THE 2ND JAPANESE MODELICA CONFERENCE

May 17-18, 2018, Tokyo

The Second Japanese Modelica Conferences is sponsored by:

Altair DASSAULT SYSTEMES つくる情熱を、支える情熱。 CYBERNET Modelon N2 Neorium Technology ANSYS

NewtonWorks ニュートンワークス株式会社 CLAYTEX SIEMENS Ingenuity for life iSiD IT Solution Innovator

The Second Japanese Modelica Conference is organized by ANSYS and Modelica Association

Proceedings of the 2nd Japanese Modelica Conference Tokyo, Japan, May 17-18, 2018

Editors:
Dr. Yutaka Hirano and Mr. Sameer Kher

Linköping Electronic Conference Proceedings, No. 148
ISSN: 1650-3686, eISSN: 1650-3740
ISBN: 978-91-7685-266-8
DOI: 10.3384/ecp18148

Organized by:
ANSYS Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
USA

in co-operation with:
Modelica Association
c/o PELAB, Linköpings Univ.
SE-581 83 Linköping
Sweden

Conference location:
Bellesalle Jimbocho
Sumitomo Fudosan Chiyoda First Building South
3-2-1 Nishi-Kanda, Chiyoda-ku, Tokyo
Japan

Welcome Message

Following the first successful conference in 2016, the 2nd Japanese Modelica Conference takes place in Tokyo again. 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). We are now proud to present a conference with:

- 3 Keynote speeches
- 37 paper presentations
- An exhibition area featuring 9 exhibitors
- Great venue location in the heart of Tokyo

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 ANSYS Inc. and ANSYS Japan K.K. for taking care of all the practical matters. 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 May 17,

Sameer Kher & Yutaka Hirano

Sameer Kher Yutaka Hirano

Keynotes Speakers

Hilding Elmqvist

Mogram AB, Modelon AB

Modelica - History, State, Needs, Trends and Possibilities

Model based product design requires both intuitive and effective user interface and powerful computing power. The presentation will contain a brief history of Modelica evolution and current status including some applications. Some new needs that are currently not covered will be discussed. New technical possibilities will be introduced, such as web apps for intuitive and effective user interaction and easy access, domain specific language extensions for advanced modeling capabilities and cloud computing for large scale simulation deployment..

Koichi Ohtomi

Meiji University

Using Modelica Effectively in Industrial Research and Development

The main task of product development is to develop a good product at lower cost and to bring it to market in a shorter period. Conventional computer-aided design and computer-aided engineering systems are well established in this regard. However, although upstream design is particularly important in product development to add value and incorporate the required functions, it is difficult to apply conventional systems to the upstream design stage due to the lack of design information at that stage. As a solution to this issue, we are developing the product development environment by applying "Delight Design by System Simulation" methodology, which can be applied to the early design stage of product development including the conceptual and functional design phases. Here Delight Design is equivalent to attractive quality. I introduce the Delight Design technique and the application of "Crane cabin design" and "Hair dryer design". Delight Design is realized by applying "Modelica-based System Simulation" including not only product model but cognitive and human models.

Torsten Blochwitz Andreas Junghanns

ESI ITI QTronic

10 Years of FMI: Where Are We Now, Where Do We Go?

The exchange of simulation models is a key enabler for the distributed model based development and verification process. This was the main motivation to start the MODELISAR project 10 years ago. The main result of this research project funded by the European Union was the tool independent Functional Mock up Interface (FMI) for Model Exchange and Co-Simulation, which is now maintained within a Modelica Association Project. By now more than 100 tools support FMI. The presentation provides an update on the current status, planned new features and the roadmap towards a new release. A few examples demonstrate typical FMI use cases. As an outlook some related ongoing research projects like ACOSAR and EMPHYSIS are presented.

Program Committee

General Chair

Mr. Sameer Kher, ANSYS Inc., USA

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

Prof. Bernhard Bachmann, Fachhochschule Bielefeld, Bielefeld, Germany
Prof. John Baras, University of Maryland, Maryland, USA
Dr. John Batteh, Modelon, Inc., Ann Arbor, USA
Christian Bertsch, Robert Bosch GmbH, Stuttgart, Germany
Volker Beuter, VI-grade GmbH, Marburg, Germany
Torsten Blochwitz, ESI ITI GmbH, Dresden, Germany
Dr. Marco Bonvini, Whisker Labs, Oakland, USA
Dr. Scott Bortoff, Mitsubishi Electric Research Laboratories, Cambridge, USA
Timothy Bourke, INRIA, Paris, France
Daniel Bouskela, Électricité de France, Paris, France
Dr. Daniel Burns, Mitsubishi Electric Research Laboratories, Cambridge, USA
Prof. Francesco Casella, Politecnico di Milano, Milano, Italy
Prof. Hyung Yun Choi, HongIk Univesity, Seoul, Korea
Dr. Johan de Kleer, Xerox PARC, Palo Alto, USA
Mike Dempsey, Claytex, Leamington Spa, United Kingdom
Dr. Hilding Elmqvist, Mogram AB, Lund, Sweden
Dr. Jens Frenkel, ESI ITI GmbH, Dresden, Germany
Prof. Takashi Fukue, Iwate University, Iwate, Japan
Leo Gall, LTX Simulation Gmbh, Munich, Germany
Dr. Rui Gao, Modelon KK, Tokyo, Japan
Khalil Ghorbal, INRIA, Le Chesnay, France
Peter Harman, CAE Tech Limited, Leamington Spa, United Kingdom
Prof. Anton Haumer, OTH Regensburg, Regensburg, Germany
Dr. Dan Henriksson, Dassault Systemes, Lund, Sweden
Dr. Yutaka Hirano, Toyota Motor Corporation, Tokyo, Japan
Prof. Bengt Jacobson, Chalmers University of Technology, Gothenburg, Sweden
Prof. Tommi Karhela, VTT Technical Research Centre , Espoo, Finland
Åke Kinnander, Siemens Industrial Turbomachinery AB, Finspang, Sweden
Dr. Jiri Kofranek, Charles University Prague, Prague, Czech Republic
Satoshi Komori, MAZDA Motor, Hiroshima, Japan
Dr. Christopher Laughman, Mitsubishi Electric Research Laboratories, Cambridge, USA
Dr. Alexandra Mehlhase, Arizona State Univeristy, Tempe, USA
Dr. Lars Mikelsons, Robert Bosch GmbH, Lohr am Main, Germany
Dr. Ramine Nikoukhah, Altair Engineering, Paris, France

Contents

Session 1a: Automotive Applications 1

Riding Comfort Simulation with air ride seat for heavy duty vehicle 12

Assessment of Ride Quality at Lane Change Maneuver Using Virtual Human Driver Model 19

Combustion Engine Mechanism Analyses Using SimulationX 27

Session 1b: Thermal System Applications 1

The CryoLib - Modelling Superconductors with Modelica 33

Robust Modeling of Directed Thermofluid Flows in Complex Networks 39

Liquid Cooling Applications in Twin Builder- Industrial Paper 49

Session 2a: Automotive Applications 2

Development of a Flex-PLI System Model and Investigations of Injury 53

Vehicle Systems Modelling and Analysis (VeSyMA) Platform 61

Dual-Clutch Transmission Model Reduction Function 66

Session 2b: Thermal System Applications 2

FluidDynamics Library for Coarse-Grid CFD-Simulation in Modelica 71

Modeling and Coordinated Control of an Air-Source Heat Pump and Hydronic Radiant Heating System 77

Semiconductor Package Thermal Impedance Extraction for Modelica Thermal Network Simulation Combined with VHDL-AMS model 86

Session 3a: Automotive Applications 3

Modeling of Fuel Cell Hybrid Vehicle in Modelica: Architecture and Drive Cycle Simulation 91

Modelling & Analysis of a Fuel Cell Hybrid Electric Vehicle using Real-World & Standard Driving Conditions 99

Hyundai framework for vehicle dynamics engineering based on Modelica and FMI 109

Session 3b: Thermal System Applications 3

Modelling of Oil Film Bearings 115

Gas Compressor System Simulation using Functional Mockup Interface for Human Machine Interface and Control 122

A New Library For Modeling and Simulation of Pneumatic Systems 127

Session 4a: HILS, Real Time Simulation

The DLR EtherCAT Library - A template based code-generation scheme for accessing real-time hardware from Modelica 130

Modelling and Development of a Pseudo-Hydraulic Power Steering Model for use in Real-Time Applications 137

Session 4b: Electronic Systems Application

Modelling silicon carbide based power electronics in electric vehicles as a study of the implementation of the semiconductor devices using Dymola 147

Session 5a: Model Based Development

AUTOMATED TEST OF CVT CONTROL SOFTWARE, USING FMI AND MODELICA MODELS 155

The Fault library - A new Modelica library allows for the systematic simulation of non-nominal system behavior 161

Application for Optimization of Control Parameters for Multi-body and Hydraulics System by using FMU 169

Session 5b: Mechanical Systems Application 1

Analysis of Lift-Generating Disk Type Blade Wind Power System Using Modelica 174

Modelling of Asymmetric Rotor and Cracked Shaft 180

Composable Modelling for a Hybrid Gearbox 187

Session 6a: FMI, Simulation Technologies 1

Toward the actual model exchange using FMI in practical use cases in Japanese automotive industry 195

Managing Heterogeneous Simulations Using Architecture-Driven Design 204

Simulation of high-index DAEs and ODEs with constraints in FMI 213

Session 6b: Mechanical Systems Application 2, Control

Universal Controllers for Architecture Simulation 223

Mission-Dependent Sequential Simulation for Modeling and Trajectory Visualization of Reusable Launch Vehicles 230

Model predictive allocation control for leg-wheel mobile robot on loose soil considering wheel dynamics 240

Session 7a: FMI, Simulation Technologies 2

Generating FMUs for the Feature-Based Language Bloqqi 245

A Web Architecture for Modeling and Simulation 255

Session 7b: Vendor Session

Multibody simulation and control of kinematic systems with FMI/FMU 261

Deployment process for Modelica-based models 265

Riding Comfort Simulation with Air Ride Seat for Heavy Duty Vehicle

Hyung Yun Choi[1] Whe-Ro Lee[1] Jong-Chan Park[2] Kee-Young Yang[2]

[1]HongIk University, Korea, hychoi@hongik.ac.kr, wirolee@gmail.com
[2]Hyundai Motor Company, Korea, {impactpart, yky}@hyundai.com

Abstract

The design of driver's seat suspension of the commercial truck differs from the one of the passenger car. The vibration and the structural characteristics of the suspension are consequently quite different. Unlike passenger cars, the vibration frequency of commercial truck suspensions is considerably low at 1 to 3 Hz. (Mayton, 2006). The truck seat has an air ride seat. The structural design of air ride seat at heavy-duty vehicle includes serial and parallel combinations of the shock absorber, air spring, and PU foam pad to achieve a good vibration damping. The 1D lumped network solution is an effective design tool with the multi-physical subcomponents. And this also enables a direct coupling into the system modeling of the vehicle body for an optimal calibration of engineering parameters taking the relevant dynamic performance of neighboring parts into account. The mechanical characteristics of each component and their assembly were identified for the 1D modeling. The result of validation and verification of the proposing 1D model of the air ride seat is also introduced.

Keywords: Air spring, Seat comfort, ISO 2631

1 Introduction

The vehicle seat is one of the important components contributing to the crash safety and riding comfort of the occupant. The design of seat frame requires a high strength and stiffness for the better mechanical performance in a crash situation. In a meanwhile, the upholstery of the seat being in a precise contact with a human occupant needs to utilize sufficiently viscoelastic materials with the good tangent of delta such as low-density polyurethane foam for both quasi-static and dynamic supports. Unlike passenger vehicles that mainly drive over the flat road, trucks occasionally undergo a driving over the rugged or even unpaved road. It brings more vigorous oscillation to the truck body. And the cabin has extra pitching motion because of the cabin suspension. It adversely affects the kinematics of occupant at a raised seat position in particular. To suppress the low-frequency oscillation, air suspension is mostly adopted in a seat design for the heavy duty vehicle. The pneumatic pressure actuator which is especially available at the commercial heavy-duty vehicle enables to add the air cell into the seat suspension design for a better isolation of the vibration. The air cell is commonly used also in cabin and chassis suspensions of the heavy-duty vehicle.

Extensive series of lab test has been conducted to characterize the mechanical behavior of major components such as hydraulic damper and pneumatic cylinder, i.e., the air cell unit in the air ride seat. A sine sweep excitation signal was applied to the seat system to measure a transmissibility on a 75kgf dead weight. This paper introduces the 1D modeling process of air ride seat system, the mechanical characteristics, and its validation result.

The verification of the 1D modeling scheme, the other "V" in Verification & Validation process for a riding comfort virtual simulation is also provided with the assessment of the associated riding comfort Index, ISO 2631 "Evaluation of human exposure to whole-body vibration". The ISO2631 provides an indicator for riding comfort and also evaluates riding quality from simulation results (Yoo, 2006). Since this application is a part of feasibility studies and so a simple rigid body mass model was employed as a surrogate human model. The adaptation of the biofidelic human body model is still in progress and remains as our future challenge.

2 System modeling of air ride seat

There are three major parts in an air ride seat design, air spring, shock absorber and PU foam pad to isolate the occupant from the vibration. Air spring and shock absorber aim to counteract jerk type impulse loadings while the PU foam pad is laid for attenuation of the oscillatory pulse in 3-10Hz transferring to the occupant. The 1D modeling process of air ride seat system consists of the selection of appropriate model libraries to represent subsystems and the calibration of necessary model parameters. An air ride seat designed for the driver side of a large truck (curb weight 9400 kg) was selected and SimulationX (ESI ITI) was adopted for the system simulation software

2.1 Seat frame structure

The dynamic mass properties of the frame structure were calculated from the CAD geometry as shown in Figure 1. The underbody in frame structure containing air spring and the shock absorber is designed with X-link such that the seat cushion can move freely in the vertical direction, up and down. Figure 2 shows 5 rotational joints and 2 translational joints in the underbody frame X-link structure.

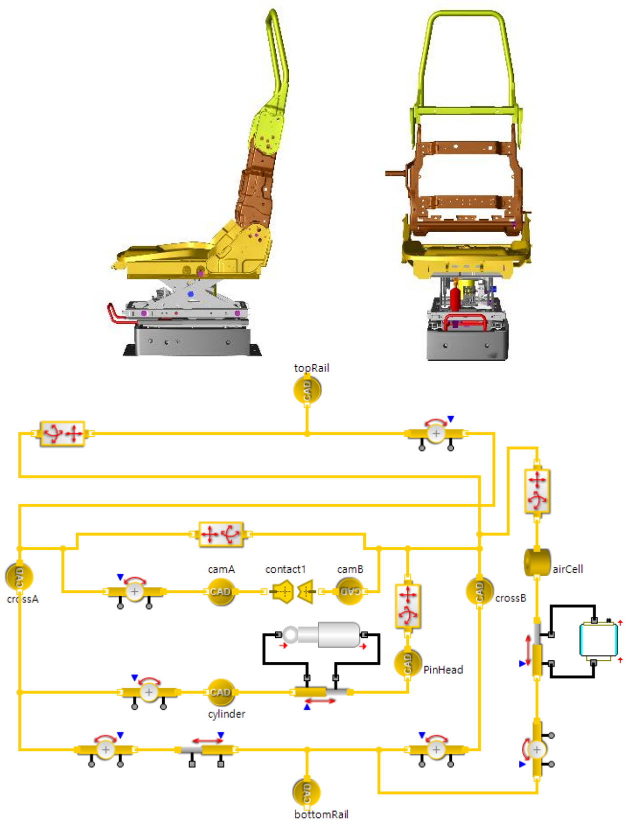


Figure 1. 3D CAD (top) and Diagram (bottom) views of air ride seat frame

2.2 Air spring modeling

The stiffness of the air spring (k_s) is expressed as in the following equation (Liu, 2008)

$$k_s = \frac{dF}{dx} = A_e \frac{d(P-P_a)}{dx} + (P-P_a)\frac{dA_e}{dx}$$

$$F = \int \left(A_e \frac{d(P-P_a)}{dx} + (P-P_a)\frac{dA_e}{dx} \right) dx \quad (1)$$

Where

k_s: Stiffness of air spring

F: Cross section force

x: Stroke

P: Air pressure

Pa: Ambient air pressure

Ae: Initial cross section area

Figure 3 shows the pressure and effective cross section area changes with axial compression of the air spring unit. The abrupt slope increase of force profile

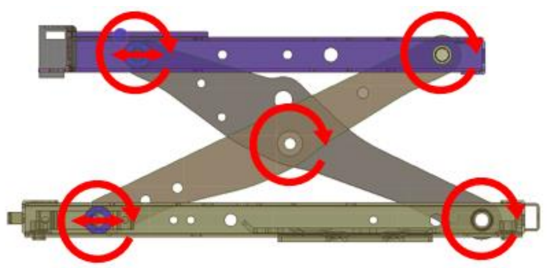


Figure 2. Joints with free DOF in under body frame structure

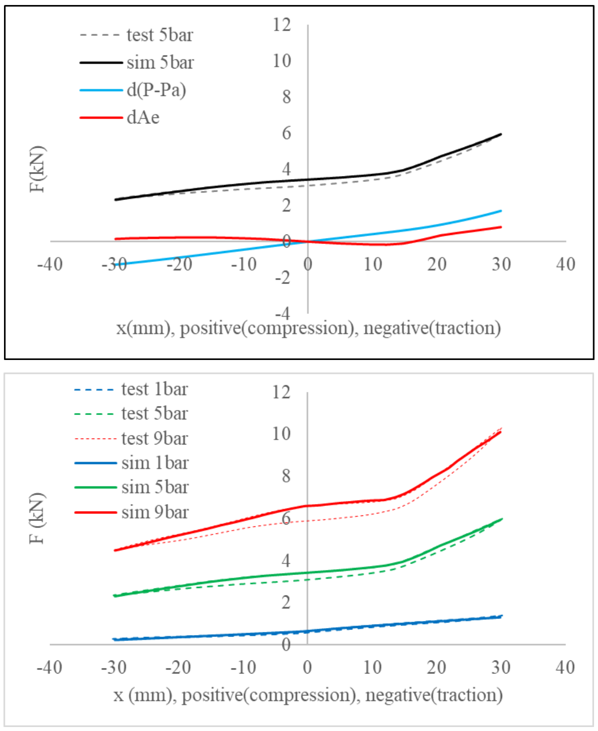


Figure 3. The air spring force is determined by terms of pressure and effective cross-sectional area

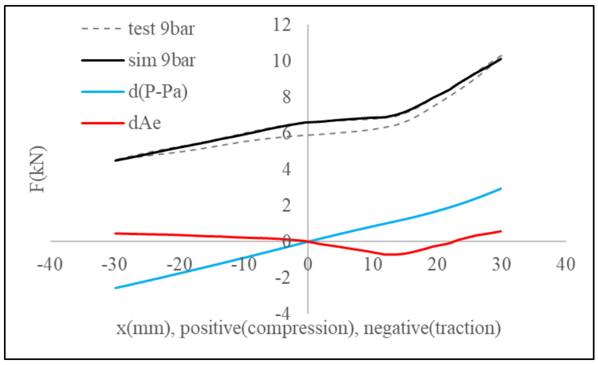


Figure 4. Comparison of force-displacement of air spring unit between test and simulation at various air pressure levels (1-9 bar)

around 10mm compression distance occurs due to the change of cross-section area in the air-cell. This non-linear behavior of the compression stiffness of air spring is also based on the ideal gas equation, the relationship between temperature, pressure, and volume. The temperature of the air in the cell is assumed constant. The compression force is determined from the effective cross-section area and the pressure of the cell at a given stroke.

Air spring shows hysteresis, different force-displacement paths at loading and unloading phases, which becomes more distinctive at higher air pressure level as shown in Figure 4. The air spring element in SimulationX model library was adopted. Those modeling parameters, i.e., characteristics of pressure and volume to axial force relationships were calibrated from measurements at compression in-vitro test of the air-spring unit. Since the temperature of air in the spring assumed constant, the hysteresis is not adequately presented in the system model.

2.3 Pneumatic system modeling

In order to maintain the uniform vertical seat position in a moving vehicle, the air spring has a control logic to inflate and deflate the air cell respectively to raise and lower the seat. The “Auto level valve” (Patent: 10-2003-0043540) consists of with a control cam and three-channel air tube. It has a kinetic mechanism for gauging the height change of the seat cushion and the system subsequently controls the pressure level in the air spring. Using the pneumatic library in SimulationX, this “Auto level valve” system was integrated as a valve logic in the air ride seat model. The coupling chain of the dynamics of seat height, the rotation of control cam, and the opening/closing of the leveling valve are explicitly simulated in the model as shown in Figure 5.

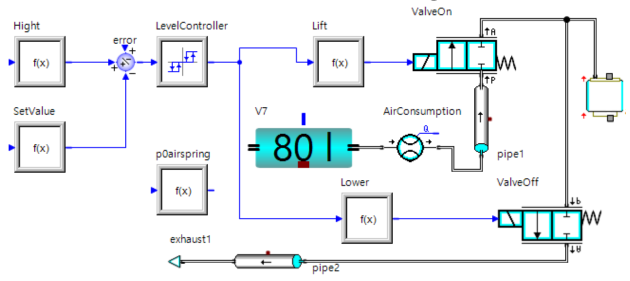


Figure 5. Diagram model of the auto leveling system

2.4 Shock absorber modeling

The operating frequency range of the shock absorber is roughly between 1 to 5 Hz, which is an approximate calculation from the typical stroke distance (20±10mm) and the traveling speed (0.05±0.03m/s) of the seat cushion. The shock absorber of the air ride seat in this study has a design with three-stage, hard, medium, and soft for the damping force control. The measured non-linear behavior of damping force with the compression speed is shown in Figure 6. This damping characteristics are precisely modeled by the shock absorber element in SimulationX model library.

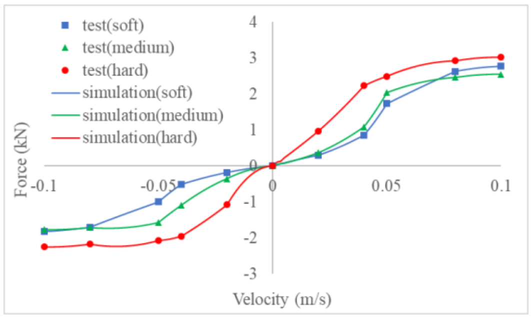


Figure 6. Comparison of axial force-compression speed relationship of the shock absorber between test and simulation

2.5 Seat foam modeling

The seat foam which is commonly made of low density polyurethane shows viscoelastic and hysteresis behaviors, respectively the strain rate dependency and the strain energy dissipation. The standard test method for characterizing these mechanical properties of PU foam are well documented in the literature (VSS manual, 2008). The numerical process of solving the mechanical interaction between seat foam and occupant by using a classical finite element analysis has been fully established and many successful applications have been reported for seat comfort design (Kim, 2007). On the other hand, a more analytic approach is proposed for a lumped network system modeling in this study. Figure 7 shows a stepwise and iterative process: 1) calculate strain and strain rate from the “Disp” element, 2) retrieve stress by making an interpolation Strain rate “curve family”, 3) calculate the reaction force by applying a scale factor that is equivalent to the contact area between occupant and seat foam, 4) iterate 1-3 steps until reaching equilibrium.

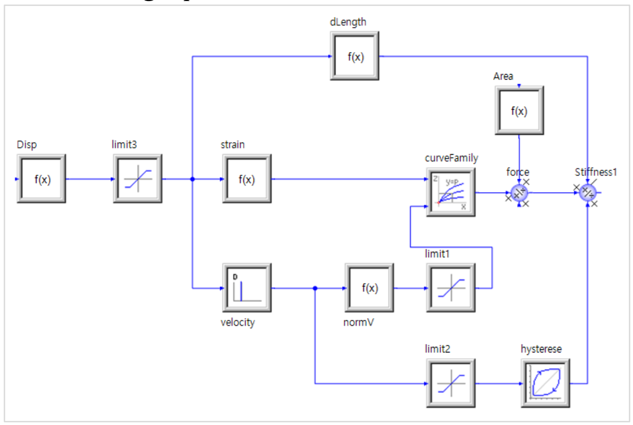


Figure 7. Process for structural analysis of seat foam in a lumped network system modeling

3 Validation of air ride seat system modeling

A two-stage excitation real test was conducted to validate the modeling of the air ride seat. As the first test, the dynamic characteristics of underbody part of the frame, mainly the reaction force produced by two parts in parallel connection, i.e., the air spring and the shock absorber, was identified by applying a harmonic excitation at the mounting floor. It was followed by the additional subcomponent serially attached on the top of underbody, the viscoelastic PU foam layer. Both excitation tests were conducted with a 75kg dead weight placed on the top as representing a body weight of the occupant. The air inlet pressure was maintained at 9bar (900kPa) as same as the target large trailer in operation. The assembled 1D subcomponent models in the underbody frame introduced at sections 2.1 to 2.4 were validated using the 1st stage test result. The 2nd stage test was subsequently utilized for the model validation of the cushion foam in section 2.5.

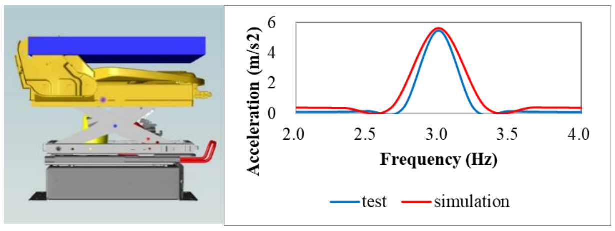


Figure 8. Harmonic excitation of underbody frame model with 75kg dead weight

Figure 8 shows the underbody frame model with the 75kg dead weight on the top, which was used for the simulation of sinusoidal harmonic excitation with 15mm amplitude and 3Hz speed. In the test, the dynamic response at the dead weight was measured by accelerometer sensor (Shimmer, Shimmer3 IMU Unit) with 256Hz sampling rate. The Fast Fourier Transformation (FFT) signal shows a bell shape acceleration profile of the dead weight with 5.6m/s^2 at 3Hz as also shown in Figure 8. The simulation result showed a bit wider bell shape than at the test but well correlated with the peak acceleration, 5.7 m/s^2 (+1.7%).

The modeling of seat foam was also validated by adding the subcomponent model described in section 2.5 on the top of the frame model. In the 2nd stage test, a sine sweep excitation from 1Hz to 20Hz, the same 75kg dead weight was used but with the buttock shape of HPM (H-Point Machine). This buttock shape weight was allowed to move only along the vertical direction. The individual amount of contributions for the vibration damping by air suspension in the frame and the viscoelastic PU foam pad was separately analyzed. The transfer function from the air suspension which is the FFT profile of acceleration ratio between the floor and the top of the frame, shown in Figure 9 (a) indicated that the resonant frequency was around 2-3Hz and overall transfer function was below 1.0. The other transfer function that was from the PU foam pad, the acceleration ratio between the top surface of the frame and buttock shape weight, shown in Figure 9 (b) revealed that its resonant frequency was around 6-7Hz and the transfer function was above 1.0 up to 10Hz. The combined transfer function of the air ride seat system, the acceleration ratio between the floor and the buttock shape weight, is shown in Figure 9 (c). The summed total resonant frequency of the buttock shape weight against the floor vibration was 2Hz and the transfer function rapidly declined after 8Hz, which was mainly due to the relatively low transfer function induced by the air suspension. Simulation result also in Figure 9 exhibits the same trend found at the test. However, there were some difference between the actual test and the system model also. There are nonlinear factors in the actual seats of commercial trucks, such as friction and tolerances, but they have not yet been covered in this study. They will consider in the next study. It is noteworthy that this kind of vibration damping performance at the air ride seat is quite distinctive in comparison with the one at the sedan seat. (Amann, 2008)

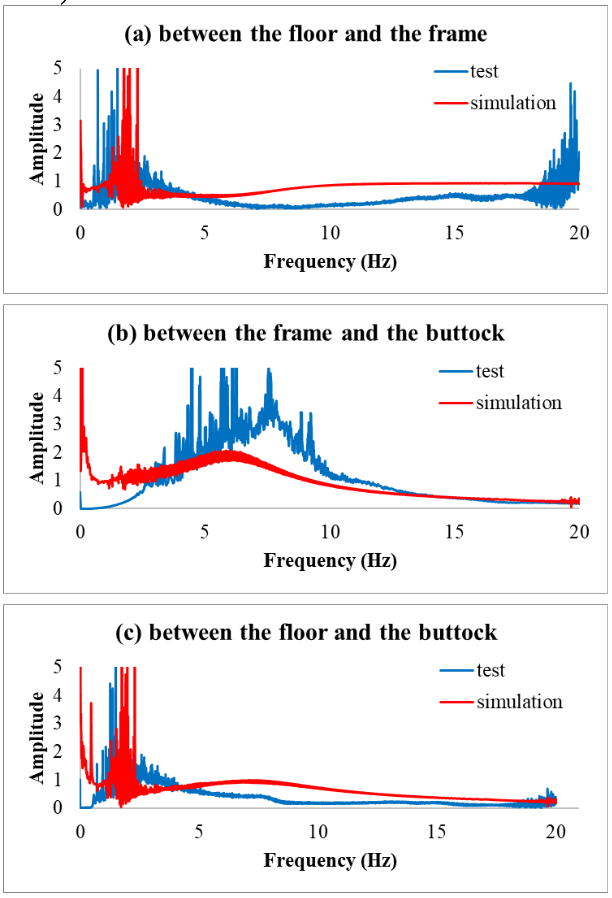


Figure 9. Harmonic excitation of air ride seat system with 75kg buttock shape weight

4 Verification of air ride seat system modeling

In order to verify the practical use of the 1D modeling of air ride seat for the assessment of its comfort design, the international standard ISO 2631 is adopted, which determines the effect of vibrations on the human body (ISO 2631 part1, 1985). ISO 2631 contains methods for evaluating average comfort which has been widely used by researchers and vehicle designers. The quantified performance measures of ISO 2631 are based on frequency weighted root mean square, RMS, computations of acceleration data in accordance with the following equation

$$a_w = \left[\frac{1}{T}\int_0^T a_w^2(t)dt\right]^{\frac{1}{2}} \quad (2)$$

Where

$a_w(t)$ is the weighted acceleration (translational or rotational) as a function of time (time history), in meters per second squared (m/s^2) or radians per second squared (rad/s^2), respectively;

T is the duration of the measurement, in seconds.

The crest factor, defined as the modulus of the ratio of the maximum instantaneous peak value of the frequency-weighted acceleration signal to its RMS value, may be used to investigate if the incoming vibration contains occasional shocks or not in relation to its effects on human occupant. For vibration with crest factors below or equal to 9, the basic evaluation method for the comfort score is normally sufficient. The ISO 2631 provides weighting function table. All necessary signal filters for the calculation of crest factor and ISO 2631 comfort index from the seat vibration that is the transient output from the 1D lumped network solver are implemented in to a total weighting function by using the "transfer function element" of Simulation X as shown in Figure 10.

Total weighting function in ISO 2631

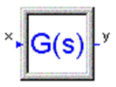
$H(p) = H_h(p) \cdot H_l(p) \cdot H_t(p) \cdot H_s(p)$

$H_h(p)$: *High pass*

$H_l(p)$: *Low pass*

$H_t(p)$: *Acceleration-velocity transition*

$H_s(p)$: *Upward step*



$$G(s) = \frac{a_0 + a_1 \cdot s + a_2 \cdot s^2}{b_0 + b_1 \cdot s + b_2 \cdot s^2}$$

e.g. Low pass

$$|H_h(p)| = \left|\frac{1}{1 + \frac{\sqrt{2}\omega_1}{p} + \left(\frac{\omega_1}{p}\right)^2}\right|$$

$$G(s) = \frac{0 + 0 \cdot s + 1 \cdot s^2}{\omega_1^2 + \sqrt{2}\omega_1 \cdot s + 1 \cdot s^2}$$

Figure 10. Transfer function for ISO2631

Two kinds of measured floor acceleration time histories are applied for an exploratory operation of the 1D modeling of the air ride seat for the ISO 2631.

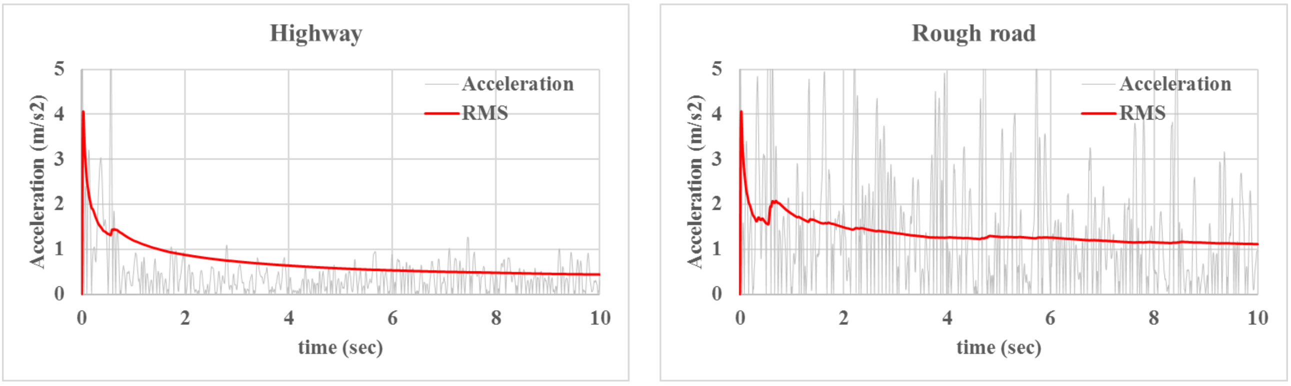


Figure 11. Room mean square acceleration of the buttock mass; the Highway(left) and the Rough road(right) condition

Table 1. ISO 2631 Comfort index

Range	Index	Highway	Rough road
Less than 0.315 m/s^2	not uncomfortable	0.4	1.1
0.315 m/s^2 to 0.63 m/s^2	a little uncomfortable		
0.5 m/s^2 to 1 m/s^2	fairly uncomfortable		
0.8 m/s^2 to 1.6 m/s^2	uncomfortable		
1.25 m/s^2 to 2.5 m/s^2	very uncomfortable		
Greater than 2 m/s^2	extremely uncomfortable		

Test driving was done with the target large truck at high speed proving track (hereafter called highway) and rough pavement test road (hereafter called rough road), respectively at 100kph and 30kph driving speeds. The 10-second clip in the middle of floor accelerations measured at the driver side seat mounting point was used as an excitation input of the 1D air ride seat model. The same configuration of the buttock shape weight in the Figure 9 gauged the vibration response of the air ride seat model. The crest factors for the highway and the rough road were respectively 1.6 and 3.5, well below 9.0 and thus applicable to the assessment of comfort index. The computed RMS of frequency weighted filtered accelerations, i.e., the comfort score of highway driving was 0.44 m/s^2 and 1.11 m/s^2 for the rough road driving. They respectively belong to "a little uncomfortable" and "uncomfortable" categories. (see the Figure 11 and Table 1)

The effects of three-stage damping forces at the shock absorber, hard, medium and soft, on the ISO 2631 comfort index for both driving over the highway and rough road are evaluated. As the damping force increases to the hard stage from the medium, the comfort score was improved by 3.6% and 15% respectively at rough road driving (1.12->1.07) and high-speed PG driving (0.44->0.37). However, the decrease of damping force to the soft stage results in raises of RMS by 1.0% and 6.8%. (see the Figure 12) As discussed in Section 2.3, the amount of damping force change between hard and medium is more than the one between medium and soft stages and this brings more substantial reduction of RMS at hard stage which is recommendable in the aspect of transfer function of the buttock. This analysis does not include interactions with seat back and foot supports but just limited to between seat cushion and buttock.

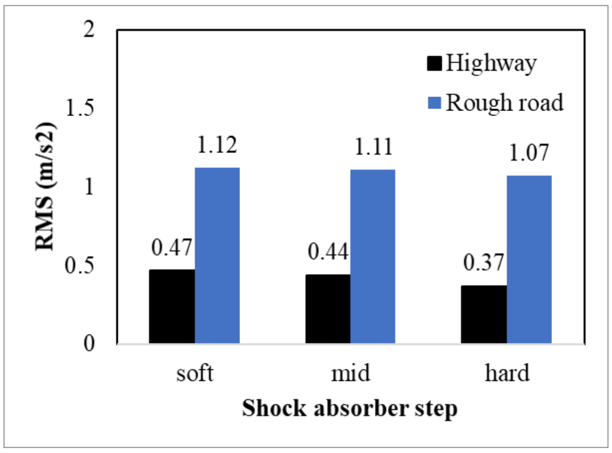


Figure 12. ISO 2631 comfort index between the highway and the rough road along the shock absorber

5 Conclusions

The air ride seat in a heavy-duty vehicle has multi-physical subcomponents such as air spring, shock absorber, and viscoelastic PU foam pad. The 1D lumped network modeling can be effectively used at the front-loading phase to optimize its engineering design parameters taking the relevant dynamic performance of neighboring parts into account. The mechanical characteristics of each major component in-vitro was experimentally identified and the corresponding 1D model was accordingly validated. The assembly of multi-physical subcomponents into the air ride seat was also validated against the harmonic excitation with a good correlation of the transfer function. In order to verify the practical application of the 1D air ride seat model for a ride comfort assessment, the standard ISO 2631 process was carried out for evaluation of each driving over the highway and the rough road. The effects of damping levels of shock absorber on the comfort score were also investigated.

The current validation and verification of 1D air ride model have a limitation such that a simple buttock shape weight is used as the occupant surrogate model. So the mechanical interactions are restricted to the seat cushion but no seat back and foot support are included. A virtual human body model with anatomically detailed joint articulation will be adopted in a subsequent study. Then, the scope of this ride comfort analysis will further expand to more precise interactions of the seat with the occupant model via deformable upholstery that may need a 3D mesh based solution, the finite element analysis. Therefore, the co-simulation of Functional Mockup Interface (FMI), in which the master and slave solutions are respectively 1D lumped network and 3D finite element analyses is also in our future plan for predicting a complete occupant kinematics for riding comfort simulation.

References

Mayton AG, DuCarme JP, Jobes CC, Matty TJ. ASME. *Laboratory Investigation of Seat Suspension Performance During Vibration Testing.*, 2006. doi:10.1115/IMECE2006-14146

Yoo W.S., Park S.J., Park D.W., Kim Min-Seok, Lim O.K., Jeong W.B... International Journal of Automotive Technology. *Comparison of ride comforts via experiment and computer simulation.* 7. 309-314., 2006

Christian Amann, Andre Huschenbeth, Raphael Zenk, Nicole Montmayeur, Christian Marca, Carole Michel. Digital Human Modeling Symposium. *Virtual Assessment of Occupied Seat Vibration Transmissibility*, 2008. doi: 10.4271/2008-01-1861

Hao Liu, Jaecheon Lee, KSAE. *An Experimental Investigation on the Characteristics of automotive air spring*, pp. 4-8, 2008

H.Y. Choi, K. Kim, C. Kim, S. Sah, S. Kim, S. Hwang, K. Lee, J. Pyun, N. Montmayeur, I. Lee. Digital Human Modeling Symposium. *Challenge of Lumbar Support Design Using Human Body Models*, 2008. doi: 10.4271/2008-01-1860

International Standard Organization ISO 2631/1. *Evaluation of Human Exposure to Whole-body Vibration*, Part1: General Requirements, 1985

N. Montmayeur, C. Marca, E. Haug, H.Y. Choi, S. Sah. Digital Human Modeling Symposium. *Numerical and Experimental Analyses of Seating and Riding Comfort*, 2004

S. Kim, S. Hwang, K. Lee, J. Pyun, H.Y. Choi, K. Kim, S. Sah, N. Montmayeur. Digital Human Modeling Symposium. *New Anthropometry of Human Body Models for Riding Comfort Simulation*, 2007. doi:10.4271/2007-01-2457

VSS manual, ESI GROUP SOLVER REFERENCE MANUAL. *GETTING STARTED WITH PAM-COMFORT*, page 18, 2010

Whe-Ro Lee, Manyong Han, Hyung Yun Choi, Jungtae Yang, Inhyeok Lee, Kee Young Yang, Jong-Chan Park. PAM User Conference Asia. *Air Ride Seat for Heavy Duty Vehicle.* 2017

Assessment of Ride Quality at Lane Change Maneuver Using Virtual Human Driver Model

Manyong Han[1] Hyung Yun Choi[2] Akinari Hirao[3] Stefan Kirschbichler[4]

[1]Department of Mechanical Engineering, HongIk University, Korea, myhan@mail.hongik.ac.kr

[2]Mechanical System Design Engineering Department, HongIk University, Korea, hychoi@hongik.ac.kr

[3]Nissan Motor Co., Ltd., Japan, a-hirao@mail.nissan.co.jp

[4]VIRTUAL VEHICLE, Austria, Stefan.Kirschbichler@v2c2.at

Abstract

The occupant kinematics occurring at a lane change maneuver affects the local ride quality. The precise analysis of the occupant kinematics requires a comprehensive understanding of the physiologic response to human body as well as the vehicle kinematics. A series of vehicle-based tests also confirmed that the alertness level of vehicle occupants is one of the important biomechanical elements. Therefore, it is necessary to have a virtual human body model (HBM), an occupant surrogate at CAE design process, with active muscle forces to represent the reflexive response of human beings. An active human body model that produces joint torques with PID closed-loop control as mimicking a bracing action to keep the sitting posture against the external jerk has been developed. In this study, this active human body model is validated against the subject test by simulating the similar occupant kinematics at a single lane change maneuvers. To further verify the use of active HBM as a design tool, an artificial lane change maneuver with a reduced lateral jerk is fabricated and good matching occupant kinematics are predicted.

Keywords: Occupant kinematics, Ride quality, Lane change, Active human body model

1 Introduction

The demand of objective and quantitative tools for an assessment of the ride quality design of vehicle occupant is quite high. Since the principal subject of vehicle driving tends to migrate from human to machine as the autonomous vehicles prevail more in the fleet, the ride quality of passengers affected by the vehicle maneuvers moves from the driving skill of individual human to the engineering car design sector. Short-range maneuvers such as a lane change cause a lateral force on the vehicle that perturbs sitting occupants and arouse them to brace to maintain the equilibrium posture. This unavoidable muscular activity of vehicle occupants lead to the discomfort of riding, the poor ride quality.

In physics, jerk is the rate of change of acceleration, which is the derivative of acceleration with respect to time. The jerk has physiological effects on occupant's perception in a moving vehicle. Occupant tends to brace as maintaining a balanced sitting posture against external perturbations, the vehicle movements. When the changes of the vehicle movements are abrupt, the acting forces on the occupant become fast as well. Then, there would not be enough time for establishing a control loop to achieve this equilibrium by adjusting agonistic and antagonistic muscle tensions, i.e., beyond the physiological bounds. Therefore an excessive acceleration (force) and jerk (rate of force) in vehicle movement may result in an uncomfortable ride.

The ride quality of a vehicle maneuvering such as lane change can be assessed common by two physical quantities, the maximum acceleration and the maximum jerk which are effective to predict the local discomforts of occupants. Local discomfort typically arises when the acceleration reaches 2.45 m/s^2 (Schofield, 2001) or when the jerk reaches 0.9 m/s^3 (AASHTO, 2001). The average ride quality can be determined by using the standard ISO 2631 or the UIC ride quality note (Lauriks, 2003) which evaluates the acceleration data over time.

Takahashi et al (Takahashi, 2013) presented "Preview G-Vectoring Control" (PGVC), which is based on the "G-Vectoring Control" (GVC) scheme by MAZDA. In GVC, the longitudinal-acceleration control algorithm is based on the actual lateral jerk. PGVC decelerates a vehicle before it enters a curve, and is based on a new longitudinal-acceleration control algorithm which uses predicted and actual lateral jerk. They concluded that it was confirmed that PGVC could reduce the jerk without an excessive slow-down. And this could be a useful driver assistance system to reduce the driver's braking task.

The virtual human body model (HBM) has extensive applications in crashworthiness design of the vehicle to predict kinematics and injury risks of both occupant and pedestrian (www.GHBMC.com; EuroNCAP TB 024, 2017). Considering the upcoming trends in the automotive industry, namely active safety and autonomous driving, the rising demand of active human body model was brought up by Kleinbach and Fehr (Kleinbach, 2017). As the focus of vehicle safety is shifting towards scenarios with lower acceleration levels, traditional surrogate models including HBM developed for high acceleration levels no longer suffice. Authors presented an efficient way to model muscle forces of vehicle occupants as they maintain the postural stability during the ride. The active joint torque controlled by a proportional integral derivative (PID) closed loop was introduced at the elbow joint to simulate the voluntary and reflexive response of the human subjects. (Han, 2016) We further extended this active human body modeling scheme to the whole body to simulate vibration response to the uncoupled translational excitations (Choi, 2017).

Yamada et al (Yamada, 2016) used THUMS v5 model for simulating the occupant kinematics in the single lane change maneuvers. When the skeletal muscles were activated at the THUMS model, a less lateral displacement was found than in the case without muscle activation. That correlated well with the findings from their reference test (Kirschbichler, 2014).

In this study, a part of the outcomes from the vehicle-based tests at OM4IS (Occupant Model for Integrated Safety) project (Kirschbichler, 2014) is utilized to verify and validate the active HBM. There were two vehicle maneuvers, emergency braking, and single lane change. Also two kinds of seat conditions, series A: wood seat and lap belt and series B: cushioned seat and lap and shoulder belt. Among two test maneuvers, the single lane change (c.f., the other one was emergency braking) is selected for this study. The single lane change was conducted at 50 km/h vehicle speed for 2 seconds period of time that produced a 10 m/s^2 peak lateral acceleration. This steering action was rather a swift movement like an evasive lane change to avoid a collision into the obstacle. The series A seat condition was selected because it provides simpler boundary condition at model simulation than series B.

2 Analysis of the test data

To investigate the human response to low-load situations prevalent in the pre-crash phase, a series of lane change tests had been conducted at OM4IS project (Kirschbichler, 2014). Twenty-one male test subjects (age: 33.4±8.8 y, mass: 78.5±6.3 kg, height: 179.2±4.6 cm, sitting height: 91.0±2.0 cm) were recruited as front seat passengers. They were requested to keep three contrasting awareness states. An unaware, an anticipated and an informed condition, respectively at each maneuver, which have different levels of active muscle contribution to the body kinematics during the lane change maneuvers. Table 1 shows age and body size of the test subjects actually participated the test. The number of subjects was reduced from 4 to 5 for the data analysis due to either missing or noisy signals.

Table 1. Age and Body Size of Test Subjects

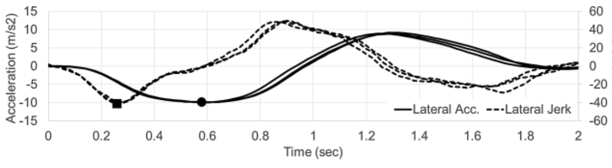
Awareness	*Number of test subjects*	*Age (years old)*	*Weight (kg)*	*Height (cm)*
Unawared	22 (18*)	32.2±8.6	77.4±6.7	179.4±4.3
Anticipated	20 (16*)	31.5±8.3	76.9±6.7	179.2±4.2
Informed	21 (17*)	32.2±8.7	76.9±6.5	179.1±4.1

*: number of test subjects used for data analysis

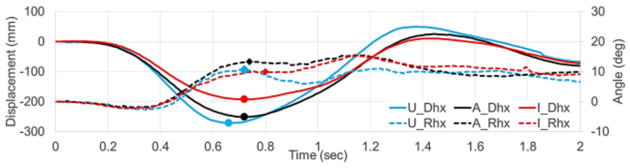
Table 2. Mean (±σ*) peak and maximum values at lane change maneuvers

Awareness	*Vehicle peak*		*Head maximum*		*Torso maximum*	
	lateral acc. (m/s^2), A_y	*lateral jerk (m/s^3), J_y*	*lateral disp. (mm), D_{hy}*	*rolling angle (degree), R_{hx}*	*lateral disp. (mm), D_{ty}*	*rolling angle (degree) , R_{tx}*
Unaware	9.9(±0.2)	41.2(±3.7)	271.7(±96.6)	10.6(±10.7)	216.1(±63.8)	14.9(±7.3)
Anticipated	9.9(±0.4)	39.6(±5.1)	251.3(±107.0)	13.3(±11.5)	189.6(±67.7)	14.5(±7.0)
Informed	9.9(±0.4)	40.2(±4.5)	193.1(±84.6)	9.9(±8.5)	147.0(±58.5)	9.2(±4.8)

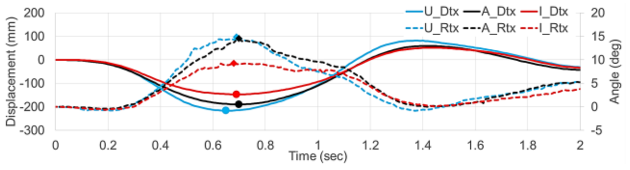
The averaged vehicle and occupant kinematics are shown in Figure 1. The first peak value of each curve involves the most meaningful physical quantity of the cause-and-effect relationship between the vehicle and occupant kinematics. Table 2 lists those mean peak vehicle and maximum head and torso values at lane change maneuvers. The more comprehensive kinematic data with standard deviations about all test subjects are presented in the references (Kirschbichler, 2014).



a) Lateral Acceleration (A_y) and Jerk (J_y) of Vehicle



b) Lateral head displacement (D_{hy}) and rolling angle (R_{hx}) of test subject

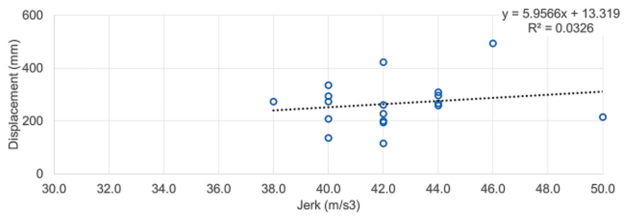


c) Lateral torso displacement (D_{ty}) and rolling angle (R_{tx}) of test subject

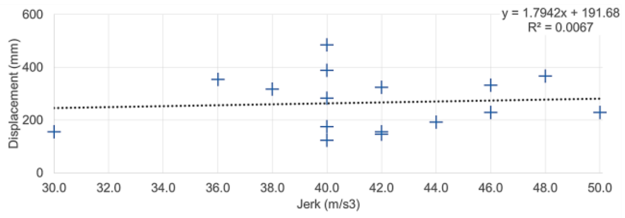
Figure 1. Mean profiles of vehicle and occupant kinematics at lane change maneuvers. Dot on each curve indicates peak (or maximum) value in Table 2 (U: Unaware, A: Anticipated, I: Informed)

2.1 Statistical Analysis

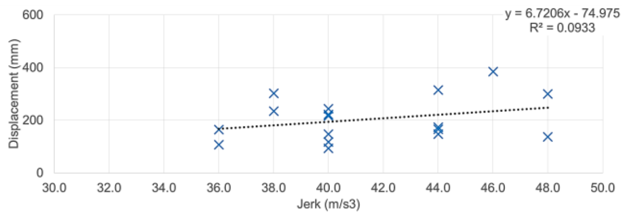
Data of each individual test subject are statistically analyzed. Even though the protocol of the lane change test was designed to provide the same vehicle motions for each lane change maneuver to identify and quantify the occupant kinematics with three awareness conditions, the peak vehicle lateral jerk has substantial variation. It reaches from 50.0m/s^3 to 30.0m/s^3 (mean: 40.0m/s^3 and σ: 4.4 m/s^3) while very small deviation of the peak lateral acceleration from 10.2m/s^2 to 9.4m/s^2 (mean: 9.9m/s^2 and σ: 0.3 m/s^2). The effect of peak lateral jerk on the maximum head lateral displacement was statistically analyzed but no significant correlation was found (p=0.214, R2=0.031). (See Figure 2)



a) Unaware



b) Anticipated



c) Informed

Figure 2. Relation between vehicle peak lateral jerk and maximum head lateral displacement of test subject

3 Virtual Human Body Model

The whole body human model developed for the ride comfort study (Choi, 2017) is adopted as a virtual occupant surrogate to simulate the occupant kinematics at lane change maneuver. This human body model has a multi-body system consisting of 15 body regions and 14 articulated joints. The skin morphology presents an average North American 50th %tile male population (Kim, 2007; SizeUSA survey, 2000-03) which has 86kg and 178cm of body size. As discussed in detail in section 2, analysis of the test data, marker trajectories were used to estimate the position and orientation, the upper body kinematics (head and torso). For improving the computational efficiency, the modeling of upper and lower limbs are switched to non-deformable articulations and rigidly mounted to upper and lower trunks. The total number of body segments then become five, head, neck, upper trunk, center trunk, and lower trunk and four joints in the trunk as listed in Table 3. The hand position is also changed from the steering wheel to laps representing a passenger posture as shown in Figure 3.

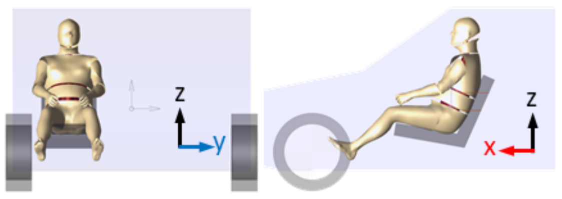


Figure 3. Human body model in multi-body system

Table 3. Four Articulated Joints with Their Anatomical Positions

#	*Articulated joint*	*DOF*	*Anatomical position*
1	Head-neck	3	OC joint
2	Neck-Upper trunk	3	C7/T1
3	Upper-Center trunk	3	T12/L1
4	Center-Lower trunk	3	L5/S1

Voluntary and reflexive muscle activation of a vehicle occupant is modeled by active joint elements at each anatomical joint position. There are two basic elements at each joint, i.e., the passive kinematic joint element and the torque actuator. Contrastively to voluntary activation of individual muscles, i.e., the pre-tension and consequent stiffening of the articulated joint represented by the passive kinematic joint element, a vestibular reflexive muscle activation for the posture stabilization is modeled by the introduction of active torques with PID closed-control (Choi, 2017). The active torque, the control signal, is a sum of proportional, integral, and derivative terms between the current and the reference (initial) joint angles. The gain values at the PID control determine the rates of torque generation. Faster torque generation with larger gain values stands for the pre-recognition of the upcoming external perturbation. Each term at PID control can be adjusted to calibrate the rate of muscle recruitment for fine control of the reflexive response of the human occupant. Authors of this paper showed a successful application of the proposing active joint modeling with the elbow reacting to the jerk loading (Choi, 2016; Han, 2016).

4 Simulation of Single Lane Change

4.1 Validation of human body model against the single lane change test data.

The forward (V_x), lateral (V_y), and yaw angular (W_z) velocities of the test vehicle measured at single lane change maneuvers in Figure 4 was used as the kinematic input condition for the lane change simulation.

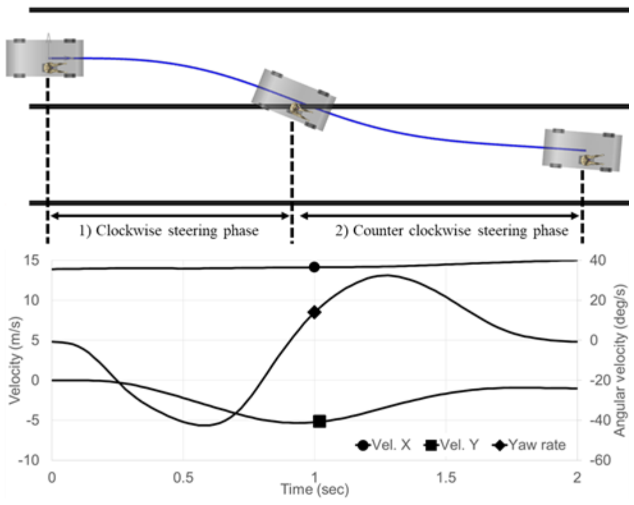


Figure 4. Kinematic input condition of single lane change maneuver for the simulation: 1) clockwise steering phase (~0.95s), 2) counter clockwise steering phase (0.95~2.0s)

Considering the lap belt restraining with a probable slackening, the lower trunk of HBM is tied to the seat cushion by an elastic spring element, which allows a small amount (< 20mm) of buttock sliding over the seat cushion. A sliding contact, the polygon-to-polygon contact type at SimulationX (ESI ITI) is defined between the seat back and occupant with 0.1 frictional coefficients.

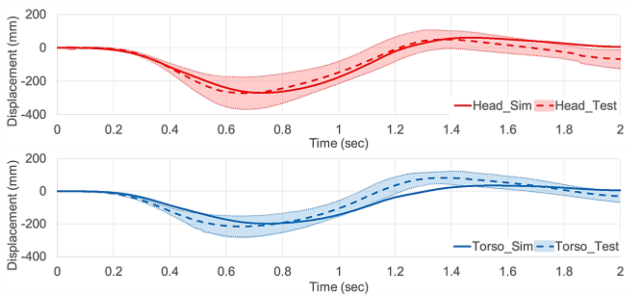
A heuristic calibration of PID gains and neural delay time were carried out until the simulated outcomes, the time plot of the head and torso lateral displacements become analogous to the test measurement. In order to quantify the correlation between the test and simulation, the CORA (CORrelation and Analysis, http://www.pdb-org.com/de/information/18-cora-download.html) score was adopted.. Figure 5 shows the comparison of head and torso lateral displacements (D_y) between test and simulation for three awareness conditions. The PID gains and neural delay times, applied to the simulation are listed together with the CORA scores in Table 4. The neural delay times were estimated based on the onset muscle latency measured at the test. (Huber, 2013) The calculation of CORA score was divided into 2 sections, phase 1 (0.0-0.95sec, clockwise steering) and phase 2 (0.95-2.0sec. counter clockwise steering). In phase 1 a significant higher average CORA score was found than in phase 2, i.e., 0.94 versus 0.46. In the PID closed-loop control, the joint torque was activated to minimize the current angles of each body segment tilting away from the initial vertical axis, i.e., upright alignment of head-neck-torso. This is assuming that the test subject tries to keep the initial upright sitting posture. The criterion for a termination of heuristic calibration trial was reaching 0.8 of the CORA score for the phase 1 (0.0-0.95sec).

Table 4. Calibrated PID gains, neural delay times and CORA scores

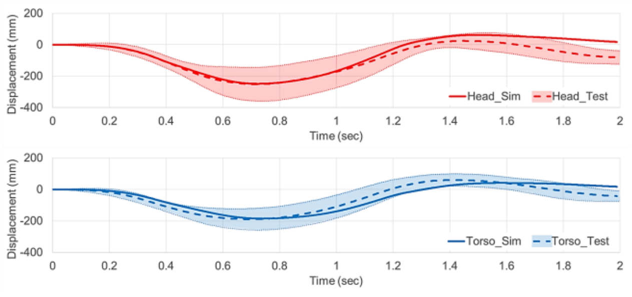
Awareness	PID Gains			Neural delay time(ms)	CORA score*		
	K_P	K_I	K_D		Head(D_{hy})	Torso(D_{ty})	Total
Unaware	35	0.01	500	200	0.879 [0.959, 0.604]	0.738 [0.857, 0.441]	0.809 [0.908, 0.522]
Anticipitated	39	0.01	700	170	0.815 [0.995, 0.418]	0.776 [0.906,0.457]	0.795 [0.950, 0.438]
Informed	55	0.01	2000	100	0.646 [0.962, 0.219]	0.842 [0.970, 0.596]	0.744 [0.966, 0.408]

*: calculated for total (0.0-2.0sec) [phase #1(0.0-0.95sec), phase #2(0.95-2.0sec)]

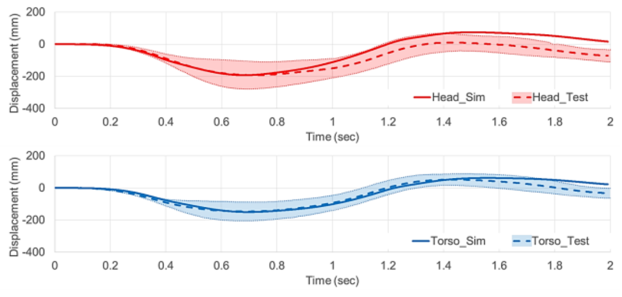
The simulated upper body motion of the unaware case is displayed in the Figure 6. The associated active torque generated at the articulated joint at the center-lower trunk (L5-S1) is also presented in Figure 7.



a) Unaware



b) Anticipated



c) Informed

Figure 5. Comparison of head and torso lateral displacements (D_y) between test and simulation

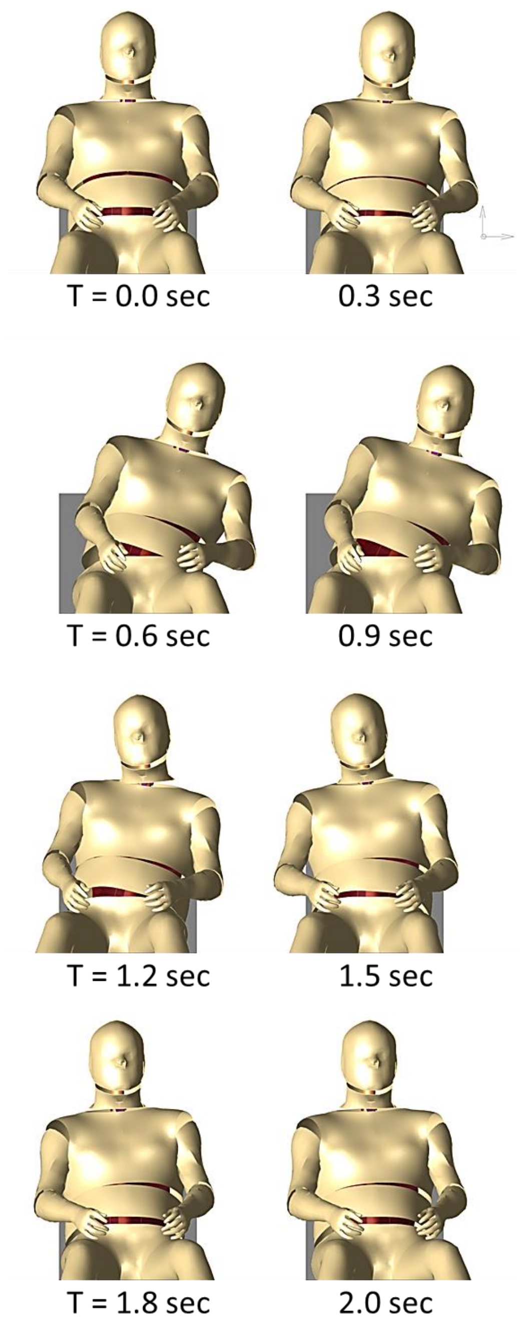


Figure 6. Frontal view of occupant kinematics at simulation (Unaware case).

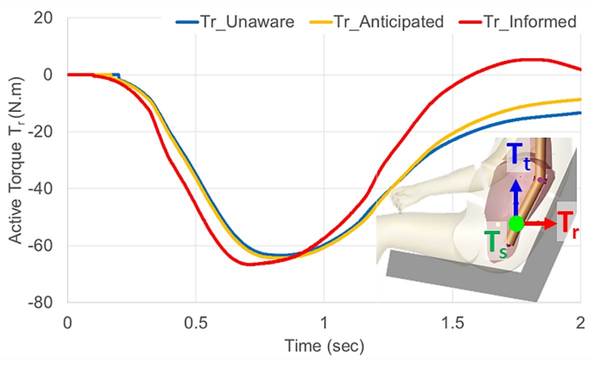


Figure 7. Simulated active torque generation at center-lower trunk joint r-direction (L5/S1 r-direction)

4.2 Creation of a lane change maneuver with reduced lateral jerk

An artificial driving movement for a single lane change is created to produce the same lateral displacement, but with a reduced maximum lateral jerk as seeking a smoother vehicle maneuver than at the human driving test in section 2. It is hypothesized that the test vehicle was driven to take an evasive lane change by a human driving expert just to avoid a crash but bearing no comfort ride in mind. However, a well-programmed machine, e.g., highly autonomous vehicle may steer the vehicle to take the evasive maneuver not just for a crash avoidance but also for a good ride quality of the passenger. Figure 8 shows the fabricated vehicle movement at a single lane change as compared with the measured one at the human driving test. The artificial maneuver was fabricated to have a similar shape of jerk time history as the original test but with more smooth profile as shown in Figure 8. In spite of the reduced maximum lateral jerk at the fabricated maneuver by 27.8% (40.0m/s^3$\rightarrow$28.9 m/s^3), the lateral acceleration and displacement are kept almost same as those at the actual test (also see the Fig, 8).

The same active human body model used for the validation of the single lane change test in Section 4.1 is utilized to predict upper body kinematics at the created lane change maneuver with the reduced maximum lateral jerk. The predicted occupant kinematics at the created lane change maneuver with the reduced jerk revealed decreased lateral head and torso excursions by 5-6 % as expected. (see the Figure 9)

The joint work, the product of joint torque and the angle, can be regarded as a muscle energy produced to maintain a desired posture against the external perturbation, the lateral acceleration (or jerk) at the single lane change in this study. The simulated change of muscle energy at four articulated joints from the original test condition to the fabricated reduced jerk condition is shown in Figure 10, which is around 10 % reduction in total for all three awareness conditions.

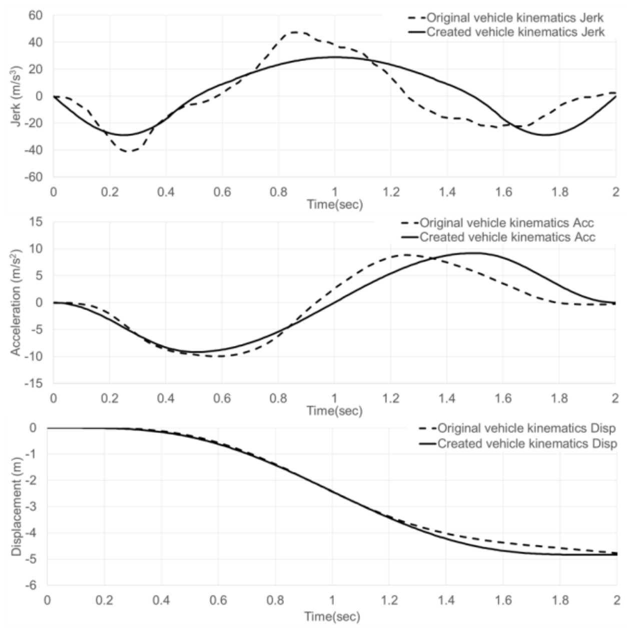


Figure 8. Comparison of vehicle kinematics between at the actual test and at the created maneuver with a reduced lateral jerk.

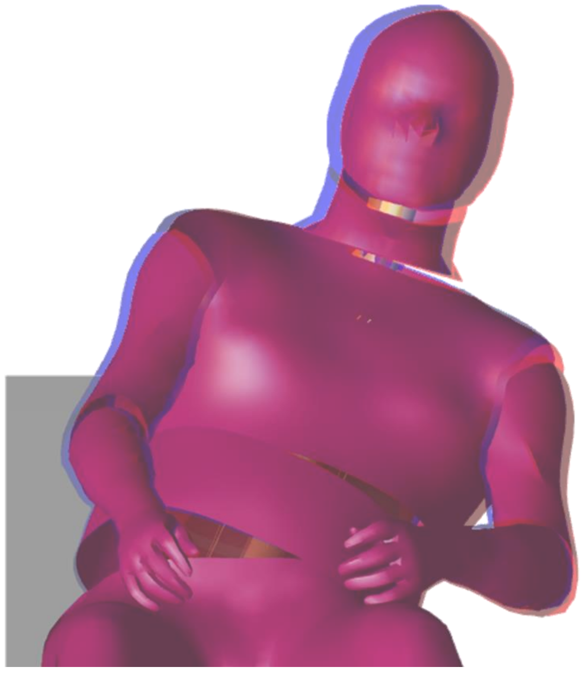


Figure9. Comparison of simulated maximum head excursions for unaware case between the original test condition (red, 0.727 sec) and the reduced jerk condition (blue, 0.695 sec)

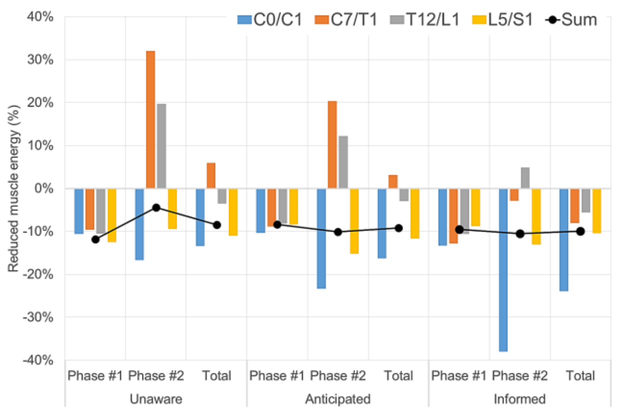


Figure 10. Muscle energy reduction at reduced jerk maneuver

5 Conclusion

The active human body model primarily developed for a ride comfort simulation to investigate the vibration response of human occupants is applied to predict relatively short-term ride quality via simulating a kinematics at an evasive lane change. The lateral movements of head and torso at three awareness conditions, unaware, anticipated, and informed were validated against the vehicle-based subject test. The simulation result showed different correlation outcomes between the 1st phase and the 2nd phase of the lane change maneuver, a good (0.94 CORA score) versus a poor (0.46 CORA score). It is speculated that the current error function in the PID closed-loop control, which is the difference of the current inclined angles of each body segment from the initial vertical axis, may not be appropriate for the 2nd phase. Unlike the 1st phase in the lane change maneuver where the head and torso are moving away from the initial upright position, that is to say, outbound direction, the 2nd phase where the head and torso are moving inbound direction may have substantially different reflexive muscle activity. In addition, the lateral G-force direction of the vehicle becomes completely opposite in the 2nd phase. Taking this human physiology into account will improve the simulation result of the 2nd phase and still remains as our future research challenge.

In order to verify a practical application of the active human body model in the design process of the ride quality, an artificial single lane change maneuver with reduced lateral jerk was fabricated and its effect on the occupant kinematics was investigated. The model predicted a joint work, regarded as a muscle energy used to maintain the sitting posture against the lateral G-force with a comparable reduction at the created lane change maneuver and successfully demonstrated its feasibility of serving as an objective design tool for quantifying the ride quality.

Acknowledgements

The underlying data were generated at the VIRTUAL VEHICLE Research Center in Graz, Austria. VIRTUAL VEHICLE Research Center would like to acknowledge the financial support of the COMET K2 - Competence Centers for Excellent Technologies Program of the Austrian Federal Ministries, the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

VIRTUAL VEHICLE Research Center would furthermore like to express its thanks to its supporting industrial and scientific project partners, in alphabetical order: Bundesanstalt für Straßenwesen-BASt, Robert Bosch GmbH, DYNAmore GmbH, Daimler AG, BMW Group, TRW Automotive, Toyoda Gosei Europe, Partnership for Dummy Technology and Biomechanics, Volkswagen, Graz University of Technology, Vehicle Safety Institute, Graz University of Technology, Institute of Automotive Engineering, Ludwig-Maximilians-Universität Munich.

References

H.Y. Choi, M. Han, A. Hirao and H. Matsuoka, *Virtual Occupant Model for Riding Comfort Simulation*, 12th International Modelica Conference, Czech Republic, 2017.

H.Y. Choi, M. Han and W. Lee, *Active Elbow Joint Model*, The First Japanese Modelica Conference, 2016

M. Han and H.Y. Choi, Elbow joint model with active muscle force, Journal of Mechanical Science and Technology 30/12 5847~5853, 2016

P. Huber, M. Christova, G. A. D'Addetta, E. Gallasch, S. Kirschbichler, C. Mayer, A. Prüggler, A. Rieser and W. Sinz, D. Wallner, *Muscle Activation Onset Latencies and Amplitudes during Lane Change in a Full Vehicle Test*, IRCOBI Conference 2013

A. S. Kilinç and T. Baybura, *Determination of Minimum Horizontal Curve Radius Used in the Design of Transportation Structures, Depending on the Limit Value of Comfort Criterion Lateral Jerk*, FIG Working Week, Knowing to manage the territory, protect the environment, evaluate the cultural heritage, Rome, Italy, 2012

S. Kirschbichler, P. Huber, A. Prüggler, T. Steidl, W. Sinz, C. Mayer and G. A. D`Addetta, *Factors Influencing Occupant Kinematics during Braking and Lane Change Maneuvers in a Passenger Vehicle*, IRCOBI Conference 2014

C. Kleinbach and J. Fehr, *Comparison of Muscle Activated HBMs in a Lane Change Manoeuvre*, IRCOBI Conference 2017

G. Lauriks, J. Evans, J. Förstberg, M. Balli and I. B. de Angoiti, *Uic comfort tests: Investigation of ride comfort and comfort disturbance on transition and circular curves. Technical report*, Swedish National Road and Transport Research Institute, 2003.

W. Schofield, *Engineering Surveying – Theory and Examination Problems for Students*, Butterworth-Heinemann, Oxford, New Delhi, 2001

J. Takahashi, M. Yamakado and S. Saito, *Evaluation of preview G-Vectoring control to decelerate a vehicle prior to entry into a curve*, International Journal of Automotive Technology, Vol. 14, No. 6, pp. 921−926, 2013

K. Yamada, H. Motojima, Y. Kitagawa and T. Yasuki, *Investigation of relations between occupant kinematics and supporting by the seat in lane change maneuvers*, 20165176, JSAE Spring Congress, 2016

AASHTO; *A Policy on Geometric Design of Highways and Streets*, American Association of State Highway and Transportation Officials, Washington DC, USA. 2001

EuroNCAP, *Pedestrian Human Model Certification*, Version 1.0 TB 024, 2017

International Organization for Standardization. *ISO 2631-1*. International standard. ISO, 1997.

www.GHBMC.com

[Industrial paper]

Combustion Engine Mechanism Analyses Using SimulationX

Tomohide Hirono[1] Takanori Watanabe[2]

[1,2] CAE Research & Development Center, NewtonWorks Corp., Japan,
{hirono.tomohide,twatanabe}@newtonworks.co.jp

Abstract

When we apply more efficient combustion profile using advanced mechanisms to ICEs, vehicles with ICEs can exceed BEVs from the view of total environmental performance. This paper illustrates SimulationX, a Modelica simulation tool, can be a tool for the developments of advanced mechanical systems such as new valve trains and cranking systems. This paper also shows a case of FMI co-simulation of a cam phaser between the tool and another hydraulic simulation tool which is used to model a conventional hydraulic system of an existing ICE.

Keywords: Internal Combustion Engine, Valve Train, Cam Phaser, Variable Compression Ratio, FMI, Co-Simulation

1 Introduction

The total (i.e. from well to wheel) environmental performance of vehicles with ICEs (Internal Combustion Engines), can exceed BEVs (Battery Electric Vehicles) when the efficiency of combustion is improved and advanced mechanical systems are required to realize the combustion profiles.[1] Resources to develop ICEs are, however, not sufficient since automotive OEMs are also required to develop xEVs in parallel. To improve the productivity of developments of the systems, more efficient tools to confirm ideas of engineers are in demand. A cam phase change mechanism (cam phaser), variable valve lift mechanisms and a variable compression ratio system are modeled using SimulationX, a Modelica simulation tool, and illustrated in this paper to show the availability of the tool to realize faster development at earlier stages in MBD cycles.

This paper also explains availabilities of FMI between two hydraulic simulation tools. Most of the conventional hydraulic tools used by automotive OEMs to model hydraulic systems of ICEs are single domain tools. The Modelica tool is a multi-domain simulation tool and it can realize simulation of hydraulic-mechanical interactions. The existing hydraulic circuit modeled by a conventional tool can be slightly modified to connect to the Modelica tool to supply hydraulic pressure to drive a hydraulic CAM phaser modeled in the tool. The scheme is realized using FMI technology and illustrated here.

[1] Mr. Hitomi presented at Automotive World 2018 in Tokyo.

2 Valve Trains

Changing combustion timing and adding air mass flow flown into cylinders are ways to increase efficiency of ICEs. They are realized by changing valve lifts in timing and in traveling distances. Cam phasers are used to change lifts in timing. A single valve with multiple cam mechanism or adding intermediate arm between a cam and a valve mechanism is used both in timing and lift changes.

2.1 Cam Phaser

A hydraulic cam phaser is shown in Figure 1 and the diagram of total cam phaser system modeled with the tool is shown in Figure 2.

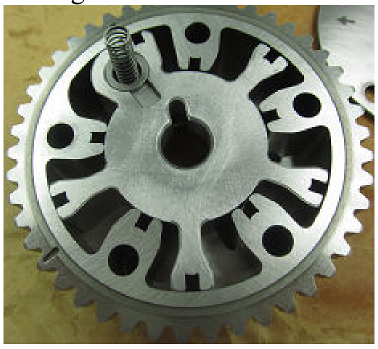


Figure 1 Hydraulic Cam Phaser (5 x 2 Rooms)

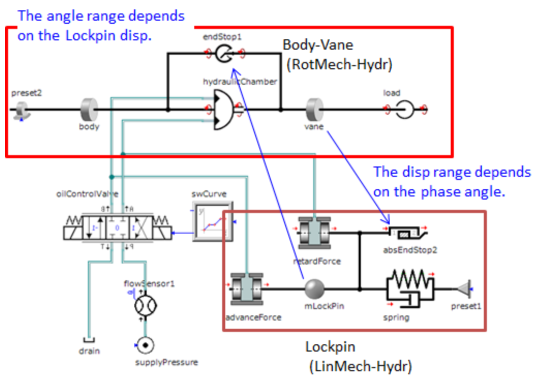


Figure 2 Diagram of Cam Phaser System

This type of hydraulic cam phasers has a body with a sprocket and inner vanes and a rotor with outer vanes. The sprocket is driven by a chain to/from a crank shaft sprocket and the rotor is connected to a cam shaft. There are some fluid volumes (rooms) between the body vanes and the rotor vanes into which lubricant flows to realize to advance or retard phases. A valve controls flow direction to fluid volumes into which the pressurized fluid flows. A lockpin is used to fix the phaser angle when the pressure is lower than a certain level to avoid free motion in low pressurized states such as starting up the engine. The phaser rotor rotates by the fluid pressure relative to the crank shaft rotations then phase advancement or retardation occurs. Figure 3 shows the results of the model. At t=0, there is no pressure in the circuits and it gradually increases to t=0.1[s]. At t=0.2[s] advance signal changes to 1 then the hydraulic pressure in advance rooms goes up to 100[kPa] to rotate the vane. The displacement of the lockpin is 5 [mm] and it indicates relative motion between the rotor and the body. The relative rotation stops at about t=0.95[s] since the rotor contacts to the other side of the body mechanically. The retardant volume is almost zero now. The retardant motion occurs at t=1.2[s] triggered by a '-1' advance signal and a motion to the opposite direction is observed. The lockpin moves back to 1 [mm] position at t= 2.0 [s]. The lockpin connects the rotor and the body mechanically.

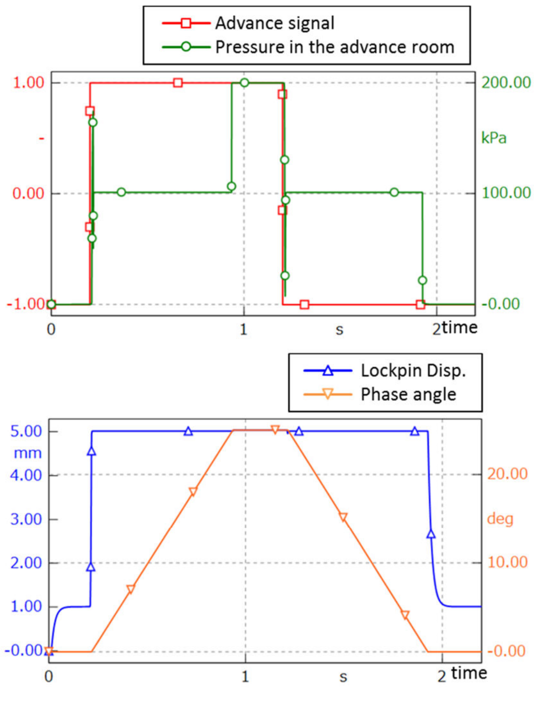


Figure 3 Simulation Results. (Top) Input Signal to Advance and Pressure in the Advance Rooms (Bottom) Lockpin Displacement and Phase Angle

2.2 Valve Mechanism

2.2.1 Multiple Cam Change Over Mechanism MCCOM

This type of mechanism has two cams with different profiles for a valve. One of the cams is selected to push the valve according to the engine rotational speed and/or load required (Figure 4).

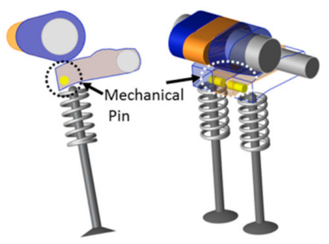


Figure 4 MCCOM (Blue: Low Lift Cam Orange: High Lift Cam)

This mechanism is modeled as seen in Figure 5 A pin is used to fix the selected cam to work. The traveling time of the pin has direct relation to the response of the cam selection. A cam element developed for the tool is used here.

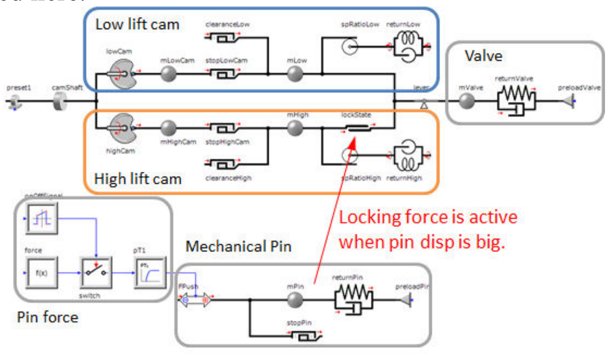


Figure 5 Diagram of MCCOM

The Figure 6 shows the valve stroke. The first stroke is invoked by the low lift cam and the second stroke is invoked by the high lift cam.

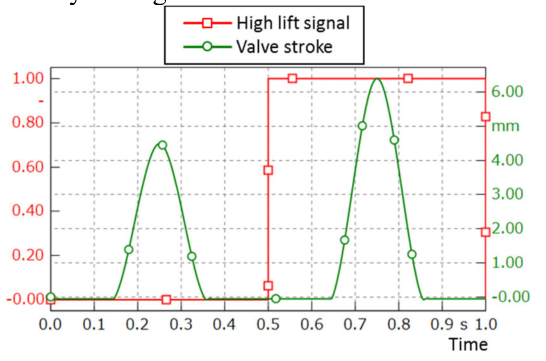


Figure 6 Valve Strokes

2.2.2 Intermediate Arm and Cam

This type of the system has a set of an intermediate arm and a cam between a cam attached to a cam shaft and a rocker arm. The traveling distance (stroke) of the valve can be changed continuously by setting the angle of the intermediate arm which I controlled by a control shaft driven by an electric motor. The mechanism can be modeled using MBS (Multi Body System) Library and contact elements of the tool as seen in Figure 7.

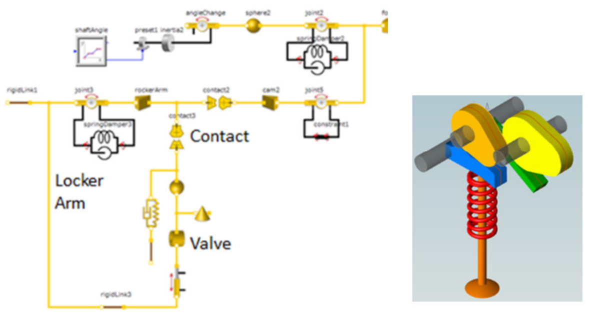


Figure 7 (Left) Diagram of Valve Train with Intermediate Cam and Arm.
(Right) Mechanism of Intermediate Arm and Cam.

The stroke changes are seen in Figure 8 at t=2.5 and 4.5[s] when control shaft angle is changed.

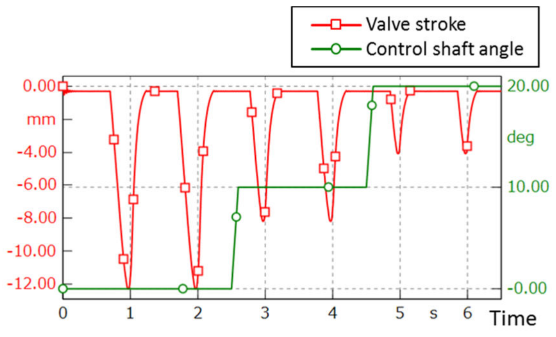


Figure 8 Changes of Valve Stroke by Control Shaft Angle

3 Variable Compression Ratio System

One of different approaches to make ICEs more efficient is changing compression ratios. Conventional ICEs have crank systems and the traveling distance (stroke) of pistons are fixed. This type of mechanisms can change the distance by an additional multiple-link (Figure 9). The diagram modeled by the tool is seen in Figure 10. Mechanical engineers can model this kind of systems easily using mechanical 3D-CAD after each of the components of the links is designed as a CAD model but they have difficulties to calculate required torque to rotate the linkage at earlier stage of design cycles. The tool can be used to estimate the torque in many circumstances of engines in practical uses. The Figure 11 shows how its compression ratio changes by its control shaft angle. Optimal quantities of fuel will be supplied to the cylinders in order to obtain required power to drive the vehicle which are dependent on then current compression ratio. Required power to rotate the control shaft can be also obtained when pressure in the cylinder is obtained.

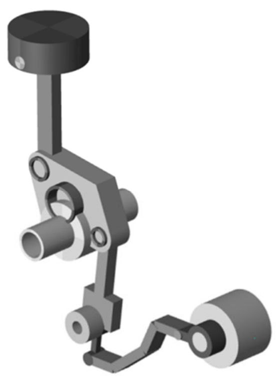


Figure 9 Variable Compression Ratio Mechanism

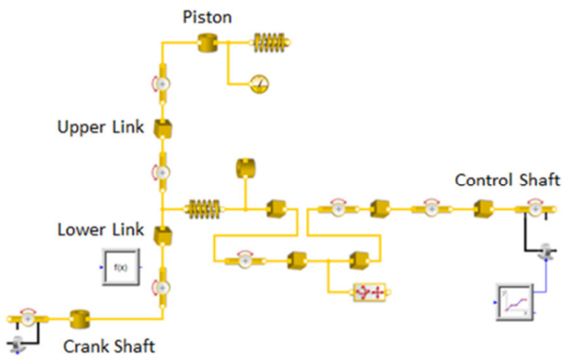


Figure 10 Diagram of VCR Using MBS

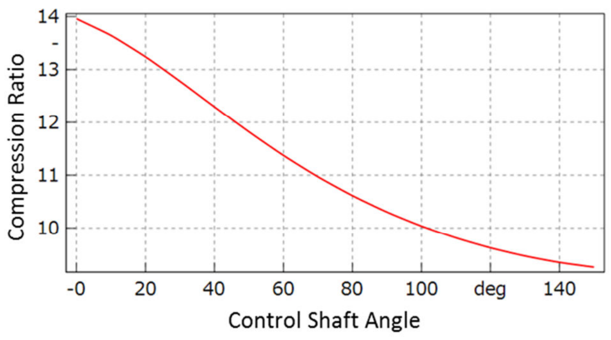


Figure 11 Compression Ratio Change by Shaft Angle

4 Co-Simulation of Hydraulic Tools with FMI

ICEs have hydraulic circuits to lubricate between pistons and cylinders, crank shaft bearings and the shafts, cam mechanisms. Lubricant is also used for sealing and cooling purposes. The circuits used to be designed and analyzed using conventional flow simulation tools. Those tools are, in most cases, designed mainly for flow and/or thermal analysis purposes and have not enough capability to simulate mechanical motion. The OEMs, however, have a large amount of legacy data generated with the fluidic/thermal tools. The engineers of the OEMs have interests in using the models to survey additional functions such as cam phasers, variable compression mechanisms and so on whose motions are driven by the power supplied by hydraulic systems. We would like to discuss a model - i) the existing circuit model supplies fluid to drive a new mechanical component, ii) the new mechanism is modeled by a different tool and iii) a co-simulation with the two tools is performed using FMI technology here in this section. Two different tools are employed for the co-simulation, one is SimulationX and the other is Flowmaster. Please note the diagrams of the models shown here in this paper are expressed with a single tool (SimulationX) due to intellectual property reasons. The model used for the co-simulation is different from the one seen in Figure 12, actually.

4.1 Connections for CoSimulation

We added a junction at a point marked as A in the Figure 12 to supply oil to a cam phaser, the new mechanism. The oil supplied to the cam phaser goes down to an oil pan via volumes for advancement or retardation. The whole system diagram that we model is seen in Figure 13. The results are shown in Figure 14. It shows the pressure at the main gallery (MG) changes by the phase switch because of additional flow to the cam phaser.

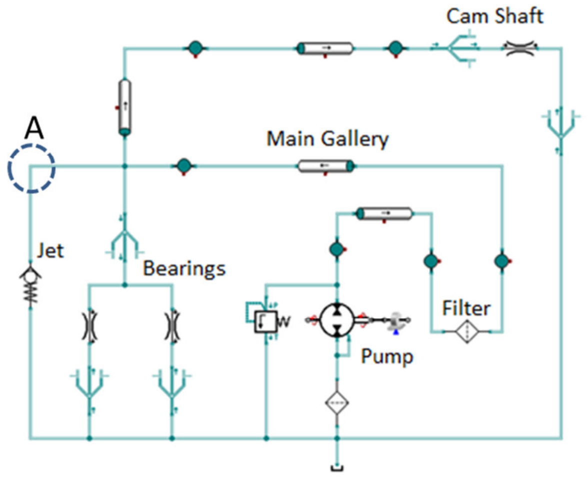


Figure 12 Diagram of Existing Hydraulic Circuit

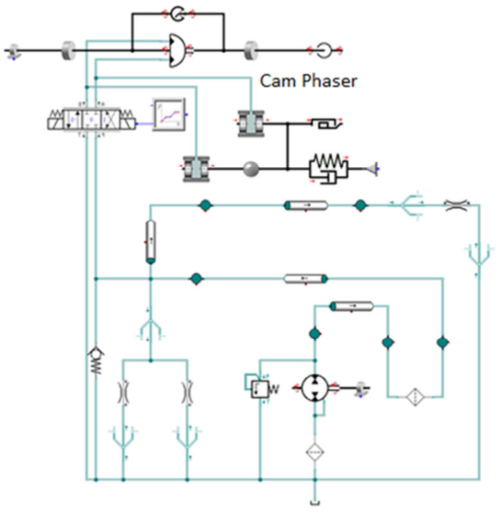


Figure 13 Diagram of Hydraulic Circuit with a Cam Phaser

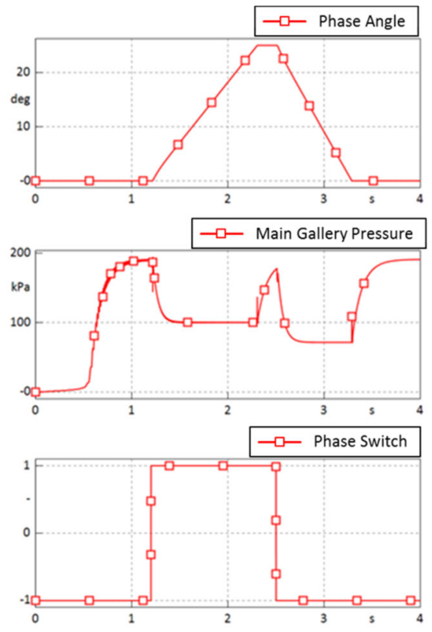


Figure 14 Results of a Single Tool Simulation Model (Top) Phase Angle (Middle) MG Pressure (Bottom) Phase Switch Signal

To branch the circuit at A in Figure 12, we added a flow source and a pressure sensor in the existing model to send the pressure signal to the sub-circuit model and receive the mass (or volumetric) flow out to the sub-circuit (Figure 15). The sub-circuit model has a pressure source as its input and a flow sensor is also added as an output of the model to send the mass flow rate signal to the main circuit model (Figure 16). The total model is shown in Figure 17.

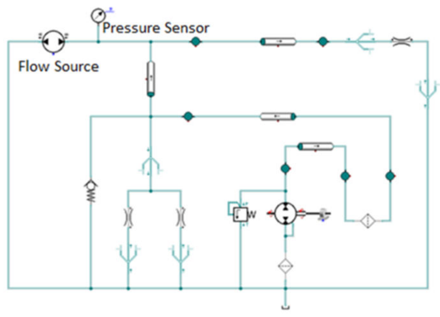


Figure 15 Modified Existing Hydraulic Circuit Model for CoSimulation

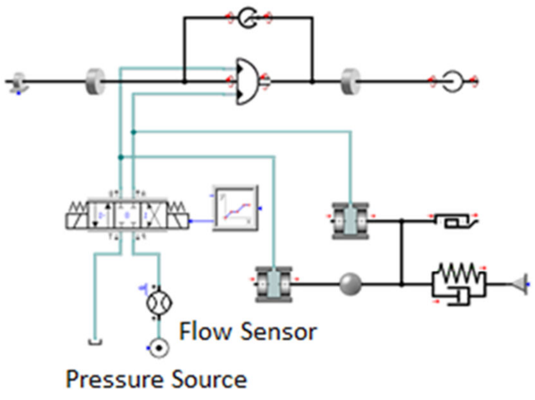


Figure 16 Connectors of Cam Phaser Circuit

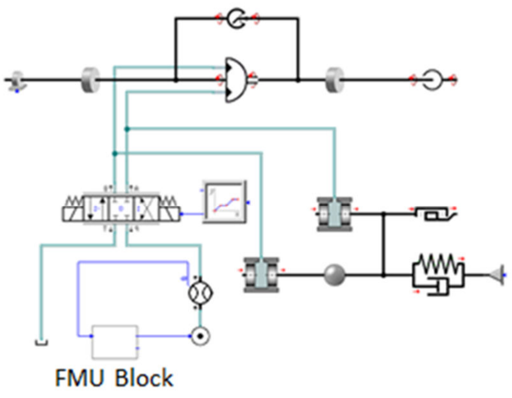


Figure 17 Diagram of Master Tool with a Slave FMU

4.2 FMI CoSimulation Results

The two models co-simulate using FMI1.0 tool co-simulation. The tool used for the sub-circuit model is now the master tool since it has a capability to import FMUs of FMI co-simulation. Adding communication ports to the conventional tool and creating a wrapper program to be a slave are relatively easy. The results with the co-simulation model are seen in Figure 18. The communication step sizes (CSS) was set to 1.0E-4 [s].

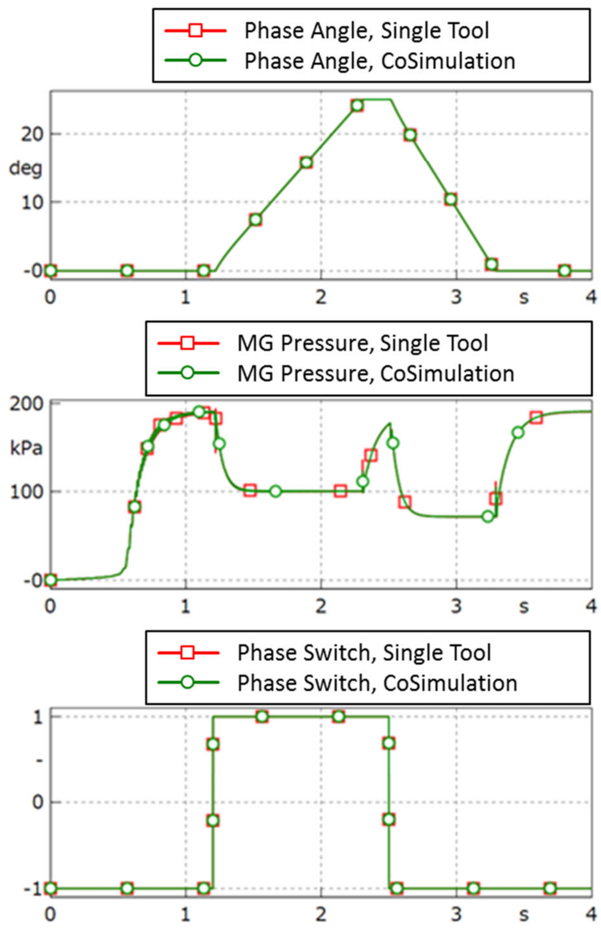


Figure 18 Results of FMI CoSimulation Model and a Single Tool Model (Top) Phase Angle (Middle) MG Pressure (Bottom) Phase Switch Signal

The pressure values at the MG of a single tool simulation and co-simulation are almost identical and so are the phase angles. The differences are less than 0.2 %.

4.3 What We Have Learned

We confirmed that FMI connection can be used in action and found followings.

1. Fluid property differences have to be discussed beforehand.
2. Initial values should meet but it is not a necessity in this case.
3. Time constant is very small in hydraulic systems. The CSS must be taken into consideration to obtain good results without using much simulation time.

5 Conclusion

We employed SimulationX to model mechanical systems used in advanced ICE mechanisms to increase efficiency of ICEs. This kind of simulation can be performed without CAD data and is useful to confirm

the availability of new mechanisms at an earlier stage of a developments process.
In the latter half of this paper, a co-simulation of a conventional hydraulic tool and a multi domain Modelica tool is discussed. FMI was used and it was revealed to be applicable for the co-simulation.

References

Shunichi Aoyama (2009): A Study of A Mulitiple-Link Variable Compression Ratio System for Improving Engine Performance. *JSME Journal*, Vol.112, No.1092, 2009.

Ryutaro Tagishi et al. (2008) : Development of i-VTEC Gasoline Engine for 2008 Model Year Honda FIT. *Honda R&D Technical Review*, Vol.20, No.1 :6-13, 2008.

The CryoLib - Modelling Superconductors with Modelica

Alexander Pollok Dirk Zimmer

Institute of System Dynamics and Control, DLR German Aerospace Center, Germany
{alexander.pollok,dirk.zimmer}@dlr.de

Abstract

A Modelica library for the thermal and electrical modelling of cryogenic and superconducting systems is presented. The library design is compatible with the electrical and fluid components from the Modelica Standard library. At the same time, several typical effects of cryogenic systems are modelled that have no equivalent in the Standard library. Easy usability and extendability of the components is emphasized. In this paper, the different parts of the library are explained in more detail. To illustrate the capabilities of the library, three showcases are presented: a simple solenoid magnet system, a current lead and a current limiter. The results show that many effects occuring in cryogenic and superconducting systems can successfully be simulated using this library.

Keywords: cryogenic, superconductivity, magnet, cryocooler, simulation, thermal, electric, HTS

1 Introduction

While the concept of superconductivity is known for over a century, it took a while for commercial applications to take root. In 2018, superconducting magnets can be found in the radiology departments of most first world hospitals. However, after all this time, the cost per kiloampere-meter for superconducting cables continues to go down, especially in the case of high temperature superconductors (HTS) (Selvamanickam et al., 2011; Rupich et al., 2013). This makes more and more applications viable. For example, cryogenic systems are envisioned for future electric aircraft projects (Gemin et al., 2015; Berg et al., 2015).

Thermal design makes up a big part of the overall effort that goes into the development of superconducting systems. For some applications like nuclear magnetic resonance tomography (NMRT) systems this can be done in Excel. In other applications, the transient behavior of the overall system is more important. This is especially true if the superconducting system is coupled to other systems and the overall system behavior cannot be neglected. In that case, Modelica is a naturally suited technology to describe the overall system behavior.

Some work has already been directed towards this topic: Łanczont (2016) presented a very detailed but also very specific electrothermal model of second generation HTS cables. A library for the simulation of supercritical helium loops is presented by Zanino et al. (2012) and verified in (Zanino et al., 2014). No general-use library for the typical aspects of the design of superconducting systems is available to best knowledge of the author.

This work closes that gap by presenting CryoLib, an easy-to-use Modelica library that provides components and interfaces to simulate the transient behavior of superconducting systems. The paper is structured as follows: In Section 2, the library structure is presented and some components are explained in more detail. Section 3 shows some examples that have been developed using the CryoLib library. These are used to present both physical effects and possible use cases of the library. The paper is discussed in Section 4 and concluded in Section 5.

2 Library

2.1 Library Concept

Based on the nature of the Modelica language, some usage scenarios for this library are more probable than others. If highly detailed three-dimensional effects have to be modelled, finite-element or similar methods should be used. For design or architecture studies, or for controller development, Modelica makes fast model development cycles possible, at the cost of accuracy. For this reason, the CryoLib is designed to be relatively light-weight and easy-to-use. Of course, an end-user can always choose to extend the library to their liking.

The library is divided into several sub-packages, see Figure 1 for the overall structure. The UserGuide subpackage contains general meta information about the library, the Examples package contains simulation models built from the library to demonstrate use cases. The other packages are presented in more detail in the following.

All component packages also contain a test-package. This is not only for validation of the components, but also serves as a tutorial to end users.

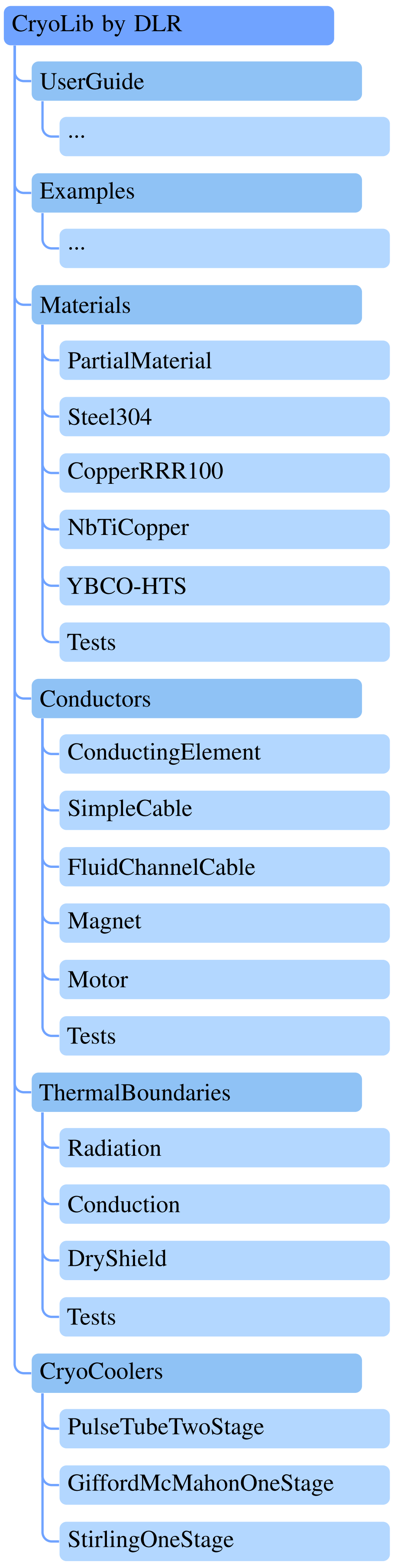


Figure 1. Structure of the CryoLib

2.2 Materials

The Materials package contains several models that estimate material properties based on temperature, current density and magnetic field density. This is especially important since material properties can change significantly at low temperatures. For instance, the electric resistivity of very pure copper can be 100 to 1000 times smaller than the resistivity at room temperature.

The library includes technical high temperature superconductors, technical low temperature superconductors and non-superconductors like austenitic steel or copper. These models are also used as a basis for the components of the other sub-packages. For the superconducting materials, users can also change the ratio of superconductor to support medium using parameters.

Each of the models takes temperature, field and current as inputs, computing volumetric heat capacity, thermal conductivity and the electrical resistivity. The resistivity is set to zero if the material is inside the limits of superconductivity, based on field, temperature and current density. The dependency between these variables is not trivial and illustrated in Figure 2.

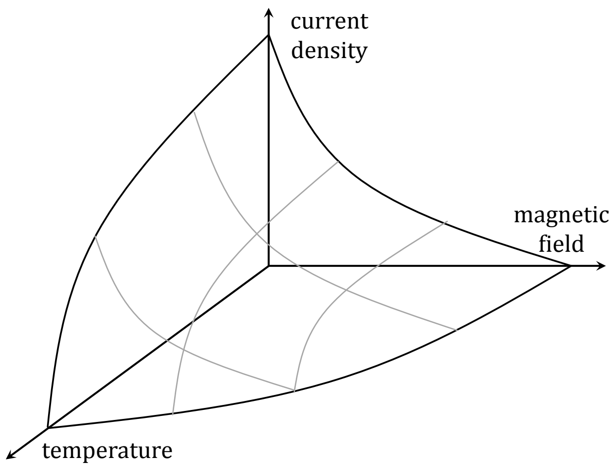


Figure 2. Illustration of the limits of superconductivity

The cryogenic data for the materials is taken from publicized manufacturer specifications (SuperPower Inc, 2014; AK Steel Corporation, 2007) and scientific papers (Bottura, 2000; Manfreda, 2011; Lu et al., 2008; Duthil, 2015). The data points are linearly interpolated afterwards.

While critical temperature and critical field are usually determined by the type of superconductor, critical current density is dependent on the manufacturing process. Therefore, the user can scale the critical surface in the direction of the current density using a parameter.

2.3 Conductors

This package contains several components that simulate superconductors. As such, it is the main package of the library.

The most basic component is called ConductingElement. It models a cylindrical shape with axial electric current. The element material can be chosen from the Materials package. Two Modelica.Electrical.Analog connectors allow using the component similar to a conventional resistor element. Of course, the resistivity of the element can drop to zero if all requirements for superconductivity are met. As such, the overall electrical circuit has to allow for this possibility. Based on the voltage drop over the element, current, heat dissipation, temperature and maximum field is simulated. A connector from Modelica.Thermal.HeatTransfer allows for evacuation of dissipated heat.

The SimpleCable component models a discretized cylindrical shape. It behaves similar to multiple ConductingElement models stacked together.

The FluidChannelCable component behaves similar to the SimpleCable component, but additionally features Modelica.Fluid connectors. These can be used to model cable-in-conduit conductors (CICCs) or current leads.

The Magnet and Motor components are specialised versions of the ConductingElement component with additionally modelled inductance.

2.4 ThermalBoundaries

This package offers helper components to model typical thermal transfer effects in cryogenic systems.

In the Radiation component, thermal radiation is modelled based on the temperature on both sides, the area and the number of layers of used Multi-layer insulation. The Conduction and DryShield components can be used to simulate the transient thermal effects of radiation shields and the suspension.

2.5 CryoCoolers

Cryocoolers as described for instance by Satoh et al. (1996) are used to cool down small cryogenic systems. In the CryoCoolers subpackage, simple models representing several types of cryocoolers are included. These models offer a certain amount of cooling heat flow base on the temperature. Also, some amount of thermal inertia is included. The performance data is taken from Xu et al. (2008), Radebaugh et al. (2007) and Sunpower (2012).

3 Examples

In this section, several simulation models are shown that have been built using the Cryolib. Additionally, components from the Modelica Standard Library (ModelicaAssociation, 2008), HelmholtzMedia (Thorade and Saadat, 2012) and the NoiseLib (Klöckner et al., 2014) are used.

3.1 Magnet System

The first example system represents a standard superconducting magnet design with low temperature superconductors, as often used for nuclear magnetic resonance applications. It is illustrated in Figure 3.

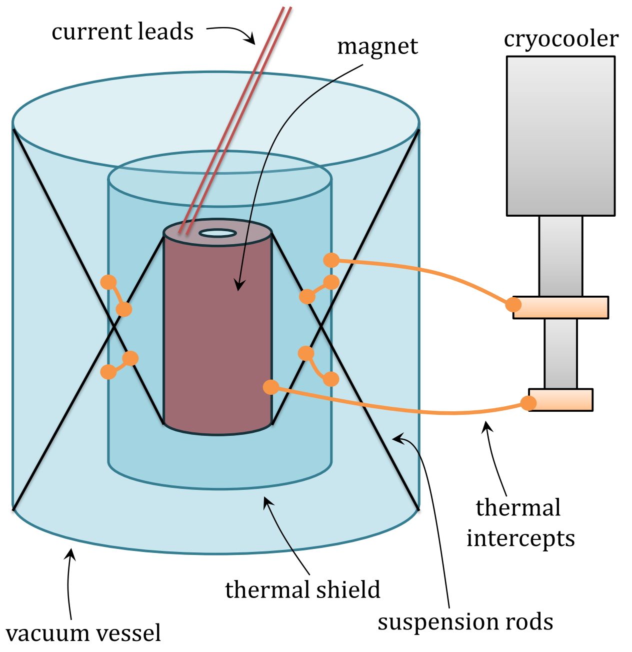


Figure 3. Illustration of a simple solenid magnet system

Refrigeration power is more costly at lower temperatures. For that reason, two-stage cryocoolers are used. These offer a small amount of refrigeration power at the operating temperature of the magnet (around 4 Kelvin) and a larger amount of refrigeration power at an intermediate temperature (around 80 Kelvin). A thermal shield is used to protect the cold side of the magnet from heat radiation and convection. The corresponding Modelica model is shown in Figure 4.

Conduction and Radiation elements are used to model the heat loads on both the shield and the actual magnet. Conduction elements are also used to model the heat resistance of the thermal intercepts between cryocooler, shield and magnet.

On the electrical side, a controller measures the magnet temperature. As soon as the temperature falls below 4.5 Kelvin, a constant voltage is applied until a target current is reached. The target current is set above the critical current, to force a quench. The thermal behavior of current leads is not modelled. The simulation results for this system are shown in Figure 5.

It is apparent that the shield side is colder than the magnet for the longest part of the cooldown. This can be explained by the smaller cooling power of the second stage compared to the first stage.

The magnet cooldown accelerates from 70 to 10 K. This is due to the very small heat capacity of most materials at low temperatures.

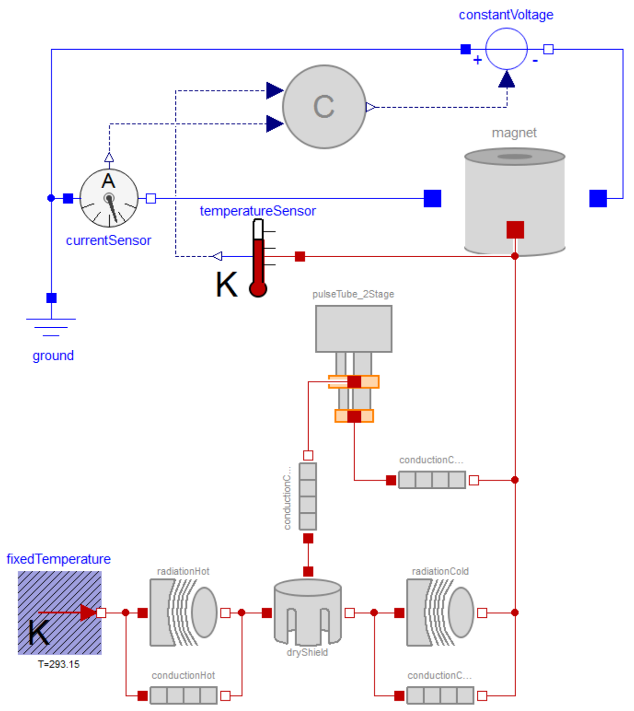


Figure 4. Modelica model of the "magnet system" example

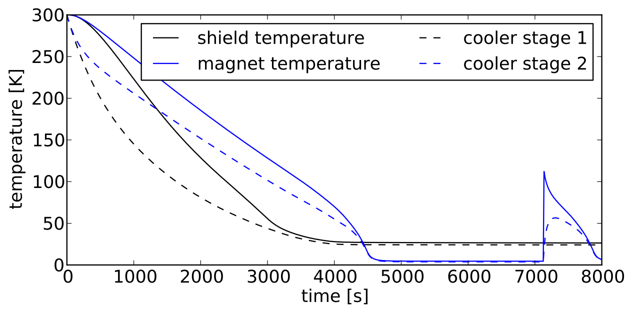


Figure 5. Simulation results for the "magnet system" example

At roughly 7000 seconds, a quench occurs. The magnet loses its superconducting property and the energy stored in the magnets inductivity is dissipated. This results in a sudden temperature spike.

3.2 Current Lead

The second example represents a current lead. Current leads are used to feed power from a conventional source to a cryogenic system. This generates a major engineering challenge, since electrical conductivity generally comes with unwanted thermal conductivity (Franz and Wiedemann, 1853)[1]. Also, the warm ends of current leads cannot be superconducting, generating additional electrical heat dissipation. These thermal loads can overwhelm most cooling systems, which have very low coefficients of performance at low temperatures.

[1]The original article is written in German. However, a search for "Wiedemann–Franz law" results in English translations.

A typical approach, which is also modelled here, is to counter-cool the current lead with helium. At the cold end, liquid helium is supplied into the current lead. It absorbs the majority of the dissipated heat and leaves the current lead at the warm end.

The modelled system is shown in Figure 6.

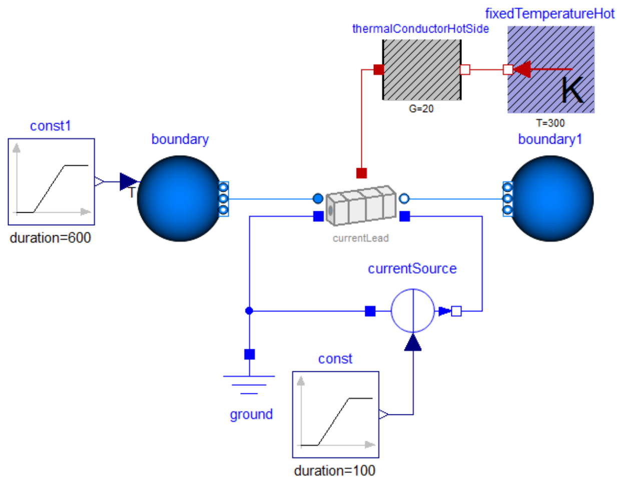


Figure 6. Modelica model of the "current lead" example

The current lead itself is modelled as copper with a residual resistivity ratio (RRR) of 100 and discretized into 6 elements. The system is initialized at room temperature and the helium source at the left is ramped down from 300 to 4 Kelvin in the first 600 seconds at slight overpressure. From 700 to 800 seconds, the current is ramped up from zero to 25.000 amperes. The right side of the current lead is connected to room temperature and atmospheric pressure.

The simulation results for this system are presented in Figure 7.

It can be seen that the temperature distribution in the current lead is not linear during the cooldown. This can be explained by the small thermal conductivity at low temperatures.

When the electrical current is ramped up from 700 to 800 seconds, a significant temperature increase can be seen throughout the current lead. The effect is much more pronounced near the warm side of the current lead. At low temperatures, the electrical conductivity of the used material is much better resulting in lower heat dissipation.

At 1000 seconds, the system is very near to the steady state. The axial heat flow in the current lead is 411 Watts at the warm end and 11.2 Watts at the cold end. That means that the cooling system only has to deliver 11.2Watts at a temperature of 4 Kelvin, in addition to the re-liquefaction of the boiled-off helium (5.5 grams per second).

Note that the used parametrization of the system was not optimized in any way. For benchmark designs, see for example Buyanov (1985). More advanced superconducting current lead designs are presented by Wesche and

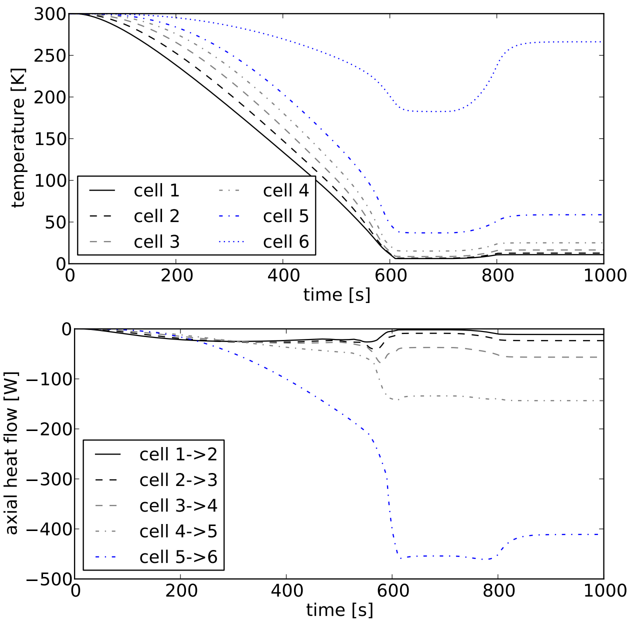


Figure 7. Simulation results for the "current lead" example

Fuchs (1994).

3.3 Current Limiter

Superconducting current limiters are used instead of fuses because they offer fast current shut-off and don't require maintenance after each triggering. High temperature superconductors are especially suited to this application. The modelled example shows a simple electric circuit with a variable resistor and inductivity. It is shown in Figure 8.

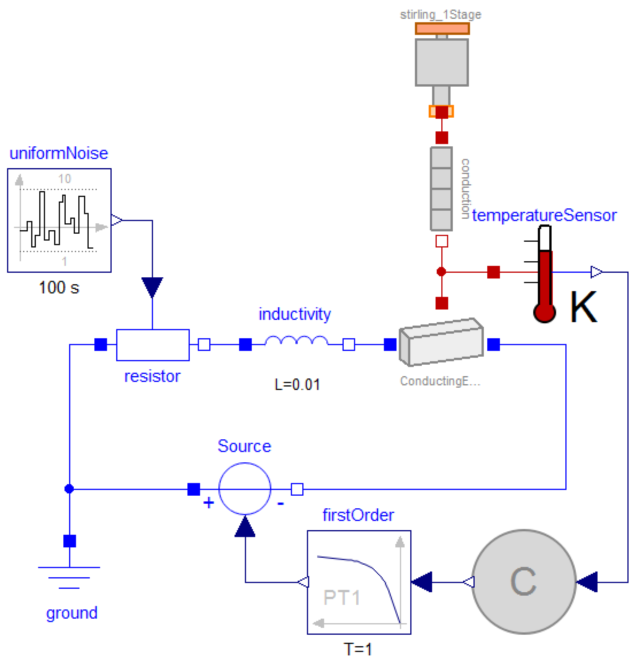


Figure 8. Modelica model of the "current limiter" example

A superconducting element is connected in parallel and cooled down using a cheap stirling-type cryocooler. A simple bang-bang controller switches on a constant voltage when the superconducting element is below 45 Kelvin and switches it off when it is above 100 Kelvin. The variable resistor is controlled by noise such that the maximum current of the superconducting element - which is also dependent on the temperature and the self-field - is sometimes exceeded. The simulation results can be seen in Figure 9.

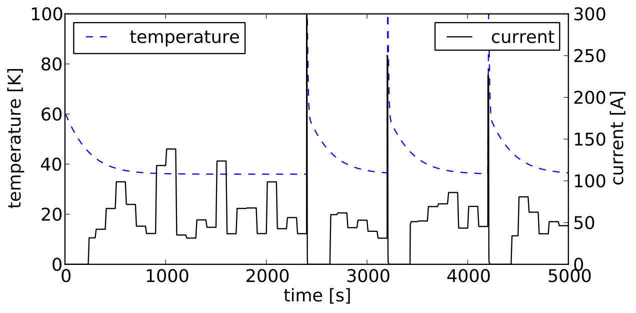


Figure 9. Simulation results for the "current limiter" example

The system reacts as designed. Every time the maximum current is exceeded, superconductivity is lost. The resistivity of the current limiter is now dominating the system and most of the energy stored in the inductivity is dissipated here. The cooler takes some time to cool the limiter down again and the cycle starts over.

4 Discussion and Future Work

The library as presented can be used as a basis for quick optimization studies, controller development or plausibility considerations. For more involved simulation tasks, extensions are necessary. For instance, a superconducting cable can not only be discretized in axial direction, but also in radial direction. This can be important since the self-induced magnetic field in an electrical conductor is linearly dependent on the distance to the central axis. The conductor can still be superconducting near the central axis when superconductivity is already lost near the border. Also at the moment, there is no anisotropy included for high temperature superconductors. These are often produced in the form of thin tapes. The field strength limits are much higher if the field vector is in the plane of the tape (Selvamanickam et al., 2011; Hazelton et al., 2009). Extensions of the library could include these effects, at the cost of added complexity.

5 Conclusion

We present a library for the modelling and simulation of transient cryogenic systems. The library can be used together with the Modelica Standard Library, while extending the range of modelled effects significantly.

Acknowledgements

We thank Trey and Matt for inspiration.

References

AK Steel Corporation. 304 and 304l stainless steel. Technical report, 2007. URL `www.aksteel.com/pdf/markets_products/stainless/austenitic/304_304l_data_sheet.pdf`.

F Berg, J Palmer, L Bertola, Paul Miller, and Graham Dodds. Cryogenic system options for a superconducting aircraft propulsion system. In *IOP Conference Series: Materials Science and Engineering*, volume 101, page 012085. IOP Publishing, 2015.

Luca Bottura. A practical fit for the critical surface of nbti. *IEEE transactions on applied superconductivity*, 10(1):1054–1057, 2000.

Yu L Buyanov. Current leads for use in cryogenic devices. principle of design and formulae for design calculations. *Cryogenics*, 25(2):94–110, 1985.

Patxi Duthil. Material properties at low temperature. *arXiv preprint arXiv:1501.07100*, 2015.

R Franz and G Wiedemann. Ueber die wärme-leitungsfähigkeit der metalle. *Annalen der Physik*, 165(8):497–531, 1853.

Paul Gemin, Tom Kupiszewski, Arthur Radun, Yan Pan, Rixin Lai, Di Zhang, Ruxi Wang, Xinhui Wu, Yan Jiang, Steve Galioto, et al. Architecture, voltage, and components for a turboelectric distributed propulsion electric grid (avc-tedp). 2015.

Drew W Hazelton, Venkat Selvamanickam, Jason M Duval, David C Larbalestier, William Denis Markiewicz, Hubertus W Weijers, and Ronald L Holtz. Recent developments in 2g hts coil technology. *IEEE Transactions on Applied Superconductivity*, 19(3):2218–2222, 2009.

Andreas Klöckner, Franciscus LJ van der Linden, and Dirk Zimmer. Noise generation for continuous system simulation. In *Proceedings of the 10th International Modelica Conference*, number 96, pages 837–846. Linköping University Electronic Press, 2014.

Michał Łanczont. Electro-thermal numerical model of superconducting tape. *Przegląd Elektrotechniczny*, 92(6):134–137, 2016.

J Lu, ES Choi, and HD Zhou. Physical properties of hastelloy® c-276 at cryogenic temperatures. *Journal of applied physics*, 103(6):064908, 2008.

Giulio Manfreda. Review of roxie's material properties database for quench simulation. *TE Technology department internal note*, 35, 2011.

Modelica-Association. The Modelica Standard Library. *Online, URL: http://www.modelica.org/libraries/Modelica*, 2008.

Ray Radebaugh, Agnes O'Gallagher, Michael A Lewis, and Peter E Bradley. Proposed rapid cooldown technique for pulse tube cryocoolers. *Cryocoolers 14, edited by SD Miller and RG Ross, Jr*, pages 231–240, 2007.

Martin W Rupich, Xiaoping Li, Srivatsan Sathyamurthy, Cornelis LH Thieme, Kenneth DeMoranville, John Gannon, and Steven Fleshler. Second generation wire development at amsc. *IEEE transactions on applied superconductivity*, 23 (3):6601205–6601205, 2013.

Toshimi Satoh, Atsushi Onishi, Rui Li, Hiroshi Asami, and Yoshiaki Kanazawa. Development of 1.5w 4k gm cryocooler with magnetic regenerator material. In *Advances in cryogenic engineering*, pages 1631–1637. Springer, 1996.

V Selvamanickam, Y Chen, I Kesgin, A Guevara, T Shi, Y Yao, Y Qiao, Y Zhang, G Majkic, G Carota, et al. Progress in performance improvement and new research areas for cost reduction of 2g hts wires. *IEEE Transactions on Applied Superconductivity*, 21(3):3049–3054, 2011.

Sunpower. Cryotel gt. Technical report, 2012. URL `http://sunpowerinc.com/cryocoolers/cryotel-family/gt/`.

SuperPower Inc. Superpower 2g hts wire specifications. Technical report, 2014. URL `http://www.superpower-inc.com/system/files/SP_2G+Wire+Spec+Sheet_2014_web_v1.pdf`.

Matthis Thorade and Ali Saadat. Helmholtzmedia - a fluid properties library. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 63–70. Linköping University Electronic Press, 2012.

R Wesche and Am M Fuchs. Design of superconducting current leads. *Cryogenics*, 34(2):145–154, 1994.

M Xu, H Takayama, and K Nakano. Development of high efficiency 4 k two-stage pulse tube cryocooler. Georgia Institute of Technology, 2008.

Roberto Zanino, Roberto Bonifetto, Laura Savoldi Richard, and Francesco Casella. Dynamic modeling of a supercritical helium closed loop with the 4c code. In *AIP Conference Proceedings*, volume 1434, pages 1743–1750. AIP, 2012.

Roberto Zanino, Roberto Bonifetto, C Hoa, and L Savoldi Richard. Verification of the predictive capabilities of the 4c code cryogenic circuit model. In *AIP conference Proceedings*, volume 1573, pages 1586–1593. AIP, 2014.

Robust Modeling of Directed Thermofluid Flows in Complex Networks

Dirk Zimmer Daniel Bender Alexander Pollok

Institute of System Dynamics and Control, DLR German Aerospace Center
{Dirk.Zimmer,Daniel.Bender,Alexander.Pollok}@dlr.de

Abstract

A new concept is presented on how to set up the equations of complex fluid networks. This concept avoids the creation of large non-linear equation systems and hence leads to a very robust modeling approach while still being competitive in levels of performance. The concept has been implemented in Modelica and tested for the rapid pre-design of aircraft environmental control systems.

Keywords: fluid systems, thermal systems

1 Motivation

Modelica has established itself as a valuable tool for the modeling of thermal fluid systems. Typical applications are the modeling of power plants (Casella 2005), building simulation (Wetter, 2016), or, as in our case, the pre-design of environmental control systems (ECS) for future aircraft (Schlabe, 2014; Sielemann, 2011).

To support these activities, several quasi-standards have been developed: a stream connector (Franke *et al*, 2009B) has been included in the Modelica language standard and a corresponding standard library supports the modeling of fluids. (Franke *et al,* 2009A) Furthermore, the Modelica.Media library (Casella, 2006) provides models for a multitude of different fluid media, so that the same fluid models can be applied to different media.

Yet despite these advances, there still remain reoccurring problems that make the application for the end user challenging. Most of them involve the solvability of (larger) non-linear equation systems. Every so often, initialization or simulation of the fluid systems fails for reasons that are hard to detect for a non-specialist. From the end-user perspective, this is perceived as a lack of robustness significantly slowing down development time of new fluid architectures.

Also here, several attempts such as homotopy (Casella 2011) have been undertaken to address this problem but so far with mixed or limited success.

This paper presents a new approach to model fluid systems that avoids the creation of large non-linear equation systems in the first place. This leads to a very robust fluid library, and also high performing and scalable models.

2 The inertial pressure

In order to understand this approach, let us examine the root of the problem. What leads to the creation of large non-linear equation systems?

Whereas a smaller non-linear equation system may occur within a component (such as a heat exchanger), larger non-linear systems are created by a network of such components. Especially critical are branches, by-passes and loops. Whenever fluid flows join, a (quasi-) static analysis will require an equivalence of pressure for each involved junction. In order to fulfill this equivalence, the corresponding mass-flows become part of a non-linear equation system.

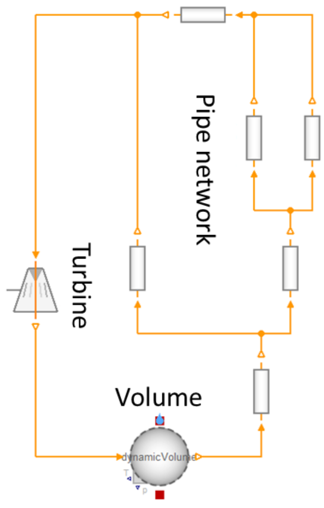


Figure 1: Simple fluid network

Figure 1 provides a simple example of a loop with two nested side-branches, leading to a non-linear equation system for the mass-flows. If modeled using conventional components and the stream connector, a system of 63 non-linear equations results that can be reduced to 8 iteration variables. More complex networks such as in (Zimmer, 2013) require more than 40 iteration variables. It is then a priori unclear how many solutions there are and whether a generic non-linear equation solver will find any of them (and which one), especially at (re-)initialization.

In order to increase robustness, we shall hence not rely on a generic solver but rather provide differential equations that lead to the desired equivalence.

Fortunately, the laws of physics offer a very favorable way to formulate this. To this end, let us review the fundamental equations of motion. Newton's second law can be formulated in terms of pressure gradient and density. This is the most basic part and form of the Euler equations for incompressible fluid flows, disregarding any external forces (such as gravity, friction, etc.):

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\,\nabla p$$

with $\boldsymbol{u}$ being the velocity, p the pressure and ρ the density. Since we work in an Eulerian framework where particles move through a parcel, we shall expand the material derivative:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\,\nabla p$$

Since we concern ourselves here only with one-dimensional flows, we can form a scalar PDE where v expresses the flow velocity $\mathbf{u}$ in direction of s:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial s} = -\frac{1}{\rho}\frac{\partial p}{\partial s}$$

Using a spatial element (such as a pipe section) of the length Δs and cross-section area A, this PDE is transformed into an ODE:

$$\frac{dv}{dt} + v\frac{\Delta v}{\Delta s} = -\frac{1}{\rho}\frac{\Delta p}{\Delta s}$$

Multiplying with $\rho\Delta s$ and substituting v in the first term with $\dot{m}/\rho$A yields finally

$$\underbrace{\frac{d\dot{m}}{dt}\frac{\Delta s}{\mathrm{A}}}_{\Delta r} + \underbrace{\rho\, v\Delta v}_{\Delta q} = \Delta p$$

In this way, the pressure difference Δp is decomposing into Δr and Δq. Δq is expressing the change in dynamic pressure for a given mass-flow, a well-established and frequently used quantity. The term expressed by Δr is all too often neglected. It expresses the inertial pressure to change the mass flow rate.

Alternative derivations or explanations of this inertial component in the pressure gradient can be found in (Truckenbrodt, 1983) and in (Longwell, 1966) for line flows of fluids and particles. These build upon a stream of fluid or particles through a pipe. For illustration let us look at a straight pipe.

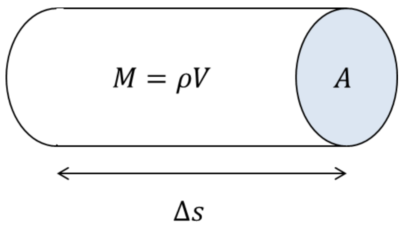


Figure 2: mass flow through a pipe

Figure 2 displays a section through a straight pipe. Its content corresponds to a mass M and its acceleration a will require a force f

$$f = \Delta r A = Ma$$

Assuming that the flow through such a pipe fulfills the mass balance so that inflow equals outflow, we can express the acceleration in terms of volume flow:

$$\Delta r A = V\rho\frac{1}{A}\frac{d\dot{V}}{dt}$$

Dividing the volume by A yields Δs and multiplying the volume flow by the density yields the mass flow.

$$\Delta r A = \Delta s\frac{dm}{dt}$$

or

$$\Delta r = \frac{d\dot{m}}{dt}\frac{\Delta \mathrm{s}}{A}$$

Again, we retrieve the formula for Δr. Let us note that it is a simple and linear equation that is independent of the thermodynamic state of the fluid and whose parameters remain constant for pipes with non-changing geometry. This will become very useful.

The reason Δr is so often neglected is that it is often very small and does not influence the thermodynamic state significantly (see Section 5). However, although mostly negligible, it is still a very useful quantity to compute changes in the mass-flow from occurring pressure differences. Using it, we can describe a dynamic that leads to pressure balance even in complex fluid networks. To this end, we shall revisit the equations for fluid junctions.

Prerequisite of this approach is that we work with fluid flows that uphold the mass flow balance $\dot{m}_{in} = \dot{m}_{out}$, meaning that the dynamics of compression are regarded as irrelevant for the fluid transport and the medium moves like being incompressible. However, the medium is allowed to change its thermodynamic state, even its density, when being subject to thermo-dynamic manipulation such as in a compressor, turbine, heat-exchanger, etc. Hence, we call this approach quasi-incompressible because the assumption of incompressibility (actually, mass flow balance) only affects the transport dynamics. In case transport phenomena of compressible flows are relevant, other approaches such as (Sielemann, 2012) have to be used.

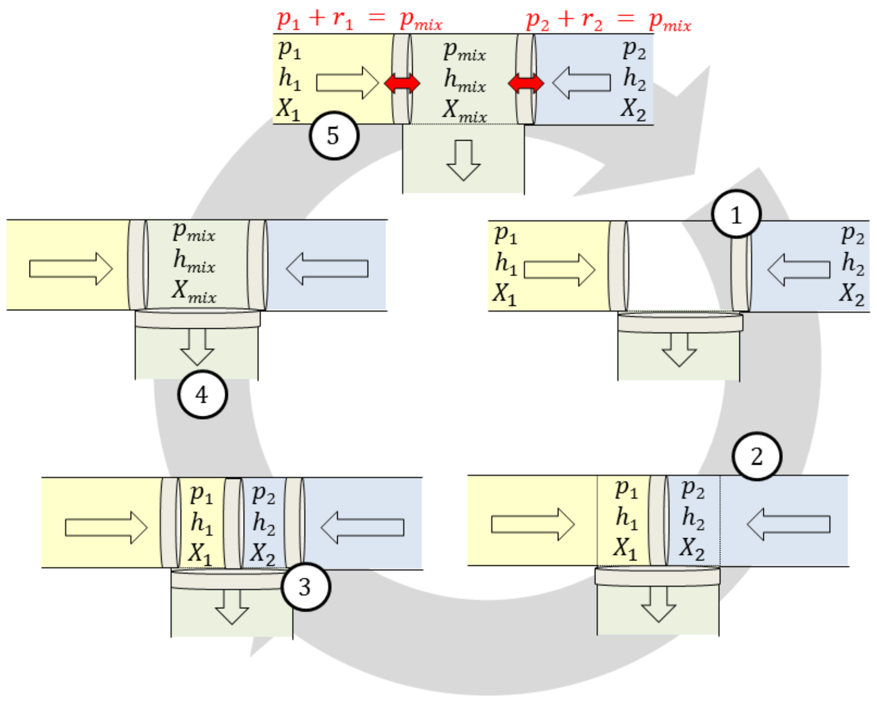


Figure 3: Illustration of the thought experiment for the mixing of two fluids

3 Pressure balance

By using this decomposition of the pressure gradient, we can revisit the model for junctions of a fluid flow. Let the pressure $\hat{p}$ be composed of the inertial pressure r and the pressure p for the case of static mass-flow:

$$\hat{p} = r + p$$

Pressure in general represents a sum of forces per area. These forces may result from macroscopic motion such as acceleration or from the microscope state of the medium to make place for its volume. Since we attribute the macroscopic acceleration of the mass flow already to r, p is now interpreted as resulting from the microscopic state that is bound to its specific volume and transported with the fluid. Hence p can here be treated similar to a specific quantity and is consequently also subject to mixing. We can hence formulate a mixing law for p:

$$p_{out} = f(p_{in,1}, h_{in,1}, \dot{m}_{in,1}, \dots, p_{in,2}, h_{in,2}, \dot{m}_{in,2}, \dots)$$

To compute this specific mixing pressure of two fluid flows, we shall conduct the following thought experiment. We fill a chamber with an amount of each fluid that is proportional to its mass-flow. The chamber is sealed on both ends and the two fluids a separated by a virtual disc. As soon as we remove this disc, mixing of the two fluids takes place and a new pressure establishes. Figure 3 illustrates this thought experiment in its stages 1 to 4.

In general, this leads to a non-linear mixing law. In case, the specific gas constant for both sections of the chamber is equal, a linear law can be used:

$$p_{out} = p_{mix} = \frac{\sum \dot{m}_i p_i}{\sum \dot{m}_i + \epsilon}$$

This linear law can also be used as approximation for the non-linear law when the exact pressure dynamics are not regarded as important during mass-flow transients or the pressure differences are low. Although we prefer to use a linear law for the sake of simplicity, it does not harm the overall equation structure when a non-linear law is used.

To get from a single stage mixing to a continuous process, let us look at stage 5 in Figure 3 that completes the circular process. We recognize that the mixing chamber has now a pressure p_{mix} that is different from both p_1 and p_2. This pressure difference occurs across the sealing virtual discs at both ends. If we now unhinge these discs, we shall attribute this pressure difference to the inertial pressure r.

Hence the pressure balance at a junction can be formulated for any inflowing stream i or outflowing stream j in terms of $\hat{p}$:

$$\hat{p}_i = \hat{p}_{mix}$$

or

$$p_i + r_i = p_j + r_j$$

4 Computing an unidirectional network

The decomposition of $\hat{p}$ into p and r does not only affect the pressure balance at a junction but it restructures the overall equation system of the network into a very favorable form. This is best explained by means of an example. Figure 4 represents a simple fluid network structure where a split (A) fluid stream of given total mass flow rate rejoins (B). On each branch there is a simple black box component that manipulates the thermodynamic state of the fluid by algebraic equations in an arbitrary way. For each section of each branch, the law for the inertial pressure applies:

$$\frac{d\dot{m}}{dt}\frac{\Delta s}{A} = \Delta r$$

This law is applied to all thermofluid components and causes all mass-flows of the system to become potential state variables. This means that if the thermodynamic state of the inlet is defined, each component can compute the thermodynamic state of the outlet in a straight forward manner. All required variables represent knowns. This forward computation is symbolized by the black variables in Figure 4. All of them are computed simply from source to sink.

Given these variables, we can now compute the inertial pressure r and the mass-flow dynamics. We start at the boundaries. For each source, r is stipulated to be 0. At each sink, $\hat{p}$ equals the desired outlet pressure. The pressure balance law is applied for junction B whereas the split at junction A defines equality of its inertial pressures. The black box components contain the law for the inertial pressure.

