Proceedings of the

Asian Modelica Conference 2020 is sponsored by:

Asian Modelica Conference 2020 is organized by Modelon and Modelica Association

Proceedings of Asian Modelica Conference 2020
Tokyo, Japan, October 08-09, 2020

Editors:
Dr. Rui Gao and Dr. Yutaka Hirano

Published by:
Modelica Association and Linköping University Electronic Press

ISBN: 978-91-7929-775-6
Series: Linköping Electronic Conference Proceedings, No. 174
ISSN: 1650-3686
eISSN: 1650-3740
DOI: https://doi.org/10.3384/ecp2020174

Organized by:
Modelon K.K.
1-10-3-209, Roppongi, Minato-ku,
Tokyo 106-0032
Japan

In cooperation with:
Modelica Association
c/o PELAB, Linköpings Iniv.
SE-581 83 Linköping
Sweden

Conference location:
Virtual platform in Whova

Welcome Message

Following the first and the second conferences in 2016 and 2018, the 3rd conference renamed as The Asian Modelica Conference 2020 takes place in Tokyo again.
This is a special year with COVID-19 spreading out worldwide, the original physical conference was delayed from May 13-14 to October 8-9 and then converted to a virtual conference since August 25. The local organizer and Modelica Association work closely to overcome various difficulties to make the conference available. We are now proud to present a conference with:

- 2 Keynote speeches
- 18 paper presentations
- An exhibition area featuring 9 exhibitors
- Virtual event platform on Whova

According to Modelica Association standards, all papers are peer-reviewed and will be freely available for download.
We want to acknowledge the support we received from the conference board and program committee. Special thanks to our colleagues at Modelon KK 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.
Even with a virtual platform, the objective of creating the conference as an arena in Asia for sharing knowledge and learning about the latest scientific and industrial progress related to Modelica and FMI (Functional Mockup Interface) is unchanged. We wish all participants an enjoyable and inspiring conference!

Tokyo, October 8, 2020

Rui Gao & Yutaka Hirano

DOI
10.3384/ecp2020174

Keynotes Speakers

James Kuffner
Toyota Research Institute - Advanced Development, Inc.

Smart Mobility in a Cloud-Connected World

The automobile industry is currently undergoing a dramatic transformation. High-performance networking, deep learning technology, and cloud computing have radically transformed how individuals and businesses manage data, and is poised to disrupt the state-of-the-art in the development of intelligent machines. This talk explores how automated driving and artificial intelligence based on connected, distributed computing enables future mobility systems that will impact the design and evolution of future cities.

Tobias Bellmann & Dirk Zimmer
DLR German Aerospace Center

From industrial applications to electric flight: Modelica as key enabler

With its high degree of versatility, Modelica has established itself for a multitude of applications ranging from complex hardware-in-the-loop simulators of automation components to the design optimization of future electric aircraft. The corresponding tool chain meanwhile embraces real-time interfaces for virtual commissioning as well as advanced 3D visualization. Robust modeling principles further promote the acceptance in industrial use and help to save costs in the development process.

DOI
10.3384/ecp2020174

Program Committee

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

Program Chair
Dr. Yutaka Hirano, Toyota Research Institute - Advanced Development, Inc., Japan

Program Board
Dr. Rui Gao, Modelon K.K., Tokyo, Japan
Dr. Yutaka Hirano, Toyota Research Institute - Advanced Development, Inc., Japan
Dr. Hilding Elmqvist, Mogram, Lund, Sweden
Prof. Peter Fritzson, Linköping University, Sweden
Prof. Martin Otter, DLR, Germany
Dr. Michael Tiller, Xogeny, Michigan, USA
Dr. Hubertus Tummescheit, Modelon Inc. USA

Program Committee
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. Peter Fritzson, Linköping University, Linköping, Sweden
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
Prof. Tadao Kawai, Osaka City University, Osaka, Japan
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
Prof. Henrik Nilsson, University of Nottingham, Nottingham, United Kingdom

Prof. Hidekazu Nishimura, Keio Gijuku University, Yokohama, Japan
Prof. Kenichiro Nonaka, Tokyo City University, Tokyo, Japan
Thierry-Stephane Nouidui, LBL, Berkeley, USA
Prof. Shigeru Oho, Nippon Institute of Technology, Tokyo, Japan
Prof. Koichi Ohtomi, Meiji University, Tokyo, Japan
Prof. Martin Otter, DLR, Oberpfaffenhofen, Germany
Dr. Andreas Pillekeit, dSPACE, Dortmund, Germany
Dr. Adrian Pop, Linköping University, Linköping, Sweden
Johan Rhodin, 84 Codes Consulting LLC, Missouri, USA
Dr. Clemens Schlegel, Schlegel Simulation GmbH, Munich, Germany
Dr. Peter Schneider, Fraunhofer IIS, Design Automation Division, Dresden, Germany
Prof. Stefan-Alexander Schneider, Hochschule Kempten, Kempten, Germany
Michael Sielemann, Modelon AB, Lund, Sweden
Dr. Martin Sjölund, Linköping University, Linköping, Sweden
Prof. Thierry Soriano, Université de Toulon, Toulon, France
Dr. Rita Streblow, RWTH Aachen University, Aachen, Germany
Dr. Ed Tate, Exa, Livonia, USA
Dr. Wilhelm Tegethoff, TLK-Thermo GmbH, Braunschweig, Germany
Matthis Thorade, Universität der Künste Berlin, Berlin, Germany
Dr. Michael Tiller, Xogeny Inc., Michigan, USA
Dr. Jakub Tobolar, DLR, Oberpfaffenhofen, Germany
Dr. Hubertus Tummescheit, Modelon, Hartford, USA
Prof. Alfonso Urquia, UNED, Madrid, Spain
Prof. Luigi Vanfretti, Rensselaer Polytechnic Institute, Troy, USA
Dr. Stéphane Velut, Modelon AB, Lund, Sweden
Stefan Wischhusen, XRG Simulation Gmbh, Hamburg, Germany
Dr. Dirk Zimmer, DLR, Oberpfaffenhofen, Germany

DOI
10.3384/ecp2020174

Contents

DOI
10.3384/ecp2020174

Author Index

DOI
10.3384/ecp2020174

Introducing the Virtual Systems Interface for Dynamic Coupling of Continuous Time Systems with Discontinuities

Jeffrey Morgan[1] Bruno Loyer[2]

[1]General Motors, United States of America, jeff.morgan@gm.com
[2]Siemens Digital Industries Software, France, bruno.loyer@siemens.com

Abstract

This paper introduces the Virtual Systems Interface (VSI) as a potential enhancement of the FMI interface or as a separate open source interface. The VSI interface is intended to simplify model exchange and dynamic model coupling of continuous time systems with discontinuities while ensuring fast, accurate and low-cost simulations. The VSI interface uses a single interface for the coupling of both continuous time and discrete time systems. It is designed for variable time step integration with proper discontinuity handling and convergence checking. It requires no run-time licenses for any model packaged using the interface.

Keywords: virtual systems interface, dynamic coupling, controls integration, functional mock-up interface

1 Introduction

Automotive and other industries are relying on the use of simulation models to reduce product development time and cost. This has generated a need for the dynamic coupling of both physical system models (called plant models in this paper) and electronic controller models. Furthermore, many systems are composed of subsystems and components produced by different companies so the exchange of models and their integration into simulation packages is required. This has led to the development of the Functional Mock-up Interface (FMI). The Function Mock-up Interface is an open specification which allows import and export of both plant and controller models into and out of any simulation packages that support the specification.

There are two types FMI interfaces: model exchange and co-simulation. The model exchange interface is intended for use with models that are described by differential, algebraic and discrete equations with or without discontinuities, and the system equations are solved simultaneously. The co-simulation interface is intended for use with models where each subsystem is solved independently, and data is exchanged between subsystems only at discrete communication points. In general, the co-simulation interface gives accurate results only for systems that physically exchange information at discrete communication points (i.e. digital controllers) or have low degrees of dynamic coupling so that the destabilizing effect of the discrete communication can be made negligible by selecting a small enough communication interval.

This paper focuses on the model exchange interface and how it can be used effectively for the dynamic coupling of continuous time systems with discontinuities. First, two example usages of the FMI 1.0 model exchange interface are presented, and the issues encountered are explained. Next, some background for effective usage of a model exchange type interface is presented. Finally, a new interface is introduced which is intended to ensure fast, accurate and low cost means for dynamic coupling of continuous time systems with discontinuities.

2 FMI 1.0 Model Exchange Interface Examples and Issues

Two examples are presented using the FMI 1.0 model exchange interface. The first example is an automotive driveline as an example of a continuous time plant model with discontinuities. The second example is an engine dynamic model as an example of a continuous time behavioral controls model with discontinuities. The results and the issues encountered are explained.

2.1 Example 1: Automotive Driveline

The first example is an automotive driveline torsional model. This model is incorporated into several vehicle systems models for the evaluation of both the vehicle hardware and control system designs. The automotive driveline model contains simple torsional elements (springs, dampers, inertias, clutches, planetary gear sets, etc.). The model is run in fixed gear with the torque convertor open, so there are no active controls. The automotive driveline model is a continuous time system with discontinuities. The discontinuities occur due to the inclusion of backlash elements in the model. The automotive driveline model is shown in Figure 1.

The automotive driveline model is implemented in Simcenter Amesim version 15.1 using Microsoft Visual Studio 2010 64 bit as the compiler. The Simcenter Amesim FMU generation tool was used to export the model to FMI 1.0 model exchange format and then it was imported back into Simcenter Amesim as an FMU to verify that it runs correctly.

