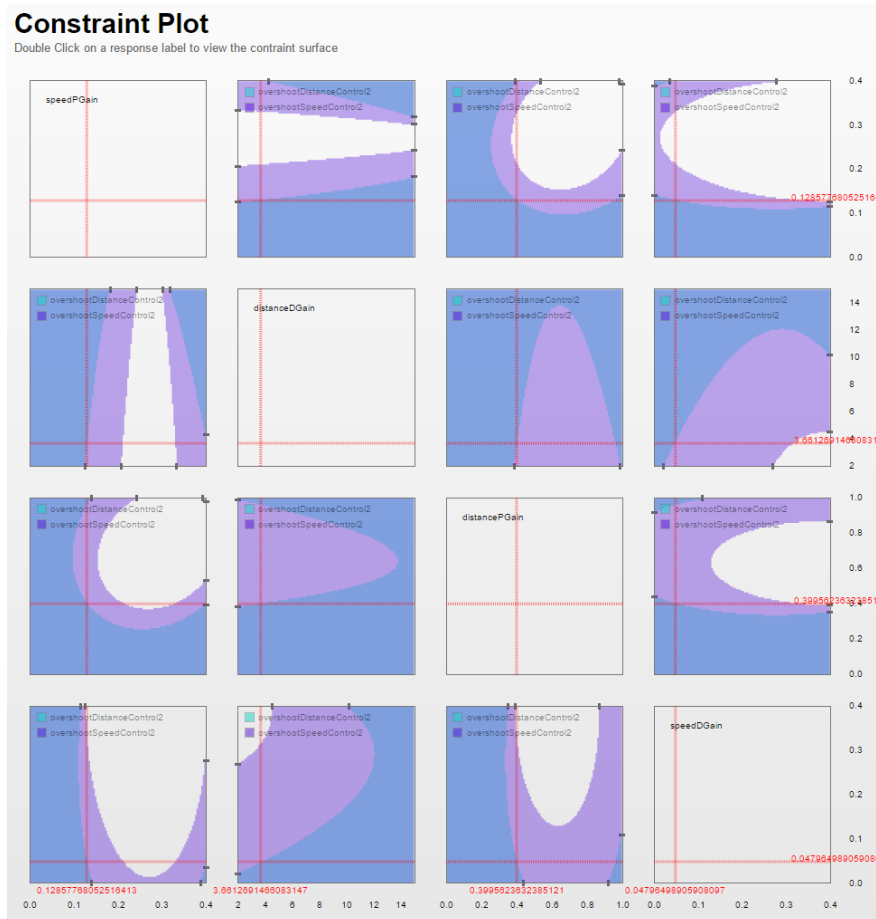


Figure 6. PET result: Constraint Plot

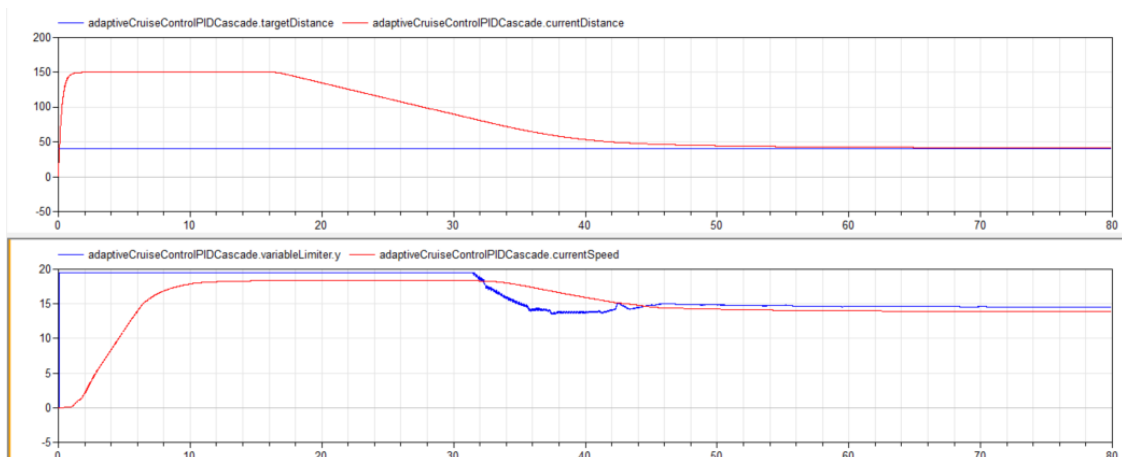


Figure 7. Waveform results of PET

virtual dealership concept in which customers participate in the design of their vehicles, and test-drive their creations. This test environment would also be used to crowd-source vehicle testing allowing for improvements to systems like ADAS and resulting in designs that have better performance and reliability.

References

- Bernhard Thiele, Tobias Bellmann, tbeu, Modelica_DeviceDrivers, 2015 URL: https://github.com/modelica/Modelica_DeviceDrivers
- Dassault Systèmes AB, Dymola 2015, 2015 URL: <http://www.3ds.com/products-services/catia/products/dymola>
- D. Gruyer, S. Glaser, S. Pechbert, R. Gallen, and N. Hautiere, Distributed Simulation Architecture for the Design of Cooperative ADAS”, *Presentation at First International Symposium on Future Active Safety Technology toward zerotraffic-accident*, September 2011.
- MathWorks, Simulink 8.6, 2015 URL: <http://jp.mathworks.com/products/simulink/>
- Modelon, Vehicle Dynamics Library 1.9, 2014 URL: <http://www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/>
- Shinichi Shiraishi and Mutsumi Abe. Automotive System Development Based on Collaborative Modeling Using Multiple ADLs. *Presentation at ESEC/FSE 2011 Industrial Track*, Sep. 2011.
- Janos Sztipanovits, Ted Bapty, Sandeep Neema, Larry Howard, and Ethan Jackson. OpenMETA: A Model- and Component-Based Design Tool Chain for Cyber-Physical Systems. *From Programs to Systems. The Systems perspective in Computing*, pp. 235-248, 2014. doi: 10.1007/978-3-642-54848-2_16
- Janos Sztipanovits, Ted Bapty, Sandeep Neema, Xenofon Koutsoukos and Jason Scott. The META Toolchain: Accomplishments and Open Challenges. *Vanderbilt University Institute for Software-Integrated Systems Technical Report*, 2015
- Toyota Motor Corporation (2014). “2014 Toyota Safety Technology Media Tour”, URL: http://www.toyota-global.com/innovation/safety_technology/media-tour/.
- Unity Technologies, Unity 5.3.0, 2015 URL: <https://unity3d.com/unity>
- Boris van Waterschoot and Mascha van der Voort. Implementing Human Factors within the Design Process of Advanced Driver Assistance Systems (ADAS) Engineering Psychology and Cognitive Ergonomics Vol. 5639, pp. 461-470, 2009. doi: 10.1007/978-3-642-02728-4_49

Active Elbow Joint Model

Hyung Yun Choi¹ Manyong Han¹ Whe-Ro Lee¹
Toru Matsui² Hisayoshi Matsuoka²

¹Mechanical System Design Engineering Dept., Hongik University, Korea, hychoi@hongik.ac.kr

²Integrated CAE and PLM Dept., Nissan Motor Co, Ltd., Japan

Abstract

Voluntary and reflexive muscle activation of human elbow joint is investigated by both subject tests and numerical simulations. A jerk loading is applied to extend the elbow joint with different muscle tensing and pre-recognition conditions. Inter- and Intra-subject variations of hand displacement are analyzed for an objective assessment of the active response at the elbow joint to the external perturbation. Both Modelica and finite element mesh models are developed using passive kinematic joint element and active torque which has PID close loop control. The simulation result from these two models are compared with test results and shows a good correlation.

Keywords: Digital human body model, Voluntary and reflexive muscle activation

1 Introduction

Digital human body models (DHBM) have been widely adopted in various CAE processes of vehicle design, e.g., car crash simulation for the prediction of injury risk and riding comfort simulation for the assessment of occupant discomfort. For most of such cases, the DHBM is in 3D finite element mesh shape so that it can mechanically interact with vehicle structures such as seat, safety belt and airbag. Thanks to efforts from many researchers, there is a significant advancement in human body modeling (www.ghbmc.com), e.g., mechanical behavior of biological tissue but the active human response with voluntary and reflexive muscle activation that affects occupant kinematics are still remaining as a great challenge.

Muscle tensing of bracing occupant produces larger axial forces, stress redistribution within bones, increase in effective mass and stiffness, altered kinematics, and less excursion and smaller joint rotations (Choi, 2005). Voluntary and reflexive muscle activation of a vehicle occupant is modeled by active joint element at each anatomical joint position (e.g., shoulder, knee, spine, and etc.). There are two basic elements at each joint, i.e., passive kinematic joint elements and torque actuators. Assuming that a co-contraction of agonist and

antagonist muscles stiffens the joint articulation, spring constant and damping coefficient of the passive kinematic joint element are adjusted for the different level of co-contraction, which is considered as a major mechanism of voluntary muscle activation. A vestibular reflexive muscle activation for the posture stabilization is modeled by active torque with PID close loop control. Active torque, the control signal is a sum of proportional, integral, and derivative terms between current and reference states of the joint angle.

Test of jerk loading applied to elbow joint which is relatively simple one dimensional articulation is performed with live human subjects to identify and quantify the active response with different muscle conditions. Two kinds of numerical elbow models, i.e., 3D finite element mesh and Modelica models are built to reproduce the active response to the jerk loading and further to elucidate those kinesiological behavior of bracing human joint.

2 Jerk loading to elbow joint extension

During the vehicle driving or just riding, external loadings are often applied to the occupant as perturbations, e.g., vertical bumping on rough road, lateral G force at cornering, and autonomous braking with ADAS (Advanced Driver Assistant System). It would be quite natural that the occupant spontaneously brace to keep his (or her) upright sitting posture. In order to mimic this kind of perturbation of vehicle in motion and bracing behavior of the occupant, jerk loadings to elbow joint extension are performed as follows.

2.1 Anthropometry of test subjects

Five male subjects are recruited and their average age and anthropometric data are listed in Table 1.

Table 1 Average data (SD) of five test subjects

age	height	Weight (kg)	Fat Free Mass(kg)*	forearm weight(kg)**
28 (2.3)	172 (4.6)	72.8 (5.1)	53.9 (2.1)	1.714 (0.11)

*: from inbody analysis

** : calculated from GEBOD (Huaining Cheng, 1996)

2.2 Jerk loading test

The elbow joint with simple 1-DOF is selected. Upper body and upper arm of test subject are restrained and the elbow joint angle is to maintain its initial position, i.e., keeping the forearm levelled before and after the jerk loading. There are two kind of loadings, 5kgf static loading on hand and 3kgf jerk loading on wrist which is initially carried by a string and just becomes a jerk load when string is cut. (See Fig. 1). The subject has two test conditions, 1) co-contraction versus single contraction and 2) recognition versus unrecognition of jerk loading. Co-contraction or single contraction is respectively attempted by contracting both agonist (e.g., biceps) and antagonist (e.g., triceps) muscles or only agonist muscles. Recognition of the jerk loading to test subject is made by letting him to make his own observation of the action of string cut, i.e., open eye condition. On the contrary, the closed eye condition does not allow the test subject to become aware of the precise moment string cut. There are thus a total four cases of test conditions, “open eye tensed” (recognized with co-contraction), “closed eye tensed” (unrecognized with co-contraction), “open eye relaxed” (recognized with single contraction), and “closed eye relaxed” (unrecognized with single contraction). All five test subjects have two trials for each case of four test conditions.

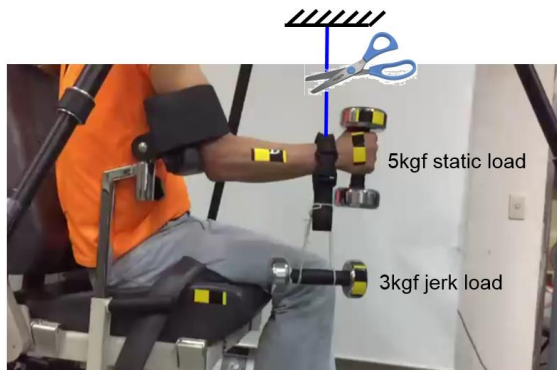
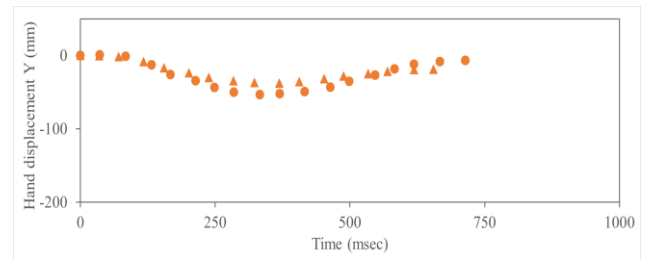


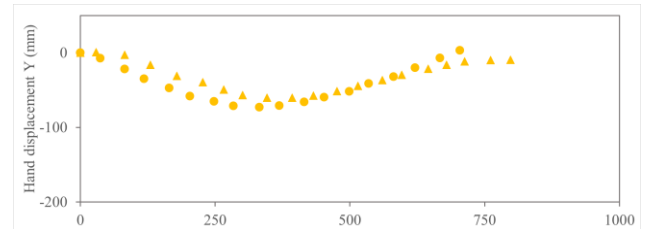
Figure 1 Test setup for jerk loading at elbow joint

2.3 Measurement of hand motion

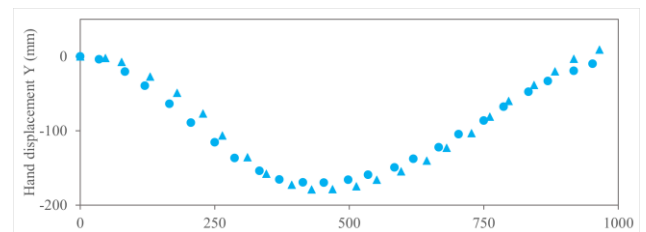
Hand displacements, digitized from video, are shown in Fig. 2. Intra-subject variations are quantitatively assessed by CORA (CORrelation and Analysis, <http://www.pdb-org.com/de/information/18-cora-download.html>) score as listed in Table 2. All five test subjects showed high CORA scores with “open eye relaxed” condition, i.e., good repeatability between two trials at recognized with single contraction muscle condition. It is speculated that the cases with low CORA score are due to the poor coordination of muscle tensing condition of the subject, e.g., closed eye relaxed case with test subject #1.



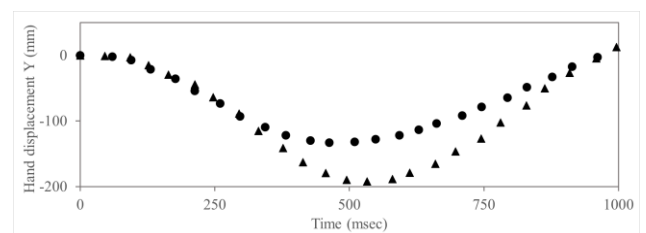
(a) Open Eye Tensed



(b) Close Eye Tensed



(c) Open Eye Relaxed



(d) Close Eye Relaxed

Figure 2 Typical hand displacements in vertical direction (y) due to the jerk loading from subject #2 (●: 1st try, ▲: 2nd try)

Table 2 CORA score for intra-subjects variation

Sub. #	Open eye Tensed	Close eye Tensed	Open eye Relaxed	Close eye Relaxed
1	0.699	0.967	0.957	0.493
2	0.746	0.732	0.948	0.768
3	0.962	0.642	0.955	0.898
4	0.365	0.828	0.720	0.851
5	0.546	0.766	0.795	0.914
Mean (S.D.)	0.693 (0.21)	0.787 (0.11)	0.875 (0.10)	0.785 (0.15)

The inter-subject variation is also represented by test corridors with mean hand displacements as shown in Fig. 3. The open eye tensed condition shows the least width between upper and lower corridors while the open eye relaxed condition has largest.

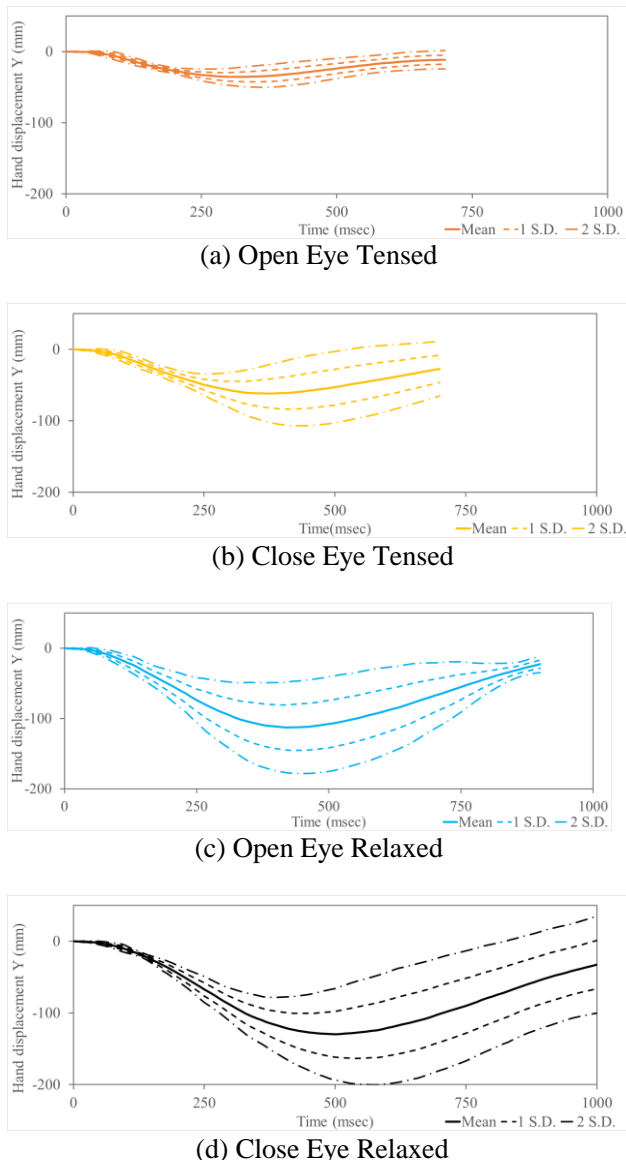


Figure 3 Test corridors and mean hand displacements of three subjects.

3 Three dimensional finite element mesh versus Modelica models

Both FE mesh and Modelica models are shown in Fig. 4. Two rigid bodies, i.e., upper and lower arms articulated by one dimensional kinematic joint element that represents the elbow joint. The dynamic properties of two rigid body are assigned from the average data of five test subjects. It is confirmed that the outcome of both model, i.e., elbow extension and hand displacement from the jerk loading is identical to each other.

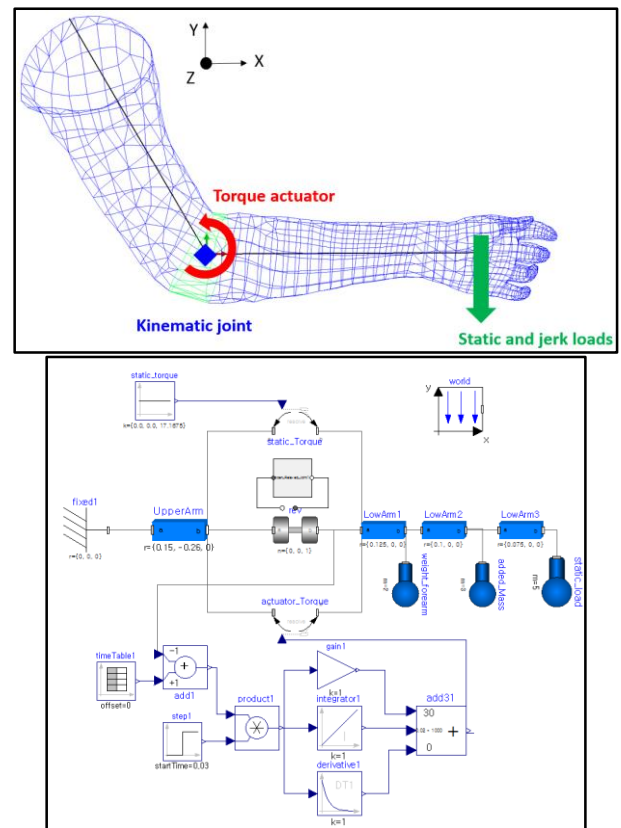


Figure 4 3D finite element mesh (top) and Modelica (bottom) models of active elbow joint

3.1 Modeling of active elbow joint

The numerical modeling of elbow joint and its active response to the jerk loading is designed by implementing two mechanical components, a passive 1D kinematic joint element and a torque actuator. The linear stiffness and damping coefficient of the passive 1D kinematic joint element present the level of co-contraction that stiffens the elbow joint articulation. Voluntary and reflexive muscle activation responding to the jerk loading is modeled by the torque actuator with a PID close loop feedback control. Considering that the test subject tries to keep the initial elbow joint angle, torque (M_z) is activated to minimize the error which is the difference between the initial and current elbow joint angles. Meijer et al (2013), and Brodin et al (2015) presented successful applications of the active torque with PID control to their active human body models. Gain values for the PID control, i.e., Proportion, Integral, and Derivative terms determine the rates of torque generation. Faster torque generation with larger gain values stands for the recognition of jerk loading, i.e., “open eye” condition in the subject test. On the other hand, “closed eye” condition for unrecognized and thus more reflexive response that is modeled by smaller gain values. Comparison of hand displacements between subject test and simulation for four cases are shown in Fig. 5. The comparison of hand position at the maximum elbow extension between subject test and simulation is also shown in Fig. 6.

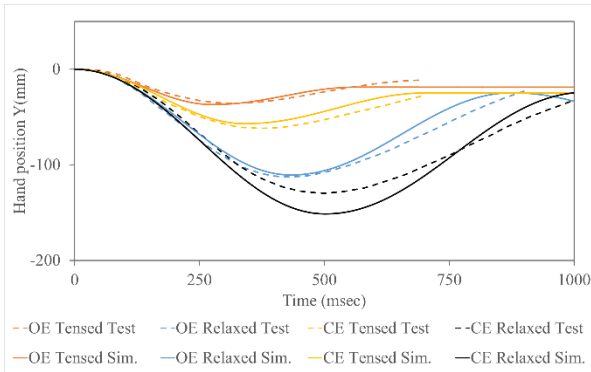
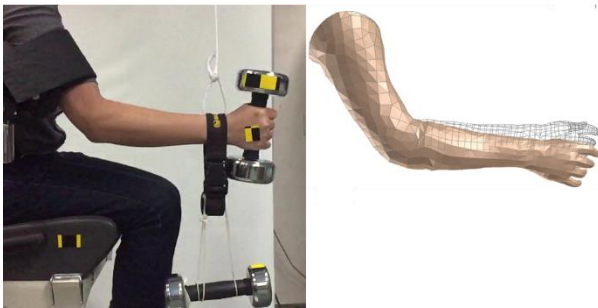
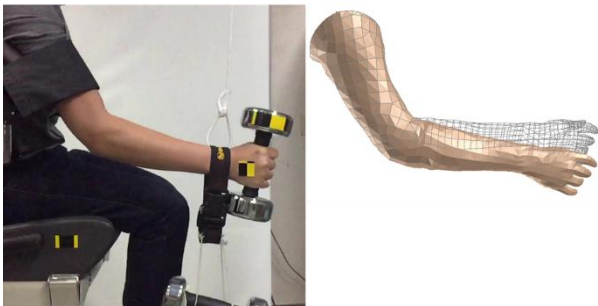


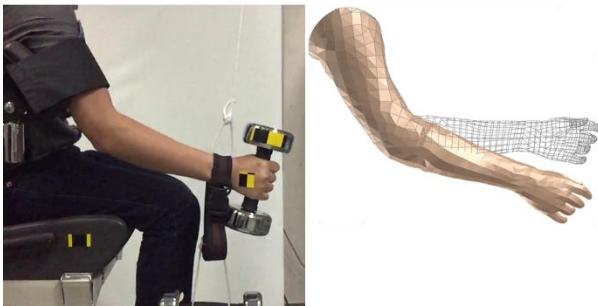
Figure 5 Comparison of hand displacements between subject test and simulation.



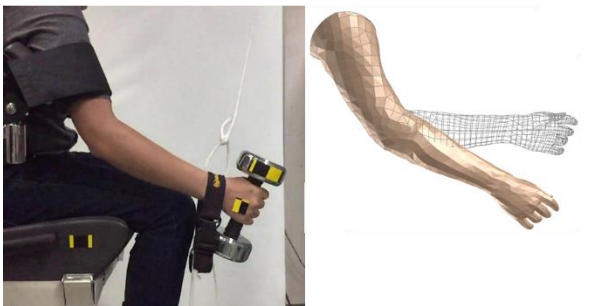
(a) Open Eye Tensed



(b) Close Eye Tensed



(c) Open Eye Relaxed



(d) Close Eye Relaxed

Figure 6. Comparison of hand position at the maximum elbow extension between subject test (1st trial of Subject #4) and simulation

3.2 Hypothesis on the modeling of active elbow response

Table 3 lists modeling parameters of active elbow joint for all four cases. The derivative term in PID close loop control turns out to be insensitive in this simulation of active response to the jerk loading and thus excluded. Those parameters were estimated by heuristic method (trial and error) and based on following hypotheses;

1. Muscle condition e.g., co-contraction (tensed condition) vs. single contraction (relaxed condition) is modeled by damping coefficient of K-joint as shown in Fig. 7.

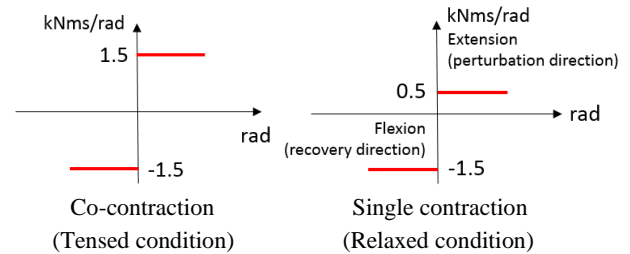


Figure 7 Damping coefficients for different muscle tensing conditions

2. Stiffness of K-joint is dependent on inter subject variations, e.g., muscular structure, gender, etc.

- Muscular build (stronger) arm \uparrow , male > female, and so on.

3. Recognition of perturbation (Open eye vs. closed eye) is controlled by gains of PID close loop control

- Relaxed condition (single contraction) has decreased gains by factor of 0.35 than tensed condition (co-contraction)

4. Muscle reflex latency (delay), 30ms is given to the closed eye condition

Table 3 Modeling parameters and CORA score of active elbow joint model

Modeling parameters	Open eye Tensed	Close eye Tensed	Open eye Relaxed	Close eye Relaxed
Damping C. (kNms/rad)	-1.5/1.5	-1.5/1.5	-1.5/0.5	-1.5/0.5
Stiffness (kNm/rad)	0.1	0.1	0.1	0.1
K_p (kNm/rad)	80	50	80*0.35	50*0.35
K_i (kNms/rad)	0.015	0.015	0.015*0.35	0.015*0.35
PID Control Latency(ms)	0	30	0	30
CORA Score*	0.916	0.897	0.950	0.892

* Calculated between test and simulation in Fig. 5

The correlation between test and simulation results for all four cases are qualitatively analyzed by CORA

score as in Table 3. Open eye condition, i.e., recognition of jerk loading, shows better correlation slightly for both tensed and relaxed muscle conditions than closed eye condition. (0.916, 0.950 > 0.897, 0.892)

4 Whole body modeling

The same modeling scheme of active response at elbow joint is extensively applied to the whole body model. The version of multi-body model (c.f., deformable body model) consists of 15 rigid body segments and 14 articulated joints (Fig. 8). Each articulated joint has either 1 DOF (e.g., elbow, knee, etc.) or 3 DOF (e.g., shoulder, hip, spine, etc.) depending on its biomechanical characteristics. Same kind of passive kinematic joint element and active torque as in the active elbow joint model are implemented but its mechanical characteristics, e.g., the moment-angle curve and damping coefficient are dissimilar to each other. The errors to be removed by active torques at articulated joints are a composite function of joint angle changes at every body segments. Human driver voluntarily and/or reflexively braces to maintain upright sitting posture against various kinds of G-forces during vehicle maneuvering such as emergency braking, lane change, cornering, etc. Validation of active human body model against the test data from open literature (Huber, 2015) is now in progress.



Figure 8 Whole body model with 15 rigid body segments and 14 articulated joint

5 Discussion

The SISO (Single-Input Single Output) problem with 1D active elbow joint model becomes MIMO (Multiple-Input Multiple Output) problem with the whole body model. Human driver's muscle recruitment strategy of active response to brace against external perturbations belong to the quite complicated behavioral kinesiology. Also inter and intra subject variations make the active human body model as one of exciting challenges.

References

Karin Brolin, et al. Development of an Active 6 Year Old Child Human Body Model for Simulation of Emergency Events, IRCOBI 2015.

Huaining Cheng et al. (1996) The development of the GEBOD program, Biomedical Engineering Conference, Proceedings of the 1996 Fifteenth Southern

Hyung Yun Choi, et al, Experimental and numerical studies of muscular activations of bracing occupant, ESV 2005

Phillip Huber, et al, Passenger kinematics in braking, lane change and oblique driving maneuvers, IRCOBI 2015

Riske Meijer, et al. Modelling of Bracing in a Multi-Body Active Human Model, IRCOBI 2013.

Modeling and simulation for leg-wheel mobile robots using Modelica

Hiroki Yoshikawa Takatsugu Oda Kenichiro Nonaka Kazuma Sekiguchi

Tokyo City University, Japan, {g1681237, g1591201, knonaka, ksekiguc}@tcu.ac.jp

Abstract

Modeling of complex robots which consist of mechanical and electric elements has attracted a lot of attention to be utilized for analysis, simulation and development. In this paper, we model the space exploration robot which has leg-wheel mechanisms using Modelica, which is an equation based language and convenient to cope with a complex physical system. In addition, to evaluate the performance of planetary exploration robots, we conduct simulations considering the space environment using the fundamental control system and the robot model. The simulation results indicate that considering load shift due to centrifugal force is important under low gravitational acceleration. *Keywords: leg-wheel mobile robots, modeling, space robots, control system*

1 Introduction

Leg-wheel mobile robots depicted in Figure 1 attract a lot of attention and are widely developed, because the robots achieve high stability utilizing leg mechanism and high mobility using wheel mechanism. Leg-wheel hybrid platform Quattroped, which has two degree-of-freedom legs, is developed (Shen et al., 2009). In order to climb up onto the steps, the control method for limb mechanism robot ASTERISK is studied (Fujii et al., 2006). The action planning algorithm for a planetary explorer robot LEON is proposed (Rohmer et al., 2010). Since these robots can move on uneven terrain, it is expected to work in planetary exploration. However, conducting



Figure 1. Leg-wheel mobile robots.

experiments in space environment require too much cost and time. Therefore, simulating the robot behavior in space environment appears as a practical choice.

Equation based language Modelica is very efficacious to model complex systems which have mechanical and electrical elements. Several studies have reported the modeling and simulation results of several industrial applications using Modelica (Otter et al., 2015)(Hirano et al., 2015). In this paper, we model and simulate the behavior of a leg-wheel mobile robot modeled using geometric parameters of ATHLETE which has the leg-wheel mechanism developed by NASA (Wilcox et al., 2007) in the space environment using Modelica.

2 Modeling leg-wheel mobile robot

2.1 Outline

Table 1 shows the characteristics of the leg and the wheel mechanism. The moving speed and efficiency of the wheel mechanisms is higher than that of the leg mechanisms. The robots equipped with the leg mechanisms can move on uneven terrain. Moreover, the leg robots which have the redundancy in leg arrangement can control the wheel position to avoid overturn. The leg-wheel mobile robots possess both characteristics which enhance the robot mobility. In this paper, we focus on the ATHLETE as a typical example of leg-wheel mobile robot.

The ATHLETE is a lunar exploration robot developed by NASA. The leg-wheel mechanisms with six degree-of-freedom consisting of the wheel mechanism and the six joints are mounted on each vertex of the hexagonal body. The ATHLETE is able to allocate loads and move on uneven terrain while maintaining the body horizontally. These leg-wheel mechanisms are utilized in order to accommodate a wide range of tasks.

Table 1. Characteristics of leg-wheel mobile robots

	Leg	Wheel	leg-wheel
Climbing steps	Good	NG	Good
Load sharing	Good	NG	Good
Moving speed	OK	Good	Good

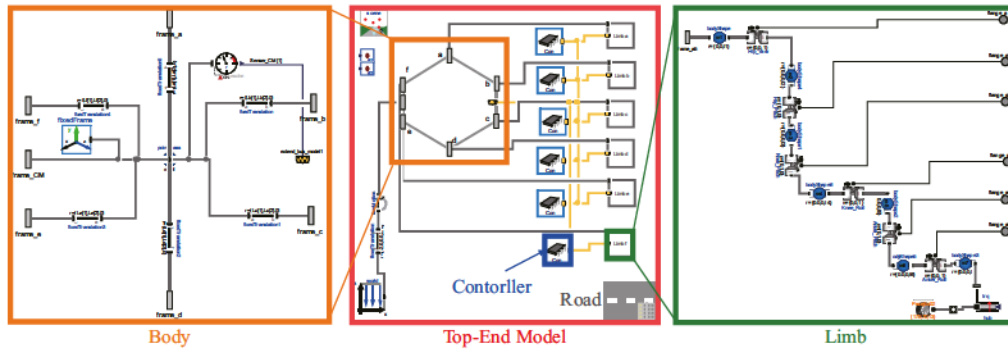


Figure 2. Top-end of Modelica model in Dymola.

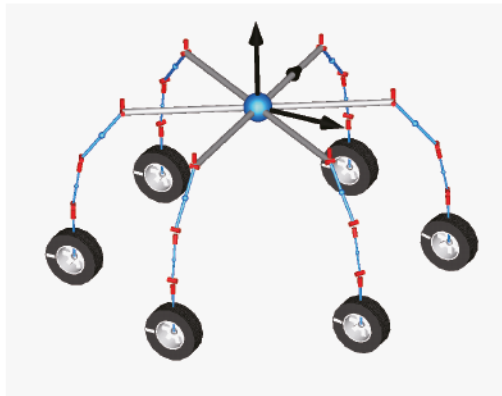


Figure 3. Leg-wheel mobile robot model.

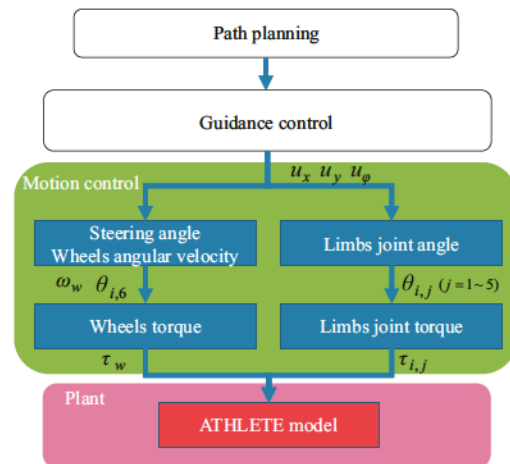


Figure 4. Control system flow.

2.2 Modeling

We model the leg-wheel robot which consists of body and limb-parts using Modelica in order to simulate the behavior of the wheel mechanism and analyze the mobility of ATHLETE. The ATHLETE model described by Modelica is shown in Figure 2 and 3. The robot model is designed as a rigid hexagonal body (orange frame) and links (green frame), as Figure 2 indicates. The body mass point which is the gravity center of the hexagon is set as the origin of the robot coordinate system (black arrows in Figure 3). A leg-wheel mechanism part consists of six revolute joints, seven links and the wheel mechanism. The limb mass points are set on the center of the wheel and the middle of each link. The tire model of Vehicle Dynamics Library (VDL) of Dymola is introduced to reproduce the actual wheel behavior.

3 Structure of the controller

3.1 Outline

As Figure 4 indicates, the controller consists of three layers: path planning layer, guidance control layer, and motion control layer. In this paper, the fundamental control system is proposed to achieve the reference vehicle velocity in the motion control layer. The motion control

layer consists of two parts. First one is the control system for driving and steering of each wheel. Another part determines the posture of the robot. Details of each block are explained in the following sections.

3.2 Motion controller

In this section, we explain the leg-wheel mobile robot model and a calculation method of the controller. Figure 5 depicts the model of the leg-wheel robot. $X_0 \ Y_0$ is the inertial coordinate system and $x \ y$ is the coordinate system fixed to the robot. (X_g, Y_g) is the position of the robot center of gravity (CoG) on the $X_0 \ Y_0$ coordinate system and ϕ is the orientation of the robot. u_x, u_y are the command translational velocity at CoG and u_ϕ is the command angular velocity on the $x \ y$ coordinate system.

Figure 6 shows the configuration of the leg-wheel mechanism of the ATHLETE and the definition of the angle and torque of the leg joints. $\theta_{i,j}$ are the rotation angle of each joint where subscript $i = 1 \sim 6$ indicates the legs number and $j = 1 \sim 6$ indicates joint names of Hip Yaw, Hip Pitch, Knee Pitch, Knee Roll, Ankle Pitch, and Ankle Roll, respectively. In this paper, the Ankle Roll angle $\theta_{i,6}$ is controlled considering moving direction and

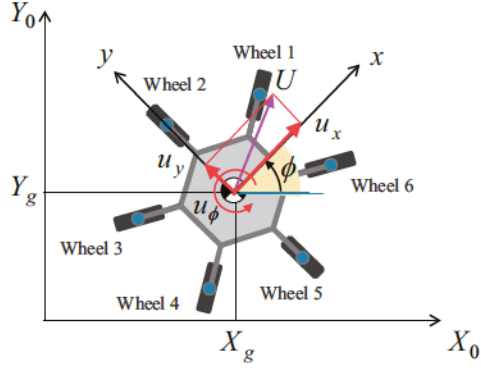


Figure 5. Top-view of plant model.

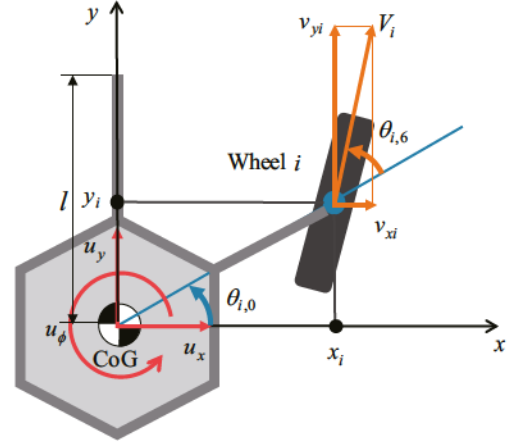


Figure 7. Velocity vector of wheel.

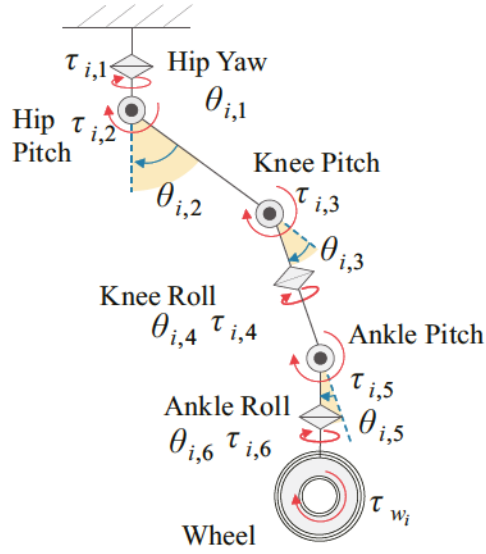


Figure 6. Configuration of limb joints of ATHLETE.

the other angles $\theta_{i,1} \dots \theta_{i,5}$ are controlled to maintain the reference posture.

Figure 7 shows the wheel and the Ankle Roll angle corresponding to the steering angle. $\theta_{i,0}$ is the angle from the CoG position to the wheel.

The translational velocities of the wheel $v_{x,i}$ and $v_{y,i}$ are calculated based on the command velocity at CoG u_x, u_y and u_ϕ as follows:

$$v_{x,i} = u_x - y_i u_\phi, \quad (1)$$

$$v_{y,i} = u_y + x_i u_\phi. \quad (2)$$

Reference angle $\hat{\theta}_{i,6}$ and angular velocity of wheel $\hat{\omega}_{wi}$ are calculated as follows:

$$\hat{\theta}_{i,6} = \tan^{-1} \left(\frac{v_{y,i}}{v_{x,i}} \right) - \theta_{i,0}, \quad (3)$$

$$\hat{\omega}_{wi} = V_i / R_w, \quad (4)$$

$$V_i = v_{x,i} \cos \hat{\theta}_{i,6} + v_{y,i} \sin \hat{\theta}_{i,6}, \quad (5)$$

where R_w is a radius of the wheel and V_i is translational angular velocity. In order to achieve the reference angle

$\hat{\theta}_{i,j}$ and angular velocity $\hat{\omega}_{wi}$, we introduce the PD and P control as follows:

$$\tau_{i,j} = P_{th}(\hat{\theta}_{i,j} - \theta_{i,j}) + K_d(\dot{\hat{\theta}}_{i,j} - \dot{\theta}_{i,j}), \quad (6)$$

$$\tau_{wi} = P_w(\hat{\omega}_{wi} - \omega_{wi}), \quad (7)$$

where P_{th} and P_w are a proportional gain and K_d is a derivative gain.

Zero moment point (ZMP) is a one of concept which is an index of stability. When the ZMP position of the robot is kept inside the support polygon, the stability of the robot body is assured. We introduce the turning limit radius r_{max} considering rolling moment and ZMP position in order to evaluate the relationship between the height of CoG and centrifugal force. If the ZMP position coincides with the tip of the wheel position, the situation of the robot is regarded as a limitation of overturn. In this situation, the turning limit radius r_{max} is calculated as follows:

$$r_{max} = \frac{u_x^2 z_c}{g l}, \quad (8)$$

where z_c is the height of the robot CoG, $l = \sqrt{x_i^2 + y_i^2}$ is the length from CoG position to the wheel position and g is gravitational acceleration.

4 Simulation

4.1 Conditions

To analyze robot behavior in the space environment, we conduct three simulations with the following conditions:

~ Case 1: Turning under the lunar gravity with the height of CoG high ($Z_{CoG}=1.45$ m)

~ Case 2: Turning under the lunar gravity with the height of CoG low ($Z_{CoG}=0.866$ m)

~ Case 3: Turning under the earth gravity with the height of CoG high ($Z_{CoG}=1.45$ m)

Comparing case 1 with 2, we analyze the effect of height of CoG z_{CoG} . The ATHLETE is able to change the height of CoG z_{CoG} by taking advantage of the redundancy of the leg joints. In these simulations, the height of CoG is changed so that the posture of the ATHLETE is maintained. Comparing case 1 with 3, we analyze the influence of the gravitational acceleration while turning. Physical parameters of the ATHLETE are determined based on the reference thesis (Wilcox et al., 2007). Commanded velocities are $\hat{u}_x = 2.5$ m/s, $\hat{u}_y = 0.0$ m/s, and $\hat{u}_\phi = \sqrt{\hat{u}_x^2 + \hat{u}_y^2}/R$. The turning radius R is designed to change smoothly using third-order polynomial from $R = 1000$ m to $R = 2.5$ m in 30 s. These commanded velocities generate a spiral trajectory that the turning radius is gradually decreased.

4.2 Results and Discussion

Figure 8, 9, and 10 show simulation results of case 1, 2 and 3, respectively. Figure 8 (a) and 9 (a) depict the trajectory of the robot, (b) depict the translational and rotational velocity, (c) depict body sideslip angle and (d) depict a vertical load of each wheel F_z . Figure 10 (a) depicts translational and rotational velocity and (b) depicts a vertical load of each wheel F_z .

As shown in Figure 8 (a), when the height of CoG is high, the robot can turn without overturn, however, the robot velocity has the error between commanded and actual velocity, as Figure 8 indicates. It is reasonable that the sideslip angle is generated, as Figure 8 (c) indicates. Since the velocity u_y is caused by centrifugal forces, the body sideslip angle arises. Figure 8 (d) indicates that the vertical load is distributed to each wheel ununiformly. Among them, limb 5 supports 70 % of the total load. The vertical load of limb 1 and 4 are equal to zero. It indicates that the robot is running using only four limbs. As shown in Figure 9 (a), when the height of CoG is lower than in case 1, the robot also can turn successfully. Figure 9 (b)(c) indicate that the tendency of velocity and body sideslip angle are similar to case 1. It is noteworthy that the load shift due to centrifugal force is suppressed, as Figure 9 (d) indicates. The reason is reducing the effect of rolling moment caused by the centrifugal forces. As shown in Figure 10 (a)(b), when gravitational acceleration is smaller than in case 1 and 2, the velocities are realized by the command velocity ; the robot does not generate the sideslip angle. In case 3, the influence of rolling moment generated by the centrifugal forces is smaller than case 1 and 2. It is because the gravitational acceleration works to suppress the roll rotational movement.

To evaluate these simulation results, we discuss the limit turning radius r_{max} . At the turning radius of $r_{max} = 3.03$ m, the robot will fall down due to centrifugal force in the case 1. The robot is not overturn but some wheels

are floating at the target turning radius of $R = 2.5$ m which is smaller than the limit turning radius r_{max} in case 1. On the other hand, when the target turning radius is modified to $R = 3.1$ m in case 1, all wheels contact with the load surface. There is little error between the limit and target turning radius within 0.1 m in this case, even though Eq. (8) assumes simplified model. Thus, we obtain the adequate simulation results. We can use the relationship of balance of moment for the control. It is expected that the turning ability in case 2 and 3 is better than that of case 1 by Eq. (8), because the limit turning radius decreases as gravitational acceleration increases and the height of CoG decreases. Accordingly, the simulation results indicate that the ZMP position close to the CoG position (X_g, Y_g) by suppressing the load shift of the robot.

The robot tends to generate skid in lunar space where the effects of gravitational acceleration are smaller than that of the earth. These results indicate that in order to achieve high mobility under the lunar environment, considering the sideslip angle and load shift is important, because the robot tends to overturn under low gravitational acceleration. To suppress load shift by centrifugal forces, leg-wheel mobile robots can lower the height of CoG using the leg mechanism. In addition, arranging the wheel position, the leg-wheel mobile robots can achieve high mobility utilizing the redundancy of the leg mechanism.

5 Conclusions

In this paper, we model the leg-wheel mobile robots which have the leg-wheel mechanism using Modelica and conduct the simulation considering the lunar environment. The simulation results indicate that the robots tend to generate the vehicle sideslip which is the cause for load shift and overturn because of low gravity acceleration. Therefore, the motion controller, which considers vehicle slippage, is required to achieve high mobility.

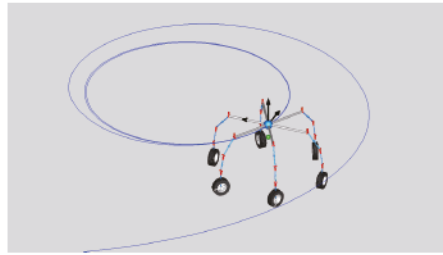
The future directions of this study are designing guidance controller, modeling motor dynamics, and considering a terramechanics which express the effects between wheel and sand called regolith.

6 Acknowledgments

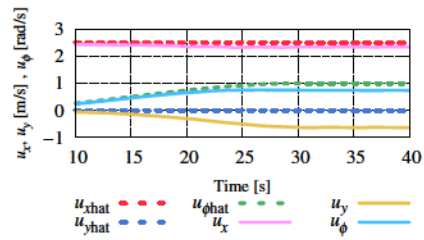
The authors gratefully acknowledge the support of Grant in Aid for Scientific Research (C) No.15K06155 of Japan.

References

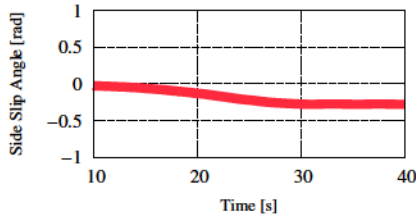
- Shota Fujii, Tomohito Takubo, and Tatsuo Arai. Climbing up onto steps for limb mechanism robot "ASTERISK". In *International Association for Automation and Robotics in Construction*, pages 225–230, 2006.



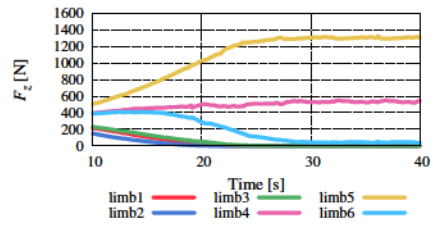
(a) Torjectory (until 40 s).



(b) Transrational and rotational velocity.

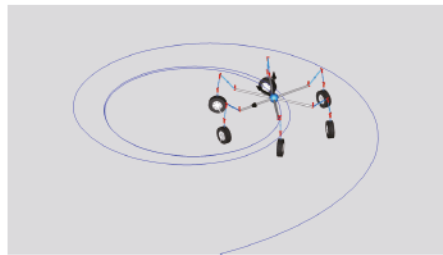


(c) Body sideslip angle.

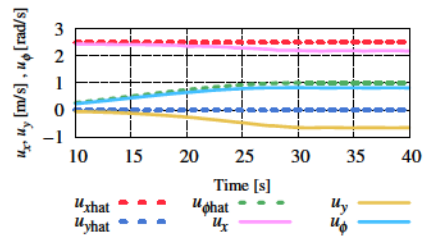


(d) Vertical load of each wheel.

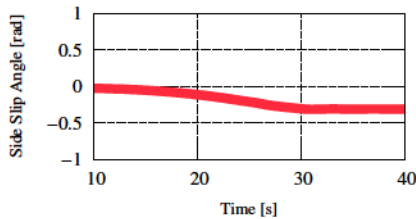
Figure 8. Case 1 -High Center of gravity with the lunar gravitational acceleration- ($R=2.5, z_{CoG}=1.45, g=1.65$)



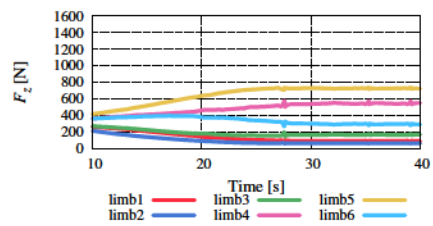
(a) Torjectory (until 40 s).



(b) Transrational and rotational velocity.

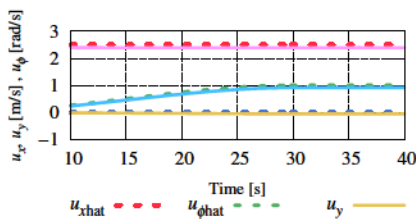


(c) Body sideslip angle.