In total, we can setup the following equation system for the mass-flow dynamics:

$$\frac{d\dot{m}_1}{dt}\frac{\Delta s_1}{A} = r_1 - r_A$$

$$\frac{d\dot{m}_2}{dt}\frac{\Delta s_2}{A} = r_2 - r_A$$

$$p_1 + r_1 = p_2 + r_2$$

$$\frac{d\dot{m}_1}{dt} = -\frac{d\dot{m}_2}{dt}$$

The last equation thereby results from the index reduction of the system: it represents the time-derivative of the mass-flow constraint $\dot{m}_1 + \dot{m}_2 = \dot{m}_0$ with $\dot{m}_0$ being given. We can rewrite the equations above as linear equation system:

$$\begin{bmatrix} -1 & & \Delta s_1/A & \\ & -1 & & \Delta s_2/A \\ 1 & -1 & & \\ & & 1 & -1 \end{bmatrix}\begin{bmatrix} r_1 \\ r_2 \\ d\dot{m}_1/dt \\ d\dot{m}_2/dt \end{bmatrix} = \begin{bmatrix} 0 - r_A \\ 0 - r_A \\ p_2 - p_1 \\ 0 \end{bmatrix}$$

The resulting equation system is not only linear. The matrix elements are either integers or describe the pipe geometry and hence likely form invariants with respect to simulation time. This means that the equation system can be inverted upfront and only occurs as simple matrix-vector multiplication during simulation time. When implementing this in Modelica, a tool like Dymola (Brück, 2002) is smart enough to do exactly this.

We can generalize the lessons from the example in Figure 4 and derive general statements for the resulting structure of the equation system of any network of fluid systems. To this end, we have to postulate two underlying modeling assumptions that have to be fulfilled by the modeler:

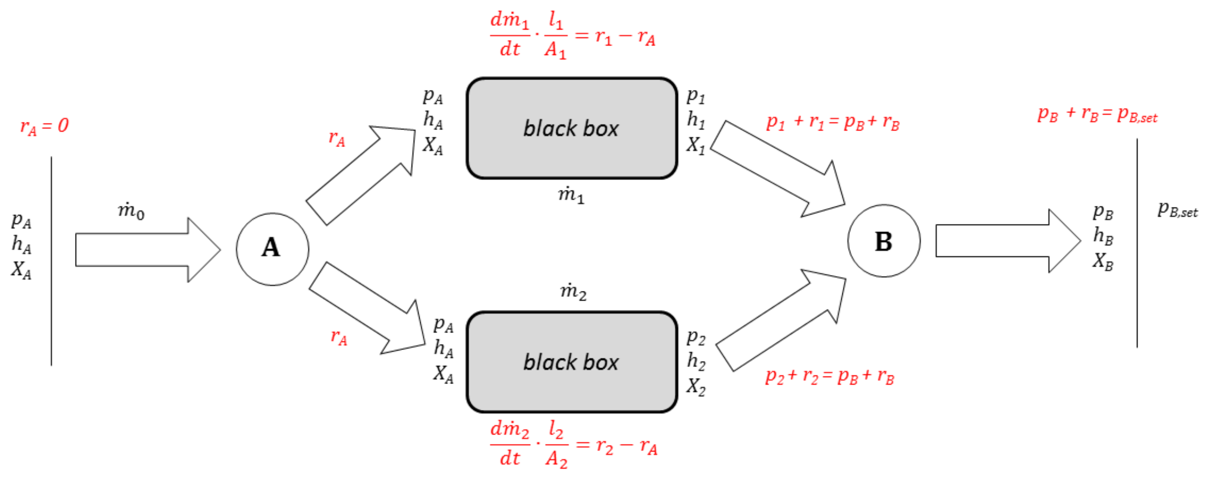


Figure 4: Computation of a simple directed fluid flow. The black terms represent a straight forward computation from source to sink. The red equations describe the mass-flow dynamics and form a linear system of equations.

- The directed fluid flow gives rise to a partial order of its fluid connectors along the flow. (This means that no circular flows must occur unless these are cut by a volume(-like) element)
- There is no direct algebraic coupling between two flows within a component. (This means in practice that on the component level differential equations may be employed instead of pure algebraic equations)

If these two assumptions are upheld by the modeler, then any fluid system will be represented by a structure incidence matrix (Cellier, 2006) of the following block-lower-triangular (BLT) form in Figure 5:

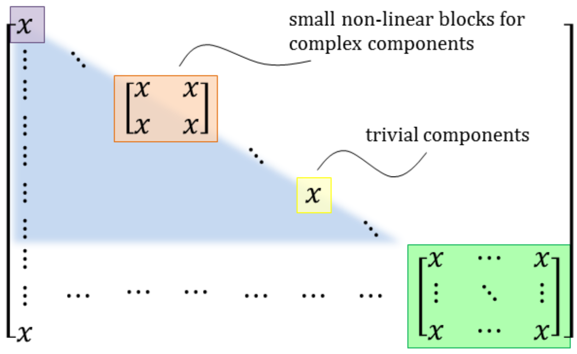


Figure 5: Structure incidence matrix in BLT Form

The blue part of this incidence matrix may represent a straight-forward computation of non-linear equations. There might be small non-linear equations systems stemming from the components but they do not grow in size when connecting these components. This means that if the solvability has been proven on the component level, it will remain solvable on the system level.

The only large equation system is marked by the green part. This is however a linear equation system with constant coefficients that can be inverted upfront. The size of this system is thereby proportional to the number of different branches in the fluid network. It is, thanks to index-reduction, invariant to the length or complexity of each branch.

Given this BLT form, it is now clear how high robustness for the end user is achieved. No large non-linear equation system will be created by building a complex network out of its components. If the components are very robust then the total system will be as well.

5 Validity of the approach

Although the above structure is very favorable it also contains a little shortcut since we have used p for the computation of the thermodynamic state and not $\hat{p}$. This means that the gradient in the thermodynamic state between junctions is not properly taken into account. It is clear that this is irrelevant for the mass-flow static case (with all $r = 0$) but what about the transient behavior?

To estimate any error, we are interested in the fraction of $\Delta r/p$. For an ideal gas flowing through a parcel of length Δs, we can do a simple analysis. Formulating the law for the inertial pressure in terms of velocity and not in terms of mass flow yields:

$$\Delta s \, \rho \frac{dv}{dt} = \Delta r$$

Using the speed of sound $c^2 = \kappa \, p/\rho$, we can express the quotient $\Delta r/p$ by:

$$\frac{\Delta r}{p} = \kappa \frac{d\frac{v}{c}}{dt} \frac{\Delta s}{c}$$

For $\Delta r/p$ to become significant, the acceleration must be expressed in Mach per second and/or the pipe length in sound-seconds. For instance, for the quotient to become roughly 10%, one must accelerate air in a (frictionless) pipe of 100 meters with 10 times the gravitational acceleration. For many typical applications, such values represent very high numbers and the error can be tolerated.

For incompressible fluids the speed of sound is determined by $c^2 = \mathrm{K}/\rho$ and the simple statement from above does not hold. Strong (mass-flow) accelerations can indeed lead here to a shift of evaporation regions or even cavitation. Nevertheless, in many cases, this can be neglected as well. In some cases such as a water hammer, strong acceleration does occur but the thermodynamic state is rather insensitive to pressure and hence the approach remains applicable (although with care).

In cases where transient effects become relevant there is a multitude of potential work-arounds that shall be omitted here in order to be concise. For our application field, this is however irrelevant.

It also shall be noted that many existing fluid libraries choose to ignore the inertial pressure completely. This is just doing a (well-accepted) error in the different direction. Important is just that the modeler understands the underlying assumptions.

6 Implementation in Modelica

The design of a suitable connector is here the most fundamental decision. The inertial pressure `r` and the mass-flow `m_flow` hereby constitute a classic pair of potential and flow variable. All the remaining variables, such as mass-flow static pressure `p`, specific enthalpy `h`, etc., are then regarded as specific quantities and transmitted via input and output connectors:

```
connector Inlet
 replaceable package Medium = Modelica.
     Media.Interfaces.PartialMedium;
 Modelica.SIunits.Pressure r;
 flow Medium.MassFlowRate m_flow;
 input Medium.AbsolutePressure p;
 input Medium.SpecificEnthalpy h;
 input Medium.MassFraction Xi[Medium.nXi];
end Inlet;
```

respectively:

```
connector Outlet
 replaceable package Medium = Modelica.
     Media.Interfaces.PartialMedium;
 Modelica.SIunits.Pressure r;
 flow Medium.MassFlowRate m_flow;
 output Medium.AbsolutePressure p;
 output Medium.SpecificEnthalpy h;
 output Medium.MassFraction Xi[Medium.nXi];
end Outlet;
```

An alternative design is to use the thermodynamic state record of the Modelica.Media Library directly in the connector instead of the variables $(p, h, \ldots)$. This can lead to a better performing solution but maybe slightly more bulky equations.

Also one can artificially enrich the connector by additional signals in order to ensure that there are no open ends or unwanted loops. For instance, an integer input/output signal in reverse flow direction counting upwards from sink to source, would lead to errors in case of open endings or unwanted loops. So far, such mechanism have however shown to be unnecessary.

Using this connector, we implemented the proprietary HEXHEX library for aircraft environmental control and cooling systems in an early design phase. It contains components for pumps, fans, compressors, turbines, heat-exchangers, etc. Boundary models enable to set the environmental conditions. A global world model enables to set common default values. Although similar in its look to the DENECS library (Sielemann, 2011) it represents a complete new implementation.

When implementing the components, robustness must be thoroughly tested. The components must not become singular at zero-mass flow and should continue to exhibit a plausible behavior even when being used outside their range of validity.

Different from the standard fluid library, junctions now demand for an extra model. There are classic T-Junctions, X-Junctions and 1-to-N Junctions. Although this could be regarded as additional burden, there were no complaints from the user base. The basic law:

$$\frac{d\dot{m}}{dt}\frac{\Delta s}{A} = \Delta r$$

for the inertial pressure of Section 2 is now present in every single component (that is not a boundary). Hence any classic two-port component extends this law from a partial base class, where this law has been implemented:

```
partial model TwoPort
  Inlet portA;
  Outlet portB;
  parameter SIunits.Area A = world.A
  parameter SIunits.Length L = world.L
  SIunits.MassFlowRate m_flow( start=0);
  SIunits.Pressure dr;

equation
  0 = portA.m_flow + portB.m_flow;
  m_flow = portA.m_flow;
  portA.r - portB.r = dr;
  dr = der(m_flow)*L/A;
end TwoPort;
```

The pressure balance of Section 3 is then implemented for instance in a junction model where two inflows `portA`, `portB` meet and result in an outflow `portC`:

```
model Junction
  Inlet portA;
  Inlet portB;
  Outlet portC;
equation
  m_flowA = portA.m_flow;
  m_flowB = portB.m_flow;
  m_flowC = portC.m_flow;
  m_flowA + m_flowB + m_flowC =0;

  portC.Xi = (portA.Xi*(m_flowA + eps)
             + portB.Xi*(m_flowB + eps))
             /(m_flowA+m_flowB+2*eps);
  portC.h = (portA.h*(m_flowA + eps)
            + portB.h*(m_flowB + eps))
            /(m_flowA+m_flowB+2*eps);
  portC.p = (portA.p*(m_flowA + eps)
            + portB.p*(m_flowB + eps))
            /(m_flowA + m_flowB + 2*eps);

  portC.p + portC.r = portA.p + portA.r;
  portC.p + portC.r = portB.p + portB.r;

end Junction;
```

In this junction model, the simple linear approximation for the resulting mixing pressure is applied. Also a small regularization is applied to cover the case of zero mass flow rate. This is especially helpful for initialization when the system is ramping up from zero mass flow.

Aside from a few basic classes such as the TwoPort component from above, the library avoids the overuse of inheritance since multiple inheritance levels have shown to be detrimental to the readability of the code (Pollok, 2016).

Because also the stream connector is avoided, the thermodynamic equations of components like heat exchangers, turbines, fans, etc. can be written in a straight forward way. This is greatly enhancing the readability of each component code. In this way, the code becomes easy to understand even by the non-expert. Also the development time of individual components has been reduced by an approximate of 50%.

The library has undergone a significant testing effort by external users for the rapid pre-design of a large variety of aircraft environmental control systems. We estimate that the development time of such architectures has been reduced roughly by at least 80%. One particular architecture is presented in the next section in order to show the application of the HEXHEX library.

7 Exemplary Use Case

To demonstrate the feasibility of HEXHEX, a complex electric architecture for an aircraft environmental control system has been modeled.

Figure 6 shows the diagram layer of the modelled electric driven vapour cycle pack (eVCP) architecture. The architecture is derived from a patent publication (Golle, 2016). Unlike the original architecture, the vapour cycle was simplified. The original vapour cycle has an additional evaporator connected to recirculated air from the cabin. Unlike conventional bleed air driven air cycle packs (Bender, 2017) unconditioned outside air instead of bleed air from the engine enters the eVCP. The cold and low pressure air is compressed in a first stage before it passes the primary heat exchanger (PHX). A second compressor further raises the pressure and temperature before entering the reheater. The main heat exchanger, mounted in the ram air channel, is passed before the evaporator cools the air. In case the saturation temperature is exceeded, a water separator extracts the condensate and leads it to the water injector located in the the ram air channel upstream the vapour cycle condenser. Before the conditioned fresh air is expanded in the turbine, it is reheated in the reheater. The discharged fresh air meets the recirculated cabin air in the mixing unit downstream the pack. The ram air channel functions as heat sink and is feed with air from the outside. During ground operations, the air flow is provided by a fan

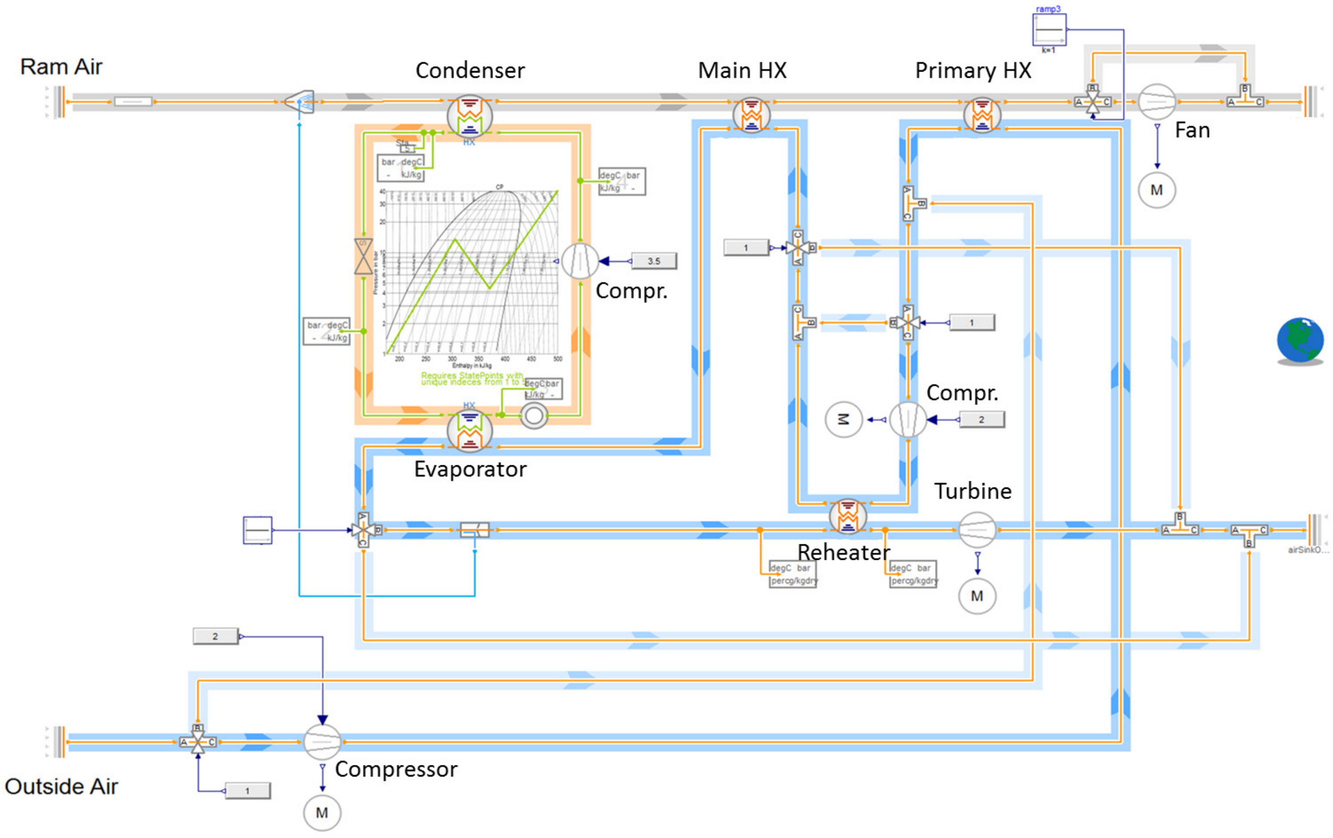


Figure 6: Modelica diagram of electric driven vapour cycle pack architecture using the HEXHEX library.

located near the ram air outlet. Contrary to a bleed air driven pack which is autonomously driven, all turbomachines within the eVCP are electrically powered.

The here presented modelling concept allows to build up such a system model as shown in Figure 6 within one day including first working simulations and a control concept for its many bypasses. Beside reasonable values for operational settings, i.e. compression/expansion ratios, flow resistances, and boundary conditions, no further settings concerning initialization have to be defined. Variants of such an architecture can then be generated much faster.

Looking at the statistics of the translated model, the advantage of the proposed concept becomes apparent: There is no large non-linear equation system at all. From the components there remain only 19 smaller non-linear equation systems with a single iteration variable.

In total the system contains 40 continuous time states. 9 of them represent mass-flows according to the proposed dynamics. 11 states originate from in-built controllers and the remaining states belong to the components. Most of them originate from quite detailed models of the evaporator and condenser within the vapour cycle.

The addition of the 9 state variables for the mass-flow by this concept do not cause any oscillation in the system. The impact on simulation performance has shown not to be detrimental.

The architecture of Figure 6 shows a number of junction valves that can open or close various different bypasses. Figure 7 displays the opening of a generic bypass valve with the resulting mass-flow dynamics and the corresponding inertial pressure. As outlined before, the inertial pressure is zero as long as the mass-flow rates remain constant. The opening of the valve then causes a pressure change and consequently the inertial pressure becomes non-zero. The affected mass flow rates then change until the new equilibrium point is reached. Evidently, a well-natured transient behavior can be observed that can be well-handled by numerical ODE solvers, especially by those suited for stiff-systems.

Although the valve is fully opened within a single second, the maximum peak for the absolute value of the inertial pressure remains around 100 Pascal. Being around one per mille of the fluid's static pressure, its effect on the thermodynamic state is negligible, which goes well in line with the analysis of Section 5.

If desired, the mass flow rate can also be stipulated by the addition of corresponding boundary conditions to the system. However, when doing so, these boundary conditions of mass flow rates must be differentiable with respect to time. Otherwise, the inertial pressure cannot be properly determined. Also one must take care that such boundary conditions do not conflict each other and cause the system to be overdetermined.

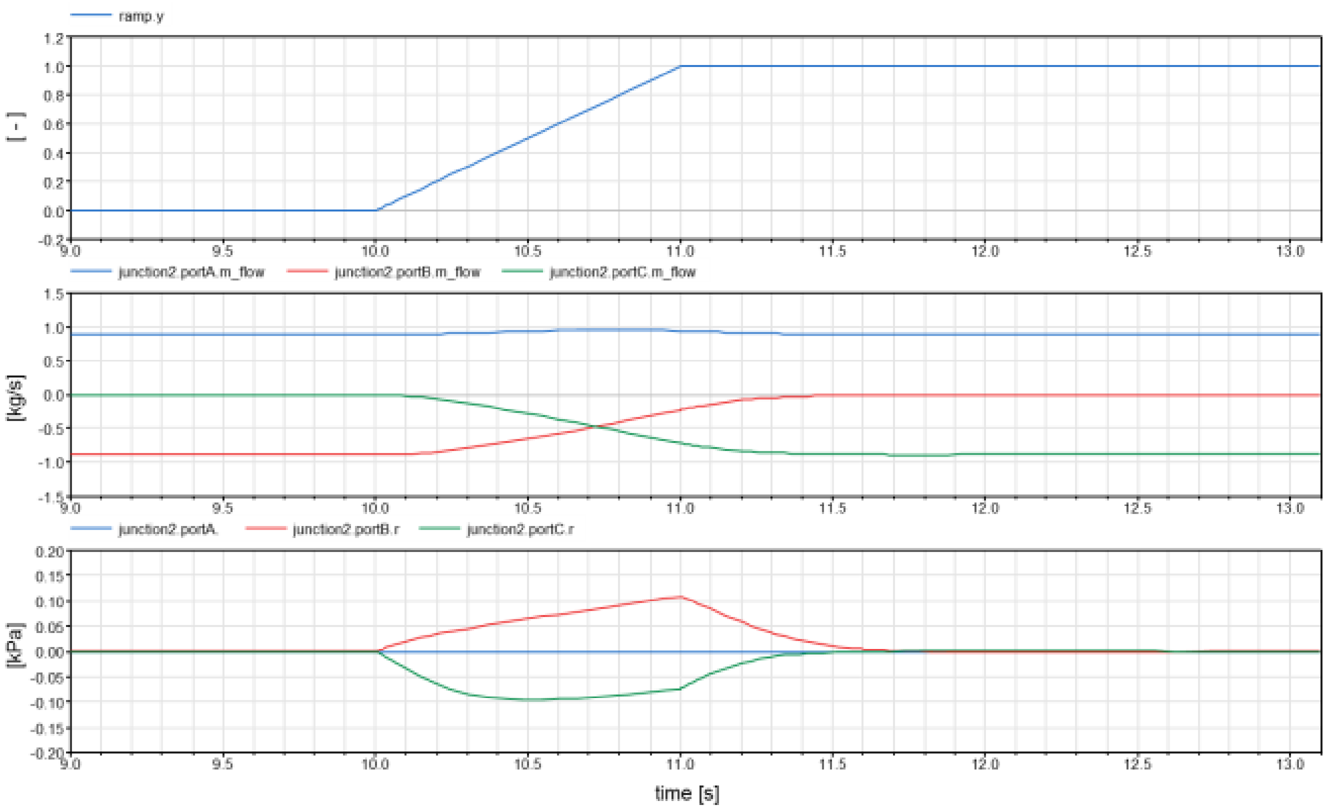


Figure 7: Mass flow dynamics for an opening valve

8 Final Remarks

8.1 Positioning of the approach

How does the approach of HEXHEX compare to the most common other approaches? Within the Modelica community, we find two main approaches.

The first approach is a purely algebraic quasi-static modeling of fluid systems. This approach may avoid any states but yields large non-linear equation systems. Most volume free components in the Modelica Standard Fluid library (Franke, 2009A) are modeled in this style.

The second approach is a realization of the finite volume method (or similar) where any flow represents the flow between two volume models. This avoids any larger non-linear equation system but at the price of creating many state-variables because of its many volumes. A classic example is (El Hefni, 2014)**.**

HEXHEX is right in the middle, looking for the sweet spot. It avoids completely any large non-linear equation system and hence robustness on the component level leads to robustness on the total system level. Yet, it creates only a small set of states. The set is much smaller than for a finite volume approach, since only the mass-flows are used as state variables and not the full thermodynamic state of a volume. Furthermore, the mass-flow can be shared for all serially connected flows. This enables index-reduction to further reduce the set of state variables.

A justified point of critique is that with the inertial pressure also fast dynamics enter the system asking for stiff-system solvers and that the mass-flow states for many modelers rather represent so-called artificial states (Zimmer, 2013). The answer to this critique is given in (Zimmer, 2013) and (Zimmer, 2014) that propose better ways how numerical ODE solvers shall deal with artificial states. Unfortunately, no Modelica tool currently supports such an approach. We hope that libraries like HEXHEX will further raise the value and importance of such a solution since after all a general support would also benefit the works of many others such as (Jorissen, 2018). Yet even without this support, the usability and performance of HEXHEX is not really impaired.

8.2 Initialization

A robust method for initialization is also important for the end-user. Ideally only the boundary conditions are defined and the components of the system do not require additional information for initialization. With HEXHEX such an approach is absolutely feasible. For many cases, an initialization at rest with all mass-flows being zero is a good starting condition. The system will then ramp-up according to the boundary condition. It is actually the same as plugging in (or switching on) the actual device.

In some cases, this ramping up may lead to a different state than expected. This typically indicates flaws in the actual system model. In any case, there will be a trajectory leading up to the issue, which allows for better diagnostics than just a failed initialization with a non-linear solver.

8.3 Adaption to Bidirectional Fluid flows

For our application field, a unidirectional solution is sufficient. Yet, can the approach be extended for bi-directional models? In principal yes, either one has to duplicate the equations for both flow directions or one takes use of the stream connector (Franke et al, 2009B). Both approaches will (in a different way) pollute the modeling equations to some degree but represent workable solutions.

In contrast to the standard fluid library, the built- in regularization scheme of the stream connector is not needed any longer since the mass-flow is a state variable.

A disadvantage of a generic bidirectional approach is that more loops will occur and hence more volume elements (or other means) will be needed to cut these loops. This is simply because there are statistically more loops in an undirected graph than in a directed graph of the same density. Hence while workable, the approach is losing some of its appeal for bidirectional flows.

8.4 Overall conclusion and future work

The dynamics of non-static fluid flows can very well be expressed using DAEs in Modelica, even for complex networks. If we regard the fluid streams as quasi-incompressible, then index-reduction enables to extract a small set of state variables for the total system.

Exploiting this fact enables a very robust modeling of complex fluid systems that frees the end-user from having to care about large non-linear equation systems and initialization.

The outlined approached has been implemented and thoroughly tested by a number of complex environmental control system architectures for aircraft and artificial testing examples. Development time of these architecture models could be drastically reduced compared to prior implementations.

Yet, this approach has not reached full maturity yet und the corresponding library is still subject to heavy development. Future work will hence more thoroughly compare this approach to others in terms of validity and performance. Furthermore, the release of an open interface standard shall be considered as soon as the development.

9 Acknowledgements

We would like to thank Airbus for providing incentive by their challenging problems and steady support by intensive testing. Last but not least we would like to thank Bibi Blocksberg as source of vital inspiration.

References

Daniel Bender, (2017) Integration of exergy analysis into model-based design and evaluation of aircraft environmental control systems, Energy, 137, p. 739-751, 2017

Dag Brück, Hilding Elmqvist, Hans Olsson, Dymola for Multi-engineering Modeling and Simulation. In: Proc. of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany

Francesco Casella and Alberto Leva (2005), Object-Oriented Modelling & Simulation of Power Plants with Modelica, Proceedings of the 44th IEEE Conference on Decision and Control, andthe European Control Conference, Seville, Spain

Francesco Casella, Martin Otter, Katrin Proelss, Christoph Richter, Hubertus Tummescheit, (2006) The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks, Proc. 5th International Modelica Conference, Vienna, Austria, Vol.2, pp.559-568.

Francesco Casella and Sielemann, Michael und Savoldelli, Luca (2011) Steady-state initialization of object-oriented thermo-fluid models for homotopy methods. In: Proceedings of 8th International Modelica Conference. 8th International Modelica Conference, 20.-22. Mrz 2011, Dresden.

Francois E. Cellier, Ernesto Kofman (2006). Continuous System Simulation, Springer Verlag, New Yorg, 643p.

Baligh El Hefni and Daniel Bouskela (2014), Dynamic modelling of a Condenser with the Thermo SysPro Library, In: Proceedings of the 10th International ModelicaConference, March 10-12, 2014, Lund, Sweden

Rüdiger Franke et. al. (2009A) Standardization of Thermo-Fluid Modeling in Modelica.Fluid. Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009

Rüdiger Franke et. al. (2009B) Stream Connectors – An Extension of Modelica for Device-Oriented Modeling of Convective Transport Phenomena. Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009

Steffen Golle et. Al., (2016), Betriebsphasenabhängig steuerbare Flugzeugklimaanlage und Verfahren zum Betreiben einer derartigen Flugzeugklimaanlage, Patent, DE102015207447A1, 27.10.2016

F. Jorissen, M. Wetter & L. Helsen (2018). Simplifications for hydronic system models in Modelica. In: Journal of Building Performance Simulation 12.01.2018

Longwell, P.A. (1966). Mechanics of Fluid Flow. McGraw-Hill.

Alexander Pollok, and Andreas Klöckner (2016) The use of Ockham's Razor in object-oriented modeling. In: Proceedings of the 7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Milano, Italy. DOI: 10.1145/2904081.2904086

Daniel Schlabe and Lienig, Jens (2014) Model-Based Thermal Management Functions for Aircraft Systems. In: SAE 2014 Aerospace Systems and Technology Conference. SAE 2014 Aerospace Systems and Technology Conference, 23.-25. Sept 2014, Cincinnati, Ohio, USA. ISSN 01487191

Michael Sielemann, T. Giese, B. Oehler, M. Gräble (2011) Optimization of an Unconventional Environmental Control System Architecture. In: SAE International Journal of Aerospace, 4 (2), Seiten 1263-1275. SAE 2011 AeroTech Congress and Exhibition, 18.-21. Oct. 2011, Toulouse, France.

Michael Sielemann (2012) High-Speed Compressible Flow and Gas Dynamics, In: Proceedings of the 9th International Modelica Conference, 3.-5. Sept. 2012, Munich, Germany

Erich Truckenbrodt, Lehrbuch der angewandten Fluidmechanik (Kapitel zu instantionäre Fadenströmung dichtbeständiger Fluide) Springer Verlag, 1983

Michael Wetter, Marco Bonvini and Thierry S. Nouidui (2016) Equation-based languages - A new paradigm for building energy modeling, simulation and optimization. Energy and Buildings, 117(1), p. 290-300, 2016

Dirk Zimmer, (2014), Handling infinitely fast processes in continuous system modeling, Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Berlin, Germany

Dirk Zimmer, (2013), Using Artificial States in Modeling Dynamic Systems: Turning Malpractice into Good Practice Proceedings of the 5th International Workshop on Equation-Based Object-Oriented Languages and Tools (EOOLT) , Nottingham, United Kingdom

Liquid Cooling Applications in Twin Builder

Anand Pitchaikani[1] Katrin Prölss[1] Mathias Strandberg[1] Hubertus Tummescheit[1] Sameer Kher[2]

[1]Modelon AB, Sweden, {anand.pitchaikani,katrin.prolss,mathias.strandberg, hubertus.tummescheit}@modelon.com

[2]Ansys INC, USA, sameer.kher@ansys.com

Abstract

This industrial paper introduces the new Liquid Cooling Library (LCL) that is available in ANSYS Twin Builder Heating and Cooling Library bundle. This library is a multi-tool compatible Modelica Library which can help customers to deploy simulation of cooling systems in a new way. In general, LCL can be used for modeling cooling circuits across many industries like aerospace, automotive and process industry. The combination with the extensive electronics and electrical machine libraries in Twin Builder makes LCL a natural fit for cooling of power electronics and thermal management of electric vehicles.

Keywords: Twin Builder, Liquid Cooling Library, Optimica Compiler Toolkit, Thermal Management, Electronics Cooling

1 Introduction

Modelica is one of the best and most efficient ways to express physical models that can be used to implement analytical model based systems engineering (MBSE) through modeling and simulation. ANSYS Twin Builder is a powerful platform for modeling, simulating and analyzing system-level digital prototypes. Twin Builder is used in model-based design workflows in automotive, aerospace, electronics, energy and industrial equipment segments to model and simulate multidomain systems. Modelon offers a comprehensive suite of multi-tool compatible libraries powered and built on the Modelica standard through the Modelon Library Suite. Modelon also provides the OPTIMICA Compiler Toolkit, the most advanced Modelica-based mathematical engine on the market through its Modelon Creator Suite. This combination of the leading library portfolio, Modelica compiler and powerful simulation platform has convinced ANSYS to market the Modelon thermal libraries to the industry through its tool Twin Builder. The industry can benefit from off-the-shelf components available for translating their system into models and simulate various real-life scenarios.

Twin Builder Heating and Cooling Library bundles together Modelon's Liquid Cooling Library (LCL), Heat Exchanger Library (HXL) and Vapor Cycle Library (VCL). This paper presents the LCL [4] that can be used to model wide variety of internal flow thermal management, hydraulic sizing and temperature control applications ranging from automotive and aerospace to industrial equipment and process industry.

2 Twin Builder

ANSYS Twin Builder helps customers improve Predictive Maintenance outcomes - allowing one to save on warranty and insurance costs and optimize their product operations.

To easily and quickly build, validate and deploy digital twins - virtual replicas of a physical system - ANSYS Twin Builder provides a variety of capabilities. To build the twin, ANSYS Twin Builder combines the power of a multi-domain systems modeler with extensive 0-D application-specific libraries, 3D physics solvers and Reduced Order Models (ROM) capabilities along with embedded software development tools, allowing system developers to reuse existing components and quickly create a systems model of their product. To validate the system, ANSYS Twin Builder combines multi-domain systems simulation capabilities with rapid Human Machine Interfaces (HMI) prototyping, systems optimization and XiL validation tools, ensuring that the system design will perform as expected. And, to connect the twin to test or real-time data, ANSYS Twin Builder easily integrates with Industrial Internet of Things (IIoT) platforms and contains runtime deployment options, allowing customers to perform predictive maintenance on their physical product. ANSYS Twin Builder offers a packaged approach for our customers' digital twin strategy.

Twin Builder's built-in libraries provide a rich collection of components used to create complete system models. The models can be selected from multiple physical domains and multiple levels of fidelity to capture the desired system dynamics at an appropriate level of detail. Twin Builder models are easily parameterized to replicate physical component behavior. Twin Builder libraries include analog and power electronics components; control blocks and sensors; mechanical components; hydraulic components; digital and logic blocks; application-specific libraries for aerospace electrical networks, electric vehicles and power systems; and characterized manufacturers' components. Twin Builder supports the Modelica Standard Library and Modelica libraries offered by Modelon AB, including libraries for hydraulics, pneumatics, liquid cooling, heat exchangers and thermal power.

3 Liquid Cooling Library

LCL is used for modeling and simulation of liquid cooling systems. Both standard and non-standard cooling circuits can be modeled with ease as shown in *Figure 1*. In particular, *Figure 1* illustrates a cooling circuit model of an internal combustion engine. The closed cooling circuits are handled well with this library. It includes more than 80 internal flow components such as pipes, bends and junctions. These components are neatly organized into different packages. These packages as shown in *Figure 2* provide high performance modeling of incompressible flows within closed circuits making the library suitable for real time as well as desktop applications. The usage of incompressible fluid flow assumptions in the library helps in reaching high performance while simulating complex cooling circuits. The library has suitable numerical operators applied in important physics like mass flow conservation and energy balance to make real time computation possible. These operators can be switched ON or OFF by the users.

3.1 Library Content & Structure

As shown in *Figure 2*, the library is divided into several sub-packages for accommodating different component types. In each package, components are readily available for use in system models. In addition to the component packages, there is an Experiments package, which contains several examples of system models that can be simulated. These models should give an idea of how the component models in the library can be used. Also found on the top level of the library is the Media package, which contain definitions of medium property models for coolant.

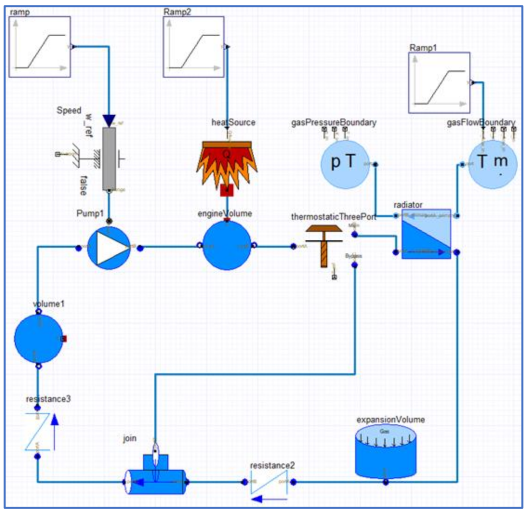
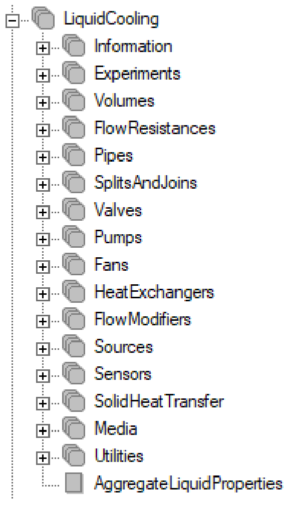


Figure 1. System model built in Twin Builder



LiquidCooling
- Information
- Experiments
- Volumes
- FlowResistances
- Pipes
- SplitsAndJoins
- Valves
- Pumps
- Fans
- HeatExchangers
- FlowModifiers
- Sources
- Sensors
- SolidHeatTransfer
- Media
- Utilities
- AggregateLiquidProperties

Figure 2. Modelon Liquid Cooling Library Structure

Volumes, flow resistances, pipes, splits and joins and pumps are the packages from which the user can drag and drop any components of interest to build the system model of his choice. Heat exchanger models are available in LCL with the assumption of homogeneous conditions of the inlet and outlet flows and with no phase transitions of the fluids. Simple models of heat exchanger stacks are also available in the library up to 8 heat exchangers per stack.

3.2 Key Assumptions, Features & Capabilities

All liquid models in the library are incompressible. More specifically that means that:

- The liquid density is independent of the pressure
- Volume components have quasi-static mass balances

- The liquid specific enthalpy is independent of pressure

Compared to compressible fluids there are some fundamental differences when assuming incompressible properties:

- Temperature is the only state variable in control volumes.
- Because of the quasi-static mass balances, the mass flow rate in branches are always the same in all components of one branch.
- For an incompressible fluid system, the mass flow rates in each branch must be given directly by boundary conditions and the state variables.

All the above assumptions make sure that models provides high performance for the system simulation performed by the user. LCL comes with a large set of fluid component models like volumes, a large collection of fluid resistances, pipes, splits & joins, valves, pumps, fans, heat exchangers and fluid properties as shown in *Figure 3*.The fluid resistance elements are based on the complete set of data from reference [1], the de-facto standard guide for the design and analysis of thermos-fluid systems and components. The fluid property models include:

- Water
- Aqueous solutions of glycol, alcohols, glycerol, ammonia, chlorides and salts
- Jet fuels and motor oil

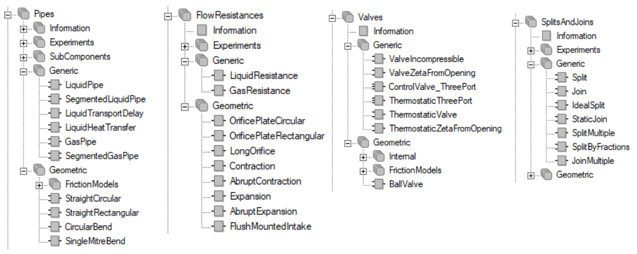
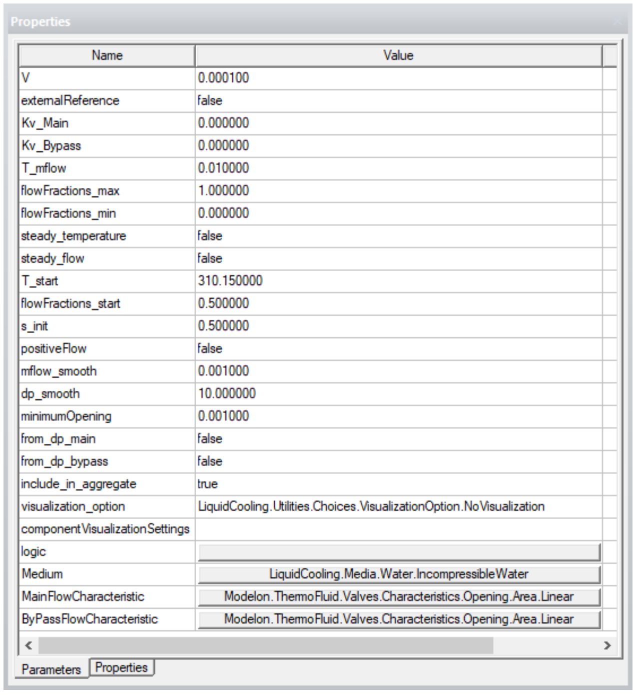


Figure 3. Large set of fluid component models in LCL

All component models come with broad parameterization possibilities. The user is provided with vital geometry, characterization and fluid related parameters through the parameter tab in Twin Builder as shown in *Figure 4*.

It is also possible to study the propagation of trace components in liquid flow networks or circuits. A trace component is a species diluted in the liquid in such low concentration that it can be assumed not to affect the thermodynamic or transport properties. This will be helpful in studying contaminations in the coolant.



Name	Value
V	0.000100
externalReference	false
Kv_Main	0.000000
Kv_Bypass	0.000000
T_mflow	0.010000
flowFractions_max	1.000000
flowFractions_min	0.000000
steady_temperature	false
steady_flow	false
T_start	310.150000
flowFractions_start	0.500000
s_init	0.500000
positiveFlow	false
mflow_smooth	0.001000
dp_smooth	10.000000
minimumOpening	0.001000
from_dp_main	false
from_dp_bypass	false
include_in_aggregate	true
visualization_option	LiquidCooling.Utilities.Choices.VisualizationOption.NoVisualization
componentVisualizationSettings	
logic	
Medium	LiquidCooling.Media.Water.IncompressibleWater
MainFlowCharacteristic	Modelon.ThermoFluid.Valves.Characteristics.Opening.Area.Linear
ByPassFlowCharacteristic	Modelon.ThermoFluid.Valves.Characteristics.Opening.Area.Linear

Figure 4. Typical parameter tab in Twin Builder

In the Twin Builder environment, cooling system models can be created using a drag and drop approach with very little effort from the user. The capabilities of LCL makes it an ideal candidate for playing a vital role in performing transient system analysis in the following cases:

- Cooling systems for automotive, aerospace, industrial equipment and process industries
- Engine cooling
- Lubrication circuit design
- Battery thermal management
- Component selection
- Pump dimensioning
- Design and analysis of non-standard cooling solutions
- Support of control system development and evaluation

4 Applications / Use cases

The LCL is an important part of libraries required for making vehicle thermal management models for automobiles as shown in [2], [3] and [5]. This can cover a range of vehicles from conventional to electric and hybrid electric vehicles. The cooling system of an internal combustion engine or a traction motor can be created using library's off-the-shelf components as shown in *Figure 1*. *Figure 1* is a dynamic model of a liquid cooling circuit of an internal combustion engine. The coolant flow is driven by a pump incorporating a table-based pump characteristic curve. The external heat load from the engine is approximated by a ramp input signal of heat flow rate. A radiator with a thermostatic bypass valve cools down the liquid coolant. For the radiator, the heat exchanger effectiveness is mapped directly from the mass flow rates of both air flow and coolant flow using a look-up table.

The components as well as the fluids in the Liquid Cooling Library are adapted for the development of cooling systems for power electronics and machines. The library has successfully been used for the analysis of cooling systems in the power electronics of wind turbines. A great number of applications, including engine design in the aerospace industry, require the fuel to act as coolant. In such cases LCL is deployed in an effective manner to model and simulate the systems.

The Liquid Cooling Library contains the relevant pressure drop models, valves and heat exchanger models to accurately represent a district heating network. Careful industrial deployments have resulted in reductions in operational costs by adapting the water supply temperature and intelligent valve control. Overall this library finds applications in a variety of applications starting from aerospace, automotive industries to solar power plants [6].

5 Possibilities

Twin Builder being a tool that covers the entire breadth and depth of physical modeling opens numerous possibilities on the type of studies that can be made through the Liquid Cooling Library. Twin Builder can let the LCL models combine with detailed control element models involving advanced power electronics devices and characterization tools (Semiconductors) along with state-machines and signal flow logic. Twin Builder's strong features like Co-simulation with 3D solvers and reduced order modeling can capture complex multi-physics interactions between the 3D and the 1D systems simulation world captured with Modelica Libraries. Twin Builder also supports Functional Mockup Interface through which models built can shared with many other Simulation tools.

6 References

1. Internal Flow Systems (BHRA fluid engineering series), Donald Stuart Miller, 1978
2. Massimo Stellato, Luca Bergianti, John Batteh, Powertrain and Thermal System Simulation Models of a High Performance Electric Road Vehicle, Proceedings of the 12th International Modelica Conference, May 15-17, 2017, Prague, Czech Republic.
3. John Batteh, Jesse Gohl, Sureshkumar Chandrasekar, Integrated Vehicle Thermal Management in Modelica: Overview and Applications, Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden, pp. 409-418, 2014
4. Modelon, "Liquid Cooling Library", Version 3.1, 2018. http://www.modelon.com/products/modelon-library-suite/liquid-cooling-library/
5. Modelon blog on "Integrating Thermal Management Systems for the Benefit of Advancing Electric and Hybrid Vehicles", http://www.modelon.com/blog/articles/integrating-thermal-management-systems/
6. Österholm, R. and Pålsson J., Dynamic modelling of a parabolic trough solar power plant, Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden, pp. 409-418, 2014.

DEVELOPMENT OF A FLEX-PLI SYSTEM MODEL AND INVESTIGATIONS OF INJURY

Yong-Ha Han[1] In-Hyeok Lee[2] Whe-Ro Lee[2]

[1]Hyundai Motor Company, Korea, yongha@hyundai.com

[2]ESI Korea, Korea, {inhyeok.lee, whe-ro.lee}@esi-group.com

Abstract

Pedestrian accidents give direct damage to the human body. Pedestrians do not have any safety devices and it results in a significant risk of injury to the pedestrians as compared to other accidents (Carroll, 2014). To protect pedestrians, EURO NCAP, JNCAP, and various pedestrian safety laws are enforced. Korea also imposes KNCAP and related laws. Assessment of pedestrian injuries is performed throughout impact tests using the head, upper leg, and lower leg impactor. (Yong, 2006)

Pedestrian injury simulation is normally performed using the Finite element method at the early design stage to reduce a cost and research period. FE simulation requires detail design data, high-performance equipment and long computation time. FE simulation gives detail results how each part is deformed, how much energy is absorbed and how much injury values are resulted in. But on the other hand, it requires well-designed simulation matrix and many simulations to find contributions to the injury values of various design parameters at the initial design stage.

The system model simulation allows more intuitive parametric studies than the existing detailed FE studies. The computation is much faster than the FE simulation, results are obtained results within in a few seconds and contributions of various parameters are directly get throughout simple parametric simulations.

In this study, the impactor and vehicle system model is developed for the lower leg injury risk assessment. The system model of lower leg impactor, Flex-PLI is developed by comparing to its FE model and system model parameters are calibrated against several static and dynamic certification tests of FLEX-PLI. The vehicle is modeled to equivalent mass-spring-damper systems and its parameters are obtained from existing FE simulation results. And finally developed system model is verified against FE simulation results.

Keywords: Pedestrian injury protection, Flex-PLI, System Model

1 Introduction

There are many ways of designing cars to improve pedestrian protection. Although the car has many parts and composed with complex sub-structures, the load transmission path of specific impact loading is relatively simple. It is possible to derive important parts for load transfer by analyzing the collision mechanism. And it is possible to construct an intuitive system model for collision problems by calculating equivalent stiffness between major parts.

In this study, system model for lower leg impactor and vehicle is constructed for evaluating lower leg injury. The impactor is based on the latest Flex-PLI (Mallory, 2005). Flex-PLI consists of three parts: Femur, knee, and tibia. Femur consists of 8 segments, tibia has 10 segments and knee consists of four spring wires that represent ACL, PCL, LCL and MCL ligaments respectively. Flex-PLI is covered with multi-layered rubber and pouch cover that expresses the muscles, flesh, and skin of the legs (HUMANETICS, 2011).

The Impactor system model is verified in a variety of ways because it must be able to reproduce injuries that occur on the human body's legs. Static certification test represents the stiffness of the bone structures. Dynamic certification tests such as a pendulum test and an inverse test are performed to verify responses during the impact. In this study, parameters of Flex-PLI subsystems are calibrated using component and static certification tests and are validated using dynamic impact tests.

The vehicle equivalent model was constructed with several equivalent masses that represent set of major parts on the load transmission path and springs and dampers that connect equivalent masses are calculated using FE simulation results.

2 System modeling of Impactor system model

2.1 Flex-PLI system model

Flex-PLI has four groups as shown at figure 1: three structural groups (femur, knee, tibia) and one flesh(skin) group. The masses of structures and flesh are 9.38kg and 3.82kg and total mass of the system is 13.2kg (ESI GROUP. *FLEX PLI GTR FE Model Users Manual*).

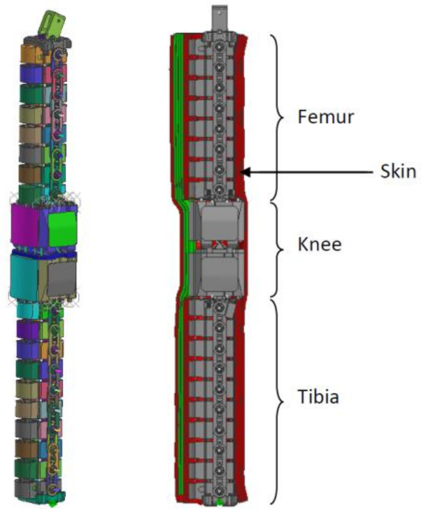


Figure 1. Components of Flex-PLI

2.1.1 Femur and Tibia assembly model

Femur and tibia are made of glass fiber reinforced plastic part, take most of the loads and produce bending moment outputs using strain gages which are bonded to it. Femur and tibia are composed of several separated bone segments as shown in figure 2. Each bone segment has a plastic frame, two aluminum spacers, and a cover. Two spacers are rigidly connected to the plastic frame by screws, and the impactor cover is bonded on the frame by a strong double-sided tape. The bone is assembled with the frame by the contact of two spacers. To make sure the spacers are properly contacting the bone, optional thin shims can be used. The link is used to connecting two neighbor bone segments and it maintains uniform space between them. Rubber buffer in the link prevents hard contact between segment. There are four steel wire cables inside the segment corner to prevent a bone damage due to over-bending. Accelerometers are mounted on some segments. (HUMANETICS, 2011)

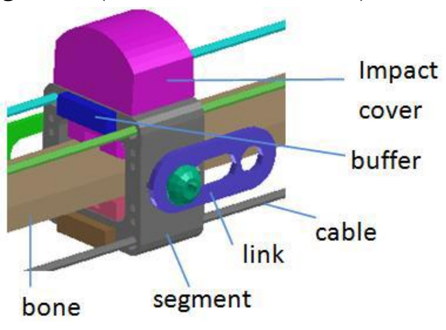


Figure 2. Bone components of Flex-PLI

The system model of bone segments and links are shown at figure 3. Each bone segment is modeled as a rigid body. The link between bone segments is modeled using a revolute joint with rotational spring and damper. The stiffness of rotational springs is calibrated against the static bending certification test as described at section 2.2

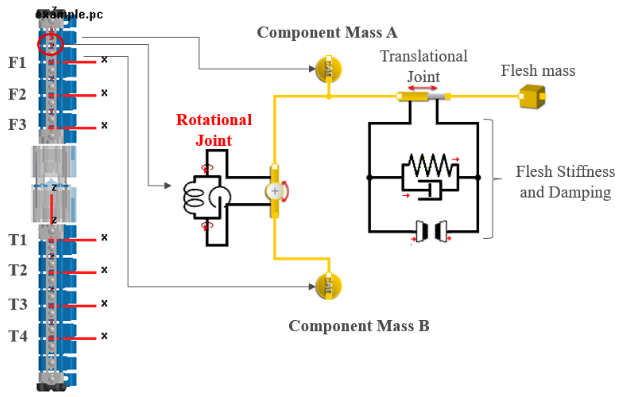


Figure 3. System modeling for femur and tibia

2.1.2 Knee

The shape of knee group is shown at Figure 4 without femur block. The knee consists of two aluminum blocks, steel wire, springs, and covers. Two aluminum blocks represent distal femur and proximal tibia and these blocks are contacted each other. Block shapes around contacting area resemble real bone shapes and mimic the real relative motion of bones of human knee. Knee ligaments are represented using steel wires and springs. Springs are designed to meet the required ligament resistive forces and range of motion. They are initially compressed, so the two blocks can hold each other. There is an accelerometer mounted on the lower block and four strings are used to measure MCL, LCL, ACL and PCL elongations. Two impact covers are mounted on the blocks using double-sided tapes. Aluminum covers are used to protect the signal wires and electronics.

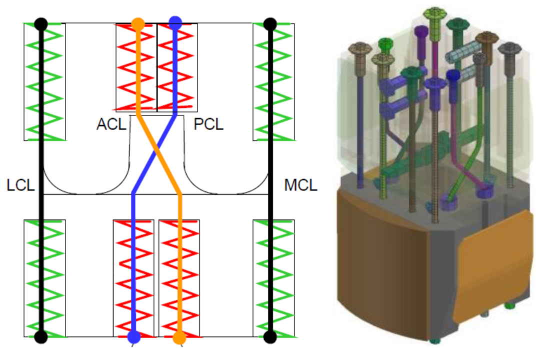


Figure 4. Knee components (source: UN-ECE)

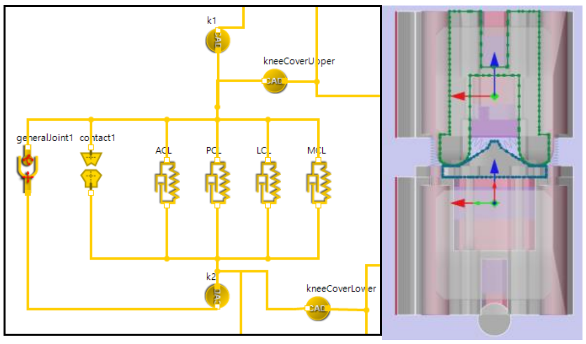


Figure 5. Diagram for Knee assembly and contact

The system model for the knee is shown at figure 5. Knee blocks are modeled using rigid body and shapes are extracted from FE model. Contact condition is imposed between blocks to represent complicated real relative motions of knee blocks accurately. Four springs connect two blocks and represent knee spring blocks. The stiffness of these springs is calculated using FE model.

2.2 Flex-PLI static validation

2.2.1 Femur and tibia bending simulation

There is a moment requirement for femur and tibia static bending loading. Maximum moments measured from femur and tibia load cells should be 400Nm when the specific forces are applied at the center of each assembly. The forces are 4,848N and 3,902N for the femur and tibia respectively. Static bending simulation using femur and tibia system model is performed to validate the bending stiffness as shown at figure 6. Simulation results show good correlation as shown at figure 7

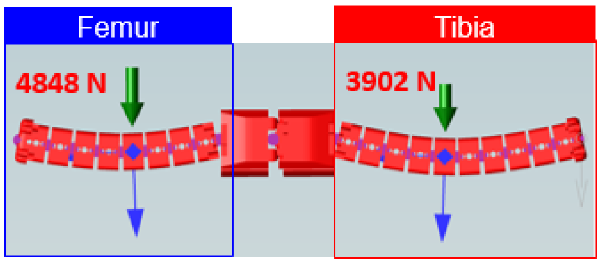
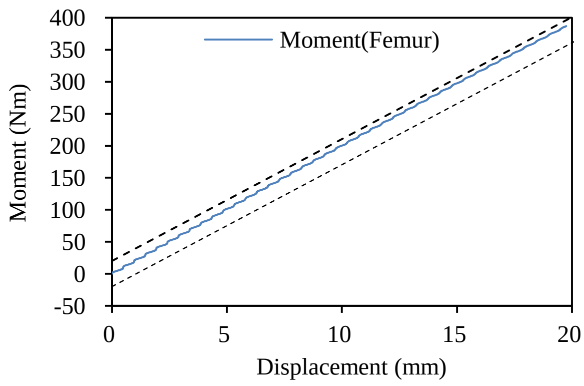


Figure 6. Static bending simulation model



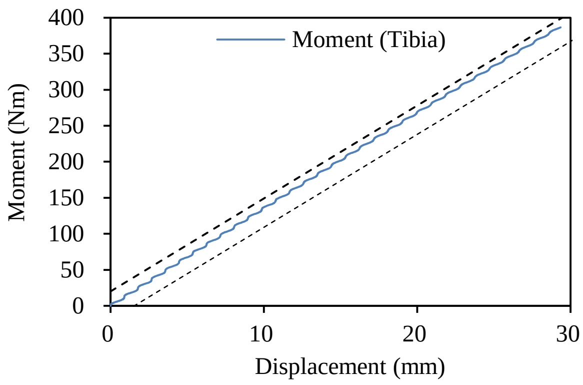


Figure 7. Static bending simulation results of femur and tibia

2.2.2 Knee bending simulation

Knee static bending test procedure is shown at figure 8. As shown at this figure, a round shape of zigs is used at the ends to represent pure bending condition. Force is applied to the proximal tibia end, but round shape loading zig begin to contact the femur right after the loading. The knee static bending simulation is performed using knee system model as shown at figure 9. Simulation results shows good correlation with knee ligament elongation corridors as shown at figure 10.

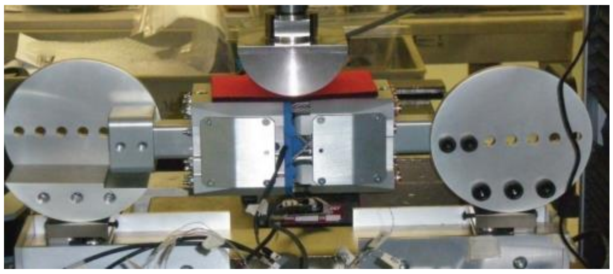


Figure 8. Knee static bending test

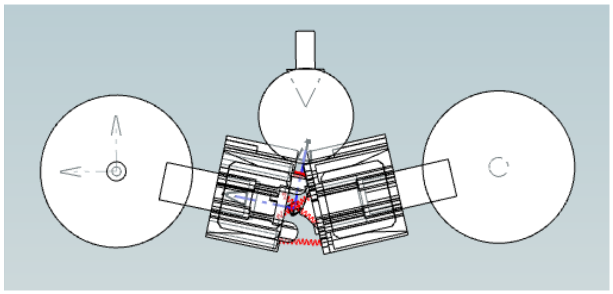


Figure 9. Knee static bending simulation

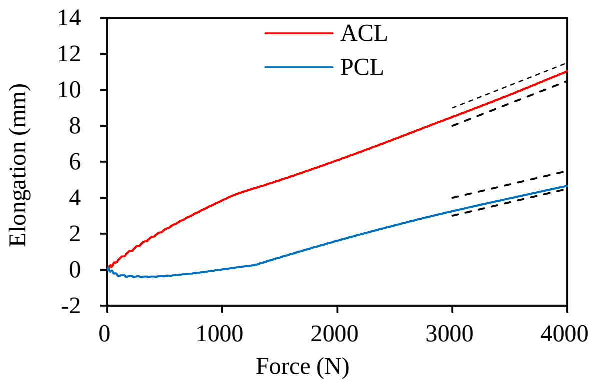


Figure 10. Comparison of knee ligament elongation result for knee static bending simulation

2.3 Flex-PLI dynamic validation

In this study, dynamic performance of system model is validated by pendulum and inverse certification tests.

2.3.1 Pendulum test

Figure 11 illustrates the pendulum test method. The end of the tibia is linked to the test zig and additional 5 kg mass installed at the femur end. Before dropping the Flex-PLI, it was lifted by 15 degrees. After it is dropped, the Flex-PLI stopped due to the impact with stopper bar. Moments of femur and tibia and elongations of four ligaments are measured until the Flex-PLI is rebounded.

Pendulum simulation is performed as shown at figure 12 and moment results obtained satisfies corridor as shown at figure 13.

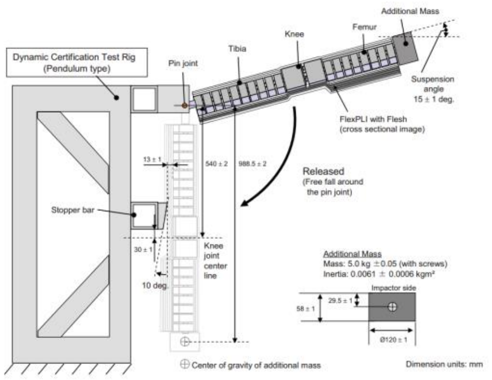


Figure 11. Pendulum test

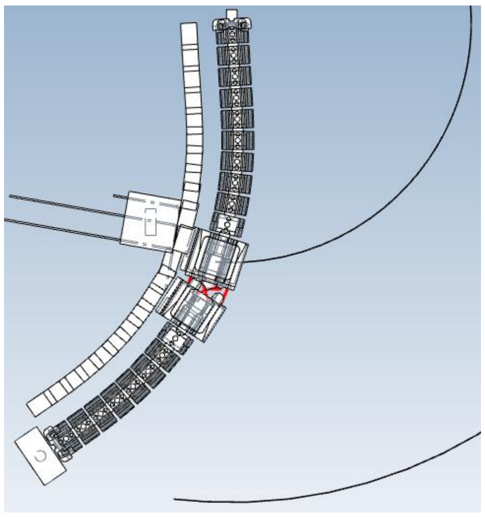


Figure 12. Pendulum simulation model

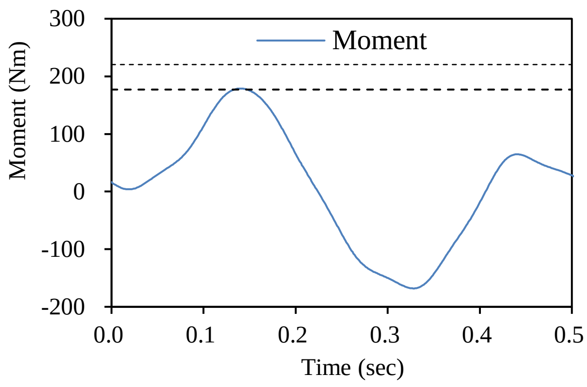


Figure 13. Comparison of moment result for pendulum simulation

2.3.2 Dynamic inverse test validation

Figure 14 shows inverse dynamic test. The end of the femur is hanging on the zig, and moving mass impact the lower knee block. The velocity of moving mass is 11m/s and weight is 8kg. To verify the system model, the inverse test was reproduced as a system model simulation as shown at figure 15.

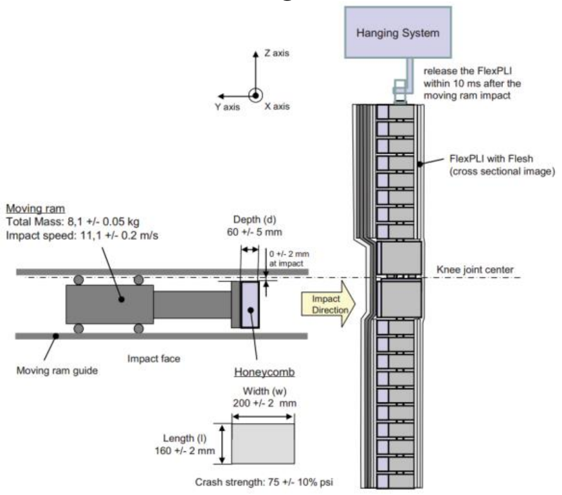


Figure 14. Inverse test

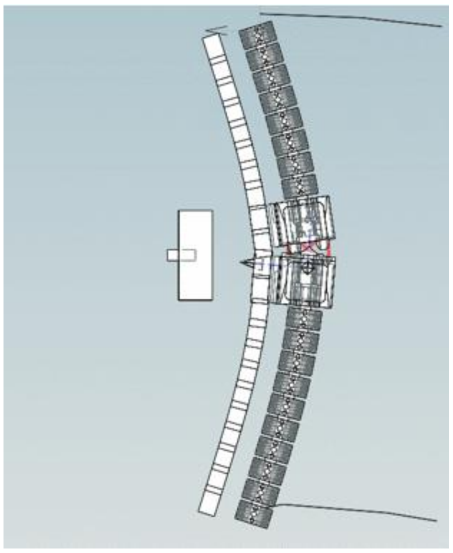


Figure 15. Inverse simulation

Table 1. Comparison of moment for dynamic pendulum simulation

Position	test	simulation	Error (%)
F3	81	52	-36%
F2	126.2	110	-13%
F1	158.3	165	4%
T1	193.5	200	3%
T2	149.7	150	0%
T3	100.9	92	-9%
T4	47.8	40	-16%

The end of the femur is restrained and a moving mass of 8 kg is impacted to the Flex-PLI with 11 m/s speed. Contact condition is set between mass and Flex-PLI. In the test, flex-PLI is mounted on the hinge, but it is represented by fixing the corresponding degree of freedom. Table 1 shows the moment results and errors at the measurement positions between tests and system models. Simulation results nearby the knee (T1 and F1) are well matched to test results. Moment errors of femur upper(F3) and tibia lower(T4) are larger and it is mainly because the tensional stiffness of flesh between bone segments are omitted in the system model.

3 Vehicle parameter extraction from FE model

The vehicle consists of many parts who transfer forces into directly connected or contacted parts during impact. Due to its complexity, it is almost impossible to extract equivalent stiffness from part results directly. In this study, indirect method is used to extract stiffness and damping parameters between vehicle subsystems using momentum and impulse conservation law via following five steps.

1. Grouping; Flex-PLI impacts to the vehicle fascia. All injury values of Flex-PLI is decided how much reaction forces are generated from the vehicle how much deformation is occurred at the fascia during impact. Thus, vehicle fascia should be divided into several rigid bodies in according to its amount of deformation. During impacting, the impact load is transferred through the hood, upper frontend structure, bumper and lower frontend structures. By analyzing these loading paths, parts included in the same loading path are grouped as a subsystem. In this study, fascia is divided into 7 groups, the hood is divided into 2 groups and engine room structure is divided into 5 groups.
2. Calculation of effective momentum per each group; During impacting, forces are applied into parts, parts move and absorb some energies due to its applied forces and transfer the same amount of forces into other connected or contacted parts. In this study, it is assumed that inertial forces of each group are small because the mass of parts that are majorly involved in the impact is relatively small compared to whole vehicle mass. Then, momentum difference of two groups becomes same as impulse between two groups and impulse becomes same as reaction forces generated from internal energy absorption between two groups. The effective momentum of each group is a summation of all momentums for parts of each group.
3. Calculation of forces between groups; Impulse is calculated from the difference of momentum between two groups. Then, forces are calculated simply by differentiating impulses with time.
4. Calculation of relative displacement between groups; For the fascia, bending behaviors of Flex-PLI are directly decided with fascia frontend displacement which is contacted to Flex-PLI. Thus, for the fascia, frontend displacement of each group is used as a group displacement. For other groups, displacement of group center of gravity is used as a group displacement.
5. Calculation of stiffness between groups; Stiffness is calculated using forces and the relative displacement between groups.

To verify this methodology, simulation is performed using simple bumper impact model as shown at figure 16 and then, stiffness between fascia and back beam is calculated using the proposed method.

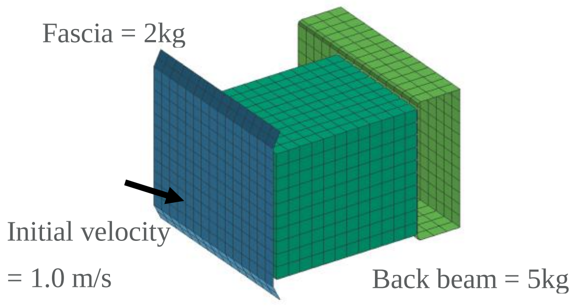


Figure 16. simple bumper model

From the FE simulation results, momentum of fascia and back beam is calculated using part results and momentum difference is obtained as shown at figure 17.

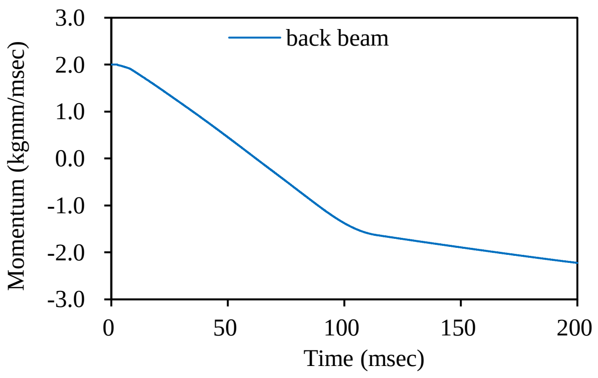


Figure 17. Obtained difference of momentum

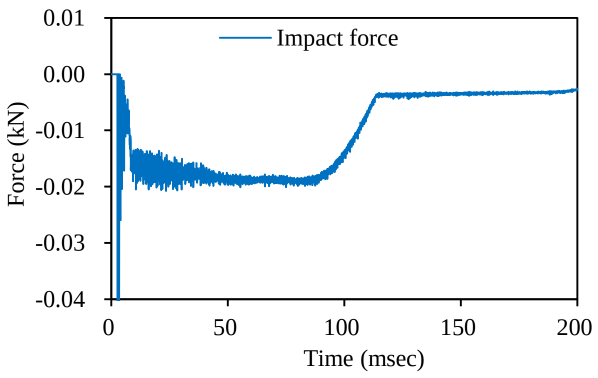


Figure 18. Impact force between fascia and back beam

Momentum difference is same as impulse and forces between fascia and back beam is calculated by differentiating impulse with time as shown at figure 18. Finally, stiffness between fascia and back beam is calculated using forces and relative displacement of two parts as shown at figure 19.

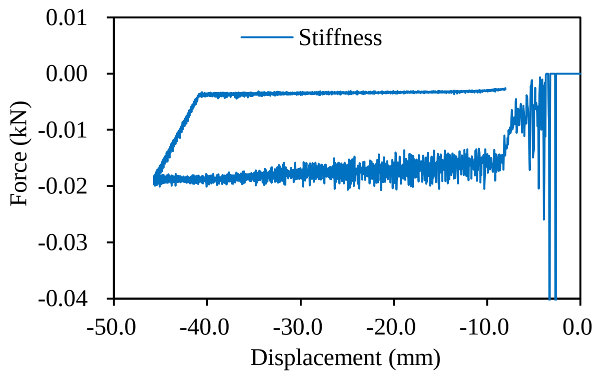


Figure 19. Stiffness between fascia and back beam

To verify stiffness results, the stiffness is converted to engineering stress-engineering strain relationship using foam length and foam area and it is compared with original foam material characteristics.

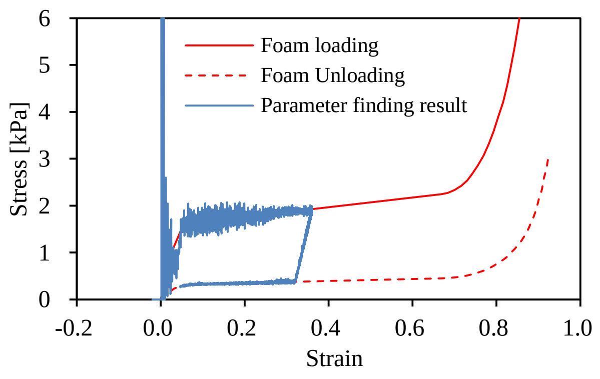


Figure 20. Comparison of stress-strain relationship from stiffness result to original foam material characteristics

As shown at figure 20, stiffness obtained from proposed method will represents internal forces generated between fascia and back beam.

4 Lower leg impact simulation

As described in section 3, vehicle system model is composed of 14 groups: 7 fascia groups, 2 hood groups and 5 engine room groups. Vehicle group is the group of parts which are connected to 4 engine room groups and which are not included into other groups.
Each group is modeled using mass. A sliding contact is defined between impactor bone segments and 7 vehicle fascia groups and one hood group and the ground group is entirely fixed. Spring stiffness between groups are calculated using proposed method as described at section 3 and damping coefficient is calibrated to represent unloading behavior of each group.
Figure 22 shows comparison of deformed shape with FE simulation results. As shown at figure 22, when the impactor crashed the vehicle, knee hits the bumper around, femur and tibia ends moved to the vehicle direction till the impactor rebound. And system model

simulation results are well matched to FE simulation results.

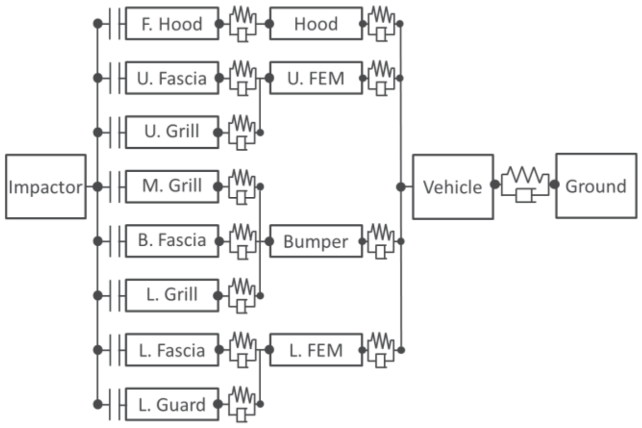


Figure 21. a diagram for the vehicle system model

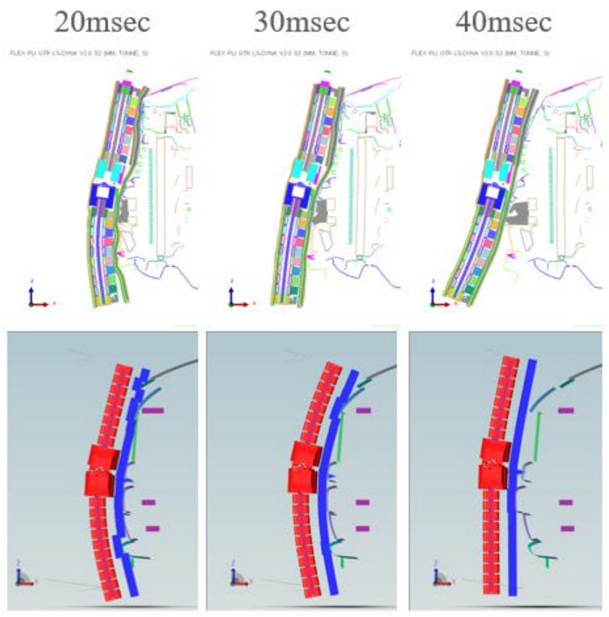


Figure 22. Comparisons between FE and system model

The momentum of bone segments is generated due to different deformation of each fascia groups. **Figure 23** shows the moment of the FE simulation and the system simulation at the main position. The first peak moment occurred at 25ms. Since the femur and tibia move in opposite directions with respect to the knee, the directions of moments are opposite to each other and the impactor is rebounded after 25ms. In the case of the System model, the behavior up to the first peak was very similar to the FE results, but there was a difference after the impactor rebound. This is because the springs in the vehicle model express loading phase well but are not sufficient to express the stiffness in unloading phase. However, since the maximum damage caused by the collision occurs mostly in the first peak, the prediction of the injury is possible from the results of this study.

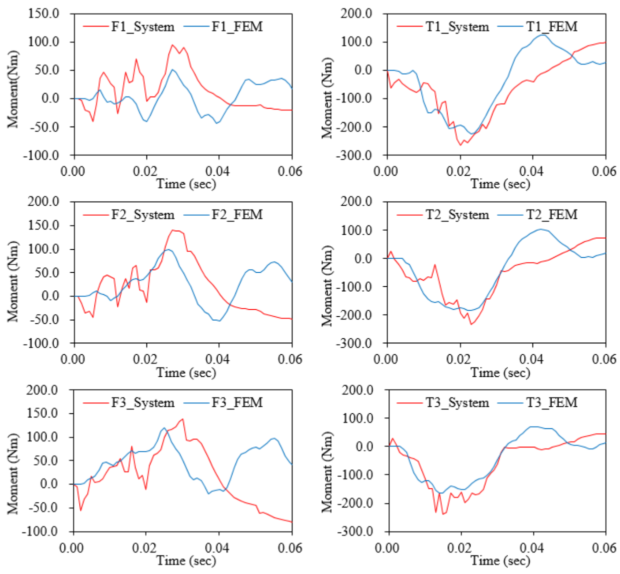


Figure 23. Compare moments between FE simulation and system simulation

Conclusion

In this study, a system model is constructed to analyze the pedestrian lower leg impact. The system model for lower leg impactor, Flex-PLI, is constructed and verified against various static and dynamic certification tests. Methodology for constructing vehicle system model is proposed based on momentum and impulse conservation law and it is illustrated using simple bumper impact model. Vehicle system model is finally constructed by grouping all vehicle parts into 14 groups, calculating stiffness between groups using the proposed method. Finally, whole lower leg impact simulation model is simulated and verified by comparing its results with FE simulation results. The impactor moment results of FE simulation and the system simulation are well matched in terms of the magnitude of the moment and the peak time. In the case of system simulation, calculation time is completed within 5 minutes in the single core machine, so it is very efficient to review various parameters at the initial stage of design.

References

HUMANETICS. *Flex PLI GTR User Manual*, 2011

ESI GROUP. *FLEX PLI GTR FE Model Users Manual*,

ESI GROUP. ITI *SimulationX User Manual 3.8*

J A Carroll, A Barrow, B J Hardy, B Robinson, Transport Research Laboratory, *Pedestrian legform test area assessment report*, 2014

Ann Mallory, Jason A. Stammen, France Legault, 19th International technical conference on the Enhanced Safety of Vehicles. *Component leg testing of vehicle front structures*, 05.0194, 2005

Boo-Joong Yong, Hyun-Deog Cho, Jae-Wan Lee, Journal of the Korean Society of Manufacturing *Process Engineers. Development of vehicle evaluation system for pedestrian protection*, Vol. 5 No. 4, pp.53~58, 2006

Vehicle Systems Modelling and Analysis (VeSyMA) Platform

Hannah Hammond-Scott[1], Mike Dempsey[1]

[1] Claytex Services Limited, United Kingdom, {hannah.hammond-scott, mike.dempsey}@claytex.com

Abstract

The Vehicle Systems Modelling and Analysis (VeSyMA) platform is a suite of compatible Modelica libraries for modelling automotive vehicles and their subsystems, where the complexity can be tailored to the user's requirements.

Keywords: Vehicle Modelling, Motorsport, Engines, Powertrains, Suspensions, Driver-in-the-Loop, rFpro

1 Introduction

In this industrial paper we will look at the Vehicle Systems Modelling and Analysis (VeSyMA) platform developed by Claytex. This suite of Modelica libraries was created to provide a modular approach to vehicle modelling, where the user can tailor the complexity of the model to meet their specific needs.

The foundation of this capability is the VeSyMA library which provides the architecture of the vehicle, and the base classes for the vehicle subsystems. It builds upon the open-source Vehicle Interfaces Library (Modelica Association 2018).

The VeSyMA extension libraries then provide more detailed modelling capabilities in specific subsystem and domain areas; such as engines, powertrain and suspensions. Because the VeSyMA extension libraries use the base classes from the VeSyMA library, all the models created are compatible.

Currently the VeSyMA platform[1] consists of the:

- VeSyMA library
- VeSyMA – Engines
- VeSyMA – Powertrain
- VeSyMA - Suspensions
- VeSyMA – Motorsports
- VeSyMA – Drive-in-the-Loop
- VeSyMA – Terrain Server

In the following sections we examine the capability that each of these libraries give the user.

Users can see the Modelica code behind the models in the VeSyMA platform, so they can fully understand how a component is modelled. Users can therefore adapt models to suit their purpose as necessary.

The VeSyMA platform was built on the concept of subsystem models being parameterized individually and then used as 'off the shelf' models to build the complete vehicle model.

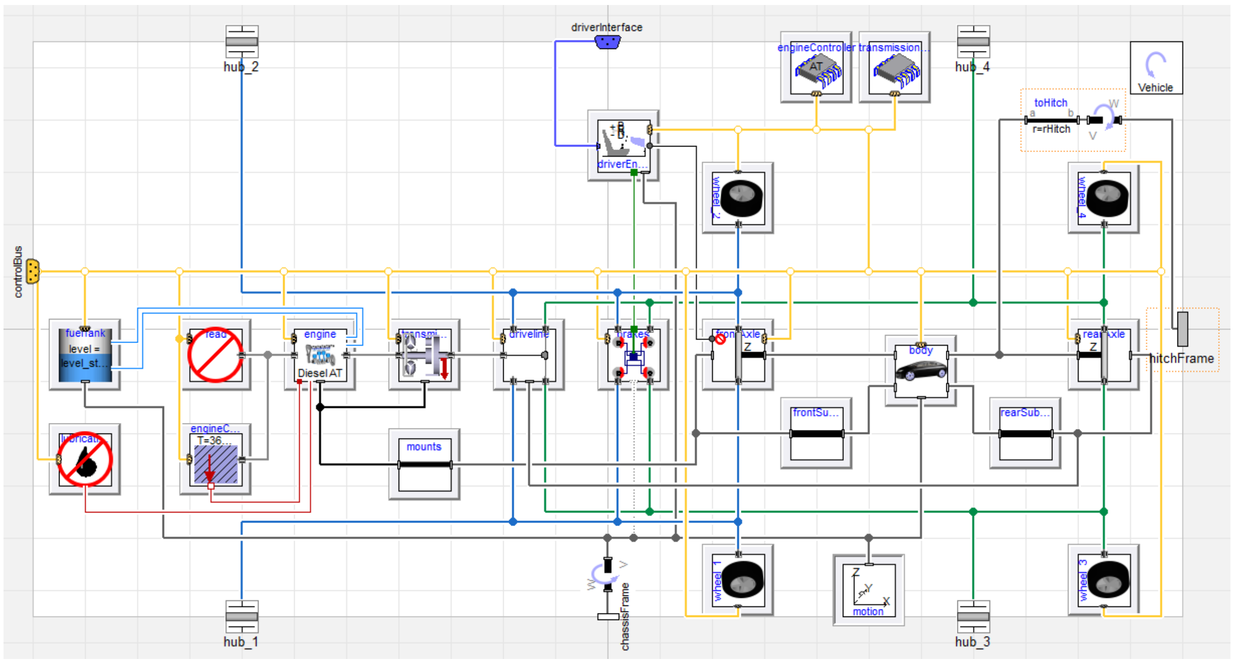


Figure 1. Large, diesel, automatic car model from the VeSyMA library

[1] At the time of publication, the VeSyMA platform is at version 2018.1

2 VeSyMA

The VeSyMA library is the cornerstone of this vehicle modelling platform. It contains vehicle templates for most common vehicle configurations, including internal combustion engine, hybrid and electrically powered vehicles. The subsystems in these vehicle templates are replaceable. As all the VeSyMA platform libraries use common bases classes, the user can easily populate the vehicle model with subsystems derived from any of the VeSyMA platform libraries.

In addition to defining the model architecture, the VeSyMA library includes a collection of idealized subsystem models. This means that vehicles suitable for performing longitudinal studies and drive cycle analysis can be built purely from components from this library. The library includes a set of vehicle examples of various configurations. Figure 1 shows the rear wheel drive, automatic, diesel, large family car example from the library.

The idealized subsystem models can be combined with detailed subsystem models built from the VeSyMA extension libraries to build a complete vehicle model (Ensbury *et al.* 2018). This allows the detailed subsystem to be tested in a vehicle with minimal effort.

The vehicle experiment templates allow the vehicle to be integrated with an open or closed loop driver. Figure 2 shows a New European Drive Cycle (NEDC) test of the vehicle model from Figure 1. There is also the option for the vehicle to tow a trailer if required.

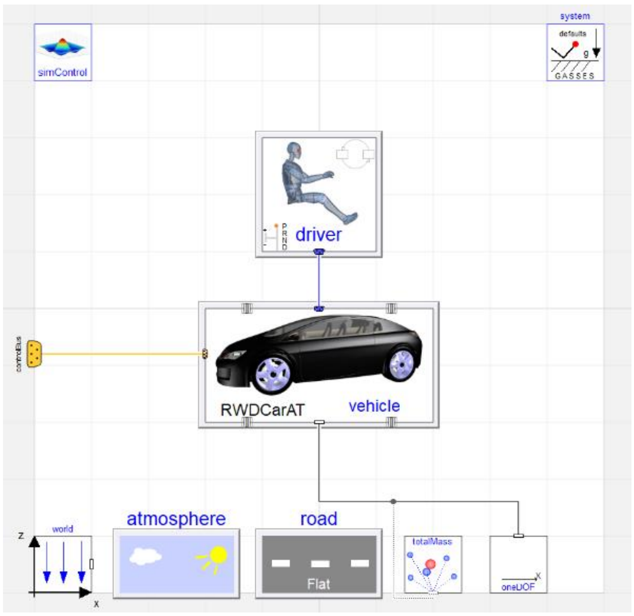


Figure 2. Drive cycle test using the closed loop driver

3 VeSyMA – Engines

The VeSyMA – Engines library is focused on internal combustion engine modelling, providing 2 levels of detail:

- Mean Value Engine Model (MVEM)
- Crank Angle Resolved Engine Model (CAREM)

The MVEMs are used to predict the cycle averaged performance of the engine, permitting fast simulation. The combustion and emissions models are map based using manifold pressure and engine speed as the primary inputs, with further corrections for spark timing and air-fuel ratio. The air mass flow rate through the engine is calculated based on a function which permits reasonable scaling of the engine displacement.

The CAREMs predict the instantaneous torque and air flow through the engine. The combustion heat release is based on the Wiebe model, where tables define the Wiebe coefficients at different engine speeds, loads and air-fuel ratios. Where a predictive functionality is required, the user can select a two-zone predictive combustion model in place of the Wiebe model. Both heat release models include knock prediction. Flow through the engine block is dictated by the valve geometry and opening characteristics as well as the piston-cylinder assembly model. Valve and spark timing effect the fluid dynamics and combustion model. Figure 3 shows the diagram of one of the CAREM examples.

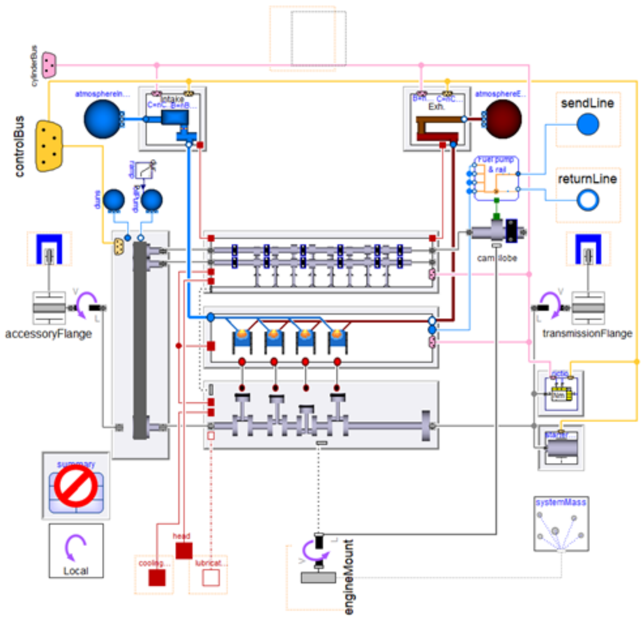


Figure 3. I4, 1800cc, spark ignition, naturally aspirated CAREM with hydraulics based variable cam timing and high-pressure fuel pump

Engine model simulation performance can be improved using surrogate models, where a multi-cylinder engine is represented by a single cylinder with flow and torque replication for the other cylinders.

Both MVEM and CAREM variants use the same engine templates with common intake, exhaust and mechanical components.

The VeSyMA – Engines library supports both naturally aspirated and forced induction engines. With turbocharger, supercharger and intercooler components in the library.

Emissions are modelled using a map based approach. The exhaust model can include emission treatment devices such as catalytic converters, selective catalytic reduction, ammonia slip catalyst and diesel oxidation catalyst.

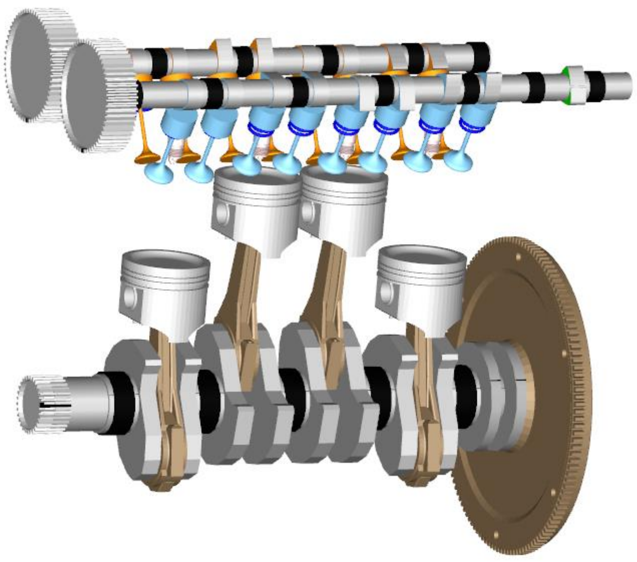


Figure 4. Dyno test animation of a CAREM with hydraulic inlet variable cam timing

This library contains a collection of parameterized MVEM and CAREM examples for a variety of engine configurations. The library provides a dynamometer experiment, allowing both steady-state and transient testing. The dynamometer can also be used to represent a chassis dyno. Figure 4 illustrates the animation from a dynamometer test of a CAREM example.

4 VeSyMA – Powertrain

The VeSyMA – Powertrain library focuses on the modelling of transmissions, gearboxes and drivelines using the same efficient, multibody systems used in VeSyMA – Engines. The library contains advanced models for:

- Shafts
- Bearings
- Gear meshes ranging from ideal to ones with backlash and mesh stiffness
- Flexible joint components

There are complex assemblies, such as epicyclic gearsets and differentials, as well as mounting systems. Figure 5 shows the dual-clutch example from the library.

VeSyMA – Powertrain models capture the full motion of the powertrain on its mounts to allow the vehicle response to be examined for drivability studies (Gillot *et al.* 2017). The library includes examples of idiot start, standing start and tip-in vehicle experiments.

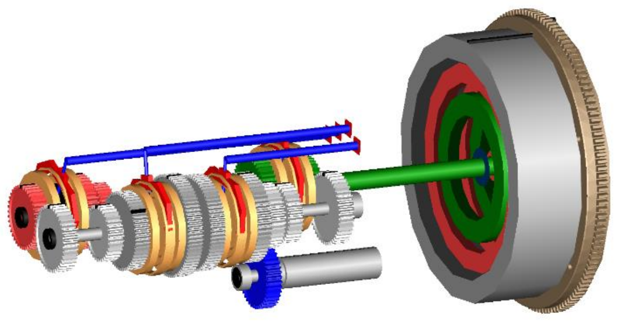


Figure 5. Dual-clutch transmission

5 VeSyMA – Suspensions

The VeSyMA – Suspensions library focuses on modelling the suspensions subsystems to perform vehicle dynamics analysis.

There are multibody suspension models of common road car suspension configurations; MacPherson, double wishbone, multilink, multilink hybrid, trailing arm and trapezoidal. These models can use ideal joints, or bushes and flexible bodies; Figure 6 show a front double wishbone linkage with ideal joints. There are double wishbone examples using aggregate joints, which have been optimized for real-time simulation.

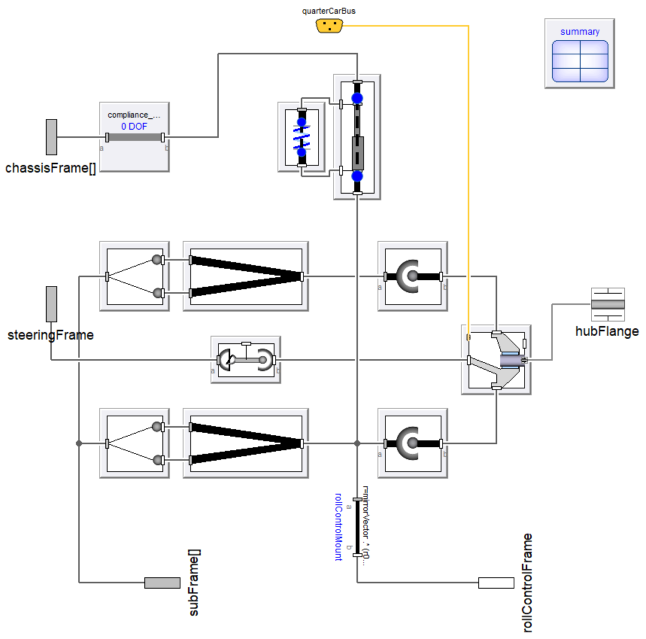


Figure 6. Front double wishbone linkage with ideal joints

There are also table-based examples of the double wishbone and multilink models, where kinematic data is used to define the hub position. These models are less complex than their multibody equivalents, which can improve simulation performance.

The VeSyMA – Suspensions library includes Pacejka MF6.1 and MF6.2 tire models (Besselink *et al.* 2010; Pacejka 2012), as well as an interface to the FTire tool (Cosin 2018). There is the option for both single and multiple contact points.

There are road building functions for creating custom road and circuit models. The VeSyMA - Suspensions road models permit friction, road roughness and curbs to be added. It is also possible to define a driving line and stopping events for the driver model to follow. There is also support for using the OpenCRG road definitions (VIRES 2017).

To perform dynamic vehicle maneuvers, the VeSyMA driver capability is extended to include closed loop drivers with lateral control. A test driver model is also available, where the closed loop control can be overridden.

The road, driver and vehicle models from VeSyMA – Suspensions have been combined to create vehicle maneuver examples, including:

- Acceleration
- Coastdown
- Constant radius
- Braking
- Double lane change
- Slalom
- J-turn
- Fishhook
- Figure of eight

There are several rig experiments provided. Kinematic rigs for testing at the quarter car and half car levels. At the vehicle level, there are KnC (Kinematics and Compliance), four post and seven post rigs, as shown in Figure 7.

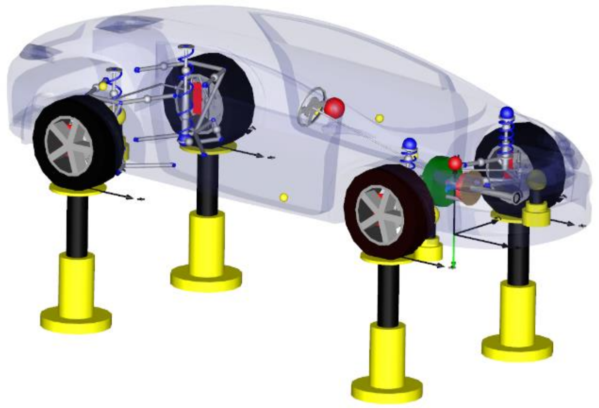


Figure 7. Seven post rig test

The VeSyMA – Suspensions library was designed with real-time simulation capability in mind. The model architecture includes support for the multi-threading features available in Dymola.

6 VeSyMA – Motorsports

VeSyMA – Motorsports is a motorsports focused extension of the VeSyMA – Suspensions library. It includes suspension configurations specific to open wheel race cars, sports cars and NASCAR style vehicles. These suspensions are optimized to minimize the nonlinear systems of equations to improve real-time simulation performance.

The suspensions are parameterized using geometry records, this allows alternative geometries to be easily applied to the same suspension models.

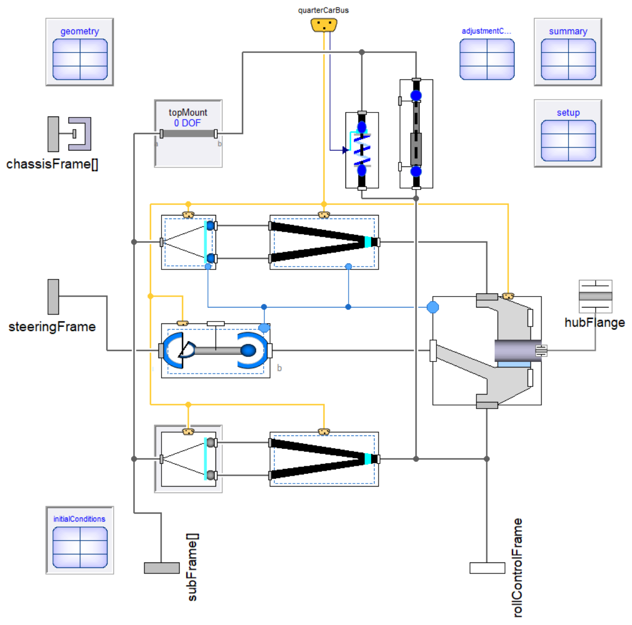


Figure 8. Front NASCAR linkage

There are adjustment shims incorporated into these suspension models, to permit realistic setups to be applied to the vehicle. Figure 8 shows the adjustments, in light blue, in a linkage component. There are two setup test variants, the adjustments can be made in sequential order or simultaneously to achieve the desired targets. Following a setup experiment, the shim values can be extracted from the results for reuse in the suspension models via the geometry records.

Additional aerodynamics models are included for race car bodies and wings. There is also the provision for adding aerodynamics to the wheel models.

7 VeSyMA – Drive-in-the-Loop and VeSyMA – Terrain Server

The VeSyMA – Driver-in-the-Loop (DiL) and VeSyMA – Terrain Server libraries allow the VeSyMA libraries and Dymola to be integrated with rFpro.

VeSyMA – DiL permits vehicles created using VeSyMA – Suspensions and VeSyMA – Motorsports to be integrated with rFpro driving simulators. It includes the templates and tools to run the vehicle model on a wide range of systems connected to the simulator.

VeSyMA – Terrain Server enables the rFpro Terrain Server to be integrated into Dymola simulations. This allows the high-fidelity LiDAR track data available from rFpro (Kangaloosh 2018) to be used in Dymola simulations, promoting consistency between the driver-in-the-loop and Dymola environments.

8 Summary

The VeSyMA platform provides a family of Modelica libraries to enable the modelling and analysis of automotive vehicles.

At its core is the VeSyMA library, which provides the architecture of the vehicle models and experiments. From the idealized vehicle subsystem models in this library, vehicle models for performing longitudinal and drive cycle studies have been built.

The VeSyMA extension libraries build upon this capability to allow users to develop detailed engine, powertrain and suspension subsystem models. As all the VeSyMA platform libraries are compatible with each other, the user can tailor the vehicle model's fidelity to allow them to perform the analysis they require. For example, the idealized rigid suspension from the VeSyMA library can be replaced in a VeSyMA vehicle by a detailed multibody suspension models from VeSyMA – Suspensions to permit vehicle dynamics investigations to be conducted.

The VeSyMA platform libraries were designed with real-time simulation in mind from the beginning to support software, hardware and driver-in-the-loop testing. Models have been optimized to improve simulation performance, and model structure supports the use of the multi-threading features available in Dymola. VeSyMA – Driver-in-the-Loop and VeSyMA – Terrain Server further support the use of VeSyMA platform vehicle models in the driver-in-the-loop simulator environment using rFpro.

References

Modelica Association. Vehicle Interfaces Library, 2018. *URL: https://www.modelica.org/libraries* (Last accessed: 19 Mar. 2018).

Ensbury T., Harman P. and Dempsey M. Modelling and Development of a Pseudo-Hydraulic Power Steering Model for use in Real-Time Applications. *Proceedings of the 2nd Japanese Modelica Conference*, 2018.

Gillot R., Picarelli A., Dempsey M. Model Reduction Techniques Applied to a Physical Vehicle Model for HiL Testing. *Proceedings of the 12th Modelica Conference*, 2017. DOI: 10.3384/ecp17132299.

Besselink I. J.M., Schmeitz A. J.C. and Pacejka H. B. *An improved Magic Formula/Swift tyre model that can handle inflation pressure changes*. Vehicle System Dynamics Vol. 48, Iss. Sup 1, 2010.

Pacejka H. B. *Tire and Vehicle Dynamics.* 3rd Edition. Elsevier Ltd, 2012.

Cosin Scientific Software. FTire, 2018. *URL: https://www.cosin.eu/ products/ftire/* (Last accessed: 19 Mar. 2018).

VIRES Simulationstechnologie GmbH. OpenCRG, 2017. *URL: http://www.opencrg.org/* (Last accessed: 19 Mar. 2018).

Kangaloosh Limited. rFpro Terrain Server, 2018 . *URL: http://www.rfpro.com/driving-simulation/terrain-server.aspx* (Last accessed: 19 Mar. 2018).

Dual-Clutch Transmission Model Reduction Function

Romain Gillot[*] Alessandro Picarelli[*] Mike Dempsey[*]

[*] Claytex Services Ltd. Edmund House, Rugby Road, Leamington Spa, CV32 6EL
{romain.gillot, alessandro.picarelli, mike.dempsey}@claytex.com

Abstract

This paper introduces a gearset model reduction function. It is an automated process that works for any type of gear set. The focus is on a dual-clutch transmission (DCT). Reducing the transmission from a multibody model to a table-based loss and inertia 1D rotational mechanics model leads up to a 70% decrease in simulation time. This performance improvement allows engineers to run a detailed physics derived model over the NEDC urban section in real-time or faster.

Many customers have expressed the need for an automated model reduction tool. This paper aims at describing the methodology to create one.

This function will be included in the next release of the VeSyMA – Powertrain library.

Keywords: model reduction, dual-clutch transmission, gear set

1 Introduction

It is an extended version of the function that was introduced in a previous paper (Gillot R., 2017). The functionalities and the process will be explained.

Whenever a full vehicle model is required to be run, simulation time becomes of prime importance. Some compromises can be made in terms of model detail in areas of the model that are not directly the subject of the study. However, the results the simplified subsystems produce still need to be close enough to the ones of the detailed subsystems in order to provide correct interactions amongst the components of interest. The gearbox is one of the most computationally expensive subsystems in a vehicle model. Bearing models with friction enabled, gear pairs with mesh and mesh loss models, shift mechanisms can slow down the model simulation.

This paper shows a method to reduce a gearset model in a quick and automated manner. Essentially, the gear set is run several times with different speed and torque inputs ranging from zero to a maximum defined by the user. This is repeated for every gear. The results of this series of experiments are then collected and mapped and the new reduced model is automatically generated and parameterised.

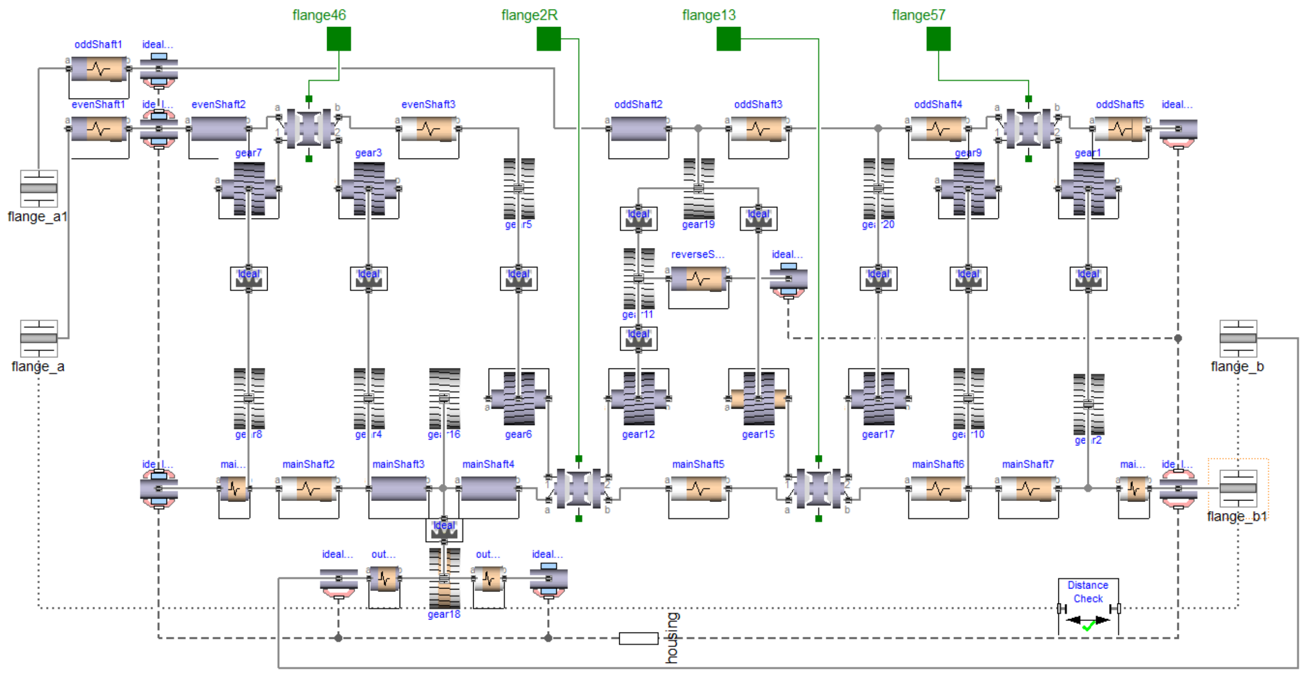


Figure 1. Detailed dual-clutch transmission gear set.

2 Presentation of the model

The 7-speed dual-clutch transmission gearset to be reduced is modelled using components from the VeSyMA – Powertrain library from Claytex. The model interface (i.e. the translational and Rotation3D flanges) is compatible with the Modelica Standard library components.

Friction can be added in the bearings and in the gear mesh models. Several friction models are available for the bearings (Coulomb, Elastic, Hydrodynamic, etc.) and for the gear mesh (Temperature dependent, Torque dependent, etc.).

As this model is detailed, it requires substantial parameterization. The gears need a value for diameter and number of teeth as well as a mesh loss model, the shafts need mass, length, inertia, compliance (optional), the bearings containdetailed friction models.

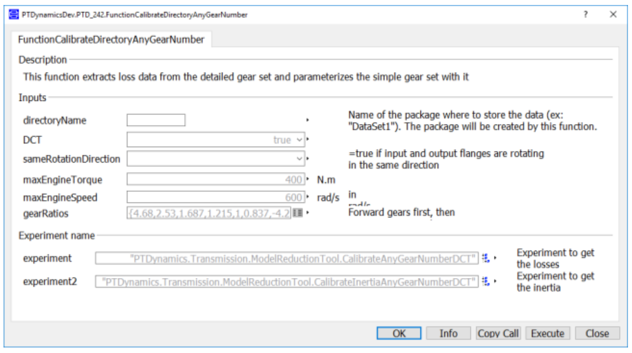
Parameterizing the detailed gearset in depth to achieve high fidelity is a time-consuming process. We do not want to lose all this valuable information when using the reduced model.

However, it is not immediately obvious what the layout and parameterization of a reduced version of the model above (Figure 1.) should be. We cannot simply use the same parameterization as for the detailed model since the models are different in structure.

3 The function

3.1 The function interface

There are not many inputs required from the user in order to be able to run the function.


PTDynamicsDev.PTD_242.FunctionCalibrateDirectoryAnyGearNumber

FunctionCalibrateDirectoryAnyGearNumber

Description

This function extracts loss data from the detailed gear set and parameterizes the simple gear set with it

Inputs

directoryName			Name of the package where to store the data (ex: "DataSet1"). The package will be created by this function.
DCT	true		
sameRotationDirection			=true if input and output flanges are rotating in the same direction
maxEngineTorque	400	N.m	
maxEngineSpeed	600	rad/s	in
gearRatios	{4.68,2.53,1.687,1.215,1,0.837,-4.2		Forward gears first, then

Experiment name

experiment	"PTDynamics.Transmission.ModelReductionTool.CalibrateAnyGearNumberDCT"	Experiment to get the losses
experiment2	"PTDynamics.Transmission.ModelReductionTool.CalibrateInertiaAnyGearNumberDCT"	Experiment to get the inertia

Figure 2. Function user interface.

The user needs to specify:

- *directoryName*: name of the package where the models created by the function (see 3.3.) will be saved
- *DCT*: a Boolean to choose if the transmission that will be reduced is a DCT or not, if not the reduction process is similar, but the model interface is slightly different (only one input flange for example)
- *sameRotationDirection*: a Boolean to specify if the input and output shafts of the detailed model are rotating in the same direction
- *maxEngineTorque*: sets the test upper torque boundary
- *maxEngineSpeed*: sets the test upper speed boundary
- *gearRatios*: transmission gear ratios

Then the correct experiments need to be selected for collecting both the inertia and the losses for generating the reduced model.

3.2 The internal process

This section describes the inner workings of the function

The whole idea is to run the gear set over all its operating points and to collect the results to then map them into a table. The process is repeated for every gear. The final step is to interpolate for speed and torque.

The data collected is: Input and Output shafts inertia, Parasitic Driving and Driven losses and Efficiency Driving and Driven losses.

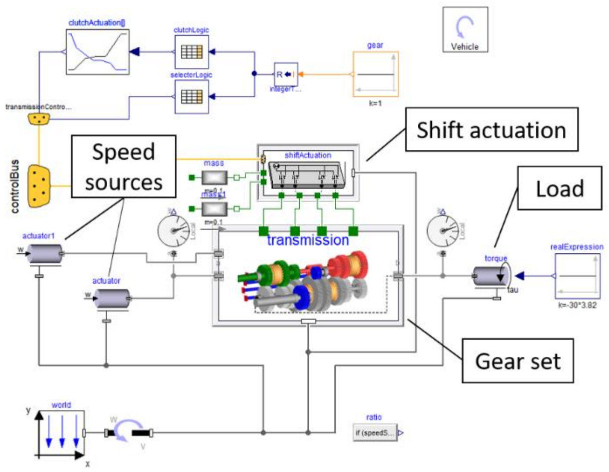


Figure 3. Gear set test rig.

The speed is ramped up to a maximum set by the user which corresponds to the maximum engine operating speed. We repeat this test several times, varying the load every time up to a value specified by the user that corresponds to the maximum engine operating torque.

To make the process more generic and quicker, the function runs the gear set in all the gears at once (including reverse) using arrays of components. This means that the experiment above (Figure 3.) is dragged and dropped into a new model and is of dimension "number of gears". This means that the function can be used for gear sets with any gear number.

The load-dependent and speed-dependent losses are calculated at the input flanges, the same goes for the inertia. This is true for all the gears except the neutral gear for which we calculate the losses and inertias for the input and output shafts separately since there is no significant physical connection between them.

The last step is to create the reduced model and to parameterise it. This is done using the print() function (Modelica.Utilities.Streams.print). The simple gear set model (see Figure 4) already exists prior to running the function, it will only be extended during the reduction process to generate a parameterised reduced model.

It should be noted that any gear set model can be reduced using this function, no matter what its inner structure or parameterisation is as long as it uses flanges with bearings as connectors.

3.3 The function output

This is the model that the function will extend from and parameterize.

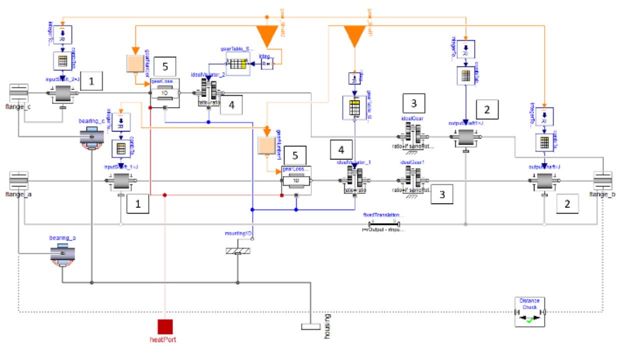
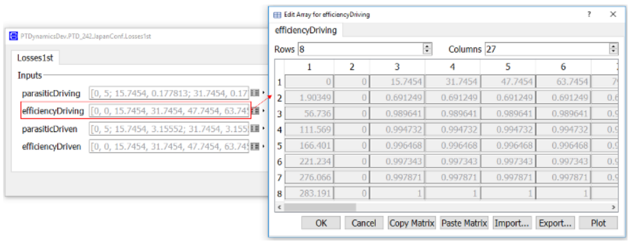


Figure 4. Reduced gear set (1. Odd and even input shafts inertia, 2. Odd and even output shafts inertia, 3. Ratio accounting for rotation direction (-1 or 1), 4. Variable ratio for odd and even shafts, 5. Losses for odd and even shafts).

This is a 1D rotational mechanics gear set model. The interface makes use of 3D flanges for the sake of compatibility with the other vehicle subsystems, but all the rotating components use 1D rotational flanges.

A variable ratio is applied between the input and output shafts. The model uses lumped variable inertias that have a different value for each gear pair selected. The total gear set inertia is lumped at flange_a and flange_c (the input flanges) except when in neutral, in which case the inertia is accounted for as an input shaft and an output shaft inertia. The losses are also lumped in a table in two places: on the odd and on the even shaft and are interpolated for angular velocity, torque and gear number.

Once the function has successfully run, it opens the model in Figure 4 - after it has been parameterized - along with a set a data records where the data is stored. One data record will be created for each gear that contained the torque-dependent and speed-dependent losses for both the driving and driven conditions. The inertias will be stored in an additional data record.


PTDynamicsDev.PTD_242.JapanConf.Losses1st

Losses1st

Inputs

parasiticDriving	[0, 5; 15.7454, 0.177813; 31.7454, 0.17
efficiencyDriving	[0, 0, 15.7454, 31.7454, 47.7454, 63.74
parasiticDriven	[0, 5; 15.7454, 3.15552; 31.7454, 3.155
efficiencyDriven	[0, 0, 15.7454, 31.7454, 47.7454, 63.74

Edit Array for efficiencyDriving

efficiencyDriving

Rows 8 Columns 27

	1	2	3	4	5	6
1	0	0	15.7454	31.7454	47.7454	63.7454
2	1.90349	0	0.691249	0.691249	0.691249	0.691249
3	56.736	0	0.989641	0.989641	0.989641	0.989641
4	111.569	0	0.994732	0.994732	0.994732	0.994732
5	166.401	0	0.996468	0.996468	0.996468	0.996468
6	221.234	0	0.997343	0.997343	0.997343	0.997343
7	276.066	0	0.997871	0.997871	0.997871	0.997871
8	283.191	0	1	1	1	1

Figure 5. Data record with the collected losses (highlight on the driving efficiency losses).

It takes about 10 to 15 min to run the function.

4 Results

To demonstrate the benefits in using the model reduction function, we ran a full vehicle model over the NEDC Urban drive cycle (see Figure 7), first with a detailed dual-clutch transmission gear set and then with its reduced equivalent.

The solver used is Cvode – variable order with a tolerance of 1e-6. The number of intervals is 50000 for a simulation time of 800s.

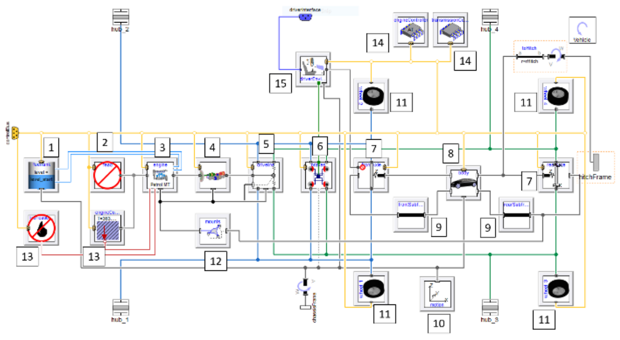


Figure 6. Detailed vehicle model (1. Fuel tank, 2. FEAD, 3. Engine, 4. Transmission, 5. Driveline, 6. Brakes, 7. Front and rear suspensions, 8. Body, 9. Front and rear subframes, 10. Motion block, 11. Wheels, 12. Engine mounts, 13. Cooling and lubrication systems, 14. ECU and PCM, 15. Driver environment).

The vehicle model above (Figure 7) has compliance in many subsystems. The engine mounts, the suspensions and the driveshafts are linearly compliant. Speed and torque dependent losses have been applied to the transmission. Such a level of detail allows to study the vibrations from the engine down the driveline but is computationally very demanding.

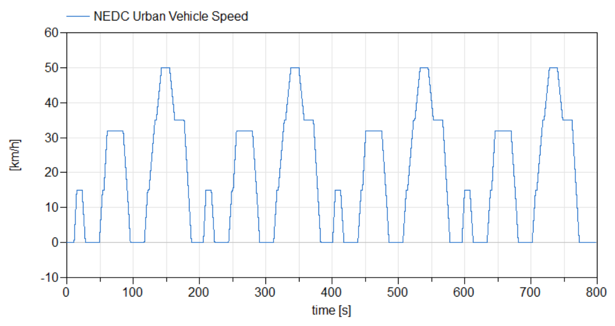


Figure 7. NEDC target vehicle speed.

All the tests described in this paper were performed on a desktop computer with the following processor: Intel® Core™ i5-7600K CPU @ 3.8Ghz. However, the results for simulation speed have been normalised to highlight the gain in performance when reducing the model rather than the overall performance itself. The percentage of speed improvement should be roughly similar on any machine whereas the actual simulation times would be highly dependent on the processor specifications.

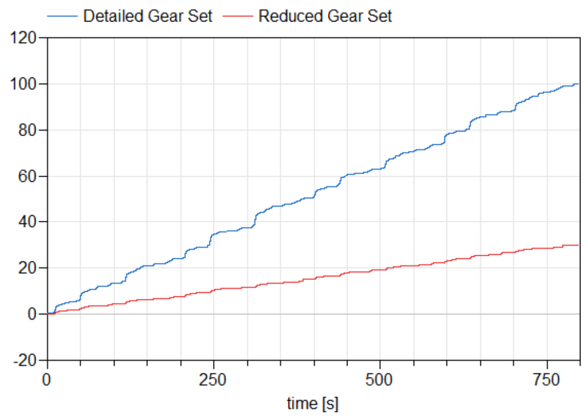


Figure 8. Percentage of CPU time improvement.

The plot above (Figure 8) shows a 70% reduction in simulation time. The vertical axis is CPU (i.e. the time taken by the computer to perform the calculation) and the horizontal axis is the time in the simulation (i.e. the duration of the drive cycle).

The results in terms of accuracy are now discussed. A closed loop driver model is used in this model in order to follow the longitudinal speed profile showed in Figure 7.

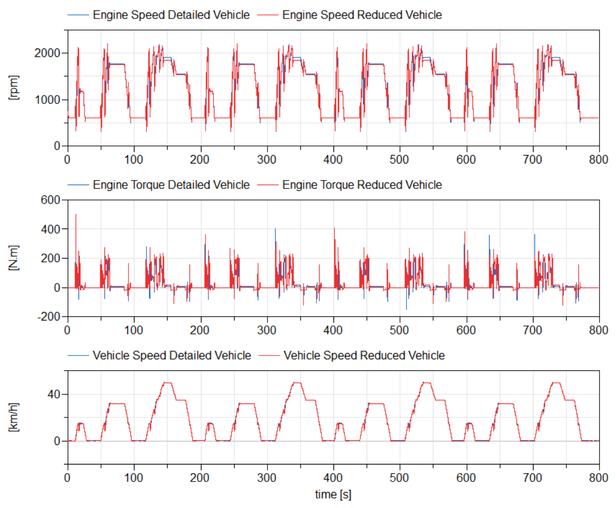


Figure 9. Engine speed and torque and vehicle speed for the detailed and reduced vehicles.

The results are matching well even if there are some small inaccuracies at times, especially visible in the torque plot. The torque map in the engine and losses tables used in the reduced transmission are rather coarse. The increment in the engine speed input of the engine map is 500 rpm and it only uses three different values for throttle opening. The transmission losses maps increase the speed input by 150 rpm and 50 Nm each time. Having coarse tables will lead to inaccuracies during interpolation. Moreover, since the simple engine model used in this paper feeds back the engine speed signal as an input to its torque source, it creates an even bigger error. The last explanation for this error is that the backlash is not taken into account in the reduced model. This means that the torque will be transmitted directly from the input to the output of the gear set with no delay other than related to the compliances in the driveline shafts and mountings. This generates a slight phase shift in the torque curve. Despite these imprecisions, the error remains small (it is investigated further in the following paragraphs).

In terms of the results directly related to the transmission:

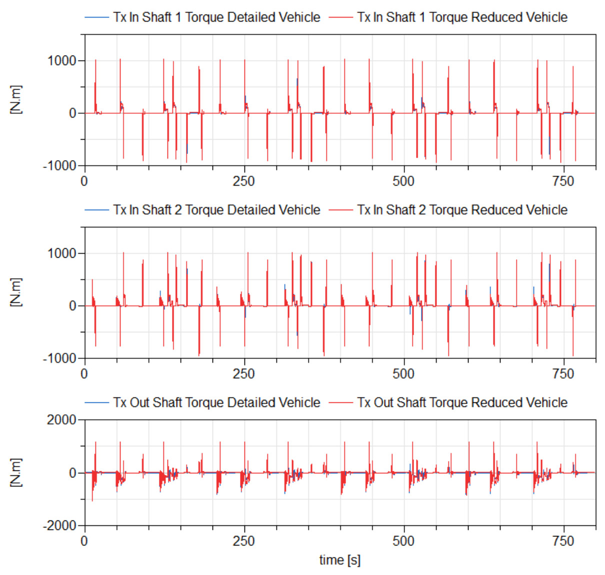


Figure 10. Comparison of the following variables for the detailed and reduced vehicles: transmission input shaft 1 (odd gears) torque (top plot), transmission input shaft 2 (even gears) torque (middle plot) and transmission output shaft torque (bottom plot).

Since the driver controls the vehicle speed, the angular velocity at the input and output flanges of the transmission matches very well the results of the detailed model. It is thus more interesting to analyse the torque curves.

In Figure 10 we can observe a slight discrepancy in the results. When the transmission input shaft speed changes sign, we can see first a slight phase shift of engine torque. The reduced model not considering backlash will thus start to transmit torque a few moments before the detailed model.

Another error happens during gear shift. The reduced gear set model does not use synchronisers to ensure high shift quality. This will result in transients that are accurate enough to allow this reduced model to be for this type of studies (gear shift, transient operation). Of course, one should keep in mind that such a simplified transmission model is not suited for studies which focus on this subsystem for driveability and shift quality. The level of detail in each vehicle subsystem needs to be adjusted depending on the area of interest.

5 Conclusion

A function has been created that provides an easy routine to reduce a dual-clutch transmission gear set model. The performance improvement is significant and the reduction process is quick (it takes less than 10 minutes on my machine). The accuracy of the results is overall satisfying even if some particular operating conditions lead to some considerable error. Attention has to be given to what type of study is to be carried before considering using this reduced model. An improved accuracy is however to be sought for in the future to broaden the range of tests than be done using this model.

It could be extended to other subsystems in the future.

Given all the gearset models in the array used to to generate data for the reduced gearset are independent from each other, we could thus take the advantage of the multicore capability and run each one of them on a separate core. This needs to be investigated as part of future work.

References

Gallagher Stephen. et al. Model-based Real-time Systems Engineering, Loughborough, England, Powertrain Modelling and Control Conference, 2016

Dempsey Mike. et al Coordinated automotive libraries for vehicle system modelling, Vienna, Austria, Proceedings of the 5th International Modelica Conference, 2006

Dempsey Mike. et al. Investigating the Multibody Dynamics of the Complete Powertrain System, Como, Italy, Proceedings of the 7th Modelica Conference, 2009

Gillot Romain, Picarelli Alessandro, Dempsey Mike. Model Reduction Techniques Applied to a Physical Vehicle Model for HiL Testing, Prague, Czech Republic, Proceedings of the 12th Modelica Conference, 2017

FluidDynamics Library for Coarse-Grid CFD-Simulation in Modelica

Dr. Stefan Wischhusen[1] Timo Tumforde[1] Hans-Herrmann Wurr[1]

[1]XRG Simulation GmbH, Germany, {wischhusen, tumforde@xrg-simulation.de}

Abstract

This paper describes the content and the use of the new FluidDynamics Library which can be applied to carry out CFD simulations using Modelica as an open modelling language. Typical applications until now have been in automotive, aircraft and buildings development. In this paper a fire dynamics and smoke removal simulation is presented. These simulations are very important in the process of approving a building permission. The FluidDynamics Library helps to identify promising ventilation and control setups and speeds up the simulation process significantly.

Keywords: CFD, Coarse grid, Computational-Fluid-Dynamics, Navier-Stokes, FluidDynamics Library

1 Introduction

For simulations of air-conditioned spaces different approaches can be suitable. In 1D-simulation-tools often a single node model is used to model the mass and energy balance for a complete room or building. Those “Lumped”-models are designed for quick simulations of longer time intervals and are supplied by many free or commercial Modelica libraries on the market (examples: HumanComfort Library, AIX Library, Buildings Library). For a more detailed analysis of the inhomogeneous air states and air velocities within the compartment the user has to model the air flow between the discrete volumes, or else he has to establish Navier-Stokes equations which can describe the air flow in 3D. Those Navier-Stokes-based equations are implemented in free or commercial CFD software, like OpenFoam, Ansys Fluent, Star CCM+ and so on. For a combined calculation of both modelling approaches, an interfacing software (Middleware) is required, which handles the exchange of variables at the model boundaries or connections (i.e., inlets and outlets of the air spaces). For this purpose TISC [TIS(2018)] or MpCCI [MPC(2018)] may be used. Although the coupling works fine in general, it requires up to three software licenses. Even in case of a built-in solution (e.g. Ansys Simplorer) a coupled solution requires more computational resources and/or the decoupled solution and data exchange at discrete time points may generate balance failures.

A Modelica-based coarse grid CFD-solution has the following advantages:

1. **Save time** by faster Modelica simulations
2. **Instant simulation success** through convergence control of the variable-step solver
3. **Reduced license costs** with only one simulation software when coupled to Modelica models
4. **Reduced elapse time** for iterative work and control design loops through faster simulations
5. **Model customization** since the code is open
6. **Efficient modeling** through symbolic manipulation of the Modelica source code

2 FluidDynamics Library Overview

The FluidDynamics Library, available in Dymola 2018 FD01, consists of the following packages:

- Basics
- Weather
- Zones
- CFD
- Examples

The **Basics package** supplies all fundamental definitions which are shared by several models, such as records, icons or functions.

The integrated weather model of the **Weather Package** is able to read arbitrary weather data tables and provides the interpolation to the zone model. Moreover, the model can convert the intensity of radiation to the effective area according to its spatial orientation. All

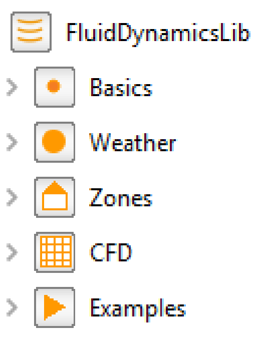


Figure 1. Package overview of the FluidDynamics Library

necessary weather information is bundled and sent to the zone model via a standardized weather connector.

The **Zones Package** contains models to develop mobile or stationary applications. Mobile applications can be built with models e.g. for aircraft [Michaelsen(2015)] vehicle cabins. All models can individually be built and fitted. The Modelica Code is readable.

The **CFD package** contains a Modelica-based, three-dimensional grid model. The model is composed of cubic cells. Each cell, which is used to apply the finite-volume-method, may represent a solid or an air cell. The energy and mass balances are calculated in energy cells, while the mass and heat flows are calculated in the so-called flow cell. Here, the influence of the turbulence, the shear forces acting to the air, the gravity and the buoyancy force is taken into account. By using the Navier-Stokes equations, as found in CFD simulation software, realistic flow conditions can be calculated. At the edges of the grid standard interfaces allow an easy connection with models from other libraries. Thus, e.g. a whole building can be represented by one-dimensional wall models from the Zones package connected to the interior represented by the CFD grid model. On demand one can easily exchange the grid model with a lumped volume model for the air side. Of course, Modelica.Fluid interfaces of the Modelica Standard Library are also available. Moreover, symmetric and periodic boundary conditions can be defined in order to reduce the computational effort for larger rooms. Furthermore, the grid model contains all geometric information about the radiation exchange between surfaces. Thus, in advance the view factors to determine the thermal radiation between the surface pairs are calculated by the software, which is required to determine for example internal shading. The spacing and relative orientation of the surfaces are relevant for the exchanged heat radiation.

The **Examples package** contains models for demonstration of the library capabilities and typical applications.

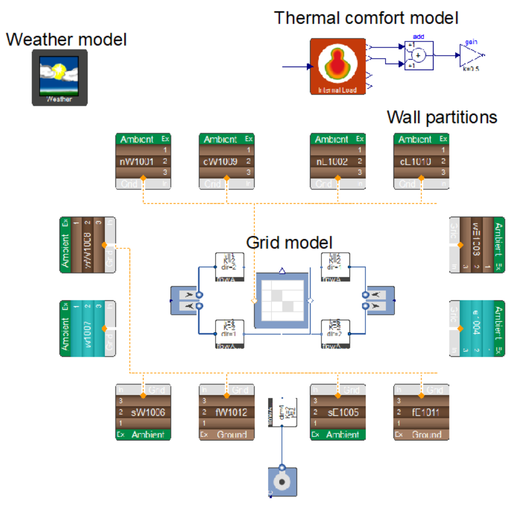


Figure 2. Building example model from Examples package

2.1 Coarse-grid CFD model

When modelling a viscous, heat conducting flow, the Navier-Stokes equations are the basic governing equations. They consist of the continuity equation which represents mass conservation, momentum equation which is derived from Newton's Law of Motion and the energy equation which stands for energy conservation. Since the analytical solution is obtainable in only a limited number of cases, one is forced to approximate these equations to obtain workable results. Therefore, using one of the discretisation methods, Navier-Stokes equations in their differential or integro-differential form are transformed into a set of algebraic equations. Discretised equations are numerically solved in a number of discrete points in space and time. There are multiple numerical approaches to CFD modelling including Finite Volume Method, Finite Element Method and Finite Difference Method. Out of them all, the Finite Volume Method is most widely used for fluid flow problems since it is conservative by default as long as the surface integrals are equal for all control volumes sharing the boundary. In the Modelica-based approach presented here, the Navier-Stokes equations are used to capture conservation of mass, momentum, energy and other associated transport phenomena for Newtonian fluids. The general form of all these equations for a conserved scalar density q in a coordinate system at rest (not moving with the fluid) is given by [Versteeg and Malalasekera(1995)]:

$$\underbrace{\partial_t(\rho\, q)}_{\text{local time dependent change}} + \underbrace{div(\rho\, \vec{c}\, q)}_{\text{convective term}} = \underbrace{div(\Gamma\, grad\, q)}_{\text{diffusive / conductive term}} + \underbrace{S_q}_{\text{source term}} \quad (1)$$

where "∂_t" is the partial derivative with respect to time, "div" represents the local divergence of a vector field and "$grad$" the local gradient vector of a scalar field. Moreover ρ denotes the mass density, Γ is a diffusion coefficient also known as conductivity in heat transport problems. The term $\vec{c} = (u, v, w)$ represents the flow velocity vector, which is the time derivative of the position vector $\vec{r} = (x, y, z)$ in a given coordinate system with coordinates (x, y, z).

Each time derivative of each balance equation is a potential state in the jacobian matrix of the system model. Due to this fact the numerical effort for solving increases exponentially with the number air volumes in the grid model. Typically, up to 2.000 volumes can be handled with state-of-the-art work stations. Thus, the fluid dynamics library offers a **coarse grid** simulation. For a detailed view on the flow field a standard CFD simulation has to be carried out.

The spatial discretization (i.e., grid generation) of the air-conditioned space into cubic volumes is assisted through the free edition of the XRG Score Application (Microsoft Excel Addin), which creates the required geometry record (incl. view factors) for simulation. This Score edition is always shipped together with the FluidDynamics library. Moreover, it provides the post-processing of temperatures, and velocities in 2d images (grid plot).

3 Application to Fire Dynamics and Smoke Removal Simulation

One possible application of the FluidDynamics Library is the usage for a fire dynamics and smoke removal simulation. In the following an exemplary use-case of a fire incident in an open-plane office is presented.

3.1 Description of the Use case scenario

The use-case scenario deals with a fire incident represented by an inflammation of a large-scale printer within a typical open-plan office with a floor space of 1000 m^2 and workplaces for 64 people. Additionally, a kitchen counter is placed within the office. An overview of the office geometry is given in Figure 3. The installed fire-fighting system consists of automatic smoke detection devices as well as a mechanical smoke exhaust ventilation system.

The heat and smoke release can be described with a quadratic increasing curve until the maximum heat release rate is reached [VDI(2009)]. After that the heat and smoke release rate stays at a constant level. An appropriate value for the maximum heat release rate of a large-scale printer is 600 kW. The smoke extraction system is sized with a volume flow rate of 60000 m^3/h and is activated simultaneously with smoke detection.

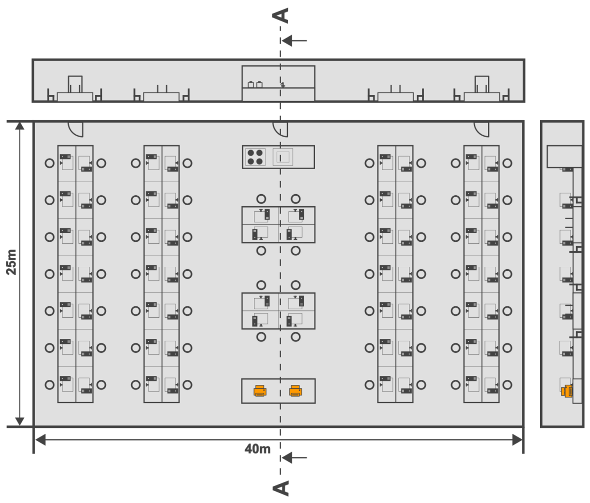


Figure 3. Overview about the office geometry

This can be assumed to be no later than 120s after ignition according to [VDI(2009)]. Figure 4 shows the transient development of the heat release rate together with the smoke release flux and the exhaust air volume flow rate.

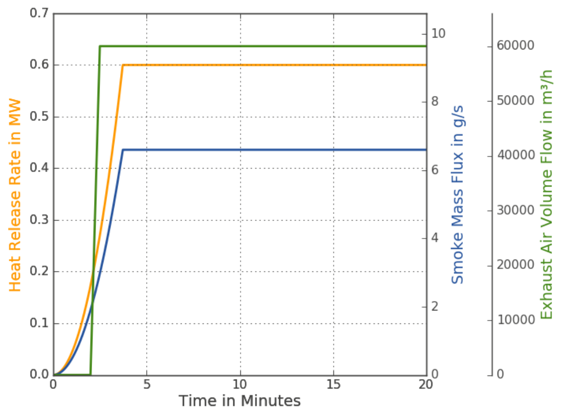


Figure 4. Course of heat and smoke release rate

The fire scenario shall be modeled and simulated both in CFD with ANSYS Fluent with a high locale resolution and with the Coarse Grid model of the FluidDynamics Library.

ANSYS Fluent is one of the most popular tools of the Computational Fluid Dynamics. The 3D-model of the office which is built in the ANSYS DesignModeler is displayed in Figure 5. The red volume is the source of fire. After activation of the smoke exhaust ventilation system the smoke is extracted via the 6 extraction points colored in orange. The green marked areas represent opening surfaces for fresh air entering the room in the case of fire. The air volume is meshed with the ANSYS Meshing Tool. The mesh consists of 4047834 cells in total.

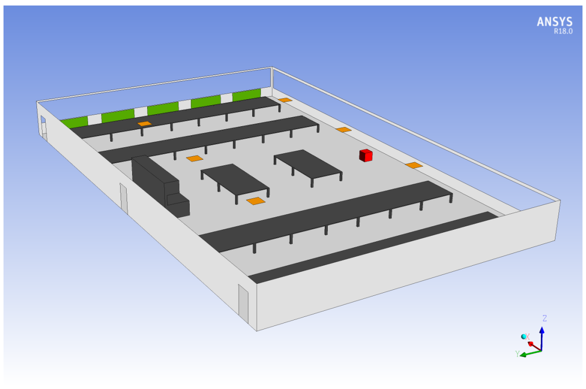


Figure 5. 3-D model of the office for CFD-Simulation with ANSYS Fluent

The Dymola/Modelica model of the same scenario using the FluidDynamics Library is displayed in Figure 6. It consists of the grid model of the room itself and further sub-components representing the main boundary conditions. A fire model computes the transient heat and smoke release within the grid. The air inlet model and 6 extraction models supply information about the ambient conditions at the opening surfaces and the volume flow rate at the extraction points. The coarse grid has a resolution of 13x8x9 hexagonal cells, so that in total 936 cells are used.

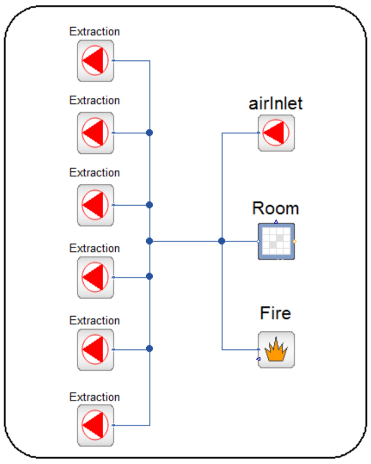


Figure 6. System model of the office for coarse grid simulation with the FluidDynamics Library

For the CFD-simulation with ANSYS Fluent a PC system with 8 cores (Intel(R) Core(TM) i7-6950X CPU @3,00 GHz) and 128GB RAM is used in parallel computation mode. The coarse grid simulation with the FluidDynamics Library in Dymola is proceeded with a PC with a Intel(R) Core(TM) i5-2500 CPU @ 3,30GHz processor and 16 GB RAM in single-core computation mode.

3.2 Comparison of Simulation results

In the following, the results of both the CFD-simulation in ANSYS Fluent and the Coarse-Grid simulation with the FluidDynamics Library are presented for the described fire incident within the office. Therefore the optical densities and air temperatures are visualized for both cases in cut A-A from Figure 3 for the exemplary moment of 600s after the ignition of the fire. Moreover, the computation time for both cases is analyzed considering the used resources.

Figure 7 shows the local optical densities after 600s for cut A-A computed with ANSYS Fluent. One can see that the smoke has spread along the ceiling into the room. Moreover, clear layers of smoke and air can be noticed. The lowest point of the smoke layer is approximately 2,20m above the floor. At the extraction points holes within the smoke layer can be found due to the so called plug-holing effect. Plug-holing describes the effect when air from beyond the smoke layer is sucked to the extraction points due to high vertical velocities at those positions which leads to the mentioned holes within the smoke layer.

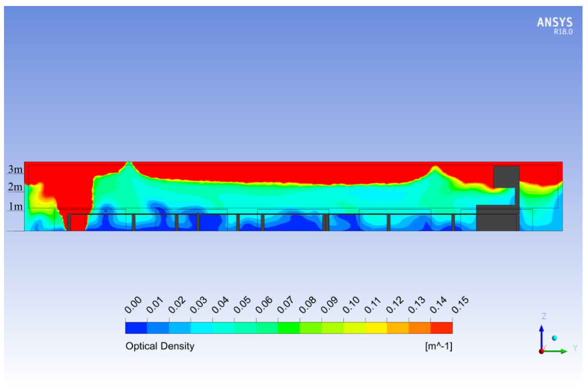


Figure 7. Local optical densities in cut A-A (see Figure 3) after 600s (ANSYS Fluent)

The local optical densities in cut A-A at 600s after ignition are displayed in Figure 8 for the results computed with the FluidDynamics Library in Dymola. It can also be seen that the smoke has spread along the ceiling into the room and a distinct smoke layer is existent. Compared to the result which is computed with ANSYS Fluent the local plug-holing effects are not visible. Nevertheless, a very similar thickness of the smoke layer can be stated. The lowest point of the smoke layer in the result created with the FluidDynamics Library is approximately 2,30m above the floor.