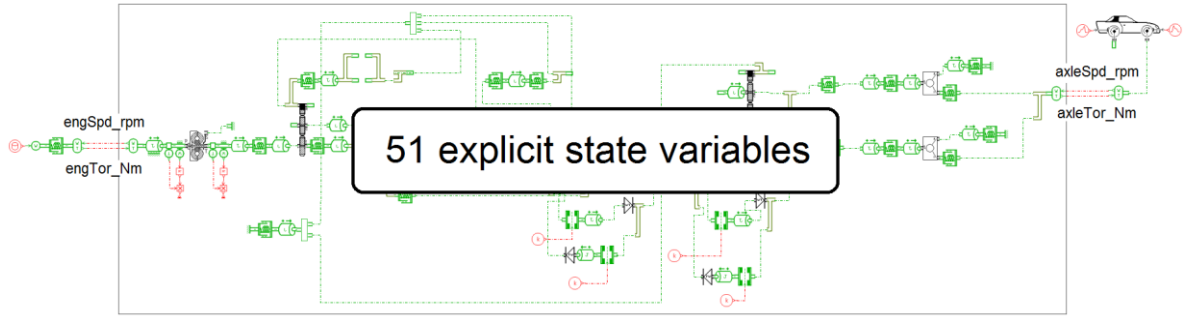


Figure 1. Automotive Driveline Torsional Model (Simcenter Amesim model)

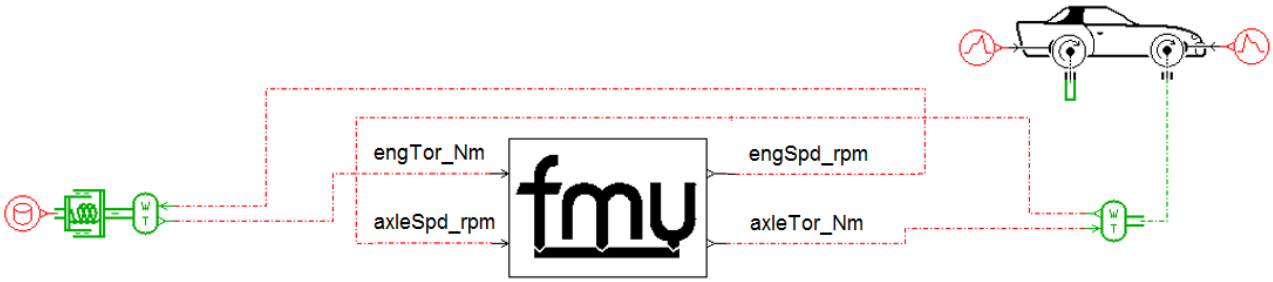


Figure 2. Automotive Driveline FMU integrated into a Vehicle System Model (Simcenter Amesim model)

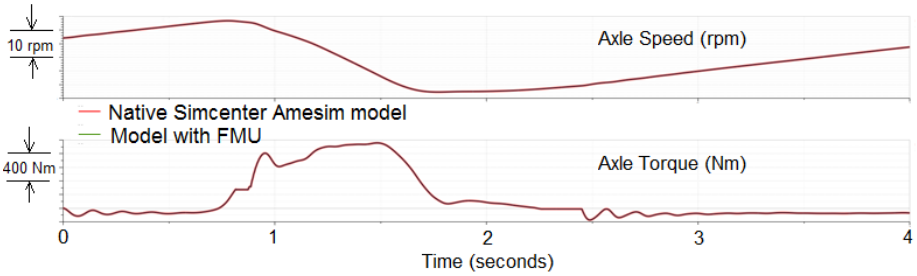


Figure 3. Results for Native Simcenter Amesim Model vs. Simcenter Amesim Model with FMU

The automotive driveline FMU is integrated into a vehicle system model as shown in Figure 2. The simulation was run using the Simcenter Amesim standard integrator and a 1 ms print interval. The results for the system simulation with the automotive driveline included as native elements and as an FMU is shown in Figure 3 (results are the same and show up as one trace).

The FMU produced accurate results but was 10.3 times slower than the native Simcenter Amsim model (24.7 seconds vs. 2.4 seconds). The slow simulation was determined to be caused by the generation of an implicit variable when connecting the FMU. This artificial algebraic loop can be avoided by a better handling of the model's input-output dependencies when exporting the FMU. It should be noted that improving the management of these artificial algebraic loops was one of the motivations behind FMI 2.0.

2.2 Example 2: Engine Dynamic Model

The second example is an automotive gasoline engine dynamic model. This model is incorporated into several vehicle systems models for the evaluation of both the vehicle hardware and control system designs. The engine dynamic model was built in Simulink and a 3rd party commercial FMI generation tool was used to export it to FMI format so that it could be used in other simulation packages. The engine dynamic model is a continuous time system with discontinuities. The discontinuities occur due to the inclusion of continuous time rate limiter elements in the model. The engine dynamic model is shown in Figure 4.

The model is implemented in Simulink version 2014a (64 bit). FMUs were generated using Modelon software version 2.6.1 and then imported into Simcenter Amesim version 14.2 using Microsoft Visual Studio 2010 64 bit as the compiler for system simulation. Even though the Model Exchange interface was used, it was found that the exported model was different depending on which Simulink solver was specified at the time of export. When a fixed time step solver (ODE4 with 1 ms time step) was specified the exported FMU was generated with each time step being a discontinuity (call this the

DOI 10.3384/ecp202017413

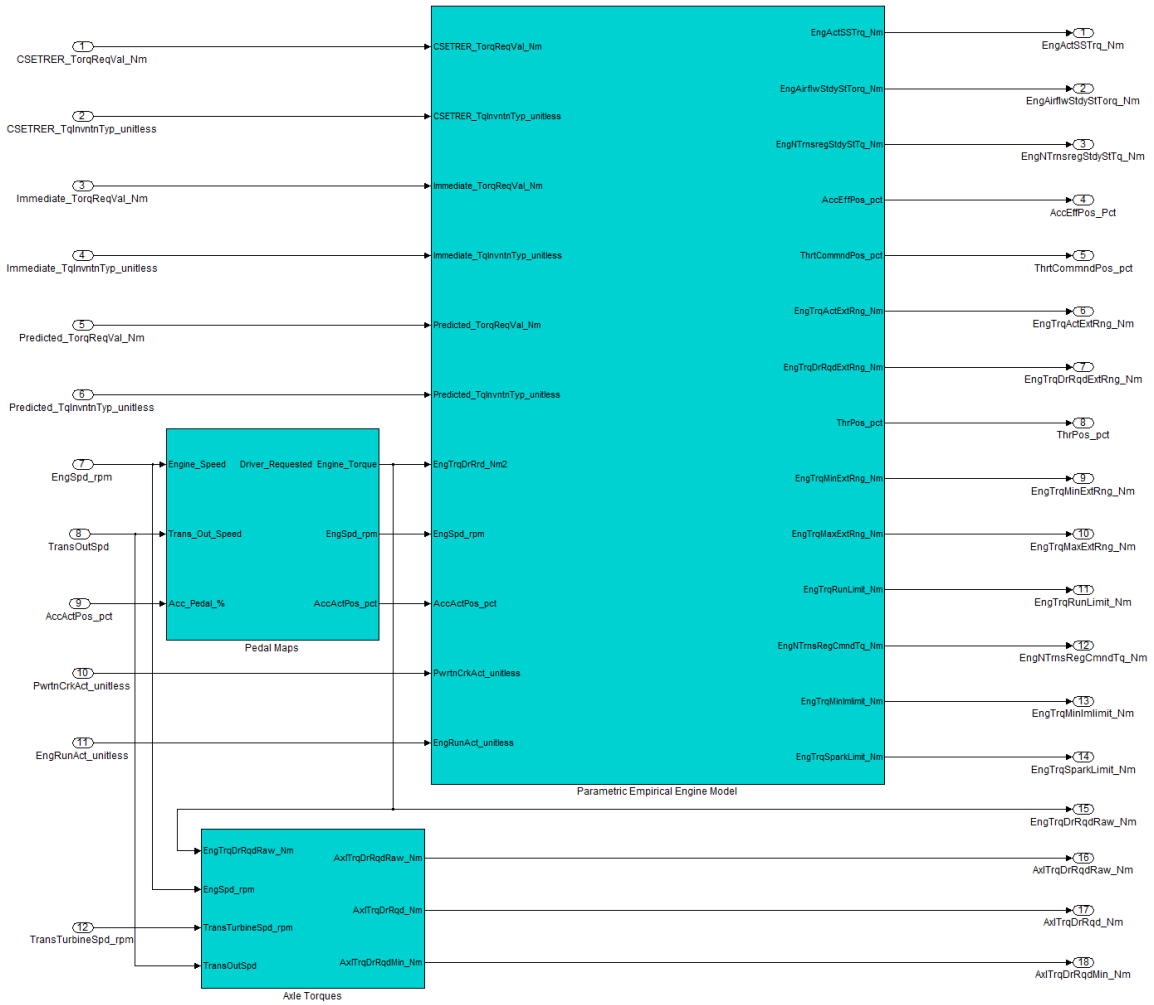


Figure 4. Engine Dynamic Model (Simulink Model)

FMU with time-based discontinuities). When a variable time step solver (ODE45 using a relative tolerance of 1e-3) was specified the exported FMU was generated without each time step being a discontinuity but including event-based discontinuities (call this the FMU with event-based discontinuities). For comparison, a Simcenter Amesim equivalent model was created so the results could be compared. The results are shown in Figure 5.

The simulation was run using the Simcenter Amesim standard integrator and a 1 ms print interval. The FMU with time-based discontinuities produced accurate results while the FMU with event-based discontinuities did not. The FMU with time-based discontinuities was 65 times slower than the native Simcenter Amesim model (1733 seconds vs. 26 seconds). The FMU with event-based discontinuities was 1.3 times slower than the native Simcenter Amesim model (33 seconds vs. 26 seconds). The slow simulation speed of the FMU with time-based discontinuities was due to the creation of discontinuities at each time point corresponding to the Simulink model's originally specified integration time step (1 ms). The inaccuracy of the FMU with event-based discontinuities was due to incorrect handing of the model's real discontinuity points.

2.3 FMI Model Exchange 1.0 Interface Issues

The learnings from the two examples was that the FMI 1.0 Model Exchange interface does not contain enough information on how the internal equations, discontinuity handling, and solution algorithms should be implemented in order to ensure that fast and accurate FMUs are generated.

This paper proposes a new (extended) interface intended to address these issues. The new interface combines Virtual Systems in-the-Loop (VSiL) technology developed jointly by General Motors with the support of LMS (now Siemens Digital Industries Software) and Autonomie technology developed by Argonne National Labs. These two technologies are described next.