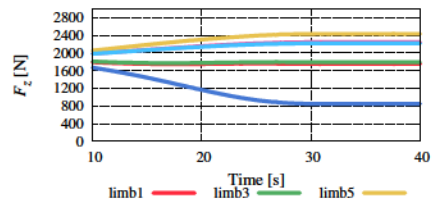


(d) Vertical load of each wheel.

Figure 9. Case 2 -Low Center of gravity with the lunar gravitational acceleration- ($R=2.5, z_{CoG}=0.886, g=1.65$)



(a) Transrational and rotational velocity.



(b) Vertical load of each wheel.

Figure 10. Case 3 -High Center of gravity with the earth gravitational acceleration- ($R=2.5, Z_{CoG}=1.45, g=9.81$)

Yutaka Hirano, Shintaro Inoue, and Junya Ota. Model based development of future small electric vehicle by modelica. In *11th International Modelica Conference*, pages 143–150, 2015.

Martin Otter, Nguyen Thuy, Daniel Bouskela, Lena Buffoni, Hilding Elmqvist, Peter Fritzson, Alfredo Garro, Audrey Jardin, Hans Olsson, Maxime Payelleville, Wladimir Schamai, Eric Thomas, and Andrea Tundis. Formal requirements modeling for simulation-based verification. In *11th International Modelica Conference*, pages 625–635, 2015.

Eric Rohmer, Giulio Reina, and Kazuya Yoshida. Dynamic simulation-based action planner for a reconfigurable hybrid leg-wheel planetary exploration rover. *Advanced Robotics*, 24, 2010.

Shuan-Yu Shen, Cheng-Hsin Li, Chih-Chung cheng, Jau-Ching Lu, Shao-Fan Wang, and Pei-Chun Lin. Design of a leg-wheel hybrid mobile platform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4682–4687, 2009.

Brian H. Wilcox, Todd Litwin, Jeff Biesiadecki, Jaret Matthews, Matt Heverly, Jack Morrison, Julie Townsend, Norman Ahmad, Allen Sirota, and Brian Cooper. ATHLETE: A cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 27(5):421–434, 2007.

System-Level Design Trade Studies by Multi Objective Decision Analysis (MODA) utilizing Modelica

Joshua Sutherland, Kazuya Oizumi, Kazuhiro Aoyama¹ Naoki Takahashi² Takao Eguchi³

¹Department of System Innovation, The University of Tokyo, Japan,
{joshua, oizumi, aoyama}@m.sys.t.u-tokyo.ac.jp

²Dassault Systèmes K.K., Tokyo, Japan naoki.takahashi@3ds.com

³Shinko Research Co. Ltd, Tokyo, Japan eguchi.takao@kobelco.com

Abstract

This paper describes an approach and tool to conduct System-Level Design Trades Studies utilizing Modelica by way of Multi Objective Decision Analysis (MODA). Requirements for this being identified from the problems experienced on student Solar-Boat project.

The proposed approach and tool utilizes Modelica to predict performance of different competing alternative designs and MODA as a way to consistently compare those alternatives subject to a range of Assessment Scenarios.

To enable alternative designs to be created with low effort the replaceable feature of Modelica components is used such that the alternatives can share common architectures subject to a defined hierarchy which includes the Assessment Scenario itself.

A tool was created to automate the placement of alternative designs into the Assessment Scenarios, run the simulations and consolidate the results via MODA. Examples utilizing the approach and tool to predict performance of competing Solar-Boat designs and compare them is provided.

Keywords: Trade Studies, Assessing Alternative Designs, System-Level Design

1 Introduction

System-Level Design is defined in (Ulrich *et al.*, 2011) to “include the definition of the product architecture and the decomposition of the product into subsystems and components”. Expanding to describe what an engineer must achieve with the System-Level Design, it is expected that there is sufficient level of detail to enable the system being designed to be assessed from the perspective of predicting its performance and the cost sufficiently accurately while simultaneously informing what is acceptable to be designed at the detailed design stage, as such bounding the number of alternatives at the detailed design stage to a reasonable level.

Further the development of a System-Level Design should involve the comparison of alternative competing designs from which one or many might be selected for further detailed design.

As described by (Parnell *et al.*, 2014) trade studies (or tradeoff studies) play a central role in decision management and can be applied throughout a systems lifecycle. With the term “tradeoff” implying that there may be the need to forgo one objective to obtain a desired level in another. As such trade studies are suitable for usage in System-Level Design which includes the selection of a design from a set of competing alternatives.

1.1 Solar-Boat Project Description

Every summer on Japan’s Lake Biwa multiple university teams participate in a competition to race fully automatus solar powered boats over a 20km course which they have designed, manufactured and tested over the previous year. The University of Tokyo, Department of Systems Innovation regularly participates in this event, where all boats are subjected to the following rules: Maximum 2m² solar panels, 25Wh lead based batteries and ability to carry a 64g payload. Figure 1 shows an example from 2014 of the hydro foiling craft constructed by University of Tokyo students.



Figure 1. 2014 University of Tokyo Solar-Boat.

1.2 Problems with Previous Solar-Boat Projects

In (Sutherland *et al.* 2015) a detailed analysis of the activities conducted on the Tokyo 2014 Solar-Boat project was conducted, the resulting summary of problems mapped to Lifecycle Stages (LS) is listed in Table 1. Reviewing the listed problems, the lack of design exploration and performance prediction at LS3: System-Level Design focused on a design target identified in LS2: Concept Development resulted in further problems downstream where alternatives generation and simulated performance prediction were

not the norm. As such much trial and error based on physically realized components was performed wasting resources.

Table 1. Solar-Boat Lifecycle Stages, 2014 problems and some proposed solutions.

<i>Lifecycle Stage (LS)</i>	<i>2014 problems</i>	<i>Proposed solutions</i>
LS1: Clarify	Slow to acquire initial knowledge.	Provide knowledge in models.
LS2: Concept Development	Unclear of the design target.	Complete trade-off analysis of multiple designs using models to simulate performance.
LS3: System-Level Design	Little exploration of alternatives or their predicted outcomes.	
LS4: Detail Design	Little prediction of performance.	
LS5: Production, Test and Refinement	Based on trial and error.	
LS6: Race	Lost race due to faults which could have been predicted with modeling.	
LS7: Knowledge Transfer		

Given that design exploration and performance prediction were conducted inadequately on the 2014 project it is important to survey work products generated and used related to early lifecycle stage numerical simulation. The work products which did exist were numerical models in Excel and MATLAB/Simulink. These models while somewhat useful for predicting performance suffered from the following problems:

1. Lack of modularity, preventing: Reuse of existing model components in new situations; generating alternative designs out of different module combinations; different students developing models independently and subsequently virtually integrate.
2. Lack of adequate libraries, resulting in: inaccurate approximations to complex components (e.g. a solar panel array is a “power generator” of a particular efficiency, rather than a component which interacts with a circuit current and voltage).
3. Lack of holistic model resulting in subsystems being designed in isolation (e.g. Powertrain designed separately to the main system structure).

4. Lack of infrastructure to assess and compare alternative designs consistently with each other making it unclear as to what design has been selected for what reason.
5. Lack of access to variables which the models were not “designed” to provide. Much modification is required to Excel and MATLAB/Simulink to expose a new variable of interest.

1.3 Proposing Solutions

Based on these problems identified at the early lifecycle stages some high level solutions were proposed in (Sutherland, 2016) to help alleviate these problems (also shown in Table 1) by:

1. Providing knowledge in models.
2. Completing trade-off analysis of multiple designs using models to simulate performance.

Given the different types of knowledge required to be stored and mechanisms to explore alternative designs it is proposed by (Sutherland, 2016) to utilize a conceptual modeling language (OPM, Object Process Methodology (ISO, 2015)) for LS2: Concept Development and numerical modeling language with subsequent simulation (Modelica) for LS3: System-Level Design. In this paper the proposed usage of an automation framework for expediting the completion of trade studies utilizing Modelica for LS3: System-Level Design is explored.

2 Developing Requirements for the Tools and Approach while Reviewing Existing Literature

Table 2 details a set of requirements for a trade study tool and approach which aims to address the issues with the previous Solar-Boat projects approach. A brief comparison to existing tools and methodologies is provided in the following sections which the focuses on an adequate numerical modeling and simulation language (Section 2.1) and systematic approach (Section 2.2).

2.1 Modelica

The use of Modelica subject to a logical approach can address many of the requirements identified in Table 2. Describing this explicitly:

1. The replaceable keyword enables subsystems and components to be replaced subject to a defined interface.
2. Large libraries of standard components exist and new ones can be developed based on equations quickly.
3. Components integrate across multiple domains.
4. Simulations provide access to all the variables of the equations which describe the components behavior.

Table 2. Solar-Boat previous numerical modeling problems and requirements for the proposed system.

2014 numerical modelling problems	Requirement for proposed system
1) Lack of modularity	Can replace components and subsystems with any other which is compliant to a defined interface.
2) Lack of adequate libraries	Access to a range of library components. Can develop new library components quickly.
3) Lack of holistic model	Integrate multiple engineering domains concurrently.
4) Lack of infrastructure to assess and compare	Can assess and compare all alternative designs consistently and automatically.
5) Lack of access to variables which the model was not “designed” to provide	Can review the details of individual components performance.

2.2 Trade Study Approaches

The International Council of Systems Engineering (INCOSE) provides by way of description in (INCOSE, 2015) and (SEBoK, 2015) a decision management process which is intended for trade studies. Two implementations of this process are provided in (Cilli *et al.*, 2014) and (Edwards *et al.*, 2015). In these implementations a common architecture of subsystems is defined for the system of interest, variation of the subsystems within this architecture enables alternative System-Level Designs to be generated. Each of these alternatives are then assessed by a common set of performance metrics which have been mapped to the functional objectives of the system of interest. In the case of (Cilli *et al.*, 2014) this involved mapping the amount of value derived for a particular functional objective from a prediction of its performance by way of a value function. Multiple objectives are then combined by way of weighting to enable Multi Objective Decision Analysis (MODA). In (Cilli *et al.*, 2014) performance prediction is provided by subject matter experts, while in (Edwards *et al.*, 2015) simulation is used, but the simulation model does not use acausal interactions between the individual components which make up the model. Instead subject matter experts define the interaction between components based on the equations and data they wish to utilize. This process is somewhat opaque.

As such, while the approaches used by (Cilli *et al.*, 2014) and (Edwards *et al.*, 2015) to implement the INCOSE decision management process can form a basis

of an approach, it is proposed for this paper and the Solar-Boat project to utilize Modelica as the numerical modeling tool such that the benefits described in Section 2.1 can be realized when completing a trade study.

3 Proposed Tools and Approach

A high level flow diagram of the proposed tools and approach, developed and demonstrated is shown in Figure 2. It is described as having three important processing elements (in green on Figure 2) listed as: Model Builder, Simulation Runner and Results Processor.

The required initial inputs of the tools and approach (in orange on Figure 2) take the form of Comparison.xml detailing what Assessment Scenarios and alternative System-Level Designs to consider and a library of Modelica models which are the Assessment Scenarios and alternative System-Level Designs referenced by the Comparison.xml. With the Assessment Scenario describing how to assess a design alternative subject to a set of stated conditions.

Ultimately the aim of running through the approach is to generate insight (in black on Figure 2) into the performance and cost characteristics of alternative designs, which can occur through the reviewing consolidated Multi Objective Decision Analysis (MODA) results or reviewing detailed raw results of the .mat file (in blue on Figure 2) generated during the simulation of the model associated with each design alternative for each Assessment Scenario (blue on Figure 2).

More detailed descriptions are provided in subsequent sections for the items in Figure 2: Inputs to the approach described in Section 3.1 while processing elements and their subsequent outputs are described in Section 3.2.

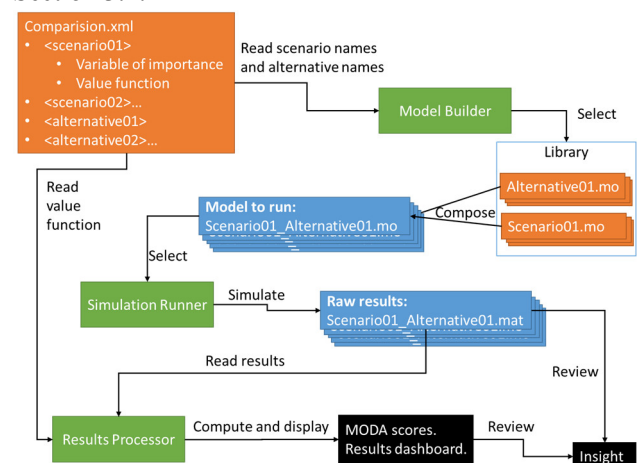


Figure 2. Proposed trade study approach utilizing Modelica. Green: System processing elements. Orange: Inputs. Blue: Intermediate results. Black: Final results.

3.1 Inputs

3.1.1 Comparison.xml

Comparison.xml is an input to the approach, it is an XML file listing and describing what Assessment Scenarios to complete (including time to simulate and how to processes its results, described in more detail in Section 3.2.3) followed by a listing of alternative System-Level Designs to assess. A code snippet is provided below of an example file (truncated and modified for simplicity).

```

<system_name="solarBoat"/>
<scenario name="Floating">
  <variable="z_top_of_hull"/>
  <variable_units name="m"/>
  <value_func_direct name="neg"/>
  <min_accep_perform val="-0.1"/>
  <stretch_goal val="-0.4"/>
  <weight val="0.5"/>
  <sim_length val="70"/>
  <extract_data_type name="mean"/>
</scenario>
<scenario name="StraightLineAvSun">
  <variable="x_velocity"/>
  <variable_units name="m/s"/>
  <value_func_direct name="pos"/>
  <min_accep_perform val="1.5"/>
  <stretch_goal val="3"/>
  <weight val="0.5"/>
  <sim_length val="3"/>
  <extract_data_type name="max"/>
</scenario>
<scenario name="Cost">
  <variable="cost_money"/>
  <variable_units name="yen"/>
  <value_func_direct name="neg"/>
  <min_accep_perform val="300000"/>
  <stretch_goal val="0"/>
  <sim_length val="1"/>
  <extract_data_type name="max"/>
</scenario>
<design name="Ideal"/>
<design name="Boat_Alternative_01"/>
<design name="Boat_Alternative_02"/>

```

3.1.2 Library – Structure and Conventions

To manage complexity the library and the models utilized are divided into distinct hierarchy levels which are outlined in Table 3. Figure 3 pictorially depicts how Level 4 models ultimately combine to form a Level 1 model which can be simulated. Each level of this hierarchy is discussed in the subsequent sections.

In addition assumptions made about the Solar-Boat are explicitly listed to enable a consistent library to be developed by way of setting rules for how these assumptions are implemented in the library. To generalize, these assumptions stem from the bottom up approach used to develop the models, where for example Subsystems are defined by their interface and internal Subsystem-Components. Assumptions

associated cost are described in Table 4, assumptions associated with mass are described in Table 5, and those regarding fluid interaction are described in Table 6.

Table 3. Hierarchy levels utilized in the models and libraries.

Hierarchy Level	Name	Example
Level 1	Assessment Scenario	Straight line good weather
Level 2	System of Interest	Solar-Boat
Level 3	Subsystems	Electrical to Thrust
Level 4	Subsystem-Components	DC Motor

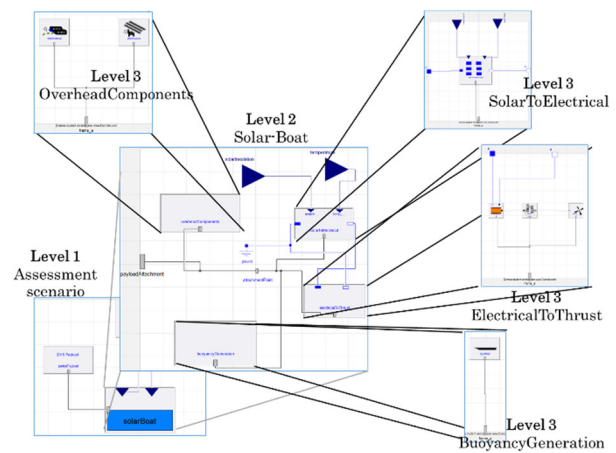


Figure 3. Representation of the different hierarchy levels combining.

Table 4. Assumptions associated with cost and how the assumptions are implemented in the library.

Assumption / Design process decision	Implementation in Library
All objects of the Solar-Boat have cost.	All models at Level 2-4 extend “PartialProcurementAttributes” with the single variable <code>cost_money_computed</code> . As such they must expose/compute their cost.
Cost properties of the System of Interest occur from the sum its Subsystem-Components cost properties.	Every Subsystem-Component defines a cost parameter. The cost of the subsystem is the sum of the cost of its components. The same logic follows up the levels.

Table 5. Assumptions associated with mass and degrees of freedom and how the assumptions are implemented in the library.

<i>Assumption / Design process decision</i>	<i>Implementation in Library</i>
All objects of the Solar-Boat have mass.	All models at Level 2-4 extend “PartialMassAttributes” with the single variable mass_computed. As such they must expose/compute their mass. Which might be the sum of lower level component masses.
Mass properties of the System of Interest occur from the sum its Subsystem-Components mass properties.	Every Subsystem-Component attaches a mass component from the Mechanics.MultiBody library to its Frame_a connector.
System of Interest is a single rigid body in a 3D world.	All Subsystems and components expose a Modelica.Mechanics.MultiBody.Interfaces.Frame_a connector.
The number of degrees of freedom in motion required at different times if the lifecycle varies.	The use of Modelica.Mechanics.MultiBody.Joints.Prismatic to prevent motion on degrees of freedom which are not going to be considered.

Table 6. Assumptions associated with fluids and their implementation in the library.

<i>Assumption / Design process decision</i>	<i>Implementation in Library</i>
All objects immersed in a fluid generate a drag_force and buoyancy_force.	Any models Level 2-4 expected to be immersed in a fluid extend “PartialInAFluidAttributes” with the variables drag_force and displaced_volume exposed. As such these must be computed.

3.1.3 Library Level 1 – Assessment Scenario

The Assessment Scenario is a Modelica model which aims to provide the necessary infrastructure around a replaceable partial model of the System of Interest (Solar-Boat) the necessary inputs to represent the scenario for simulation. Figure 4 shows an example of one such of these for driving in a straight line subject to average sun conditions. With Figure 4 (left) showing

prior to population with a valid alternative and Figure 4 (right) showing this after having been populated with a valid alternative Solar-Boat design.

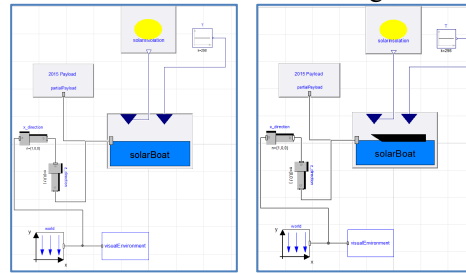


Figure 4. Assessment Scenario: Straight line average sun. Left: Before being populated with a valid Solar-Boat design alternative. Right: After being populated with a design alternative such that it can be simulated.

3.1.4 Library Level 2 – System of Interest

This level describes the system which is being attempted to be designed and assessed (i.e. Solar-Boat). As such all valid alternative designs should be compliant with the interface used for the System of Interest in the Assessment Scenarios. In addition the variables of interest defined in the Comparison.xml (e.g. x_velocity) must be declared such that they can be extracted by the Results Processor. Figure 5 shows the partial model interface, while Figure 6 (left) shows an example architecture created by the population with partial replaceable Subsystems interfaces. In this case four Subsystems are utilized: Solar to Electrical, Electrical to Thrust, Buoyancy generation and Overhead mass components. This architecture is then populated with Subsystem models to generate a Solar-Boat alternative as shown in Figure 6 (right).

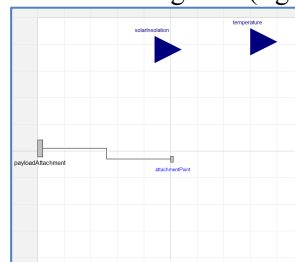


Figure 5. Partial model defining the interface of the System of Interest (Solar-Boat).

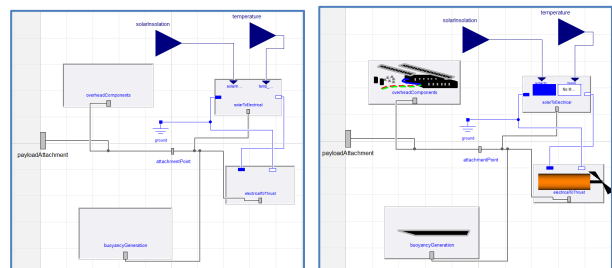


Figure 6 Left: Extending the partial model of the System of Interest (Solar-Boat) and subsequently defining an architecture by placing partial Subsystems on it. Right: Populating this architecture with Subsystems.

3.1.5 Library Level 3 – Subsystems

Subsystems have an interface to define their interaction at the System of Interest level. An example is shown in Figure 7 for an electrical to thrust Subsystem. A Subsystem architecture (Figure 8 left) can then be defined for the interface (Figure 7) by the population with partial replaceable Subsystem-Component interfaces. Subsequent population with Subsystem-Components results in an alternative Subsystem being defined (Figure 8 right).

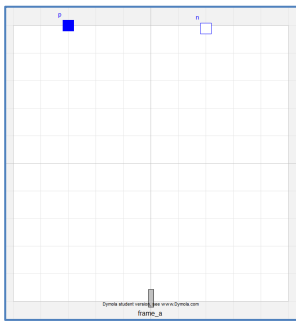


Figure 7. Partial model defining the interface of a Subsystem (electrical to thrust).

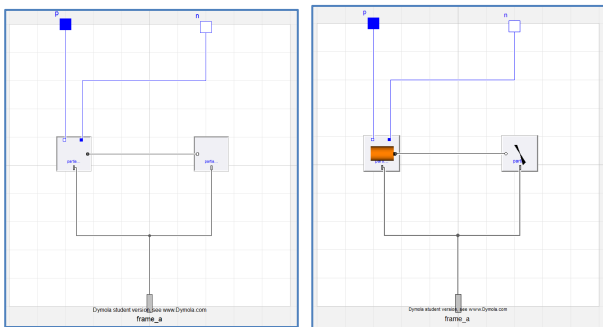


Figure 8. Left: Extending the Electrical to thrust partial model and subsequently defining an architecture by placing partial Subsystem-Components on it. Right: Populating this architecture with Subsystem-Components.

3.1.6 Library Level 4 – Subsystem-Components

The lowest level of the defined hierarchy are Subsystem-Components. Similar to the other levels an interface is used to define their interaction at the higher levels, as shown in Figure 9. However architecture implementation takes a different form, generally being made of additional models (custom and standard library) which have their parameters provided by way of redeclaring a partial record. The aim of this approach is to create a large library of components based on the specification sheets of commercial products which can be transferred to a record in the Modelica language. As per the assumptions discussed in Section 3.1.2 each Subsystem-Component must declare a mass and cost which can then be used to compute the mass and cost of the Subsystem it is included in. Further as shown at the bottom of Figure 10 each Subsystem-Component includes a mass component from Mechanics.MultiBody library.

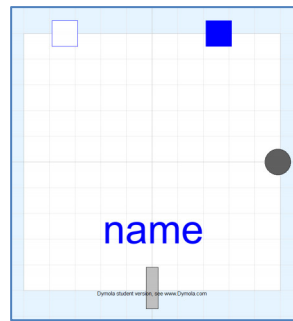


Figure 9. Interface of Subsystem-Component (Electrical to rotation).

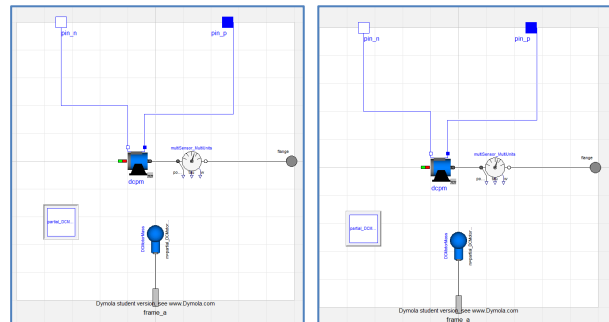


Figure 10. Left: Implementing the Electrical to rotation component with a partial record. Right: Populating the partial record to create a Subsystem-Component.

3.2 Processing Elements

In this section the processing elements of the approach proposed (shown in Figure 2) which process inputs to generate output are discussed. This was implemented with Python code as Dymola and OpenModelica were unable to automate the variation of Modelica blocks or the consolidation of multiple results.

3.2.1 Model Builder

The Model Builder processing element generates a Modelica model for each combination of Assessment Scenario and System of Interest (Solar-Boat) alternative described in the Comparison.xml file. The Model Builder requires that the Assessment Scenarios and System of Interest (Solar-Boat) alternatives named in the Comparison.xml are available from the library. This is achieved programmatically by duplicating existing model for the Assessment Scenario and manipulating the .mo text file to change the Solar-Boat alternative to the one for assessment.

3.2.2 Simulation Runner

Simulation Runner subsequently simulates all the models created by Model Builder for the simulation length specified in the Comparison.xml file. This is achieved programmatically by utilizing Dymola's python interface. The subsequent results (in the .mat file) can then be further reviewed by the engineer if they wish.

3.2.3 Results Processor

The results processor extracts for each model simulated (Assessment Scenario and design alternative pair) the time series of the raw simulation results the variable of interest for each Assessment Scenario to measure the System of Interest's performance (e.g. max x_velocity). This extraction is enabled by Dymat python package (Dymat, 2015). For simplicity weighted sum MODA scheme described in (Cilli *et al*, 2014) was used which is explained as follows.

The extracted variable (e.g. max x_velocity) is used to compute unweighted value by the utilization of the value function (see Figure 11 for an example) which is defined in Comparison.xml and maps performance on a particular scenario to unweighted value (minimum acceptable performance corresponding to zero value and stretch goal corresponding to value of 1). The unweighted value is then multiplied by the weight assigned in Comparison.xml to create weighted value for that scenario. Summing the for all the scenario runs for an individual alternative design results in the total weighted value for that particular alternative design. With the "ideal system" having a value of 1 as its performance is assumed to always be at the stretch goal and the weights sum to 1.

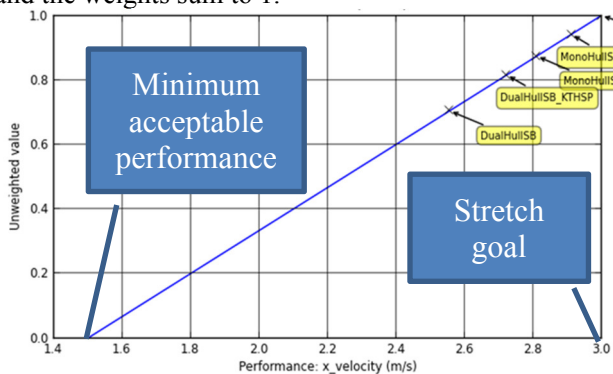


Figure 11. Example linear value function.

These results can then be consolidated on to a single dashboard (example shown in Figure 12). On all charts the y-axis displays the total weighted value for the alternative. For the top chart and middle chart the x-axis indicates the alternatives being considered. With the middle chart further displaying a breakdown of the weighted value contributions from each Assessment Scenario. For the bottom scatter chart, each point on the bottom chart indicates a design alternative of the System of Interest and the x-axis indicating cost in yen for that particular design alternative.

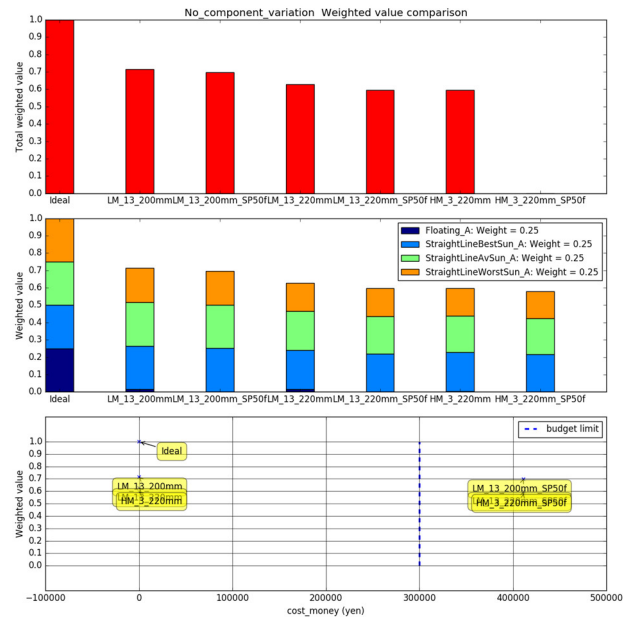


Figure 12. Example output of the Results Processor. Top: Total weighted value (y-axis) computed for all the Assessment Scenarios (x-axis) for all the Alternative System of Interests. Middle: Breakdown of the weighted value contributions from each assessment scenario. Bottom: Weighted value (y-axis) compared to cost in yen (x-axis) for each alternative design. (Results are from Section 4.3 comparison of the introduction of new solar panels).

4 Demonstration

This section provides examples of the utilization of the approach to demonstrate how it can rapidly enable the fast comparison of alternative Solar-Boats. The assessment scenarios used are described in Table 7, which are used to define a Comparison.xml and library of Level 1 models. For simplicity all weights were set equally to 0.25 in all the demonstrations, the Solar-Boat architecture of Figure 6 (left) is utilized.

4.1 Component Variation

An initial set of Solar-Boat alternative designs are outlined in Table 8 and created as models by populating the Solar-Boat architecture of Figure 6 (left). These designs are identical other than the variation in the Subsystem-Components used for a direct drive electrical to thrust Subsystem. The Subsystem-Component variation involves motor variation (high mass and low mass variants) and propeller variation. By following the flow diagram of Figure 2 for the Assessment Scenarios of Table 7 and alternative designs of Table 8, it is possible to generate the results as shown in Figure 13 automatically. Reviewing these results it is possible to see three alternatives fail to meet the minimum acceptable performance on at least one scenario (red ring on Figure 13). The complex interaction between boat mass, water line, thrust, drag and velocity has simplified into a single chart.

Table 7. Assessment Scenarios used in the demonstration.

Measure of interest	Scenario conditions	Min accep perform	Stretch goal	Data type
Top of hull z position (m)	Floating	-0.1	-0.4	Mean
x velocity (m/s)	Best ever insolation (870 Wm ²)	2	4	Max
x velocity (m/s)	Average insolation (550 Wm ²)	1.5	3	Max
x velocity (m/s)	Worst ever insolation (260 Wm ²)	0.5	2.5	Max

Table 8. Alternative Solar-Boat designs created by electrical to thrust Subsystem variation (H = High mass motor, L = Low mass motor).

Alternative name	Buoyancy Generation	Solar To Elec	Elec To Thrust
HM_160m m	Single hull	FT-136SE	H motor: No gearbox: 160mm prop
HM_200m m	Single hull	FT-136SE	H motor: No gearbox: 200mm prop
HM_220m m	Single hull	FT-136SE	H motor: No gearbox: 220mm prop
LM_160m m	Single hull	FT-136SE	L motor: No gearbox: 160mm prop
LM_200m m	Single hull	FT-136SE	L motor: No gearbox: 200mm prop
LM_220m m	Single hull	FT-136SE	L motor: No gearbox: 220mm prop

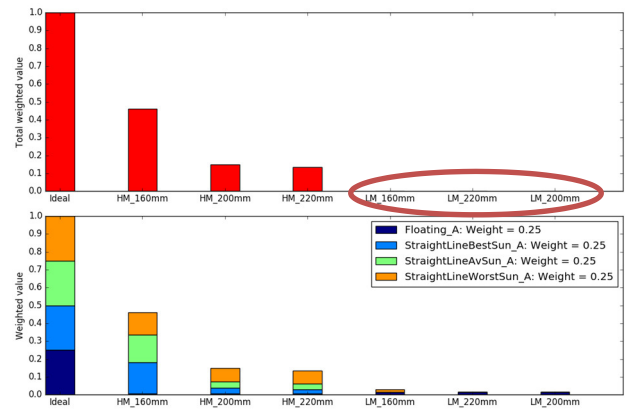


Figure 13. Multi Objective Decision Analysis results for Solar-Boat alternatives of Table 8.

4.2 Subsystem – Architecture Variation

Given the approach makes use of a common architecture for the Solar-Boat it is possible to rapidly compare alternative Solar-Boat designs utilizing different Subsystem architectures. As such Figure 14 displays an alternative electrical to thrust Subsystem architecture (incorporating a gearbox) to the one used previously (Figure 8). As such it is possible to create alternative Solar-Boats utilizing this architecture. Creating several alternatives by varying the Subsystem-Components in this model and simulating results in Figure 15 (where the designs have significant performance increase over the results of Section 4.1).

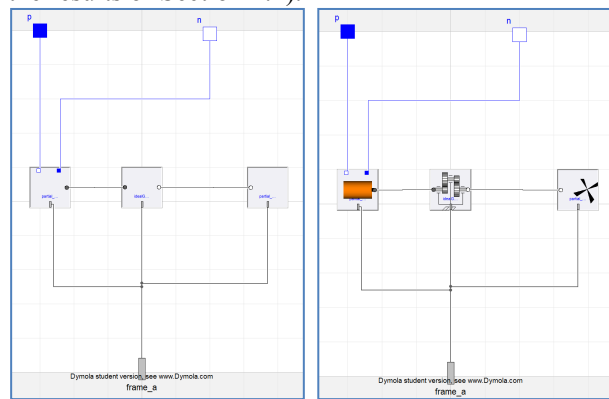


Figure 14. Left: Extending the electrical to thrust partial model but defining a different Subsystem architecture to that in Figure 8 by incorporating a gearbox. Right: Populating this architecture with Subsystem-Components.

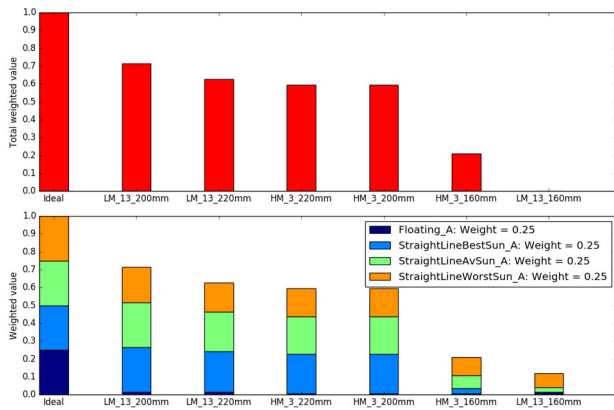


Figure 15. Multi Objective Decision Analysis results for Solar-Boat alternatives incorporating a gearbox.

4.3 Cost Benefit of a New Component

Further component exploration of interest could involve the performance evaluation associated with incorporating higher efficiency, high mass and high cost solar panels (solar to electrical Subsystem). Creating alternatives based on these and simulating results in Figure 12. The bottom chart clearly displays the large cost of the new solar panels (exceeding the project budget). While the weighted total value of alternatives incorporating the panels is not significantly different to those utilizing existing panels. Indicating they are not a wise purchase.

5 Discussion

The proposed approach described in this paper has a number of benefits when compared to other approaches. By clearly describing each Assessment Scenario and processing the results formally by way of Multi Objective Decision Analysis (MODA) each alternative design is compared consistently and decision making is simplified. Further, the utilization of defined interfaces and common architectures based on them enables alternative designs compliant with the Assessment Scenarios to be created quickly. By using Modelica as the modeling language the engineer benefits from accessibility to the rich simulation results of many variables and can compose System-Level Designs using extensive existing libraries.

However the approach and tool has further work to be done to it to make it more useful including: automation of the generation of alternative designs, support for parameter variation and implementation on a more complex design.

6 Conclusions

This paper has described an approach and tool for performing System-Level Design trade studies using Modelica. In the form of a Model Builder, Simulation Runner and Results Processor which take a suitable library and XML description file as input.

The aim of the approach was to consistently assess multiple design alternatives and summarize their results for fast comparison.

This was achieved by defining a common interface for the System of Interest (Solar-Boat) and placing it in an Assessment Scenario as a replaceable partial model into which programmatically, different SolarBoat alternatives were placed by the Model Builder. Each of these was then simulated and the results processed by Multi Objective Decision Analysis (MODA).

The rapid automated assessment of the alternatives and processing of results by MODA enables engineers to quickly understand the benefits of different designs, but by retaining the rich results associated with Modelica simulation further (manual) analysis can be performed to gain greater insight into how individual components are performing.

This was demonstrated for some simple examples of several Solar-Boat alternative System-Level Designs being subject to the same four different assessment scenarios.

References

- Cilli, M. V., & Parnell, G. S. (2014). Systems engineering tradeoff study process framework. In 24th INCOSE Int'l Symposium, Las Vegas, NV.
- DyMat 0.7 (2015), A package for reading and processing the result files of Dymola and OpenModelica. <https://pypi.python.org/pypi/DyMat>
- Edwards, S., Cilli, M. V., Peterson, T., Zabat, M., Lawton, C., & Shelton, L. (2015). Whole Systems Trade Analysis. In 25th INCOSE Int'l Symposium, Seattle. Seattle, WA, USA.
- INCOSE. (2015). Wiley: INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition - INCOSE.
- ISO. (2015). Automation systems and integration -- Object-Process Methodology (No. ISO/PAS 19450:2015).
- Parnell, G. S., Cilli, M. V., & Buede, D. (2014). Tradeoff Study Cascading Mistakes of Omission and Commission. INCOSE International Symposium, Las Vegas.
- SEBoK. (2015). Guide to the Systems Engineering Body of Knowledge (SEBoK).
- Sutherland, J., Kamiyama, H., Aoyama, K., & Oizumi, K. (2015). Systems Engineering and the V-Model: Lessons from an Autonomous Solar Powered Hydrofoil. Presented at the 12th International Marine Design Conference (IMDC), Tokyo Japan.
- Sutherland, J. (2016, March 4). Knowledge Management and System-Level Design Tools utilizing OPM and Modelica for a Student Solar-Boat Project (Master's Thesis). University of Tokyo, Tokyo Japan.
- Ulrich, K., & Eppinger, S. (2011). Product Design and Development (5 edition). New York: McGraw-Hill/Irwin.

FMI for Co-Simulation of Embedded Control Software

Nicolai Pedersen^{1,2} Tom Bojsen² Jan Madsen¹ Morten Vejlggaard-Laursen²

¹ Technical University of Denmark, Embedded Systems Engineering, Kgs. Lyngby DK-2800, Denmark ,
{nicp, jama}@dtu.dk

²MAN Diesel & Turbo, Teglholmsgade 41 Copenhagen DK-2450, Denmark,
{nicolai.pedersen, tom.bojsen, morten.laursen}@man.eu

Abstract

Increased complexity of cyber-physical systems within the maritime industry demands closer cooperation between engineering disciplines. The functional mockup interface (FMI) is an initiative aiding cross-discipline interaction by providing, a widely accepted, standard for model exchange and co-simulation. The standard is supported by a number of modelling tools. However, to implement it on an existing platform requires adaptation. This paper investigates how to adapt the software of an embedded control system to comply with the FMI for co-simulation standard. In particular, we suggest a way of advancing the clock of a real time operating system (RTOS), by overwriting the idle thread and waiting for a signal to start execution until return to idle. This approach ensures a deterministic and temporal execution of the simulation across multiple nodes. As proof of concept, a co-simulation is conducted, showing that the control system of an SCR (selective catalyst reduction) emission reduction system can be packed in a functional mockup unit (FMU) and co-simulated with a physical model, built in Ptolemy II. Results show that FMI can be used for co-simulation of an embedded SCR control software and for control software development. *Keywords:* Co-Simulation, RTOS, FMI, FMU, Embedded Systems

1 Introduction

Designing the next generation of embedded cyber-physical systems (CPS) requires close collaboration between physical model developers and the engineers implementing the computation, communication and control. The amount of sub-systems, deviation in the tool chain and standards are often barriers between these disciplines. Teams are divided into different departments within organisations or in cross-company collaborations, further complicating the cooperation. One of the recent initiatives to lower this barrier is the functional mockup interface (FMI) (Blochwitz et al., 2009). It is a tool-independent standard for model exchange and co-simulation. FMI was initiated by the automotive industry

and released in a version 1.0 in 2010 followed by a 2.0 version in 2014. This paper does not explain the standard, but aims to show the process of adapting an embedded system to comply with FMI. Implementing the FMI standard on an existing modelling platform is straightforward, especially since many of the open-source and commercial tools already support it. Forcing a specialised embedded system to comply is, however, a demanding task that requires adaptation.

At MAN Diesel & Turbo, legislation on pollution and a demand for support of alternative fuel types are increasing the amount of distributed sub-systems and the complexity of the traditional two-stroke diesel engine. The increased distributed complexity makes the cooperation between cyber and physical parts of the system even more crucial. Currently, simplified physical models are used for control algorithm development, and only estimations of the control system dynamics is considered when modelling the physical behaviour. The objective of this project is to enhance the modelling development and distribution at MAN Diesel & Turbo by introducing a more comprehensive system simulation. We wish to simulate both physical behaviour and control dynamics, combined with a model of the software. The software model will enable us to investigate system behaviour such as alarm handling, IO scaling and network communication/protocols. The main challenge is to adapt the embedded engine control system into a functional mockup unit (FMU). The process of this adaptation is what will be presented in this paper. As use case, a simple model of the SCR (Selective Catalyst Reduction) emission reduction system and its control software will be co-simulated.

FMI 2.0 for co-simulation has been chosen due to its strict type/execution structure combined with its freedom of implementation. The standard is highly recognised and applied within the automotive industry (Abel et al., 2012; Stoermer and Tibba, 2014), which has many similarities with the maritime. Recently, applications within energy and grid systems (Vanfretti et al., 2014; Elsheikh et al., 2013) and HVAC systems (Nouidui et al., 2014) are emerging as well. FMI applications within the maritime industry, like this, is limited (Pedersen et al., 2015).

This project uses the heterogeneous simulation software framework Ptolemy II (Liu et al., 2001; Brooks et al., 2010) to co-simulate a simple physical model with an imported FMU. Ptolemy II has been used for various FMI applications (Broman et al., 2013; Liu et al., 2001; Lee et al., 2015) Much attention has been put on implementing the standard, such as FMI++ (Widl et al., 2013) the FMI Library from (Modelon) and the FMU SDK from (QTronic). Examples of how to build an FMI master algorithm has been provided as well (Bastian et al., 2011; Broman et al., 2013). In (Bertsch et al., 2015) a prototypical realisation of an FMU executing on a Bosch electronic control unit was presented. However, the non-trivial process of adapting the software of an embedded system, with at real-time operating system (RTOS), into a co-simulation FMU, has not yet been described, but will be in this paper.

First the cyber-physical system at hand will be introduced in Section 2. Section 3 shows how to move from a target embedded application to an FMU running in a regular Linux environment. A use-case implementation is presented in Section 4 and conclusions are drawn in Section 5

2 Cyber-Physical System

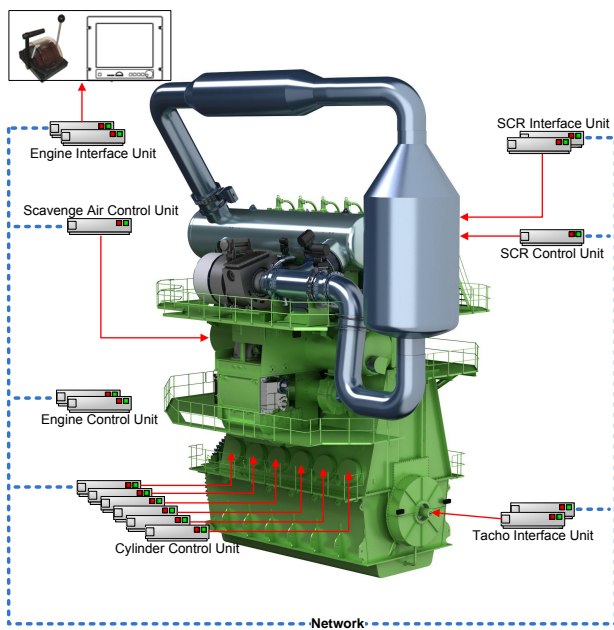


Figure 1. An MAN Diesel & Turbo two-stroke low-speed diesel engine with the SCR and the engine control system illustrated

MAN Diesel & Turbo designs large-bore diesel engines and turbomachinery for marine propulsion systems and stationary applications, such as power plants. With the introduction of the electronically controlled line of ME engines in 2002, MAN Diesel & Turbo moved into the development of Cyber-Physical System. In recent

years, the demand for new emission reduction systems and alternative fuel types have made the core engine even more dependent on the surrounding control system. This dependency demands a more advanced simulation environment including co-simulation. The engine control system consists of numerous distributed controllers with each their specific control objective connected by a wired network. Figure 1 illustrates a 6-cylinder two-stroke ME-engine with an SCR system and engine control system. The main controllers are the engine interface units communicating with the operator, and the scavenge air control unit ensuring that pressures are balanced between the turbocharger and scavenging. The engine control units ensure that the cylinder control units perform the correct temporal injection ect. according to the information about the crankshaft position from the tacho interface units. Finally, if the engine is fitted with an auxiliary system e.g. an SCR system, it will be controlled and monitored by its own SCR units.

3 From Embedded Target to FMU



Figure 2. A multi-purpose controller of the MAN Diesel & Turbo engine control system

To achieve the objective of co-simulating the software control system together with a physical model, in a different environment(Ptolemy II), we need to make our target application code run in a functional mockup unit 0(FMU). It should be noted that the main objection of this solution is to aid physical modelling and control algorithm development. The solution will therefore demonstrate a deterministic simulation of both computational execution and network. Despite the previously described system behaviour investigation benefits, of including a software model in the FMU, the decision is also based on future ambitions and the current control system development at MAN Diesel & Turbo. Future

plans include a stochastic network model and HIL-nodes combined with FMI nodes.

3.1 Configuration Abstractions

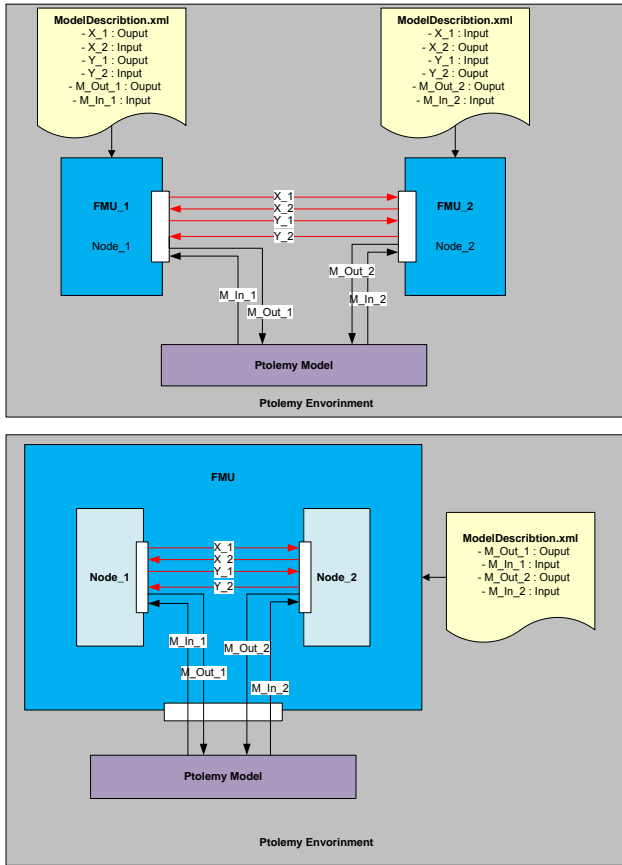


Figure 3. It is possible to change the level of configuration complexity exposed to the user. The top figure shows how each control system node can be packed in an FMU for maximal configuration flexibility. The bottom figure shows how multiple nodes can be packed and configured in a single FMU for a simpler user configuration setup.

One of the most important concerns when introducing FMI was the configuration complexity. The system is to be used by different disciplines, and it is important that the configuration level can be abstracted to fit the user objective - meaning that if a hydraulic engineer wishes to investigate the dynamic effects of the control system on his model, he should not have to connect all the wires of the control system to get started, but rather have one FMU with only relevant variables and parameters exposed. We found it beneficial to maintain the possibility of interconnecting multiple nodes of the control system before wrapping them into the functional mockup interface. As shown in Figure 3, this allows for different levels of configuration complexity. If we are interested in both the interaction between two nodes and a physical model, we can provide all variables, parameters and IOs through multiple FMUs and connect them in our environment, see top Figure 3. However, if we are only

interested in the variables interacting with our external model, it is possible to connect the nodes internally, and only expose the relevant variables, bottom figure 3. The latter option provides a much simpler configuration and "ModelDescription.xml" for the user and lets the control system experts ensure that nodes are connected correctly.

3.2 Target to PC simulation

The target controllers used are multi-purpose, meaning e.g. that cylinders and SCR-control units are identical. The only deviation determining the specific controller objective is the software executed on the embedded system. A controller interfaces with sensors and other computational units, using the information to interact with the system through actuators. A controller contains a CPU module with an FPGA-based embedded system utilising a real-time operating system. The strategy for simulating our embedded system is to model the entire embedded system from the operating system and up, wrapping this into an FMU. Conclusively, our model is not simulating the behaviour of the embedded processor, but builds the target code for an x86 architecture in a so called PC-simulation application (PCSIM).

3.3 FMI implementation of PC simulation

To implement FMI 2.0 for co-simulation, we need further access to some main functionality embedded in the PCSIM. Looking at the FMI co-simulation state machine (Blochwitz et al., 2009), we need to access relevant data for *fmi2Set()* and *fmi2Get()* and a way of stepping the simulation according to the *fmi2DoStep()* function. Furthermore, the network communication is to be reconnected and the FMI functions implemented.