In addition to the local optical densities the computed air temperatures shall be compared for both approaches. Figure 9 shows the local temperature after 600s for cut A-A computed with ANSYS Fluent. It is obvious

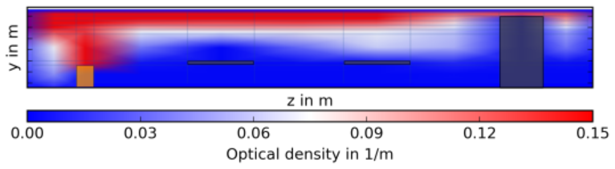


Figure 8. Local optical densities in cut A-A (see Figure 3) after 600s (FluidDynamics Library)

that the highest air temperatures can be found in the surrounding of the fire. The further away from the fire the lower are the temperatures in the smoke layer. The previously described plug-holing effect at the extraction points can also be seen in this temperature plot. In the areas in which workers are potentially present, temperatures only slightly above the ambient and starting temperature can be found.

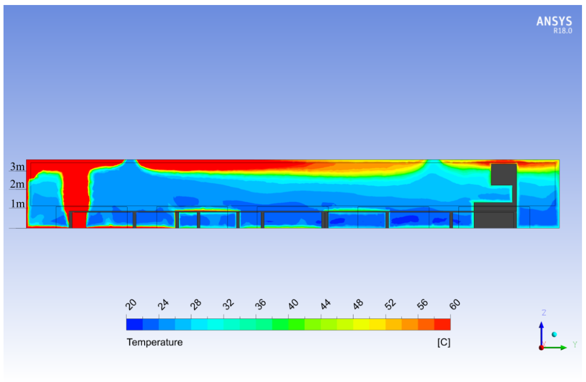


Figure 9. Local temperatures in cut A-A (see Figure 3) after 600s (ANSYS Fluent)

The temperature plot in Figure 10 shows the local temperatures in cut A-A for 600s after ignition, which are computed with the coarse grid model of the FluidDynamics Library. Generally, a very similar situation compared to the result of the CFD simulation with ANSYS Fluent can be seen. The highest temperatures in the smoke layer can also be found in the direct surrounding of the fire source. With increasing distance from the fire source the smoke layer temperatures decrease. In the lower zones in which people can be potentially present, the temperatures hardly exceed the starting room temperatures of 20°C.

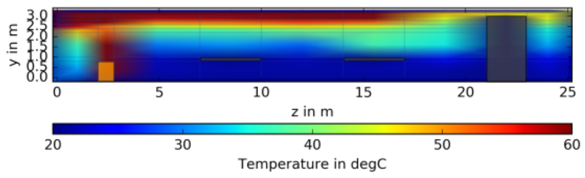


Figure 10. Local temperatures in cut A-A (see Figure 3) after 600s (FluidDynamics Library)

Considering the generally well matching results of both simulations, the computational effort is of special interest. As described above the CFD simulation with ANSYS Fluent is conducted with a multi-core PC in parallel computation mode while for the coarse grid simulation in Dymola a PC is used in single-core mode. While the CFD simulation can be started directly after initialization, the coarse grid model needs to be initialized dynamically fading in the physical effects step by step. This initialization process needs most of the computation time but only has to be done once, so that a simulation can be repeated with a modified set of input parameters with significantly less effort.

Figure 11 shows the time needed for computation for the single simulations. Comparing just the pure effort for one single simulation the coarse grid simulation with the FluidDynamics Library in Dymola took 21.4h while 14.75h were needed for the initialization process and 6.65h for the simulation itself. The CFD simulation with ANSYS Fluent took 29.63h in total. Considering the differences in the hardware of 8 cores computing in parallel in the CFD simulation and one core being used in the coarse grid simulation it can be stated that the FluidDynamics approach needs significantly less resources for computation. This makes the usage of the coarse grid model of the FluidDynamics Library interesting for optimization processes, e.g. for finding a volume flow rate which is as small as possible to fulfill certain safety requirements.

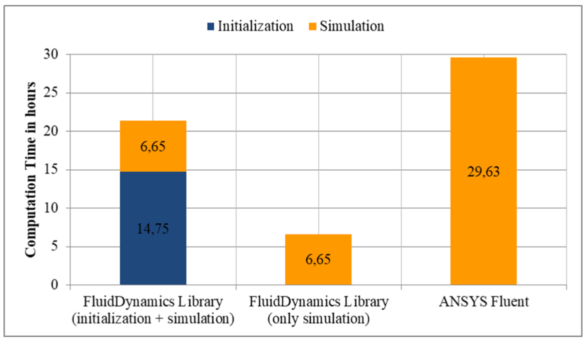


Figure 11. Comparison of computational effort

4 Conclusions

The comparison of the results of the simulated use case shows that the coarse grid approach of the FluidDynamics Library in Dymola delivers very similar results as the detailed CFD approach in ANSYS Fluent does. Even though local effects like plug-holing at extraction points are not captured in the coarse grid results the overall distribution of optical densities and air temperatures correspond very well to the ones computed with ANSYS Fluent.

The significantly less computational effort of the coarse grid approach in the FluidDynamics Library makes it possible to conduct several simulations with the aim of optimization of e.g. necessary volume flow rates even with medium tier hardware. With this advantage the FluidDynamic Library can be used to plan, size and optimize smoke extraction systems in less time and with significantly lower costs than with detailed CFD tools and several iteration loops. In the process of approving a building permission this can cause a noticeable speed-up and also a significant reduction of costs due to the fact that the amount of necessary expensive CFD simulations can be limited to one single loop in the best case.

References

[VDI(2009)] *Engineering methods for the dimensioning of systems for the removal of smoke from buildings*. Verein Deutscher Ingenieure, 2009.

[MPC(2018)] Mpcci - multiphysics interface, 2018. URL `https://www.mpcci.de/`.

[TIS(2018)] Tisc suite, 2018. URL `https://www.tlk-thermo.com/index.php/en/software-products/tisc-suite`.

[Michaelsen(2015)] et al. Michaelsen. Dynamic simulation of an aircraft compartment coupled to a ventilation system. *AST 2015*, 2015.

[Versteeg and Malalasekera(1995)] H. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics, The finite volume method.* Longman Scientific & Technical, 1995.

Modeling and Coordinated Control of an Air-Source Heat Pump and Hydronic Radiant Heating System

Christopher R. Laughman   Scott A. Bortoff   Hongtao Qiao

Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
{laughman,bortoff,qiao}@merl.com

Abstract

Hydronic radiant heating systems embedded in building constructions are receiving increased interest due to their potential for high energy efficiency and improved thermal comfort, but their slow time constants pose challenges when controlling space conditions. We address this problem via a system architecture that combines the radiant heating system with a separate air-source heat pump serving the same space. In this paper, we develop a new coordinating control method for this proposed system by using a set of reduced order models generated from a set of coupled Modelica models of the individual subsystems. This new control architecture does not require significant modification of standard heat pump control architectures, and results in both improved thermal comfort and reduced energy consumption.

Keywords: radiant heat transfer, heat pump, thermally active building systems, control, Modelica

1 Introduction

Buildings continue to be a focused target of societal efforts to reduce carbon-based energy consumption, as they consume a significant fraction of the total energy produced in many countries. Improving energy efficiency of equipment, systems and buildings has been a long-term trend since the mid 1970s. A more recent trend is the accelerated growth of renewable energy generation, which makes electric heat pumps, for example, an increasingly attractive alternative to conventional boilers and furnaces. In addition, increased awareness of the impact that buildings have on the health and safety of their occupants is prompting extensive study of approaches that improve the overall thermal comfort of occupied spaces, rather than just air temperature.

As described in Rhee et al. (2017), this set of concerns has motivated the increased exploration and application of radiant heating and cooling systems. The main benefit of these systems is that they are able to provide equivalent thermal comfort at a lower temperature difference, due to their extended surface areas and the fact that the mean radiant temperature experienced by an occupant contributes significantly to thermal comfort. The smaller difference between the desired room temperature and the source temperature generally makes it possible for heating and cooling systems to operate with higher energy efficiency. These systems also tend to be quiet, as they do not rely upon the transport of air to remove the thermal load, and can be integrated into building systems, such as when the polyethylene tubes transporting the water most often used as a heat transfer medium are embedded in structural concrete slabs used in the building. Such radiant heating and cooling systems that are integrated into the building envelope are often called thermally-active building systems (TABS). The high thermal inertia of TABS is beneficial because it can facilitate the reduction of energy consumption peaks by allowing the space to be conditioned when occupants are absent and the cost of electrical power is low. The corresponding reduction in the required heating and cooling capacity also can reduce the size and cost of system components.

While TABS have a number of advantageous characteristics, the significant thermal inertia associated with the time constants of the slab and low temperature differences in the system impose constraints on their controllability (Gwerder et al., 2008). In particular, the long time delays in the thermal response of the slab, which typically range from hours to days, make it either impractical or impossible to change the slab temperature in response to sudden changes in the room air temperature setpoint or large changes in the room load. Consequently, the dynamic response due to heat load disturbances in a room conditioned by a TABS is much larger than it would be if the room were conditioned by a heat pump or other more conventional system. The slow TABS system lacks the control authority and bandwidth to reject the heat load disturbance.

A variety of previous work has been done to address these challenges in applying radiant heating systems. The control of on/off valves have been an object of particular study due to their low cost; for example, Tang et al. (2018) developed a 3-D discretized model of a TABS to describe the transient thermal dynamics of the slab to study a pulsed flow control strategy under cooling

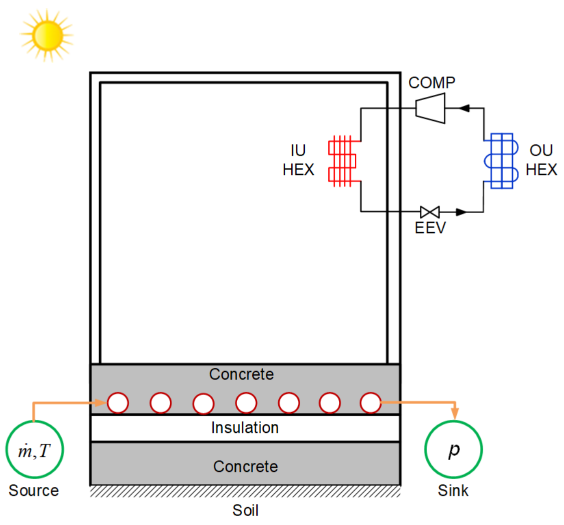


Figure 1. Diagram illustrating building model incorporating heat pump and thermally-active slab for radiant heating.

operation while assuming constant thermal conditions on the slab boundaries. They determine through simulation and experiment that the water flow rate can be reduced by approximately 25% while modulating capacity over a wide range of values. Gwerder et al. (2009) also explores the control of these systems using pulse width modulation strategies with a limited number of tuning parameters, and demonstrates these methods' efficacy in both simulation and experiment. Márquez et al. (2017) investigated a state machine-based control strategy for controlling the heating operation of a TABS, a fan coil unit, or a combination of both systems by using a TRNSYS model of the space and a curve-based model of a heat pump to provide the water supply, and demonstrate that the thermal comfort and energy consumption are strongly dependent upon the control methods used. Beghi et al. (2011) also investigated a relay controller architecture that manages the water flow rate through both radiant slabs and fan coils using a dynamic model of the space and curve-based models of the fan coil to directly control thermal comfort during both heating and cooling operation, and reduced energy consumption by 5-13% via this approach. Finally, Gayeski et al. (2011) used a series of temperature- and load-dependent maps for a variable capacity chiller to perform predictive precooling of a concrete-core radiant floor, and experimentally demonstrated energy savings of 19-25% in Atlanta and Phoenix climate conditions.

One system architecture that addresses the limitations on the control bandwidth of the TABS, which are fundamental to its construction, involves the combination of a TABS and a variable-capacity air-source heat pump, as seen in Figure 1. This combination of systems provides a means to reject thermal disturbances and provide rapid setpoint changes while also providing the increased thermal comfort and efficiency of the TABS. While such an architecture has a variety of benefits, one inherent challenge is that both the heat pump and the TABS will have a high control gain to the temperature and thermal comfort of the room. The operation of the parallel systems must therefore be coordinated to simultaneously minimize the overall energy consumption and maintain thermal comfort.

The dynamic nature of the interaction between these subsystems necessitates the use of dynamic models and controls to identify a feasible design. Because model-based control design practices can be an invaluable tool to quickly and inexpensively explore the control design space to develop and refine candidate controller concepts for nonlinear thermofluid systems, we use a set of dynamic models, as opposed to performance maps, of both the air-source heat pump and the room with the TABS to design the controls for the overall system. As has been noted in Wetter et al. (2016), the equation-oriented language Modelica (Modelica Association, 2017) is well-suited for such system-level analysis and design tasks because of its ability to separate concerns between the activities of physics-based modeling and simulation, and because it facilitates the use of models beyond simulation, such as numerical optimization or linearization for control design. Such capabilities make it possible to make individual dynamic models of the air-source heat pump, the TABS, and the room-air dynamics, and then to couple all of these models together in a heterogeneous simulation for the use in the design of the coordinated control system and subsequent nonlinear simulations.

In this paper, we describe a set of Modelica models that represent the coupled dynamics of an air-source heat pump and a room with a TABS during heating operation, and then describe a candidate approach for coordinating the operation of both systems to meet thermal comfort guidelines while managing the power consumption. Our objective is the design of a control architecture that requires minimal modification of existing heat pump control algorithms, while providing a high degree of energy efficiency and thermal comfort. A secondary objective is the illustration of the use of Modelica models for frequency-domain control system design, which is a use case that is different than time-domain simulation. In Section 2, we describe the models used, first for the air-source heat pump operating in a heating configuration, and then for the thermally-active slab. We also describe a few salient details about the Modelica implementation as well as a number of adaptations required to make the individual system models compatible. We then linearize the nonlinear model and develop a control design for the overall system in Section 3 that meets a specified set of requirements and objectives, and evaluate the performance of this controller in simulation. Finally, we briefly review a set of conclusions and next steps in Section 5.

2 Models

A control-oriented analysis of the dynamics of the proposed heating systems requires a two distinct models: a model of the vapor compression heat pump, and a model of the room with the TABS. As the dynamics of the integrated system are of primary concern, the subsystem models must correctly capture their behavior and encode it in a representation that can be coupled with the other system models. This analysis thus builds on previous modeling work relating to the subsystems; these heat pump models and information about correlations and further details are described at length by Laughman et al. (2015), while the building models are similarly discussed in Wetter et al. (2014). We will briefly review the structure of these subsystem models and then describe the manner by which they were integrated into an overall system model in the following subsections.

2.1 Vapor Compression Heat Pump Model

A conventional variable-capacity air-source vapor compression cycle was used as the basis for the development of this model. This cycle includes a variable speed rotary compressor, a refrigerant-to-air condensing heat exchanger with a variable-speed fan in the occupied space, an evaporating heat exchanger with a variable-speed fan in the ambient environment, and an electronic expansion valve. The availability of four control inputs provides the capability to regulate both the internal refrigerant-side variables of the cycle, such as pressures or condenser subcooling, as well as overall cycle performance variables, such as its heating capacity or outlet air temperature. We also assumed the availability of a variety of temperature measurements for the purposes of control, but did not assume that other measurements were available that could not be realistically included in a commercial product, such as mass flow meters.

In considering the behavior of the cycle, the dynamics of the compressor and electronic expansion valve are much faster than those of the heat exchangers, so that the heat exchanger dynamics dominate the system response over the time scales of interest (multiple seconds to hours). We consequently used static (algebraic) models for the compressor and expansion valve, and dynamic models of the heat exchangers. A finite volume discretization approach was used to describe the behavior of both heat exchangers to ensure the physically correct treatment of mass, momentum, and energy conservation. A number of simplifying assumptions were used in these models; the most important of these is that of one-dimensional refrigerant flow, meaning that the flow field is assumed to be uniform in the r and θ directions at each point along the length of the heat exchanger. Other important assumptions included that of a Newtonian fluid, negligible axial heat conduction along the direction of the fluid flow, negligible viscous dissipation in the fluid,

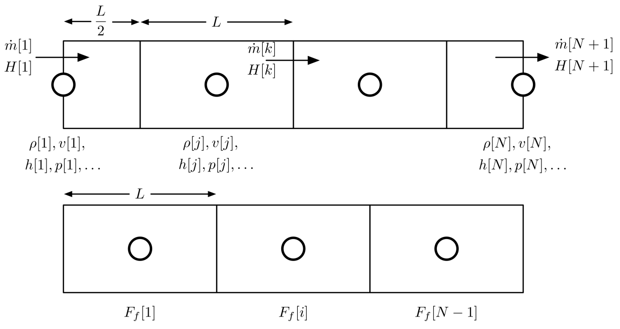


Figure 2. Staggered discretization grid used to model the dynamics of the fluid.

negligible contributions to the energy equation from the kinetic and potential energy of the refrigerant, negligible dynamic pressure waves in the momentum equation, and thermodynamic equilibrium in each volume for which the refrigerant is in the two-phase region.

The resulting conservation equations are then discretized using the upwind approximation and can be written as

$$\frac{d(\rho_j V_j)}{dt} = \dot{m}_k - \dot{m}_{k+1} \tag{1}$$

$$\begin{aligned}\frac{d(\dot{m}_i)}{dt} l = {} & \rho_j v_j^2 A_j - \rho_{j+1} v_{j+1}^2 A_{j+1} + \\ & \frac{A_j + A_{j+1}}{2}(P_{j+1} - P_j) + F_{f,i}\end{aligned} \tag{2}$$

$$\begin{aligned}\frac{\partial(\rho_j u_j A_j)}{\partial t} = {} & H_k - H_{k+1} + \\ & v_j A_j (P_{j+1} - P_j) + v F_{f,i} + Q_j.\end{aligned} \tag{3}$$

These equations are implemented on a staggered flow grid to eliminate numerical oscillations in the pressure, which is illustrated in Figure 2. The indices in Equations 1 through 3 match those used in the figure, where the i indices refer to the momentum grid, the j indices refer to the thermal grid, and the $k = j+1$ indices refer to the boundaries of the thermal grid for each volume V_j. In addition, the term H_k is defined as

$$H_k = \dot{m}_k \bar{h}_{upstream,j}, \tag{4}$$

and the mixed-cup specific enthalpy $\bar{h}$ is equal to the *in situ* specific enthalpy under the homogeneous flow assumption.

A set of simplified closure relations for the frictional pressure drop related to F_f and the refrigerant-side heat transfer coefficients were used to connect those variables to the properties of the refrigerant flow through each control volume. Simplified forms were used because many published correlations from the literature have poor numerical properties that make them very difficult to include in a dynamic simulation. The frictional pressure drop was expressed as

$$\Delta P = K \frac{(\Delta P)_0}{\dot{m}_0^2} \dot{m}^2, \tag{5}$$

where the nominal values of K, $(\Delta P)_0$, and $\dot{m}_0$ were determined by using the Colebrook correlation for the single-phase friction factor and the Friedel correlation for two-phase multipliers. Similarly, phase dependent heat transfer coefficients were used, in which the value of the heat transfer coefficient in each phase was only dependent on the refrigerant mass flow rate, and trigonometric interpolation was used to smooth the transition between phases over a small range of thermodynamic quality (Richter, 2008). The constants used for this simplified correlation were calculated by the Gnielinski correlation for single-phase fluids, the Shah correlation for condensing flows, and the Gungor-Winterton correlation for evaporating flows.

The pressure P and mixture specific enthalpy h in each control volume were used to define the thermodynamic state of the refrigerant, as well as the state derivatives used in the numerical integration routines used to simulate the behavior of the cycle. The derivatives of the mass and energy in the control volume were formulated in terms of these state variables. While this selection of state variables can result in fluctuations in total refrigerant mass in the cycle due to accumulated numerical errors (Laughman and Qiao, 2017), these errors were found to be negligible in this work.

A moist-air formulation was used to describe the heat transfer from the outer surfaces of the tubes to the air, as described in Equation 7, where the mass transfer coefficient was given by a modified Lewis correlation. The humidity dynamics were negligible in these models due to heating mode operation, but were included for compatibility with the room models.

$$\dot{m}_{air} c_{p,air} \frac{dT_{air}}{dy} \Delta y = \alpha_{air} \left(A_{o,tube} + \eta_{fin} A_{o,fin} \right) \left(T_w - T_{air} \right) \tag{6}$$

$$\dot{m}_{air} \frac{d\omega_{air}}{dy} \Delta y = \alpha_m \left(A_{o,tube} + \eta_{fin} A_{o,fin} \right) \times \min\left(0, \omega_{water,sat} - \omega_{air}\right) \tag{7}$$

The cycle model also included a variable-speed high-side rotary compressor, in which the motor is cooled by the compressed high-pressure refrigerant discharged from the compressor mechanism. While mass and energy balance equations can be formulated for this system, a model of realistic machine performance must take into account aspects of the machine behavior that are difficult to capture with low-dimensional models. The compressor was therefore described via the volumetric efficiency η_v and isentropic efficiency η_{is}, as well as a function for the power consumption $\dot{W}(P_{rat}, \omega)$ that relates the total power consumption of the machine to other refrigerant-side parameters, given by

Parameter	Value
Refrigerant	R410A
Total refrigerant mass (kg)	0.529
OU HEX tube diameter (mm)	9.5
IU HEX tube diameter (mm)	7.9
OU HEX tube length (m)	1.1
IU HEX tube length (m)	0.9
OU HEX number of tubes	26
IU HEX number of tubes	18

Table 1. Cycle model parameters.

$$\eta_v = \frac{\dot{m}_{comp}}{\rho_{suc} V f} \tag{8}$$

$$\eta_{is} = \frac{h_{dis,isen} - h_{suc}}{h_{dis} - h_{suc}}, \tag{9}$$

and the formulation of the expressions for η_v, η_{is}, and $\dot{W}$ are provided in Laughman et al. (2017).

A simple isenthalpic model of the electronic expansion valve was also used, as described by a standard orifice flow equation

$$\dot{m} = C_v a_v \sqrt{\rho_{in} \Delta P}, \tag{10}$$

where the flow coefficient C_v can be determined by regression using experimental data and the flow area a_v is specified by the user. Algebraic fan models specified by standard fan laws (ASHRAE, 2008) were also used, and were scaled by experimentally measured values of the fan speed, flow rate, and power for a representative system. The building was also assumed to maintain a constant pressure, allowing us to assume that the supply air flow rate was equal the exhaust air flow rate in the conditioned space.

Following the construction of these component models, they were interconnected to form a complete cycle model. This model was configured with a number of geometric and fluid parameters that governed the behavior of the overall system, which are given in Table 1. The resulting system had a nominal heating capacity of 2631 W at a compressor frequency of 60 Hz, with 2.5 °C of condenser subcooling when the indoor zone was at 26 °C and 2.3 °C of evaporator superheating when the ambient environment was at 2 °C.

2.2 Building Model

The increasing adoption of the Modelica language by many different engineering communities has resulted in the growth of an ecosystem of high-quality open source libraries that can be freely used and incorporated into other models. The Modelica Buildings library (Wetter et al., 2014) is a prime example of such a library, as it has been under development for 10 years with the extensive support of many researchers and organizations and has

Parameter	Value
Room area (m^2)	55.7
Room height (m)	3.66
Concrete slab thickness (m)	0.33
Window size (m^2)	13.9
Building location	Boston, MA, USA

Table 2. Room model parameters.

been extensively tested and used. This library is designed to provide dynamic models of moist air and other media, heat transfer, multizone airflow, and electrical distribution for building and district energy and controls methods; the many component models that are provided in the library can be assembled into models of rooms and single- or multi-story buildings. We employed this library to simulate the building physics in this work; a diagram of this model is illustrated in Figure 1, with pertinent parameters provided in Table 2.

The multiphysical and object-oriented nature of this modeling paradigm is reflected in the structure of the room models developed for this work, as the constructions that constitute the room are composed of fundamental materials and elements that are directly related to the building materials used in practice. Models of the walls and the room are composed of multiple layers representing the individual materials, each of which is parameterized by fundamental material properties like thickness, thermal conductivity, and density. These layers are further discretized into control volumes to approximate the behavior of the partial differential equations describing heat conduction in the material. The default number of control volumes used for a layer of material is automatically chosen, based upon the Fourier number of the material, so that the time constants of each volume are approximately equal. One wall of the room model also incorporates a double-paned window construction over a large portion of its surface. Further information about these models is provided in Wetter (2006).

A similar level of detail was employed to construct the physical models of the zone air and the thermal loads. The zone air model is a mixed air single-node model with one bulk air temperature that interacts with all of the radiative surfaces and thermal loads in the room, where the zone is assumed to have convective, radiative, and latent gains specified on a per area basis. The radiative heat transfer representing the solar heat gains and the infrared heat transfer between the interior surfaces of the room is also modeled with a similar degree of accuracy. The absorptivity and emissivity of each surface is characterized, and is used in the calculation absorbed and reflected heat transfer to all of the surfaces in the room (Wetter et al., 2011). A set of simplified view factors between the surfaces in the room was used as an approximation to avoid the complexity of incorporating the detailed room geometries. Detailed weather data from TMY3 files was also used to describe the ambient conditions, including but not limited to the dry bulb temperature, wet bulb temperature, and direct and diffuse solar radiation terms, to accurately describe the influence of these conditions on the room thermal dynamics.

The overall building model was defined by using the standard interface established in the Buildings library, in which the room constructions can either be defined as part of the room or only defined via their surfaces, to establish a partial model was developed that only defined the surface absorptivity and emissivity properties of the slab. This partial model could thus be extended to incorporate either an inactive slab or a thermally active slab. The inactive slab used a standard construction with a 8 cm layer of concrete, a 5 cm layer of insulation, and a second 20 cm layer of reinforced concrete, whereas the thermally-active slab used a similar structure but also incorporated crosslinked polyethylene (PEX) tubes to provide a radiant thermal input. The overall model for this slab is based on models described in Koschenz and Dorer (1999), in which the PEX tubes embedded in the concrete were modeled using a simplified fluid model, and a limited (5) number of regions were used to discretize the pipe to limit the computational complexity of the overall system. The inclusion of the TABS in our system provided two additional control inputs from which to influence the thermal comfort of the room: the mass flow rate of water through the TABS, and the inlet temperature of that water.

Conventional physical boundaries between subsystems were used as abstraction layers for these models. The cycle model of the heat pump was encapsulated as one discrete model, where the flow and temperature sources for the outdoor unit were wrapped into the overall cycle model and two air-side connectors constituted the main interface between the cycle model and the room model. Similarly, the room model was also encapsulated as a separate discrete model with air-side connectors to interface with the heat pump. This overall system model incorporating the connected room and cycle models was then itself wrapped in another enclosing model that exposed the limited set of inputs and outputs used for the controller design through the generation of linearized system models in Dymola, and performance assessment through the testing of the controller performance on a single plant model.

One technical concern that emerged in the course of this work is that the standard moist air media model used in the Modelica Standard Library (MSL) and the simplified moist air model provided in the Buildings Library are not interchangeable, due to the different assumptions made in these models. While `Modelica.Media.Air.MoistAir` uses a mixture model in which both the air and steam components obey the ideal gas law $P = \rho RT$, `Buildings.Media.Air` simplifies this model further so that the pressure and temperature are decoupled via

$$\rho = \rho_{STP} \frac{P}{P_{STP}}, \tag{11}$$

where STP denotes the moist air properties at a temperature of 273.15 K, a pressure of 101325 Pa, and a humidity ratio of 10%. This simplification is used in the building models because the nonlinear equation blocks in the flattened model are much smaller, resulting in an increase in simulation speed as well as a marginal decrease in accuracy. Because the air-side models of the heat exchangers for the vapor compression cycle did not assume this simplification, we initially used the MSL media model in the building model, but found that the simulation was very slow. Rather than modify the air-side model of the heat exchangers, we slightly modified `Buildings.Fluid.Interfaces.ConservationEquation` so that the mass in the control volume m was dependent only on the pressure in the volume, e.g.,

$$\mathtt{m} = \mathtt{fluidVolume} * \mathtt{medium.p}/\mathtt{P_{STP}} * \rho_{\mathtt{STP}}, \tag{12}$$

where `fluidVolume` represents the volume of the room.

3 Control Design

The set of system models described in Section 2 can be used to design a control system that coordinates the operation of the heat pump and the TABS. This design is based on frequency-domain methods and demonstrates an important use case of Modelica models that goes beyond time-domain simulation.

Contemporary variable-capacity heat pumps have four control inputs that are used to regulate their behavior, including the compressor speed f_c, an electronic expansion valve (EEV) position, and the indoor and outdoor fan speeds. We assume that the heat pump is controlled using two PID feedback loops: one that actuates f_c based on room air temperature T_a, and a second that actuates EEV based on an internal measurement y_1 such as condenser subcooling. In comparison, the indoor fan speed is typically controlled by the customer, and the outdoor fan speed is used for purposes outside our scope. Because this and other similar control laws are well-established in commercial heat pumps, we do not wish to modify this architecture in the coordinated controller. Furthermore, we assume that the inlet water temperature T_w is the available control input for the TABS.

3.1 Linearization and Model Reduction

Our control design process begins by computing a linearization of the overall system that includes the heat pump, the TABS and the building at a nominal operating condition. The models described the previous section were coupled to refrigerant property models from the Vapor Cycle Library Modelon AB (2018), compiled in Dymola 2018 FD01 AB, and run on a PC with an Intel i7 processor with 16 Gb of RAM. We simulate the model with constant inputs for 5×10^6 s (58 days) to ensure that the linearization is computed an equilibrium solution. This ensures convergence of the slowest mode of the system, which has a time constant of approximately 1 week. For our particular system the resulting raw linearized model has 344 states and is 99.5% sparse.

The raw model is then carefully reduced in dimension. Frequency-based design methods such as loop-shaping (or any formal control design method, for that matter) can fail if the raw model is used directly. This is because the raw model is highly sparse, stiff and numerically ill-conditioned. In fact, many Matlab Control Toolbox commands, e.g., `bode`, reduce the model automatically, with little visibility to the user, and can produce misleading or erroneous results.

Our model reduction process requires several steps. We begin by symbolically eliminating the states associated with refrigerant energy and mass using the variable names that accompany the state vector in the linearized model, as these states are unobservable, uncontrollable and have eigenvalues at the origin. This results in a 240-dimensional model. We then compute a balanced realization of the system and its associated Hankel singular values, which are typically logarithmically spread as a function of frequency, and then truncate the balanced model, keeping approximately 15 states in a partially reduced-order model. As a result of this calculation, we frequently encounter very slow non-physical, weakly controllable and observable modes which we speculate are associated with the closed-loop topology of the vapor compression cycle and airflow in buildings. These non-physical modes are removed by modal decomposition. In addition, we also compute a singular perturbation if the partially reduced-order model includes pressure states. This process typically produces a 10^{th}-order reduced-order model that is numerically well-conditioned and whose spectrum has modal time constants ranging from 20 s to 1.5 days, as is expected from practical experience.

3.2 Dynamic Analysis

Figure 3 shows the frequency response of the reduced model from the two actuators, f_c and T_w, to the room air temperature T_a and room mean radiant temperature T_{rad}. We see that f_c has strong authority over T_a over a frequency range $0 - 0.01\,\text{rad/s}$ (10 min), but *neither* input is effective at controlling T_{rad} beyond $0.0002\,\text{rad/s}$ (8 hours). In fact, both T_w and f_c are equally effective at regulating T_{rad} at frequencies below $0.0002\,\text{rad/s}$. Fortunately, the disturbance bandwidth is correspondingly similar, as can be seen in Figure 4, which shows the frequency responses from the radiative and convective heat loads Q_{rad} and Q_{sen}, respectively, to the temperatures T_a and T_{rad}. This figure demonstrates that both disturbances

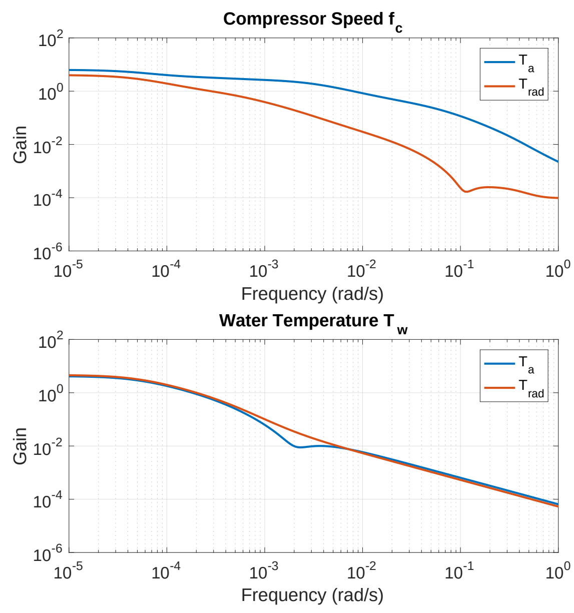


Figure 3. Frequency response of T_a and T_{rad} to controls f_c and T_w.

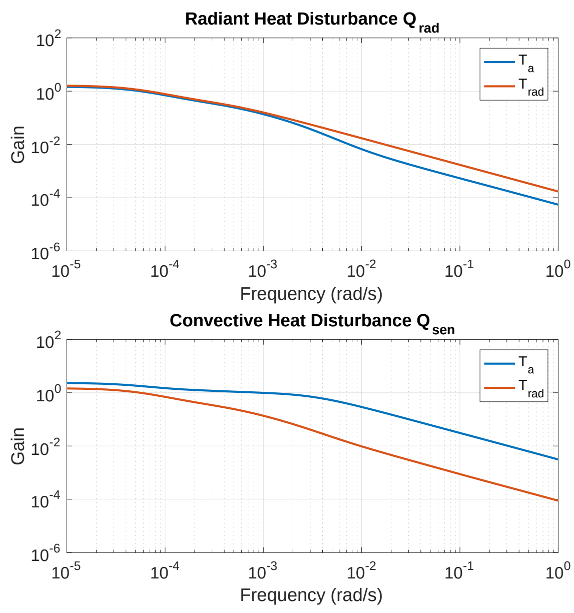


Figure 4. Frequency response of T_a and T_{rad} to disturbances Q_{rad} and Q_{sen}.

have a low-bandwidth effect on T_{rad}, while only the gain from Q_{sen} to T_a has a relatively broad bandwidth. Moreover, both are within the effective bandwidths of the controls.

3.3 Controller Design

Controlling both T_a and T_{rad} is important because perceived comfort, as measured by the predicted mean vote (PMV) (ASHRAE, 2017), weights both T_a and T_{rad} equally for low levels of human metabolic activity associated with typical office conditions. Room occupants are therefore equally sensitive to both T_a and T_{rad}, though only T_a is typically measured. These facts, together with our requirement to use the production heat pump control algorithm, suggests the control architecture shown in Figure 5.

We consider the heat pump and radiant systems as two separate actuators that both contribute to T_a and T_{rad} (not shown). In this architecture, the PID controller is designed to regulate the room air temperature to the setpoint r by controlling the compressor frequency of the heat pump f_c while neglecting the radiant system entirely. This PID command is filtered through a low-pass filter (LPF) whose bandwidth is tuned to approximately 4 hours to speed-up the radiant system response. The LPF splits the frequency content of the PID output into a low-frequency component, which is applied to the radiant system, and a high-frequency component that is applied to the heat pump. The compensator C_2 is set equal to P_1/P_2, where P_1 is a low-order approximation of the transfer function from f_c to T_a, and P_2 is a low-order approximation of the transfer function from T_w to T_a. These approximate models are computed from the frequency

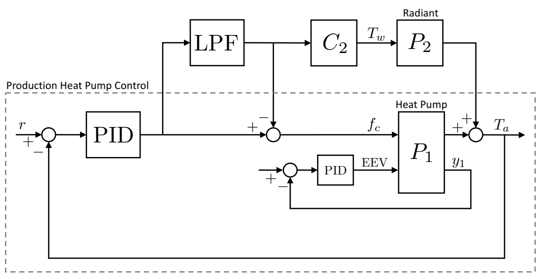


Figure 5. Closed-loop block diagram of the overall system. The production heat pump control is inside the dashed box.

responses described in the previous section and are both minimum-phase, so that C_2 is stable and realizable. We find that a first-order P_2 is sufficient, while P_1 is second-order. The compensator C_2 effectively provides phase lead to the radiant system, so that $C_2 \cdot P_2$ is approximately equal to P_1, which implies that the frequency response of the compensated radiant system is equal to that of the transfer function from f_c to T_a. The low-pass filter ensures that only low-frequency signals are applied to the compensated radiant system. One benefit of this architecture's simplicity is that the only parameters required by the controller are the low-order models P_1 and P_2, as well as the gain and bandwidth of the low-pass filter that controls the frequency split and steady-state load split between the heat pump and radiant system. Such an architecture does not require any significant modification of the heat pump controller, aside from subtracting the output of the LPF from the output of the PID, and applying the result to f_c. Note that the complete design depends on only the frequency response.

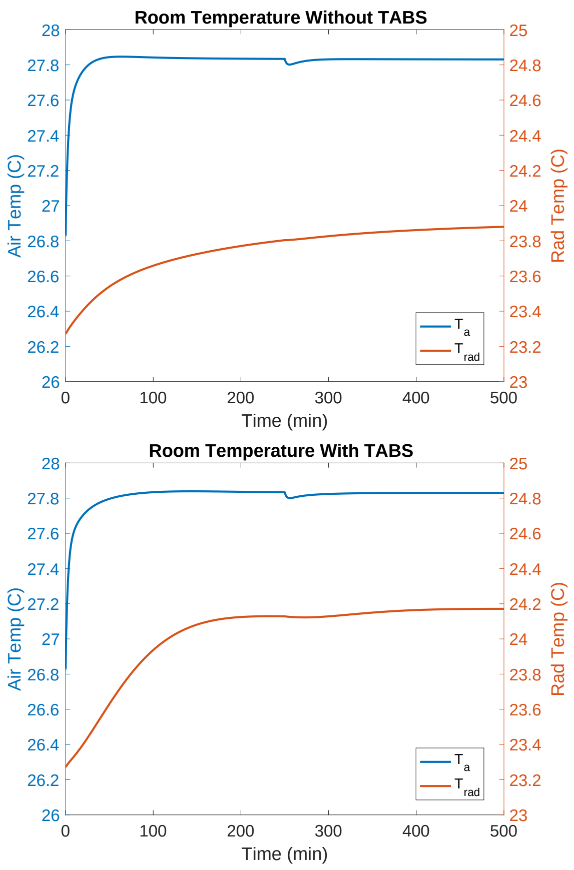


Figure 6. Room temperature step response. The air temperature reference is stepped 1 °C at $t = 0$, and a heat load disturbance step of $Q_{sen} = -1\,\text{kW}$ is applied at $t = 250\,\text{min}$. T_a response is similar for both, but the radiant response is sped-up TABS with coordinated control (bottom).

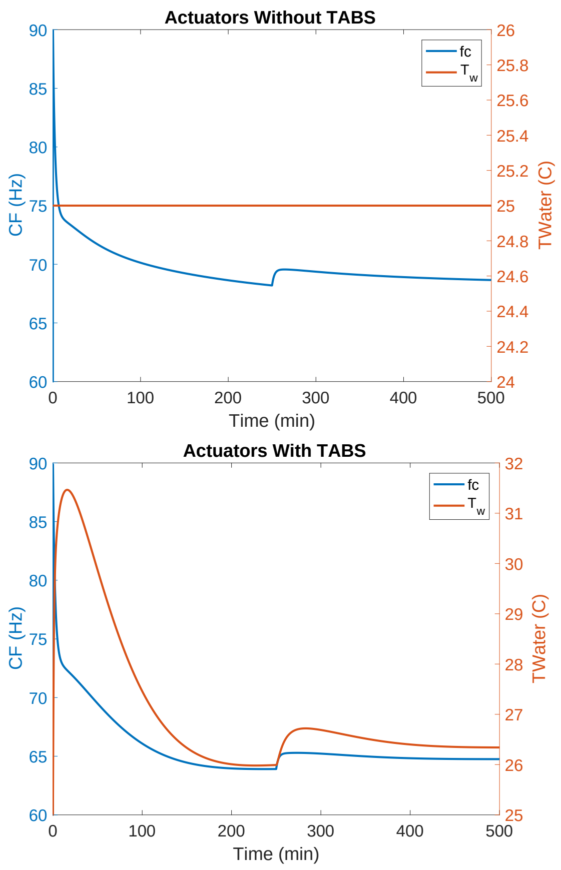


Figure 7. f_c and T_w Actuation. The air temperature reference is stepped 1 °C at $t = 0$, and a heat load disturbance of $Q_{sen} = -1\,\text{kW}$ is applied at $t = 250\,\text{min}$. f_c is similar for the short term, but f_c is reduced because of the effect of the radiant system (bottom), and in the steady-state, the load is split among the systems. The effect of the lead filter on T_w is apparent and not too large.

4 Results

We first simulate the operation of the system without the TABS. The shortcomings are readily seen by considering the responses of T_a and T_{rad} to a step in the room temperature setpoint (at $t = 0$ s) and to a $-1\,\text{kW}$ heat load step disturbance at $t = 250$ min, as seen in Figure 6. While the air temperature T_a responds quickly and the disturbance is rejected, the radiant temperature responds much more slowly because of the limited control authority of the heat pump over the surface temperatures of the room. This slow change in T_{rad} implies that a building occupant will feel about 0.5 °C colder for a sustained period of time after the room temperature setpoint change due to the slow radiant temperature response, despite the fact that the air temperature responds quickly.

In comparison, the performance of the heat pump and TABS system under coordinated control is much improved. The response of T_a is nearly identical to that of the previous case, but the response of T_{rad} is significantly faster. While the time constant of T_{rad} for the room without coordinated control is nearly 6 hours, the time constant of T_{rad} is closer to 150 minutes when the two systems are operated together. This results in a much smaller difference between T_a and T_{rad}, and consequently improved thermal comfort. The disturbance rejection properties of the system are still quite good, as would be expected due to the fact that the heat pump is largely responsible for these dynamics.

The actuator commands for this simulation are shown in Figure 7. Whereas the steady-state compressor frequency f_c is approximately 68 Hz for the base control case, this is reduced by approximately 5 Hz for the coordinated control architecture. This reduction in compressor frequency is also accompanied by a reduction in the needed heating capacity of the heat pump in steady-state for the coordinated control case. As a result, the benefits of this coordinated control method are both a reduction in the power consumption of the heat pump, and potentially a smaller capacity heat pump is required. In addition, consideration of the T_w signal indicates that the improvement in the T_{rad} response is achieved with only a 6 °C increase in the inlet water temperature, which is easily achieved by commercially available water heating systems.

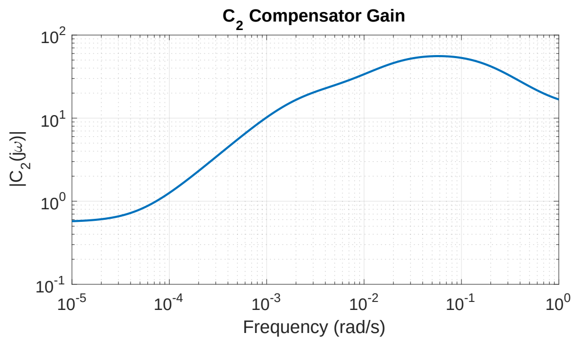


Figure 8. Frequency response of C_s showing lead behavior between 10^{-4} and 10^{-2} rad/s.

5 Conclusions and Future Work

In this paper we have described a Modelica model of a heat pump and radiant heating system integrated into a building, and used the model to construct a coordinating feedback control algorithm using frequency-domain methods. This represents a good use-case example of Modelica models beyond that of time-domain simulation. The use of these linearized models, which are readily computed using tools such as Dymola or OpenModelica, requires some care because they represent large-scale, sparse, and stiff systems. We show a simple coordinated controller for the combined radiant and heat pump system provides improved comfort and responsiveness with a minimum impact on the production control algorithm in the heat pump.

This result can be extended in a variety of different directions. For example, it would be interesting to consider acausal low-pass filters for frequency separation, which may be possible if the references are known apriori. Such an approach constitutes a form of predictive control, and may improve the results and allow further downsizing of the heat pump. Formal MPC methods could also be applied to this same end. Alternate directions could also include the coordination of this system architecture with ventilation systems, as well as consideration of cooling mode operation for these system architectures in light of the effects of humidity.

References

Dassault Systemes AB. Dymola. 2018 FD01.

ASHRAE. *HVAC Systems and Equipment Handbook*. ASHRAE, Atlanta, GA, 2008.

ASHRAE. Thermal environmental conditions for human occupancy. ASHRAE 55:2017, Atlanta, GA, USA, 2017.

A. Beghi, L. Cecchinato, and M. Rampazzo. Thermal and comfort control for radiant heating/cooling systems. In *IEEE Conference on Control Applications*, 2011.

N.T. Gayeski, P.R. Armstrong, and L.K. Norford. Predictive pre-cooling of thermo-active building systems with low-lift chillers. *HVAC&R Research*, 18(5):858–873, 2011.

M. Gwerder, B. Lehmann, J. Tödtli, V. Dorer, and F. Renggli. Control of thermally-activated building systems (TABS). *Applied Energy*, 85:565–581, 2008.

M. Gwerder, J. Tödtli, B. Lehmann, V. Dorer, W. Güntensperger, and F. Renggli. Control of thermally activated building systems (TABS) in intermittent operation with pulse width modulation. *Applied Energy*, 86:1606–1616, 2009.

M. Koschenz and V. Dorer. Interaction of an air system with concrete core conditioning. *Energy and Buildings*, 30:139–145, 1999.

C.R. Laughman and H. Qiao. On the influence of state selection on mass conservation in dynamic vapor compression cycle models. *Mathematical and Computer Modelling of Dynamical Systems*, 23:262–283, 2017.

C.R. Laughman, H. Qiao, V. Aute, and R. Radermacher. A comparison of transient heat pump cycle models using alternative flow descriptions. *Science and Technology for the Built Environment*, 21(5):666–680, 2015.

C.R. Laughman, H. Qiao, S.A. Bortoff, and D.J. Burns. Simulation and optimization of integrated air-conditioning and ventilation systems. In *Proc. of the 15th IBPSA Conference*, pages 660–669, 2017.

A.A. Márquez, J.M.C. López, F.F. Hernández, F.D. Muñoz, and A.C. Andrés. A comparison of heating terminal units: Fan-coil vs. radiant floor, and the combination of both. *Energy and Buildings*, 138:621–629, 2017.

Modelica Association. Modelica specification, Version 3.4, 2017. URL `www.modelica.org`.

Modelon AB. *Vapor Cycle Library User Guide*, 2018. v2.0.

K-N. Rhee, B.W. Olesen, and K.W. Kim. Ten questions about radiant heating and cooling systems. *Building and Environment*, 112:367–381, 2017.

C.C. Richter. *Proposal of New Object-Oriented Equation-Based Model Libraries for Thermodynamic Systems*. PhD thesis, Technical University of Braunschweig, 2008.

H. Tang, P. Raftery, S. Schiavon, J. Woolley, and F.S. Bauman. Performance analysis of pulsed flow control method for radiant slab system. *Building and Environment*, 127:107–119, 2018.

M. Wetter. Multizone building model for thermal building simulation in Modelica. In *International Modelica Conference*, pages 517–526, 2006.

M. Wetter, W. Zuo, and T.S. Nouidui. Modeling of heat transfer in rooms in the Modelica "Buildings" library. In *Proceedings of Building Simulation 2011*, pages 1096–1103, 2011.

M. Wetter, W. Zuo, T. Nouidui, and X. Pang. Modelica Buildings library. *Journal of Building Performance Simulation*, 7 (4):253–270, 2014.

M. Wetter, M. Bonvini, and T.S. Nouidui. Equation-based languages – a new paradigm for building energy modeling, simulation, and optimization. *Energy and Buildings*, 117: 290–300, 2016.

Semiconductor Package Thermal Impedance Extraction for Modelica Thermal Network Simulation Combined with VHDL-AMS model

Eiji Nakamoto[1] Kentaro Maeda[2] Takayuki Sekisue[3]

ANSYS Japan K.K., Japan, {eiji.nakamoto,kentaro.maeda,takayuki.sekisue}@ansys.com

Abstract

Because of the emerging market demand for higher power with higher efficiency for the power semiconductor devices, thermal design of the semiconductor package and its cooling method has become one of the key elements for the power supply systems in power electric design. Thus, many thermal designers now require the junction-to-case thermal Impedance $Z_{\theta JC}$ since it is one of the most important thermal characteristics of semiconductor devices and thus, in November 2010, the more reliable and sufficiently reproducible measurement method without a case temperature measurement has been standardized by JEDEC as JESD 51-14

(https://www.jedec.org/standards-documents/docs/jesd51-14-0)

This paper shows the new feature in ANSYS simulation tool, ANSYS Electronics Desktop, which extracts $Z_{\theta JC}$ from JESD 51-14 compliant measurement data. The extracted $Z_{\theta JC}$, or its cumulative expression of thermal resistance $\sum_{i=1}^{n}(R_{thi})$ and capacitance$\sum_{i=1}^{n}(C_{thi})$ called "structure function", is transformed to the Modelica thermal ladder network model. This Modelica model was simulated by ANSYS TwinBuilder, Multi-domain system simulator, and the junction temperature is reproduced by this simulation, that agreed well with the original measured temperature data. Further, $Z_{\theta JC}$ is split into two components, Junction-to-Die part(IC package DUT) and Heat-sink part(cold plate) in accordance with the guideline of Transient Dual Interface Measurement Procedure principle described in JESD 51-14. Then, $Z_{\theta JC}$ corresponding to IC package structure part is transformed to the VHDL-AMS model (as IC Package thermal compact model) while Heat-sink structure part is transformed to Modelica model(as testing fixture structure model) . Those models built by two well-known physical model description languages were connected with the acausal (i.e., conservative) condition in ANSYS TwinBuilder and the thermal response of the combined model is evaluated. The result of the simulation matches to the full Junction-to-Heat-sink Modelica thermal ladder network model, that ensures Modelica and VHDL-AMS models can be connected in a single physical multi-domain system simulation environment in ANSYS TwinBuilder under the energy conservative principle, that might expand the potential applicability and the coverage for Modelica simulation for the broader application area.

Please send an email to eiji.nakamoto@ansys.com if there are any questions or suggestions regarding this paper.

Keywords: IC Package transient dual interface (TDI) measurement, IC Package Thermal Impedance, Structure Function, VHDL-AMS, Acausal Connection

1 Introduction

The new feature to extract structure function of semiconductor package is developed for ANSYS Electronics Desktop product in compliant with JESD 51-14. For its validation, comparison tests using sample measured data were implemented.

1.1 Extraction of Structure Function for IC Package and Validation Test

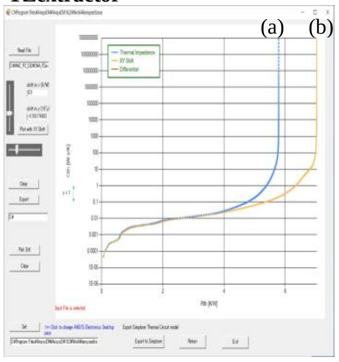
There are details described about Thermal Impedance $Z_{\theta JC}$ and the methodology of extracting IC Package structure function in JESD 51-14. In compliance with this JEDEC standard, the Visual C++ based software program for extracting structure function from the TDI measurement data (named TZextractor feature in ANSYS Electronic Desktop) was developed.

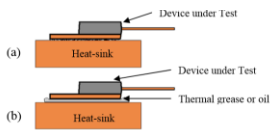
Dual interface measurement models, i.e. with / without thermal grease on the contact surface between Die-pad and Heat-sink , were evaluated. Fig.1 shows the result of the validation tests compared with the sample semiconductor package models in Software TDIM-MASTER[1], which serves as reference and example implementation of the algorithms described in JESD 51-14. The extracted structure functions by TZextractor are well matched to those of TDIM-Master (Figure. 1).

1.2 Thermal Network and State Space Models

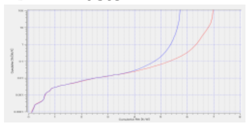
Now, another IC model shown in Figure 2 (2SK2173 MOSFET, TOSHIBA) was evaluated for the further testing and validation. Initial input heating current

TZextractor



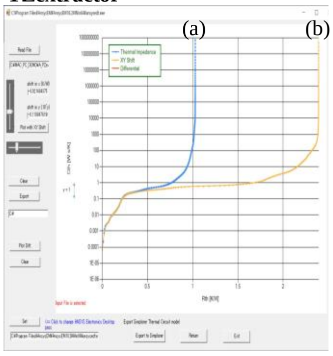


TDIM-Master

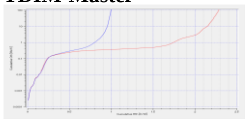


HBRIDGE Package model

TZextractor



TDIM-Master



MOSFET Package model

Figure 1. Structure Function by TZextractor and TDIM-Master with(b)/without(a) thermal grease for HBRIDGE and MOSFET package models

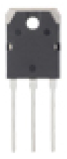


Figure 2. DUT for validation test (2SK2173 MOSFET)

of 20 Ampere was applied to DUT(Device Under Test). Figure 3 shows the transient response of junction temperature of DUT measured after the input current is switched-off. The structure function of DUT is extracted from this data by TZextracor and compared with the output data given by the other commercially available JESD 51-14 compliant TDI measurement system (Figure 4).

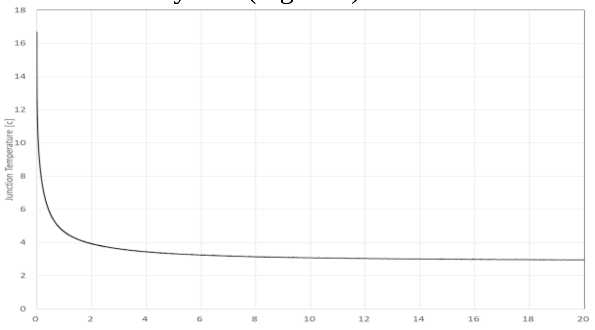


Figure 3. Junction Temperature by TDI measurement

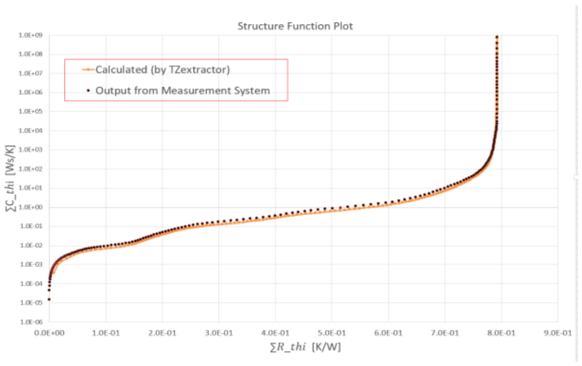


Figure 4. Structure Function of 2SK2173 MOSFET by TZextractor vs Meaurement System

2. Thermal Network and State Space Models

Then, Modelica model of Cauer type thermal RC circuit network is generated from $Z_{\theta JC}$ data for 2SK2173 MOSFET model (Fig.5) in ANSYS TwinBuilder.

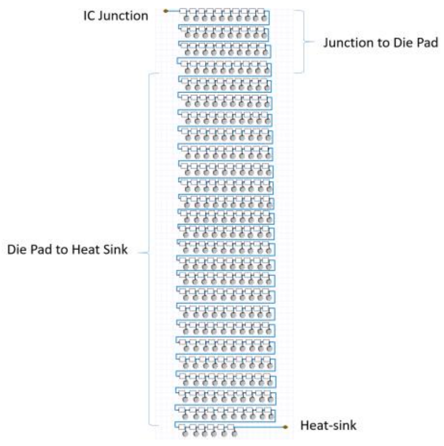
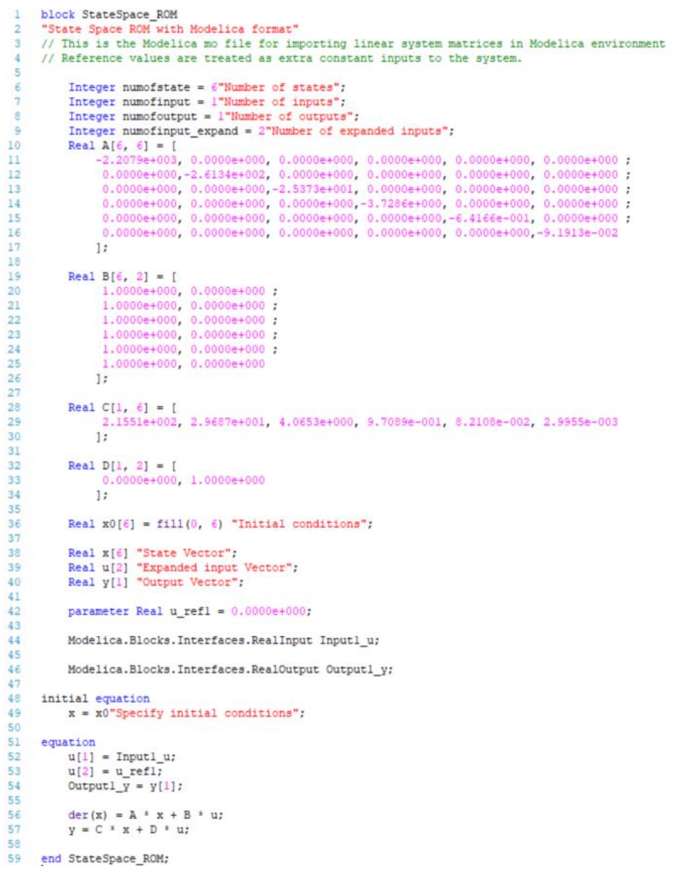


Figure 5. Thermal RC circuit network from $Z_{\theta JC}$

On the other hand, TDI measurement data of the junction temperature after power switch-off state was reversed to artificially represent the step response of the power switch-on state. Then, LTI (Linear Time Invariant) State Space mode from this step response was automatically generated as Modelica ROM (Reduced Order Model) by ANSYS TwinBuilder(Figure 6). Then, the input power of 17.14 Watt was applied for the first 100 second until the package Junction temperature reaches to the steady state temperature and then, the power was turned-off to replicate the TDI measurement condition. The junction temperature of the simulation models were monitored and compared with the TDI measurement data and the results are compared each other as shown in Figure 7.

ROM model shows good match to the measurement data after two second from switch-off while Thermal RC network model matches well from 0.5 to 1.0 second period. However, thermal RC network models shows a slight deviation when the decreasing speed of the junction temperature becomes slowed down after one second from the switch-off until reaching to the steady state temperature. It may be because JESD 51-14 theory assumes only one directional heat flow pass between Junction and Heat-sink by thermal RC series network, that may not be good enough to represent the three dimensional thermal characteristic of the Package at the lower temperature dissipation state.



```
block StateSpace_ROM
"State Space ROM with Modelica format"
// This is the Modelica mo file for importing linear system matrices in Modelica environment
// Reference values are treated as extra constant inputs to the system.

    Integer numofstate = 6"Number of states";
    Integer numofinput = 1"Number of inputs";
    Integer numofoutput = 1"Number of outputs";
    Integer numofinput_expand = 2"Number of expanded inputs";
    Real A[6, 6] = [
        -2.2079e+003, 0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000 ;
         0.0000e+000,-2.6134e+002, 0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000 ;
         0.0000e+000, 0.0000e+000,-2.5373e+001, 0.0000e+000, 0.0000e+000, 0.0000e+000 ;
         0.0000e+000, 0.0000e+000, 0.0000e+000,-3.7286e+000, 0.0000e+000, 0.0000e+000 ;
         0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000,-6.4166e-001, 0.0000e+000 ;
         0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000, 0.0000e+000,-9.1913e-002
        ];

    Real B[6, 2] = [
         1.0000e+000, 0.0000e+000 ;
         1.0000e+000, 0.0000e+000 ;
         1.0000e+000, 0.0000e+000 ;
         1.0000e+000, 0.0000e+000 ;
         1.0000e+000, 0.0000e+000 ;
         1.0000e+000, 0.0000e+000
        ];

    Real C[1, 6] = [
         2.1551e+002, 2.9687e+001, 4.0653e+000, 9.7089e-001, 8.2108e-002, 2.9955e-003
        ];

    Real D[1, 2] = [
         0.0000e+000, 1.0000e+000
        ];

    Real x0[6] = fill(0, 6) "Initial conditions";

    Real x[6] "State Vector";
    Real u[2] "Expanded input Vector";
    Real y[1] "Output Vector";

    parameter Real u_ref1 = 0.0000e+000;

    Modelica.Blocks.Interfaces.RealInput Input1_u;

    Modelica.Blocks.Interfaces.RealOutput Output1_y;

initial equation
    x = x0"Specify initial conditions";

equation
    u[1] = Input1_u;
    u[2] = u_ref1;
    Output1_y = y[1];

    der(x) = A * x + B * u;
    y = C * x + D * u;

end StateSpace_ROM;
```

Figure 6. Modelica State Space Thermal model for 2SK2173 MOSFET

3 Modelica with other modeling language models

VHDL-AMS is the international standard description language registered by IEEE and ICE, that has a long history since its origin in ’70 and is one of the well-known modeling languages used in semiconductor, electrical and electronics design arena. There are many open or IP protected models and libraries. Because almost all the machines, hardware products and systems are now electrically controlled by the controller (ECU, PLC) with the electrical drive trains (actuators, motors) by using power electric technology, so it shall be beneficial if Modelica simulation could be connected or has an interface to other modeling language models such as VHDL-AMS.

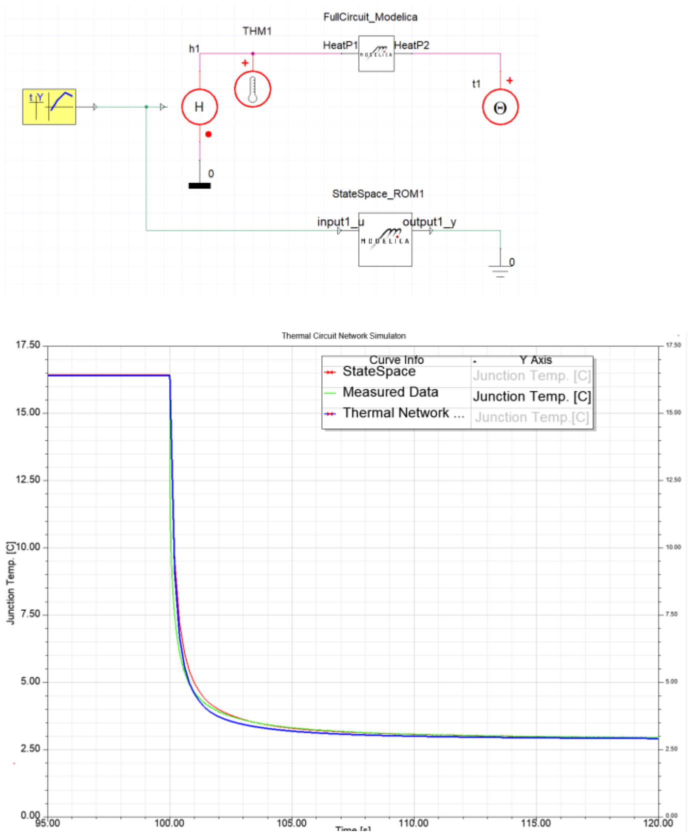


Figure 7. Junction temperature after switch-off from steady state. Simulation results of Modelica Thermal Network and State Space models compared with TDI Measurement data

3.1 Modelica model combined with VHDL-AMS

To connect those two different models, an interface port that enables Modelica acausal pins interpreted to VHDL-AMS conservative pins are available in ANSYS TwinBuilder. Figure 8 is the result of the simple test for mechanical mass/spring/damping component models with different Modelica-to-VHDL-AMS combinations. As result shows, the response of mass displacement are all the same for three different model connections which ensure the validity of the interface port.

3.2 Validation check with IC package model

According to JEDEC 51-14, $Z_{\theta JC}$ can be split into two components, IC package part and Heat-sink part by TDI Measurement Procedure principle. As for the validation, test model of 2SK2173 MOSFET is evaluated. In this model, the first thirty series RC thermal network shall corresponds to IC package and the latter part of the network shall be Heat-sink according to the two different measurements (i.e. with and without thermal grease on DUT Die-pad). Then, IC package part is made by VHDL-AMS model while Heat-sink part is made by Modelica model in ANSYS TwinBuilder. Those two different models are connected via interface ports and simulated. The simulation of combined thermal network models shows identical result to the full Modelica network model (Figure 9).

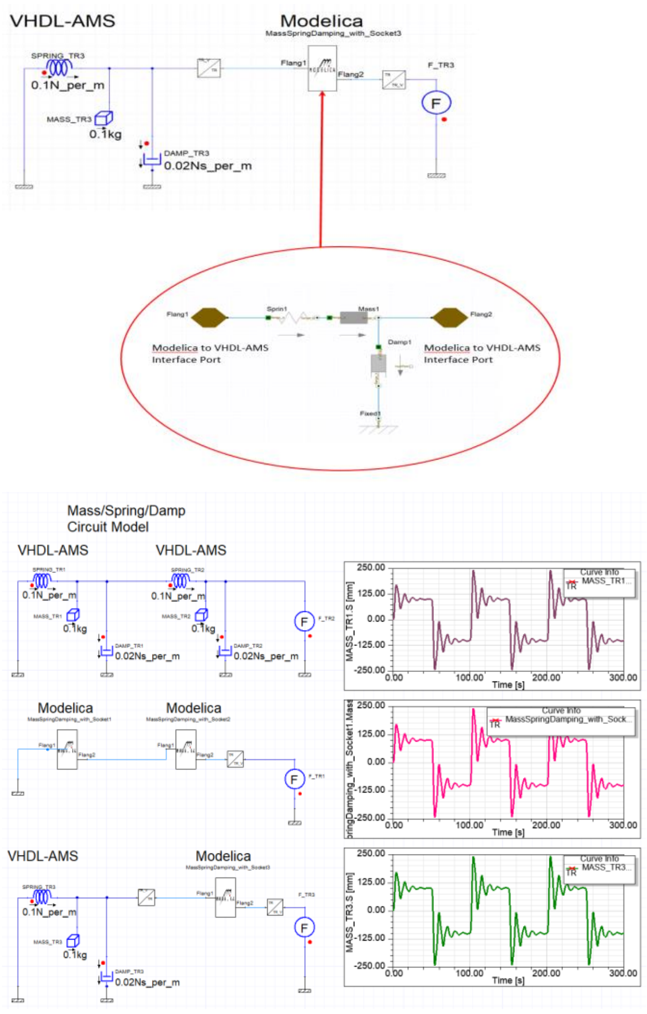

VHDL-AMS
Modelica
Modelica to VHDL-AMS Interface Port
Mass/Spring/Damp Circuit Model
0.1N_per_m
0.1kg
0.02Ns_per_m

Figure 8. Mechanical mass/spring/damping component models with different combinations by Modelica-to-VHDL-AMS interface port

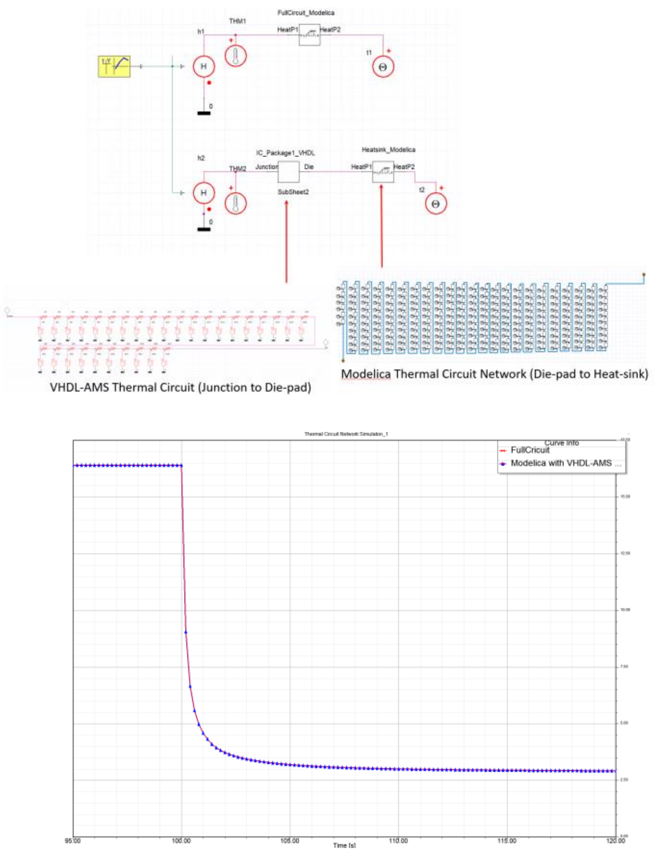


Figure 9. Combination of Modelica and VHDL-AMS thermal network components compared with full Modelica network model

4 Summary

The new feature to extract Structure Function of semiconductor package was developed which is available for ANSYS Electronics Desktop product. This feature was validated by the measurement data in compliant with JESD 51-14 standard of JEDEC. Extracted result was transferred to Modelica thermal network and State Space models and the result of the thermal circuit simulations were compared with the original temperature measurement data with reasonable correlation. Then, the extracted thermal network data was split into two thermal structure models by different modeling language, Modelica and VHDL-AMS. Those are connected via Modelica acausal to VHDL-AMS conservative interface port in ANSYS TwinBuilder. Result of the combined model simulation identically matched to the full Modelica thermal network model.

Acknowledgements

Modelica thermal network model for semiconductor package can be created from JEDEC JESD 51-14 standard measurement, that is useful to evaluate its thermal characteristics by thermal network circuit simulation. The network model can also be created for other modeling language such as VHDL-AMS.

Combination of VHDL-AMS models with Modelica may be useful to expand its applicability and coverage for broader application area.

References

1. D. Schweitzer, Software TDIM-MASTER: Program for the evaluation of transient dual interface measurements of Rth-

JC. This software serves as reference and example implementation of the algorithms described in this standard and can be downloaded from the JEDEC homepage: http://www.jedec.org.

2. JEDEC JESD51-14, “Transient Dual Interface Test Method for the Measurement of the Thermal Resistance Junction-to-Case of Semiconductor Devices with Heat Flow Through a Single Path”, 2010

Modeling of Fuel Cell Hybrid Vehicle in Modelica: Architecture and Drive Cycle Simulation

Sara Sigfridsson[1] Lixiang Li[2] Håkan Runvik[3] Jesse Gohl[2] Antonin Joly[4] Kristian Soltesz[1]

[1] Department of Automatic Control, Lund University, Sweden,
sara.sigfridsson.152@student.lu.se, kristian.soltesz@control.lth.se

[2]Modelon Inc, USA, {lixiang.li, jesse.gohl}@modelon.com

[3]Modelon SE, Sweden, hakan.runvik@modelon.com

[4]Modelon KK, Japan, antonin.joly@modelon.com

Abstract

This paper highlights recent development of fuel cell hybrid vehicle (FCHV) models using the Fuel Cell Library (FCL), the Vehicle Dynamics Library (VDL), and Electrification Library (EL) from Modelon. A flexible model architecture is implemented to support physical modeling of such large scale, multi-domain vehicle system. The top-level model consists of a hydrogen fuel cell subsystem with detailed power characteristics and humidification, a hybrid powertrain including battery, converter and electric motor, a vehicle model with chassis and brakes, and a driver model. Drive cycle simulations are performed using these models to analyze system dynamics under different operating conditions.

Keywords: fuel cell, hybrid vehicle, vehicle modeling, drive cycle simulation

1 Introduction

This section introduces the motivations of the work and provides background about the Fuel Cell Library (FCL), the Vehicle Dynamics Library (VDL), and Electrification Library (EL) used to build the FCHV models.

1.1 Motivations

As air pollution and global warming are becoming increasingly severe world-wide and well recognized by the public, hydrogen is gradually gaining attention as a future alternative to traditional fossil fuels in vehicle propulsion system due to its zero emission of pollutants and CO_2. As part of the electrification trend in the automotive industry, development of hydrogen fuel cell cars is booming in recent years: most car manufacturers have announced investment in research on FCHV, from component design to system integration. Some have already launched their FCHV products on the market.

While a massive amount of work has been published on detailed modeling, water management and control strategies of vehicle fuel cell stack, few can be found on building flexible and complete FCHV model architecture to evaluate the overall performance of the system, not to mention full drive cycle simulation for FCHV. Such effort, however, is crucial to get more insights into system integration of FCHV and to stimulate innovations in the whole product development cycle. The goal of this paper is to take the first step towards complete FCHV system modeling using well-validated, industrial level Modelica libraries: the FCL, VDL, and EL (see Figure 1).

1.2 The Fuel Cell Library

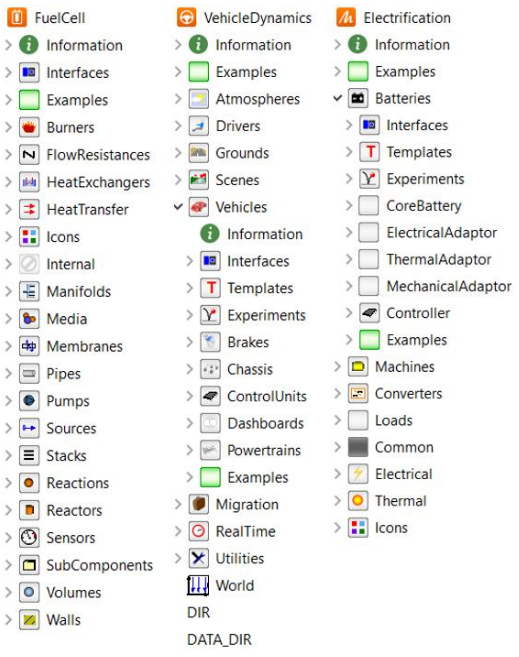
The Fuel Cell Library, or FCL, is a commercial and licensed Modelica® library by Modelon. It is compliant with multiple Modelica tools including JModelica/OCT, ANSYS Simplorer, Ricardo IGNITE and Dymola. It is well suited for component sizing, system design and analysis, control development, and optimization of fuel cell systems for both stationary and mobile applications [1][2][3]. Solid Oxide (SOFC) and Proton Exchange Membrane (PEMFC) fuel cells are included as examples in the library with expandable templates for other types of fuel cells. The library also contains a large number of ready-to-use components for modeling chemical reactions in pre-reforming reactors and fluid distribution in fuel/air manifolds.

1.3 The Vehicles Dynamic Library

The Vehicle Dynamics Library, or VDL, is another commercial and licensed Modelica® library by Modelon. It is intended for vehicle dynamics analysis related to mechanical design and control design of automotive chassis. The library can be used to analyze partial and complete vehicles. Models are designed in a way that is similar to the structure of real vehicle assemblies. The focus is on chassis, wheels, driver, and road but basic models for e.g. engines, transmissions, drivetrains, and brakes are supplied to offer full vehicle modeling and simulation capability in a single environment.

1.4 The Electrification Library

The Electrification Library is a new commercial and licensed Modelica® library from Modelon. The library provides a number of typical components needed when modeling electrified systems including vehicles (like full electric, battery hybrid, and fuel cell vehicles), aircraft, auxiliary power systems, etc. This includes batteries, machines, converters, and loads. It also includes thermal implementations of the components in order to support full thermal management modeling. The library uses a flexible architecture for the components to allow any level of fidelity to be used for any portion of the components as needed throughout the design process.


FuelCell
- Information
- Interfaces
- Examples
- Burners
- FlowResistances
- HeatExchangers
- HeatTransfer
- Icons
- Internal
- Manifolds
- Media
- Membranes
- Pipes
- Pumps
- Sources
- Stacks
- Reactions
- Reactors
- Sensors
- SubComponents
- Volumes
- Walls

VehicleDynamics
- Information
- Examples
- Atmospheres
- Drivers
- Grounds
- Scenes
- Vehicles
  - Information
  - Interfaces
  - Templates
  - Experiments
  - Brakes
  - Chassis
  - ControlUnits
  - Dashboards
  - Powertrains
  - Examples
- Migration
- RealTime
- Utilities
- World
- DIR
- DATA_DIR

Electrification
- Information
- Examples
- Batteries
  - Interfaces
  - Templates
  - Experiments
  - CoreBattery
  - ElectricalAdaptor
  - ThermalAdaptor
  - MechanicalAdaptor
  - Controller
  - Examples
- Machines
- Converters
- Loads
- Common
- Electrical
- Thermal
- Icons

Figure 1. Content in the FCL, VDL, and EL.

2 System Model

This section of the paper outlines the top-level system architecture and key subsystems in the FCHV model. They are viewed as a proof-of-concept to build a large-scale, multi-domain system model for analysis of the system dynamics of an FCHV in drive cycles, which will support various levels of model fidelity and will facilitate control development in the future. Hence, no customer-proprietary data was involved in the parameterization of the models. The vehicle model was based on existing sedan component models from VDL. Most of the other parameters were based on open product information of the Toyota Mirai [4][5] or estimated from the authors' experience.

2.1 Top-level System Architecture

The top-level FCHV system model consists of a fuel cell subsystem, a hydrogen tank, a hybrid powertrain, a vehicle model, a driver model, a controller and other system level information (weather, road, drive cycle data etc.). The control bus is used to pass control signals to various subsystems while expandable connectors of different domains, e.g. electrical, fluid, mechanical etc., allows physical coupling of different subsystems. The architecture shown in Figure 2 represents the topology of the FCHV system. All the subsystems are replaceable, which supports future development of more detailed models. In VDL, there are templates that place the powertrain in the vehicle model, however, we decided to expose the hydrogen tank, fuel cell stack, and powertrain on the top-level in the case of FCHV for easier access to these key subsystems.

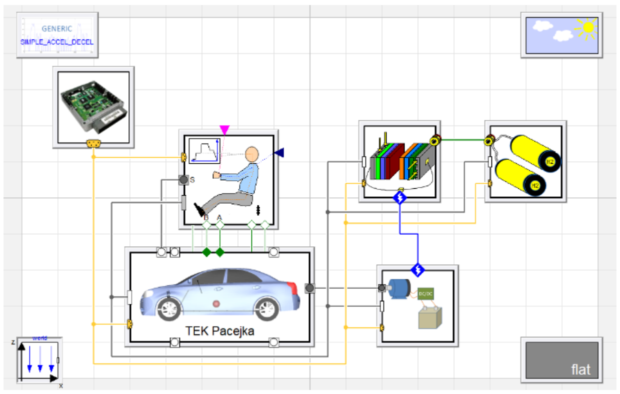


Figure 2. Top-level architecture of the Hydrogen fuel cell hybrid vehicle model.

2.2 Fuel Cell Subsystem

The fuel cell subsystem mainly consists of a proton exchange membrane fuel cell (PEMFC) stack, a humidifier, a heat exchanger and several fluid machines for the air flow, hydrogen flow, and coolant flow, as shown in Figure 3.

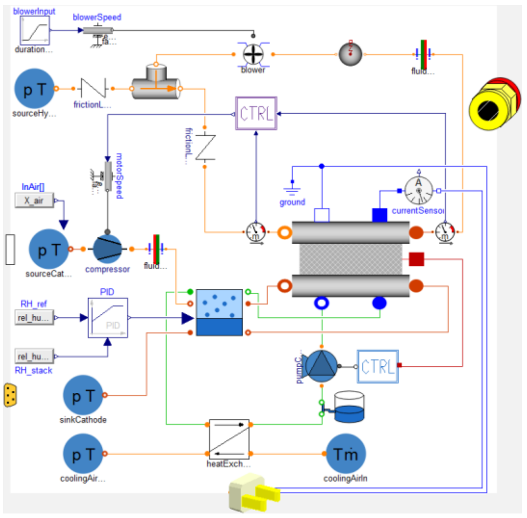


Figure 3. Hydrogen fuel cell subsystem.

On the anode side, excessive hydrogen at the exit is circulated to improve efficiency and to avoid using after-burner. On the cathode side, the air flow rate is controlled using a P-controller based on the consumption of hydrogen in the anode and stoichiometric calculation. The cooling water pump is controlled by a PID controller based on the stack temperature.

In order to keep the cells hydrated, incoming air from the atmosphere goes through a humidifier before reaching cathode channel. The humidifier makes use of the produced water coming out from the cathode to maintain the relative humidity of the air supply at the desired level. More details on the humidifier are discussed in Section 2.2.3.

The stack model has a hierarchical structure and includes a cooling channel, anode, cathode and the cell membrane. The anode and cathode channels are modeled as lumped volumes with condensation effects. Gases are transported into the membrane which handles calculations of power characteristics and water transport across the membrane. Geometric based, distributed model can be further developed by adopting existing templates of discretized membrane and flow channels from FCL.

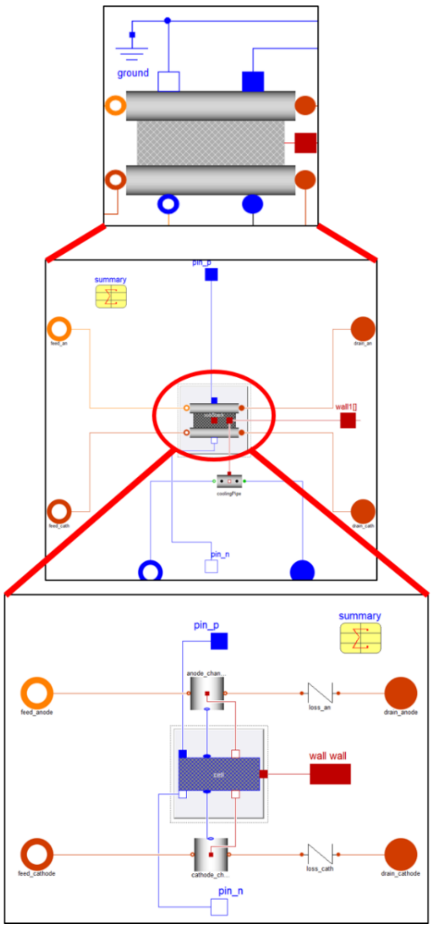


Figure 4. Hierarchical structure of the fuel cell stack.

2.2.1 Power Characteristics

The open circuit voltage of the stack is obtained by the Nernst equation:

$$E = -\frac{\Delta G_f{}^0}{2F} + \frac{RT}{2F}\left(\frac{p_{H_2} p_{O_2}{}^{0.5}}{p_{H_2O}}\right) \quad (1)$$

where the $\Delta G_f{}^0$ [J/mol] is the change in molar Gibbs free energy of formation of hydrogen fuel cell at standard pressure; T [K] is the temperature of the cell; p_{H_2}, p_{H_2} and p_{H_2O} [Pa] are partial pressure of H_2, O_2 and H_2O vapor. R is the ideal gas constant and F is the Faraday constant.

Activation loss, Ohmic loss, and concentration loss are implemented using the following models respectively [6]:

$$\Delta V_{act} = \frac{RT}{2\alpha F}\ln\left(\frac{i+i_n}{i_0}\right) \quad (2)$$

$$\Delta V_{Ohmic} = ir \quad (3)$$

$$\Delta V_{conc} = m\exp(ni) \quad (4)$$

i_0 $[\mathrm{A/m^2}]$ is the exchange current density; i_n is the fuel crossover equivalent current density; i is the cell current density; r $[\Omega\cdot\mathrm{m^2}]$ is the area-specific resistance dependent on the thickness and water content of the membrane [7]; m [V] and n $[\mathrm{m^2/A}]$ are empirical constants for concentration loss. The open circuit voltage with irreversibility is obtained as

$$V = E - \Delta V_{act} - \Delta V_{Ohmic} - \Delta V_{conc} \quad (5)$$

The polarization curve of a single cell and the total power curve of the FC stack model used in drive cycle simulations are shown in Figure 5 and Figure 6 respectively.

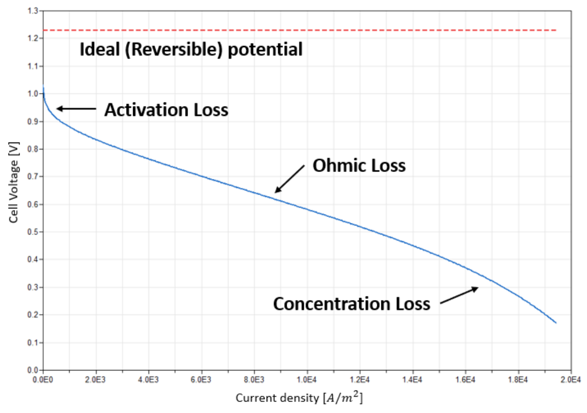


Figure 5. Polarization curve of a single cell at 80 degC.

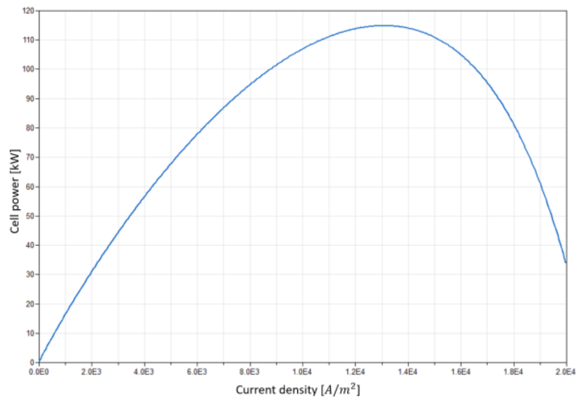


Figure 6. Total power of the FC stack at 80 degC.

Voltage efficiency of the fuel cell can be defined as

$$\eta_{voltage} = \frac{E}{E^0} = \frac{E}{1.229V} \qquad (6)$$

which is a measurement of irreversible losses of the fuel cell. Although the total stack power peaks at a current density of 1.3e4 A/m^2, the corresponding voltage efficiency is less than 40%. So, the current drawn from the stack need be limited by the hybrid drivetrain control algorithm to avoid significant irreversible losses, which is discussed in Section 2.3.

2.2.2 Water transport across the Cells

Back diffusion and electro-osmotic drag are both considered for water transport across the membrane[7]:

$$N_{osmotic} = n_d \frac{i}{F} \qquad (7)$$

$$N_{diff} = D_w \left(\frac{c_{cath} - c_{an}}{t_m}\right) \qquad (8)$$

$$N_{membrane} = N_{diff} - N_{osmotic} \qquad (9)$$

where $N_{osmotic}$ [mol/(s · m^2)] is water flow from anode to cathode due to electro-osmotic drag; N_{diff} is water flow from cathode to annode caused by back diffusion; D_w [m^2/s] is the diffusion coefficient of water; c_{cath} [mol/m^3] and c_{an} are the water concentration at the cathode and anode; t_m [m] is the thickness of the membrane. The water mass flow rate [kg/s] across the membrane for a single cell can be obtained by

$$\dot{m}_{water} = 18.01528 \cdot N_{membrane} \cdot A_{cell} \qquad (10)$$

2.2.3 Humidifier

The humidifier model, shown in Figure 7, assumes ideal mixing and instantaneous thermodynamic equilibrium.

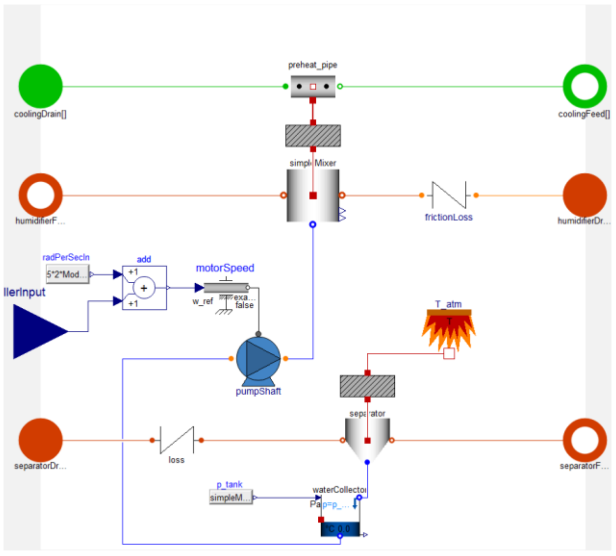


Figure 7. Humidifier model assuming ideal mixing.

The exhaust gas out of the cathode is cooled by the ambient in the separator so that water can be condensed and collected (flow route at the bottom of Figure 7). Then the condensed water is sent to the ideal mixer to humidify the supplied air. The humid air is pre-heated by the cooling water coming out of the stack, in order to maintain the stack temperature better and to increase the saturation vapor pressure of water.

This model can be replaced by diffusion-based membrane humidifier model [8][9][10] in the future.

2.3 Hybrid drivetrain

The hybrid drivetrain is constructed in a similar configuration to Toyota Mirai [11]: both the fuel cell and the battery are connected to DC/DC converters to match the maximum motor voltage. While Mirai has an AC motor and an inverter, the Modelica model uses a DC motor model because the high-frequency dynamics are not of interest in this study.

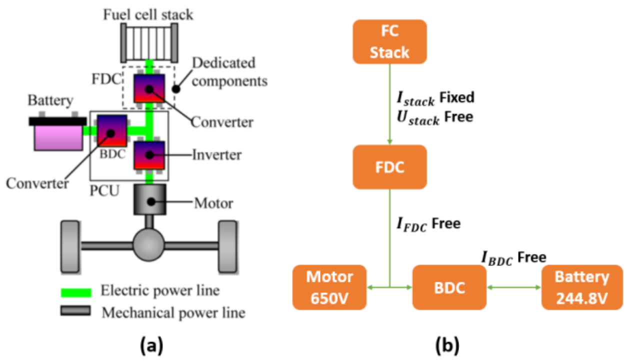


Figure 8. (a) Hybrid drivetrain of TOYOTA Mirai (picture taken from [11]). (b) Schematic plot of the hybrid drivetrain implemented in the Modelica model

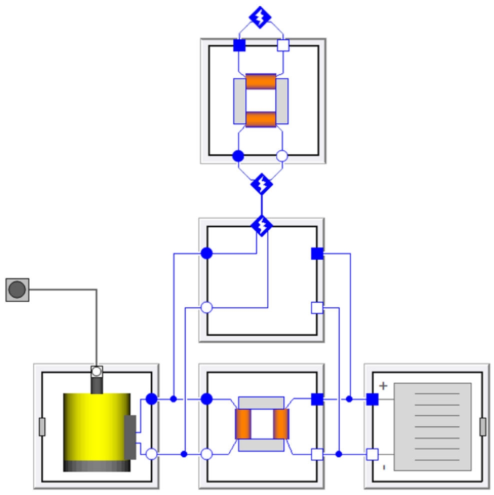
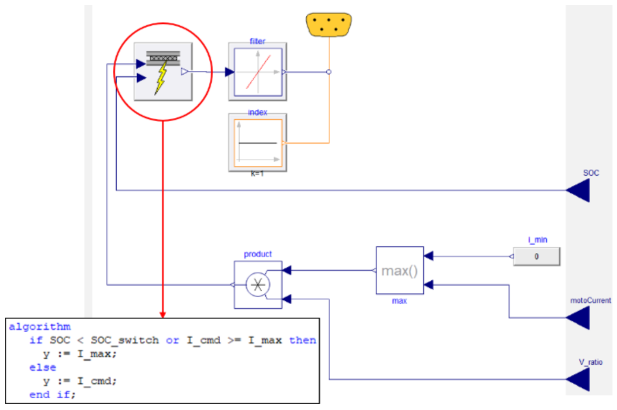


Figure 9. Hybrid drivetrain including battery, battery converter, fuel cell converter, and DC motor.

Mathematical formulation and data of the Ni-MH battery (1.6V, 6.5Ah per cell) model can be found in [12], hence not to repeat here. The state of charge (SOC) of the battery and the motor current demand I_{cmd} are used to determine the current drawn from the fuel cell. When the demand is above the maximum current of the fuel cell or if the SOC is below a threshold SOC_{switch} , the fuel cell will work at maximum current; otherwise, the fuel cell will directly drive the motor at the demanded current. A low-pass filter is added to remove high frequency oscillation in the control signal for fuel cell current.



```
algorithm
  if SOC < SOC_switch or I_cmd >= I_max then
    y := I_max;
  else
    y := I_cmd;
  end if;
```

Figure 10. The control algorithm for the fuel cell. *I_cmd* is the motor current demand, *I_max* is the maximum current of the fuel cell and *y* is the control signal of current drawn from the fuel cell.

2.4 Vehicle Model

The fuel cell, drivetrain, vehicle, and hydrogen storage models are the focus at the top level of the system model, see Figure 2. The vehicle is based on standard component models from the VDL including the chassis, drivetrain, and brakes. These standard models are used in the template shown in Figure 11. Also shown in this figure is an expansion of the chassis model which is a standard implementation included in VDL of a sedan with Pacejka tire models and tabular elasto-kinematic front and rear suspension models. The drivetrain and brake models are also standard passenger sedan implementations included in VDL. The drivetrain is a front-wheel, two-wheel drive implementation. The brake model was a power-assisted, four-wheel disk brake implementation. The only customization needed in this case was a template with a connector at the top level for the mechanical 3D rotational connection to the motor. With the template created all the components were simply selected for use in the complete vehicle model.

2.5 Driver and Drive cycle

The driver model is also a standard model included in VDL that follows a desired vehicle speed trace defined by a drive cycle profile. While a number of standard cycles are available, the profile used for this test was the 600-s supplemental federal test procedure, SC03 cycle as shown in Figure 13. The driver model produces the accelerator and brake commands necessary to follow this cycle. These pedal commands are combined in the motor controller to produce the motor torque command and the foundation brakes provide supplemental braking to the motor torque.

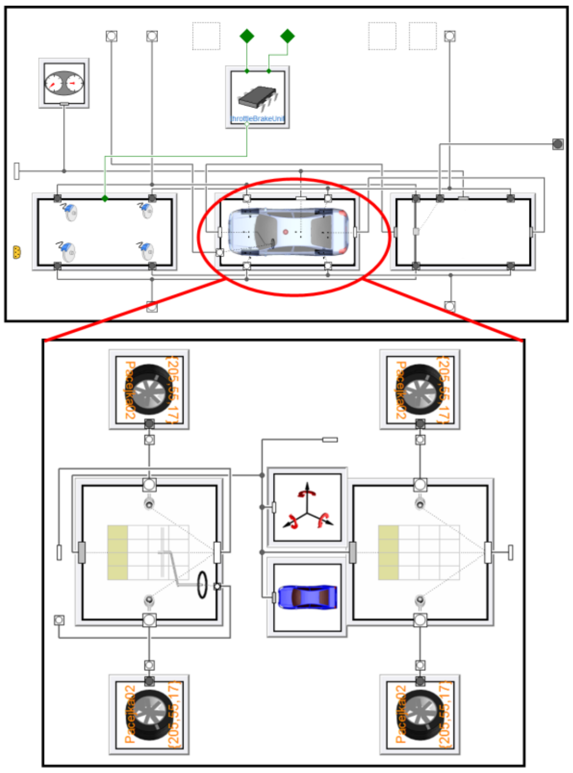


Figure 11. Vehicle model template from the VDL.

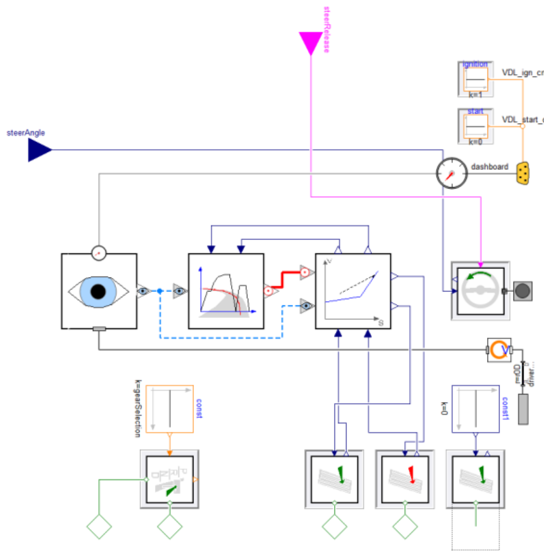


Figure 12. Driver model from the VDL.

3 Drive Cycle Simulations

The SC03 drive cycle speed profile, shown in Figure 13, is used for verification of the FCHV system model. The cycle represents a 5.8 km drive that lasts for 596 s (600 s in the simulation, zero speed for the last 4 s) with an average speed of 34.8 km/h, the maximum speed of 88.2 km/h. The system model includes all the subsystems shown in Figure 2 except the high-pressure hydrogen storage tank and valve subsystem.

The simulations are performed in Dymola 2018 FD01 on a Dell Precision M2800 laptop with 2.8GHz CPU and 16GB RAM.

3.1 The Parameterization

Key parameters in the model are listed in Table 1-4.

Table 1. Parameters of the fuel cell membrane.

Parameters and unit	*Value*
Number of cells (in series)	370
FC Stack weight [kg]	56
Exchange current density i_0 [A/m^2]	0.12
Fuel crossover current density i_n [A/m^2]	15
Empirical constant *m* [V]	3e-4
Empirical constant *n* [m^2/A]	3.2e-4
Membrane thickness [μm]	100
Active cell area [m^2]	0.05
Parameters for Ohmic loss from [7]	

Table 2. Parameters of the hybrid drivetrain [11][12].

Parameters and unit	*Value*
Number of cells (in series) in the battery	204
Cell nominal voltage [V]	1.2
Cell nominal capacity [Ah]	6.5
Motor maximum power [kW]	113
Motor peak torque [N·m]	335
Motor maximum speed [rad/s]	340
Motor nominal voltage [V]	650
DC/DC converter efficiency	90%

Table 3. Parameters of the vehicle model [4].

Parameters and unit	*Value*
Curb weight [kg]	1850
Wheelbase [m]	2.7813
Tires: P215/55R17	

Table 4. Operation condition of the FC stack.

Parameters and unit	*Value*
FC stack T_{ref} [degC] for coolant controller	60
Hydrogen inlet pressure [bar]	2
Air inlet pressure [bar]	1.013
Reference relative humidity of air for humidifier controller	0.9

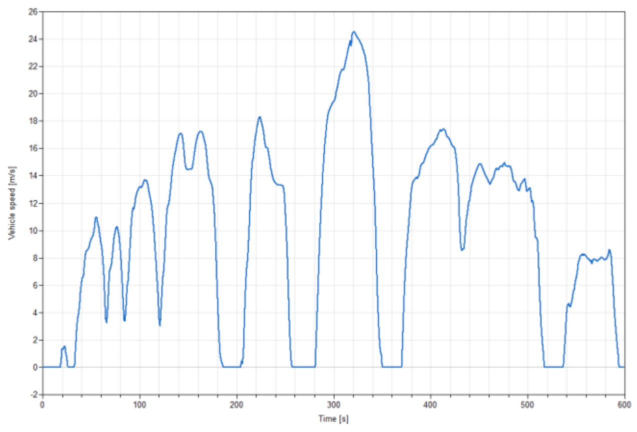


Figure 13. Speed profile of SC03 cycle.

3.2 Results and discussions

The results obtained from the SC03 drive cycle simulations can be found in Figure 14 - Figure 17. Dynamics of the system at different ambient air condition is analyzed and discussed in this section.

In the simulations, Hydrogen is supplied at 25 degC and 2bar. Ambient air comes in at fixed atmospheric pressure but various temperatures (5, 25 and 38 degC) to represent different seasons. The air gets compressed and humidified before it is fed to the cathode. The compressor draws a different amount of air from the ambient according to the consumption of hydrogen in the fuel cell (the demanded current load equivalently). Pressures in the anode and cathode flow channels can be found in Figure 14.

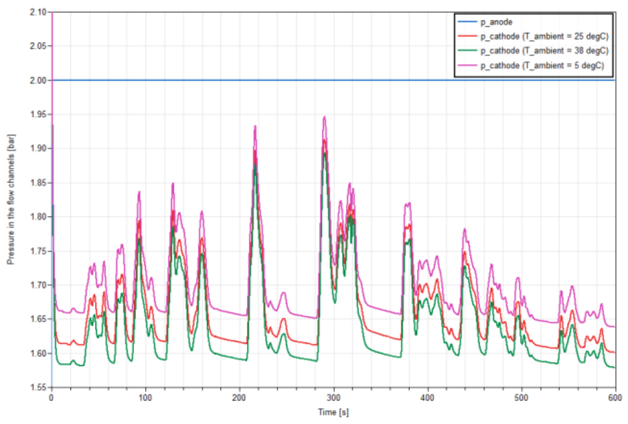


Figure 14. Pressure in the anode channel (Blue) and in the cathode channel (Purple: $T_{ambient}$ = 5 degC, Red: $T_{ambient}$ = 25 degC, Green: $T_{ambient}$ = 38 degC).

The power output of the FC stack and battery, as well as the battery SOC, are plotted in Figure 15. SOC of the battery is initialized at 0.6. In general, the battery responds much faster to electric load than the fuel cell does. When the vehicle accelerates hard, the motor draws current from both the battery and the fuel cell. When the electric load decreases, the fuel cell has a lag in response, so the generated electricity is stored in the battery, resulting in an increase in SOC.

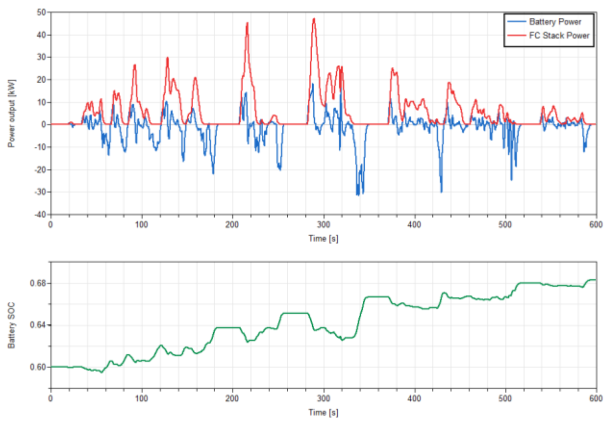


Figure 15. Power outputs and battery SOC during the SC03 cycle with $T_{ambient} = 25$ dgC.

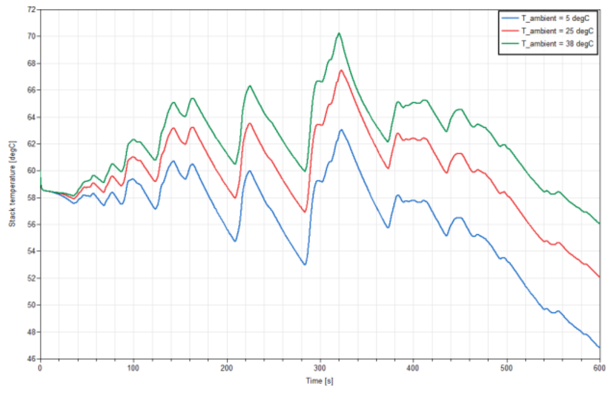


Figure 16. Stack temperatures during the SC03 cycle under different ambient air temperatures: Blue – 5 degC, Red – 25 degC, Green – 38 degC.

Temperature control of the FC stack is challenging, as illustrated in Figure 16. The stack temperature is initialized at 60 degC and varies according to the current demand. In 290 s to 310 s, the vehicle accelerates from 0 km/h to 88.2 km/h, which put a large load on the fuel cell. As a result, the stack temperature increases rapidly by about 10 degC. When the stack power output is too low, for example in the last 100 s in the drive cycle, the stack temperature drops significantly.

Figure 16 also compares stack temperatures during the drive cycle with different ambient air temperature. It shows that the stack temperature is sensitive to the ambient air temperature. More sophisticated coolant pump control strategies and an additional air pre-heater are needed to better maintain the stack temperature.

The computational performance of the FCHV system model can be seen in the CPU time plot in Figure 17. The 600-s drive cycle simulation is finished in 390s CPU time, which is less than 2/3 of the real time. Hence, the FCHV system model as it stands is real-time capable and it will be of great interest for controller development and HIL application. Further improvement of computation speed can be achieved as demonstrated in [2].

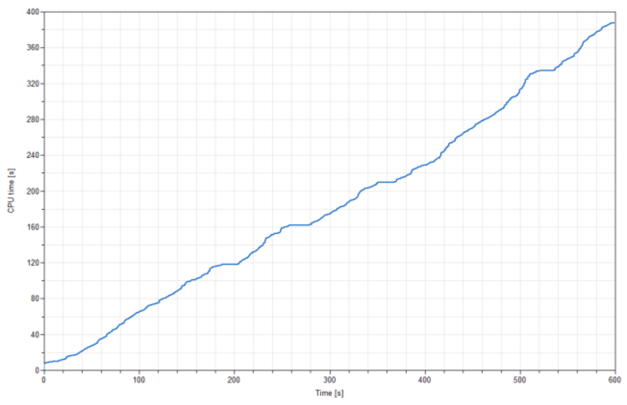


Figure 17. CPU time plot of the SO03 cycle simulation.

4 Conclusions and Future Development

A flexible model architecture for Hydrogen FCHV has been developed using a coordinated suite of Modelica libraries: FCL, VDL, and EL. It covers both physical modeling (across thermal, fluid, electrical, and mechanical domains) and controls modeling. Drive cycle simulations of the SC03 cycle are performed using the FCHV system model for verification. The model is found to be real-time capable. Such simulations demonstrate the use of sophisticated Modelica libraries to enable multi-physical modeling of the full FCHV system for future design, control, and optimization purposes.

With the flexible architecture developed in this work, subsystem or component models of higher fidelity can be developed and plugged into the system as a future improvement, for example:

- Discretized FC stack with cross-flow channels
- Diffusion-based membrane humidifier
- High-pressure Hydrogen tank with pressure regularization valve and its control system
- Hydrogen recirculation using an ejector.

Beyond dynamic simulations and analysis of the FCHV, the use of such full FCHV system model for other applications is also worth exploring:

- Study of temperature and water management of the fuel cell under real drive cycle
- Component selection and system integration
- Exploration of hybrid drivetrain configurations
- Controller development and optimization.

References

[1] Andersson, D., Åberg, E., Eborn, J., Yuan J. and Sundén, B. Dynamic Modeling of a Solid Oxide Fuel Cell System in Modelica. *Modelica 2011 Conference*, Dresden, Germany, pp.593-602, Mar. 20-22, 2011.

[2] Fröjd, K., Axelsson, K., Torstensson, I., Åberg, E., Osvaldsson, E., Dolanc, G., Pregelj, B., Eborn, J. and Pålsson, J. Development of a Real-Time Fuel Processor Model for HIL Simulation. *Modelica 2014 Conference*, Lund, Sweden, pp.675-682, Mar. 10-12, 2014.

[3] Åberg E., Pålsson, J., Fröjd K., Axelsson K., Dolanc G., Pregelj B. HIL simulations of a Real-Time Fuel Processor Model. *5th European PEFC & H2 Forum 2015*, Lucerne, Switzerland, June 30-July 3, 2015.

[4] TOYOTA 2017 Mirai Product Information. https://ssl.toyota.com/mirai/assets/core/Docs/Mirai%20 Specs.pdf

[5] Nonobe, Y. Development of the fuel cell vehicle Mirai. *IEEE Transactions on Electrical and Electronic Engineering*, 12, pp. 5–9, 2017

[6] Dicks, A.L. and Rand, D.A., Fuel cell systems explained，2nd edition. John Wiley & Sons, 2018.

[7] Pukrushpan, J.T., Modeling and control of fuel cell systems and fuel processors. *PhD Thesis*, Ann Arbor, Michigan, USA: University of Michigan, 2003

[8] Chen, D. and Peng, H., A thermodynamic model of membrane humidifiers for PEM fuel cell humidification control. *Journal of dynamic systems, measurement, and control*, *127*(3), pp.424-432, 2005.

[9] Chen, D., Li, W. and Peng, H., An experimental study and model validation of a membrane humidifier for PEM fuel cell humidification control. *Journal of Power Sources*, *180(1)*, pp.461-467, 2008.

[10] Solsona, M., Kunusch, C. and Ocampo-Martinez, C., Control-oriented model of a membrane humidifier for fuel cell applications. *Energy conversion and management*, 137, pp.121-129, 2017

[11] Hasuka, Y., Sekine, H., Katano, K., and Nonobe, Y., Development of Boost Converter for MIRAI, *SAE Technical Paper*, 2015-01-1170, 2015

[12] Tremblay, O., Dessaint, L.A. and Dekkiche, A.I., A generic battery model for the dynamic simulation of hybrid electric vehicles. *Vehicle Power and Propulsion Conference, 2007. IEEE*, pp. 284-289, 2007.

Modelling & Analysis of a Fuel Cell Hybrid Electric Vehicle using Real-World & Standard Driving Conditions

Raees B. K. Parambu, Mike Dempsey, Alessandro Picarelli

Claytex Services Ltd, Edmund House, Rugby Road, Leamington Spa, CV32 6EL, United Kingdom

E-mails: {raees.parambu, mike.dempsey, alessandro.picarelli}@claytex.com

Abstract

This paper presents an acausal model-based system-level simulation of a fuel cell plug-in hybrid electric vehicle (FCPHEV) (also known as the H2EV) in Dymola. The modelling part includes the development of a full vehicle and its subcomponents. The simulation (analysis) part involves investigation of the vehicle performance, fuel economy (Wh/km or gH_2/km) and carbon footprint (C_f) using both standard & real-world (UK) and homologation (Japan) driving conditions. The effect of the addition of auxiliary load on vehicle range is also explored based on these two countries in conjunction with corresponding drive cycles. Comparing to a commercial FCEV, the well-to-wheel (WTW) analysis results show that by adopting the proposed H2EV during Japan Olympics 2020, C_f can be reduced and fuel economy improved with an assumption that Japan produces hydrogen fuel from renewable energy resources only.

Keywords: Acausal, Model, Fuel cell, Battery, Hybrid, Vehicle, Fuel Economy, Carbon Footprint, Real-world, Standard, Drive, Cycle, Dymola.

1 Introduction

In the past five years, an increased penetration of electric vehicles (EVs) into the road transportation system has been recorded worldwide [1]. This includes the battery (BEV), plug-in hybrid (PHEV) and fuel cell (FCEV) type electric vehicles. In the year 2016, China and US were the two major contributors of EVs, followed by Japan, and the UK comes in the 6th position [1]. Although the FCEV or its variant (fuel cell plug-in hybrid electric vehicle (FCPHEV)) have a potential to decarbonise our road transport completely, this can only be achieved by a joint effort of producing hydrogen fuel solely from renewable energy resources [2] and with smart refuelling stations.

Despite various challenges, the development of fuel cell (FC) based vehicles has been accelerated and mainly noticed in the past major events (e.g., Olympics and World Expo) worldwide. In 2012, during the London Olympics, several FC powered Taxis were introduced [3]. In the past ten years, China has been continuously developing FC powered vehicles (such as cars, buses, etc.), and demonstrated their technical capabilities during their previous major events [4].

According to [5], both the national and local governments of Japan have been spending about $360 million for developing up to 6000 FCHVs and 35 stations by 2020. The main focus of this project is to achieve a target of running the Tokyo Olympics 2020 mostly on hydrogen-powered vehicles. This has generated a significant industrial opportunity and challenges for many companies (like Honda and Toyota). For the successful implementation of this kind of project within the time-frame, it is worth looking at existing FC based vehicles across the world and considering model-based systems engineering feasibility studies using these vehicles under different operating conditions. Such methods can save significant amount of time and engineering resources.

Microcab Ltd is a UK based spin-off company of Coventry University and has a connection with the University of Birmingham and several hydrogen-based research institutes both in the UK and globally [6]. They have been developing lightweight FCPHEVs for several years, and some of their successful projects are given in [7,8]. The latest version of their vehicle is known as H2EV (a plug-in series hybrid FC-Battery powertrain) primarily designed for urban use. This vehicle has proved to work well in the West Midlands area of the UK and potentially in Northern/Central Europe [9]. However, the majority of their work is based on the development of physical prototypes and testing of H2EVs on both a chassis dynamometer (rolling road) and real-world driving conditions [9]. Furthermore, there is a lack of publicly available resources in relation to an acausal system-level model of H2EVs.

Dymola (Dynamic Modeling Laboratory) is a systems-engineering tool for modelling multi-engineering multi-physical complex systems based on Modelica (an equation-based object-oriented) modelling language. There are few research papers in the literature focused on modelling of FC powertrain especially using Modelica language [10-12]. However, neither of them based on the modelling of an existing H2EV nor validated against realistic drive test data especially based on a normal everyday driving scenario in the West Midlands area in the UK. Moreover, there is no feasibility study available on adoption of H2EV for an upcoming large event, the Japan Olympics 2020.

Therefore, this research paper aims to develop an acausal system-level model of a H2EV in Dymola and analyse the performance, fuel economy and carbon footprint (C_f) of this vehicle under different driving conditions. For this purpose, a full H2EV model is developed by mostly adapting component models from a commercial library. This full vehicle model is then validated against publicly available (including real-world, also called Coventry Drive Cycle (CDC) experimental test data based in the UK. For predicting the performance of H2EV in Japanese road conditions, the developed H2EV model is then simulated with the standard Japanese JC08 drive cycle. Following on, two case studies are conducted to explore the impact of parasitic loads on the performance of the H2EV in the above chosen two countries. Finally, a well-to-wheel (WTW) analyses are performed for comparing the fuel economy and carbon footprint (C_f) of H2EV relative to a commercial FCHEV in Japan.

2 Dymola Model Overview

A system-level H2EV model is created in Dymola with the help of a commercial Modelica library, VeSyMA (Vehicle Systems Modelling and Analysis) from Claytex Services Limited [13].

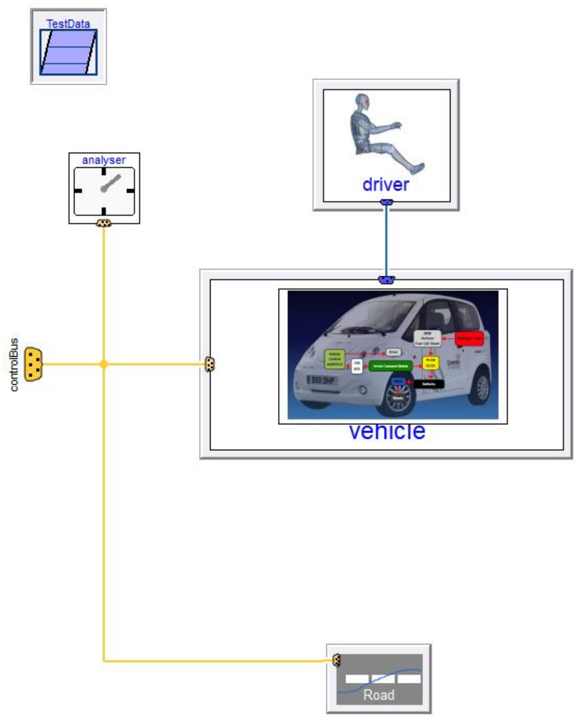


Figure 1. The full system-level model of a H2EV [Courtesy: Microcab H2EV image is used].

The VeSyMA library provides a top-level adaptable vehicle template layout in Figure 1. This paper has modified and used this template as the framework to integrate all the systems. In Figure 1, the top-level model consists of a driver, vehicle, road, analyser (to collect or calculate the most important vehicle variables such as power consumption, efficiency, emission, (C_f) etc.) and a Modelica record (TestData) with parameters of the selected H2EV. Many component models are taken directly from VeSyMA, and others such as fuel cell, hydrogen tank, auxiliary load unit (ALU) and analyser are additionally created. For enhancing the simulation speed, the component models in VeSyMA (like battery, motor and controllers) are modified.

2.1 Vehicle Propulsion System Architecture

The architecture of the vehicle model in Figure 1 is shown in Figure 2. This vehicle model is divided into the following subsystems: (1) fuel cell, (2) tank, (3) battery, (4) electric drive ((ED) motor & power converter), (5) ALU, (6) driver environment, (7) chassis and (8) vehicle parameters' record. The controller of each subsystem is built within its corresponding model.

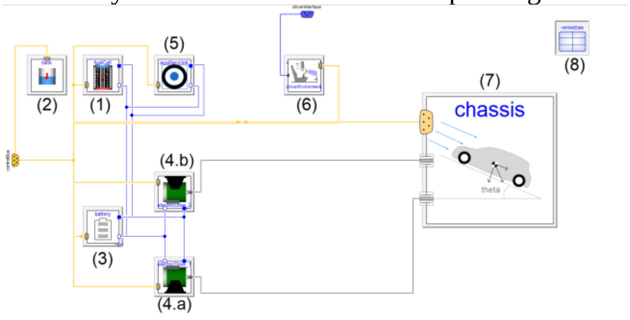


Figure 2. The architecture of H2EV with its subsystems models.

The structure of the chassis model in Figure 2 shown in Figure 3, which is further partitioned into driveline (contains gearboxes), resistance (contains aerodynamic, rolling and inclination resistances, wheels and mass) and brake models.

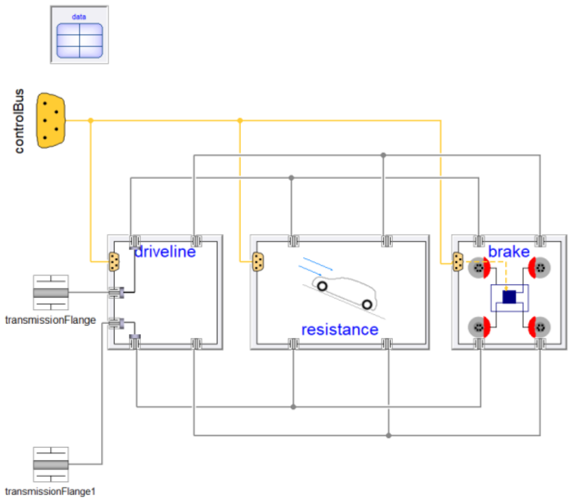


Figure 3. The structure of the chassis model.

The overall operation of the system-level model of H2EV is summarised as follows: when driver presses the accelerator pedal, a positive torque is requested to vehicle and the powertrain delivers power. On the other hand, when the driver is pressing on the brake pedal, a negative torque is requested. As in Figure 1, the driver demands are fed to the vehicle model via a bus system.

2.2 Vehicle Specification and Model Parameters

Vehicle parameter values used in this paper are those for an existing H2EV but are taken from various resources [9, 13-15] and only public domain values or educated estimates are used. Moreover, H2EV is a multi-purpose vehicle, and can be used in four different situations. These details are also included in the bottom of Table 1.

Table 1. System-level specifications and model parameters used in this paper [9, 13-15].

Description	**Parameter (value)**
Vehicle general description	Four seats lightweight niche vehicle
Kerb (gross vehicle) weight	750 kg
Top speed	under 90 km/h (55mph)
Propulsion (powertrain)	a plug-in series hybrid FC-Battery (FCPHEV)
Fuel cell	3kW (continuous) air-cooled Horizon open cathode H-3000
Hydrogen tank	Pressure= 350 bar Capacity= 74 litre (holding capacity of 1.8 kg's of hydrogen)
Battery	72V 4.3kWh $LiFePO_4$
DC motors	12.5 kW front wheel drive (Lynch LEM-200/d127)
Power converter type	DC/DC converter
Total no. of motors	2
Combined peak power of motors	40 kW
Fuel cell weight and its associated systems	Appro. 60 kg
Range (fully charged battery & full tank)	up to 290 km (180 miles)
Refuelling time	3 min
Dimensions	Length=3.5 m, Height= 1.7m, Width=1.6m
Chassis	Lotus bounded aluminium
Transmission type	Belt drive to front wheels
Multi-purpose format	
Car format	4 seats
Van format	2 seats (200 kg payloads)
Taxi format	Diver and 2/3 passengers
Taxi details	Flat floor for full accessibility

2.3 Component Models

2.3.1 Driver

For simulating a vehicle in Dymola, it is important to provide an input excitation (such as a drive cycle) using a driver model shown in Figure 1. A real driver typically commands propulsion through accelerator and brake pedals. The chosen model is a longitudinal driver model from VeSyMA. It reads the speed-time demands from the provided drive cycle and has a longitudinal closed loop feedback control.

2.3.2 Fuel Cell Stack

In this paper, the system under observation is the commercially available 3kW, 72 cells, Proton Exchange Membrane fuel cell (PEM FC) stack from Horizon Fuel Cells Technologies [15].

The main advantage of this type of FC is that it produces water as a residue, with high efficiency compared to Internal combustion engines (ICEs). It can operate at low temperatures and allows a fast start-up. Because of these reasons, as of today, it is one of the best options for automotive traction applications. A solid-polymer based electrolyte is very common in this type of FCs.

The overall operation of a PEM FC can be expressed as;

Overall reaction:

$$H_2(g) + \frac{1}{2}O_2(g) \rightarrow \mathrm{H2O}(l) + \text{electric energy} + \text{wasteheat}$$

At anode:

$$H_2(g) \rightarrow 2\,H^+(aq) + 2e^-$$

At cathode:

$$\frac{1}{2}O_2(g) + 2\,H^+(aq) + 2e^- \rightarrow \mathrm{H2O}(l)$$

In this paper, a dynamic FC cell model based on [16,17] is implemented as shown in Figure 4. It includes five major parts; (1) V_{rev}, the reversible (Nernst) voltage, (2) V_{act}, an activation polarisation (to represent voltage drop associated with the activation of anode and cathode), (3) V_{conc}, the concentration polarisation (voltage drop resulting from the decrease in the concentration of oxygen and hydrogen), (4) an ohmic loss (to represent resistance of the proton transfer through the polymer membrane). Both activation and concentration polarisations are modelled within the polarisation model in Figure 4, and (5) a double layer capacitance is connected parallel to the polarisation model.

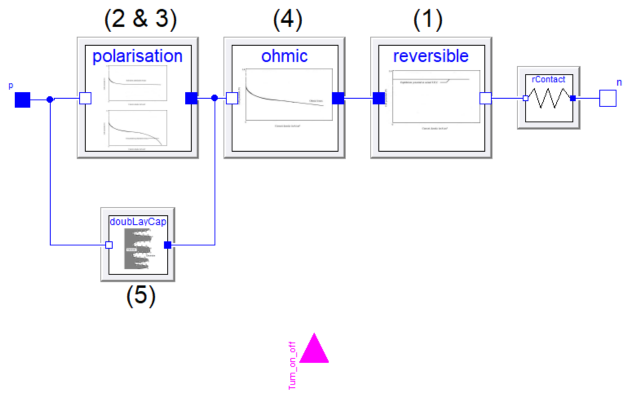


Figure 4. Structure of the FC cell model.

The FC cell voltage, $V_{FC_{cell}}$ and its each term can be defined as below [16];

$$V_{FC_{cell}} = V_{rev} + V_{act} + V_{ohm} + V_{conc} \qquad (1)$$

$$V_{rev} = -E^0 - \frac{RT}{2F}\ln\left(\frac{P_{H_2O}}{P_{H_2}\cdot P_{O_2}^{1/2}}\right) \quad (2)$$

$$V_{act} = -\frac{RT}{2\alpha F}\ln\left(\frac{i_{FC_{cell}} + i_{loss}}{i_0}\right) \quad (3)$$

$$V_{ohm} = -(i_{FC_{cell}} \cdot ASR) \quad (4)$$

$$V_{conc} = -m \cdot i_{FC_{cell}}{}^{k} \ln\left(1 - \frac{i_{FC_{cell}}}{i_L}\right) \quad (5)$$

where, $E^0 = \frac{G_{f,liq}}{2F}$ is the electromotive force (EMF) of the cell at standard pressure, $G_{f,liq}$ is the Gibbs function in liquid form, R and F are the molar and Faraday's constants, T is the cell internal temperature, P_{H_2}, P_{O_2}, and P_{H_2O} are the partial pressure of hydrogen, oxygen and water respectively, α is the charge transfer coefficient, $i_{FC_{cell}}$ is the cell current density, i_{loss} and i_0 are internal current and exchange current densities, ASR is the area specific resistance, m and k are the amplification and mass transport constants, and i_L is the limiting current.

In this paper, the individual FC cell model described above is scaled using a specific number of cell in series (N_{cell}) to represents a full FC stack model as shown in Figure 5.

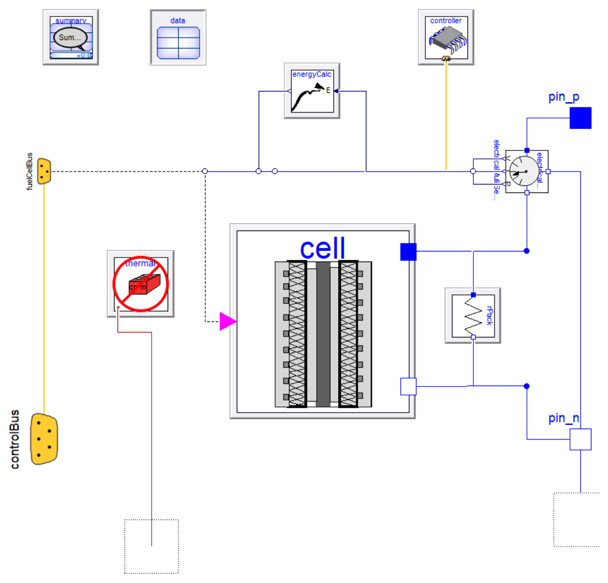


Figure 5. Structure of the FC stack model.

The FC stack current (I_{FC}), power (P_{FC}) and efficiency (η_{FC}) are defined as below;

$$I_{FC} = i_{FC_{cell}} A_{cell} \quad (6)$$

$$P_{FC} = N_{cell} \times V_{FC_{cell}} \times I_{FC} \quad (7)$$

$$\eta_{FC} = \frac{V_{FC_{cell}}}{1.482} \quad (8)$$

where A_{cell}, the cell active area. FC cell model parameters are taken mainly from [16] and rest of them are estimated to match with Table 1 system level values.

2.3.3 Hydrogen Tank

Hydrogen tank model is a simple mathematical model which calculates the hydrogen consumption and by which to compute the amount of hydrogen in the tank ($tank_{level}$), but, does not include the movement of the hydrogen laterally. The hydrogen consumption can be calculated from the hydrogen consumption rate (f_{H_2}) and mass flow rate ($\dot{m}_{H_2}$).

The f_{H_2} can be estimated based on applied current I_{FC}, the number of cells (N_{cell}), and Faraday's constant.

$$f_{H_2} = \frac{I_{FC} \cdot N_{cell}}{2F} \quad (9)$$

The $f_{H_2}$is then related to mass flow rate as:

$$\dot{m}_{H_2} = M \cdot f_{H_2} \quad (10)$$

where, M, the molecular weight of hydrogen is 2 g/mol. Then, by using the initial hydrogen content in the tank M_{tank_init} used to determine the tank level or remaining hydrogen in the tank as below.

$$tank_{level} = \frac{M_{tank\_init}}{\dot{m}_{H2}} \quad (11)$$

Note that a normalised $tank_{level}$ value (0-1) is used to indicate level of hydrogen in the tank.

2.3.4 Battery

Because of a simple mathematical formulation and ease of implementations, a circuit based electrical model of a Li-ion battery is created. The electrical behaviour of the battery cell is modelled using an equivalent circuit-based model [18] as shown in Figure 6.

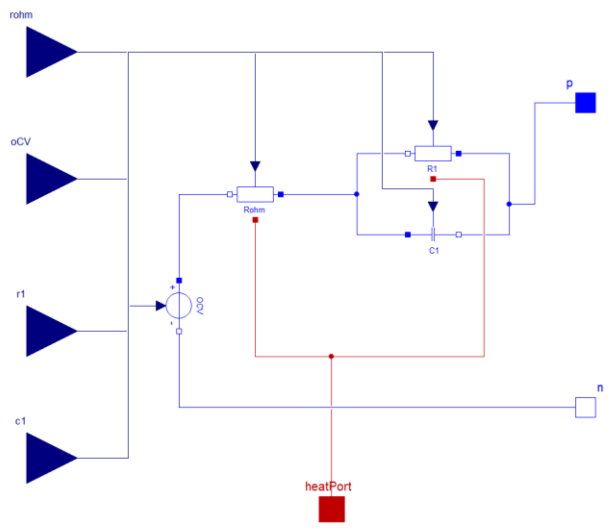


Figure 6. Structure of the electrical circuit model of a Li-ion battery cell.

In Figure 6, the electrical model comprises OCV to represent the open circuit potential (V_{ocv}), a parallel resistor-capacitor (R_1C_1) network to model the dynamics part and a series resistor (R_0) to represent instantaneous voltage drop of the battery cell. A set of differential and algebraic equations describe the overall electrical behaviour of the cell is provided below.

$$V_{Batt_{cell}} = V_{ocv} - I_{Batt_{cell}} R_0 - U_d \tag{12}$$

$$\frac{d}{dt} U_d + \frac{U_d}{R_d C_d} = \frac{I_{Batt_{cell}}}{C_d} \tag{13}$$

$$SOC = SOC(t_0) - \frac{100}{Q_{rated}} \int_{t_0}^{t_f} (I) \cdot dt \tag{14}$$

The equation (14) shows the state-of-charge (SOC) of the battery cell calculated using the ratio of applied current integrated over time to the rated capacity of the battery, Q_{rated}.
Similar to the FC model presented in Section 2.3.2, the individual battery cell model described above is scaled using a specific number of cell in series (N_s) and in parallel (N_p) to represent battery pack characteristics shown in Figure 7.

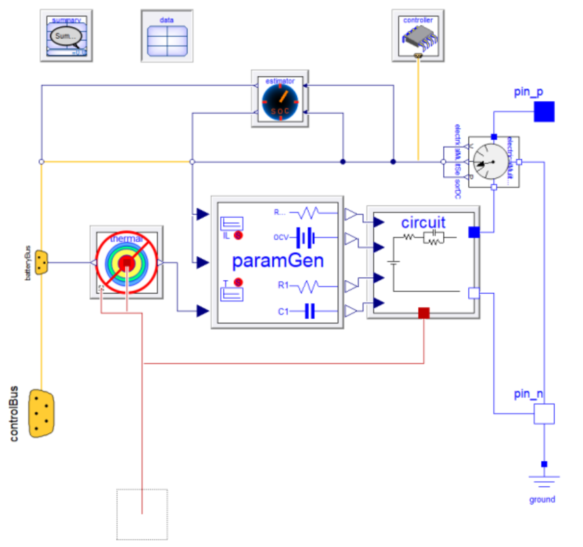


Figure 7. Structure of thermally coupled electrical model of Li-ion battery pack**.**

Battery energy (E_{batt}) consumption is calculated as

$$E_{batt} = \int_{t_{init}}^{t_{op}} P_{batt} \ dt \tag{15}$$

whereas, t is the time of operation.
An optional 1-D thermal model integrated with electrical model is also available, but not used in this introductory paper. The parameter used in this paper are given in Table 2.

Table 2. Battery model parameter.

General	
Manufacture	LiFeBATT
Model	X-2E 40166
Chemistry	LiFePO4
Total number of cells	88
Battery cell	
Capacity	15 Ah
Nominal cell voltage	3.3V
Inner resistance	3 mΩ
Weight	470 g
Battery pack	
Pack configuration	4 N_p, 22 N_s
Main drive voltage	72V
Energy	4.3kWh
Capacity	59.7 Ah (~=60Ah)
Inner resistance	18 m Ω
Weight	45.12 kg

2.3.5 Auxiliary Load Unit

The ALU is simple model to simulate electrical accessories in the H2EV, such as A/C (heater & de-mister), electronics etc. For the sake of simplicity, within this model, the electrical power consumption is assumed to be constant provides a constant power demand if active and no power demand if deactivated.

In this paper, this ALU is used carry out some investigation on how the additional load impacts the vehicle range under different operating conditions.

2.3.6 Electric Drivetrain

The electric drive (ED) model consists of motor and power electronics. The implemented ED is a map-based model which absorbs or delivers power based on the input torque signal received. Acceleration demands will send a positive torque request whereas braking demands will send a negative torque request.

2.3.7 Driveline (Transmission)

H2EV uses a belt drive transmission to power the front wheels; however, in this paper, given we are not interested in more detailed driveability effects, a fixed ratio reduction gear is implemented.

2.3.8 Brakes (includes Regenerative) & Wheels

The braking subsystem design is used in this paper uses a force input from the brakes control bus dictating the interaction of the braking regeneration systems and the physical braking system through the addition of the HybridDemand block.
For the sake of simplicity, an ideal rolling wheel is used.

2.3.9 Body with Mass & Aerodynamics

An essential part of vehicle modelling is mass and aerodynamics of the vehicle. The vehicle is subject to the propulsive force due to powertrain torque and the aerodynamic and rolling resistance force to movement of the vehicle.

This model defines basic aerodynamic forces applied at the centre of pressure; using drag coefficients. The forces are calculated in the chassis frame, so the forces are parallel to the axes of the frame and are affected by the attitude of the vehicle.

A mathematical description of this part of model is provided in the following Section 3.1.

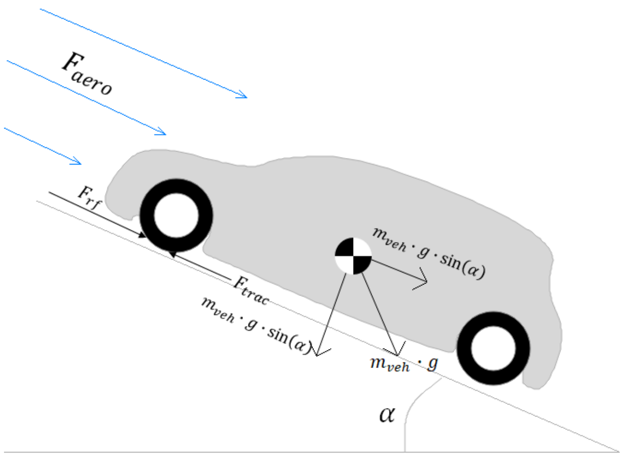
2.3.10 Control Bus

In Dymola, a control bus creates a communication link between the physical and control parts of the model.

3 Mathematical Description

This section of the paper is intended to provide the major equations used to build the entire vehicle model in Dymola. We are interested only the longitudinal vehicle dynamics. As we shown in the rest of this paper, one can combine the following equations together with the previously provided equations to do various system-level analysis of a vehicle virtually.

3.1 Chassis model



F_{aero}
F_{rf}
F_{trac}
$m_{veh} \cdot g \cdot \sin(\alpha)$
$m_{veh} \cdot g \cdot \sin(\alpha)$
$m_{veh} \cdot g$
α

Figure 8. Forces acting on a vehicle**.**

The longitudinal vehicle dynamics of vehicle can be written as [19,20]

$$F_{trac}(t) = F_{aero}(t) + F_{roll}(t) + F_{grad}(t) + F_{net}(t) \tag{16}$$

where, F_{trac} and F_{roll} are traction force and rolling friction of the front tyres (due to a front-wheel drive topology of H2EVs), F_{aero} is the aerodynamic friction and F_{grad} is the grade climbing resistance.

The aerodynamic drag force can be calculated as

$$F_{aero} = \frac{c_d \cdot A_f \cdot \rho_{air} \cdot {v_{veh}}^2}{2} \tag{17}$$

where, $c_d = 0.482$ is the drag coefficient, $A_f = 2.72$ m2 is the frontal area, ρ_{air} is air density at current location (about 1.2 kg/m3 at sea level and normal temperature) and v_{veh} speed of the vehicle.

The rolling resistance can be defined as

$$F_{roll} = c_r \cdot m_{veh} \cdot g \cdot \cos(\alpha) \tag{18}$$

whereas, $c_r = 0.017$ is the rolling resistance coefficient, m_{veh} mass of the vehicle, $g = 9.81$ m/s2, the acceleration due to gravity and α is the inclination angle.

The force required to overcome grade climbing resistance is given as

$$F_{grad} = m_{veh} \cdot g \cdot \sin(\alpha) \tag{19}$$

The net force is calculated as

$$F_{net} = m_{veh} \cdot a_{veh} \tag{20}$$

where, a_{veh} as the acceleration of the vehicle.
Finally, tractive effort delivered by powertrain is given by

$$F_{trac} = \frac{G}{r} T_{mot} \tag{21}$$

where, G, gear ratio of the system connecting the motor to the axle, r, tyre radius and T_{mot} is the torque of the motor.
In order to overcome all the resistances outlined above, a sufficient power must be generated by main and/or secondary power units. The aggregation of these powers can be collectively called as electrical power of the powertrain.
The electric power to the motor represented as

$$P_{el} = P_{batt} + P_{aux} - P_{FC} + P_{reg} \tag{22}$$

where, P_{batt} the battery power, P_{aux} represents the parasitic (auxiliary) load of the vehicle, P_{FC} is the fuel cell power, and P_{reg} is the power due to regeneration.

In eq. (22), if motor power is positive, then, the battery power is calculated as:

$$P_{batt} = \frac{P_{aux} + P_{mot}}{\eta_t} \tag{23}$$

where P_{aux} is the constant power drain from other vehicle systems, such as air conditioners, other sub-systems etc.

On the other hand, in eq. (22), if the motor power is negative, then, the regenerative power is used as:

$$P_{reg} = P_{aux} + P_{mot} \times \eta_t \tag{24}$$

4 Energy Efficiency & Range Estimation

4.1.1 Energy Efficiency

The efficiency of drivetrain is calculated as [9]

$$\eta_t = \eta_{ele} \cdot \eta_{trans} = \frac{P_{wheels}}{[P_{ele}]} \tag{25}$$

whereas, η_{ele}, the motor and its controller's electrical efficiency, η_{trans} represents the efficiency of the transmission, P_{wheels}, mechanical power at wheels.

4.1.2 Range Estimation

As stated in the introduction section, the H2EV is a series hybrid vehicle most suited for urban trips. Two modes of operation are possible: (1) EV and (2) FCEV modes. Under the normal FCEV operating condition, the FC stack (3kW Horizon stack) allows a constant load to the battery pack up to the battery voltage rises above a specific pre-set value of 87V [9]. By this time the FC stack will power down to 300W (as an ideal mode). The FC stack kicks in again if a sufficient power demand is required otherwise it is automatically turned off after half an hour. Therefore, the following equations are used for estimating the range of the vehicle using both EV and FCEV modes.