3 Virtual Systems in-the-Loop (VSiL)

Virtual Systems in-the-Loop (VSiL) has been used at General Motors since 2005 (Glaue, 2006). This

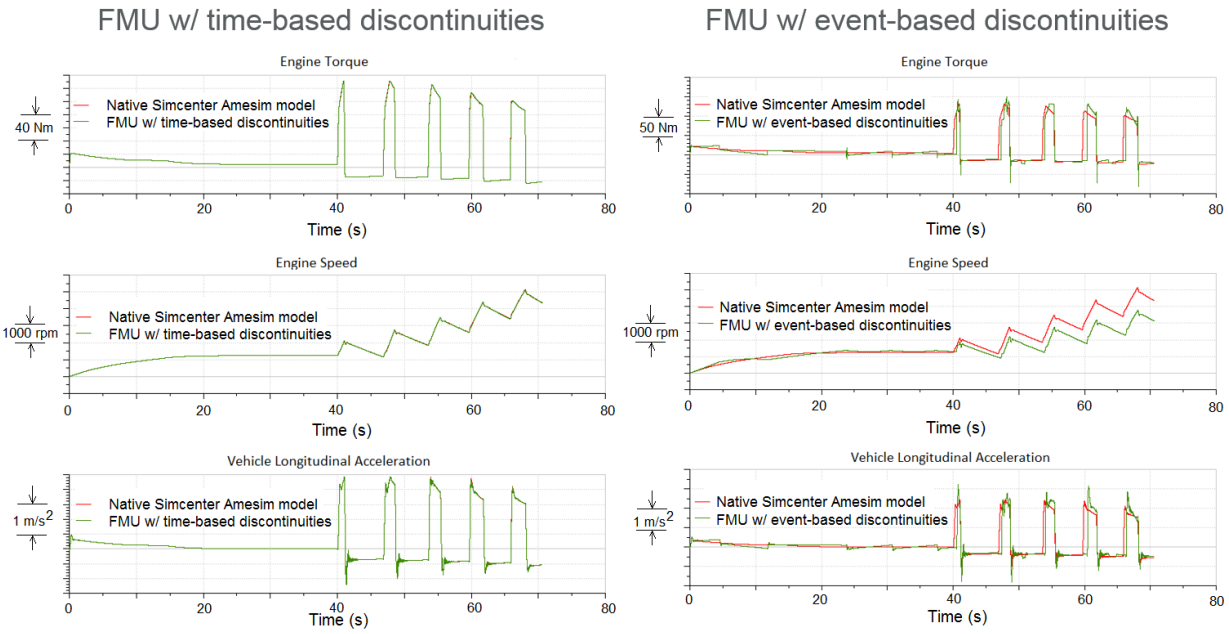


Figure 5. Results for the Engine Dynamic Model Represented Using FMI 1.0 Model Exchange

methodology allows electronic control units (ECUs) to be integrated with Simcenter Amesim plant models and run as a single executable. The Simcenter Amesim plant model is modeled using standard and custom sub models based on Bond Graph (powerflow) methodology (Rosenberg and Karnopp, 1983). This approach defines causality between the powerflow variables which determines how the equations are assembled and solved.

The controller model is built using the actual software source code corresponding to the application layer. The application layer includes all the control algorithms and calibrations which are independent of the actual controller hardware. A hardware input/output (HWIO) layer is built to replace the actual controller hardware specific software which includes all other software code needed to run the ECU, but which is not included in the application software code (i.e. the hardware input and output drivers, task scheduler (RTOS), CAN messaging, etc.). The application code is compiled into an executable file which connects to the plant model thru the glue layer. The glue layer includes method declarations for all the external methods and variables referenced by the application code and any default definitions used when only partial ECU functionality is being analyzed. A set of Simcenter Amesim custom submodels is created to implement the sensors, actuators, scheduler, CAN messaging and peek-inside-ECU functionality. An example engine camshaft phaser VSiL controller model is shown in Figure 6.

Sensors read continuous time signals in engineering units from the plant model and convert these into controller variables defined by the HWIO interfaces. Actuators take controller variables defined by the HWIO interfaces and convert them into continuous time signals in engineering units which are sent to the plant model. The scheduler determines when controller

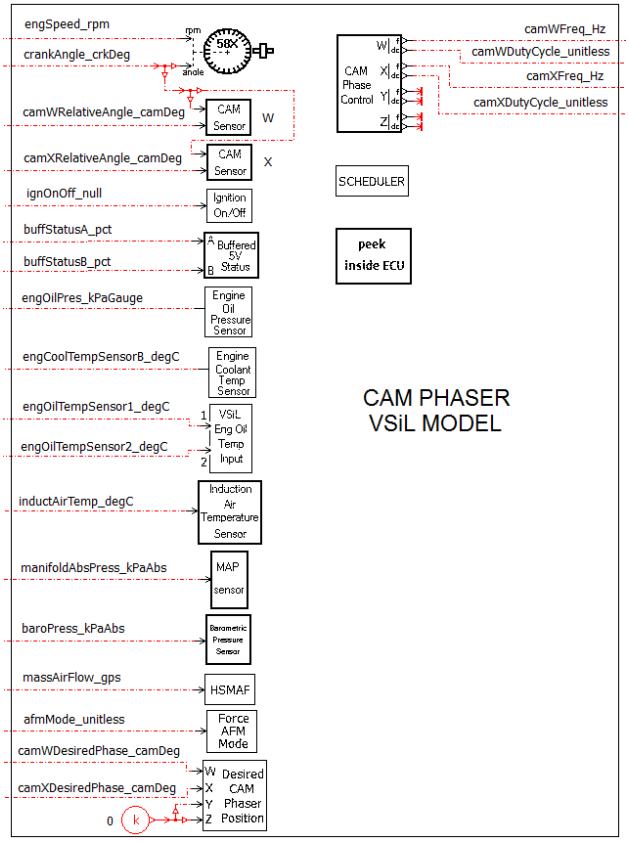


Figure 6. Engine camshaft phaser VSiL controller (Simcenter Amesim model)

methods are executed including controller initialization and the updating of a limited set of calibrations. The CAN messaging sub models handle inter-controller communication signals. The peek-inside-ECU sub models expose a limited set of internal ECU variables for printing and plotting.

DOI 10.3384/ecp202017413

VSiL is intended for hardware focused analyses where the controls are treated as a black-box with only a limited number of controller functionalities (rings), calibrations and internal variables are of interest. Other approaches are used for fully functional software in-the-Loop (SIL) models where all functionality of the controller is of interest.

4 Autonomie

Autonomie is a software package primarily used to analyze vehicle performance and fuel economy (Halbach *et al*, 2010). Autonomie is based on Bond Graph methodology (powerflow) concepts but is implemented using a signal flow approach. This makes it easy to connect plant models to controller models which are also generally implemented using a signal flow approach. Autonomie uses MATLAB/Simulink as the master simulator and any external systems can be integrated as S-functions. The S-Function interface has both a standard (model exchange) and co-simulation interface type. The system level interface used in Autonomie is shown in Figure 7.

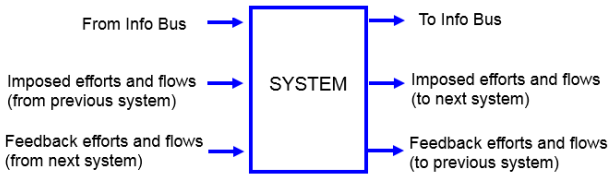


Figure 7. System Level Interface used in Autonomie

The key feature of the Autonomie interface is that it separates the power flow variables from the non-powerflow variables. This concept will be used in the VSI Interface and its importance will be explained later in this paper.

5 Fundamental Problem with Co-simulation Type Interfaces

Before introducing the new interface, we will explain the fundamental problem with co-simulation type interfaces which led to the development of the Virtual Systems Interface. First, consider the operation of a real electronic control unit (ECU) and how data is logged. This is illustrated in Figure 8.

The current time stamp is used for all variables updated during the execution of controller code that is run as a result of a task initiator. This makes it look like these variables are updated at the same time. In reality, there is a time delay between the task initiator and the updated variables.

For the co-simulation interface (parallel execution and fixed time step communication) the execution is done as shown in Figure 9.

The inputs at the previous time step (t_{n-1}) are used to calculate the outputs at the current time step (t_n). Then, the inputs at the current time step (t_n) are updated. We will call this Execution Order 1:

Step Forward > *Update Outputs* > *Update Inputs*

A different execution order can also be used which updates the inputs at the current time step (t_n) first and then calculates the outputs at the current time step (t_n). We will call this Execution Order 0:

Update Inputs > *Step Forward* > *Update Outputs*

Execution order 0 requires sequential (serial) execution of the co-simulating models. During testing of a two-model co-simulation system (one plant and one controller model), it was found that to match the closed loop simulation results, execution order 1 was needed and to match open loop simulation results, execution order 0 was needed. This was a surprising result which