3.3.1 Hook to OS clock

For the co-simulation to work correctly, we need to control the execution between the discrete communication points on each node. The approach is to access the clock of the operating system and let a simulation manager control the temporal execution. This is made possible by building the target code as a shared library and overwriting the idle thread method of the RTOS. The RTOS used in this project supports an x86 architecture and provides the board support package, which includes a *bsp_idle_thread* to be manipulated. The solution proposed will require customisation to work with different RTOS versions, however, the concept is generic. Besides the idle thread hook, we need to be able to start and stop the application by calling the main function through the library. The main function is executed in a separate thread until we force it to stop, having the main function return. The new idle thread function has an idle callback function that implements ticking of the RTOS clock. Each tick lasts for a simulated 1 ms, implemented

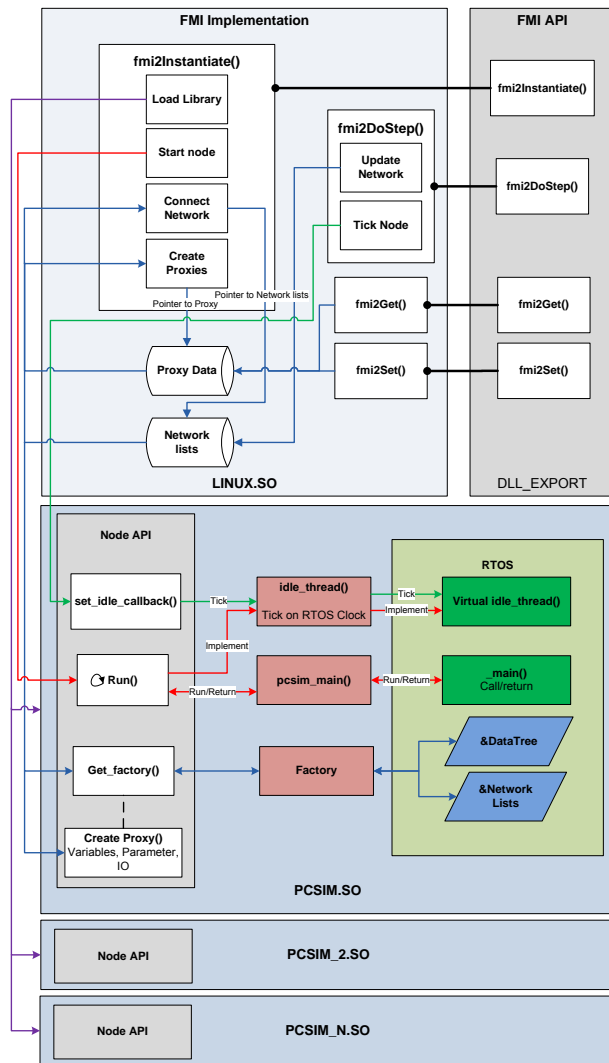


Figure 4. The implementation of FMI on the MAN engine control system

by assuming unlimited CPU power - thus an execution time of zero for every node, followed by a 1 ms delay. A node will run until it returns to idle, meaning for every tick, all task will finish and never be interrupted. This guaranties a common perception of time across nodes. The assumption of unlimited processing power will obviously make the simulation results deviate slightly from a real stochastic execution. However, it ensures a determinism which is important during control algorithm development and regression testing. All interrupts are currently software simulated and scheduled as regular tasks. Further work will aim to implement a more temporal scheduling of especially high frequency interrupts.

Having a hook to the clock and a joint time perception makes it possible for a manager to call the *fmi2DoStep()* function and orchestra a correct temporal execution of the co-simulation.

3.3.2 Connecting variables, parameters and IO channels

On the target application all variable, parameters and IO channels are organised in a component-oriented data tree structure with unique IDs. Using a factory method design, we make it possible to create proxies for both variables, parameters and IO channels, providing a *Proxy.Get()* and *Proxy.Set()* function that will effect the source on the specific node. For IO channels, we communicate on micro-ampere level, so proper conversion is needed.

The *fmi2Set()* and *fmi2Get()* functions will write and return the value of the proxies. The instantiation of proxies are done in the *fmi2Instantiate()* function and is based on the "ModelDescription.xml". One of the advantages of FMI is the strict data type definition. However, the target application utilises more data types than the ones allowed by FMI, such as fix-point and unsigned short. As a result, a type conversion layer had to be added.

3.3.3 Solving network communication

To simulate the network communication between nodes, we replace the RTOS network driver with a deterministic input/output queue implementation. Each node is given an address corresponding to the unique *node_id* already provided by the controller. Through the factory design from 3.3.2 input and output lists are made available across nodes. A network manager then redirects packages from output to input queues according to network address. The network manager support both unicast, multicast and broadcast. Communication is done at every discrete communication point, and the network driver is activated every ms tick of the OS clock, if any data is available in the input or output queue. Currently, the network is only available with interconnected nodes and not as an output through the FMU. However, this is something we are working on.

3.3.4 FMI implementation

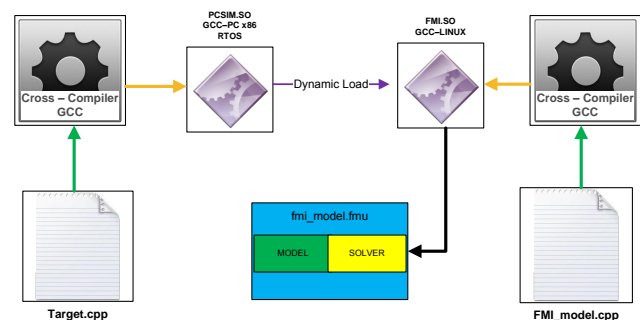


Figure 5. The compiling routine from target to functional mock-up unit.

According to the FMI standard the application should be compiled into a shared library with the FMI functions exported. As described, we are able to build each of our control nodes into PC shared libraries (PCSIM.so) including a, x86 RTOS. We now need to wrap these into a Linux shared library (FMI.so) implementing the FMI application interface. One or more PCSIM.so are loaded into the FMI.so which is the main binary in the co-simulation FMU, see figure 5. A MAN Diesel & Turbo FMU will have the 2.0 FMI for Co-simulation API. The *fmi2Instantiate()* will load the PCSIM.so's required for the specific scenario and create the relevant parameters, inputs and outputs according to the *ModelDescription.xml* and start each node executing. The *fmi2DoStep()* is able to call the idle callback function on each node, signalling the idle thread to tick the RTOS. If an FMU contains more than one node the network will be updated at every communication point. The remaining FMI functions have been implemented but not illustrated in Figure 4.

4 Use Case: SCR Temperature Dynamics and Control

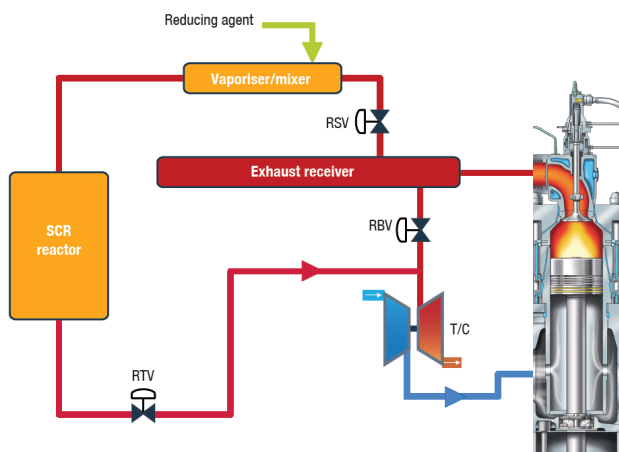


Figure 6. Diagram of the SCR system

As a simple use case, we look at the dynamics and control of heating up the SCR reactor. When a vessel is to comply with the Tier III emission limits (IMO, 2008) for NO_x reduction, a command is sent from the operator to activate the SCR control. The SCR control unit will then redirect the exhaust gas through the reactor by opening the reactor sealing valve (RSV) and the reactor throttle valve (RTV). The controller has to balance the RTV opening, to ensure that the flow to the turbine inlet of the turbocharger is sufficient. As soon as the reactor is properly heated, the reactor bypass valve (RBV) can be closed; consequently, only cleaned air from the reactor leaves the system as exhaust. A diagram of the SCR control is illustrated in Figure 6. The SCR controller uses the difference between the reactor input and

output temperature as a reference residual signal for controlling the position of the RTV valve. By modelling the time delay of heating the reactor and passing the resulting output temperature back to the SCR controller, we will show that it is possible to investigate the dynamic interaction between a physical model and the actual control software.

Many additional observations regarding the engine physics are required for all aspects of the SCR controller to perform correctly. An advantage of being able to connect more nodes within a single FMU is that the so-called engine simulation unit (ESU) used for hardware in the loop test can be included. The ESU contains numerous physical models executing within the embedded controller environment. Model execution on the ESU must comply with real-time requirements and should therefore not be too complex. With FMI, it is possible to make a hybrid simulation of the engine physics where ESU models can be combined with Ptolemy models. In this use case, the reactor heating model provides physical insight into the SCR controller together with the ESU.

4.1 SCR Heating Model

The reactor heating model chosen as proof of concept is described below. The output temperature can be modelled as the relationship between the RTV position, the flow through the reactor and the input temperature, resulting in two low-pass filters with a significant time constant. The inputs to the model is provided by the SCR controller and ESU.

The mass flow into the reactor \dot{M} is estimated from the engine load L .

$$\dot{M}_n = \dot{M}_{n-1} + \frac{L - \dot{M}_{n-1}}{1 + \tau_{Scavenge} \cdot T} \quad (1)$$

where T is the sampling frequency.

The time constant of the reactor output temperature, is estimated as the RTV valve opening with the mass flow plus a time constant, converted into seconds.

$$\tau_{out} = (\dot{M}_n \cdot RTV + \tau_{reactor}) \cdot 3600 \quad (2)$$

Finally, the output temperature is calculated as

$$T_{out_n} = T_{out_{n-1}} + \frac{T_{in} + T_{out_{n-1}}}{1 + \tau_{out} \cdot T} \quad (3)$$

This is naturally a simplified approach, however, it goes to show, that it is possible to distribute the control system and co-simulate with other thermodynamic models regardless of the abstraction level.

4.2 Ptolemy II as simulation framework

As simulation framework, the open-source Ptolemy II was chosen due to its heterogeneous actor-oriented

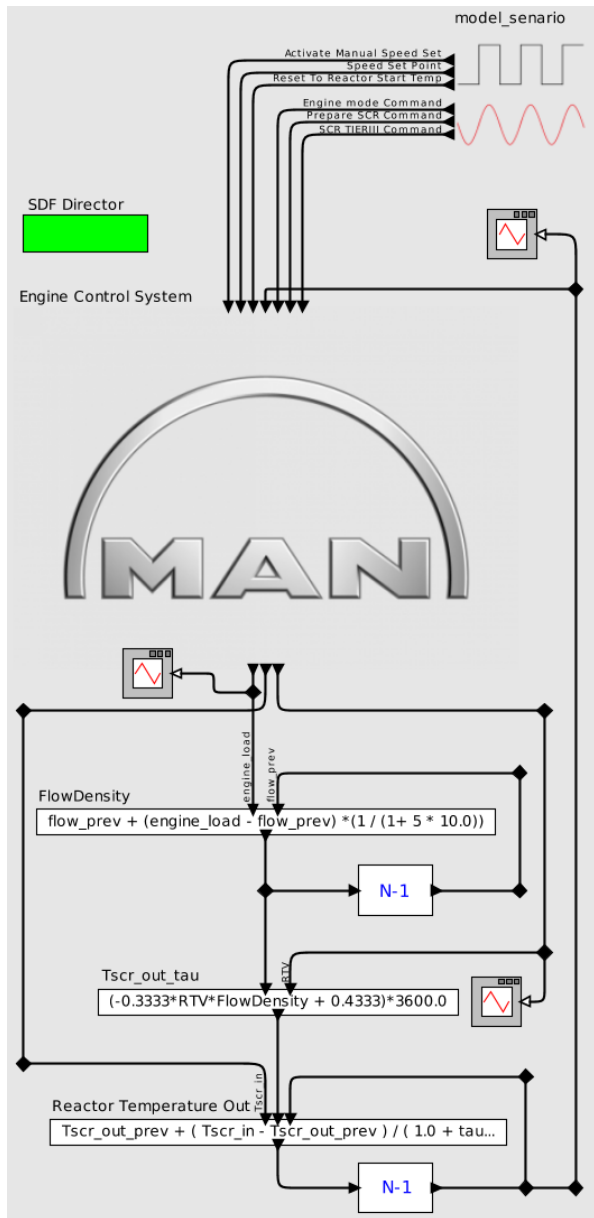


Figure 7. FMU import in Ptolemy II and simple physical model implementation

design and comprehensive support for different software components and the FMI interface as described in (Broman et al., 2013). The FMU is imported as a co-simulation actor automatically configured by the "ModelDescription.xml". Using "Vergil", the graphical user interface shipped with Ptolemy, the equations from 4.1 are created and connected to the FMU outputs. A simulation scenario is likewise defined in Vergil and connected to the input ports of the FMU, see Figure 7. The scenario sets a reactor start temperature and an engine speed set point. After 700 seconds, a simulated bridge command is send to the SCR controller, activating the SCR control strategy.

To execute the simulation, a synchronous dataflow (SDF) director was chosen. The SDF director is appro-

priate because we have a predictable and regular execution (firing) of the FMU. At regular communication points, inputs/outputs are updated in a predefined order.

4.3 Results

binaries	1 item
linux32	1 item
model.so	2.4 MB
resources	2 items
lib	4 items
esu_target.so	47.5 MB
scrcu_target.so	58.8 MB
scri1_target.so	51.1 MB
scri2_target.so	50.4 MB
par	4 items
esu.manbw-paf	285.9 kB
scrcu.manbw-paf	179.5 kB
scri1.manbw-paf	44.3 kB
scri2.manbw-paf	23.9 kB
model.png	75.7 kB
modelDescription.xml	2.2 kB

Figure 8. The use-case example of a functional mock-up unit containing the MAN SCR control nodes

To run the simulation, an FMU was build as seen in Figure 8. Here four PCSIM.so corresponding to the code of four embedded controllers, are packet into "resources/lib". The engine simulation unit (*esu_target.so*) models the entire engine, except the SCR heating model, using the target solver ect. An SCR Control Unit (*scrcu_target.so*) containing all the control algorithms for the reactor control and two SCR interface controllers (*scri1_target.so*, *scri2_target.so*) redirecting all the sensor values connected as simulated cables from the ESU to the SCRCU by network. Configuration of the PCSIM applications are provided via the MAN parameter files located at "resources/par"

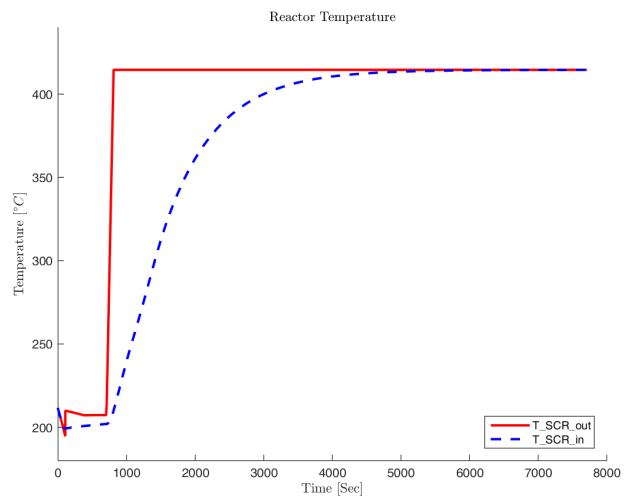


Figure 9. The in- and output temperature of the simulated SCR heating

The simulation of the FMU and reactor heating model is presented in Figure 9. Here we see that the SCR reactor out temperature start to increase after 700 seconds when the SCR start command is sent. The heating has the expected low-pass behaviour and takes approximately 1.5 hours to heat up.

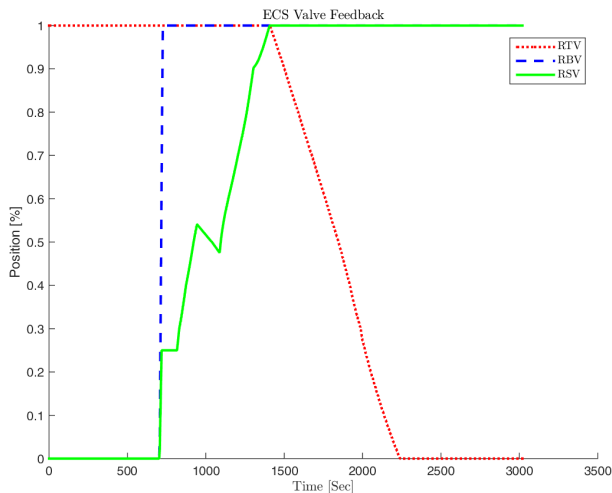


Figure 10. Valve feedback from the SCR simulation

In Figure 10, we clearly see that the SCR control works as intended, even though we have replaced the SCR heating model from the original ESU and replaced it by a Ptolemy implementation. As soon as the SCR activation occurs, the RTV and RSV valves start to open. The RTV valve is clearly controlled to balance the flow to the turbocharger. This actuation is filtered from the temperature by the low-pass behaviour of the reactor, as expected. As soon as the RTV valve is fully open the RBV valve can be closed, and output temperature keeps increasing until it eventually reaches the inlet temperature.

Each node in the simulation executes an application task running on top of the RTOS, updating variables at a specific sampling frequency. From Figure 11, we clearly see how the SCR control unit runs at 5 Hz and the engine control unit at 10 Hz. The SCR temperature is calculated in Ptolemy, resulting in the same frequency as the simulation time step of 1 ms.

5 Conclusion

This paper showed the non-trivial process of implementing FMI for co-simulation of an embedded system. We proposed to compile a target platform RTOS into an x86 architecture, which most RTOS systems support. By replacing the idle thread of the RTOS, a hook for the system clock can be provided and used to advance through the application. To match the "Get()/Set()" structure of the standard, the same was implemented through sim-

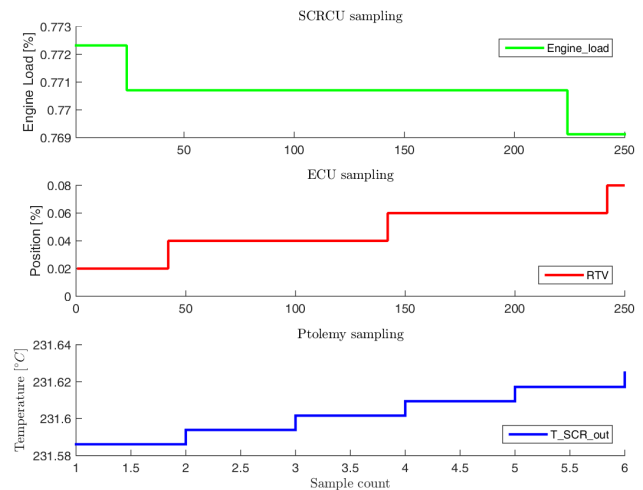


Figure 11. Illustration of the different sub-system sampling frequencies

ulation proxies identified by unique ID numbers of target variables. The FMI API is wrapped around the x86 RTOS by loading it as a shared library, with the FMI step function "fmi2DoStep()" activate the RTOS clock through a callback function. The configuration of an entire control system results in a vast amount of connections, not necessary relevant for all modelling purposes. One of the advantages of the proposed method is that the configuration abstraction can be varied. If relevant, each node of the control system can be packed in individual FMUs, or all nodes can be enclosed in a single FMU, with all configuration and data/network exchange done internally. We have provided a use case where part of the engine control system is packed in an FMU and imported into Ptolemy II. By connecting the FMU to a physical model, we proved that the system could be co-simulated with an external tool, resulting in correct control system behaviour.

References

- Andreas Abel, Torsten Blochwitz, Alexander Eichberger, Peter Hamann, and Udo Rein. Functional mock-up interface in mechatronic gearshift simulation for commercial vehicles. *9th Int. Model. Conf.*, pages 775–780, 2012. doi:10.3384/ecp12076775.
- Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. Master for Co-Simulation Using FMI. *8th Int. Model. Conf. 2011*, pages 115–120, 2011. doi:10.3384/ecp11063115.
- Christian Bertsch, Jonathan Neudorfer, Elmar Ahle, Siva Sankar Arumugham, Karthikeyan Ramachandran, and Andreas Thuy. FMI for Physical Models on Automotive Embedded Targets. *Proc. 11th Int. Model. Conf.*, pages 43–50, 2015. doi:10.3384/ecp1511843.

- T Blochwitz, M Otter, M Arnold, C Bausch, C Clauß, H Elmqvist, A Junghanns, J Mauss, M Monteiro, T Neidhold, D Neumerkel, H Olsson, J V Peetz, and S Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *8th Int. Model. Conf. 2011*, pages 173–184, 2009. doi:10.3384/ecp12076173.
- David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of FMUs for co-simulation. *2013 Proc. Int. Conf. Embed. Software, EMSOFT 2013*, 2013. doi:10.1109/EMSOFT.2013.6658580.
- Christopher Brooks, Edward A Lee, and Stavros Tripakis. Exploring Models of Computation with Ptolemy II. *10 Proc. eighth IEEE/ACM/IFIP Int. Conf. Hardware/software code-sign Syst. Synth.*, pages 331–332, 2010.
- Atiyah Elsheikh, Muhammed Usman Awais, Edmund Widl, and Peter Palensky. Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. *2013 Work. Model. Simul. Cyber-Physical Energy Syst. MSCPES 2013*, pages 1–6, 2013. doi:10.1109/MSCPES.2013.6623315.
- IMO. MARPOL : Annex VI and NTC 2008 with guidelines for implementation. Technical report, 2008.
- Edward A. Lee, Mehrdad Niknami, Thierry S. Noudui, and Micheal Wetter. Modeling and Simulating Cyber-Physical Systems. *2015 Int. Conf. Embed. Softw.*, pages 115–124, 2015. doi:doi: 10.1109/EMSOFT.2015.7318266.
- Jie Liu, Xiaojun Liu, and Edward A Lee. Modeling Distributed Hybrid Systems in Ptolemy II. *Proc. 2001 Am. Control Conf.*, 6:4984–4985, 2001. doi:10.1109/ACC.2001.945773.
- Modelon. FMI Library. URL <http://www.jmodelica.org/FMILibrary>.
- Thierry Noudui, Michael Wetter, and Wangda Zuo. Functional mock-up unit for co-simulation import in Energy-Plus. *J. Build. Perform. Simul.*, 7(3):192–202, 2014. doi:10.1080/19401493.2013.808265.
- Nicolai Pedersen, Jan Madsen, and Morten Vejlggaard-Laursen. Co-Simulation of Distributed Engine Control System and Network Model using FMI and SCNSL. *10th IFAC Conf. Manoeuvring Control Mar. Cr. MCMC 2015*, 48(16):261–266, 2015. doi:10.1016/j.ifacol.2015.10.290.
- QTronic. FMU SDK. URL <https://www.qtronic.de/en/fmusdk.html>.
- Christoph Stoermer and Ghizlane Tibba. Powertrain Co-Simulation using AUTOSAR and the Functional Mockup Interface standard. *Proc. 51st Annu. Des. Autom. Conf. Des. Autom. Conf. - DAC '14*, (March):1–1, 2014. doi:10.1145/2593069.2602975.
- Luigi Vanfretti, Tetiana Bogodorova, and Maxime Baudette. Power system model identification exploiting the Modelica language and FMI technologies. *2014 IEEE Int. Conf. Intell. Energy Power Syst. IEPS 2014 - Conf. Proc.*, pages 127–132, 2014. doi:10.1109/IEPS.2014.6874164.
- Edmund Widl, Wolfgang Muller, Atiyah Elsheikh, Matthias Hortenhuber, and Peter Palensky. The FMI++ library: A high-level utility package for FMI for model exchange. *2013 Work. Model. Simul. Cyber-Physical Energy Syst. MSCPES 2013*, 2013. doi:10.1109/MSCPES.2013.6623316.

Deployment of high-fidelity vehicle models for accurate real-time simulation

Johan Andreasson¹ Naoya Machida² Masashi Tsushima² John Griffin³ Peter Sundström⁴

¹ Modelon KK, Japan

² Nissan Motor Co., Japan

³ Modelon Inc., USA

⁴ Modelon AB, Sweden

johan.andreasson@modelon.com nao-machida@mail.nissan.co.jp

masashi-tsushima@mail.nissan.co.jp john.griffin@modelon.com peter.sundstrom@modelon.com

Abstract

In the effort to shorten development cycles and with the reduced ability to test in real life, driver-in-the-loop simulators are increasingly used by automotive OEMs and in Motorsports to enable engineers and drivers to experience a new vehicle design in a realistic environment before it is built. With the right level of accuracy, the same model can be applied in other real-time vehicle dynamics applications to allow for testing and verification in the development of new vehicle functions.

This paper gives an overview of the requirements for automotive real-time application and the solution chosen. Emphasis is given on the model definition and real-time configuration as well as parameterization from existing data sources and integration of third party subsystem models.

Keywords: vehicle simulators, vehicle dynamics, real-time, hardware-in-the-loop, driver-in-the-loop

1 Introduction

Virtual representations that can predict a vehicle's real life behavior have become more and more important in the development process. There are some well-known reasons for this, such as overcoming the cost, time, safety and repeatability issues with physical prototypes (Rauh, 2003).

Recently, deployment of real-time vehicle dynamics simulation for hardware-in-the-loop testing and driving simulators has gained significant attention (Yasuno, 2014). This trend is largely driven by the demand for shorter development cycles that also should result in better products.

Figure 1 illustrates three typical applications that drive the use of real-time models. The ability to per-

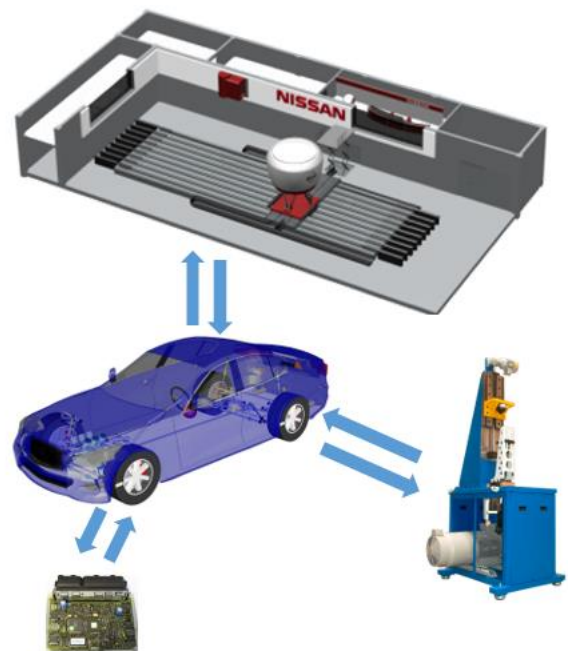


Figure 1. Applications of real-time capable models: Driver-in-the-loop simulator, Yasuno (2014) (top), ECUs (bottom left) and component hardware testing (bottom right).

form early assessment of drivers' perception of the vehicle being designed is one of the key driving factors (top).

Another common use is for the integration of Electronic Control Units (ECUs) from suppliers where the actual implementation is often hidden or partly hidden as a so called black box (bottom left). A third case is to include parts of the vehicle as hardware to provide realistic boundary conditions for component testing (bottom right).

Ultimately, all these applications can be combined, with the virtual vehicle representation as the glue to be able to assess the complete driver-vehicle system at any point during the development process. Here, we

try to give a brief overview of the requirements a modern engineering tool chain should meet and pointers to where in the paper the topics are addressed.

1.1 Accuracy and real-time capability

The late testing, tuning work and potential design iterations associated with physical prototypes are known to not just be expensive, but also to extend the development time with associated risks to delay market introduction. To be able to reduce physical testing further, the required simulation accuracy increases, which in turn typically increases the computational effort which is in conflict with real-time requirements.

For vehicle dynamics, multi-body representations of the vehicle mechanics are a standard approach in the automotive industry as they are able to capture the relevant phenomena for vehicle dynamics, while providing a straight-forward parameterization.

For motorsports applications, the vehicle chassis is traditionally close to an ideal kinematic behavior. As such, the number of degrees-of-freedom (DOF) can be kept to around 30-50. Such multi-body representations have been used successfully in real-time applications for a long time. Applications include driver training, driver perception evaluation of new designs and complete driver-vehicle systems integration, see e.g. (Toso, 2014).

For passenger vehicles the vehicle design is generally different, especially due to the many elastic elements that are used to enhance comfort and tune vehicle attributes. This leads to more complex representations with typically 150-300 DOF that traditionally were not possible to execute in real-time. Therefore, the current industry practice is to use a vehicle model with significantly reduced complexity that results in a lower computational load. Unfortunately, this means lower accuracy, a limited valid frequency range and additional pre-processing of the model to generate the required parameterization, which slows down iteration time.

With the introduction of new technology for parallelizing vehicle models (Andreasson et al., 2014) and (Elmqvist et al., 2014), the high model fidelity used offline (see Section 2) is made executable in real-time applications, as explained further in Section 5.

1.2 Inter-operability

In the context of shorter lead-times, no tool can be an island, they must connect to form an efficient tool chain. With the amount of legacy tools and methods used by OEMs today, this puts some additional requirements on any new model:

1. The model must be able to share data, meaning reading and writing existing formats.
2. It must also be possible to plug the model into the tool chain in such a way that it can use existing tools for pre- and post-processing.
3. Additionally, subsystem models from different sources need to be included in the complete vehicle model to allow for incremental improvement, multi-fidelity and use of heritage/legacy code.

These topics are further described in Sections 3 and 4.

2 Model overview

The vehicle in this example is a production vehicle featuring a front double wishbone suspension, a rear multi-link suspension, front engine, rear wheel drive, and an automatic transmission, Figure 2. The models are based on the (Vehicle Dynamics Library, 2015) and largely implements the template and interface structure provided, Figure 3.

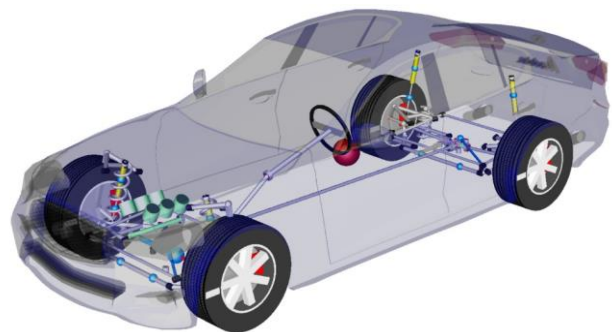


Figure 2 Vehicle model used in the presented work.

For each component or sub-system, the vehicle model can be reconfigured by replacing subsystems with plug-in compatible variants. This allow for changing both configuration (e.g. from automatic to manual transmission) and fidelity (e.g. from multi-body to tabular suspension).

For each subsystem or component, there is a well-defined interface that specifies the boundaries and provides an established framework to connect pieces from existing various simulation tools as described in the next section.

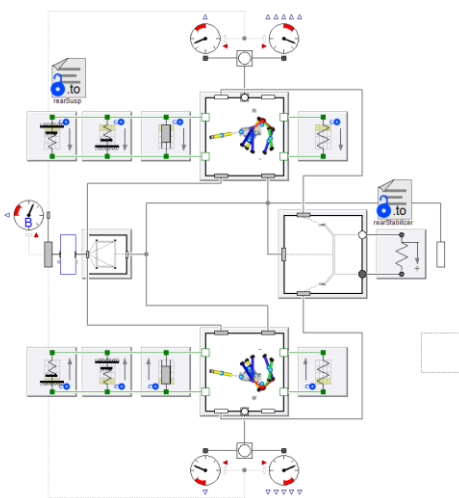
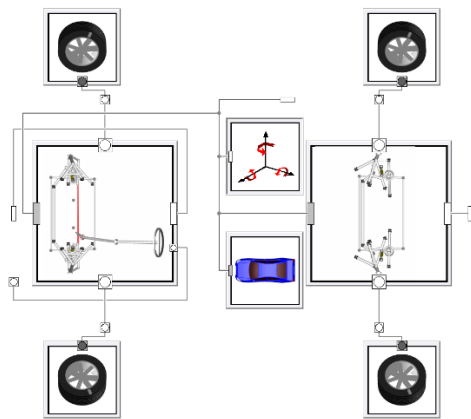
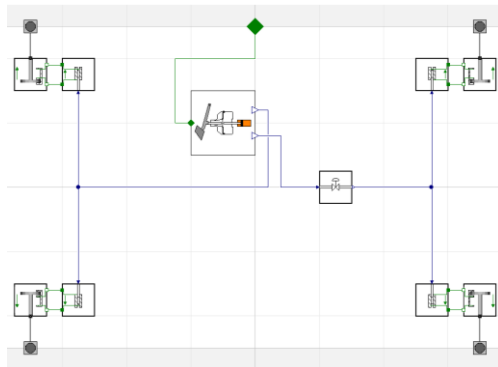
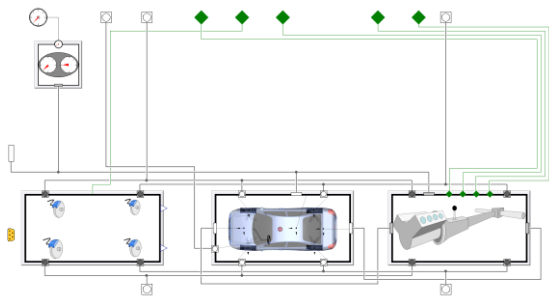


Figure 3. Architecture of the complete vehicle model. Vehicle (top), brake system (second), chassis (third) and rear suspension (bottom).

3 Deployment

To ensure convenient interoperability with subsystem models from different sources, these models connect either from within the Modelica framework or from the tools originally used. This section illustrates the two cases as well as the final integration of the complete real-time model.

3.1 Multi-physics brake model

To include higher detail in the brake simulation and get more realistic brake pedal feel, the pneumatic vacuum booster and the hydraulic master cylinder are modelled as physical components (Hydraulics Library, 2015) and (Pneumatics Library, 2015). Figure 4 shows the layout of the brake system model.

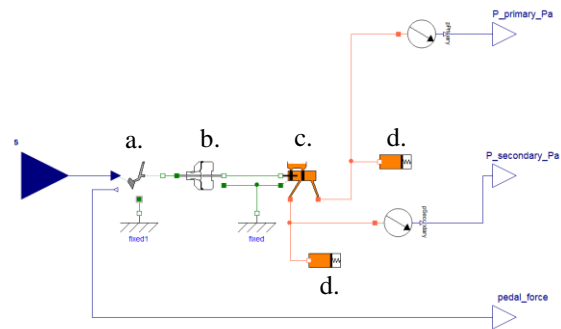


Figure 4. Diagram layer of the brake system model. Components are; pedal actuator (a), vacuum booster (b), master cylinder (c) and brake lines and calipers (d).

The core of the booster model is a double acting pneumatic cylinder corresponding to the booster diaphragm. Based on the pedal actuating the poppet valves, vacuum and atmosphere pressure is applied to the different sides of the diaphragm to boost the pedal force acting on the master cylinder piston.

The force characteristics of the vacuum booster are affected by elasticities and gaps in the mechanisms that open and close the valves to the diaphragm. High stiffness in combination with low mass in these components can result in fast dynamics. This is typically not relevant for the application, and may also be too fast for the desired integrator time step. To handle this, the spring-mass combinations are replaced with elements where the bandwidth can be explicitly defined.

The master cylinder is a two circuit variant with a mechanical gap connecting the two cylinders. As long as the first circuit is pressurized, the gap is open and the hydraulic pressure also activates the second cylinder. If the pressure in the first circuit is lost, the gap will close so the pedal force is still transferred to the second cylinder though with changed pedal travel.

To show the more detailed function of the booster in combination with the hydraulic system can give a more refined accurate prediction of the driver's perception, Figure 5 illustrates how the brake system responds to a press-and-release cycle of the pedal force. The upper plot shows the brake line pressure which also will affect the actual and experienced retardation generated. The lower plot shows the pedal stroke which provides the driver with feedback through the foot.

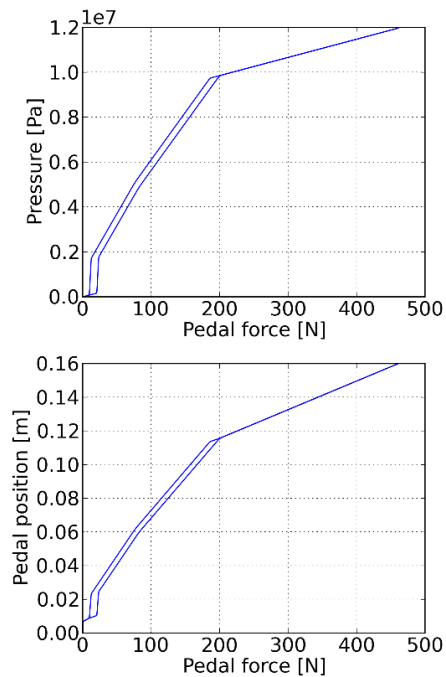


Figure 5. Top plot: Brake line pressure (vertical axis) as a function of brake pedal force (horizontal axis). This shows the knee points where the booster engages and disengages. Bottom plot: Corresponding pedal travel (vertical axis) for the same pedal force excitation. For confidentiality reasons, the numerical values have been removed.

3.2 External steering model

External subsystem models not implemented in Modelica can either be imported into the Modelica framework, or the vehicle model is exported without the corresponding subsystems for integration on an external platform.

External models are brought into the Modelica model using external functions or objects, or to include so called Functional Mock-up Units (FMUs) adhering to the Functional Mock-up Interface (FMI, 2015). In either case, these are wrapped into the subsystem interfaces to ensure plug-in compatibility. A variety of external models, including (OpenCRG, 2015), (DelftTyre, 2015) and (FTire, 2015) are predefined and ready to use.

Correspondingly, when exporting the vehicle model, the systems that should be external are replaced with models that provide no contents. Figure 6 illustrates a Modelica steering system model (top), an external steering model included in the vehicle model (middle) and an empty steering model with external inputs and outputs (bottom).

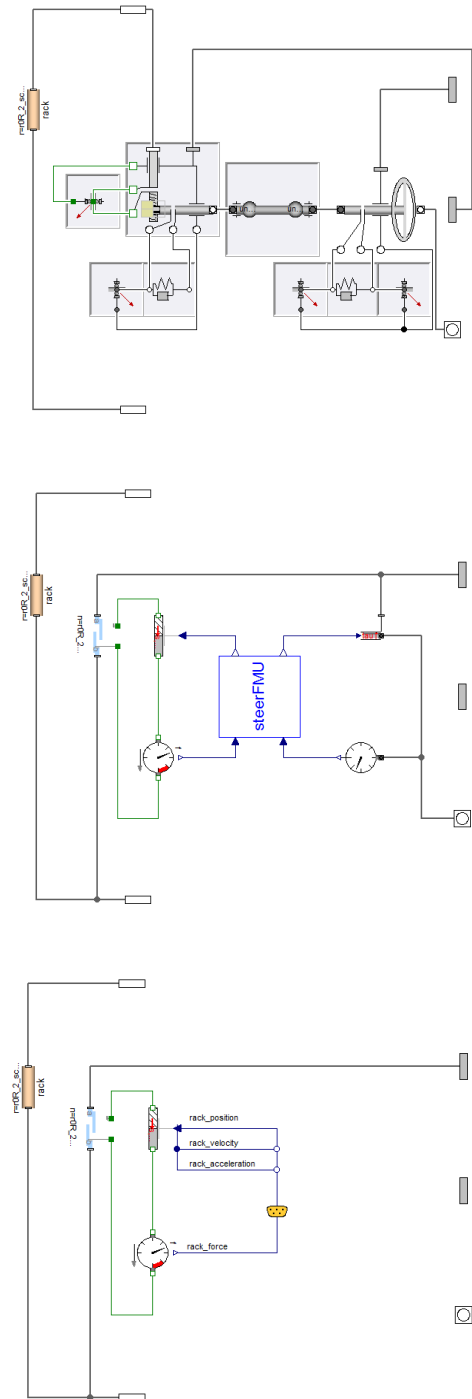


Figure 6. Modelica steering model (top), steering model imported as and FMU (middle), empty steering model requiring rack motion and providing rack force.

3.3 Complete real-time model integration

After configuring the vehicle model to allow for external subsystems, it can be exported in different formats (such as FMI, C, etc.) for system integration elsewhere. For deployment, the system integration is currently done in SIMulationWorkbench (Concurrent, 2015).

The system architecture is shown in Figure 7. The shaded area indicates what is handled by the real-time host. It contains the models (2) of chassis (2a), brake system (2b), steering (2c), power train (2d), controllers (2e) and I/O devices (3) for interaction with the motion platform (3a) and driver interface (3b).

The Communications between the models (2) and the I/O devices (3) is handled through shared memory interconnection (1) called Real Time Data Base (RTDB), Concurrent (2015). RTDB allows access to any variables stored in RTDB for any of the included models and is achieved to wrap the inputs and outputs of each of the model with dedicated read/write functionality.

The real-time host then communicates with the operators through a set of clients that provides a Graphical User Interface (GUI) for configuration of I/O (5a), data recording (5b), playback (5c) and configuration of RTDB (5d).

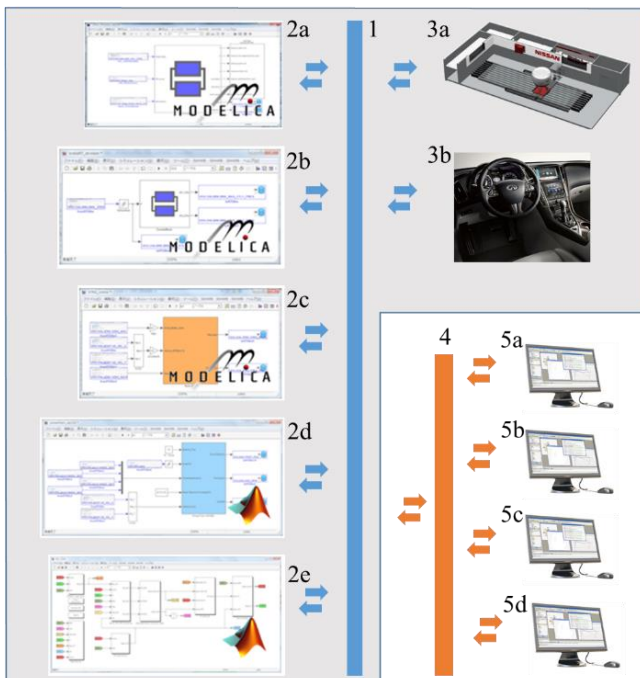


Figure 7. Software architecture of the real-time simulation host.

To allow for communication through the RTDB, each model must contain its own numerical integrator and

can only expect to share information at each communication point. So, effectively, the real-time host works like a co-simulation platform. A benefit with this approach is the ability to let each model integrate at multiples of the communication interval, so called multi-rate integration.

Multi-rate integration is suitable as it allows for models from tools that only supports explicit integration schemes to be executed at small enough time step to ensure numerical stability. Correspondingly, the method allows for plant models to be integrated at shorter time steps than the controllers.

With co-simulation, there is also inherent support to distribute the integration of each model to its own set of cores. Additionally, the chassis model is parallelized on multiple cores as described further in Section 5.

4 Parameterization

For the real-time model to be an efficient tool, it is crucial that it can access the latest state of development. Therefore, the model is designed to read the same data set that is used in existing offline tools, in this case the TeimOrbit format.

4.1 Accessing data

The data management is accomplished using a general-purpose data management method to read and write external data called DataAccess. This method can handle a variety of different file formats such as .xml, .json and .mat. Since DataAccess is compiled into the simulation code, it is well suited for model export as it allows users to conveniently change model parameters in a consistent way regardless of how and to what format the model is exported, Figure 8.

For this work, the handling of the TeimOrbit format was added to allow data sharing with the offline tools. Initially, the real-time model was parameterized by manually accessing a data value required for a particular attribute, such as the mass of a part. This resulted in significant effort and duplication of code to read the necessary data. As the project progressed, it became evident that duplicate coding could be eliminated by creating data-aware components, Figure 9.

A data-aware component is responsible for reading all the data that it requires from the data file. For example, a data-aware part is responsible for reading all of the mass and inertia data associated with it; and a data-aware bushing is responsible for reading all the force and torque characteristics that describe it. Data-aware components are also easier to validate because

correlation occurs at the component-level and the only configuration work is to point the component to the right data source.

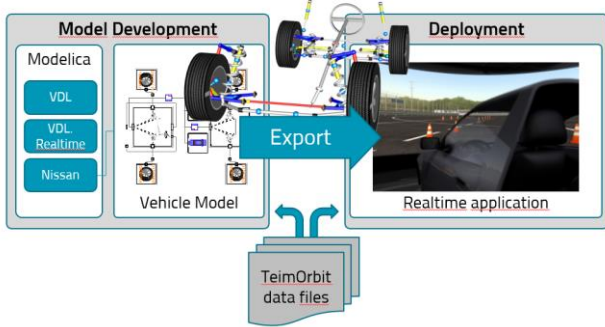


Figure 8. The data management (DataAccess) is compiled into the executable model. This ensures that the parameters are read from the same source regardless of the deployment scenario.

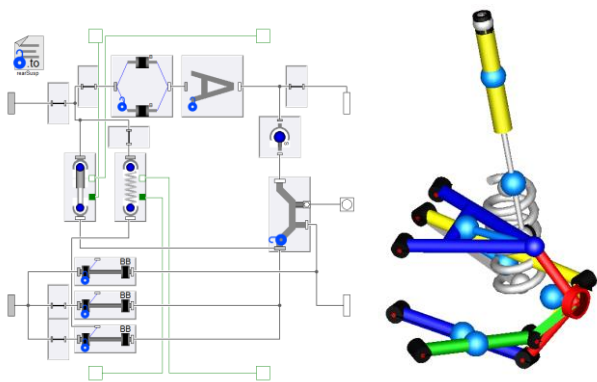


Figure 9. Linkage topology view (left) configured with data aware components that read TeimOrbit data and resulting 3D view (right).

4.2 Validation against offline tool

To ensure consistent behavior between offline tool used in development and the new real-time model, a validation procedure was carried out. This procedure included comparison on different levels, from components to chassis on road. For the suspension level, the standard offline procedure was replicated in Modelica, and configured to read the test specifications produced by the offline tools. Figure 10 shows the resulting test model that contains three main components; the test rig, the suspension, and the signal source.

The test rig provides a constraint between vehicle body and ground. The wheel centers are excited

through moving the wheel pads which induces forces through the tires. Additionally, forces and torques can be applied either at the wheel center, or at the tire contact patch. For the front suspension, either steering wheel position or steering wheel torque is given.

The suspensions are the same suspension that is used in the complete vehicle (Figure 3). The source block is configured using DataAccess to read the configuration information used by the off-line tool eliminate manual reconfiguration of test scenarios. Figure 10 shows some example correlation plots from the front suspension for parallel wheel travel.

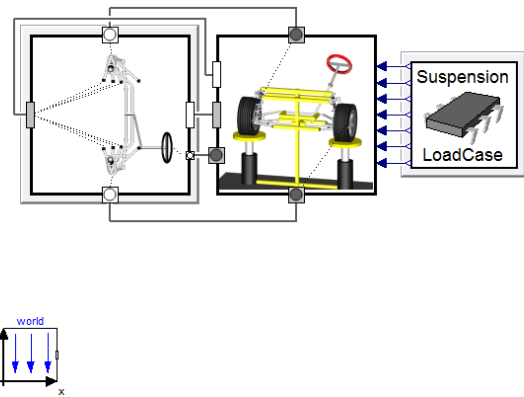


Figure 10. Suspension test model implementation showing the tested suspension, the boundary conditions, and the source block that reads the test specification.

5 Realtime configuration

The Vehicle Dynamics Library have been used to model real-time capable multi-body vehicle models for more than a decade, (Elmqvist et al., 2004). These models are heavily adopted in the Motorsports industry for various applications, see e.g. (Toso, 2014).

With recent development (Andreasson et al., 2014), it is now possible to execute high fidelity vehicle models with more than 150 DOF (300 states). This allows the models used for vehicle development at CAE departments to be executed directly in real-time applications. Key methods to achieve the performance is the inlining of the real-time solver and parallelization of the executable code.

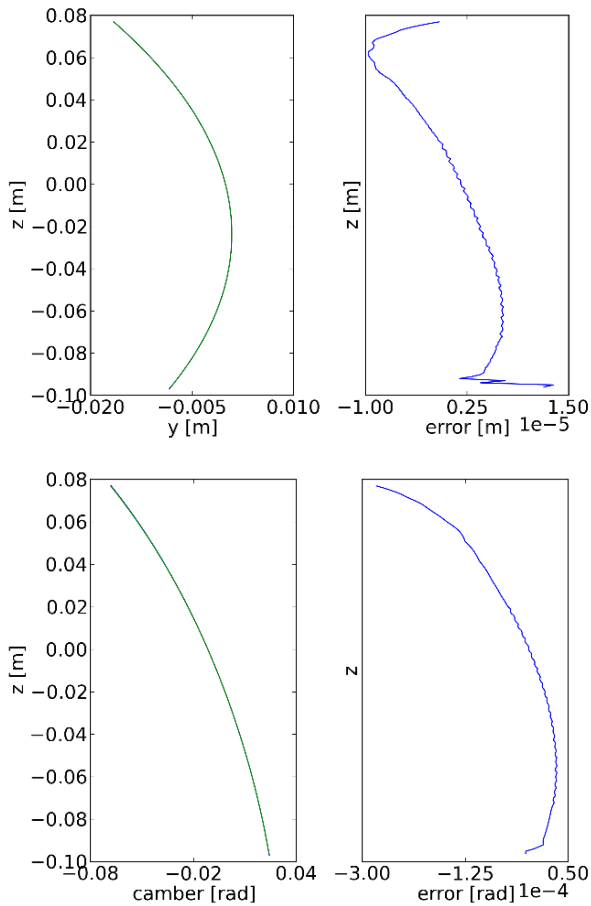


Figure 11. Comparison of results generated from offline tool (blue) and real-time model (green) for a parallel wheel travel. Plots show lateral (top) and camber (bottom). Each plot shows the curves overlaid and error, respectively.

5.1 Inlining

So called inlining (Elmqvist et al., 1995), has proven to be a successful way of achieving real-time performance of stiff systems, (Elmqvist et al., 2004). Inlining essentially means that the discretization formulas of the integration methods are substituted into the differential equations of the model, and then structural analysis and computer algebra are applied on the augmented system of equations. The method can be combined with both explicit and implicit discretization schemes. The explicit scheme is straight-forward but requires small enough steps to ensure stability. The implicit scheme has better stability properties but typically results in non-linear system of equations that need to be solved iteratively.

5.2 Parallelization

Solving large systems of non-linear equations is $O(n^3)$, meaning cost to solve the problem grows with the third power of the number of equations. For the model in this work, the size of the manipulated inlined

implicit integration system is 178 ($n_1=178$), which would be difficult to solve robustly in any real-time application without further manipulation.

As described in (Elmqvist et al., 2014), parallelized code can now be generated from Modelica models according to the (OpenMP, 2015) standard. The real-time model takes advantage of this functionality to distribute the workload of solving the systems of equations across multiple cores according to the following principle:

After the implicit integration scheme has been inlined with the model, the resulting system of equations is then divided into several smaller systems corresponding to the dynamics of the rear and front left and right suspension linkages, the powertrain, the steering and the wheels. The resulting impact on the structural side is that one large system of equations is now reduced to several smaller systems after manipulation, here $n_2=\{40, 40, 32, 32, 12, 11, 1, 1, 1, 1, 1, 1, 1, 1\}$.

This split gives two advantages, first several smaller systems solve significantly faster than one big system due to the cubic growth described above. In this particular case the reduction corresponds roughly to $n_1^3/\sum n_2^3$, which is about 30 times.