4.1.2.1 EV Mode

The driving range of pure battery electric mode, R_{EV} can be estimated from the simulation of the vehicle model. This can be expressed are as following;

$$R_{EV} = v_x \cdot t_{EV\_op} \quad (26)$$

4.1.2.2 FCEV Mode

The range estimation of FCEV mode is given as,

$$R_{FCEV} = v_x \cdot t_{FECV\_op} \quad (27)$$

where, t_{FECV_op} is calculated based on hydrogen consumption rate.
However, the total range of the vehicle can be estimated by combining above two ranges.

5 Results and Discussion

As aforementioned, in this paper, we use three types of drive cycles (1) UDC, (2) CDC and (3) JC03 to initially validate the model and then to predict the H2EV performance under different operating conditions. Although this paper is intended to present both EV and FCEV modes operation, but, because of the page limit, the FCEV part is removed from the rest of the discussion section. However, this part will be provided in the future paper.

5.1 Model Validation

The first step in the analysis part is to validate the developed model using both standard and real-world drive cycles.

5.1.1 Standard Driving conditions (Rolling Road Test)

This section of the paper focuses on validation of the developed H2EV model using standard (UDC or ECE-15) drive cycles.

In Figure 9, the estimated range on EV mode is 7.951 km, whereas on the real test is 7.952 km [9]. battery SOC is depleted much quicker than CDC drive cycle. Both, the target and actual velocities matches well, therefore, its hard to distinguish the difference.

ECE-15 on EV mode

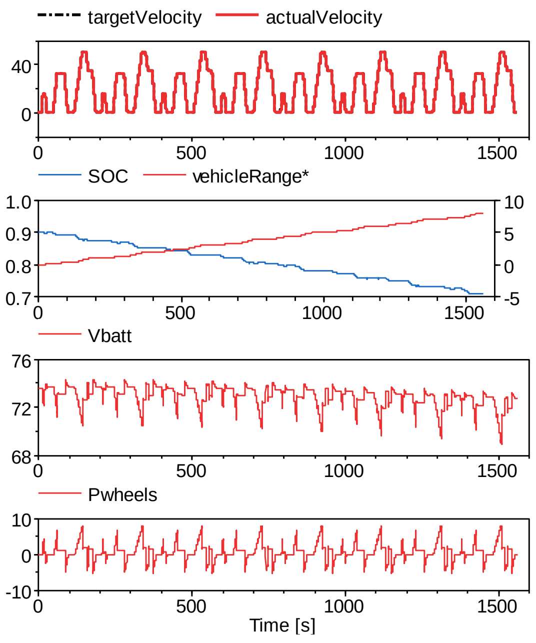


Figure 9. H2EV simulation result using 8 UDC drive cycles.

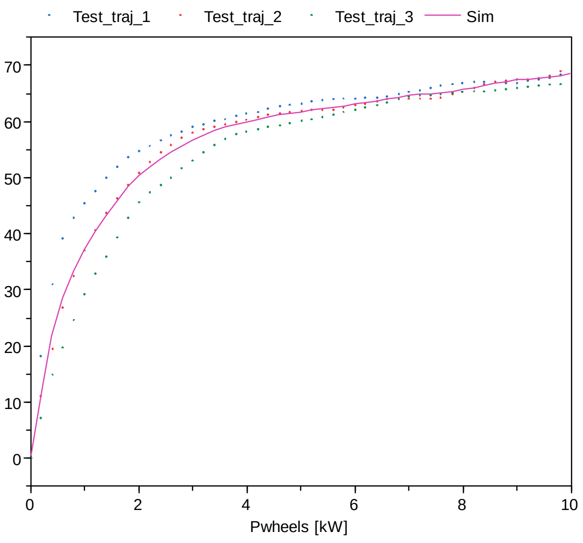


Figure 10. H2EV powertrain efficiency versus power at wheels.

Comparing to the result provided in [9], the predicted value from the model is obtained within the range plotted in Figure 10.
Figure

5.1.2 CDC on EV mode (with and without FC weight)

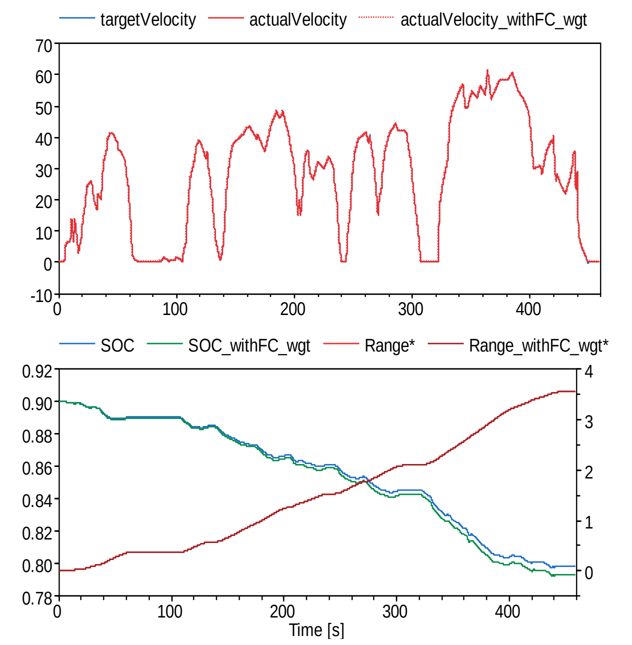


Figure 11. Typical CDC based on [9] with 3.58 km lap.

Figure 11 shows simulation result obtained from the H2EV model present in this paper using Dymola. The target velocity (test data) matches well with actual velocity (simulation result). In Figure 11 also depicts the EV range of the vehicle which is 3.54 km from simulation shows good agreement with the 3.58 km provided in [9].

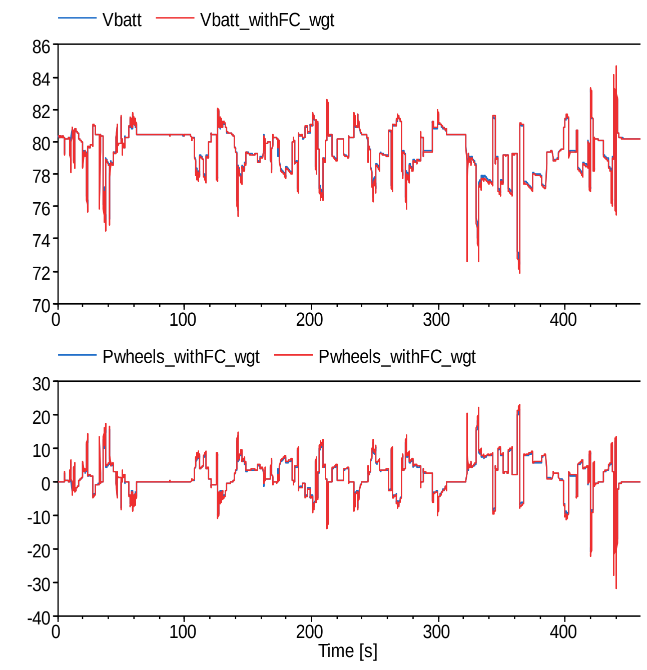


Figure 12. Battery voltage and power at wheel using CDC.

5.2 Model Prediction

In this section we simulate the model using JC08 drive cycle to investigate vehicle powertrain performance and energy efficiency in Japanese road driving conditions.

5.2.1 Standard (Japanese) Drive Cycle

JC08 on EV mode

In Figure 13, the both the simulation result and test data have good agreements using JC08 drive cycle. However, battery SOC is depleted much quicker than CDC drive cycle.

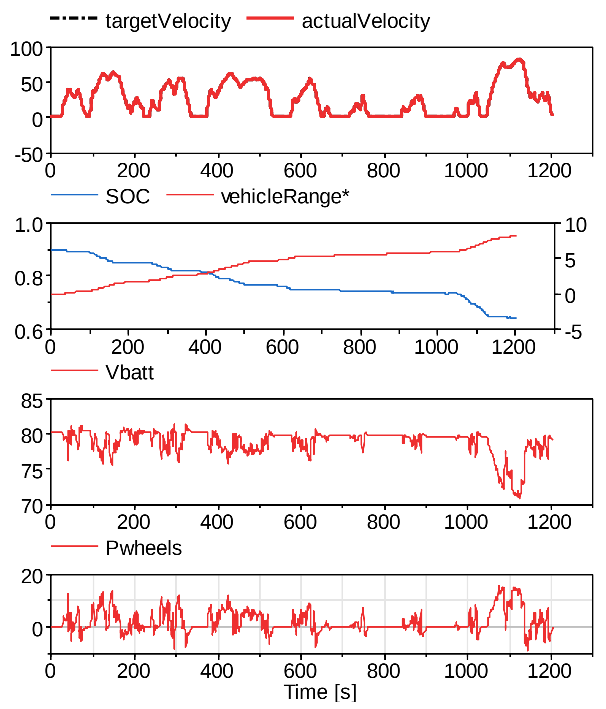


Figure 13. Standard Japanese JC08 drive cycle

Table 3. Summary of the fuel economy estimation of Microcab H2EV and Commercial [9,22, 23].

Vehicle	*H2EV*					*Commercial*
Drive cycle	UDC (EV mode)	Repeated UDC (FCEV mode)	CDC (EV mode)	JC08 (EV mode)	Repeated JC08 (FCEV mode)	Repeated JC08 (FCEV mode)
Battery Power (Wh/km)	102	128.3 (FR=33.5km)	126			-
FC Power (Wh/km)	-	274.9	-			
g H2/km	-	6.82 (FR=265 km)	-			6.57 (FR=700 km)

5.2.2 Effects of Auxiliary Loads (UK and Japan) [9,21].

For determining the effect of climatic load on driving range of the vehicle, a repeated drive cycle test using both CDC and JC08 is conducted. Then, auxiliary loads based on Table 3 is used to compare the result obtained with and without additional loads using a ALU model is described in Section 2.3.5.

Table 4. WTW calculation parameters based on [9,21].

Load [W]	*UK*	*Japan*
Electronics	400 (max)	400 (max)
Fuel cell operation	300	300
Cabin Heater & demister or A/C	2400	2400

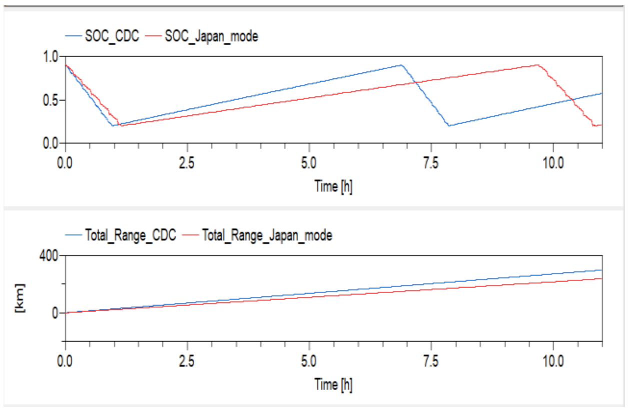


Figure 14. Repeated drive cycle simulation using CDC and JC08 drive cycles.

It is clear from Figure 14 that using FCEV mode, CDC got total range of about 300 km, whereas the Japanese JC08 is 282 km. This is mainly because of require more energy from the battery pack to complete its drive cycle.

5.3 WTW Analysis

5.3.1 Well-to-Wheel (WTW)

Carbon footprint (C_f) of the H2EV for both EV and FCEV modes can be calculated based on following equations [9];

5.3.1.1 EV Mode

$$\mathrm{Cf} = \frac{P_{battEV} \cdot C_{grid}}{\eta_{grid} \cdot \eta_{charger}} \tag{28}$$

whereas, P_{battEV} is the energy consumed by battery in EV mode alone, C_{grid}, grid carbon intensity of the UK or Japan, η_{grid}, efficiencies for the grid and $\eta_{charger}$, efficiencies for the plug-in charger.

5.3.1.2 FCEV Mode

$$\mathrm{Cf} = \frac{f_{H2} \cdot C_{H2}}{\eta_{comp} \cdot \eta_{trans}} \tag{29}$$

whereas, f_{H2} is the hydrogen consumption rate in FCEV mode, C_{H2}, carbon intensity of the hydrogen production in the UK or Japan, η_{comp}, efficiency related to compression pressure of the hydrogen gas and η_{trans}, efficiency related to transportation of hydrogen.

Table 5. WTW calculation parameters based on [9,21,22].

	UK (EU)	Japan
Grid efficiency [%]	86	96
Grid carbon intensity [gCO2/kWh]	507	554
Charger efficiency [%]	90	90
Hydrogen production carbon intensity [gCO2/kg H2]	[NG: 11888] [Electrolysis: 970]	
Efficiency related to compression pressure of H2 [%]	[700 bar: 78] [350 bar: 87]	

Efficiency related to transportation of H2 [%]	88	

Table 5 shows the parameter used in the calculation of WTW in both UK and Japan.

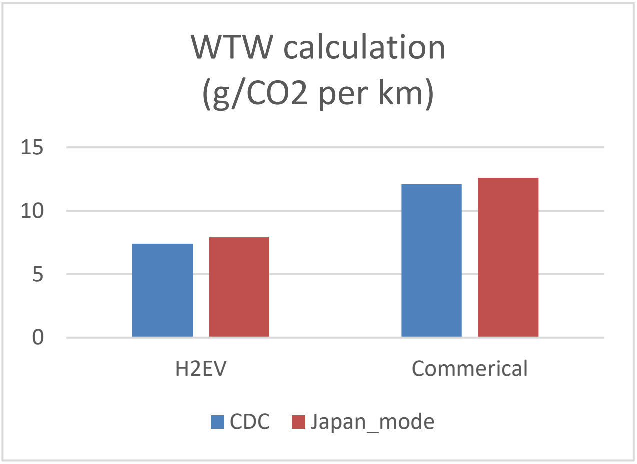


Figure 15. Carbon footprint (C_f) based in the UK and Japanese using H2EV and a commercial vehicle.

Figure 15 shows that by adapting H2EV or similar vehicle during Japan Olympics 2020 can reduce carbon footprint (C_f) and increase the fuel economy. A detailed analysis based on this calculation will be included in the final paper.

6 Conclusion

In this research paper an acausal system-level model of a fuel cell plug-in series hybrid electric vehicle (H2EV) is presented. With the help of this system-level model, performance, energy efficiency and carbon foot print of H2EV are analysed based on different realistic and standard drive cycle tests. Climatic condition of both the UK and Japan has considered while investigating the vehicle range due to additional climatic load. By comparing the results obtained from WTW analysis of H2EV and a commercial FCHEV show a promising result.

References

[1] Cazzola, P., Gorner, M., Munuera, L., Schuitmaker, R. and Maroney, E., 2017. Global EV Outlook 2017. International Energy Agency, France.

[2] Steenberghen, T., & Lopez, E. (2008). Overcoming barriers to the implementation of alternative fuels for road transport in Europe. Journal of Cleaner Production, 16(5), 577-590.

[3] Hayter D., LONDON: a capital for hydrogen and fuel cell technologies, April 2016, HYDROGEN LONDON.

[4] Ju, W. Overview of Fuel Cell Vehicle Development in China, October 16, 2017, Society of Automotive Engineers of China International Hydrogen Fuel Cell Association (IHFCA).

[5] Industry Opportunities: Japan 2020 Olympics, July 2015, Marine International Trade Center, Global Resources. Local Expertise.

[6] Microcab Hydrogen Fuel Cells, http://www.coventry.ac.uk/research/research-campus/microcab-hydrogen-fuel-cells/. (Last accessed: 05 Feb. 18)

[7] Staffell, I. (2011). Results from the Microcab fuel cell vehicle demonstration at the University of Birmingham. International Journal of Electric and Hybrid Vehicles, 3(1), 62-82.

[8] Barrett, S. (2011). Microcab launches new hydrogen fuel cell microcar for UK demo. Fuel Cells Bulletin.

[9] Ryan, D., Shang, J., Quillivic, C., & Porter, B. (2014). Performance and energy efficiency testing of a lightweight FCEV Hybrid Vehicle. Eur Electr Veh Congr (EEVC).

[10] Steinmann, W. D., & Treffinger, P. (2000). Simulation of fuel cell powered drive trains. In Modelica WorkShop.

[11] Treffinger, P., & Goedecke, M. (2002). Development of fuel cell powered drive trains with Modelica. In Proceedings of the 2nd Modelica Conference (pp. 125-131).

[12] Hellgren, J. (2002, March). Modelling of hybrid electric vehicles in Modelica for virtual prototyping. In Proceedings of the 2nd International Modelica Conference (pp. 247-256).

[13] VeSyMA Library, http://www.claytex.com/products/dymola/model-libraries/vesyma/. (Last accessed: 05 Feb. 18).

[14] Microcab H2EV Vehicle Specification, http://www.microcab.co.uk/the-new-h2ev/ . (last accessed 31 Jan 2018).

[15] Horizon Fuel Cell Technologies, 3000W Fuel Cell Stack User Manual, Updated 18 Aug. 2013.

[16] Spiegel, C. (2011). PEM fuel cell modeling and simulation using MATLAB. Academic press.

[17] Larminie, J., Dicks, A., & McDonald, M. S. (2003). Fuel cell systems explained (Vol. 2). Chichester, UK: J. Wiley.

[18] Uddin, K., Picarelli, A., Lyness, C., Taylor, N. and Marco, J., 2014. An acausal Li-ion battery pack model for automotive applications.Energies, 7(9), pp.5675-5700.

[19] Guzzela, L., & Sciarretta, A. (2007). Vehicle propulsion systems.

[20] Larminie, J., & Lowry, J. (2012). *Electric vehicle technology explained*. John Wiley & Sons.

[21] Yuu, N., & Tsuguhiko, N. (2014). A novel concept of AI-EV (air-conditioner integrated electric vehicle) for the advanced smart community. International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics.

[22] Measures to Suppress CO2, http://www.fepc.or.jp/english/environment/global_warming/suppress_co2/ (Last accessed: 05 Feb. 18).

[23] Tokieda, J., Ozawa, T., Yoshida, T., Aida, S, and Oya, L. (2015). The MIRAI Life Cycle Assessment Report for communication. Toyota Motor Corporation.

Hyundai Framework for Vehicle Dynamics Engineering based on Modelica and FMI

Kwang Chan Ko[1] Erik Durling[2] Jong Chan Park[1] Wonyul Kang[3] Johan Andreasson[2]

[1]Commercial Vehicle CAE research lab., Hyundai Motor Group, Korea, {kcko,impactpark}@hyundai.com
[2]Modelon, Sweden, {erik.durling,johan.andreasson}@modelon.com
[3]Institute of Vehicle Engineering, Korea, wykang@ivh.co.kr

Abstract

This paper describes a framework for systems engineering with primary application in the field of vehicle dynamics, addressing the need to be able to broadly deploy models to accelerate innovation and design. The framework is based on open standards Modelica, FMI and SSP.

Keywords: Vehicle Dynamics, Modelica, FMI, SSP

1 Introduction

This paper describes a framework for vehicle dynamics engineering with focus on evaluation of ride and handling. Especially the following key requirements are addressed:

1. Scalability: The framework should be able to handle both subsystems such as suspensions, and complete vehicle configurations
2. Multi-fidelity: The framework should allow for models of different fidelity to be combined and executed together
3. Deployment: The framework should support broad usage among engineers, including non-simulation experts
4. Future-proof: The framework should be designed with future scope extensions in mind; control design, system integration and applications to other vehicle types such as passenger cars.

Similar requirements are found in several other applications, e.g. (Andreasson, 2016), (Henningsson, 2014), (Sundström, 2016a), (Sundström, 2016b).

2 Framework approach

The approach was to define a framework to allow for separation of different types of engineering work, and engineers skills according to but not limited to the following categories:

1. Model execution: Most engineers are using the models to support engineering decisions. Their need is to configure the vehicle/subsystem, chose/edit data sets, execute analysis and post-process the results. All with a high degree of automatization. Their background in simulation is typically limited and there is limited need to see details in the models. They prefer to work in a streamlined environment that they are familiar to. If they run into problems with simulations, they want to get help rather than solving it themselves.
2. Model authoring: A limited number of engineers need to make new models and change existing ones, they also execute very specialized analysis. They have stronger background in simulation, and typically have experience in using simulation tools. They typically want to have great flexibility and have less need for streamlined workflow. These people are expected to help engineers working with Model execution.
3. System integration: Engineers working with system integration may need to edit how models from different suppliers are connected, but rarely need to see inside each model for details. They need some more flexibility than those working with Model execution, but less compared to those working with Model authoring.

The implementation of the framework is carried out in a sequence of steps with gradually expanded capabilities:

1. Execution of fixed configurations, Figure 1.
2. Execution of reconfigurable models, Figure 2.
3. System integration, Figure 3.

2.1 Execution of fixed configurations

Figure 1 illustrates a common framework to manage the separation of engineering task assuming that model execution does not involve system reconfiguration. System reconfiguration typically means replacing a suspension, change the number of suspensions, or any other change that change the topology of the model.

In this case, the Model author must prepare all combinations needed by the Model executor and compile them to executable form. If a combination is missing, the Model executor has to order it from the Model author. The output from the Model author is a set of FMUs (executable models according to the FMI standard). These FMUs can then be used by both the Model executor and the System integrator as seen in Figure 1.

With the separation, the Model executor can work in any environment that supports the FMI standard (FMI, 2018), such as MS Excel (FMIE, 2018), MATLAB/Simulink (FMIT, 2018), or python, (pyFMI, 2018). Here, Excel sheets are used for configuration and data management, and MATLAB for Execution and post processing. This is consistent with the setup chosen in e.g. (Sundström, 2016a).

Using the FMI standard, the System integrator can manage models from any tool that supports the standard, see (FMI, 2018) for a list of available tools.

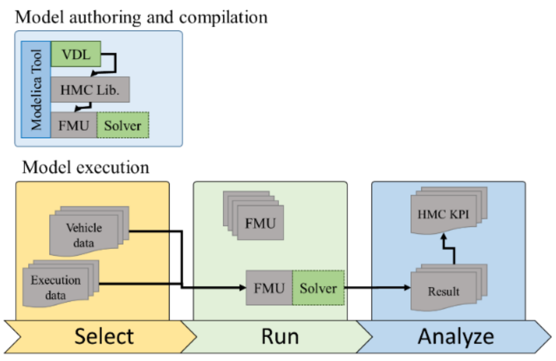


Figure 1 Execution of fixed configurations.

2.2 Execution of reconfigurable models

When there is a need for the Model executor to also reconfigure the model, add compile-on-demand functionality can be added as illustrated in Figure 2.

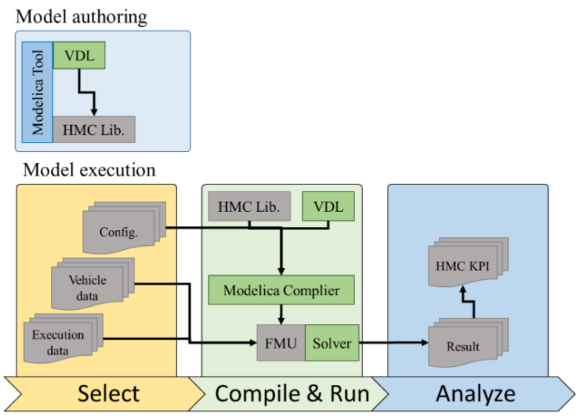
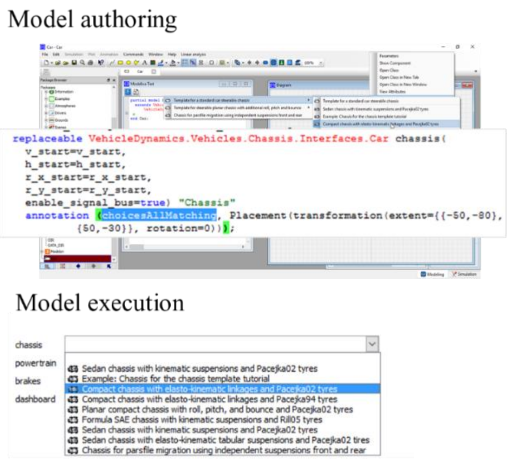


Figure 2 Execution of reconfigurable models.

Here, the output from the Model author is the Modelica library. In addition, the execution environment is equipped with a Modelica compiler. This allow for the execution environment to reconfigure Modelica model from the HMC (Hyundai Motor Group) Library according to the commands of the Model executor. Configuration options can be set up in the HMC Library so that the Model author can control what options are available for the Model executor as illustrated in Figure 3.



```
replaceable VehicleDynamics.Vehicles.Chassis.Interfaces.Car chassis(
  v_start=v_start,
  h_start=h_start,
  r_x_start=r_x_start,
  r_y_start=r_y_start,
  enable_signal_bus=true) "Chassis"
  annotation (choicesAllMatching, Placement(transformation(extent={{-50,-80},
          {50,-30}}, rotation=0)));
```

Sedan chassis with kinematic suspensions and Pacejka02 tyres
Example: Chassis for the chassis template tutorial
Compact chassis with elasto-kinematic linkages and Pacejka02 tyres
Compact chassis with elasto-kinematic linkages and Pacejka94 tyres
Planar compact chassis with roll, pitch, and bounce and Pacejka02 tyres
Formula SAE chassis with kinematic suspensions and Rill05 tyres
Sedan chassis with kinematic suspensions and Pacejka02 tyres
Sedan chassis with elasto-kinematic tabular suspensions and Pacejka02 tires
Chassis for parsfile migration using independent suspensions front and rear

Figure 3 Reconfiguration in Model authoring environment (top) and in Model execution environment (bottom).

2.3 System integration

To allow to combine the model from HMC Library with 3rd party models, the suggestion is to also do this using FMI standard. Here an intermediate step is introduced to bundle the FMUs together prior to executing the simulation. The 3rd party models are maintained in a data base.

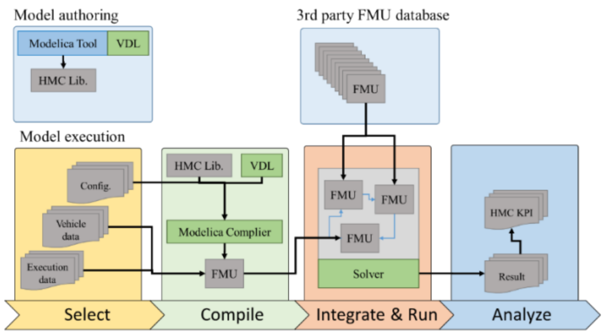


Figure 4 Toolchain with ability to include 3rd party FMUs.

3 Model creation

The HMC_Commercial_Vehicle library is seen in Figure 5. It is based on the Vehicle Dynamics Library (VDL, 2018) and makes use of the Interface – Template structure introduced in (Andreasson, 2006) now broadly used to build system libraries, see e.g. (Sielemann, 2017a), (Sielemann, 2017b).

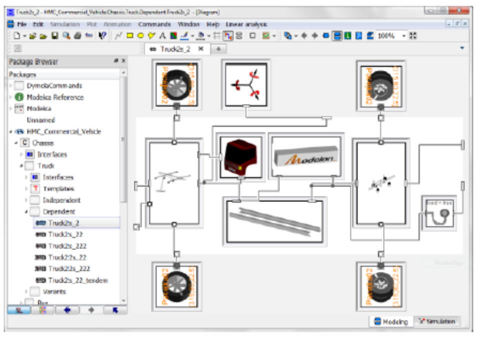


Figure 5 HMC_Commercial_Vehicle library.

In brief, the structure promotes separation of interfaces, topology and configuration as illustrated in Figure 6. Here, a partial model is defined containing all common interface attributes such as connectors and parameters (left) that in turn is inherited by all implementations. An implementation is then separated into the topology definition (middle) and the configuration (right). The topology is called a template and is defined using subsystem interfaces as placeholders, and the connections between these. Based on the template, variants can easily be defined by just stating the contents of each placeholder and thus eliminating the need to duplicate information for layout and connections.

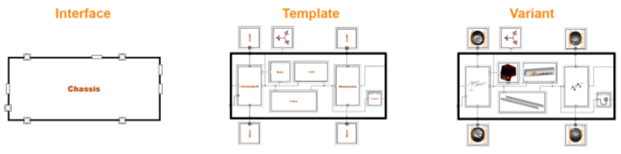


Figure 6 Interface - template structure according to (Andreasson, 2006).

3.1 Modelica library export of FMUs

To enable execution without recompilation, a structure that supports automatic generation of a large number of preconfigured variants was set up. In brief this is set up based on a strict package structure and naming convention with the corresponding representation on the file system as illustrated in Figure 7.

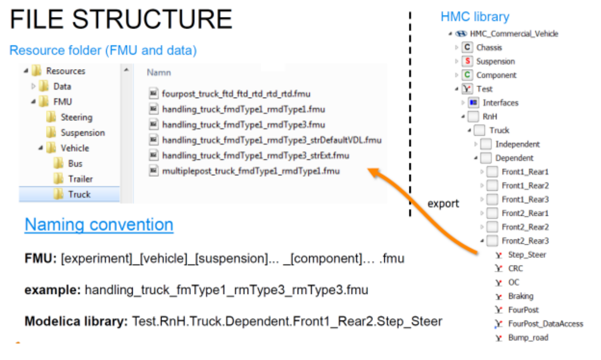

FILE STRUCTURE
Resource folder (FMU and data)
Resources
Data
FMU
Steering
Suspension
Vehicle
Bus
Trailer
Truck
Namn
fourpost_truck_ftd_ftd_rtd_rtd_rtd.fmu
handling_truck_fmdType1_rmdType1.fmu
handling_truck_fmdType1_rmdType3.fmu
handling_truck_fmdType1_rmdType3_strDefaultVDL.fmu
handling_truck_fmdType1_rmdType3_strExt.fmu
multiplepost_truck_fmdType1_rmdType1.fmu
Naming convention
FMU: [experiment]_[vehicle]_[suspension]... _[component]… .fmu
example: handling_truck_fmType1_rmType3_rmType3.fmu
Modelica library: Test.RnH.Truck.Dependent.Front1_Rear2.Step_Steer
export
HMC library
HMC_Commercial_Vehicle
Chassis
Suspension
Component
Test
Interfaces
RnH
Truck
Independent
Dependent
Front1_Rear1
Front1_Rear2
Front1_Rear3
Front2_Rear1
Front2_Rear2
Front2_Rear3
Step_Steer
CRC
OC
Braking
FourPost
FourPost_DataAccess
Bump_road

Figure 7 Library and FMU file structure

3.2 Model parameterization

Parameterization is separated from the model definition using DataAccess, see e.g. (Andreasson, 2016) for more elaborate examples. DataAccess allow parameter to be pulled and pushed by the model itself, and thus makes models natively able to interact with data repositories regardless of implementation and execution tool. In this case it means that the FMUs that are exported from the Modelica environment can read and write data to the HMC data-base also when executed in MATLAB and Excel.

The directory and filenames are string parameters that can be changed after export, before initializing the FMU.

3.3 Multi-FMU support

For some applications, for example when there are subsystems from a 3rd party tool. The framework needs to support multiple FMUs. From the Modelica-perspective, the HMC_Commercial_Vehicle library is prepared to support this by the introduction of causal adaptors. The causal adaptors converts the Modelica connectors to a directional signal flow that fits with other causal model implementations. To make this compatible with the interface-template structure, the methodology presented in (Andreasson, 2016) was reused with the addition that compound signal connectors is defined to handle the common cases. Figure 8 shows an example where the model is prepared for an external 3rd party steering FMU.

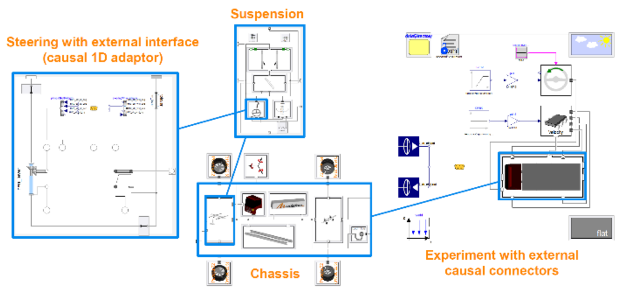

Suspension
Steering with external interface (causal 1D adaptor)
Chassis
Experiment with external causal connectors

Figure 8 Model prepared for external steering FMU.

4 Model deployment

Figure 9 illustrates the model execution of single FMUs, without recompilation. The implementation is based on FMI functionality in MATLAB, (FMIT, 2018) and provides an API for high level operations as well as a graphical user interface. The workflow is based on three steps; configure, parameterize and simulate.

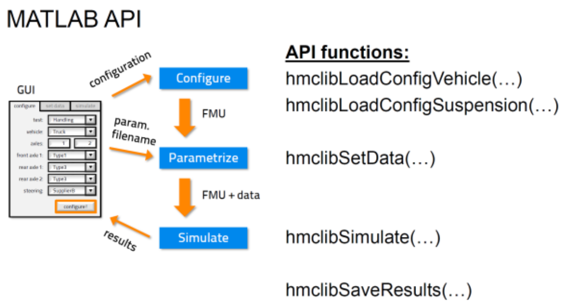
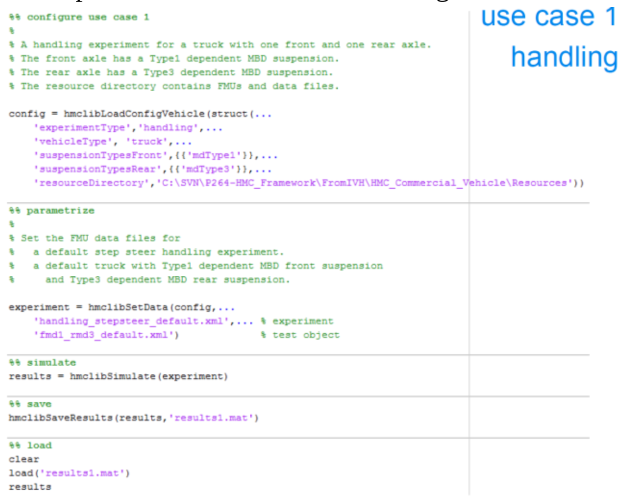


Figure 9 Model execution without recompilation.

4.1 MATLAB API

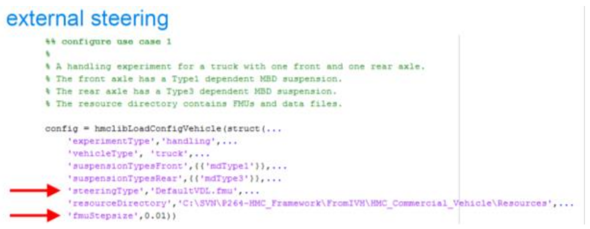
Figure 10 shows an example of how the API can be used to set up and execute simulations of single FMUs.



```
%% configure use case 1
%
% A handling experiment for a truck with one front and one rear axle.
% The front axle has a Type1 dependent MBD suspension.
% The rear axle has a Type3 dependent MBD suspension.
% The resource directory contains FMUs and data files.

config = hmclibLoadConfigVehicle(struct(...
    'experimentType','handling',...
    'vehicleType', 'truck',...
    'suspensionTypesFront',{{'mdType1'}},...
    'suspensionTypesRear',{{'mdType3'}},...
    'resourceDirectory','C:\SVN\P264-HMC_Framework\FromIVH\HMC_Commercial_Vehicle\Resources'))

%% parametrize
%
% Set the FMU data files for
%   a default step steer handling experiment.
%   a default truck with Type1 dependent MBD front suspension
%     and Type3 dependent MBD rear suspension.

experiment = hmclibSetData(config,...
    'handling_stepsteer_default.xml',... % experiment
    'fmd1_rmd3_default.xml')             % test object

%% simulate
results = hmclibSimulate(experiment)

%% save
hmclibSaveResults(results,'results1.mat')

%% load
clear
load('results1.mat')
results
```

Figure 10 Example of API usage.

When adding third party FMUs, an intermediate aggregation step is included, that calls the kernel of FMI Composer (FMIC, 2018). In brief this kernel combines multiple FMUs into one FMU such that all subsequent steps can remain intact. From the API user perspective, this appears as two optional inputs in the function call, Figure 11.



```
%% configure use case 1
%
% A handling experiment for a truck with one front and one rear axle.
% The front axle has a Type1 dependent MBD suspension.
% The rear axle has a Type3 dependent MBD suspension.
% The resource directory contains FMUs and data files.

config = hmclibLoadConfigVehicle(struct(...
    'experimentType','handling',...
    'vehicleType', 'truck',...
    'suspensionTypesFront',{{'mdType1'}},...
    'suspensionTypesRear',{{'mdType3'}},...
    'steeringType','DefaultVDL.fmu',...
    'resourceDirectory','C:\SVN\P264-HMC_Framework\FromIVH\HMC_Commercial_Vehicle\Resources',...
    'fmuStepsize',0.01))
```

Figure 11 Configuring experiment with 3rd party FMU.

4.2 GUI for pre-processing

GUI is built based on MATLAB-Simulink to give engineers convenience of execution. First, pre-processing is performed by inputting the model parameters like Figure 12.

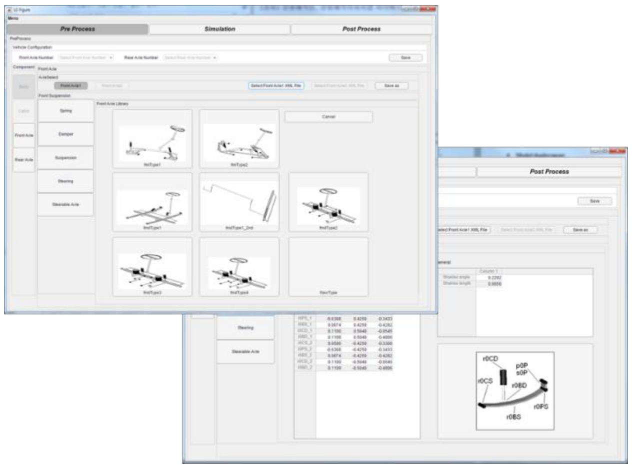
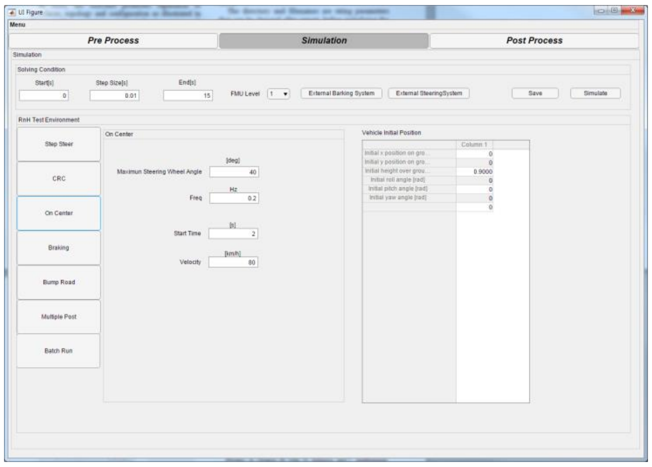


Figure 12 MATLAB GUI for FMUs parameterization

The GUI is employed not just to make connection between model parameters and FMUs, but also to give simulation condition. Figure 13 illustrates the GUI of simulation mode definition. By technical assistance of MATLAB FMI toolbox by Modelon, FMIT (2018), directly MATLAB start simulation of FMUs after this parameterization and simulation condition definition.



Simulation

Solving Condition

Start[s] 0 Step Size[s] 0.01 End[s] 15 FMU Level 1

RnH Test Environment

On Center

Maximun Steering Wheel Angle [deg] 40

Freq Hz 0.2

Start Time [s] 2

Velocity [km/h] 80

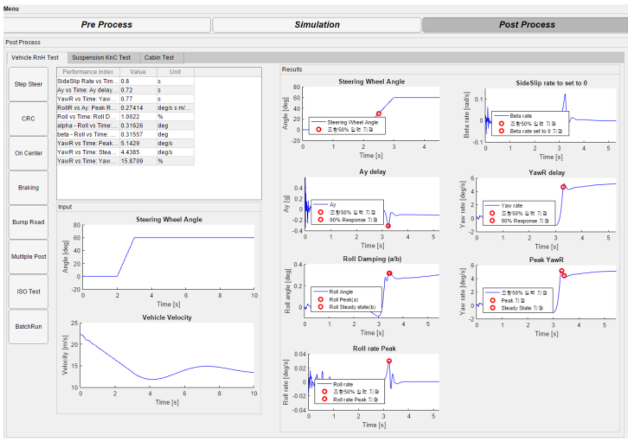
Vehicle Initial Position

	Column 1
Initial x position on gro...	0
Initial y position on gro...	0
Initial height over grou...	0.9000
Initial roll angle [rad]	0
Initial pitch angle [rad]	0
Initial yaw angle [rad]	0
	0

Figure 13 MATLAB GUI for definition of Ride and Handling simulation condition

4.3 Results

Finally, the simulation results are reproduced by plots and Key Performance Indexes that HMC has already defined. Once again, post-processor was developed on the MATLAB for the purpose of establishing consistent work flow from pre to post process. Figure 14 illustrates this automated post process results.



Performance Index	Value	Unit
SideSlip Rate vs Tim...	0.8	s
Ay vs Time: Ay delay...	0.72	s
YawR vs Time: Yaw...	0.77	s
RollR vs Ay: Peak R...	0.27414	deg/s s m/...
Roll vs Time: Roll D...	1.0022	%
alpha - Roll vs Time:...	0.31626	deg
beta - Roll vs Time: ...	0.31557	deg
YawR vs Time: Peak...	5.1429	deg/s
YawR vs Time: Stea...	4.4385	deg/s
YawR vs Time: Yaw...	15.8709	%

Figure 14 Automatically generated Simulation Results

5 Summary and future work

From this collaborating project, HMC is able to set up stable and consistent simulation tool for Ride and Handling performance using FMI technology and it will be utilized during our vehicle development process. Also HMC plan to set up similar workflow in many other research and development section like CFD, fuel consumption.

On the other hand, there is now the possibility to have a client-server approach to Modelica-based engineering (Elmqvist, 2018), which allows such functionality as FMU generation and execution of simulation and optimization to be remotely from the user and without the need for installation of software on the local machine. Such workflows as described here can therefore be carried out from a web browser using the same models as previously described, Figure 12.

References

J. Andreasson, M. Gäfvert: The Vehicle Dynamics Library – Overview and Applications, In proceedings of the 5th International Modelica Conference, Wien, September 2006. https://modelica.org/events/modelica2006/Proceedings/sessions/Session1b3.pdf

J. Andreasson, N. Machida, M. Tsushima, J. Griffin, P. Sundström: Deployment of high-fidelity vehicle models for accurate real-time simulation, In proceedings of 1st Japanese Modelica Conference, May 23-24, Tokyo, Japan, 2016. http://www.ep.liu.se/ecp/article.asp?issue=124&article=0

H. Elmqvist, M. Malmheden, J. Andreasson. A Web Architecture for Modeling and Simulation, In proceedings of 2nd Japanese Modelica Conference, May 17-18, Tokyo, Japan, 2018.

FMI, Functional Mockup Interface, 2018 http://www.fmi-standard.org

FMIC, FMI Composer, 2018. http://www.modelon.com/products/modelon-deployment-suite/fmi-composer

FMIE, FMI Add-in for Microsoft Excel, 2018. http://www.modelon.com/products/modelon-deployment-suite/fmi-add-in-for-excel

FMIT, FMI Toolbox for MATLAB/Simulink, 2018. http://www.modelon.com/products/modelon-deployment-suite/fmi-toolbox-for-MATLABsimulink

M. Henningsson, J. Åkesson, H. Tummescheit: An FMI-Based Tool for Robust Design of Dynamical Systems, In proceedings of 10th International Modelica Conference, March, Lund, Sweden, 2014. https://www.modelica.org/events/modelica2014/proceedings/html/submissions/ECP1409635_HenningssonAkessonTummescheit.pdf

pyFMI, FMI support for python, 2018. https://pypi.python.org/pypi/PyFMI

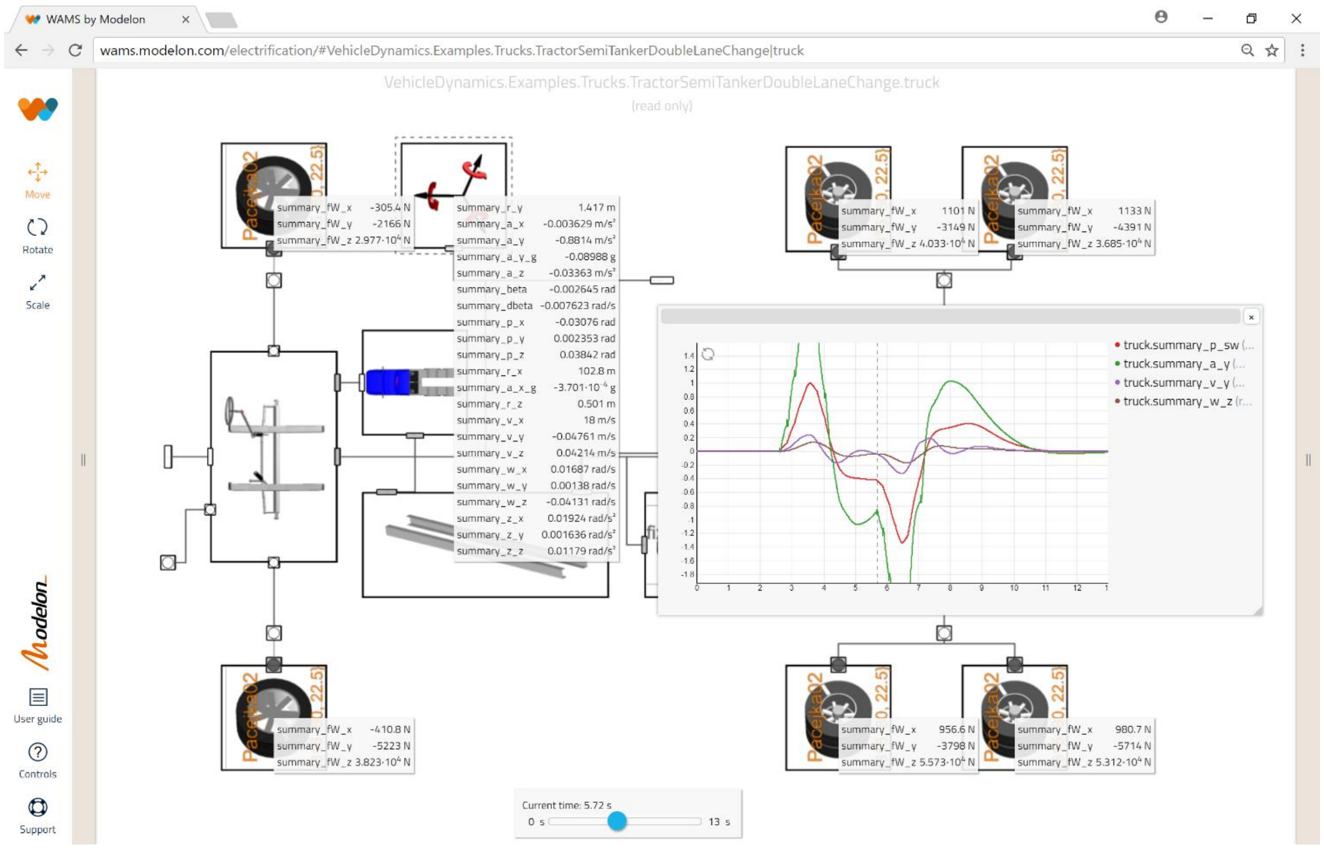
M. Sielemann, A. Pitchaikani, N. Selvan, M. Sammak: The Jet Propulsion Library: Modeling and simulation of aircraft engines, 2017. In proceedings of 12th International Modelica Conference, May, Prague, 2017.

M. Sielemann, J. Andreasson: Towards Model-Based Design of Aircraft Systems, 2017. NAFEMS World Congress 2017.

P. Sundström and J. Andreasson: Model-based design and control of long heavy vehicle combinations, In proceedings of 2016 IEEE Intelligent Vehicles Symposium, June 19-22, Gothenburg, Sweden, 2016.

P. Sundström, M. Henningsson, X. Carrera Akutain, Y. Hirano, A. Ocariz, H. Iida, N. Aikawa, and J. Andreasson: Virtual Vehicle Kinematics and Compliance Test Rig, In proceedings of 1st Japanese Modelica Conference, May 23-24, Tokyo, Japan, 2016. http://www.ep.liu.se/ecp/article.asp?issue=124&article=004

VDL, Vehicle Dynamics Library, 2018 http://www.modelon.com/products/modelon-library-suite/vehicle-dynamics-library


WAMS by Modelon
wams.modelon.com/electrification/#VehicleDynamics.Examples.Trucks.TractorSemiTankerDoubleLaneChange|truck
VehicleDynamics.Examples.Trucks.TractorSemiTankerDoubleLaneChange.truck
(read only)
Move
Rotate
Scale
Modelon
User guide
Controls
Support
● truck.summary_p_sw (...
● truck.summary_a_y (...
● truck.summary_v_y (...
● truck.summary_w_z (r...

summary_fW_x	-305.4 N
summary_fW_y	-2166 N
summary_fW_z	$2.977 \cdot 10^4$ N

summary_r_y	1.417 m
summary_a_x	-0.003629 m/s²
summary_a_y	-0.8814 m/s²
summary_a_y_g	-0.08988 g
summary_a_z	-0.03363 m/s²
summary_beta	-0.002645 rad
summary_dbeta	-0.007623 rad/s
summary_p_x	-0.03076 rad
summary_p_y	0.002353 rad
summary_p_z	0.03842 rad
summary_r_x	102.8 m
summary_a_x_g	$-3.701 \cdot 10^{-4}$ g
summary_r_z	0.501 m
summary_v_x	18 m/s
summary_v_y	-0.04761 m/s
summary_v_z	0.04214 m/s
summary_w_x	0.01687 rad/s
summary_w_y	0.00138 rad/s
summary_w_z	-0.04131 rad/s
summary_z_x	0.01924 rad/s²
summary_z_y	0.001636 rad/s²
summary_z_z	0.01179 rad/s²

summary_fW_x	1101 N
summary_fW_y	-3149 N
summary_fW_z	$4.033 \cdot 10^4$ N

summary_fW_x	1133 N
summary_fW_y	-4391 N
summary_fW_z	$3.685 \cdot 10^4$ N

summary_fW_x	-410.8 N
summary_fW_y	-5223 N
summary_fW_z	$3.823 \cdot 10^4$ N

summary_fW_x	956.6 N
summary_fW_y	-3798 N
summary_fW_z	$5.573 \cdot 10^4$ N

summary_fW_x	980.7 N
summary_fW_y	-5714 N
summary_fW_z	$5.312 \cdot 10^4$ N

Current time: 5.72 s
0 s 13 s

Figure 12 Client-server based truck simulation using web architecture for modeling and simulation. (Note that web link is not available without special access.)

Modelling of Oil Film Bearings

Tatsuro Ishibashi[1] Tadao Kawai[2]

[1]Meidensha Corporation, Japan, ishibashi-tat@mb.meidensha.co.jp
[2]Department of Mechanical & Physical Engineering, Osaka City University, Japan, kawai@mech.eng.osaka-cu.ac.jp

Abstract

In this paper, we model the oil film bearings and estimate the fluid-induced instability for the design and the diagnosis of the rotating machinery system. The presented model is implemented in our original rotating machinery library by Modelica. An example of a Jeffcott rotor system supported by plain circular journal bearings is simulated. To check the behavior of the model, Campbell diagrams and stability maps are computed by using the Modelica_LinearSystems2 library.

Keywords: Rotor Dynamics, Oil Whirl, Oil Whip, Campbell Diagram, Stability Map

1 Introduction

Oil film bearings are widely used for the large rotating machinery systems such as turbines and generators. The oil film bearing has the following advantages compared to the rolling bearing. The oil film bearings provide the higher damping, which is required to pass through a critical speed and suppress vibration. Those also reduce noise, and have very long life under normal load condition because of the lack of contact between rotating parts. In harsh operating conditions such as high load and moderately high speed rotation, the oil film bearing has a superior performance. Some disadvantages of the oil film bearing are the higher friction, the higher susceptibility to particulate contamination and that it cannot run without a lube system.

At high rotating speed, self-excited vibration due to the motion of the oil film may occur in the rotating machinery system supported by oil film bearings. This instable vibration, generally called oil whirl or whip causes damage to the machine. To design the high rotating speed machinery, it is necessary to understand this instable vibration mechanism and prevent it. To diagnose the rotating machinery system supported by oil film bearings, it is important to grasp the behavior with faults such as unbalance and shaft bending etc.

In this paper, the oil lubricated plain journal bearing model is implemented in our original Rotating Machinery library (Ishibashi *et al*, 2017). We simulate the rotating machinery system supported by oil film bearings with static unbalance in a rotor. The components in this system are from our original rotating machinery library based on transfer matrix method. To check the behavior of the model, Campbell diagrams and stability maps are computed by using the Modelica_LinearSystems2 library.

2 Hydrodynamic lubrication

In this Section, we derive the Reynolds equation in order to determine the force that the oil film exerts on the shaft, that is, the pressure distribution of the oil film (Hori and Kato, 2008; Ishida and Yamamoto, 2012; Matsushita *et al*, 2017). Figure 1 schematically shows both walls of the shaft and the bearing are separated by an oil film.

By treating the oil film as a continuum, the equation of motion of the oil film in three directions is derived from the Navier-Stokes equation. The majority of oil lubricated journal bearings operate in the slow viscous flow regime where the viscous forces are much greater than the inertia forces. In most of the industrial application, the bearing Reynolds number is usually below 1,500. Here, by simplifying various assumptions (rigid wall, incompressible Newtonian fluid, isothermal, isoviscous and isopycnic fluid, laminar flow, superior of differential coefficient in oil film thickness direction and ignoring inertial forces), equations of motion in the *x*, *y* and *z* directions are expressed as follows at any coordinate (*x*, *y*, *z*).

$$\mu \frac{\partial u^2}{\partial y^2} = \frac{\partial p}{\partial x} \quad (1)$$

$$\mu \frac{\partial v^2}{\partial y^2} = 0 \quad (2)$$

$$\mu \frac{\partial w^2}{\partial y^2} = \frac{\partial p}{\partial z} \quad (3)$$

Here, μ is the uniform fluid viscosity of the lubricating oil, *u*, *v*, *w* are the oil film flow velocity in the *x*, *y*, *z* direction, respectively. The pressure generated in the oil film varies depending on the in-plane coordinate *x*, *z*, but it is uniform in the *y* direction. By assuming that the oil film flow velocity on the wall surface is equal to the velocity of the wall surface motion, the boundary conditions are the velocities on the wall surfaces respectively. By integrating Equations 1, 2 and 3, the flow velocity is obtained as follows.

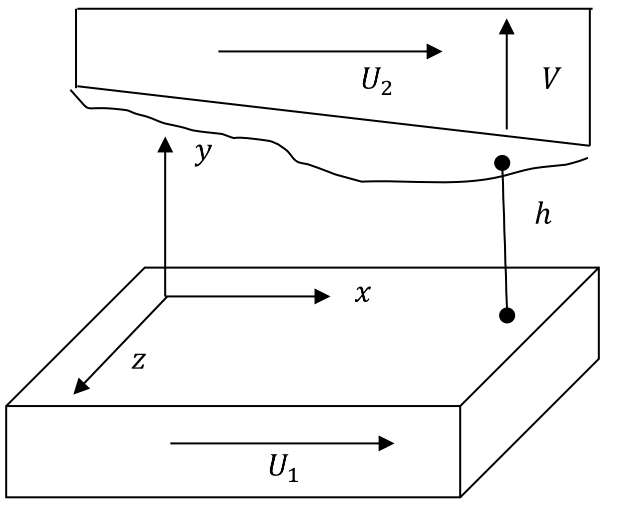


Figure 1. Lubrication surface model. *x* is the journal rotating direction, *y* is the oil film thickness direction, and *z* is the bearing width direction. The oil film thickness *h* does not change in the *z* direction, but narrows down gradually in the *x* direction. On the bearing surface *y*=0, the oil film moves at velocity U_1 in *x* direction. On the journal surface *y*=*h*, it moves at velocity U_2 in *x* direction. And it moves at velocity *V* in *y* direction.

$$u = \left\{U_2 + (U_1 - U_2)\frac{h-y}{h}\right\} + \left\{-\frac{y(h-y)}{2\mu}\frac{\partial p}{\partial x}\right\} \tag{4}$$

$$v = V\frac{y}{h} \tag{5}$$

$$w = -\frac{y(h-y)}{2\mu}\frac{\partial p}{\partial z} \tag{6}$$

These flow velocity equations are substituted into the equation obtained by integrating the following continuous Equation 7 once for *y*.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{7}$$

A formula for converting the order of differentiation and integration is applied with respect to terms of *x* derivative and *z* derivative of the pressure *p*,

$$\int_0^{h(x)} \frac{\partial}{\partial x} f(y,x)dy = \frac{\partial}{\partial x}\int_0^{h(x)} f(y,x)dy - f(h(x),x)\frac{\partial h(x)}{\partial x} \tag{8}$$

A three-dimensional dynamic Reynolds equation is obtained as follows.

$$\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial z}\left(h^3\frac{\partial p}{\partial z}\right) = 6\mu(U_1 - U_2)\frac{\partial h}{\partial x} + 6\mu h\frac{\partial}{\partial x}(U_1 + U_2) + 12\mu V \tag{9}$$

3 Static property of oil film bearing

Equation 9 is adapted to the plain circular journal bearing which have the radius *R*, the width *L* and the radial clearance *C* shown in Figure 2. The uniform fluid viscosity μ, the constant journal circumferential velocity *U* and the constant bearing load *W* are assumed.

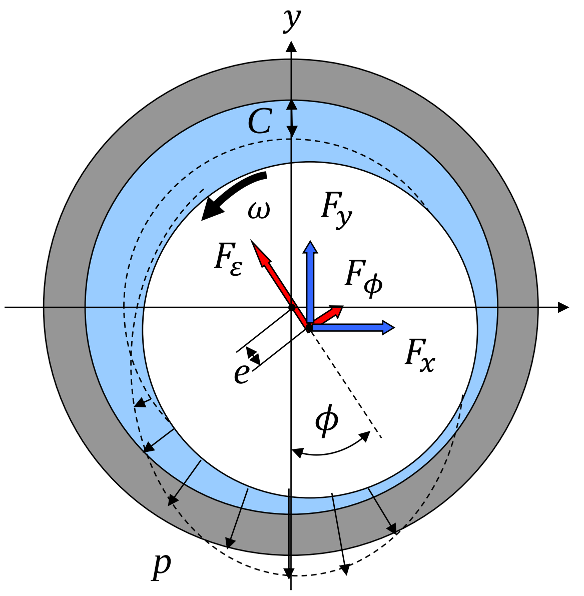


Figure 2. Oil film force and pressure of the plain circular journal bearing.

Substituting $U_1 = 0, U_2 = U,\ V = U\frac{\partial h}{\partial x} + \frac{\partial h}{\partial t}$ into Equation 9 and applying the short bearing approximation $\frac{\partial p}{\partial x} \ll \frac{\partial p}{\partial z}$, Equation 9 can be calculated as follows.

$$h^3\frac{\partial^2 p}{\partial z^2} = 6\mu U\frac{\partial h}{\partial x} + 12\mu\frac{\partial h}{\partial t} \tag{10}$$

As the actual *C/R* is very small value, the oil film thickness *h* is approximately expressed as follows.

$$h = C(1 + \varepsilon\cos\phi) \tag{11}$$

ε is the eccentricity ratio *e/C*, where *e* is the distance between the bearing and the journal center, ϕ is the journal center angle from the maximum clearance position for the direction of the journal rotation. Here, $x = R\phi$. Equation 10 is expressed by the following equation.

$$\frac{d^2p}{dz^2} = -\frac{6\mu}{C^2}\frac{\varepsilon\sin\phi}{(1+\varepsilon\cos\phi)^3}\left(\omega - 2\dot{\phi}\right) + \frac{12\mu}{C^2}\frac{\dot{\varepsilon}\cos\phi}{(1+\varepsilon\cos\phi)^3} \tag{12}$$

Integrated twice under the boundary condition *p*=0 at the bearing edge (*z=0*, *L*) with respect to *z*.

$$p = \left(\frac{3\mu}{C^2}\frac{\varepsilon\sin\phi}{(1+\varepsilon\cos\phi)^3}\left(\omega - 2\dot{\phi}\right) - \frac{6\mu}{C^2}\frac{\dot{\varepsilon}\cos\phi}{(1+\varepsilon\cos\phi)^3}\right) \times z(L-z) \tag{13}$$

The oil film force is calculated by multiplying the oil film pressure by $-\cos\phi$ and $\sin\phi$, applying Gumbel's boundary condition and integrating in the bearing area $\theta = 0\sim\pi\ ,\ z = 0\sim L$.

$$F_\varepsilon = \int_0^L\int_0^\pi (-p\cos\phi)Rd\phi dz = \frac{\mu R L^3}{C^2}\left(\frac{\varepsilon^2\left(\omega - 2\dot{\phi}\right)}{(1-\varepsilon^2)^2} + \frac{\pi\dot{\varepsilon}(1+2\varepsilon^2)}{2(1-\varepsilon^2)^{5/2}}\right) \tag{14}$$

$$F_\theta = \int_0^L \int_0^\pi (\mathrm{p}\sin\phi) R d\phi dz$$
$$= \frac{\mu R L^3}{C^2}\left(\frac{\pi\varepsilon\left(\omega - 2\dot{\phi}\right)}{4(1-\varepsilon^2)^{3/2}} + \frac{2\varepsilon\dot{\varepsilon}}{(1-\varepsilon^2)^2}\right) \tag{15}$$

For the static equilibrium position $\dot{\varepsilon} = \dot{\phi} = 0$, the oil film force is calculated as follows,

$$F_{\varepsilon_0} = \frac{\mu R L^3}{C^2}\left(\frac{\varepsilon_0^2\omega}{(1-\varepsilon_0^2)^2}\right) \tag{16}$$

$$F_{\phi_0} = \frac{\mu R L^3}{C^2}\left(\frac{\pi\varepsilon_0\omega}{4(1-\varepsilon_0^2)^{3/2}}\right) \tag{17}$$

The eccentricity angle at the static equilibrium position is determined as follows.

$$\tan\phi_0 = \frac{F_{\phi_0}}{F_{\varepsilon_0}} = \frac{\pi\sqrt{1-\varepsilon_0^2}}{4\varepsilon_0} \tag{18}$$

The oil film force *x*, *y* direction is written as follows

$$F_x = -F_{\varepsilon_0}\sin\phi_0 + F_{\phi_0}\cos\phi_0 \tag{19}$$

$$F_y = F_{\varepsilon_0}\cos\phi_0 + F_{\phi_0}\sin\phi_0 \tag{20}$$

At the static equilibrium position, the oil film force balances with static load *W*. Hence ε_0 is written as follows.

$$S\left(\frac{L}{2R}\right)^2 = \frac{(1-\varepsilon_0^2)^2}{\pi\varepsilon_0\sqrt{\pi^2 + (16-\pi^2)\varepsilon_0^2}} \tag{21}$$

Here, *S* is Sommerfeld number written as follows.

$$S = \frac{2\mu N R L}{W}\left(\frac{R}{C}\right)^2 \tag{22}$$

Here, *N* is the shaft rotating speed (rps, 1/s). When $S\left(\frac{L}{2R}\right)^2$ is given, the static equilibrium position ε_0, ϕ_0 is uniquely determined.

4 Dynamic property of oil film bearing

Next we consider the dynamic property. The oil film force is highly nonlinear to solve. In order to discuss the linear stability of a shaft, the oil film force is linearized beforehand in the neighborhood of the static equilibrium position of the journal center. Further, in order to consider the journal motion in the rectangular coordinates system (*x*, *y*) shown in the same figure, let us transform the polar components F_ε and F_ϕ of the oil film force to the rectangular components F_x and F_y. Then the oil film forces F_x and F_y can be written in the following form.

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{bmatrix}\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{23}$$

The oil film force is expressed by the four spring coefficients and the four damping coefficients. Each coefficient is made dimensionless by converting as follows.

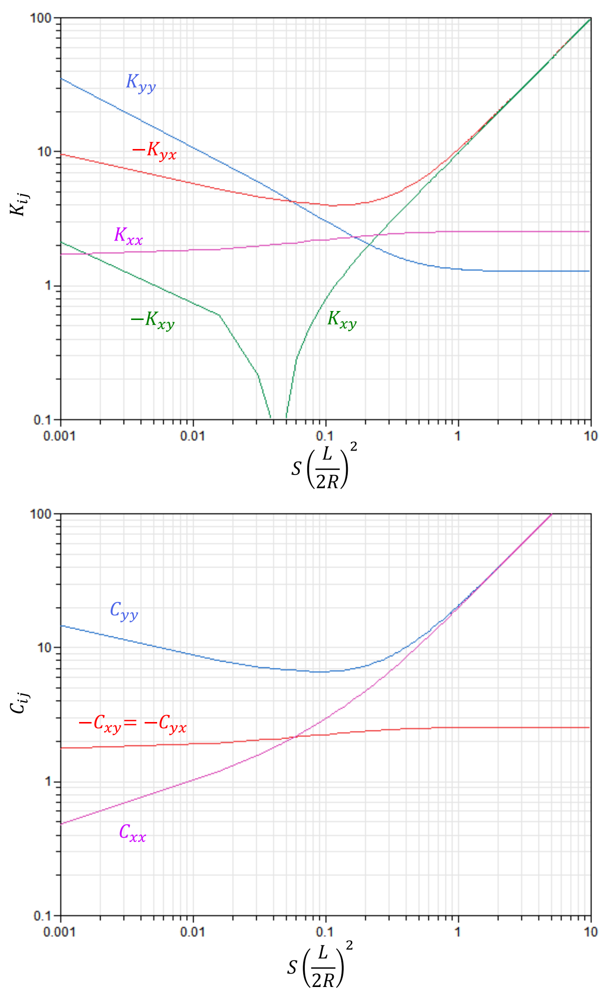


Figure 3. The dimensionless coefficients of the plain circular journal bearing in case of the short bearing approximation. Spring coefficients (upper). Damping coefficients (lower).

$$K_{ij} = \frac{C}{W}k_{ij} \tag{24}$$

$$C_{ij} = \frac{C\Omega}{W}c_{ij} \tag{25}$$

For example, the dimensionless coefficients of the plain circular journal bearing in case of the short bearing approximation and Gumbel's boundary conditions are calculated as follows.

$$K_{xx} = \frac{4[2\pi^2 + (16-\pi^2)\varepsilon_0^2]}{K_\alpha} \tag{26}$$

$$K_{xy} = \frac{\pi[\pi^2 - 2\pi^2\varepsilon_0^2 - (16-\pi^2)\varepsilon_0^4]}{\varepsilon_0(1-\varepsilon_0^2)^{0.5}K_\alpha} \tag{27}$$

$$K_{yx} = -\frac{\pi[\pi^2 + (32+\pi^2)\varepsilon_0^2 + 2(16-\pi^2)\varepsilon_0^4]}{\varepsilon_0(1-\varepsilon_0^2)^{0.5}K_\alpha} \tag{28}$$

$$K_{yy} = \frac{4[\pi^2 + (32+\pi^2)\varepsilon_0 + 2(16-\pi^2)\varepsilon_0^4]}{(1-\varepsilon_0^2)K_\alpha} \tag{29}$$

$$C_{xx} = \frac{2\pi(1-\varepsilon_0^2)^{0.5}[\pi^2 - 2(8-\pi^2)\varepsilon_0^2]}{\varepsilon_0 K_\alpha} \tag{30}$$

$$C_{xy} = C_{yx} = -\frac{8[\pi^2 - 2(8-\pi^2)\varepsilon_0^2]}{K_\alpha} \quad (31)$$

$$C_{yy} = \frac{2\pi[\pi^2 + 2(24-\pi^2)\varepsilon_0^2 + \pi^2\varepsilon_0^4]}{\varepsilon_0(1-\varepsilon_0^2)^{0.5}K_\alpha} \quad (32)$$

$$K_\alpha = \{\pi^2 + (16-\pi^2)\varepsilon_0^2\}^{1.5} \quad (33)$$

The dimensionless coefficients are expressed as the functions of the eccentricity ε_0 at the static equilibrium position. Therefore, as the static equilibrium position changes, the dimensionless coefficients change. In Figure 3, we plot the dimensionless coefficients against $S\left(\frac{L}{2R}\right)^2$, which uniquely detemines the static equilibrium position ε_0, ϕ_0.

The dimensionless coefficients of the other type of journal bearings are obtained by computing Reynolds equation with the specific boundary condition or referring to the database (Someya, 1989).

5 Modelica Implementation

The presented oil film force models must be supplied by constraints in the transverse direction *x*, *y* and rotating angle direction. Our original Rotating Machinery library is used to supply these constraints (Ishibashi *et al*, 2017). The basic `flange` of this library has 5 DOF (degree of freedom), consisting of 4 DOF (two dimensional deflections and slopes) for transverse vibration of the rotor system and 1 DOF (rotating angle) for torsional vibration, neglecting axial vibration. Features like unbalanced rotors, flexible beams (shaft), supports, springs and dampers are all represented. The library is used to create the total rotating machinery system. The above static and dynamic of the oil film force models are implemented respectively.

The oil film force models are implemented with two connectors, each with 5 DOF. Since the above oil film force models has the only 3 DOF, the moments are set as zero. These connectors are the connections to the journal in the bearing (3 connectors rotor without unbalance) and the support.

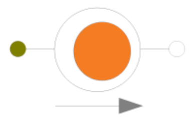


Figure 4. Modelica Icon of the oil film force.

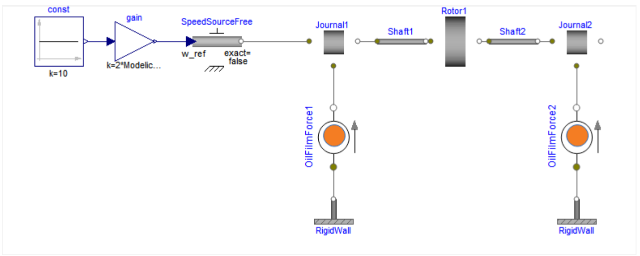


Figure 5. The rotating machinery system supported by the oil film bearing.

In Figure 4 the icon of the oil film force model is shown. The static and dynamic models have the same icon. No inertias or constraints are included in the model. Using our original rotating machinery library, it is possible to create rotating machinery systems. A simple rotating machinery system supported by the plain circular journal bearing is easily generated. Here, we treat Jeffcott rotor system as a test case in Figure 5. In the models, the static and dynamic oil film force models (Figure 4) are defined as described in this paper, all other components are components of our rotating machinery library.

6 Simulation Results

6.1 Static property

A static equilibrium position of a simple rotor machinery system supported by the oil film bearing (as shown in Figure 5) is estimated. Here, the oil film bearing is the plain circular journal bearing explained in the above section. By replacing the constant input with the ramp input and simulating the static model (in which rotors and shafts have only the loads of weights), the position of the Journal1 `flange` connected to the oil film force model i.e. the static equilibrium position is calculated. Also, the static equilibrium position is the same as the journal center. Figure 6 shows the trajectory of the journal center when the rotating speed increases. In the plain circular journal bearing, the static equilibrium position draws the trajectory close to semicircular arc shape. As the journal rotating speed increases infinitely, the journal center reaches the bearing center position.

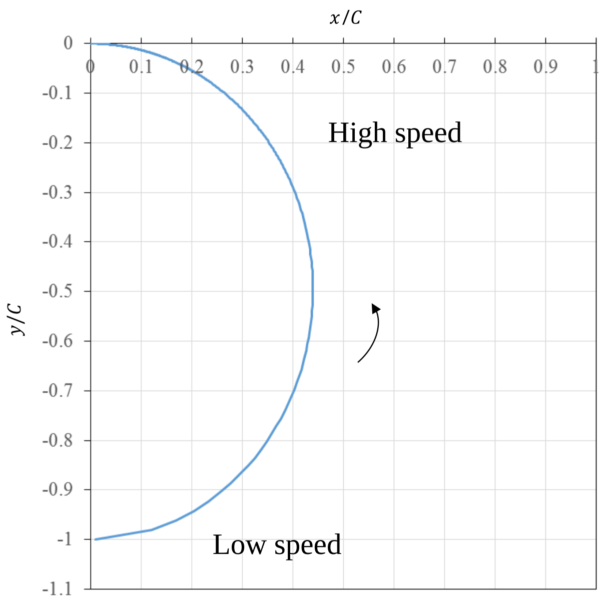


Figure 6. The static equilibrium position.

6.2 Dynamic property

At the constant rotating speed, the static equilibrium position and the linearized dynamic oil film forces are uniquely determined. Using the model in Figure 5, the dynamic property of the journal center around the static equilibrium position is computed. The model has just a static unbalance in Rotor1. This system has the critical speed (the natural frequency of lateral vibration) around 42 Hz (see Figure 8). Figure 7 shows the transient simulation results of the journal center position in *x* direction at the different constant rotating speed. At 80rps, the vibration gradually diverges. This implies the instable vibration. To analyze these simulation results and the dependency of the rotating speed, these simulation results are processed by FFT (Fast Fourier Transformation). The waterfall plot is created as shown in Figure 8 (a). In the light shaft system such as Figure 8 (a), the vibration of the half rotating speed increases as the rotating speed increases. When the rotating speed reaches twice the critical speed, a large whirl occurs. Because the whirling speed of the half-speed whirl coincides with the natural frequency of the system. Beyond this point, a large whirling still continues.

To check the oil film bearing response, we simulate the model by increasing the constant bearing load *W*. Figure 8 (b) shows the result. Below the critical speed, the half-speed whirl is not observed. The static unbalance vibration synchronized with the rotating speed is observed. Beyond the critical speed, the large vibration of the natural frequency is still observed.

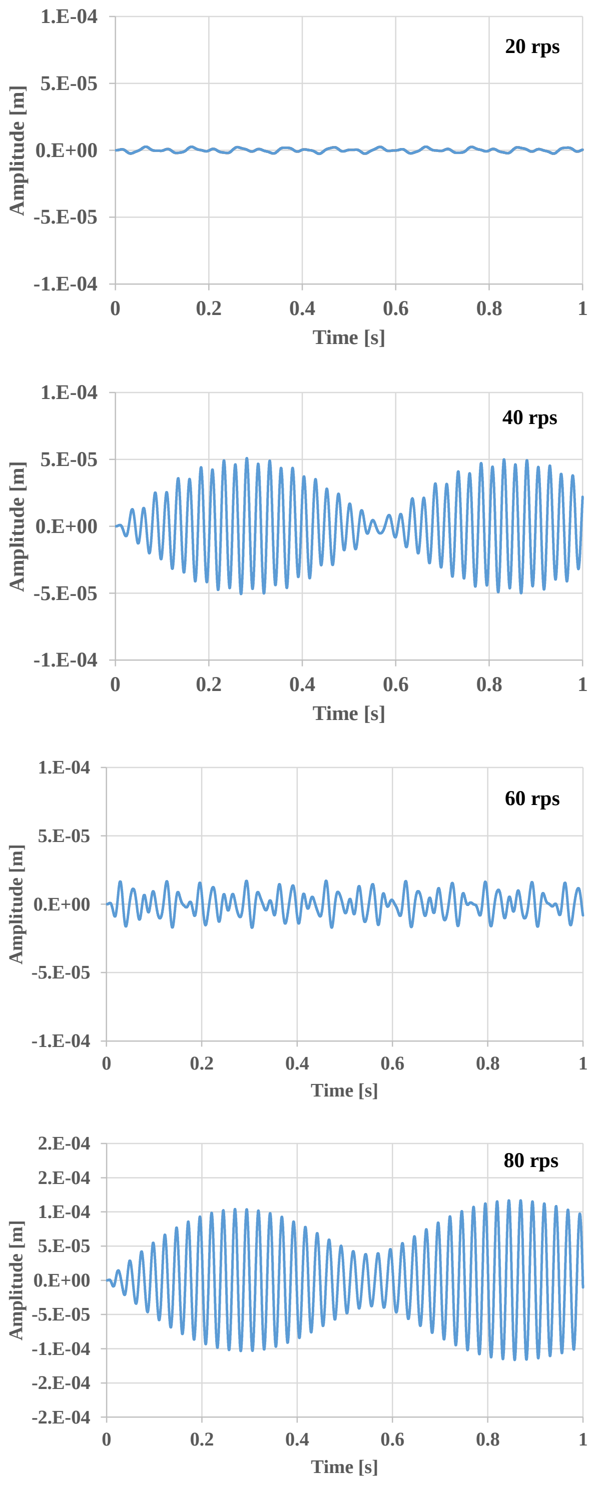


Figure 7. The simulation results of the journal center vibration at the constant rotating speed in *x* direction. This system is the light shaft system.

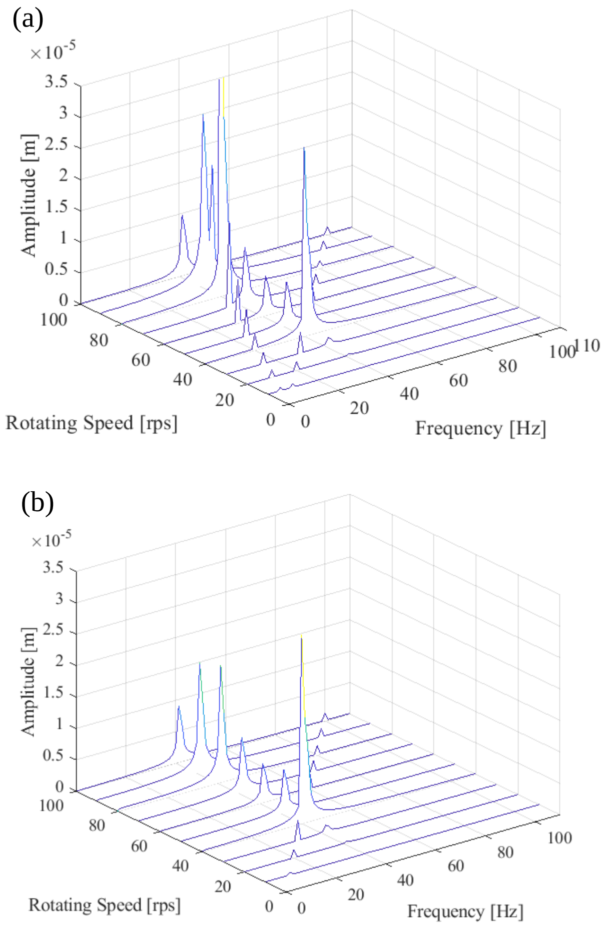


Figure 8. Waterfall plots of the simulation results.

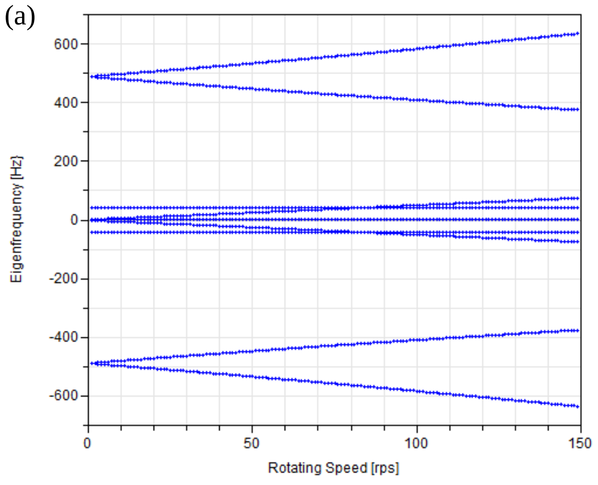
(a) Light shaft. (b) Heavy shaft.

6.3 Eigenfrequency and Stability Analysis

Using the Modelica_LinearSystems2 library (Bauer *et al*, 2009; Otter, 2006) and creating the functions, it is possible to create Campbell diagrams (also known as whirl speed maps) and stability maps, which show variations of eigenfrequencies and damping ratios of rotors with respect to the rotating speed. The models are linearized into the state space and represented in ABCD matrices by the Modelica_LinearSystems2 library. Computing eigenvalues of the matrix A and transforming into eigenfrequencies and damping ratios at the constant rotating speed, Campbell diagrams and stability maps are obtained. In rotating machinery systems, the eigenfrequencies often depend on the rotating speeds due to the induced gyroscopic effects or variable hydrodynamic conditions in fluid bearings. The intersection between the synchronous excitation line and the eigenfrequencies in Campbell diagrams are referred as critical speeds. In fluid bearing, damping ratios turn negative from positive as the rotating speed increases in stability maps. This speed is known as the instability threshold. Self-excited instability occurs at the speeds above the instability threshold. It is very important for design and diagnose the rotating machinery systems to compute analytically and measure experimentally critical speeds and instability thresholds.

Here, we analyze the model corresponding to Figure 8. Figure 9 shows the Campbell diagrams. The Campbell diagrams show both forward and backward vibration modes. The Campbell diagrams of both of the light and heavy shaft show the totally same behavior. The critical frequency is around 42 Hz, at which frequency the lateral vibration occurs. The eigenfrequency curves starting from around 500Hz split due to the gyroscopic effects. Since the model only have a static unbalance, the vibration amplitude is too small to observe. The half-speed whirl is observed due to the oil film.

Figure 10 shows the stability map. There is differences in the damping ratio curves between the light shaft and the heavy shaft. In the light shaft system, the damping ratio turns negative below the critical speed. From this speed the instable self-excited vibration called oil whirl occurs. The damping ratio reaches minimum at twice the critical speed. Over twice the critical speed, it increases. The oil whirl develops to the large whirl called oil whip at twice the critical speed and shrinks over twice the critical speed.

(a)


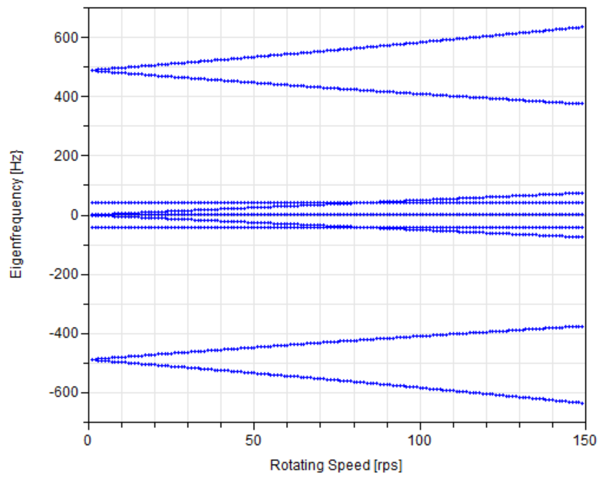
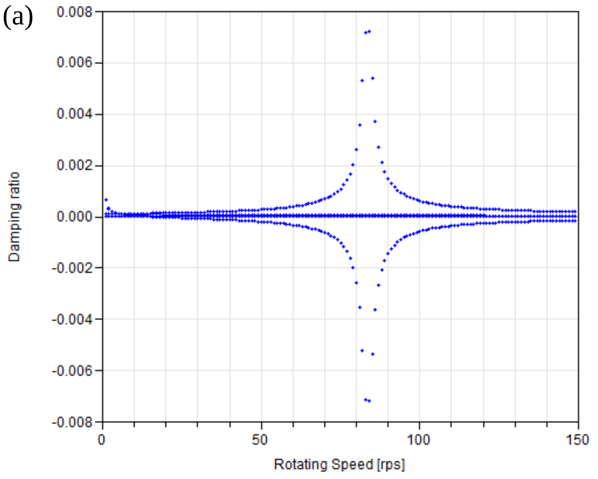


Figure 9. Campbell diagrams**.** (a) Light shaft. (b) Heavy shaft.

(a)


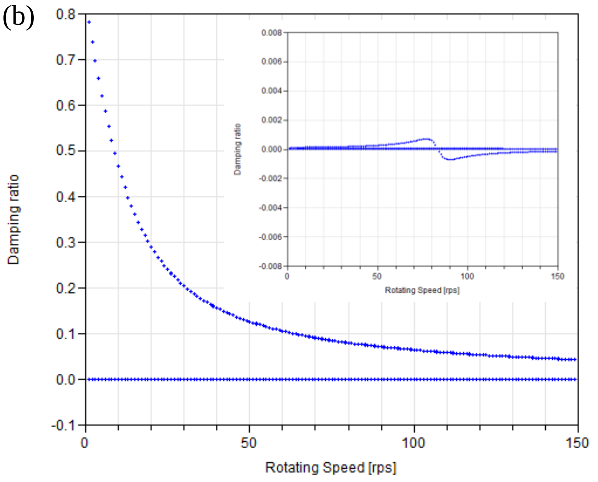
(b)


Figure 10. Stability maps. (a) Light shaft. (b) Heavy shaft. The inset shows the enlarged view.

In the heavy shaft system, the damping ratio turns negative just below twice the critical speed. In the light shaft system, the damping ratios are smaller than in the heavy shaft. This means the light shaft system is more instable. These behaviors are well consistent with Figure 8.

7 Conclusion

In this paper, models are presented to describe the oil film force. Using our original rotating machinery library, it is possible to model a rotating machinery system supported by an oil film bearing. An example of Jeffcott rotor system supported by plain circular journal bearings is simulated and analyzed. The presented models make it possible to estimate Campbell diagrams and stability maps of the rotating machinery system by using the Modelica_LinearSystems2 as well as transient simulation results. The presented models show the abilities to design and diagnose rotating machinery systems.

References

Marcus Baur, Martin Otter and Bernhard Thiele. Modelica Libraries for Linear Control Systems. *Proceedings of the 7th International Modelica Conference,* 2009. doi: 10.3384/ecp09430068.

Yukio Hori and Koji Kato. Studies on tribology. *Proceedings of the Japan Academy, Series B*, 84(8): 287–320, 2008.

Tatsuro Ishibashi, Han Bing and Tadao Kawai. Rotating Machinery Library for Diagnosis. *Proceedings of the 12th International Modelica Conference,* 2017. doi:10.3384/ecp17132381.

Yukio Ishida and Toshio Yamamoto. *Linear and Nonlinear Rotordynamics: A Modern Treatment with Applications, 2nd Edition*. Wiley-VCH, 2012.

Osami Matsushita, Masato Tanaka, Hiroshi Kanki, Masao Kobayashi and Patrick Keogh. *Vibrations of Rotating Machinery*. Springer Japan. 2017. doi: 10.1007/978-4-431-55456-1.

Martin Otter. The LinearSystems library for continuous and discrete control systems. *Proceedings of the 5th International Modelica Conference,* 2006. URL https://www.modelica.org/events/modelica2006/Proceedings/sessions/Session5c1.pdf/view.

Tsuneo Someya. *Journal-Bearing Databook.* Springer-Verlag Berlin Heidelberg. 1989. doi: 10.1007/978-3-642-52509-4.

Gas Compressor System Simulation using Functional Mockup Interface for Human Machine Interface and Control

Ryan Magargle[1], Hemanth Kolera-Gokula[1]

[1]ANSYS, Inc. USA, {ryan.magargle, hemanth.kolera-gokula}@ansys.com

Abstract

This paper describes a gas compressor (GCS) system simulation for the purpose of verifying a controller's operation and interfacing simulation with measured data. The GCS is used to collect and compress low pressure gas streams for transmission into larger higher-pressure lines. This system is simulated with ANSYS TwinBuilder. It includes models of the compressor, motor, bypass valve, and piping along with pressure sources for the low and high-pressure lines, which are controlled with measured or test data. To cycle and pressurize the GCS, a controller is implemented using an FMU from ANSYS SCADE. To interact with the system and monitor current operating conditions, a human machine interface (HMI) is also implemented using ANSYS SCADE as an FMU.

Keywords: digital twin, vapor recovery unit, FMI, ANSYS, TwinBuilder, SCADE, control, HMI, Rapid Prototyper

1 Introduction

The GCS system shown in Figure 1 was created using ANSYS TwinBuilder (ANSYS, 2017). The GCS system compresses the low-pressure gas in the storage unit on the left through a low pressure reservoir and sends it to the outlet line on the bottom right. While it is pressurizing and sending the gas to the outline line, the bypass valve is closed so no gas is recycled back into the scrubber tank. If the input pressure fed from the storage goes too low, the compressor can enter a low power mode and the bypass valve will open allowing gas to recycle back through the scrubber tank. The scrubber tank is responsible for removing contaminants from the gas flow, removing them as a second liquid phase, but is not critical for the flow analysis considered in this model, and is treated as a simple tank.

The usage cases for this type of model are 1) verification of controller operation, and 2) connection with real-time data to provide analytical support in health monitoring applications. The discussion considered within this paper primarily addresses the first usage case. Extension of the existing model to be a connected simulation based digital twin will be left to future work and has been discussed in previous papers.

In verifying the controller operation, measured source pressure data from the field can be replayed as an input to the system schematic and run under the supervision of the controller to ensure all states are properly entered and exited. Alternatively, test signals can be input at the source to cycle through all possible input pressures. The models implemented in the system described here are lower-fidelity, high speed quasi-steady state models able to simulate large time periods at the expense of local dynamic behavior.

2 Model Overview

The various models implemented for the system components are described in this section. The main schematic consists of numerous top-level macro-

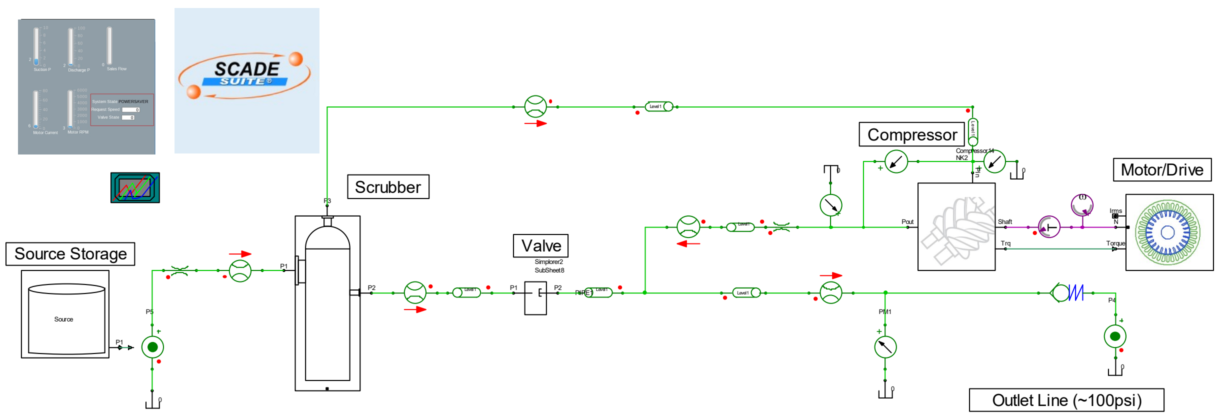


Figure 1. System Schematic in ANSYS TwinBuilder

models, or packaged models containing various sub-models and elements. This keeps the top-level schematic from getting too cluttered from components that may have multiple sub-components in the system. Macro-models schematics are preserved and maybe be entered and modified at any time, and any of their internal states may be made available to the main schematic.

2.1 Source Storage

The storage source is treated as a pressure source. This behaves as the boundary to the simulation where measured data can be input or test signals can be applied.

As seen in Figure 2, the macro-model feeds a pressure source. The macro-model has two paths defined. The signal flow path on the top simply applies a triangular waveform to sweep the pressure. The signal flow path on the bottom imports a text file containing the pressure data obtained from the field and applies this to the output of the macro-model to feed the pressure source. One signal flow path can be active at a time, the other signal flow path is simply deactivated.

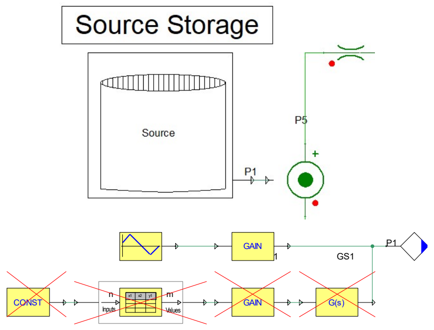


Figure 2. Macro-model and sub-circuit of source storage model.

2.2 Scrubber Tank

The scrubber tank is treated as a basic tank in this model. As seen in Figure 3, The macro-model has three connections, the source storage (left), the compressor suction (top), and the recycle from the bypass valve (right). The sub-circuit model is a standard Modelica model of a close volume vessel from the Open Modelica library (Modelica-Fluid-Vessels-ClosedVolume).

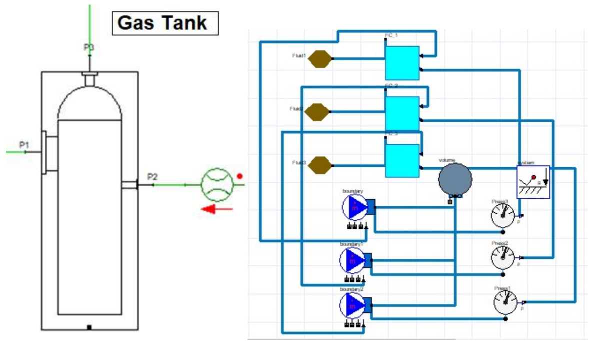


Figure 3. Macro-model and sub-circuit of scrubber tank.

2.3 Compressor

The compressor model uses an analytical pressure-flow relation,

$$\boldsymbol{Q = \eta \frac{Vol\,\omega}{2\pi} \Delta P} \tag{1}$$

Where Q is the flow rate, η is the volumetric efficiency, Vol is the displacement volume of the compressor, ω is the compressor speed, ΔP is the compressor pressure. η is fit to an analytical function based on a regression on the measured datasheet values. And can be therefore customized to any manufactured pump.

Based on the datasheet information, shown in Figure 4, the flow pressure relations are derived. In general, a multi-dimensional function vs pressure and speed can be generated, but based on the low variation vs speed, the average curve was used to calculate the flow vs pressure. The resulting equation used for η is shown in Figure 4.

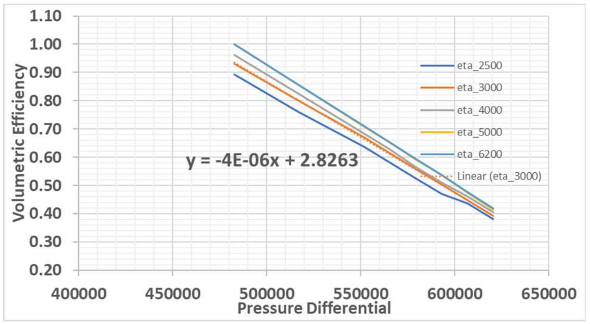


Figure 4. Datasheet values for compressor, shown as volume efficiency vs differential pressure. The different curves are for different compressor speeds.

The compressor sub-circuit then consists of an equation block that calculates the flow as a function of pressure and additionally calculates the compressor's flow torque according to:

$$\boldsymbol{\tau = \frac{Vol}{2\pi} \Delta P} \tag{2}$$

The resulting flow is applied to a flow source in series with the macro-model flow input and output ports, as seen in Figure 5. The torque is passed as a quantity out of the compressor macro-model for use with the Motor/Drive to calculate the electrical loading, described in the following section.

The compressor model also contains pressure relief valves and volumetric capacity to numerically stabilize the solution and account for some compressibility in the excess volumes and connectors.

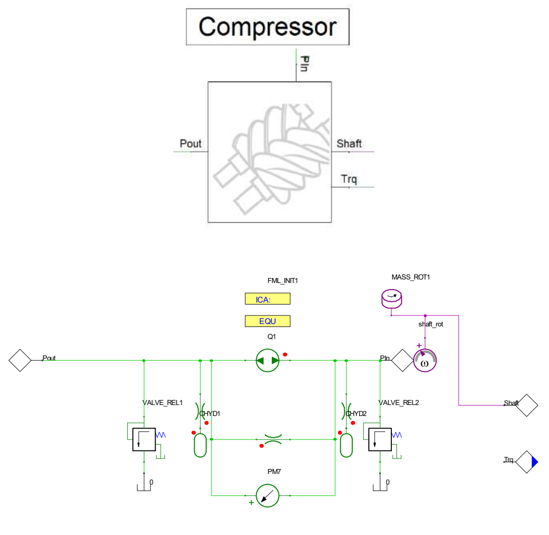


Figure 5. Macro-model and sub-circuit of compressor.

2.4 Motor/Drive

The motor and drive model in this system are a steady state representation of an induction machine with basic constant V/f drive. A standard TwinBuilder induction machine model was combined with a V/f drive, which was then parametrically analyzed over all possible input drive request speeds and required torques. Then the actual speed with slip and the resulting RMS current were saved to create a steady state table, as seen in Figure 6.

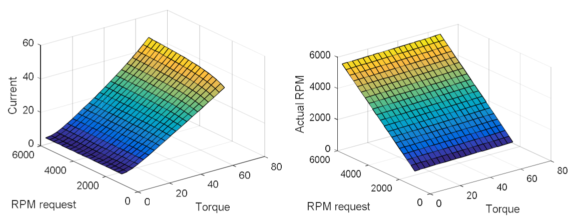


Figure 6. Steady state response surfaces for induction machine paired with V/f drive, given torque and RPM request inputs and current and actual RPM outputs.

The response surfaces are imported as CSV files into the TwinBuilder ND table element. Inside the motor/drive macro-model, the inputs of Torque and Speed are wired to input ports and outputs of RMS current and Speed are wired to output ports, as seen in Figure 7.

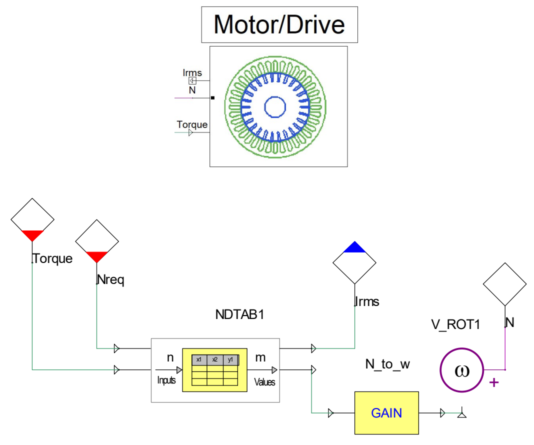


Figure 7. Macro-model and sub-circuit of electric motor and drive.

2.5 Bypass Valve

The bypass valve takes an input command from the controller and opens when in standby or low-pressure conditions and closes when pressurizing. As seen in Figure 8, the valve consists of a variable orifice component that fully opens and closes with a mechanical actuation of 0 to 10cm.

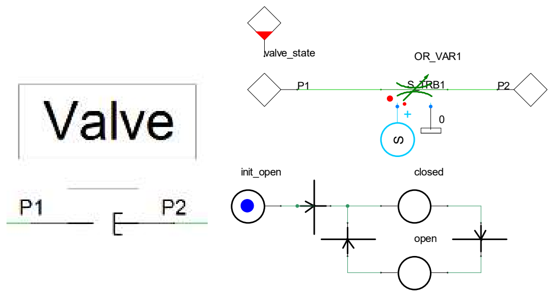


Figure 8. Macro-model and sub-circuit for bypass valve.

The actuation is described by a function, *S*, that transitions from on to off and off to on continuously (Funahashi, 1989),

$$S = \frac{cmd_{on}}{1+e^{(-\alpha(time-t_o)-t_d)}} + \left(\frac{1-cmd_{on}}{1} - \frac{1-cmd_{on}}{1+e^{(-\alpha(time-t_o)-t_d)}}\right) \quad (3)$$

Where cmd_{on} is the controller command, α is the switch rate, t_o is the activation time of cmd_{on}, and t_d is the delay from activation time until switching begins. In this case, α= 2000, and $t_d = 5\text{x}10^{-3}$s, as seen in Figure 9, with $t_o = 5\text{x}10^{-3}$s.

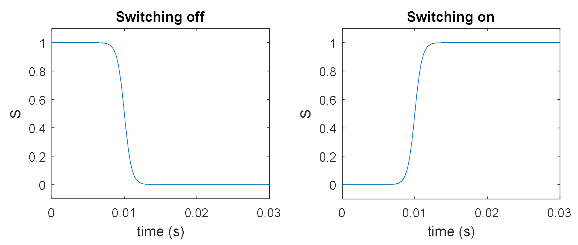
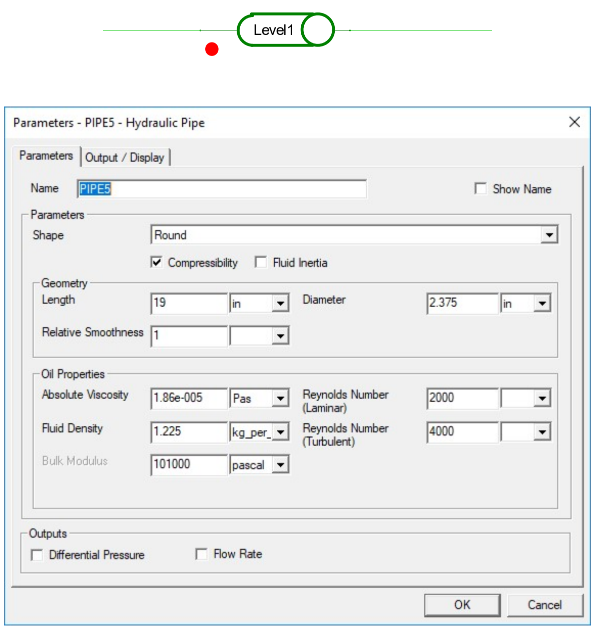


Figure 9. Switching function for opening and closing valve continuously when closing (left) and opening (right).

The state machine in Figure 8 sets t_o equal to the current time whenever it transitions from open to closed and vice versa. The transition event will occur whenever the controller sends the open or close command. This will create a 5ms delayed continuous transition from open to closed or closed to open. The continuity of the transition function and its derivatives helps with the stability and speed of the simulation.

2.6 Piping

The pipe models used to connect all components use standard TwinBuilder pipe elements. As seen in Figure 10, the pipe elements are characterized by the geometry and material properties of the flowing media. These are single phase media models only.



Name: PIPE5 — Show Name [ ]

Parameters
Shape: Round
[✓] Compressibility [ ] Fluid Inertia

Geometry
Length: 19 in — Diameter: 2.375 in
Relative Smoothness: 1

Oil Properties
Absolute Viscosity: 1.86e-005 Pas — Reynolds Number (Laminar): 2000
Fluid Density: 1.225 kg_per_ — Reynolds Number (Turbulent): 4000
Bulk Modulus: 101000 pascal

Outputs
[ ] Differential Pressure [ ] Flow Rate

Figure 10. The pipe element and component dialog box for parameter input.

They use a standard flow, pressure relation based on the pipe friction function (Haaland, 1983):

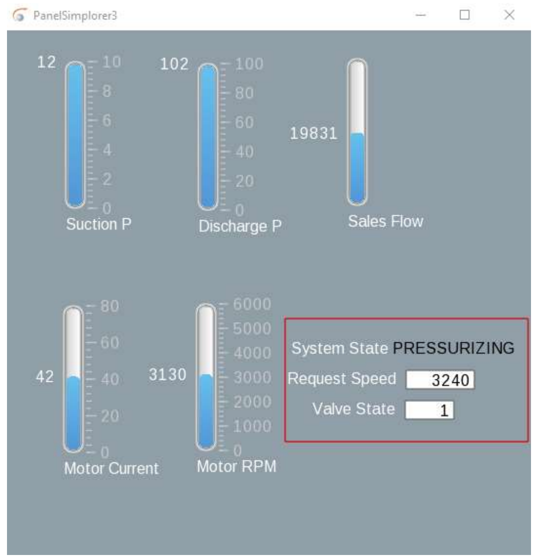
$$\boldsymbol{P} = \boldsymbol{f}\frac{\boldsymbol{l\rho V^2}}{\boldsymbol{2D}} \tag{4}$$

Where f is the friction function using Hagen-Poiseuille and Blasius laws in laminar and turbulent flow respectively, l is the pipe length, ρ is the density, V is the flow velocity, and D is the diameter.

2.7 Functional Mockup Interface (FMI) Controller and Human Machine Interface (HMI)

The controller and HMI use ANSYS SCADE (ANSYS, 2017) to write and compile the code into an FMU for use in TwinBuilder.

The HMI uses the standard rapid-prototyping editor to create the display shown in Figure 11. This display is used to monitor the motor current and speed, in addition to the pump input and output pressures together with the outlet line flow (sales flow). This HMI can also be used to manipulate the simulation, such as manually opening or closing valves, turning pumps on or off, or modifying control gains on the fly. The current HMI, shown in Figure 11, is used for monitoring only.



System State PRESSURIZING
Request Speed 3240
Valve State 1

Figure 11. HMI interface for monitoring the simulation states.

Similarly, the controller is also created in SCADE and imported into the TwinBuilder schematic as an FMU. The controller monitors the input and output pressures and regulates the bypass valve together with the compressor speed. The current state of the controller is reflected in the HMI.

3 Results

The results of a sweep on the source pressure are shown in Figure 12. The input pressure is swept from 1 psi to 17 psi. The pump remains off until the pressure passes a given threshold of 6 psi, at which point the outlet pressure rises to match the outlet flow line, then steps up the motor speed as the inlet pressure continues to rise. The outlet flow rises accordingly.

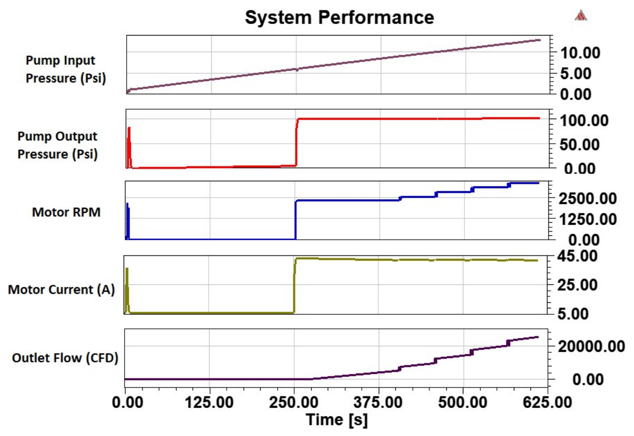


Figure 12. System results for swept input pressure.

4 Summary and Future Work

The current model can be used to validate controller operation for various input conditions. Interaction and display can be achieved using a human machine interface Future models can be expanded to include real time connection with pressure readings from the field. This can enable additional operational insight using the simulation based digital twin virtual sensors.

References

ANSYS, Inc, Canonsburg, PA. (2017). *TwinBuilder*. http://www.ansys.com.

ANSYS, Inc, Canonsburg, PA. (2017). *SCADE*. http://www.ansys.com.

S.E. Haaland. Simple and Explicit Formulas for the Friction Factor in Turbulent Pipe Flow. *J. Fluids Eng*, 105(1): 89-90, 1983. doi: 10.1115/1.3240948

K-I Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3): 183-192, 1989. doi: 10.1016/0893-6080(89)90003-8

[Industrial paper] A New Library for Modeling and Simulation of Pneumatic Systems

Maximilian Kormann[1] Dr. Clément Coïc[2] Guillaume Viry[3]

[1]Dassault Systèmes, Germany, maximilian.kormann@3ds.com
[2]Dassault Systèmes, Germany, clement.coic@3ds.com
[3]Dassault Systèmes, Germany, guillaume.viry@3ds.com

Abstract

The Pneumatic Systems library (PSL) by Dassault Systèmes is aimed at modeling pneumatic power systems. Typically, such systems involve actuators in industrial plants, pneumatic brakes or suspension systems, etc. Also, this library suitable for aerospace applications such as cooling or engine bleed air systems. The library deals with common problems modelling fluid flow in Modelica like accuracy in throttles or multi-sided connectors.

Keywords: pneumatics, heat transfer, fluid model, pneumatic power, pneumatic system, fluid flow

1 Introduction

In the following section, basic attributes of the library are given.

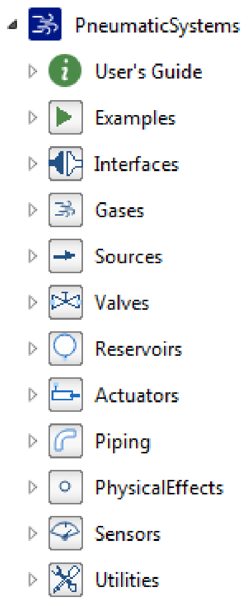

PneumaticSystems
User's Guide
Examples
Interfaces
Gases
Sources
Valves
Reservoirs
Actuators
Piping
PhysicalEffects
Sensors
Utilities

Figure 1. Library structure.

1.1 Library structure

An overview over the top level of the library is given in figure 1. To uncouple physics and their technical application, pneumatic systems' physical effects serve as the base class of the library object-orientation. These effects typically involve: capacitance, resistance, heat transfer and other power transformation (e.g. pneumatic to mechanical transformation).

Models of `Valves`, `Reservoirs`, `Actuators` and `Piping` instantiate or extend models from `Physical Effects` for technical application.

The `User's Guide` provides some tutorials based on `Examples` for getting started with the library.

1.2 Provided models

Different types of `Sources` define the origin of pneumatic power. Depending on the intended modeling depth, a simple pressure source can be used or the user can model a complex compression system utilizing the vane compressor model, for example.

`Valves` are used to control the fluid flow direction or imply a logic on the pneumatic system. `Actuators` transform pneumatic power into translational or rotational power. The icons of these components are compatible to (ISO 1219).

The most common example is a pneumatic cylinder. `Reservoirs` provide the possibility to store pneumatic power and/or apply heat transfer. `Piping` and `Restrictions` enable modeling the fluid transportation dynamics and losses.

2 Library features

In the following section, the advantages of the library will be outlined.

2.1 Fluid Model

Choosing the fluid model is done by instantiating a fluid model from the `Gases` package into the top level of the model. All deeper levels will access the fluid model by using the `outer` keyword. The fluid model is graphically represented in the top level as it can be seen in the lower left component in Figure 3.

The `Ideal Gas` model is suitable for most applications below temperatures of 500 K. Its gas equations are defined according to (White, 2016) and the dynamic viscosity is estimated by the law of (Sutherland, 1893). In order to improve the library compatibility and genericity an `ImportFromMSL` model is provided enabling to utilize any compressible fluid model from the Modelica Standard Library.

2.2 Symbols and Animation

The symbols in the icon layer of `Valves` and `Actuators` comply to ISO 1219. By default, the icons represent the current position of the valve or actuator during simulation to visualize the effects of a model. For complex models the animation can be deactivated to increase simulation performance.

2.3 Convenient computation of port properties

The variables in Table 1 have been chosen as properties in connectors. For enthalpy transport the stream concept (Franke et al., 2009) is being used. The templates for all PSL components (except multiple ports) contain a `GasProperties` class for each connector. By default, it is set to an empty class without computation of any values to increase performance. As the port variables can be insufficient for testing and debugging of a model, the choice is let to the user to replace `GasProperties` by other classes calculating total, static and/or critical (at sonic speed) values of the flow including density, temperature, viscosity, inner energy and more.

Table 1. Properties in a connector.

Formula sign	*Description*
$\dot{m}$	Mass flow through the connector
p	Absolute pressure in the connector
$h_{outflow}$	Specific enthalpy of the exiting flow

2.4 Discussion on kinetic energy

To estimate the mass flow through restrictions, most available formulas depend on the pressure ratio between downstream static pressure and upstream total pressure (Andersen, 1976). The calculation of static properties requires additional computational effort and may reduce the stability of the model. A slight inaccuracy is accepted for the benefit of improved simulation performance by setting `neglectKineticEnergy=true` by default for all components extending `PartialRestriction`. With this setting both total values for upstream and downstream pressures are being used to calculate the pressure ratio. If high accuracy is more important than computational speed, `neglectKineticEnergy=false` can be set.

2.5 Heat transfer

By default, all components are assumed to be adiabatic components. The first law of thermodynamics (Çengel and Boles, 1994) is applied without heat flow over the system border. Additionally, the user has two non-adiabatic options (see Figure 2): Environment heat transfer models heat losses of components due to their surface being exposed to environment temperature. External heat transfer provides a heat port for the component allowing the user to individually model the heat transfer of the component.

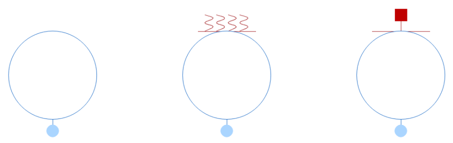


Figure 2. Adiabatic component, heat transfer to environment and to an external port.

3 Example

To describe the use of the library, an example will be given.

3.1 Description of the model

The example (see Figure 3) compares a pneumatic cylinder with cushioning to a cylinder without. Both cylinders are connected to a pressure source over a 3-2-Valve, which is actuated by a boolean pulse, to imply a cylinder movement. The rod of the cylinder is connected to a fixed spring damper system.

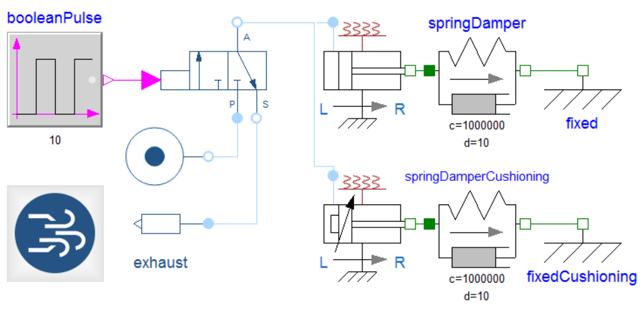


Figure 3. Example model comparing a cylinder with and without cushioning.

Cushioning means an end point damping for cylinders realized by the pneumatic throttle effect. When the distance between piston and end point is lower than a defined value, the area of the inlet restriction to the cylinder is reduced to throttle the fluid flow of the cylinder. As the cylinder movement is coupled to the fluid flow, this effect results in damping.

This example covers the main physical effects involved into pneumatics: capacitance as the gas volume inside the cylinder, restriction as the inlet flow restriction into the cylinder and transformation as the cylinder converts pneumatic into mechanic power.

3.2 Simulation results

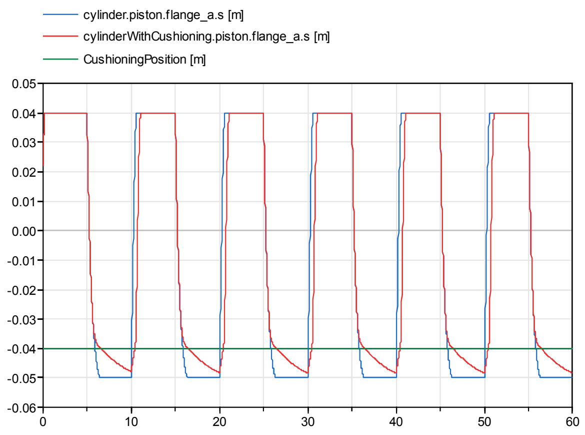


Figure 4. Comparison of the cylinder movement with and without cushioning.

As the example uses a one-sided cylinder, cushioning only happens on the left side of the cylinder actuation (see Figure 4). When the piston enters the cushioning zone (below the green line), the cylinder with cushioning (red plot) receives less mass flow rate from the inlet port and thus its speed decreases regarding the cylinder without cushioning (blue plot) and does not reach the same end position.

4 Conclusion

The Pneumatic Systems Library provides models for various applications of pneumatic power. The option to deactivate the neglecting of kinetic energy in orifices offers the possibility for more accurate simulations.

References

B. W. Andersen. *The Analysis and Design of Pneumatic Systems*. R.E. Krieger Pub. Co., University of Michigan, 1976. ISBN 1-575-24164-1.

Y.A. Çengel and M.A. Boles. *Thermodynamics: an engineering approach*. McGraw-Hill, 1994. ISBN 9780071132497.

R. Franke, F. Casella, M. Otter, M. Sielemann, H. Elmqvist, S. E. Mattson, and H. Olsson. Stream connectors – an extension of modelica for device-oriented modeling of convective transport phenomena. *Proceedings 7th Modelica Conference, Como, Italy*, pages 108–121, 2009. doi:10.3384/ecp09430078.

ISO 1219. Fluid power systems and components - graphical symbols and circuit diagrams, 2012.

W. Sutherland. The viscosity of gases and molecular force. *Philosophical Magazine 5*, pages 507–531, 1893.

Frank M. White. *Fluid Mechanics, Eighth Edition*. McGraw-Hill, 2016. ISBN 978-0-07-339827-3.

The DLR EtherCAT Library
A template based code-generation scheme for accessing real-time hardware from Modelica

Tobias Bellmann[1] Fabian Buse[1]

[1]Institute of System Dynamics and Control , German Aerospace Center (DLR), Germany,
{Tobias.Bellmann,Fabian.Buse}@dlr.de

Abstract

In this paper, a new concept to access real-time hardware from within Modelica via the EtherCAT bus is introduced and the implementation of a prototype library is demonstrated. The DLR EtherCAT library uses the open source EtherCAT library EtherLab to gather information about the connected bus slaves. Thereupon, the slave information is used in a code generation process to build native Modelica blocks providing the interfaces to their hardware counterparts. These blocks subsequently can be used to build real-time models, running on a Linux based real-time system and therefore controlling the hardware directly from the model. The application of the library is shown in a robotic testbed where a motor drive is controlled via EtherCAT.

Keywords: Real-time, EtherCAT, Code-Generation

1 Introduction

Using Modelica models in real-time embedded applications can be achieved via a multitude of interface technologies. In most cases, the Modelica model is compiled into C-code or FMU and integrated in a wrapping simulation software like e.g. Mathworks Simulink Real-Time. It is then executed on a dedicated real-time platform, e.g. vxWorks (Hofmann et al., 2015), dSPACE RT Hardware (Ritzer et al., 2016), xPC Target (Richard Kuchar and Andreas Klöckner, 2015), etc. providing the interfaces to the field devices like motor drives or sensors via an industrial bus system. However, the interfaces to the controlled hardware are in the domain of the simulator software, and can not be accessed directly via Modelica code. Furthermore, most of these solutions are costly industrial products, generating licence fees. With EtherCAT (The EtherCAT Technology Group, 2017), a real-time capable industrial bus is available, compatible to standard PC Ethernet components. It is possible to use open-source solutions like EtherLab (Florian Pose, 2013) or SOEM (Open EtherCAT Society), to communicate with EtherCAT field devices from a Linux PC with real-time kernel, achieving cycle times sufficient for many (control-) applications.
By integrating such open source solutions in a Modelica library, it becomes possible to communicate directly with the field devices from within Modelica models. The approach to integrate hardware interfaces directly into Modelica models and communicate with the hardware from within the Modelica Developer Tool is known from the Modelica *DeviceDrivers Library* (Thiele et al., 2017),(Bellmann, 2009). However, the Device Drivers library uses static interface models to communicate with the hardware.
In this new DLR EtherCAT library, we use a template based code-generation scheme to automatically generate the interface blocks from the EtherCAT slaves information, containing the also auto-generated communication interface C-Code. Generating Code from structured template files by replacing placeholders with often changing code is a helpful technique, applied in several Modelica projects (e.g. in (Nytsch Geusen et al., 2017)).

1.1 Basics of EtherCAT

EtherCAT is a field-bus communication protocol, defined in the IEC-Standard 61158 (International Electrotechnical Commision, 2014). It has been initially developed by Beckhoff and is an industry norm since 2005. The EtherCAT Master is the only participant in the EtherCAT Network, who sends data packages actively. The master can run on consumer PC hardware and uses a standard Ethernet media-access-card (MAC) to send and recieve the data packages. All EtherCAT slave controllers (ESC) only extract and insert their data at predefined locations in the data package, normally using a pure hardware implementation with an emphasis on short processing times. EtherCAT components make use of standard Ethernet cables and allow cable lengths up to 100 meters, whereas topologies as daisy-chain, star or trees are supported.

1.2 EtherCAT communication process

At boot time of the master, the topology of the bus is determined and information about the attached slaves is gathered. There are two possible types of communication between the master and its slaves: On the one hand, data can be exchanged with asynchronous datagrams (so called Service Data Objects, SDOs), addressing a slave by its position in the bus or a fixed address (which is defined by the master when connecting a new slave). These packages can be used for event-based communication. On the other hand, in cyclic operation, a so called process image is defined by the master, to be sent up and down the chain of slaves. It is up to the master to update and read the information in the process image. This can be done in different sample times for different data objects, collected in so-called domains. The data is organized in so called Process Data Objects (PDOs). PDOs represent the I/O channels of the attached bus hardware (e.g. a digital output or a motor controller's reference speed) and can be used to receive (Rx-PDO) or transmit data (Tx-PDO) from a device to the master. At the beginning of operations the accessible memory addresses for each slave are defined by the master.

1.3 The EtherLab EtherCAT Master

In this work, the open-source EtherCAT Master of the EtherLab package is used, however, the principles can as well be transferred to be used with other EtherCAT Masters. The EtherLab EtherCAT Master is installed under Linux as a kernel module and interacts with the MAC either via a generic interface or via direct memory access with modified drivers for chosen network card chipsets from Realtek and Intel. However, the generic interface does not have exclusive access to the network interface card and therefore can not be used in hard real-time environments. Nevertheless, in practice, sample times sufficient for control applications (1-2 kHz) can be achieved with the generic driver even under a "‘soft-realtime"’ low-latency kernel.
The EtherLab EtherCAT Master provides a set of command-line instructions, e.g. to print out XML formatted information about the connected bus slaves or active domains. It is also possible to write out the C interface code providing address information about the PDOs of the single slaves. The derived C code can then be used in the software in order to access memory pointer information for read-/write-operations. However, every time the connected EtherCAT bus is changed in composition or order, the memory addresses also change, resulting in the need for auto generated code on the user-software side. The master is available as open source licensed under the GPL Version 2, whereas the interface library is licensed under LGPL Version 2.1, allowing the linking of closed source components.

2 Library overview

The DLR EtherCAT library primarily consists of only a few blocks, as the variety of the hardware is then reflected by auto-generated Modelica libraries implementing the slaves' hardware interfaces. Figure 1 shows the most important library blocks.

The important function *generateConfiguration* starts the code-generation process, and takes the new plant library name and path as input. The *EtherCATMaster* block controls aspects as active EtherCAT domains and their I/O sample times, as well as the initialization and cleanup of the EtherCAT master. It is implemented as an inner/outer

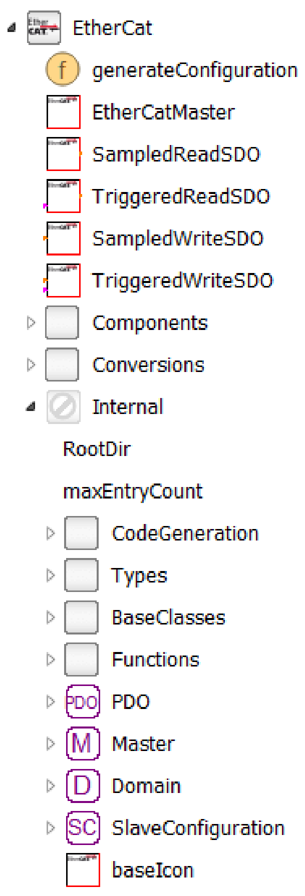
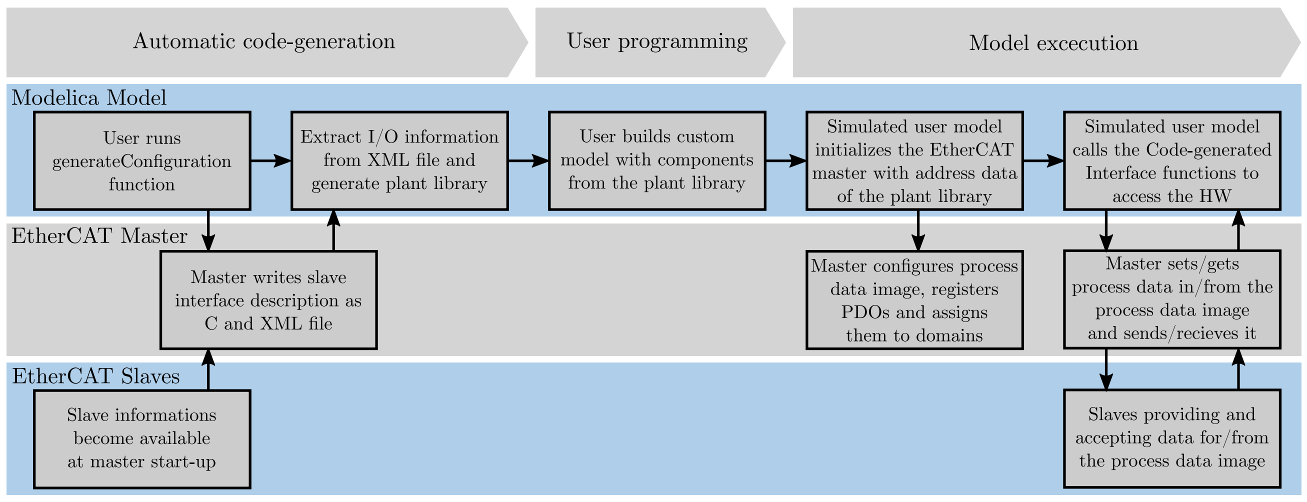


Figure 1. Overview of the available library blocks.

construct to allow the single slave interface blocks access to the master. Blocks to write and read Service Data Objects (SDOs) are provided as sampled and triggered version. In order to organize the slave data and to provide the building blocks for the code generation, several External Objects as the *Master*, *Domain*, *PDO* and *SlaveConfiguration* are available. These are not directly available to the library user but used as building blocks in the code-generation process.

3 The EtherCAT C/Modelica code-generation process in Modelica

The implementation of the EtherLab EtherCAT master requires several hard-coded address structures provided by the user program accessing the master. As they are changing every time the bus composition is modified, it would be cumbersome and error-prone to update the according Modelica and C Code fragments by hand. In the following section, the EtherLab communication procedure will be described in detail and the code-generation process providing the necessary Modelica and C code will be outlined. Figure 2 shows an overview of the code-gerneration and model execution process.



Automatic code-generation
User programming
Model excecution
Modelica Model
User runs generateConfiguration function
Extract I/O information from XML file and generate plant library
User builds custom model with components from the plant library
Simulated user model initializes the EtherCAT master with address data of the plant library
Simulated user model calls the Code-generated Interface functions to access the HW
EtherCAT Master
Master writes slave interface description as C and XML file
Master configures process data image, registers PDOs and assigns them to domains
Master sets/gets process data in/from the process data image and sends/recieves it
EtherCAT Slaves
Slave informations become available at master start-up
Slaves providing and accepting data for/from the process data image

Figure 2. Overview of the code-gerneration and model execution process.

3.1 Communication procedure with EtherLab

Every user program using the EtherLab EtherCAT master has to follow this procedure to establish the communication cycle with the attached slaves:

1. Initialize the EtherCAT master

2. Register at least one domain (to organize the slaves' PDOs)

3. Retrieve configuration for slave *1...n* from the master

4. Configure the PDOs of slave *1...n* using their memory address image (provided by the master as C-Struct)

5. Register the single PDOs of slave *1...n* and assign them to their domain

6. Start the communication cycle with the slaves by activating the master

7. Periodically read and write from and to the PDOs and send/receive the process image

Once the communication is running, the user program has to produce and consume the data in real-time.

3.2 Using Modelica's External Objects to organize the master/slave data

In order to organize all the necessary information to set up and run the communication cycle, several External Objects are used to store for example data and pointers used by EtherLab interface. External Objects enable the user to control the use of C-Source Code with a guaranteed execution in a constructor and destructor routine. This is the standard way to administrate the allocation and freeing of external resources as memory or hardware in C-code used by Modelica models, as the constructor and destructor are guaranteed to be called pairwise during the simulation. Normally the constructor is called during the initialization of the External Object and the destructor is called at the end of the simulation run, cleaning up used resources.

In Listing 1, the implementation of the master external object is shown as an example for the typical hardware initialization and cleanup process.

In total, four different External Objects are used to handle the process data and pointers from the EtherLab API (see Figure 3). The *Master* class handles the initialization of the EtherLab EtherCAT master. The *Domain* External Object is used to register a domain, and can be provided as input for the PDOs collected in that domain. The *SlaveConfiguration* External Object makes the information about the slave available for the EtherCAT master, e.g. the bus position of the slave, vendor data and hardware name. Every slave on the bus has to be registered with the master in order to provide it with the overall bus topology. Additionally, a record, *SlaveConfiguration*, is used to store this general information about the slave in Modelica. The *PDO* External Object holds the information about the PDO address in the process image (index and subindex) as well as its corresponding domain. Every PDO of every slave has to be registered with the master to be accessed periodically, this is performed in the constructor of the PDO class.

3.3 Generating the plant library

By executing the *generateConfiguration* function of the library, the user starts the code-generation process, as described in Table 1. The function subsequently calls the master's shell commands to write out the respective bus interface control document (ICD) as XML-based file and the process image memory addresses as C header file. The *generateConfiguration* function uses standard Mod-

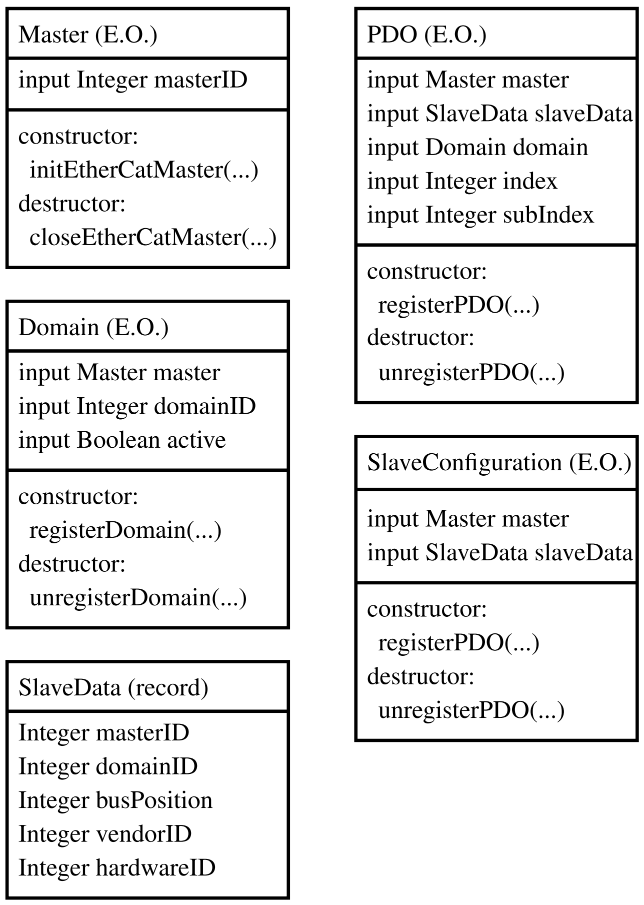


Figure 3. External Objects and Data structures, used by the EtherCAT library and the generated plant library.

Listing 1. The External Object EtherCAT.Internal.Master

```
class Master "External Object handling the
   ethercat master creation"

  extends ExternalObject;

  function constructor
    import EtherCAT;
    import EtherCAT.Internal.Master;
    input Integer masterID "ID of the
        master, normally it should be 0";
    output Master master;
  external"C" master =
      EtherCAT_initEtherCATMaster(masterID)
      ;
    annotation (...);
  end constructor;

  function destructor
    import EtherCAT;
    import EtherCAT.Internal.Master;
    input Master master;
    external "C"
        EtherCAT_closeEtherCATMaster(master
        );
  end destructor;

end Master;
```

elica String functions to parse the XML file. With this information, code-fragments for every slave are generated individually using the External Objects described in the last section. These code blocks, for example the input/output interface definitions and their according PDO objects are then inserted in a slave model template file. In this template file (Listing 2), placeholders in the format *$(IDENTIFIER)* are replaced with the code generated by the *generateConfiguration* function. At the end of the code-generation process, the new plant library is written out as file at the user-defined disk location and loaded into the Modelica editor using the vendor specific API function.

Table 1. The code-generation process.

Code-generation steps
`Write slave information as XML-file and C-interface code to disk`
`Parse XML file for number of slaves attached to the bus`
For each slave:
`Parse XML file for vendor ID, product code, number of Rx- and Tx-PDOs`
For each PDO:
`Extract PDO's sub-index, bit length, data type and name`
`Generate a Modelica External Object definition to handle the PDO`
`Generate Modelica code accessing the PDO`
`Generate Modelica code defining the inputs and outputs`
`Insert the code generated parts into a prototype file`
`Save modified prototype file as new Modelica model of the slave`

An example of a generated plant library can be seen in Figure 4. In this plant configuration, an ELMO motor drive (slave 0), a Beckhoff EtherCAT Coupler (slave 1), a Beckhoff 8-channel digital input terminal (slave 2), a Beckhoff 8-channel digital output terminal (slave 3), two Beckhoff 8-channel analog input terminals (slaves 4 & 5) as well as two master/slave terminals for CANopen (slaves 6 & 7) are attached to the EtherCAT master. In this example, the slaves 1, 6 and 7 have no input or output connectors; slave

1 is a passive bus coupler with no physical inputs or outputs besides its connectors to other rail attached Ethercat slaves. Slave 6 and 7 are CANOpen Adapters and do not communicate via PDOs but with the CAN over EtherCAT protocol, which is not supported by this library yet.

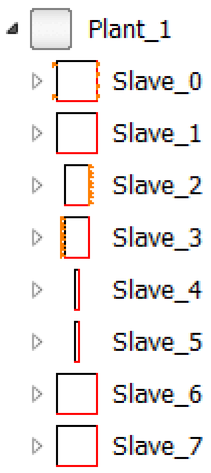

Plant_1
Slave_0
Slave_1
Slave_2
Slave_3
Slave_4
Slave_5
Slave_6
Slave_7

Figure 4. Example for a code-generated plant library.

4 Usage of generated blocks

Figure 5 shows a simple example model using the slave interface blocks from the bus configuration in Figure 4. In this example, the ELMO motor drive receives its reference torque signal as integer value from the block *torqueSource*. A state machine provides the values for the initialization of the motor drive via the block *startUpStates*. The digital input and output terminals EL1018 and EL2008 are also easily accessed via integer values. In order to synchronize the model with real-time the *SynchronizeRealtime* block from the Modelica Device Drivers library is used. This block also changes the priority level of the simulation process to "'real-time"' to avoid process interruption by other processes.

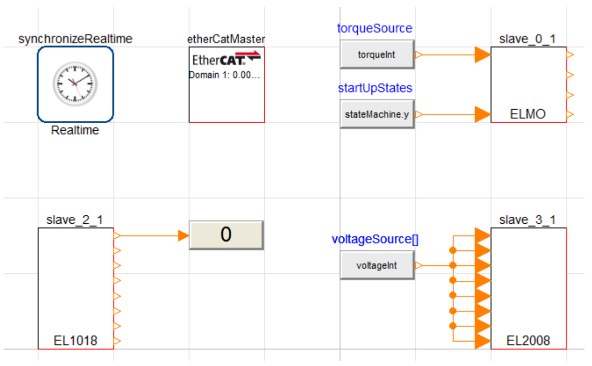


Figure 5. Example model, communicating to external systems via EtherCAT.

5 Application example: The TROLL terramechanics testbed

The Terramechanics Robotics Locomotion Lab (TROLL) is a novel testbed with the goal of automated terramechanics testing (see Figure 6) for planetary rover applications. Its main components are an ingress protected industrial robot, a force torque sensor and a ELMO motor controller driving the wheel drive unit. Additional application specific sensors can be mounted and used if needed. To unify the communication setup EtherCAT has been chosen. The most important component that is currently connected with EtherCAT is the ELMO motor controller of the drive unit. The control architecture of the TROLL

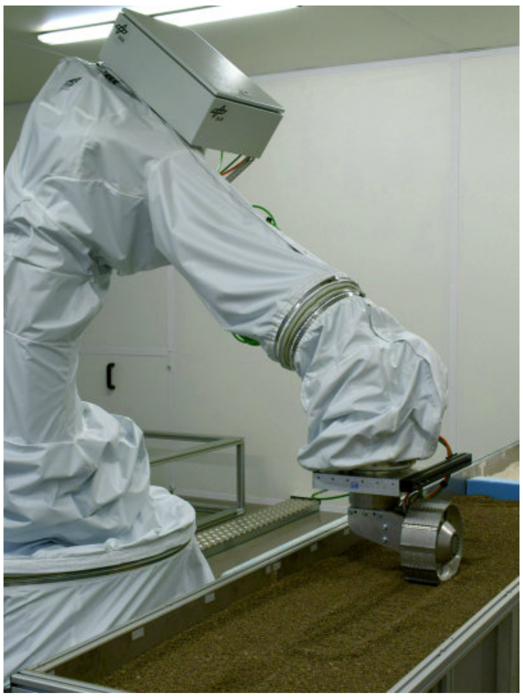


Figure 6. Side view of the TROLL showing the robot, force torque sensor, drive unit with wheel and a soil bin filled with lava sand.

uses a combination of conventional robot programming and a Modelica model running on a real-time Linux system (Linux Ubuntu 14.04, kernel-3.16.077 low-latency). The Modelica model manages the process control, unifies the communication between the different sensors via the DLR EtherCAT Library and is directly driving the robot during experiments. All auxiliary motion, like moving to start position or similar are taught to the robot and can be executed on command. The DLR EtherCAT Library is necessary in this context to build a plant library of the connected elements. The generated models within the plant library are then used where needed within the Troll Control model. More complex elements, like the ELMO are embedded within an interface model to enable comfortable standalone use of the component. In case of the ELMO controller interface, conversions for the input and output values envelop the code-generated ELMO EtherCAT block, as well as a Modelica state machine setting up the internal state machine of the controller (see Figure 7).

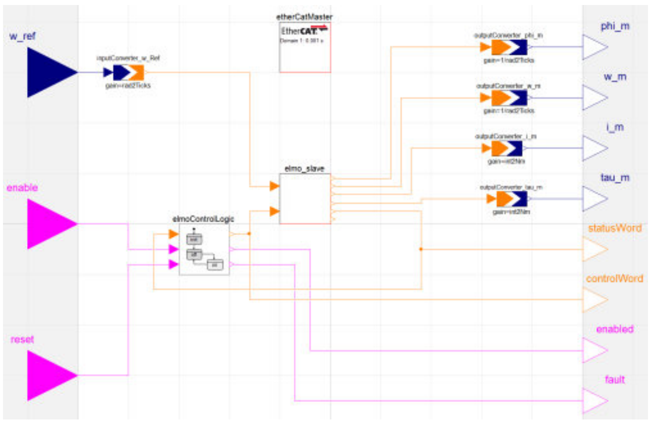


Figure 7. ELMO interface model that is generated by the Library with auxiliary elements to generate a standalone model with automated start-up secquence.