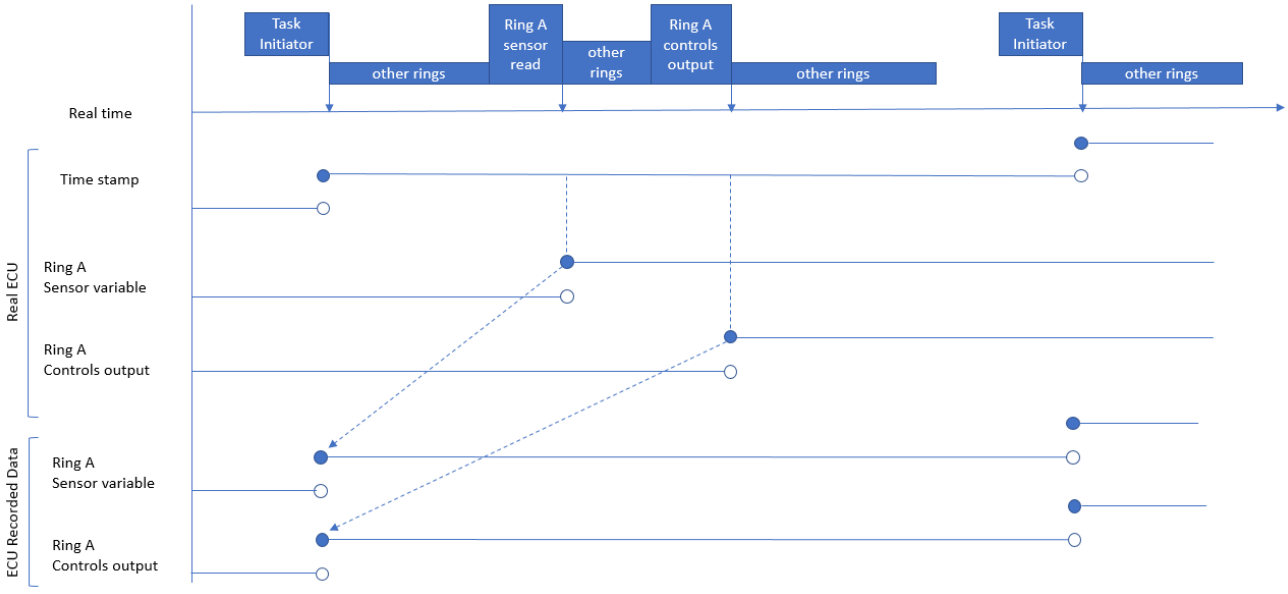


Figure 8. ECU data logging schematic

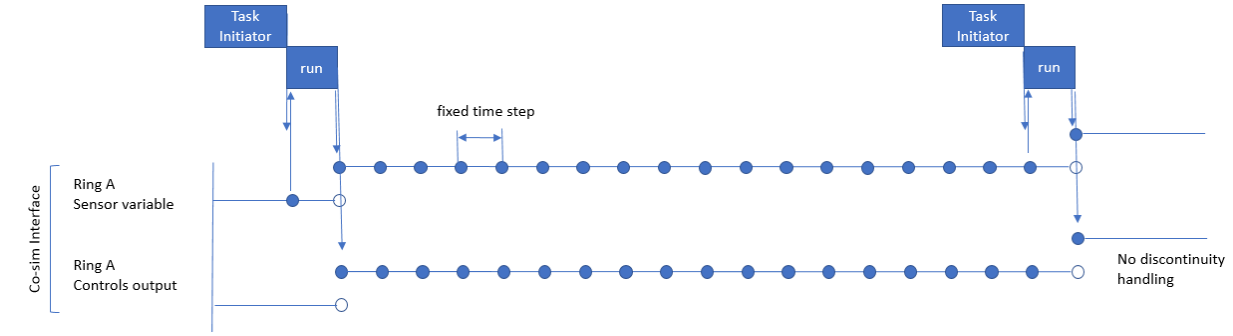


Figure 9. Co-simulation execution schematic

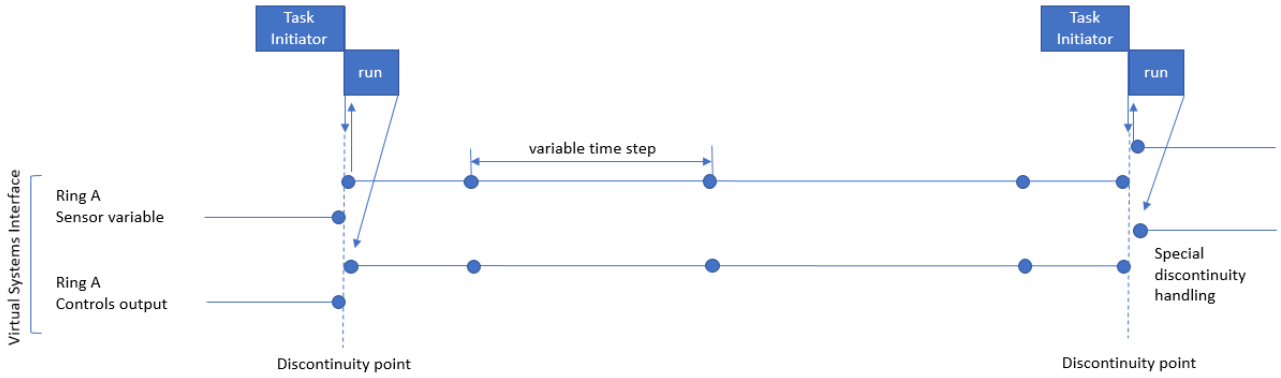


Figure 10. VSI interface execution schematic

led to an extensive investigation into co-simulation data exchange and discontinuity handling. The result of this investigation was that to accurately predict the dynamics of coupled systems (at all coupling strengths and at all time steps) a simultaneous solution of the equations of motion is needed with proper discontinuity handling. The lack of this feature is the fundamental problem with co-simulation type interfaces and was the motivation for the development of the Virtual Systems Interface.

6 Virtual Systems Interface Execution Schematic

The VSI interface has simultaneous solution of the equations of motion and proper discontinuity handling. This is shown in Figure 10.

Discrete systems (like ECUs) are treated as continuous time systems with discontinuities. The VSI interface does not have the sensing and controls delays that are present in actual ECUs (compare Figure 8 and Figure 10). These delays are either neglected or can be added to the sensing and/or actuation part of the plant model (i.e. as continuous time delays or first order lags). These delays are often tuned using current or voltage signals measured directly on the physical wiring of the ECU (not using the ECU recorded variables which do not contain these delays – see Figure 8). The VSI interface is functionally equivalent to the VSiL approach previously described.

7 Virtual Systems Interface Description

The Virtual Systems Interface (VSI) is being introduced as a potential enhancement of the FMI interface or as a separate open source interface intended to simplify model exchange and dynamic model coupling while ensuring fast, accurate and low-cost simulations. The enablers for these features are:

1. Single interface for coupling both continuous time and discrete time systems (for simplicity)
2. Designed for variable time step integration with proper discontinuity handling and convergence checking (for fast and accurate results)
3. Requires no run-time licenses for any model packaged using the interface (for low cost)

The proposed Virtual Systems Interface (VSI) is shown in Figure 11.

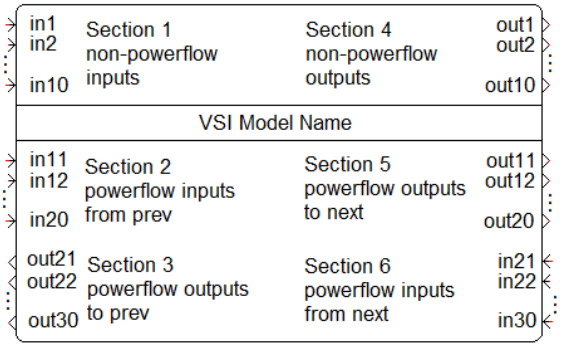


Figure 11. VSI interface Format

DOI 10.3384/ecp202017413

The VSI interface is similar to the Autonomie interface except that the location of the power flow ports is revised to have the I/O relative to the previous system and the I/O relative to the next system to be on the same sides. This is done to allow a more direct connection between upstream and downstream models and easier identification of the power flow signal pairs and corresponding causality. The non-powerflow variables are located above the VSI Model Name and the powerflow variables are located below the VSI Model Name. The inputs and outputs are numbered top to bottom as shown.

To simplify the connection to other models, all of the I/O signals are double precision continuous time variables. Any conversion to discrete time variables is done internal to the interface.

The I/O signals in section 1 and 4 of the interface are used for non-powerflow variables. Both the inputs and outputs are piecewise continuous, and the interface includes a mechanism to report the location of discontinuities so that these are handled correctly. There can be state variables that are integrated internally and/or state variables that are sent to the external integrator for integration.

The I/O signals in section 2, 3, 5 and 6 of the interface are used for the physical coupling of the models using the power flow concept from Bond Graph methodology. The power flow approach requires that all signals be done in pairs where one signal is an effort type variable (force, torque, voltage, pressure, thermodynamic temperature, etc.) and the other signal is a flow type variable (linear velocity, angular velocity, current, volume flow rate, entropy flow rate, etc.). The multiplication of the two paired signals give the input or output power. The power flow signals from Section 2 and 3 of the interface define the input power to the model. The power flow signals from Section 5 and 6 of the interface define the output power from the model. The sign convention for the power flow signals is defined as shown below:

Output Power = Input Power + Internal Power Generation

Internal power generation is negative for internal power loss. Since Bond Graph methodology is used, these signals produce ordinary differential equations (ODEs) with explicit state variables only. The interface includes a mechanism to report the location of discontinuities to the external integrator so that these are handled correctly.

Both continuous time and discrete time systems can be modeled using the VSI interface. A discrete time system is defined as any system that includes one or more discrete time components.

8 Master Algorithm

The VSI interface tries to re-use as much of the FMI nomenclature and programming specifications as possible. When a model is compiled using the VSI interface it is called a VSU (Virtual Systems Unit). The VSI interface is designed to use a single master algorithm that implements variable time step communication and/or integration. The master solver high level flow chart is shown in Figure 12.

A variable time step master algorithm was selected to ensure the fast and accurate solution of the system equations. The VSI interface is not intended for use with fixed time step communication/integration algorithms as these do not handle discontinuities properly.

The VSI interface requires that all components of the system (including each VSU) be continuous time systems with or without discontinuities. Any discrete time elements must be encapsulated inside the VSI interface along with their corresponding analog-to-digital and digital-to-analog convertors. This approach is specifically done so that digital controllers can be

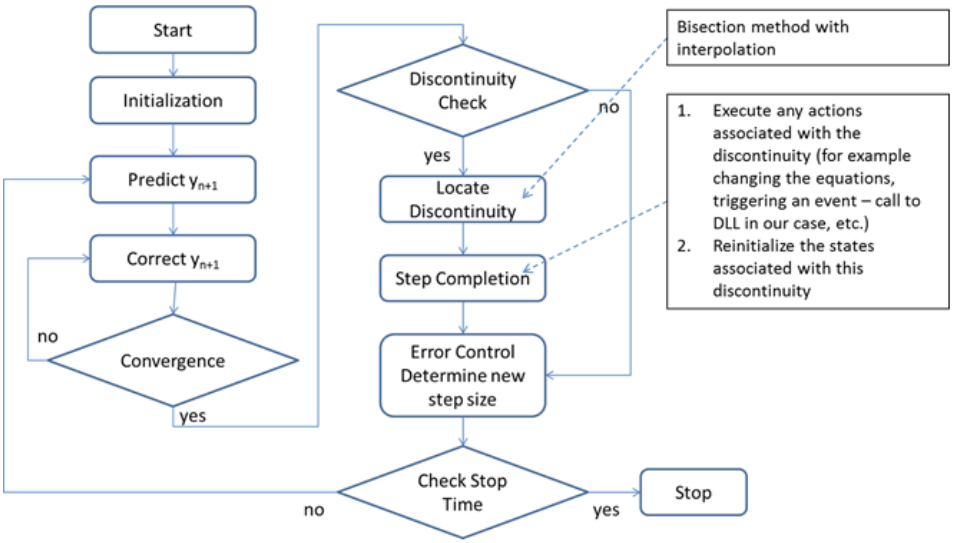


Figure 12. VSI Interface Master Solver High Level Flow Chart

packaged into VSUs and coupled to continuously time plant models using physical signals corresponding to force, torque, voltage, pressure, thermodynamic temperature, linear velocity, angular velocity, current, volume flow rate, entropy flow rate etc. as opposed to the sensed electrical versions of these signals. This approach allows the HWIO layer of the digital controller to be implemented in varying levels of fidelity inside the VSU without having to change the plant model interface. Low fidelity HWIO layers would be typical of ideal sensing (no errors). Medium fidelity HWIO layers would be typical of sensing with empirical models of the sensing errors (bias errors, random errors, first order delays and lags, etc.). High fidelity HWIO layers would be typical of physics-based models of the actual sensors and actuators with state variables that are either solved internally or externally (or combination of both).

To be considered compliant with the VSI specification, the master solver must implement the flow chart shown in Figure 12 and provide a PECE integrator based on Heun's method (Dobrushkin, 2014) as a user selectable option. Having a common integrator option is done to provide a means to validate and compare different venders' master solvers. It is allowed and encouraged that each venders' simulators contain more advanced integration options. It is expected that some systems will not be able to be solved with some integrator types so having either a manual or automatic selection of the integrator is desired. This will automatically encourage venders to create and optimize their master solvers so they can effectively handle a wide variety of equations types. Since the master solvers are expected to contain proprietary implementations, licensing of the master simulator is allowed.

The VSI interface is designed to produce the same set of ODEs regardless of which external integrator (and simulation environment) is used. It is also expected that the same results will be generated regardless of which external integrator (and simulation environment) is used (provided convergence at each time step). This was not the case with FMI 1.0 model exchange interface.

The required features of the VSI interface are shown in Table 1 along with a comparison to FMI 2.0.