The second advantage is that the parallelization of the code allows the execution to be distributed on multiple cores. This also means that as long as there are cores available to distribute the calculations onto, any added model complexity will have a limited effect on the overall turn-around time as long as it does not add to any of the largest systems of equations.

5.3 Simulation accuracy and performance

The real-time configuration has been validated both with respect to accuracy and performance. To ensure robustness to high amplitude and high frequency inputs, the test suite contains a broad range of excitations such as jump and police turn in addition to the more traditional simulation set-ups. Figure 12 shows the vertical acceleration of the body while the car accelerates over an uneven ground surface.

The performance of the real-time model is defined by the time it takes to solve for each time step, so called turn-around time. Indications of the performance can be done directly on a desktop or laptop using timers, and on a Windows laptop (i7-3520M @ 2.90 GHz) the performance is roughly real-time. In Figure 13, the timing plot on the hardware-in-the-loop platform (Concurrent Xeon E5-2687w v2) is shown.

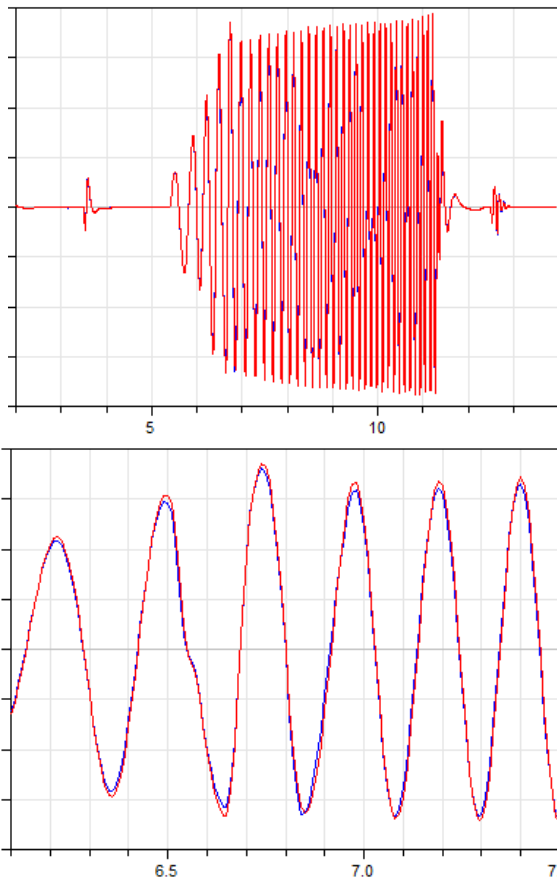


Figure 12. Time trace of vertical acceleration while accelerating over an uneven road. Reference trajectory generated with Dassl solver using relative tolerance $1e-6$ (blue), and real-time solver (red). For confidentiality reasons, the numerical values of vertical axis have been removed.

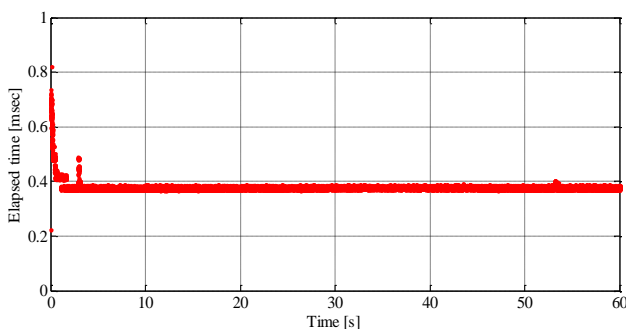


Figure 13. Execution time on the hardware platform. Each time step is 1ms.

6 Conclusions

This paper presents a real-time capable, high-fidelity model of a production vehicle. It is shown how this model can add real-time capability to the existing toolchain without having to replace or re-implement existing functionality. This is achieved by combining an open architecture with the ability to read and write

legacy data formats. It is also shown how to enable real-time simulation of high-fidelity vehicle models using inlining and parallelization.

All-in-all, the model presented in this paper can respond accurately to inputs and realistically predict the vehicle behavior as of the latest state of the development process. The deployment in real-time environments such as driver-in-the-loop and hardware-in-the-loop simulators enables both subjective and objective evaluation. This in turn allow for early assessment of the human-vehicle interaction and the integration of vehicle safety systems.

7 References

- Andreasson, J. et al. (2014). Realtime simulation of detailed vehicle models using multiple cores, in proceedings of *International symposium of Advanced Vehicle Control, AVEC*, Tokyo.
- Concurrent (2015). www.ccur.com
- DelftTyre (2015). <https://www.tassinternational.com/delft-tyre>
- Elmqvist, H. et al. (1995). Inline Integration: A new mixed symbolic/numeric approach for solving differential-algebraic equation systems, Proceedings of *European Simulation Multiconference*, June, Prague, pages XXIII-XXXIV.
- Elmqvist, H. et al. (2004). Realtime Simulation of Detailed Vehicle and Powertrain Dynamics. In Proceedings of the *SAE World Congress 2004*, Paper no 2004-01-0768, Detroit, Michigan.
- Elmqvist, H. et al. (2014). Parallel Model Execution on Many Cores, Proceedings of *10th International Modelica Conference*, Lund, Sweden.
- FMI (2015). www.fmi-standard.org
- FTire (2015). www.cosin.eu
- Hydraulics Library (2015). www.modelon.com/products/modelica-libraries/hydraulics-library/
- Modelica (2015). www.modelica.org
- OpenCRG (2015). www.opencrg.org
- OpenMP (2015). openmp.org
- Pneumatics Library (2015). www.modelon.com/products/modelica-libraries/pneumatics-library/
- Rauh, J. (2003). Virtual development of ride and handling characteristics for advanced passenger cars. *Vehicle System Dynamics*, 40(1-3), 135-155.

Toso, A. and Moroni, A. (2014). Professional Driving Simulator to Design First-Time-Right Race Cars. *SAE Technical Paper 2014-01-0099*.

Vehicle Dynamics Library (2015).

www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/

Yasuno, Y. et al. (2014). Nissan's New High Performance Driving Simulator For Vehicle Dynamics Performance & Man-Machine Interface Studies. In proceedings of *Driving Simulation Conference 2014* Paris, France, September 4-5, 2014

Validation of a Battery Management System based on AUTOSAR via FMI on a HiL platform

Leonard Janczyk¹ Klemens Esterle¹ Stephan Diehl¹
Michael Seibt¹ Arthur Gauthier² Viry Guillaume³

¹Dassault Systèmes Deutschland GmbH, Munich, Germany

²Dassault Systèmes SE, Plouzané, France

³Dassault Systemes KK, Tokyo, Japan

leonard.janczyk@3ds.com, klemens.esterle@3ds.com, stephan.diehl@3ds.com,
michael.seibt@3ds.com, arthur.gauthier@3ds.com, guillaume.viry@3ds.com

Abstract

In systems which are sourcing their electric energy from a battery system, such as electric or hybrid electric vehicles, it is of crucial importance to monitor the battery's condition in order to ensure its usability and longevity. The battery management system (BMS) is a control unit which supervises the physical variables in order to assess the condition of the battery.

For the development and testing of control units in the automotive industry, such as the BMS, the AUTOSAR standard was introduced, which separates application code from platform-specific software. By using AUTOSAR tools and the model exchange via the Functional Mock-up Interface (FMI), this paper shows how BMS algorithms can be validated and tested in several abstraction layers. A sub-function of the algorithm is tested first in the Modelica-based system simulation tool Dymola on a personal computer and then on Hardware-in-the-Loop (HiL) platform which emulates the hardware of an automotive ECU.

In order to provide realistic inputs of the physical variables, a battery model in Modelica is built using the Dymola add-on Battery Library by Dassault Systèmes. In order to run on the HiL platform the battery model is implemented such that it is real-time compliant.

For both, the BMS algorithm and the battery model, it is described along the process which adjustments need to be made when switching from the simulation framework to the HiL platform.

Keywords: battery model, battery management system, AUTOSAR, FMI, ASim, MiL, SiL, HiL, XiL, Co-Simulation

1 Introduction

Today's system- and software development teams work quite isolated from one another. Information exchange is usually limited on written specifications. With the example of the battery management system (BMS) we will show a method in which information can

effectively be exchanged through a model based on the Functional Mock-up Interface (FMI) as executable specification. This allows both parties closed-loop simulation at different stages of the V-Cycle. This way, software developers can more thoroughly test their software in a virtual environment. At the same time the system simulation teams can simulate their whole system without the need to manually re-implement the software algorithms of the ECU code.

This paper illustrates how based on FMUs (Functional Mock-up Units) source code from an AUTOSAR Battery Management Algorithm can be simulated on different abstraction levels in order to verify the algorithm for a failure mode.

At first, in section 2, the physical battery model will be introduced along with example battery module which it represents. In a second step, the function and tasks of a battery management system will be explained. The focus will shift on the specific algorithm, the charging status estimation, which is chosen as an example in order to demonstrate the process for the overall BMS. In section 3, the AUTOSAR standard and the used tool chain will be described.

2 Battery Simulation and Battery Management

Proper battery modelling plays an important role in this context. On the one hand, the model needs to provide a proper representation of the inner workings of the battery so the battery management system receives a realistic and complete set of signals.

On the other hand, the battery model needs to be performant enough in order to be compliant with real-time requirements. In the following two sections, the battery model will be introduced.

2.1 Battery Simulation Model

The battery pack which is modelled is a 48 V module. It could be deployed in micro-hybrid systems for the on-board electric power supply or as part of a traction

battery system. The module features two parallel-connected rows of each 13 battery cells in serial electric connection. Their combined capacity of the 26 battery cells amounts to 140 Ampere hours.

Table 1. Battery Cell Parameters.

Parameter	Unit	Value
Nominal Cell Capacity	Ah	2.7
Nominal Cell Voltage	V	3.6
Maximum Voltage	V	4.2
Minimum Voltage	V	2.5
Maximum Internal Resistance	mΩ	30
Shape	-	round

The battery module is modelled in Modelica using the Battery Library from Dassault Systèmes (Gerl, et al. 2014). The physical battery cell models is made up of the physical domains relevant for the batteries behavior: electric and thermal.

The main requirement for cell models used in system simulation is to provide accurate information on the macroscopic characteristics (e.g. voltage, current and state of charge) combined with reasonable computation time. This way, the impedance characteristics of the real cell are replicated. In many applications models using an electrical equivalent circuit fulfill these requirements

The voltage of a battery U can be described as the difference between the open circuit voltage U_{OCV} and a number of over potentials η_i caused by different electrochemical effects:

$$U = U_{OCV} + \sum \eta_i \quad (1)$$

These over potential can be modelled with equivalent electric circuit networks. In Figure 2 the voltage characteristic for the step current discharge of a NiMH cell is shown. The effect is similar for Lithium-Ion based cells, such as the ones used for this example.

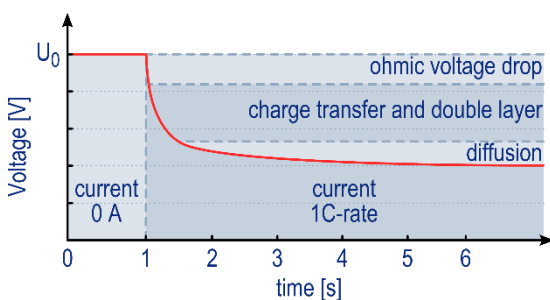


Figure 1 Voltage characteristic of an electrochemical cell (NiMH) (Jossen und Weydanz 2006)

The over potential is divided into an ohmic over potential η_{ohm} , over potential caused by charge transfer and the electrical double layer η_{trans} and over potential due to diffusion η_{diff} . An electrical equivalent circuit capable of reproducing the voltage characteristic from Figure 1 is shown in Figure 2, whereas the dynamic

behavior of the over potentials are modelled using RC-circuits.

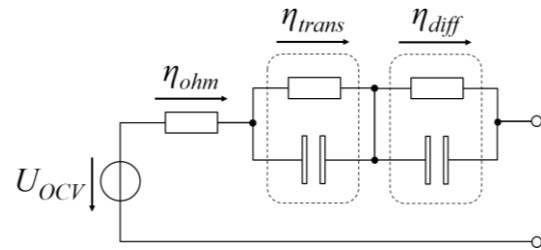


Figure 2 Voltages in the equivalent circuit model

In order to determine the influence of varying temperatures on electrical and aging behavior a thermal model of the cell and its surrounding environment is required. The heat inside the cell is generated mainly due to Joule effects, while the chemical reactions are exothermic or even endothermic to a minor degree. Thus the generated heat corresponds to the calculated power losses of the resistors of the equivalent electric circuit which are therefore connected to the thermal model.

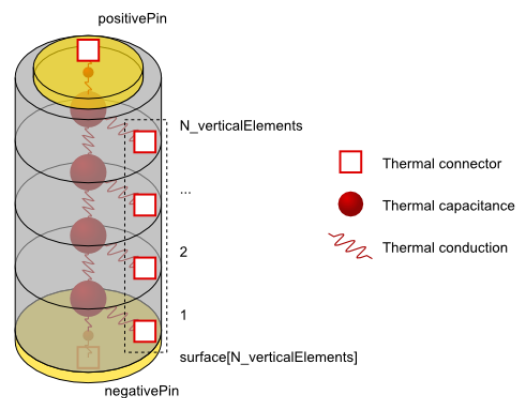


Figure 3 Representation of the thermal cell model

At the module level, where several cells form an electric, geometric and thermal entity, the major advantage in this context is that the battery pack can be adjusted according to the performance needs. In practical terms, this results in the question whether the battery cells are each represented as a Modelica object. A simplified approach would be modelling just one cell and scaling up the results to module size by multiplying the inputs and outputs by the cell numbers in accordance with their electrical wiring. Also, the thermal representation of the cells can be adjusted in the number of discretized elements. In the case of a round cell these elements are vertically slices which help to calculate the cell internal flow, as sketched in Figure 3.

Of practical importance is the fact that the Hardware-in-the-loop platform usually does not feature a data system. Modelica models in industrial environments might be parameterized by external parameter files. When exporting the model for the HIL environment, the data needs to be placed within the model without any external dependencies.

2.2 Battery Management System

The battery management system has to process sensor data and on-board model simulation results in order to obtain information about the state of the battery. Tasks of the battery management system (BMS) include the determination whether the monitored variables are still within the acceptable limits. Apart from state-of-charge (SoC) and the state-of-health (SoH) usually the cell temperature, the cell voltage and the system voltage are monitored. In case one of these variables appears to be out of the operational limits, the battery management

$$SoC = f(U) = f(U_{OCV}) \text{ for } \eta_i \rightarrow 0 \quad (3)$$

system sends a signal to the overall power management control unit which restricts the power usage of the consuming components.

Especially the SoC and the SoH are variables, which need to be monitored to ensure overall system availability at any given moment (He, Wei and Brian 2010). When the SoC reaches a critically low level in general in the range of 5-10%, the electrodes of the battery take severe damage and the battery voltage might drop below a level at which the battery system cannot provide the required power anymore. On the other hand, when the SoC exceeds 100% by too much, the battery cell stores excessive amounts of energy beyond a level which it can safely handle. In severe cases, this might even cause a “thermal runaway”, a strong exothermal reaction after which the battery system is completely dysfunctional.

Therefore in any case the battery management system should encompass a SoC estimation algorithm. In the following, the realization of such an algorithm will be discussed.

2.3 Estimation of Battery State-of-Charge

In a battery simulation model the change of the SoC can be calculated by balancing the electric charge and discharge current such as in equation (2). The SoC is by definition part of the overall amount of electric charge available for discharge with C_n being the nominal battery capacity in Ampere seconds (He, Wei and Brian 2010).

$$\Delta SoC = \frac{\int_{t_0}^{t_{end}} I dt}{C_n} \quad (2)$$

However the SoC determination is more complex when being implemented on a battery ECU.

First, integration is a mathematical operation which requires more resources in terms of on-board memory and computational time compared to other mathematical operations.

Secondly, current sensors do not necessarily deliver a constantly precise measurement output. Calibration errors result in constant drifts of the recorded battery current. This drift might not significantly influence the

quality of the estimation during a short period such as a short inner city ride. However during longer trips such as an inter-city highway tour the drift in the charging status estimation might accumulate to a point where the battery is depleted while the battery management system assumes the charging status as being sufficient.

In order to ameliorate the quality of the SoC estimation, corrective back-up algorithms need to be included. A viable alternative is measuring the voltage of the battery when the battery is in electrochemical equilibrium, meaning that no electric load or charging is applied and excitation of previous electric load has faded. In this state the over potentials are negligible leaving the open circuit voltage as the dominant factor determining the cell voltage:

As a matter of fact, the open circuit voltage is usually measured during the initial rating of new cell type and also typically used for the parameterization of equivalent circuit cell models as shown in Figure 2.

Implementing the relationships presented in equations (2) and (3) in Modelica code could be drafted as followed:

```
der(SoC_count) * C_n = current + error;
SoC_est = SoC_count;
when zeroCurrentTimer > fadingTime then
  reinit(SoC_est, SoC_ocv);
end when;
```

In the first code line the charge counter (`SoC_count`) is implemented after the fashion of equation (2) with the `error` signal applied on the `current` signal. The output `SoC_est` is directly loaded with the result of the integration over the current and standardization with the nominal capacity C_n .

The `when`-clause representing equation (3) becomes active at the time point at which the current has been close to zero for a time period, implementation shown in Figure 4, with the influence of charge transfer over potential has most likely faded, in this case more than the time constant `fadingTime`. The calculated SoC will be replaced then with a charging status which has been extracted from a look-up-table describing SoC over OCV.

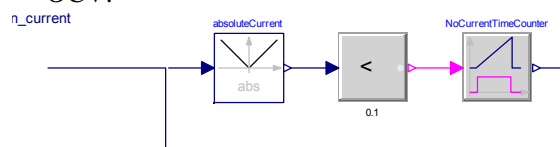


Figure 4 Counting time with current close to zero

During the verification phase of the system engineering process, the battery management system needs to be verified if it lives up to battery safety requirements, i.e. if the variables describing the battery state are recorded properly, the operational limits are correctly determined and their violations duly signaled.

In context of this paper, the ability of the charging status estimation algorithm to correct an erroneous

current measurement signal will be verified. For demonstration purposes the implementation is limited to charge counter and correction by voltage comparison as laid down in this chapter. A typical question answered during this process might be whether an estimation correction via cell voltage is sufficient to ensure that the battery management system and the driver are provided with the correct battery charging status.

3 Virtual Testing of AUTOSAR compliant controller software using FMI

3.1 What is AUTOSAR?

AUTOSAR (AUTomotive Open System ARchitecture) is a well-accepted standard for developing software for automotive electronic control units (ECU), as documented by (Bertsch, et al. 2015) for Bosch ECUs. It defines a layered architecture, separating hardware, application software and basic software through standardized interfaces.

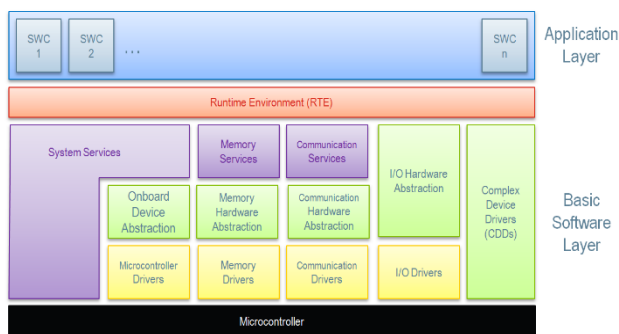


Figure 5 AUTOSAR Layered Architecture

As shown in Figure 5 the interface between application software components and the interfaces between application software and basic software (BSW) is handled through the runtime environment (RTE), which implements different types of communication mechanisms (AUTOSAR 2016).

3.2 AUTOSAR Unit Test

One essential part in the AUTOSAR software development is the testing of individual software components and the whole software architecture (top level composition). Ideally those tests should be executable without any hardware-dependencies to enable testing as soon as possible in the development cycle. AUTOSAR addresses this through the Virtual Function Bus (VFB) Abstraction Level. The AUTOSAR test environment ASim from Dassault Systèmes is also applying this concept taking a real AUTOSAR compliant operating system (OS) and RTE into account. This allows testing of software components on a very granular level, also considering effects through the OS, e.g. scheduling, or through the RTE, e.g. synchronize queued and non-queued

communication. Even fixed-point arithmetic is taken into account using AUTOSAR datatypes.

3.3 FMI-based Export of virtual AUTOSAR ECUs

Unit tests are in general open-loop tests, which means the user has to define sufficient and reasonable test-vectors and test-constraints, which is often quiet challenging and time-consuming. Hence integrating the software “model” in a virtual environment which closes the loop through a plant-model would make the conditioning of many of the software component-ports obsolete, as they will be fed directly through the connected plant model. In addition to that timing effects and delays could also be taken into account by a plant model. ASim opens this possibility through FMU export the extraction of a virtual AUTOSAR ECU which can then be integrated into other simulation platforms supporting the FMI-Standard.

3.4 FMI-based XiL Tool Methodology

The Modelica-based simulation tool Dymola supports the import, export and simulation of FMUs (FMI for Model Exchange and Co-Simulation, (Blochwitz, et al. 2011)). Software- and plant-models can be simulated on different abstraction levels, which allows MiL- and SiL-testing. FMUs can also be exported via the source-code generation capabilities. These can then be compiled for different HiL platforms. Based on a Battery Management Unit it is illustrated, how XiL-tests can be performed using the FMI-standard.

4 Results and Discussion

4.1 Model-in-the-Loop

In general, system simulation starts at an earlier stage in product development than the software development. In this context both battery model and BMS algorithm are implemented as models to evaluate system behavior and the response algorithm together in a Model-in-the-Loop (MiL) simulation. One could argue that the BMS algorithms could be coded from the beginning in a software development platform for the ECU software instead of being implemented in the same simulation environment as the model itself. However when taking a closer look at the model equations and the required solvers, the advantage of this method will become obvious:

Using an acausal object-oriented Modeling language like Modelica for modeling physical systems often results in a higher-order differential algebraic equation system (DAE) with slow dynamics, looking at the thermal behavior of the battery case and fast dynamics caused by the electrical cell behavior. An implicit solver like DASSL is designed to deal with those type of systems. As only explicit fixed-step solvers can be used in a real-time environment, numerical stability for the

sampling rate of the ECU has to be ensured. Testing the battery model in the modeling environment with the step-size matching the sampling rate reduces the use of generally much more expensive real-time hardware.

In addition by using this approach the functionality of the battery model can be verified in parallel to the development of the battery management system by the functional development engineer resulting in an acceleration of the design phase. Moving forward in the product development, this procedure allows early simulation experiments.

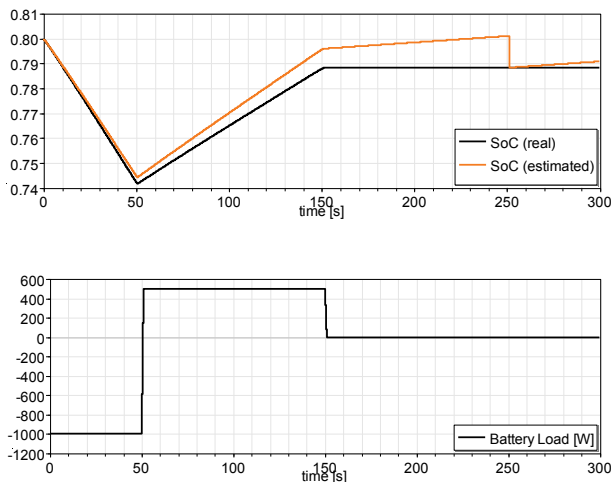


Figure 6 Model-in-the-Loop Simulation: SoC (*top*) and power load (*bottom*) over time in seconds. Negative power indicates discharging of the battery module.

When coupling the Modelica implementation of the SoC estimation algorithm with the Battery Library model and applying a load cycle and environmental conditions, the following results are obtained as shown in Figure 6. The load power cycle consists of a discharge phase of 1 kW, an immediate recharge of 500 W and a subsequent unstressed time period at room temperature. One can observe that due to the forced condition offset error of +0.5 A on the current sensor, the output of the charging status estimation drifts away from the actual SoC up to the point where the divergence amounts to over one percentage point. A short time period later the power load is stopped. The cell voltage is largely no longer influenced by the electrochemically induced overpotentials but only by the open circuit voltage. The corrective algorithm steps into action, looks up the SoC value which matches the measured voltage. At second 250 the *reinit* command is ignited and replaces the calculated SoC value with the one based on the measured cell voltage.

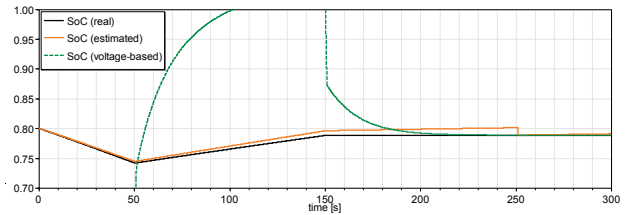


Figure 7 Model-in-the-Loop Simulation: Different Methods of SoC over the course of the battery load cycle

A comparison in Figure 7 between the SoC values also shows at this early software design stage why charging status based on the measured cell voltage cannot serve as a lone signal source, and why a certain time period needs to pass before the correction is applied.

As the focus is the evaluation of the concept per se, the simulation is performed on a high-performing workstation with characteristics described in Table 2.

Table 2 Technical Characteristics of the workstation (Intel Corporation 2016)

<i>Parameter</i>	<i>Value</i>
CPU Type	i7-4810MQ
Instruction Set	64 Bit
Number of CPU Cores	4
Base Frequency	2.8 GHz
L2 Cache size	1 MB
L3 Cache size	6 MB
RAM	16 GB

With summoning such computing power while using a language which is native for the Dymola solver shows a satisfying result for simulation time: The 17671 equations of the Modelica are integrated in 110 seconds while the CPU-time for one GRID interval is 0.365 milliseconds.

4.2 Software-in-the-Loop

Once the algorithms of the BMS have been drafted and evaluated during the MiL testing, they are implemented as software functions for the ECU. Using the ASim plugin of the Autosar Builder, the BMS algorithm can be exported as FMU and coupled with the physical battery simulation model in Dymola. As the algorithm is now in the same format as on the ECU, this stage is called Software-in-the-Loop (SiL) simulation.

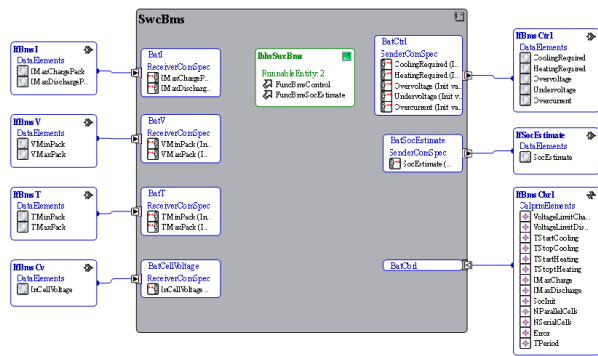


Figure 8 AUTOSAR Software Component of BMS. The interfaces for I/O and calibration are marked in blue, the internal behavior is marked green (presented in Figure 9).

In this stage of the process, the stability is first verified with the ECU code coupled with the battery model in Modelica in Dymola and in a second step as in FMU both again imported in Dymola. Taking into account the sampling time of the Hardware-in-the-Loop platform, it is to be tested whether coupled entities are running stable when being operated with a fixed-step solver with a sampling time of one millisecond.

When implementing the charge status algorithm as sketched out in Modelica for the MiL simulation, some methods need to be altered in order to ensure a sufficient performance on an ECU for implementing the charging status estimation algorithm:

As mentioned before in 2.3, the integration operation used in equation (2) would use too much on-board memory and is not always available in the ECUs instruction set, so it needs to be replaced by a discrete sum operation which accumulates the measured current with each time step.

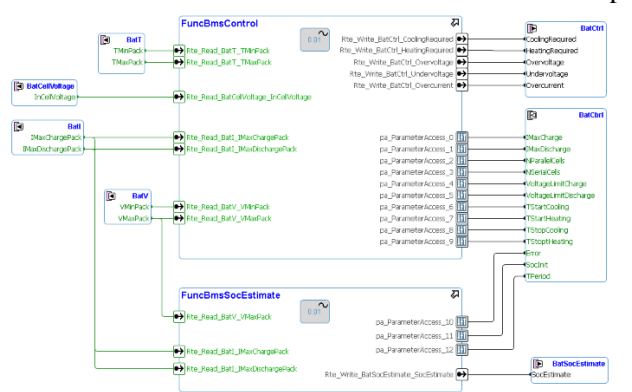


Figure 9 AUTOSAR Builder Screenshot of the internal behavior of BMS. The supervision algorithm for the operational limits is in the upper block *FuncBmsControl*, while the charging status estimation is in bottom block *FuncBmsSocEstimate*.

As shown in Figure 10, in both cases the system of physical model and algorithm works stable with

reasonable results similar as obtained in the Model-in-the-Loop simulation in chapter 4.1.

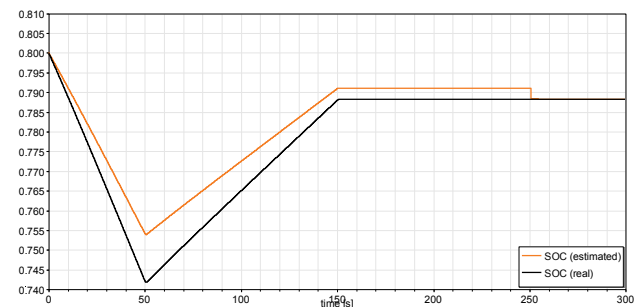


Figure 10 SiL Simulation: Results AUTOSAR-FMU and Modelica Model in Dymola.

In the SiL simulation, the Dymola solver is now slowed down by processing the FMU. The integration time amounts now to 187 seconds with a 0.618 milliseconds per grid interval.

4.3 Hardware-in-the-Loop

In the final stage of the verification of the charging status algorithm, the battery model in Modelica and the BMS algorithms in AUTOSAR C-code are exported as FMUs and executed on a platform which emulates the hardware of an ECU of the target system. This stage is therefore called Hardware-in-the-loop. At this point, the stability of the software in a real-time environment can be verified. Additionally, hardware specific effects, such as the influence of signal propagation delays, limited memory, cache and processing speed are playing out as well.

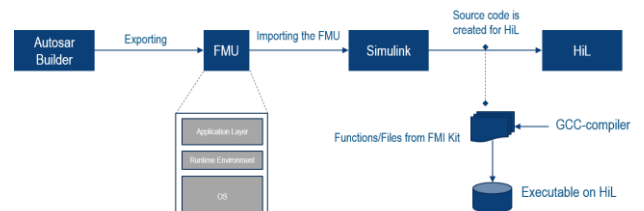


Figure 11 Toolchain and process for creating an FMU for a HiL platform.

The battery model FMU and the Autosar FMU are both set on the HiL platform. For this purpose, a dSpace DS1006 Processor Board is employed as specified in Table 3.

For being executed on the HiL platform, the AUTOSAR FMU had to be equipped with operating system functionalities such as scheduling. The toolchain is visualized in Figure 11 **Toolchain and process for creating an FMU for a HiL platform**.Figure 11.

Table 3 Technical Characteristic HiL platform's CPU (dSpace GmbH 2016)

Parameter	Value
CPU Type	Opteron
Instruction Set	64 Bit
Number of CPU Cores	4
Base Frequency	2.6 GHz
L2 Cache size	1 MB
L3 Cache size	-
RAM	128 MB

For the virtual validation of the charging estimation algorithm, the output signals are compared to the results in the MiL simulation in chapter 4.1, as shown in Figure 12.

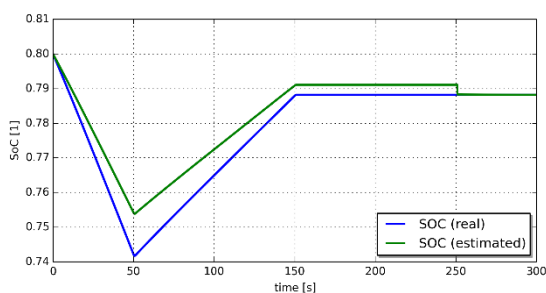


Figure 12 HiL Simulation: Results of the combined BMS FMU exported from AUTOSAR and battery model FMU exported from Dymola on the HiL platform.

The important characteristic for ensuring real-time requirements is the turnaround rate in percentage points. It states which fraction or multiple of the fixed-step sample time interval is consumed for the execution of the software code.

The turnaround time of the combined BMS FMU exported from AUTOSAR and battery model FMU exported from Dymola on the HiL platform is below 0.4 milliseconds. With the HiL platform processing according to a step time of 1.0 millisecond, the turnaround indicating a performance fast enough in order to be real-time capable.

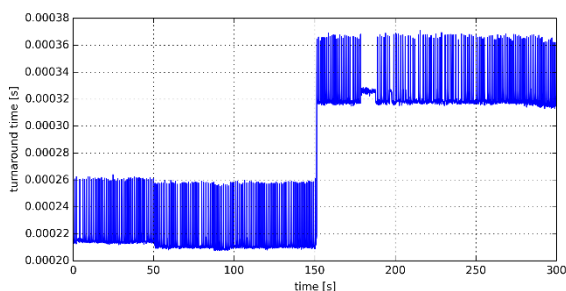


Figure 13 HiL Simulation: Turnaround time when executing the combined FMUs on the HiL platform.

5 Conclusion and Outlook

In this paper, it could be shown that the detailed battery model based on Dassault Systèmes Battery Library can be used for real-time applications as the derived system could be solved using a fixed-step integration method with a step-size of one millisecond. The BMS functionalities developed in AUTOSAR could be validated using the battery model as a FMU on the HiL platform.

With the example of the charging status estimation algorithm, it has been shown how BMS functions could be developed from draft to real-time verification using FMI across all stages of the process from MiL over SiL up to HiL.

As outlook from the perspective of the tool chain it should be noted that currently the FMU generated from the ASim in the AUTOSAR Builder uses the VFB level, which doesn't take the Basic Software or Complex-Device Drivers (CCDs) into account. In a next step, parts of the AUTOSAR Basic Software or CDDs could be also modelled and exported with the FMU. This would then also allow the consideration of propagation delays induced by the Basic Software Layer.

Acknowledgements

We would like to thank Dan Henriksson from Dassault Systèmes AB in Lund, Sweden, for his support on the HiL platform.

References

- AUTOSAR. 2016. *Technical Overview*. February 06. <http://www.autosar.org/about/technical-overview/>.
- Bertsch, Christian, Jonathan Neudorfer, Elmar Ahle, Siva Sankar Arumugham, Karthikeyan Ramachandran, and Andreas Thuy. 2015. "FMI for physical models on automotive embedded targets." *11th International Modelica Conference*. Versailles, France. 43-50. doi:10.3384/ecp1511843.
- Blochwitz, Otter, Bausch, Clauß, Elmquist, Junghans, Mauss, et al. 2011. "The Functional Mockup Interface for Tool independent Exchange of Simulation Models." *8th International Modelica Conference*. Dresden, Germany.
- dSpace GmbH. 2016. "DS1006 Processor Board." *Technical Details*. February 06. <http://www.dspace.com/en/pub/home/pr>

oducts/hw/modular_hardware_introduction/processor_boards/ds1006.cfm.

- Gerl, Johannes, Leonard Janczyk, Imke Dr Krüger, and Nils Modrow. 2014. "A Modelica Based Lithium Ion Battery Model." *10th International Modelica Conference*. Lund, Sweden: Linköping University Electronic Press. 335-341.
- He, Yongsheng, Liu Wei, and Koch J. Brian. 2010. "Battery algorithm verification and development using hardware-in-the-loop testing." *Journal of Power Sources* (195): 2969-2974.
doi:10.1016/j.powersour.2009.11.036.
- Intel Corporation. 2016. "Intel® Core™ i7-4810MQ Processor." *Specifications*. February 06.
http://ark.intel.com/products/78937/Intel-Core-i7-4810MQ-Processor-6M-Cache-up-to-3_80-GHz.
- Jossen, Andreas, and Wolfgang Weydanz. 2006. *Moderne Akkumulatoren richtig einsetzen*. Reichhardt Verlag.

Chattering-Free Simulation of Hybrid Dynamical Systems with the Functional Mock-Up Interface 2.0

Ayman Aljarbouh¹ Benoit Caillaud¹

¹Centre de Recherche INRIA-IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France,
{ayman.aljarbouh,benoit.caillaud}@inria.fr

Abstract

The numerical simulation of non-smooth hybrid systems exhibiting chattering behavior requires high computational costs. In the worst case, the simulation appears to come to a halt, since infinitely many discrete transitions would need to be simulated. In this paper we present an FMI-based framework and prototypical implementation for robust and reliable detection and elimination “On the Fly” of chattering behavior in run-time simulation of non-smooth hybrid systems. The main benefit of the developed framework is that it establishes solvability requirements and theorems for simulating hybrid systems while performing the chattering path avoidance internally in the master algorithm of the interface. This increases the efficiency of the chattering-free simulation as no enumeration of modes is required during the chattering detection and elimination process. The developed FMI-based framework can generate a chattering-free simulation for any generic chattering Functional Mockup Unit (FMU) conforming to the FMI standard v2.0 Specification for model exchange.

Keywords: Functional Mockup Interface (FMI), Functional Mockup Unit (FMU), Non-smooth Hybrid systems, Discontinuity mappings, Chattering

1 Introduction

In the literature, the term “hybrid systems” is used to describe a very wide class of dynamical systems with interacting continuous and discrete dynamics. The state variables in such systems are capable of evolving continuously (flowing) and/or evolving discontinuously (jumping). That is, the presence of two different behaviors, continuous and discrete, is the cause of heterogeneity (Zhang et al., 2001; Cai et al., 2008). However, even simple hybrid systems can exhibit many unique phenomena, such as *chattering behavior*. The interaction between time-driven continuous variable dynamics (i.e. ODEs, DAEs) and event-driven discrete logic dynamics (i.e. If-then-else) may lead to this non-smooth be-

havior, which can be intuitively thought of as involving infinitely fast and continuous switching between different control actions or modes of operation (Aljarbouh and Caillaud, 2015b). Models of physical hybrid systems may be chattering due to modeling over-abstraction, actuators limitations, time discretization, or unmodeled dynamics (usually from servomechanisms, sensors and data processors with small time constants).

1.1 Problem Statement

As in physical hybrid systems there is no chattering, it is not reasonable then to assume that the control signal time evolution can chatter or switch at infinite frequency. This undesirable significant oscillation with an infinitely fast frequency components of the control propagate through the system, because of chattering, affects the system output. In particular, chattering control is harmful because it leads to low control accuracy, and once applied, can lead to high wear of moving mechanical parts, as well as high heat losses in electrical power circuits. In addition, the numerical simulation of hybrid systems exhibiting chattering behavior requires high computational costs as small step-sizes are required to maintain the numerical precision. For both non-adaptive and adaptive time stepping with event localization, root finding to locate the exact time of occurrence of the chattering event causes continuous integration to become dramatically and excessively slow. The system converges fast to the point in time at which infinitely many discrete transitions need to be simulated, and the simulation then appears to come to a halt. Chattering behavior has to be treated in an appropriate way to ensure that the numerical integration progress terminates in a reasonable time. This has been investigated by means of different methods. A smooth sliding motion can be induced on the switching manifold on which the chattering occurs (Leine and Nijmeijer, 2004; di Bernardo et al., 2008; Biák et al., 2013; Weiss et al., 2015). Filippov Differential inclusion approach (Filippov, 1988) can be used in this case to define equivalent sliding dynamics on the switching manifold on which the chattering occurs. Another approach (the so-called equivalent control) proposed by Utkin (Utkin,

1992) can also be used. However, the computation of the equivalent dynamics turns to be difficult whenever the system chatters between more than two dynamics. This arises when the chattering behavior occurs in dynamical systems having multiple discontinuous control variables. In the Functional Mock-up Interface (FMI) specification, Functional Mockup Units (FMUs) should add a small hysteresis to the event indicators to avoid chattering (Blochwitz et al., 2012). This approach has the following disadvantages: I) A Modelica tool will also add a hysteresis when handling state events, to ensure that the zero crossings happen with non-zero values of the input arguments of the event functions at the integration restart. Therefore, when calling the FMI function `fmi2GetEventIndicators` from the Modelica model, it will introduce the hysteresis twice to the event indicators, and as a result, the resulting event triggered by the imported FMU is slightly inaccurate. II) Adding hysteresis to the event indicators does not guarantee an efficient treatment of the chattering behavior, as the physics in chattering hybrid systems make the solution $x_\varepsilon(\cdot)$ be a saw-toothed, or zigzag function, i.e., a function that oscillates around the switching surface, with peaks at $-\varepsilon < 0$ and $+\varepsilon > 0$, with $t_{i+1} - t_i = 2\varepsilon$ (see Example 1 in Section 2.3). III) The size of the small value ε shall be related to the size of the event indicator z_j . The interface then would become more complicated, because, in order to determine the size of ε in the simulation environment which imports an FMU, the “nominal” value of z_j has to be reported by the FMU, which requires more information from the tool that generated the FMU, but cannot be handled efficiently in the simulation environment that calls the FMU. IV) If this would be handled in the simulation environment, there is always the danger that the environment does not handle it properly, but the FMU would be blamed for a failure.

1.2 Contribution

In this paper, we present methods and techniques for treating chattering behavior of non-smooth hybrid dynamical systems in the context of the Functional Mock-up Interface (FMI), and a prototypical implementation. In particular we discuss technical issues and implementation of a generic FMI which rigorously detects and eliminates chattering behavior in run-time simulation without modes enumeration, and without any need to add a small hysteresis to the event indicators in the FMUs. The developed chattering-free FMI localizes the non-smooth structural changes in the system in an accurate way and allows sliding mode simulation when the chattering occurs. It treats the chattering non-smoothness in the trajectory of the state variables by a smooth correction after each integration time-step. Furthermore, our chattering-free FMI can robustly handle the case of chattering on the intersection of finitely many switching manifolds iteratively without any need

to solve stiff nonlinear equations for the computation of the chattering-free coefficients. In addition, this paper provides a guidance for development of a hybrid chattering-free version of the Functional Mockup Interface (FMI) standard, giving a computational framework for an ideal manipulation of chattering behavior.

The paper is organized as follows: Section 2 gives a closer look into how the chattering behavior occurs in hybrid systems, as well as the challenges when simulating hybrid systems with chattering executions. Afterwards, we present in Section 3 the chattering-free semantics for reliable detection and elimination of chattering behavior in run time simulation. In Section 4, a prototype implementation is sketched for applying the chattering-free computational framework from Section 3 to the Functional Mock-Up Interface v2.0 for Model Exchange. Finally, the simulation results and conclusions of the work are given in Section 5 and Section 6 respectively. We illustrate the concepts with examples throughout the paper.

2 Chattering in Hybrid Systems

Formally we define chattering executions as solutions to hybrid systems having infinitely many discrete transitions in finite time. This happens when nearly equal thresholds for the transitions conditions of different modes are satisfied and the system start to oscillate around them. Numerical errors may also be the source of chattering as transitions conditions can be satisfied because of local errors. In chattering behavior, the system moves back and forth between modes, that is, the gradient of continuous-time behavior in each one of two adjacent modes is directed towards their common switching surface. When in either of the two adjacent modes on the common switching surface, an infinitesimal step causes a mode change. In the new mode, the gradient directs behavior to the previous mode and after another infinitesimal step a change to the previous mode occurs.

2.1 Chattering Execution

An *execution* χ of a hybrid system is chattering if there exist finite constants τ_∞ and C such that

$$\lim_{i \rightarrow \infty} \tau_i = \sum_{i=0}^{\infty} (\tau_{i+1} - \tau_i) = \tau_\infty \quad (1)$$

$$\forall i \geq C : \tau_{i+1} - \tau_i = 0 \quad (2)$$

where $\{\tau_i\}_{i \in \mathbb{N}}$ is a set of strictly increasing time instants represents discontinuity points (state events instants).

2.2 Chattering and Simulation

An essential element of numerical simulation of a hybrid dynamical system is the generation of discrete events

from continuous variables that exceed thresholds. Generating these events is generally implemented using relational operations (e.g. $>$, \geq , $<$, \leq). For an accurate simulation, the point in time at which these relations change their truth value has to be located within a small tolerance. A zero-crossing function $g(t, x)$ can be used to identify the boundary at which the change takes place. Usually state variables x are used as argument to the event indicator $g(t, x)$. The nature of zero-crossing detection and location is to compare the sign of the function value $g(t, x)$ at the beginning and the end of each time integration step, and if it changes, declare that it crossed zero and then bracketing the zero-crossing event (i.e. bisectional search) to locate the zero-crossing. During the search process, the values of state variables x , needed for the computation of $g(t, x)$, are evaluated by interpolation, using the values $x(t_i)$ and $x(t_{i+1})$. Because of the nature of finite precision arithmetic on digital computers, the time that the event occurred can only be located within an interval $[T_{left}, T_{right}]$ that corresponds to machine precision. During each iteration of the zero-crossing location, the zero-crossing function is evaluated twice: at the left and the right side of the reducing interval. After the event is bracketed by T_{left} and T_{right} , the ODE solver first advances integration time from t_i to T_{left} . The solver is then reset before advancing to T_{right} followed by switching the mode. In doing so, the assumption of continuity holds throughout the numerical integration. However, this approach may fail if the system exhibits a chattering execution. The zero crossing function $g(t, x)$ is a function of the model state, but it does not contribute to its continuous dynamics $f(t, x)$. Therefore, the numerical integration can proceed without taking the dynamics of $g(t, x)$ into account, and when these are faster than the dynamics $f(t, x)$, the chattering execution then causes the previous T_{right} to become the T_{left} of the next time step, and the integration will move with the minimum step size allowed. In order to illustrate the simulation of a chattering execution, a simple example shall be given.

2.3 Example 1: Relay Feedback

The relay feedback system is a good candidate to show the chattering behavior of a hybrid dynamical system (Aljarbough and Caillaud, 2015a). The relay feedback system consists of a dynamical system and a sign function connected in feedback. The sign function leads to a discontinuous differential equation (Johansson et al., 2002). Consider the following example for $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (3)$$

$$y(t) = Cx(t) \quad (4)$$

$$u(t) = -sgn(y(t)) \quad (5)$$

$$A = \begin{bmatrix} -3 & 1 & 0 \\ -3 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ -2\beta \\ \beta^2 \end{bmatrix} \quad (6)$$

$$C = [1 \ 0 \ 0] \quad (7)$$

The system in this example is represented as a hybrid system with two control modes q_1 and q_2 where the phase space of the system is split by a single switching manifold $\Gamma = \{x \in \mathbb{R}^n : g(t, x) = 0\}$ into two domain: $D_1 = \{x \in \mathbb{R}^n : g(t, x) < 0\}$ and $D_2 = \{x \in \mathbb{R}^n : g(t, x) > 0\}$ so that opposed zero crossing of the switching function $g(t, x) = x_1(t)$ defines the switching from q_1 to q_2 and vice-versa (e.g. a switching from D_1 to D_2 occurs when $g(t, x)$ changes its domain to $g(t, x) \geq 0$). It is important to recognize that the “zero crossing” approach defined by available integrators, for detecting state events, requires that the event function variables are non-zero at the event instant and after initialization. So, suppose one integrates the differential equation 3 with some delay in the control switch between $+1$ and -1 because some kind of hysteresis function implemented around the switching surface $x_1 = 0$. In addition to use it for handling the non-zero domain change of event functions, such a procedure is sometimes used in order to avoid too many switches. Even with adding such hysteresis, the physics in this system, because of chattering, makes the solution $x_{1\epsilon}(\cdot)$ to be a saw-toothed, or zigzag function, i.e., a function that oscillates around $x_1 = 0$, with peaks at $-\epsilon < 0$ and $+\epsilon > 0$, where $t_{i+1} - t_i = 2\epsilon$. Let the hysteresis size go to zero, i.e., $\epsilon \rightarrow 0$. Then $x_{1\epsilon}(\cdot)$ converges uniformly towards the zero function. Clearly the number of “events” goes to infinity on any interval of time with positive measure.

Hybrid systems simulation tools struggle even with such naive chattering hybrid system. For example, consider OpenModelica, and Acumen. In OpenModelica, for a data set $\beta = 0.5$ and $x_0 = [0.5 \ 3 \ 0.1]^T$, the solver gets stuck and the simulation terminates with a halt when the execution of the hybrid system start to exhibit a chattering. OpenModelica reports the following error message: *Chattering detected around time 1.88743591101..1.88743593454 (100 state events in a row with a total time delta less than the step size).*

```

model Example1
parameter Real x10 = 0.5 ;
parameter Real x20 = 3.0 ;
parameter Real x30 = 0.1 ;
Real x1 , x2 , x3 , u ;
initial equation
x1 = x10 ;
x2 = x20 ;
x3 = x30 ;
equation
der(x1) = -3 * x1 + x2 + u ;
der(x2) = -3 * x1 + x3 - u ;
der(x3) = -x1 + 0.25 * u ;
when x1 < 0 then
u = 1 ;

```

```

elsewhen x1 > 0 then
  u = -1;
end when;
end Example1;

```

Acumen language was developed as an extension of event-driven formalisms that have a similar flavor to synchronous languages. In Acumen, models are simulated by a fixed time stepping with fine interleaving of a sequences that can consist of multiple discrete computations followed by a single computation updating the values that should evolve continuously (i.e. global fixed point semantics). Thus, simulating what is happening at any single instance in time consists of zero or more discrete steps followed by a single continuous step. The Acumen model of the system in Example 1 can be written as following:

```

model Main(simulator) =
initially
  x1 = 0.5, x2 = 3, x3 = 0.1,
  x1' = 0, x2' = 0, x3' = 0, u = 0

always
  x1' = -3*x1 + x2 + u,
  x2' = -3*x1 + x3 - u,
  x3' = -x1 + 0.25*u,
if x1 >= 0 then u = -1
elseif x1 <= 0 then u = 1 noelse,
  simulator.timeStep += 0.0001,
  simulator.endTime += 10

```

Figure 1 shows the fixed time step simulation of Example 1 in Acumen language without events localization. With a fixed step size of 0.0001, the solution trajectory exhibits an undesirable oscillations around the switching surface Γ , with high frequency components of the control switching propagate through the system.