To initialize the controller correctly, the ELMO slave is configured by sending SDO commands from the Modelica state-machine during start-up of the simulation. The model and the EtherCAT communication are executed at a sample rate of 1000 Hz, whereas the robot controller is provided with data every four simulation steps at a sample rate of 250 Hz. The model is synchronized with the robot using a blocking network call, waiting for the input data from the robot. In between these synchronization cycles the *SynchronizeRealtime* block from the Modelica Device Drivers library is used to adjust the simulation rate for the EtherCAT communication. This setup allows the simultaneous communication with the robot and motor controller.

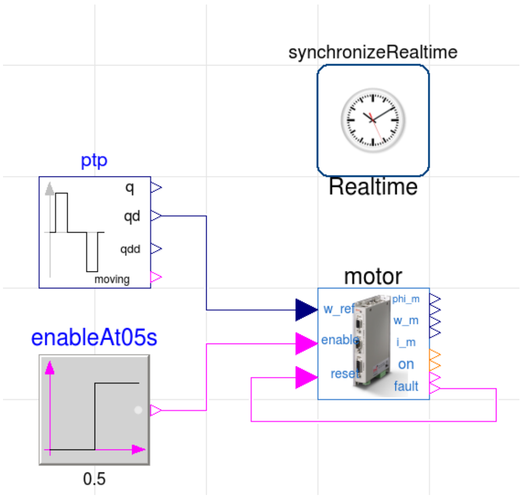


Figure 8. Simple Modelica model, generating a PTP movement for the ELMO motor controller from Figure 7.

In Figure 8 a simple Modelica test model is depicted showing a PTP source as generator for a reference velocity. The reference velocity is sent to the ELMO EtherCAT slave via the DLR EtherCAT library, driving a synchronous motor with a rover wheel attached (no ground contact). The resulting motor speeds and currents are shown in Figure 9, demonstrating that the velocity controller on the ELMO controller is working properly. The model is run directly from Dymola under Linux Ubuntu 14.04 (kernel-3.16.077 low-latency).

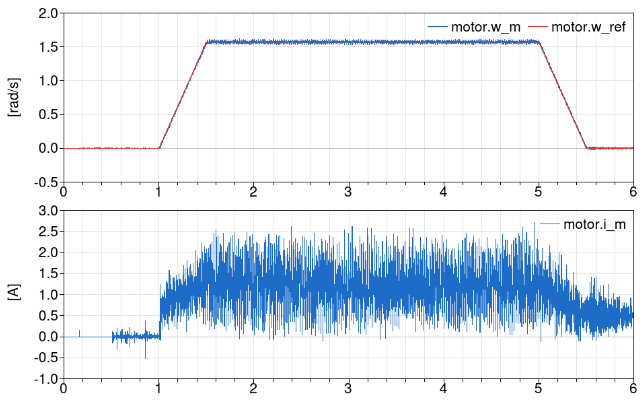


Figure 9. Motor speed (w_m) and current (i_m) from the ELMO controller as well as reference speed (w_ref) from the Modelica model. Motor is controlled in real-time with 1000 Hz sample rate from within Dymola under Linux Ubuntu 14.04 (kernel-3.16.077 low-latency).

6 Conclusion

In this paper, a new method to control EtherCAT based hardware in real-time and directly from within Modelica models has been shown. The DLR EtherCAT library enables the user to easily read out the EtherCAT bus configuration and to auto-generate interface code for the connected EtherCAT slaves. The generated interface blocks are integrated by the user to form models controlling and reacting to EtherCAT components. No additional licences are necessary, as the EtherCAT master from EtherLab is an open source project. The next steps in development should focus on including support for CAN over EtherCAT to control CAN-bus hardware attached via a CAN/EtherCAT coupler. For now, the library remains an internal DLR tool, but in the future a release as part of the Modelica *DeviceDrivers* library or as a commercial library is planned.

Acknowledgments

The authors would like to thank the Ingenieurgemeinschaft IgH for creation and maintaining the open-source EtherCAT master EtherLab, as well as Dipl.-Ing. Johann Heindl for providing the test motor for the library development.

Listing 2. The template for slave models

```
model $(MODELNAME)
 extends EtherCatSlave(
  slaveData(
   domainID=domainID,
   busPosition=$(BUSPOS),
   vendorID=$(VENDORID),
   hardwareID=$(HWID)));
 parameter Integer domainID=1
  "Id of PDO domain (1..5)";
 ...
protected
 outer EtherCatMaster etherCatMaster;
 SlaveConfiguration slaveConfiguration=
  SlaveConfiguration(etherCatMaster.master,
  slaveData);
 Integer ret=configurePDOs(
     slaveConfiguration);
 String hwName = "$(NAME)" "Name of Slave
     hardware";
 //PDO Definitions:
 $(PDODEF)
public
 //Inputs and Outputs:
 $(IODEF)
equation
 when etherCatMaster.sampled[domainID] and
     etherCatMaster.ddo[domainID] then
  $(TXRXCALLS)
 end when;
 annotation (...);
end $(MODELNAME);
```

References

Tobias Bellmann. Interactive simulations and advanced visualization with modelica. In *Proceedings of the 7th International Modelica Conference; Como; Italy; 20-22 September 2009*, number 043, pages 541–550. Linköping University Electronic Press, 2009.

Florian Pose. IgH Master 1.5. 0 Documentation. *Ingenieurgemeinschaft IgH*, 2013.

Andreas Hofmann, Nils Menager, Issam Belhaj, and Lars Mikelsons. Integrated Engineering based on Modelica. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 893–901. Linköping University Electronic Press, Linköpings universitet, 2015.

International Electrotechnical Commision. *Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series.* 5 2014. ISBN 9782832216309.

Christoph Nytsch Geusen, Alexander Inderfurth, Werne Kaul, Katharina Mucha, Jörg Rädler, Matthis Thorade, and Carles Ribas Tugores. Template based code generation of Modelica building energy simulation models. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 199–207. Linköping University Electronic Press, 2017.

Open EtherCAT Society. SOEM Reference manual. URL `https://openethercatsociety.github.io/`.

Richard Kuchar and Andreas Klöckner. Automatic flight code generation from multi-physics models. In *ODAS 2015*, 2015. URL `http://elib.dlr.de/100124/`.

Peter Ritzer, Michael Panzirsch, and Jonathan Brembeck. Robotisch bewegt-Interaktive Bewegungssimulation. *dSPACE Magazin*, (1/2016):52–57, 2016.

The EtherCAT Technology Group. EtherCAT - the Ethernet Fieldbus. Online, 7 2017. URL `https://www.ethercat.org/download/documents/ETG_Brochure_EN.pdf`.

Bernhard Thiele, Thomas Beutlich, Volker Waurich, Martin Sjölund, and Tobias Bellmann. Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 713–723. Linköping University Electronic Press, 2017.

Modelling and Development of a Pseudo-Hydraulic Power Steering Model for use in Real-Time Applications

Theodor Ensbury[1] Peter Harman[2] Mike Dempsey[1]

[1]Claytex Services Ltd. Edmund House, Rugby Road, Leamington Spa, CV32 6EL, UK
{theodor.ensbury, mike.dempsey}@claytex.com
[2]CAE Tech Limited, Leamington Spa, UK, peter.harman@cae.tech

Abstract

Driver-in-the-loop (DiL) simulation is playing an increasing role in automotive OEM development processes. Vehicle models used in these activities therefore need to be as accurate, and realistic, as possible. This paper will present the modelling and development of a pseudo-hydraulic power steering model, designed for usage in DiL applications. Specific focus during development has been towards the quantification and analysis of the torque feedback from the steering model to the simulator rig steering wheel, to produce as realistic a steering 'feel' as possible. Metrics derived from physical testing of vehicle steering systems have been deployed to analyze the torque feedback of the steering system. Subsequent assessment of the steering model and specific parameterization has been used to inform the model parameters utilized. Results quantifying the performance of the steering model during full vehicle testing using the Claytex VeSyMA platform are presented.

Keywords: Driver-in-the-loop, DiL, steering feel, hydraulic power steering

1 Introduction

As real-time Driver-in-the-loop (DiL) vehicle simulation has increased in deployment across the automotive industry, the need for accurate models correctly depicting qualitative aspects of vehicle behavior, has increased. Vehicle steering, as one qualitive aspect of vehicle experience, should be as close to real life as possible for driver immersion (Ansible Motion, 2015). To understand why, the role steering torque feedback plays in the driver/vehicle interaction must be considered.

Despite the recent trend of OEMS moving to fully electric systems, some, such as Nissan, state preference for using hydraulic assistance systems based on a perception of superior feel. This has led to the development of hybrid electro-hydraulic systems, with an electric pump assisting the steering through hydraulic fluid (Nissan, 2018). Thus, it is important to model the dynamics of the hydraulic assistance system to capture the steering feel. Therefore, the object of this paper is not to present a model of an electro-hydraulic steering system, but rather to present a pseudo-physical steering model accounting for the dynamics of hydraulic assistance, such as the decay rate of force when the torque applied drops off suddenly, without the numerical complexity of a fully physical system model.

1.1 Impact of steering feel in DiL applications

When driving within a DiL environment, the driver senses what the vehicle model is doing through several senses; one of these is haptic (touch). This concerns the human/physical interface of which the steering forms a part (Ansible Motion, 2015). Essentially, the steering in a DiL simulator is a haptic feedback device. As a primary feedback on vehicle behavior, correct steering feel is therefore important to enable the driver to control the vehicle in as realistic method as possible. Dynamic limit control of the vehicle by the driver is therefore impacted by the accuracy of the steering feel. This applies to DiL simulation in both handling studies and driver/vehicle control system interaction, such as electronic stability program (ESP) development. For ESPs to be correctly developed, the driver must thus react as realistically as possible given the confines of a simulation environment (Ansible Motion, 2015). Accuracy of the steering model must therefore be retained in both normal driving condition, characterized by on-center driving (Pfeffer et al. 2008), as well as off-center scenarios.

1.2 Real-time modelling constraints

As the target application of the steering model will be in DiL simulation scenarios, the steering model utilized must therefore be fast enough to run in real-time. By nature, hydraulic power steering torque feedback displays a non-linear hysteretic characteristic. Previous implementations of steering models in detailed real-time scenarios have required the use of external models devoted towards modelling the steering (Andreasson, et al. 2016). Whilst this is one solution to capturing the correct characteristics in a computationally efficient manor, a steering model integral to the vehicle model is a preferable solution with regards ease of use by the model user.

The vehicle model utilized to test, develop and parameterize the steering model must also be of required quality and run fast enough for smooth real-time running. As the steering feel and vehicle response to

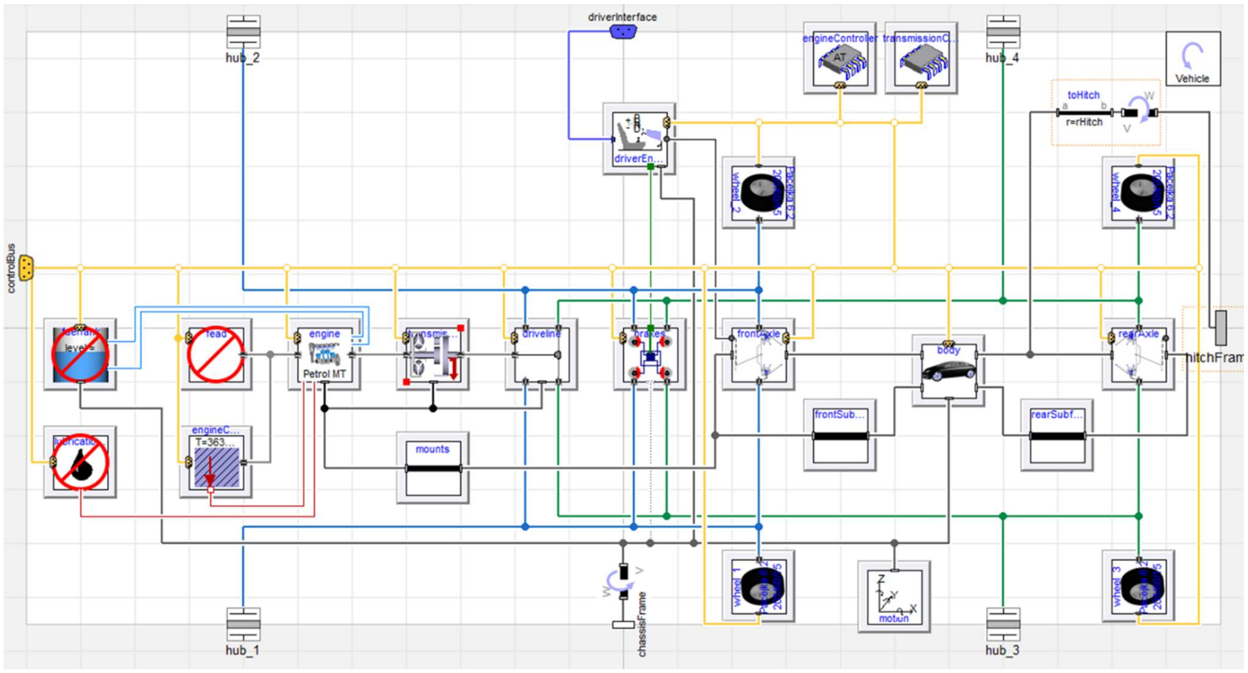


Figure 1. The vehicle model built to test and develop the Pseudo-Hydraulic power steering model. Built from a template from the VeSyMA library, this model features subsystems and components from the VeSyMA, VeSyMA - Suspensions libraries. Note, steering wheel connection is handled acausaly through the driver interface.

steering input are interlinked (Pfeffer et al. 2008), it is paramount the vehicle and tire model are of sufficient fidelity with respect to the analysis for the study to be valid. Due to the consideration in this paper of the transient response of the steering/vehicle, a transient, non-linear vehicle and tire model is required. The object-oriented nature of Modelica enables the vehicle model to be tailored to the application, therefore simplified systems for vehicle aspects such as the powertrain can be deployed.

2 Modelling

The steering model described in this paper, as well as the vehicle model and vehicle test used in the development and analysis of the steering model form part of the Claytex VeSyMA suite of libraries.

2.1 Overview of Claytex VeSyMA suite

Developed for use with the Dymola simulation environment, the Vehicle Systems Modelling and Analysis (VeSyMA) suite is a complete vehicle simulation solution built from Modelica. Utilizing the object-oriented nature of the Modelica language, the principle of the VeSyMA suite is several separate subject specific libraries all of which are compatible with one another, as they share a common parent, the VeSyMA library, which defines vehicle level templates. As the open source Vehicle Interfaces library provides the basis of the templates in the VeSyMA library, third party models can be capable of being compatible with VeSyMA vehicle models.

On its own, the VeSyMA library is capable of straight line vehicle analysis; each subsequent subject specific library adds in fidelity and complexity to the overall model. Examples of subject libraries include the VeSyMA – Suspensions library, which defines 3D multibody suspension systems, road models and high-fidelity tire models, or the VeSyMA – Engines library which defines various high-fidelity powertrain components. A schematic overview of the VeSyMA suite is presented in Figure 2.

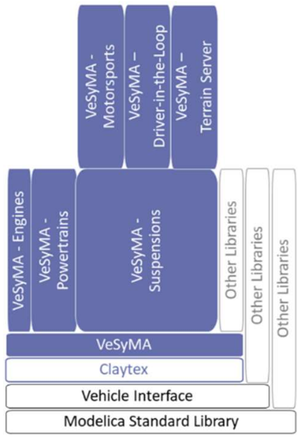


Figure 2. Schematic overview of the VeSyMA suite of libraries from Clayex. Note, the VeSyMA - Suspensions, VeSyMA - Driver-in-the-Loop and VeSyMA libraries were utilized in the study this paper describes.

2.2 Vehicle model

A complete multibody vehicle model utilized to test and develop the pseudo-hydraulic power steering model, free to move within the simulation environment in all 6 degrees (lateral, longitudinal, heave; roll, pitch and yaw) is presented as Figure 1. All elements within the vehicle model were taken from the VeSyMA or VeSyMA – Suspensions libraries. The vehicle platform employed was loosely based upon a passenger car, specifically the executive/E class. Double wishbone suspension was used for both the front and rear axles of the vehicle, with drive torque sent to the rear wheels as part of a standard rear-wheel drive (RWD) layout. No electronic stability programs, active yaw control or other electronic driver aids were used in the vehicle.

The vehicle model is required to be of sufficient fidelity, whilst simultaneously being as computationally efficient as possible to be capable of real-time DiL running. Compromises were therefore made regarding the fidelity of non-essential components. Specifically, idealized powertrain components were used, including:

- Mapped engine model
- Idealized paddle shift transmission model
- No lubrication or FEAD systems
- 1D rotational driveline model

Compliances within the suspension mounting points were also omitted and a simplified aerodynamic model detailing the lift, side and drag forces used. A four-wheel disc brake model with a controlled elasto-plastic friction model (Dankowicz, 1999) was also used.

Priority was given to the fidelity of the suspensions and tire models employed within the vehicle model. Full multibody suspension linkages were used, featuring Claytex aggregate joints in place of standard spherical joints; aggregate joints are computationally more efficient than standard multibody joints, developed specifically for real-time applications. The block diagram of the suspension linkage model used is presented as Figure 3.

Each link within the suspension model included specific mass and inertial properties, but linkage mounting compliance or suspensions member flex was omitted. Translation force elements were deployed as the ride springs and dampers, connected directly to the lower control arm. Full multibody anti-roll bars were also utilized on both the front and rear axles.

A fully combined lateral and longitudinal slip tire model was also used, therefore the lateral and longitudinal forces produced by the tires are non-linear. This comprised of a Modelica implementation of the Pacejka Magic Formula (corresponding to MF6.2). Included were asymmetric tire behavior due conicity and ply-steer; a kelvin spring damper modelled the vertical dynamics of the tire (Pacejka, 2012).

An explicit Runga-Kutta time integration method with a fixed time step of 1 m/s was utilized for both the steering model study and real-time running. Approximately 100 time-states were present in the translated vehicle model.

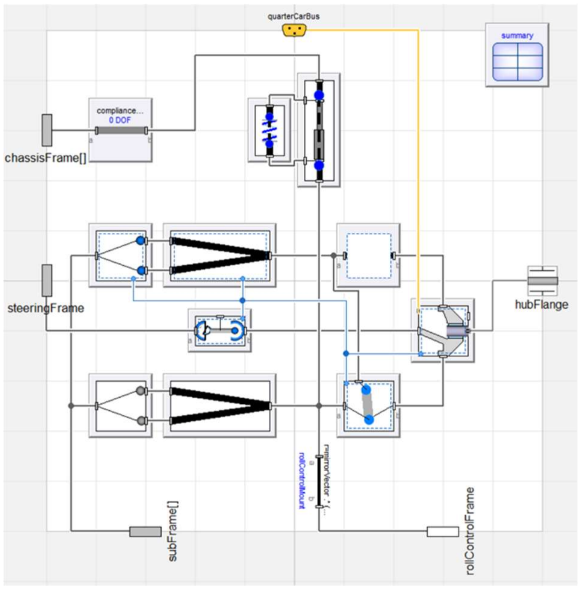


Figure 3. Double wishbone Quarter Car linkage model deployed once per wheel in the frontAxle model. Note the use of the aggreagted joints in place of spherical joints.

2.3 Steering model concept

Pfeffer et al. (2008) presented a steering model designed to capture the correct feel of the vehicle steering during on-center driving, due to the dominance of this range in the experience of daily road drivers. Lateral dynamics up to 4 ms^{-1} were deemed to be of interest in their study. Validation of the model presented was achieved against vehicle measurements from testing a BMW E46 with a hydraulic power steering system; therefore, this steering model presents a robust basis for steering model development. Furthermore, computational concerns were considered during modeling by Pfeffer et al. (2008) further increasing the suitability of the model to the application described in this paper.

The basic premise of the Pfeffer et al. (2008) model is to split the steering system into two sections; mechanical and hydraulic. Work on the hydraulic element of the steering model presented in this paper is presented in section 2.4. Comprising the mechanical elements of the steering model are the column and the rack with pinion. Thus, the mechanical system has 2 degrees of freedom, one rotational (column) and one translational (rack). Inertias for the two mechanical elements are considered separate, with the compliance of the steering system modelling as part of the column. Figure 4 presents the holistic steering model developed for this paper. Of note is the lack of upper column and steering wheel inertia shown in Figure 4; this is present and modelled separately (but rigidly connected) to the steering model shown during offline simulations, but it is replaced with the physical column and wheel in simulator rigs. Inertia below the column compliance is modelled in this steering model. Power assistance is applied to the steering system at the rack in the Pfeffer et al. (2008) model.

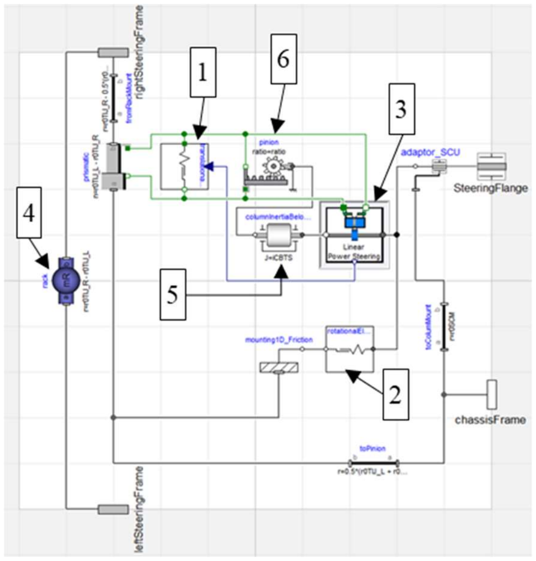


Figure 4. Steering model developed for this paper, deployed as part of the frontAxle model. Section 2.4 details the Linear Power Steering block, featuring the pseudo-hydraulic model. Notes: 1 is the rack friction model, 2 is the column friction model, 3 is the pseudo-hydraulic power assistance model, 4 is the rack mass, 5 is the column inertia below the compliance, 6 is the pinion/rack model. The upper column inertia is modelled externally to this model, rigidly connected to the steeringFlange.

As during physical operation both the column and the rack move within their respective housing, they cannot be considered ideal in motion. Therefore, friction has been included by separate models for both the column and the rack. Friction losses between the pinion gear and the rack are considered as part of the rack friction model. Pfeffer et al. (2008) utilized Exponential-Spring-Friction-Elements (ESF-Element) in their work to model the friction effects on the column and rack. A key aspect of the ESF-Element friction models is the dynamic state behavior. This is to capture the hysteretic characteristic required, resulting from the differing sliding and pre-sliding aspects of dynamic and static friction present within a steering system.

However, a changing state behavior renders them unsuitable for real-time simulation, due to the computational penalty associated with state events during time integration. A single-state friction model must therefore be utilized to avoid introducing events. Dupont et al. (2002). presented an 'elasto-plastic' single state friction model suitable for use in real-time applications. Termed 'elasto-plastic', the model developed by Dupont et al. (2002) differs from standard single state friction models (such as Dahl or LuGre). In this model, the presiding behavior is modeled in an elastic method initially before transitioning into plastic behavior, hence the term 'elasto-plastic' being coined. As the elastic component is reversible (whereas the plastic is irreversible), it is argued the total drift (deemed "spurious") of the elasto-plastic friction during pre-sliding is greatly reduced compared to other single state models, which rely solely on plastic behavior during pre-sliding. It is stated that the elasto-plastic model is therefore valid in periodic conditions dominated by pre-sliding; conditions a vehicle steering system experiences, as proved by the hysteretic behavior displayed during on-center driving.

2.4 Pseudo-hydraulic power assistance model

The model relating the steering column dynamics to the assistance force is termed 'pseudo-hydraulic', as it aims to capture the key dynamics of a hydraulic power steering but without physical modelling of the internal elements of that system.

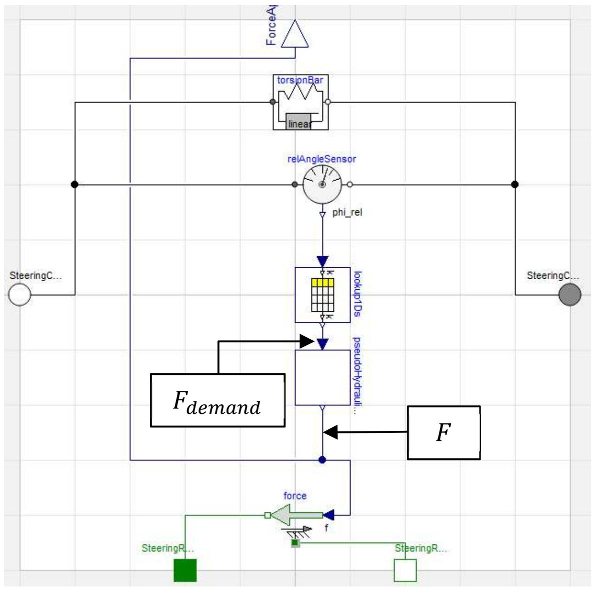


Figure 5. Inside the Linear Power Steering block. The rotational flanges of the steering-column are connected by the torsion-bar. The relative angle is used to calculate the assistance force demand, with the assistance force generated by the pseudo-hydraulic block.

The assistance force from a hydraulic power steering system is applied by a linear actuator with two chambers. The force is calculated from the pressure difference in the chambers. This actuator is connected to a spool valve, which can be either linear or rotary, that connects the chambers to either a high supply pressure from the pump or a low return pressure to tank. The steering column contains a torsion-bar giving compliance between handwheel angle and the rack. It is the relative angle across this torsion-bar that opens and closes the spool. In Figure 5, we have connected a relative angle sensor to a lookup table, which contains the steady-state assistance force demand for the torsion-bar twist. Assistance force is then produced by the pseudo-hydraulic block relative to the assistance demand.

If we neglect non-linearities due to fluid viscosity and compressibility effects, the rate-of-change of the fluid pressure is proportional to the square-root of the pressure-drop across the spool. The effect of the profile of the spool is incorporated in the steady-state assistance force table, and hence our model only needs to calculate an actual assistance force based on a demanded assistance force. A typical power-steering valve comprises a rotary spool, with a bevel profile that smooths the transition of opening area. Identification and modelling of spool geometry is described in detail by Rösth (2007).

Critical to the steering feel is the decay in assistance force. This differs from the rate of increase due to the internal flow areas in the hydraulic system. We therefore define 2 parameters for a minimal model of hydraulic behavior.

- rate [1/s]: The gain from the force demand to rate-of-change of force from one side of the actuator
- emptyingFactor [1]: The factor of emptying rate vs filling rate, 1 means equal, 2 means twice as fast to empty (and force to decay)

The variables involved in the model are divided into Left and Right, as they are equivalent to the chambers in the actuator.

- F_{demand} [N]: The steady-state assistance force
- $F_{demandL}$ [N]: Demand on the left
- $F_{demandR}$ [N]: Demand on the right
- F_L [N]: The force on the left
- F_R [N]: The force on the right
- F [N]: The assistance force applied to the steering rack

The equations are as follows:

$$F_{demandL} = \max(F_{demand}, F_L) - F_L \tag{1}$$

$$F_{demandR} = \max(-F_{demand}, F_R) - F_R \tag{2}$$

$$\frac{dF_L}{dt} = rate\sqrt{F_{demandL}(1 + emptyingFactor^2)} - emptyingFactor\sqrt{F_L} \tag{3}$$

$$\frac{dF_R}{dt} = rate\sqrt{F_{demandR}(1 + emptyingFactor^2)} - emptyingFactor\sqrt{F_R} \tag{4}$$

$$F = F_L - F_R \tag{5}$$

Due to square-roots having an infinite derivative at zero, the implementation uses an approximation with a finite derivative to help the solver performance. There is therefore a 3rd parameter in the model, delta, which defines the closeness to the true square-root function.

3 Steering feel quantification

Objective analysis of steering hysteresis loop is a challenging task, therefore attempting to quantify steering feel from looking at links between subjective ratings and objective parameters is a valid course of action (Pfeffer et al. 2008).

Therefore metrics, derived from statistical studies of subjective physical vehicle steering feel assessment, can be applied to quantify the qualitive aspects of steering feel with regards to simulation models.

3.1 Vehicle steering metrics and characteristics

Eluded to in previous sections of this paper, physical vehicle steering displays a non-linear, hysteretic profile of torque feedback to the driver. van Daal (2007) cites previous physical studies, such as Farrer (1993) and Chrstos and Grygier (1997) in presenting a basic, graphical definition of 'good' steering feel in terms of time trajectories. These depict the relationship between handwheel angle and steering torque, lateral acceleration and steering torque and finally yaw rate and steering torque. Figures 6, 7 and 8 respectively present the characteristic hysteresis loops.

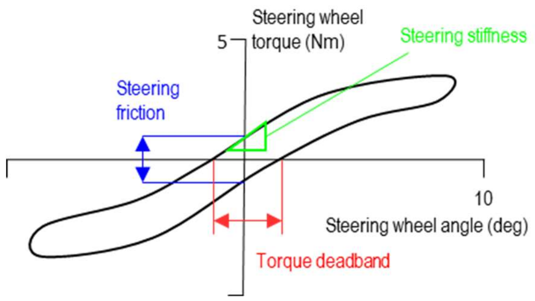


Figure 6. Characteristic hysteresis loop for a steering system. Recreated from van Daal (1997).

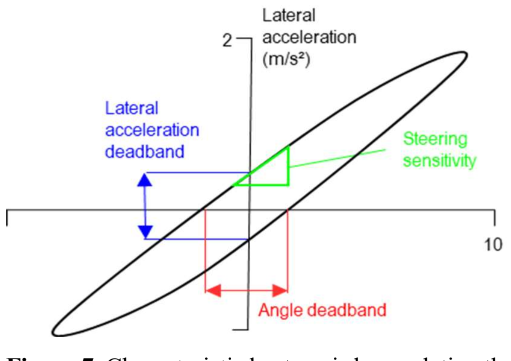


Figure 7. Characteristic hysteresis loop relating the vehicle lateral acceleration response to steering handwheel input Recreated from van Daal (2007).

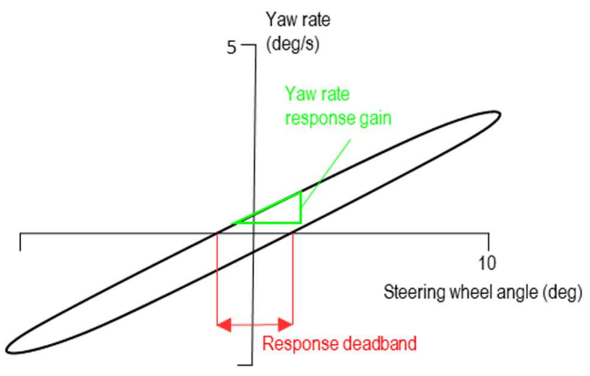


Figure 8. Characteristic hysteresis loop depicting the relationship between vehicle yaw rate response and handwheel input. Recreated from van Daal (2007).

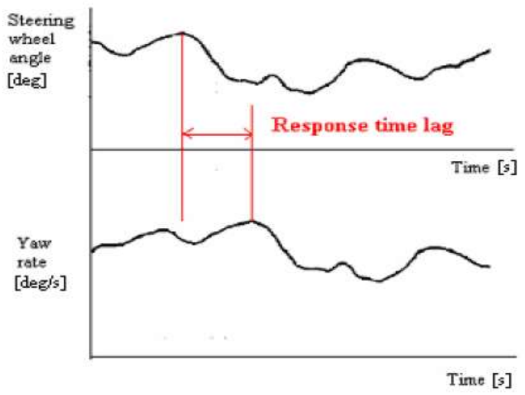


Figure 9. Response lag between steering input and vehicle response, in this case yaw rate (van Daal, 2007).

Of note when considering Figures 6, 7 and 8 is that only Figure 6 solely depicts the performance of the steering model in isolation. As Figures 7 and 8 reference vehicle variables (lateral acceleration and yaw rate), it can be deduced that the vehicle platform plays a significant role in producing good steering feel. Various metrics are defined by van Daal (2007) from these plots, which are displayed on Figures 6, 7 and 8. Figure 9 presents the lag in time between the steering input and vehicle response. A characteristic such as this is entirely predictable, given basic vehicle dynamics understand pertaining to the transient period of vehicle response.

As van Daal's (2007) work is not specifically focused upon the quantification of what constitutes 'good' steering feel, rather the identification of friction and compliance within the steering system, the application of that study to this paper is limited to holistic consideration of the dynamic relationship between steering angle/torque and vehicle response.

Xuxin and Zhicheng (2012) have built upon van Daal (2007), defining further metrics to characterize steering feel. Specific emphasis in their work is given to the attempt to define and quantify what 'good' steering feels like. A statistical study was conducted on various drivers, with more than 20 vehicles tested across 4 vehicle classes to quantify what drivers considered to be 'good' steering feeling independent of vehicle platform differences.

One upshot of this is Xuxin and Zhicheng (2012) differ in their definitions of basic metrics presented by van Daal (2007), namely the definition of torque deadband. Considering Figure 10 with reference to Figure 6, Xuxin and Zhicheng (2012) calculate the torque deadband angle at 1.3Nm of steering torque, to define the feeling of play within the steering as experience by a driver. van Daal's (2007) definition of torque deadband is termed as on-center hysteresis by Xuxin and Zhicheng (2012).

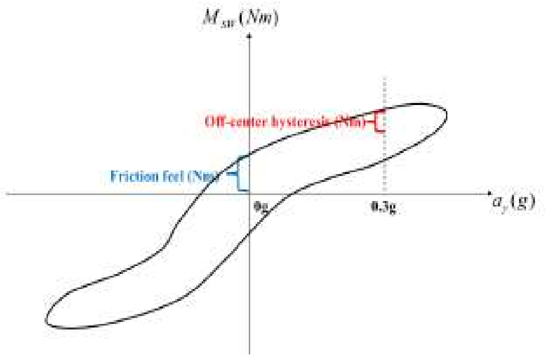


Figure 10. Steering hysteresis loop depicting steering torque against steering angle (Xuxin and Zhicheng, 2012)

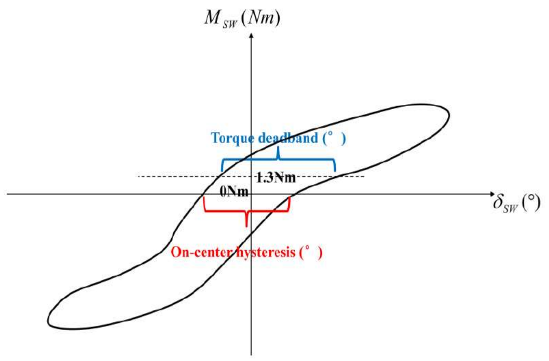


Figure 11. Relationship between steering torque and lateral acceleration, showing the off-center hysteresis (Xuxin and Zhicheng, 2012).

Interestingly, Xuxin and Zhicheng (2012) focus beyond the steering feel during on-center driving, also considering the off-center performance, although on-center driving appears to form the basis of their study. This is consistent with the Pfeffer et al. (2008) assertion that the majority of road driving concerns on-center steering events. Therefore, Xuxin and Zhicheng (2012) define another metric, off-center hysteresis to define the effort required to correct the steering whilst cornering. This is presented graphically as Figure 11. Whilst a formal definition of off-center hysteresis is not given, and the graphical presentation is somewhat unclear, off-center hysteresis has been interpreted in this paper as meaning the difference between the highest and lowest point of the hysteresis loop at 0.3g. The written definition of the metric Xuxin and Zhicheng (2012) gives reference to the need to correct the steering, which would entail the steering feedback torque following the lower, return boundary of the hysteresis loop.

Of primary importance to this paper are the quantified values Xuxin and Zhicheng (2012) present regarding what constitutes steering feel with a highly positive rating. Table 1 presents these values.

Table 1. Metric values considered to deliver a 'good' steering feel (Xuxin and Zhicheng, 2012).

Metric	*Range*
Response Gain Straight Path (°/s/100°SWA)	25-30
Response Time Delay (m/s)	>95
Torque Deadband (°)	<2.2
Yaw Response Gain (°/s/100°SWA)	28-32
Torque Buildup Cornering (Nm/g)	4-6
Off Center Hysteresis (Nm)	1.5-2.2
Effort Level (Nm)	3.6-4.5
Parking Efforts Standstill (Nm)	<3.3
Parking Efforts Rolling (Nm)	>1.5

The Response Gain Straight Path is the yaw gain when the vehicle deviates from a straight path (step steer scenario); Yaw Response Gain is the yaw gain during a sinusoidal steering event. Effort level was calculated at 0.3g lateral acceleration. Note that these metrics concern both vehicle response variables (relating to the vehicle platform a whole) and metrics directly describing the steering in isolation.

3.2 Steering model evaluation method

Knowledge obtained from the research presented in section 3.1 was synthesized to produce a series of Dymola/Modelica experiments. These are designed to evaluate the performance of the steering model presented in this paper against the values in Table 1, and the hysteresis loops in Figures 6, 7 and 8. These experiments enabled the objective assessment of the steering feel in the Dymola simulation environment prior to the model being tested in real-time simulators. Each experiment is described in Table 2. Note, vehicle velocity used was tuned so that other metric conditions (such as specific lateral acceleration, i.e. off-center hysteresis) were met.

All experiments used position actuation of the steering wheel, with the torque feedback (to the simulator rig) visually assessed to be smooth and consistent. Torque values used in metric assessment were handwheel torques, to assess the steering from a driver's viewpoint. Frequency of steering input during testing was 0.2Hz. A high, medium and low amplitude sinusoidal test was used to evaluate the steering results both with and without engaging the power assistance on-center (<±10°) as well as off-center performance. Closed loop throttle control was used to hold the target vehicle velocity for the tests.

Table 2. Full vehicle experimental setup to evaluate steering model.

Experiment	*Handwheel angle*	*Vehicle velocity*
Sinusoidal Steer (high amp)	±30°	60kph
Sinusoidal Steer (mid amp)	±10°	60kph
Sinusoidal Steer (low amp)	±3°	60kph
Stationary Steer	+150°	0kph
Rolling Steer	+150°	7kph
Steering Ramp	+35°	75kph
Step Steer	+45°	45kph

Different metrics were therefore evaluated by different experiments. Table 3 details which metric was evaluated by which test, noting if the metric was required to be evaluated at a specific condition (e.g. specific lateral acceleration).

Table 3. Tests used to evaluate objective metrics.

Metric	*Experiment*
Response Gain Straight Path	Step Steer
Response Time Delay	Sin. Steer (high amp)
Torque Deadband	Sin. Steer (low/mid)
Yaw Response Gain	Sin. Steer (low/mid)
Torque Buildup Cornering	Steering Ramp
Off Center Hysteresis	Sin. Steer (high amp)
Effort Level	Steering Ramp
Parking Efforts Standstill	Stationary Steer
Parking Efforts Rolling	Rolling Steer

4 Results

Table 4 presents assessment of the steering model against the optimal metric values presented in the previous section.

Table 4. Objective metric results from steering model full vehicle testing.

Metric	*Result*
Response Gain Straight Path (°/s/100°SWA)	19.426
Response Time Delay (m/s)	80.000
Torque Deadband - mid amplitude (°)	1.661
Torque Deadband - low amplitude (°)	0.125
Yaw Response Gain – mid amplitude (°/s/100°SWA)	14.045
Yaw Response Gain – low amplitude (°/s/100°SWA)	20.000

Torque Buildup Cornering (Nm/g)	4.847
Off Center Hysteresis (Nm)	0.194
Effort Level (Nm)	4.687
Parking Efforts Standstill (Nm)	4.694
Parking Efforts Rolling (Nm)	3.409

Figures 12, 13 and 14 detail the characteristic hysteresis loops for the steering model presented in this paper. For direct comparison with Figures 6, 7 and 8, which present the expected appearance, these plots were taken from the Sinusoidal Steer (mid amplitude) simulation. Figures 15 and 16 present the characteristic steering hysteresis loop for the high and low amplitude Sinusoidal Steer tests. To reach these results, the steering model friction parameters (both rotational and translational stiffness, damping and breakaway ratio) were optimized, as well as various power steering parameters such as the emptyingRate, F_{demand}, rate and the stiffness/damping of the torsion bar. This was done once the vehicle model has been developed to produce a satisfactory handling performance.

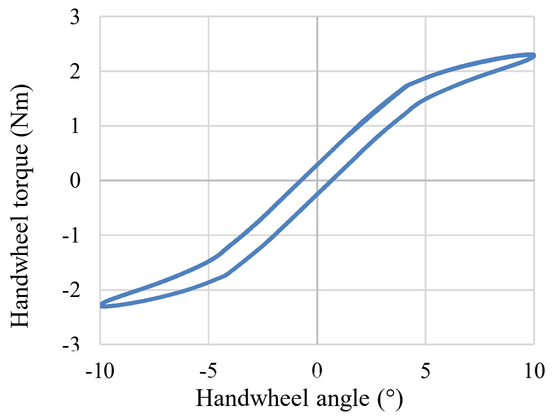


Figure 12. Steering torque hysteresis loop for Sinusoidal Steer test (mid amplitude).

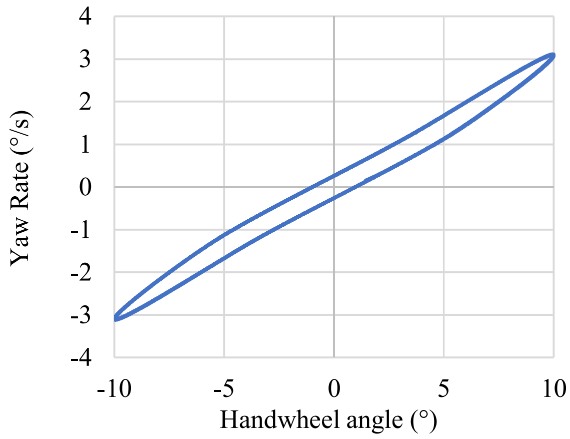


Figure 13. Vehicle yaw response to steering input for the Sinusoidal Steer test (mid amplitude).

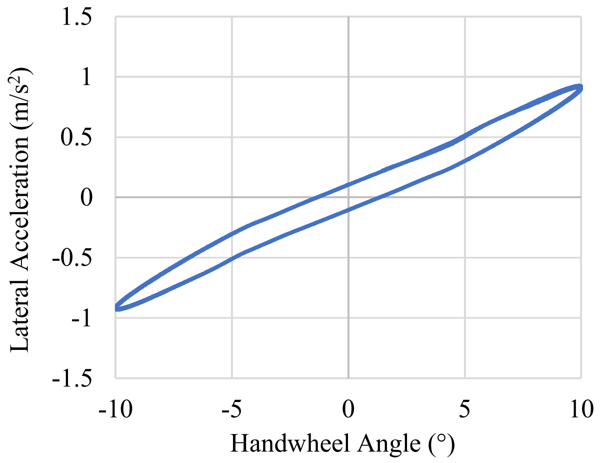


Figure 14. Vehicle lateral acceleration response to steering input for the Sinusoidal Steer test (mid amplitude).

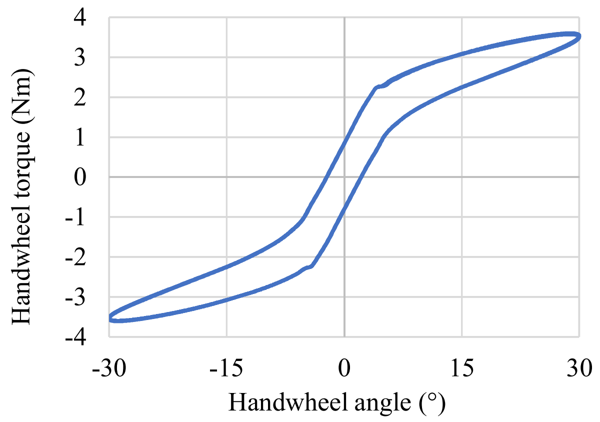


Figure 15. Steering torque hysteresis loop for Sinusoidal Steer test (high amplitude).

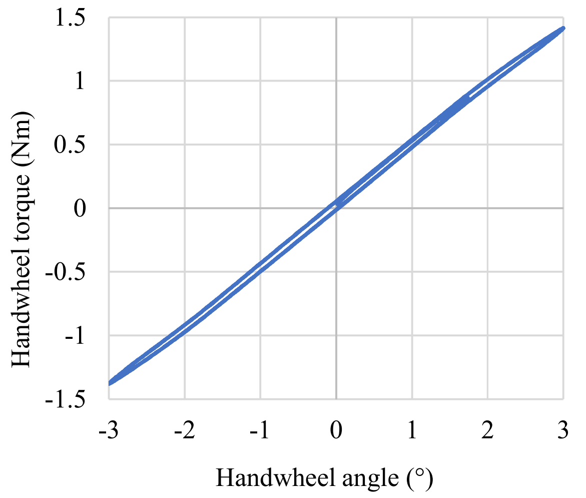


Figure 16. Steering torque hysteresis loop for Sinusoidal Steer test (low amplitude).

4.1 Discussion of metrics

Interrogation of Table 4 with respect to ideal target values detailed in Table 1 indicates the steering model has not achieved the set targets for Response Gain Straight Path, Response Time Delay, Yaw Response Gain and Parking Efforts Standstill. The value calculated for Effort Level it is sufficiently close to the upper boundary to omit discussion into the difference between value and target. Considering the failed metrics, they can be broken up into two distinct groups; metrics indicative of vehicle yaw response (Response Gain Straight Path, Response Time Delay and Yaw Response Gain) and metrics relating to a stationary vehicle condition (Parking Efforts Standstill).

Vehicle yaw response is a function of the vehicle platform as a wider system. Specifically, the performance of the tires and the physical vehicle dynamics setup variables (spring/anti roll bar rates, damper values, mass distribution etc.). Concerning the metrics, Response Time Delay is slightly below the optimal value, meaning the vehicle is achieving peak yaw rate quicker than is ideally desired; the Yaw Response Gains suggest the yaw magnitude for a given steering input is not high enough. Combining these two points, evidence indicates that the vehicle yaw response differs from the vehicles used as part of the Xuxin and Zhicheng (2012) study. Even though that study included cars of this type (executive/E class) as well as larger ones (SUVs), it also included lighter, nimbler vehicles of different classes (small family/C and large family/D class); specific vehicle parameters are not given for the vehicles used in their study. Therefore, it can be expected that the target values their study provides are not totally representative of the class of vehicle used in this study, rather a general idealised target. Thus, it is difficult to draw a direct comparison of vehicle yaw response between the vehicle model used in this study and the vehicles used in Xuxin and Zhicheng (2012). Of further interest, Xuxin and Zhicheng (2012) comment on how they found a lack of correlations in driver feedback for E class/segment vehicles due to a low sample size. Given these points, it is hard to consider the failure to meet the specific vehicle response target metrics to be the result of an incorrect vehicle or steering model, as a direct comparison is problematic. The proximity of the metric scores in this study to the published ideals therefore seems reasonable, indicating the steering model is valid.

Considering the Parking Efforts Standstill result, the source of the failure of achieving this metric can be attributed to the Pacejka (2012) tire model. The Pacejka (2012) model specifically employs an artificial damping factor at zero forward speed to prevenient an undamped vibration from occurring (Pacejka, 2012). It is envisaged that this could be the source of the high comparative magnitude of torque required to move the tire at zero velocity, rather than as a function of the steering model.

Differences in Yaw Response Gain and Torque Deadband results at low/medium amplitudes of steer angle can be considered a result of the experimental setups for each test. The Torque Deadband, being reduced at low steer angles (which do not result in power assistance being applied) suggests most of the hysteresis within the steering is being generated by the power assistance; a claim backed up by the assertion that the rack friction affects the hysteresis less at higher steering wheel angles (Pfeffer et al. 2008). This suggests the power assistance should be dominant over friction in generating hysteresis, which the results indicate. Furthermore, Yaw Response Gain is seen to deteriorate at the mid amplitude steering angle, suggesting the power assistance is adding to the compliance of the steering system during sinusoidal steering, further validating the performance of the pseudo-hydraulic power assistance model.

4.2 Discussion of plots

Analysis of Figures 12, 13 and 14 can be considered validation of the generalized performance of the steering model/pseudo-hydraulic power assistance model; the shape and form compare favorably with expected loops of real-world power steering systems presented in Figures 6, 7 and 8. Smooth variation of steering torque in Figure 11 indicates that the friction models are working correctly, with the pronounced bend at 3°-5° steer angle indicating the power assistance is correctly assisting the steering wheel. As the test used to produce Figure 12 utilized a positional input, the feedback torque from the steering model to the handwheel is consistent and smooth, a requirement of DiL steering models. Both Figures 13 and 14 show the vehicle responds correctly to steering input, further reinforcing the view that the vehicle model is valid, with the difference in the yaw metric scores is due to the yaw response metrics being not directly applicable to a vehicle of this class.

Moving onto Figure 15, a Sinusoidal Steer test of up to 30°, the same general hysteresis loop shape is presented as in Figure 12, a test of up to 10° steer. In effect this plot confirms the general validity of the steering model presented, as the same characteristics are present during both on and off-center steering situations. Noting the slight notch as the steering moves from unassisted to power assisted; not present in Figure 12, it can be deduced that this is an artefact of friction model/power assistance interaction at that point. This is most likely to be the result of non-optimal parameterization rather than model deficiencies, due to the small localized nature and the absence of a similar notch on the return bound of the hysteresis loop. Figure 16, where the power assistance is not engaged, supports this hypothesis, as the friction model torque response is smooth and consistent, with both the stationary and dynamic states encountered by the models. Of final note is the comparative lack of hysteresis shown in Figure 16, reflected in the Torque Deadband result for the Sinusoidal Steer test (low amplitude). This indicates that the current parameterization of the steering model does not include enough friction force, explaining the lack of hysteresis shown at each end of the steering torque hysteresis loops in Figures 12 and 15, and indicating this phenomenon is the result of non-optimal friction model parameterization rather than a modeling error.

4.3 Comments on real-time running

Various parameterization setups of the steering model (friction settings, steering assistance levels) were tested successfully on a workstation DiL setup before being deployed successfully on a full DiL simulator rig.

5 Conclusions

Overall, the work conducted during this study supports the presentation of the pseudo-hydraulic steering model as applicable for DiL applications for both on-center and off-center driving, based primarily on the generation of feedback curves characteristic of a real-world physical steering system. Steering model parameters were adjusted to influence the steering feel. The model has been shown to produce a representative vehicle and torque feedback response, comparing favorably with idealized objective metrics derived from physical vehicle testing. This indicates that the pseudo-hydraulic power assistance block is functioning correctly, with the friction model selection being valid. Whilst evidence indicates that the parameterization of the steering model is not optimal, this paper serves as a proof of concept of the pseudo-hydraulic steering model presented.

5.1 Further work

Initial further work would pertain to the analysis of free steer capability/performance of the steering model, as one of key advantage of using a pseudo-hydraulic power assistance model is the decay rate of force when the torque applied drops off suddenly. This would be using the methods/targets in ISO 17288:2011. Following this, obtaining a dataset of a real physical vehicle (dynamic setup etc.), would enable a parameterization setup to be developed with further confidence.

Developing a physical model of a hydraulic power steering would be of interest. The VeSyMA approach to the vehicle and subsystem architecture enables comparison between different implementations of the same system. Such a comparison between a physical model and the pseudo-hydraulic model could be used to fit the parameters for the pseudo-hydraulic model to suit a specific design, so that the design can be evaluated in a DiL environment.

References

Johan Andreasson, Naoya Machida, Masashi Tsushima, John Griffin and Peter Sundström (2016). Deployment of high-fidelity vehicle models for accurate real-time simulation. *Proceedings of 1st Japanese Modelica Conference,* pp78-86, 2016. DOI:10.3384/ecp1612495

Ansible Motion: Looking Down the Road: Driving Simulator Technology and How Automotive Manufacturers Will Benefit. White paper, 2015.

J.P. Chrstos, P.A. Grygier (1997). Experimental Testing of a 1994 Ford Taurus for NADSdyna Validation. *SAE-Technical Paper 970563,* 1997.

H. Dankowicz (1999). Modelling of dynamic friction phenomena. *ZAMM* 1999;79:399–409.

Pierre Dupont, Vincent Hayward, Brian Armstrong and Friedhelm Altpeter (2002). Single state elastoplastic friction models. *Automatic Control, IEEE Transactions on Automatic Control*, 47.5 (2002): 787-792.

D.G. Farrer (1993). An Objective Measurement Technique for the Quantification of On-Centre Handling Quality, *SAE Technical Paper 930827*, 1993.

Hans B. Pacejka (2012). *Tyre and Vehicle Dynamics*, 3rd edition. Oxford: Elsevier, pp 356-404.

Nissan. 2018. *Electro-Hydraulic power steering system.* Online. Accessed 18 March 2018. Available at: *https://www.nissan-global.com/EN/TECHNOLOGY/OVERVIEW/ehpss.html*

P. E. Pfeffer, M. Harrer and D.N Johnston (2008). Interaction of vehicle and steering regarding on-centre handling. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility,* 46:5, 413-428, DOI: 10.1080/00423110701416519

Marcus Rösth (2009). *Hydraulic Power Steering System Design in Road Vehicles.* Doctoral Thesis, Department of Management and Engineering, Linköping University, 2009.

B.A.M van Daal (2007). *Friction and compliance identification in a vehicle's steering system*. Extern traineeship report, Eindhoven University of Technology, 2007.

Xuxin He and Zhicheng Su (2012). *Links between Subjective Assessments and Objective Metrics for Steering*. Master's Thesis, KTH Royal Institute of Technology, Stockholm Sweden, 2012.

Modelling silicon carbide based power electronics in electric vehicles as a study of the implementation of semiconductor devices using Dymola

Leonard Janczyk[1] Yoshihisa Nishigori[2] Yasuo Kanehira[3]

[1]Dassault Systèmes, Germany, leonard.janczyk@3ds.com
[2] ROHM Co., Ltd., Japan, yoshihisa.nishigori@rohm.dsn.co.jp
[3]Dassault Systèmes, Japan, yasuo.kanehira@3ds.com

Abstract

In a joint effort, Dassault Systèmes and Rohm Semiconductor demonstrate how the introduction of silicon carbide (SiC) as a base material in power electronics improves the energy efficiency of a typical electric vehicle. As an application example simulation models of an electric drive and an electric vehicle are chosen.

Keywords: Power electronics, inverter, electric vehicles, Silicon carbide semiconductor, Dymola, Electrified Powertrains Library

1 Introduction

With the evolution of electric vehicles, electric powertrains with electric motors powered by a battery are increasingly becoming part of vehicles as well as of the onboard power supply system. In the industry market, renewable energies have different specifications and must be converted to fit to existing power networks. Additionally, in many cases, renewable energy system need a battery to operate continuously (Nakamura, 2015).

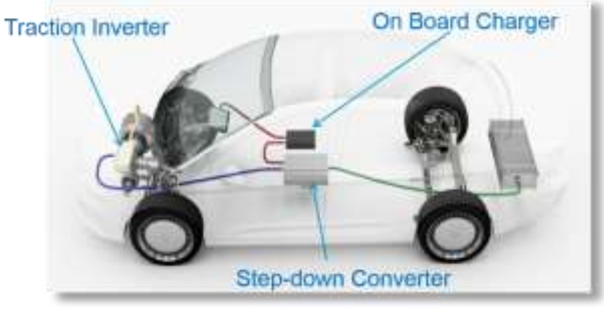


Figure 1 Concept drawing with power electronics linking the plug-in-charger (red), the DC battery circuit (green) the electric motor (blue).

In the mentioned applications, an electric system needs to link circuits with different voltage levels or circuits with alternating current (AC) and direct current (DC). This connection is facilitated by transformers or power electronics. Power electronics are often used in systems where dimensions and weight are important, such as electric vehicles (EV). For example, power electronics are required to link the battery pack, the electric machine and plug-in cable chargers, as illustrated in Figure 1.

Dassault Systèmes and Rohm Semiconductor are jointly investigating the use of system simulation for these cases, which utilizes semiconductor devices in a system simulation environment like Dymola.

After providing background information on power electronics and the advantages of SiC in chapter 2, the inverter models from the power electronics package in the Electrified Powertrains Library (EPTL) are introduced in chapter 3.

The thermal-electric modeling and calibration of an inverter module are explained in chapter 4.

In chapter 5, the inverter model is used in a system model of an electric drive to analyze the effect of the new power electronics material on the system electric efficiency and cooling requirements.

After converting the electric drive system model into a multi-dimensional table model in order to improve simulation speed, the improvement in energy efficiency and vehicle range is shown by a simulation of an electric vehicle.

2 Background

2.1 Silicon carbide (SiC)

Like most semiconductors, power electronics use silicon as a base material. Other types of materials include germanium and gallium nitride. A newcomer to this list is silicon carbide (SiC). Its material properties make it ideal for usage in power electronics: it is among the most mechanically robust materials (after carbon diamonds) and is able to withstand operating temperatures up to 200°C. Its specific thermal conductivity is five times higher than that of silicon, which facilitates the dissipation of electric loss power. Finally, the specific electric resistance is lower than traditional semiconductor materials.

SiC is a compound semiconductor which consists of silicon (Si) and carbon (C). It has superior performance compared to Si, with the breakdown electric field being ten times stronger and the band gap three times wider.

2.2 SiC characteristics in power electronics devices

Since the breakdown electric field of SiC is a magnitude stronger than Si, it is possible to build high-voltage devices with maximum operation voltages ranging from 600 V to a few thousand Volts with high doping density and thin drift layer. As most of the resistance of a high-voltage power device is in the drift layer, SiC can realize a high breakdown voltage device with extremely low on-resistance per unit area. In theory, the drift layer resistance per area can be reduced by a factor of 300 compared to Si for the same breakdown voltage (Nagano, 2018).

Minority carrier devices, i. e. bipolar devices such as IGBTs (Insulated Gate Bipolar Transistor) have been mainly used for Si in order to improve the increase in on-resistance accompanying high breakdown voltage. It has a problem on larger switching loss, however, the switching frequency is limited due to the dissipated heat generated by the electric loss power.

With SiC, a majority carrier device such as Schottky Barrier Diode (SBD) and MOSFET, which is a high-speed device structure, can be manufactured with high breakdown voltage. The device simultaneously enables three characteristics for power transistors: high breakdown voltage, low on-resistance, and high speed. Moreover, the wider band gap than Si allows power devices that can operate at very high temperatures (Tanaka, 2018) and (Ogawauchi, 2018).

Historically, Rohm financed the development of SiC devices and established an integrated production ranging from pulling wafer ingots to packaging and test. Rohm is starting to offer high-quality SiC devices, SBDs, and modules.

The adoption in industrial high-voltage power supplies and onboard electric vehicles chargers has started (Nakamura, 2015). Rohm is an official technology partner of the Venturi Formula E team and provides SiC devices for motor drive inverters.

In this way, SiC is a key state-of-the-art device to increase the efficiency of electric power systems around the world.

3 Electrified Powertrains Library

3.1 Overview

The Electrified Powertrains Library (EPTL) contains models covering the key components of an electric drive system in different levels of detail, i.e. physical, switched, averaged and energy-based. The key components are electric machines and inverters with their respective controllers.

3.2 Power electronics package

The power electronics package contains models to describe switching electronic devices required in the context of e.g. electric drive modeling.

It contains the semiconductor switches at the heart of the power electronic devices as well as power electronics devices such as inverters and converters.

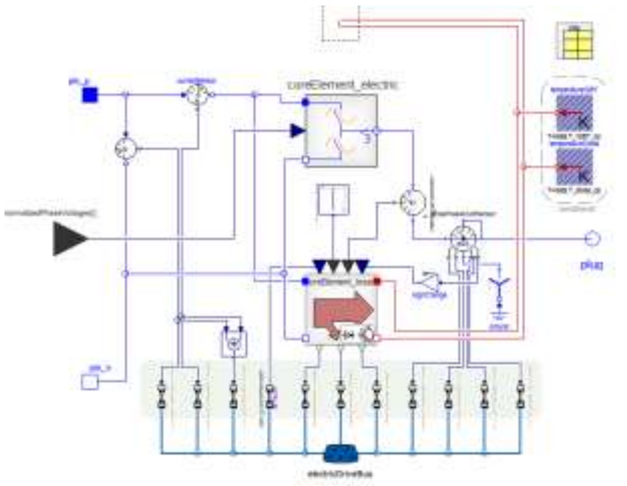


Figure 2 Electrical and loss model of an inverter.

In this context, inverters are of special interest, as their functionality allows to bridge AC and DC electric circuits. Inverter models are available in several levels of detail:

- idealized averaged electric models,
- averaged electric models with losses,
- idealized switching electrical models,
- switching electrical models with losses,

3.2.1 Electrical Model

Among the model variants available in the EPTL, the electrical behavior is modelled by the averaged model with constant efficiency in order to increase the simulation speed in large scale system models in later stages.

The input load signal is the desired three-phase voltage.

The active current that has to be drawn from the battery determined by recording the phase currents and phase shift. For the active current, the Clarke transformation is subsequently applied on the three-phase currents for converting the AC currents into a space vector with real and imaginary part.

Assuming a constant efficiency η, the relationship between the current on AC and DC side of the inverter is modelled by the following equation:

$$I_{DC} = I_{AC} \cdot \left(1 + \frac{1-\eta}{\eta}\right)$$

Major differences between the averaged and the switching electrical models include that the averaged model is an approximation of a switched inverter and does not describe the harmonics of the AC electric signal apart from the fundamental wave. Hence, the behavior is different to a model of a switched inverter as soon as the load is not 100% sinusoidal anymore. For example, switched inverter models coupled detailed with the respective loss models can be used to find a trade-off between current ripple and inverter losses.

A detailed description of the electric behavior models for MOSFET, IGBT and switches can be found in (Denz, 2014). The mentioned publication comprises also a validation of the EPTL models using electric measurement data.

3.2.2 Losses Model

The loss model computes the power losses incurred by the following effects:

- Conduction losses of the diode with regards to forward voltage V_{f0} taken from the datasheet $P_{cond,diode} = \left(\frac{1}{2\pi} - m \cdot \cos\left(\frac{\phi}{8}\right)\right) \cdot V_{f0} \cdot \sqrt{2} \cdot I$
- Conduction losses of the switching element with regards to drain-source voltage V_{ds0}, drain current and device temperature:
$$P_{cond,switch} = \left(\frac{1}{2\pi} + \frac{m \cdot \cos(\phi)}{8}\right) \cdot V_{ds0} \cdot \sqrt{2} \cdot I$$
- Reverse recovery of the diode with regards to forward voltage forward current, device temperature and forward voltage:
$$P_{loss,RR} = n_s \cdot E_{rr} \cdot \frac{\sqrt{2}}{\pi} \cdot \left(\frac{V}{V_{ref}}\right)^{K_v}$$
- Loss energy per switching operation with regards to current, device temperature and voltage to switch:
$$P_{switching} = n_s \cdot \left(E_{on} + E_{off}\right) \cdot \frac{\sqrt{2}}{\pi} \cdot \left(\frac{V}{V_{ref}}\right)^{K_v}$$

In this context m is the modulation index, I the load current, n_s the number of switches, V_{ref} the reference voltage for the datasheet parameters, K_v a coefficient for accounting the non-linear influence of the normalized voltage on the losses.

Forward voltage, drain-source voltage as well as the energies for on/off switching and reverse-recovery need to be entered as parameter table as specified in the datasheet, as given in the Appendix.

The power losses are turned into heat flow and can be connected to the thermal model in order to predict the device temperature.

3.2.3 Thermal Model

Along with the model describing the electrical behavior and the loss generation, shown in Figure 2, a thermal representation of the respective inverter is comprised. The diode and the MOSFET are modelled as so called Foster elements. The Foster element is an electric equivalent circuit model which describes the transient thermal behavior of the device as concatenated RC circuits. In this context the thermal mass is treaded as capacitor element and the thermal conductance as resistor element. The thermal conductance from junction to the casing of the module's switch and diode is given by the datasheet and multiplied by the number of elements, in that application with the number of the module’s switches, which is six for this device.

4 Modelling and calibration of an Silicon carbide based inverter module

4.1 Topology

The inverter model is based on the half bridge module BSM120D12P2C005 by Rohm Semiconductor (Rohm, 2016) with the characteristics as listed in Table 1.

Table 1. Characteristics of the used device (SiC) and a Silicon (Si) based benchmark device.

Category	Device	Benchmark
Technology	SiC	Si
Type	MOSFET	IGBT
Maximum Voltage	1200 V	1700 V
Voltage Drain-Source	600 V	900 V
Maximum Current	120 A	150 A
Operating Temperature	150 °C	80 °C
Number of switches	6	6
Voltage coefficient K_v	1.35	1.35

It consists of a MOSFET and a Schottky Barrier Diode (SBD), both based on silicon carbide.

The module is modelled using an averaged electric model with losses and a thermal model

4.2 Calibration of the power module

Additionally to the scalar parameters as shown in Table 1, the electric model of the diode includes the multi-dimensional relationship of source current and source drain voltage with regards to the operating temperature of the device.

Moreover, the calculation of the dissipated heat losses requires to enter the loss energy per switching operation as well as the recovered loss energies with regards to drain current and operation temperature.

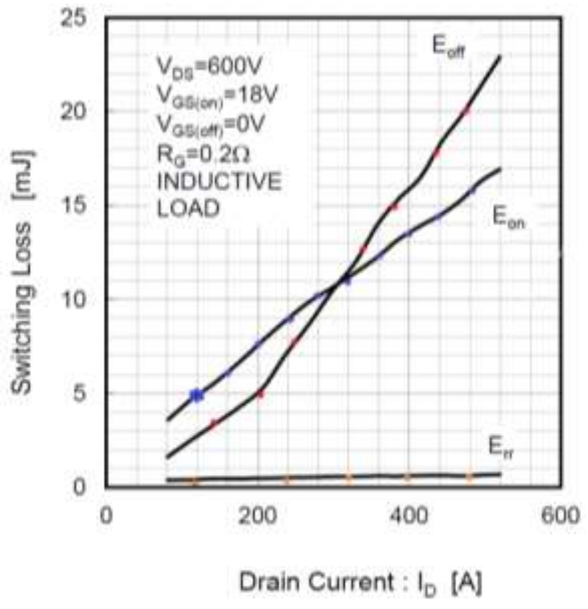


Figure 3. Sampling the characteristic curve of the used SiC device for switching losses when operating at 25°C.

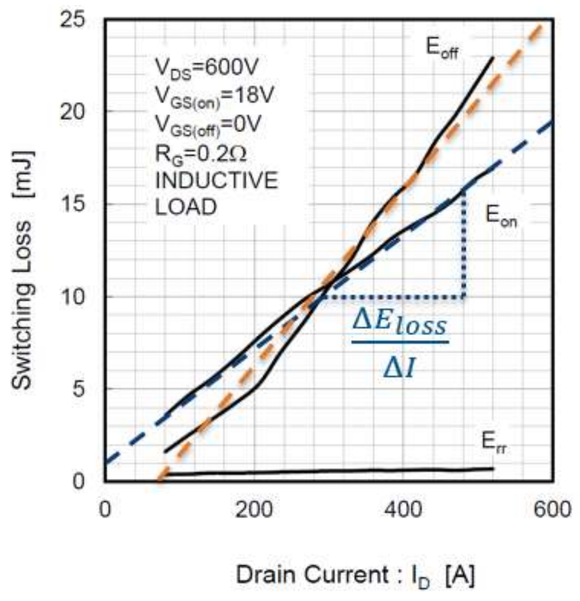


Figure 4. Linearizing the characteristic curve of the used SiC device for switching losses when operating at 25°C.

There are two options for calibrating the SiC device in the EPTL:

(1) Table Based: Either by sampling the characteristic curves given for the respective device, as indicated for example in Figure 3. The recorded data set is interfacing the Modelica model via the `NDTable` class. The output for given operating load and conditions is generated by performing a multi-dimensional interpolation in the data set.

(2) Linear equation: When the device characteristics feature sufficiently linear dependencies, the curve can be captured by linear equations within a confined area of operation as sketched in Figure 4.

With the switch-off loss energy curve with regards to drain current displays discontinuities, especially at 200 amperes, the table based method is chosen for modelling the losses.

The class `NDTable` for is implemented as described by (Schmitt et al, 2015) for an example for the forward characteristic of a Si-based diode.

4.3 Testbench simulation

In order to assess the impact of the increased efficiency the inverter is simulated as part of an electric drive system model as shown in Figure 5.

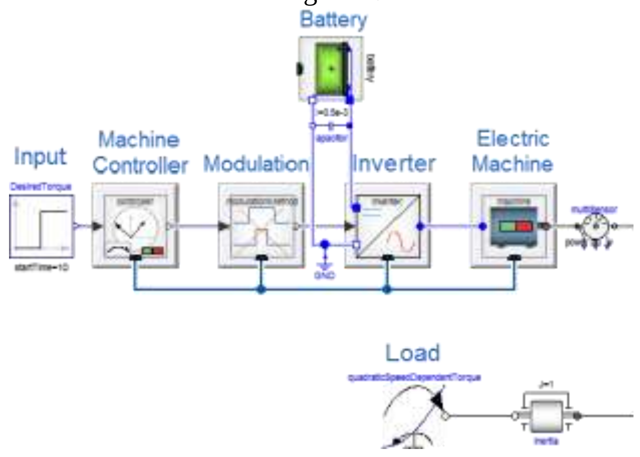


Figure 5 Simulation set up: Electric drive system including electric and loss model of the inverter comprising the power electronics.

The electric drive consists of five components: the electric machine with its machine controller, the inverter with modulation and the battery pack. The impedance effects of the cables of the high-voltage circuit as well as additional electric consumers of auxiliary devices are neglected in this setup.

Table 2 Simulation starting values and boundary conditions for the simulation.

Variable [Unit]	Value
Operating Temperature [°C]	80
Switching frequency [kHz]	25
Modulation harmonic [type]	sinus
Electric machine type [type]	PSM
Nominal machine frequency [Hz]	195
Nominal torque [Nm]	122
Nominal phase RMS voltage [V]	114
Nominal RMS current per phase [A]	170
Electric machine power [kW]	50
Desired Torque as input [Nm]	+10
Load Torque [Nm]	-10
Nominal machine speed [rpm]	1000

The electric machine is a 50 kW three-phase permanent magnet synchronous machine (PSM) with three pairs of poles. The corresponding machine controller has the target torque as input and features field-weakening control and the maximum-torque-per-ampere control.

With 25 kHz, a comparably high value has been chosen as switching frequency in order to illustrate the contrast in terms of dissipated loss power.

The overall boundary conditions of the simulation are laid down in Table 2.

The simulation in Figure 6 illustrates that lower switching energies result into lower heat loss power, hence a lower power consumption on the DC power side.

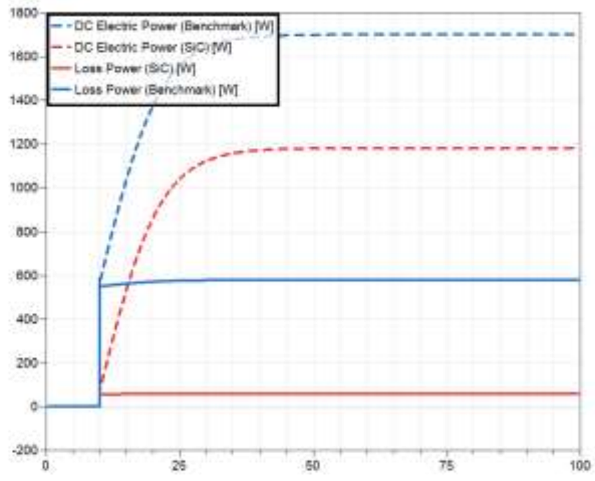


Figure 6 Comparison of DC load power and loss power between Si-based benchmark device (blue) and SiC-based device (red) for the first 100 seconds simulation time.

With the DC power load reduced significantly, the battery experiences a lower draining for providing a similar level of AC output. The battery charging state (SOC, that is the state-of-charge) therefor remains at a higher level.

5 Simulation and discussion

5.1 Water-cooled electric drive system

In order to assess the implication of the reduced loss power as calculated in chapter 4.3, the electric testbench component models are complemented by the respective thermal representation.

The use case as illustrated in Figure 7 consists of a permanent magnet synchronous machine which is water-cooled with the used liquid cooling media type being water/ethylene-glycol (50:50), as given in the `FluidHeatFlow` package in the `Thermal` library of the Modelica Standard Library 3.2.2. It represents a typical automotive setup with an external pump causing a volume flow through the inverter first and the machine subsequently. The inlet and outlet of the cooling cycle are modelled by the class `Sources.Ambient`. The component pump is an instance of the `VolumeFlow` class in the same package.

As modelling complexity increased significantly with the liquid cooling cycle, the battery was replaced by voltage source.

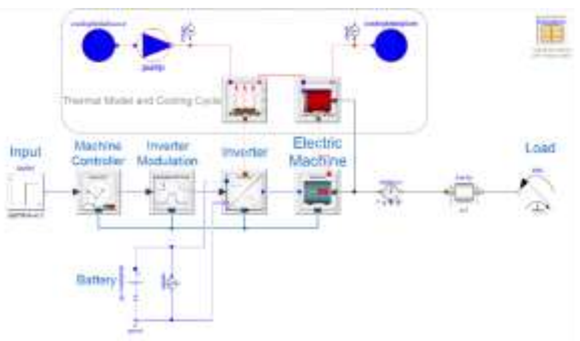


Figure 7 Simulation set up: Water-cooled electric drive system including thermal-electric models of inverter and electric machine.

The resulting reduction of the load for the cooling cycle is assessed by a simulation with the configuration as laid down in Table 3.

Table 3 Simulation starting values and boundary conditions for the simulation.

Variable [Unit]	Value
Inlet temperature coolant [°C]	65
Volume flow [l/min]	6.0
Operating Temperature [°C]	transient
Switching frequency [kHz]	25
Modulation harmonic [type]	sinus
Electric machine type [type]	PSM
Electric machine power [kW]	50
Desired Torque (Input) [Nm]	+122
Load Torque [Nm]	-122
Nominal machine speed [rpm]	3000

In Figure 8 is result of the testbench simulation is shown. With the generated heat losses reduced, the coolant experiences a significantly lower temperature increase.

With using a cooling cycle with liquid cooling, the indicator by choice is the temperature of the cooling medium. The coolant's temperature is increased by the dissipated heat loss power. The interfacing thermal conductance model is provided by Foster circuit parameters.

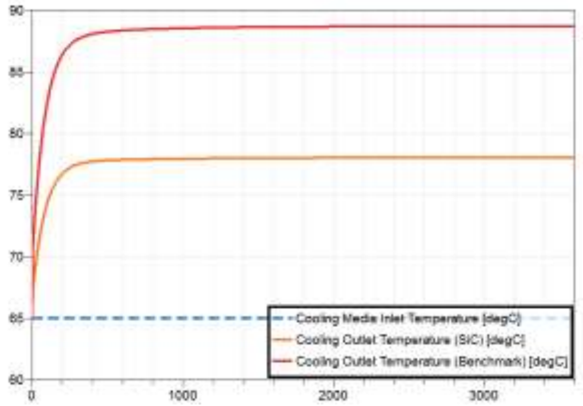


Figure 8 Simulation result: cooling medium temperature at the inlet and the outlet of the cooling cycle.

5.2 Electric vehicle simulation

The vehicle simulation package allows setting up example use cases for electric drives in different characteristics, providing an expandable and configurable infrastructure as well as all required components from environment to operational control models.

Hence characteristic maps are created for a specific supply voltage and specific temperatures. These values must be defined before the map generation process is started. For this purpose the boundary conditions as described in chapter 4.3 are chosen assuming that the inverter are operated in thermal equilibrium by using the cooling cycle as defined in chapter 5.1.

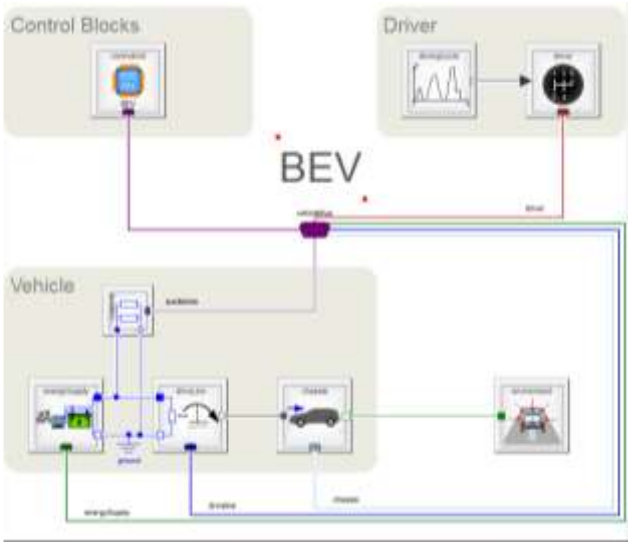


Figure 9 Electric Vehicle Simulation Testbench

With the battery being the primary source of energy, the electric powertrain as shown in Figure 10, the driving range increases significantly according to the simulation displayed in Figure 11.

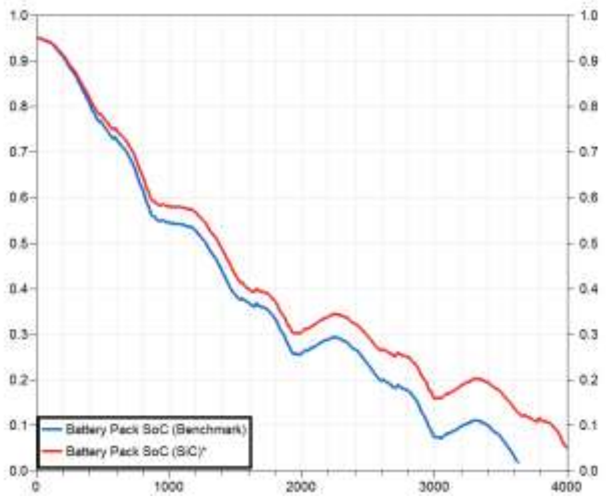


Figure 10 Charging state (SOC) of the EV traction battery when adopting the driving profile "Artemis Motorway".

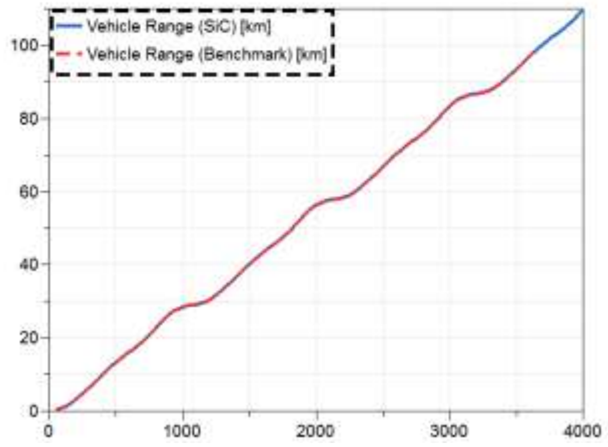


Figure 11 Maximum EV driving range when adopting the driving profile "Artemis Motorway".

6 Outlook

Tuning, increased accuracy in the simulation environment still be necessary, but it was already possible to demonstrate the improvement generated by device replacement just with high-level simulation.

Application of SiC in DC/DC converters is currently increasing, and with improvements in the relationship of performance and cost, the authors are confident that the application of SiC will expand.

References

Patrick Denz, Thomas Schmitt, Markus Andres. Behavioral Modeling of Power Semiconductors in Modelica. *Proceedings of the 10th International Modelica Conference*, Lund, Sweden, 2014. doi:10.3384/ECP14096343

Rohm Co., Ltd. SiC Power Module BSM120D12P2C005. *Data sheet*. 2016.09 – Rev C. 2016.

Thomas Schmitt, Markus Andres, Stephan Ziegler, Stephan Diehl. A Novel Proposal on how to Parameterize Models in Dymola Utilizing External Files under Consideration of a Subsequent Model Export using the Functional Mock-Up Interface. *Proceedings of the 11th International Modelica Conference,* Versailles, France, 2015. doi:10.3384/ecp1511823

T. Nakamura, Y. Nakano, T. Hanada. Advanced SiC Power Devices and Their Prospects. 4_S10_5, VOL. 3, *IEE-Japan Industry Applications Society Conference*, Japan, 2015.

K. Tanaka, S. Kyogoku, R. Iijima, S. Harada. Impact of the trench bottom shielding region on switching characteristics in SiC Double-trench MOSFETs. 4-010 *Annual Meeting of the IEEJ* 2018.

T. Nagano, K. Matsubara, H. Takubo, A. Toba. Suppression of Surge Voltage by Active Gate Driving for SiC-MOSFETs. 4-017, *Annual Meeting of the IEEJ* 2018.

Y. Ogawauchi, K. Nakahara "A Study on High-speed Gate Drive Circuit Using Hybrid Current/Voltage Source ofor SiC-MOSFET" 4-018 *Annual Meeting of the IEEJ* 2018

Appendix

Rohm BSM120D12P2C005 parameters

Output Characteristics

MOSFET drain-source voltage with regards to drain current and device temperature, measured in Volt.

ID [A]	25°C	125°C	150°C
40	0.6507	0.9771	1.105
80	1.3582	1.9828	2.224
120	2.1082	3.0735	3.457
160	2.9151	4.2496	4.818
200	3.8213	5.6101	6.377
240	4.8	7.1269	8.500

Diode Forward Characteristics

Source-drain voltage with regards to source current and device temperature, measured in Volt.

I [A]	25°C	125°C	150°C
30	0.53	0.80	0.83
40	0.70	0.92	0.95
60	1.00	1.20	1.75
80	1.13	1.30	1.40
120	1.35	1.65	1.80
160	1.55	2.00	2.20
200	1.75	2.40	2.60
240	1.97	2.75	3.05

Switch-on energy

Loss energy per switch-on operation with regards to load current and device temperature, measured in Joule.

I [A]	25°C	125°C
20	0.8000e-3	1.0000e-3
40	1.4946e-3	1.3441e-3
80	2.7321e-3	2.2566e-3
120	3.9696e-3	3.1816e-3
160	5.2071e-3	4.0440e-3
200	6.5571e-3	5.0066e-3
240	8.2607e-3	5.9441e-3

Switch-off energy

Loss energy per switch-off operation with regards to load current and device temperature, measured in Joule.

I [A]	25°C	125°C
20	0.3000e-3	0.3000e-3
40	0.4982e-3	0.5809e-3
80	1.1732e-3	1.2807e-3
120	1.9445e-3	2.1932e-3
160	2.8125e-3	3.1432e-3
200	3.8893e-3	4.3435e-3
240	5.0143e-3	5.6813e-3

Reverse Recovery

Reverse recovery energy per switching operation with regards to load current and device temperature, measured in Joule.

I [A]	25°C	125°C
20	0.0482e-3	0.0934e-3
40	0.0321e-3	0.0801e-3
80	0.0161e-3	0.0792e-3
120	0.0161e-3	0.0659e-3
160	0.0161e-3	0.0300e-3
200	0.0161e-3	0.0300e-3
240	0.0161e-3	0.0300e-3

Silicon benchmark device parameters

Output Characteristics

IGBT collector-emitter voltage with regards to collector current and device temperature, measured in Volt.

I[A]	25°C	125°C	150°C
40	1.31	1.34	1.33
80	1.56	1.75	1.78
120	1.83	2.1	2.18
160	2.04	2.43	2.54
200	2.25	2.76	2.92
240	2.47	3.1	3.31

Diode Forward Characteristics

Drain-source voltage with regards to source current and device temperature, measured in Volt.

I	25°C	125°C	150°C
40	1.17	1.06	1.06
80	1.39	1.35	1.35
120	1.54	1.55	1.56
160	1.68	1.74	1.77
200	1.81	1.9	1.95
240	1.92	2.06	2.12

Switch on Energy

Loss energy per switch-off operation with regards to load current and device temperature, measured in Joule.

I [A]	125°C	150°C
40	0.018	0.020
80	0.032	0.035
120	0.046	0.050
160	0.061	0.070
200	0.076	0.080
240	0.093	0.100

Switch off Energy

Loss energy per switch-off operation with regards to load current and device temperature, measured in Joule.

I [A]	125°C	150°C
40	0.013	0.015
80	0.025	0.028
120	0.036	0.040
160	0.045	0.050
200	0.054	0.060
240	0.063	0.069

Reverse Recovery

Reverse recovery energy per switching operation with regards to load current and device temperature, measured in Joule.

I [A]	125°C	150°C
40	0.019	0.023
80	0.028	0.031
120	0.034	0.038
160	0.040	0.044
200	0.045	0.048
240	0.048	0.051

AUTOMATED TEST OF CVT CONTROL SOFTWARE, USING FMI AND MODELICA MODELS

[2]Zeng, Weihua [1]Liu Fei [1]Belmon, Lionel

[1]Global Crown Technology Co. Limited, China
[2]Hunan Jianglu & Rongda Vehicle Transmission Ltd., China
{lionel.belmon,feil}@globalcrown.com.cn

Abstract

Hunan Jianglu & Rongda Vehicle Transmission Ltd., Co. has recently completed the development of a Continuously Variable Transmission (CVT). The transmission and the control software must be fitted to various vehicle platforms and must suit various driving conditions and driver requests. Because of the complexity of the system and due to the large number of driving conditions, systematic tests and validation methods are required. These methods should guarantee correctness and quality of the software and system behavior. For the test of the control software, Jianglu Rongda used an innovative test method based on automatic test scenario generation, Software in the Loop (SiL) simulation, and finally Hardware in the Loop (HiL) tests. The CVT control software is executed with a vehicle simulation model developed with SimulationX/Modelica, which is used both for the SiL and the HiL tests. The model accurately represents the CVT transmission including mechanical and hydraulic systems based on mechanical and physical parameters. In this paper, we describe the corresponding test process and tool chain along with the model validation. We also discuss the advantages and costs of this approach.

Keywords: embedded software, automotive, transmission, testing

1 Introduction

Transmission systems are continuously improved with respect to efficiency, robustness, costs and comfort. Most of these requirements have direct relationship with the transmission control software, which is becoming more and more intelligent and complex. Many situations have to be detected rapidly and reliably. Testing the controller, by covering all relevant driving situations and faults, repeatedly during the development cycle can be very time consuming and ineffective. Traditional methods based on hand-written test scripts do not perform well for validation and test of transmission controllers.

In this paper, we present a method based on automated scenario generation, execution and evaluation of useful test cases. We explain how the corresponding tools have been used to validate and to iteratively improve the control software of the CVT for a minivan, the Dongfeng Xiaokang.

The development environment for the CVT software integrates the following components:

- ETAS ASCET is used for model-based development of the CVT control software and for turning the model into production embedded C code for the Transmission Control Unit (TCU).
- A high-precision simulation model of the Xiaokang and of the CVT hardware. The model has been developed using SimulationX, based on Modelica.
- QTronic Silver is a tool for building virtual ECUs, used in Software-in-the-loop simulation. Silver imports both the vehicle model exported from SimulationX as FMUs and the TCU embedded software. Silver executes both in a co-simulation on a normal Windows PC. Furthermore, Silver provides interfaces to automated system level tests, the A2L/dcm database to integrate production calibration data into the simulation. Silver also supports various methods for stimulating sensor and actuator faults by modifying signals between plant and controller.
- QTronic TestWeaver, a coverage-driven test case generator, automatically generates, runs and evaluates thousands of different driving scenarios for system level tests during the development of the CVT control software.

By using this tool chain, we achieved the following main benefits:

- Accelerating the development process: Shortening development time is partly credited to the simulation-based tool chain sketched above. Furthermore, simulation lets the developers perform more engineering and test tasks on their PCs, and avoids blocking of rare resources like such as real ECUs, HiL (Hardware-in-the-Loop) test benches or prototype vehicles.

- Increased robustness: The automated generation of high-quality test cases enabled us to perform a much higher number of test cases than possible with the same effort using traditional test methods. Especially for extreme driving conditions and fault insertion tests the new methods we apply here found many problems that would have been difficult to find on the vehicle.

2 Virtual integration of the CVT software and of the plant model

The development environment for the CVT software contains a build system used to integrate and build the control software into a binary file for the TCU hardware. We extend this build system by adding a SiL target to it, so that developers can execute their code directly on PC first.

The developer can compile the module that he is currently developing, link it with all the other modules' object files and run the resulting, integrated control software as a DLL file on his PC immediately, to test the relative effects of his last modifications in a closed loop environment. The system is described in Figure 1. This compilation and build process takes less than a few minutes normally. The SiL setup can also provide access to thousands of variables of the control software that are listed in the A2L file, and also to every variable of the simulation model. The Silver simulation can also be attached to a debugger for step-by-step C code execution, and for injecting faults through changing values of variables. In addition, the simulation can be driven by a measurement MDF/CSV file, from vehicle test drive, or by Python scripts in order to trigger a specific driving condition of interest. Besides, the calibration data (DCM file for this project) can be 'flashed' into the controller of the simulation. This means we can perform some pre-calibration work based the accurate plant model.

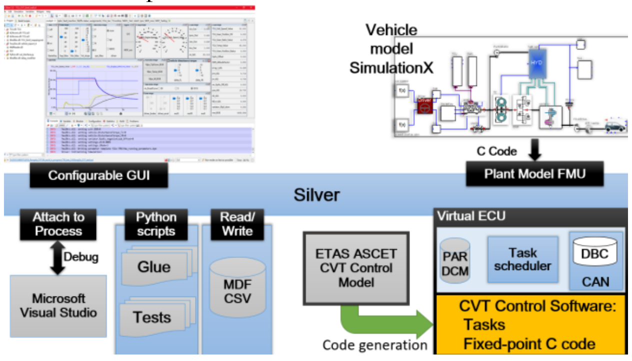


Figure 1 : Virtual integration of TCU and plant model

We execute the tasks for initialization as well as periodic tasks generated in the fixed-point C code of the CVT control software. We also configure inputs and outputs variables needed for simulation and interaction between ASW (application software) and BSW (basic software). The original BSW is not included in the closed-loop simulation, it is emulated by SBS (Silver Basic Software). SBS emulates features such as task scheduling and non-volatile memory.

3 Vehicle model development and validation

3.1 Model development

To build a closed loop simulation environment, we need a vehicle model to match the CVT control software. For better simulation quality, we need a well calibrated simulation model, that can reproduce the vehicle behavior well, both in terms of logic, input/outputs dynamics and performances. Here the required vehicle model of the Xiaokang van was developed by Global Crown using SimulationX. It models the longitudinal dynamics of the vehicle and the following components are included (see Figure 2): a model of the combustion engine with engine maps, start/stop control logic and CAN, a torque converter based on a SimulationX library, a DNR clutch, a detailed CVT model with hydraulic and solenoid systems based on real parameters and diagrams, drive shafts and differential with stiffness, left/right tires model with slip characteristics, a car body with given air and driving resistance, a road model with different surface properties, a simple ABS model which can simulate blocking sequences with ABS control of the wheel speed. Detailed attention was paid to the push-belt and hydraulic models. The belt models can simulate slipping if the pressure of primary and secondary pulleys are not suitable with the respect to the input drive torque.

The model was also required to run on a HiL system, which means it had to satisfy real-time performance and uses a fixed-step solver.

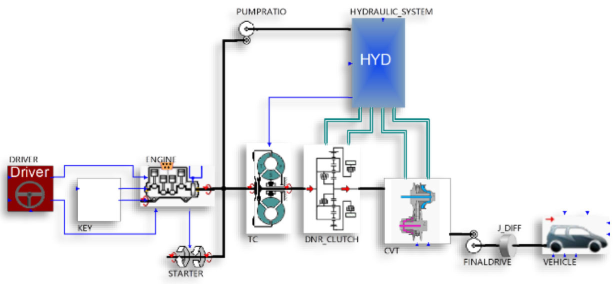


Figure 2 : Vehicle plant model schematic

The hydraulic system of the CVT has been modeled as detailed as possible based on real parameters and mechanical diagrams for every solenoid valves and oil channel. However, some high frequency physical effects are neglected because they are assumed to not influence the TCU. Such an assumption is also required because the model must be solved with a fixed-step solver. The neglected high frequency effects include pressure dynamics inside feedback volumes of spool valves, pressure dynamics inside solenoid valves. The resulting model has 65 states.

One difficulty in running Modelica models for real-time application lies in the event management and the Differential Algebraic Equation (DAE) formulation of the models. Such DAE with events usually require solvers with iterative methods, which is in conflict with real-time requirement of predictable execution time for one time step. For real-time applications, explicit fixed-time step integration is commonly used in the industry for many years. SimulationX can generate a ODE system from the DAE system and a classical explicit fixed-step solver method can thus be applied. Concerning events, most of them are removed during the modeling, using Modelica NoEvent() function. Most of events indicators are not meaningful for a fixed-step solver and are only meaningful for zero-crossing finding algorithms of variable step solvers. Such events can thus be safely removed of the model.

Concerning the time step selection, the TCU having a 10ms sampling rate, a 1ms step or smaller was defined as a 'higher limit' on the time step. The 'lower limit' time step depends on the model execution time and simulator CPU speed. In the present case, the model could run fast enough to execute real-time with 0.25ms step. Finally a 0.5ms time step was used.

3.2 Model open-loop validation

Model open-loop validation is done to verify some properties such as hydraulic pressure control. In the model it is possible to directly control solenoid input currents and measure hydraulic output pressure on primary and secondary pulleys of the CVT. It is then possible to compare to measurements on the vehicle or on a hydraulic test bench. Some calibration of pressure-current (I-P) characteristics of solenoid valves is done.

3.3 Model export to FMI and integration

For integration within the Silver co-simulation, to achieve a closed-loop system with the TCU embedded software, we used the Functional Mock-Up Interface and exported the model as a FMU2.0 for co-simulation. Using "co-simulation" or "model exchange" should not make a significant difference in our case since both the TCU embedded software and the plant models are using fixed step methods. The TCU software is updated every 10ms, the plant model is computed every 0.5ms with a fixed step solver. All required inputs/outputs are configured and mapped to the TCU software inside Silver. Some debugging work is usually needed to make the system works properly. Once debugging is done and the simulation works (engine start-up, ratio control, RND control, etc.), we proceed to closed-loop validation.

3.4 Closed-loop vehicle level validation

The closed-loop vehicle level validation is done by comparing simulation and prototype vehicle results for around 20 different drive situations, at various speeds, with various maneuvers. The maneuvers are selected so that we know the functional coverage of the model is good enough (DNR clutch, torque converter, hydraulic control, brakes, engine torque requests…).
When comparing vehicle results and simulation results, we must first make sure that TCU versions are close enough and that calibration parameters are the same, otherwise we might have very different behavior between vehicle and simulation. The simulation is executed in Silver with the plant model FMU, the TCU embedded software, calibration parameters and test sequences inputs read from measurements data files.
We give in Figure 3 an example of validation results obtained during the project. The key control variables are in good agreement between real vehicle and simulation. Such qualitative validation is done for the 20 drive test cases selected before. The time of building the initial vehicle model took about 6 weeks, the hydraulic system being the complex part of the model, and another 4 weeks for debugging and validation.

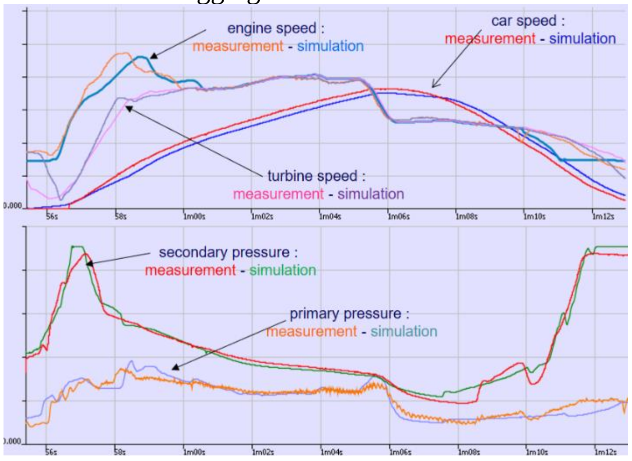


Figure 3 : Example of vehicle validation results

4 Automated testing for the CVT control software

It is possible to manually run tests in Silver, but this lacks the test automation and test reporting. Many simulations were done manually in Silver during debugging phases and for experimentation. For test automation and software validation, we applied TestWeaver, a test case generator, driving the SiL simulation through a sequence of inputs like acceleration and brake pedals, shift lever, road conditions, and fault insertions. As system states and system variables for monitoring, we selected meaningful signals, like engine speed, vehicle speed and the current of solenoids. All these variables will be recorded during each test case generation and stored in a test database. TestWeaver tries to drive the vehicle to new states and new driving conditions not reached before, trying to maximize coverage of requirements system states and also executed code.

As shown in Fig. 5, the state space is made up by all inputs and outputs that connect the system for the test with TestWeaver. For example, TestWeaver cannot set the vehicle speed (a reporter/output here), but it can learn that pushing the acceleration pedal (a input/chooser here) for a given amount of time then the system gets a higher vehicle speed. And then TestWeaver stores this behavior in a test database for later tests. This way, TestWeaver successively learns how to 'drive' the vehicle to any specific condition.

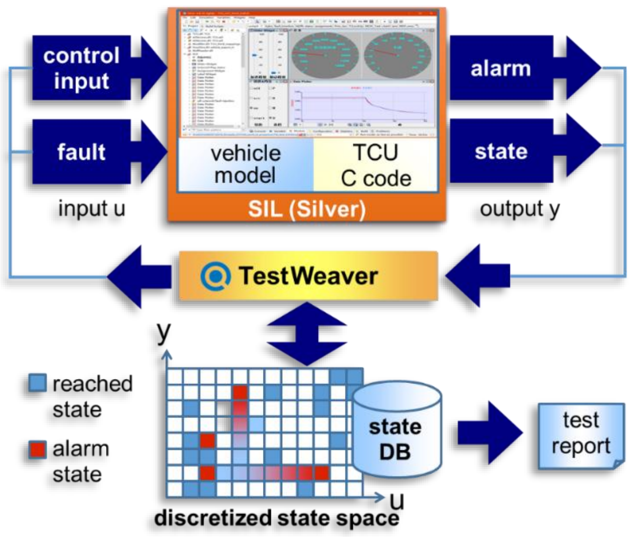


Figure 4 : Automated test of CVT control software by TestWeaver

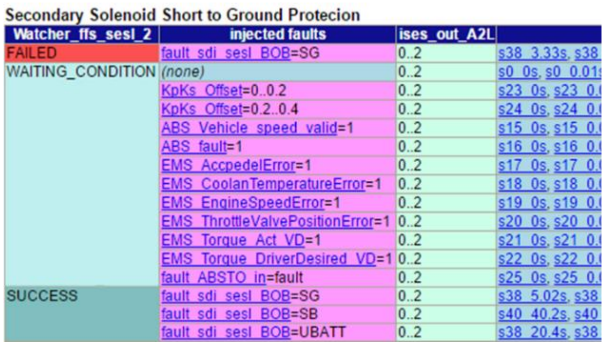
Before we start automated system testing with TestWeaver, a test or development engineer will do the following work:

- ❑ Configuration of input signals choosers, partitions, occurrence definition for each partition. Fault injection variables are choosers too.
- ❑ Configuration of output signals reporters, partitions, severity definition for each partition, to support automated evaluation of generated scenarios during testing.
- ❑ Defining requirements monitoring using TestWeaver "watcher" instruments. Here complex logical requirements can be defined and TestWeaver can automatically report if the requirements have been tested, with success or failure.
- ❑ Defining report templates for the variables we are interested in state coverage or code coverage or other specific testing results.
- ❑ General experiment configuration, such as the maximal duration of each scenario, maximal number for fault insertion per scenario, etc.

For testing the CVT control software, we need pay close attention to the following subjects:

- ❑ Ratio control of CVT: we need to monitor the difference between the actual ratio and target ratio to see if the CVT control software could control the ratio smoothly and precisely. Also if there are any oscillations in the ratio change, then there are probably issues on the ratio control.
- ❑ Diagnostic Trouble Code: for this part of the control code, we systematically check, using requirement watchers, that for a given fault condition the TCU executes a suitable fault reaction.
- ❑ Belt slip monitoring: for CVT, if belt slipping occurs, the lifetime would be shortened dramatically, even to the point of destroying the CVT. Therefore, we need monitor the slipping to analyze why this condition occurs.
- ❑ Code coverage: here the code coverage tool integrated in TestWeaver, was used. This feature is achieved by compiling the CVT control software with a special flag that tracks the C code coverage during execution in the Silver simulation.

In Figure 5, we show a report generated by TestWeaver that summarizes fault insertions on a solenoid and the monitoring of the fault reaction *Watcher_ffs_sesl*. TestWeaver detected several scenarios in which the TCU fails to apply a suitable fault reaction to the solenoid fault, which might lead to a dangerous situation for the CVT with a loss of transmission ratio control.



Secondary Solenoid Short to Ground Protecion

Watcher_ffs_sesl_2	injected faults	ises_out_A2L	
FAILED	fault_sdi_sesl_BOB=SG	0..2	s38_3.33s, s38
WAITING_CONDITION	*(none)*	0..2	s0_0s, s0_0.01
	KpKs_Offset=0..0.2	0..2	s23_0s, s23_0.
	KpKs_Offset=0.2..0.4	0..2	s24_0s, s24_0.
	ABS_Vehicle_speed_valid=1	0..2	s15_0s, s15_0.
	ABS_fault=1	0..2	s16_0s, s16_0.
	EMS_AccpedelError=1	0..2	s17_0s, s17_0.
	EMS_CoolanTemperatureError=1	0..2	s18_0s, s18_0.
	EMS_EngineSpeedError=1	0..2	s19_0s, s19_0.
	EMS_ThrottleValvePositionError=1	0..2	s20_0s, s20_0.
	EMS_Torque_Act_VD=1	0..2	s21_0s, s21_0.
	EMS_Torque_DriverDesired_VD=1	0..2	s22_0s, s22_0.
	fault_ABSTO_in=fault	0..2	s25_0s, s25_0.
SUCCESS	fault_sdi_sesl_BOB=SG	0..2	s38_5.02s, s38
	fault_sdi_sesl_BOB=SB	0..2	s40_40.2s, s40
	fault_sdi_sesl_BOB=UBATT	0..2	s38_20.4s, s38

Figure 5: Fault reaction table for Solenoid Short

In the first column of the above table, we have the “state” of the “requirement watcher”.

Failed: the requirement has been “violated”.

Waiting_condition: requirement does not apply yet.

Success: the requirement has been tested and results are success.

The second column shows the injected faults.

At last for the state column, it lists some of the scenarios relevant to the state of the requirement watcher.

A scenario (s38 3.33s for example) listed in the table can be inspected in TestWeaver and replayed in Silver for detailed debugging by clicking on it.

After several test cycles, engineers removed most of the problems in the control software. For each cycle, there are thousands of test cases for different driving conditions, many usual but also many unusual conditions are included. This systematic analysis through Silver/TestWeaver is an important complement to other test methods, such as HIL tests, test benches and real vehicle tests.

5 Plant model for Hardware in the loop simulation

Since the plant model was developed from the beginning with the objective of satisfying real-time requirements, and was using a fixed-step solver, the adaptation to the HiL was rather straightforward.

The HiL system used in this project is an ETAS LabCar. The system was purchased several years ago and is an older version that does not support the FMI interface yet. To port the plant model on the HiL, we thus used the ETAS Labcar ‘export format’ of the SimulationX code export. SimulationX can generate the model c-code according to the ETAS LabCar module format. The SimulationX model can then be added to the Labcar project and compiled for its Linux real-time operating system. With newer ETAS Labcar versions, we could probably use directly the FMU generated for the SiL since it also contains the model c-code and can be recompiled for the ETAS Labcar target. The FMI standard would simplify interfacing of tools.

The model itself does not need to be changed from the SiL to the HiL, it was developed so that it can satisfy both cases, with a fixed-step solver. We only had to modify the code generation target from FMU to Labcar. During this process, we found some minor bugs and compilation issues that were solved thanks to ESI/ITI SimulationX technical support.

6 Conclusion and future work

We presented an approach for an automated test of transmission control software based on SiL simulation (Silver) on standard PC and intelligent generation of test scenarios (TestWeaver) and how this testing method can be applied to the development of a CVT control software. For most of the state coverage, the intelligent automated test could generate thousands of scenarios, each scenario executing a 60 second drive maneuver. We also added specific test cases and test scripts to execute directly various standard maneuvers and fault injection.

To summarize, through the above approach of testing, we speed up the development cycle time for the CVT control software, and simulated most of the extreme driving conditions on a PC which is much safer than testing these conditions in real vehicles. After we solve most of the problems in the model/C code we found using SiL tests, then we move onto HiL tests and real vehicle tests using a more mature control software quality, reducing test time for the later, more expensive and less available platforms.

As future work, we plan to re-use TestWeaver “requirements watchers” and TestWeaver generated scenario database from SiL and port them on the ETAS HiL.

References

[1] http://www.fmi-standard.org/

[2] A.Abel, T.Blochwitz, A.Eichberger et al. Functional Mock-up Interface in Mechatronic gearshift simulation for commercial vehicles, 9th International Modelica Conference, 2012, Munich.

[3] A.Junghanns, R.Serway, T.Liebezeit, M.Bonin. Building virtual ECUs quickly and economically. ATZ Elektronik, 03/2012, Volume7.

[4] E.Chrisofakis, A.Junghanns. Simulation-based development of automotive control software with Modelica. Dresden: Modelica international conference, 2011.

[5] M.Tatar, Schaich, Breitinger. Automated test of the AMG speedshift DCT control software. Berlin: 9th CTI Innovative Automotive Transmissions Symposium, 2010.

[6] J.Mauss, M.Simons. Chip simulation of automotive ECUs. 9th symposium

Steuerungssysteme fur automobile Antriebe, 2012, Berlin

[7] N.Papakonstantinou, S.Klinger, M.Tatar. Test-driven development of DCT Control Software. 8th International CTI Symposium Innovative Automotive Transmissions, 2009, Berlin.

[8] M.Neumann,M.Nass,M.Tatar. Absicherung von Steuerungssoftware fur Hybridsysteme, Autoreg 2011, Friedrichshafen.

[9] L.Belmon, J.Yan. Modeling and simulation of DCT gearshifting for real-time and high-fidelity analysis. SAE China-FISITA conference, F2012-C04-014, 2012

[10] A.Abel, U.Schreiber, Valsania, Fornelli. Simulation based design of gearboxes for high-performance sports cars. Modena: 11th HTCES conference, 2005.

[11] L.Belmon, Y.Geng, H.Qian, Virtual integration for hybrid powertrain development, using FMI and Modelica models, Lund, Modelica international conference, 2014

The Fault Library - A New Modelica Library Allowing for the Systematic Simulation of Non-Nominal System Behavior

Julia Gundermann[1] Artem Kolesnikov[1] Morgan Cameron[2] Torsten Blochwitz[1]

[1]ESI ITI GmbH, Dresden, Germany
[2]ESI Group, Aix-en-Provence, France,
{julia.gundermann,artem.kolesnikov,morgan.cameron,torsten.blochwitz}@esi-group.com

Abstract

We introduce a library developed at ESI ITI for modeling faults in physical systems. We outline the motivation of how and why to model faults as well as a description of the library structure. The new library is exemplified in different fields of applications. In addition, we demonstrate a number of complementary tools and techniques for analyzing the results of simulations of faulted models.
Keywords: Fault, FAME, Reliability, Test case generation, Model based diagnosis

1 Introduction

To date, most Modelica libraries consider physical systems in their nominal configuration. There have been limited attempts to extend model coverage towards behavior beyond that. These include the Fault triggering library (van der Linden, 2014), which allows for the insertion of *faults* into models of existing components; and the FAME library (de Kleer et al., 2013). The latter has served as a basis for the development presented in this paper.

The range of applicability of the Fault library we introduce below is broad. We want to illustrate this based on the examples we studied in the scope of our development. They are listed below.

1. Braille printer: We take as a starting the point, a model of a braille printer, which includes mechanical, electrical and magnetic parts. Under normal operation, the force from the magnetic system pushes the needle into the paper. We would like to know, which *faults* in the system might prevent the embossing, leading to a rupture of the paper, prevent the needle from moving back to its initial position, or lead to overheating of the magnetic part.

2. Automotive transmission: The specification for a car contains requirements such as "The car should be able to accelerate from 0 to 100 km/h in 5 Seconds." We want to test the fulfilment of these and other criteria based on the model of a car. Furthermore, we want to identify critical *faults* in the transmission system that lead to a violation of these criteria. On the other hand, one could ask the question whether it is possible to determine the presence of *faults* from some measurable output variables such as velocity or fuel economy, or some other sensor data.

3. Battery package: Battery management systems are used in industry for the protection and operation of Li-Ion battery packs only within safe operating conditions. Because of various requirements the battery management system has to monitor and protect each cell against over-/under-voltage, over-/under-temperature, over-current and other *faults* (Xing et al., 2011). This leads to the necessity of an uniform testing of the battery management system for every type of battery pack. We want to generate *faults* for the battery management system in the battery cell simulator systematically and automatically reuse the model in different battery packs.

4. Feed axis: In view of Industry 4.0, predictive maintenance becomes more feasible in production machinery because of the technical ability to handle large data volumes. Modern machine tools have intelligent drives with control systems e.g. main and feed axis drives, or auxiliary drives (Friedrich et al., 2016). The control system data can be used for *faults* detection and prediction without additional measurement equipment. We would like to know how signal *faults* in the control system of a feed axis might be detected by using measured signals in the feed axis drive.

2 Defining Faults

In the preceding section we spoke about *faults*. In the following we will define more clearly what is meant by this term.

The reasons a physical system can deviate from its designed ("nominal") behavior are manifold. First of all, the manufacturing of products is only exact to a certain point - there are variations and sometimes systematic deviations from the specified system design. Secondly, during the use of a product, its behavior can change. Materials and material pairings are subject to wear, aging, abrupt failures etc. Effects include loss of lubrication, corrosion and hardening of materials.

Throughout this paper the term *fault* refers to any of these processes - i.e. it is a deviation from nominal behavior. It shall be emphasized that we do not care about the underlying physical processes, let alone its dynamical evolution and/or interaction with other processes. The Modelica models developed for the Fault library simulate only the effect of a process on the system's behavior. For example, mechanical *faults* in joints, gears, shafts, springs or clutches because of breakage or slipping leads to the reduction of transmitted force/torque. The current reduction because of bad or open connections in an electrical system, the mass flow decrease in a hydraulic system because of leakage or obstruction as well as changes of entropy flows in the thermodynamics have related behavior and can be similar structured and analyzed by modeling (Herrmann, 2006).

3 The Fault Library

In the following section the Fault library is described. Its development is based on the FAME library (de Kleer et al., 2013; Minhas et al., 2014). This library is the basis of three fields of applications outlined in Sec. 4. The Fault library consists of a `Basic` package containing elementary type definitions to parametrize *faults*. Beyond that, its structure is oriented according to the structure of composed models in the Modelica language (van der Linden, 2014)

3.1 Type Definition: Fault - Continuous and Discrete

Based on the processes and effects outlined above (cf. Sec. 2) two types of *faults* are defined: continuous and discrete. The library contains appropriate type definitions for both these *fault* types- Continuous.Fault and Discrete.Faults are records, which contain an editable variable, respectively `Real intensity` or `Integer active`. Rather than being fixed parameters, these are dynamic variables which can change their value during simulation. By using a type definition for the *faults*, it is easy to systematically read out and control the *faults* in a model.

Continuous faults parameterize the strength of a gradual effect. Typical examples are wear and aging phenomena which can lead to a gradually different value of a backlash parameter, or friction coefficient, etc. In the `record Continuous.Fault` this gradual change is translated into a normalized `Real` named *intensity*, ranging between zero (nominal) and one (maximal effect). If the *fault* modifies a parameter or variable, the extended `record Continuous.FaultWithFunction` can be used, which allows for the definition of the functional dependency of parameter change, e.g. multiplicative $p = p_0(1 + c_0(\texttt{intensity} * \texttt{scale})^n)$, exponential $p = p_0 \exp\{c_0(\texttt{intensity} * \texttt{scale})^n\}$, or others. Herein, p_0 is the nominal parameter value, and c_0, n are parameters of the function. The `scale` parameter is needed to define a typical magnitude of the *fault* effect, which depends on the specific application, e.g. the mass of the surrounding components, acting forces, etc. For details, see Sec. 3.3.

Discrete faults are either *active* or not, i.e. the *fault* is switched on or off. Typical examples are abrupt phenomena, such as failure of an electronic component, or breaking of a mechanical connection. The `record Discrete.Fault` contains the `Boolean active`, which is `false` in the nominal case, and `true` if the *fault* is active. If the *fault* modifies a parameter or variable, the extended `record Discrete.FaultWithFunction` can be used, which allows for the definition of the functional dependency of parameter change: proportional to p_0 -including setting the prefactor d_0, or as an alternative value - including setting the value p_1.

3.2 Types of Faults in a Model

A Modelica model can consist of components taken from different libraries. These components are connected with each other. *Faults* change the behavior of components (FaultAugmentedModels), alter the transport of quantities between components (ConnectorFaults) or add connections (BridgeFaults). A snapshot of a model which was augmented with components from the Fault library, is shown in Figure 1.

ConnectorFaults can be inserted at (connected) connectors or into existing connections. The first type cuts the connection (in Figure 1, below), it has two connectors, and can be used to model, for example, the breaking off of a mechanical component, or a mechanical obstruction. The second type has only one connection (in Figure 1, at the top), it is added to a connection, and can be used to model, for example an additional friction force/-torque in a mechanical connection, or a leakage in hydraulics. The models of these *faults* consist of a *fault* and a set of equations which depend on the intensity/active(-ity) of this *fault*, and, in general, on a scale.

BridgeFaults are models with two connectors (in the middle of Figure 1). For each domain there exists one BridgeFault. They can be inserted between existing connections, adding further connect-statements, if the *fault* is `active`.

FaultAugmentedModels with parametric *faults* are fault-augmented counterparts to the nominal SimulationX library models. For example, Figure 1 shows a `FaultAugmentedModels.Electricity.Analog.Basic.Resistor`, which is an extension of the `Electricity.Analog.Basic.Resistor`. It contains a *fault*, which changes the resistance parameter `r=Continuous.ChangeParameter(1,fault)` from its default value 1, as defined by `intensity`, `scale`, and `functionType` in the `fault`-record. The Fault triggering library (van der Linden, 2014) models *faults* in a similar way to those contained within FaultAugmentedModels.

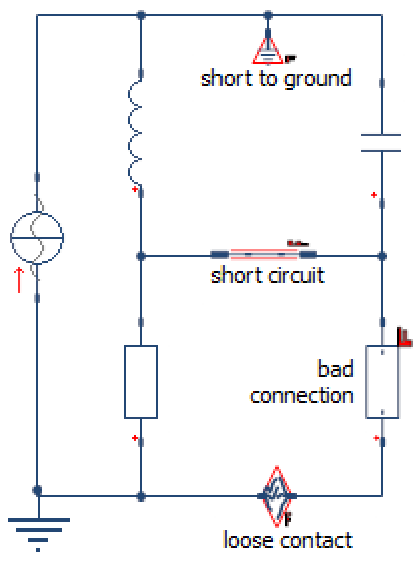


Figure 1. Screenshot of a model of an electrical circuit which has been augmented with faults. One of the resistors is replaced by its `FaultAugmentedModels` counterpart. To the connection between the inductor and the capacitor a `ConnectorFault` modeling a short to ground has been added. The connection between the two resistors has been cut by a `ConnectorFault` modeling a loose contact. An additional switchable connection (BridgeFault) modeling a short circuit between two junctions in circuits of consuming components.

3.3 The Fault Scale

It became clear, especially when modeling the connector or bridge *faults*, but also during the development of the fault-augmented counterparts of the library components, that for most of the *faults* the amplitude of the strongest effect (intensity=1, or active=true) differs depending on the surroundings of the system into which the *fault* is inserted. For example, the frictional torque in a rotational connection which prevents the latter from moving is several orders of magnitude higher in a ship's propeller than in a clockwork mechanism.

In the library this phenomenon is captured with the definition of a positive Real parameter named `scale`, which can be included and defined in the definition of a *fault*, if needed. It should be emphasized that the meaning of the scale differs between different *faults*. Furthermore, scales of the same *faults* in a model can differ in different places - e.g. if the model contains both the ship's propeller and the clockwork mechanism.

For a meaningful effect of the inserted *faults*, it is crucial to find ways to estimate the scale for each *fault*. In the optimal case, one can calculate the scale from the amplitudes of variables in the surroundings of the *fault* as recorded during a single simulation of the nominal behavior. For example, the scale of the translational sticking *fault* is a prefactor to the friction force added to the connection. The latter is defined as F= 1N`*scale*intensity` for scale 1. The scale should be chosen to be high enough to ensure that the connection stops moving when the *fault* intensity is equal to one. On the other hand it should be low enough that one is not faced with numerical issues, because the system is stiff or the solver has trouble integrating when the *fault* is switched on during simulation time. In the optimal case the scale value should be set to ensure that the range of the friction force over increasing intensity is such that for low intensities it already has some (recognizable) effect on the connection but does not already prevent it from motion at all (sensitivity to *fault* intensity).

Estimating the scale - even in the seemingly simple case of the friction force from above - is not as straightforward as one might think and is indeed a dedicated research topic. Currently, several methods (reading from base unit force, energy, power, etc.) are under investigation.

4 Fields of Application

Having illustrated the definition of a *fault* and the structure of the Fault library in detail, in the following section we want to give more details about what this can be used for. There are three fields of application we see for the Fault library.

4.1 Reliability

This field is probably the most obvious application. Since any real component has been manufactured, and is subject to wear and aging processes, it can be helpful to estimate the impact of such *faults* at an early development stage. With the Fault library the following questions can be addressed: How does my system behave when *faults* are active? What are the critical *faults* or *fault* combinations with respect to some performance goal? Is it always true that increasing the intensity of a *fault* worsens the performance of the system?

With the Fault library we can use system simulation to answer these questions. Beyond that we have developed another library named *Performance Indicators* to systematically measure whether a system meets or violates predefined criteria (cf. Sec. 6).

The examples "Braille Printer" and "Automotive Transmission" introduced in Section 1 fit into this field of application. For details about the Braille Printer and Automotive Transmission models, as well as the analysis of *fault* states, see Figure 2 and Figure 3.

In general, this analysis can serve as a basis to estimate the system's reliability. In principle, when assigning a probability to each *fault*, one could calculate the overall system failure probability. This could serve as a simulation-based counterpart to graph-based fault tree analysis - which is also possible in SimulationX. Since literature (Bertsche et al., 2009) and customer feedback suggest that these probabilities are in general hard to procure, we have not developed this analysis any further.

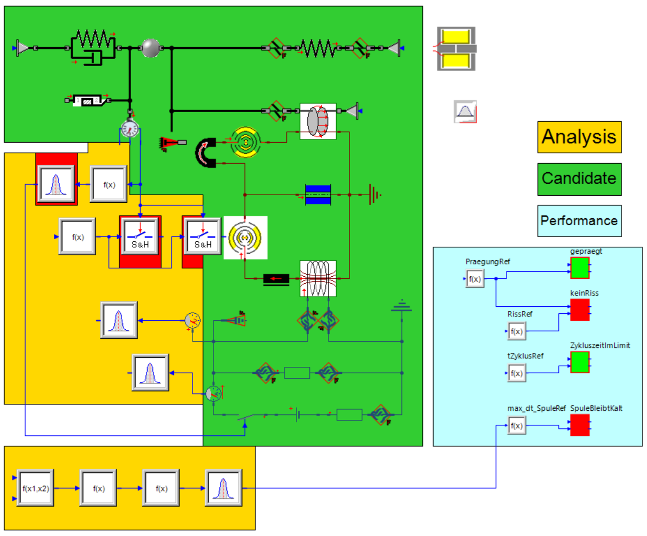


Figure 2. Screenshot of the SimulationX model of a Braille Printer (Kamusella, 2016). The components in the model are categorized - different categories are highlighted by color: green represents the physical components which are subject to faults. Mechanical connector faults (slipping, broken) and faults in the electrical (short to ground, open contact) connections are considered. The yellow area highlights components used for the analysis of the model, which are helpers for the components in the blue area. With the latter the performance of the system is evaluated. From top to bottom the performance indicators (cf. Sec. 6.2) check whether: the paper gets embossed, the paper is not pushed through, the needle is back to its initial position in time, and the magnetic part is not overheated. The fulfillment (green) or violation (red) of a performance criterion in the current simulation are displayed explicitly in the diagram.

4.2 Virtual Testing

The application "virtual testing" refers to model-in-the-loop, software-in-the-loop or hardware-in-the-loop (HiL) tests. In this scenario, the functionality of a controller is investigated through its connection to a model of the physical system. The model or code of the controller has to be able to handle nominal and non-nominal behavior. The latter can manifest itself in a multiplicity of ways - a high amount of modeling/coding effort is devoted to this subject (Plummer, 2007).

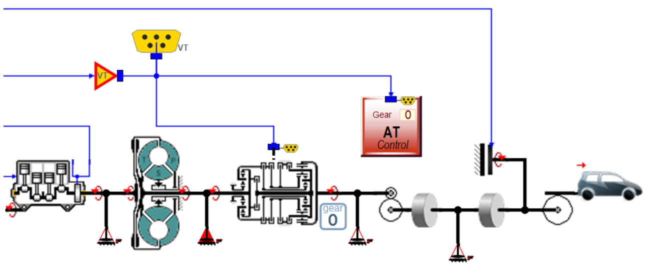


Figure 3. Screenshot of a SimulationX model of an automatic transmission. In the mechanical connections faults adding friction are included. One fault - between converter and transmission - is active (marked in red).

By inserting *faults* from the Fault library into the physical (SimulationX) model of the surrounding system, a large number of test cases for the controller can be generated. The number of *fault* combinations can likewise be very large, hence a semi-automatic algorithm can be used to insert the *faults* (cf. Sec. 5). Since *faults* can arise abruptly during simulation it is necessary that the *fault* intensity/active(-ity) are dynamic variables instead of parameters.

The example "Battery Package" introduced in Section 1 is subject to various *faults* and their combinations. Depending on the surrounding system, it is important to be able to detect the *fault* and behave according to the mechanical, electrical and environmental restrictions for the safe operation of the battery. For details about the Battery Package model see Figure 4. One can easily imagine that the exploration space of *fault* combinations is very large, especially when the option of activating/deactivating *faults* during runtime is included - therefore efficient ways of checking combinations and planning further tests have to be investigated (Tatar and Mauss, 2014; Ruiz et al., 2018).

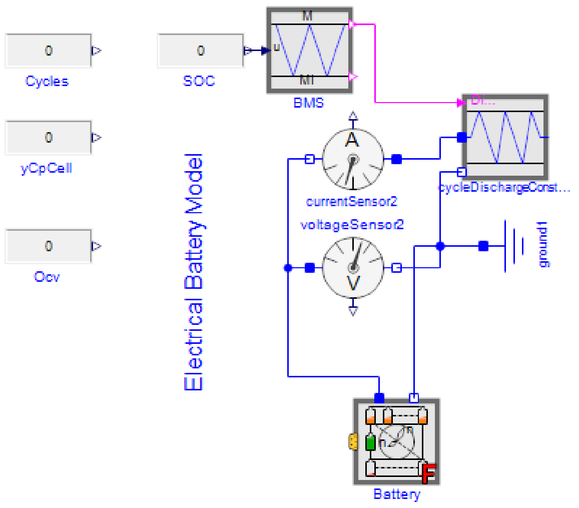


Figure 4. Screenshot of the (generated) faulty model for battery management system tests. The component "Battery" modeled in the battery package contains integrated electrical and thermal faults as well as their combination e.g. because of balancing failure. The battery package parameters are used in HiL for virtual parametrization and management system testing of the battery.

4.3 Diagnosis

The third field of application is concerned with the prediction and identification of (*faults*) in a system. We envisage the case for the non-nominal behavior of a real physical system. Sometimes measurement data from the real system (especially for the non-nominal behavior) is not available or insufficient. This could be because the system is still under development, or because it is a subsystem physically encapsulated by other components and hence inaccessible. In this case the nominal model of the system can be extended to include the non-nominal behavior by including *faults* from the Fault library - but also specific own *fault* models (Ishibashi et al., 2017). This model can be used as a basis for diagnosis on the model itself or only on the (arbitrarily large amount) of data produced by simulations of the model.

As mentioned above, data acquisition in real systems is generally limited, in particular due to a lack of historical data and measurable signals values (Dürr, 2016). In the case where insufficient data from the real system is available, the model may be used to complete the necessary data. In this case, the developed model describes some *faults* behavior of a real system with sufficient accuracy and the subject of validation can be used in place of field tests for generating reference data with various *faults* and time durations. In this way, the produced data allows an appropriate machine learning algorithm to analyze and diagnose *faults* and to identify appropriate *performance indicators* for different *faults* and their combinations.

The model "Feed Axis" in Figure 5 can be used to obtain some of the unavailable signals of the feed drive for the purposes of *fault* diagnosis. Moreover, combining *fault* modeling with the data acquisition of signals in the bus system enables the use of supervised machine learning algorithms for signal *faults* diagnosis. The feature analysis of signals and *faults* with various classification algorithms such as support vector machine or decision tree offers the possibility to choose an appropriate algorithm and to define *performance indicators* for each kind of *fault*.

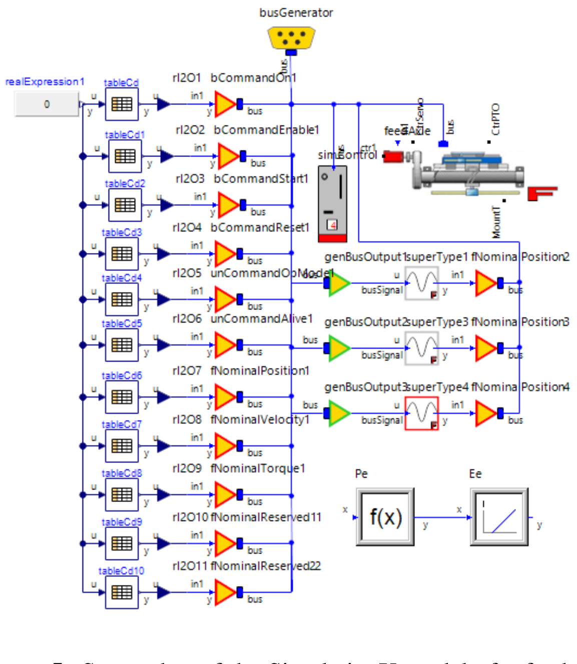


Figure 5. Screenshot of the SimulationX model of a feed axis with faults. The model consists of a feed axis model with servomotor and its control system. The fault signals causes a loss of accuracy and drift in the actual current and position signals in the control in-the-loop.

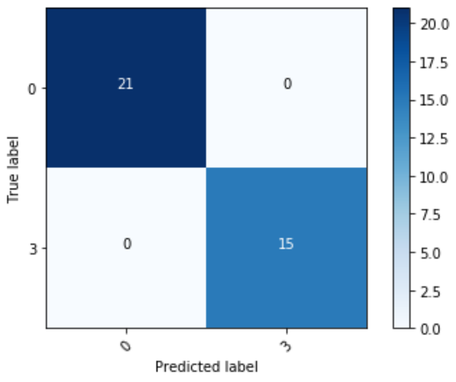


Figure 6. Confusion matrix of faults with decision tree classifier without normalization. The matrix shows identification of 15 faults from 36 test cases and confirms the feasibility of the algorithm configuration.

For example the confusion matrix in Figure 6 shows the identification of drift *faults* in the control system of a feed axis by using the CART algorithm with the Gini impurity metrics. The data is shuffled and split, with 75 percent of it being used for training and 25 percent for testing. The confusion matrix confirms the quality of the chosen parameters in the decision tree for the signal *fault* classification. The decision tree depicted in Figure 7 extracts the importance of features for the signal *faults* identification. As shown in the decision tree, the signal *faults* cause changes in the intensities of the torque and the active power of the feed axis drive which can hence be used as key *performance indicators*.

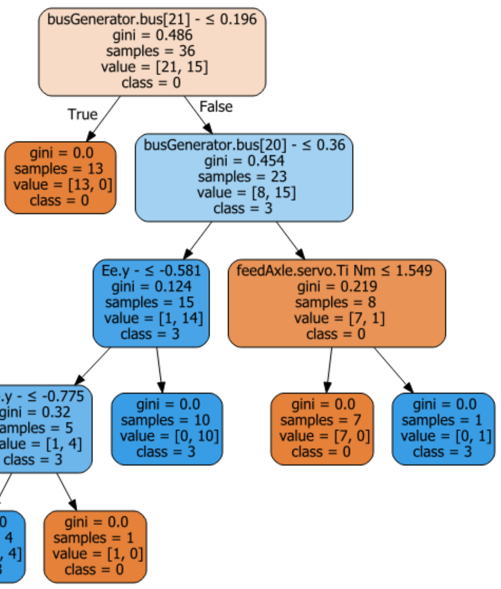


Figure 7. Decision tree of faults diagnosis from bus signals of the feed axis. The decision three shows the key performance indicators for the fault diagnosis in the decision tree - energy consumption, active power, motor torque, etc.

5 Fault Insertion - the FaultAugmenter in SimulationX

Augmenting a model of the nominal system with *faults* can be time-consuming and error-prone. For example, when replacing the nominal type of a component by its fault-augmented counterpart, one has to ensure that the parameters are kept or modified correctly.

To support the augmentation, an AddIn to SimulationX has been developed. A wizard guides the user through the augmentation process. Some input is needed to avoid including too many *faults* - which is helpful for the following analysis. The supported augmentation contains the following steps:

Definition of the candidate: The user has to define the *candidate*, i.e. all those components and connections in the model, which are to be augmented (at most) by drag-and-dropping to a list (Figure 8). Additionally or alternatively he/she can decide whether to exclude or include the augmentation of components (Figure 9 on the left), or the insertion of ConnectorFaults only into connections of specific domains (mechanical, electrical,...).

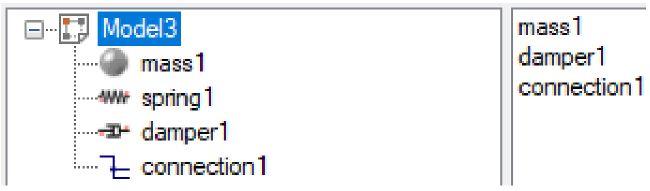


Figure 8. Choosing the components and connections for the fault augmentation with FaultAugmenterAddIn.

Augmentation: The actual augmentation can be started by clicking the button "Augment"- the model structure is changed as shown in Figure 9 on the right. Components have been replaced by their fault-augmented counterparts. To each connected connector a *fault* with two pins ("broken") was added, and one ConnectorFault with one pin ("sticking") was added to the whole connection.

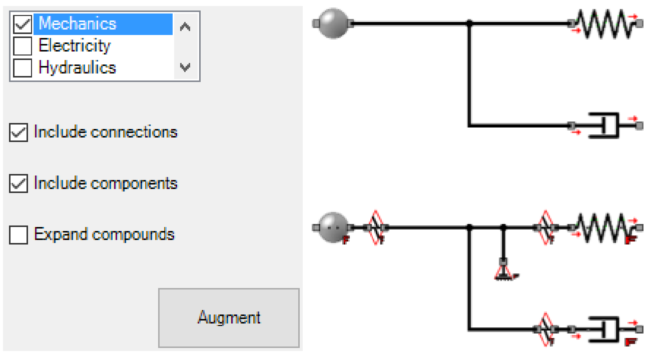


Figure 9. SimulationX model before and after the automatic fault augmentation (right) for mechanical components supported by the FaultAugmenter AddIn (left).

Scaling: The FaultAugmenter AddIn allows to set the `scale` parameter of groups of *faults* or individual *faults* to a certain value (cf. Sec. 3.3). If, for example, the mechanical part of the model represents heavy machinery, the scale in all mechanical connections can be increased by the same order of magnitude. The insertion of BridgeFaults is not conducted by the FaultAugmenter, since on the graph the number of possible additional connections is very high, but BridgeFaults only make sense in very few places. Hence, the user has to insert BridgeFaults manually.

6 Analyzing Faults

As well as structuring the modeling and insertion of *faults*, it is also helpful to structure and systematize the output of the model of interest. In all three fields of application some output quantities of the model are analyzed. However, the goals of the analyses in the different applications are quite different. The Modelica packages presented in this section serve as helpers for these analyses.

6.1 Extracting Features

6.1.1 What for?

In the field of diagnosis the model output is compared to real systems' data. Such data is, in general, sampled, can sometimes be averaged, noisy, or even in frequency domain representation. To prepare the model output in the same way, one needs ways to extract features from the variables.

To avoid dealing with (different) sampling rates and the quality of time series, the diagnosis algorithms based on data presented in Sec. 4.3 - support vector machines, decision trees, etc. - use single-valued data only. To produce such data from output time series, meaningful features have to be extracted.

Furthermore, in the field of reliability system performance is categorized by blocks returning a single value - successful, failed, or undecided/not clear. The determination of these categories (for more details see next subsection) is based on calculations of output variables, which are again features of the latter.

6.1.2 Technical Details

The `Features` package contains helper blocks that support feature extraction. Helper blocks are provided for extracting features such as minimum, maximum, mean, variance, the FFT, or short-term-mean. The list is extended based on the examples studied. One important requirement is that all features are insensitive to numerical side effects. For example, the extraction of the maximum of a variable should not pick a "numerical" peak, the height of which is dependent on tolerance or other numerical artefacts. The extraction of mean values should be possible over restricted time spans to avoid a dependency on the overall simulation time (e.g. the average velocity of the car decreases to zero, because the model driving scenario depicts an unnecessary amount of time span after the car has come to rest).

The additional sub-packages `ChecksInFixedWindow`, `ChecksInSlidingWindow`, `SignalAnalysis` from (Otter et al., 2015) - although motivated by a completely different application - serve similar purposes.

6.2 Measuring System Performance

The Braille printer and automotive transmission examples address the question of the system performance when it is subject to *faults*. Stated differently: which *fault* or combination of *faults* leads to the violation of pre-defined criteria. Such a criterion, taken from the transmission example, is: The car shall be able to accelerate from 0 to 100 km/h in 5 Seconds. To assess this in the model, output variables (velocity, time) are read, features are extracted (velocity at 5 seconds after startup) and tested against a criterion (larger than 100 km/h). The formalization of the last step is supported by the elements modeled in the package `PerformanceIndicators`.

The basic definition of a performance indicator contains an array of `assertions` as an input. In the example of the velocity test this array has one entry: $v(t_{Startup}+5)-100[km/h]$. If this entry is greater than zero, the criterion is fulfilled, if not, it is violated. If the array of the assertions contains more than one entry, it has to be defined whether they are connected by an AND (i.e. all must be fulfilled to fulfill the whole criterion) or an OR (i.e. only one must be fulfilled). Sometimes it does not make sense to test a criterion at all - for example, if there was no ignition, there is no startup time. To address this scenario, the performance indicator contains a second input named `validityIndicator` defined in a similar way as the `assertion` - i.e. if this variable is positive, the validity is given and the assertion can be tested, if not, the assertion does not need to be tested. The integer output `perfInd` is based on the validity indicator and the assertions, and is restricted to three values, which represent the categories as listed in Table 1.

For convenience, the assertions are fed to an output variable `assertionsOut`. The output `perfInd` can only tell if the simulation fulfilled or violated the criterion, but not *how far* it was from violating or fulfilling it. To have a measure for this, the relevant continuous variables should be analyzed. Sometimes it is important to have information about how close to violating/fulfilling the specification, e.g. since due to noises/variations which are not in the model the outcome might not be robust. Furthermore, it proves helpful to have some information about whether an increasing *fault* intensity has an effect on an assertion. This information cannot be obtained from the integer output.

Table 1. Categories of the output of the performance indicator based on the incoming validity and insertion.

perfInd	category	conditions
1	fulfilled	validity>0, assertion>0
0	violated	validity>0, assertion<0
-1	undecided	valididy<0

Figure 10 contains components of the `PerformanceIndicators` blocks. Currently, the connection of assertions via AND, OR and NOT is possible, since any logical expression can be brought into either of these forms (disjunctive/conjunctive normal form, see e.g. (Hazewinkel, 1994)). More flexible definitions of performance indicators will be developed as applications demand.

The Modelica_Requirements library presented in (Otter et al., 2015) contains a similar 3-valued logic to those presented here. Its motivation comes from a connection between system simulation and the formal definition of requirements. For the Modelica_Requirements library an extension of the Modelica language is proposed to handle the 3-valued temporal logic (satisfied, violated, undecided). The formalization of proving the logical expressions built up from validity and assertions (as described above) could be improved if the handling of the 3-valued logic becomes part of the language standard. However, it is not necessary in our case.

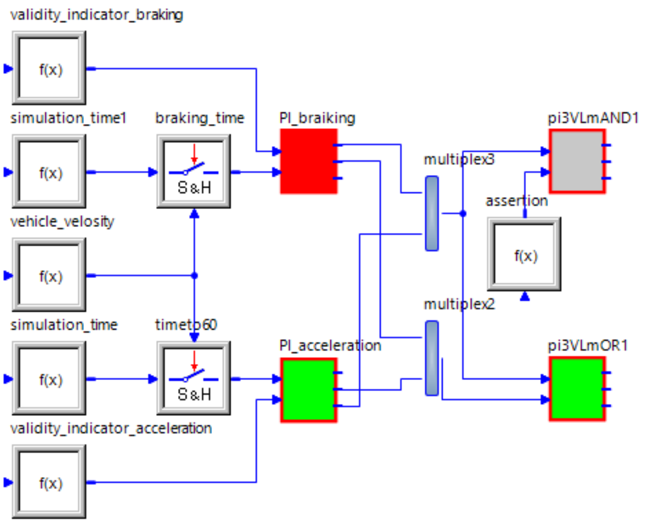


Figure 10. Screenshot of PerformanceIndicators in the SimulationX model of the automatic transmission. The connection of assertion via AND and OR allows to check whether: the car shall be able to accelerate from 0 to 100 km/h in 5 Seconds and to brake from highway speed to stop in 4 Seconds. The fulfillment (green), violation (red) or undecided (grey) states of a performance criterion are displayed.

7 Conclusion

In this publication, we introduced the Fault library , which enables the user to model and simulate physical systems outside their nominal behavior in a systematic way. We motivated the utility of the Fault library with four examples having very different analysis goals. Based on these, the broad range of applicability was outlined. Preliminary results - based on the given examples - were presented. In addition, the necessity and implementation of helper libraries (Features, PerformanceIndicators) and wizards (FaultAugmenter AddIn) was described.