Table 1. Required Features of VSI

Required Features of VSI	Part of FMI 2.0 Model Exchange Interface?
Register discontinuities	Yes
Indicate a passed discontinuity	Yes
Reinitialize after a discontinuity	Yes
Indication that a time step is converged	Theoretically feasible
Indication that a time step is used for printing	No
Separation of power flow and non-power flow signals in the interface	No
Restriction of input and output signals to be double precision variables	No
Standardization on how the internal equations are implemented and solved	No

The FMI 2.0 specification already contains some of the VSI interface features required. Thus, the VSI interface can be thought of as an abstract super-class (objective oriented programming terminology) of the FMI interface.

9 VSI Interface Examples

Five examples are presented to illustrate typical use case types for the VSI interface. These examples are described in Table 2.

Table 2. VSI Interface Examples

#	Example Name	Has Power Flow Signals?	Has states requiring external integration?	Has discontinuities?
1	Translational Mass	Yes	Yes	No
2	Dynamic Rate Limiter	No	Yes	Yes
3	Automotive Driveline	Yes	Yes	Yes
4	Behavioral Engine Dynamic Model	No	Yes	Yes
5	Engine Camshaft Phaser VSiL Controller	No	No	Yes

9.1 Example 1: Translational Mass

The first example is a one degree of freedom translational mass with power flow connections on both sides. This shows how simple components can be implement using the VSI interface. The free-body diagram of the translational mass is shown in Figure 13. The representation using the VSI interface is shown in Figure 14.

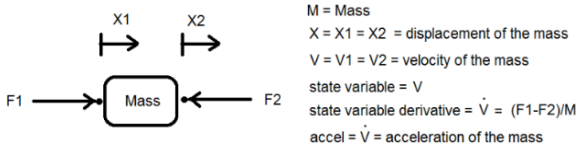


Figure 13. Free Body Diagram of a 1- DOF Translational Mass

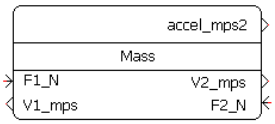


Figure 14. 1-DOF Translational Mass VSI Interface block

The state variable *V* is integrated using the external integrator using the supplied initial condition and state variable derivative value. Causality of the signals is clearly shown by the I/O nature of the signals. Forces are imposed on the mass and velocities are returned. Input power is *F1* times *V1* and output power is *F2* times *V2*. Acceleration of the mass was implemented as an output signal, but it could have been kept as an internal variable used for printing purposes only.

9.2 Example 2: Dynamic Rate Limiter

The second example is a dynamic rate limiter. This is an example of a continuous time system with discontinuities in the state variable derivative and no power flow signals. The functional description of the dynamic rate limiter is shown in Figure 15. The

DOI 10.3384/ecp202017413

representation using the VSI interface is shown in Figure 16.

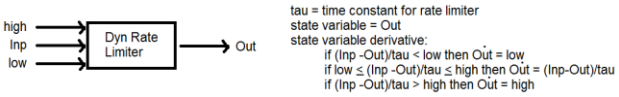


Figure 15. Functional Description of the Dynamic Rate Limiter

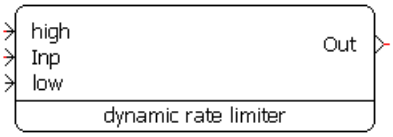


Figure 16. Dynamic Rate Limiter VSI Interface Block

The state variable *Out* is integrated using the external integrator using the supplied initial condition and state variable derivative value. A discontinuity is reported to the external integrator whenever the state variable derivative is discontinuous, and the external integrator takes the proper action. The numerical method used to identify the discontinuity point (time) can be any suitable type but the time corresponding to the discontinuity point must be between the last converged time step and the time step which identified the discontinuity. One approach of discontinuity handling is to linearly extrapolate the discontinuous equation and interpolate the time corresponding to the discontinuity as shown in Figure 17 for the dynamic rate limiter. The FMU must report only real discontinuities to ensure fast execution.

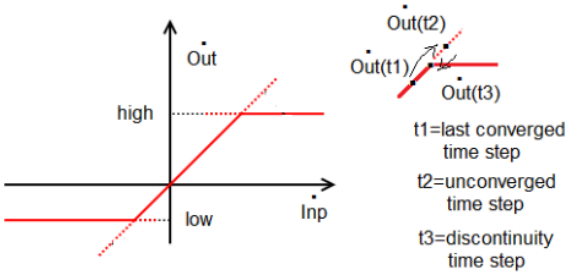


Figure 17. Discontinuity Handling Example

9.3 Example 3: Automotive Driveline

The third example is an automotive driveline torsional model. This is an example of a continuous time system with discontinuities and power flow signals. The automotive driveline torsional model was previously shown in Figure 1. The representation using the VSI interface is shown in Figure 18.

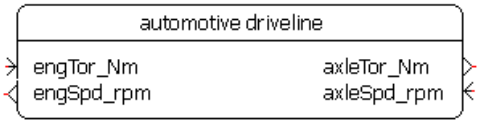


Figure 18. Automotive Driveline VSI Interface Block

The state variables are integrated using the external integrator using the supplied initial conditions and state variable derivative values. Causality of the signals is clearly shown by the I/O nature of the signals. The automotive driveline is modeled in a fixed gear state so there are no control signals sent to or from the model.

9.4 Example 4: Engine Dynamic Model

The forth example is an engine dynamic model. This is an example of a continuous time system with discontinuities and no power flow signals. The engine dynamic model was shown previously in Figure 4. The representation using the VSI interface is shown in Figure 19.

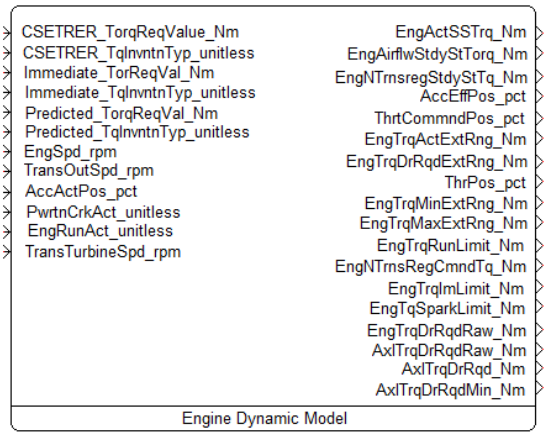


Figure 19. Engine Dynamic Model VSI Interface Block

The engine dynamic model does not have any power flow signal pairs identified in the interface (no signals below the VSI Interface name), but this does not mean there is no power flow. Any system with inputs and outputs can produce power flow bonds either intentionally or unintentionally. The engine dynamic model is intended to provide engine dynamic torque to a downstream system with the downstream system providing the engine speed as feedback. Thus, there is a hidden power flow in this interface. This occurs because of the implementation is done using a signal flow model and not a power flow model. Generally, the same physics can be represented in either a signal flow model or a power flow model and the VSI interface can handle either model type and get the same results provided that all the state variables in the non-power flow part of the interface are exposed to the external integrator. This is the case for the Engine Dynamic Model shown in this section.

9.5 Example 5: Engine Camshaft Phaser VSiL Controller

The fifth example is an engine camshaft phaser VSiL controller. This is an example of a discrete time system with no power flow signals. The VSiL controller model was previously shown in Figure 6. The representation using the VSI interface is shown in Figure 20.

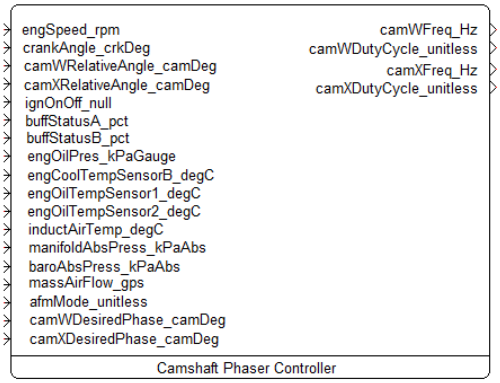


Figure 20. Engine Camshaft Phaser VSIL controller VSI interface block

There are no external or internal state variables to integrate. The camshaft phaser VSIL controller reads the input signals and calculates the output signals whenever the controller detects a time based or event-based task initiator (trigger). The external integrator is notified of each task initiator as a discontinuity point and takes the proper action. The camshaft phaser controller ring does not use any signals from other controllers so no CAN (Controller Area Network) communication signals are included in the VSiL controller.

10 Connecting VSI Interface Models

VSI interface models can be connected to other signal flow models simply by connecting the input and output signals. VSI interface models can be connected to power flow models using ether sensor/actuator pairs or signals to power flow elements as shown in Figure 21.

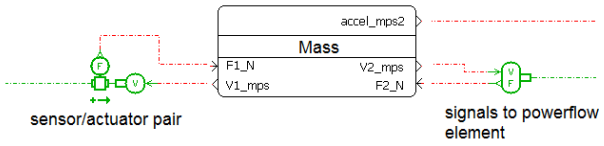


Figure 21. Connection of VSI interface Models

11 Algebraic Loops and Implicit State Variables

The VSI interface is intended to produce the same set of equations as the source models, thus if the source model when combined into a feedback system produces algebraic loops or implicit state variables, the VSU model will likewise produce algebraic loops or implicit state variables. Conversely, if the source model when combined into a feedback system does not produce algebraic loops or implicit state variables, the VSU model will likewise not produce algebraic loops or implicit state variables.

The VSI interface will not solve systems containing algebraic loops or implicit state variables. The VSI interface is designed to handle systems with ODEs (and possibly DAEs in the future).

12 Why the VSI interface Requires No Run-Time Licenses

The reason why the VSI Interface requires no run-time licenses for any model packaged using the interface is so that system simulation cost will be comparable to single component and/or sub-system simulation cost. This also ensures that all models packaged with the VSI interface will run in any simulation package that implements the interface. Software venders can and do charge extra for the FMU generation feature and this is still allowed in the VSI specification.

13 Application Areas for VSI Interface

The intended application areas for the VSI interface are shown in Table 3.

Table 3. VSI Intended Application Areas

Area #	Application Area	Intended for VSI Interface
1	Plant Models for HIL bench	No
2	Plant Models for MIL & SIL	Yes
2	Continuous Time Behavioral Controllers MIL	Yes
3	Discrete Time Ring Level Controllers for SIL & VSiL	Yes
4	Discrete Time Full Controllers for SIL	No

14 Conclusions

The VSI interface is presented in this paper as a high-level overview and will require additional work to formalize its content into a usable specification. Future development of the VSI interface will be dependent on the feedback from the FMI developers and the model coupling and co-simulation user community as to whether these types of enhancements are of significant value for the type of simulations being done for product development in their specific industries.

Acknowledgements

The authors would like to acknowledge the original developers of the Virtual Systems in-the-Loop (VSiL) technology and the Autonomie software which led to the development of the VSI interface.