3 Detection and Elimination of Chattering

We consider a hybrid system \mathcal{H} with a finite set of discrete states $q \in \mathcal{Q}$ with transverse invariants (Lygeros et al., 2008), where the state space is split into different regions (invariants) $D_q \in \mathbb{R}^n$ by the intersection of p transversally intersected \mathbb{R}^{n-1} switching manifolds Γ_j defined as the zeros of a set of scalar functions $g_j(t, x)$ for $j = 1, 2, \dots, p$,

$$\Gamma_j = \{x \in \mathbb{R}^n : g_j(t, x) = 0; j = 1, 2, \dots, p\} \quad (8)$$

All switching functions $g_j(t, x)$ are assumed to be analytic in their second arguments, (i.e. $\frac{\partial g_j(t, x)}{\partial x} \neq 0$), so that, for each one of the intersected switching manifolds Γ_j , the normal unit vector $\perp_j = \frac{\frac{\partial g_j(t, x)}{\partial x}}{\|\frac{\partial g_j(t, x)}{\partial x}\|}$, orthogonal to the tangential plane $T_x(\Gamma_j)$, is well defined. Moreover, the normal unit vectors are linearly independent for all

the $\mathbb{R}^{(n-r)}$ swicthing intersections where $r \in \{2, 3, \dots, n\}$. The flow map vector field $f(t, x)$ of the hybrid system is discontinuous on all the switching surfaces Γ_j . Therefore, we can associate to each discontinuity surface Γ_j a discontinuous vector field of the form:

$$\dot{x} = f_j(t, x) = \begin{cases} f_{j1}(t, x) & \text{for } x \in D_{j1} \\ f_{j2}(t, x) & \text{for } x \in D_{j2} \end{cases} \quad (9)$$

$$D_{j1} = \{x \in \mathbb{R}^n : g_j(t, x) < 0\} \quad (10)$$

$$D_{j2} = \{x \in \mathbb{R}^n : g_j(t, x) > 0\} \quad (11)$$

so that opposed zero crossing of $g_j(t, x)$, defines the switching from D_{j1} to D_{j2} and vice versa. Note that, equation 9 represents the necessary condition for the hybrid system to accept a chattering execution between D_{j1} and D_{j2} . If this necessary condition is satisfied for all $j = \{1, 2, \dots, k\}$ with $k \leq p$, then the hybrid system is said to accept a chattering on switching intersection. As each discontinuity surface Γ_j splits the phase domain into two different invariants $D_{j1} \in \mathbb{R}^n$ and $D_{j2} \in \mathbb{R}^n$, the entire continuous domain of the hybrid system \mathcal{H} is then partitioned into 2^p open convex regions $D_q \in \mathbb{R}^n$, in which the solution trajectory flow is governed by the dynamics $f_q(t, x)$, where $q = 1, \dots, 2^p$, and p is the total number of the intersected switching manifolds Γ_j . It is assumed that f_q are smooth in the state x for all D_q . For more details on how the chattering occurs on switching intersection, we refer the reader to (Aljarbough and Cailaud, 2015b).

3.1 Chattering Detection

Upon crossing a switching manifold Γ_j , the behavior of the solution trajectory can uniquely be characterized by the gradients of the continuous-time behavior according to the dynamics f_{j1} and f_{j2} in a small neighborhood on the both sides of Γ_j . This is given by the normal projections of the dynamics f_{j1} and f_{j2} onto Γ_j (i.e. directional derivatives or Lie derivatives $\mathcal{L}_{f_j} g_j(t, x)$), given by

$$f_{j1}^{\perp j}(t, x) = \mathcal{L}_{f_{j1}} g_j(t, x) = \left(\frac{\partial g_j(t, x)}{\partial x} \right) \cdot f_{j1}(t, x) \quad (12)$$

$$f_{j2}^{\perp j}(t, x) = \mathcal{L}_{f_{j2}} g_j(t, x) = \left(\frac{\partial g_j(t, x)}{\partial x} \right) \cdot f_{j2}(t, x) \quad (13)$$

The sufficient condition for the hybrid system \mathcal{H} to exhibit a chattering back and forth between the two domains D_{j1} and D_{j2} requires that the necessary condition of chattering is satisfied (equation 9), as well as the following two constraints:

1. a zero crossing on Γ_j (state event) is detected in the integration time interval $[t_i, t_{i+1}]$, that is,

$$g_j(t_i, x_i) \cdot g_j(t_{i+1}, x_{i+1}) < 0 \quad (14)$$

2. the scalar inner product of the normal projections $f_{j1}^{\perp j}(t_i, x_i)$ and $f_{j2}^{\perp j}(t_{i+1}, x_{i+1})$ is strictly negative,

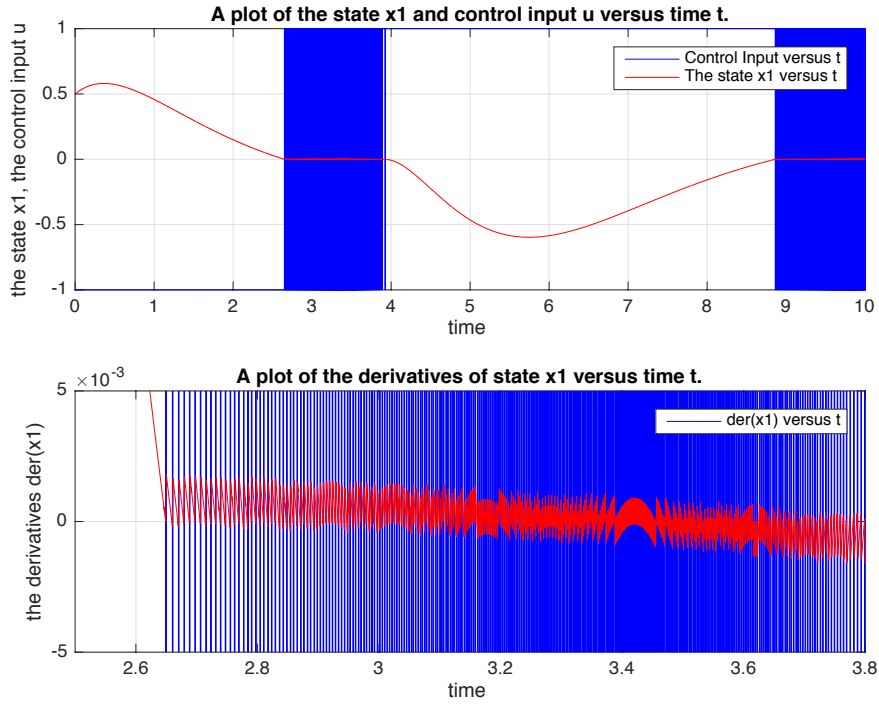


Figure 1. Fixed time step simulation of Example 1 in Acumen without event localization for $\beta = 0.5$ and $x_0 = [0.5 \ 3 \ 0.1]^T$: Up: time evolution of the event function and the control input with high chattering oscillation. Down: zoom on the first chattering window around the switching surface $x_1(t) = 0$.

that is, the projections of the two different dynamics (normal onto Γ_j), before and after the zero-crossing, have opposed signs (Figure 2),

$$f_{j1}^{\perp j}(t_i, x_i) \cdot f_{j2}^{\perp j}(t_{i+1}, x_{i+1}) < 0 \quad (15)$$

A chattering takes place on the intersection of $k \in \mathbb{N}$ switching manifolds Γ_j if the necessary condition of chattering is satisfied, and for all $j = 1, 2, \dots, k \leq p$, the following three constraints are satisfied:

$$g_j(t_i, x_i) \cdot g_j(t_{i+1}, x_{i+1}) < 0 \quad (16)$$

$$g_j(t_{i+1}(\sigma)) = \kappa; \quad \sigma \in (0, 1); \quad \kappa \in (-\varepsilon, \varepsilon) \quad (17)$$

$$f_{j1}^{\perp j}(t_i, x_i) \cdot f_{j2}^{\perp j}(t_{i+1}, x_{i+1}) < 0 \quad (18)$$

In this case of chattering on the intersection of finitely many switching manifolds, the execution of the hybrid system \mathcal{H} chatters back and forth between all the domains D_q in the neighborhood of the intersection.

3.2 Chattering Elimination

One way to prevent the chattering is to keep the solution trajectory in a sliding motion on the switching manifold/intersection on which the chattering occurs. An additional mode, sliding mode, can be inserted into the system to represent the equivalent chattering-free dynamics. For all the switching manifolds Γ_j , the dynamics

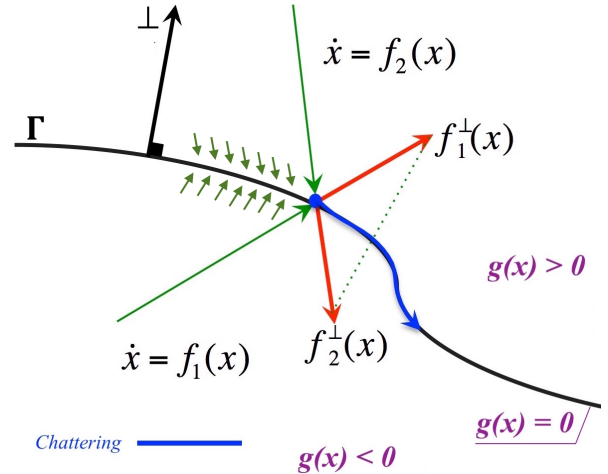


Figure 2. The chattering between two dynamics along a switching surface.

$\dot{x} = f_j(t, x)$ can be replaced by a differential inclusion $\dot{x} \in \eta(x)$ given as a convex set containing all the limit values of $f_j(x)$ for small neighbor $x(t, x) \notin \Gamma_j$ approaching Γ_j from the both sides (Biák et al., 2013). Equation 3 can be replaced then by:

$$\eta_j \in \frac{1 - \delta_j(g_j(t, x))}{2} \cdot f_{j1}(t, x) + \frac{1 + \delta_j(g_j(t, x))}{2} \cdot f_{j2}(t, x) \quad (19)$$

where $\delta_j(\cdot)$ is a multi-valued sign function given by:

$$\delta_j(g_j(t,x)) = \begin{cases} -1 & \text{for } g_j(t,x) < 0 \\ (-1, 1) & \text{for } g_j(t,x) = 0 \\ 1 & \text{for } g_j(t,x) > 0 \end{cases} \quad (20)$$

Roughly speaking, when a chattering occurs on Γ_j , we seek a smooth function δ_j , taking the value $\delta_j(g_j(t,x)) \in (-1, 1)$, so that the new equivalent chattering-free dynamics f_{jCHF} is given for $\delta_j(g_j(t,x)) \in (-1, 1)$ by:

$$f_{jCHF} = \frac{1 - \delta_j(g_j(t,x))}{2} \cdot f_{j1}(x) + \frac{1 + \delta_j(g_j(t,x))}{2} \cdot f_{j2}(x) \quad (21)$$

The idea behind forcing the solution trajectory to stay on the swicthing manifold during chattering execution is by forcing the normal projection of the equivalent chattering-free dynamics onto the swicthing manifold Γ_j to be tangential to Γ_j , that is,

$$f_{jCHF}^\perp(t,x) = \left(\frac{\partial g_j(t,x)}{\partial x} \right) \cdot f_{jCHF}(t,x) = 0 \quad (22)$$

which implies

$$\left[\frac{1 - \delta_j(g_j(t,x))}{2} \quad \frac{1 + \delta_j(g_j(t,x))}{2} \right] \cdot \begin{bmatrix} f_{j1}^\perp(t,x) \\ f_{j2}^\perp(t,x) \end{bmatrix} = 0 \quad (23)$$

and then

$$\delta_j(g_j(t,x)) = \frac{f_{j1}^\perp(t,x) + f_{j2}^\perp(t,x)}{f_{j1}^\perp(t,x) - f_{j2}^\perp(t,x)} \quad (24)$$

where $f_{j1}^\perp(t,x)$ and $f_{j2}^\perp(t,x)$ are given in equation 12 and equation 13, respectively.

By the substitution of equation 24 in equation 21, the equivalent chattering-free dynamics is given then by:

$$f_{jCHF}(t,x) = \frac{f_{j1}^\perp(t,x) \cdot f_{j2}(t,x) - f_{j2}^\perp(t,x) \cdot f_{j1}(t,x)}{f_{j1}^\perp(t,x) - f_{j2}^\perp(t,x)} \quad (25)$$

A smooth exist from sliding takes place instantly at the time instant at which the sufficient condition of chattering is no longer satisfied, that is, when either $f_{j1}^\perp(t,x)$ or $f_{j2}^\perp(t,x)$ starts to change its signs (i.e. when either $f_{j1}^\perp(t,x) = 0$ or $f_{j2}^\perp(t,x) = 0$).

Once a chattering execution is detected during the simulation process, the following Algorithm 1 is employed to generate the chattering-free dynamics internally in the simulation loop of the simulator. The number of iterations need to be performed by Algorithm 1 to compute the chattering-free dynamics is equal to the total number of the switching manifolds Γ_j on which the chattering occurs instantly. That is, when the system chatters between two dynamics, i.e. a chattering onto

a single switching manifold (as in Example 1), the equivalent chattering-free dynamics will be generated by Algorithm 1 in one iteration.

The main benefit of the iterative approach of Algorithm 1 is that it allows us to eliminate chattering efficiently in run-time simulation without any need to modes enumeration, even when the chattering is occurring on the intersection $\Delta = \bigcap_j(\Gamma_j), j = 1, 2, \dots, p$ of a large number p of intersected switching manifolds. Another benefit is that there is no need to solve stiff nonlinear equations for the computation of the chattering-free coefficients $\delta_j(g_j(t,x))$ in case of chattering on switching intersection with $p > 1$.

Data: Discontinuous dynamics $f(t,x)$, swicthing functions $g_j(t,x)$.

Result: $f_{\Delta CHF}(t,x) = f_{jCHF}(t,x)$

Initialization:

$j = 1$;

$f(x(t)) = f_j(x(t))$ (equation 9);

while $j \leq p$ **do**

Use $f_j(x(t))$ to build a differetial inclusion η_j (equation 19);

Compute $f_{jCHF}(t,x)$ (equations 21 to 25);

Set $f_j(x(t)) = f_{jCHF}(t,x)$;

$j = j + 1$;

Repeat;

end

Algorithm 1: How to generate the equivalent chattering-free dynamics $f_{\Delta CHF}(t,x)$.

In the following two simple examples we illustrate the functionality of Algorithm 1 in case of chattering on switching intersection.

Example 2:

Consider the simplest case of chattering onto the intersection of two switching manifolds, Γ_1 and Γ_2 , defined as the zeros of a set of scalar functions $g_1(t,x) = x_1(t)$ and $g_2(t,x) = x_2(t)$, respectively.

$\dot{x}_1 = 0$ **init** $-sgn(g_{10})$ **reset** $[-1;1]$ **every up** $[g_1;-g_1]$

$\dot{x}_2 = 0$ **init** $-sgn(g_{20})$ **reset** $[-1;1]$ **every up** $[g_2;-g_2]$

$g_1 = x_1$ **init** g_{10} ; $g_2 = x_2$ **init** g_{20}

where the zero-crossing is described as an expression of the form **up**(z) that becomes true when the sign of the event function $z(t,x)$ switches from negative to positive during an execution, that is, **up**(z) = *True* if $z(t_{i-1}, x_{i-1}) \leq 0 \wedge z(t_i, x_i) > 0$ (Schrammel, 2012). In this example, the trajectories initialized outside the origin reach the origin in finite time and with an infinite number of crossings of the switching surfaces $x_1(t) = 0$ and $x_2(t) = 0$. The finite time convergence is easy to establish as the time intervals between two switches satisfy a geometric series and consequently have a finite sum. This system has also an infinity of spontaneous switches from the origin, that is, there is an infinity of trajectories which start with the initial data (0,0), and except for the trivial solution that stays at the origin, they all cross the switching surfaces an infinity of times.

To generate the intersection chattering-free dynamics $f_{\Delta_{CHF}}(t, x)$ on the intersection (the origin) $\Delta = \Gamma_1 \cap \Gamma_2$, Algorithm 1 performs two iterations:

- In **Iteration1**, the algorithm computes the equivalent chattering-free dynamics on Γ_1 (equation 26).
- In **Iteration2**, the algorithm computes the equivalent chattering-free dynamics on the intersection $\Delta = \Gamma_1 \cap \Gamma_2$ (equation 27).

$$f_{1_{CHF}}(t, x) = \begin{bmatrix} 0 \\ \left\{ \begin{array}{l} -1 \text{ for } x_2(t) > 0 \\ 1 \text{ for } x_2(t) < 0 \end{array} \right\} \end{bmatrix} \quad (26)$$

$$f_{\Delta_{CHF}}(t, x) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (27)$$

Example 3: Stick-Slip Frictional System

Consider the following non-smooth mechanical system with friction elements.

$$f(x) = \left\{ \begin{array}{l} \dot{x}_{m_1} = v_{m_1} \\ \dot{v}_{m_1} = \frac{1}{m_1} \mathcal{F}_1 \\ \dot{x}_{m_2} = v_{m_2} \\ \dot{v}_{m_2} = \frac{1}{m_2} (u - kx_{m_2} - \mathcal{F}_1 - \mathcal{F}_2) \\ \dot{x}_{m_3} = v_{m_3} \\ \dot{v}_{m_3} = \frac{1}{m_3} \mathcal{F}_2 \end{array} \right\} \quad (28)$$

In this example, the entire discontinuity region is given as the union of two transversally intersected swithing manifolds Γ_1 and Γ_2 defined as the zeros of a set of the scalar functions $g_1(t, x) = v_{m_2}(t) - v_{m_1}(t)$ and $g_2(t, x) = v_{m_2}(t) - v_{m_3}(t)$, respectively.

$\mathcal{F}_1 = 0$ **init** $F_{c_1} \text{sgn}(g_{10})$ **reset** $[F_{c_1}; -F_{c_1}]$ **every up** $[g_1; -g_1]$
 $\mathcal{F}_2 = 0$ **init** $F_{c_2} \text{sgn}(g_{20})$ **reset** $[F_{c_2}; -F_{c_2}]$ **every up** $[g_2; -g_2]$
 $g_1(t, x) = v_{m_2}(t) - v_{m_1}(t)$ **init** g_{10}
 $g_2(t, x) = v_{m_2}(t) - v_{m_3}(t)$ **init** g_{20}

We have $p = 2$ intersected swithing manifolds. The algorithm, then, performs two iterations to generate $f_{\Delta_{CHF}}(t, x)$.

The output of **Iteration1**:

$$f_{1_{CHF}}(t, x) = \left\{ \begin{array}{l} \dot{x}_{m_1} = v_{m_1} \\ \dot{v}_{m_1} = \frac{1}{m_1+m_2} (u - kx_{m_2} - \mathcal{F}_2) \\ \dot{x}_{m_2} = v_{m_2} \\ \dot{v}_{m_2} = \frac{1}{m_1+m_2} (u - kx_{m_2} - \mathcal{F}_2) \\ \dot{x}_{m_3} = v_{m_3} \\ \dot{v}_{m_3} = \frac{1}{m_3} \mathcal{F}_2 \end{array} \right\} \quad (29)$$

The output of **Iteration2**:

$$f_{\Delta_{CHF}}(t, x) = \left\{ \begin{array}{l} \dot{x}_{m_1} = v_{m_1} \\ \dot{v}_{m_1} = \frac{1}{m_1+m_2+m_3} (u - kx_{m_2}) \\ \dot{x}_{m_2} = v_{m_2} \\ \dot{v}_{m_2} = \frac{1}{m_1+m_2+m_3} (u - kx_{m_2}) \\ \dot{x}_{m_3} = v_{m_3} \\ \dot{v}_{m_3} = \frac{1}{m_1+m_2+m_3} (u - kx_{m_2}) \end{array} \right\} \quad (30)$$

4 Generic Implementation Scheme in FMI 2.0

In this section, a prototype implementation is sketched for applying the chattering-free computational framework from the previous section to Functional Mock-Up Interface v2.0 for Model Exchange. The goal is to provide in FMI, a rigorous chattering-free simulation, in run-time, without modes enumeration, for any chattering FMU which may be either generic or generated from a modeling environment in which chattering models can not be simulated rigorously, whenever the compliance with FMI specification for model exchange is fulfilled. The FMI chattering-free implementation has been performed by embedding the chattering detection and elimination algorithm in the Event Mode of the FMI.

4.1 The Functional Mock-Up Interface FMI

FMI is an open standard for model exchange and co-simulation between multiple software systems. This new standard, resulting from the ITEA2 project MOD-ELISAR, in 2010, is a response to the industrial need to connect different environments for modeling, simulation and control system design. It is used to create an instance of a model which can be loaded into any simulator providing an import function for FMI. A software instance compatible to the FMI is called an FMU. An FMU is distributed as a compressed archive with a .fmu file extension. It contains a concrete mathematical model described by differential, algebraic and discrete equations with possible events of a dynamic physical system. An FMU consists basically of two parts:

- an XML format for model interface information,
- C API model interface functions according to the FMI specification, for model execution.

The XML format, specified by an XML schema conforming to the FMI specification, contains all static information about model variables, including names, units and types, as well as model meta data. The C API, on the other hand, contains C functions for data management, as setting and retrieving parameter values, and evaluation of the model equations. The implementation of the C API may be provided either in C source code format or in binary forms (e.g. in the form of Windows dynamic link library .dll or a Linux shared object library .so files) to protect the model developer's intellectual property. Additional parts can be added and compressed into the FMU, as the documentation and the icon of the model. FMUs can be written manually or can be generated automatically from a modelling environment.

4.2 Chattering-Free Support in FMI

In this section we explain the functionality of our chattering-free FMI framework as well as how the chattering behavior is treated internally in the main simulation loop of interface without any need to add hysteresis to the event indicators in the FMU.

Prior to a simulation experiment, the model has to be instantiated. This includes extracting the files in the FMU, loading the DLL and XML files and calling the instantiation function available in the DLL. A model can be instantiated multiple times for which the function `fmi2SetupExperiment` is provided.

Simulating an FMI model means to split the solution computation process in three different phases, categorized according to three modes: Initialization Mode, Continuous-Time Mode, and Event Mode.

In the Initialization Mode, the model is initialized with `finit(⋯)` by calling the FMI function `fmi2EnterInitializationMode` in order to compute the continuous-time states and the output variables at the initial time t_0 . There are FMI functions used in this Mode as `fmi2GetContinuousStates` as well as functions for setting and getting values for Type Real, Integer, String, and Boolean values, of the form `fmi2(Get/Set)(Type)`. The input arguments to the Initialization Mode functions consist of the all variables that are declared with "input" and "independent" causality in the FMU XML files, as well as all variables that have a start value with `initial = "exact"`. Once the model is instantiated and initialized it can be simulated.

The main simulation loop starts once the FMI function `fmi2ExitInitializationMode` is called. The simulation is performed by calculating the derivatives and updating time and states in the model via the FMI functions `fmi2SetContinuousStates`, `fmi2SetTime`, `fmi2GetContinuousStates`, `fmi2GetDerivatives`, as well as the four `fmi2(Get/Set)(Type)` functions mentioned above. To retrieve or set variable data during a simulation, value-references are used as keys. All variables are connected to a unique number defined and provided in the FMU XML-file. This number can then be used to retrieve information about variables via functions in the interface or can be used to set input values during a simulation. During the simulation, events are monitored via the functions `fmi2GetEventIndicators` and `fmi2CompletedIntegratorStep`. Events are always triggered from the environment in which the FMU is called, so they are not triggered inside the FMU (Blochwitz et al., 2012). Step-events are checked in the model after calling the completed step function `fmi2CompletedIntegratorStep` when an integration step was successfully completed. A step event occurs if indicated by the return argument `nextMode = Event-Mode`. For capturing state events during continuous integration, the algorithm monitors, at every completed

integrator step, the set of event indicator functions $z_j(t, x)$ provided in the function `fmi2GetEventIndicators`. All event indicators $z_j(t, x)$ are piecewise continuous and are collected together in one vector of real numbers (Blochwitz et al., 2012). A state event occurs when the event indicator changes its domain from $z_j(t, x) > 0$ to $z_j(t, x) \leq 0$ or from $z_j(t, x) \geq 0$ to $z_j(t, x) < 0$. If a domain change of one of the indicator functions is detected, a state event has occurred and the simulation environment then informs the FMU by calling the function `fmi2NewDiscreteStates`.

During the continuous integration, we distinguish, for each time integration step, the following cases:

1. If $z_j(t_i, x_i) \cdot z_j(t_{i+1}, x_{i+1}) > 0$ for all $j = 1, 2, \dots, p$ where p is the total number of the event indicators, then we continue integrating the system with the same dynamics.
2. If there exist $j \in \{1, 2, \dots, p\}$ for which: $\forall \tau \in [t_i, t_{i+1}[: z_j(\tau, x) < 0 \wedge \exists m \leq \text{margin} : \forall \tau \in [t_{i+1}, t_{i+1} + m] : z_j(\tau, x) \geq 0$, or $\forall \tau \in [t_i, t_{i+1}[: z_j(\tau, x) > 0 \wedge \exists m \leq \text{margin} : \forall \tau \in [t_{i+1}, t_{i+1} + m] : z_j(\tau, x) \leq 0$, a zero crossing in the time interval $[t_i, t_i + 1]$ is then detected. The algorithm performs an iteration over time between the previous and the actual completed integrator step, in order to determine the time instant of the switching point up to a certain precision. In this case we have a continuous smooth switching function $z_j(t_{i+1}(\sigma))$ taking opposed signs at $\sigma = 0$ and $\sigma = 1$ and therefore there exist a zero at $\sigma_e \in (0, 1)$ which defines the state event $x_e = x_{i+1}(\sigma_e) \in \Gamma_j$, where $\Gamma_j = \{x \in \mathbb{R}^n \mid z_j(t, x) = 0\}$ is the switching surface.
3. The case in which there exist finitely many event indicator functions $z_j(t, x)$, $j \in \{1, 2, \dots, p\}$, all satisfy: $\forall \tau \in [t_i, t_{i+1}[: z_j(\tau, x) < 0 \wedge \exists m \leq \text{margin} : \forall \tau \in [t_{i+1}, t_{i+1} + m] : z_j(\tau, x) \geq 0$, or $\forall \tau \in [t_i, t_{i+1}[: z_j(\tau, x) > 0 \wedge \exists m \leq \text{margin} : \forall \tau \in [t_{i+1}, t_{i+1} + m] : z_j(\tau, x) \leq 0$, and $z_j(\sigma_e) = 0$ for all $j = 1, 2, \dots, k$ where $k \leq p$ and $\sigma_e \in (0, 1)$, indicates that the solution trajectory has reached the intersection of $k \leq p$ of transversally intersected $\mathbb{R}^{(n-1)}$ switching manifolds Γ_j .

At an event, the function `fmi2NewDiscreteStates` has to be called. This function updates and re-initializes the model in order for the simulation to be continued. Information is also given about if the states have changed values, if new state variables have been selected and information about upcoming time events.

In our chattering-free semantics, the master algorithm has to decide, at the state event, whether the solution trajectory should cross the switching surface transversally or slide on it (to eliminate chattering). The computation of the chattering-free solution is split in two phases: i) chattering detection, and ii) chattering elimination.

The chattering detection phase starts once a state event is detected and located. The algorithm inspects whether the state event is a chattering event or not. This implies checking, at the state event, whether or not the sufficient condition of chattering is satisfied, by analyzing the gradients of the continuous time behavior before and after the state event. For doing so, the directional derivatives of the dynamics (the normal projection of dynamics onto the switching surface) should be computed and evaluated at the beginning and at the end of the completed integration step at which the state event has been detected. A state event $x_e \in \Gamma_j$ detected in the time interval $[t_i, t_{i+1}]$ is said to be a chattering event if the condition: $NP_j(t_i, x_i) \cdot NP_j(t_{i+1}, x_{i+1}) < 0$ is satisfied, where $NP_j(t_i, x_i) = f_{j1}^{\perp j}(t_i, x_i)$, respectively $NP_j(t_{i+1}, x_{i+1}) = f_{j2}^{\perp j}(t_{i+1}, x_{i+1})$, is the normal projection of the dynamics f_{j1} (before the state event), respectively f_{j2} (after the state event), onto the switching manifold $\Gamma_j = \{x \in \mathbb{R}^n \mid z_j(t, x) = 0\}$, at t_i , respectively t_{i+1} . A chattering occurs on a switching intersection $\Delta = \bigcap_j \Gamma_j$ (i.e. intersection state event), detected in the time interval $[t_i, t_{i+1}]$, if for all $j = 1, 2, \dots, k$: $NP_j(t_i, x_i) \cdot NP_j(t_{i+1}, x_{i+1}) < 0$, where as mentioned in Section 4 (equation 12 and equation 13), the normal projection $NP_j(t_i, x_i)$, respectively $NP_j(t_{i+1}, x_{i+1})$, is computed as a scalar product of the dynamics $f(t_i, x_i)$, respectively $f(t_{i+1}, x_{i+1})$, with the partial derivatives the event indicator function z_j . The partial derivatives of z_j are computed numerically in the integration step $[t_i, t_{i+1}]$ at which the state event is detected. As the nature of our chattering detection semantics is to compare the sign of the directional derivatives (normal projections) at the beginning and the end of the time integration step $[t_i, t_{i+1}]$ in which a state event is occurred, and if it changes, declare a chattering event, the environment then should be able to have an access to the dynamics at t_i (i.e. in the previous domain before the event) and at t_{i+1} (i.e. in the next domain after the event). For doing so, we use two arrays, $xdot_{pre}$ and $xdot_{post}$, where during the continuous integration, and for each time step $[t_i, t_{i+1}]$ in which a state event has been detected, the dynamics $f(t_i, x_i)$, and $f(t_{i+1}, x_{i+1})$ are computed and evaluated via `fmi2GetDerivatives` and then stored in $xdot_{pre}$, and $xdot_{post}$, respectively. In the chattering elimination phase, Algorithm 1 (Section 3) is employed in the environment's master algorithm in order to compute the smooth equivalent chattering-free dynamics internally giving the dynamics before and after the state event, f_{j1} and f_{j2} , respectively, as well as the event indicators $z_j(t, x)$. Once the solution is at the final time of a simulation, the function `fmi2Terminate` is called to terminate the simulation. After a simulation is terminated, memory has to be deallocated. The function `fmi2FreeInstance` is then called to deallocate all memory that have been allocated since the initialization.

5 Simulation Results

Figure 3 shows the chattering-free simulation of the system in Example 1 for the data set: $\beta = 0.5$, $x_0 = [0.5, 3, 1]^T$. During a simulation time $t = 10$, 241685 chattering events have been detected and replaced by two sliding windows. The first chattering event is detected at $t = 2.649$ (Figure 4), the algorithm switches to integrate the the system with the chattering-free dynamics generated internally. In Figure 5 and Figure 6, the Stick-Slip frictional system in Example 3 was simulated for $m_1 = m_2 = m_3 = 1[kg]$, $k = 0.88[N \cdot m^{-1}]$, $F_{c1} = 0.01996[N]$ $F_{c2} = 0.062[N]$, and $x_0 = [0.8295 \ 0.8491 \ 0.3725 \ 0.5932 \ 0.8726 \ 0.9335]^T$. The external force u was simulated as a sine wave of frequency of $\omega = 0.073[rad/sec]$. The sliding bifurcations depend on the effect of the external force u and the level of Coulomb frictions F_{c1} and F_{c2} . At the time instant $t = 32.69 \ sec$, two masses m_2 and m_3 stick together and the solution trajectory start a sliding motion on the switching manifold $\Gamma_2 = \{x \in \mathbb{R}^n : (v_{m_2}(t) - v_{m_3}(t) = 0)\}$ (Figure 5). A smooth exit from sliding on Γ_2 to evolve into q_3 was detected at the time instant $t = 77.23 \ sec$. A transversality switching from the discrete state q_3 to the discrete state $q_1 = \{x \in \mathbb{R}^n : (v_{m_2}(t) - v_{m_1}(t) > 0) \wedge (v_{m_2}(t) - v_{m_3}(t) > 0)\}$ at the intersection $\Delta = \Gamma_1 \cap \Gamma_2$ was detected at $t = 92.04 \ sec$.

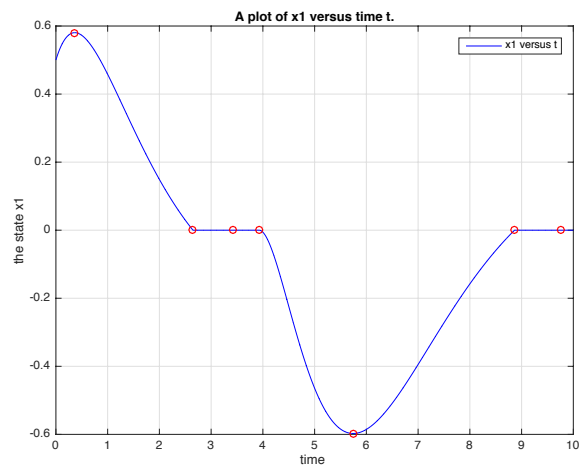


Figure 3. The time evolution of the continuous state x_1 with chattering-free simulation.

6 Conclusions

In this paper we presented an FMI-based computational framework, and a prototypical implementation of a generic chattering-free FMI for robust and reliable detection and elimination "On the Fly" of chattering behavior in run-time simulation of non-smooth hybrid systems, without modes enumeration, and without any need to add a small hysteresis to the event indicators in the FMUs.

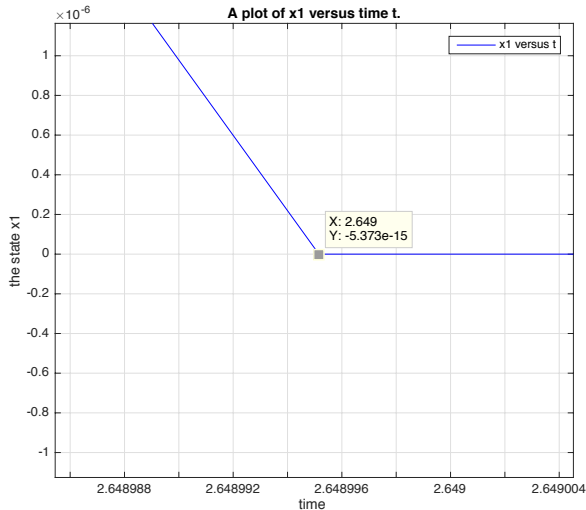


Figure 4. A smooth entering to sliding: First chattering state event detected at $t = 2.649$.

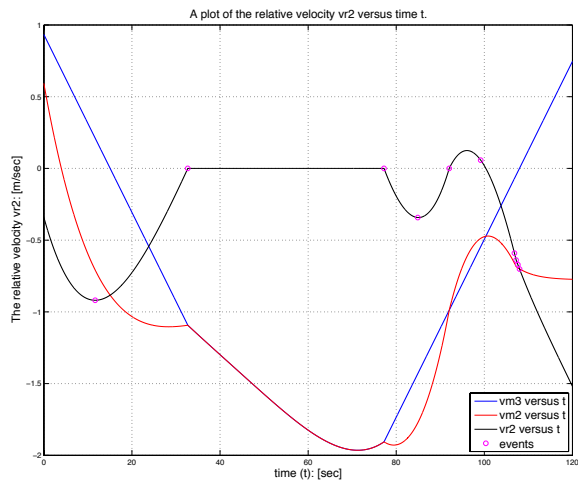


Figure 5. A chattering-free simulation of Example 3: The time evolution of the relative velocity $v_{m_2}(t) - v_{m_3}(t)$.

The developed chattering-free FMI switches between the transversality modes and the sliding modes simulation automatically, integrates each particular state appropriately, and localizes the non-smooth structural changes in the system in an accurate way. It treats the chattering non-smoothness in the trajectory of the state variables by a smooth correction after each integration time-step. Our chattering-free FMI can robustly handle the case of chattering on switching intersection without any need to solve stiff nonlinear equations for the computation of the chattering-free coefficients. Furthermore, a guidance for development of a hybrid chattering-free version of the FMI standard, was provided in this paper. Finally, the simulation results on a set of representative examples have demonstrated that our FMI-based chattering-free framework is efficient and precise enough to provide a

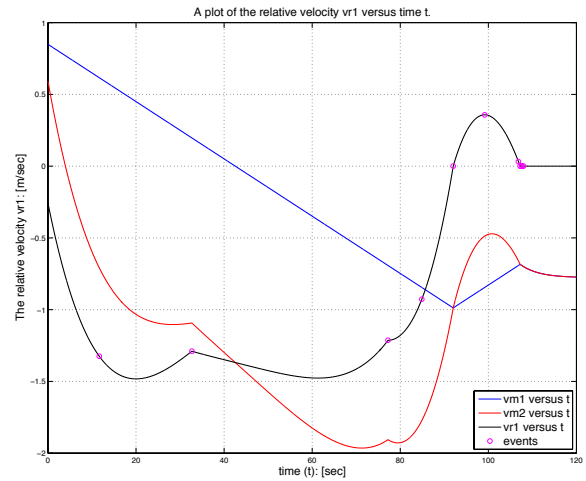


Figure 6. A chattering-free simulation of Example 3: The time evolution of the relative velocity $v_{m_2}(t) - v_{m_1}(t)$.

rigorous chattering-free simulation for any generic chattering Functional Mockup Unit (FMU) conforming to the FMI standard v2.0 Specification for model exchange.

Acknowledgements

This work was supported by the ITEA2 MODRIO project under contract N° 6892, and the ARED grant of the Conseil Régional de Bretagne.

References

- Ayman Aljarbough and Benoit Caillaud. On the regularization of chattering executions in real time simulation of hybrid systems. *Baltic Young Scientists Conference Proceedings*, pages 49–66, 2015a. URL <https://hal.archives-ouvertes.fr/hal-01246853v2>.
- Ayman Aljarbough and Benoit Caillaud. Robust simulation for hybrid systems: Chattering path avoidance. *Linköping Electronic Conference Proceedings*, 119(018):175–185, 2015b. ISSN 1650-3686. doi:10.3384/ecp15119175. URL <http://www.ep.liu.se/ecp/119/018/ecp15119018.pdf>.
- Martin Biák, Tomáš Hanus, and Drahoslava Janovská. Some applications of filippov’s dynamical systems. *Journal of Computational and Applied Mathematics*, 254:132–143, 2013. ISSN 0377-0427. doi:http://dx.doi.org/10.1016/j.cam.2013.03.034. URL <http://www.sciencedirect.com/science/article/pii/S0377042713001428>.
- Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph ClauB, Hilding Elmquist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. *In Pro-*

ceedings of 9th International Modelica Conference, Munich, Germany, 076(017):173–184, 2012. ISSN 1650-3686. doi:10.3384/ecp12076173. URL <http://www.ep.liu.se/ecp/076/017/ecp12076017.pdf>.

Chaohong Cai, Rafal Goebel, Ricardo Sanfelice, and Andrew Teel. *Hybrid systems: limit sets and zero dynamics with a view toward output regulation*. Springer-Verlag, 2008. URL <https://hybrid.soe.ucsc.edu/files/preprints/21.pdf>.

Mario di Bernardo, Chris J. Budd, Alan R. Champneys, Piotr Kowalczyk, Arne B. Nordmark, Gerard Olivari Tost, and Petri T. Piironen. Bifurcations in nonsmooth dynamical systems. *SIAM Review*, 50(4):629–701, 2008. doi:10.1137/050625060. URL <http://dx.doi.org/10.1137/050625060>.

A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Springer Netherlands, 1988. ISBN 978-94-015-7793-9.

K.H. Johansson, A.E. Barabanov, and K.J. Astrom. Limit cycles with chattering in relay feedback systems. *Automatic Control, IEEE Transactions on*, 9(018):1414–1423, 2002. ISSN 0018-9286. doi:10.1109/TAC.2002.802770. URL <http://www.ep.liu.se/ecp/119/018/ecp15119018.pdf>.

Remco I. Leine and Henk Nijmeijer. *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*. Springer Berlin Heidelberg, 2004. ISBN 978-3-642-06029-8.

John Lygeros, Claire Tomlin, and Shankar Sastry. *Hybrid Systems: Modeling, Analysis and Control*. Lecture Notes on Hybrid Systems, 2008. URL <http://inst.cs.berkeley.edu/~ee291e/sp09/handouts/book.pdf>.

Peter Schrammel. *Logico-Numerical Verification Methods for Discrete and Hybrid Systems*. PhD dissertation, 2012.

Vadim I. Utkin. *Sliding Mode in Control and Optimization*. Springer Berlin Heidelberg, 1992. ISBN 978-3-642-84379-2.

D. Weiss, T. Küpper, and H.A. Hosham. Invariant manifolds for nonsmooth systems with sliding mode. *Mathematics and Computers in Simulation*, 110:15–32, 2015. doi:<http://dx.doi.org/10.1016/j.matcom.2014.02.004>. URL <http://www.sciencedirect.com/science/article/pii/S037847541400041X>.

Jun Zhang, Karl Henrik Johansson, John Lygeros, and Shankar Sastry. Zeno hybrid systems. *International Journal of Robust and Nonlinear Control*, 11(5):435–451, 2001. ISSN 1099-1239. doi:10.1002/rnc.592. URL <http://dx.doi.org/10.1002/rnc.592>.

Acceleration of FMU Co-Simulation On Multi-core Architectures

Salah Eddine Saidi¹ Nicolas Pernet¹ Yves Sorel² Abir Ben Khaled¹

¹IFP Energies nouvelles, Rueil-Malmaison, France,

{salah-eddine.saidi,nicolas.pernet,abir.ben-khaled}@ifpen.fr

²INRIA, Paris, France, yves.sorel@inria.fr

Abstract

The design of cyber-physical systems is a complex process and relies on the simulation of the system behavior before its deployment. Co-simulation allows system designers to simulate a whole system composed of a number of interconnected subsystems. Traditionally, these models are modeled by experts of different fields using different tools, and then integrated into one environment to perform simulation at the system-level. This results in complex and heavy co-simulations and requires adequate solutions and tools in order to reduce the execution time. Unfortunately, most modeling tools perform only mono-core simulations and do not take advantage of the omnipresent multi-core processors. This paper addresses the problem of efficient parallelization of co-simulations. It presents a multi-core scheduling heuristic for parallelizing FMI-compliant models on multi-core processors. The limitations of this heuristic are highlighted and two solutions for dealing with them are presented. The obtained speed-up using each of these solutions is illustrated and discussed for further improvements.

Keywords: FMI, co-simulation, multi-core, scheduling, heuristic

1 Introduction

Cyber-physical systems incorporate a combination of computational elements which collaborate in order to control physical processes. The complex nature of such systems requires cost, time and effort-effective design methodologies; therefore predicting their behavior and functioning scenarios before testing the real system is becoming more and more an indisputable step. Co-simulation aids in achieving these requirements as it allows the assessment of the design of the system by imitating its behavior. It consists mainly in simulating, on a computer, the global behavior of a multi-physics system composed of a number of interconnected subsystems. System designers can then identify potential design flaws and correct them before deploying the system.

Co-simulation faces however a number of challenges.

Actually, the simulated system is described by several interacting models which are often developed by experts of different fields using different tools and following different design approaches. The diversity of modeling tools and involved teams makes the coupling of the models a complex task. In fact, co-simulation necessitates efficient synchronized communications between the models where each model must be able to detect and respond to events of other models. Thanks to the FMI (Functional Mock-up Interface) standard (Blochwitz et al., 2011), it is now possible to easily couple diverse models originating from different developers and tools. Nevertheless, executing FMI-compliant models raises some issues, which unless well handled, may reduce the co-simulation performance and limit the benefits of FMI.

One major issue is the question of how to reduce the co-simulation execution time. Integrating heterogeneous models into one environment usually results in a complex and heavy to execute co-simulation which increases the demand of processing power.

As is well-known, increasing CPU frequency by means of silicon integration has reached its possible limits and semiconductor manufacturers switched in last years to building multi-core processors, i.e. integrating multiple processors into one chip allowing parallel processing on a single computer. Multi-core processors can reduce the execution time of a computational task by dividing it into several subtasks and assigning a subset of subtasks to each core to be processed in parallel. Most simulators, however, have mono-core simulation kernels and do not take advantage of the computation power brought by multi-core architectures. Therefore, enabling parallel execution of heavy co-simulations on multi-core processors is keenly sought by the developers and the users of simulation tools. However, fulfilling this objective is not trivial and appropriate parallelization schemes need to be applied on co-simulation models in order to accelerate their execution on multi-core processors. It is worth noting that in this paper the term co-simulation is generic and is used to refer to the simulation of FMUs generated from FMI for Co-Simulation as well as FMI for Model Exchange.

FMI gives information about inputs and outputs relationships inside a model that is exported as an FMU

(Functional Mock-up Unit). An FMU is a package that encapsulates an XML file containing among other data the definitions of the model's variables, and a library defining the equations of the model as C functions. Input, output and state variables are updated by what we name "operations" which may call different functions provided by the FMU.

Given these features, various execution possibilities can be realized and the parallelization of co-simulation models on a multi-core processor can be seen as the following problem: Find an allocation of the different operations to the different cores and define an execution order, i.e. schedule the operations that are allocated to each core. When solving this problem, the utilization of the available cores has to be optimized in order to achieve the best acceleration. Using parallel computing terminology, the problem consists in finding a schedule for all the operations of the co-simulation on a multi-core processor. This paper deals with the problem of scheduling operations of heavy complex co-simulation models on multi-core processors in order to accelerate the simulation execution. It follows the approach presented in (Ben Khaled et al., 2014) by addressing two limitations of the previous work. First, an efficient multi-core scheduling can not be obtained without taking into account a good estimation of each operation's execution time. Second, the non-thread-safe implementation of FMUs prevent full exploitation of the potential parallelism of co-simulation graphs. Techniques for dealing with these limitations are here compared.

The rest of the paper is organized as follows. Next section presents related work on multi-core execution of simulations. Then our parallelization approach, firstly presented in (Ben Khaled et al., 2014), is described in section 3, including a discussion about its present limitations. The fourth section presents our contribution, including the use of a toolchain for profiling co-simulation graph parallelism and explores the theoretical gain in execution speed-up over different architectures. Theoretical results are discussed and compared to real co-simulation executions in xMOD¹. xMOD is a co-simulation and a virtual experimentation platform, which allows mixing stand-alone and tool coupling co-simulations and the optimization of complex models execution. It provides a user-friendly interface in order to extend the simulation use to non-experts and ensure the continuity from Model-in-the-Loop to Hardware-in-the-Loop simulations. The last section concludes the paper and gives an outlook into our ongoing and future work.

2 Related Work

In order to achieve simulation acceleration using multi-core execution, different approaches are possible and were already explored. From a user point of view, it is

¹<http://www.xmodsoftware.com/>

possible to modify the model design in order to prepare its multi-core execution, for example by using marked functions or Modelica extensions as in (Elmqvist et al., 2015; Gebremedhin et al., 2012). From a modeling tool provider point of view, if providing OpenMP ready libraries is possible, the key feature for simulation acceleration is to provide techniques which offer speed-up whatever the model is. Proposing parallel solvers or automatic parallel executions of model equations as in (Elmqvist et al., 2014; Sjölund et al., 2010) is also an efficient way. In this paper, we address the problem from a co-simulation tool provider point of view. In such a tool, the user connects different FMUs, embedding solvers or not. In this case, it is not possible to change the models, the solvers, or the modeling tools. Such FMU assembly defines a graph of operations and the main opportunity to improve the co-simulation execution is consequently to accomplish an automatic parallelization of this graph. As shown in (Ben Khaled et al., 2012), splitting a model into several FMUs, by isolating discontinuities, may reduce the simulation time, even in the case of a mono-core execution. (Ben Khaled et al., 2014) presented the RCOSIM approach. It consists in using each FMU information on input/output causality to build a graph, with an increased granularity and then exploiting the potential parallelism by using a heuristic to build an off-line multi-core schedule. This method has been tested on a real industrial model and significant speed-up was obtained. This approach was implemented in the co-simulation tool xMOD and is available in its 2015 release.

3 Parallelization approach

3.1 Principle

The parallelization concept of xMOD is based on a task DAG (Directed Acyclic Graph) scheduling approach. Thanks to FMI, it is possible to access information about the internal structure of a model encapsulated in an FMU. In particular, FMI allows the identification of Direct Feedthrough and Non Direct Feedthrough outputs of a model. Since connections between different models of the co-simulation are also known, all data dependencies between the operations are known. Figure 1 shows an example of two models and their inter and intra-model dependencies.

The co-simulation can be described by a DAG where each vertex represents one operation and each edge describes a precedence constraint between two operations. The approach proceeds in two steps: First, the co-simulation DAG is constructed and then, the operations are allocated to the available cores in such a way to minimize the makespan of the graph. The makespan corresponds the execution time of the whole DAG.

The transformation of each model into an operation

graph allows the parallelization of the model instead of considering it as an atomic block. Consequently the potential parallelism of the entire co-simulation is increased and can be better adapted to the hardware parallelism (number of cores in the case of a multi-core processor). The potential parallelism of a graph corresponds to vertices that are not dependent which characterize the partial order of the graph.

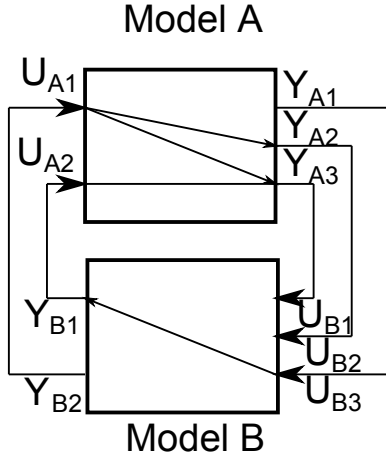


Figure 1. Inter and intra-model dependencies of two models.

3.2 Multi-core scheduling heuristic

The co-simulation DAG is built by exploring the relations between the models and between the operations of the same model. The operations are either *update_{output}*, *update_{input}* or *update_{state}*. An *update_{output}* operation corresponds to an FMI *Get* function that allows getting the value of an FMU output and an *update_{input}* operation corresponds to an FMI *Set* function which allows setting the value of an input. An *update_{state}* operation corresponds to calling FMI functions needed to perform an integration step (*SetTime*, *GetDerivatives*, and *SetContinuousStates*, etc., in the case of Model Exchange or *DoStep* in the case of Co-Simulation) (FMI development group, 2014). A vertex is created for each operation and edges are then added between vertices if a data dependency exists between the corresponding operations. This information can be extracted from the model's FMU. When using FMI 1.0 which does not give information about the dependencies between the state variables computation and the input and output variables computations, it is necessary that edges connect all *update_{input}* operations and the *update_{state}* operation of the same model, since all inputs at instant k need to be updated before updating the state to X_{k+1} . Furthermore, edges are placed between all *update_{output}* operations and the *update_{state}* operation of the same model, because the computation at instant k of an output Y_k must be performed with the same value of the state as for all the outputs belonging to the same model. Running the co-simulation consists in executing the graph repeatedly.

At each co-simulation step the whole graph is executed and a new execution of the graph cannot be started unless the previous one was totally finished. Figure 2 illustrates the graph constructed from the two models of Figure 1.