Acknowledgments

Part of the work lies in the scope of the Romesa project (Robustness and Reliability Simulation of Mechatronic Systems including Aging and Wear (BMBF, 2015)). The authors would like to thank the German Federal Ministry of Education and Research (BMBF) represented by the project coordinator German Aerospace Center (DLR) for supporting financially. Furthermore, thanks are due to Palo Alto Research Center for their development and input on the Fault library, which is based on the FAME library (de Kleer et al., 2013). Thanks go also to Alfred Kamusella, who provided the model of the Braille Printer that served as a basis for the results reported in Section 4 (cf. also Figure 2).

References

Bernd Bertsche, Peter Göhner, Uwe Jensen, Wolfgang Schinköthe, and Hans-Joachim Wunderlich. *Zuverlässigkeit mechatronischer Systeme: Grundlagen und Bewertung in frühen Entwicklungsphasen*. Springer-Verlag, 2009.

Federal Ministry of Education and Research BMBF. *Projektblatt Romesa*, 2015. URL `http://www.pt-sw.de/media/content/Projektblatt_ROMESA.pdf`.

Johan de Kleer, Bill Janssen, Daniel G. Bobrow, Tolga Kurtoglu, Bhaskar Saha, Nicholas R. Moore, and Saravan Sutharshana. Fault augmented modelica models. In *24th International Workshop on Priciples of Diagnosis*, pages 71–78, Jerusalem,Israel, 2013.

Hans Dürr. In *New Validation Method for Models for Grid Studies*, pages 1–6. Utility Variable Generation Integration Group, 2016. URL `https://www.uvig.org/resources/model-validation-workshop/`.

Christian Friedrich, Salim Chaker, Christoph Schramm, and Steffen Ihlenfeldt. Generic feed-axis library for machine tools. In *ESI SimulationX User Forum 2016, Dresden, Germany, November 24-25, 2016*, pages 134–143, 2016.

Michiel Hazewinkel. *Encyclopaedia of Mathematics (set)*. Encyclopaedia of Mathematics. Springer Netherlands, 1994. ISBN 9781556080104. URL `https://books.google.de/books?id=uxUBQwAACAAJ`.

Friedrich Herrmann. Eine analogie zwischen mechanik, elektrizitätslehre, wärmelehre und stofflehre. *PdN PhiS*, 55(2): 2–5, 2006.

Tatsuro Ishibashi, Bing Han, and Tadao Kawai. Rotating machinery library for diagnosis. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 381–387. Linköping University Electronic Press, 2017.

Alfred Kamusella. SimulationX model of a braille printer, 2016. URL `https://www.ifte.de/lehre/optimierung/uebung.html`.

Raj Minhas, Johan de Kleer, Ion Matei, Bhaskar Saha, Bill Janssen, Daniel G. Bobrow, and Tolga Kurtoglu. Using fault augmented modelica models for diagnostics. In *Proceedings of the 10th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 96, pages 437–445. Linköping University Electronic Press; Linköpings universitet, 2014.

Martin Otter, Nguyen Thuy, Daniel Bouskela, Lena Buffoni, Hilding Elmqvist, Peter Fritzson, Alfredo Garro, Audrey Jardin, Hans Olsson, Maxime Payelleville, et al. Formal requirements modeling for simulation-based verification. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 625–635. Linköping University Electronic Press, 2015.

Andrew R. Plummer. Control techniques for structural testing: A review. *Journal of Systems and Control Engineering*, pages 139–169, 2007.

V. Ruiz, A. Pfrang, A. Kriston, N. Omar, P. Van den Bossche, and L. Boon-Brett. A review of international abuse testing standards and regulations for lithium ion batteries in electric and hybrid electric vehicles. *Renewable and Sustainable Energy Reviews*, pages 1427–1452, 2018.

Mugur Tatar and Jakob Mauss. Systematic test and validation of complex embedded systems. *ERTS-2014, Toulouse*, pages 05–07, 2014.

Franciscus L. J. van der Linden. General fault triggering architecture to trigger model faults in modelica using a standardized blockset. In *10th International Modelica conference*, number 96 in Linköping Electronic Conference Proceedings, pages 427–436. LiU Electronic Press, 3 2014. URL `http://elib.dlr.de/90576/`.

Yinjiao Xing, Eden WM Ma, Kwok L Tsui, and Michael Pecht. Battery management systems in electric and hybrid vehicles. *Energies*, 4(11):1840–1857, 2011.

[Industrial paper] Application for Optimization of Control Parameters for Multi-body and Hydraulics System by using FMU

Nobumasa Ishida[1] Hideyuki Muramatsu[2]

[1]Dassault Systèms K.K., Japan, Nobumasa.ISHIDA@3ds.com
[2]Dassault Systèms K.K., Japan, Hideyuki.MURAMATSU@3ds.com

Abstract

We developed a simulation environment using FMU (Function Mockup Unit) as performance prediction method in the early stage of development.

In this environment, hydraulic controller parts are modeled in Dymola which is one of 1D simulation tools based on Modelica language and controlled mechanics parts are modeled in Simpack which is multibody simulation tool. We created a total system model connected to hydraulic controller parts and mechanics parts by using FMI (Function Mockup Interface). Hydraulics and controller parts are converted to FMU by Dymola, and are incorporated in Simpack mechanical model.

Furthermore, we applied the system model optimization to control parameters by using Isight.

Keywords: 1D-Simulation、*Multibody*、*Optimization*、*FMU*、*Dymola*、*Simpack*、*Isight*

1 Introduction

Recently, in the manufacturing industry, shortening the development cycle and reducing the cost of product development has an important issue, so that it is required to develop an efficient method for the development.

In the product development cycle before, prototyping was essential for production and testing to performance review. Nowadays, though, thanks to improvement of simulation accuracy and calculation environment, simulation can work as virtual testing and performance review, which leads to the reduction of trial production and physical testing cost.

However, as high precision simulation needs a lot of detailed physical properties and detailed FE model, it is difficult to apply this approach to the early stage of development in terms of complexity of model creation and calculation cost.

Further, products have been more complex system than before including many components and controller. Therefore, we need to consider whole system level to evaluate their performance. However, there are many different simulation tools that are used in each component department and parts supplier.

When creating whole system model, each modeling tool has their own interface to connect other modeling tools. Since the interface has a specification, it takes time and effort to learn and create whole system model.

1D modeling tools like Dymola has a variety of libraries such as controller, hydraulics, electrics and so forth, and it is possible to create simple holistic system model. Simulation with simple system model is suitable for the early phase of development and reducing cost.

FMI is a generic interface to connect different modeling tools that provides more flexible simulation environment. In addition, FMI is useful for model transfer between parts supplier and OEM or across different simulation domains and it is promising to spread widely in future.

In this paper, we built Excavator model consisting of hydraulic controller and mechanics parts. We used Simpack to model mechanics parts and Dymola for hydraulic controller. We generated FMU on hydraulic controller from Dymola and coupled it with Simpack, via both Co-simulation and Model Exchange to see the difference of those two approaches in light of simulation accuracy as well as calculation time. Then we developed an optimization workflow with Isight to determine controller parameters to see if the approach is effective for the early conceptual phase of product development.

2 Excavator Model

Figure 1 shows structure of Excavator system model.

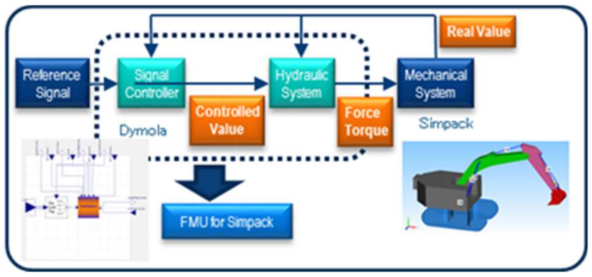


Figure 1. Structure of excavator system model

The Excavator model consists of two parts, mechanics and hydraulic controller. Mechanics parts are constructed by Multi-body simulation software Simpack and hydraulics controller is created with 1D Simulation tool Dymola.

Simpack and Dymola are connected by FMI. Simpack outputs real value of each actuator cylinder length, and Dymola outputs force and torque value of each actuator through FMI. We developed this Excavator model based on publicly available OpenHydraulics by Georgia Institute of Technology.

2.1 Vehicle Model

Figure 2 shows the mechanics parts model.

The model is constructed by Simpack. All parts (Base, Carriage, Boom, LinkArm, and Bucket) are defined as Rigid Body. The Rigid body has a mass, inertia, center of gravity position.

Each body parts have degrees of freedom on connecting point by Joints and Constraints element in Simpack.

Each actuator is modelled by Force element in Simpack. Actuator between Base and Carriage is defined as torque around normal direction (y). Cylinder actuator of Boom, LinkArm, and Bucket are defined as translational force. In addition, Boom is equipped with two actuators on the left and right. All value of actuator force and torque are provided from hydraulic controller in Dymola via FMI.

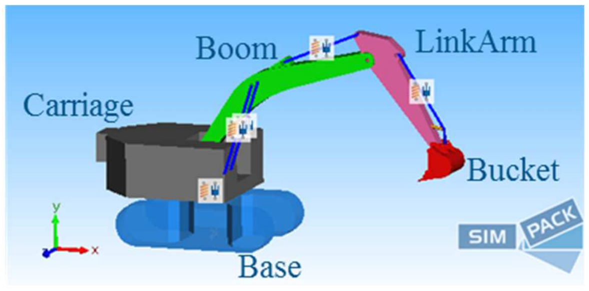


Figure 1. Vehicle Model (Simpack)

2.2 Hydraulics model

Figure 3 and Figure 4 shows the hydraulic and controller model modeled in Dymola.

Hydraulic consist of Power Unit, Valve Unit and cylinder model. Controller gives control values to Valve Unit.

Controller model shown in Figure 4 gives control value calculated by PI control differential of real cylinder length and target length to Valve unit.

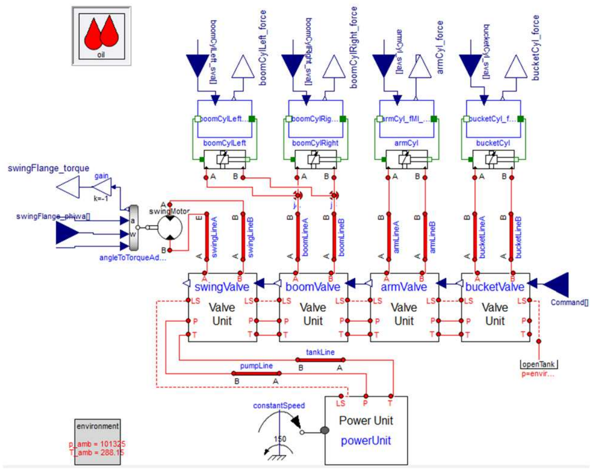


Figure 3. Hydraulics Model (Dymola)

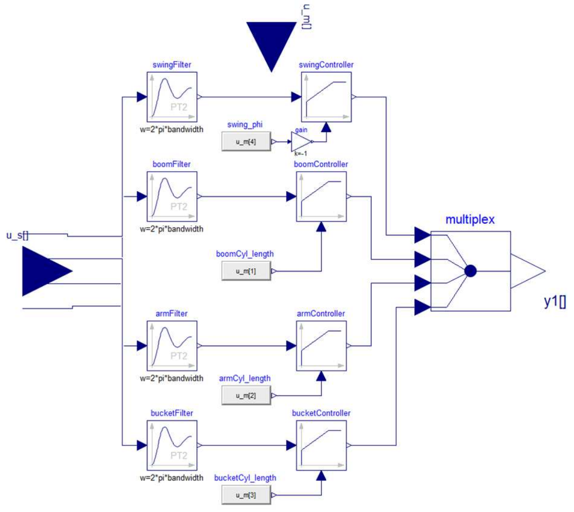


Figure 4. Controller Model (Dymola)

To simulate whole Excavator system creating integrate model using FMI, to connect mechanics and hydraulics controller models created in different modeling tools.

In this case, hydraulic controller model is converted to FMU in Dymola and integrate it to Simpack model. We used version 2.0 of FMU.

3 Comparison of Co-Sim and ME

There are two types of FMU, Co - Simulation and Model Exchange, which use different solvers. Co - Simulation uses solver included in each FMU, and sets communication time between tools/models and execute iteration step for each. In this case, the communication time was set to 0.1 ms in Co-Simulation. On the other hand, in Model Exchange FMU contains only models and not the solver, and the calculation is done by the solver on the simulation platform. In our case, Co-Simulation executes both Dymola and Simpack solver and Model Exchange executes only Simpack solver.

Optimization calculation requires a lot of iteration, so, it is desirable to have shorter simulation time. To see the impact of two different FMU modes onto the calculation time, we executed both Co-Simulation and Model Exchange FMU cases and compared the calculation result and calculation time.

As simulation scenario, we defined following steps shown as Figure 5.

1) Vehicle speed is 0 at all steps.
2) Extend Boom and LinkArm
3) Scoop sand
4) Lift Bucket
5) Turn Carriage
6) Dump sand

We simulate only the behavior of the working machine and do not consider the behavior of sand and ground.

In order to perform this series of operations, we set target cylinder length as time domain data, and gave force via feedback control. Figure 6 shows the comparison results between Co-Simulation and Model Exchange taking Boom cylinder length, and Figure 7 takes the difference of Boom cylinder length between two FMU approaches.

From Figure 6, we see that the results of Co-Simulation and Model Exchange are almost same. Figure 7 shows the difference between these two and it is smaller than the actual values by three orders of magnitude so we can consider that these two FMU approaches brought almost the same result.

In Figure 6, there is a difference between the target value and the calculation result. However, in this case, since temporary control parameters are used so accuracy of target value was not reviewed.

Regarding comparison of the computation time in Table 1, calculation time of Model Exchange was 1/20 from Co-Simulation. Therefore, we decided using Model Exchange for the subsequent optimization calculation.

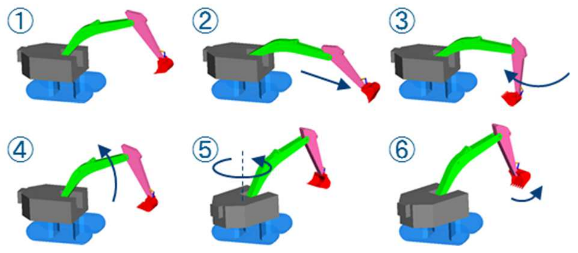


Figure 5. Calculation step

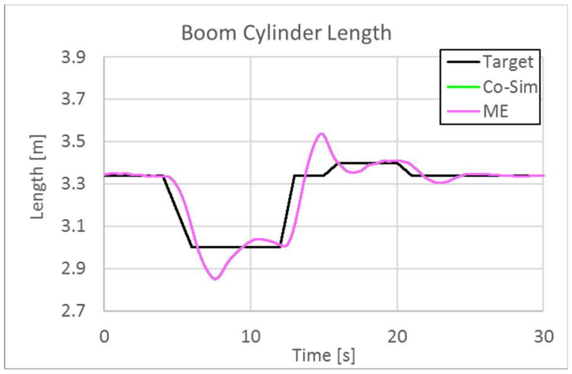


Figure 6. Boom Cylinder length (Co-sim vs ME)

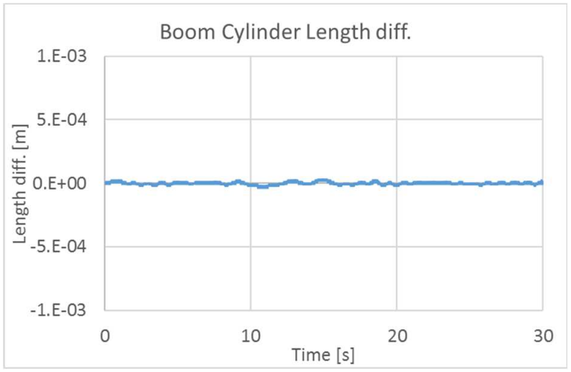


Figure 7. Difference of Boom Cylinder length between Co-simulation and model Exchange.

Table 1. Sizes of Compiler Phases, Lines of Code.

FMU Type	Calculation Time	CPU Time
Co-Sim	46m42s	95.09s
ME	1m43s	28.46s

※CPU : Intel®Core™ i7 2.7GHz、Momory:16.0GB

In the comparison of CPU Time, there is no difference as much as calculation time, so we consider that Co-Simulation took longer calculation time than

Model Exchange because the communication time between tools is an important factor.

4 Control Parameters Optimization

We use the coupled model using FMU to explorer design parameters for control.

In this study, as an example of the control robustness against the difference in the operating environment of the working machine, we optimized control parameters for two cases with different Bucket mass parameters.

The operating conditions are the same as the series of operations shown in Figure 5.

Table 2 outlines the optimization calculation.

Table 2. Optimization Outline

Bucket Condition	・Payload off ・Payload 50kg
Design variable	Controller gain and time constant for each cylinder （Total 6 parameter）
Objective	Minimize cylinder length error ε

We set the error ε between the objective and the target value of cylinder length as following equation.

$$\varepsilon = \sqrt{\frac{1}{n-1}\sum_{n=1}^{n}\left(y_{real} - y_{obj}\right)^2}$$

In the design exploration, we ran DOE calculation at the beginning to create an approximate model from the result. DOE was performed by Latin Hypercube, and approximate model was created using RBF (Radial Basis Function). Optimization was carried out to minimize the error defined by the equation above using the created approximate model. The optimization method was NCGA (Neighborhood Cultivation Genetic Algorithm). These workflows were created using Isight. Figure 8 shows the outline of the design search.

Figure9 shows the comparison of approximate model and actual model. The approximate model is created based on DOE results by RBF method. The result of approximate model is in good agreement with the result of actual model, so the approximate model can be applied to optimization.

Figure 10 to Figure 13 show the calculation results.

Figure 10 and Figure 11 show the time domain data of the difference with respect to the target value of the Boom cylinder length, and Figure 12 and Figure 13 show the time domain data of the target value and the calculated cylinder length.

As can be seen from Figure 10 and Figure 11, the error with respect to the target value becomes smaller by using the optimized parameters.

We can also confirm the effectiveness of the optimization by comparing the cylinder lengths in Figure 12 and Figure 13.

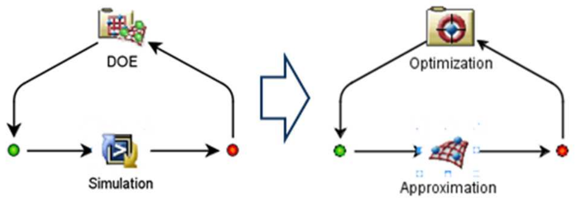


Figure 8. Workflow of optimization in Isight

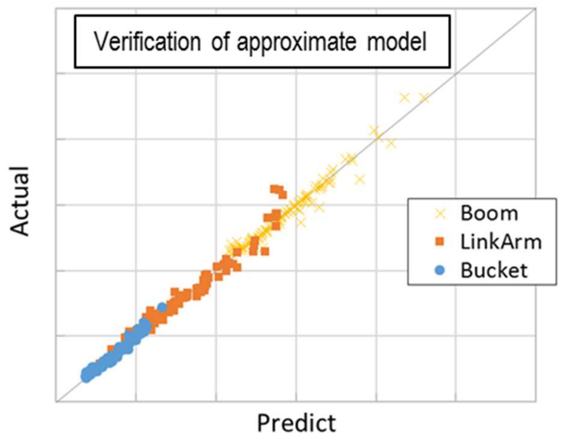


Figure 9. Verification result of approximate model

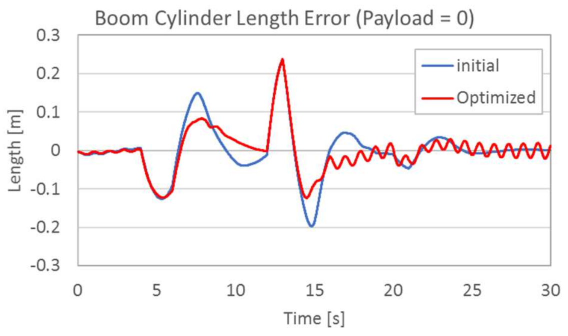


Figure 10. Boom Length Error (Payload off)

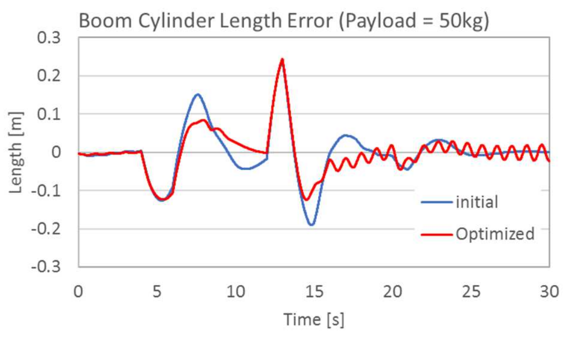


Figure 11. Boom Length Error (Payload 50kg)

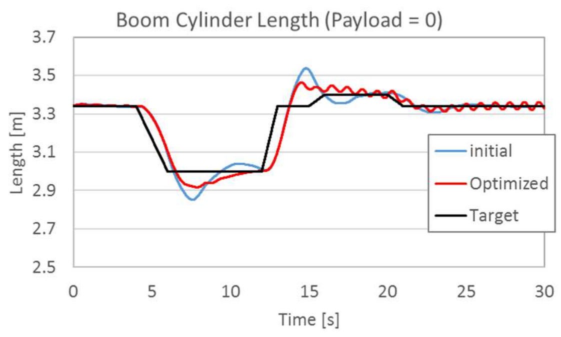


Figure 12. Boom Length (Payload off)

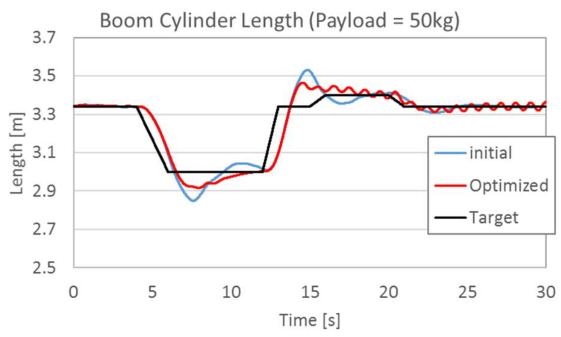


Figure 13. Boom Length (Payload 50kg)

5 Conclusion

For a construction machine, we modeled a system including hydraulic controller with 1D simulation tool and mechanism analysis software and created an integrated simulation model using FMU.

In this case, it turned out that FMU Model Exchange worked efficiently for the optimization of control parameters.

In this study, we performed design space exploration on control parameters, but we think that it can be applied not only to control parameters but also to dimensioning components such as the shape of structural parts.

As a conclusion, we consider that the use of the system model with FMU is effective for the design search at the early stage of development where we need to evaluate systems performance coupling different disciplines efficiently.

Acknowledgements

We gratefully acknowledge the support of Mr. Youngwon HAHN from Dassault Systemes Simulia Corp for his great cooperation in technical advice and model creation for this thesis.

References

Dassault Systems, Dymola User Manual Volume1,2 Version 2018 (2017-04-10)

Dassault Systemes Simulia Corp., Simpack Documentation Version 2018

Georgia Institute of Technology, OpenHydraulics v1.0.1 (2013-02-26) *https://github.com/cparedis/OpenHydraulics*

Analysis of Lift-Generating Disk Type Blade Wind Power System Using Modelica

Yeongmin Yoo[1] Soyoung Lee[1] Jaehyun Yoon[1] Jongsoo Lee[1]

[1]Department of Mechanical Engineering, Yonsei University, Seoul, Korea,
{yym9514, leesoy77, yunjh00, jleej}@yonsei.ac.kr

Abstract

The research and application technology of wind power generation have attracted great attention in relation to the development of renewable energy. As wind power systems such as a propeller type are being enlarged to increase power output, various problems such as natural landscaping damage and shadow problems are occurring. In this paper, we propose a lift-generating disk type blade power generation mechanism that can effectively generate wind power even with simple structure considering the problems of existing system. The modeling method of the wind power system is explained using the Modelica. After that, a wind tunnel test is conducted through a small scale model of the disk type blade created for simulation verification. As a result, the modeling of the wind power system is verified and the results of the generator power output are presented through simulation.

Keywords: Wind Power System, Disk Type Blade, Multi-Physics System, Integrated Simulation

1 Introduction

Wind power technology is one of the various methods of obtaining electricity from natural energy and has received much attention because it can use infinite resources for free. Thus, wind energy is expected to play a decisive role in the future world energy supply. To produce wind power system (WPS), efficiency of development is required to lower development cost. With this objective, computer simulation technology is used in all development fields. Therefore, the simulation technique is a very important means for streamlining development work. The WPS to be designed includes various physical phenomena such as mechanical dynamics, electricity, control and flow, and a program suitable for the required integrated simulation of the system. Typically, different development departments often use different software because they deal with different physical phenomena. Thus, a dedicated analysis software must be employed for detailed analysis. However, it is difficult for this design method to effectively cope with frequent design changes occurring in the initial conceptual design, which results in unnecessary work and, reduction in efficiency. In the initial design phase, it is necessary to use integrated simulation software because it is important to communicate, coordinate, and cross check between concept verification and development personnel rather than conduct detailed analysis.

For this reason, the Modelica language is adopted for integrated simulation. So far, various researches on WPS using Modelica have continued. Strobel et al. (Strobel, 2011) proposed a Modelica library by designing offshore WPS and verifying simulation results. Petersson et al. (Petersson, 2012) presented a mathematical modeling of a vertical axis WPS and presented simulation results accordingly. Eberhart et al. (Eberhart, 2015) constructed an open source library for the simulation of the horizontal axis WPS.

In this study, a new type of WPS utilizing Modelica is designed based on these studies. The conventional WPS is called a horizontal-axis-type WPS because the rotation axis of the blade is placed horizontally. These types of blades have characteristics suitable for high-speed rotation, but various problems, such as modification of landscapes for natural environment, and social pressure issues because of the shadows cast by the structures are raised. Therefore, it is the aim of the authors to design a lift-generating disk type WPS that can generate electricity in response to an omni directional wind instead of a conventional WPS. The disk type blades were first designed using computer-aided design (CAD), and then they were designed according to design parameters. We designed the WPS by applying the designed blade to Modelica simulation. Wind tunnel test were performed for simulation verification and the results were in good agreement. Finally, we designed the Modelica simulation to derive the generator power output of the disk type blade WPS.

2 System Mechanism

The proposed WPS is shown in Figure 1. The system in which the blade is disk type and power is generated while it moves along a vertical direction. The basic principle of the system is that vertically supported tower structures from the ground move along the vertical axis with the disk type blade lifted by the wind. This type of system generates electric energy through a

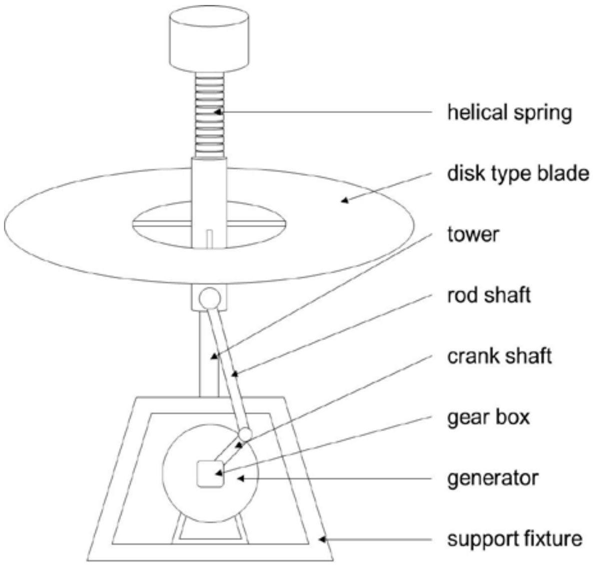


Figure 1. Schematic diagram of the lift-generating disk type wind power generation system.

power transmission system (PTS) that converts such motion into rotary motion along with a power conversion system (PCS), such as a generator. This model is very different from typical WPS, such as conventional horizontal or vertical axis turbines. The proposed WPS is advantageous for areas in which flow-induced noise and blade flickering/shadows become critical. The small scale WPS developed by the authors includes a disk type blade, spring for managing vibration generated by the vertical movement of the blade, crank-rod mechanism (Khemili, 2008) for converting vertical motion into rotational motion, and tower for installing a system. Thus, through this PTS, power is produced by the generator.

The proposed disk type blade has some distinct characteristics. First, the disk can move up and down to generate lift force that is converted into electrical energy. Second, owing to its symmetric shape, it is easily activated by the wind coming from all directions. Finally, the proposed model has a relatively weak contibution to shadow flickering because it does not feature multi-blade rotation and the space required for realizing its motion is comparatively small.

3 System Modeling

3.1 Shape Design of Disk Type Blade

The shape of the disk type blade to be designed using CAD is shown in Figure 2. The material used for the blades is polymer-type polycarbonate with a density of 1.12e^3 kg/m^3, Young's modulus was 2.3e^9 Pa, and Poisson's ratio of 0.33. The disk type blade is hollow, and its shell thickness is 10.3 mm. The shape was designed considering the force acting on the blade. The lift and drag are different depending on the shape. For instance, the longer the airfoil chord length of the blade

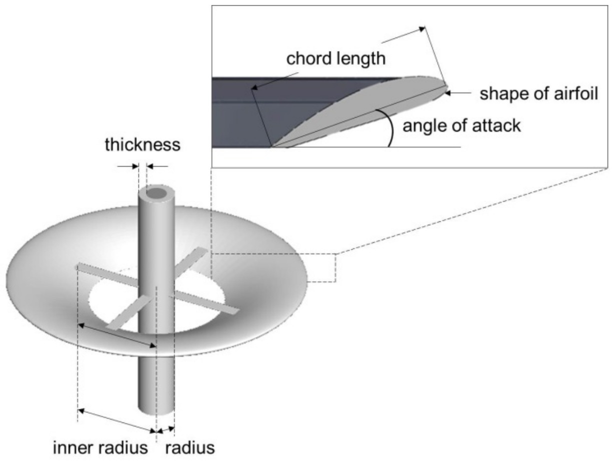


Figure 2. Geometric model and design parameters of a disk type blade.

is, the more lift can be generated. However, such a longer length increases the weight of the product, resulting in a non-economical design. Considering these conditions, analysis of parameters that affect the shape design was carried out in order to create a light-weight blade with minimal airfoil chord length. Additionally, the fixed variables and control variables to be changed were accordingly selected. As fixed variables, the radius of the center tower connected to the blade is 0.30 m, thickness is 0.05 m, and the inner radius between the tower and the disk type blade is 0.55 m. For the aforementioned fixed parameters, a total of 3 disk type blade parameters, such as the airfoil chord length, angle of attack (AOA), and NACA series airfoil cross-section shape of the blade are designated as design variables, as shown Table 1.

For each parameter, the three values represent the three levels considered when varying the design of the experiment. The chord lengths were 0.20 m, 0.45 m, and 0.70 m. The values of the AOA were 5°, 10°, and 15°. For the NACA series, the airfoil section was selected from NACA632615, NACA64A410, and NACA631412, and the airfoils were cambered. In this study, we used a central composite design (CCD) table that allowed the smallest number of numerical experiments.

Table 1. Design variables of a disk type blade.

	Factor	Level-1	Level-2	Level-3
x_1	chord length (m)	0.20	0.45	0.70
x_2	angle of attack (°)	5	10	15
x_3	shape of airfoil	632615	64A410	631412

In the CCD, the overall number of experiments was 15, wherein the value of α for each design variable was ±4.20 in x_1, and ±8.41 in x_2. Particularly in x_3 of the NACA series, -α was selected using the level-1 value of NACA632615, and +α was selected using the level-3 value of NACA631412. The CCD method was applied to derive the blade shape of 15 cases according to design variables (Montgomery, 2017).

3.2 Computational Fluid Dynamics Simulation

Flow simulation was conducted using ANSYS ICEM-CFD for computational fluid dynamics (CFD) simulations. The governing equations solved in the fluid domain were the incompressible Navier–Stokes equations with the Reynolds stress term. The Reynolds stress was treated as a shear gradient and eddy viscosity in the Boussinesq approximation. The inlet wind speed in the present study was 12 m/s, which is the standard speed for small scale WPS. When the projection diameter of the blade was set to be the characteristic length with this inlet velocity, the Reynolds number became 861,000–2,100,000 considering the wind speed, the disk diameter, and the air viscosity.

We built the tetrahedral mesh in the fluid domain. The unit size for this mesh was 0.012 m, and the ratio of the maximal size to the unit size was 20. A finer mesh was set near the disk type blade and tower, while a coarser mesh was set in the region that was far away from the disk type blade. There were 7 million mesh cells. The boundary conditions of the flow simulation were location-dependent. The inlet condition was Dirichlet’s boundary condition, according to which boundary nodes had a constant velocity. The outlet condition was the open boundary condition, which prescribes zero static pressure for no disturbance of the fluid motion by the boundaries. The boundary condition at the surface was the no-slip condition, according to which the flow velocity is zero. Other boundary conditions were related to some initial conditions. Figure 3 shows the CFD simulation results for the blade designed by the CCD method. From this result, we want to confirm the response of the system according to lift and drag values.

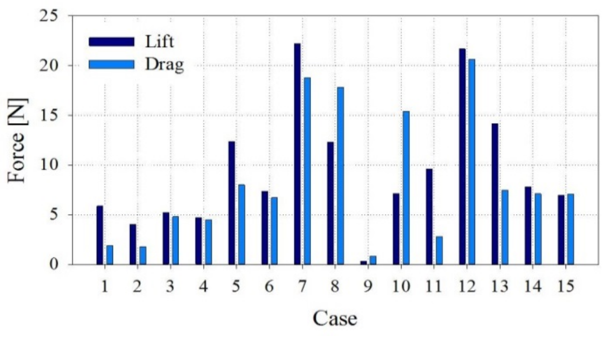


Figure 3. CFD simulation results using CCD method.

The case that generates the largest lift is Case 7, but the simulation is compared with several cases in order to confirm that the output of the WPS is the best. For example, we will compare the smallest lift case and the largest drag case.

3.3 Integrated Simulation using Modelica

Modelica is an object-oriented simulation language developed to simulate complex and large heterogeneous physics. It is more practical to implement simulation because it has an acausal relationship with the elements of the before and after steps even if the elements needing modifications are changed during the course of the simulation design through acausal programming (Mattsson, 1998). In order to conveniently model a system, such as a WPS, a study was conducted using a multibody system library that provides a three-dimensional mechanical component.

The two primary forces acting on the disk type blade are lift and drag. In order to simulate the lift-generating disk type WPS, the lift force in the y-axis direction and the drag in the x-axis direction were selected as input values. Each force acting on the blade was applied with a constant value and design was carried out with the force acting on the center of mass of the blade. Since the blade model is considered as a rigid body in Modelica, the mass, center of mass, and inertia tensor are selected as input variables. These data were applied as parameter values to MultiBody.Parts.Body. The MultiBody.Parts.BodyCylinder was selected as a cylindrical tower with a diameter of 0.25 m and height of 10 m. The disk type blade is a system that reciprocates vertically along the vertical axis and can not utilize the rotational force generated from the rotor shaft as the blades and rotors of existing WPS rotate. Therefore, a crank-rod mechanism is considered to convert the motion of the disk type blade into rotational motion. Table 2 shows the parameters for PTS and PCS mechanism design included in the WPS.

Table 2. Specifications for the PTS and PCS mechanism.

Component	Parameter	Value
crank	mass (kg)	3.08
	length (m)	1
rod	mass (kg)	9.24
	length (m)	3
gear box	gear ratio	5
generator	moment of inertia ($kg \cdot m^2$)	0.0035

These values were selected considering the small-scale WPS (Lim, 2017). The length of the crank with the rod was calculated to allow the crank to rotate in a circle according to the displacement of the blade. The models were designed using MultiBody.Parts.BodyBox and the selected parameters were applied as input values. For connecting the system, MultiBody.Joints.Revolute is the connection model used that connects the system. The rotation speed of the low speed shaft is increased by the gearbox ratio whereas torque is reduced. The torque in the high-speed shaft connected from the gear model is used for the rotation of the generator and the generation of the torque drawn out from the generator. The power of the system was finally obtained by using the angular velocity and torque output from the generator. Figure 4 is model of whole system in OpenModelica.

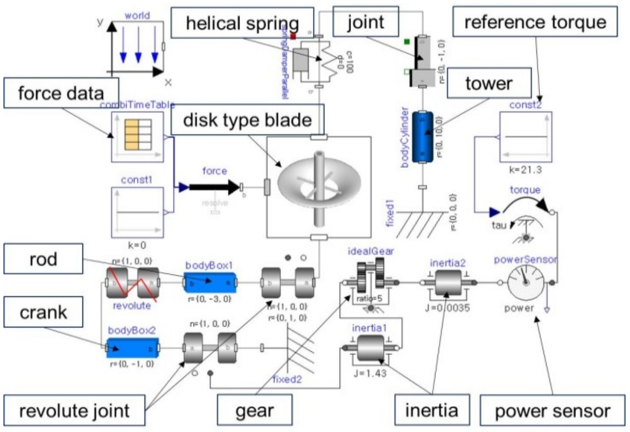


Figure 4. Model of lift-generating disk type wind power generation system in OpenModelica.

4 Simulation Verification

4.1 Experiment Composition

The wind tunnel test was conducted for the simulation verification of the WPS designed based on the Modelica and compared with the experimental data. Among the 15 blade types, the case where the most lift occurs is Case 7, and the AOA is 15°. Based on this, the blade is reduced in size by 1:10 and 1:15. The 4 reduced models (Case A, B, C, and D) are manufactured by 3D printer. In the case of the blade shape derived from the CCD method, the reduced model corresponding to case 7 was Case C. Based on this, Case A and D had the same AOA. To compare various results, Case B was made by greatly increasing the AOA. In addition, Case A and B were solid types, and Case C and D were hollow.

The wind tunnel test equipment was 300 mm in height, 400 mm in width, 1,200 mm in length and the maximum wind speed was increased to 20 m/s. Experiments were carried out to observe the upward displacements of the fabricated small scale model blades with varying wind speeds. As the vertical displacement of the disk type blade increased, the upward movement was observed in Case C and D, and the Case A and B did not move, so the displacement result could not be obtained. Case A is a 1:15 small scale model. Since the area under pressure is small, the generation of lift for disk movement is not sufficient. In Case B, there was no change of motion up to wind speed of 20 m/s, because the weight increased due to the rise of the angle of attack. In Case C, the movement was observed at wind speeds of 12 m/s or more, and it was confirmed that the maximum movement was 110 mm. In Case D, the movement was observed at wind speeds of 12 m/s or more, and it rose up to 40 mm, but increased at a much lower rate than Case C.

4.2 Validation

As a result of the wind tunnel test, we compare the experimental value and the simulation value for Case C, which showed a large upward movement in the wind speed range of 12~20 m/s. As shown in Figure 5, the simulation design was constructed considering the wind tunnel test environment.

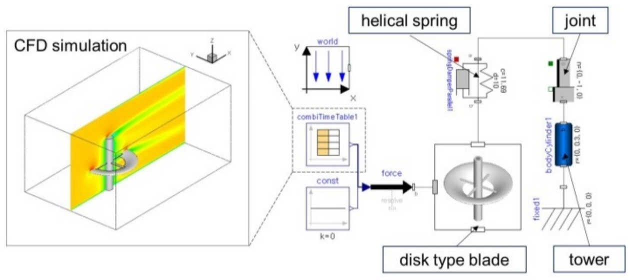


Figure 5. Modeling of 3D printing disk type blade in OpenModelica for simulation verification.

Experiments were performed to obtain the upward displacement of the blades at the wind speeds of 12, 15 and 20 m/s. The lifting force derived from the CFD simulation under the same conditions as the experimental environment was applied to the Modelica to obtain the upward displacement of the blade. The comparison between the wind tunnel test result and the Modelica simulation result is shown in Figure 6.

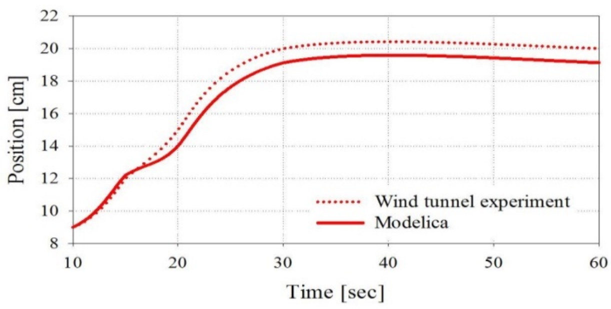


Figure 6. Simulation verification results compared with wind tunnel test.

The initial position of the small scale model is 90 mm. The simulated time was 60 seconds considering actual wind tunnel test time. Through these comparisons, it can be seen that the experimental results and the simulation results are similar.

5 Results and Discussions

Through this study, we apply the lift derived from CFD simulation to Modelica and compare the power of the WPS according to the blade shape. It is to derive the results with only the simulation. Among the 15 types of blades designed by CCD method, the comparison objects are as follows:

- Case 2 and 7 were selected to compare the case that generates the lowest lift and the highest.
- Case 7 and 12 were selected to compare the case that generates the highest lift and highest drag.

The generator power results for the WPS of the three blade cases are shown in Figure 7. In order to compare the blade with the lowest and highest lift, the average power for one hour was obtained. Case 9 was excluded from the case of the lowest lift because the drag force was large. Case 2 (chord length 0.2 m, AOA 5°) is 0.437 kW and Case 7 (chord length 0.7 m, AOA 15°) is 0.530 kW. It can be seen that the larger the lift generated on the blade, the higher the average power. Case 12 (chord length 0.45 m, AOA 23.41°) is 0.514 kW. This is the case where the lift is higher than Case 2 and similar to Case 7, but the drag is the largest. From the simulation results, it can be seen that the Case 7, which has the highest lift, has the highest average power of the WPS. Therefore, it is possible to derive the power of the proposed system through simulation.

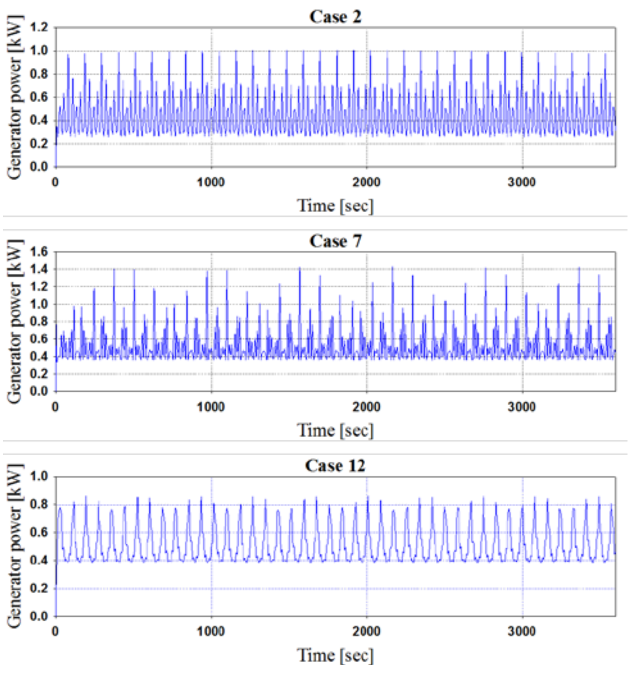


Figure 7. Simulation results of generator power output.

6 Conclusions

Considering the various problems of the conventional WPS, a mechanism of a generation system of a disk type blade capable of generating wind power with a simple structure is proposed. In order to determine the disk type blade shape, three design parameters affecting the generation of lift were selected and 15 blade shapes were constructed by applying the CCD method. CFD simulation was carried out to derive lift force data for the designed blades and a constant wind speed condition of 12 m/s was applied. We modeled disk type blade, springs, tower, crank, gearbox, generator, etc. for integrated simulation and conducted simulation verification through wind tunnel test. In order to compare the power of the WPS according to the influences of the design variables among the blades, comparison was made with three blade cases. In the wind tunnel test, as shown in Figure 6, the blades were vertically moved up to the final wind speed of 20 m / s and almost stopped. Because the blade of the reduced model used in the experiment is light in weight, it only moves upward. However, in the simulations designed considering PTS and PCS, the blades periodically move according to the increase of the blade weight and the spring constant, and it is confirmed that the power is derived as shown in Figure 7.

As a result, the mechanism of the WPS for the new type of blades was established and the power result for the system was derived by applying the simulation technique.

Acknowledgements

This research is supported by Hyundai NGV (No.2016-11-0906) and the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science, ICT & Future Planning (2017R1A2B4009606).

References

Strobel M, Vorpahl F, Hillmann C, Gu X, Zuga A, and Wihlfahrt U. The Onwind Modelica Library for Offshore Wind Turbines-implementation and First Results. *Proceedings of the 8th International Modelica Conference*, March 20-22, 2011, Dresden, Germany.

Petersson J, Isaksson P, Tummescheit H, and Ylikiiskila J. Modeling and Simulation of a Vertical Wind Power Plant in Dymola/Modelica. *Proceedings of the 9th International Modelica Conference*, September 3-5, 2012, Munich, Germany.

Eberhart P, Chung TS, Haumer A, and Kral C. Open Source Library for the Simulation of Wind Power Plants. *Proceedings of the 11th International Modelica Conference*, September 21-23, 2015, Versailles, France.

Khemili I and Romdhane L. Dynamic Analysis of a Flexible Slider-Crank Mechanism with Clearance. *European Journal of Mechanics A-Solids*, 27(5):882-98, 2008.

Montgomery DC, Design and Analysis of Experiments. 9th ed. *John Wiley and Sons Press*, New Jersey, 2017.

Mattsson SE, Elmqvist H, and Otter M. Physical System Modeling with Modelica. *Control Engineering Practice*, 6(4):501-10, 1998.

Lim CW, Design and Manufacture of Small-Scale Wind Turbine to emulate Torque Response of MW Wind Turbine. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 4(4):409-418, 2017.

Modelling of Asymmetric Rotor and Cracked Shaft

Tatsuro Ishibashi[1] Atsushi Yoshida[2] Tadao Kawai[2]

[1]Meidensha Corporation, Japan, ishibashi-tat@mb.meidensha.co.jp
[2]Department of Mechanical & Physical Engineering, Osaka City University, Japan,
m17ta040@ka.osaka-cu.ac.jp, kawai@mech.eng.osaka-cu.ac.jp

Abstract

The object of this paper is to present asymmetric rotor and shaft models in rotating machinery systems. Using these models it is possible to analyze the electrical motors or generators which have different lateral stiffness or the moments of inertia in two orthogonal directions. The asymmetry causes unstable vibrations in some rotating speed ranges. A cracked shaft model is presented as the extension of the asymmetric shaft model. These models are implemented in our original rotating machinery library.

Keywords: Rotor Dynamics, Asymmetrical Rotor, Asymmetrical Shaft, Cracked Shaft

1 Introduction

Rotating machinery systems have been dominant in most of heavy equipment such as turbine, generator, motor and so forth. Rotating machinery systems are always the key component of these equipment and the dynamic properties of which can determine the performance of the whole system. Before manufacturing, the modelling and simulation of rotating machinery systems can help the engineers to design a better system. Rotating machinery systems may suffer from different kinds of faults. A proper modelling and simulation of rotating machinery system with common faults should help engineers understand the performance with faults well, so that faults can be discovered or diagnosed in the early stage. Approaches to dynamic analysis of rotor systems can be divided into two main branches. One is the widely used finite element method (FEM), and the other is the relatively more traditional one, i.e. the transfer matrix method (TMM). This method is relatively simple and straightforward in application. The rotating shaft are decomposed into the concentrated mass and flexible beam. The main advantage of TMM is low computational cost.

Modelica is an object-oriented, declarative, acausal, multi-domain modelling language for component-oriented modelling of complex systems. It is suitable for modelling and simulation of rotating machinery systems with faults, whose parameters are frequently changed. Causal modelling method appears to be inefficient because the code is difficult to modify and reuse. Modelling by Modelica decomposes rotating machinery systems into several basic components. Basic components are reusable, and their parameters can be simply modified. By using the Modelica_LinearSystems2 library, it is possible to do eigenfrequency analysis in the linearized system. These are the big advantage for modelling and simulating of the rotating machinery systems with several faults by TMM.

Several papers have been written relating modelling rotor dynamics in Modelica. Vibrations of gears in the plane have been implemented and refined (Dahl *et al*, 2017; Kosenko and Gusev, 2012; van der Linden, 2012). However those are focusing on only a gear. Previously 3 DOF (degree of freedom) rotor dynamics library with multi-faults is reported (Ming *et al*, 2013). It only handle Jeffcott rotor systems neglecting the gyroscopic effect, which is very important for rotor dynamics.

We have created the rotating machinery library which has 5 DOF rotor dynamics model components (Ishibashi *et al*, 2017). The library has common faults of rotating machinery systems including static and dynamic unbalance, shaft bending, and faulty bearing. In this paper, we implemented asymmetrical rotor and shaft models. The actual rotating machinery systems have asymmetries such as two pole generators and propellers etc. The cracked shaft model is also implemented as the extension of the asymmetric shaft model.

2 Asymmetrical systems

The actual rotating machinery such as propeller rotors and two-pole generator rotors have the asymmetry. The former are systems with a directional difference in the rotor inertia (Figure 1 (a)). The latter are systems with a directional difference in the shaft stiffness (Figure 1 (b)). As the shaft with these directional differences rotates, terms with time-varying coefficients appear in the governing equations. The most characteristic property of asymmetrical system is the appearance of unstable vibrations in some rotating speed ranges (Ishida and Yamamoto, 2012; Matsushita *et al*, 2017). The asymmetrical rotor and shaft models are implemented in our rotating machinery library.

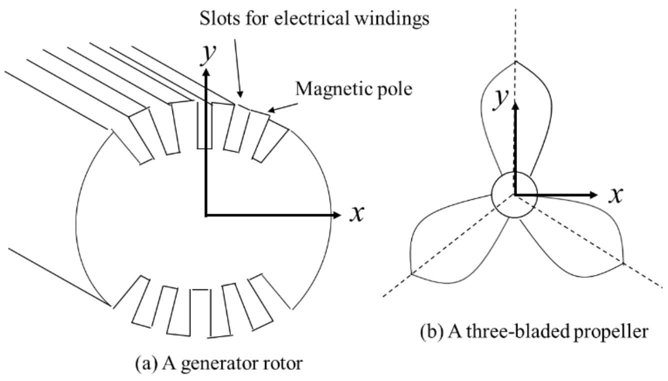


Figure 1. Asymmetries of the rotor and the shaft.

2.1 Asymmetrical rotor

The asymmetrical rotor is considered as a single point with a rigid disc or a long rigid shaft. The directional difference in the moment of inertia is expressed as $I_1 = I + \Delta I, I_2 = I - \Delta I$. The equations of asymmetrical rotor motion are following.

$$\begin{aligned} &I\ddot{i_x} + I_p\dot{\theta}\dot{i_y} - \Delta I\frac{d}{dt}(\dot{i_x}\cos 2\omega t + \dot{i_y}\sin 2\omega t) \\ &+\tau\dot{\theta}^2\{(I - I_p)\cos(\omega t + \tau_0) - \Delta I\cos(\omega t - \tau_0)\} \\ &= M_x \end{aligned} \qquad (1)$$

$$\begin{aligned} &I\ddot{i_y} - I_p\dot{\theta}\dot{i_x} - \Delta I\frac{d}{dt}(\dot{i_x}\sin 2\omega t + \dot{i_y}\cos 2\omega t) \\ &+\tau\dot{\theta}^2\{(I - I_p)\sin(\omega t + \tau_0) - \Delta I\sin(\omega t - \tau_0)\} \\ &= M_y \end{aligned} \qquad (2)$$

Here,
i_x, i_y: Deflection angle of each direction,
M_x, M_y : Moment of each direction,
θ : Rotational angle,
τ : Slope of dynamic unbalance,
τ_0 : Initial phase of dynamic unbalance,
ΔI : Difference of moment of inertia,
I : Average moment of inertia,
I_1 : Moment of inertia about y axis,
I_2 : Moment of inertia about x axis,
I_p : Polar moment of inertia.

2.2 Asymmetrical shaft

According to the coordinate system O - ξ η rotating with the shaft shown in Figure 2, the asymmetrical elastic shafts which have a directional difference in the shaft stiffness, holds the following equation.

$$\begin{aligned} &E\begin{pmatrix} I_\xi i_{b\xi} \\ I_\eta i_{b\eta} \end{pmatrix} = \frac{L^2}{2}\begin{pmatrix} F_{a\xi} \\ F_{a\eta} \end{pmatrix} + L\begin{pmatrix} M_{a\xi} \\ M_{a\eta} \end{pmatrix} + E\begin{pmatrix} I_\xi i_{a\xi} \\ I_\eta i_{a\eta} \end{pmatrix} \\ &+\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} 0 \\ -\dfrac{mgL^2}{6} \end{pmatrix} \end{aligned} \qquad (3)$$

$$\begin{aligned} &E\begin{pmatrix} I_\xi u_{b\xi} \\ I_\eta u_{b\eta} \end{pmatrix} = \frac{L^2}{6}\begin{pmatrix} F_{a\xi} \\ F_{a\eta} \end{pmatrix} + \frac{L}{2}\begin{pmatrix} M_{a\xi} \\ M_{a\eta} \end{pmatrix} + EL\begin{pmatrix} I_\xi i_{a\xi} \\ I_\eta i_{a\eta} \end{pmatrix} \\ &+E\begin{pmatrix} I_\xi u_{a\xi} \\ I_\eta u_{a\eta} \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} 0 \\ -\dfrac{mgL^3}{24} \end{pmatrix} \end{aligned} \qquad (4)$$

The second moments of area changes twice per revolution at the stationary coordinate system as expressed by Equation 5.

$$\begin{pmatrix} I_x \\ I_y \end{pmatrix} = \frac{I_\xi + I_\eta}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{I_\xi - I_\eta}{2}\cos 2\omega t\begin{pmatrix} -1 \\ 1 \end{pmatrix} \qquad (5)$$

Here,
m : Shaft mass,
L : Shaft length,
E : Young's modulus,
I_x, I_y : Second moments of area in the stationary coordinate system,
I_ξ, I_η : Second moments of area in the rotating coordinate system,
g : Constant of gravitation,
ω : Rotating speed.
The indices a and b mean left and right flange respectively.

This time-varying coefficients cause an unstable range in machine operating frequency.

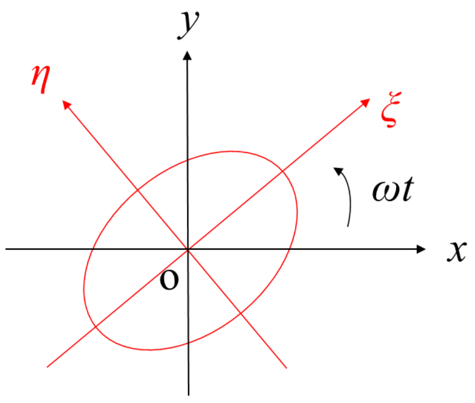


Figure 2. The stationary (black) and rotating coordinate system (red).

3 Cracked shaft

One of the most important causes of accidents in rotating machinery is the crack caused by fatigue. In horizontal shaft system, since the weight of the rotors and shafts bend shafts, expanding force acts on the underside of the shaft and compressing force acts on the upper side. As the result, when the shaft rotates, periodic stress is generated and cracks sometimes occur. In the cracked shaft, the stiffness differs depending on the direction in which the shaft bends. In the crack opening direction, the shaft stiffness is small, and on the other hand, in the closing direction, the stiffness becomes large (Figure 3 (a)). The restoring force of the shaft has a nonlinear spring characteristic in a fragment linear fashion along the η direction at the rotating coordinate system O - ξ η rotating with the shaft (Figure 3 (b)). Along the ξ direction, the restoring force has a linear spring (Gasch, 1976; Henry and Okah-Avae, 1976). The cracked shaft model is expressed as the extension of the asymmetrical shaft model. In Equation 3 and 4, the second moments of area at the rotating coordinate system changes as follows.

$$I_\eta = \begin{cases} I_\eta - \delta, & i_{b\eta} - i_{a\eta} \geq 0 \\ I_\eta + \delta, & i_{b\eta} - i_{a\eta} < 0 \end{cases} \tag{6}$$

Here,
δ: Crack strength parameter.

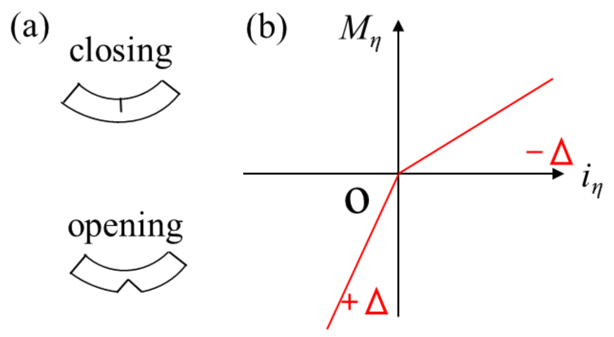


Figure 3. The cracked shaft model. (a) Conceptual diagram (b) A nonlinear spring characteristic in a fragment linear fashion.

4 Example Models

The above models are using our original rotating machinery library. Dymola is used for the simulations. The asymmetrical rotor model is implemented by modifying rotor. The asymmetrical and cracked shaft models are implemented by modifying the shaft. Figure 4 shows a simple Jeffcott rotor model using our rotating machinery library. Here, we simulate the asymmetrical rotor and shaft and the cracked shaft in the Jeffcott rotor system.

4.1 Symmetrical system

Before the transient simulation, we check the symmetrical model by eigenfrequencies analysis using the Modelica_LinearSystems2 library (Bauer *et al*, 2009; Otter, 2006) and creating the function. Figure 5 shows the Campbell diagram. Red curve shows the

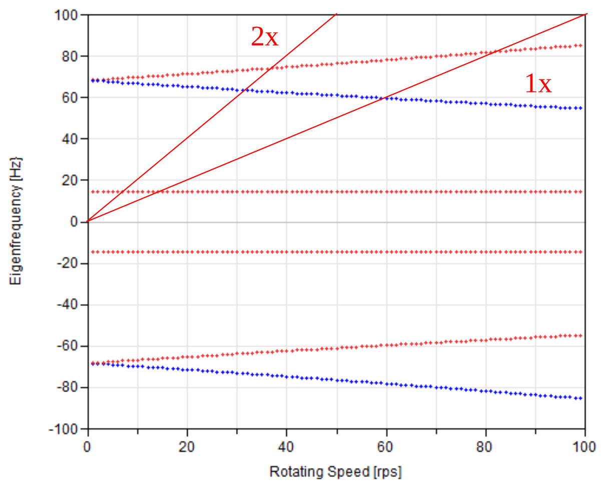


Figure 5. The Campbell diagram of the symmetrical system. Red shows the counterclockwise rotation whirl mode. Blue shows the clockwise rotation whirl mode.

counterclockwise rotation whirl mode and blue curve shows the clockwise rotation whirl mode by eigenvector analysis. The eigenfrequency curves starting from around 70Hz split due to the gyroscopic effects. The intersection between the synchronous excitation line and the eigenfrequencies in Campbell diagrams are referred as critical speeds. Also, the intersection between twice the synchronous excitation line and the eigenfrequencies are referred as secondary critical speeds. This system has two critical speeds around 14Hz and 83Hz. Since it is easy to understand the simulation results, in this case, time derivatives terms in the equations of journals motion are ignored.

To check this system, we simulate the ramp response of this system. Rotor with a diameter of 80mm has both the static and dynamic unbalance. Shaft3 with a diameter of 8mm also has the shaft bending. Figure 6 and Figure 7 show the deflection in the horizontal direction and the deflection angle vibration of Rotor against the rotating speed respectively. The rotating speed is raised from 0 rps to 100 rps at the rate of 10 rps/s by replacing the constant input with the ramp input in Figure 4. The deflection has the peak at the a little bit higher speed of the 15 rps and the deflection angle has the peak around 83 rps.

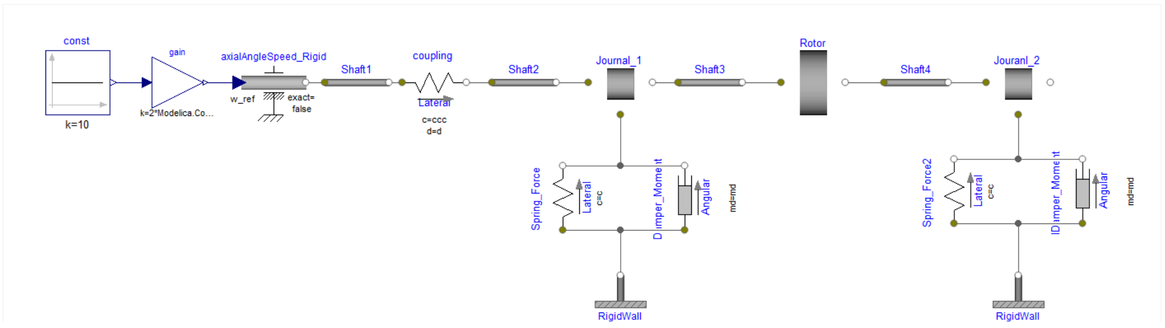


Figure 4. Modelica model of a Jeffcott rotor system.

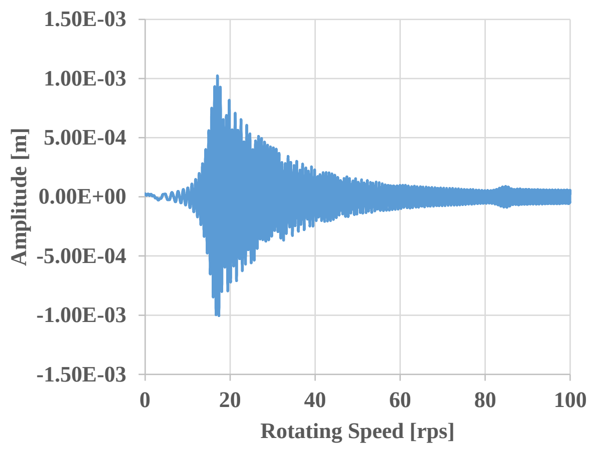


Figure 6. Ramp response of Rotor deflection.

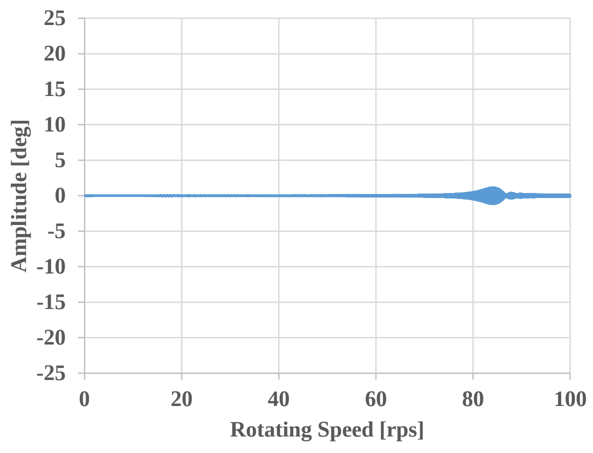


Figure 7. Ramp response of Rotor deflection angle.

4.2 Asymmetrical system

4.2.1 Asymmetrical rotor

We simulate the model replacing Rotor with the asymmetrical rotor in Figure 4. The asymmetrical rotor is expressed in Equation 1 and 2. The asymmetrical rotor is assumed as the elliptic cylinder with the long axis of 45mm and the short axis of 35mm which has the directional difference in the moment of inertias $\Delta I = 6.4e^{-4}$ kgm^2.

Figure 8 and Figure 9 show the ramp response. The deflection angle peak amplitude is larger than that of the symmetrical system, although the peak positions in both the deflection and deflection angle are the totally same positions as the symmetrical system. This implies that the rotating speed around 83 rps is unstable. To examine the unstable region, we simulate the model at several constant rotating speeds around 83 rps.

Figure 10 shows the transient simulation results of rotor deflection angles at the different constant rotating speeds respectively. At the rotating speed of 80 and 86 rps, Rotor deflection angle converges. From 81 to 85 rps, Rotor deflection angle vibration diverges. At 83 rps, Rotor deflection angle diverges rapidly. These clearly indicate that this system is unstable in this region. Unstable regions are demonstrated in the asymmetrical rotor system, whereas in the symmetrical system they are not observed. Due to the directional difference in the moment of inertias, the unstable region exists only in this region.

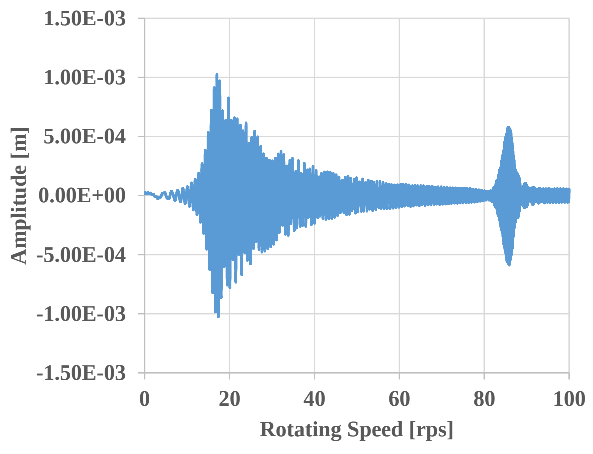


Figure 8. Ramp response of Rotor deflection with the asymmetrical rotor.

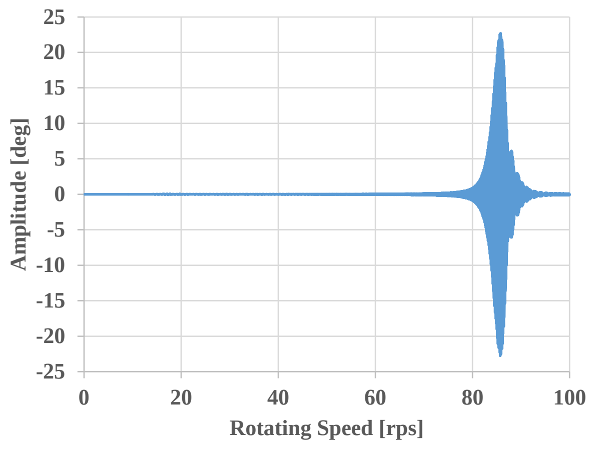


Figure 9. Ramp response of Rotor deflection angle with the asymmetrical rotor.

4.2.2 Asymmetrical shaft

We simulate the model replacing Shaft3 with the asymmetrical shaft in Figure 4. The asymmetrical shaft model is expressed in Equation 3, 4 and 5. The asymmetrical shaft is assumed as the elliptic cylinder with the long axis of 4.5mm and the short axis of 3.5mm which has the directional differences in second moments of area.

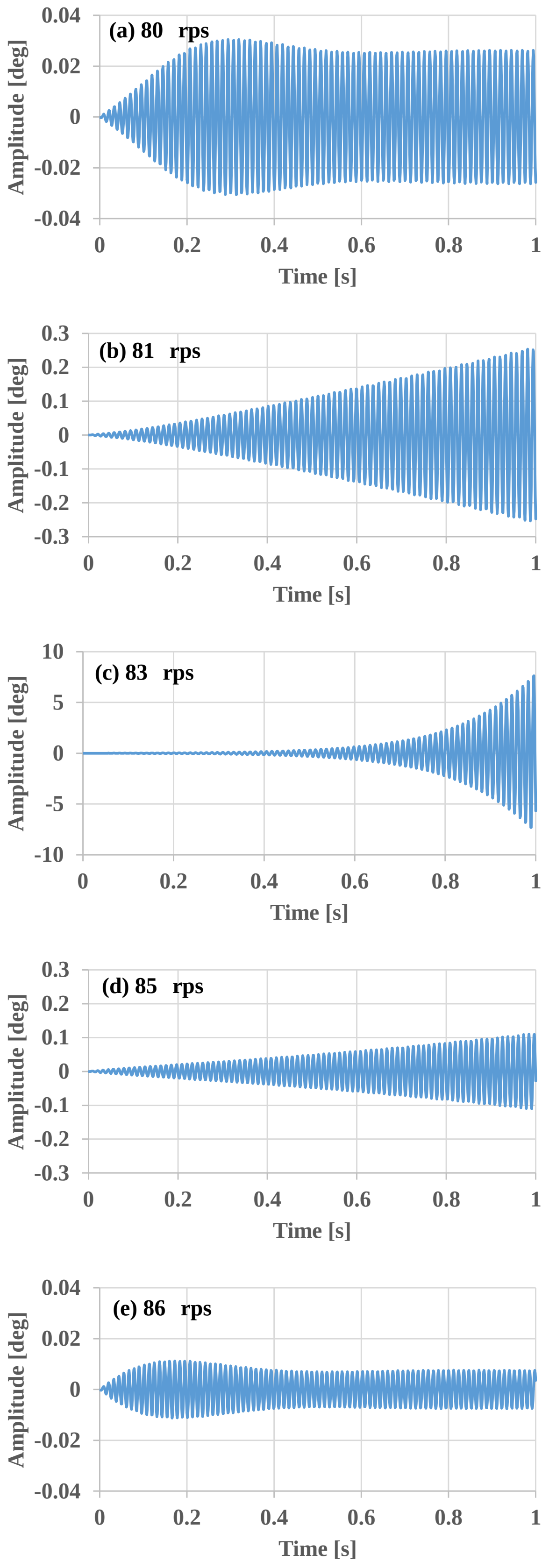


Figure 10. Unstable regions of the asymmetrical rotor system. Transient results of Rotor deflection angle with the asymmetrical rotor at different constant speeds.

Figure 11 and Figure 12 show the ramp response. The behavior is the same as the asymmetrical rotor system. Since the equations of motion of both the deflection and deflection angle have the directional stiffness differences in the asymmetrical shaft system, the asymmetrical shaft system has much more unstable regions such as the region around 14 rps.

To examine the unstable region, we simulate the model at several constant rotating speeds around 14 rps. Figure 13 shows the transient simulation results of Rotor deflections in the horizontal direction at the different constant rotating speeds respectively. At 14.1 and 14.7 rps, Rotor deflection converges. From 14.2 to 14.6 rps, Rotor deflection diverges. At 14.6 rps, Rotor deflection diverges rapidly. These clearly indicate that the system is unstable in this region. Similarly the system is unstable at the region around 83 rps. Also, the other several unstable regions corresponding to the minute peaks at 45 and 62 rps exist in simulation.

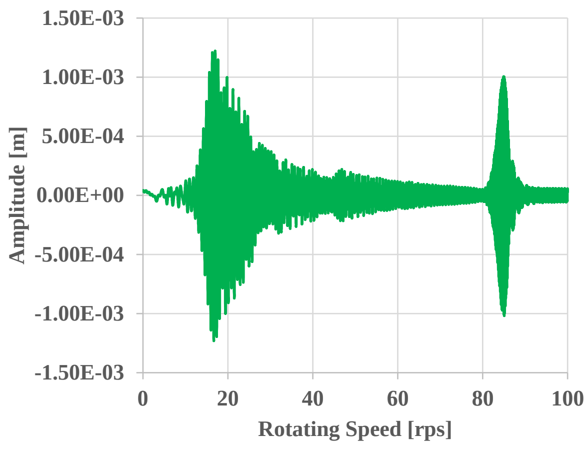


Figure 11. Ramp response of Rotor deflection with the asymmetrical shaft.

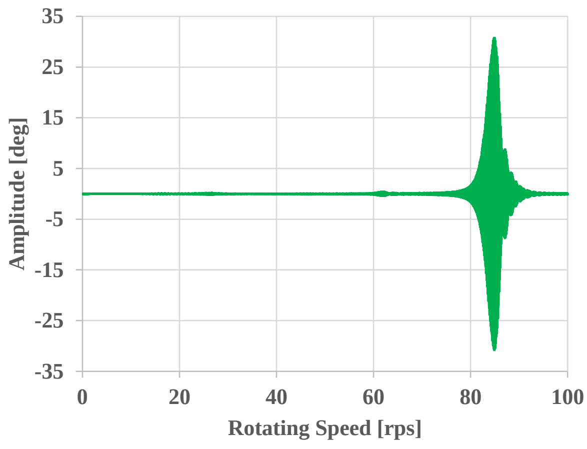


Figure 12. Ramp response of Rotor deflection angle with the asymmetrical shaft.

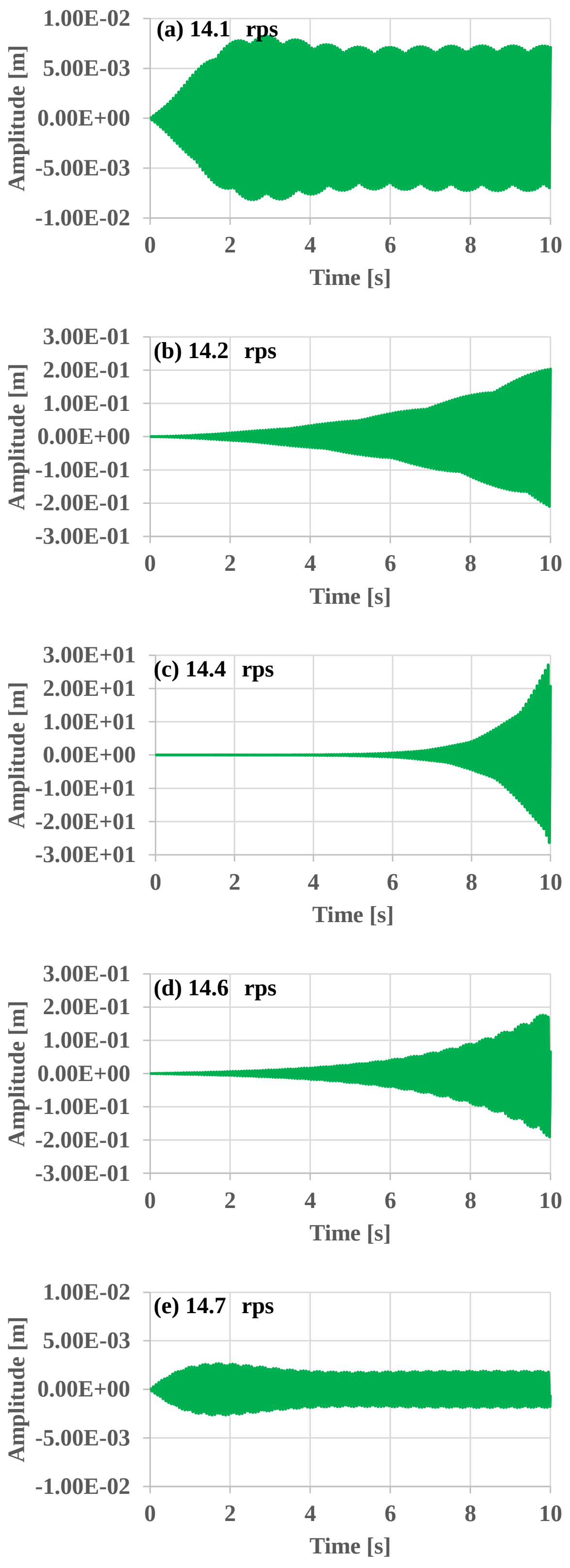


Figure 13. Unstable regions of the asymmetrical shaft system. Transient results of Rotor deflection with the asymmetrical shaft at different constant speeds.

5 Cracked shaft

We simulate the model replacing Shaft3 with the cracked shaft in Figure 4. The cracked shaft model is expressed as the extension of the asymmetrical shaft by adding Equation 6. Here, we examine the nonlinear vibration of the cracked shaft. The cracked shaft shape is symmetrical. It has the crack strength parameter δ with half the second moment of the area of the symmetrical shaft.

The asymmetrical shaft and the symmetrical system are simulated to check the differences. Figure 14 shows the transient simulation results of the rotor deflection angle at the constant rotating speed of 5 rps.

We analyze frequency characteristics of results by FFT signal processing with a rectangular window in the range of 4 to 6 s. The range is chosen to remove the initial transient effect of this system. Figure 15 shows the results of FFT analysis. The symmetrical system has the only component synchronized with the rotating speed. The asymmetrical shaft has the other components. The frequency of 14 Hz is the eigenfrequency of the

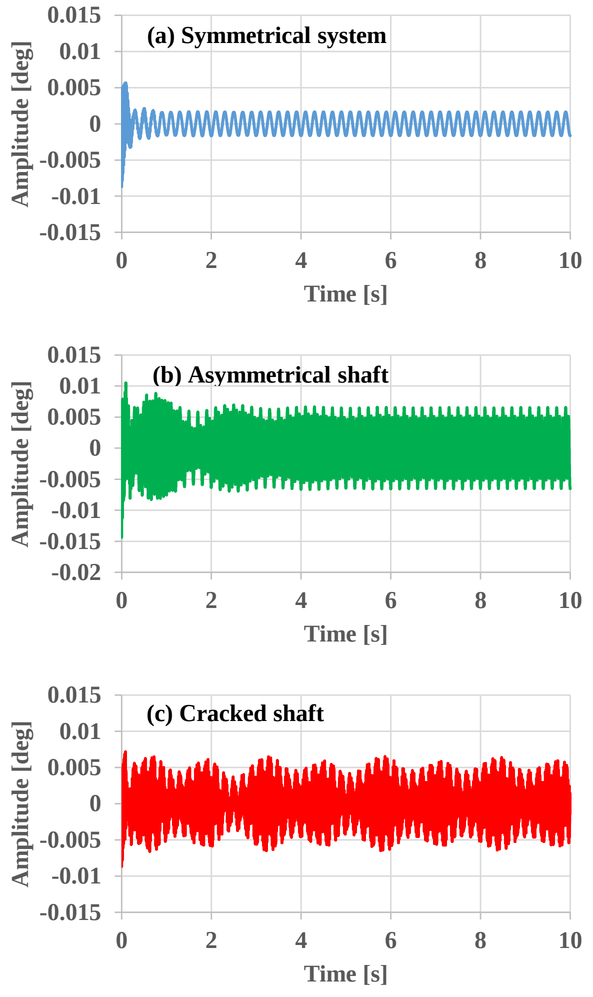


Figure 14. Comparison of transient simulation results at constant speed 5 rps. (a) Symmetrical system. (b) Asymmetrical shaft. (c) Cracked shaft.

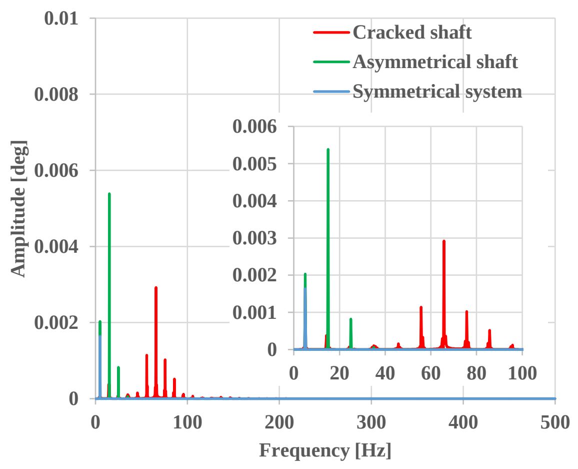


Figure 15. Comparison of FFT analysis. Blue line: Symmetrical system. Green line: Asymmetrical shaft. Red line: Cracked shaft. The inset shows the enlarged view.

rotor horizontal vibration mode in this system. The cracked shaft has the same components as the asymmetrical shaft and the other components due to the nonlinear spring characteristic in a fragment linear fashion.

6 Conclusion

In this paper, models are presented to simulate the asymmetrical rotor and shaft and the cracked shaft. Using our rotating machinery library, it is possible to model an actual rotating machinery such as propeller rotors and two-pole generator rotors with asymmetries. As the extension of the asymmetrical shaft model, the cracked shaft model is implemented. Examples of simple rotor systems are demonstrated. The presented model shows the abilities to design and diagnose rotating machinery systems.

References

Markus Dahl, Håkan Wettergren and Henrik Tidefelt. Modelica Spur Gears with Hertzian Contact Forces. *Proceedings of the 12th International Modelica Conference*, 2017. doi:10.3384/ecp17132755.

Robert Gasch. Dynamics Behavior of a Simple Rotor with a Cross-Sectional Crack. *Proceedings of the International Conference on Vibrations in Rotating Machinery*, I. Mech. E. 123-128, 1976.

T. A. Henry and Okah-Avae. B. E. Vibrations in Cracked Shaft. *Proceedings of the International Conference on Vibrations in Rotating Machinery*, I. Mech. E. 15-17, 1976.

Tatsuro Ishibashi, Han Bing and Tadao Kawai. Rotating Machinery Library for Diagnosis. *Proceedings of the 12th International Modelica Conference*, 2017. doi:10.3384/ecp17132381.

Yukio Ishida and Toshio Yamamoto. *Linear and Nonlinear Rotordynamics: A Modern Treatment with Applications, 2nd Edition*. Wiley-VCH, 2012.

Ivan Kosenko and Ilya Gusev. Revised and improved implementation of the spur involute gear dynamical model. *Proceedings of the 9th International Modelica Conference*, 2012. doi:10.3384/ecp12076311.

Osami Matsushita, Masato Tanaka, Hiroshi Kanki, Masao Kobayashi and Patrick Keogh. *Vibrations of Rotating Machinery*. Springer Japan, 2017. doi: 10.1007/978-4-431-55456-1.

Li Ming, Wang Yu, Li Fucai, Li Hongguang, and Meng Guang. Modelica-based Object-orient Modeling of Rotor System with Multi-Faults. *Chinese Journal of Mechanical Engineering*, Vol. 26, No. 6, 2013.

F.L.J. van der Linden. Modelling of elastic gearboxes using a generalized gear contact model. *Proceedings of the 9th International Modelica Conference*, 2012. doi:10.3384/ecp12076303.

Composable Modelling for a Hybrid Gearbox

Joshua Sutherland[1] Rui Gao[2]

[1]Department of System Innovation, The University of Tokyo, Japan, joshua@m.sys.t.u-tokyo.ac.jp
[2]Modelon KK, Tokyo, Japan rui.gao@modelon.com

Abstract

Gearbox modelling is an important topic to the simulation community due to its difficulty in representing mixed continuous and discrete behaviour. Such kind of models also usually couple the loss of power that is dependent on the angular velocity and the load. However, one single gearbox model with a certain fidelity cannot satisfy the broader needs of modelling during product development. In this paper, we will address hybrid modelling issues and take the gearbox model as an example to illustrate how to model gearboxes from a Systems Engineering perspective. A supervisor and supervised model structure is investigated that could also be regarded as a use case to a general cyber-physical modelling approach. The resultant model achieves the same output as the LossyGear model in Modelica Standard Library.

Keywords: Hybrid model, Gearbox, Loss power, Cyber-physical

1 Introduction

Modelica is well accepted as the multi-domain modelling and simulation standard and its acausal model features enable efficient representation of cyber-physical systems. One of the important applications is to use it for simulation-based design in product development to reduce the use of the real prototypes. However, this makes it difficult for engineers who have experience in experimental based product development or who have deep knowledge in high fidelity simulation technologies such as Finite Element Method (FEM) to understand to what extent Modelica can capture and reproduce the physical phenomena. Such engineers tend to compare the accuracy of Modelica model with more traditional FEM models.

Any model of physics including a Modelica model is no more than a projection of the real-world behaviour in terms of a design or analysis purpose. Referring to the physical phenomena, it is difficult (or even impossible) to represent it exhaustedly by one single model. However, such an exhaustive model would be undesirable as a good model is one with the simplest form that satisfies the certain design or analysis purpose. As such, defining the adaptable scope of the model becomes a crucial task to model based design.

The hybrid modelling features in Modelica is enabled in the Modelica language by the synchronous data flow principle (known as hybrid (Elmqvist, Mattsson, & Otter, 2001; Otter, Elmqvist, & Mattsson, 1999)). It is important because these principles help to define the boundary of the physics-based model. Enabling the physics-based model to be compartmentalised by a discrete mode model.

Further, Modelica provides mode transition, finite state machine functionality by Modelica text and libraries including: StateGraph, StateGraph2 and Synchronous. With the former two being Dymola specific extensions, and the latter aiming to supersede them both as part of Modelica Language 3.3 (Elmqvist, Gaucher, Matsson, & Dupont, 2012). These modelling technologies broaden the cyber-physical modelling capability of the Modelica.

Given any model that is developed for use, either for design or analysis purposes, is the outcome of human activity with a specific objective. To capture the development requirements and setup the specification is one of the most important starting points in modelling and simulation-based design.

Take the gearbox example, the simplest IdealGear model represents the basic functionality and captures the most important requirements of a gearbox. However, the gearbox designer may also need to consider the gearbox performance with for example power loss that is dependent on the rotational velocity and load level. Such kind of functional/performance requirements need a high-fidelity model.

1.1 Past Research

Given the importance of gearboxes to many different machines significant amounts of effort have gone into creating models of them. One review paper (Parey & Tandon, 2003) presents 63 different references to spur gear dynamic models alone. Regarding Modelica modelling of gearboxes two models stand out for further investigation: Modelica Standard Library (MSL) LossyGear presented in (Pelchen, Schweiger, & Otter,

2002) and a composed gearbox model presented in (Schlegel, Hösl, & Diel, 2009).

LossyGear is a highly popular model of a single gear interaction, which successfully models not only load- and speed-dependent energy losses (associated with mesh and bearing friction) while the gears are moving, but also the stiction affects which can lead to the gears becoming stuck. To model stiction required the usage of the hybrid modelling described previously. However, all the phenomena of the model are lumped into a single model which makes it difficult to separate phenomena or modify. A representation of LossyGear is presented in Figure 1.

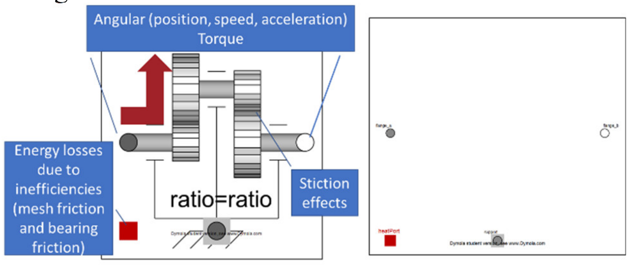

Angular (position, speed, acceleration)
Torque
Energy losses due to inefficiencies (mesh friction and bearing friction)
ratio=ratio
Stiction effects

Figure 1. LossyGear. **Left side:** Icon view, annotated with the key functions of the model. **Right side:** Diagram view, showing the lack of composition of other model blocks.

In (Schlegel et al., 2009) a complex automotive gearbox model is presented composed of many different components. The focus of this model is to represent the steady state behaviour of the gearbox including energy losses. However, by focusing on the steady state work was not done on ensuring it works appropriately including the times the gearbox is stuck (what LossyGear does so well).

It seems that there is a gap between models which are highly composable but cannot capture the discontinuous behaviour of when the gear is stuck (Schlegel et al., 2009) and models which have very accurate discontinuous behaviour but lack an emphasis on composition of a large number of components (Pelchen et al., 2002).

1.2 The Focus of this Paper

This paper aims to address the gap described in the previous section and construct gearbox models which are both highly composable but also model the discontinuities associated with being stuck.

We believe a focus on composability can drive a more function-based design where the specific behaviours required of the subsystem should be identified and then created by the composition of specific components in the subsystem, which then leads to greater innovation (there is opportunity to compose highly novel new systems). Ideally such an approach would be able to do the following:

- Quickly build subsystem models with sufficient fidelity for the larger system model's purpose.
- Provide insight into the performance of the individual components which make up the subsystem such that meaningful insight can be drawn on why the subsystem is performing as it should be.

While ensuring the resulting subsystem model satisfies the needs of the system model in which the subsystem is to be embedded.

There are many phenomena worth investigation regrading gearboxes (e.g. fatigue, vibration, and defects) discontinuities associated with being stuck and fictional energy losses are to remain the focus of this paper.

Clearly the approach any indervidual modeler takes to building a model of a particular subsystem will result in different models. Our basic assumption is that each model is more or less successful at fulfilling its purpose and such variability is undesirable. However, by defining an approach and applying it to a Gearbox example it is our aim to share potential best practices across the community.

To accomplish our aim the paper is arranged as follows:

- Section 2: The problem formulation of gearbox modelling with the different design requirements is explored.
- Section 3: Operational mode modelling using the StateGraph2 and other methods is presented.
- Section 4: The composable modelling of gearbox is shown and comparison to the existing MSL LossyGear model is demonstrated.
- Section 5: Conclusions and further work are presented.

2 Gearboxes Modelling

2.1 Elegant Modelling

How "good" any model is, is a highly subjective matter that is dependent on the context in which the model will be constructed, verified, validated, used and modified. However, it is possible to draw out various quality measures which we believe any model maker would at least have a passing interest in. In the context of Systems Engineering the term "elegant" has been applied to systems which are considered holistically better than others (Griffin, 2010). With elegant designs being characterised as answering a yes to the questions on the left-hand side of Table 1. For this paper's purposes, we characterise those questions as shown on the right side of Table 1. We believe being able to create models consistently exhibiting the listed characteristics would be highly valuable.

Table 1. Characterization of elegant models.

Elegant question:	*Characterization for modelling*
Does it work?	Model provides sufficient fidelity and parametrization for its purpose. Stakeholders have sufficient confidence in the model.
Is it robust?	It can be used in contexts outside of those it was originally intended, or it fails gracefully.
Is it efficient?	Time resources needed to create the model, simulate and maintain it are not wasted.

2.2 Composing Subsystems from Components: The Preferred Solution?

A great strength of Modelica is its hierarchical object-oriented nature which enables the composition of models out of component models. Enabling the benefits recognised from the software industry of complexity management and facilitating code reuse.

Given a physical gearbox is created by the composition of individual components (e.g. bearings and interlocking gears) and then imbedding in a larger more complex system (e.g. a car) to deliver some desirable behaviour (e.g. torque change, speed change, energy losses and stiction effects) it is a good example for reviewing the described problem.

2.3 Modelling Purposefully

As described in the introduction (Section 1) we acknowledge that any model of physics including a Modelica model is no more than a projection of the real-world behaviour in terms of the design or analysis purpose. Further, it is difficult (or even impossible) to represent all potential physical phenomena exhaustively. As such we assert that one must take a highly purposeful approach when creating models.

Given that the language associated with describing the purpose is often ambiguous, we refer to the glossary of the Systems Engineering Body of Knowledge (Adcock (EIC), 2017) to obtain definitions which we subsequently use to describe the gearbox model.

2.3.1 Purpose

Defining purpose to simply be "What the system is for…" (Blockley & Godfrey, 2000) the modeller must also be clear of the potentially large difference between the purpose of system model compared to the system itself. As shown in Table 2 the purpose of a model is often to investigate negative aspects of the system such as energy loss. Clearly the purpose of a real gearbox is not to lose energy.

Table 2. Some example purposes of a gearbox and a gearbox model.

System:	*Some example purposes:*
Gearbox	• Change angular speed of a rotating shaft. • Change the torque being transferred on a rotating shaft. • Make a profit for a gearbox manufacturing company.
Gearbox model	• Investigate potential gear ratios which will work well with the system the gearbox is to be imbedded in. • Investigate potential energy losses from the gearbox. • Investigate potential longevity of the gearbox. • Make a profit for a gearbox modelling company.

2.3.2 Requirements

Given the definition of a requirement to be a "Statement that identifies a product … characteristic … necessary for product … acceptability" (ISO/IEC, 2007), requirements are a way to state an intention of a system such that it might fulfil its purpose. Given that the purposes of the system and model are different it then follows that the requirements would also be different.

2.3.3 Behaviour

Taking the definition of systems behaviour from (Ackoff, 1971) to be "a change which leads to events in itself or other systems. Thus, action, reaction or response may constitute behaviour in some cases." given this paper is concerned with dynamic modelling in Modelica the capturing of behaviour is incredibly important.

However, given our previous acknowledgement that a model cannot represent all phenomena and the model's behaviour will represent a subset of the physical gearboxes total behaviour.

2.3.4 Functions and Functional Requirements

Defining a function to be "An action, a task, or an activity performed to achieve a desired outcome." (Hitchins, 2008) it is possible to then create a definition for Functional Requirements being: An action, a task, or an activity performed to achieve a desired outcome necessary for product acceptability.

Given the purpose of a dynamic Modelica model is generally to investigate and predict the behaviour of a system which has yet to be built the most important functional requirements for the model are around modelling a subset of the behaviours of the physical system.

2.4 Functional Requirements Break-Down: An Example of a Gearbox Model

In this section we present some functional requirements for a gearbox model such that in later sections we can compose a new model.

2.4.1 Torque and motion conversion functional requirements

The basic and most important functionality of the gearbox is to convert the torque and motion by following underlying equations 1 and 2. As such any model interested in the behaviours of the rotating shafts will need to model these.

$$\varphi_a = ratio \cdot \varphi_b \qquad 1$$

$$0 = ratio \cdot flange_a\tau + flange_b\tau \qquad 2$$

2.4.2 Loss power functional requirements

In the real world the contact of parts (gear teeth and bearings) results in the loss of power. If the purpose of the model is to investigate such losses such behaviour must be a functional requirement.

It should be noted in the previous section that the functional requirements of a model are not the same of the system being built; i.e. stating that a model must be capable of simulating energy loss is not saying that energy loss is desirable in the physical system. Instead it is stating that the model must have a representation of this physical reality.

As per LossyGear (Pelchen et al., 2002) such losses are generally modelled as a speed dependent load torque (bearing friction) and a speed dependent efficiency (mesh friction). Both of which manifest as heat.

2.4.3 Operational mode functional requirements

Even without friction a gearbox can be in the states of moving forward, backward or be stopped. If friction is modelled closely to that of the real world it a non-linear force. It exhibits stuck and sliding behaviour (stiction) where to get two surfaces to move past each other a sufficient amount of initial force must be provided.

When applied to a gearbox it can be appropriate to think of the gearbox as having several discrete states in which it can be in at any time. The states and state transitions associated with a gearbox are presented in Figure 2. These states are important as they enable the capture of additional functional requirements of the gearbox model by which to describe the differnte operational modes or use cases of the model.

The number of transitions shown in Figure 2 make it somewhat complex however it is important to explicitly model all the transitions possible to be made such that an appropriate modelling approach can be taken.

2.4.4 Combined functional requirement of a gearbox model

The best practice is generally to start from a simple model and incrasingly add functions to it (but also complexity).

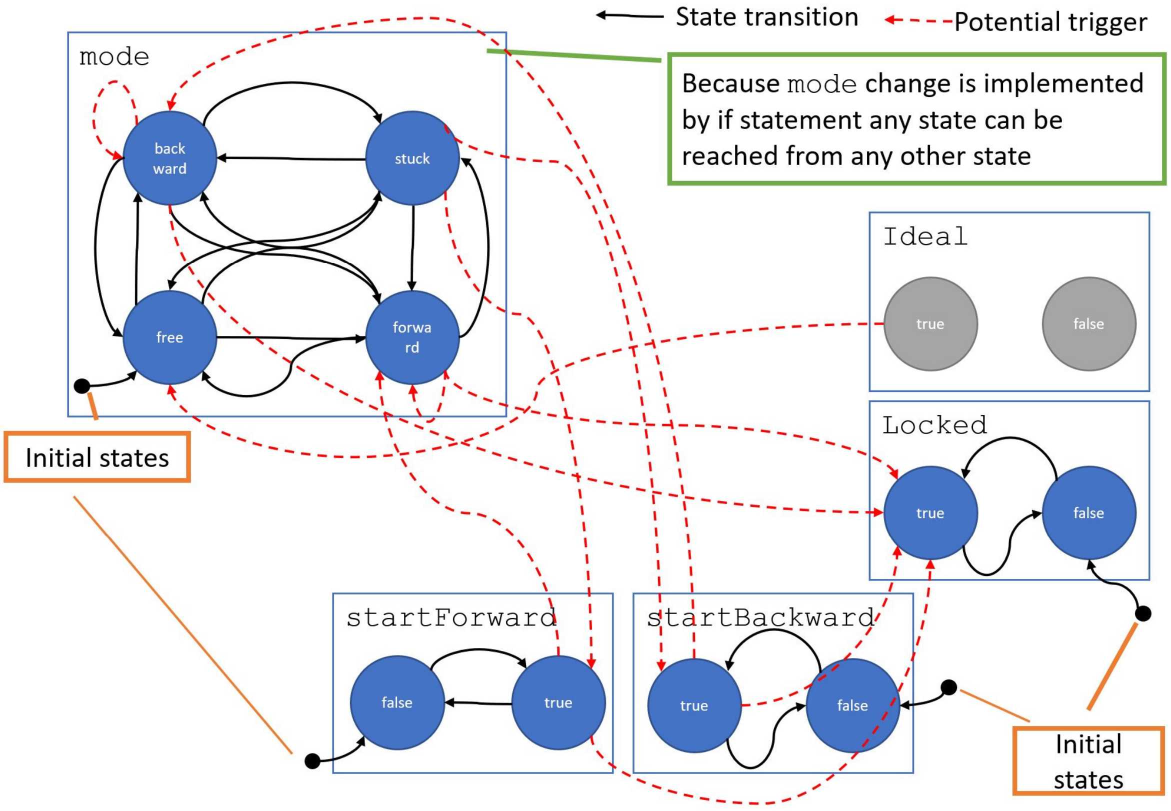


Figure 2. State and state transition representation of gearbox for depicting operational modes.

For our case of the gearbox a typical system engineer is most likely to start modelling from idealgear and then add the loss power functional requirments, then they will finally come to the real operational mode design. This is usually referred as the functional requirement that defines the fidelity of the targeted model.

Based on the investigation, the main functionality of the gearbox model is presented in Figure 3 which along with Figure 2 act as the functional requirements for the gearbox model.

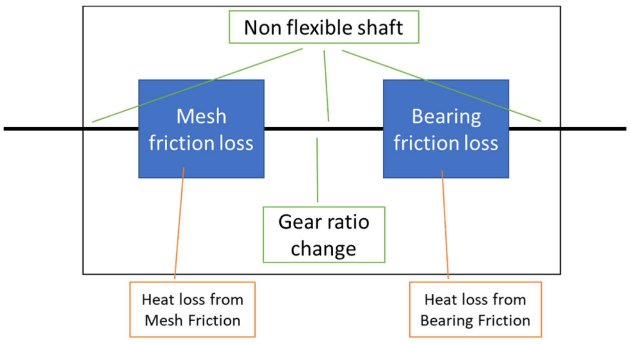


Figure 3. Describing the key aspects of a gearbox.

3 Operational Mode Modelling

Dymola has discrete event modelling capability including libraries that handle Petri Nets e.g. StateGraph library. We use the more sophisticated library StateGraph2 to implement the operational mode transition modelling presented in Figure 2. Modelica 3.3 has been shown to support a state machine approach by way of the Synchronous feature (Elmqvist et al., 2012) but currently there is no graphical library ready to be used. Given the focus of this paper is implementation of a graphical approach Synchronous was not investigated further.

We term the control logic which handles the operational mode of the model a supervisor. The gearbox model who is controlled by the supervisor is known as the supervised. This is depicted in Figure 4.

In the rest of this section we present various alternative implementations of the supervisor and evaluate how successful they are.

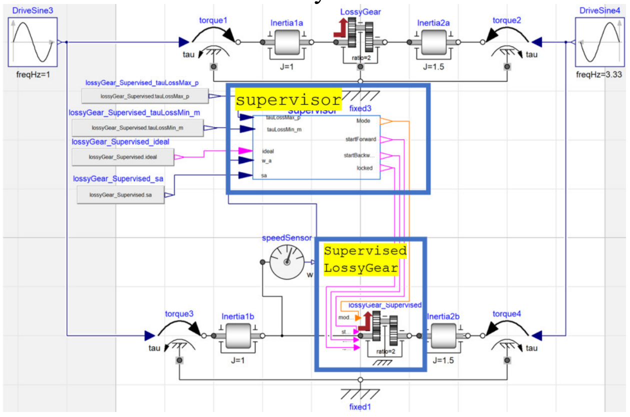


Figure 4. Supervisor for Supervised LossyGear to handle operational mode modelling.

3.1 Supervisory Control using StateGraph2

StateGraph2 is presented in (Otter, Malmheden, Elmqvist, Matsson, & Johnsson, 2009) to create "hierarchical state machines in combination with any Modelica model". Given the state representation we present in Figure 2 it was felt that this would offer a promising option for clearly modelling the operational modes of the gearbox.

Two supervisor models are presented Figure 5 (implementing the state transitions of Figure 2 as explicitly as possible) and Figure 6 (implementing the state transitions of Figure 2 where each state is provided two state steps).

The first model, Figure 5, does succeed in a creating a control architecture which explicitly shows all the possible transitions between the states. However due to the very large density in the transitions of mode (top left of Figure 2) it does not make the model clear. Further the model fails due to chattering.

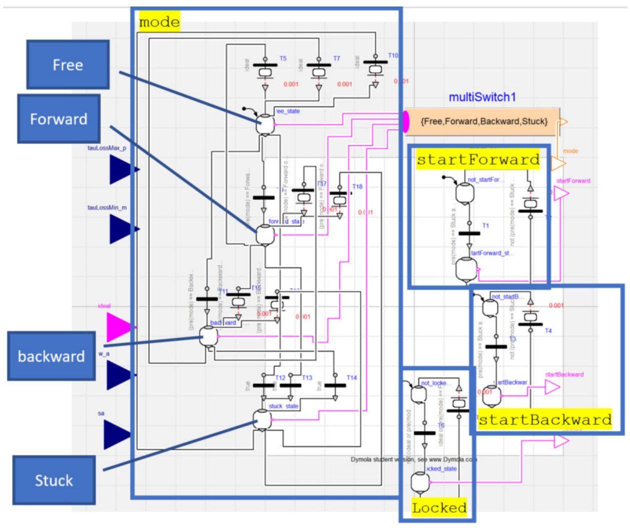


Figure 5. Implementing the state transitions of Figure 2 as explicitly as possible. Fails due to chattering.

The second model, Figure 6, takes a different approach where the mode states are modelled in a similar way to the outer edge states of Figure 2. This method is less preferred as it would make it possible for conflicting states to be held at the same time (e.g. forward and backward) if the control logic was not implemented correctly to prevent it.

Through this studies we have found that the StateGraph2 models in Figure 5 and Figure 6 capture the functional requirements depicted as states and state transitions in Figure 2 quite straightforwardly, because they can represent the structure of the state transition diagram well. Further, the simulation results of the gear behaviour using the model depicted in Figure 6 follow similar trends to those of the LossyGear model. However, this approach fails to achieve identical results to that of the reference target LossyGear model. This is regarded as the limitation to the StateGraph2 perhaps based on a timing issue.

Based on the limited success of using StateGraph2 decided to investigated alternative ways of supervisory modelling.

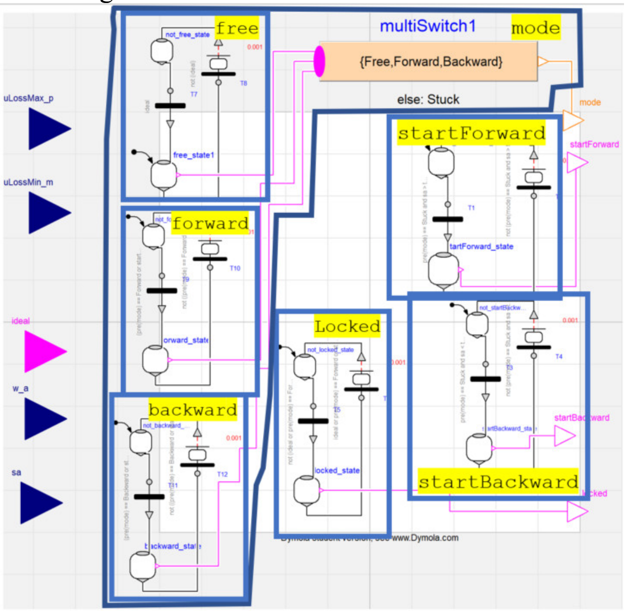


Figure 6. Implementing the state transitions of Figure 2 where each state is provided two state steps. Fails to produce identical results to the original LossyGear model.

3.2 Supervisory Control by Logical Modelling

Alternative methods for implementing the supervisor (depicted in Figure 7 and Figure 8) involve explicitly modelling the supervisory controller are described in this section. Both of which were deemed highly successful as both were able to produce identical results to the reference LossyGear model (see Figure 9).

```
equation

  startForward = pre(mode) == Stuck and sa > tauLossMax_p/unitTorque
    or initial() and w_a > 0;

  startBackward = pre(mode) == Stuck and sa < tauLossMin_m/unitTorque
    or initial() and w_a < 0;

  locked = not (ideal or pre(mode) == Forward or startF
    or pre(mode) == Backward or startBackward);

  mode =
    if ideal then Free
    else
      (if (pre(mode) == Forward or startForward) and w_a > 0
          then Forward
       else if (pre(mode) == Backward or startBackward) and w_a < 0
          then Backward
       else Stuck);
```

Figure 7. Equations implementing the supervisor.

Figure 7 shows the equations used to determine the state of the gearbox based on various physical phenomena as described in (Pelchen et al., 2002) while Figure 8 depicts the same equations implemented using Modelica.Blocks.Logical.

Given the success of these approaches they are used for the rest of this paper. However, the equation-based approach lacks the visual depiction of state available with the StateGraph2 approach.

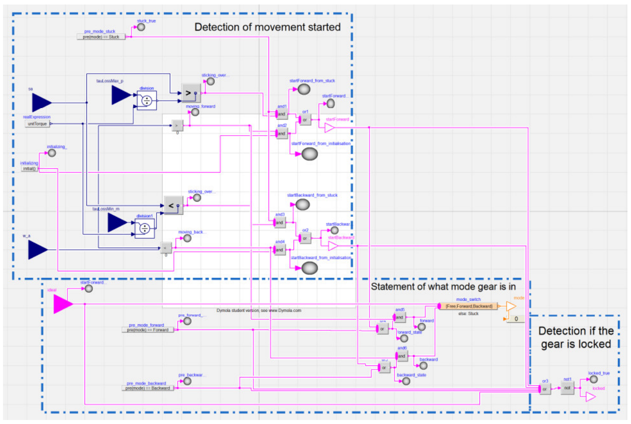

Detection of movement started
Statement of what mode gear is in
Detection if the gear is locked

Figure 8. Implementing the supervisor using components from the library Modelica.Blocks.Logical.

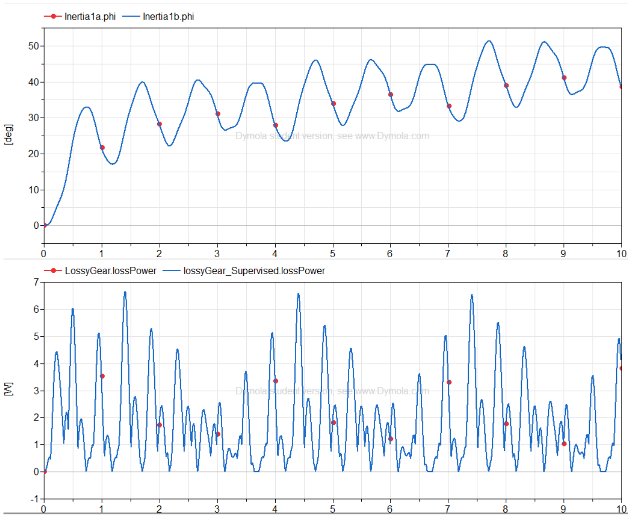


Figure 9. Comparing results of the supervisor/supervised LossyGear combination and the original model. **Top:** Position. **Bottom:** Loss power.

Now, we have achieved modelling of functionality, now it is time to combine the design requirements into the model.

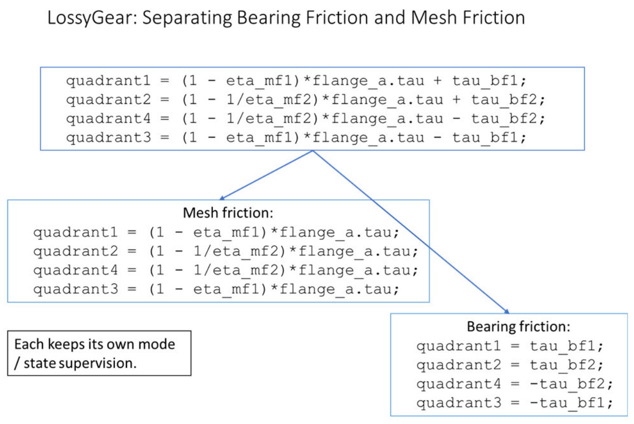
4 Composable Modelling of the Gearbox

This section describes the general approach we have developed for composing a gearbox based on the functionality described in Section 2 which lead to the creation of the key aspects of the gearbox in Figure 3 and the operational mode approach presented in Section 3.

4.1 Phenomena Decomposition

As shown in Figure 3, two of the primary functional components are the two sources of energy loss, the mesh friction loss and the bearing friction loss.

The paper (Pelchen et al., 2002) provides a very detailed description of LossyGear where speed dependent energy losses are parameterised in the form of mesh efficiency and bearing friction. **Error! Reference source not found.** depicts how the key equations representing these in the reference LossyGear model can be separated from one another to build separate models of mesh and bearing friction. The separation then offers more insight these components.


LossyGear: Separating Bearing Friction and Mesh Friction

```
quadrant1 = (1 - eta_mf1)*flange_a.tau + tau_bf1;
quadrant2 = (1 - 1/eta_mf2)*flange_a.tau + tau_bf2;
quadrant4 = (1 - 1/eta_mf2)*flange_a.tau - tau_bf2;
quadrant3 = (1 - eta_mf1)*flange_a.tau - tau_bf1;
```

Mesh friction:

```
quadrant1 = (1 - eta_mf1)*flange_a.tau;
quadrant2 = (1 - 1/eta_mf2)*flange_a.tau;
quadrant4 = (1 - 1/eta_mf2)*flange_a.tau;
quadrant3 = (1 - eta_mf1)*flange_a.tau;
```

Bearing friction:

```
quadrant1 = tau_bf1;
quadrant2 = tau_bf2;
quadrant4 = -tau_bf2;
quadrant3 = -tau_bf1;
```

Each keeps its own mode / state supervision.

Figure 10. Modifying the equations of LossyGear to only focus on mesh friction or bearing friction.

4.2 Independent Power Losses and Independent Supervisor

In this section, we extend the study such that separate supervisors are investigated to handle the operational modes of the mesh and bearing frictions.

The benefit of such composable method enables the model user to switch on/off the supervisor to make the component run free with no friction. This is helpful to identify the location of the energy loss and improve the design e.g. if the loss occurs at bearing, then lubricating the bearing should be considered.

4.2.1 Separate supervised components

Figure 11 shows a composed gearbox model of Mesh Loss, Bearing Loss and with separate supervision for each of those components. This simulation produces results (shown in Figure 12) identical to a LossyGear model, but now with the benefit that bearing and mesh friction are separated such that their associated losses can be interrogated separately and the heat associated with them can be directed to different locations. Hence, the work is considered a success given we have created a composed gearbox model with identical behaviour to that of the reference LossyGear model.

It should be noted that the order of the components and to which the ratio is applied matters for producing identical results to the reference LossyGear model. In that Mesh Friction component should have a ratio of 1 and Bearing Friction a ratio equal to what the gear is to represent.

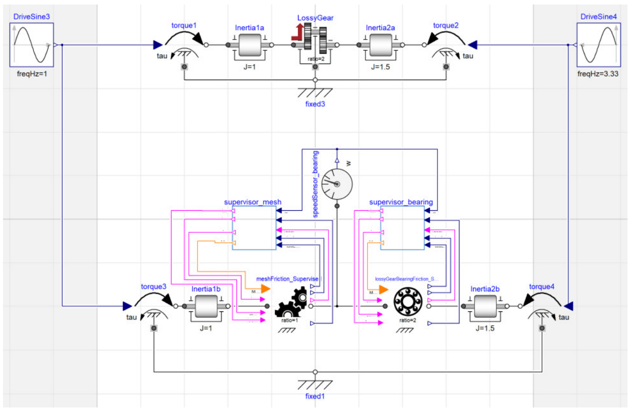


Figure 11. Composed gearbox model of mesh loss, bearing loss and with separate supervision for each of those components.

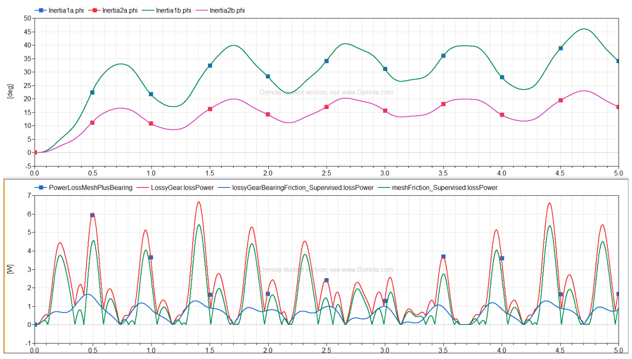


Figure 12. Verifying the composed gearbox (Figure 11) compared to a LossyGear model. **Top:** Position. **Bottom:** Loss power.

4.3 Application of the Combined Model

Given the composable nature of the new components it is possible to insert additional components into the LossyGear model. This is demonstrated in Figure 13 where a spring damper is added between the components. As shown in Figure 14 such an approach can enable the modelling of flexibility on the shaft resulting in different speeds between the components.

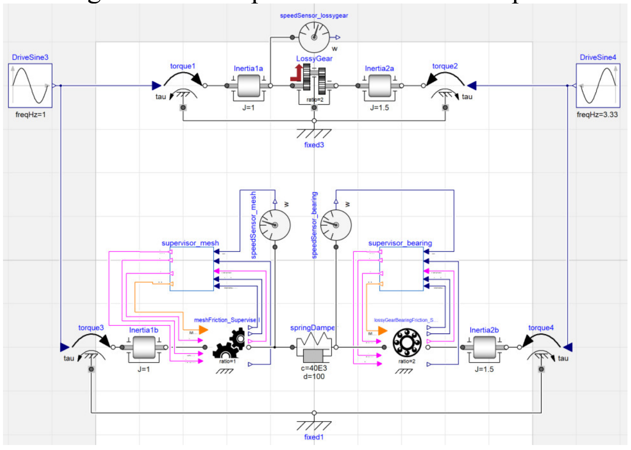


Figure 13. Adding a spring damper between the mesh friction and bearing friction.

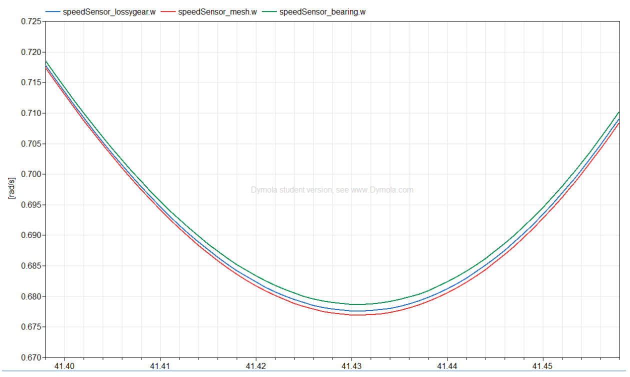


Figure 14. Comparison of shaft speeds of the model shown in Figure 13. With LossyGear compared to a composed model of mesh and bearing friction with an additional spring damper.

5 Conclusions and Further Work

This paper set out to develop and use a composable approach to create cyber-physical models driven by the requirements. We take the gearboxes design as an example to illustrate the workflow of building models with different fidelity. In particular:

- Models should be composed based on the functional requirements of the component.
- The supervisor part is designed explicitly according to the functional requirements using StateGraph2, logical blocks and Modelica text.
- While the physics part (the supervised) is kept independent but interacting with the supervisor.
- The proposed structure increases the freedom to switch on/off the mode transition functionality at the mesh and bearing parts independently, which enables the detailed analysis of the performance of individual components.

The resultant model achieves the same result as the reference LossyGear model.

5.1 Further Work

Further work to be done includes extending the approach to a general cyber-physical modelling method and applying the approach to another domain (such as electrical).

Investigation into application of Modelica 3.3 Synchronous is also worth consideration.

References

Ackoff, R. L. (1971). Towards a system of systems concepts. *Management Science*, *17*(11), 661–671.

Adcock (EIC), R. D. (2017). *The Guide to the Systems Engineering Body of Knowledge (SEBoK) v. 1.9*. Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society. Retrieved from http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)

Blockley, D. I., & Godfrey, P. (2000). *Doing it differently: Systems for rethinking construction*. Thomas Telford.

Elmqvist, H., Gaucher, F., Matsson, S. E., & Dupont, F. (2012). State machines in modelica. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany* (pp. 37–46). Linköping University Electronic Press. Retrieved from http://www.ep.liu.se/ecp/076/003/ecp12076003.pdf

Elmqvist, H., Mattsson, S. E., & Otter, M. (2001). Object-oriented and hybrid modeling in modelica. *Journal Européen Des Systèmes Automatisés*, *35*(4), 395–404.

Griffin, M. D. (2010). How do we fix System Engineering? In *61st Annual International Congress, Prague, Czech Republic* (Vol. 27). International Astronautical Federation (IAF).

Hitchins, D. K. (2008). *Systems engineering: a 21st century systems methodology*. John Wiley & Sons.

ISO/IEC. (2007). *ISO/IEC 42010: 2007-Systems and software engineering–Recommended practice for architectural description of software-intensive systems*. Technical report, ISO.

Otter, M., Elmqvist, H., & Mattsson, S. E. (1999). Hybrid modeling in Modelica based on the synchronous data flow principle. In *Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on* (pp. 151–157). IEEE. Retrieved from http://ieeexplore.ieee.org/abstract/document/808640/

Otter, M., Malmheden, M., Elmqvist, H., Matsson, S. E., & Johnsson, C. (2009). A new formalism for modeling of reactive and hybrid systems. In *Proceedings of the 7th International Modelica Conference; Como; Italy; 20-22 September 2009* (pp. 364–377). Linköping University Electronic Press.

Parey, A., & Tandon, N. (2003). Spur gear dynamic models including defects: A review. *The Shock and Vibration Digest*, *35*(6), 465–478.

Pelchen, C., Schweiger, C., & Otter, M. (2002). Modeling and simulating the efficiency of gearboxes and of planetary gearboxes. In *Proceedings of 2nd International Modelica Conference* (pp. 257–266). Retrieved from http://elib.dlr.de/11828

Schlegel, C., Hösl, A., & Diel, S. (2009). Detailed loss modelling of vehicle gearboxes. In *Proceedings of the 7th International Modelica Conference; Como; Italy; 20-22 September 2009* (pp. 434–443). Linköping University Electronic Press. Retrieved from http://www.ep.liu.se/ecp/article.asp?issue=043&article=48

Toward the actual model exchange using FMI in practical use cases in Japanese automotive industry

Yutaka Hirano[1], Junichi Ichihara[2], Haruki Saito[3], Yosuke Ogata[4], Takayuki Sekisue[5], Satoshi Koike[6],

[1] Toyota Motor Corporation, Japan {yutaka_hirano@mail.toyota.co.jp},
[2] AZAPA Co. LTD, Japan {junichi-ichihara@azapa.co.jp},
[3] Nissan Motor Corporation, Japan {haruki-s@mail.nissan.co.jp},
[4] Siemens K.K., Japan {yosuke.ogata@siemens.com},
[5] ANSYS Japan K.K., Japan, {takayuki.sekisue@ansys.com},
[6] DENSO CORPORATION{ SATOSHI_L_KOIKE@denso.co.jp}

Abstract

The working group in JSAE (Society of Automotive Engineers of Japan) proposed and published a guideline about a method using adapters to enable the connection of FMUs in acausal modeling tools in 2015. After that, more combination of exporting tools and importing tools of FMUs were tested for more realistic models. The success of the method was confirmed in the case of using FMI for Model Exchange (ME). Additionally the connection of FMUs generated from various modeling tools was tested using FMI for Co-Simulation (CS) for a full-vehicle test model. Through the above-mentioned activities, we obtained much knowledge about utilizing FMI for practical model exchange in the industry. Finally the requests for future realization of FMI are described.

Keywords: FMI, Model Exchange, Co-Simulation, Benchmark Model, Numerical Stability

1 Introduction

Importance of utilizing simulation is increasing for the development of automotive systems. Both high functionality and high reliability are required while the development time is becoming shorter. For large-scale and multi-domain development of automotive systems, environment for connecting models developed in various organizations is increasing its importance (Sekisue *et al*, 2013). For this purpose, FMI (Functional Mockup Interface) is one of the most important technologies to realize the exchange of models developed by various tools. (Blochwitz *et al*, 2011) (FMI)

Upon above background, the Working Group about Research for FMI Utilization and Expansion in the Committee on Automotive Control and Model in JSAE has been continuing its efforts to 1) find problems about practical usage of FMI for exchanging models in actual development, 2) research about the measures to solve the problems, 3) make guidelines about utilizing FMI for exchanging models and 4) educate and encourage users about using FMI for exchanging models in Japan.

In the previous research a method to use special adaptor models to divide and make FMUs of the sub-models and to connect FMUs in acausal modeling environment was proposed (Hirano *et al*, 2015). The method was validated by a benchmark model in the case of using ME. In this paper, many combinations of tools to create FMUs and connect FMUs were tested for the proposed method. There were some problems found in those tests not only when using ME but also when using CS. Another test model of a full vehicle model to connect FMUs of each sub-system of the vehicle using CS was tested in various combinations of the tools. Through those activities, we encounter the important points about connecting FMUs in the practical use cases such as handling of algebraic loops and stability of simulation for stiff systems. Finally future plans about our activities will be described.

2 Method to use adaptor models

In the previous research, a method to use adaptor models of electric domain and mechanical domain as shown in **Figure 1** was introduced (Hirano *et al*, 2015).

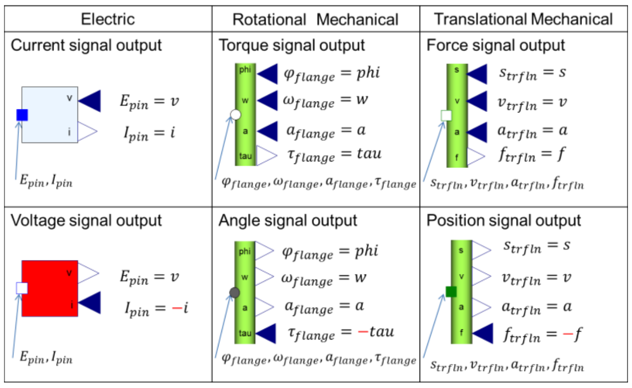


Figure 1. Adapter models to connect acausal physical port and causal signal ports

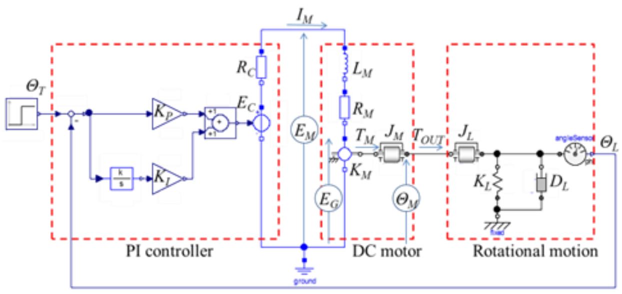


Figure 2. Benchmark model (Simple control system)

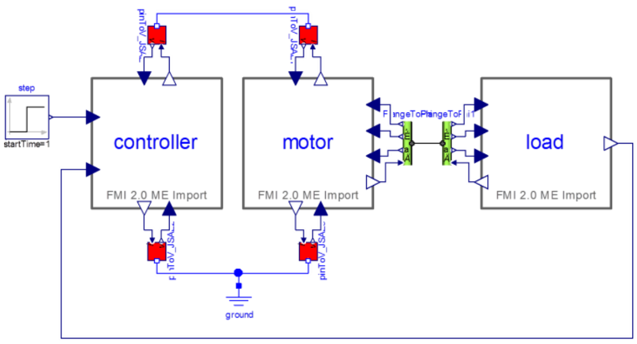


Figure 3. System model using 3 FMUs

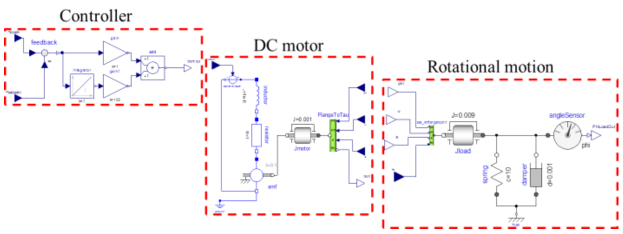


Figure 4. Modified connection of the benchmark model

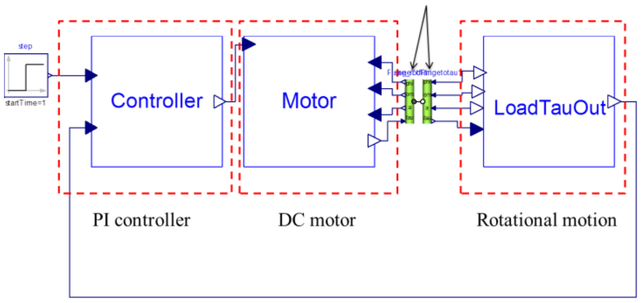


Figure 5. Modified benchmark model with 3 FMUs

This method was tested by a benchmark model shown by **Figure 2**. Three sub-systems of the benchmark model indicated by dotted square in **Figure 2** were converted to FMUs by connecting the adaptor models for each inputs and outputs. Then three FMUs were connected by using corresponding adaptors of each inputs and outputs as shown in **Figure 3**. Simulation results of the original acausal model and the model using 3 FMUs were compared. It was confirmed that the results were almost identical. Though, in this case, only one Modelica tool was used to create and connect FMUs.

This time many combinations of the tools for creating FMUs and connecting FMUs were tested. First, it became clear that some combinations of tools failed to simulate the benchmark model when using the adaptors for electric domain. Thus, the benchmark model was a little modified about its connection. In this case, the electric connectors were not used but just controller command as the voltage input to the motor was used as shown in **Figure 4**. After generating FMUs from each subsystem of **Figure 4** by various tools, each FMUs were connected in the system model as shown in **Figure 5**. Five tools were used to create FMUs and four tools were used to connect FMUs. Finally the result of whether the simulation worked became as shown in Table 1. Here, the meanings of the symbols in each column are as follows.

・for 'adaptor':

Δ - Contains only adaptors for some physical domain

× - Could not create adaptor models

・for 'Run':

Δ - Do not use adaptors and connect directly

The number in the lower part of each cell means the ratio of calculation time of each model with FMUs compared with that of the original acausal model.

Almost all tools could create and run FMUs using the adaptor models. Only Tool D could not create adaptor models. Also Tool D could run FMUs created by other tools by connecting causal signals directly but not using the adaptors. Also it became clear that the calculation time became very long in some combinations of the tools. The reasons of this phenomenon were assumed as 1) that the compatibility of the master algorithm was bad for the embedded FMUs (ex. the algorithm for solving the algebraic loops and /or solving the stiff systems), 2) difficulty of determining the initial values when the re-initialization occurred, 3) improper setting of the simulation parameters of the solvers, and so on. But the details are not clear yet because these investigations need deep insight of the algorithms of each tool but it is not easy for the users' side.

Table 1 Results of connecting FMUs and run

Create FMUs		adaptor	Connect FMUs			
			ToolA	ToolB	ToolC	ToolD
			Run	Run	Run	Run
ToolA	✓	Δ	✓	✓	✓	Δ
			6.45	22.22	8.00	18.62
ToolB	✓	Δ	✓	✓	✓	Δ
			7.82	22.22	5.00	2.25
ToolC	✓	Δ	✓	✓	✓	Δ
			505.36	18.22	5.00	2.84
ToolD	✓	×	✓	✓	✓	Δ
			8.55	68.89	5.00	2.90
ToolE	✓	Δ	✓	✓	✓	Δ
			2428.57	35.56	36.00	2.08

Above results were obtained in the case of using ME. We also tried to test same cases by using CS. But for CS, it became clear that the adaptor model of the rotational mechanics domain using all of rotational angle, velocity and acceleration resulted in the simulation error because of over constrained condition. Thus it was suggested that we should modify the adaptor model to use just one state variable of rotational motion as the across variable. This method will be investigated in near future.

Additionally there was a remark from Modelica Association that adaptor models in many physical domains have been added to the master branch of the Modelica Standard Library (MSL) (https://github.com/modelica/Modelica) and will become available in the next release of the MSL.

3 Connection of full vehicle model using CS

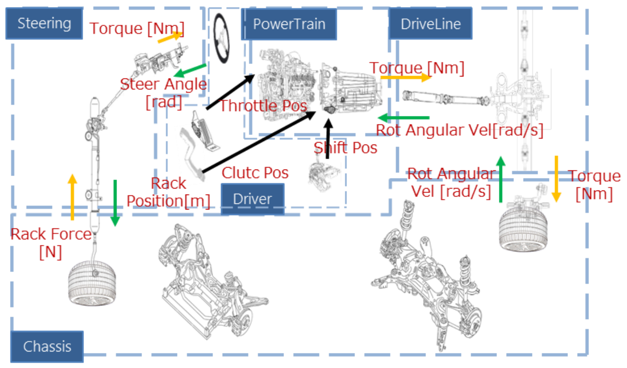


Figure 6. Full vehicle benchmark model

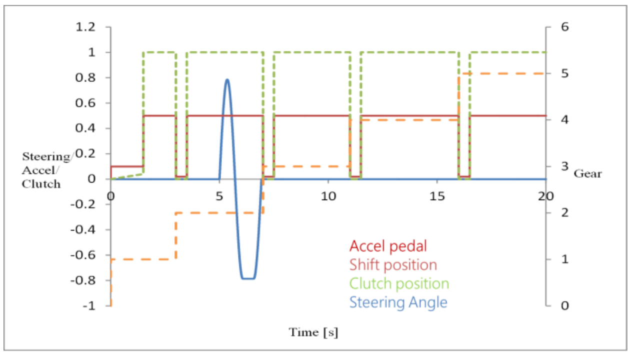


Figure 7. Driver's maneuver inputs to the full vehicle model

Table 2 Characteristics of each sub-system of the full vehicle model

Sub-system	Number of state variables	Dynamics
Driver	0	-
Power train	63	2.3 [kHz]
Drive line	6	3.2 [kHz]
Chassiss	40	10 [Hz]
Steering	12	240 [Hz]

For the research of connecting sub-systems of the full vehicle model by CS, the benchmark model shown in **Figure 6** was used. In this case, signals between FMUs were connected directly without using the adopter models mentioned above. Sub-systems consist of the steering system, the powertrain system, the driveline system, the chassis system and the driver model. Physical variables transferred between each sub-system and their directions are shown in **Figure 6.** This selection of physical variables and their directions were decided by the guideline about the model interface of full vehicle model written by a Working Group of METI (Ministry of Economics, Trade and Industries) of Japanese Government. (METI, 2017)

Driver's open loop maneuver inputs to the full vehicle model were decided as shown in **Figure 7**. Table 2 shows the number of the state variables and maximum frequency of the dynamics of each sub-systems. Simulation tests were done by following sequences.

1. Simulate the full vehicle model of original acausal model in one host tool (Tool A).
2. Replace each sub-system model to FMUs generated by the host tool (Tool A) and simulate the full vehicle model with replaced FMUs in the host tool (Tool A).
3. Replace the sub-system models to FMUs generated by different tools of the host tool. Each tool to generate the FMUs of sub-systems are shown in Table 2. Then simulate the changed model in the host tool (Tool A).
4. Change the host tool of the full vehicle model to Tool B, C and D using FMUs from the different tools mentioned in the above sequence 3.

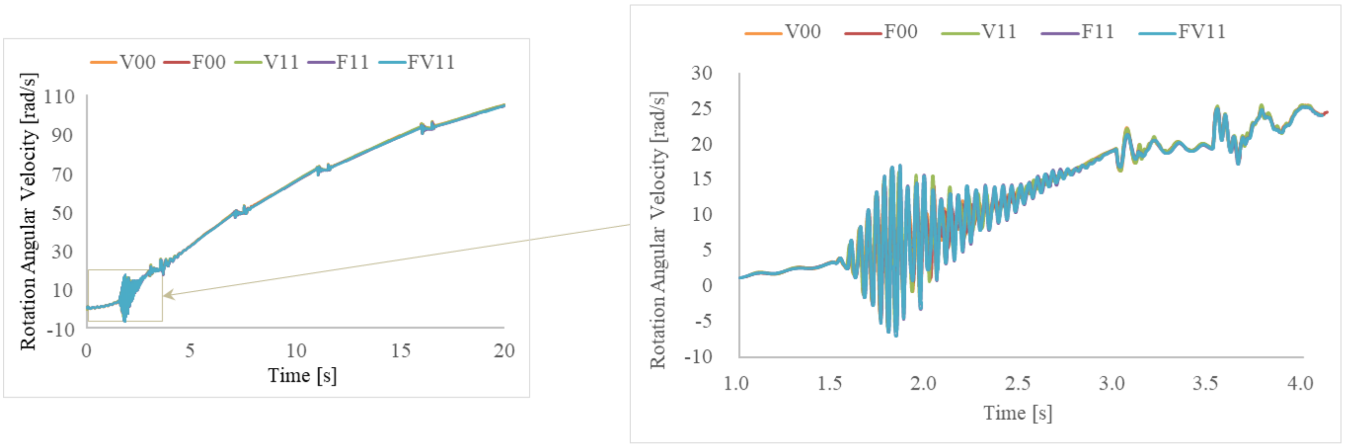
Here, tested host tools (master of CS) are as follows. Tool A: Amesim, Tool B: Simulink + Modelon FMI Toolbox, Tool C: Dymola, Tool D: SimulationX. (Please note that these tools are different from that of the chapter 1.)

Table 3 shows the various test conditions for the above test sequence 1 and 2. Both of the variable step solver (LSODA) and the fixed step solver (RK4) were compared. **Figure 8** shows one example of the results. There were some oscillation caused by the stepwise change of the driver's maneuver inputs. But the results of all the cases were almost identical regardless of whether FMUs were used or not. Also the results were not changed so much by the changes of the simulation algorithm and its parameter if the integration step size and the communication step size (described as 'interval' in the table) of FMUs were small enough. Also for the above test sequence 3 and 4, many cases of changing integration algorithm and communication step size were tested as shown in Table 4. The results of all the cases in Table 4 were almost identical. But there were a slight change in the oscillation of the results as shown in **Figure 9** when the communication step size of FMUs was changed. It was confirmed that larger oscillation of the simulation result appeared for the larger communication step size. As shown in Table 4, there were trade-off between the calculation time of the host tool and the preciseness and stability (oscillation) of the simulation results according to the selection of the integration algorithm, its calculation interval and the communication step size of FMUs by CS. **Figure 10** shows one simulation result of the sequence 4. The results of the simulation were almost identical when different CS master tool were used.

Table 3. Simulation test conditions of the full vehicle benchmark model (Sequence 1 and 2)

	Case	Setting	Driver	Power Train	Drive Line	Chassis	Steering
Original (Sequence 1)	V00	Solver	LSODA				
		Interval	-				
	F00	Solver	RK4 5E-5				
		Interval	5E-5				
FMU models (Sequence 2)	V11	Solver	LSODA	LSODA	LSODA	LSODA	LSODA
		Interval	5E-5	5E-5	5E-5	5E-5	5E-05
	F11	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 5E-5
		Interval	5E-5	5E-5	5E-5	5E-5	5E-5
	FV11	Solver	RK4 5E-5	RK4 5E-5	LSODA	LSODA	LSODA
		Interval	5E-5	5E-5	5E-5	5E-5	5E-05

🟧 Variable Time Step Solver
🟦 Fixed Time Step Solver



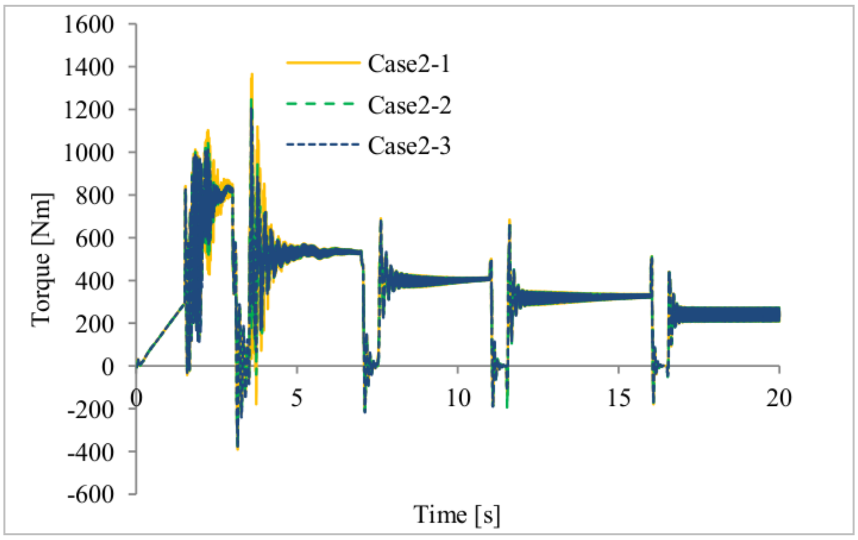
Driveshaft Rotation Angular Velocity (Chassis)

Figure 8. Simulation results of the full vehicle model (Sequence 1 and 2)

Table 4. Simulation test conditions of the full vehicle benchmark model (Sequence 3 and 4)

		Slave Tool FMU					**Master Tool CPU time***
Case	**Setting**	**Tool A (Amesim)**	**Tool A (Amesim)**	**Tool B (Simulink + Modelon Tool)**	**Tool C (Dymola)**	**Tool D (SimlationX)**	**Tool A (Amesim)**
		Driver	**Power Train**	**Drive Line**	**Chassis**	**Steering**	**Rk4, 5e-5**
1	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 1E-3	CVODE	6.2
	Interval	5E-5	5E-5	5E-5	1E-3	1E-3	
2-1	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 5E-5	CVODE	42.5
	Interval	5E-5	5E-5	5E-5	1E-3	1E-3	
2-2	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 5E-5	CVODE	44.6
	Interval	5E-5	5E-5	5E-5	5E-4	5E-4	
2-3	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	RK4 5E-5	CVODE	86.4
	Interval	5E-5	5E-5	5E-5	5E-5	5E-5	
3-1	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	DASSL	CVODE	129.7
	Interval	5E-5	5E-5	5E-5	1E-3	1E-3	
3-2	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	DASSL	CVODE	264
	Interval	5E-5	5E-5	5E-5	5E-4	5E-4	
3-3	Solver	RK4 5E-5	RK4 5E-5	RK4 5E-5	DASSL	CVODE	2716
	Interval	5E-5	5E-5	5E-5	5E-5	5E-5	

* CPU time shows the ratio of calculation time standardized by that in the case of acausal original model



Rear Left Driveshaft Torque (DriveLine)

Figure 9. Simulation results of the full vehicle model for the different communication step size

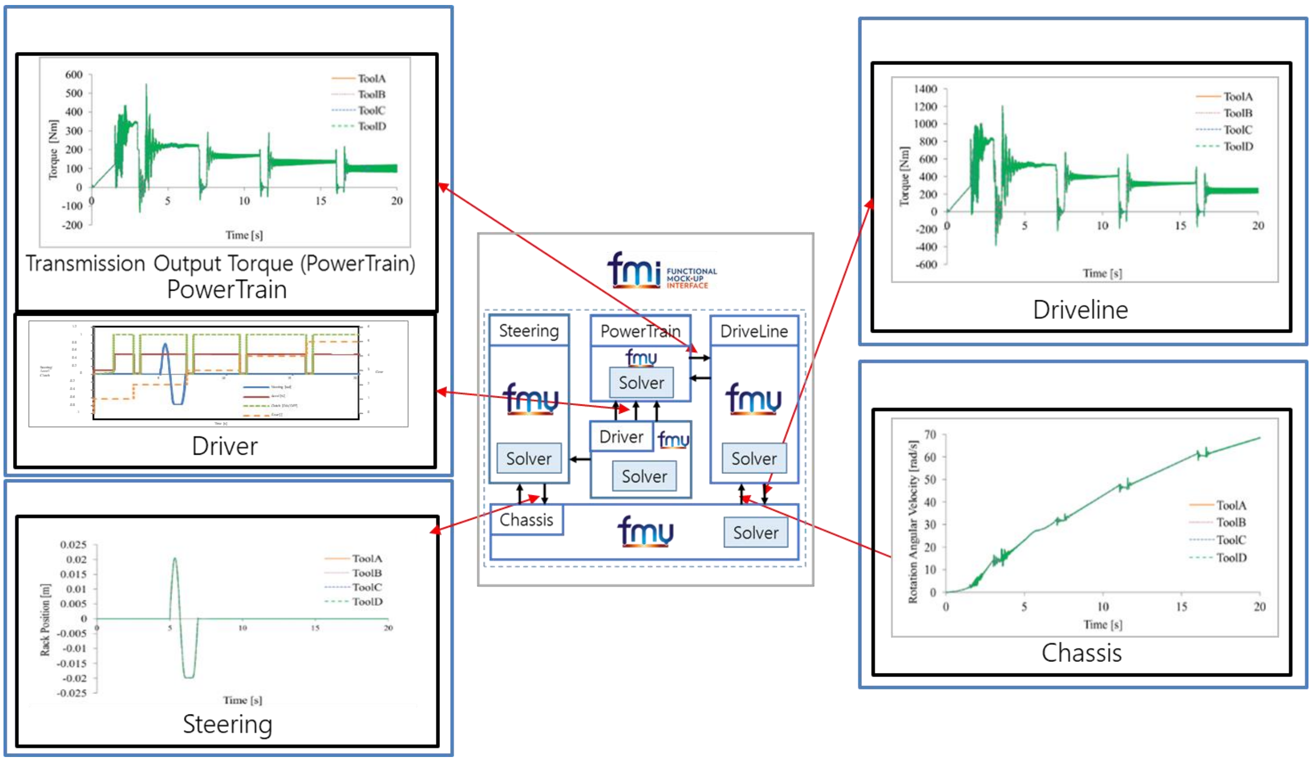


Figure 10. Simulation result of the full vehicle model (Sequence 4, Case 3-3)

4 Important points about connecting FMUs

Through the above researches, we obtained some knowledge and important points about connecting models by FMI. There are three major phases when we try to connect and exchange models by FMI.