References

Ronald C. Rosenberg and Dean C. Karnopp. Introduction to Physical System Dynamics, McGraw-Hill, 1983.

Shane Halbach, Phillip Sharer, Sylvain Pagerit, Aymeric P. Rousseau and Charles Folkerts. Model Architecture, Methods, and Interfaces for Efficient Math-Based Design and Simulation of Automotive Control Systems. SAE 2010-01-0241, SAE World Congress, Detroit, April 2010.

Tim Glaue. Virtual Systems-in-the-Loop (VSiL): System Modeling for Quality Controls Integration. 2006 Simcenter Amesim European Users Conference, March 30, 2006.

Vladimir A. Dobrushkin, Applied Differential Equations: The Primary Course, Mathematica Tutorial for the First Course. Part III: Heun Methods, CRC Press, 1st Edition, 2014.

Timoshenko Beam based Coupled Torsion Beam Axle Modeling Method

Minsoo Hyun[1] Deaoh Kang[1] Seung-jin Heo[2] Jingyu Moon[2]

[1]Graduate School of Automotive Engineering, Kookmin University, Korea, slay@kookmin.ac.kr
[1]Institute of vehicle Engineering, Korea, bigfive@ivh.co.kr
[2]School of Automotive Engineering, Kookmin University, Korea, sjheo@kookmin.ac.kr
[2]Institute of vehicle Engineering, Korea, jkmoon@ivh.co.kr

Abstract

This paper investigates the coupled torsion beam axle (CTBA) modeling method based on Timoshenko beam theory. Timoshenko beam elements are applied to torsion beams that cause large deformations, and rigid bodies are applied to parts such as spring mounts, trailing arms and knuckles. Timoshenko beam modeling was modeled using the modelica language, and the remaining components and test rigs were modeled using Modelon's Vehicle dynamics library (VDL). To evaluate the accuracy of the model constructed in this way, the opposite wheel travel simulation is performed and the results are compared with the CTBA model based on the full flexible body. As a result of the test, the error of the roll stiffness and the trailing arm axial force are all within 3%.

Keywords: Timoshenko Beam Theory, Coupled Torsion Beam Axle(CTBA), Roll Stiffness

1 Introduction

Coupled torsion beam axle (CTBA) is a suspension in which the centrally located torsion beam for the opposite wheel travel generates roll torque. Generally used for rear suspension of small vehicles. This CTBA is mainly applied to the flexible body model because of the torsion beam causing large deformation. However, the flexible body model is inefficient when designing hardpoints and torsion beam properties because the model can be modeled only after the shape is defined.

This paper proposes a Timoshenko beam-based CTBA dynamic model with hardpoints and torsion beam properties as modeling parameters. In Section 2, we introduce the 6 DOF(degree of freedom) Timoshenko beam modeling method based on Modelica language and propose CTBA dynamic modeling method using it. Section 3 describes the opposite wheel travel simulation, an example of the Timoshenko beam based CTBA model, and Section 4 evaluates the accuracy by comparing the simulation results with those of Adams / car's flexible body CTBA model.

2 Timoshenko Beam based CTBA Model

2.1 Timoshenko Beam Model

The 6 DOF Timoshenko beam model used in the dynamics model is shown in Eq. (1).

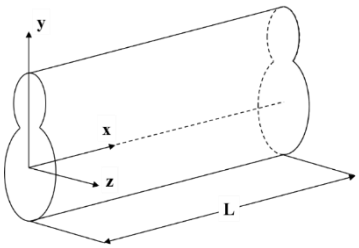


Figure 1. Coordinate System of Beam Model

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ T_x \\ T_y \\ T_z \end{bmatrix} = -\begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_{zz}}{L^3(1+P_y)} & 0 & 0 & 0 & \frac{-6EI_{zz}}{L^2(1+P_y)} \\ 0 & 0 & \frac{12EI_{yy}}{L^3(1+P_z)} & 0 & \frac{6EI_{yy}}{L^2(1+P_z)} & 0 \\ 0 & 0 & 0 & \frac{GI_{xx}}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI_{yy}}{L^2(1+P_z)} & 0 & \frac{4EI_{yy}}{L(1+P_z)} & 0 \\ 0 & \frac{-6EI_{zz}}{L^2(1+P_y)} & 0 & 0 & 0 & \frac{4EI_{zz}}{L}(1+P_y) \end{bmatrix} \begin{bmatrix} dx - L \\ dy \\ dz \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} - C \begin{bmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (1)$$

P_y and P_z are factors for considering shear deformation. This equation was compared with the results of Euler-Bernouli beam, finish element model by creating a model using Modelica language and bending simulations.

Table 1. Bending test results of beam models

Input force [N]	Beam length [mm]	Displacement [mm]		
		Euler-Bernoulli	Timoshenko	FEM
1200	500	0.3575	0.3710	0.3710
	1000	2.8599	2.8869	2.8869
	1500	9.6521	9.6926	9.6926

As a result of comparison, as the beam length gets shorter, the Euler-Bernoulli beam has little error with the Timoshenko beam, which has a larger error with the finite element model. This shows that Timoshenko beams are more suitable for beams with small aspect ratios.

2.2 Torsion Beam Modeling

The torsion beam used in CTBA is divided into constant region with constant cross section and transition region with gradually changing cross section as shown in Figure 2. To realize this, in this paper, the constant region consists of one beam element and the transition region represents the change of the cross section by applying five beam elements to each of the left and right sides. Since the transition region is short, a Timoshenko beam of Section 2.1 is suitable because a beam element with a small aspect ratio is used.

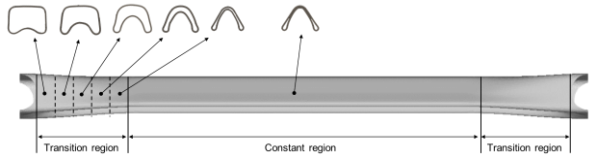


Figure 2. Regions of Torsion Beam

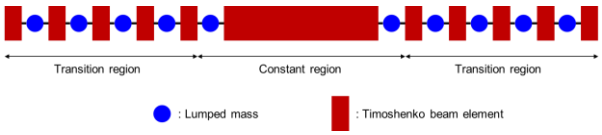


Figure 3. Timoshenko Beam based Torsion Beam Model

The torsion beam model in Figure 3 is implemented in Modelica language as shown in Figure 4. The red model is the Timoshenko beam element and the blue model is the lumped mass.

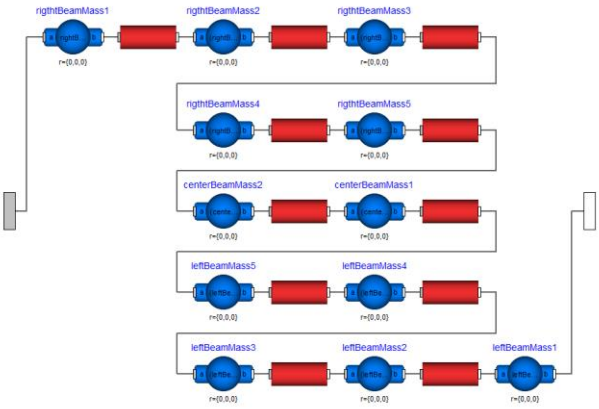
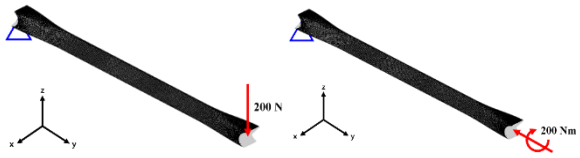


Figure 4. Modelica based Torsion Beam Model

To compare the developed torsion beam model with the FE (finite element) torsion beam model, static bending test and static torsion test were conducted. The bending test and torsion test were conducted by fixing one side of the torsion beam as shown in Figure 3 and applying a force of 200N downward and 200Nm torque in the axial direction, respectively. As shown in Table 4, all test results showed the accuracy within 5%.



a) Static Bending Test b) Static Torsion Test

Figure 5. Static Test

Table 2. Torsion test results of beam models

Test	Output variable	Timoshenko torsion beam model	FE torsion beam model	Error
Bending Test	Displacement [mm]	1.2024	1.2486	3.7%
Torsion Test	Angle [°]	2.2551	2.3592	4.4%

2.3 CTBA Modeling

The torsion beam model constructed in Section 2.1 is combined with the remaining parts to form the CTBA model. First, the components of the trailing arm, knuckle, spring lower mount, etc. are configured as a rigid body model so that the hardpoints of the CTBA can be directly changed. Force elements such as springs, dampers, bushes, and bump stoppers form the model based on the parameters in Table 3. Figure 4 shows the major components of the CTBA model. Modeling was done using Modelon's Vehicle Dynamics Library (VDL).

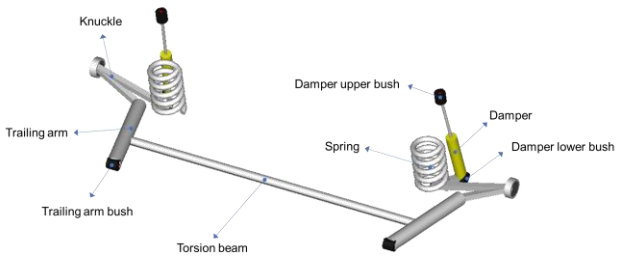


Figure 6. Components of CTBA Model

Table 3. Parameters of CTBA Model

Category	Parameters	Units
Hardpoints	Wheel center	mm
	Trailing arm mount	
	Damper upper mount	
	Damper lower mount	
	Spring upper mount	
	Spring lower mount	
	Torsion beam mount	
Spring	Stiffness	N/mm
	Preload	N
Damper	Force-Velocity table	-
Bump stopper	Force-Displacement table	

DOI 10.3384/ecp202017423

Bush	6 DOF stiffness table 6 DOF damping coefficient	

3 CTBA Multibody Dynamic Simulation

In this paper, the opposite wheel travel simulation is performed to verify the accuracy of the proposed Timoshenko beam CTBA model and to proceed with the application case. The opposite wheel travel is a test that generates roll torque by inputting vertical displacement in the opposite direction to the left and right wheels. The test rig is constructed by connecting an actuator that can input vertical displacement to the wheel center of the CTBA model, as shown in Figure 5. The input profile used in this paper is shown in Figure 6.