In order to achieve fast execution of the co-simulation on a multi-core processor, an efficient allocation and scheduling of the DAG vertices has to be performed. xMOD uses an off-line scheduling heuristic similar to the one proposed in (Grandpierre et al., 1999). (Ben Khaled et al., 2014) presented the use of this heuristic and the speed-up obtained by applying it on an industrial combustion engine model. The heuristic considers the execution time of each operation and aims at computing a schedule that minimizes the makespan of the graph.

Using the execution time C_i of each operation OP_i , the heuristic computes first the earliest start and end dates from the graph start denoted S_i and E_i , then the critical path $CP := \max E_i$ (Algorithm 1). After that, the latest start and end dates from the graph end denoted S_i^* and E_i^* and then the flexibility $F_i = CP - E_i - E_i^*$ are computed (Algorithm 2).

```

Initialization;
Set  $\Omega$  the set of all the operations;
Set  $O$  the set of operations without predecessors;
foreach  $OP_i \in O$  do
  |  $S_i := 0; E_i := S_i + C_i;$ 
end
Set  $O'$  the set of operations whose all immediate predecessors
were treated;
while  $O' \neq \emptyset$  do
  | foreach  $OP_i \in O'$  do
    | |  $S_i := \max(E_h : OP_h \rightarrow OP_i);$ 
    | | ( $OP_h$  are the immediate predecessors of  $OP_i$ );
    | |  $E_i := S_i + C_i;$  Remove  $OP_i$  from the set  $O'$ ;
    | | Add to the set  $O'$  all successors of  $OP_i$  for which all
    | | predecessors were already scheduled;
    | end
  | end
  |  $CP := 0;$ 
  | foreach  $OP_i \in \Omega$  do
    | | if  $CP < E_i$  then
    | | |  $CP := E_i;$ 
    | | end
  | end

```

Algorithm 1: Computation of S_i , E_i and CP

At each step, the heuristic computes for a given operation the schedule pressure on a specific core. The schedule pressure is the difference between the makespan increase, by allocating this operation to this core, and the operation's flexibility. The heuristic updates the set of candidate operations to be scheduled at each step. An operation is added to the set of candidate operations if it has no predecessor or if all of its predecessors have already been scheduled. The set of candidate operations holds the partial order associated to the graph. Then, for each candidate operation, the schedule pressure is computed on each core in order to find its best core, the one that minimizes the pressure. After this step, a list of candidate operation-best core pairs is obtained. Finally, the

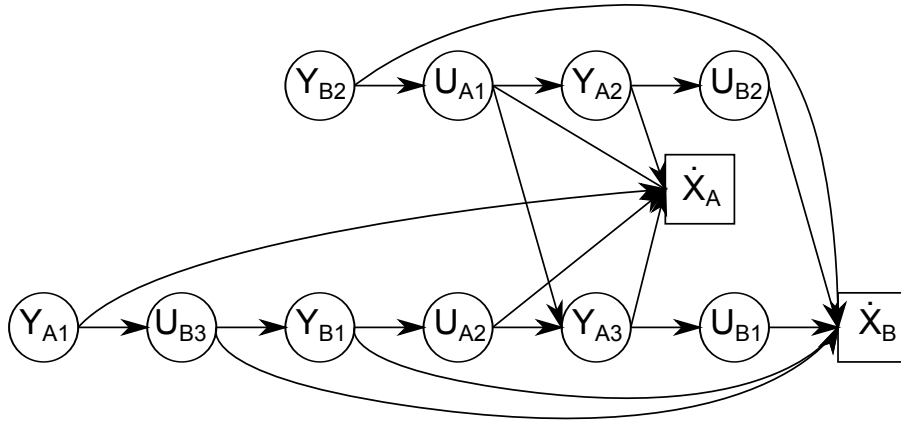


Figure 2. Dependency graph of the models of Figure 1.

```

Initialization;
Set  $\Omega$  the set of all the operations;
Set  $O$  the set of operations without successors;
foreach  $OP_i \in O'$  do
  |  $E_i^* := 0; S_i^* := E_i^* + C_i;$ 
end
Set  $O'$  the set of operations whose all immediate successors were
treated;
while  $O' \neq \emptyset$  do
  foreach  $OP_i \in O$  do
    |  $E_i^* := \max(S_h^* : OP_i \rightarrow OP_h);$ 
    | ( $OP_h$  are the immediate successors of  $OP_i$ );
    |  $S_i^* := E_i^* + C_i$ ; Remove  $OP_i$  from the set  $O'$ ;
    | Add to the set  $O'$  all predecessors of  $OP_i$  for which all
    | successors were already scheduled;
  end
end
foreach  $OP_i \in \Omega$  do
  |  $F_i := CP - E_i - E_i^*;$ 
end

```

Algorithm 2: Computation of S_i^* , E_i^* and F_i

operation with the largest pressure on its best core is selected and scheduled. The heuristic repeats this procedure until all operations are scheduled (Algorithm 3).

This heuristic has originally been used to implement critical hard real-time applications where the execution times are usually estimated as the WCET (Worst Case Execution Time). On the contrary, co-simulation is not safety critical and the main goal here is to achieve fast execution, so average computation times can be used. So far, execution times in xMOD are estimated based on the observation of practical examples as follows: *update_{state}* operations are by far more costly so they are assigned significantly higher execution times than *update_{output}* operations, whereas *update_{input}* operations are just data copy whose cost is negligible.

3.3 Limitations of the approach

Although the presented scheduling heuristic resulted in interesting co-simulation speed-ups, it has some limitations that have to be considered in the multi-core

```

Initialization;
Set  $\Omega$  the set of all the operations;
Set  $\Gamma$  the set of all the available cores;
Set  $O$  the set of operations without predecessors;
while  $O \neq \emptyset$  do
  foreach  $OP_i \in O$  do
    | Set  $cost_i$  to  $\infty$ ; (cost of  $OP_i$  is set to the maximum
    | value);
    | foreach  $Core_j \in \Gamma$  do
      |  $S'_{i,j} := \max(S_i, T_{Core_j});$  (new start date of  $OP_i$  when
      | executed on  $Core_j$ );
      |  $cost_{i,j} := S'_{i,j} + C_i + E_i^* - CP$ ; (cost of  $OP_i$  when
      | executed on  $Core_j$ );
      | if  $cost_{i,j} < cost_i$  then
      | | Set  $cost_i := cost_{i,j}$ ;
      | | Set  $BestCore_i := Core_j$ ;
      | end
    | end
  end
  Find  $OP_i$  with maximal  $cost_i$  in  $O$ ;
  Schedule  $OP_i$  on its core  $BestCore_i$ ;
  Set  $k := BestCore_i$ ;
   $T_{Core_k} := T_{Core_k} + C_i$ ; (Advance the time of  $Core_k$ );
  Remove  $OP_i$  from the set  $O$ ;
  Add to the set  $O$  all successors of  $OP_i$  for which all
  predecessors are already scheduled;
end

```

Algorithm 3: Multi-core scheduling heuristic

scheduling problem in order to obtain better performances. First, so far, the multiprocessor scheduling heuristic uses empiric operations execution times. By using realistic execution times for each operation, the multi-core execution of the simulation should be improved. In this paper, we present some results, based on a profiling technique.

Second, FMI standard does not presently require that FMU functions have to be thread-safe, i.e. they cannot be executed simultaneously as they may share some resource (variables for example) that might be corrupted if two operations try to use it at the same time. This implies that at any instant during the execution of the co-simulation, one and only one operation of the same FMU can be executed. Consequently, if the scheduling

heuristic allocates two or more operations belonging to the same FMU to different cores, a mechanism that ensures these operations are executed in strictly different time intervals must be set up.

4 Proposed solutions

This section presents a theoretical study of the achievable speed-up on a use-case, using the SynDEX² software (Sorel, 2004, 2005). Then, these theoretical results are compared with xMOD co-simulation runs, with two different implementations for guaranteeing a mutual exclusion between different operations of the same FMU.

4.1 Toolchain

A toolchain is proposed to assist the developer in parallelizing co-simulations. Using this toolchain, it is possible to assess new solutions before implementing them in xMOD thanks to the SynDEX software. SynDEX is a system level CAD software based on the Algorithm-Architecture Adequation (AAA) methodology (Sorel, 1996). It was developed to optimize the implementation of real-time distributed applications onto multicomponent architectures. The workflow is illustrated in Figure 3. When different FMUs are imported into xMOD and connected together, a file which describes inter-model connections is generated. This file and the XML files of the different FMUs of the co-simulation are passed to a converter which parses the files and produces equivalent files (.sdx) compliant to the SynDEX format. The co-simulation code is profiled in order to obtain the execution times of the different operations which are introduced in SynDEX. SynDEX offers the possibility to use the multi-core scheduling heuristic outlined in this paper, as well as other kinds of heuristics, and therefore makes it possible to study the achievable co-simulation speed-up before implementing the heuristic in xMOD.

4.2 Use-case description

In this work, experiments have been carried out on a Spark Ignition (SI) RENAULT F4RT engine co-simulation using 5 FMUs. It is a four-cylinder in line Port Fuel Injector (PFI) engine in which the engine displacement is 2000 cm^3 . The air path is composed of a turbocharger with a mono-scroll turbine controlled by a waste-gate, an intake throttle and a downstream-compressor heat exchanger (Figure 4). The engine model was developed using ModEngine library (Benjelloun-Touimi et al., 2011). ModEngine is a Modelica library that allows for the modeling of a complete engine with diesel and gasoline combustion models. The engine model was imported into xMOD using the FMI export

features of the Dymola³ tool. This use-case has over 100 operations which are scheduled by the multi-core scheduling heuristic.

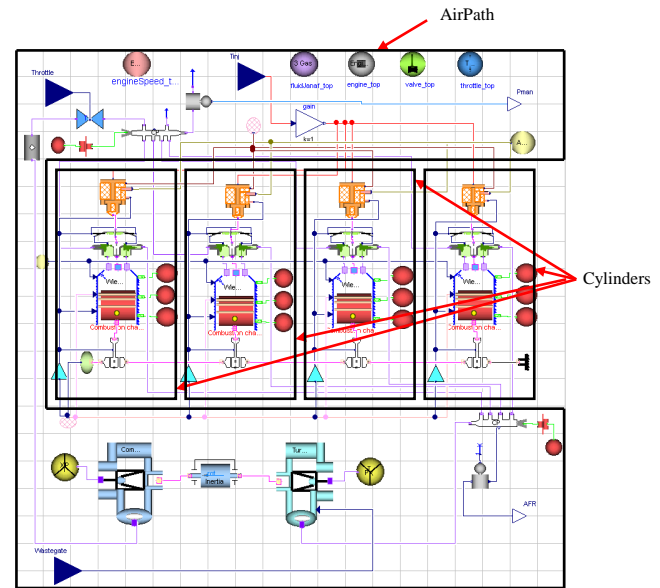


Figure 4. Spark Ignition (SI) RENAULT F4RT engine model.

4.3 Results and Discussions

Using the toolchain, a .sdx file of the use-case was generated in order to evaluate, in SynDEX, the theoretical speed-up obtained by parallelizing the model on different numbers of cores, using the multi-core scheduling heuristic of section 3.2. For each schedule the speed-up is computed by dividing the mono-core schedule makespan by the schedule makespan. Figure 5 gives the different theoretical speed-ups. The best speed-up is close to 3,6 and is reached with 6 cores. Finding the minimal number of cores which offers the maximum speed-up is interesting if a large number of simulation runs (possibly with different parameters) have to be performed: If a large number of cores is available, multiple runs could be launched in parallel with the adequate number of cores dedicated to each run. This research of the minimal number of necessary cores to reach the maximum speed-up could be scripted and automatically performed before the simulation.

In order to tackle the constraint of non thread-safe FMU functions, two mutual exclusion strategies have been implemented in xMOD and the performance obtained using each of them has been evaluated. The first one does not modify the multi-core scheduling heuristic result and uses a dedicated mutex (system object that guarantees mutual exclusion) for each FMU: Every time an FMU function call is made at runtime, the associated mutex have to be acquired before the execution of

²<http://www.syndex.org/>

³<http://www.3ds.com/products-services/catia/products/dymola>

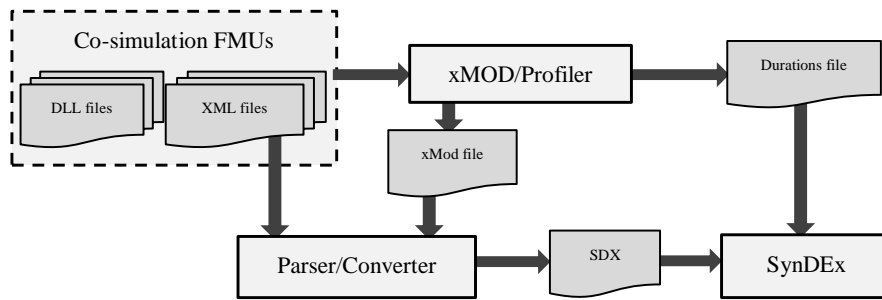


Figure 3. Proposed toolchain to assist in the development and assessment of scheduling heuristics.

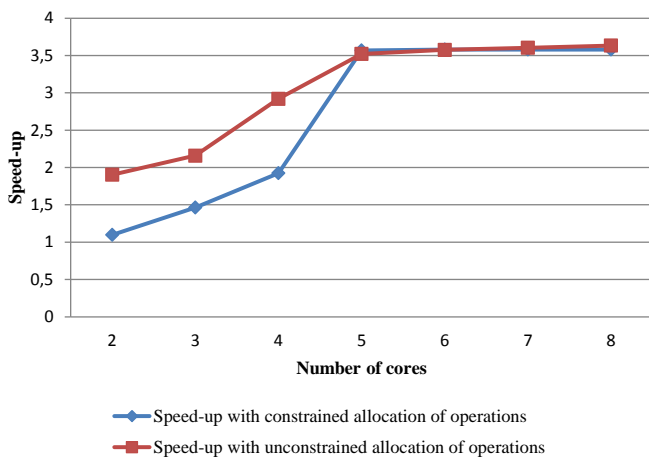


Figure 5. Theoretical speed-up.

the function code can be started. The second solution is explained in (Ben Khaled et al., 2014) and consists in modifying the multi-core scheduling heuristic to always allocate the operations of a same FMU to the same core (constrained allocation). If constrained allocation is used, the search space of the scheduling heuristic is reduced, i.e. at each step, for a given candidate operation, if there is another operation of the same FMU that has already been allocated to a specific core, the candidate operation is allocated to this same core without the need to test it on the other cores. Thanks to SynDEX, it is easily possible to theoretically estimate the impact of using the constrained allocation in the multi-core scheduling heuristic. Results are given in Figure 5. It shows that the expected speed-up in the case of constrained allocation is less than the one using unconstrained allocation, when the number of cores is less than 5, but similar when 5 cores or more are available. When using less than 5 cores, the large number of $update_{output}$ operations can be efficiently allocated only if the unconstrained allocation is used: The speed-up difference between the constrained and unconstrained allocation cases is due to this restriction on the allocation. Five is the minimal number of cores for enabling the execution of each $update_{state}$ operation on a different core. Due to the predominant execution times of the $update_{state}$ operations, their impact on the speed-up overrides the possibility of optimizing

the allocation of the other operations. This explains why the speed-up difference between the unconstrained and the constrained allocation cases becomes very small with 5 cores or more.

In order to compare the two mutual exclusion strategies, we implemented them in xMOD. Execution times measurements were performed by getting the system time stamp at the beginning of the simulation and after 30 seconds of the simulated time. As previously, we compare the speed-up by dividing the mono-core simulation execution time by the simulation execution time on a fixed number of cores. Figure 6 sums up the results, where unconstrained allocation corresponds to the use of mutex objects. It shows the impact of mutex overhead on the speed-up. Whatever the number of the available cores, the speed-up remains close to 1,3. On the contrary, the implementation in xMOD of the constrained allocation gives similar results in terms of speed-up improvement when increasing the number of cores until 5. Nevertheless, the maximum measured speed-up (2,4) remains smaller than the theoretical one (3,5). In fact, the theoretical speed-up computation considers the makespan ratio without estimating any synchronization cost between cores. The real implementation in xMOD contains synchronization objects between operations to ensure the consistency of data dependencies which certainly have an important impact on the speed-up.

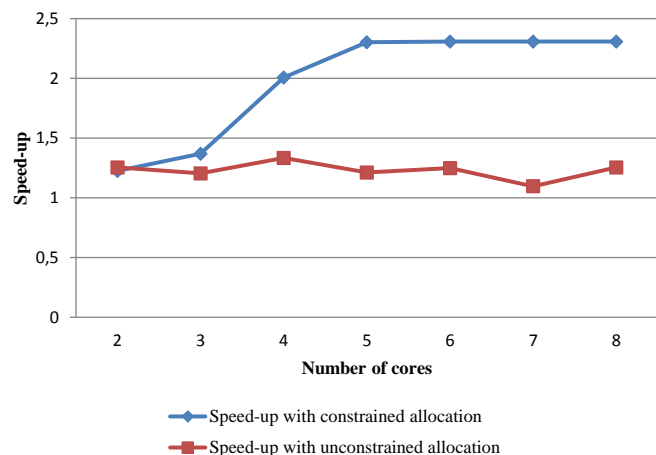


Figure 6. Measured speed-up.

5 Conclusion and Future Work

The work presented in this paper dealt with the problem of co-simulations acceleration by means of parallelization on multi-core processors. We proposed to extend our previous work by taking into account operations execution times in the multi-core scheduling heuristic. This allows the optimization of the number of the dedicated cores to the simulation, by performing architecture exploration with SynDEX. Our experiments in xMOD on an industrial use-case, gave important speed-up results (2,4). Nevertheless, it also shows the impact of the mutual exclusion constraint on the co-simulation acceleration. Providing thread-safe FMU implementation could offer important simulation acceleration opportunities. In our ongoing work, we are exploring graph transformation techniques to improve the handling of the mutual exclusion problem of FMUs. We also envision to extend these results to the multi-rate co-simulation of FMUs by developing an efficient multi-core scheduling heuristic to handle it.

References

- A. Ben Khaled, M. Ben Gaïd, D. Simon, and G. Font. Multicore simulation of powertrains using weakly synchronized model partitioning. In *IFAC Workshop on Engine and Powertrain Control Simulation and Modeling ECOSM*, pages 448–455, Reuil-Malmaison, France, 2012. doi:10.3182/20121023-3-FR-4025.00018.
- A. Ben Khaled, M. Ben Gaïd, N. Pernet, and D. Simon. Fast multi-core co-simulation of cyber-physical systems: Application to internal combustion engines. *Simulation Modelling Practice and Theory*, 47:79 – 91, 2014. ISSN 1569-190X. doi:http://dx.doi.org/10.1016/j.simpat.2014.05.002. URL <http://www.sciencedirect.com/science/article/pii/S1569190X14000665>.
- Z. Benjelloun-Touimi, M. Ben Gaïd, J. Bohbot, A. Dutoya, H. Hadj-Amor, P. Moulin, H. Saafi, and N. Pernet. From physical modeling to real-time simulation: Feedback on the use of Modelica in the engine control development toolchain. In *8th Int. Modelica Conf.*, Dresden, Germany, Mar 2011. Linköping Univ. Electronic Press.
- T. Blochwitz, T. Neidhold, M. Otter, M. Arnold, C. Bausch, M. Monteiro, C. Clauß, S. Wolf, H. Elmqvist, H. Olsson, A. Junghanns, J. Mauss, D. Neumerkel, and J.-V. Peetz. The Functional Mockup Interface for tool independent exchange of simulation models. In *8th Int. Modelica Conf.*, Dresden, Germany, Mar 2011. Linköping Univ. Electronic Press. ISBN 978-91-7393-096-3. doi:10.3384/ecp11063105.
- H. Elmqvist, S.E. Mattsson, and H. Olsson. Parallel model execution on many cores. In *10th Int. Modelica Conf.*, Lund, Sweden, 2014.
- H. Elmqvist, H. Olsson, A. Goteman, V. Roxling, D. Zimmer, and A. Pollok. Automatic gpu code generation of modelica functions. In *11th Int. Modelica Conf.*, Versailles, France, 2015.
- FMI development group. Functional mock-up interface for model exchange and co-simulation, July 2014. URL <https://www.fmi-standard.org/>.
- M. Gebremedhin, A. Hemmati Moghadam, F. Fritzson, and K. Stavaker. A data-parallel algorithmic modelica extension for efficient execution on multi-core platforms. In *9th Int. Modelica Conf.*, Munich, Germany, 2012.
- T. Grandpierre, C. Lavarenne, and Y. Sorel. Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors. In *Proceedings of 7th International Workshop on Hardware/Software Co-Design, CODES'99*, Rome, Italy, May 1999. URL <http://www-rocq.inria.fr/syndex/publications/pubs/codes99/codes99.pdf>.
- M. Sjölund, R. Braun, P. Fritzson, and P. Krus. Towards efficient distributed simulation in modelica using transmission line modeling. In Linköping Univ. Electronic Press, editor, *3rd Int. Workshop on Equation- Based Object-Oriented Languages and Tools EOOLT*, page 71–80, Oslo, Norway, 2010.
- Y. Sorel. Real-time embedded image processing applications using the algorithm architecture adequation methodology. In *Proceedings of IEEE International Conference on Image Processing, ICIP'96*, Lausanne, Switzerland, September 1996. URL <http://www-rocq.inria.fr/syndex/publications/pubs/icip96/icip96.pdf>.
- Y. Sorel. Syndex: System-level cad software for optimizing distributed real-time embedded systems. *Journal ERCIM News*, 59:68–69, October 2004. URL <http://www-rocq.inria.fr/syndex/publications/pubs/ercim04/ercim04.pdf>.
- Y. Sorel. From modeling/simulation with scilab/scicos to optimized distributed embedded real-time implementation with syndex. In *Proceedings of the International Workshop On Scilab and Open Source Software Engineering, SOSSE'05*, Wuhan, China, October 2005. URL <http://www-rocq.inria.fr/syndex/publications/pubs/sosse05/sosse05.pdf>.

Rankine Cycles, Modeling and Control

Ylva Teleman¹ Pieter Dermont² Hak Jun Kim³ Kil Sang Jang³

¹Faculty of Engineering (LTH), Lund University, Sweden, ylva.teleman@gmail.com

²Modelon AB, Sweden, pieter.dermont@modelon.com

³Hanon Systems, South Korea, {hkim18, kjang1}@hanonsystems.com

Abstract

As the need for increased energy efficiency grows, the use of new energy sources is a topic of investigation for research and industrial applications. The ability to use low temperature heat sources via a Rankine or organic Rankine cycle is one of the options. In this paper such a cycle is modelled and simulated using a Modelica based thermal management library suite as well as the simulation tool Dymola. Experimental test bench data provided by Hanon Systems allowed calibration and verification of the simulation results. Simulation results shows good agreement with experimental data. Additional dynamic simulations are performed to illustrate potential applications of the model for system optimization and control development.

Keywords: Rankine cycles, organic Rankine cycles, modeling, dynamic simulations, Modelica, Dymola, thermofluid

1 Introduction

1.1 Rankine Cycle

A Rankine cycle is a thermodynamic cycle which utilizes heat to create mechanical power, harvested by an expander that can drive a generator. The medium of the Rankine cycle, usually water, is pressurized in a pump, evaporated in a heat exchanger, subsequently passed through an expander and finally condensed in condenser. The process and essential components in the cycle can be seen in Figure 1. By choosing a medium with appropriate properties, different heat source temperatures can be used.

1.2 Purpose

This paper is a result of an academic collaboration with the Faculty of Engineering (LTH) of Lund University. The purpose of the project was to model and simulate an industrial use-case: a Rankine cycle that utilizes waste heat from a combustion engine vehicle to generate mechanical power. The project scope was defined by Mod-

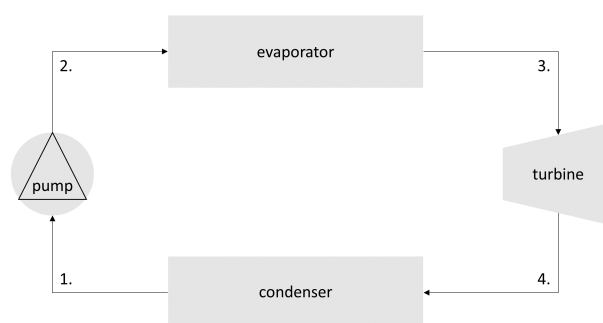


Figure 1. Essential components in a Rankine cycle.

elon, a company specialized in simulation and optimization using open standard technology. The aim of the project was to model or use existing models of components, parameterize the components, build the complete cycle, implement appropriate control and gain knowledge of the cycle's behavior.

1.3 Thermofluid Modeling

The models were implemented in Modelica, an open standard modeling language (ModelicaAssociation, 2015), using a 1-D thermofluid approach. The principles of thermofluid modeling using Modelica are laid out in (Tummescheit, 2002; Eborn, 2001).

The different components were either selected from preexisting model libraries or modeled. A specialized thermal management suite was used, based on three compatible model libraries: Vapor Cycle library (Modelon, 2015c), Heat Exchanger Library (Modelon, 2015a; Batteh et al., 2014) and Liquid Cooling Library (Modelon, 2015b; Batteh et al., 2014).

The modeling was performed in the simulation environment Dymola 2016 FD01 (DassaultSystèmes, 2015).

1.4 Method

After the physical components were either selected or modeled and calibrated, they are connected to create the complete cycle. Simulations with different control

strategies and conditions were tested. In order to verify results, data received from Hanon Systems was used. Hanon Systems is formerly known as Halla Visteon Climate Control and specializes in thermal management solutions for automotive applications. The components and cycle were parameterized according to 11 experimental data sets.

2 System Presentation

2.1 Refrigerant

A refrigerant called R134a, also known as 1,1,1,2-tetrafluoroethane, is the working fluid of the cycle. While utilizing low grade waste heat is most commonly done using an organic refrigerant, the waste heat temperature is sufficiently high in Hanon System's test setup to use a conventional refrigerant.

Results are repeatedly represented in the specific enthalpy - pressure diagram of the working fluid. The diagram for R134a can be seen in Figure 2. Within the thumb-like shape, the refrigerant is a two-phase fluid, a mixture of both liquid and gas. To the left and right of the dome, only liquid respectively vapor exists. The isotherms transverse the two-phase dome horizontally.

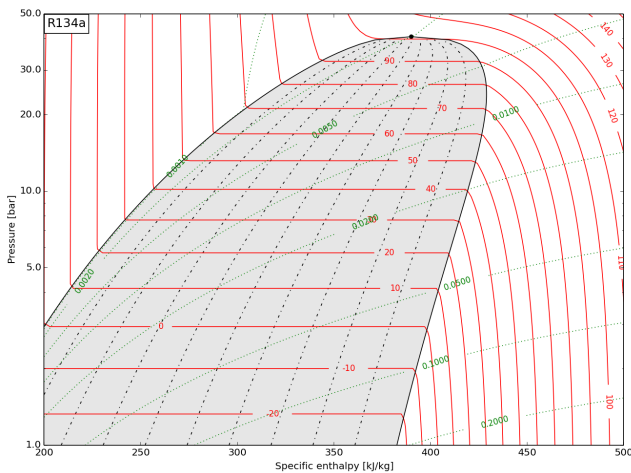


Figure 2. Pressure/enthalpy diagram for R134a.

2.2 Components

All but one component used for the Rankine cycle were readily contained in one of Modelon's thermal management libraries suite. Hanon Systems' test bench utilizes a positive displacement machine, more specifically a diaphragm pump that was implemented as a custom component.

A *diaphragm pump* operates conceptually similar to a human diaphragm does - a chamber and a membrane that moves outward sucking working fluid into the space or move inwards in order to push out the working fluid. The pump has a 4.46 kW power output at 1750 rpm. Its losses

can be described through three distinct efficiencies; volumetric, isentropic and mechanical. The volumetric efficiency describes ratio of the volume in the pump that is displaced to the geometric volume. The isentropic efficiency describes how much energy is lost during the process of pressuring, and the mechanical efficiency describes losses in shaft and other components of the pump. The efficiencies of the pump were estimated with the experimental data, partly relying on assumptions since data for the fluid at the outlet of the pump was missing.

The *expander* used in Hanon Systems' cycle is a positive displacement machine: a scroll turbine. It has two intertwined spirals, one stator and one rotor. The hot high pressured gas enters in the middle, pushing the rotor around as it makes its journey between the spirals until it at last exits the positive displacement machine. Similar to the pump, the expander is subject to losses and they are described using equivalent efficiencies; volumetric, isentropic and mechanical. Again, these efficiencies were calculated from the data provided by Hanon Systems.

The cycle includes a *plate heat exchanger* which acts as evaporator, using glycol on the secondary side. Not all geometry parameter data was available, and the missing parameters were estimated or set to typical values. A detailed geometry-based plate heat exchanger model exists in the model libraries, with a higher number of states due to its flow configuration. A less complex counter-flow heat exchanger model with equivalent parameterization was used assuring fast solver convergence but less accurate behavior representation. In Table 1 some geometry data provided by Hanon Systems for the plate heat exchanger can be seen.

Item	Value
Type	MCV Plate
# plates	36 rows
Fin	N.A.
Path	6-12-18
Effective size mm	93.1*170.5*70.4

Table 1. Plate heat exchanger data

The condenser of the cycle is a *flat tube heat exchanger* using ambient air to cool down the working fluid. A detailed geometry-based heat exchanger model was used, parameterized with both available geometry data and typical geometry data. In Table 2 some geometry data provided by Hanon Systems for the flat tube heat exchanger can be seen.

A *tank* is integrated in the cycle between condenser and pump. The purpose of the tank is twofold: (1) ensure only liquid entered the pump as gas would damage it and (2) balance the amount of working fluid in the cycle. A static head of 1 m was introduced between condenser and pump to confirm that the dynamic pressure at the inlet of the pump was sufficiently high to avoid the creation of bubbles in the fluid. The tank has a 8 L volume.

Item	Value
Type	SC 20t
# tubes	54 rows
Fin	80 fpm
Path	44-10
Effective size mm	570*383

Table 2. Flat tube heat exchanger data

2.3 Control

Based on the cycle configuration different control strategies are relevant. The following strategies were considered:

- Adequate super heating to let only gas into the turbine
- Adequate sub cooling or other measure to avoid gas entering the pump
- Preferred power or torque from the turbine
- Optimal evaporation temperature
- Low condenser pressure
- Appropriate amount of refrigerant in the cycle

Except for choosing appropriate components that will match the requirements, it is possible to change the following variables (Quoilin et al., 2011):

- The speed of the turbine
- The speed of the pump
- Add tanks after condenser/evaporator
- Drain or charge the cycle with WF
- Change the temperature of the heat source for the evaporator
- Change the temperature of the cooling air in the condenser

The temperature of the heat source in the evaporator as well as the cooling air in the condenser are typically not entities that can be controlled in automotive applications as the heat source typically is the heat from the engine and the temperature of the air is dependent on ambient conditions. It is however of interest to gain knowledge about the behavior of the cycle under different conditions.

The speed of the pump has a direct consequence on the amount of superheat of the working fluid leaving the evaporator. Decreasing the speed of the pump, the

amount of superheating is increased. Effective superheat control is important to ensure no two-phase fluid enters and potentially damages the expander. The speed of the expander was varied in order to achieve desired power or torque as they are correlated according to Equation 1.

$$Torque(Nm) = \frac{Power(W)}{Speed(rad/s)} \quad (1)$$

For both superheat and torque control PI-controllers were used.

In this project the heat source and temperature of cooling air were kept constant. The overall efficiency is calculated according to Equation 2.

$$\eta_{overall} = \frac{W_{turbine} - W_{pump}}{Q_{evaporator}} \quad (2)$$

2.4 Initialization

Initialization is important to ensure a fast solver convergence. In this cycle, the most complex components are the heat exchangers. Since the heat exchangers contain the majority of continuous time states, their initialization is key. Additionally, it is necessary to set the PI-controller parameters correctly in order to achieve robust control.

The amount of working fluid in the cycle can be changed either by:

- Changing the initialization of the cycle, i.e. the continuous times states of the thermofluid model, which results in a refrigerant mass.
- Directly setting the tank level.
- Adding a charge component that can either charge or drain the cycle to a desired amount of working fluid during simulation.

2.5 Data

Hanon Systems provided 11 data sets that varied in the amount of working fluid and super heating. For every data set the speed, power and torque were measured for pump and turbine, and inlet and outlet pressure and temperature were measured for all the components (with exception of outlet of pump).

On the test bench test conditions such as speed of the pump, ambient temperature, air velocity, mass flow and boiler coolant temperature were controlled. The speed of the expander was controlled such that the torque remained constant at 9 Nm, and a tank was placed between the condenser and the pump. Controls for dynamic operation of the cycle weren't implemented at the time of the measurements. However, the maximum power of the expander was determined by changing expander brake torque for different pump speeds and air temperatures in the condenser.

2.6 Test Scenarios

The following control strategies or conditions were tested for the cycle:

- The different data sets were simulated.
- The cycle was tested with different torques on the turbine to match the maximum power point tracking diagram Hanon Systems had implemented. Hanon Systems concluded from experimental data that maximum power is achieved at 9 Nm brake torque.
- Superheat control was tested.
- The cycle was tested with varying amounts of refrigerant.

3 Creating the Model

3.1 Expander

The isentropic efficiency for the expander was calculated using the data. For all the data sets it varied between 43-55% depending on pressure ratio and speed. No correlation could however be demonstrated, likely due to measurement errors and thus a mean value was used. When calculating the mechanical efficiency, only 3 of the 11 data sets give values under 100%. For 9 of the data sets the mass flow was too low, and it was assumed that an error in either the data or the measuring of data had occurred. Consequently a mean value of the mechanical efficiency from the three good data sets was used and resulted in a base value of 87%.

3.2 Pump

Unlike the expander, the pump's efficiencies were mapped in a grid depending on speed and pressure ratio. Since it was known that the mass flow was inconsistent for 9 of the data sets, only the three data sets with consistent data were used to calculate the efficiencies. The mechanical efficiency was set to 73%, and the isentropic ranged between 58% and 70% depending on speed and pressure ratio over the pump.

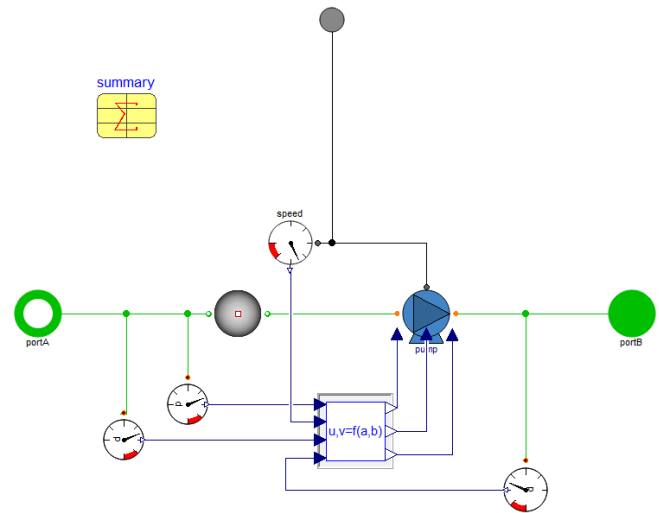


Figure 3. The model of the pump as seen in Dymola.

The top level class of the pump in the Dymola diagram layer is illustrated in Figure 3. The model is based on a generic pump model from the Liquid Cooling Library, in which a function prescribes the pump's behavior. Through an interface, this function can be exchanged. The speed of the shaft, density at the inlet and pressure at the inlet, and at the outlet are inputs to the interface. Outputs of the interface are isentropic and mechanical efficiencies, as well as mass flow. To create a diaphragm pump, appropriate behavior-prescribing tables dependent on speed and pressure ratio were inserted in the interface, and the efficiencies are then extrapolated from the tables. The interface additionally calculates the mass flow based on the density at the inlet, speed, maximum displacement volume, and volumetric efficiency.

3.3 Heat Exchangers

In order to calibrate the heat exchangers, virtual test benches were set up. In these virtual test benches, the heat exchanger was connected to mass flow and pressure source components (Figure 4). After setting geometry parameters, applying correct boundary conditions, and selecting appropriate heat transfer and pressure drop correlations, the calibration factors were used to calibrate the heat exchanger model. The calibration factors tune the pressure drop as well as the heat transfer coefficients.

3.4 Complete Cycle

The complete assembled cycle in the Dymola diagram layer is illustrated in Figure 5. The cycle contains a simple evaporator, a tank between the condenser and pump, and includes all necessary control for dynamic simulation.

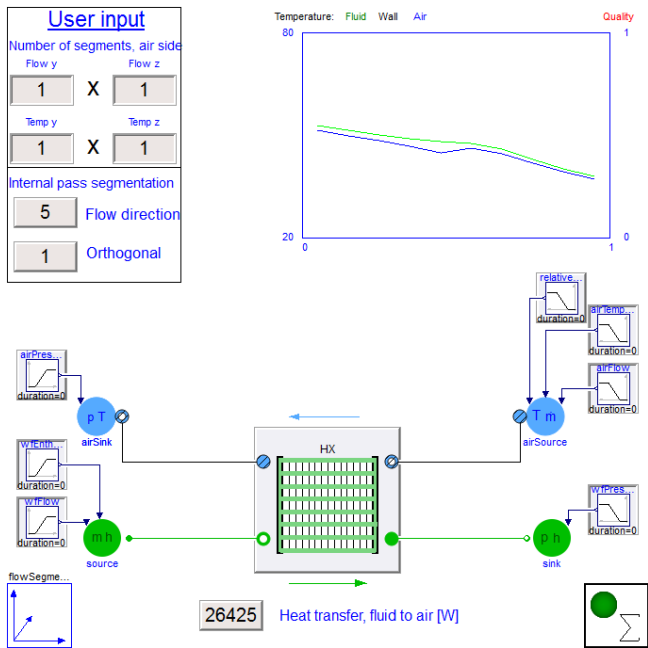


Figure 4. Virtual testbench for the condenser. It was based on a template with a default heat exchanger and boundary condition parameterization (HeatExchanger.HeatExchangers.FlatTube.Experiments.TestBenches.AirTwoPhaseHomogeneous).

Data set	9	Simulation
Power W	834	836
Overall efficiency %	2.2	1
Mass flow kg/s	0.12	0.12
SH °C	2.5	2.5
Speed of turbine rpm	885	886
Speed of pump rpm	350	343
Torque of turbine Nm	9	9
Turbine pressure ratio	1.87	2.03
Pressure ratio error %	-	8.6
Isentropic efficiency turbine	-	61

Table 3. Simulation results for data set 9.

Data set	10	Simulation
Power W	820	827
Overall efficiency %	1.7	1.8
Mass flow kg/s	0.138	0.138
SH °C	0.5	0.5
Speed of turbine rpm	870	879
Speed of pump rpm	400	406
Torque of turbine Nm	9	9
Turbine pressure ratio	1.89	1.84
Pressure ratio error %	-	2.6
Isentropic efficiency turbine	-	60

Table 4. Simulation results for data set 10.

4 Results

The result section is composed of two parts. The first part validates the model by comparison with the experimental data and demonstrates that the model can mimic its physical counterpart under static conditions. A comparison of a relevant cycle metric is performed: a power tracking diagram is plotted. Secondly, the dynamic capabilities of the model are shown through a set of simulation experiments. Simulations took between 2 and 10 min for a cycle with the simple evaporator, and 0.5-1.5 h with the complex one.

4.1 Simulation of Different Data Sets

All 11 data sets, each with different boundary conditions, matched the simulation results well. The sets with coherent data resulted in the best match as expected. In Figure 6 two specific enthalpy - pressure diagrams with thermodynamic cycle are depicted; on the left is the cycle constructed with experimental data of data set 10 and on the right the cycle as obtained from the simulation using data set 10 boundary conditions.

In Table 3 and 4 simulation results as well as experimental data are compared. The simulation of data set 10 is more accurate than the one of set 9; data set 10 had coherent data and set 9 did not.

4.2 Maximum Power Tracking Diagram

The graph in Figure 7 illustrates the behavior of the power harvested by the expander as a function of the torque applied to the expander. Experimental and simulation data match well. The largest difference is approximately 3%.

4.3 Superheat Control

Dynamic superheat control setpoint control response is plotted in Figure 8. The super heating was measured by the sensor component in Figure 5. The output was then sent to the PI-controller which compares the measured value with the desired value and subsequently adjusted the pump speed.

4.4 Refrigerant charge

The refrigerant source components allows charging or draining the cycle during the simulation. It was demonstrated that overcharging the cycle results in any increase of pressure in all components with a ultimately a loss of pressure difference between evaporator and condenser. Similar observations could be made for a starved cycle where the pressure in components decreases as illustrated in Figure 9. Expander power loss was significant, ranging from approximately 850 W with sufficient refrigerant charge, to 137 W for a starved cycle.

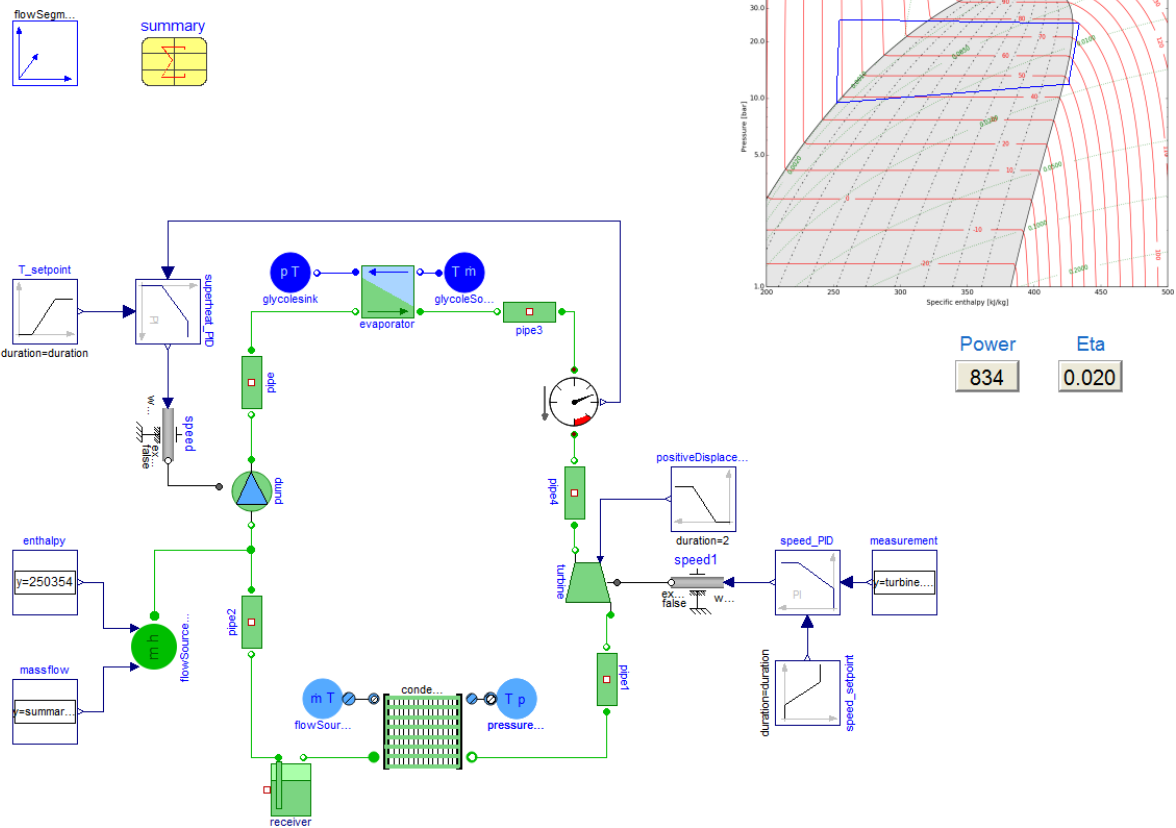


Figure 5. The whole Rankine cycle as seen in Dymola.

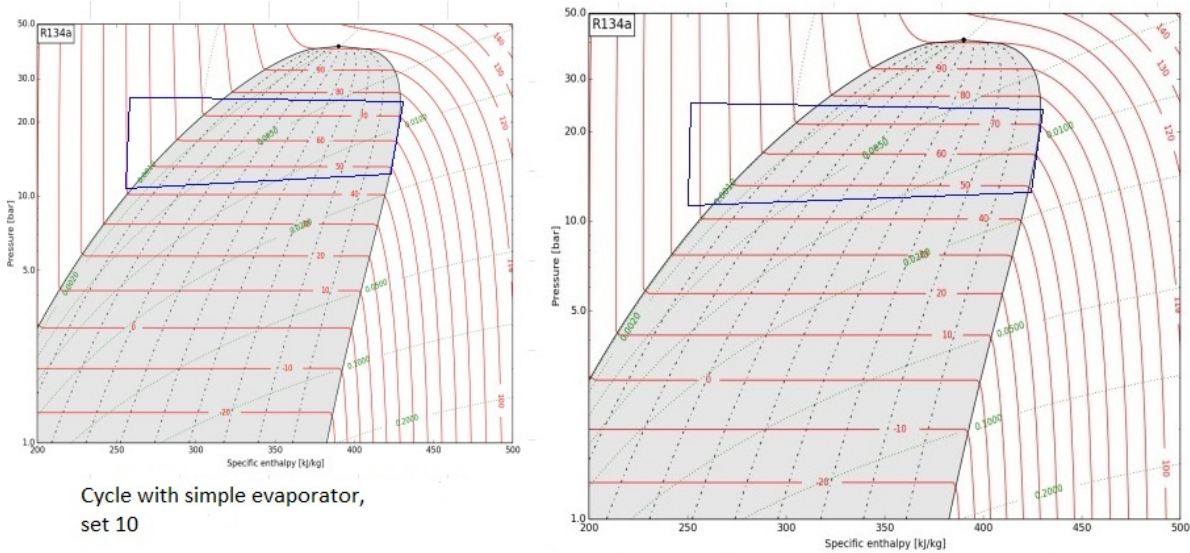


Figure 6. Simulation result for data set 10, on a pressure/enthalpy diagram to the left, data to the right.

5 Discussion and Conclusion

A Rankine cycle was modeled and parameterized according to data from Hanon Systems, and control strategies were implemented to gain insight into the behavior of the cycle and for controls virtual prototyping. Some

variables had to be estimated since data was missing or inaccurate.

Steady state simulation results matched experimental data well. Additional geometrical information on individual components as well as the piping between components would allow to create a more trustworthy model,

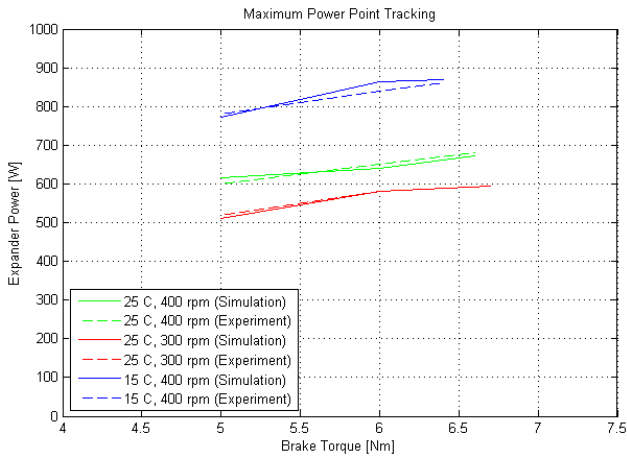


Figure 7. Maximum power point diagram, comparison of experimental and simulation data

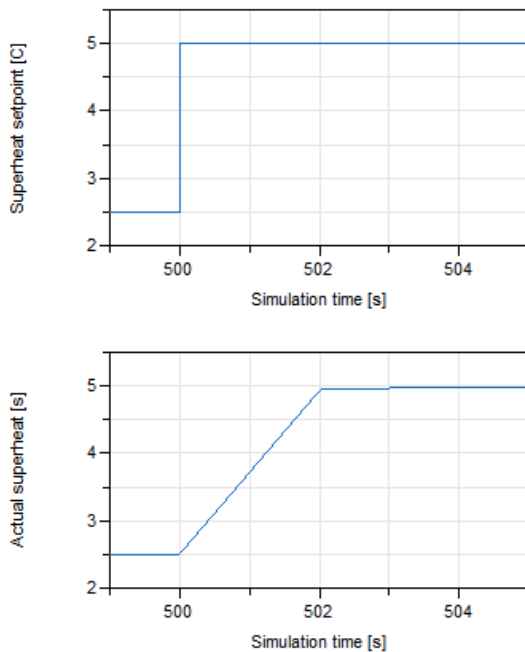


Figure 8. Superheat control. Setpoint and system response.

with an accurate total volume and all pressure drops included. The power point tracking diagram measured out by Hanon Systems is created with the model with only small deviations of the expander power output.

The control strategies that were implemented and the initialization worked effectively. At the point of completion of the model, no control was specified for the system test bench. Dynamic simulations were carried out based on typical control mechanisms for Rankine cycles.

The cycle created in this project was a generic and simple one intended for automotive applications. In order to gain more knowledge about industrial Rankine cycles different setups should be modeled and simulated. The overall efficiency of the cycle varied between approximately 1.7 and 2.2 %. Hanon Systems' primary in-

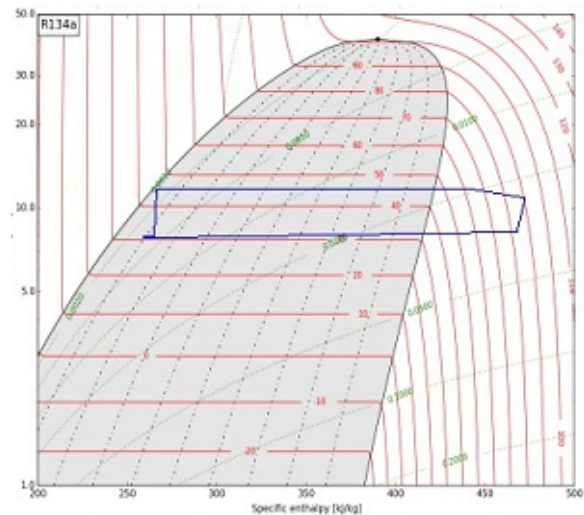


Figure 9. Thermodynamic cycle after drain to a specific charge of $300\text{kg}/\text{m}^3$ - turbine power output is at 137 W.

tention with the test bench was to focused on the development and testing of an expander. Higher efficiencies can be achieved with state-of-the-art Rankine cycles.