4.1 Creation and export / import of FMU

Sometimes there still occurred errors by inconsistency between tools about exchanging information about the models.

1) Some tools could not read the XML information when there were @ character in the name of input / output / state variables.
2) There were inconsistency about type and unit of variables between some tools.

It is desirable that those specifications will be unified between every tool and also checked by FMU compliance checker as soon as possible.

4.2 Connection of FMUs in the host tool

Even when the import and connection of FMUs was successful, there were problems of inconsistency about the direction and sign of the signals exchanged between FMUs and the host tool. This is because of that FMI is based on the causal signal interconnection. It is necessary that those specifications should be identical and agreed with the both parties of exporting FMUs and connecting FMUs.

For the full vehicle model, the Working Group of METI about unifying the definition of signals exchanged between automotive systems has recently announced the standard guideline of the model interconnection for automotive industries in Japan. The benchmark models of our research also keep this guideline. (Some members of both Working Groups of METI and JSAE are same.)

4.3 Stability and preciseness of the simulation

When running the simulation, there are many points we should be careful. The points differ between the case of ME and CS.

4.3.1 In the case of using ME

There are mainly three problems when using ME.

1) Generation of algebraic loop

In the case of ME, there is no time delay between the inputs and outputs of FMUs. Thus there occurs a possibility that the 'real' algebraic loop is generated when connecting different FMUs as shown in **Figure 11**. On the other hand, in the case of CS, the possibility of the generation of the algebraic loop is low because there always is time delay of communication step size as shown in **Figure 11**. Here, the 'artificial algebraic loop' means that if there is no information about the causality of inputs and outputs of FMUs provided as the additional information by XML file, the solver of the master tool tries to solve the loop by iterative loop solver algorithm. This can be happened in the both cases of CS and ME.

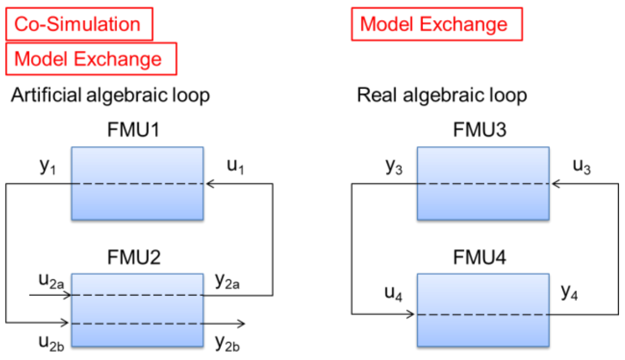


Figure 11. Two types of the algebraic loop

When the real algebraic loop occurs in the case of using ME, it is necessary to solve by some algebraic loop solver using an iteration method. Whether it would be successful to solve the algebraic loop and also the efficiency and the correctness of the simulation heavily depends on the performance of the algebraic loop solver of each host tool. Also some users try to avoid the algebraic loop by inserting time delay unit into the loop. Though this method is dangerous. If the absolute value of the loop gain exceeds one, then the calculation will diverse.

2) Selection of the solver

Because only the solver of the host tool which import FMUs can be used when using ME, the selection of the solver for the numerical integration is very important. If the time constants of each FMU differs widely, then the total system connecting FMUs becomes stiff system. In such a case, it is necessary to use an integration algorithm which can cope with the stiff system. Also the parameter of the solver such as tolerance and/or step size should be selected carefully. When using fixed step solvers, the step size should be selected so that all the eigenvalue of the system should be within the stable region of z-plain as shown in **Figure 12**. Here, λ is the minimum eigenvalue and Δt is the calculation step size. However, in the current situation, there is no guideline or suggestion about the selection of the proper solver and its parameters. This results in the difficulty of using FMI for ME as the method of connecting and exchanging the FMUs between different organizations. One remedy would be to make the output of the information about the time constants such as Jacobian as mandatory and let the every tool to use this information to recommend the best selection of the solver and its parameter to the users.

3) Initialization and event handling

Inconsistency of initial values of the state variables between FMUs is also a problem. Currently it is not mandatory to specify the initial values of the state variables. It is highly desired that this specification would be unified and become mandatory for all the tools. Also the ability of event detection and re-initialization of the state variables is not same between tools. Sometimes this results in the difference of the simulation results.

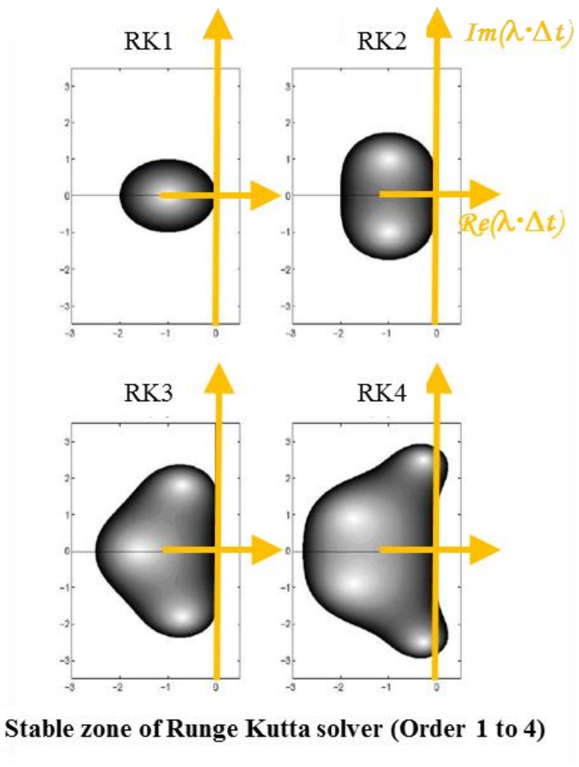


Figure 12. Stable zone of Runge Kutta solver

4.3.2 In the case of using CS

As shown in the previous sub-section, there is no possibility that the real algebraic loop would be generated when using CS. Also because each FMU has the proper solver from the master tool (if it was tested well), the problem of the selection of the solver is not so difficult as in the case of using ME. It is supposed that the benchmark test using full vehicle model by CS mentioned in chapter 3 worked well because of these reasons.

However, there always occurs the problem of time delay of communication step size (CSS) for CS. Because the dead time of CSS acts same as the sample and hold unit of a discrete system, the simulation tend to become unstable if the CSS is too big. Concerning the selection of CSS, there is a trade-off between calculation time and the preciseness of the simulation as mentioned in chapter 3. To consider the proper value of the CSS, the information about the eigenvalues of each FMU is also important as same as in the case of ME. The measure to solve this problem would be same as mentioned above.

5 Future plans

To figure out what will be necessary for our future plan, we gathered answers of questionnaires from the members of JSAE and tool users. **Figure 13** shows the distribution of belonging industries of the respondents. **Figure 14** shows the current status of on-the-job usage of FMI of them. About 22% of the respondents were utilizing or trying FMI. On the other hand, 31% of them didn’t know FMI yet.

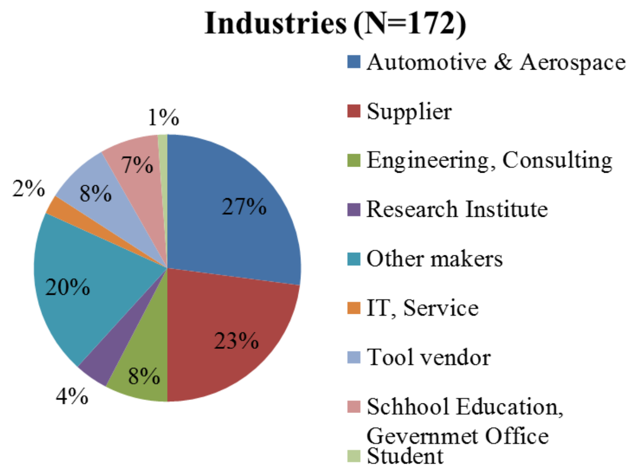


Figure 13. Distribution of industries of respondents

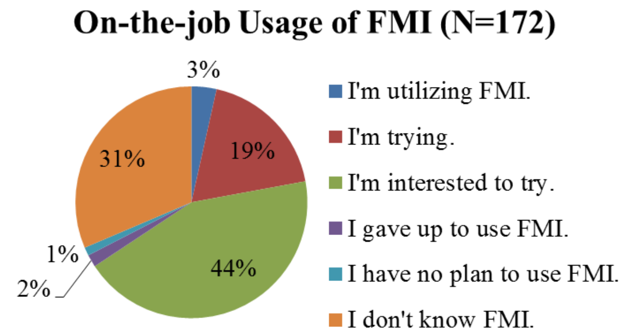


Figure 14. On-the-job usage of FMI

For the actual users of FMI, the purpose of using FMI was as shown in **Figure 15**. (Multiple number of the answers were possible.) It became clear that about 40% was for model exchange between organizations by either of ME or CS. Also utilization for HIL / Real time simulation was higher than expected. **Figure 16** shows problems and requests about FMI from the respondents. The similar problems with our research such as the problems about numerical solvability and stability, inconsistency of parameters and signal specification were reported. Also the requests about speed-up of calculation and large scale connection of FMUs including hierarchical connection were clarified. Considering these problems and requests, we are extending the activity of our Working Group to 1) extension of physical region to evaluate (to include thermal system and fluid system), 2) Coping with hierarchical connection and speed-up of simulation, 3) Updating JSAE guideline, 4) Collaborating more with the related groups such as METI, Modelica Association FMI Project, etc. and 5) Organizing seminars and an Organized Session of JSAE Conference to educate and expand knowledge of FMI users in Japan.

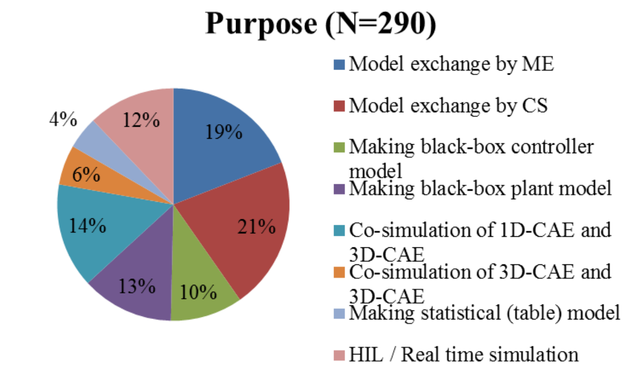


Figure 15. Purpose of utilizing FMI

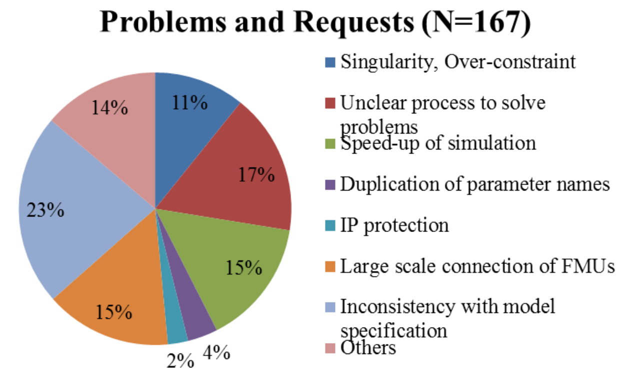


Figure 16. Problems and requests about FMI

6 Conclusion

Many tests using benchmark models for both of the cases of using ME and using CS were done to confirm the properness of using FMI for connecting and exchanging models in automotive industry in Japan. Though there still are many remained problems about using FMI for practical model exchange, most of the tests were successful. Also the reasons of the problems were estimated and countermeasures for those problems were proposed. It is highly desirable that the proposal from us will be realized in the future specification of FMI and also in the implementation of the related tools.

References

T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold1, D. Neumerkel, H. Olsson, J.-V. Peetz, S. Wolf, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, *Proc. of the 8th International Modelica Conference*, 2011

FMI. The functional mockup interface. http://www.functional-mockup-interface.org/.

Y. Hirano, S. Shimada, Y. Teraoka, O. Seya, Y. Ohsumi, S. Murakami, T. Hirono, T. Sekisue, Initiatives for acausal model connection using FMI in JSAE, *Proc. of the 11th International Modelica Conference*, 795-801, 2015. Doi:10.3384/ecp15118795

JSAE Committee on Research of Model Development and Circulation Methods Based on Global Standardized Description, Guideline of Model Connection using FMI in Acausal Modeling Tools, 2014 (in Japanese. Available online as *http://www.jsae.or.jp/tops/topics/1241/1241-1A.pdf).*

METI (Ministry of Economics, Trade and Industry) of Japan, The Study Group for Ideal Approaches to Model Utilization in the Automobile Industry Releases a Compilation of Discussion Results, 2017 (Available online as http://www.meti.go.jp/english/press/2017/0331_004.html, Guidelines are available online but only in Japanese as http://www.meti.go.jp/press/2016/03/20170331010/20170331010-1.pdf and http://www.meti.go.jp/press/2016/03/20170331010/20170331010-3.pdf)

T. Sekisue, K. Tsuji, M. Ogawa, T. Fukada, K. Tanimoto, S. Hikida, M. Ueda, T. Kato, Alternator model for full vehicle simulation, *Proc. JSAE annual conference 2013 Spring*, No.407-20135490, 2013 (in Japanese).

Managing Heterogeneous Simulations Using Architecture-Driven Design

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

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

Abstract

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

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

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

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

1 Introduction

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

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

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

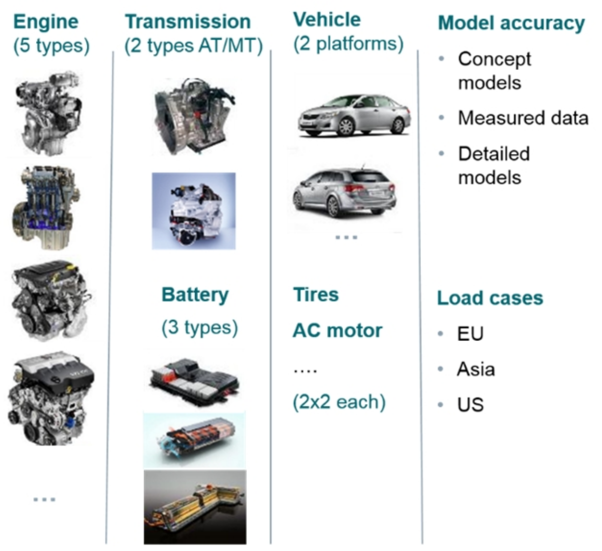


Figure 1. The challenge of dealing with many variants

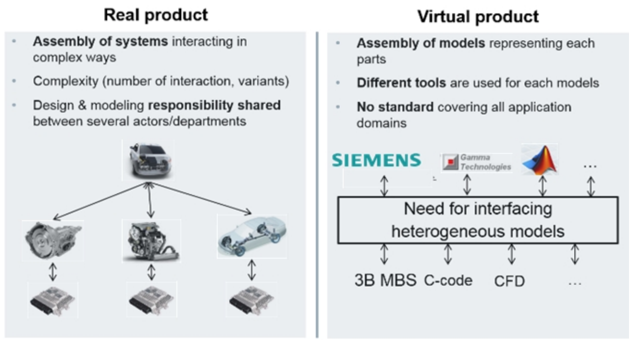

Real product
Assembly of systems interacting in complex ways
Complexity (number of interaction, variants)
Design & modeling responsibility shared between several actors/departments
Virtual product
Assembly of models representing each parts
Different tools are used for each models
No standard covering all application domains
SIEMENS
Gamma Technologies
...
Need for interfacing heterogeneous models
3B MBS
C-code
CFD
...

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

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

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

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

2 Use case description

2.1 Base line behavioral model

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

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

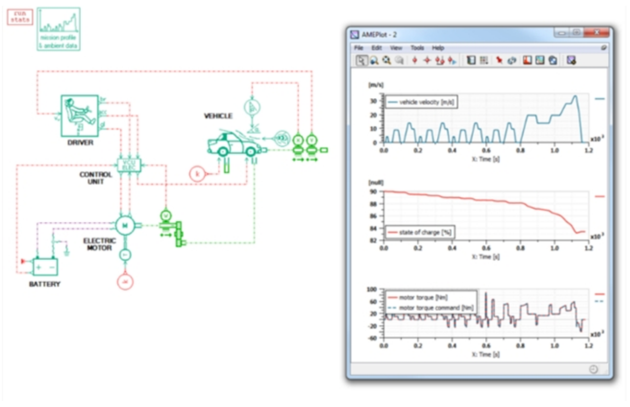


Figure 3. Baseline model for the electric vehicle use case in Simcenter Amesim

2.2 Dymola subsystem models

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

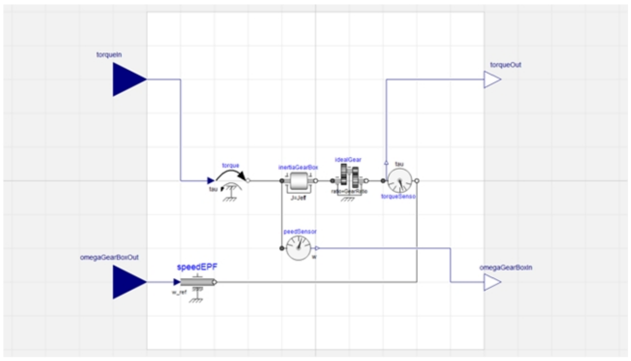


Figure 4. Transmission subsystem implemented in Dymola

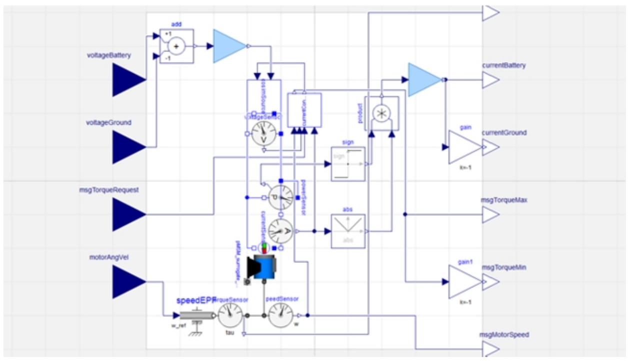


Figure 5. Electric motor subsystem implemented in Dymola

2.3 Integration of Dymola subsystem models through FMI standard

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

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

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

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

3 Architecture-driven simulation framework

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

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

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

3.1 Architecture and template definition

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

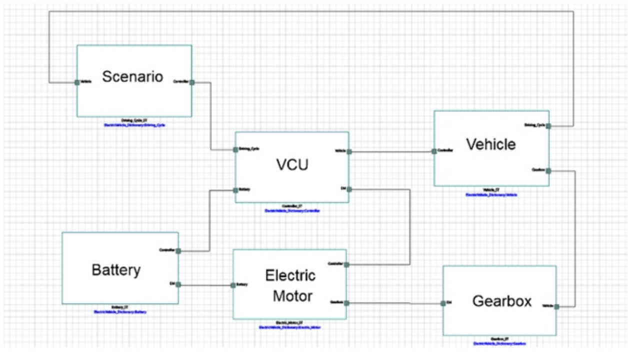


Figure 6. Architecture definition

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

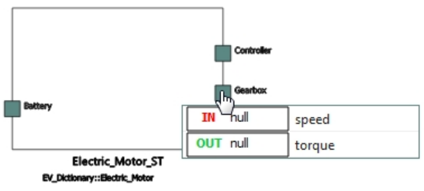
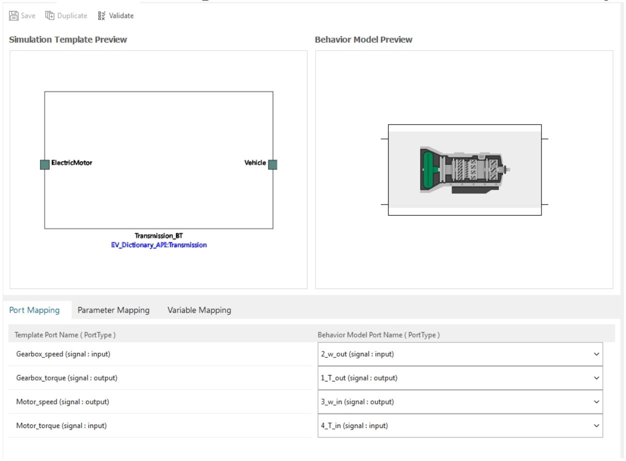


Figure 7. Electric motor simulation template

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

3.2 Model instrumentation

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



Template Port Name (PortType)	Behavior Model Port Name (PortType)
Gearbox_speed (signal : input)	2_w_out (signal : input)
Gearbox_torque (signal : output)	1_T_out (signal : output)
Motor_speed (signal : output)	3_w_in (signal : output)
Motor_torque (signal : input)	4_T_in (signal : input)

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

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

3.3 Model assembly creation

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

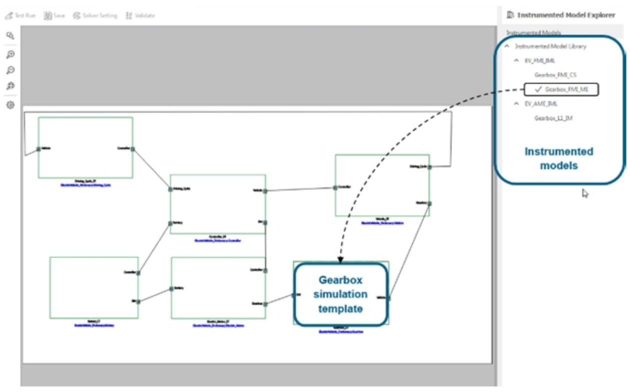

Instrumented Model Explorer
Instrumented Model Library
EV_FMI_BML
Gearbox_FMI_CS
Gearbox_FMI_ME
EV_AME_BML
Gearbox_L1_IM
Instrumented models
Gearbox simulation template

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

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

3.4 Simulation execution

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

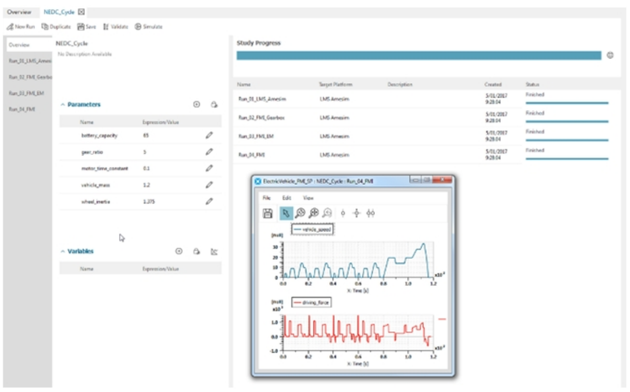


Figure 10. Heterogeneous simulation execution

4 Heterogeneous model assembly with gearbox FMU

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

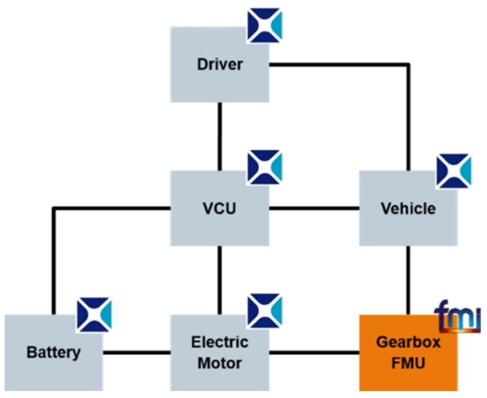
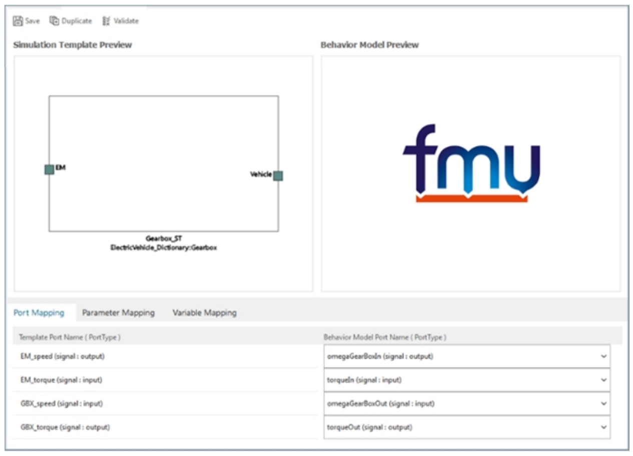


Figure 11. Heterogeneous model assembly after integration of gearbox FMU

4.1 Model instrumentation

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

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



Template Port Name (PortType)	Behavior Model Port Name (PortType)
EM_speed (signal : output)	omegaGearBoxIn (signal : output)
EM_torque (signal : input)	torqueIn (signal : input)
GBX_speed (signal : input)	omegaGearBoxOut (signal : input)
GBX_torque (signal : output)	torqueOut (signal : output)

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

4.2 Simulation results

4.2.1 Accuracy

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

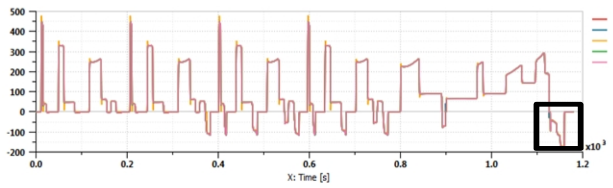


Figure 13. Gearbox output torque for full NEDC Cycle

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

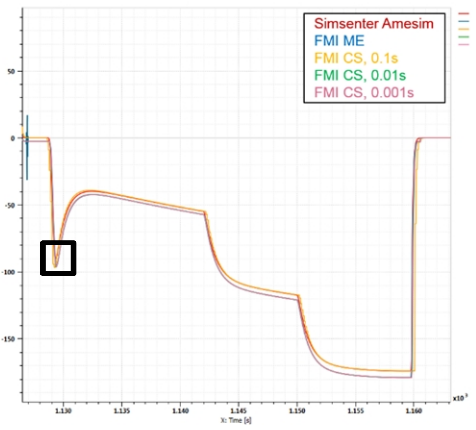


Figure 14. Gearbox output torque (zoomed in to the black region in the right left corner of in Figure 13)

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

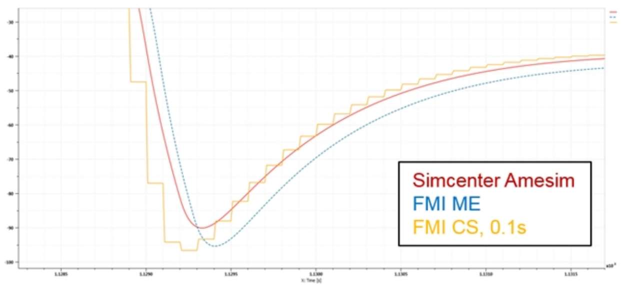


Figure 15. Co-simulation discretizes the system and introduces a delay (detail of the black area on the left in Figure 14)

4.2.2 Performance

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

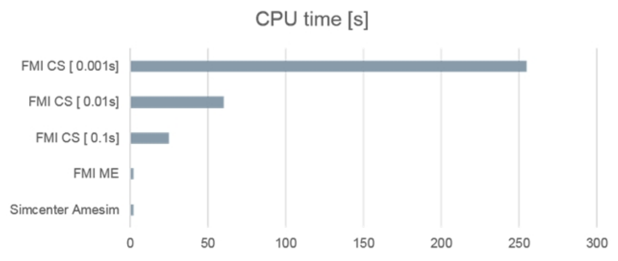


Figure 16. CPU time Performance

4.2.3 Discussion

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

5 Heterogeneous model assembly with electric motor FMU

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

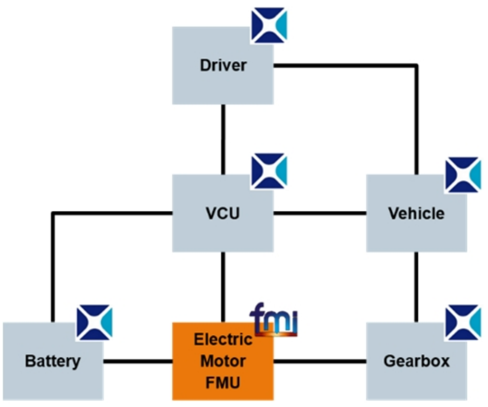
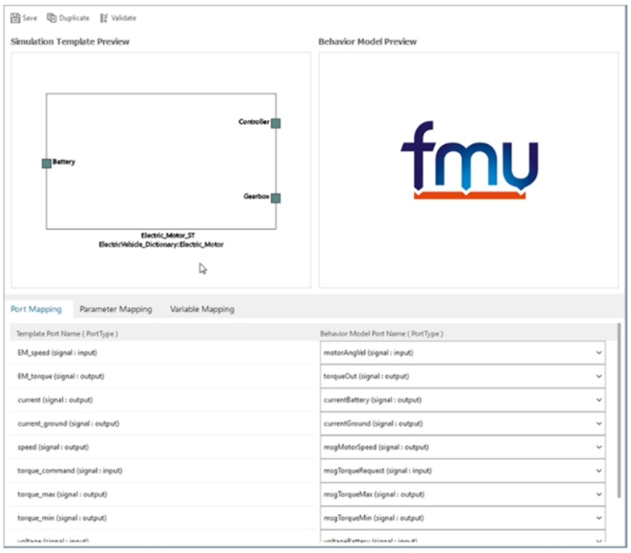


Figure 17. Heterogeneous model assembly after integration of electric motor FMU

5.1 Model instrumentation

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



Template Port Name (PortType)	Behavior Model Port Name (PortType)
EM_speed (signal : input)	motorAngVel (signal : input)
EM_torque (signal : output)	torqueOut (signal : output)
current (signal : output)	currentBattery (signal : output)
current_ground (signal : output)	currentGround (signal : output)
speed (signal : output)	msgMotorSpeed (signal : output)
torque_command (signal : input)	msgTorqueRequest (signal : input)
torque_max (signal : output)	msgTorqueMax (signal : output)
torque_min (signal : output)	msgTorqueMin (signal : output)

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

5.2 Simulation results

5.2.1 Accuracy

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

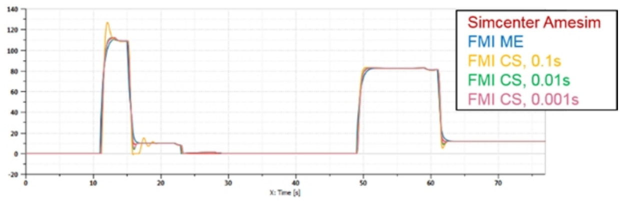


Figure 19. Detail of electric motor torque in a transient region

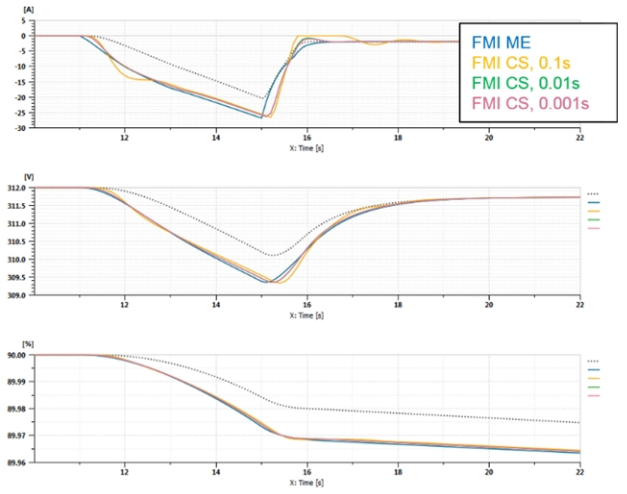


Figure 20. Detail of electric behavior in a transient region: a. current [A]. b. Voltage [V] and c. State of charge [%]

5.2.2 Performance

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

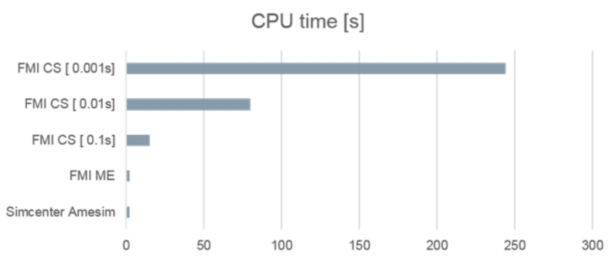


Figure 21. CPU time Performance

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

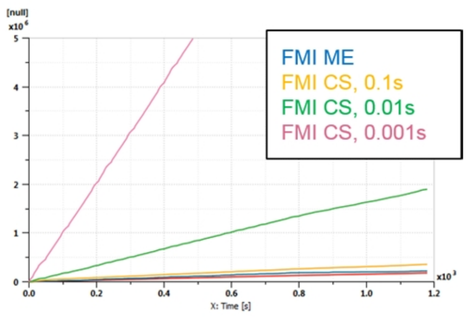
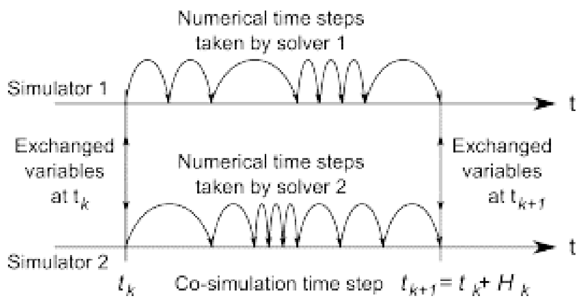


Figure 22. Number of function evaluations over time

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



Numerical time steps taken by solver 1
Simulator 1
t
Exchanged variables at t_k
Numerical time steps taken by solver 2
Exchanged variables at t_{k+1}
Simulator 2
t
t_k
Co-simulation time step $t_{k+1} = t_k + H_k$

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

5.2.3 Discussion

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

6 Heterogeneous model assembly with gearbox and electric motor FMU

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

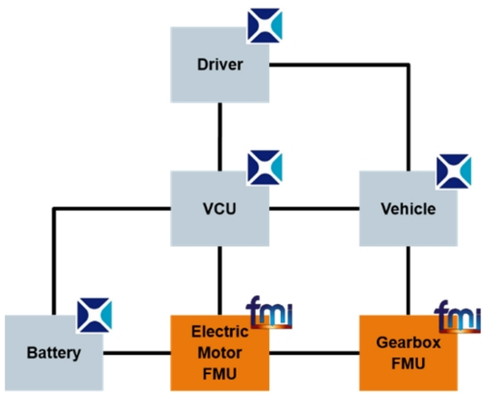


Figure 24. Heterogeneous model assembly after integration of both the gearbox and electric motor FMUs

6.1 Simulation results

6.1.1 Accuracy

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

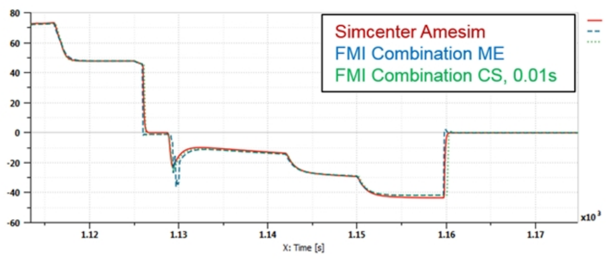


Figure 25. Detail of electric motor torque in a transient region

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

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

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

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

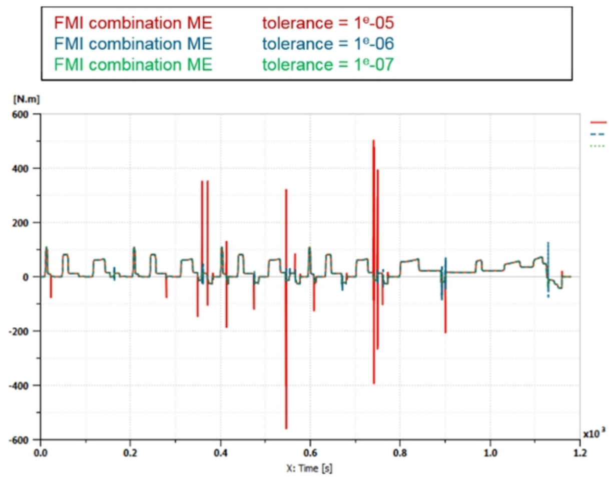


Figure 26. Effect of solver settings (tolerance) on the simulation results

6.1.2 Performance

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

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

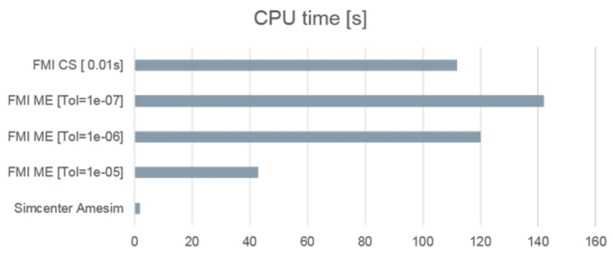


Figure 27. CPU time Performance

6.1.3 Discussion

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

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

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

7 Conclusion

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

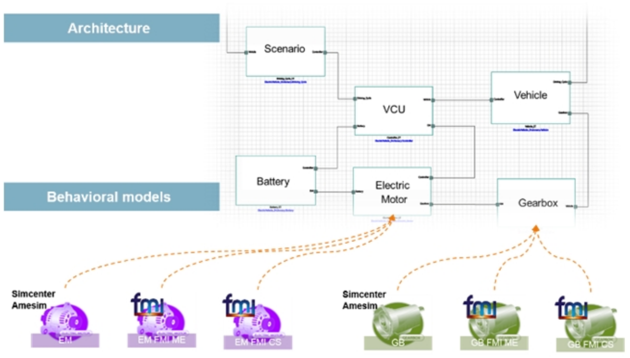
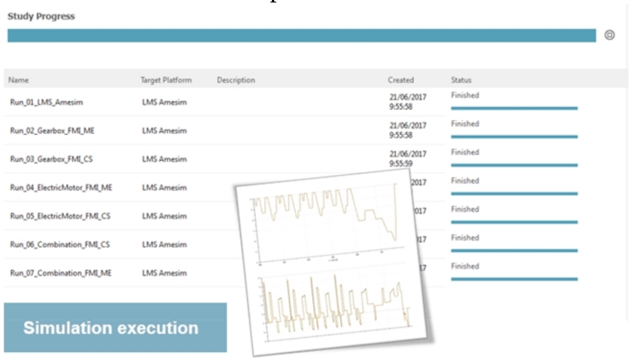


Figure 28. Simcenter System Synthesis as a framework for managing heterogeneous simulations

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


Study Progress

Name	Target Platform	Description	Created	Status
Run_01_LMS_Amesim	LMS Amesim		21/06/2017 9:55:58	Finished
Run_02_Gearbox_FMI_ME	LMS Amesim		21/06/2017 9:55:58	Finished
Run_03_Gearbox_FMI_CS	LMS Amesim		21/06/2017 9:55:59	Finished
Run_04_ElectricMotor_FMI_ME	LMS Amesim		2017	Finished
Run_05_ElectricMotor_FMI_CS	LMS Amesim		017	Finished
Run_06_Combination_FMI_CS	LMS Amesim		17	Finished
Run_07_Combination_FMI_ME	LMS Amesim		7	Finished

Figure 29. Simulation execution from Simcenter System Synthesis

References

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

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

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

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

Simulation of high-index DAEs and ODEs with constraints in FMI

Masoud Najafi

Altair Engineering, France, masoud@altair.com

Abstract

In the current FMI standard the dynamical behavior of a model can only be defined as a system of Ordinary Differential Equations (ODE). The dynamics of many physical systems, such as the equations of motion of constrained mechanical multibody systems, are expressed by high-index Differential Algebraic Equations (DAE) so they cannot be simulated directly using standard ODE or DAE solvers. These systems can be converted through index-reduction into ODE or index 1 DAE systems. However FMUs based solely on these latter systems suffer from drift in hidden constraints on the states. As a consequence, the simulation may results in physically meaningless solutions. In this paper, we propose an extension of the FMI standard to handle DAE Systems of index 1 or higher and ODE with constraints. This FMI extension requires only few additions to the FMI specification, all of which can be omitted for FMUs that represent ODE systems or FMUs that do not support DAE handling. The extension has been implemented in solidThinking Activate$^{\text{TM}}$ and two examples that illustrate the ease of implementation and the effectiveness of the method will be discussed.

Keywords: Modelica, FMI, High index DAE, ODE with constraints, Coordinate Projection

1 Introduction

Activate is primarily a signal-based modeling and simulation environment, but it supports also the Modelica language. Modelica components can be mixed with standard signal blocks in a same diagram. Activate formalism proposes a unique harmonious environment in which signal-based Activate blocks and Modelica components can co-exist in a same model.

In order to simulate an Activate model, the model should be compiled. Compiling a model consists of producing a structure to be used by the simulator. This structure contains all the information needed by the simulator that can be computed before the start of the simulation. It contains in particular all signal types and sizes information, in addition to scheduling tables specifying the condition and the order in which the computational functions of the blocks are to be called during simulation.

The way Activate compiler handles the Modelica components is by grouping them into a single Modelica model with inputs and outputs that are clearly specified by special interfacing blocks. This Modelica model is then compiled by the Modelica compiler (the MapleSim$^{\text{TM}}$ Modelica compiler is used in Activate), which in turn generates an FMU for ModelExchange to replace the Modelica part. The FMI has been chosen as the exchange format because it is a standard already supported both by Activate and MapleSim. The ModelExchange implementation is used because it allows taking advantage of different numerical solvers available in Activate.

A simple example is provided in Figure 1. This model contains an electrical circuit, modeled for the most part using Modelica components. The regular Activate blocks are the Sine Wave Generator and the Scope. There are three interfacing blocks (green blocks) connecting the Activate environment to the Modelica environment.

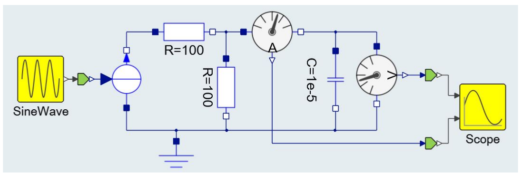


Figure 1. Simple Activate diagram containing Modelica components.

The Modelica part is aggregated into a single block as shown in Figure 2. This step is of course transparent to the user and is presented here as an illustration of the way the mechanism operates. The newly created block has one input and two outputs, as expected.

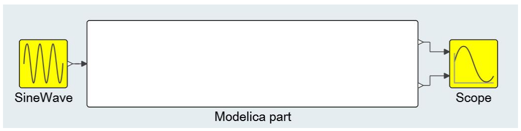


Figure 2. Equivalent Activate model after aggregation of Modelica components.

The Modelica code corresponding to the Modelica part is generated automatically by Activate and sent to the Modelica compiler for compilation. The Modelica compiler then generates a corresponding FMU, which replaces the Modelica part as shown in Figure 3. This step is of course again transparent to the user and is presented here as an illustration. Interested readers are referred to (Nikoukhah, 2017) for more details.

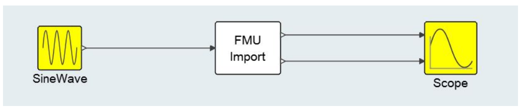


Figure 3. Resulting regular Activate model with Modelica parts replaced by an FMU block.

The compiling process requires that any Modelica model can be converted into FMU. The current FMI standard allows this conversion for many situations but in some cases an extension of this standard would be useful.

Compiling complex Modelica models, in particular mechanical models, very often results in high index DAEs or sometime ODEs and DAEs with constraints. Keeping the constraints and making sure they are satisfied is important to avoid drift in the solution. In the current FMI specification, only ODEs are supported. Activate currently supports, ODEs, index 1 DAEs, and ODEs with constraints. But these solvers cannot be used for the Modelica extension since the FMI does not support DAEs and ODEs with constraints.[1]

Consider the overdetermined system:

$$\dot{x} = f(x) \quad x(t_0) = x_0 \tag{1a}$$
$$0 = \phi(x), \tag{1b}$$

The constraint (1b) is supposed to be consistent with the ODE (1a) in the sense that the solution of this ODE satisfies (1b). So theoretically, Constraint (1b) is redundant. However for numerical simulation, it provides valuable information that can be used by the solver to reduce numerical errors. This can be done by keeping $\phi(x)$ close to zero.

Such constraints may be available in different scenarios. For example in a conservative physical system, where the total energy is conserved, the conservation of energy may be expressed as such a constraint. But the scenario that is of particular interest here is when the ODE (1a) is obtained by differentiating algebraic equations such as (1b). This is done when the original system is a DAE. The algebraic equations are differentiated until an ODE is obtained so that an ODE solver can be used for simulation. In such cases ignoring the original algebraic constraints results often in unacceptable drift in the numerical solution of the system.

[1] The DAE support is currently being considered in FMI design meetings for next releases of FMI.

In the current FMI 2.0 standard, which supports only ODEs, a way to impose the constraint and avoid drift in the solution is to trigger step-events. At solver steps, *i.e.,* at `fmi2CompletedIntegrationStep` calls, the constraint can be checked, and if the error is found to be larger than some user defined tolerance, a coordinate projection method (will be discussed in 2.3) can be performed bringing back the state of the system on the constraint manifold. Then `fmi2CompletedIntegrationStep` should report a step-event, *i.e.*, `enterEventMode=fmi2True`. The simulator treats the step-event and the FMU reports a change in the value of continuous-time states, *i.e.*, `valuesOfContinuousStatesChanged=fmi2True`. A change in the value of continuous-time states usually requires restarting the numerical solver (specially multi-step solvers) with smaller step-sizes and lower order methods which slows down the simulation. This way of treating constraints works but due to large number of step events, the use of multi-step solvers and in some extends single-step solvers becomes uneconomical.

We have extended the current FMI APIs to provide functionalities to take into account the system constraints. This information can be used by Activate or any importing tool to correct the solution so that the constraints are satisfied.

In the following sections, first high index DAE and the coordinate projection method will be discussed. Then our extension of the FMI standard to avoid drift and obtain fast simulation will be discussed. Finally, two test examples to illustrate the advantages of this FMI extension will be presented.

2 DAE Description

DAE systems arise in many applications such as constrained mechanical systems. One attribute of DAE systems is the differentiation index of the system, which can be defined as the number of differentiations of each equation necessary to convert the system into an ODE system. For the sake of simplicity, in this paper, index is used instead of differentiation index. ODE can be considered as an index 0 DAE. Further information on DAEs, and numerical methods for DAEs, can be found in (Ascher and Petzold, 1988) and (Hairer and Wanner, 1996).

2.1 index 1 DAE case:

As an example, consider the following system

$$\dot{x} = f(x,y) \tag{2a}$$
$$0 = g(x,y), \tag{2b}$$

Differentiation of (2b) once gives:

$$\dot{x}\frac{\partial g}{\partial x}+\dot{y}\frac{\partial g}{\partial y}=0 \tag{3}$$

If $\frac{\partial g}{\partial y}$ is not singular, then (2) with (3) can be used to compute the values of $\dot{x}$ and $\dot{y}$. Hence we now have an ODE system. Equation (2) is an index 1 DAE, because one differentiation yields an ODE. This process (of differentiation to obtain an ODE system from a DAE system) is called index-reduction (Pantelides, 1988).

Index 1 systems can be treated in this way, but this introduces a constraint (*i.e.,* $g(x,y) = 0$) that will not be taken into account when using a standard numerical ODE solver.

An alternative approach for index 1 problems is to treat the original system as-is, using the equation (2a) to solve for $\dot{x}$, and treating y as a purely algebraic variable, to be solved using the equation (2b). Solution of this system requires modifications to standard ODE solvers to accommodate the algebraic variables. Ideally these variables should have some error control measures applied that is similar in effect to the error control on $\dot{y}$ of the index reduced system. Advantages of the direct approach are twofold. No unnecessary state y is introduced. Fewer constraints is always better, both making the error through constraint handling smaller, and in this case removing the need for constraint handling altogether. But this method needs the numerical integrator to be modified to accommodate error control on algebraic variables. There are also standard DAE solvers such as DASSL, IDA[2], or RADAU-IIA[3] that can take the equation (2) as input and solve it over time. In these DAE solvers, consistent initial values of x and y are provided by the user. Some solvers can help the user to initialize the DAE by solving the initialization equation. In this case, the user should indicate which variable are differential and which are algebraic.

There is also an alternative approach which is usually used for FMU export. In this approach the y variable is left as an internal variable, and only the x is exposed. This has the advantage that an FMU constructed in this way can be used directly with an ODE solver. In our extension of the FMI specification for handling of algebraic variables, the algebraic section can be safely ignored, and a pure ODE solver can be used for the FMU.

2.2 Higher index DAE case:

For higher index systems, constraints cannot be avoided, even if only performing index-reduction to make the system index 1 DAE or ODE, so a mechanism for handling constraints is required. As an example, consider the unit length planar pendulum in Cartesian coordinates, which can be expressed by the following equations:

$$\left\{\begin{array}{lcl} \ddot{x} & = & Fx \\ \ddot{y} & = & Fy-g \\ 0 & = & x^2+y^2-1 \end{array}\right. \tag{4}$$

One differentiation of the constraint in x, y gives:

$$0=2x\dot{x}+2y\dot{y}$$

And a second differentiation gives:

$$0=2x\ddot{x}+2y\ddot{y}+2\dot{x}^2+2\dot{y}^2$$

which after replacing $\ddot{x}$ and $\ddot{y}$ from (4) and simplification we get:

$$0=2Fx^2+2Fy^2-2yg+2\dot{x}^2+2\dot{y}^2$$

which gives

$$F=yg-(\dot{x}^2+\dot{y}^2). \tag{5}$$

In order to fully index reduce this model, one further differentiation of (5) would be needed to obtain an equation that can be used to get $\dot{F}$ as a function of other states, so this problem is an index 3 system.

Leaving F in algebraic form, *i.e.*, keeping (5) in the system, instead of its derivative) gives us the following system of equations

$$\left\{\begin{array}{lcl} F & = & yg-(\dot{x}^2+\dot{y}^2) \\ \ddot{x} & = & Fx \\ \ddot{y} & = & Fy-g \end{array}\right. \tag{6}$$

where x, y, $\dot{x}$, $\dot{y}$ are differential states and F is the algebraic variable. The two hidden constraints on the states are:

$$\left\{\begin{array}{lcl} 0 & = & x^2+y^2-1 \\ 0 & = & 2x\dot{x}+2y\dot{y} \end{array}\right. \tag{7}$$

In order to solve such a system with constraints, various approaches are possible:

- Simply treat the ODE and index 1 portion of the system ignoring the hidden constraints. Problem: Over time the solution will drift away from the constraints giving an inaccurate or even non-physical solution for the model.
- Use Baumgarte constraint stabilization (Baumgarte, 1972), by adding correcting terms to the ODEs. Problem: This can only reduce (not eliminate) the drift for the problem. Furthermore, the parameter values for Baumgarte are not known in advance.
- High index DAEs may also be handled with FMI in some special cases. For example, if the FMU is exported from a Modelica model

[2]`https://computation.llnl.gov/projects/sundials/ida`

[3]`https://www.unige.ch/~hairer/software.html`

and the modeler has enough knowledge about the states of the model, by using `stateSelect = StateSelect.always` in an appropriate way, it is possible to transform the Modelica model to ODE and export it as FMU. There are several drawbacks. First, the user needs to have a good knowledge about the model to provide appropriate `stateSelection`. Also, nonlinear algebraic equations might need to be solved inside the FMU. Furthermore, the static selection of states might not valid over the whole simulation run and dynamic state selection may be required. With the dynamic dummy derivative method (Mattsson and Soderlind, 1993), it is possible to transform to an ODE and export the Modelica model as FMU. During simulation, step events might be used to hold integration and switch to a new set of states that is numerically more appropriate (S.E. Mattsson and Elmqvist, 2000).

- Pantelides index reduction and dummy derivatives algorithms (Pantelides, 1988) , (Mattsson and Soderlind, 1993), (S.E. Mattsson and Elmqvist, 2000), usually reduce the DAE index to zero or one. Hence, another solution would be enriching the FMI standard to support directly index 1 DAEs. Then, index reduction methods can be used to reduce the DAE index to one and exporting it to FMI (Otter and Elmqvist, 2017). One of drawbacks of this method is the lack of backward compatibility, *i.e.*, the FMUs exported in this way can no longer be simulated with FMI-2.0 compatible simulators.
- Another solution is simulating the ODE part of the system using an ODE integrator, but project back the solution onto the constraint manifold after each time step. After completion of each integrator step, the required projection is computed, and if its norm is large enough, it is applied to the solution so that the constraints are satisfied. In this method, monitoring the magnitude of the projection and integrate it into the error control mechanism is required. We chose this solution to implement our FMI export which requires adding a few new APIs for handling constraints and projection. This method will be explained in the rest of this paper.

2.3 Coordinate projection

The key idea to reduce or avoid drift is to project the solution points found by the numerical solver of the index 1 DAE or ODE system back on the manifold defined by the original system. Consider the ODE with constraints (1). The coordinate projection method essentially consists of two steps for each integration step.

1. Suppose that x_{n-1} is a point consistent with the original system (1). Using x_{n-1} as the initial value, the ODE numerical solver takes a step applying some numerical integration method on the equation (1a), and gets the point $\tilde{x}_n$ at t_n.
2. The solution point $\tilde{x}_n$, computed by the ODE solver, is then projected orthogonally back onto the manifold (1b) given by constraints, *i.e.,* the projected solution is computed as the solution of (8)

$$\begin{cases} \|x_n - \tilde{x}_n\|_2 & = & \underset{x_n}{\text{minimize}} \\ \phi(x_n) & = & 0 \end{cases} \tag{8}$$

which is a nonlinear constrained least squares problem. The projection gives the orthogonal projection to the manifold to get the next point x_n. The projected value x_n is then used to advance the solution for the next step (Eich-Soellner and Fuhrer, 1998).

In (Shampine, 1986), (Gear, 1986), (Ascher et al., 1994), (Ascher and Petzold, 1992), and (Hairer and Wanner, 1996) the coordinate projection was discussed for one-step methods such as Runge-Kutta methods. In case of BDF-methods or, more generally, multi-step methods, the projection is more complex, since the correction computed by the projection method should enter into the error equation (Eich, 1993).

3 Implementation of ODE with constraint in FMI

Applying the index-reduction algorithm (Pantelides, 1988) to a high index DAE to convert it to an ODE, introduces hidden constraints. In this paper we assume that the index reduction algorithm reduces the index to one or zero (ODE). In case of index 1 DAE, the algebraic variables are treated as local variable in the FMU which are computed as a function of continuous-time states, so they can be ignored. As a result, we will consider only ODE with constraint case, *i.e.*, the equation set (1).

The projection process simply computes the changes required for each state variable so that the current values of the system lie on the constraint manifold. Ideally this should be computed as the minimum (or near minimum) change to accomplish this, as the constraint problem is typically under-determined, so many solutions are possible.

For error-controlled integrators, the change required to move the solution back to the constraint manifold can be integrated into the error control mechanism, so if too large a change is needed, the step can be rejected, and step with a smaller step size can be attempted.

When a high index DAE or an ODE with constraint is exported as FMU, the importer tool needs to know the number of constraints present in the FMU. We have used the attribute `maxNumberOfConstraints` in fmi://ModelDescription.xml element to indicate the maximum number of constraints in a model. The default value is zero to keep the FMU backward compatible.

If `maxNumberOfConstraints` is non zero then the FMU should define one or more of the following API functions (depending on the capability flags defined below). .

-
  ```
  fmi2Status fmi2Constraint(
      fmi2Component c, fmi2Real C[])
  ```

  It computes the residual values for all constraints in the FMU. Argument C is `maxNumberOfConstraints` in length. When the solution is on the manifold of the constraint, the norm of the `C` vector is zero or nearly zero.

-
  ```
  fmi2Status fmi2ConstraintJacobian(
      fmi2Component c, fmi2Real J[])
  ```

  It computes the Jacobian for all residual values for all constraints in the FMU with respect to state variables. The length of the array `J` is `maxNumberOfConstraints x numberOfContinuousStates`, and the Jacobian matrix data is storage in row-major. Note that in many cases the constraint Jacobian can become rank-deficient even at non-event points (e.g. bifurcation points in mechanical systems), so caution must be used in using these functions for projection.

  The use of `fmi2Constraint` and `fmi2ConstraintJacobian` provides the master with complete control over the projection process, so one can implement his own scaling method or apply a different solution technique than least squared. In our implementation in Activate, since only one FMU is being used for the Modelica part, the `fmi2ConstraintJacobian` is sufficient, but in case of multiple FMUs with constraint, directional derivative of the constraint should be used.

  Note that `fmi2ConstraintJacobian` is of no use unless `fmi2Constraint` is also defined. If only `fmi2Constraint` is provided, the Jacobian of the constraints can be computed through numerical differentiation. Please note that `fmi2GetDirectionalDerivative` can also be used for our purpose, but some modifications in the variables that can be used in the argument list of this API, *i.e.*, `vKnown_ref` and `vUnknown_ref` would be required.

-
  ```
  fmi2Status fmi2ProjectionStep(
      fmi2Component c, fmi2Real S[])
  ```

  It provides the current local projection step for the constraint residual minimization problem. Argument `S` is `numberOfContinuousStates` in length. For example, in a very simple case, `S`, can be computed as follows. First the Constraint vector `C` and its Jacobian `D` are updated from the model.

  ```
  fmi2Constraint(c,C)
  fmi2ConstraintJacobian(c,D)
  ```

  Then `S` is computed as the pseudo inverse of the matrix `D`, *i.e.*,

  $$S = (D^T D)^{-1} D^T C$$

  Note that this is just for illustration purposes and here we have not considered variable scaling, or cautious handling for rank deficiency. This function returns `fmi2Error` if it is unable to compute the step (for example, the above simplified algorithm is used and $D^T D$ is singular), otherwise it returns `fmi2OK`.

  Interface `fmi2ProjectionStep` is also useful when the master has to iterate on a system composed of multiple FMUs with constraint. In such as system it may not be possible to apply the projection once, because a projection might affect an output of an FMU or when it depends on another FMU's inputs. So it may require some iterations. Note that this is not an issue when only one FMU has constraints, and there is no feedback mechanism present for the FMU with constraints.

-
  ```
  fmi2Status fmi2Projection(
      fmi2Component c, fmi2Real P[],
      fmi2Real projectionTolerance,
      size_t iterationLimit,
      fmi2Boolean apply)
  ```

  It provides a full projection of the current solution back onto the constraint manifold. The projection is applied to the current state until the states satisfy the constraints to within `projectionTolerance` or until we exceed `iterationLimit`. The length of array `P` is `numberOfContinuousStates`. The option `apply` specifies if projection should be applied to the FMU state. If `apply=fmi2False` then this function will return only the difference for the states. If `apply=fmi2True` then the projection will be performed and the internal state of the FMU will be updated. The updated states can be retrieved with a subsequent call to `fmi2GetContinuousStates`. This function returns `fmi2Discard` if it is unable to project onto the manifold, otherwise it returns `fmi2OK`. Note that this function returns `fmi2Discard` for the failure case, so the numerical ODE solver can go back and try to take a smaller step until get a successful step and projection. In the `fmi2Discard` case, the current state of the FMU is not altered, even if called with `apply=fmi2True`. The following pseudo-code demonstrates a simple way for implementation of `fmi2Projection` using `fmi2ProjectionStep`.

```
c_copy = c
delta = infinity
niter = 0
while delta>Tol and niter<=iterationLimit do
 if fmi2ProjectionStep(c,S) != fmi2OK then
   return fmi2error
 end
 delta = |S|
 Update states 'X' in c: X = X+S
 niter++
end while
P = ('X' in c) - ('X' in c_copy)
if not apply then
  c = c_copy
end
if niter>iterationLimit then
  return fmi2Error
end
```

Interface `fmi2Projection` is provided for ease-of-use, but for a system with multiple DAE FMUs it may be necessary to iterate this function at each time step.

- `fmi2Status fmi2GetNumberOfConstraints(`
  `    fmi2Component c, size_t *N)`

  The number of constraints in a model may change during the simulation when the model configuration changes due to a discrete event. We call this kind of systems variable constraint systems. Since in a variable constraint system, the number of (active) constraints can change, we have used this API function to query the number of constraints that are currently active. So in case of variable constraint systems, the use of this function together with `fmi2Constraint` and `fmi2ConstraintJacobian` is necessary. If this function returns zero, the model does not require coordinate projection. This may happen during the simulation of a variable constraint system.

  This function needs only be defined when `maxNumberOfConstraints` is non zero. If the current number of constraints is less than `maxNumberOfConstraints`, say `Ncon`, then only the first `Ncon` entries of constraints and rows of the Jacobian matrix will be populated.

The following three capability flags are being used for Model Exchange FMUs to indicate to the importing tool which of the API functions are supported within the FMU:

- `providesProjection` (Boolean): If `true` the FMU computes projection via fmi2Projection interface. The default value is `false`.
- `providesProjectionStep` (Boolean): If `true` the FMU provides projection step vector via `fmi2ProjectionStep` interface. The default value is `false`.
- `providesConstraints` (Enumeration with `true`, `false`, and `withJacobian`): If `true` the FMU can compute the constraint residual via fmi2Constraint interface. If set to `withJacobian` then additionally the FMU can compute the constraint Jacobian via the fmi2ConstraintJacobian interface.

3.1 Multiple FMU model

The extension to FMI should support the case where several FMUs containing constraints are interconnected such as in a *System Structure Parameterization (SSP)*[4] module. In multiple FMU case, the internal constraints of FMU may depend on FMU inputs which are outputs of other FMUs with constraints. For multiple FMU case, `fmi2Constraint` and `fmi2ProjectionStep` are quite useful. In order to integrate such systems with multiple FMUs, after a complete input/output update of FMUs at a given time and given continuous-time state value, `fmi2Constraint` or `fmi2ProjectionStep` functions can be called by the solver to check and compute the required projection.

3.2 Backward compatibility

This extension in the FMI APIs does not introduce any backward compatibility issue in the FMI standard. Any importing tools will simply have to make sure they can ignore the new capability flags. The exporting tools do not have to generate the new features as the capability flags are `false` by default. So if the importing tool cannot take advantage of the new APIs such as constraints and projections, they are ignored and only ODE is integrated with the price of possible drifts in states.

3.3 Required numerical solvers

In order to take advantages of this extension, the importing tool should include solvers that can support overdetermined systems, in particular, ODEs coupled with algebraic constraints. An example of such solver is CPODES[5]. CPODES is a numerical integrator for solving ODE problems using coordinate projection. It is based on the CVODES integrator which is part of the DOE Sundials[6] suite. CPODES is a multi-step integrator providing variable order Adams (up to 12th order)

[4]https://modelica.github.io/ssp-standard.org/
[5]https://simtk.org/projects/cpodes
[6]https://computation.llnl.gov/projects/sundials

and BDF (up to 5th order) methods for non-stiff problems and BDF (up to 5th order) for stiff problems. It uses CVODES to advance the ODE (2a), and then performs coordinate projection back to the constraint manifold (2b) to exactly solve the DAE (2). The projection is also incorporated back into the error test where it permits larger steps.

Other single-step solvers can also be modified to support the coordinate projection method. In Activate we have modified the RADAU-IIA solver[7] to apply projection computed by `fmi2Projection`.

4 Test cases

4.1 Pendulum model

The pendulum model explained in section 2.2 can be defined with the following Modelica model.

```
model Pendulum_DAE
   constant Real g=9.81;
   constant Real L=1;
   constant Real Ls=L*L;
   Real x(start=L, fixed=true), y(start=0), Lambda ;
   Real vx(start=0, fixed=true), vy(start=0);
   Real drift, totalEnergy;
equation
   der(x)=vx;
   der(y)=vy;
   der(vx)=Lambda*x;
   der(vy)=Lambda*y-g;
   x*x+y*y=Ls;
   drift=x*x+y*y-Ls;
   totalEnergy=(vx*vx+vy*vy)/2+g*y;
end Pendulum_DAE;
```

In this Modelica model, `drift` and `totalEnergy` are output variables defining the drift in the pendulum length and the total energy of the system, respectively. A standard FMU can be generated for this Modelica model. If the two constraints (7) are needed to be satisfied, the FMU can be augmented with the following new APIs.

```
fmi2Status fmi2GetDerivatives(fmi2Component c,
    fmi2Real derivatives[],  size_t nx) {
    double F, g=9.81;
    ModelInstance* comp = (ModelInstance *)c;

    x =comp->state[0];
    y =comp->state[1];
    xd=comp->state[2];
    yd=comp->state[3];

    F = y*g-(xd^2+yd^2);
    derivatives[0]=xd;
    derivatives[1]=yd;
    derivatives[2]=F*x;
    derivatives[3]=F*y-g;
    return fmi2OK;
}

fmi2Status fmi2Constraint( fmi2Component c,
    fmi2Real C[]) {
    ModelInstance* comp = (ModelInstance *)c;
    x =comp->state[0];
    y =comp->state[1];
    xd=comp->state[2];
    yd=comp->state[3];

    C[0]=x*x + y*y- 1;
    C[1]=x*xd+ y*yd;

    return fmi2OK;
}

fmi2Status fmi2Projection(fmi2Component
c, fmi2Real P[], fmi2Real projectionTolerance,
   size_t iterationLimit, fmi2Boolean apply) {
      ModelInstance* comp = (ModelInstance *)c;
      double R;
      x =comp->state[0];
      y =comp->state[1];
      xd=comp->state[2];
      yd=comp->state[3];

      R = sqrt(x*x+y*y);
      P[0] = x/R - x;
      P[1] = y/R - y;
      P[2] = ( xd*y*y- yd*x*y)/R/R - xd;
      P[3] = (-xd*x*y+ yd*x*x)/R/R - yd;

      If (apply){
         comp->state[0] = x/R;
         comp->state[1] = y/R;
         comp->state[2] = ( xd*y*y- yd*x*y)/R/R;
         comp->state[3] = (-xd*x*y+ yd*x*x)/R/R;
      }
      return fmi2OK;
}
```

[7]https://www.unige.ch/~hairer/software.html

In the first test, the pendulum model is exported as a standard FMU, i.e., exported as a pure ODE without constraints. The RadauII-A solver with error tolerance=1e-4 is used. In Figure 4, the left plot displays the `x` and `y` variables and the right plot is the `drift` variable. `drift` is growing as time advances.

Then the same model is exported as an FMU with additional constraints and projection APIs. As shown in Figure 5, the drift in the solution is kept below the requested error tolerance. Whenever the drift exceeds the error tolerance, the projection is applied and the drift becomes zero.

In these tests, only constraints (7) have been considered. We need also to consider the energy conservation law, *i.e.,* the amount of total energy of the system should not change. The energy constraint has not been considered in the Modelica model. So a drift in the total energy due to numerical errors may happen. In order to check the total energy the simulation time is increased to T=300 seconds. A drift in the energy with the value of 9e-3 is obtained. If we add the additional energy constraint to the FMU, *i.e.,* adding the following constraint to the FMU, we can keep the energy constraint valid.

`C[2]= (xd*xd + yd*yd)/2+g*y`

The result is given in Figure 6 where the left-hand plot is the total energy of the pendulum with there is no energy constraint in the model and and the right side plot is the amount of drift in the energy when the above ad-

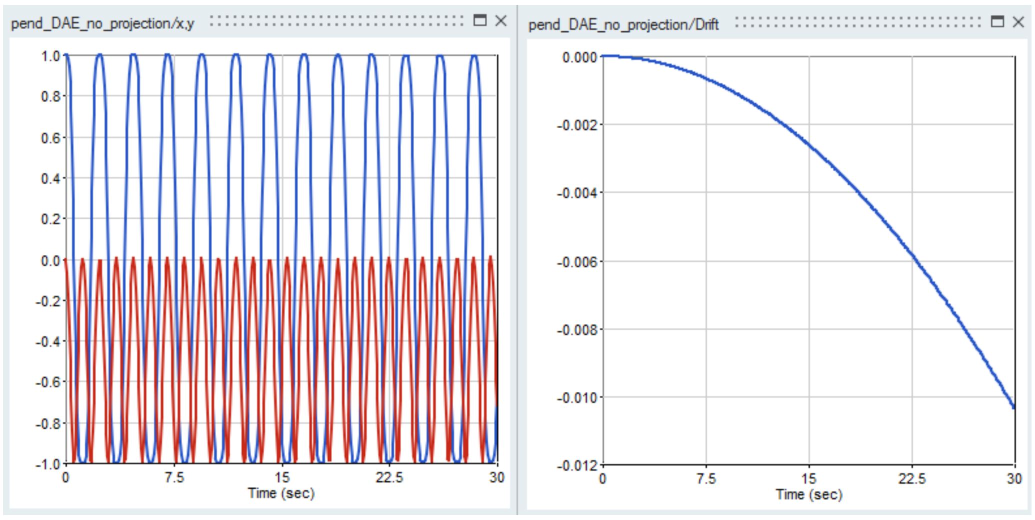


Figure 4. Simulation result of the Pendulum without constraints.

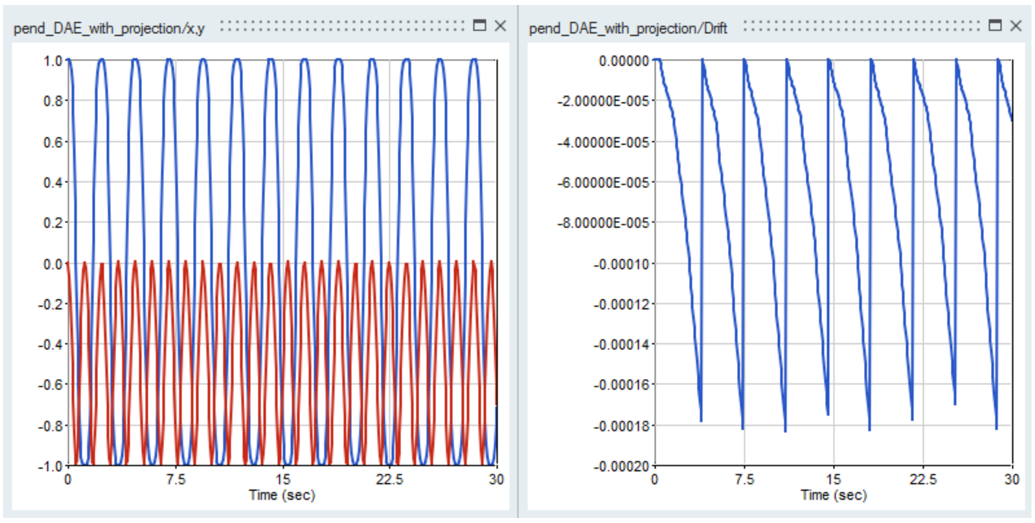


Figure 5. Simulation result of the Pendulum with constraints.

ditional constraint is applied. For this experiment, the modified RADAU-IIA solver with ATOL=RTOL= 1e-5 and maximum step-size=0.1 has been used.

Although the drift in the constraints (middle figure) is kept below the requested tolerance, the amplitude of the `y` variable as well as the total energy of the system is decreased. This is due to lack of the energy constraint in the original model.

4.2 Li-Ion Battery model

The second model (taken from the MapleSoft's Battery Library) represents an electric vehicle, powered by a battery stack consisting of 99 Li-ion cells wired in series. The model features battery temperature changes while the vehicle is controlled to follow an EPA highway drive cycle, defined in a lookup table. *LiFePO4* is used as the cathode material to provide good thermal stability. The model, as shown in Figure 7 is developed in Activate using Modelica components. This example was one of the motivation to handle the constraints efficiently.

This model contains a constraint that should be monitored to keep it near zero during the simulation. In order to ensure that the constraint stays near zero, in the standard FMI, on every `fmi2CompletedIntegrationStep` call, the constraint is checked, if exceeds the error tolerance, the numerical solver is restarted which, as explained in section 1, slows down the simulation. After the development of the new extension of FMI in in the MapleSim Modelica compiler and in Activate, we were able to simulate this model in a few seconds compared to hours. The simulation result is given in Figure 8.

5 Conclusion

In order to simulate Modelica components in Activate, they are regrouped and exported into an FMU. Since, in the current FMI standard only ODE is supported, the

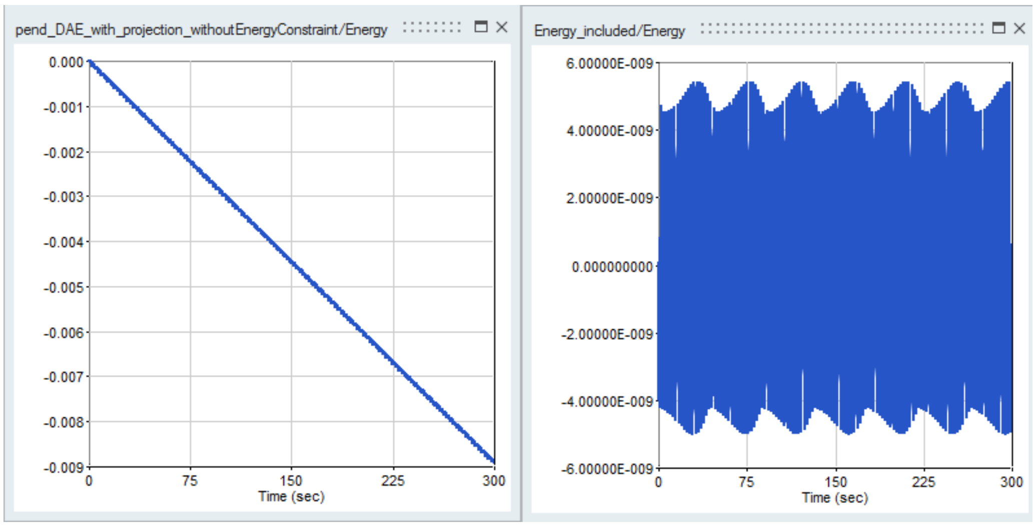


Figure 6. Drift in the total energy of the pendulum: with considering the energy constraint (right) and without considering the energy constraint (left).

Modelica models, even high index ones, are converted into ODE to be exportable into FMU. Due to conversion of high index DAEs to ODE, some constraints in the DAE may be ignored and that may cause the solution drift off the constraints. In this paper we have presented the way the FMI standard can be extended in a backward compatible way to to deal with systems whose states should satisfy hidden constraints on its continuous-time states, such as constraints resulting from DAE index reduction of mass or energy conservation. In this extension, after applying the index reduction method to high index DAE, an ODE with constraints is obtained. The resulting ODE is simulated using standard ODE integrators, but the solution is projected back onto the constraints after each time step. In other words, after completion of each integrator step of the ODE numerical solver, the required projection to bring back the solution on the constraints is computed, and if its norm is large enough, it is applied to the solution so that the constraints are satisfied. The new extension of FMI has been implemented in Activate and two examples are illustrated in this paper.

6 Acknowledgement

This work has been done with close collaboration with MapleSim^{TM} development team who provides the Modelica compiler of Activate.

References

U. Ascher and L. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1988.

U. M. Ascher and L. R. Petzold. Projected implicit runge-kutta methods for differential-algebraic equations. *SIAM, Numerical Analysis, 28(4), 1097-1120.*, 1992.

U. M. Ascher, H. Chin, and S. Reich. Stabilization of daes and invariant manifolds. *Numer. Math. 67: 131*, 1994.

J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering Volume 1, Issue 1, Pages 1-16*, 1972.

E. Eich. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM J. on Numerical Analysis 30(5):1467-1482*, 1993.

E. Eich-Soellner and C. Fuhrer. *Numerical Methods in Multibody Dynamics*. European Consortium for Mathematics in Industry, B.G. Teubner, 1998.

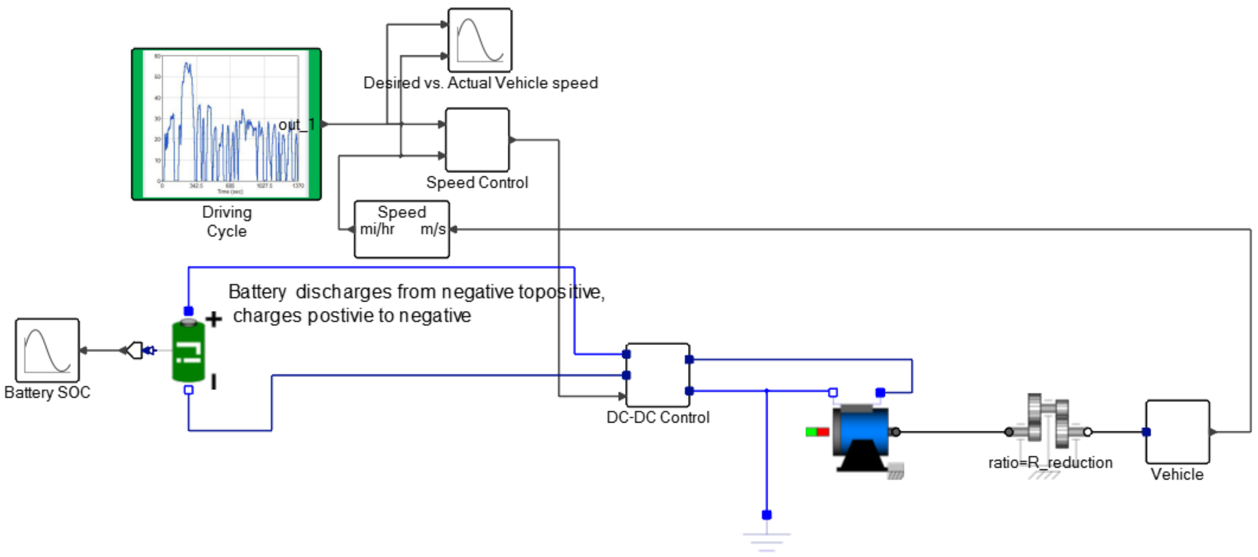
C.W. Gear. Maintaining solution invariants in the numerical solution of odes. *Journal on Scientific and Statistical Computing, Vol. 7, No. 3*, 1986.

E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, 1996.

S.E. Mattsson and G. Soderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal of Scientific Computing. 14(3), pp. 677-692*, 1993.

R. Nikoukhah. A simulation environment for efficiently mixing signal blocks and modelica components. *12'th International Modelica conference*, 2017.

M. Otter and H. Elmqvist. Transformation of differential algebraic array equations to index one form. *Proceedings of the 12th International Modelica Conference, Prag, Czech Republic*, 2017.



Desired vs. Actual Vehicle speed
out_1
Speed Control
Driving Cycle
Speed
mi/hr
m/s
Battery discharges from negative topositive,
charges postivie to negative
+
Battery SOC
DC-DC Control
ratio=R_reduction
Vehicle

Figure 7. Battery Electric Vehicle model

C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988. doi:10.1137/0909014.

H. Olsson S.E. Mattsson and H. Elmqvist. Dynamic selection of states in dymola. *Modelica Workshop 2000, Lund, Sweden, pp. 61-67*, 2000.

L. Shampine. Conservation laws and the numerical solution of odes,. *Comput. Math. Appls, Part B., 12, pp. 1287-1296*, 1986.

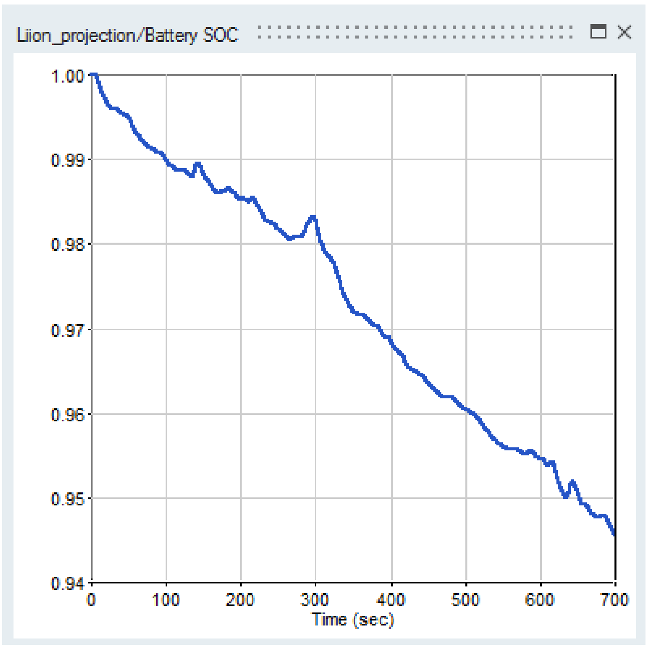


Figure 8. State of charge of the battery

Universal Controllers for Architecture Simulation

Alexander Pollok*[a,b] Francesco Casella†[b]

[a]Institute of System Dynamics and Control, DLR German Aerospace Center, Germany
[b]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

Abstract

For optimization studies of dynamical systems, it is common practice to model and tune local controllers for miscellaneous subsystems. For instance, a model of a chemical plant may contain a valve motor model, and a model of a PID controller may be included to control the motor. The associated controller tuning effort is ultimately wasted. The actual controller will be retuned anyway after finalization of the system design, or will be structurally different.

For this reason, control algorithms are needed that just provide the functionality of the actual control algorithm that will be designed in a later phase of the system design. These temporary algorithms need to have low tuning requirements, and it must be possible for non-control-specialist to generate them. On the other hand, they only need to function inside a simulation environment.

Several mainstream control approaches are reviewed, and boundary layer sliding mode control is proposed as a suitable approach for this kind of task. This class of controllers can be used without any tuning effort, and is able to compete with tuned PID-controllers in terms of tracking performance. An end-user friendly implementation of a universal controller in the equation-based and object-oriented modelling language Modelica is presented. Several examples are shown to demonstrate the performance of the proposed approach.

Keywords: Modelling, Modelica, Sliding Mode, Modelling aids, Optimization, Local Controller

1 Introduction

In industrial projects, typical phases are modelling, architecture optimization, and control systems design. For larger projects, the overlap between the responsible groups of people can be small to nonexistent. Also, not every modelling expert is also a control expert.

For preliminary studies on an architecture level, equation-based object-oriented modelling languages (EOOML) like Modelica are well suited, since individual components can be connected and rearranged quickly and flexibly.

However, additional efforts can arise when the system architecture also includes controlled components, which are also to be sized. An example for this might be pumps that control the mass flow through a pipeline. The actual pump characteristics are parameterized, to be determined during the architecture optimization. During the optimization, each function evaluation corresponds to a simulation of the architecture. For these simulations, the pump needs a controller model that controls the pump in such a way that the target mass flow is reached. This holds, even if the function evaluation is only dependent on the steady state behavior of the system.

A typical workflow for the development of such controllers looks like this: A predefined PID-controller-model from a standard library is included. The controller output is connected to the valve input. Two additional elements are created and connected to the controller model to retrieve or define controller target and actual value. The PID controller is set to P-mode with a k_p of 1, and the system is simulated. The dynamic behavior of the controlled valve is checked, and the controller gain is adjusted for correct order of magnitude and correct sign, if necessary. This can take a few iterations. Subsequently, the controller is set to PI-mode, and a small k_i value is defined to assure zero steady state error. Again, this can take a few iterations until the correct order of magnitude is found. If the dynamical behavior stays insufficient, or if the modelling expert is sufficiently motivated, a few experiments with added derivative action (PID-mode) might follow. Alternatively, optimization tools might be used, shifting the bulk of the effort into the creation of the optimization setup.

This workflow takes some time until the results are acceptable. Also, the resulting controller might not be robust against model changes, making additional effort during the optimization phase necessary. If an architecture contains several controlled components, the necessary effort grows accordingly. This effort is ultimately wasted, since the controllers will be redeveloped by actual control experts anyway, as soon as the architecture is finalized. This is illustrated in Figure 1.

In this paper, preliminary controllers are developed that control a wide range of system models without any

*alexander.pollok@dlr.de
†francesco.casella@polimi.it

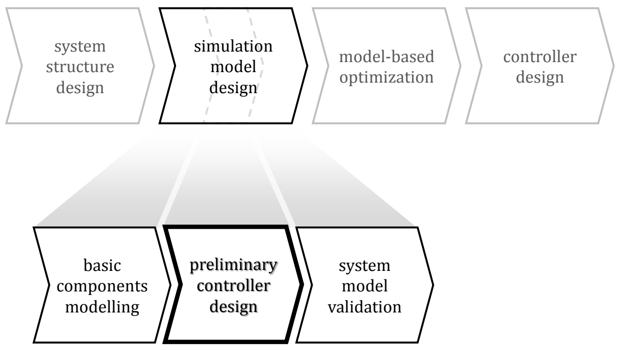


Figure 1. typical workflow for the development of controlled technical systems

tuning effort, or any control systems expertise on the side of the modeller. They are easy to set up, and perform well inside a simulation environment. This is achieved using the class of first-order boundary-layer sliding mode controllers, and implemented in the equation-based modelling language Modelica. The paper is structured as follows: In Section 2, the most important control approaches are reviewed and categorized according to their suitability. The best candidate is identified, then modified to suit the needs of modelling experts in Section 3. Section 4 presents several examples to demonstrate the performance of the resulting control approach. Limits of application of the proposed control approach are discussed in Section 5. The paper is concluded in Section 6.

2 Review of Feedback Control Approaches for SISO systems

2.1 PID

Principle: The controller output is computed as a sum of three components: One component is proportional (P) to the control error, the second component is proportional to the derivative of the control error (D), and the third component is proportional to the time integral (I) of the control error.

Advantages: PID-controllers are the most widely used controllers in the scope of object-oriented modelling. They are easy to understand, offer reasonable performance for most SISO-systems, as well as zero steady state error and simple addition of anti-windup measures. As PID is very commonly used in control algorithms, such a controller would be most representative for the eventual controller implemented with the system.

Disadvantages: PID-controllers require the user to tune 3 parameters, either by hand or using optimization. As soon as a good tuning is found, it might not be suitable for different parameterizations on architecture level. It basically implies control design effort in a stage were just representative functionality is required. The effort will be wasted.

Variants: The concept of PID-controllers has been generalized into Fractional PIDs, as shown in Vinagre et al. (2007). Here, fractional differential operators are used instead of the usual integer operations (integral and derivative action). Fractional differential operators require infinite memory during simulation, but can be approximated in EOOML, see Pollok et al. (2015). However, the number of tuners increases from 3 to 5, therefore the usability for one-size-fits-all-control is even more limited.

2.2 Native Model Inversion

Principle: EOOML are not causal. That means that there is no inherent direction of computation (as opposed to, for instance, a block diagram). Nothing prevents the modeller from defining the nominal output of a system, leaving the computation of the system input to the solver. This can be used for control: The controlled variable is equated to the target value, and the virtual controller output is computed during simulation.

Advantages: The dependency of the controller on the specific system is largely encapsulated in inverse model equations that a simulation environment like Modelica is able to generate automatically. This has enabled automatic control system generation in eg. Aircraft design. No additional effort is necessary on the modellers side. Also, perfect tracking of the controlled variable is possible. The design effort is not wasted, as, once configured appropriately, the control laws can continuously evolve with the system and eventually be implemented.

Disadvantages: Bounds on the controller output cannot be implemented. Also, model inversion usually fails or becomes quite involving, if the system has a high relative degree, gets big, or if there is no unique solution. This affects all but the simplest models. The control architecture must be configured appropriately at the first time, still requiring control expertise.

Variants: Many modern control approaches are based on model inversion. Examples are nonlinear dynamic inversion as used in Thümmel et al. (2005) or incremental nonlinear dynamic inversion as used in Acquatella et al. (2012). However, these approaches lose much of the simplicity of direct inversion, and are therefore not suited for a modelling expert.

2.3 LQR/LQG

Principle: Linear Quadratic Regulators (LQR) represent a static state-feedback-law that optimizes a quadratic cost function (H_2-norm) for LTI-systems. Since they use state-feedback, they require a complete state vector to compute the controller output. Typically, LQRs are combined with a Linear Quadratic Estimator (LQE, or Kalman Filter), to get an estimate of the state vector based on measured variables. The combination is known as Linear Quadratic Gaussian Control (LQG) Kalman et al. (1960).

Advantages: These control laws are more capable in terms of robustness and MIMO systems, thus reducing control design effort in case of systems with strong interaction between different control inputs. For simulation studies, the exact state vector is known to the solver; therefore LQR based control is possible without any need for an estimator. LQR offers guaranteed robustness properties, in contrast to LQG Doyle (1978). The tuning variables of the LQR/LQG approach are design specifications, a skilled control systems engineer can anticipate the effect of the tuning variables on the system behavior.

Disadvantages: Since the computed controller is only optimal for one fixed system, the control design effort is wasted as soon as the system architecture is modified. The computation of the LQR matrix requires a solution of the ricatti-equation, making the use of external tools like Matlab necessary. LQR is based on the assumption that the system is LTI, which is often not the case for complex real world applications. There is no treatment of actuator limitations, as well as no guaranteed zero steady state error. Retrieving the actual state vector in the correct shape might not be that straightforward for the modeller, since environments for EOOML usually encapsulate this information from the user; for this reason alone, the use of LQG in EOOM environments might make sense.

Variants: While the computation of LQR and LQG controllers is straightforward, there are possibilities to extend the method. In Skogestad and Postlethwaite (2007) a variant of an LQG controller is described, where the system model is extended by artificial integral elements. Since the output of these integrators is controlled as well, zero steady state error of the native states can be guaranteed.

2.4 Sliding Mode Control

Principle: In Sliding Mode Control (SMC), a desired subspace (sliding surface) of the system state space is defined in a way that exhibits desirable dynamics. Nonlinear control laws are used to drive the system state onto the sliding surface in finite time. Typically, this takes the form of a bang-bang control-law, where the controller tries to drive the system state into the direction of the sliding surface with maximum authority.

Advantages: SMC is robust against matched uncertainties. As soon as the sliding surface is reached, plant deviations don't lead to deviations in plant dynamics at least for low-order systems. No tuning based on experiments or complex calculations is necessary, only the desired dynamics have to be defined. No control expertise is needed on the side of the modeller. The same controller can be used even if the system is modified later.

Disadvantages: As soon as the sliding surface is reached, chattering occurs. The controller output then changes significantly with each time step. Simulations using implicit solvers and clean event-detection get stuck.

Variants: Several approaches have been described to alleviate the chattering effect: In Filtered SMC as described in Edwards and Spurgeon (1998), the controller output is first-order filtered. In Boundary SMC as described in Utkin et al. (2009), the hard nonlinearity at the sliding surface is replaced by a smooth transition inside a small boundary layer. A number of second Order SMC are described, see for example Bartolini et al. (1998). Here, both the sliding variable (roughly translatable as the distance of the state to the sliding surface) as well as its derivative are driven towards zero in finite time.

2.5 Others

H-infinity synthesis methods have been left out, but their structure, advantages and disadvantages are very similar to LQG-based methods.

Still, the approaches mentioned here don't constitute a complete list. Naturally, it is difficult to review each and every control strategy that has been published. Examples of approaches that have been left out are model predictive control, data driven control system design, adaptive control as described in Åström and Wittenmark (2013) or Intelligent Control techniques like fuzzy control or neural networks Zilouchian and Jamshidi (2000).

However, to the best knowledge of the authors, most of these approaches are either concerned with only a subset of controllable systems, are overly complicated for the task at hand, or have other relevant disadvantages.

3 Universal Controllers for equation-based simulation environments

3.1 Concept

With direct inversion and sliding mode control being the only exceptions, all of the other approaches considered require either manual tuning or external design software. Direct inversion cannot be used for all but the simplest of systems. Second order sliding mode controllers are overly complex, first order sliding mode controllers produce chattering effects, making them unusable for the implicit solvers typically used in EOO modelling environments.

However, modified variants of first order sliding mode control are available to alleviate this problem. Two of them, filtered sliding mode and boundary layer sliding mode were implemented in Modelica and used to regulate the states of a deflected double-integrator back to zero, with first-order decay as sliding surface. For comparison, a conventional first order sliding mode controller was also implemented[1]. The controller output was limited to ± 5. The system behavior for all three controlled system was nearly identical, but the associated controller outputs showed stark differences. While the standard controller exhibited the expected amount of chattering, it was significantly reduced for the filtered sliding mode controller. The Boundary layer variant showed no chattering. These results can be seen in Figure 2. Using implicit solvers, boundary layer sliding mode control simulated roughly 6000 times faster than filtered sliding mode control, probably as a result of generating no state events (compared to 92.942 state events for filtered sliding mode control). That makes the boundary layer variant of first-order sliding mode control a promising candidate for the control of simulated architectures. A commonly cited drawback for boundary layer sliding mode controllers is their sensitivity to noise Young et al. (1996). However, this is not a problem in the context of simulation studies using implicit solvers, where noise is minimal to nonexistent.

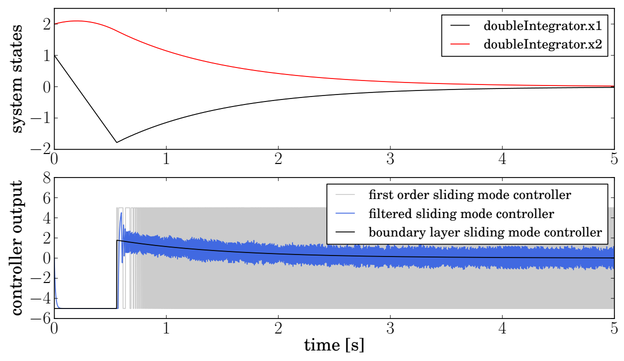


Figure 2. Comparison of different First Order Sliding Mode Controller variants

3.2 Boundary Layer Sliding Mode Control

In sliding mode control, the principle is to force the system state to stay on a defined subspace with beneficial properties. A sliding variable s is defined based on the system output and its derivatives. To enforce first-order decay of a SISO-system, the sliding variable can for instance be defined as

$$s = w_1 \cdot (y - t_{target}) + w_2 \cdot \frac{d}{dt}(y - t_{target}) \tag{1}$$

with w_1 and w_2 as weighting parameters, determining the time constant of the first order behavior. In the simplest case, the control law takes the form of

$$u = u_{max} \cdot sgn(s) \tag{2}$$

The idea of boundary layer sliding mode control is to replace the hard discontinuity at $s = 0$ with a linear approximation:

$$u = \begin{cases} -u_{max} & \text{if } s < -l \\ u_{max}\frac{s}{l} & \text{if } -l < s < l \\ u_{max} & \text{if } l < s \end{cases} \tag{3}$$

with the layer width l. During simulation, this generates state events every time the layer thresholds are crossed, slowing down the simulation. For this reason, a formulation based on the tangens hyperbolicus was used instead:

$$u = u_{max} \cdot tanh\left(\frac{s}{l}\right) \tag{4}$$

This has the advantage of being solver-friendly, since no state events are generated during simulation. Also, the tangens hyperbolicus is C^∞-continuous, which can be exploited by Dassl and other DAE-solvers that use polynomial expansion.

3.3 Implementation

For implementation in Modelica, usability aspects have to be considered as well as technical aspects. Therefore the controller model is designed to ensure applicability for a wide range of problems, while having a simple interface to the end user.

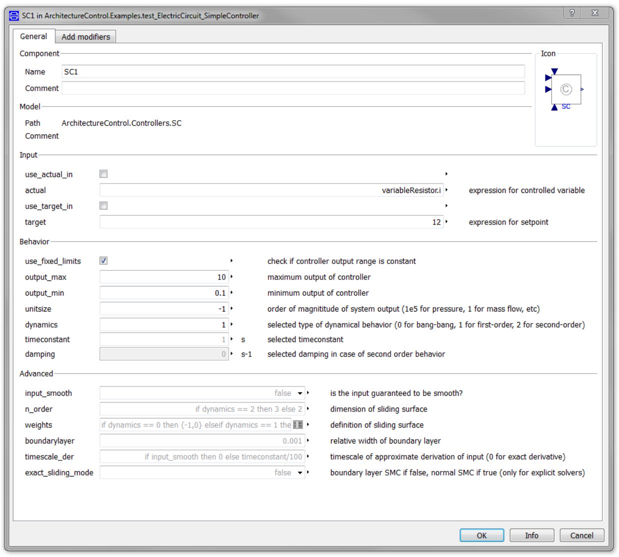
- For input of controlled variable and set point, the user can choose between conditionally defined connectors and input fields, where arbitrary expressions can be included.
- Minimum and maximum controller output can be set as fixed parameters, or can be defined adjustable using conditionally defined connectors.
- Three types of behavior can be selected, corresponding to different structures of the sliding surface: Zero-order dynamics corresponds to perfect matching between set point and actual value whenever possible. For first-order dynamics, a target time constant for control error decay can be entered. For second-order dynamics, both time constant and a damping value are used. In every case, the unused values are greyed out as to not confuse users.

[1]Standard First Order Sliding Mode Controllers cannot be simulated using Dassl Petzold et al. (1982), the standard solver for many Modelica environments. For this reason, an explicit solver (RK4) was used for this comparison.

- The relative width of the boundary layer is dependent on the order of magnitude of the controlled variable. If a pressure variable is controlled, typical values are in the order of 10^5. This makes the boundary layer much smaller in comparison. Therefore the user can also enter a unit size variable to compensate. This variable can also be used to change the sign, if the controller output goes in the wrong direction, thereby making it the only tuning decision which is necessary, albeit a binary one.

- For users that know what they are doing, more options are available. These are grouped in an "Advanced"-tab. Here, the input can be flagged as smooth, so that exact derivatives are used for the calculation of the sliding variable instead of approximate derivatives. Also, the definition of the sliding surface can be overwritten. Finally, the boundary layer formulation can be replaced with a standard sliding mode controller, which might make sense if an explicit solver is used.

The user interface of the controller model is shown in Figure 3. The shortened Modelica code listing can be seen in the Appendix in Section A.



SC1 in ArchitectureControl.Examples.test_ElectricCircuit_SimpleController
General
Add modifiers
Icon
SC
OK
Info
Cancel

Component

Name	SC1
Comment	

Model

Path	ArchitectureControl.Controllers.SC
Comment	

Input

use_actual_in	☐		
actual	variableResistor.i		expression for controlled variable
use_target_in	☐		
target	12		expression for setpoint

Behavior

use_fixed_limits	✓		check if controller output range is constant
output_max	10		maximum output of controller
output_min	0.1		minimum output of controller
unitsize	-1		order of magnititude of system output (1e5 for pressure, 1 for mass flow, etc)
dynamics	1		selected type of dynamical behavior (0 for bang-bang, 1 for first-order, 2 for second-order)
timeconstant	1	s	selected timeconstant
damping	0	s-1	selected damping in case of second order behavior

Advanced

input_smooth	false	is the input guaranteed to be smooth?
n_order	if dynamics == 2 then 3 else 2	dimension of sliding surface
weights	if dynamics == 0 then {-1,0} elseif dynamics == 1 the	definition of sliding surface
boundarylayer	0.001	relative width of boundary layer
timescale_der	if input_smooth then 0 else timeconstant/100	timescale of approximate derivation of input (0 for exact derivative)
exact_sliding_mode	false	boundary layer SMC if false, normal SMC if true (only for explicit solvers)

Figure 3. User Interface of SimpleController parameters

4 Examples

4.1 Low-Order-Systems

As a first test, the controller is used to control three different low-order-systems. The first system is a static unity-gain, the next system is a first-order system with a time constant of three seconds, the last system is a second-order system with an angular frequency of 0.5 rad/s and zero damping.

All controllers use the standard values: First-order target behavior with a time constant of one second. Output limits are set to ± 5. All systems were initialized at state vectors of zero, the target output was defined as a step at $t = 1s$. The results are shown in Figure 4.

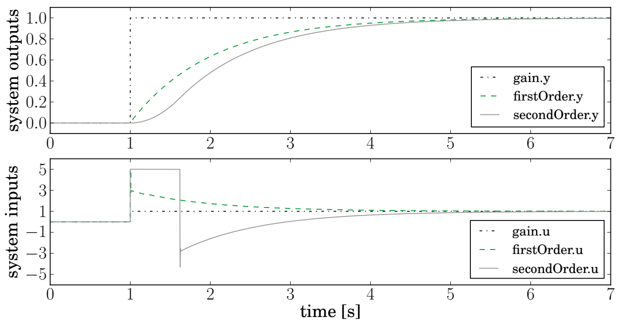


Figure 4. Dynamic behavior of several low order systems controlled by boundary layer sliding mode controller

The first-order system is forced on the reference behavior almost exactly. A small overshoot in the system input/controller output is visible, caused by the time constant of the approximate derivative used for the controller formulation.

The behavior of the second-order system is significantly limited by the controller limits. At 1.62 seconds, this limitation is no longer relevant, and the system reverts to first-order behavior. Again, a small overshoot occurs at the controller output, caused by the approximate derivative.

Interestingly, the static gain doesn't show any dynamic behavior; instead the system output perfectly matches the target. This can be explained by the feedthrough behavior of the controller, which instantly compensates the set point change. Since no control error is being generated, no control error has to fade away with a first-order behavior. In contrast to this, a control error was being generated for the first- and second-order systems, when the controllers couldn't track the set point perfectly due to controller output limitations.

4.2 Thermal Network

To estimate controller performance on systems with a high relative degree, a model of a thermal network was created. The network consists of 9 thermal masses, connected in a two-dimensional grid. At one corner, a fixed temperature boundary condition of -10 egrees C is applied, at the opposite corner, a temperature is measured. This measured temperature is also the control variable, with a set point of 0 degrees C. At one of the remaining corners, a disturbance heat flow is applied, taking the shape of a saw tooth function. The last corner is reserved for the actuator: a variable heat flow, with a range of 0 to 10 Watts. The lower limit of 0 Watt corresponds to a pure heating element, without any cooling capability. It also introduces a strong nonlinearity to the system. All thermal masses were initialized at 20 degrees C. The system can be seen in Figure 5.

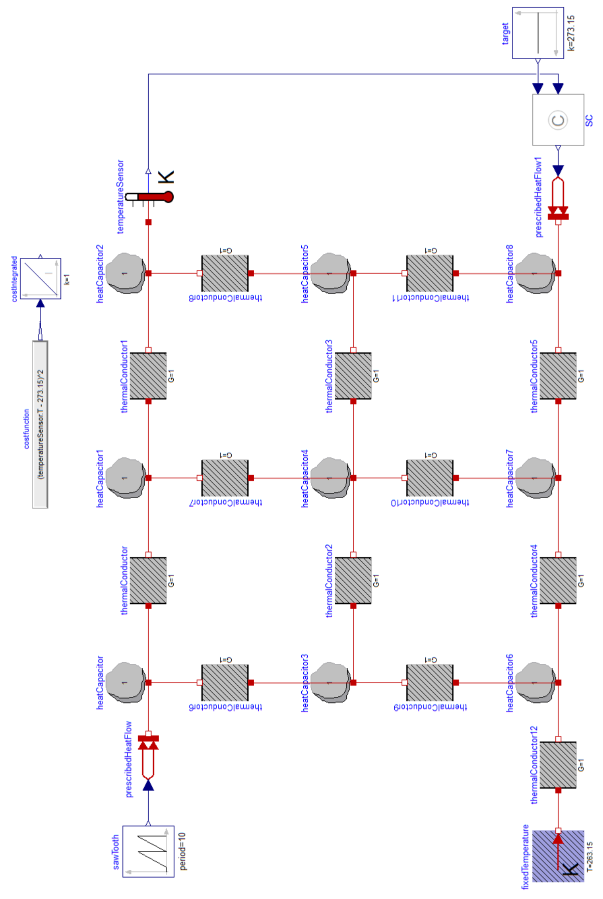


Figure 5. thermal network test model

The system was instantiated two times, using different controllers. One time, the proposed boundary layer sliding mode controller was used, with a selected second-order behavior, a time constant of 1 second and a damping of 1.

For comparison, the system was instantiated with a PID-controller, including anti-windup functionality. The 3 parameters of the PID were optimized using the Modelica Optimization library Pfeiffer (2012). As an optimization goal, the integrated quadratic control error (IQCE) for a simulated time of 100 seconds was used. The optimization achieved an IQCE of 3016.3.

The sliding mode controller achieved an IQCE of 3014.8 without any optimization. The associated temperatures and controller outputs can be seen in Figure 6.

5 Discussion

In the previous Sections it was shown that Boundary Layer Sliding Mode Controllers are a very easy to use tool, suitable for a wide range of simulation tasks. Performance wise, this class of controllers can compete with other control approaches. At the same time, configuration efforts for the modelling expert are minimal.

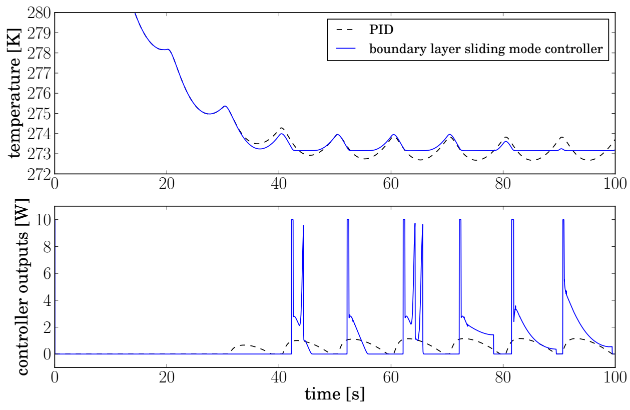


Figure 6. thermal network test results

There are however situations, where the proposed controller is not adequate. For instance, this applies to MIMO-systems with strong coupling. If the water level in n water tanks has to be controlled using 2 valves, and there is no clear one-to-one assignment between the tanks and valves, the proposed controller will probably fail. But as long as the influence of each valve on a single tank is strong, and the influence of the respective valve on all other tanks is weak, a number of boundary layer sliding mode SISO-controllers can be used. For a rule of thumb: If a system can be controlled by a number of PID-controllers, the proposed approach has a good chance of regulating the system in a satisfactory manner.

Another limitation is given by systems that include a hard dead time. Several numerical experiments were done were hard delay-elements were introduced into otherwise simple and good-natured systems. It showed that the introduction of these delay-elements, however small, completely prevented any form of convergence. However, as long as physical dead-times are approximated by first-order elements (this is done for instance in the dynamic pipe model of Modelica.Fluid) the proposed approach should work out fine.

Controllers based around sliding mode concepts don't have a well-defined gain. Small deviations in behavior of the controlled system can however result in large changes in controller output. In this sense, they behave similar to linear controllers with large gain. As such, they can be classified as aggressive controllers. If simulation models are not robust, there may be situations where a boundary layer sliding mode controller prevents the numerical solver from finding simulation results. For the same system, a cautiously tuned PI-controller will probably give better results.

Nevertheless, for a large class of simulation models, the proposed control concept will deliver good results, taking up only very little of the modelling experts time. As long as modelling experts keep the mentioned limitations in mind, the proposed controller can greatly improve productivity.

6 Conclusion

For modelling and simulation projects of complex technical systems, often many local controllers have to be modelled, without them being used in the final architecture. We identify boundary layer sliding mode control as a suitable approach, offering good performance without any tuning effort for many - but not all - systems. The major drawback of this class of controllers - sensitivity to measurement noise - is irrelevant in the context of simulation models.

Acknowledgements

We thank Trey and Matt for inspiration.

References

Paul Acquatella, Wouter Falkena, Erik-Jan van Kampen, and Q Ping Chu. Robust nonlinear spacecraft attitude control using incremental nonlinear dynamic inversion. In *AIAA Guidance, Navigation, and Control Conference*, page 4623, 2012.

Karl J Åström and Björn Wittenmark. *Adaptive control.* Courier Corporation, 2013.

G Bartolini, A Ferrara, and E Usai. Chattering avoidance by second-order sliding mode control. *IEEE Transactions on Automatic control*, 43(2):241–246, 1998.

John Doyle. Guaranteed margins for lqg regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978.

Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications.* Crc Press, 1998.

Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

Linda R Petzold et al. A description of dassl: A differential/algebraic system solver. *Scientific computing*, 1:65–68, 1982.

Andreas Pfeiffer. Optimization library for interactive multicriteria optimization tasks. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 76, pages 669–680. Linköping University Electronic Press; Linköpings universitet, 2012.

Alexander Pollok, Dirk Zimmer, and Francesco Casella. Fractional-order modelling in Modelica. In *Proceedings of the 11th International Modelica Conference*, 2015.

Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

Michael Thümmel, Gertjan Looye, Matthias Kurze, Martin Otter, and Johann Bals. Nonlinear inverse models for control. In *Proceedings of the 4th International Modelica Conference*, pages 267–279, 2005.

Vadim Utkin, Jürgen Guldner, and Jingxin Shi. *Sliding mode control in electro-mechanical systems*, volume 34. CRC press, 2009.

Blas M Vinagre, Concepción A Monje, Antonio J Calderón, and José I Suárez. Fractional pid controllers for industry application. a brief introduction. *Journal of Vibration and Control*, 13(9-10):1419–1429, 2007.

K David Young, Vadim I Utkin, and Umit Ozguner. A control engineer's guide to sliding mode control. In *Variable Structure Systems, 1996. VSS'96. Proceedings., 1996 IEEE International Workshop on*, pages 1–14. IEEE, 1996.

Ali Zilouchian and Mohammad Jamshidi. *Intelligent control systems using soft computing methodologies*. CRC Press, Inc., 2000.

A Modelica Code of Simple Controller

Listing 1. Modelica Code of SimpleController

```
model SC
  ...
equation
  ...
  x[1] = actual_in_internal
    - target_in_internal;
  for i in 2:n_order loop
   x[i] = td1[i-1].y;
   x[i-1] = td1[i-1].u;
  end for;
  if exact_sliding_mode then
    y = if x*weights > 0 then output_max
    else output_min;
  else
    y = mean + authority*Modelica.Math.tanh(
         x*weights/(boundarylayer*unitsize));
  end if;

  annotation (...);
end SC;
```

Mission-Dependent Sequential Simulation for Modeling and Trajectory Visualization of Reusable Launch Vehicles

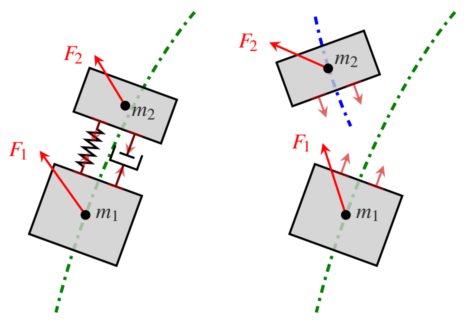
Lâle Evrim Briese

Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany,
Lale.Briese@dlr.de

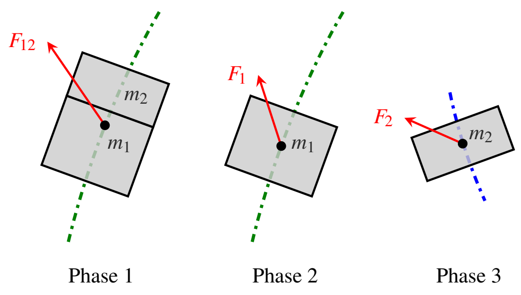
Abstract

The multibody modeling and visualization of reusable launch vehicles is a challenging task due to their variable structure regarding component separation and engine cutoffs during ascent and descent. However, the number of states within a MODELICA-based multibody model has to remain constant during a simulation. Therefore, the variable structure of launch vehicle models is often considered by using time- and state-dependent conditional statements and separation components. Such an approach can lead to a higher number of equations in the model and to a higher model complexity, respectively. In this paper, a mission-dependent sequential simulation approach for the modeling and trajectory visualization of launch vehicle systems is introduced. Here, the system is divided into characteristic phases, which are modeled with the DLR LauncherApplications Library capitalizing its modular, reusable and user-friendly structure to maintain compatibility between phases and to decrease the overall model complexity and the number of equations.

Keywords: launch vehicle modeling, trajectory, visualization, mission-dependent modeling, sequential simulation



(a) End-to-End Simulation.



(b) Sequential Simulation.

Figure 1. Schematical Representation of the Main Differences between End-to-End Simulation and Sequential Simulation.

1 Introduction

The multibody modeling of reusable launch vehicles can be a challenging task due to multiple disciplines involved in the modeling and simulation process, such as environment, aerodynamics, propulsion, structural dynamics, separation dynamics, as well as Guidance, Navigation and Control (GNC).

Especially, the variable structure of the overall multidisciplinary launch vehicle system has to be taken into account during the multibody modeling approach. Depending on the launch vehicle design and its mission requirements, the launch vehicle configuration can experience significant changes in its system structure. This can include the separation of stages, fairing, and payload as well as the time- and state-dependent *main engine cutoff* (MECO). These characteristic events for launch vehicles often lead to a change in the overall model structure and to a high number of states and equations due to separation or MECO as well as to non differentiable time-dependent step commands during engine ignition or cutoff.

In the object-oriented modeling language MODELICA (Modelica Association, 2014) the number of states and equations has to remain constant during a simulation. In recent developments, launch vehicle multibody models were obtained which use time- and state-dependent conditional statements to characterize the variable system structure, as presented in (Acquatella B., 2016). Also, separation models as introduced in (Acquatella B. and Reiner, 2014) can be used to fulfill the requirements for a fixed number of states and equations during the overall simulation by defining a connectivity condition between separated components and continuously simulating all components even after separation. These kind of models and methods are used within end-to-end simulations, which consider the multibody dynamics of all separated and connected launch vehicle components simultaneously as shown in Figure 1(a).

However, this approach for end-to-end simulations can lead to higher model complexity, number of equations and processing times. For instance, the separation models double the number of equations of motion since the generalized forces for each body have to be calculated, whether connected or separated, producing an overhead of unnecessary simulation data and calculations. Additionally, time-dependent conditional statements can result in time and state events, which can be undesired due to their disadvantages during the integration process.

As a consequence, new modeling and simulation methods are needed, which can reduce the model complexity and avoid an overhead of equations. Several methods have been developed over the past years for robust and efficient handling of variable structure systems with index changes as described for example in (Mehlhase et al., 2014; Mehlhase, 2015), (Zimmer, 2010) and (Elmqvist et al., 2014; Mattsson et al., 2015). These methods include a new experimental modeling language with an interpreter for variable structure DAE systems as proposed in (Zimmer, 2010), the usage of a *multi-mode Pantelides algorithm* to enable the simulation of models similar to state machines as shown in (Elmqvist et al., 2014; Mattsson et al., 2015) or the usage of complementary software languages and tools such as PYTHON or MATLAB for iterative simulations restricted by user-defined events.

Within this paper, a script-based sequential simulation method is proposed using only DYMOLA's built-in commands without the need for conventional variable structure modeling techniques or the usage of multiple software languages. The sequential simulation method shown in this paper is used especially for the modeling and visualization of launch vehicle systems, but can be implemented in any other application field. The modeling strategy is based on a similar approach used for multi-phase trajectory optimization of launch vehicle systems as described by (Schnepper, 2014). Therefore, the overall simulation is divided into characteristic *phases* or *modes* as referred to by variable structure modeling methods. These phases are then simulated sequentially while using consistent models with reduced complexity corresponding to each phase as shown in Figure 1(b).

First, a brief overview of the launch vehicle modeling framework is presented in Section 2. In Section 3 the sequential simulation method is introduced while taking a closer look into the multi-phase models used within trajectory optimization and visualization. In Section 4, the results regarding two launch vehicle concepts obtained by the proposed methods are presented. Finally, in Section 5 a summary and outlook will be provided.

2 Launch Vehicle Modeling

The launch vehicle modeling framework used for the modeling and visualization of expendable and reusable launch vehicles provides generic model components for launch vehicle simulations considering three or six degrees of freedom (DoF). Complementary to the *Modelica Standard Library* (Modelica Association, 2014) the following MODELICA-based libraries are used:

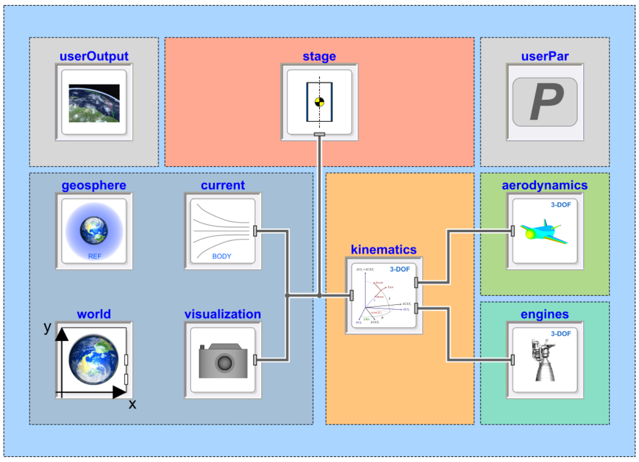


Figure 2. Overview of the 3-DoF / 6-DoF Launch Vehicle Modeling Framework provided by the DLR LauncherApplications Library.

- **DLR Environment Library** for modeling of environmental effects such as gravity acceleration of a rotating non-spherical planet (Briese et al., 2017).
- **DLR SpaceSystems Library** for modeling of satellite (Reiner and Bals, 2014) and launch vehicle systems (Acquatella B., 2016).
- **DLR LauncherApplications Library** for modeling of launch vehicle systems for example as consistent 3-DoF multibody models (Briese et al., 2018).

In Figure 2, a basic overview of the modular and internally consistent model structure of the modeling framework for a 3-DoF launch vehicle is given. All components are declared as `replaceable` models, which depend on dedicated and mutually shared *BaseClass* partial models as conceptually introduced in the DLR Environment Library. Since all components are based on the same parametric dataset *userPar*, this approach leads to a consistent model behaviour even if the level of detail is changed between simulations to conduct conceptual or detailed analyses assuming existent transition conditions.

All multibody related components are connected using multibody frames as defined in the *Mechanics.MultiBody* package of the *Modelica Standard Library* capitalizing the acausal structure of MODELICA. Mutually shared variables between main components are declared as input parameters and accessed by the parameter interface instead of using signal-based connectors to avoid using generic signal-based interfaces and thus unclear unit definition.

Within the launch vehicle modeling framework as shown in Figure 2 the *world* component provides the basic planet-dependent coordinate systems as well as more accurate gravity acceleration models. The *geosphere* component deliveres atmospheric parameters based on planet-specific atmosphere models. Since only the 3-DoF representation of the launch vehicle model for trajectory optimization is considered here, the usage of the *current* (wind) component will not be explained in this paper.

Table 1. Subset of the Basic Parameter Dataset *userPar*.

	Name	*Description*
3-DoF	start_lat	Initial Latitude
	start_lon	Initial Longitude
	start_h	Initial Altitude
	start_vel	Initial Velocity
	start_fpa	Initial Flight Path Angle
	start_chi	Initial Azimuth Angle
6-DoF	start_psi	Initial Heading Angle
	start_theta	Initial Pitch Angle
	start_phi	Initial Bank Angle
	start_p	Initial Roll Rate
	start_q	Initial Pitch Rate
	start_r	Initial Yaw Rate

2.1 Parameterization

In the context of sequential simulation and therefore changing parameter datasets, it is important to provide a common parameterized dataset interface which is consistent for any derived model complexity. The user-defined record *userPar* has to remain structurally constant in terms of model definition and yet has to remain accessable for the user to change certain parameters during the model redeclaration between two subsequent phases. This is ensured by using the `inner` and `outer` concept of MODELICA as well as extendable records which are based on a mutual baseclass.

For example, in Table 1 a subset of parameters provided by the *userPar* component for the definition of kinematic position, velocity and orientation of the launch vehicle is shown. Depending on the level of detail, the record must contain at least start values for the initial latitude, longitude, altitude, velocity as well as the flight path and azimuth angles for 3-DoF launch vehicle models. For higher levels of detail, the basic parameter dataset can be extended to include additional initial values such as the bodies' orientation angles and roll rates.

2.2 Kinematics & Dynamics

The kinematic state variables chosen for the 3-DoF representation of a launch vehicle system depend on the geocentric position $\boldsymbol{r}_G$ as well as the velocity $\boldsymbol{v}_N$ with respect to the local horizontal coordinate system as shown in Equation (1):

$$\boldsymbol{r}_G = \begin{bmatrix} \text{latitude} \\ \text{longitude} \\ \text{radius} \end{bmatrix}, \ \boldsymbol{v}_N = \begin{bmatrix} v_{\text{North}} \\ v_{\text{East}} \\ v_{\text{Down}} \end{bmatrix}. \qquad (1)$$

The 3-DoF equations of motion describing the translational launch vehicle dynamics are derived using the sum of external forces. However, the variable mass of the launch vehicle system has to be taken into account by using dedicated variable mass models based on a similar structure as

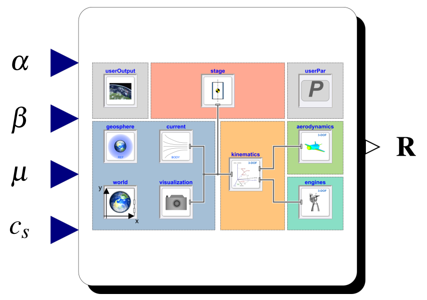


Figure 3. Overview of the Replaceable Main Model provided by the Launch Vehicle Modeling Framework for Sequential Simulation.

the body components in the *Modelica Standard Library*, where the mass m is a state variable defined by the equation $dm = \dot{m}$ and where the mass flow rate dm is calculated using the engine specification data within the *userPar* component.

The external forces consider the gravity G provided by the environment component *world*, the aerodynamic forces A given by the *aerodynamics* component and the thrust force T calculated by the *engines* component. The aerodynamic forces can be determined using aerodynamic coefficients which can be interpolated using multi-dimensional look-up tables. Similar to the aerodynamic forces, the thrust force can be determined using engine specification data or multi-dimensional time-dependent tables.

Since only the translational motion is taken into account, a time-scale separation between translational and rotational dynamics is assumed. Consequently, all angular velocities and accelerations are set to zero. The 3-DoF model is obtained from a constrained 6-DoF point mass model using dedicated kinematic models provided by the DLR Environment Library.

2.3 Model Structure

In this section, a generic model structure which can be used for the sequential simulation of launch vehicle systems is introduced. Therefore, the overall launch vehicle modeling framework as shown in Figure 2 has to be modified for further usage within the sequential simulation.

For instance, to visualize the trajectory of a launch vehicle based on trajectory optimization results or to conduct controllability studies, a set of input and output parameters are required which have to influence the system dynamics. For a 3-DoF launch vehicle model necessary input parameters are generally defined as the angle of attack α, sideslip angle β, bank angle μ, and the engine throttling factor c_s. The output parameters are stored in the result vector $\mathbf{R}$. This is schematically shown in Figure 3.

Based on the chosen analysis type of the sequential simulation these input and output parameters along with the replaceable system components and the parameterized dataset have to be adapted to the corresponding analysis requirements.

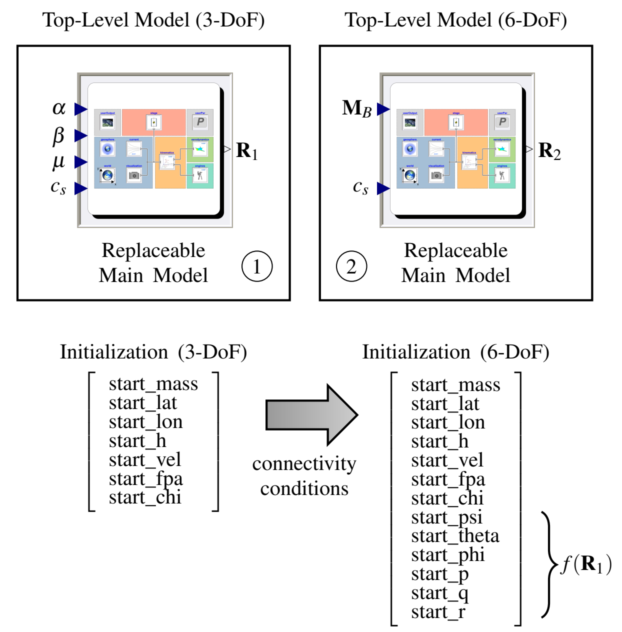


Figure 4. Overview of Multiple Top-Level Simulation Models.

For this purpose, it can be necessary to define a consistent top-level simulation model which is used as an interface during the sequential simulation and where the main launch vehicle modeling framework can be replaced according to the current phase definition and requirements. For example, the simplified top-level simulation model as used in this paper contains the replaceable main model shown in Figure 3 to allow its redeclaration at transition times between phases. For the trajectory visualization, additional multi-dimensional interpolation tables are included in the top-level model to directly provide input parameters for the replaceable main model as obtained by previously performed trajectory optimization. Alternatively, a control structure can be wrapped around the replaceable main model to conduct controllability analyses for specific phases while taking into account uncertainties in the system dynamics.

Another possibility is to define multiple top-level simulation models with consistent interface elements and connectivity conditions to allow the usage of models with different levels of detail. This approach is illustrated in Figure 4 for a sequential simulation containing two different setups for 3-DoF and 6-DoF point mass models. Since the number of states increases from seven for the 3-DoF case to 13 for the 6-DoF case, connectivity conditions as well as additional calculations depending on the output parameters of the previous simulation have to be defined to obtain corresponding initial conditions for the transition between the two top-level models. This approach is similar to the transition conditions as issued for multi-mode DAE systems in (Mattsson et al., 2015).

Since the influence of different levels of detail on the sequential simulation process is out of scope for this paper, this option will not be further investigated here.

3 Sequential Simulation

Within this section, the sequential simulation method based on DYMOLA's built-in command *simulateExtendedModel* will be discussed further. This function simulates a model with modified parameters and provides a subset of results at final simulation time which can be used for subsequent simulations (DYMOLA, 2018). Often, the *simulateExtendedModel* function is used for user-defined parameter studies, in which a certain set of parameter values is propagated into the model using a script-based approach within DYMOLA which is similar to the DYMOLA built-in command *experiment*.

3.1 *simulateExtendedModel* Function

In the DYMOLA documentation, the arguments of the *simulateExtendedModel* function including their default values are defined as shown in Code 1:

Code 1. Arguments of the *simulateExtendedModel* command.

```
simulateExtendedModel(
 problem           = "",
 startTime         = 0.0,
 stopTime          = 1.0,
 numberOfIntervals = 0,
 outputInterval    = 0.0,
 method            = "Dassl",
 tolerance         = 0.0001,
 fixedstepsize     = 0.0,
 resultFile        = "dsres",
 initialNames      = fill("",0),
 initialValues     = fill(0,0),
 finalNames        = fill("",0),
 autoLoad          = true)
```

The *problem* corresponds to the MODELICA path name of the top-level sequential simulation model as shown in Figure 4. Furthermore, the start and stop time of the simulation can be accessed, as well as the number of intervals, the output interval, the integration method, its tolerance and if applicable its fixed stepsize. For each simulation run a unique name for the result file can be chosen. Most importantly, the initial values of chosen parameters defined by their initial names can be set at the start of the simulation and the final values of a user-defined set of parameters can be derived after the simulation run.

This command has been applied for example within the *Hybrid Decomposition* method in combination with DYMOLA-external tools like PYTHON and MATLAB as described in (Mehlhase, 2015) and (Stüber, 2017). For these methods, the online evaluation of the results are indispensable, since the chosen long time simulations depend on event-based information about the end-point of the simulation. Because the sequential simulation models as shown in this paper are constrained in time, the use of DYMOLA-external tools to control the simulation time is not needed. However, the replaceable main models as shown in Figure 3 can be translated into *Functional Mock-up Units* (FMU) and used for multi-phase trajectory optimization in MATLAB as introduced in (Briese et al., 2018).

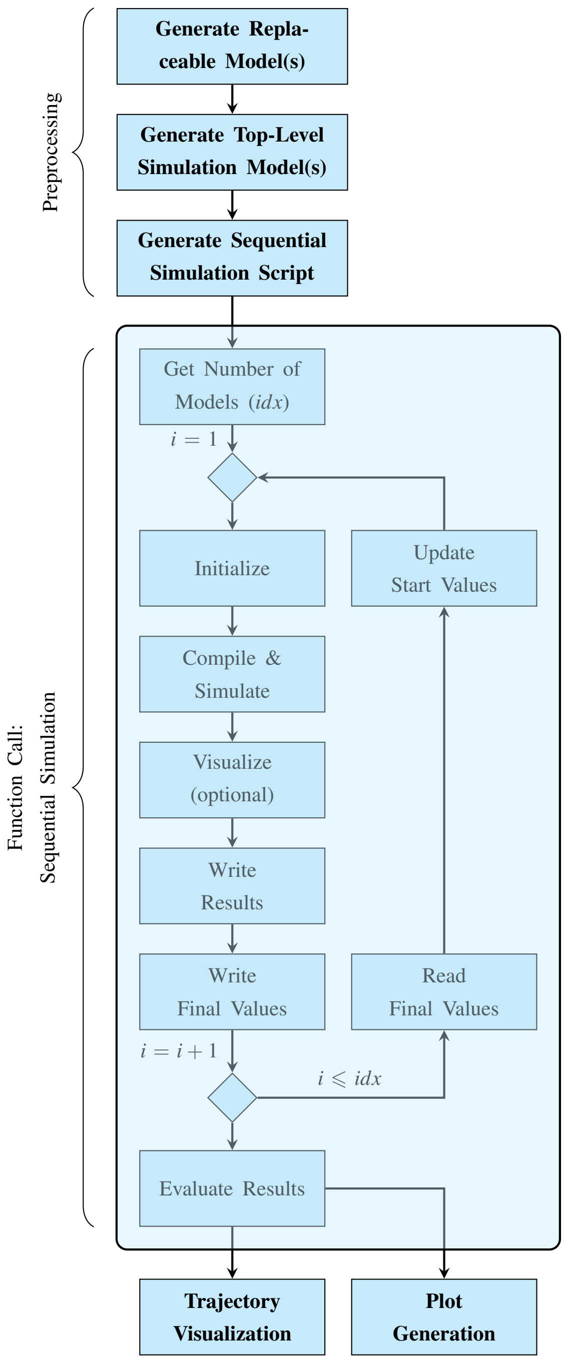


Figure 5. Overview of the Sequential Simulation Method.

3.2 Script-based Sequential Simulation

A basic overview of the sequential simulation method is illustrated in Figure 5. First, several preprocessing steps have to be conducted including the generation of (replaceable) models for each phase, at least one or more top-level simulation models, as well as the corresponding sequential simulation script which is implemented as a function within DYMOLA as shown in Code 2. Since sequential simulation scripts depend on the chosen modeling and simulation purpose, the adaptivity of this script allows a flexible handling of variable model structures and their simulation purpose.

Code 2. Sequential Simulation Script Example.

```
function sequentialScript
 import SI=Modelica.SIunits;
 input SI.Time phaseTimes[:,:];
 input String  stateNames[:];
 input String  modelNames[:];
 input String  exampleName;

protected
 Boolean ok;
 Real out[:];

algorithm
 for i in 1:size(modelNames,1) loop
  (ok,out) := simulateExtendedModel(
   exampleName + "(redeclare " +
     modelName[i] + " test)"
   startTime  = phaseTimes[i,1],
   stopTime   = phaseTimes[i,2],
   resultFile = "phase_" + String(i),
   initialNames  = if i == 1 then
     fill("",0) else stateNames,
   initialValues = if i == 1 then
     fill( 0,0) else out,
   finalNames = stateNames);
  end for;
annotation (__Dymola_interactive=true);
end sequentialScript;
```

Following the preprocessing steps, the sequential simulation script is executed. The current top-level sequential simulation model defined by the DYMOLA path `exampleName` is executed iteratively for each phase using the DYMOLA built-in function *simulateExtendedModel*. The annotation `__Dymola_interactive=true` has to be included into the function call to use the built-in function without errors or warnings.

In this sequential simulation example, the replaceable main models are referenced by their DYMOLA path `modelName`. They can be redeclared within the corresponding top-level sequential simulation model at each phase. Therefore, the top-level model has to be translated and simulated for each phase separately due to possible major changes in the system structure of the replaceable main models. This is for example the case, if for each phase different datasets have to be obtained from external sources or if the number of states changes.

The arguments of the DYMOLA command *simulateExtendedModel* allow for the initialization of certain parameters within the simulation model and to store chosen final values as a function output in the vector `out`. Within the sequential simulation method, these arguments are only used for the initialization of state variables depicted by the `stateNames` parameter since any other changes within the models are defined using either the parameter setup in the record `userPar` or direct parameter propagation. In the first iteration, the state variables are initialized using the start values stored within the record *user-*

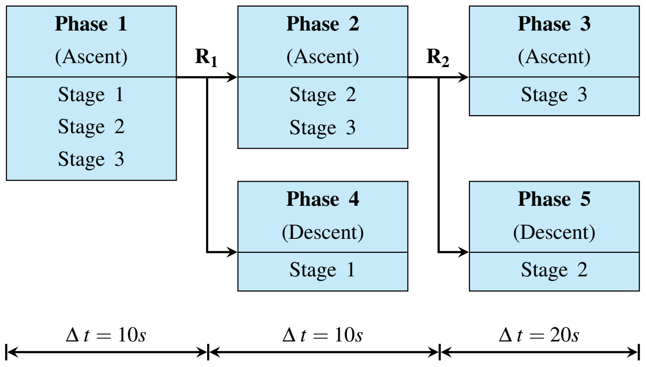
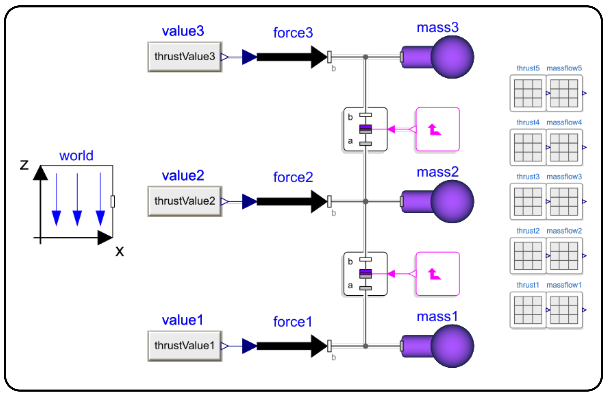
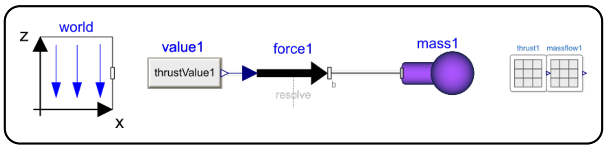


Figure 6. Test Case: Overview of the Phases.



(a) Test Case: Overview of the End-to-End Simulation Model.



(b) Test Case: Overview of the Sequential Simulation Model.

Figure 7. Test Case: Overview of the Generic Model Structure.

Par. Therefore, the function parameters `initialNames` and `initialValues` are declared as empty vectors. Additionally, the simulation time limits given by `startTime` and `stopTime` depend on user-defined inputs or alternatively on the final time of the previous model.

For each phase, a result file containing only the state variables, their derivatives as well as input and output variables is saved. Depending on the application requirements, the result files can be further reduced by deactivating certain outputs for example if the sequential simulation method has to be used in an optimization. At the end of the sequential simulation script the result files are evaluated for further post-processing steps. For example, the post-processing step can include an overall trajectory visualization based on the final result files using the visualization components provided by the DLR Visualization Library (Bellmann, 2009) and SimVis as shown in Figure 11 in Section 4.

4 Results

In this section, the sequential simulation method is applied for two different launch vehicle concepts. First, a generic test case is introduced to demonstrate the capabilities of the sequential simulation method while using a rather simplified multibody model. Second, the delta-winged reusable launch vehicle concept AURORA is modeled and visualized while using the sequential simulation method.

4.1 Test Case

To demonstrate the capabilities and advantages of sequential simulation, a simplified yet generic 6-DoF multibody model with three stages, two separations and therefore five dedicated phases has been analyzed as depicted in Figure 6. For better comparability, two dedicated models for the end-to-end as well as the sequential simulation concept have been generated as shown in Figure 7.