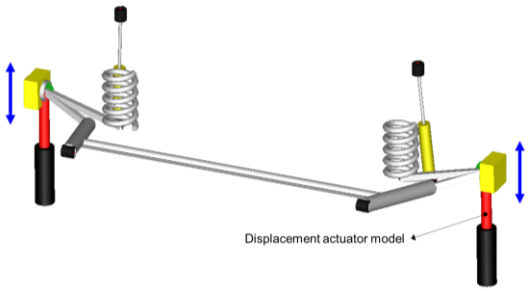


Figure 7. Opposite Wheel Travel Test Rig

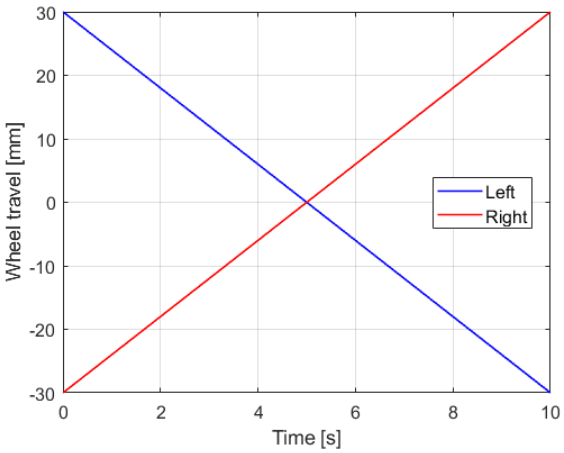


Figure 8. Input Signal of Opposite Wheel Travel

4 Validation Result

In the opposite wheel travel simulation, the roll torque of the suspension is compared with the axial force of the trailing arm. Figure 7 shows the output curves of the two models for roll angle. Table 4 shows the error of roll stiffness value, peak to peak value of trailing arm axial force of the two models.

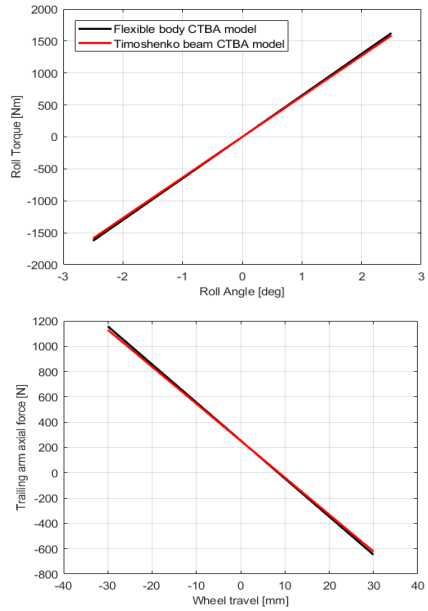


Figure 9. Opposite Wheel Travel Simulation Results (top: roll torque, bottom: trailing arm axial force)

Table 4. Opposite Wheel Travel Simulation Results

	Timoshenko beam CTBA	Flexible body CTBA	Error
Roll stiffness [Nm/deg]	635.07	650.18	2.32 %
Peak to Peak value of trailing arm axial force [N]	1751.94	1804.28	2.90 %

The root mean square(RMS) error between the output values of both models was within 3%. This demonstrates that the accuracy of the Timoshenko beam CTBA model proposed in this study is close to that of the flexible body CTBA model.

5 Conclusion

In this paper, we developed a Timoshenko beam based CTBA model that can calculate the dynamic response of CTBA according to hardpoints and torsion beam properties. The model was created using the Modelica language and Modelon's Vehicle Dynamics Library (VDL).

Modeling method is as follows.

- The torsion beam section of the CTBA consists of Timoshenko beam elements. The rest of the parts, such as the trailing arm and the knuckle, were applied with the rigid body. In this way, the model is configured to have hardpoints and torsion beam properties as variables.

- A total of 11 Timoshenko beam elements were used to implement the transition region of the torsion beam section.

To evaluate the accuracy of the constructed Timoshenko beam CTBA model, opposite wheel travel simulation was performed. The model compared is the flexible body CTBA model. The root mean square error between the output values of the two models was high within 3%. In this paper, the opposite wheel travel simulation is a quasi-static test, and the verification of the dynamic test has not been performed. In the future, we will also verify dynamic tests such as virtual testing labs(VTL) and full vehicle based R&H(Ride and Handling) simulation.

References

Vehicle Dynamics Library, Modelon, 2019.

Chen,J., Jiang, Y. Qin, M. et al. (2015). CAD/CAE and Optimization of a Twist Beam Suspension System. *SAE Technical Paper Series.*

Chen,J., Jiang, M. Jiang, Y. et al. (2015). Modeling, analysis and optimization of the twist beam suspension system. *SAE International Journal of Commercial Vehicles 8(2015-01-0623):38-44.*

Choi, B., Choi, D., Min, J. et al. (2009) Torsion beam axle system design with a multidisciplinary approach. *International Journal of Automotive Technology 10(1):49-54.*

CFichera, G., L:acagnina, M., Petrone, F. (2004) Modelling of torsion beam rear suspension by using multibody method. *Multibody system dynamics 12(4):303-316.*

DOI 10.3384/ecp202017423

Vehicle Dynamics Model With Non Linear Bush Model and Tire Filter for Ride Comfort Analysis

Jae-hun Jo[1] Dae-oh Kang[1] Gwang-woo Lee[2] Seung-jin Heo[2]

[1]Institute of vehicle Engineering, Korea, jhjo@ivh.co.kr
[1]Institute of vehicle Engineering, Korea, bigfive@ivh.co.kr
[2]Graduate School of Automotive Engineering, Kookmin University, Korea, gwlee@kookmin.ac.kr
[2]School of Automotive Engineering, Kookmin University, Korea, sjheo@kookmin.ac.kr

Abstract

There are many non-linear factors affecting ride comfort. In this study, various nonlinear factors affecting vehicle vertical dynamics are modeled to ensure robustness in the analysis of ride comfort performance.
Another factor that affected the interpretation of ride comfort performance is road model. To simulate driving situation of vehicle, a Tire Filter Model is presented.
Conclusion In this paper, the robustness of vehicle dynamics model with nonlinear elements and ride comfort performance analysis model is secured through Tire Filter Model.

Keywords: Vehicle Dynamics Model, Ride Comfort, Non-Linear Bush Model, Hydro Mount

1 Introduction

Nonlinear dynamic factors affecting vehicle's ride comfort performance are as follows. Engine Mount, Spring, and Damper. In this study, we constructed Vehicle Dynamics Model, including these nonlinear dynamic factors, and analyzed how it affected the ride comfort analysis.
Also, use Tire Filter to calculate Drive Signal used in 4-post simulation. Tire Filter calculates effective road profile that affects Vehicle by considering the deformation of Tires.
Using Vehicle Dynamics Model and Tire Filter, which contain the above nonlinear dynamics, the ride comfort analysis improves accuracy.

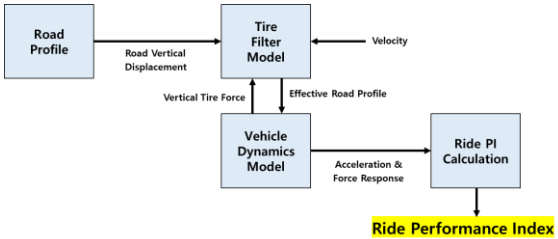


Figure 1. Ride Comfort Analysis Process

2 Vehicle Dynamics Model

Nonlinear dynamic factors such as Engine Mount, Spring, and Damper influence analysis of ride comfort performance. Therefore, Vehicle Dynamics Model was modeled to reflect the above nonlinear dynamic factors.

2.1 Engine Model

Engine Model is a 1D model based on Torque Map.

2.1.1 Engine Mount Model

Engine Mount Model used Hydro Mount Model. Hydro Mount Model reflects mass and rotational inertia values of fluid and calculates time response and stiffness of the translational /rotational direction.

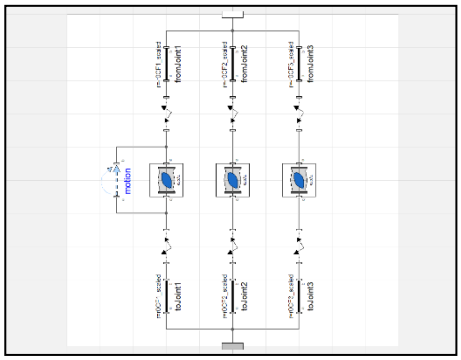


Figure 2. Hydro Engine Mount Model

2.2 Suspension Model

Suspension Model used TEKS Model.

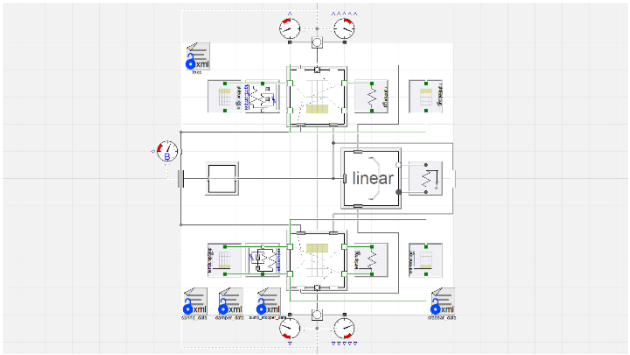


Figure 3. TEKS Model

2.2.1 Spring / Damper Model

Spring models used linear models. On the other hand, Damper Model used Spencer Bouc Wen Model.

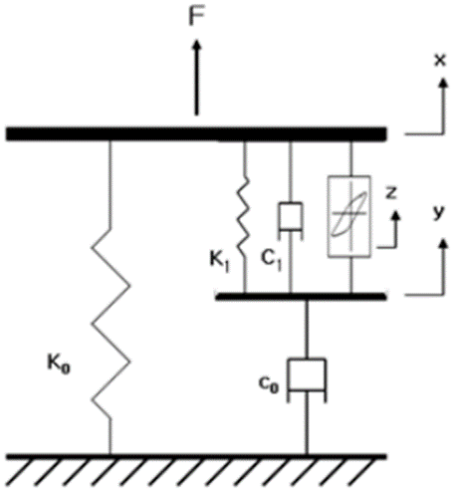


Figure 4. Spencer Bouc Wen Model

Figure 4 illustrates the concept of Spencer Bouc Wen Model. The models modeled through Modelica are shown in Figure 5.
Using the Spencer Bouc Wen Model, we can accurately predict the Hystersis phenomenon of Damper, thereby improving the accuracy of the Ride Comfort analysis.

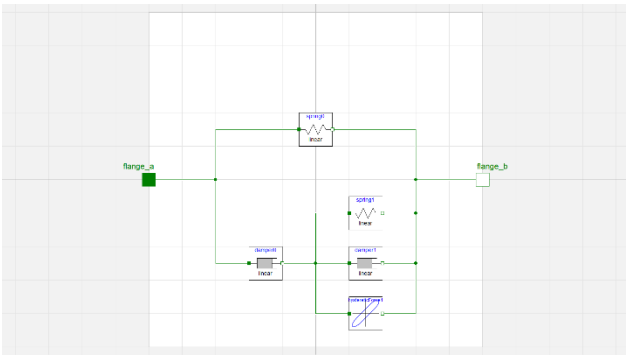


Figure 5. Spencer Bouc Wen Modelica Model

3 Tire Filter Model

Actual Road Profile is different from the input to the vehicle. Therefore, the Ride Comfort performance analysis should be performed using the Road Profile that is delivered to the actual vehicle.
To evaluate the ride comfort of an irregular road surface, Tire Filter model predicts deformation by considering the characteristics of Tire and calculates Effective Road Profile that affects actual vehicle.