References

- John Batteh, Jesse Gohl, and Sureshkumar Chandrasekar. Integrated vehicle thermal management in modelica: Overview and applications. In *Proceedings of the 10th International Modelica Conference*, 2014.
- DassaultSystèmes. Dymola 2016 fd01, 2015. URL <http://www.3ds.com/products-services/catia/products/dymola>.
- Jonas Eborn. *On model libraries for thermo-hydraulic applications*. PhD thesis, Lund University, 2001.
- ModelicaAssociation, 2015. URL www.modelica.org.
- Modelon. Heat exchanger library, 2015a. URL www.modelon.com/products/modelica-libraries/heat-exchanger-library/.
- Modelon. Liquid cooling library, 2015b. URL www.modelon.com/products/modelica-libraries/liquid-cooling-library/.
- Modelon. Vapor cycle library, 2015c. URL www.modelon.com/products/modelica-libraries/vapor-cycle-library/.
- Sylvain Quoilin, Richard Aumann, Andreas Grill, Andreas Schuster, Vincent Lemort, and Hartmut Spliethoff. Dynamic modeling and optimal control strategy of waste heat recovery organic rankine cycles. *Applied Energy*, 88(6): 2183–2190, 2011.

Hubertus Tummescheit. *Design and implementation of object-oriented model libraries using modelica*. PhD thesis, Lund University, 2002.

Thermal Deformation Analysis Using Modelica

Eunkeyeong Kim¹ Tatsuou Yashiki¹ Fumiyuki Suzuki² Yukinori Katagiri¹ Takuya Yoshida¹

¹Hitachi, Ltd., Research & Development Group, Japan,
{eunkeyeong.kim.mn, tatsuro.yashiki.zn, yukinori.katagiri.gf,
takuya.yoshida.ru}@hitachi.com

²Mitsubishi Hitachi Power Systems, Ltd., Japan,
fumiyuki_suzuki@mhps.com

Abstract

This paper presents a thermal deformation analysis method fully utilizing the non-causality of the Modelica language as a means of solving large scale simultaneous equations including equilibrium equations related to stresses, stress-strain relations and strain-displacement relations. As an illustrative example, a model for thermal deformation analysis of a cylindrical object in the two-dimensional circular polar coordinate system is described. Simulations are performed for a cylindrically shaped object under a uniform temperature distribution and a radial temperature distribution. The results of the simulations show that the differences in displacements between the proposed model and a model based on finite element (FE) methods are less than 9% while the number of elements that compose the proposed model is about 1/8 compared to that of the FE model.

Keywords: thermal deformation, stress-strain relation, strain-displacement relation, equilibrium equations, displacement gradient, physical modeling, finite volume method, non-causality, Modelica, Dymola

1 Introduction

Thermal deformation is conventionally analyzed by FE methods. Conventional FE methods supply highly precise outputs; however they require a lot of work and time. In order to conveniently analyze thermal deformation without using FE methods, analytical solutions have been extensively reported in the literature (see for example, Gere, 2006). These analytical solutions can only be used for simple geometries such as blocks and round bars and simple temperature distributions such as steady-state conditions. This is because it is difficult to analytically solve large scale simultaneous equations including equilibrium equations related to stresses, stress-strain relations and strain-displacement relations.

In this paper, a method to loosen these restrictions of simple geometries and simple temperature distributions is presented. The non-causality of the Modelica language (Elmqvist and Mattsson, 1997a, 1997b; Elmqvist et al, 1998a, 1998b; Fritzson and Engelson,

1998; Fritzson, 2003, 2011) is fully utilized as a means of formulating and solving the large scale simultaneous equations of thermal deformation analysis. The main feature of the proposed method is that the positions of displacements and forces are defined based on a finite volume method (Ferziger and Peric, 2002; Voller, 2009) to effectively describe these complex physical phenomena in non-causal manner and also to implement them as a network of Modelica component models.

2 Thermal deformation model

Here a model for the thermal deformation analysis of a cylindrical object in the two-dimensional circular polar coordinate system is described.

A cylindrical object represented by one quarter as illustrated in Figure 1 is discretized the 9 control volumes (referred to as elements in this paper) by dividing the object in the r and θ directions into 3×3 elements. A graphical image of the Modelica component models interacting with each other for analyzing thermal deformation of the cylindrical object is shown in Figure 2. There are 3 types of component models; a model for calculating the force balances inside the element (element model), a model for calculating the interactions of forces and deformations between elements (linkage model), and a model for calculating the boundary conditions (BC model). There are also connectors designed to connect these component models so that structural analysis of the entire object is performed (SA connector).

In the following, the SA connector (Sec. 2.1), the element model (Sec. 2.2), the linkage model (Sec. 2.3) and the BC model (Sec. 2.4) are described, and then the method to calculate the displacements at the vertices of the elements (Sec. 2.5) is given.

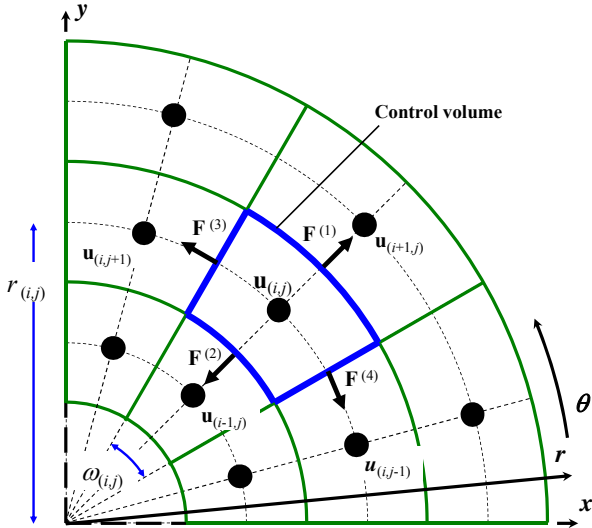


Figure 1. A cylindrical object and its discretization.

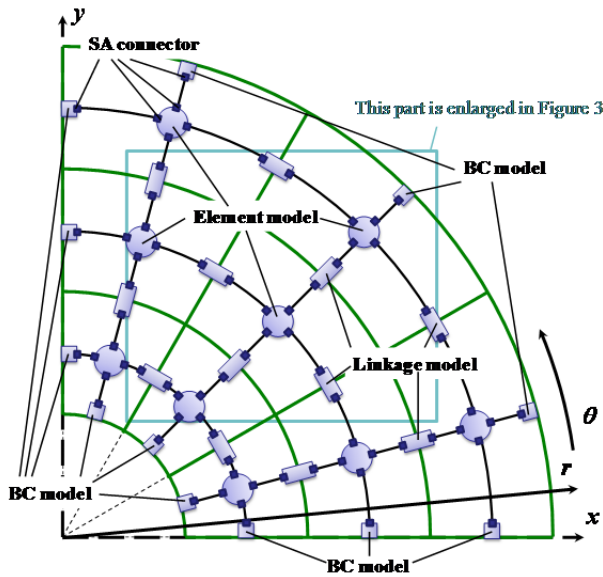


Figure 2. A graphical image of the Modelica component models for analyzing thermal deformations of the cylindrical object in Figure 1.

Table 1. Declarations of variables in the SA connector

Type	Name	Description
Length	ur	displacement in the r direction
Length	ut	displacement in the θ direction
Real	durd r	displacement gradient $\partial u_r / \partial r$
Real	dut r	displacement gradient $\partial u_\theta / \partial r$
Real	durd θ	displacement gradient $\partial u_r / \partial \theta$
Real	dut θ	displacement gradient $\partial u_\theta / \partial \theta$
flow Force	Fr	force acting in the r direction
flow Force	Ft	force acting in the θ direction

2.1 SA connector

The SA connector passes the information of forces, displacements, and displacement gradients between the component models. Each SA connector has 6 variables as shown in Table 1: 2 variables representing displacements at the center position of the element; 4 variables representing displacement gradients; and 2 variables representing forces acting on a boundary surface of the element which are the flow variables.

2.2 Element model

In this model, (1) the balance of the forces acting on the boundary surfaces of an element is calculated, and also (2) the displacement gradients at the center position of the element are calculated and passed to the SA connectors, then (3) the information of the displacements at the center position is passed to the SA connectors.

Each element model has 6 dynamic variables as shown in Table 2: 2 variables representing displacements at the center position of the element; and 4 variables representing displacements gradients at the center position of the element. The information of displacements and displacement gradients are passed to the linkage models located on the boundary surfaces of the element via the 4 SA connectors Crb, Cra, Ctb and Cta for interacting with neighbor elements (The connections between the element model and the linkage models via the SA connectors are shown in Figure 3).

(1) The balance of the forces acting on the boundary surfaces of an element is calculated by

$$\mathbf{F}^{(1)} + \mathbf{F}^{(2)} + \mathbf{F}^{(3)} + \mathbf{F}^{(4)} = \mathbf{0} \quad (1)$$

where $\mathbf{F}^{(1)}$, $\mathbf{F}^{(2)}$, $\mathbf{F}^{(3)}$ and $\mathbf{F}^{(4)}$ are representing the forces acting on boundary surfaces 1, 2, 3 and 4 (Figure 1). This equation is described by the following Modelica code.

$$\begin{aligned} \text{Crb.Fr} + \text{Cra.Fr} + \text{Ctb.Fr} + \text{Cta.Fr} &= 0; \\ \text{Crb.Ft} + \text{Cra.Ft} + \text{Ctb.Ft} + \text{Cta.Ft} &= 0; \end{aligned}$$

(2) The displacement gradients at the center position of the element are calculated by

$$\left(\frac{\partial u_\xi}{\partial r} \right) \approx \frac{1}{2} \left[\left(\frac{\partial u_\xi}{\partial r} \right)^{(1)} + \left(\frac{\partial u_\xi}{\partial r} \right)^{(2)} \right] \quad (2)$$

$$\left(\frac{\partial u_\xi}{\partial \theta} \right) \approx \frac{1}{2} \left[\left(\frac{\partial u_\xi}{\partial \theta} \right)^{(3)} + \left(\frac{\partial u_\xi}{\partial \theta} \right)^{(4)} \right] \quad (3)$$

where u_ξ denotes the displacements u_r and u_θ . The displacement gradients in the r direction ($\partial u_\xi / \partial r$) are calculated as the averages of $(\partial u_\xi / \partial r)$ on boundary surfaces 1 and 2. The displacement gradients in the θ direction ($\partial u_\xi / \partial \theta$) are calculated as the averages of

$(\partial u_\theta / \partial \theta)$ on boundary surfaces 3 and 4. These equations are described by the following Modelica code.

```
durdr = (Crb.durdr + Cra.durdr) / 2;
dutdr = (Crb.dutdr + Cra.dutdr) / 2;
durdt = (Ctb.durdt + Cta.durdt) / 2;
dutdt = (Ctb.dutdt + Cta.dutdt) / 2;
```

The displacement gradients with respect to r or θ axis are passed to the linkage models located on the boundary surfaces along the other axis via the SA connectors. The Modelica code to form these information flows is as follows.

```
Ctb.durdr = durdr;
Ctb.dutdr = dutdr;
Cta.durdr = durdr;
Cta.dutdr = dutdr;
Crb.durdt = durdt;
Crb.dutdt = dutdt;
Cra.durdt = durdt;
Cra.dutdt = dutdt;
```

(3) The displacements at the center position of the element are determined by the interactions of forces and displacements between the neighbor elements calculated by the linkage models mentioned in section 2.3. The Modelica code to share the information of the displacements among the surrounding linkage models is as follows.

```
Crb.ur = ur;
Cra.ur = ur;
Ctb.ur = ur;
Cta.ur = ur;
Crb.ut = ut;
Cra.ut = ut;
Ctb.ut = ut;
Cta.ut = ut;
```

Table 2. Declarations of variables in the element model

Type	Name	Description
Length	ur	displacement in the r direction
Length	ut	displacement in the θ direction
Real	durdr	displacement gradient $\partial u_r / \partial r$
Real	dutdr	displacement gradient $\partial u_\theta / \partial r$
Real	durdt	displacement gradient $\partial u_r / \partial \theta$
Real	dutdt	displacement gradient $\partial u_\theta / \partial \theta$

Table 3. Declarations of variables in the linkage model and the BC model

Type	Name	Description
Stress	tau rr	stress in the r direction
Stress	tau rt	stress in the θ direction
Real	epsilon rr	strain in the r direction
Real	epsilon rt	shear strain
Real	epsilon tt	strain in the θ direction
Real	durdr	displacement gradient $\partial u_r / \partial r$
Real	dutdr	displacement gradient $\partial u_\theta / \partial r$
Real	durdt	displacement gradient $\partial u_r / \partial \theta$
Real	dutdt	displacement gradient $\partial u_\theta / \partial \theta$

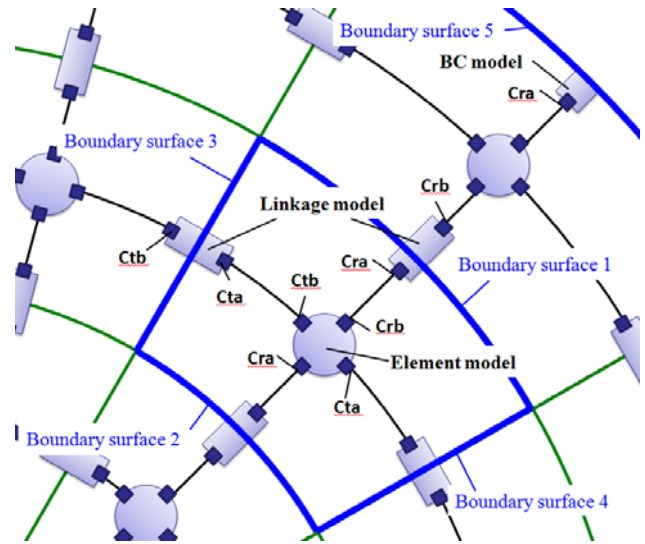


Figure 3. Enlarged illustration of Figure 2.

2.3 Linkage model

In this model, (1) the forces acting on the boundary surfaces, (2) the stress-strain relations, (3) the strain-displacement relations, and (4) the displacement gradients on the boundary surfaces between the elements are calculated, and then the resulting displacement gradients are passed to the SA connectors.

There are several types of linkage model according to the number of coordinates: in this paper, the first is for dealing with the relations between the adjacent elements in the r direction and the second is for dealing with those in the θ direction. Here the model concerning boundary surface 1 located in the r direction (Figure 3) is described as an example.

The linkage model has 9 dynamic variables as shown in Table 3: 2 variables representing stresses acting on boundary surface 1; 3 variables representing strains on boundary surface 1; and 4 variables representing displacement gradients on boundary surface 1. This information on displacement gradients is passed to the element models adjacent to the linkage model via the 2 SA connectors Crb and Cra.

(1) The force acting on boundary surface 1 is calculated by

$$\mathbf{F}^{(1)} = \begin{bmatrix} \tau_{rr}^{(1)} \\ \tau_{r\theta}^{(1)} \end{bmatrix} r^{(1)} \omega L_z \quad (4)$$

This equation is described by the following Modelica code.

```
Crb.Fr = tau_rr * r * omega * Lz;
Crb.Ft = tau_rt * r * omega * Lz;
Cra.Fr = -Crb.Fr;
Cra.Ft = -Crb.Ft;
```

where r is the radius at this boundary surface, ω is the angle of the element, and Lz is the axial length of the element.

(2) The stress-strain relations on boundary surface 1 are calculated by the following equations.

$$\tau_{rr}^{(1)} = \frac{E}{1-\nu^2} \left[\varepsilon_{rr}^{(1)} + \nu \varepsilon_{\theta\theta}^{(1)} - \alpha(1+\nu)(T^{(1)} - T_0) \right] \quad (5)$$

$$\tau_{rr}^{(1)} = \frac{E}{(1+\nu)(1-2\nu)} \left[(1-\nu)\varepsilon_{rr}^{(1)} + \nu\varepsilon_{\theta\theta}^{(1)} - \alpha(1+\nu)(T^{(1)} + T_0) \right] \quad (6)$$

$$\tau_{r\theta}^{(1)} = \frac{E}{2(1+\nu)} \varepsilon_{r\theta}^{(1)} \quad (7)$$

Equation (5) is used for the plane stress problem in which the whole vertical stress on the r - θ plane is zero, and equation (6) is used for the plane strain problem in which the whole vertical strain on the r - θ plane is zero (Voller, 2009). For example, the Modelica code to deal with the plane stress problem is as follows.

```
tau_rr=E/(1-nu^2)*(epsilon_rr+
nu*epsilon_tt-(1+nu)*alpha*(Temp-Temp0));
tau_rt=E/2/(1+nu)*epsilon_rt;
```

where E is Young's modulus, ν is the Poisson ratio, α is the coefficient of thermal expansion, $Temp_0$ is the initial temperature, and $Temp$ is the temperature.

(3) The strain-displacement relations on the boundary surface 1 are calculated by

$$\varepsilon_{rr}^{(1)} = \left(\frac{\partial u_r}{\partial r} \right)^{(1)} \quad (8)$$

$$\varepsilon_{\theta\theta}^{(1)} = \frac{u_r^{(1)}}{r^{(1)}} + \frac{1}{r^{(1)}} \left(\frac{\partial u_\theta}{\partial \theta} \right)^{(1)} \quad (9)$$

$$\varepsilon_{r\theta}^{(1)} = \frac{1}{r^{(1)}} \left(\frac{\partial u_r}{\partial \theta} \right)^{(1)} + \left(\frac{\partial u_\theta}{\partial r} \right)^{(1)} - \frac{u_\theta^{(1)}}{r^{(1)}} \quad (10)$$

where $u_r^{(1)}$ and $u_\theta^{(1)}$ denote the displacements on boundary surface 1 and they are approximated by the averages of the displacements of the adjacent elements.

$$u_\xi^{(1)} \approx \frac{u_\xi^{(i+1,j)} + u_\xi^{(i,j)}}{2} \quad (11)$$

Here u_ξ means the displacements u_r and u_θ . These relations are described by the following Modelica code.

```
epsilon_rr=durdr;
epsilon_tt=(Crb.ur+Cra.ur)/2/r + dutdt/r;
epsilon_rt=durdt/r+dutdr-
(Crb.ut+Cra.ut)/2/r;
```

(4) The displacement gradients perpendicular to boundary surface 1 are approximated from the displacements of the adjacent elements.

$$\left(\frac{\partial u_\xi}{\partial r} \right)^{(1)} \approx \frac{u_\xi^{(i+1,j)} - u_\xi^{(i,j)}}{r_{(i+1,j)} - r_{(i,j)}} \quad (12)$$

This equation is described by the following Modelica code.

```
durdr=(Crb.ur-Cra.ur)/(r_b-r_a);
dutdr=(Crb.ut-Cra.ut)/(r_b-r_a);
```

where r_b and r_a are the radii at the center positions of the adjacent elements.

The displacement gradients are passed to the element models adjacent to boundary surface 1 via the SA connectors. The Modelica code to form these information flows is as follows.

```
Crb.durdr=durdr;
Cra.durdr=durdr;
Crb.dutdr=dutdr;
Cra.dutdr=dutdr;
```

The displacement gradients tangential to boundary surface 1 are interpolated using the values at the center positions of the adjacent elements.

$$\left(\frac{\partial u_\xi}{\partial \theta} \right)^{(1)} \approx \frac{1}{2} \left[\left(\frac{\partial u_\xi}{\partial \theta} \right)_{(i+1,j)} + \left(\frac{\partial u_\xi}{\partial \theta} \right)_{(i,j)} \right] \quad (13)$$

This equation is described by the following Modelica code.

```
durdt=(Crb.durdt+Cra.durdt)/2;
dutdt=(Crb.dutdt+Cra.dutdt)/2;
```

2.4 BC model

This model is located on the boundaries of the entire object (Figure 2). The BC model is similar to the linkage model; however it is different in that it deals with either a restraint or a loading condition that has to set the boundaries of the entire object.

Here the model concerning boundary surface 5 located in the r direction (Figure 3) is described as an example.

The BC model has 9 dynamic variables as shown in Table 3, and the information of displacement gradients is passed to the element model adjacent to boundary surface 5 via the SA connector Cra .

(1) The restraint condition is represented using the displacement gradients.

$$\left(\frac{\partial u_\xi}{\partial r} \right)^{(5)} \approx \frac{u_\xi^{BC} - u_\xi^{(i+1,j)}}{r^{BC} - r_{(i+1,j)}} \quad (14)$$

$$\left(\frac{\partial u_\xi}{\partial \theta} \right)^{(5)} \approx \left(\frac{\partial u_\xi}{\partial \theta} \right)_{(i+1,j)} \quad (15)$$

Here u_ξ denotes the displacements u_r and u_θ , u_ξ^{BC} is the displacements of boundary surface 5, and r^{BC} is the radius on boundary surface 5. These equations are described by the following Modelica code.

```
durdr=(ur_bc-Cra.ur)/(r_bc-r);
dutdr=(ut_bc-Cra.ut)/(r_bc-r);
durdt=Cra.durdt;
dutdt=Cra.dutdt;
```

(2) The loading condition is represented by

$$\mathbf{F}^{BC} \approx \begin{bmatrix} \tau_{rr}^{(5)} \\ \tau_{r\theta}^{(5)} \end{bmatrix} r^{BC} \omega L \quad (16)$$

$$\left(\frac{\partial u_\xi}{\partial \theta} \right)^{(5)} \approx \left(\frac{\partial u_\xi}{\partial \theta} \right)_{(i+1,j)} \quad (17)$$

where \mathbf{F}^{BC} is the force acting on boundary surface 5. This equation is described by the following Modelica code.

```
Fr_bc=tau_rr*r_bc*omega*Lz;
Ft_bc=tau_rq*r_bc*omega*Lz;
durdt=Cra.durdT;
dutdt=Cra.dutdt;
```

2.5 Calculation of the displacement at the vertices of the elements

In this section, a method for calculating the displacements at the vertices of the elements (○ marks in Figure 4) is described using an example illustrated in Figure 4. Here the displacements at the center positions (◇ marks in Figure 4) have already been obtained by the models described in the previous sections.

The displacements at the vertices are calculated using the positions, the displacements and the displacement gradients of the surrounding elements. First, the displacement of the position \mathbf{r}_1 is obtained using the information of the position \mathbf{r}_2 as follows.

$$\mathbf{u}_{1f(r_2)} = \mathbf{u}_2 + \text{grad}(\mathbf{u}_2)(\mathbf{r}_1 - \mathbf{r}_2) \quad (18)$$

Also the same displacement of the position \mathbf{r}_1 is obtained using the information of the other positions \mathbf{r}_3 , \mathbf{r}_4 , and \mathbf{r}_5 , in the same manner. The displacement of this point is defined as the average of these values obtained using the information of the surrounding elements.

$$\mathbf{u}_1 = \frac{1}{4} (\mathbf{u}_{1f(r_2)} + \mathbf{u}_{1f(r_3)} + \mathbf{u}_{1f(r_4)} + \mathbf{u}_{1f(r_5)}) \quad (19)$$

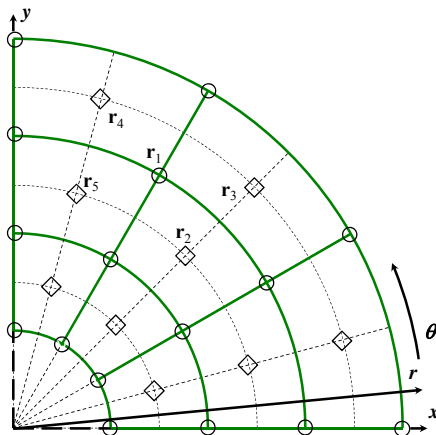


Figure 4. Vertices and center positions of the elements.

3 Simulation examples

The simulations were performed for a cylindrically shaped object using Dymola and the results were compared with those obtained by FE analyses.

3.1 Simulation target and conditions

Figure 5 shows the simulation target and examples of its FE analyses. The object consisted of the separate upper and lower halves of cylindrically shaped parts, and the upper half was placed on the lower half. Since this object was symmetrical between the left and right sides, the left half was analyzed. The analyses were performed for two cases of temperature conditions: a uniform temperature under which the object was thermally expanded in a uniform manner (Figure 5 (a), Case 1), and a radial temperature distribution under which both the upper and lower halves were deformed (Figure 5 (b), Case 2). To deal with the original three dimensional deformation by the two dimensional model, the deformation was analyzed in several r - θ planes on which the plane stress problem was applied.

Figure 6 (a) shows the discretization of the simulation target for the proposed method. A quarter was analyzed assuming that the upper and lower halves having the same geometries deformed symmetrically.

The original shape (not exactly a cylinder) used in the above FE analyses was simplified into an exact quarter cylinder; the inner and outer radii were determined so that the volume was kept equal to the original. The object was discretized into 60 elements (4×15 in the r and θ directions). For the boundary conditions, the circumferential displacements on one side of the circumferential boundary surfaces were restrained. The length L_y in the y direction between the positions A and B was used to evaluate the accuracy of the proposed method.

Figure 6 (b) shows the temperature conditions for Case 1 and 2. In Case 1, the object was maintained at a uniform temperature of 486°C . In Case 2, the inner surface of the object was maintained at 486°C and the outer surface at 342°C . The initial temperature before the occurrences of the thermal deformations was determined to be 15°C .

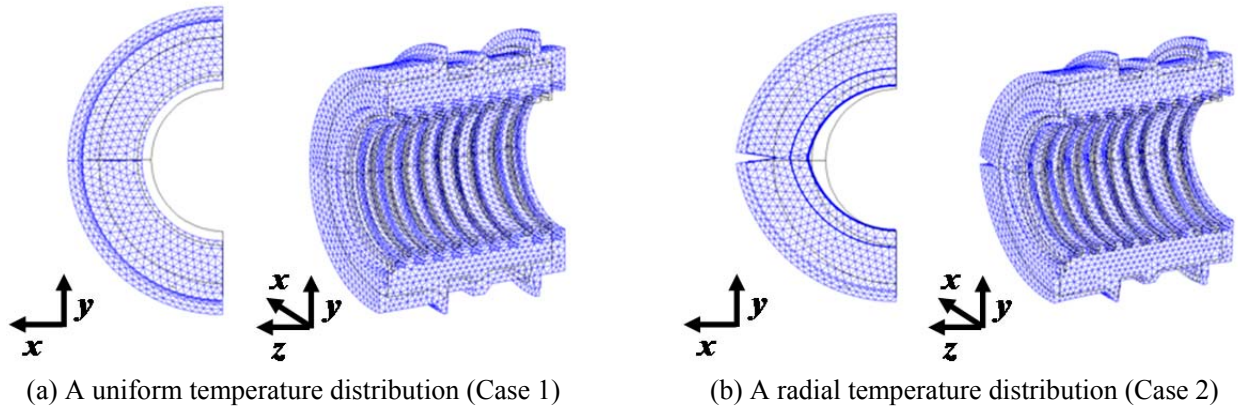
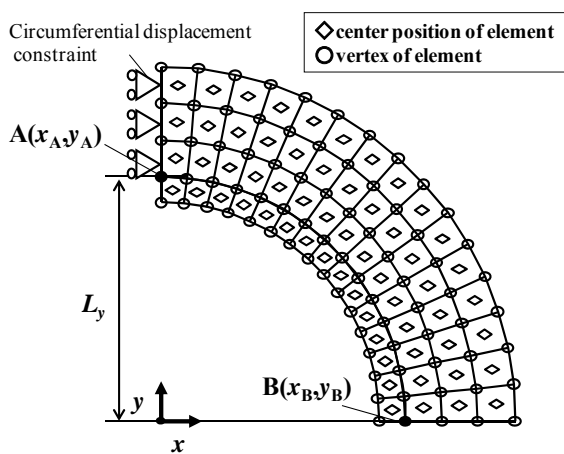
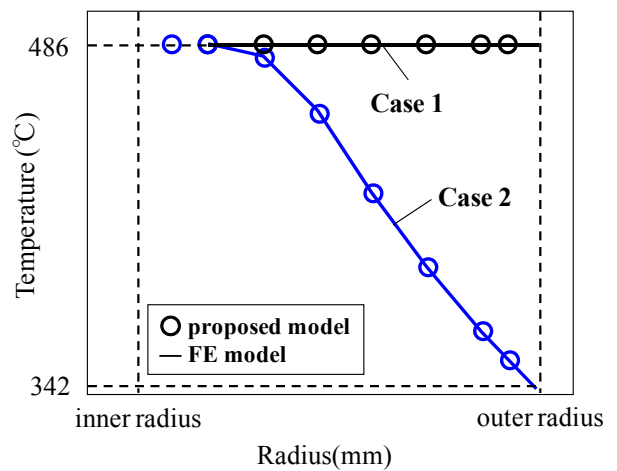


Figure 5. Simulation target.

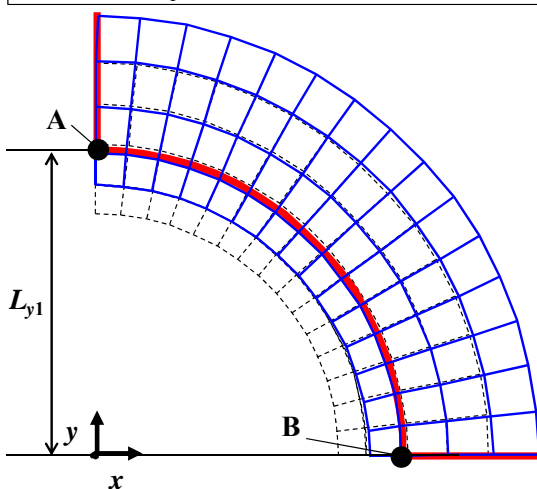
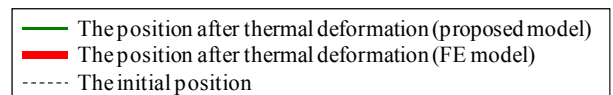
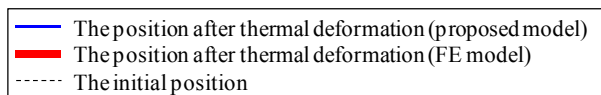


(a) Discretization of the simulation target for evaluating the proposed model

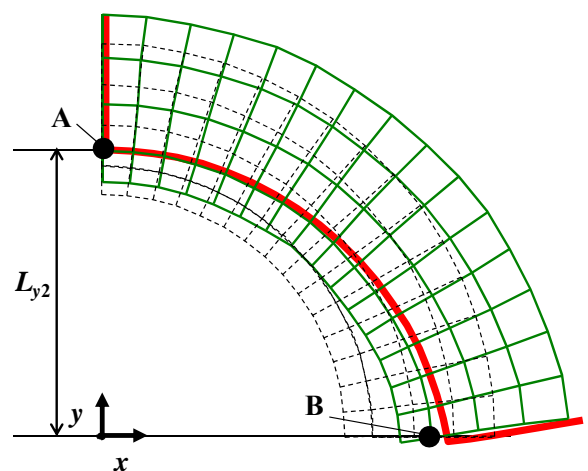


(b) Temperature conditions

Figure 6. Simulation conditions.



(a) A uniform temperature distribution (Case 1)



(b) A radial temperature distribution (Case 2)

Figure 7. Simulation results. The displacements were magnified 20 times.

Table 4. Differences in the number of elements and the simulation results between the proposed model and the FE model.

	Proposed model	FE model	Difference
Number of elements in the x - y plane	60	500	—
Length of initial condition (L_{y0})	1(-) ^{*1}	1(-) ^{*1}	—
Displacement of Case 1 ($\Delta L_{y1} = L_{y1} - L_{y0}$)	6.0×10^{-3} (-) ^{*1}	6.5×10^{-3} (-) ^{*1}	-7.6(%) ^{*2}
Displacement of Case 2 ($\Delta L_{y2} = L_{y2} - L_{y0}$)	3.1×10^{-3} (-) ^{*1}	3.7×10^{-3} (-) ^{*1}	-9.0(%) ^{*2}

*1: Dimensionless value using the length of the initial condition L_{y0}

*2: Difference(%) = $(\Delta L_{y_Proposed} - \Delta L_{y_FE}) / \Delta L_{y_FE} \times 100$

3.2 Simulation results

Figure 7 (a) and (b) show the initial position and the positions after thermal deformations in Case 1 and 2, respectively. The positions after thermal deformations were compared those calculated by an FE model. The number of elements in the x - y plane of the FE model was about 500 (The total number of elements in three dimensions was 26335). The simulation results obtained by the proposed model were in good agreement with those obtained by the FE model in both Cases 1 and 2.

The differences in the number of elements and the simulation results between the proposed model and the FE model in both Cases 1 and 2 are summarized in Table 4. The differences in simulated displacements were less than 9% while the number of elements that compose the proposed model was about 1/8 ($\approx 60/500$) compared to that of the FE model.

4 Summary and discussion

A thermal deformation analysis method was proposed that fully utilizes the non-causality of the Modelica language as a means of formulating and solving large scale simultaneous equations including equilibrium equations related to stresses, stress-strain relations and strain-displacement relations. The main feature of the proposed method is that the positions of displacements and forces are defined based on a finite volume method to effectively describe these complex physical phenomena in a non-causal manner and also to implement them as a network of Modelica component models.

As an illustrative example, a Modelica model for thermal deformation analysis of a cylindrical object in a two-dimensional circular polar coordinate system was presented. Simulations were performed for a cylindrically shaped object under a uniform temperature distribution and a radial temperature distribution. The results of simulations showed that the differences in the displacements between the proposed model and the FE model were less than 9% while the number of elements that compose the proposed model was about 1/8 compared to that of the FE model.

The proposed method can deal with more complex thermal deformation analyses than presently used analytical solutions and can obtain precise outputs comparable to that of FE methods with a fewer number of elements. These advantages make it possible to perform thermal deformation analyses in system level Modelica simulations containing structural objects and control devices. Furthermore, the method should be applicable to optimizations of geometries of such objects or control systems.

Nomenclature

A	position	[-]
B	position	[-]
E	Young's modulus	[Pa]
\mathbf{F}	force vector	[N]
G	shear modulus($=E/2/(1+\nu)$)	[Pa]
L_y	length between positions A and B	[m]
L_{y0}	length of L_y in the initial condition	[m]
L_{y1}	length of L_y in case 1	[m]
L_{y2}	length of L_y in case 2	[m]
L_z	axial length of an element	[m]
\mathbf{r}	position vector	[m]
r	radius	[m]
T	temperature	[°C]
T_0	temperature at the initial condition	[°C]
\mathbf{u}	displacement vector	[m]
u_r	displacement in the radial direction	[m]
u_θ	displacement in the circumferential direction	[m]
α	coefficient of linear expansion	[1/°C]
ϵ_{rr}	strain in the radial direction	[-]
$\epsilon_{r\theta}$	shear strain	[-]
$\epsilon_{\theta\theta}$	strain in the circumferential direction	[-]
τ_{rr}	stress in the radial direction	[Pa]
$\tau_{r\theta}$	shear stress	[Pa]
ν	Poisson's ratio	[-]
ω	angle of an element	[rad]

Subscripts

- $f(\mathbf{r})$ calculated using the positions, displacements and displacement gradients of position \mathbf{r}
- i number of elements in the radial direction
- j number of elements in the circumferential direction
- ξ in the radial and circumferential directions

Superscripts

- BC boundary surface of simulation target
- (1)-(4) boundary surface of element
- (5) between boundary surface 5 and center position

References

- Elmqvist H., Mattsson S.E. (1997a): Modelica - The next generation modeling language an international design effort, Proceedings of the 1st World Congress on System Simulation
- Elmqvist H., Mattsson S.E. (1997b): An introduction to the physical modeling language Modelica, Proceedings of the 9th European Simulation Symposium
- Elmqvist H., Mattsson S.E., Otter M. (1998a): Modelica - The new object-oriented modeling language, 12th European Simulation Multiconference
- Elmqvist H., Mattsson S.E., Otter M. (1998b): Modelica - An international effort to design an object-oriented modeling language, Summer Computer Simulation Conference
- Ferziger J.H., Peric M. (2002): Computational Methods for Fluid Dynamics, Springer-Verlag, pp.21-37
- Fritzson P. (2003): Principles of object-oriented Modeling and Simulation with Modelica 2.1, John Wiley & Sons, Inc., pp.19-25
- Fritzson P. (2011): Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica, John Wiley & Sons, Inc., pp.29-39
- Fritzson P., Engelson V. (1998): Modelica—A unified object-oriented language for system modeling and simulation, ECOOP'98, LNCS 1445, pp. 67-90
- Gere J.M. (2006), Mechanics of materials, Thomson learning, pp.93-104
- Voller V.R. (2009): Basic control volume finite element methods for fluids and solids, World Scientific Publishing Co. Pte. Ltd., pp.10-20

Validated Modelica Building Package for Energy Performance Simulation for Educational and Teaching Purposes

Shan Hua¹ Fabian Reuß² Manuel Lindauer¹ Jochen Stopper¹ Christoph van Treeck³

¹Centre for Sustainable Building, TU München, Germany, {shan.hua, manuel.lindauer, jochen.stopper}@tum.de

²Institute of Energy Efficient and Sustainable Design and Building, TU München, Germany, fabian.reuss@tum.de

³Chair of Energy Efficiency and Sustainable Building E3D, RWTH Aachen University, Germany, treeck@e3d.rwth-aachen.de

Abstract

This paper introduces the work and application of the Modelica IBPSBuilding package which has been developed at TU München. The aim of the package is to simulate building energy performance, especially the thermal behaviour of buildings. Its application is focused on research and education in order to help application-oriented researchers and students to understand the physical processes of a building performance simulation and gain persuasive simulation results. The package describes all basic processes including conduction, convection, radiation and ventilation, and it is validated with German VDI-Guidelines.

Keywords: building modeling, energy simulation, adaptive façade

1 Introduction

The teaching module “Building Performance Modeling and Simulation” was established at TU München in 2006 which is based on a formerly created block course about simulation modeling using Maple. It consists of a lecture with the same name as the module and the practical exercise “Implementing a Building Performance Simulation”. The lecture explains the physical basics of thermal building simulation including heat conduction, convection, short- and longwave radiation processes, but also introduces mathematical and numerical aspects of simulation.

Theories explained in the lectures are applied in the exercise, in which Modelica is used to create models of all the involved processes (van Treeck, 2010). These models evolved over the years and new ideas with a whole building model that had been developed in master theses and exercises were implemented. The development was conducted with OpenModelica since 2012 and then several functional enhancements and adaptations to

research projects were conducted with ITI SimulationX. The actual state of the model package and the relative development is shown in this paper. Additionally, the methodology and results of a validation of the model package according to a German technical guideline is presented.

2 The Library IBPSBuilding

The aim of the IBPSBuilding library is to have a thermal simulation tool, in which every single process related to building thermal performance is fully understood and can be modified in any desired way in order to have maximal flexibility. Especially for the aspect of understanding every detail, it is necessary to create a library from scratch and not to use existing libraries.

The main structure (in Figure 1) of this model library meets the lecture syllabus, and contents are complemented and explained to students step by step along the pace of the lecture. Taking advantage of the polymorphism in Modelica language, complicated physical processes and building elements are designed to be based on abstract concepts and structures. Base classes for interfaces, functions and records for materials are arranged into sub-packages, which are provided as modules that students can use to create their own elements for heat transfer processes and construction elements. The methodology of development of the standard components of heat transfer processes provided in this library is introduced in section 2.1. At the end of each semester, typical rooms should have been created and simulated by the students with support of the lecturers. The simulation results of these test cases can be used to get insights on the most responsive processes and indicators of thermal building performance.

In order to let the library structure be commensurate to the domain structure of a BIM data format like IFC, a skeletal framework is provided by a master thesis for de-

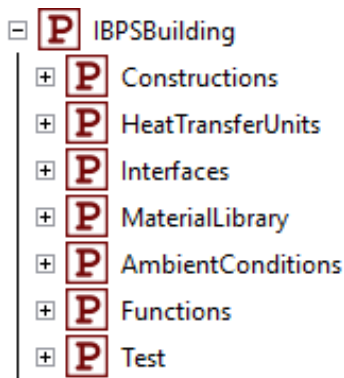


Figure 1. main structure of IBPSBuilding-package

velopment of appropriate interfaces in research according to this library (Hua, 2014).

2.1 Implementation of heat transfer processes

The heat transfer processes described in this library can be classified into two groups: the heat transfer processes determined by temperature difference, and the shortwave radiation unrelated to temperature. All the building elements are simplified into one-node thermal mass with capacity models of the material and the key positions, e.g. surfaces. Each building element is integrated into a chain of connections of various physical processes (Figure 2). Coefficients and properties for concretization of those physical processes are given as parameters or calculated dynamically in the construction component model.

In order to reduce the complexity of modeling and to increase the simulation efficiency on the building level, several restrictions are contained in the development and modeling:

1. Currently, thermo fluid processes are not considered in the teaching version. For any adoption with fluid materials, simulations are conducted under ideal conditions with a single heat capacity, convection and steady state mass transfer - pressure losses, change of density and enthalpy are disregarded in the current phase of development. So no interaction with elements in the 'Modelica.Fluid' library is adopted.
2. The discretization follows a nodal approach for calculating heat transfer through the building fabric in a one-dimensional manner, i.e. where heat transfer perpendicular to the main direction of heat flow is neglected. Multi-dimensional effects such as cold bridges are accordingly disregarded.

2.1.1 Conduction and convection

The modules for conduction and convection process are based on the same abstract partial model:

$$\dot{Q} = C \cdot (T_i - T_e)$$

Basic properties like the temperature difference between the two sides of a layer, the inlet / outlet heat flow etc. are declared as prescribed in the abstract model, so that programming effort can be minimized, especially if the concretized modules would be coded by students in practical fields, who are less familiar with programming. The coefficient C is first specialized in different successors from the abstract model, e.g. in a conduction module for half of a solid material layer, $C = A \cdot \lambda / d / 2$, here A is the net area of a material layer, λ is the material conductance, and d is the layer thickness.

Concretization of values of heat transfer coefficients for convection depends on the boundary condition of the building element very much. Within the concept of flexibility and full controllability, a bunch of parameters are added, so that it can be chosen if fixed values for the heat transfer coefficients should be assigned for validation or conceptual design phases, or the values should be calculated dynamically and internally according to geometry, wind velocity, temperature differences etc. A theoretical reference of most of the physical processes described in this package is from Clarke (2001). Chances are given to students who are taking part into the seminar to create new functions and codes to implement other theoretical methods. An example is to calculate the convection coefficients according to boundary conditions and tilt angle. A function `calConvectionCoeff` is implemented like this:

```

if useFixConvectionCoeff then
  alphaC_e:=fixAlphaC_e;
  alphaC_i:=fixAlphaC_i;
else
  if tilt >=-pi/4 and tilt <=pi/4 then
    alphaC_i:=1.63 * abs(dT_i)^(1/3);
    alphaC_e:=if connectToAdiabat then
      9999.9 elseif isExternal then
      1.8 + 4.8 * windSpeed else
      0.6 * (abs(dT_e) / L^2)^(1/5);
    elseif tilt >pi/4 and tilt <pi*3/4 then
      ...

```

where the values of Boolean parameters such as `useFixConvectionCoeff`, `connectToAdiabat` and `isExternal` are assigned by high-level modelers from input-windows. The variable `tilt` is transmitted from the geometric parameter-set of the building element, and serves as the arbiter of which formula should be taken into consideration.

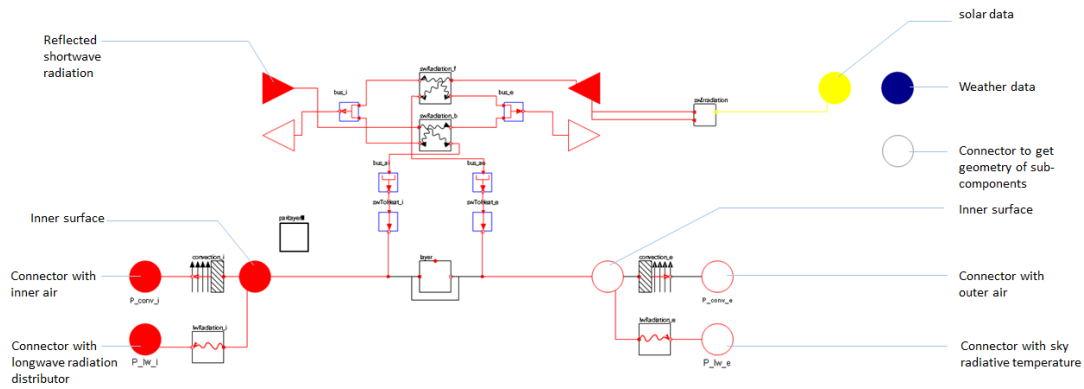


Figure 2. An opaque building element and its accessible connectors

2.1.2 Radiation and window modeling

By contrast with heat transfer models like conduction and convection, a shortwave radiation process is not based on `HeatPorts`¹ with temperature as the potential variable because of its independence of the temperature difference, therefore the direction of the heat flow by radiation is given explicitly. Besides the heat flow variable, a factor is stored in a connector `SwPort` for distributions of radiation to its parent model. A shortwave radiation module has four `SwPorts`: one as input to receive incoming radiation, and the other three stand for the outputs through transmission, reflection and absorption processes. Calculation methods of the factors are first assigned either in inheritance or in instantiation process.

A transparent construction element should have four shortwave radiation modules because shortwave radiation takes place on a plane between two mediums (in case of window are glass and gas). Moreover, each glass pane includes two surfaces, and each of them conducts radiation from both sides, that is to say, from glass to gas, and from gas to glass. Intermediate connections are shown in the following figure 3.

Within this structure, radiation inside a glass plane between the two surfaces are taken into account in numerical handling process by simulation solver. It can be solved less efficiently than an analytical method, but more clearly represented for non-programmers to help them understand actual physics behind the model. Absorbed energy is summed up into a hub, and converted into a thematic adaptable format into `HeatPort`, which can be connected to the layer heat capacitor. (connected with `P_a`, Figure 2)

¹The connector `HeatPort` is inherited from the Modelica standard library - `Modelica.HeatTransfer.Interfaces`. It consists of a potential variable temperature and a flow variable heat flow rate.

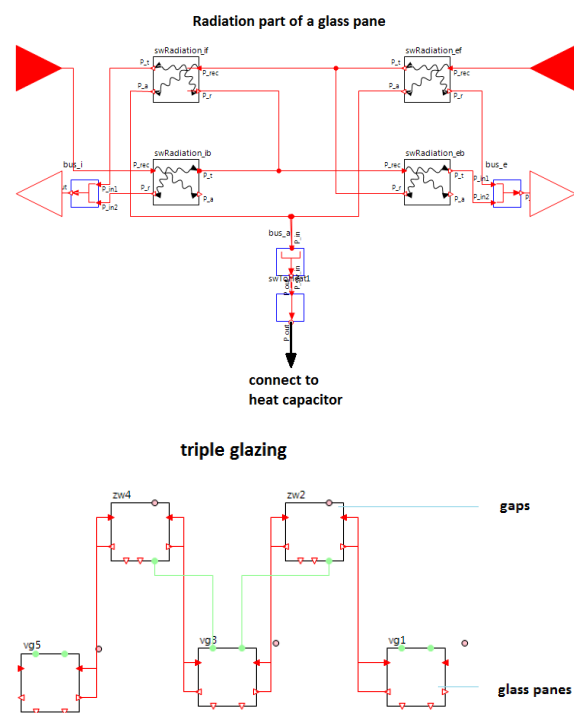


Figure 3. Connections between radiation inside a glass pane and between glass panes and gaps

2.2 Room-modeling

Thermal space or room is considered to be the basic unit of building energy performance simulation. Different rooms can be connected by using shared building elements, general boundary conditions and central service systems. Connections inside one room between building elements and other components such as internal loads could be more complicated, and require special attentions. Therefore, a template `RoomEmpty` is available to help students and engineers build a simulative room model by using “drag-and-drop”. In this template, indoor air mass, internal load, radiation distributors and hub-connectors are given. This structure can also provide

support for automatic generation of a Modelica room model from building information models.

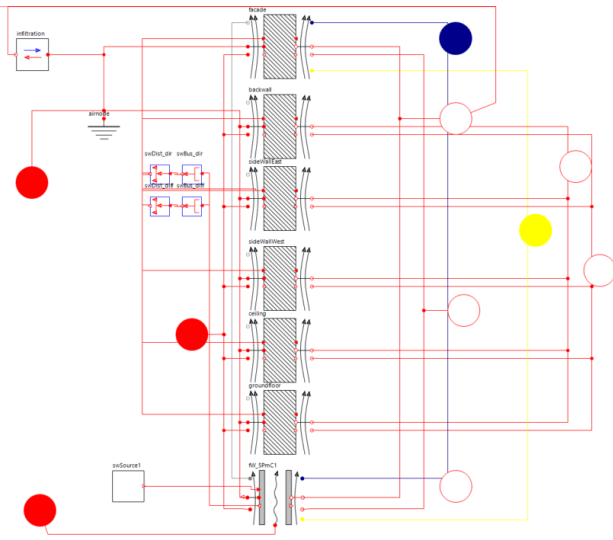


Figure 4. An example room model with six walls (1 of these is an external wall) and a window

2.3 Heating system

As part of a master thesis, the existing IBPBuilding library was extended by components for simulation of hot water central heating systems, which are the most important plant types in Germany. The implemented component models are widely accepted best practice models and are primarily taken from existing Modelica libraries, such as the Annex60 library (Wetter and van Treeck, 2016) and the Modelica Buildings Library 2.0 from Lawrence Berkeley National Laboratory (Wetter et al., 2013). The library contains all essential components of pipes, fittings, pumps, heat generators, storages, expansion tanks, transfer systems, control strategies and domestic hot water (DHW) systems to simulate a wide range of complex heating and DHW systems. The newly implemented heating system models of IBPBuilding library should therefore be seen as a basis for further developments.

2.4 Compatibility and external connections

In the framework of a public license, the library is considered to be compatible with the Modelica standard library:

1. The heatports of the sub-package `Interface` are inherited from the `HeatPort` of the standard Modelica library. Connectors in the models of IBPBuilding can be connected directly with standard modules such as with heat sources in `Modelica.HeatTransfer` package or indirectly through converter-modules.

2. Variables are defined using `Modelica.SIunits` types.

3. Use of external functions are minimized. Models and functions are verified with test rooms in the environment of OpenModelica and ITI SimulationX.

2.4.1 Compatibility by encapsulating the parameter sets

To enhance the compatibility and adaptability of this library, instantiation of room or building models should be conducted automatically by using interfaces or under coordination by platform-software. Automated Modelica-code generation has restrictions such as that no new variables and equations should be constructed, so that the generated graphical representation of models should be readable for engineers. Furthermore, it should be proved that the generated model can be simulated without errors which could be produced by incomplete interpretation and partly combination with manually inserted components.

For this purpose, parameters that essentially can come from an exchangeable data format are encapsulated into records as a property of appropriate building element. The following example shows the instantiated and concretized parameter-set in a concrete wall:

```

Constructions.ConstructionComponent extWall(
  par(
    azimuth=0, height=3, length=5,
    nLayers=2, nWindows=1),
  parLayer1(
    redeclare replaceable
    MaterialLibrary.BuildingMaterials.
    Plaster_gips mat,
    thickness=0.02),
  parLayer2(
    redeclare replaceable
    MaterialLibrary.BuildingMaterials.
    Concrete mat,
    thickness=0.2), ...)