The end-to-end simulation model contains three stages represented by variable point mass models. Each stage is connected to the next stage with separation components. As long as the boolean signal is defined as `false` the stages remain connected and they are separated if otherwise (see (Acquatella B. and Reiner, 2014)).

The mass flow rate and thrust magnitude for each stage are obtained from MATLAB files using multi-dimensional time-dependent interpolation tables to include a realistic modeling challenge since import and interpolation of external engine or aerodynamic datasets have to be performed very often in design studies. For this purpose, interpolation tables for each phase have to be included into the simulation model. The thrust is applied only in vertical z-direction such that a vertical ascent of all stages is required since only uniform gravity in z-direction and no side forces are applied on the point mass models. Additionally, the thrust and mass flow rate are set to zero using state dependent conditional statements if the overall mass of the components reaches a certain threshold.

The sequential simulation model on the other hand only provides one variable mass point to represent the stages. Consequently, the individual mass contribution of each stage is summed up into an overall mass representation of the body. This is generally the case if the connection between each rigid stage is assumed as an ideal and not an energy-consuming connection. Additionally, only the interpolation data for the current phase is needed and can be referenced directly by changing the table name to obtain correct results for the corresponding stages.

The function introduced in Code 2 is a simplified example for a generic sequential simulation. Any other variation of the sequential simulation script can be implemented according to the use case. For instance, the script has to be adapted to new requirements if splitted trajectories are taken into account as presented in Figure 6. In this case, the first phase is simulated providing the output vector $\mathbf{R_1}$ for the final states. This output vector is then used to initialize the states for the second and fourth phase while using fixed start values for the mass as defined by

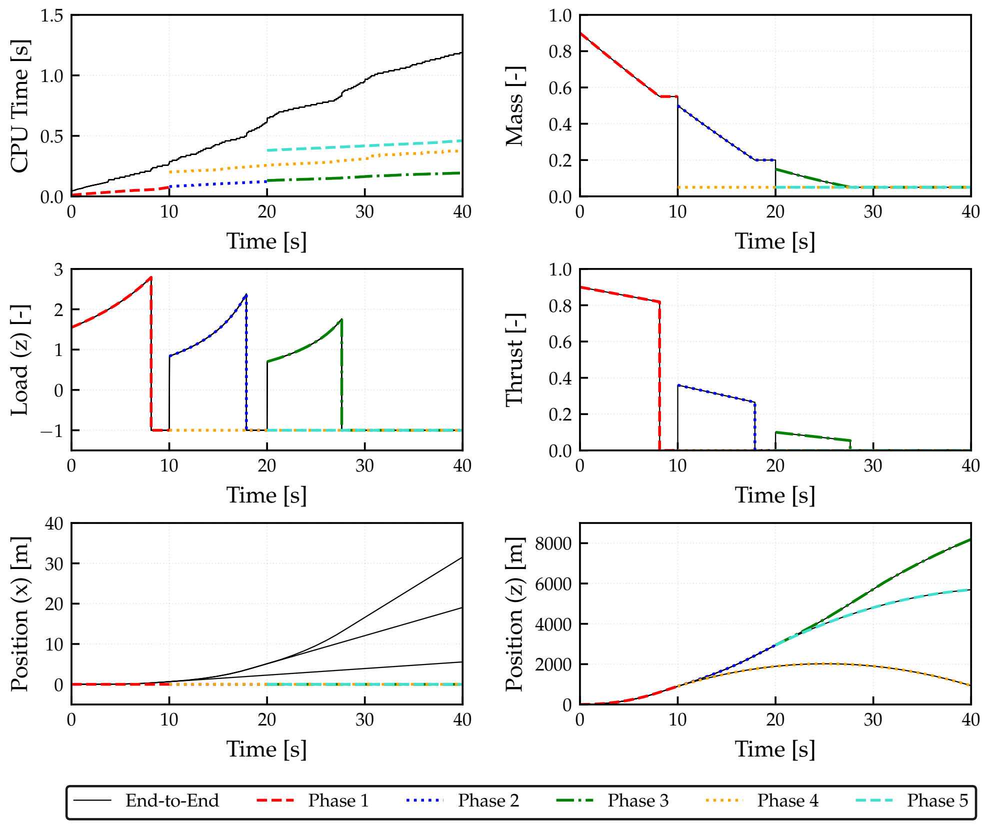


Figure 8. Test Case: Results obtained for the End-to-End as well as the Sequential Simulation Concept.

Table 2. Test Case: Comparison of Simulation Parameters.

	End-to-End	*Sequential*
Integration method	dassl	dassl
Integration Tolerance	$1e-7$	$1e-7$
Number of States	39	13
Number of State Events	3	3 (total)
Number of Time Events	5	3 (total)
Number of Equations	3791	1134

the corresponding stage structure. The same approach is considered for the third and fifth phase using the second output vector $\mathbf{R}_2$.

In Table 2 an overview of general simulation parameters for both modeling and simulation concepts is given. For better comparability both simulation setups use the same integration method and tolerances as well as similar number of intervals. In the sequential simulation case, the simulation parameters are shown for one phase. Only the number of state and time events are given as a sum over all phases for the sequential simulation. The number of state events cannot be reduced since they are used to conditionally set the mass flow rate to zero if a certain threshold for the propellant mass in the variable point mass models is reached. On the other hand, the time events can be reduced. Three time events result from the time-dependent table interpolation while the other two are needed to define the point of time where separation occurs. Another advantage of the sequential simulation concept in this context is that it only contains linear systems of equations, while the end-to-end simulation model has to initialize and solve two nonliner systems of equations due to the separation components.

The number of equations and states for each simulation phase can be significantly reduced by only using one single point mass model and by removing the separation components as well as additional table interpolation components. Obviously, this is only true if one specific phase model is considered. But since the *world* model has to be considered separately in each phase of the sequential simulation and since the end-to-end simulation model uses only three variable point mass models compared to five for the overall sequential simulation process, the number of equations is higher and can therefore also result in higher compilation times. Although the compilation time of variable structure models plays a major role regarding comparability aspects, the additional compilation time is not further considered since the focus of the paper is set on the reduction of the model complexity.

However, the overall CPU time can be reduced significantly as can be seen in Figure 8 summed up for all sequentially simulated phases. The simulation time is especially sensitive towards the chosen number of integrals for each phase. While this parameter can be only defined once for the end-to-end simulation, the number of intervals for each phase can be chosen individually according to the simulation needs. For instance, if the third phase is not relevant for the overall simulation purpose, this parameter could be decreased in order to reduce the simulation time which would not be possible for an end-to-end simulation concept.

As shown in Figure 8, the results for both concepts are the same regarding loads, thrust, mass flow rates as well as the position in z-direction of each body component except for the translational drift in the horizontal x-direction after each separation while using the end-to-end simulation model. Since all forces are applied only vertically in the bodies' z-direction according to a flat *world* assumption, this drift error should be ideally zero as can be seen by the results of the sequential simulation models. The resulting drift of the multibody parts after separation is caused by the acceleration residual at separation time, which in turn is caused by the *Baumgarte stabilization* solution during joint body motion whose accuracy is sensitive to computational erros as mentioned in (Acquatella B. and Reiner, 2014).

4.2 Use Case

In this section, the results obtained by the sequential simulation method as applied for the *horizontal takeoff and horizontal landing* (HTHL) delta-winged launch vehicle AURORA are presented.

A generic trajectory for a winged reusable launch vehicle configuration is illustrated in Figure 9. Here, the ascent phase can be divided into multiple phases which can include an horizontal takeoff, a powered ascent including the gravity turn as well as MECO after which an intermediate ballistic flight phase can follow. After stage separation the upper stage continues with the ascent into the desired orbit, including several phases which can be constrained by fairing and payload separation. For descent, the flyback vehicle and its phase definitions are mainly dependent on the descent configuration and the horizontal or vertical landing mode.

This reusable launch vehicle concept as described by (Kopp et al., 2017) consists of seven dedicated phases as shown in Table 3. The definition of these phases depends highly on the mission and has to be considered before the trajectory optimization. The results obtained with the Trajectory Optimization Package *trajOpt* by (Schnepper, 2014) and the multi-objective and multi-phase optimization tool MOPS as described in (Joos, 2016) will be used in the following.

The top-level simulation model as shown in Figure 4 remains the same for each phase and only the replaceable main models as described in Section 2.3 are redeclared within the sequential simulation script using a similar script structure as shown in Code 2. The fixed start and final time values are provided by the results of the trajectory optimization. The state definition of all models corresponds to the state vector given in Equation (1) including the varying overall mass of the launch vehicle system.

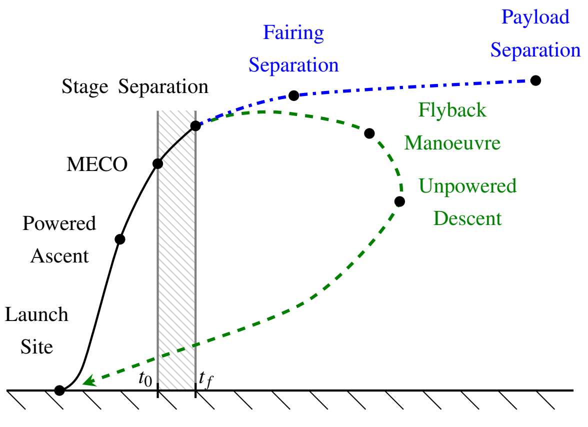


Figure 9. Schematic Trajectory of a Winged Launch Vehicle.

Table 3. Use Cases: Overview of the Phases (AURORA).

Phases	*Stages*	*Description*
Phase 1	US+MS	Horizontal liftoff
Phase 2	US+MS	Ascent phase (rocket engines)
Phase 3	US+MS	Ballistic phase & separation
Phase 4	US	Ascent of the upper stage
Phase 5	MS	Return maneuvre 1
Phase 6	MS	Return maneuvre 2
Phase 7	MS	Return to the launch site

Furthermore, the initialization of the models can be realized either using the output vectors of previously simulated phases or the start values of the trajectory optimization results for each phase. Consequently, the iterative call of the *simulateExtendedModel* function in Code 2 is divided into three parts:

1. **Sequential Simulation of Phases 1 to 3**:
   Phase 1 is initialized by itself using default values defined by parameters in the *userPar* component. Phase 2 and 3 are initialized depending on the final values of the corresponding previous phases. The final values of Phase 3 at its end-point are stored in a separate output vector `out_sep`.

2. **Simulation of Phase 4**:
   This phase is initialized using only the position- and velocity-related final values stored within the `out_sep` vector. The state parameter referencing the mass of the system is overwritten by the overall mass of the upper stage (US).

3. **Sequential Simulation of Phases 5 to 7**:
   Phase 5 is initialized using the position- and velocity-related final values stored within the `out_sep` vector. The parameter referencing the mass state of the system is overwritten by the overall mass of the main stage (MS). The last two phases are initialized depending on the final values of the corresponding previous phases.

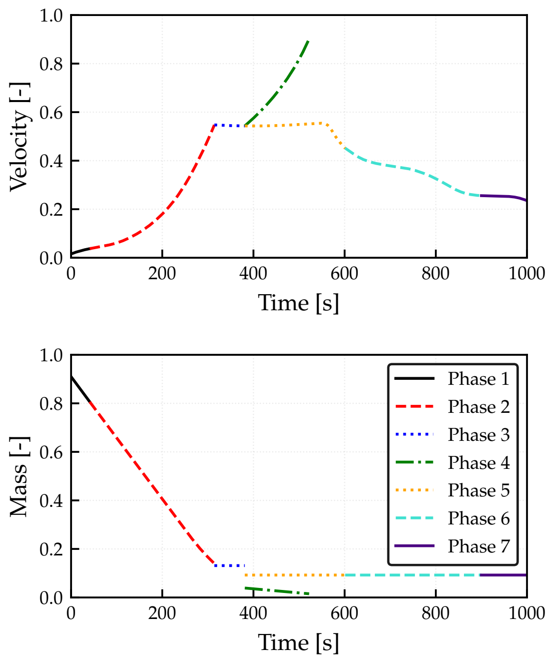


Figure 10. Use Case: Results obtained for each Phase (AURORA).

The main challenges in the modeling of this reusable launch vehicle concept are related to changing engine configurations as well as aerodynamic characteristics for different flight conditions. For this purpose, at least two different aerodynamic datasets for each aerodynamic coefficient have to be considered which can be either used for the ascent or the descent phase depending on the vehicle's current Mach number and angle of attack. Using the replaceable main models, simply one aerodynamic dataset has to be inclcuded into the current simulation model, which can reduce the overhead of parameters used within the compiled top-level simulation model.

Since a change of the number of states is not required within this simulation, the overall system configuration can remain the same. Therefore, the same replaceable main model setup can be used by reconfiguring the parameter initialization and importing different datasets for each phase. This way, the same number of states and equations is generated which reduces the model complexity as well as the overhead of unnecessary calculations and also leads to a consistent model structure througout the sequential simulation.

In addition to the results presented in (Briese et al., 2018), the normalized velocity and the overall mass of the vehicle during the sequential simulation are shown in Figure 10. The overall mass within each phase is varying continuously based on the mass flow rate obtained by the vehicle's engine specification. Due to the separation after the third phase at approximately $390s$ steps occur in the mass formulation resulting from the reinitialization of the masses of the upper stage (US) in Phase 4 and the winged flyback stage (MS) in Phase 5. While these step commands do not effect the integration within each phase, these kind of steps can lead to numerical errors within end-to-end simulations and would have to be smoothed using approximation functions by increasing the model complexity. Furthermore, as shown by the velocity results, the transition between each phase is performed smoothly by initializing all kinematic state variables consistently.

Finally, the result of the subsequent trajectory visualization performed with the DLR Visualization Library as a post-processing step is shown in Figure 11. In this case, the ascent of the combined stages (Phase 1 to 3), the ascent of the upper stage (Phase 4) as well as the descent of the winged main stage (Phase 5 to 7) are visualized separately using the corresponding post-processed result files of the sequential simulation.

5 Conclusion

In this paper, the script-based sequential simulation method based on DYMOLA's built-in command *simulateExtendedModel* has been introduced.

The purpose was a simplification of end-to-end simulations containing time- and state-dependent conditional statements and separation models to handle the transition within variable structure launch vehicle models. For this purpose, the replaceable launch vehicle models within the launch vehicle modeling framework have been introduced and the implementation of these models within top-level simulation models to be used by the sequential simulation has been shown. The significance of a common and consistently defined parameter set has been underlined.

Additionally, the sequential simulation method has been further investigated and subsequently applied on a winged reusable launch vehicle configuration. The results show that the transition conditions between each phase can be determined using the initialization arguments within the *simulateExtendedModel* function. Furthermore, time- and state-dependent conditional statements as well as the usage of separation models can be avoided. Also, the number of equations remains the same for each phase and can be significantly lower than the overall launch vehicle model for end-to-end simulations.

As an example regarding the application options for the sequential simulation method, the subsequent trajectory visualization as a post-processing step in DYMOLA was presented for the AURORA launch vehicle configuration.

For future research, the *Hybrid Decomposition* method could be extended by using DYMOLA-internal methods. Also, the simulation arguments could be designed flexibly depending on the simulation characteristics to dynamically adapt to the computational requirements of each phase.

Acknowledgements

This study was performed within the DLR internal project AKIRA for the comparison of key technologies for future reusable launch vehicles.

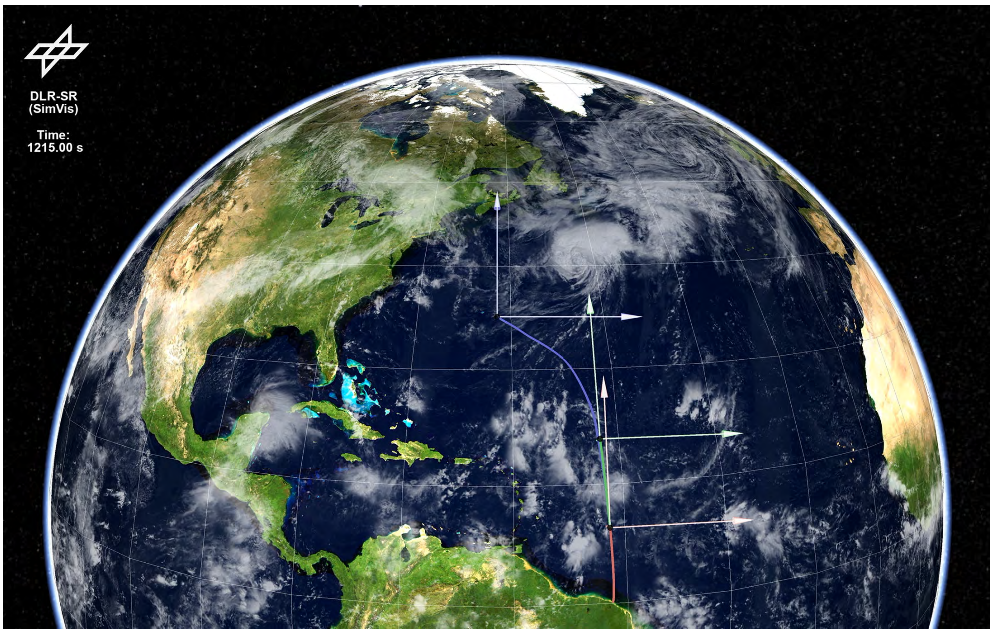


Figure 11. Trajectory Visualization of the Reusable Launch Vehicle AURORA.

References

P. Acquatella B. Launch Vehicle Multibody Dynamics Modeling Framework for Preliminary Design Studies. In *6th International Conference on Astrodynamics Tools and Techniques, ICATT*, 2016.

P. Acquatella B. and M. J. Reiner. Modelica Stage Separation Dynamics Modeling for End-to-End Launch Vehicle Trajectory Simulations. In *Proceedings of the 10th International Modelica Conference*, 2014.

T. Bellmann. Interactive Simulations and advanced Visualization with Modelica. In *Proceedings of the 7th International Modelica Conference*, 2009.

L. E. Briese, A. Klöckner, and M. Reiner. The DLR Environment Library for Multi-Disciplinary Aerospace Applications. In *Proceedings of the 12th International Modelica Conference*, 2017.

L. E. Briese, K. Schnepper, and P. Acquatella B. Advanced Modeling and Trajectory Optimization Framework for Reusable Launch Vehicles. In *Proceedings of the IEEE Aerospace Conference (in press)*, 2018.

DYMOLA. *Version 2018*. Vélizy-Villacoublay, France, Dassault Systèmes, 2018.

H. Elmqvist, S. E. Mattsson, and M. Otter. Modelica extensions for Multi-Mode DAE Systems. In *Proceedings of the 10th International Modelica Conference*, 2014.

H.-D. Joos. MOPS - Multi-Objective Parameter Synthesis. Technical Report DLR-IB-SR-OP-2016-128, DLR German Aerospace Center, 2016.

A. Kopp, M. Sippel, S. Stappert, N. Darkow, J. Gerstmann, S. Krause, D. Stefaniak, M. Beerhorst, T. Thiele, A. Gülhan, R. Kronen, K. Schnepper, L. E. Briese, and J. Riccius. Forschung an Systemen und Technologien für wiederverwendbare Raumtransportsysteme im DLR-Projekt AKIRA. In *Deutscher Luft- und Raumfahrtkongress*, 2017.

S. E. Mattsson, M. Otter, and H. Elmqvist. Multi-Mode DAE Systems with Varying Index. In *Proceedings of the 11th International Modelica Conference*, 2015.

A. Mehlhase. *Konzepte für die Modellierung und Simulation strukturvariabler Modelle*. PhD thesis, Technical University Berlin, Germany, 2015.

A. Mehlhase, D. G. Esperon, J. Bergmann, and M. Merkle. An example of beneficial use of variable-structure modeling to enhance an existing rocket model. In *Proceedings of the 10th International Modelica Conference*, 2014.

Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification Version 3.3. Revision 1*, 2014.

M. J. Reiner and J. Bals. Nonlinear Inverse Models for the Control of Satellites with Flexible Structures. In *Proceedings of the 10th International Modelica Conference*, 2014.

K. Schnepper. Trajektorienoptimierung in MOPS - Das Paket trajOpt Version 1.0. Technical report, DLR German Aerospace Center, 2014.

M. Stüber. Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python. In *Proceedings of the 12th International Modelica Conference*, 2017.

D. Zimmer. *Equation-Based Modeling of Variable-Structure Systems*. PhD thesis, ETH Zurich, Switzerland, 2010.

Model predictive allocation control for leg-wheel mobile robot on loose soil considering wheel dynamics

Takatsugu Oda[1] Hiroki Yoshikawa[1] Naoki Shibata[1] Kenichiro Nonaka[1] Kazuma Sekiguchi[1]

[1]Mechanical systems engineering, Tokyo city university, Japan, {knonaka,ksekiguc}@tcu.ac.jp

Abstract

For the planetary exploration rover, to cope with the layer of heterogeneous superficial deposits called regolith is important so as to achieve designed traction. In order to consider effects of wheel motion, model predictive allocation control is proposed. To cope with complex terramechanics in *MPC (model predictive control)*, the identification technique is introduced; the proposed MPC is formed as a linear optimization problem. The rover model and terramechanics are described using *modelica* and simulate to evaluate the performance of the proposed method. The suppressed of superfluous slip and enhancement of traction performance is numerically shown.

Keywords: Terramechanics, Allocation control, Model predictive control, Model identification.

1 Introduction

Developments of autonomous robots are widely studied to apply for various environments in which human cannot work. Especially for planetary exploration, exploration rovers are generally adopted, because these robots can move planetary surface efficiently (Weisbin et al., 1997). However, on many planets, planetary surface is covered with a layer of heterogeneous superficial deposits called regolith; wheels might slip and not generate desired forces on these soil. In addition, since the deformable loose soil on which wheels passed are compressed, the terrain property (e.g. soil density, resistance force etc.) would change (Senatore and Sandu, 2011). Then, for autonomous robots equipped with drive wheel, wheel control technique is important to suppress the slip and wheel sinkage.

Based on the terramechanics which deal with the interaction of the wheel and soil (Taheri et al., 2015), many vehicle control method is studied. In order to enhance traction and energy performance, dual-criteria objective function for control allocation is proposed in (Iagnemma and Dubowsky, 2004). For the traction control of the rover, PI controller with slip ratio estimator (Yoshida and Hamano, 2002) and robust adaptive fuzzy control strategy (Zhengcai and Yang, 2014) are developed. In addition, *MPC (model predictive control)* which could consider the dynamics and constraint explicitly is applied to planetary rovers (Krenn et al., 2013). However, this method adopts the linearization technique to consider complex model and does not consider muli-pass effect.

In this paper, the model predictive allocation control for multi-wheel rovers considering the terramechanics including the multi-pass effect is proposed. The wheel dynamics is explicitly considered using MPC, dynamically infeasible allocated solution which static optimization method might have is avoided. To consider complex terramechanics model in linear MPC, black box identification technique is introduced; for identification, the rover model is operated in only linear region of terramechanics. Effectiveness of the model predictive allocation control is verified through numerical simulation using *modelica*; suppression of superfluous slip and enhancement of traction performance is shown. Note that to focus on the longitudinal motion, we assume lateral motion of the rover could be sufficiently small.

2 Vehicle and wheel model

2.1 Leg-wheel mobile robot

In this paper, a leg-wheel mobile robot consists of a body and limbs, wheels, as depicted in Fig. 1 is considered. By using redundancy of limbs equipped with six joint, the desired wheel arrangement is achieved (Yoshikawa et al., 2017). The wheels could drive independently and generate tire force based on terramechanics which express wheel-soil interaction mechanisms.

2.2 Wheel model based on terramechanics

We introduce the wheel model based on terramechanics (Senatore and Sandu, 2011) into the robot model in order to simulate the force and moment characteristics of the tire on a deformable terrain (Yoshikawa et al., 2017). Roughly, in this model, generated force of each wheels are calculated by integrating normal σ and shear τ_x stress distribution model which described in Fig. 2.

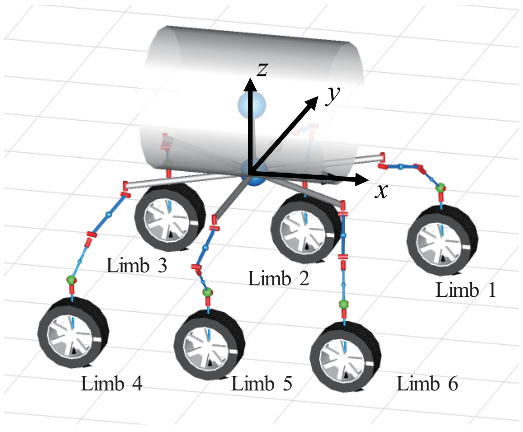


Figure 1. Leg-wheel mobile robot with six joints of each limb.

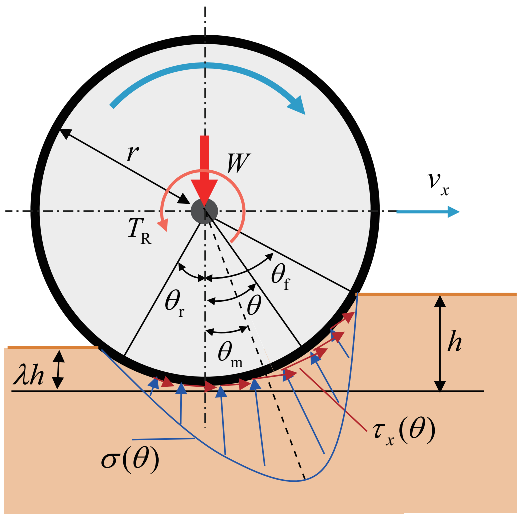


Figure 2. Normal and shear stress distribution concept of terramechanics while rolling.

The maximum stress angle is empirically estimated as following a linear function of the slip ratio κ and the entry angle θ_f:

$$\theta_m = (a_0 + a_1\kappa)\theta_f, \tag{1}$$

where a_0 and a_1 are constant parameter. The slip ratio is explained as follows with the wheel radious r, angular velocity ω and translational velocity v_x:

$$\kappa = \frac{r\omega - v_x}{\max(r\omega, v_x)}, \tag{2}$$

The empirical normal stress distribution based on Reece's formula are expressed as follows:

$$\sigma(\theta) = \begin{cases} \sigma_f(\theta) & (\theta_m \le \theta < \theta_f), \\ \sigma_r(\theta) & (\theta_r < \theta \le \theta_m), \end{cases} \tag{3}$$

$$\sigma_f(\theta) = \left(ck'_c + \rho b k'_\phi\right)\left[\frac{r}{b}(\cos\theta - \cos\theta_f)\right]^n, \tag{4}$$

$$\sigma_r(\theta) = \left(ck'_c + \rho b k'_\phi\right) \left[\frac{r}{b}\left(\cos\left\{\theta_f - \frac{\theta - \theta_r}{\theta_m - \theta_r}(\theta_f - \theta_m)\right\} - \cos\theta_f\right)\right]^n, \tag{5}$$

where the Bekker-Reece sinkage exponent n is a linear function of slip ratio; $n = n_0 + n_1|\kappa|$. b is wheel width and c is cohesion stress of the soil, ρ is soil density, k'_c is cohesion related soils parameter, k'_ϕ is angle internal friction related soil parameter, respectively.

The empirical shear stress distribution introduced by Janosi and Hanamoto is widely used (Taheri et al., 2015):

$$\tau_x = (c + \sigma(\theta)\tan\phi)(1 - e^{-j_x(\theta)/k_x}), \tag{6}$$

$$j_x(\theta) = r[\theta_f - \theta - (1-\kappa)(\sin\theta_f - \sin\theta)], \tag{7}$$

where j_x is the shear displacement in longitudinal direction and ϕ is angle of internal friction of the soil, k_x is shear deformation modulus in the longitudinal direction, respectively.

In order to express the multi-pass effect, as which the effect of repetitive loading of deformable soils is expressed, some parameters are modified; the modified soil density ρ_p, soil cohesion c_p and shear deformation modulus in the longitudinal direction k_{xp} are formulated as follows:

$$\rho_p = \rho\left(1 + (1 - e^{\frac{-\kappa_0}{k_1}})k_2 + k_3 n_p\right), \tag{8}$$

$$c_p = c\left(1 + (1 - e^{\frac{-\kappa_0}{k_1}})k_2 + k_3 n_p\right), \tag{9}$$

$$k_{xp} = k_x\left(1 - (1 - e^{\frac{-\kappa_0}{k_1}})k_2 - k_3 n_p\right), \tag{10}$$

where κ_0 is slip ratio of previous pass and n_p is number of passes, k_1, k_2, k_3 are constant parameters, respectively.

The vertical force F_z and the longitudinal force F_x, rolling resistance torque T_R are calculated by integrating normal σ and shear τ_x stress distribution as follows:

$$F_z = rb\int_{\theta_r}^{\theta_f}\{\tau_x(\theta)\sin\theta + \sigma(\theta)\cos\theta\}\,d\theta, \tag{11}$$

$$F_x = rb\int_{\theta_r}^{\theta_f}\{\tau_x(\theta)\cos\theta - \sigma(\theta)\sin\theta\}\,d\theta, \tag{12}$$

$$T_R = r^2 b\int_{\theta_r}^{\theta_f}\tau_x(\theta)\,d\theta. \tag{13}$$

Note that firstly in order to accord calculated load Fz to given load W, i.e. $W = Fz$, Eq. (11) is solved with respect to sinkage h, and then Eqs. (12)(13) are calculated with the calculated sinkage.

3 Wheel model identification

Because the original wheel model based on terramechanics Eqs. (11)–(13) are too complex to consider in controller design, we extract a more simple model. In this paper, we adopt black-box identification technique; tire dynamics is expressed as a first order delay system and tire model is expressed as a static linear function, as shown in Fig. 3. Because of dependent properties, we make and refer the *LUT (look up table)* for each parameter.

Wheel dynamics is expressed as follows:

$$G_{\mathrm{w}}(s) = \frac{K}{T_{\mathrm{l}}s+1}, \tag{14}$$

where the gain $K(T_{\mathrm{w}}, W, \kappa_0, n_{\mathrm{p}})$ and the time constant $T_{\mathrm{l}}(T_{\mathrm{w}}, W)$ are expressed using the LUT. On the deformable terrain, small slip ratio region called linear region is mainly used; a linear tire model is adopted. The tire model, which express relationship between slip ratio κ and longitudinal force F_x, is expressed as follows:

$$F_x = a\kappa + b, \tag{15}$$

where $a(W, \kappa_0, n_{\mathrm{p}})$ and $b(W)$ are parameter using the LUT.

4 Control methods

4.1 Conposition and guidance control

In this paper, the proposed method consists of guidance controller and model predictive allocation control. In the guidance controller, the vehicle total traction force is calculated based on the vehicle position and longitudinal velocity. Then, in the allocation controller, the vehicle total traction force is allocated among the wheels.

In the guidance controller, to achieve the target vehicle position x_r and velocity $v_{x,r}$, a linear quadratic regulator is adopted. The vehicle total traction force $F_{x,\mathrm{all}}$ is desinged as follows:

$$F_{x,\mathrm{all}} = -K_{\mathrm{LQ}} \begin{bmatrix} x - x_r \\ v_x - v_{x,r} \end{bmatrix}, \tag{16}$$

where K_{LQ} is the optimal feedback coefficient matrix and given by solving the continuous time algebraic Riccati equation.

4.2 Model predictive allocation control

Control allocation could be solved without considering any dynamics, because it is essentially static optimal problem. However, without considering the dynamics of the wheel, optimal solution might not be achievable. To deal with this problem, a model predictive allocation control is proposed.

In this control allocation, state $\xi = [\kappa_1, \cdots, \kappa_6]^{\mathrm{T}}$ and input $u = [T_{\mathrm{w},1}, \cdots, T_{\mathrm{w},6}]^{\mathrm{T}}$ are introduced to formulate the proposed method as follows:

$$\min_{u(i|k)} \sum_{i=1}^{N+1} (||\xi(i|k) - \xi_r(i)||_Q^2 + ||u(i|k)||_R^2), \tag{17a}$$

subject to

$$\xi(i+1|k) = A\xi(i|k) + Bu(i|k), \tag{17b}$$

$$F_{x,\mathrm{all}}(i|k) = \sum_{i=1}^{6} F_{x,i}(\kappa_i(i|k), f_{z,i}, \kappa_{0,i}, n_{p,i}), \tag{17c}$$

$$|\xi(i|k)| \leq \overline{\xi}, \tag{17d}$$

$$|u(i|k)| \leq \overline{u}. \tag{17e}$$

Eq. (17a) express the index function and $Q, R > 0$ are weight matrix and N is predicive horizon. ξ_r is reference state which calculated by assuming equal traction force distribution. Eq. (17b) is the discretized state-space representation model of Eq. (14). Eq. (17c) is the condition for achieve vehicle total traction force $F_{x,\mathrm{all}}$. Eqs. (17d)(17e) are the region of state and input expressed by bound of state $\overline{\xi}$ and input $\overline{u}$. Model predictive allocation control Eq. (17) is solved every sampling times with the condition $\xi(1|k) = \xi(k)$; the first element of imput $u(1|k)$ is adopted as applied input at t.

5 Numerical simulation

5.1 Conditions

In order to verify the effectiveness of the proposed method, we model the leg-wheel mobile robot with wheel model based on terramechanics using *modelica* (see more details in (Yoshikawa et al., 2017)), and apply the model predictive allocation control.

Some environmental parameters including soil parameters are set, so as to simulate robot behavior in lunar environment (Ishigami et al., 2007; Senatore and Sandu, 2011). The vehicles are commanded to run at constant speed 0.4 m/s. A static control allocation control technique, which does not consider any dynamics, with PI wheel velocity controller is introduced as a comparing method; same guidance controller Eq. (16) is adopted in the both (Prop. and Comp.) controller. Then, the effectiveness of considering wheel dynamics in control allocation is clearly evaluated.

5.2 Results and discussions

Simulation results are shown in Fig. 4; we especially show the response of limb 1–3, because of symmetric leg position. As Fig. 4(a) indicate, both methods almost achieve the target velocity. The allocated force in compared method is not achieved because of discontinuous change; on the other hands, using the proposed method, which considers the wheel dynamics, the wheel torques change continuously. Then, wheel torques of the proposed method are smaller than that of comparing method and is generated efficiently, as depicted in Fig. 4(b). Result of the acceleration, Fig. 4(c) shows increased vertical load of rear limb. This load shift leads to the bias of the slip ratio, as depicted in Fig. 4(d). Neither considering

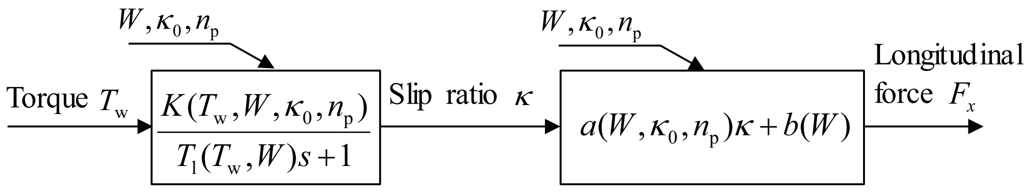

$W, \kappa_0, n_\mathrm{p}$
$W, \kappa_0, n_\mathrm{p}$
Torque T_w
$\frac{K(T_\mathrm{w}, W, \kappa_0, n_\mathrm{p})}{T_1(T_\mathrm{w}, W)s + 1}$
Slip ratio κ
$a(W, \kappa_0, n_\mathrm{p})\kappa + b(W)$
Longitudinal force F_x

Figure 3. Structure of identified wheel model.

dynamics or not, the low load wheels are commanded large slip ratio so as to equalize tire forces. The longitudinal force distribution is, however, different from load distribution, because of considering the muti-pass effect and the compressed soil property, as depicted in Fig. 4(e). This slip ratio distributions should be suppressed, because slip ratio occurs the sinkage as Fig. 4(d)(f) shows. Then, the amounts of sinkage are reduced in the proposed controller by reducing the slip ratio distribution shown in Fig. 4(g). Moreover, from the point of view of the effect of considering wheel dynamics, desired acceleration in LQR controller is achieved, because the proposed controller allocates facile longitudinal force; then, overshooting phenomenon of vehicle velocity is removed. It is why total slip ratio value is reduced.

6 Conclusion

In this paper, to consider wheel dynamics in allocation control, model predictive control method is adopted. To consider the complex and nonlinear terramechanics including multi-pass effect in linear MPC, identified wheel model consists of wheel dynamics and tire model is introduced. Numerical simulation is conducted using *modelica*, the effectiveness of the proposed control method is clearly shown.

Future direction of this study is to consider the lateral motion and to evaluate the traction efficiency.

References

Karl Iagnemma and Steven Dubowsky. Traction control of wheeled robotic vehicles in rough terrain with application to planetary rovers. *The international Journal of robotics research*, 23(10-11):1029–1040, 2004.

Genya Ishigami, Akiko Miwa, Keiji Nagatani, and Kazuya Yoshida. Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil. *Journal of Field robotics*, 24(3):233–250, 2007.

Rainer Krenn, Andreas Gibbesch, Giovanni Binet, and Alberto Bemporad. Model predictive traction and steering control of planetary rovers. 2013.

C Senatore and C Sandu. Off-road tire modeling and the multi-pass effect for vehicle dynamics simulation. *Journal of Terramechanics*, 48(4):265–276, 2011.

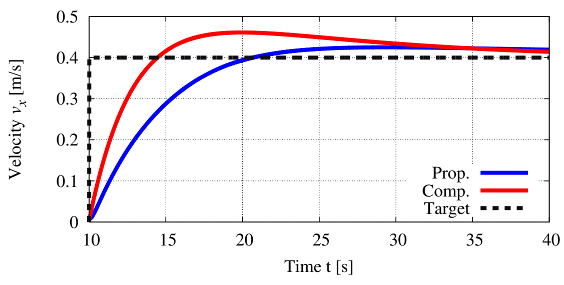
Sh Taheri, C Sandu, S Taheri, E Pinto, and D Gorsich. A technical survey on terramechanics models for tire–terrain interaction used in modeling and simulation of wheeled vehicles. *Journal of Terramechanics*, 57:1–22, 2015.

CR Weisbin, D Lavery, and G Rodriguez. Robotics technology for planetary missions into the 21st century. 1997.

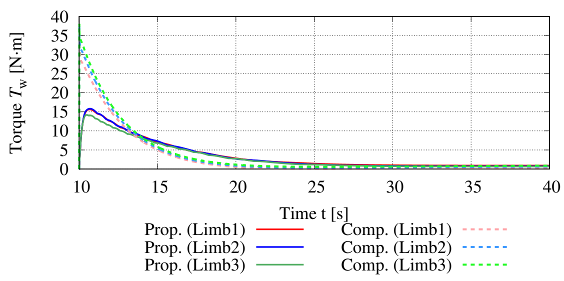
Kazuya Yoshida and Hiroshi Hamano. Motion dynamics and control of a planetary rover with slip-based traction model. In *Unmanned Ground Vehicle Technology IV*, volume 4715, pages 275–287. International Society for Optics and Photonics, 2002.

Hiroki Yoshikawa, Takatsugu Oda, Kenichiro Nonaka, and Kazuma Sekiguchi. Modeling and simulation of wheel driving systems based on terramechanics for planetary explanation rover using modelica. In *Proceedings of the 12th International Modelica Conference*, number 132, pages 901–907. Linköping University Electronic Press, 2017.

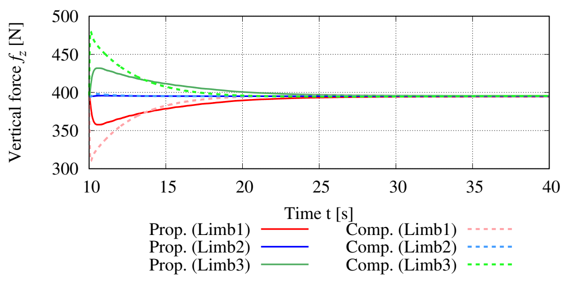
Li Zhengcai and Wang Yang. Robust adaptive fuzzy control for planetary rovers while climbing up deformable slopes with longitudinal slip. *Advances in Aerospace Engineering*, 2014, 2014.



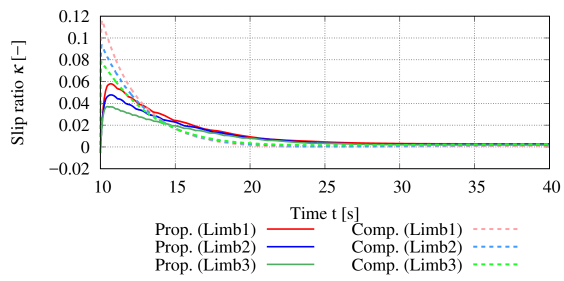
(a) Vehicle longitudinal velocity v_x.



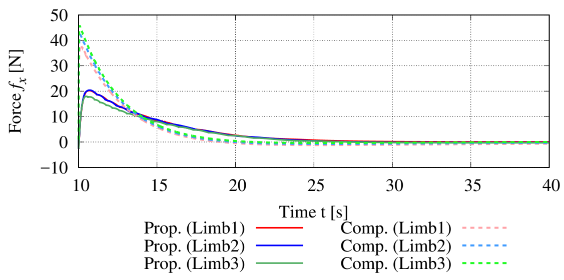
(b) Wheel torque T_{w}.



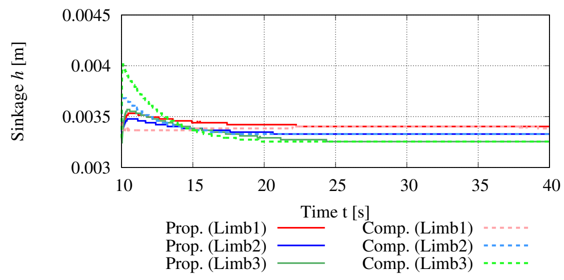
(c) Vertical force F_z.



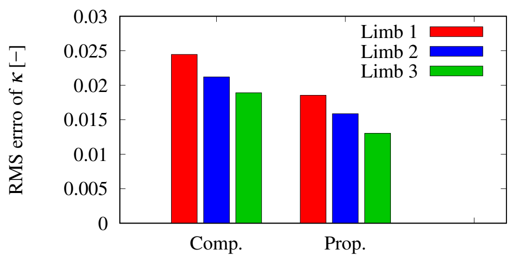
(d) Slip ratio κ.



(e) Longitudinal force F_x.



(f) Wheel sinkage h.



(g) RMS of slip ratio κ.

Figure 4. Simulation results.

Generating FMUs for the Feature-Based Language Bloqqi

Niklas Fors[1] Joel Petersson[2] Maria Henningsson[2]

[1]Department of Computer Science, Lund University, Sweden, `niklas.fors@cs.lth.se`
[2]Modelon AB, Sweden, `{joel.petersson, maria.henningsson}@modelon.com`

Abstract

In this paper, we describe how we generate Functional Mock-up Units (FMUs) for the automation block language Bloqqi. This allows Bloqqi control programs to be tested with simulations of the physical processes they control. The physical process can be specified in any tool that supports the Functional Mockup-Interface (FMI) standard. For example, we have successfully run Bloqqi programs together with Modelica models exported as FMUs. Bloqqi programs execute at discrete times, and we describe how this is handled in the implementation of the `DoStep` function, specified in the standard.

Keywords: FMI; Bloqqi; code generation

1 Introduction

Automation control systems are usually programmed using block diagrams. These diagrams contain blocks and connections that describe the data-flow between the blocks. Examples of languages include Function Block Diagrams from the standard IEC 61131, Bloqqi (Fors and Hedin, 2016) and ControlBuilder from the company ABB. Before these programs are deployed in a plant, they are tested, often with test cases written in the language itself. For more complex dynamic processes, there are other tools more suitable for describing the dynamics of the processes, for example, the modeling language Modelica (Modelica, 2018). Thus, it would be useful to write the control program as a block diagram and specify the model of the physical process in a modeling language and test them together.

The *functional mock-up interface* (FMI) (Blochwitz et al., 2012) is a standard that allows dynamic models described by different tools to be used together. For example, one part of a composed model can be exported by one tool and another part can be exported by another tool. Many Modelica tools allow models to be exported as *functional mockup units* (FMUs).

In this paper, we describe how programs in the block language Bloqqi can be exported as (Co-Simulation source) FMUs. Bloqqi is an object-oriented language for automation control systems that supports the specialization mechanisms *connection interception* and *block redeclaration*. The language also has support for feature mechanisms, making it possible to describe variants in libraries that the user can select from. The Bloqqi tools are open source and covered by the Modified BSD License.

The contribution of this paper is partly a description of how Bloqqi programs are executed by describing how they are translated to C code. The C code can be generated without any external dependencies, making it easy to integrate the code into existing systems and run it on embedded systems. The paper also contributes with a description of how the C code is adapted for the FMI standard, and how the `DoStep` function defined in the standard is implemented. A Bloqqi FMU is a bit different than a FMU for a continuous-time physical system, since a Bloqqi FMU only executes at discrete time steps according to the sampling period. We also give some examples of Bloqqi programs exported as FMUs and simulated together with models specified in Modelica.

This paper begins with background on FMI and SSP in Section 2. SSP is a standard for specifying the composition of FMUs. Then, the Bloqqi language is introduced in Section 3, and Section 4 describes how Bloqqi diagrams are translated to C programs. These C programs are then adapted for FMI, which is described in Section 5. Examples of how we have used FMUs are described in Section 6. The paper ends with a discussion of related work in Section 7 and conclusions in Section 8.

2 Background

2.1 FMI

The Functional Mock-up Interface (FMI) is a tool-independent standard with support for both model exchange and co-simulation of dynamic models (Blochwitz et al., 2012). Version 1.0 of the standard was released in 2010, followed by version 2.0 in 2014. Using the FMI standard, models can be shared across all the 100+ tools that are currently supporting the standard. FMI uses a combination of XML-files and compiled C-code to create a Functional Mock-up Unit (FMU). An FMU is a zip-file with two major parts: a model description in XML-format, and a number of compiled binaries. Each FMU can contain multiple binaries to support different platforms. The standard defines two kinds of FMUs: Model Exchange FMUs and Co-Simulation FMUs. A Model Exchange FMU requires an external solver for the FMU to be simulated. A Co-Simulation FMU on the other hand has a solver embedded.

2.2 SSP

In order to create simulation models for complex systems it is often convenient to separate the system into its components, and perform the necessary simulations in a tool suitable for the domain. However, with increasingly complex systems and large dependencies between components, full system simulations are essential. System Structure and Parameterization (SSP) (Köhler et al., 2016) is a new open standard (under development by the Modelica Association) defining a standardized format for the connection structure of a network of FMUs. It also defines a standardized way to store and apply parameters to such a structure. Utilization of the FMI and SSP standard will allow for components to be developed in the tool best suited for the domain, while still allowing for system simulations including all system components.

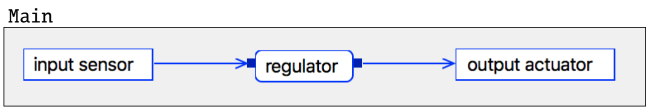
3 Bloqqi

Bloqqi is a data-flow language (Fors and Hedin, 2016; Fors, 2016) for programming the control subsystem part of automation systems. The language is a prototype language used for experimenting with language mechanisms for reuse. It has been developed in collaboration with ABB Automation Systems and specifically with the department responsible for development of tools for distributed control systems. Existing tools are based on the IEC 61131 standard for automation languages, which contains a family of five programming languages. The Bloqqi language is inspired by the Function Block Diagram language defined in this standard.

In Bloqqi, programs are specified as diagrams, where each diagram consists of blocks and connections between them that describe the data-flow. The blocks can be instances of other diagrams (which may be user-defined), making it possible to create hierarchical programs.

Bloqqi programs are executed periodically. For example, a program may run ten times per second. In each period, input values are read that are used to compute control signals. The control signals are then the output values of the program. Typically, input values are read from sensors and output values are sent to actuators that control the physical process.

This is illustrated in the diagram in Figure 1. Input values are blocks prefixed with the `input` keyword and the prefix `output` is used for output values. The block `regulator` is a normal block with an input port and an



```
diagramtype Main {
  input sensor: Int;
  output actuator: Int;
  regulator: Regulator;
  connect(sensor, regulator.in);
  connect(regulator.out, actuator);
}
```

Figure 1. A simple program in Bloqqi with an input value, a block and an output value. Both the visual and textual syntax are shown.

output port and is defined by another diagram. As can be seen in the figure, Bloqqi has both a visual and a textual syntax. The textual syntax is used as the serialization format when the diagrams are stored.

3.1 Diagram Inheritance

The Bloqqi language is similar to Modelica in that it supports diagram inheritance. A diagram S can extend another diagram T, making all blocks, connections, and parameters reusable in the subtype S. The subtype can also introduce new blocks, connections and parameters. There are two specialization mechanisms in Bloqqi: *connection interception* (Fors and Hedin, 2014) and *block redeclaration*. Connection interception allows the subtype to replace a connection defined in a supertype to instead go via a block added in the subtype. The connections in Bloqqi are directed, in contrast to Modelica where connections are undirected. Block redeclaration allows the subtype to specialize the type of a block that is defined in a supertype, similar to redeclare in Modelica.

3.1.1 Inheritance Example

Inheritance and the interception mechanism are illustrated in Figure 2. The diagram `P` describes a proportional regulator (`P`) with three input parameters: reference value (`r`), mesaured value (`y`) and the proportional coefficient (`kP`). The output parameter `u` is computed by taking the difference between the reference value and the measured value (the error), and then multiplying that with the coefficient. An instance of diagram `P` would then show the input parameters as input ports and the output parameter as output port.

The P-regulator is then extended with an integral part to incorporate the history of the error as well, as seen for diagram `PI`. The grey parts with dashed lines of the diagram are inherited from the supertype `P` and the blue parts with solid lines are declared locally in `PI`. The history of the error is stored in the block `acc` that simply accumulates the error value over time. The accumulated value is then multiplied with the integral coeffi-

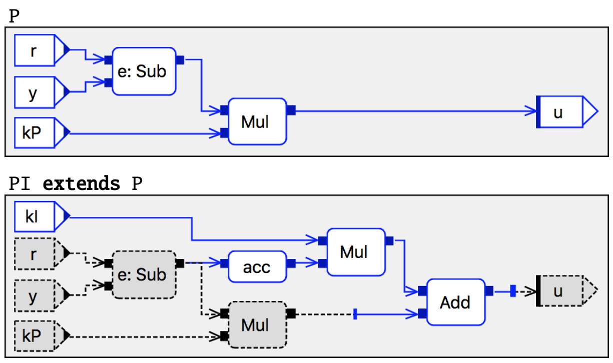


Figure 2. P-regulator (P) that is extended with an integral part in diagram PI. The connection to the output parameter u defined in P is intercepted in the subtype PI to add the integral part.

cient, which is added to the control signal. Here, the addition block intercepts the connection defined in the supertype P to go via locally declared addition block.

The accumulator block in the diagram PI contains a state to store the value to be used in the next execution period. The Bloqqi language has support for states in the form of variables, which are stored between the periods. When the variable is read, the value from the previous period is used, and when the variable is written to, a new value is stored to be used in the next period. Thus, the accumulator block contains a variable to accumulate the error value.

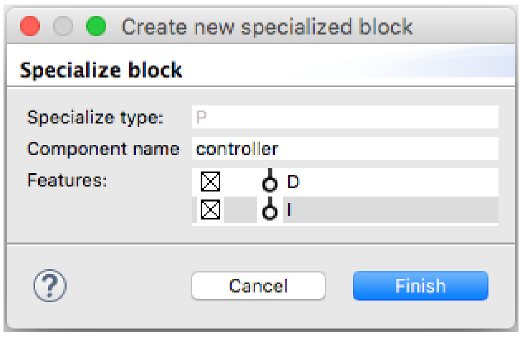
Currently, the Bloqqi language does not allow data-flow cycles. Instead, the user needs to break the cycle by introducing a variable.

3.2 Features

Bloqqi has also feature-based language mechanisms (Fors and Hedin, 2016). These allow the library developer to add optional features to diagrams, which can be selected by the library user, when the diagrams are instantiated. For example, consider the P-regulator in Figure 2; the integral part and the derivative part (not shown) can be seen as optional features to the diagram P. Speciyfing this using the feature-based mechanisms in Bloqqi would show an automatically generated wizard when the diagram P is instantiated as a block, as can be seen in Figure 3. The user can then select what features the block contains (in this case, both the features are selected).

4 Code Generation

Bloqqi programs are executed by first compiling them to C code. Running a Bloqqi program one period amounts to calling a C function. The function needs to know the input values and the state variables from the previous period, and store the new state variables to the next period


Create new specialized block

Specialize block

Specialize type: P
Component name: controller
Features: ☒ D
☒ I

Figure 3. Feature-wizard for diagram P when the integral and derivate part are specified as features using the feature-based mechanisms in Bloqqi.

and the output values. All the values are stored in a C struct and passed around as parameters. For example, the following C code will run the program in Figure 1 one period with the input value 10 for the sensor and printing the output value actuator.

```
Main_VARS v;
v.input.sensor = 10;
bloqqi(&v);
print(v.output.actuator);
```

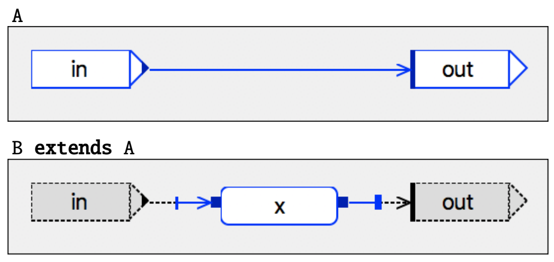
The Bloqqi compiler can generate a *driver function* that calls the bloqqi function a fixed number of times per second.

Each diagram is translated to a C function and each block is translated to a function call. The connections determine how the data flows between the function calls. Input parameters are mapped to function parameters and output parameters are mapped to the return value. Since diagrams can have several output parameters and C functions can only have one return value, the output parameters are wrapped in a struct and a value of this struct is returned. For example, the code implementing the diagram Main in Figure 1 is as follows:

```
void Main(
    Main_INPUT* _input,
    Main_OUTPUT* _output) {
  Regulator_RES regulator
    = Regulator(_input->sensor);
  _output->actuator = regulator.out;
}
```

The function has two parameters: a struct representing input values and another struct representing output values. The block regulator is translated to a function call and the value of the output port is stored in the struct Regulator_RES. Here, the only argument to the Regulator call is the value of the input port. If this block would have contained input values and output values, these would also be passed as arguments.

The entry point of a Bloqqi program is the diagram called Main with no parameters. The generated function bloqqi will just call the generated function Main.



```
diagramtype A(in: Int => out: Int) {
  connect(in, out);
}
diagramtype B extends A {
  x: X;
  intercept out with x.in, x.out;
}
```

Figure 4. Diagram B extends diagram A and intercepts the connection declared in A.

```
diagramtype B(in: Int => out: Int) {
  x: X;
  connect(in, x.in);
  connect(x.out, out);
}
```

Figure 5. Inheritance removed for diagram B defined in Figure 4 in the flattening process before C code is generated.

4.1 Inheritance Flattening

Before the C code is generated, the inheritance is removed by the source-to-source transformation *inheritance flattening*. This means that the inheritance is removed by copying all declarations in the supertype to the subtype. This transformation is possible since all block types are known statically.

For example, consider the two diagrams in Figure 4. Here, diagram B extends diagram A and intercepts the connection, from the input parameter (in) to the output parameter (out), and adds an extra block x in-between. The flattening transformation will remove the inheritance and transform the diagram B to a diagram without inheritance, as shown in Figure 5. We can see that the flattened diagram has the parameters declared in diagram A, the block x and the corresponding connections. The code generation will use this flat diagram when generating code.

4.2 Features

The feature mechanisms are based on inheritance and anonymous diagrams. The latter allows the block type to be an anonymous subtype of a diagram, which is illustrated in the following block declaration:

```
  b: Block { ... };
```

The type of block b is an anonymous subtype of diagram Block and may have the same content as a normal subtype. Before code generation, these anonymous subtypes are given a unique name and moved to the same scope level as all other diagrams.

4.3 Simple Integration

The C code generated by the Bloqqi compiler is simple to integrate into an existing system. The generated code does not depend on any external library, only timing functionality from the C POSIX library when the driver function is generated. The Bloqqi compiler can also generate code without a driver function with minimal dependencies. In that case, the only dependency is to the header file stdbool.h in the standard library. This allows the code to run on any ordinary operating system. The generated C code is compatible with the C99 standard.

4.4 Embedded Systems

We have successfully run Bloqqi programs on an Arduino Uno, which is a single-board with a microcontroller, and on a Raspberry PI, which is a single-board computer running Linux. The generated driver function was used on the Raspberry PI and an adapted driver function was written specifically for Arduino, since Arduino does not have any operating system and thus not support the C POSIX Library. The adapted driver function called the generated main function of the Bloqqi program and added a delay between the periods using the Arduino library.

4.5 Implementation

There are two tools for the Bloqqi language: a graphical editor and a compiler. Both these tools use the textual syntax as the serialization format. The editor visualizes the program and allows the user to change it, and the compiler generates C code for the program.

The semantical analysis for both the editor and compiler is specified using the metacompiler JastAdd (Ekman and Hedin, 2007), which supports the semantic formalism reference attribute grammar (Hedin, 2000) (RAGs). RAGs make it easy to reuse the semantic specification between the tools. The analysis includes inheritance flattening, which is briefly described in Section 4.1, that is used both in the editor and the compiler. When a diagram is opened in the editor, the editor shows the flattened diagram, which includes blocks, connections, etc., defined in the supertype (as seen in Figure 4). Thus, the shown diagram is computed based on the semantics of the language.

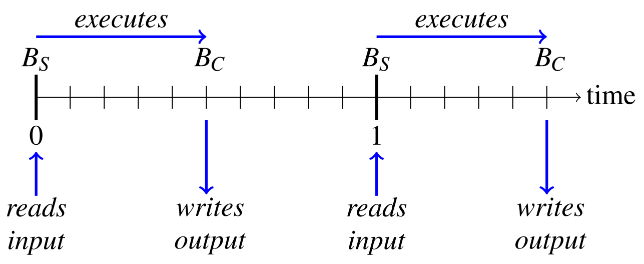


Figure 6. Simulation of a Bloqqi FMU with sampling period of 1 second and execution time of 0.5 seconds. The Bloqqi program starts executing at B_S and completes at B_C. The inputs are read at B_S and the outputs are set at B_C. The time on the axis is the simulated time.

5 FMU Generation

A Bloqqi program can be exported as a Co-Simulation FMU, which makes it easy to use Bloqqi programs together with models specified with other tools that support the FMI standard. The input and output values of the Bloqqi program are mapped to corresponding inputs and outputs of the FMU. The FMU executes periodically and the period can be set by the parameter `sampling-period`. It is also possible to simulate an estimated execution time of the Bloqqi program, which is set using the parameter `execution-time` and it will delay the output values with `execution-time` seconds. This is illustrated in Figure 6. The FMU can be generated as a *source FMU*, which includes the generated C code in the FMU.

The FMU generation for Bloqqi[1] is implemented using a port by Christopher Brooks[2] of FMU SDK[3] by QTronic which runs under Linux and MacOS. The Bloqqi compiler generates C code, as described in Section 4, and adds wrapper code for the FMU SDK and our own `DoStep` function. The Bloqqi compiler also generates an XML file describing the structure of the FMU.

5.1 Master Algorithm

When simulating a system of Co-Simulation FMUs, a *master algorithm* is responsible for exchanging data (inputs and outputs) between the FMUs at discrete *communication points*. The FMUs are then solved independently from each other between the communication points with their respective solver. The master algorithm uses functions, defined in the standard, for getting and setting inputs and outputs of the FMUs. Each FMU also defines the function `DoStep` that simulates one *communication step* with a given *communication step size* (which might vary between the calls). This function is called by the master algorithm. The following code illustrates one simple master algorithm with a fixed communication step size (`hc`) that simulates between the times `tStart` and `tEnd`:

```
curTc = tStart
while (curTc < tEnd) {
  read outputs from all FMU:s
  set inputs to all FMU:s
  call DoStep(curTc, hc) on all FMU:s
  curTc += hc
}
```

When using this algorithm together with a Bloqqi FMU, it is important that the communication points reflect the discrete times at when the Bloqqi program starts and completes execution. The master algorithm should exchange data every time the Bloqqi program starts an execution and when it completes an execution. Otherwise, the Bloqqi FMU will use old input values and delay its output values.

For the example shown in Figure 6, where the sampling period is 1 second and the execution time is 0.5 seconds, a master algorithm with the communication step size of 0.5 seconds will work fine without introducing any delays.

5.2 Implementation of DoStep Function

As described earlier, we have used FMU SDK but implemented our own `DoStep` function, with the goal of not introducing any unnecessary input/output delays. Our implementation internally keeps track of the next time the Bloqqi FMU should execute (according to the parameter `sampling-period`). If the Bloqqi FMU has started to execute, but has not yet completed, the function also keeps track of when it will complete. The function uses two variables to represent these times (`starts` and `completes`).

The function `DoStep` is called with the current communication point (`curTc`) and a communication step size (`hc`) by the master algorithm, and it will check if any of the time variables are before `curTc+hc`. An implementation sketch of `DoStep` is shown in Figure 7. As can be seen, if the Bloqqi program should start executing, the function will get the FMU inputs and set them to the inputs of the Bloqqi program, and then run the Bloqqi program one period. After that, the times for when the execution completes and when the next execution starts are calculated. It might happen that the execution completes during the same invocation of `DoStep`, and this is handled afterwards with the call to `set_outputs`, which sets the outputs of the FMU as the outputs of the Bloqqi program.

The `DoStep` function will only start executing if the simulated time has *passed* the value of `starts`. This is illustrated in Figure 8. Here, the second invocation of `DoStep` is between the times t_1 and t_2, and the invocation will not start the Bloqqi execution since the simulated time has not passed t_2. However, the third invocation will start the Bloqqi execution since it passes the time t_2.

[1] https://bitbucket.org/bloqqi/bloqqi-fmi
[2] https://github.com/cxbrooks/fmusdk2
[3] https://qtronic.de/en/fmusdk.html

```
// Internal state for Bloqqi program
Main_VARS v;

// When the next execution starts
starts = 0.0

// When a started execution completes.
// The value ∞ is used when there is
// no active execution.
completes = ∞

void DoStep(
    curTc, // current communication point
    hc     // communication step size
    ) {

  // ... error checking ...

  nextTc = curTc + hc

  // If an execution started in a previous
  // call to DoStep and completes during
  // this call.
  if (completes <= nextTc) {
    set_outputs(v.output)
    completes = ∞
  }

  if (starts < nextTc) {
    // Start a new execution
    v.input = get_inputs()
    bloqqi(&v)
    completes = starts + EXECUTION_TIME
    starts    = starts + SAMPLING_PERIOD

    // If the execution completes during
    // the same call to DoStep as when
    // it was started.
    if (completes <= nextTc) {
      set_outputs(v.output)
      completes = ∞
    }
  }
}
```

Figure 7. Implementation sketch for DoStep

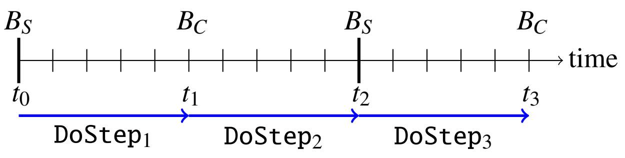


Figure 8. The execution starts when the simulated time has *passed* the value of the variable starts. This means that the first and third invocation of DoStep will start an execution, but not the second invocation. The second invocation is between the times t_1 and t_2, but it has not passed t_2, which is required for it to start executing.

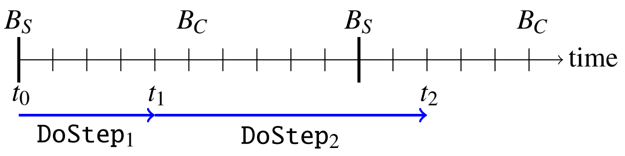


Figure 9. When the master algorithm makes the second call to DoStep in this example, the Bloqqi FMU should first write outputs from the previous execution before starting a new execution. This is handled by checking the variable completes before starting a new execution in DoStep.

We think that this behaviour makes it easier to use Bloqqi FMUs together with standard master algorithms, and where the Bloqqi FMU gets fresh input values when the execution starts. In the implementation of DoStep (Figure 7), this behaviour is captured by using the less than operator (<). On the other hand, when the output values are set, it is not needed to pass the time for when the execution completes, which is captured by using the less than or equal operator (<=). This will make the outputs visible when the execution is completed. For example, in Figure 8, the output values will be set by the first and third invocation of DoStep, making the outputs available at the times t_1 and t_3.

5.2.1 Handling Asynchronous Periods

It might happen that a call to DoStep starts an execution, but does not complete it, and that the execution is completed in a subsequent call to DoStep. This situation is handled by checking in the beginning of DoStep the variable completes. This code fragment also handles the situation when a previous call to DoStep started an execution, and the current call both completes the previous execution and starts a new execution. This is illustrated in Figure 9. Here, the first call to DoStep by the master algorithm is from t_0 to t_1, and which starts an execution. The second call is from t_1 to t_2, and which first writes the outputs from the previous execution and then starts a new execution of the Bloqqi program. In this example, the output values will be delayed (to time t_2 for the first execution) and old input values will be used (from time t_1 for the second execution). This delay is undesirable, but cannot be avoided without changing the master algorithm.

The error checking in the beginning of DoStep checks that the communication step size is not larger than the sampling period of the Bloqqi program, which in that case raises an error (fmi2Discard). We have chosen this behaviour as the Bloqqi program and the controller it represents can produce faulty results, as well as large delays, if used with a too large communication step size. For example, suppose the derivative in the controller is calculated by using the slope between two samples. It is then important that input values are updated between controller execution loops, as the difference would be zero otherwise. This would not happen with a DoStep that includes multiple executions of the Bloqqi program using the same sample of inputs.

5.2.2 Zero Execution Time

The estimated execution time can be set to 0. In this case, the master algorithm should make small communication steps from when the Bloqqi execution starts. For instance, assume that the first time the Bloqqi FMU should start executing is at time t_0 and that the Bloqqi sampling period is h, then the master algorithm should first call DoStep(t_0, hc_0) with a small communication step size hc_0 (it needs to pass the time t_0). Then, the master algorithm should take a longer communication step $hc_1 = h - hc_0$ to when the Bloqqi program is to start executing again, hence, it should make the call DoStep(t_1, hc_1), where t_1=$t_0 + hc_0$. If it is not possible for the master algorithm to make communication steps of variable length, then it should instead make the communication steps small enough to reduce the delay of the control signal to an acceptable level.

5.3 Selection of FMU Kind

As mentioned in Section2.1, there are two different kinds of FMUs specified in the standard, Model Exchange and Co-Simulation FMUs. There are advantages with both variants of the standard and selecting which one to use is not always straightforward. Due to its simple interface and ease of implementation, Co-Simulation FMUs are supported in most tools and provides a good platform for sharing the control models. It also provides an inherently sampled platform which reflects the sampled nature of a controller executing the Bloqqi program. For these reasons Co-Simulation FMUs were choosen as an appropriate first target kind for the Bloqqi FMUs.

Generally when performing simulations with FMUs an external entity will have control of the sampling time, and for the Bloqqi FMUs it cannot be assumed that the sampling times coincide with the discrete times of the Bloqqi program. The extreme case, when the sampling period of the master algorithm is larger than the sampling period of the Bloqqi FMU, is handled through a FMI2Discard of the DoStep. The mismatch of sampling times will inevitably lead to unnecessary delays in the control algorithms during simulations. The additional delays brought by the sampling might have a negative effect on the stability region of a controlled process that would not be present in the actual physical system, which could lead to inaccurate results of the simulations. For these reasons the possibilities of using Model-Exchange FMUs and event states to assure that the Bloqqi execution points are executed at the correct times would be interesting to investigate during the continuation of the project. Such an FMU could perform all its calculation, including registering its next event, while in event mode, and not contribute to the integration during the continuous time mode.

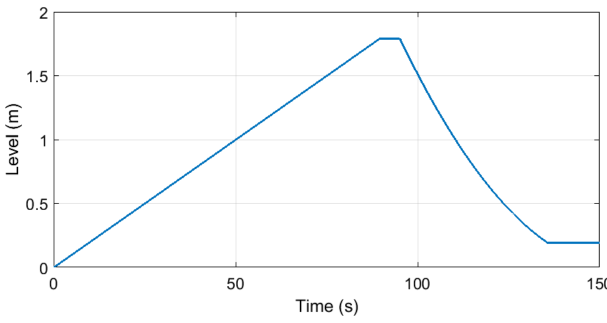


Figure 10. Simulation of the liquid level in a tank. The set point is first 1.8 meter and then changed to 0.2 meter.

6 Examples

We have exported Bloqqi programs as FMUs and run them together with plant model FMUs exported from Modelica models.

6.1 Tank Example

One example is a simple regulator for a tank of liquid. The regulator controls the liquid level using one input valve and one output valve, both valves can be opened or closed.

We have implemented the regulator in Bloqqi and modelled the tank in Modelica, and exported both models as FMUs. These two FMUs have then been aggregated (see Section 6.2) and simulated together. The simulation result can be seen in Figure 10. The figure shows how the liquid level changes over time. In this example, first, the set point of the liquid level is set to 1.8 m, and after it is reached, the set point is changed to 0.2 m.

The Bloqqi diagram for the tank regulator that opens and closes the valves is shown in Figure 11 (the diagram is called Tank). The parameter setLevel is the set point and the current liquid level is read from a sensor, which is represented by the block levelSensor. The blocks lowerValve and upperValve represent the actuators for the valves and take a boolean as input (true to open the valve and false to close the valve). The di-

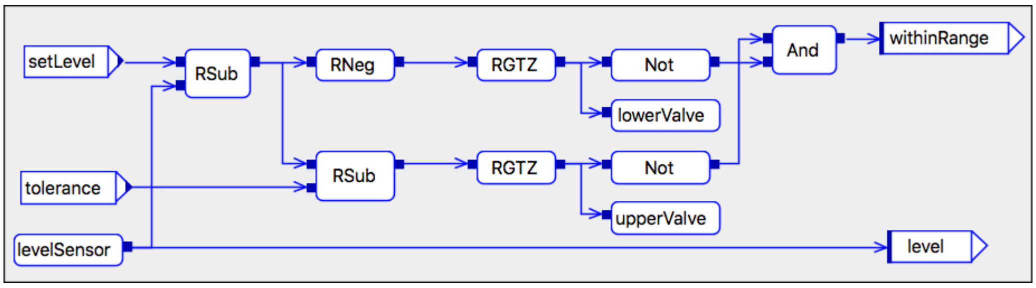


Figure 11. Simple tank regulator in Bloqqi. The block `levelSensor` reads the current liquid level. The blocks `lowerValve` and `upperValve` are actuators to the valves. The block `RGTZ` means greater than zero (for reals).

agram also has a parameter `tolerance` that allows the level to be within a range of the set point (`setLevel` $\pm$ `tolerance`). The output parameter `withinRange` yields true if this is the case. There is also another diagram `Main`, which is not shown in the figure, that has the `Tank` diagram as a block. The `Main` diagram will use the output parameter `withinRange` to change the set level.

The tank model in Modelica is specified as a *mass balance equation* (Fritzson, 2004), where the curent liquid level is specified in terms of the input flow and output flow of the tank. The equations for the tank are specified in the following manner:

```
der(level) = (inFlow-outFlow)/AREA;
inFlow = if upperValveOpen
  then IN_FLOW
  else 0.0;
outFlow = if lowerValveOpen
  then (OUT_VALVE_AREA)*sqrt(2*9.82*level)
  else 0.0;
```

Thus, the time derivative of the current liquid level is defined in terms of the difference in flow. The input flow depends on if the input valve is open and the output flow depends on if the output valve is open. The output flow is also dependent on the pressure, thus, on the current liquid level. This can be seen in the simulation results in Figure 10, where the emptying of the tank from 1.8 m to 0.2 m is not a straight line.

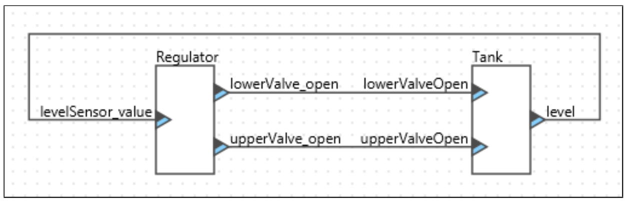


Figure 12. FMU composition of the tank regulator and tank model specified in FMI Composer.

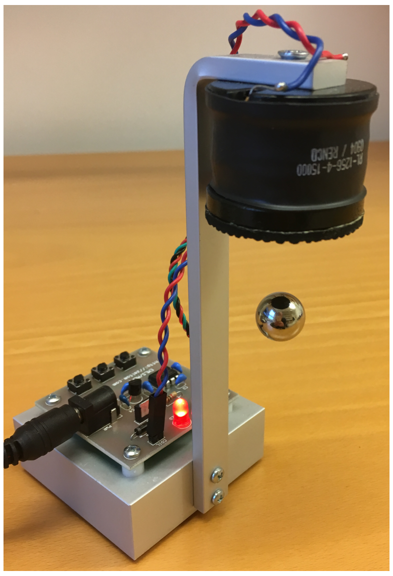


Figure 13. Magnetic levitation system.

6.2 Composing FMUs

We have used FMI Composer[4] from Modelon for co-simulation of the Bloqqi FMU of the controller and the Modelica FMU of the tank. FMI Composer supports the SSP standard and can convert a system of FMUs to a aggregated FMU. The simulation of the composed FMU is then run in another tool, for instance, the FMI Toolbox for Simulink, also from Modelon, that we have used for the tank simulation shown in Figure 10.

For example, the composition of the tank regulator and the tank model is shown in Figure 12. The regulator FMU has one input, the current liquid level, and two outputs, which tells if the valves should be open or not. The tank FMU uses the output from the regulator to model the current liquid level, which is the output from the tank FMU.

6.3 Magnetic Levitation

Another example we have experimented with is a magnetic levitation system. The system uses an electromagnet to control a magnet in the air. This is shown in Figure 13, which is an education system from Zeltom[5].

We have run a PD-regulator defined in Bloqqi for controlling a simluation of the electromagnet, where the control signal is the voltage to the electromagnet and the set

[4]http://www.modelon.com/products/

[5]http://zeltom.com/products/magneticlevitation

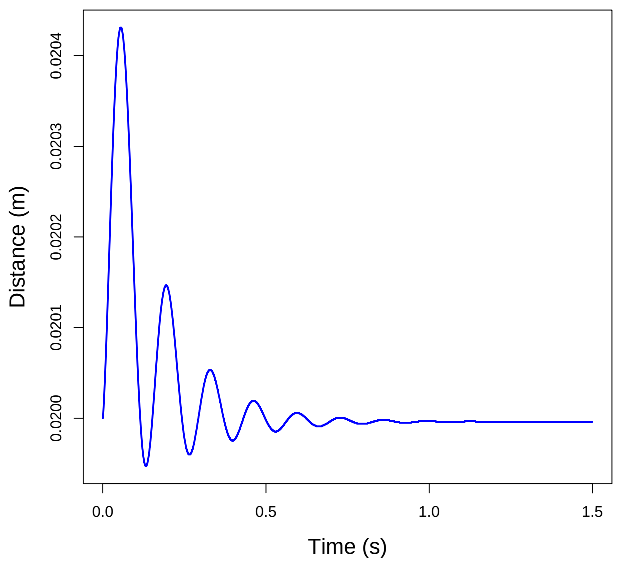


Figure 14. Simulation of magnetic levitation. The set point is 0.02 m distance between the electromagnet and the magnet.

point is the distance between the electromagnet and the magnet. The regulator has been simulated together with a Modelica model that describes the distance between the electromagnet and the magnet, given a voltage to the electromagnet. The Modelica model is written by Bernhard Thiele. Thiele has also specified a PD-regulator in Modelica for the system, which is the basis for our PD-regulator in Bloqqi. A simulation of the regulator and the model can be seen in Figure 14 (with the set point of 0.02 m and sampling period of 0.0005 s).

7 Related Work

Typically, code generation for block diagrams removes the hierarchical structure by flattening the diagrams, where each non-atomic block is replaced with its definition. Lublinerman et al. (2009) propose a technique for modular code generation, where the code generation for one diagram is independent of the context and uses minimal information about the internals of its blocks. Their technique handles diagrams that look cyclic, but when the diagrams are flattened, they are in fact acyclic. The Bloqqi compiler will not allow these kinds of diagrams, as described in Section 3.1.1.

The implementation of code generation for Bloqqi has similarities with the implementation of JModelica (Åkesson et al., 2010). Both implementations are specified using reference attribute grammars (Hedin, 2000) in the metacompilation tool JastAdd (Ekman and Hedin, 2007). In JModelica, the user selects a Modelica model to compile, and JModelica compiler will then remove the object-oriented and hierarchical structure for the selected model that results in a flat equation system. During this process, each model instance reachable from the selected model is substituted with the content of the model definition. In contrast, the Bloqqi compiler will only flatten the inheritance, by copying declarations in the supertype to the subtype, which results in a set of diagrams as described in Section 4.1. Thus, the Bloqqi compiler will retain the the hierarchy between diagrams. Both implementations make extensive use of *non-terminal attributes* (Vogt et al., 1989), which allows dynamically computed subtrees to be added to the abstract syntax tree in the compiler.

The problems, as discussed in 5.3, with representing discrete time systems, or any system including events, using the Co-Simulation FMI technology with regards to matching the master algorithm's sampling with the internal events of the FMUs is discussed in Cremona et al. (2017). In Cremona et al. (2017), a suggestion to add new step function called "doStepHybrid" which allows for an early return of the step function. This in combination with a possibility for the master to interrogate the system using functions like `getMaxStepSizeHybrid` would allow for the Bloqqi FMU to assure that each sample and execution event is properly matched. Additionally the concept of clocks and hybrid Co-Simulation is included in the alpha feature list of FMI 2.1 that was announced 2017-12-18 (fmi standard, 2017), this to support synchronization of variable changes across FMUs, and allow for co-simulation with events. Cremona et al. (2017) also proposes an integer representation of time in order to increase the accuracy of the time representation in simulation of FMUs.

Additional problems related to synchronization of discrete-time models is presented in Franke et al. (2017). Where a synchronus discrete-time extension to the FMI-standard is proposed to enable synchronization between FMUs with the environment, and other FMUs. The proposal includes a possibility to declare clocks and discrete-time states in the `modelDescription.xml`, and enables the environment to activate clocks in order to synchronize the environment with other FMUs.

TrueTime (Cervin et al., 2003) is a tool for simulating control systems, where it is possible to simulate task scheduling and network transmissions. These aspects are not covered in the Bloqqi FMUs, but it would be possible, for example, to model network delays as other FMUs connected to the Bloqqi FMUs.

8 Conclusions

We have in this paper described how Bloqqi programs can be exported as Co-Simulation FMUs. This is done by translating the programs to C code, which are then wrapped as FMUs. In this translation, each Bloqqi diagram is translated to a C function and each block (an instance of diagram) is translated to a function call. Before the translation, the inheritance structure is removed (but the hierarchical structure is retained). The paper also describes how the function `DoStep` is implemented for Bloqqi FMUs to handle that Bloqqi programs execute at discrete time, according to the sampling period and the estimated execution time.

Exporting Bloqqi programs as FMUs allow them to be easily imported to a simulation environment to be tested together with a simulation of the physical process. The physical process can be specified with any tool that supports FMI export/import. We have successfully tested running Bloqqi programs as FMUs together with FMUs exported from Modelica models.

In the future, we would like to support more of the FMI standard, for example, the possibility to make rollback, which is useful for simulating a larger class of FMU compositions (Broman et al., 2013). It would also be useful to generate Model-Exchange FMUs, and not only Co-Simulation FMUs. This would allow the Bloqqi FMU to tell the master algorithm about the discrete times at when it will start and complete its execution. We would also like to experiment with more and larger examples.

Acknowledgements

This research was supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA), within the strategic innovation program Process Industrial IT and Automation, under contract number (2017-02371). We thank Bernhard Thiele for the magnetic levitation example, and Görel Hedin for comments on earlier draft of this paper, and Ulf Hagberg, Christina Persson, Stefan Sällberg and Alfred Theorin at ABB for sharing their expertise about the ABB tools for control systems.

References

Johan Åkesson, Torbjörn Ekman, and Görel Hedin. Implementation of a Modelica compiler using JastAdd attribute grammars. *Science of Computer Programming*, 75(1-2):21–38, 2010.

Torsten Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *9th International Modelica Conference*, 2012.

David Broman, Christopher X. Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of fmus for co-simulation. In *Proceedings of the International Conference on Embedded Software, EMSOFT 2013*, pages 2:1–2:12, 2013.

Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine*, 23(3):16–30, June 2003.

Fabio Cremona, Marten Lohstroh, David Broman, Stavros Tripakis, and Edward A. Lee. Hybrid Co-simulation: It's About Time. Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley, April 2017.

Torbjörn Ekman and Görel Hedin. The JastAdd system - modular extensible compiler construction. *Science of Computer Programming*, 69(1-3):14–26, 2007.

fmi standard. Functional mock-up interface, 2017. URL `http://fmi-standard.org/`.

Niklas Fors. *The Design and Implementation of Bloqqi - A Feature-Based Diagram Programming Language*. PhD thesis, Lund University, October 2016.

Niklas Fors and Görel Hedin. Intercepting dataflow connections in diagrams with inheritance. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 21–24, 2014. doi:10.1109/VLHCC.2014.6883016.

Niklas Fors and Görel Hedin. Bloqqi: modular feature-based block diagram programming. In *2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward! 2016, Amsterdam, The Netherlands, November 2-4, 2016*, pages 57–73, 2016. doi:10.1145/2986012.2986026.

Rüdiger Franke, Sven Erik Mattson, Martin Otter, Karl Wernersson, Hans Olsson, Lennart Ochel, and Torsten Blochwitz. Discrete-time models for control applications with fmi. In *12th International Modelica Conference*, 2017.

P.A. Fritzson. *Principles of object-oriented modeling and simulation with Modelica 2.1.* Wiley-IEEE Press, 2004.

Görel Hedin. Reference Attributed Grammars. In *Informatica (Slovenia)*, 24(3), pages 301–317, 2000.

Jochen Köhler, Hans-Martin Heinkel, Pierre Mai, Jürgen Krasser, Markus Deppe, and Mikio Nagasawa. Modelica-association-project "system structure and parameterization" – early insights. In *1st Japanese Modelica Conference*, 2016.

Roberto Lublinerman, Christian Szegedy, and Stavros Tripakis. Modular code generation from synchronous block diagrams: modularity vs. code size. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 78–89, 2009.

Modelica. *The Modelica Association*, 2018. `http://www.modelica.org`.

Harald Vogt, S. Doaitse Swierstra, and Matthijs F. Kuiper. Higher-order attribute grammars. In *PLDI*, pages 131–145, 1989.

A Web Architecture for Modeling and Simulation

Hilding Elmqvist Martin Malmheden Johan Andreasson
{Hilding.Elmqvist, Martin.Malmheden, Johan.Andreasson}@Modelon.com
Modelon AB, Sweden, www.Modelon.com

Abstract

A Web Architecture for Modeling and Simulation (WAMS) is presented which enables system modeling in your browser using the Modelica language. Compilation and simulations are done on a server using the Optimica Compiler Toolkit (OCT) from Modelon.

Such an architecture is appropriate for making design space explorations such as sensitivity analysis, DOE, Monte Carlo analysis, optimizations, parameter estimation, etc. efficiently.

Keywords: Web app, Cloud Simulations, Model Based Product Design, Modelica, Modeling, Simulation, Optimization

1 Introduction

Model based product design requires both intuitive and effective user interfaces and large computing power. Fortunately, modern computer systems provide a solution with client software, web apps, running in a web browser and use of cloud computing and cloud storage for simulations and storing models and results. HTML5 and WebGL provides an appropriate basis for web app development. When making design space explorations, multi-core, clusters and cloud computing provide the means for performing multi-simulations such as sensitivity analysis, DOE, Monte Carlo analysis, optimizations, parameter estimation, etc. efficiently.

Such client-server architecture and use of web apps also opens up for performing modeling, simulations and optimizations from a tablet or a smart phone as well as from your computer.

Figure 1. Simulation on the road

A future oriented use case utilizing such a solution is shown in Figure 1 (Berggren, 2017):

- Your self-driving car is taking you home after work.
- You get an idea for how to solve today's frustrating problem.
- You just need to:
  - insert one model component
  - change some parameters
  - simulate
  - look at some plots to verify
- You are eager to test it immediately on your smart phone.

This paper describes an architecture for modeling in a web app using the Modelica language and performing simulation on the cloud.

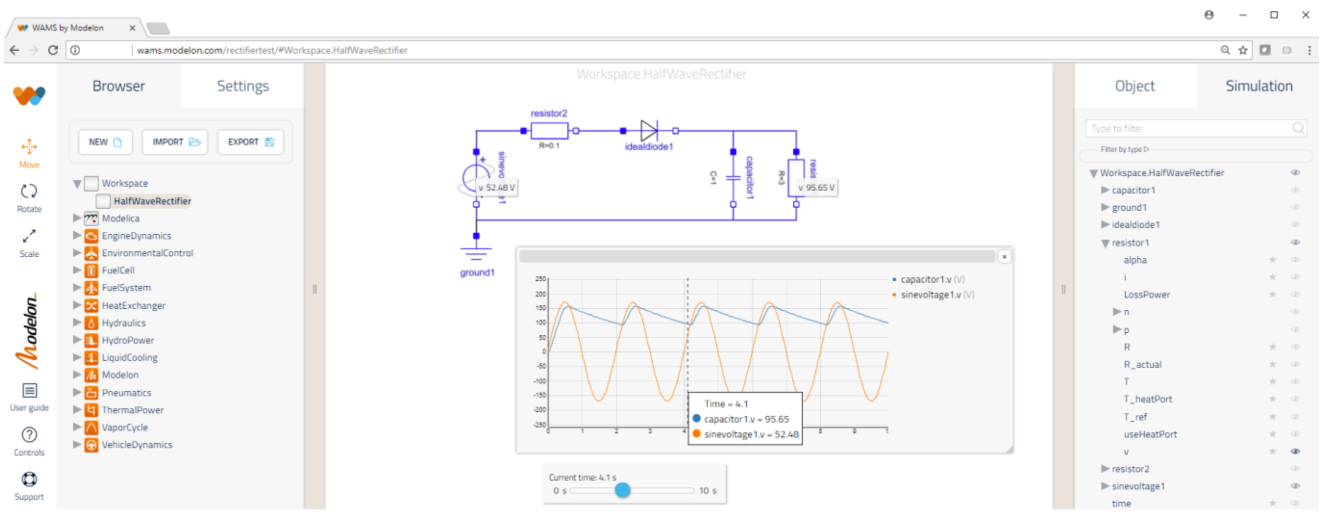


Figure 2. Web app for modeling and simulation

2 Modelica Systems Modeling

The layout of the web app is shown in Figure 2. The left pane contains a package browser for model components described in the Modelica language (Modelica Association, 2014).

The middle part contains the diagram of a system model, i.e. a model consisting of connected model components as well as variable plots.

The right part contains a variable browser for setting parameters, selecting variables to plot, etc. There are elaborate features for searching, for filtering and to declare favorite variables.

The web app has features to create and modify models, such as:

- Drag components from the model browser to the diagram
- Set parameters of a component
- Connect connectors of components
- Multiple select
- Resize, rotate component
- Copy, Paste and Duplicate
- Undo, redo
- Display of web app and model documentation
- Open hierarchical sub-levels

3 Multi-Simulation

A Modelica model is built up on the server while the model is being edited. When a simulation is requested, this Modelica model is compiled using OCT (Modelon, 2018c) into an FMU (Blochwitz, et al., 2012) which is used for the simulation.

The variable browser in the right pane of the web app contains the model hierarchy tree with all variables. A plot of a variable is obtained by dragging a variable into the diagram. If a variable is dragged into an already present plot, the variables are plotted together as shown in Figure 2.

It is also possible to show instantaneous values of variables close to the components using a concept called a *sticky*, see Figure 2. Such stickies are created by selecting a tool close to the variable names in the browser.

Parameter values can be changed in the variable browser and new simulations performed without recompiling the model.

It is possible to specify ranges of values for parameters to perform design space exploration. Multi-simulations are then automatically performed using a specified sampling technique. It is also possible to specify Monte Carlo simulations, optimizations and parameter estimation.

Figure 3 shows how uncertainties in behavior are shown as a "spaghetti" plot when parameters are specified as ranges or with random distributions. In this case, 10 simulations of a robot were made and the tool positions plotted.

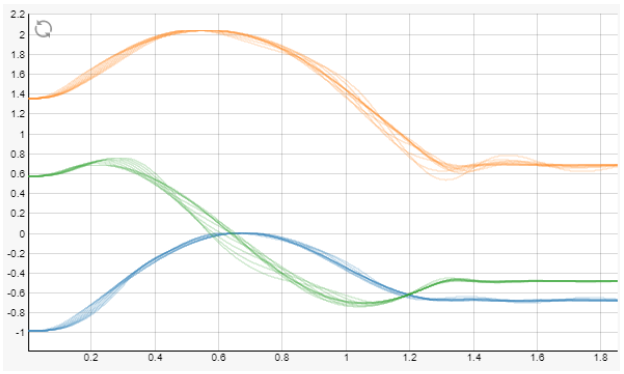


Figure 3. Uncertainties in robot motion for loads 0-300 kg.

3D plots are also available to present variations which depend on two parameter ranges. Figure 4 shows how the camber angle of a wheel suspension depend on wheel travel and steering angle.

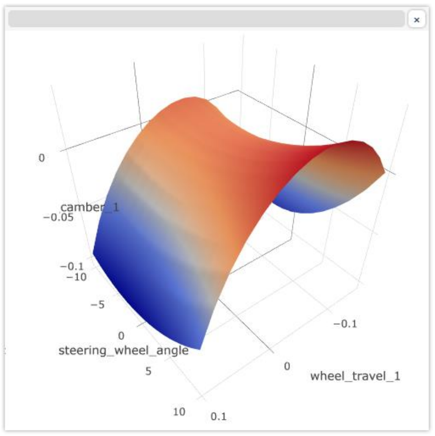


Figure 4. Camber angle versus steering wheel angle and wheel travel

3D animation is provided for models based on the MultiBody library of MSL. Figure 5 shows an animation from the Vehicle Dynamic Library (Modelon, 2018b).

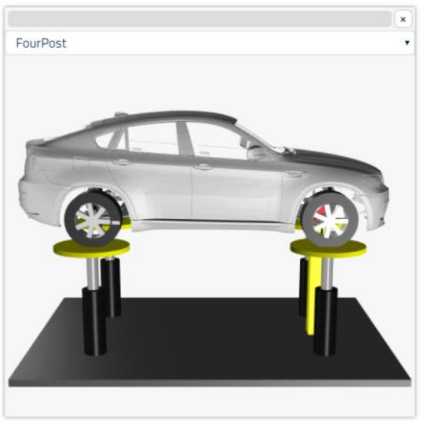


Figure 5. 3D animation of VDL model

4 Application Example: Electric Powertrain

Figure 6 shows a full vehicle thermal analysis of a battery electric vehicle with a small internal combustion engine (ICE) serving as a range extender. The vehicle has active thermal management of the electric powertrain and cooling of the ICE.

Figure 6 shows the electric powertrain, with battery and two electric machines where one serves as the traction motor and the other as generator connected to the ICE. Battery and machines are all connected to the high voltage bus. Additionally, all components have thermal connectors that allow them to interact with the cooling system. Internally, as seen in Figure 7, the machine has thermal dynamics described individually by motor and power electronics. Correspondingly, the battery thermal dynamics can be resolved at cell level. Further information about the models can be found in Modelon (2018d), Batteh (2018).

Here, the model is configured to study heat propagation as response to a standard drive cycle. The diagram is configured with plots to see the dynamic temperature change over time. Note that as a user, to generate the plot it is enough to know that you are looking for temperatures and then select for what components you want to see the values.

To get an overview of how the heat is propagated through the system, stickies are activated for the thermal components, where the display follows the time slider.

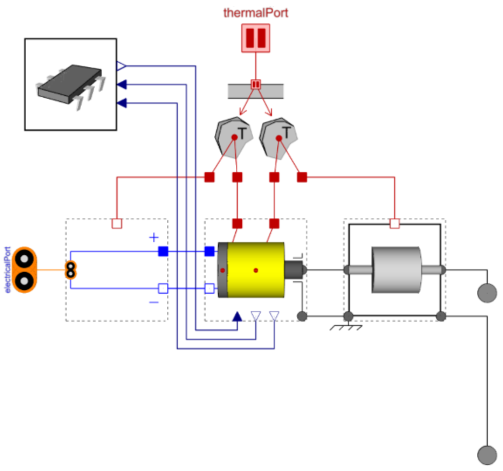
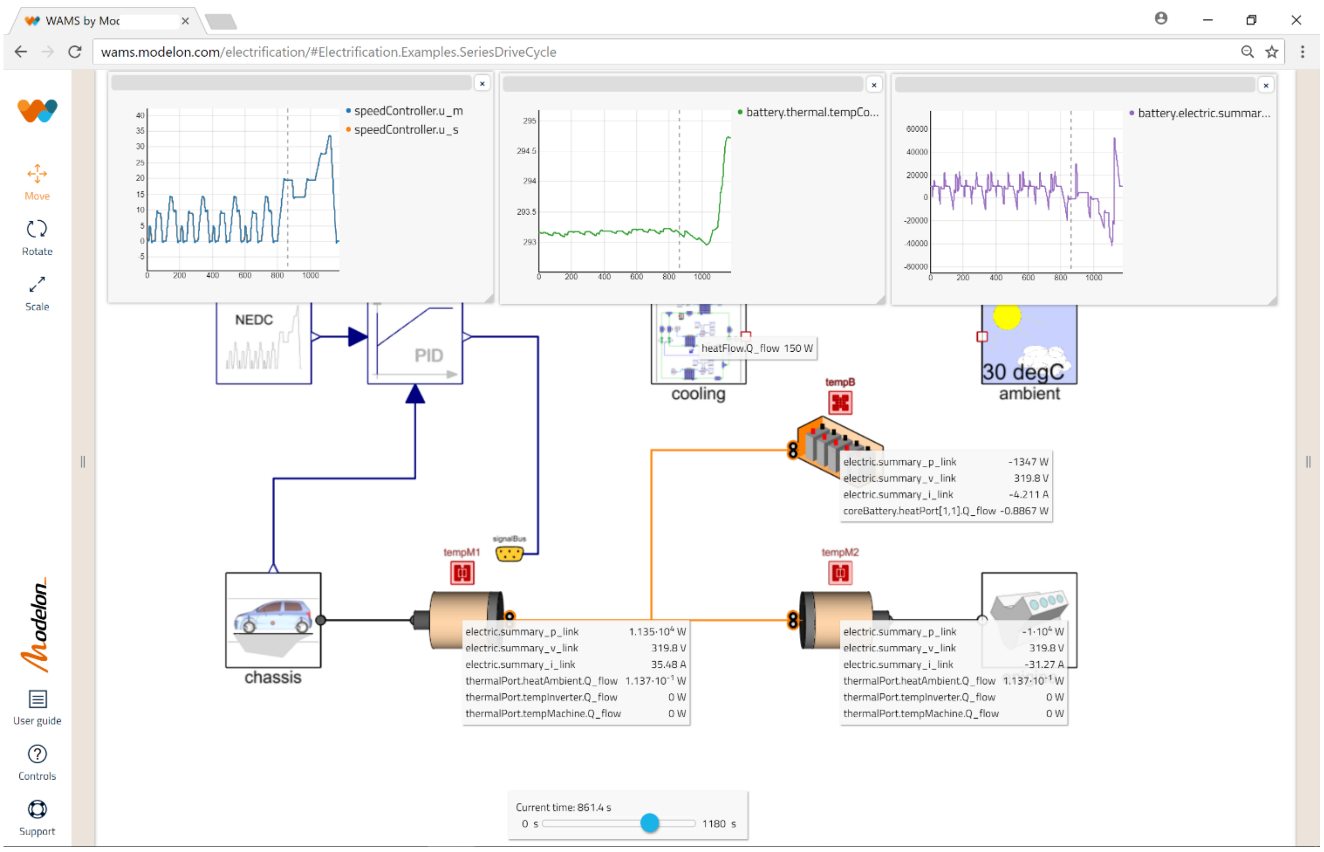


Figure 7. Electric machine.



electric.summary_p_link -1347 W
electric.summary_v_link 319.8 V
electric.summary_i_link -4.211 A
coreBattery.heatPort[1,1].Q_flow -0.8867 W

electric.summary_p_link $1.135 \cdot 10^4$ W
electric.summary_v_link 319.8 V
electric.summary_i_link 35.48 A
thermalPort.heatAmbient.Q_flow $1.137 \cdot 10^{-1}$ W
thermalPort.tempInverter.Q_flow 0 W
thermalPort.tempMachine.Q_flow 0 W

electric.summary_p_link $-1 \cdot 10^4$ W
electric.summary_v_link 319.8 V
electric.summary_i_link -31.27 A
thermalPort.heatAmbient.Q_flow $1.137 \cdot 10^{-1}$ W
thermalPort.tempInverter.Q_flow 0 W
thermalPort.tempMachine.Q_flow 0 W

Current time: 861.4 s
0 s 1180 s

Figure 6. Series hybrid from the Electrification Library.

5 Application Example: Suspension design

Figure 8 illustrates how a model can conveniently be set for multiple executions, in this case for suspension design. Here, a suspension is mounted in a test rig from the Vehicle Dynamics Library (Modelon, 2018a), for evaluation of its kinematic behavior depending on steering and wheel travel.

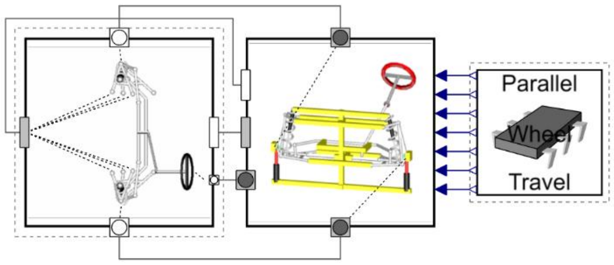


Figure 8. Wheel suspension mounted in test rig.

The analysis is relevant from an engineering perspective to characterize the main behavior of a given suspension design, and is frequently used in suspension design.

The experiment is set-up by defining ranges of the two independent variables and then the resulting executions are automatically triggered and managed. Such an execution can be either dynamic or steady-state.

As can be seen in the plots, Figure 4, the corresponding illustration is in this case a response surface over steering (x) and wheel travel (y). The response variable (z), in this case wheel hub rotations camber, could be either a value at a given time from a dynamic simulation, or the result from a steady-state calculation. In this case, 15 x 15 dynamic simulations were made, and the final steady-state solutions were plotted. The simulations were made on 20 Microsoft Azure cloud nodes.

6 Application Example: Distribution Network

Figure 9 shows a heat distribution network from the Liquid Cooling Library (Modelon, 2018b) that can be used to tune flow resistances to reach a passive distribution of the heat in the network.

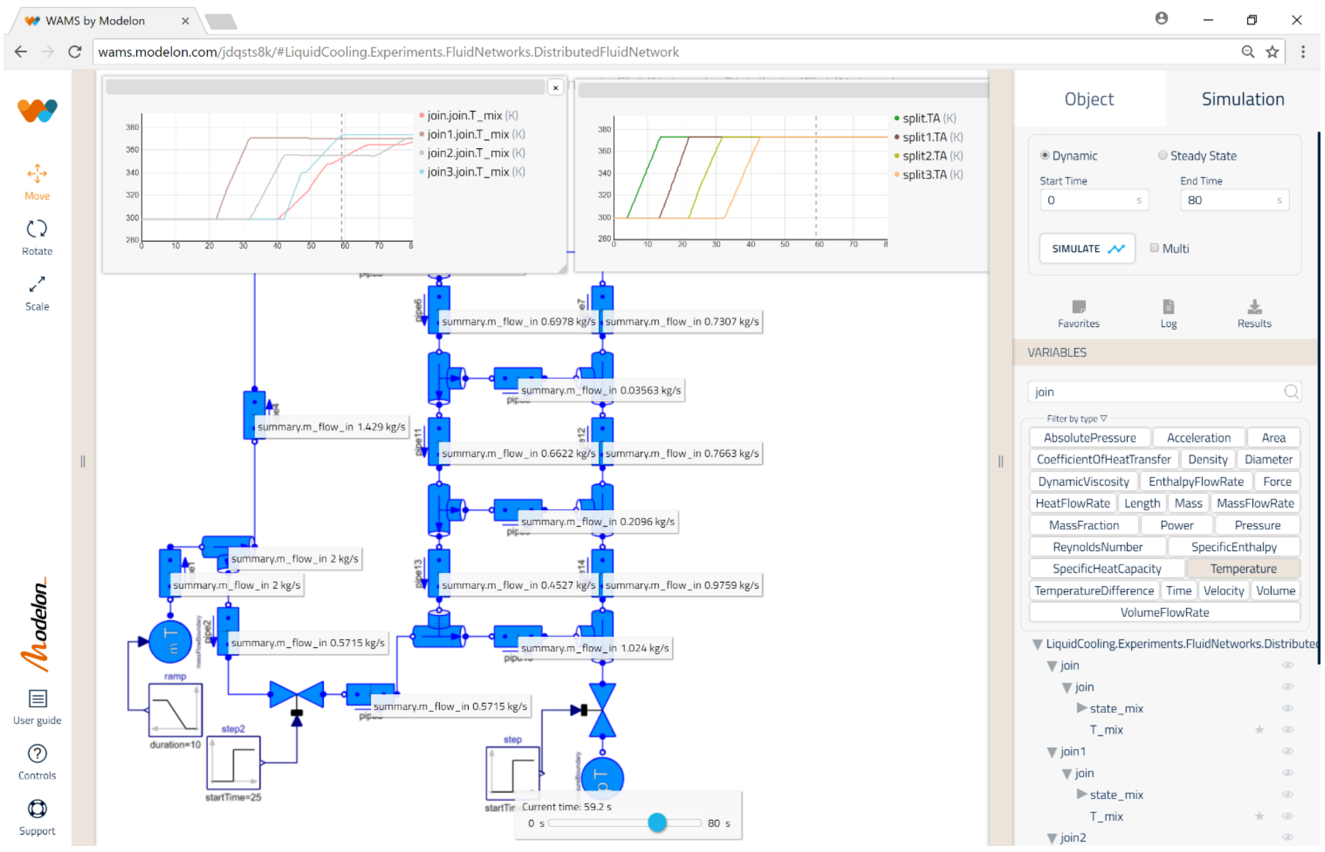


Figure 9. Heat distribution network from Liquid Cooling Library.

Here it is configured to study heat propagation as response to a ramp change of the input flow to the network. The diagram is configured with a plot to see the dynamic temperature change over time.

To get an overview of how the mass flow is propagated through the system, stickies are activated for each pipe, where the display follows the time slider.

7 Architecture

The software architecture is shown in Figure 10. The web app is written in TypeScript and utilizes the React framework and the three.js 3D package. It communicates with the server using a REST API. The Modeling API has functions for creating, deleting and changing models, instances, connections and variables as well as compiling a Modelica model into an FMU. The simulation API can change parameters, perform multi-simulations and access simulation results.

The server uses the Optimica Compiler Toolkit (OCT) for maintaining the abstract syntax tree of a model being built up in the web app as well as performing the compilation. The simulation API is implemented in Python and uses the PyFMI API (Andersson, et al., 2016). The server hosts the Modelica model repository and the data repository.

The modeling and simulation platform also utilizes Jupyter/JupyterLab notebooks for interactive exploration of models by invoking simulations in the server. JupyterLab can also be used for editing of Modelica code for component authoring. Python scripts can be uploaded to the server to extend the simulation API.

Customized user experiences can be implemented as web apps using the server API functionality and uploaded Python scrips.

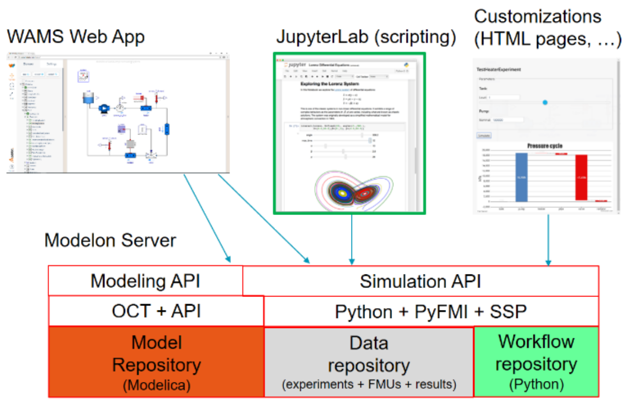

WAMS Web App
JupyterLab (scripting)
Customizations (HTML pages, …)
Modelon Server
Modeling API
Simulation API
OCT + API
Python + PyFMI + SSP
Model Repository (Modelica)
Data repository (experiments + FMUs + results)
Workflow repository (Python)

Figure 10. Software architecture

8 Related Work

WebMWorks (Qi, et al., 2012) is a web-based modeling and simulation environment for Modelica. The modeling capabilities are similar to the desktop application MWork Studio. WebGME (Web Generic Modeling Environment) (Maróti, et al., 2014) is a web-based cyberinfrastructure to support the collaborative modeling, analysis, and synthesis of complex, large-scale information systems. It has support for the Modelica language. The Playmola web app (Elmqvist, et al., 2015) is an effort to utilize the power of HTML5 and three.js for Modelica modeling and simulation in a VR environment using Google Cardboard.

Tiller (2013) discusses running simulations on the cloud for sensitivity analysis. Cloud based simulation for FMUs were introduced in Johansson (2015) to speed up simulations initiated by a plugin to Excel. Different parameter cases were defined in columns of the Excel sheet. Bittner, et al. (2015) describes cloud deployment of FMU simulations which are defined in a web app as either enumerated sets of values or ranges of values associated with parameters. All combinations are simulated.

Web apps have also been introduced for CAD, for example TinkerCAD which is one easy to use web app and OnShape and Clara which are professional tools (Ishtiaque, 2017).

9 Conclusions

The paper demonstrates the advantages of using web apps and cloud computing for model-based product design. The presented web app is used for systems modeling and to invoke simulation experiments. The simulations are done on the server which might be installed locally, on premise or on the cloud. Simulations can be distributed on the cloud for an improved user experience avoiding waiting long times for simulation results.

Acknowledgements

The authors want to thank the development teams at Modelon for the excellent implementation effort.

References

C. Andersson, J. Åkesson, and C. Führer: PyFMI (2016): A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface. https://lup.lub.lu.se/search/publication/961a50eb-e4a8-43bc-80ac-d467eef26193

J. Batteh (2018): Development and Implementation of the NASA X-57 Electric Aircraft with Aircraft Dynamics Library, Submitted for publication.

S. Berggren (2017): Web Application for Modeling and Simulation, Malmö University. http://muep.mau.se/handle/2043/23515

T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, A. Viel (2012): Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models, 9th International Modelica Conference, Munich, 2012. http://www.ep.liu.se/ecp/076/017/ecp12076017.pdf

S. Bittner, O. Oelsner, T. Neidhold (2015): Using FMI in a cloud-based Web Application for System Simulation. Proceedings of the 11th International Modelica Conference September 21-23, 2015, Versailles, France. https://www.modelica.org/events/modelica2015/proceedin gs/html/submissions/ecp15118845_BittnerOelsnerNeidhold.pdf

H. Elmqvist, A.D. Baldwin, S. Dahlberg (2015): 3D Schematics of Modelica Models and Gamification. Proceedings 11th International Modelica Conference, Versailles, September 21-23, 2015. http://www.ep.liu.se/ecp/118/057/ecp15118527.pdf

J. Ishtiaque (2017): Top 5 Online CAD Tools for 3D Modeling, https://www.linkedin.com/pulse/top-5-cloud-based-online-cad-tools-3d-modeling-jabid-ishtiaque/

D. Johansson (2015): Implementing a cloud-based scalable dynamic model simulator, Chalmers University of Technology. Department of Computer Science and Engineering. Gothenburg, Sweden, January 2015. http://publications.lib.chalmers.se/records/fulltext/212552/212552.pdf

M. Maróti, T. Kecskes, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendoszky, A. Ledeczi. (2014). Next generation (Meta)modeling: Web- and cloud-based collaborative tool infrastructure. CEUR Workshop Proceedings. 1237. 41-60. https://webgme.org/WebGMEWhitePaper.pdf

Modelica Association (2014): The Modelica Language Specification, Version 3.3 Revision 1, https://www.modelica.org/documents/ModelicaSpec33Revision1.pdf

Modelon (2018a): Vehicle Dynamics Library: http://www.modelon.com/products/modelon-library-suite/vehicle-dynamics-library/

Modelon (2018b): Liquid Cooling Library: http://www.modelon.com/products/modelon-library-suite/liquid-cooling-library

Modelon (2018c): OPTIMICA Compiler Tool Kit. http://www.modelon.com/products/modelon-creator-suite/optimica-compiler-toolkit/

Modelon (2018d): Liquid Cooling Library: http://www.modelon.com/products/modelon-library-suite/electrification-library

L. Qi, X. Tifan, L. Qinghua, C. Liping (2012): WebMWorks: A General Web-Based Modeling and Simulation Environment for Modelica. Proceedings of the 9th International Modelica Conference, September 3-5, 2012, Munich, Germany. https://pdfs.semanticscholar.org/30a6/c036da254693d416b754867ae0243a8baf23.pdf

M. Tiller (2013): The FMQ Web Platform. Detroit FMI Technology Day, November 6, 2013. https://www.youtube.com/watch?v=5wYvQmqCvBo

Multibody simulation and control of kinematic systems with FMI/FMU

Francois Chapuis[1], Jean Daniel Beley[2], Stephane Garreau[3], Olivier Roll[4], Tim Puls[5], Leon Voss[6], Sameer Kher[7]

[1]ANSYS Inc., France, francois.chapuis@ansys.com
[2]ANSYS Inc., France, jean-daniel.beley@ansys.com
[3]ANSYS Inc., France, stephane.garreau@ansys.com
[4]ANSYS Inc., France, olivier.roll@ansys.com
[5]ANSYS Inc., Germany, tim.puls@ansys.com
[6]ANSYS Inc., Germany, leon.voss@ansys.com
[7]ANSYS Inc., USA, sameer.kher@ansys.com

Abstract

Our connected world and environment make us interact every day with very complex devices. Driving our cars, monitoring health using smart phones, the extensive use of robots are all applications which involve large quantities of embedded functions and physics. To design and simulate efficiently such systems, individual physics simulators are not sufficient and coupled simulations are required. A new standard called FMI (Functional Mock-Up Interface) [1] has been created, allowing to federate these interactions between a wide variety of physical, digital and reduced models, either through a co-simulation approach or through model exchange strategy using a standardized and neutral interfacing mechanism. In this article, we illustrate through an example how it's possible to simulate mechanical assemblies, kinematics, dynamics and control systems in the same system model. Each mechanical sub-assembly is represented by a FMU (Functional Mock-Up Unit) exported from a multibody dynamics solver and includes a mix of rigid and flexible components. Flexible components are reduced order models of the full fidelity finite element model using the well-known CMS (Component Mode Synthesis) method. [2]

We apply the coupling through the FMI standard to a robot model, composed of rigid parts and one flexible sub-assembly. The highly non-linear behavior of the equations of motion of the multibody assembly is captured and consumed as a co-simulation FMU. The actuators detailed model – from the voltage source to the electric motors – are modeled in the system simulation platform ANSYS Twin Builder, while the control loops use SCADE which offers different control laws. The co-simulation of these 3 sub-systems can then be performed in an efficient manner, without the prerequisite of having on-off coupling developed between each of the individual simulators.

Keywords: FMU, multibody dynamics, CMS, actuator, control

1 Introduction

As a prerequisite to a system simulation, it is commonly known that one must build, embed and link all relevant components in a simulation platform. Some of the standard base components are directly available in the Modelica libraries and some other higher fidelity models may also be reduced through ROMs (for instance LTI) or using the FMI standard for model exchange. When the sub-model transient behavior is highly non-linear, no reduction of the model is possible and a co-simulation approach is needed. In that case, a co-simulation FMU is exported, which allows full transient simulation on the fly along with the global simulation. In the shown example, all ROMs and FMUs have been generated from ANSYS Mechanical owing to the finite element and multibody solvers and brought to the systems environment for the whole system simulation.

2 Generation of the FMUs and the ROM

The application example is shown in figure 1. It is a 6-axis robot, each of the axis being modeled by a revolute joint on which is added a controlled actuator.

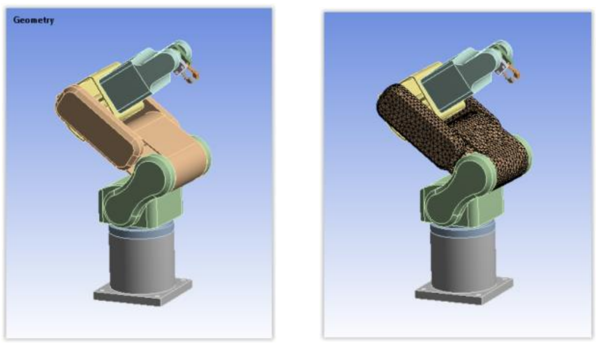


Figure 1 : robot arm geometry and mesh (full rigid on the left, rigid/flexible on the right)

To illustrate multi-level reduced order modeling, one of the arms is considered as deformable. A full fidelity finite element model is built. These 100,000 degrees of freedom model is then reduced to a few dozens of DOFs using a Component Mode Synthesis (CMS) method, accounting for the elastic behavior as well as the coupling between small elastic displacements and large rotations and displacements. A modal analysis of the deformable part, followed by a series of static analysis are performed to generate the CMS reduced order model. During the preprocessing steps, the interface pins are defined. For each robot axis, angle and torque are defined as pins. Finally, we export a co-simulation FMU component (shown in figure 2) with its interface pins. The reduced order model of the flexible part is included in this FMU and will be consumed by the multibody solver during the simulation.

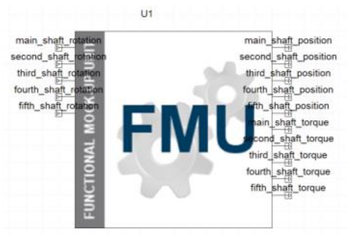


Figure 2 : co-simulation FMU component

It is possible to combine several kinematic models designed separately in the same system assembly, each kinematic model may be converted as a FMU and link to the other in ANSYS Twin Builder.

For the electric motor device, FMUs may also be generated following the same workflow, including also flexible parts such as the housing. It is worth mentioning that when the flexible parts are not undergoing large displacement and rotations, a mechanical State Space model coming from ANSYS finite element solver may also be integrated in this workflow as a model exchange FMU.

3 System workflows

The robot model uses 5 motors (one per axis), powered by a circuit mixing voltage sources, MOSFETs and diodes, as shown in figure 3. Each motor provides a rotation angle and get a resistive torque from the FMU pins.

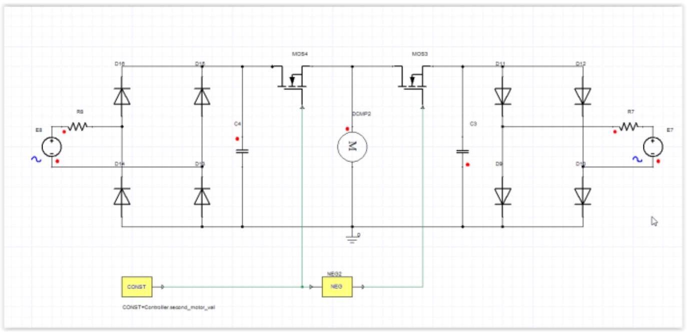


Figure 3 : electric power circuit supplying each robot joint

To achieve good performance, the kinematic function of the robot, a controller has been defined with SCADE suite and linked to this workflow (figure 4). A target signal has been provided to all electric motors and the system.

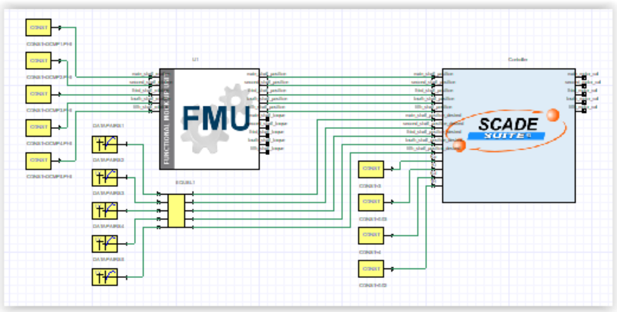


Figure 4 : control of the robot model

Regarding the electric motors, instead of the idealized system blocks (used in figure 3), we may also consider a complex mechanical system involving different FMUs : a rotor with some magnets spinning in a mechanical system will induce some forces/torques on the other mechanical parts, especially on the stator housing which also entails vibrations and unexpected behaviors on the robot arm. Reciprocally, the vibration of the electric motor may be affected by the robot arm connected to this motor and this is also addressed using the co-simulation approach with FMUs. Figure 5 shows the typical combination of FMUs used for the electric motor.

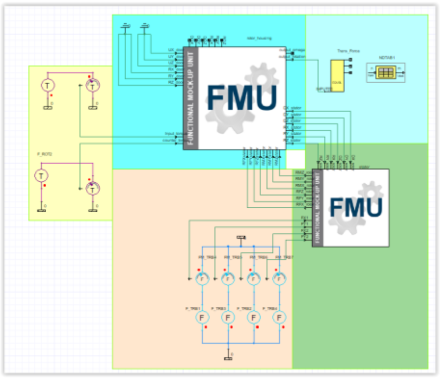


Figure 5 : definition of the workflow for the electric motor

4 Results

Figure 6 shows the results at the end of the co-simulation and it emphasizes the behavior of each axis regarding the target signal that the robot arm needs to follow. Here, the controller provides more efficiency for the fourth than for the third axis, especially in terms of response time and smoothness. This may be improved with the use of other components and a smarter controller.

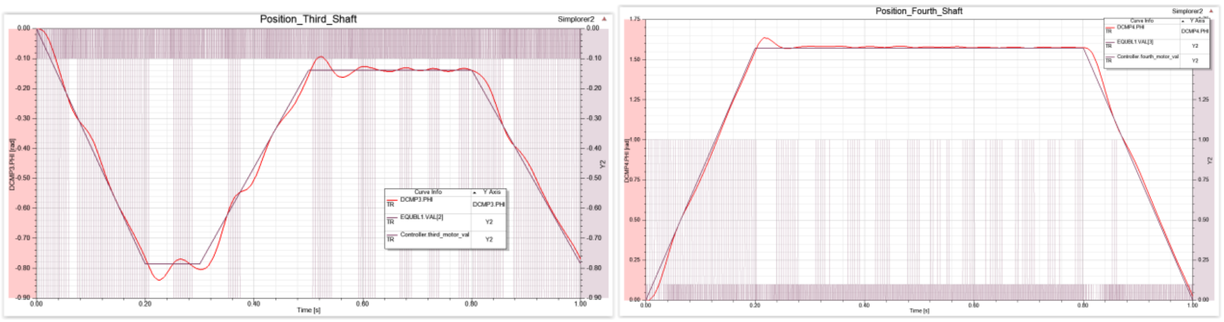


Figure 6 : position results for two different joints of the robot in function of time (s)

The figure 7 illustrates the difference on the third shaft when we compare the current robot (with one flexible arm) and the same robot with all arms considered as rigid parts. Because of the flexible behavior of arm, the time response of the structure is slightly different, then the controller does not behave exactly in the same way for both cases and we see a small shift between each couple of curves.

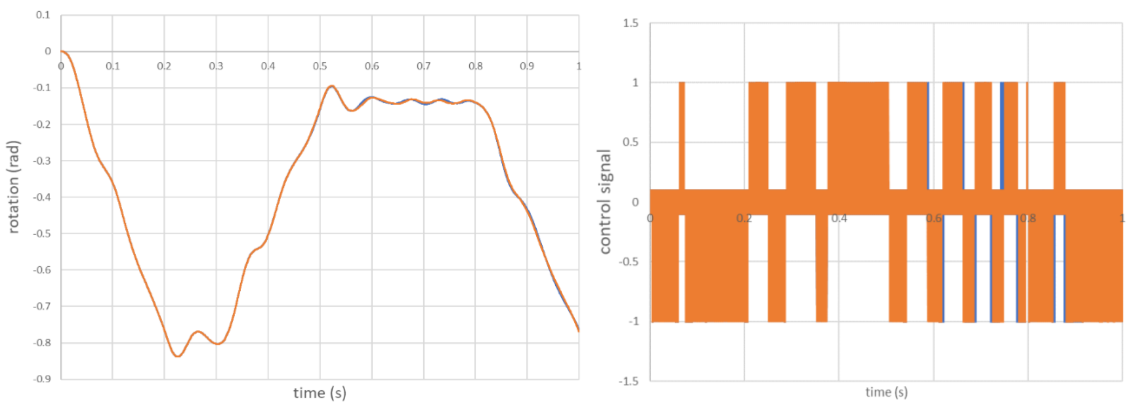


Figure 7: comparison of the position results and the control signal for the full rigid model (orange) and the mixed rigid/flexible (blue) in function of time (s)

5 Conclusion: Limitations and Extensions

The FMI standard allows linking together multiple physics without one-off development of coupling interfaces between each of the participant of complex system simulations, thanks to well defined API. Models and Physics that can be easily "reduced" can use the model exchange standard, while highly non-linear system is more subject to use co-simulation FMUs.

While the standard is facilitating exchanges, it also comes with some limitations and difficulties.

- The first of these difficulties lies in validating the produced FMUs. The "Cross Check" tests and the certification procedures are providing some useful information, but it is not sufficient to give confidence on the accuracy and performance of the co-simulation FMUs when coupled to other units.
- Multibody systems, where large rotation usually make it impossible to reduce the model behavior, require co-simulation FMUs. Another source of non-linearities in these systems is collisions and contacts between bodies. Co-simulation becomes in that case more tedious, forcing the synchronization between the participants more frequent due to the non-smooth behavior of the equations of motion [3]. An extension of the standard to events could help handling more efficiently this non-smooth behavior.
- A valuable extension of the standard would be to support full high dimensionality interface, such as a full pressure field on a surface.

6 References

[1] fmi-standard.org

[2] Component Mode Synthesis, Jaap Wijker, in Mechanical Vibrations in Spacecraft Design, pp 369-398 Springer, Berlin, Heidelberg

[3] Nonsmooth Mechanics, Models, Dynamics and Control. Third Edition. Bernard Brogliato, Springer, Berlin, Heidelberg

[Industrial paper] Deployment process for Modelica-based models

Takashi Iwagaya[1]   Chad Schmitke[2]   Tetsu Yamaguchi[2]

[1]Cybernet Systems Co., Ltd., Japan, iwagaya@cybernet.co.jp
[2]Maplesoft, Canada, {cschmitke,tetsuy}@maplesoft.com

Abstract

This paper introduces Maplesoft's solutions for deploying Modelica-based models for non-experts of simulation. In order to apply models to analysis, either non-experts need to learn how to use simulation tools, or simulation experts need to prepare easy-of-use GUIs, like Excel and Web technologies. Maplesoft's solutions, MapleSim Explorer and MapleSim Server, allow more flexible and rapidly developed analysis tools for non-experts. These solutions encourage wide use of simulation models.

Keywords: Model execution tool, Deployment, Web-base, Cloud

1 Introduction

Modelica-based models should be applied to analyze design, and explore the design space of parameters and determine design parameters. However, it is often difficult to use the simulation tools for general design engineers who do not have simulation expertise. One solution for this problem is to educate engineers, but they don't have enough time to learn new tools and methodologies. To solve this difficulty, simulation experts can develop models and ease-of-use GUI applications in many cases (Figure 1). Thus, it takes time to provide analysis environment to non-expert users.

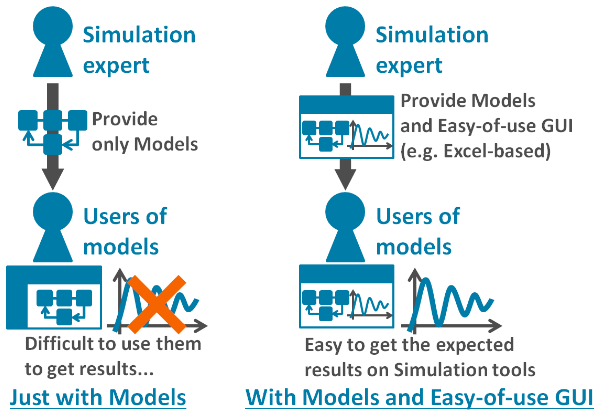

Simulation expert
Provide only Models
Users of models
Difficult to use them to get results...
Just with Models
Simulation expert
Provide Models and Easy-of-use GUI (e.g. Excel-based)
Users of models
Easy to get the expected results on Simulation tools
With Models and Easy-of-use GUI

Figure 1. How to provide models to the non-experts of simulation.

The other difficulty for non-experts is how to utilize models developed by experts. The usage of them must be related to given design tasks. Non-experts may have difficulties when parameters and evaluation points for the design do not correspond directly with the parameters and variables of simulation models. In other words, post processing may be required to evaluate the design. For example, frequency analysis may be applied to simulation results in the time domain. Additionally, parameters of a model tuned by design engineers may be several selected parameters in a model. Thus, the limited accessibility in the parameter setting is sometimes useful for non-experts. In these cases, simulation experts will provide customized analysis tools for each design tasks, which make operations of tools easier for non-experts.

In the following section, we introduce Maplesoft's solutions to simplify such deployment framework, and expand the use of Modelica models.

2 Deployment types and use-cases

Maplesoft product can offer three types of deployment approach other than that based on code generation (for example, using FMI) (Figure 2). In the figure, models are Modelica-based models created in MapleSim, and analysis tools mean pre and post processing tools which are connected to models.

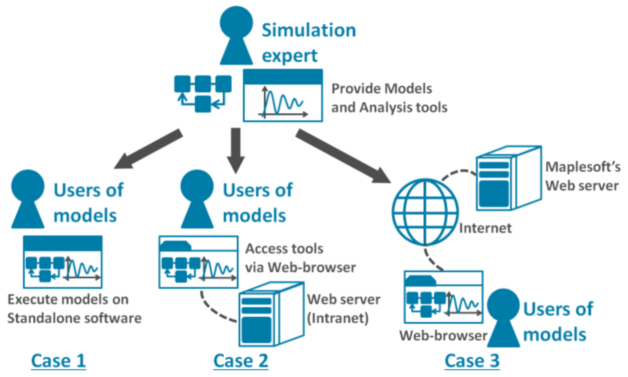


Figure 2. Three types of deployment with Maplesoft products.

The three types of deployment are :

- Case 1 : With Standalone software, MapleSim Explorer (MapleSim Explorer, 2018). Users need to install software and store models and analysis tools developed by experts on their own computer

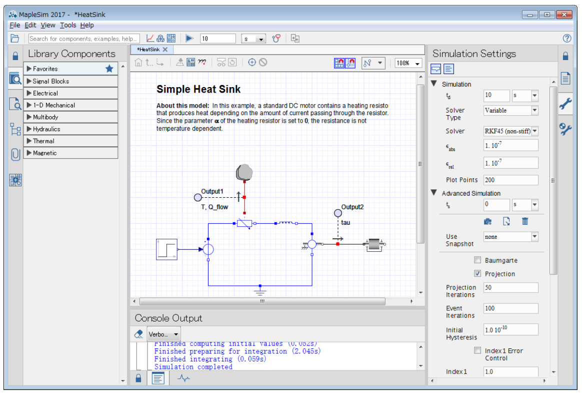
- Case 2 : Web-based environment, MapleSim Server (MapleSim Server, 2018). Software, models and analysis tools are installed on Web server. User can access the analysis tools via Web-browser.
- Case 3 : Cloud-based environment, MapleCloud (MapleCloud, 2018). Analysis tools containing models are uploaded to cloud platform. Users can execute tools on cloud and download them to their own environment.

As fundamental capability of Maplesoft product suite, it is easy to build user-defined analysis tools in Maple, which consist of calculation procedures and GUI, to set parameters to defined Modelica-based models and apply post processing with the simulation results. Additionally, one of Maple's file formats (.maple file) can contain Modelica-based models (MapleSim model file .msim and Modelica code .mo). And, MapleSim model file (.msim) also can store analysis tools created in Maple. Thus, a packed file consisted of model and analysis tool can be easily uploaded to MapleCloud or Web-server, and shared between simulation experts and non-expert users. Thus, users do not need to worry about connectivity between models and analysis tools.

Advantages of above three types of deployment approach are outlined in the following sections.

2.1 Case 1 : MapleSim Explorer

MapleSim Explorer gives almost same usability as MapleSim itself (Figure 3). Users can set value of parameters, simulate models and use analysis tools, but cannot modify model structure and save any changes for value of parameters in MapleSim Explorer.


Simple Heat Sink

About this model: In this example, a standard DC motor contains a heating resisto that produces heat depending on the amount of current passing through the resistor. Since the parameter **α** of the heating resistor is set to **0**, the resistance is not temperature dependent.

Figure 3. GUI of MapleSim Explorer

The benefits of using MapleSim Explorer are :

- There is no risk of errors for inadvertent user changes models or analysis tools.
- Especially in terms of using models and analysis tools, user experience in MapleSim Explorer is almost same as MapleSim. Therefore, simulation experts can easily teach non-experts on how to use given application.

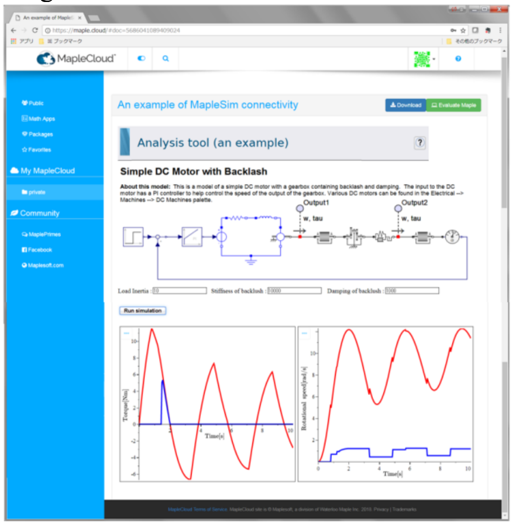
On the other hand, this workflow suffers from some minor development inconveniences. Users need to install software in local computer. Additionally, version management of models and analysis tools should be handled by simulation experts or tool manager. In other words, these are weaknesses compared with two other deployment approaches.

2.2 Case 2 and 3 : MapleSim Server and MapleCloud

MapleSim Server and MapleCloud are built on top of the same underlying technology. User experiences about execution for non-experts are the same in both case. Users can use GUI components on analysis tools developed in Maple (Figure 4).

For users in either environments, they do not need to install anything locally; they simply access the specified URL in their web browser. Major difference between two environments is whether person who manages the analysis tools also needs to manage the web server or not.

As other benefit of MapleSim Server, it can be integrated with existing web-based systems in user's organization especially about GUI because of Web technologies.


An example of MapleSim connectivity

Analysis tool (an example)

Simple DC Motor with Backlash

About this model: This is a model of a simple DC motor with a gearbox containing backlash and damping. The input to the DC motor has a PI controller to help control the speed of the output of the gearbox. Various DC motors can be found in the Electrical --> Machines --> DC Machines palette.

Load Inertia : 10
Stiffness of backlash : 10000
Damping of backlash : 1000

Figure 4. An example of analysis tool on MapleCloud

3 Workflow of Analysis tools development

In this section, typical workflow of Analysis tools development is shown by using a simple case study. simulation experts need to develop models on MapleSim and analysis tools on Maple for all types of deployment (Figure 5). First, models are created with components which are defined by Modelica as general Modelica based modeling tool. Next step is to develop analysis tools in Maple. GUI in tools is easily developed by using Maple's predefined GUI components. And, to define the behavior of each components, action code can be implemented in the compoment.

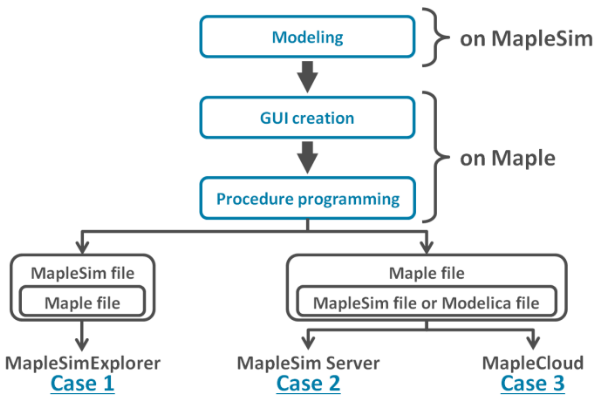
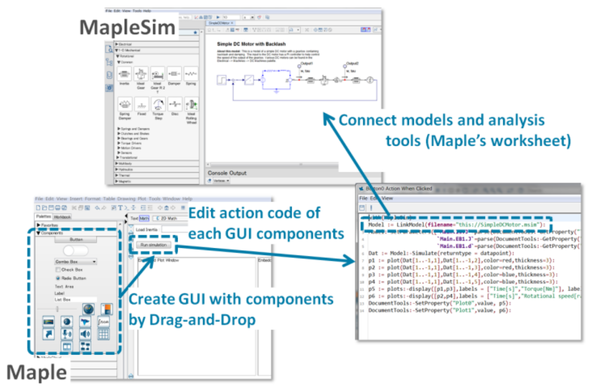


Figure 5. **Workflow of Analysis tools development**

Especially, connection between model and analysis tool is established by using three options:

1. Attach one or multiple Maple files of analysis tool (.mw or .maple) to MapleSim model file (.msim). Then use the MapleSim component of Maple's GUI component to establish connection between model and analysis tools.
2. Attach one or multiple MapleSim model files (.msim or .mo) to Maple file of analysis tool (.maple). And then use Maple command in Maple file to establish connection between models and analysis tools. Figure 6 shows this case.
3. Use Maple command in Maple file of analysis tool (.mw or .maple) to specify file stored path of MapleSim model file (.msim or .mo), which is located at outside the Maple file.


MapleSim
Connect models and analysis tools (Maple's worksheet)
Edit action code of each GUI components
Create GUI with components by Drag-and-Drop
Maple
Button0 Action When Clicked
File Edit View

```
with(MapleSim):
Model := LinkModel(filename="this://SimpleDCMotor.msim"):
Model:-SetParameters([`Main.EB1.J`=parse(DocumentTools:-GetProperty("
                      `Main.EB1.J`=parse(DocumentTools:-GetProperty(
                      `Main.EB1.d`=parse(DocumentTools:-GetProperty(
Dat := Model:-Simulate(returntype = datapoint):
p1 := plot(Dat[1..-1,1],Dat[1..-1,2],color=red,thickness=3):
p2 := plot(Dat[1..-1,1],Dat[1..-1,3],color=red,thickness=3):
p3 := plot(Dat[1..-1,1],Dat[1..-1,4],color=blue,thickness=3):
p4 := plot(Dat[1..-1,1],Dat[1..-1,5],color=blue,thickness=3):
p5 := plots:-display([p1,p3],labels = ["Time[s]","Torque[Nm]"], labe
p6 := plots:-display([p2,p4],labels = ["Time[s]","Rotational speed[r
DocumentTools:-SetProperty("Plot0",value, p5):
DocumentTools:-SetProperty("Plot1",value, p6):
```

Figure 6. Modeling and Analysis tools development

First option in above case is for using MapleSim Explorer, which is Case 1 in Figure 2 and 5. As typical workflow, simulation experts can simply share the model file which contains one or multiple analysis tools with non-expert users. Then, they can get several analysis results instantly for given target design, which can be expressed within the model. (Figure 7).

Second option can be used for MapleSim Server and MapleCloud, which are corresponding to Case 2 and Case 3. Simulation experts need to upload analysis tool that contains one or multiple models to own web server or MapleCloud's. Then, non-expert users just need to access the specified URL from their web browsers to run analysis tools. With this framework, it is possible to analyze multiple attached models and to do multiple types of analysis which is implemented in the associated Maple file.

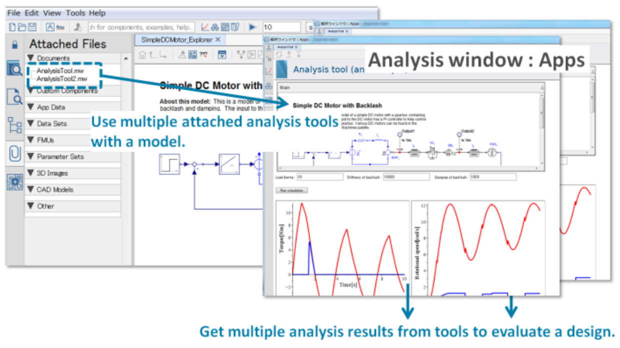

Attached Files
AnalysisTool.mw
AnalysisTool2.mw
Simple DC Motor with Backlash
Analysis window : Apps
Use multiple attached analysis tools with a model.
Get multiple analysis results from tools to evaluate a design.

Figure 7. Using MapleSim Explorer

Third option is effective for simple routine tasks with different models only in Case 2. For example, if the definition of interface is fixed, the code generation (FMU and S-Function) can be applied, though it is not analysis task.

Simulation experts need to select what the appropriate file structure is, based on use-cases when they develop models and analysis tools. Meantime, models and analysis tools can be used in common use case for all types of deployment, hence simulation experts can provide analysis environment which is suitable for non-expert users' demand appropriately.

4 Conclusions

In this paper, we introduced typical deployment approaches by Maplesoft product suite. One of difficulties that may occur when simulation experts provide models and analysis tools to non-experts, can be clarified with described approaches. The key factor of how simulation experts can develop analysis tools efficiently is related to flexible file structure for connection between model files and analysis tool files. In this context, Maplesoft technologies allow to access Maple files directly from web browser. We are already working closely with some customers in Japan to see practical increase of usage for Modelica-based models by providing these deployment approaches discussed in this paper.

References

MapleCloud (2018):
https://maple.cloud

MapleSim Explorer (2018):
https://www.maplesoft.com/products/maplesimexplorer/

MapleSim Server (2018):
https://www.maplesoft.com/products/maplesimserver/