```

2.4.2 Weather reader and external CombiTables

The IBPBuilding-package uses `CombiTable`(e.g. `Modelica.Blocks.Tables.CombiTable1D`) for indicator-related properties of components (e.g. temperature-related fluid density, wavelength-related refractive index or spectral sensitivity) and schedules. An Excel-Macro is developed to convert any data-sheets into Modelica-readable format or time table. A reader of weather data based on the appropriate component in ITI GreenBuilding-package of SimulationX (Unger et al., 2012) is implemented for semantic analysis of `CombiTables` of weather data according to date and site location, and provides connectors for other components. It is designed to be compatible

with components of GreenBuilding-package and standard Modelica libraries.

2.5 Collaboration with students

Since 2015, for version management the software revision system Git was applied for code organization in teaching activities. Creating modules in different student groups and merging these development trees was the main motivation. In the exercise of the winter semester 2015/16 this idea was realized by creating a Github repository with a basic version of the source code of the IBPSBuilding library and establishing student development groups for the topics convection and longwave radiation processes. This repository is created under the Github-Education program², which is held in private use, and can at the moment only be published with the scope of a virtual ‘classroom’ with access control of tutors in the lecture. As most of the students with background of architecture and civil engineering had never been working before with software development environments, a lot of technical problems arise. One of the biggest challenges was to motivate the use of Github, which in the end was only used by the students for downloading the basic version of the library. Pushing new source code versions created by the student groups and merging the development trees had to be done by the supervisors. In the next course Github will be used again, but with more introduction examples to increase the usage of Git as a collaboration platform in and between the groups. Additionally Git will be used in future Master’s theses on the library, as in such a thesis more time can be spent on clarifying technical problems. The repository is under GNU GPL license.

3 Validation

Thermal building simulation models should be tested to eliminate model errors and to state the quality of the results in relation to calculation accuracy and reality. Quantitative testing can be done with a relative or a physical validation. By a physical validation the simulation model is tested with measured values from a real system, e.g. a well-documented real experiment. In practice, however, a relative validation often is the more convenient choice. In the case of a relative validation a new simulation model is compared to existing simulation models, e.g. by evaluation parameters of national and international validation standards or guidelines. They usually provide different test cases to verify the implemented algorithms and models of certain areas of the program. To evaluate the calculated results comparison values are specified. In this paper, the German guideline

²More information: <https://education.github.com/>.

VDI 2078³ from the German Association of Engineers (VDI) has been used.

3.1 Selection of the validation case

The thermal solver of the IBPS Building library is not based on the 2-capacity (2-K) model by Rouvel and Zimmermann (Rouvel, 2015) as described in the VDI 6007-1. For simulation programs with another thermal solver than described in VDI 6007-1 and / or radiation model as in VDI 6007-3 the VDI 2078 prescribes a validation according to “Case B”(VDI 2078). This validation case includes:

1. Validation according to test examples 1 to 16 (except 11) of VDI 6020-1 (implicitly included in the test examples of VDI 6007-1 and VDI 2078)
2. Validation of test examples of VDI 6007-1 (Test Example 1-12)
3. Validation of test examples of VDI 2078 (Test Example 1-16)

For this validation case the guideline specifies limiting values for the mean values of the hourly and standard deviation from the reference results of VDI 6020-1. The reference results are based on the n-K model by Rouvel and were calculated using the building simulation program GEBSIMU (Rouvel, 2015). The simulation program results must lie in these limits.

Though, a complete validation of the IBPSBuilding library according to VDI 2078 is not expedient. Part of the test examples of VDI 6020-1 and VDI 2078 require expensive whole-year simulations which clearly complicate the assessment and analysis of test results. Furthermore, features such as window ventilation and daylight calculation, are required in these test examples which are not yet included in the existing library. Therefore, a validation with the test examples of VDI 6007-1 and limiting values according to VDI 2078 is of primary importance for an evaluation of the thermal solver.

3.2 Type rooms

For the calculation of the test examples, a simple test room (type room) is defined in the guideline VDI 6007-1. The construction of the room corresponds, depending on the test example, either to a lightweight construction

³The guideline VDI 2078 “Calculation of cooling load and room temperatures of rooms and buildings (VDI Cooling Load Code of Practice)” is used to determine the cooling load, the ambient air temperature and the operative room temperature for rooms of all kinds, with and without air conditioning (VDI 2078). The guideline summarizes selected test examples of VDI 6007-1 (“Calculation of transient thermal response of rooms and buildings - Modeling of rooms”) and VDI 6020-1 (“Requirements on methods of calculation to thermal and energy simulation of buildings and plants - Buildings”). These should serve both the validation as well as support for programming.

(room type L) or a massive construction (room type S). The resulting two different amounts of thermal mass allow to examine the corresponding reactions of the room to these differences. The test room is defined as part of a building and therefore has adiabatic boundaries. From case to case, the outer wall is adjacent to different external climate.

3.3 Test cases

The guideline VDI 6007-1 comprises a total of twelve test examples (VDI 6007). The test examples 1 to 7 correspond to the guideline VDI 6020-1 and examine reactions of the room to internal loads and setpoint changes. The test examples 8 to 12 contain important functions of the extended 2-K model by Rouvel, e.g. surface heating or cooling, non-adiabatic internal components, air exchange and several external components.

The thermal reaction of the test room is calculated over a period of 60 days in hourly increments. The pre-determined external and internal heat sources and sinks are in temporal pattern of a day the same for the whole period of 60 days. Thus, the transient and the steady state can be compared to the reference results in the guideline.

The kernel of VDI 6007-1 is based on the 2-K model by Rouvel and Zimmermann (Rouvel, 2015). Since the IBPS Building library is designed for the simulation of detailed room models, as suggested by Clarke (Clarke, 2001), the test examples are created accordingly. The basic structure corresponds more to the n-capacity (n-K) model of VDI 6020-1. However, there are some basic differences to the reference n-K model of the building simulation program GEBSIMU, as described below.

With the existing components of the IBPS Building library it is not possible to calculate all of the test examples of the VDI 6007-1. It is therefore necessary to implement some additional components or extend existing components that are comprised in the kernel of VDI 6007-1.

3.4 Component model

The n-K model (of the building simulation program GEBSIMU) uses a simplified model suggested in VDI 6020-1 for capacitance and resistance of the individual components of a room (VDI 6020). For each single or multilayer component a substitute model is calculated using a chain matrix from concentrated thermal resistances and capacitances. These capacitances and resistances reflect the thermal mass of room components that can be activated. By this, the common solution methods such as Fourier or Laplace transformation can be avoided. The active thermal mass is described by the aperiodic depth of penetration and is calculated regarding the considered period duration.

A substitute model for a wall with any number of different layers can be combined into an equivalent resistor-

capacitor (RC) circuit with a maximum of three resistors and two capacitors. Asymmetric thermally loaded components, such as external components or internal components to different tempered neighboring spaces are described by two resistors and one capacitor. Symmetrical thermally loaded components, i.e. adiabatic internal components are modeled as a damped heat storage with one resistor and one capacitor (see Figure 5).

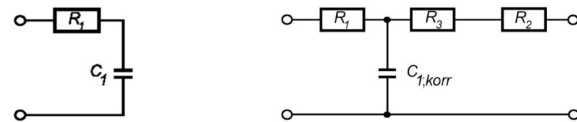


Figure 5. Equivalent circuit for a symmetrical (left) and asymmetrical (right) loaded component (Rouvel, 2015)

The IBPS Building library's equivalent models for single- or multi-layer components are based on the Beuken model. The Beuken model is based on the agreement between the differential equation of the thermal conduction and the processes in an idealized electrical cable (VDI 6020), as shown in Figure 6.

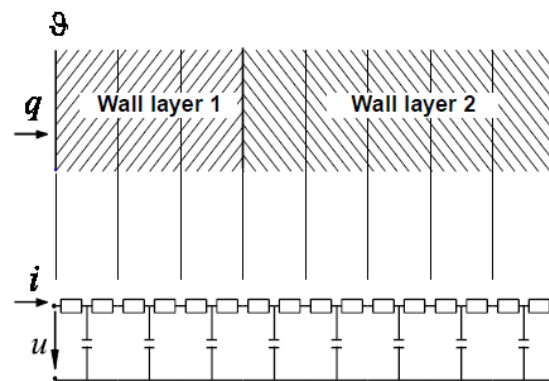


Figure 6. Component as Beuken model (VDI 6020)

In the Beuken model any wall layer is represented by an RC circuit with two resistors and one capacitor. A component (e.g. a wall) can basically be divided into any number of layers with an equivalent circuit. The degree of subdivision of the wall layers is normally chosen based on the required calculation accuracy but in principal any required calculation accuracy is possible. No requirements are stipulated regarding the boundary conditions (such as linearity etc.). In the IBPS Building library the spatial discretization of component models is limited to one RC circuit per component layer, i.e. per material layer. This allows a significantly faster computation time.

In the IBPSBuilding library based detailed models components to adiabatic boundaries, that is, interior walls, door, floor and ceiling, are approximated by only half of the input thickness. In VDI 6020-1, the "corrected heat storage capacity" of the symmetrically loaded components, i.e. those to adiabatic boundaries, are cal-

culated in dependence of the aperiodic depth of penetration. The simplified assumption in the detailed model leads to overall smaller heat capacities, whose impacts can be observed in the simulation results. The windows are modeled in accordance with VDI 6020-1 as opaque component with thermal resistance but without heat storage.

3.5 Longwave radiation

Longwave radiation exchange between different surfaces in the building simulation program GEBSIMU is assumed to be calculated by the approach for “more complicated” geometrical conditions of the VDI 6020-1 (VDI 6020). This simplified linear approach assumes that all surfaces of the room are involved in the radiation exchange relative to their size:

$$\dot{Q} = A \cdot \alpha_{\text{str}} \cdot (T_1 - T_2)$$

In VDI 6007-1 the heat transfer coefficient is defined as $\alpha_{\text{str}} = 5 \text{ W}/(\text{mK})$ (VDI 6007). According to ISO 6946:2007 “Building components and building elements - Thermal resistance and thermal transmittance - Calculation method“ this value corresponds to a temperature of $T_m = 10^\circ\text{C}$. Thus, it is assumed that this value is also used in the GEBSIMU model. The IBPS Building library uses an approach based on the Stefan-Boltzmann law:

$$\dot{Q} = A \cdot \sigma \cdot \varepsilon \cdot (T_1^4 - T_2^4)$$

This leads to higher radiative heat flows between the walls for temperatures over 10°C . Therefore, the indoor air temperature on day 60 in the guideline is higher than in our model. This originates in the radiative heat flow from the inner to the outer wall, which is underestimated in the guideline and leads to lower heat losses to the outside.

3.6 Simulation

After entering all data the test cases are simulated over the base period of 60 days. The output is given in hourly steps as specified in the directive. By evaluating the results, programming errors in the IBPSBuilding library could be detected and eliminated.

The reference results of validation “Case B” are based on the n-K model by Rouvel which is included in VDI 6020-1 (VDI 6020). These were calculated using the building simulation program GEBSIMU version 7.30.0011. The basics of the n-K model were outlined by Rouvel in 1972 based on the detailed Beuken model. The compliance of both models calculation results has been proved by Rouvel (Rouvel, 2015).

3.7 Results analysis and discussion

The simulation results of the test cases of VDI 6007-1 are compared to the reference results of validation “Case B” according to the specifications of VDI 2078. The reference results are in this case based on the n-K model as included in VDI 6020-1.

For analysis of the test examples, day 1, 10 and 60 of the base period are compared in respect to room air temperature, operative temperature, and for test examples 6, 7 and 11, to heating and/or cooling load. By this, initial values, transient response and consistency in steady state can be examined. Figure 7 exemplarily shows the transient response of Test Example 1 over the whole base period of 60 days.

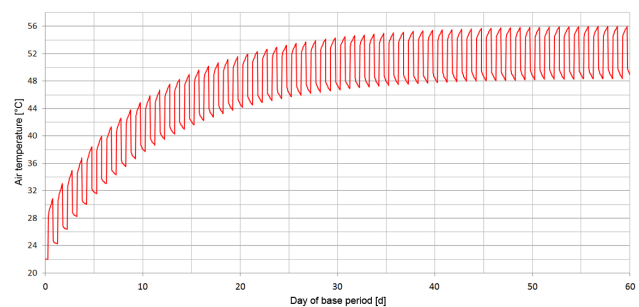


Figure 7. Development of the room air temperature over the simulation period

Figure 8 exemplifies the diurnal variations of the indoor temperatures of Test Example 1. The increase in air temperature due to the internal loads in office hours (6.00 - 18.00 o'clock) can be clearly seen. Compared are the reference results of the 2-K model of VDI 6007-1 and the reference results of the n-K model of VDI 6020-1 with the simulation results of the IBPS Building Library. The maximum standard deviation of the hourly variations of the simulation results from the reference results occurs for the room air temperature and the operative temperature on day 1 with 0.28 K. The maximum absolute deviation is about 1.0 K and occurs on day 60. The test example shows an overall good compliance with the reference results. However, differences in the calculation engine between the n-K model and the IBPS Building library’s model result in significant variations in all test examples.

It is apparent that the indoor air temperature shows a wider fluctuation range on all days than the reference results. This can be justified by the smaller overall effective heat capacity of the zone. As described before, the heat capacity of the reference model is calculated in accordance with VDI 6007-1, i.e. in dependence of the aperiodic depth of penetration. The heat capacity to adiabatic boundaries of the IBPSBuilding library model is simply estimated by half the wall thickness. In this way the total heat capacity of the IBPS Building model is smaller and the room air temperature is responding

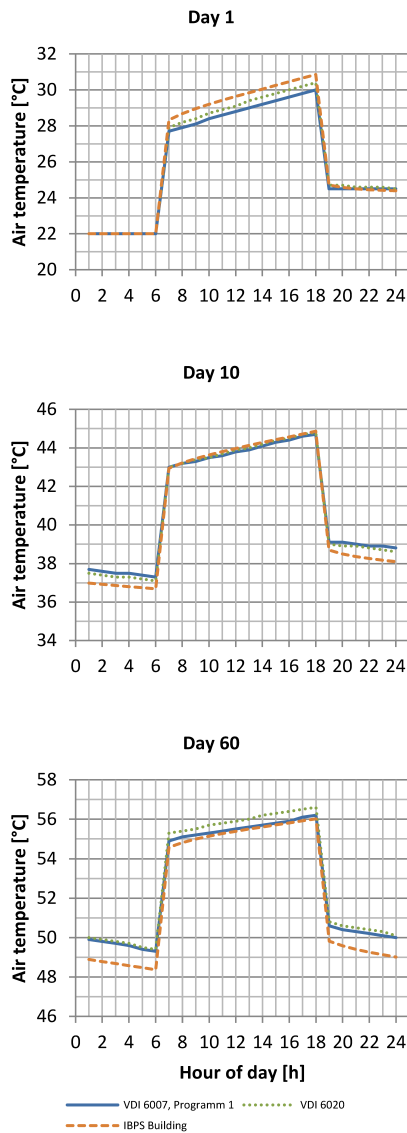


Figure 8. Results of test example 1

faster to heat sources and heat sinks on every day, until it reaches the upper limit in steady state.

Furthermore, the room air temperature of the IBPS-Building model increases slower during the simulation period than in the reference model. This is attributable to the heat flow from the inner to the outer wall by long-wave radiation, which is considered too small in the reference model. As described before, in contrast to the n-K model, the IBPSBuilding library does not use the simplified linear approach for the long-wave radiation exchange, but a model based on the Stefan-Boltzmann law. This leads to higher radiative heat flows between the walls at temperatures above 10°C and thus to higher thermal losses through the exterior walls. The room air temperature is therefore, on day 60 lower than in the reference results. The two behaviors described above occur in all test examples.

Minor discrepancies at all considered days can be at-

tributed to slight differences in the calculation results by using different calculation programs (VDI 6007). Depending on the programming, programming language, compiling, etc. slight differences are inevitable.

In heating and cooling load significantly higher deviations from the reference results are conveyed. In Test Example 11 the room air temperature is in good consistence with the reference results with a maximum standard deviation of 0.47 K on the 1st day. Though the system load has high standard deviations from the reference results of 65.6 W on day 1 and approximately 56.4 W on day 10 and 60. The maximum absolute deviation occurs on day 1 with about 190 W.

This is primarily due to the low spatial discretization which aggravates the correct identification of the activated thermal mass. Another factor is the use of a PID (proportional-integral-derivative) controller to determine the required heating and cooling load. Here, the n-K model of GEBSIMU according to VDI 6020-1 uses a discrete analytical approach to determine system loads, which leads to fundamentally different results (VDI 6020).

Evaluation is performed as a statistical analysis by mean value and standard deviation corresponding to VDI 6007-1 (VDI 2078). The mean value is the mean of the hourly variation, i.e. simulation result minus reference result, in the evaluation period. The evaluation period includes day 1, 10 and 60 of the simulation period. The standard deviation is the standard deviation of the hourly variation in the evaluation period. The discrepancies between the program to be validated and the reference results are also compared to the deviations of the 2-K model of VDI 6007-1 to the reference results (see Table 1⁴).

A successful validation requires the maximum values of the mean values and the standard deviation of the evaluation period to be within the applicable limiting values. The limiting values are 1.0 K for the mean value and 1.5 K for the standard deviation of the room air temperature and the operative temperature and 50 W for the mean value and 60 W for the standard deviation of the heating or cooling load.

The overall results for room air temperature and operative temperature of the test examples are continuously within the limiting values. However, with heating and cooling load calculation limiting values are not always respected. Although the mean value of the hourly deviation is within the limiting value of 50 W in all test examples, the standard deviation exceeds the limiting values of 60 W in Test Example 6 with 60.0 W and significantly in Test Example 11 with 65.6 W. This can primarily be referred to the low spatial discretization of building components which aggravates the correct identification of the activated thermal mass. Thereby, this problem can be

⁴The table includes only the maximum values of the evaluation period. The respective higher values are underlined and are those which apply for validation according to VDI 2078.

easily remedied by expanding the number of RC circuits per component layer of the IBPS Building library's component model.

Table 1. Maximum values of validation results

Test Example	Maximum					
	IBPS Building model			VDI 6007-1 model		
	Air temp.	Oper. temp.	Sys. load	Air temp.	Op. temp.	Sys. load
mean value	K	K	W	K	K	W
standard deviation						
1	0,83	0,84		0,27	0,27	
	0,28	0,28		0,16	0,15	
2	0,42	0,42		0,34	0,35	
	0,28	0,28		0,13	0,13	
3	0,74	0,76		0,42	0,42	
	0,50	0,49		1,09	1,05	
4	0,61	0,60		0,64	0,66	
	0,38	0,39		0,74	0,72	
5	0,54	0,57		0,19	0,18	
	0,23	0,23		0,26	0,25	
6		0,09	23,2		0,04	17,1
		0,24	60,0		0,10	40,9
7	0,23	0,17	23,6	0,17	0,18	16,0
	0,24	0,32	56,0	0,18	0,17	28,7
8	0,88	0,53		0,41	0,43	
	0,73	0,33		0,55	0,51	
9	0,90	0,54		0,41	0,42	
	0,74	0,33		0,55	0,53	
10	0,44	0,41		0,16	0,17	
	0,23	0,20		0,26	0,27	
11	0,44	0,44	43,4	0,07	0,07	4,3
	0,47	0,50	65,6	0,09	0,09	12,3
12	0,16	0,17		0,15	0,15	
	0,26	0,27		0,25	0,26	

4 Conclusion and outlook

4.1 Adoption to IEA EBC Annex 60 framework

As a complement of other Modelica libraries for building energy performance simulation, the development of the IBPSBuilding library is focused on building modeling and simulation with Modelica mainly for educational and teaching purposes. The main structure of this model library and its compatibility is emphasized in this work, so that it can be easily adopted into learning work and used by non-programmers.

Further activities shall link the activities on hand with the ongoing work of the international IEA EBC Annex 60 project. The Annex within the International Energy Agency (IEA) in Buildings and Communities programme (EBC) is a project to promote research and development of new computational methods for energy efficient buildings and communities. The focus is to develop and demonstrate next-generation computational tools for buildings and energy systems within buildings based on Modelica and the Functional Mockup Interface standards (Wetter and van Treeck, 2016).

One of the subtasks of the Annex framework focuses on harmonizing and unifying model development in Modelica. As a result of the Annex work, fragmented developments were merged into a common and open set of Modelica base classes for the various libraries such as the AixLib from RWTH Aachen University, see e.g.

(Remmen et al., 2015), (Fuchs et al., 2015) and the references therein, or the Building library from LBNL Berkeley (Wetter et al., 2013) and others. It is therefore the intention to merge the developments on hand which primarily focus on educational issues with the research- and application-oriented models of the Annex.

4.2 Validation

The validation of the IBPSBuilding Library on the basis of VDI Directive 6007-1 with the limit values according to VDI 2078 shows overall good results. Thus demonstrating the resilience of the thermal calculation kernel. By evaluating the results programming errors were detected and remedied, e.g. the incorrect definition of the heat transfer coefficient. In addition, existing components were expanded to include useful additional functions. With the successful validation, the basis for the future development of requested complementary functions has been made, e.g. moisture transport processes, multi-zone modeling, detailed window models, etc..

4.3 FLUIDGLASS

The IBPSBuilding-Package is adopted in the EU-project FLUIDGLASS to support validations of the new façade construction and the integrated control system for fluid and its technical systems. It is a research project coordinated by University of Liechtenstein and funded by European Commission with the seventh Framework Program (EU-FP7 Grant Agreement No. 608509).⁵ In this new façade system in development, two fluidized layers are implemented and proposed to be regulated by an intelligent control system, in order to improve the building energy efficiency by acting as a replacement to cover the functions of shading device, solar thermal collector, heating and cooling elements (Stopper et al., 2013).

In order to accomplish the detailed validation process for the performance of various construction configurations of FLUIDGLASS, an extended version of IBPS-Building is implemented, which includes:

1. consideration of glass coating and its different optical performances according to direction of irradiation and incident angles;
2. consideration of different behavior with light in different wavelengths;
3. free or enforced convection in a gap between glass panes with detailed calculation method and temperature-related material properties;
4. shading of reveal and its geometric interpretation according to incident solar angles;

⁵More information: <http://www.fluidglass.eu/>

5. visual indicators such as illuminance and luminance;
6. improvement of interactions with 3rd-party libraries, MEP-models and modeling optimization

Additionally, a group of modules are created to help building a detailed FLUIDGLASS model in Modelica, which is compatible with the IBPSBuilding-Package and most of the standard Modelica libraries (mainly SIunits and HeatTransfer). The theoretical numerical parts, especially the behavior according to wavelengths, refer to the former research works and their equation system in the equation solver program EES from F-Chart Software (Gstoehl et al., 2011). The FLUIDGLASS components with IBPSBuilding modules are tested in ITI SimulationX, and are also validated with the EES model and sample measurements.

The extended version of the library is planned to be adopted in network simulations on district level in further steps of FLUIDGLASS project. Approaches of co-simulation are in research. It can collaborate with Matlab and Simulink at the moment in practice by using the COM-interface of SimulationX. Furthermore, the library is adopted in Hardware-in-Loop simulations by interacting with LabVIEW to improve the smart controlling of mechanical systems with FLUIDGLASS. The compatibility of the FLUIDGLASS add-on is also going to be tested with the standard Annex60 Buildings Library.

5 Acknowledgements

The authors acknowledge the financial support by the European Commission within the FLUIDGLASS-project (EU-FP7 Grant Agreement No. 608509).

References

- VDI 6020 Blatt 1: Anforderungen an Rechenverfahren zur Gebäude- und Anlagensimulation, 2001-05.
- VDI 2078: Berechnung der Kühllast und Raumtemperaturen von Räumen und Gebäuden (VDI-Kühllastregeln), 2012.
- VDI 6007 Blatt 1: Berechnung des instationären thermischen Verhaltens von Räumen und Gebäuden - Raummodell, 2012-03.
- J Clarke. *Energy Simulation in Building Design*. Taylor & Francis, 2001. ISBN 9780750650823. URL <http://books.google.de/books?id=ksNIQ4kx6UIC>.
- Marcus Fuchs, Ana Constantin, Moritz Lauster, Peter Remmen, Rita Streblov, and Dirk Müller. Structuring the building performance Modelica model library AixLib for open collaborative development. In *Proceedings of 14th IBPSA Conf. Building Simulation 2015*, pages 331–338, Hyderabad, 2015.
- Daniel Gstoehl, Jochen Stopper, Stefan Bertsch, and Dietrich Schwarz. Fluidised glass façade elements for an active energy transmission control. In *World Engineers' Convention*, Geneva, 2011.
- Shan Hua. Entwicklung einer Schnittstelle zwischen IFC-Gebäudemodellen und Modelica. 2014.
- Peter Remmen, Jun Cao, S. Ebertshäuser, J. Frisch, Moritz Lauster, Tobias Maile, J. O'Donnell, S. Pinheiro, J. Rädler, R. Streblov, M. Thorade, R. Wimmer, Dirk Müller, C. Nytsch-Geusen, and Christoph van Treeck. An open framework for integrated BIM-based building performance simulation using Modelica. In *Proceedings of 14th IBPSA Conf. Building Simulation 2015*, pages 379–386, Hyderabad, 2015.
- Lothar Rouvel. Thermische Gebäudesimulation GEBSIMU - Berechnungsverfahren zum instationären thermischen Gebäudeverhalten, 2015. URL <http://www.gebsimu.de>.
- Jochen Stopper, Felix Boeing, and Daniel Gstoehl. Fluid Glass Façade Elements : Influences of dyeable Liquids within the Fluid Glass Façade . In *Energy Forum on Solar Building Skins*, Bressanone, 2013.
- Matthis Thorade, Jörg Rädler, Peter Remmen, Tobias Maile, Reinhard Wimmer, Jun Cao, Moritz Lauster, Christoph Nytsch-Geusen, Dirk Müller, and Christoph van Treeck. An Open Toolchain for Generating Modelica Code from Building Information Models. pages 383–391, sep 2015. doi:10.3384/ecp15118383. URL http://www.ep.liu.se/ecp/{_}article/index.en.aspx?issue=118;article=41.
- René Unger, Torsten Schwan, Beate Mikoleit, Bernard Bäker, Christian Kehrner, and Tobias Rodemann. "Green Building" - Modelling renewable building energy systems and electric mobility concepts using Modelica. *Proceedings of the 9th International Modelica Conference*, pages 897–906, 2012. doi:10.3384/ecp12076897. URL http://www.ep.liu.se/ecp/{_}article/index.en.aspx?issue=76;article=93.
- Christoph van Treeck. *Introduction to Building Performance Modeling and Simulation*. Habilitation thesis, Technische Universität München, 2010.
- Michael Wetter and Christoph van Treeck. IEA EBC Annex 60 Website, 2016. URL <http://www.iea-annex60.org/>. Accessed: 2016-04-04.
- Michael Wetter, Wangda Zuo, Thierry S Noudui, and Xiufeng Pang. Modelica Buildings library. *Journal of Building Performance Simulation*, pages 1–18, mar 2013. ISSN 1940-1493. doi:10.1080/19401493.2013.765506. URL <http://dx.doi.org/10.1080/19401493.2013.765506>.

Advances of Zero Flow Simulation of Air Conditioning Systems using Modelica

Pieter Dermont¹ Dirk Limperich² Johan Windahl¹ Katrin Prölss¹ Carsten Kübler³

¹Modelon AB, Sweden, {pieter.dermont, johan.windahl, katrin.prolss}@modelon.com

²Daimler AG, Germany, {dirk.limperich}@daimler.com

³TWT GmbH, Germany, {carsten.kuebler}@twt-gmbh.de

Abstract

This paper describes recent advances in simulation of zero flow conditions based on work with Daimler using the Air Conditioning Library from Modelon. The Air Conditioning Library is based on the open standard modelling language Modelica. Simulating refrigerant loops at (near) zero flow for large vapor compression cycles is challenging, due to the fast dynamics in the model under those conditions that drastically reduce the step size of the solver. Findings on solver selection and pressure drop correlations are presented. An approach to improve zero flow simulation based on a systematic analysis of heat transfer coefficients is suggested and demonstrated to increase simulation robustness under (near) zero flow conditions.

Keywords: air conditioning systems, zero flow, 1-D fluid modelling, dynamic simulation, numerical issues

1 Introduction

Operating modes with low and zero refrigerant mass flow rates in air-conditioning refrigeration systems have gained significance in the past years. Additional consumers of cooling power, such as batteries, evaporators for multi-zone air cooling or other integrated parts of the thermal management system have become more common.

Naturally, not all branches of the refrigeration cycle are active at all times but may be switched off during given operation modes. In addition, for the purpose of complete vehicle simulation, the entire cycle may be switched off at any time. The refrigeration system model is expected to handle these conditions smoothly and efficiently.

Characteristic for air conditioning cycles is the small total volume, often less than 1 [l]. In addition, the cycle traverses the two-phase dome, and as a consequence a wide and rapidly varying set of fluid properties.

1.1 Thermofluid Modelling and the Air Conditioning Library

Thermofluid models in Modelica are 1-D models which are effectively a string of control volumes and control surfaces (referred to as volume and flow components respectively in the Modelica community). Control volumes account for the conservation of energy and conservation, whereas the control surfaces incorporate the conservation of momentum (Figure 1). (Tummescheit, 2002) explains this concept in detail.

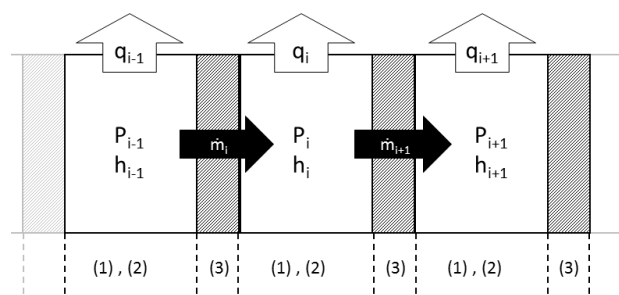


Figure 1. The balance equations. Conservation of mass (1) and energy (2) occurring in the control volume and the conservation of momentum (3) defined across an control surface.

In the Air Conditioning Library the models are discretized using an upwind discretization scheme, intended for (strong) convective flows with suppressed diffusion effects.

Both the above concepts are essential in this article and will be referred to at a later stage.

The Air Conditioning Library contains a set of predefined components focused on air conditioning applications and is developed by Modelon AB. It is currently mostly used by automotive OEM's and suppliers (Tummescheit et al., 2005).

1.2 Test Models

All investigations in the paper are based on a large system model, provided by Daimler. The system uses

R134a refrigerant and contains 2 evaporator branches as depicted in Figure 2. Five *use cases* of the system model were described by Daimler:

1. Normal operation
2. Evaporator branch #1 shutdown
3. Evaporator branch #2 shutdown
4. Compressor shutdown through reduction of positive displacement volume. A complete simulation contains three shut-down - start-up cycles.
5. Compressor shutdown through reduction of shaft speed. Only one shut-down is performed and the system remains at rest for the rest of the simulation.

At the beginning of the investigation *use cases 4-5* experienced significant increase in CPU time right after compressor shut-down, which prevented the simulation to complete within a reasonable time frame.

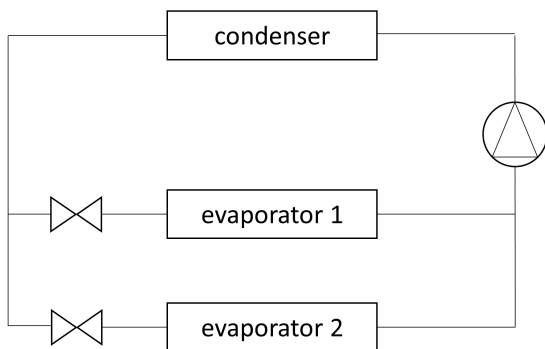


Figure 2. Schematic representation of the refrigeration system

1.3 The Zero Flow Problem

Close to zero flow, some oscillations appear due to the fast dynamics in the model. These dynamics are caused by the mathematical description that at nominal flow rates has reasonable time constants, but at low flow rates become very small. Hence, the solver reduces drastically its step size. A reduced solver step size during a prolonged period results in a dramatic increase in CPU time which makes it infeasible to compute the model within a reasonable time.

The fast dynamics can be observed in the simulation results by oscillations of a given set of variables, in particular the mass flow as is illustrated in Figure 3.

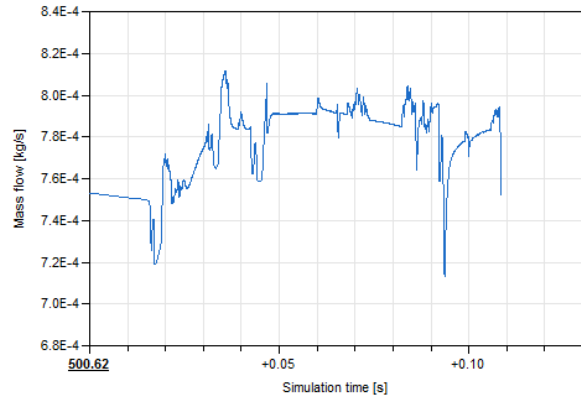


Figure 3. Oscillations in mass flow are a typical symptom of the zero flow problem.

The (simulation time, CPU time) graph is used repeatedly throughout this paper, as an illustration the progress the solver is making. The gradient of the graphs is an indication of the progress of the solver. Figure 4 shows the the graph for *use case 4* before any improvements to the model, where the vertical graph gives a clear indication of the zero-flow problem. 3600 [s] is considered the limit of a reasonable simulation time. All simulations are automatically stopped at 3600[s] - a reasonable computation time for a system with approximately 270 continuous time states and many transients.

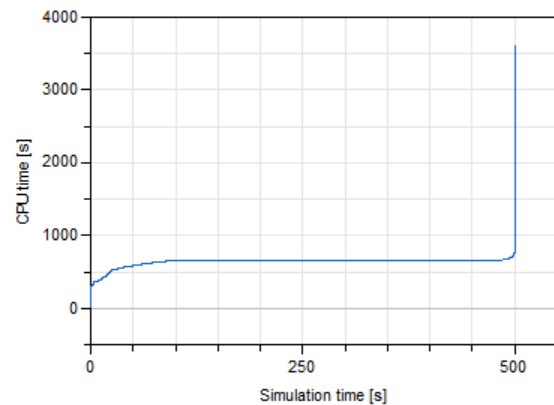


Figure 4. This graphs shows CPU time as a function of simulation time for *use case 4*. The vertical line at $t = 500$ [s] indicates the zero flow problem.

2 Pressure Drop Correlation Regularization

The pressure drop correlation presents in essence the relationship between pressure drop and mass flow, which is approximately a quadratic function (1) :

$$\dot{m} = f(\Delta P) \sim \sqrt{\Delta P} \quad (1)$$

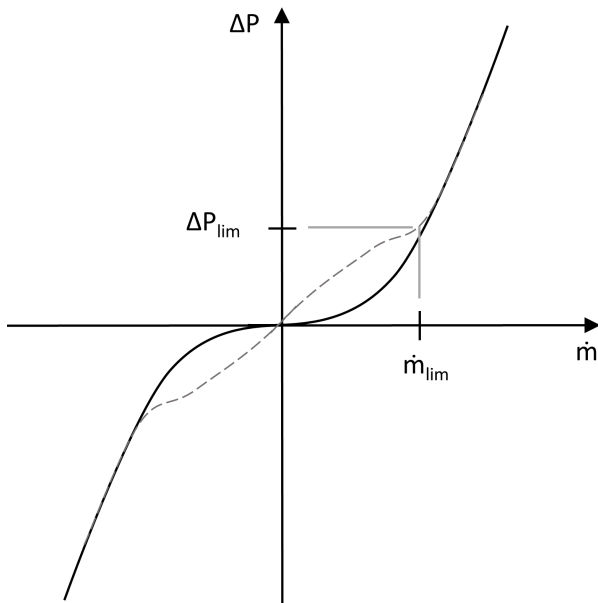


Figure 5. A regularized pressure drop

Note that the derivative of this function approaches zero when approaching a zero mass flow. This will disrupt the search algorithm as very small pressure differences correspond to very large variations in mass flow, causing the solver to jump between different solutions which are beyond solver tolerance. Regularized pressure drop correlations address this issue by increasing the derivative within a user specified region (Figure 5) (Tummescheit, 2002). All pressure drop correlations in the Air Conditioning Library can be regularized with a parameter.

Operation point based pressure drop correlations, as opposed to geometry based pressure drop correlations where the pressure drop is specified by thermodynamic properties and the pipe geometry, allow the user to specify the nominal mass flow and pressure drop to define the pipe's behaviour. Note that depending on how the user sets the nominal mass flow and pressure drop, the correlation will be less or more sensitive to zero flow issues. A small pressure drop for a large mass flow will increase the region in which the derivative of the pressure drop - mass flow function is beyond solver tolerance. It is thus beneficial to lump small pressure drops into one larger pressure loss.

A regularized pressure drop correlation is a necessary requirement for a complex thermo-fluid model to compute under zero flow conditions.

3 Solver selection

Using an appropriate solver is critical to ensure fast convergence of models that experience (near) zero flow operation. Three popular solvers are integrated in the simulation environment Dymola: Dassl, Lsodar, RadauII.

The former two are multi-step algorithms whereas the latter one is a one-step algorithm. One-step algorithms are more efficient at handling stiff problem formulations, which is the case for a (near) zero flow simulation, therefore RadauII is the preferred solver.

A solver tolerance of $10e-6$ was used in Dymola 2016FD01.

4 Heat Transfer To Control Volumes at Zero Flow

As explained in Section 1.1, each control volume has an associated energy balance. For heat exchangers, a large contributor to the overall energy balance of the two phase channel is the heat transfer from or to the secondary channel governed by the heat transfer coefficient correlation.

4.1 Observations

In *use case 4*, oscillations are localised in an evaporator component. One can observe significant oscillations in mass flow and large oscillations in two phase fractions, in the control volumes located within the heat exchanger model. To address the problem, all heat transfer correlations are replaced by a constant ($\alpha = 1500 [W/m^2K]$) to investigate a potential improvement in the simulation progress. The result is remarkable; it does not improve the simulation at zero-flow, on the contrary, it reduces its robustness. The oscillations for this case occur for the control volumes located at the boundaries of the heat exchanger models.

Based on these *observations*, the fast dynamics are attributed to the (large) difference in heat transfer coefficients between adjacent control volumes:

1. Caused by to the difference in heat transfer between one-phase and two-phase flow, as is observed for *use case 4*.
2. Caused by the difference in heat transfer between heat exchangers and adiabatic pipe, as is observed when the heat transfer coefficient is set constant, i.e. without any mass flow dependency.

These hypotheses are strengthened by the result of a simulation run where all heat transfer coefficient correlations are replaced by a function for which the heat transfer coefficient is linearly dependent of the mass flow. The heat transfer coefficient is zero for zero mass flow and takes a given value at nominal flow rate (e.g. $\alpha = 1500 [W/m^2K]$). Consequently, at zero-flow conditions no difference in heat transfer coefficient between adjacent control volumes exists, neither due to different phases nor at the interface heat exchanger - adiabatic pipe. Using this substitute correlation, all *use cases* compute effortlessly to the end.

4.2 Reduce the Difference in Heat Transfer Coefficient Between One-Phase and Two-Phase at Zero-Flow

To accommodate for *observation 1*, heat transfer coefficients of one-phase and two-phase should converge to identical values at zero flow (Figure 6). Key points that were considered are:

- *What value should the heat transfer coefficient converge to for zero flow?* Since all two-phase heat transfer coefficient correlations that are used in this context are not valid at low flow rates but the one-phase heat transfer coefficient correlations is, it is judicious to let the former converge towards the latter. However, its actual value is likely influenced by the droplets and bubbles.
- *At what mass flow should such a transition begin?* Two options are considered to set the transition threshold. In this section, (a fraction) of the nominal mass flow rate is used as limit. In the next section we explore the possibilities of using the Reynolds number instead.

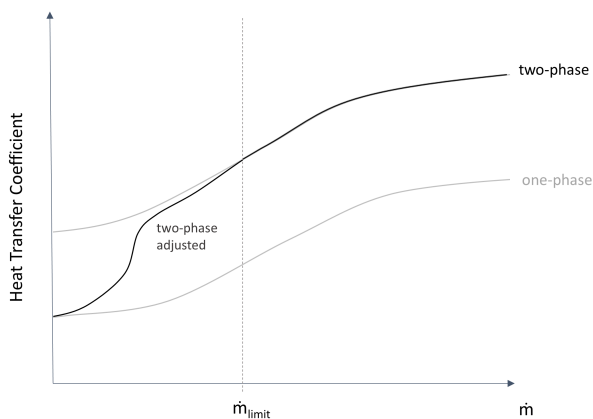


Figure 6. The heat transfer coefficient for one- and two-phase flow in function of the mass flow. To address the zero flow problem, the coefficient is smoothly interpolated between different phases

Note that this effectively not only addresses *observation 1* but also applies to *observation 2*, since the heat transfer coefficient for one-phase flow is generally lower than for two-phase flow. This in turn decreases the difference between the heat transfer coefficients between control volumes located at the border of a heat exchanger and an adiabatic pipe.

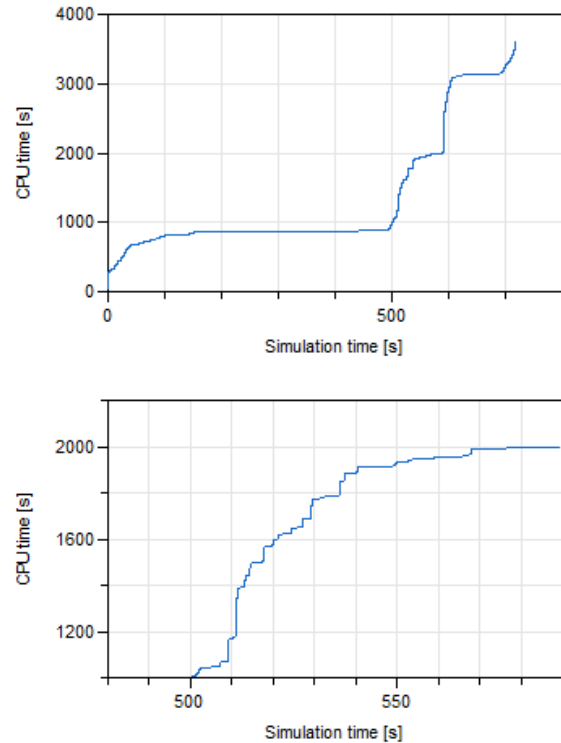


Figure 7. This graphs shows CPU time as a function of simulation time for *use case 4*, with smoothing based on mass flow. At 500 [s] the first compressor shut-down occurs. The simulation is stopped after 3600[s] CPU time, which allows for one complete compressor restart and another shut-down.

Performing this change proved to be highly beneficial to improve zero-flow behaviour. The simulation continues to make progress where it previously got stuck, as is demonstrated in Figure 7. The initial transients take approximately 800 [s] CPU time. Subsequently there is a plateau up to 500 [s] simulation time, where the cycle reaches steady state operation and the solver makes fast progress. The transients for cycle shut-down and restart are located between 500 - 600 [s] simulation time and take approximately 1600 [s] CPU time, which is considered reasonable given the cost of the initial transient. The plateau located at approximately 600-700 [s] simulation time indicates a fast solver progress and coincides with normal refrigerant cycle operation.

4.3 Filtering the Heat Transfer Coefficient

While complete compressor shut-down and restart can be achieved without stalling the simulation, distinct steps in the (simulation time, CPU time) graph can still be observed (see Figure 7). These steps are caused by small variations of the two-phase fraction in a control volume, in its turn causing large variation in heat transfer coefficient between adjacent control volumes. The two-phase fraction indicates what the ratio of one-phase and two-phase within a control volume is. The fast and local variations of heat transfer coefficient contribute to density

variations and thus oscillating flow. A first order filter is applied to the heat transfer coefficient, with time constant τ , as illustrated in Equation (2). Note that F_{user} and y are a calibration factor and the interpolation factor described in Section 4.2 respectively. This introduces an additional state in each control volume, which in principle will increase simulation time.

$$\frac{d\alpha_i}{dt} = \frac{-\alpha_i + F_{user} \left[y_i \alpha_i^{2ph} + (1.0 - y_i) \alpha_i^{1ph} \right]}{\tau} \quad (2)$$

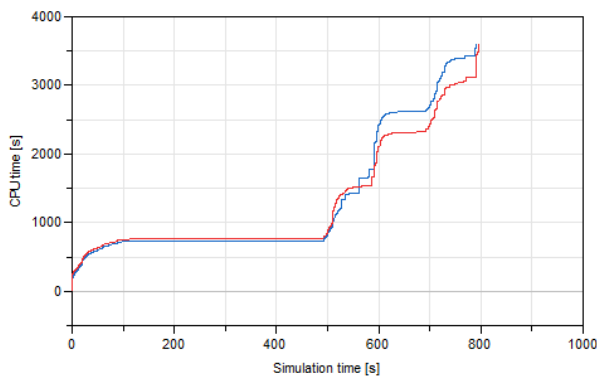


Figure 8. This graphs shows CPU time as a function of simulation time in compressor shut-down (case 4) using mass flow smoothing and a filter in heat transfer correlation. At 500 [s] the first compressor shut-down occurs. In this image, one can clearly see the different consecutive compressor shut-downs, 2-3 are achieved within 3600 [s] simulation time. The blue line corresponds to $\tau = 0.01$, and the red $\tau = 0.001$.

Applying the filter has a recognizable effect on the simulation robustness. In Figure 8, within 3600[s] of CPU time, between 1-2 compressor shut-downs are achieved. The value of the time constant was varied $\tau = 10^{-2} - 10^{-6}$ [1/s]; the concept was beneficial in all cases but no universally optimal value could be determined.

4.4 Reynolds Number Smoothing

In order to achieve a more physically correct result, it was suggested to set a Reynolds number as threshold value for the two-phase heat transfer coefficient to converge towards the one-phase heat transfer coefficient when approaching zero mass flow. The Reynolds number is a more meaningful measurement, independent of the pipe geometry and as it is an indication of the flow regime, a metric often used to describe the validity of heat transfer correlations.

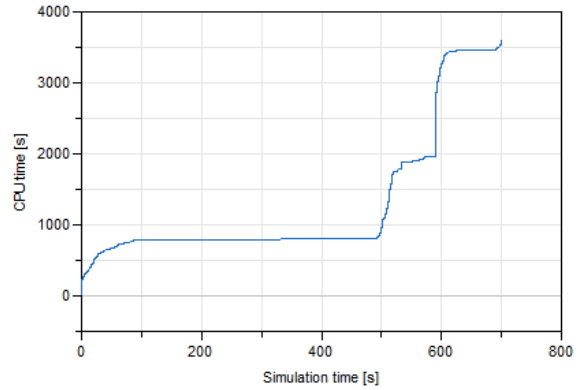


Figure 9. This graphs shows CPU time as a function of simulation time in compressor shut-down (case 1) using Reynolds based smoothing and a filter in heat transfer correlation. At 500 [s] the first compressor shut-down occurs. A complete compressor shut-down and restart cycle is completed within 3600 [s] CPU time.

As default value, $Re = 3000[-]$ is used, located in the transition between laminar ($Re < 2300[-]$) and turbulent flow ($Re > 4000[-]$). Using this threshold the simulation proceeds steadily, however at a slower rate as the previously used nominal mass flow threshold (Figure 9).

4.5 Physical Behaviour at Zero Flow

The suggestions for improved zero flow robustness described above rely on altering the heat transfer coefficients from existing heat transfer coefficient correlations when approaching zero flow. This has consequences on the model behaviour at zero flow but not in the nominal operation range, provided the threshold value of the mass flow or Re is correctly set. If the time constant τ is set sufficiently low, it has no significant impact on the result in nominal operation mode.

Topics for future consideration are:

1. *What is the value of the heat transfer coefficient at zero flow?* All two-phase correlations implemented in the Air Conditioning Library cease to be valid at $Re = 3000[-]$. One-phase correlations however, remain valid to a much lower Reynolds number well into the laminar region. It appears therefore justified to converge towards the one-phase heat transfer coefficient correlation. However, for a more accurate value an advection correlation would need to be included in the overall heat transfer coefficient calculation.
2. *What is the effect of gravity on the system?* The models in the Air Conditioning Library do not take into account the effect of gravity nor introduce a slip factor between phases. At low flow rates, phase separation will occur.

3. *How do these two items above interact?* If we assume that at zero flow, after the system has come to rest, the channels are filled up with either liquid or vapour only, the one-phase correlation is more likely to mimic the physics better. However, boiling is likely to occur in given sections of the cycle.

5 Considerations for Future Work

The discretization scheme currently implemented only accounts for convective flow as previously specified. At zero flow, the energy balances of adjacent control volumes are not linked and the only relationship between adjacent control volumes is the momentum balance. By altering the heat transfer coefficient, the need for linking the energy balances is reduced, as the the energy accumulation/dissipation is more uniform.

Instead, one could imagine creating a thermal link between adjacent control volumes. Diffusion-like discretization schemes can be included in the energy conservation equations of each control volume. This has been done previously however did not yield satisfactory results. While diffusion is beneficial to the zero flow problems, it cannot offset the very large energy accumulation differences between adjacent control volumes that have very different heat transfer coefficients, unless the diffusion factors are unreasonably high.

And alternative method to link the energy balances of the control volumes is to define a discretized (metal) wall along side the thermo-fluid model, through which the control volumes can indirectly interact. The additional states will increase simulation time under nominal conditions but may likely improve model robustness under zero flow conditions.

6 Conclusion

The approach for improved zero flow behaviour of detailed air conditioning system models uses the heat transfer coefficient correlations which prescribe the heat transfer from the refrigerant control volumes to the secondary side. The implementation requires that the value of these coefficients for adjacent control volumes approach each one another for (near) zero flow simulation. In refrigeration simulation, the two natural occurrences of large variations of heat transfer coefficients exist between adiabatic pipes and heat exchangers and between one- and two-phase coefficients within heat exchangers. An approach to numerically smooth these transitions based on nominal mass flow and Reynolds number for the latter suggested. The approach is tested with large system models and demonstrated to increase simulation robustness under (near) zero flow conditions. Systems simulations during which the compressor is repeatedly shut-down previously got stuck but now run to the end.

References

- Hubertus Tummescheit. *Design and implementation of object-oriented model libraries using modelica*. PhD thesis, Lund University, 2002.
- Hubertus Tummescheit, Jonas Eborn, and Katrin Prolss. Airconditioning—a modelica library for dynamic simulation of ac systems. In *4th International Modelica Conference*, 2005.