3.1 Tire Enveloping Model

Based on the vertical load, speed and road profile of the tyre, Wheel Vertical position is predicted.

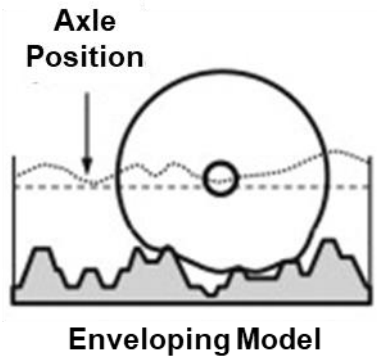


Figure 6. Tire Enveloping Model

3.2 Road Perception Model

Calculate required road profile that can be utilized in the analytical model for wheel specific position predicted by Tire upgrade model.

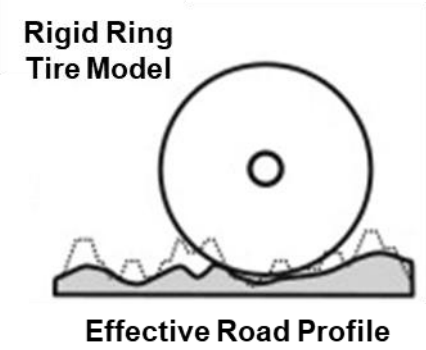


Figure 7. Road Perception Model

Effective road height is determined by Formula (1).

$$-W_e(X) = \frac{Z_f(X_f) + Z_r(X_r)}{2} - b_e \quad (1)$$

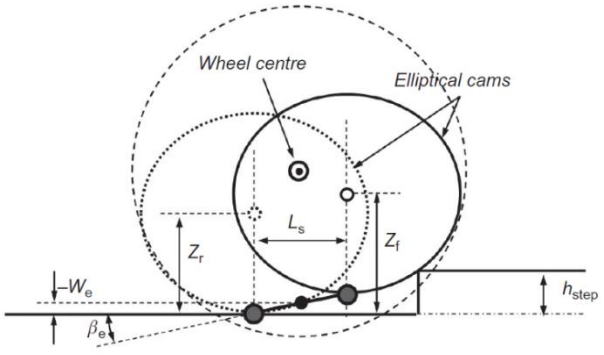


Figure 8. Effective Road Height Calculation

3.3 Tire Filter Result

Using Tire Filter to calculate the Road Profile, the results are as shown in Figure 9. This results can be used for 4-post simulation.

DOI
10.3384/ecp202017427

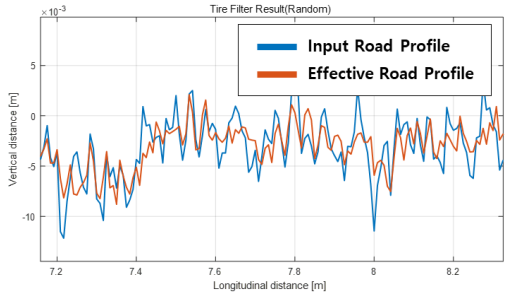


Figure 9. Tire Filter Result

4 Simulation

4.1 Performance Index

Ride comfort Performance Index is defined as the commonly used Bounce, Wheel Hopping, Choppy, and the Ride Filter where values are calculated by weighting the acceleration signals at each location.

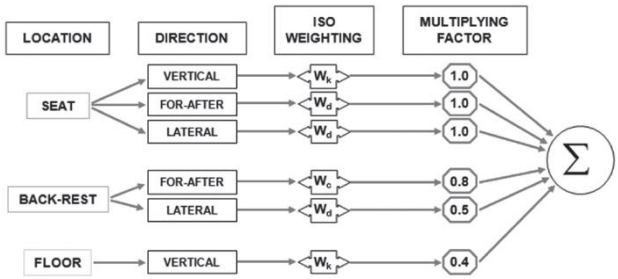


Figure 10. Ride Filter

4.2 Simulation Result

4-post simulation was performed by entering the Random Road Profile as Drive Signal in vehicle. To verify the usefulness of the dynamic features of vehicle, the results of the analysis of ride comfort performance of the vehicle model with the engine mount removed were compared. The comparison result is a picture 11. As shown in , a vehicle model with a dynamic feature has a smaller tolerance to field test compared to a vehicle model that does not.

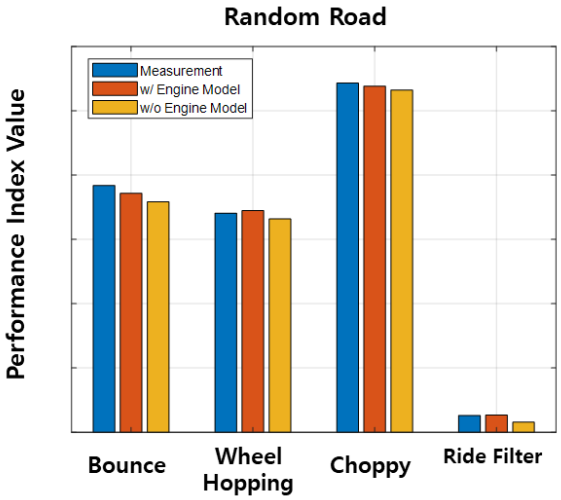


Figure 11. Engine Mount Verification Result

Table 1. Engine Mount Verification Result

Performance Index	With Engine Mount	W/O Engine Mount
Bounce	3.20%	6.65%
Choppy	1.14%	2.66%
Wheel Hopping	0.87%	1.97%
Ride Filter	2.03%	40.29%

Similarly, to verify the usefulness of tire filter, the results of the 4-post simulation using the actual road profile were compared with the results of the 4-post simulation using the efficient road profile calculated through tire filter. The comparison result is a picture 11. While other performance indexes do not have a significant effect on tire filter, for the Ride Filter, we can see that the model used in tire filter is similar to the field test results.

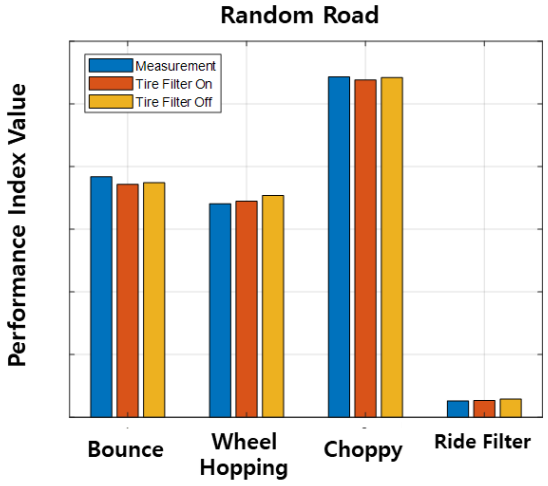


Figure 12. Tire Filter Verification Result

d

Table 2. Tire Filter Verification

Performance Index	Tire Filter On	Tier Filter Off
Bounce	3.20%	2.51%
Choppy	1.14%	3.87%
Wheel Hopping	0.87%	0.16%
Ride Filter	2.03%	11.07%

5 Conclusion

In this study, vehicles with modelica based dynamic features were modeled. Dynamic element models are Hydro Engine Mount, Damper (Spencer Bouc Wen Model). In addition, the tire filter was modeled to predict road profile to be delivered to vehicle.
A 4-post simulation was conducted to verify the nonlinear dynamic model and tire filter. Simulation results showed that models with nonlinear dynamics were more similar to field test results. When calculating road profile using tire filter, the results of field test were more similar than those of the other road profile.
Analysis has been achieved through the vehicle dynamic model with non-linear dynamics component model and tire filter model, this will be applicable to the vehicle's design process.

References

Vehicle Dynamics Library, Modelon, 2019.

Phenomenological Model of a Magnetorheological Damper. B.F. Spencer Jr., S.J. Dyke, M.K. Sain and J.D. Carlson. The ASCE Journal of Engineering Mechanics, 1996.

Vehicle Dynamics – theory and application, Reza N. Jazar, Springer, 2015.

A Semi-Empirical Three-Dimensional Model of the Pneumatic Tyre Rolling over Arbitrarily Uneven Road Surfaces, Schmeitz, A.J.C. Mechanical Maritime and Materials Engineering, 2004.

Ridemeter – Calculated Ride Comfort, Detlef K. Kudritzki, SAE Technical Paper 2007-01-2388, 2007.

DOI 10.3384/ecp202017427

Reinforcement Learning for Thermostatically Controlled Loads Control using Modelica and Python

Oleh Lukianykhin[1] Tetiana Bogodorova[2]

[1]The Machine Learning Lab, Ukrainian Catholic University, Ukraine, lukianykhin@ucu.edu.ua
[2]Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA
bogodt2@rpi.edu

Abstract

The aim of the project is to investigate and assess opportunities for applying reinforcement learning (RL) for power system control. As a proof of concept (PoC), voltage control of thermostatically controlled loads (TCLs) for power consumption regulation was developed using Modelica-based pipeline. The Q-learning RL algorithm has been validated for deterministic and stochastic initialization of TCLs. The latter modelling is closer to real grid behaviour, which challenges the control development, considering the stochastic nature of load switching. In addition, the paper shows the influence of Q-learning parameters, including discretization of state-action space, on the controller performance. [1]

Keywords: Modelica, Dymola, Open AI Gym, JModelica.org, OpenModelica, Python, Reinforcement Learning, Q-learning, Thermostatically Controlled Loads, Power System, Demand Response

1 Introduction

Despite of the successful application in the past, classic methods and solutions in power systems are not capable to handle new challenges. In particular, when stability margins have decreased due to stochastic behaviour of renewable energy sources that increase presence in the power grid (Begovic et al., 2001). In addition, the rise of IoT-related technologies contributed to appearance of distributed smart grid (Ipakchi and Albuyeh, 2009). These challenges require and allow for new solutions, one of those is an application of reinforcement learning algorithms for controller design.

The reinforcement learning learns from the interaction of a controller (agent) with a system (environment). Examples of successful applications for complex tasks in various domains are winning complex games (Silver et al., 2016; Vinyals et al., 2019), and pretraining robots for performing different tasks (Riedmiller et al., 2009). In (Ernst et al., 2008), authors have shown on an electrical power oscillations damping problem that RL can be competitive with classic model-based methods, even when a good analytical model of the considered system is available. This lets to be optimistic about RL application for ancillary services.

Ancillary services that involve a relatively small amount of energy to change power consumption rely on the capacity that is held in reserves, including thermostatically controlled loads (Ma et al., 2013), (Heffner, 2008). The authors in (Claessens et al., 2018) applied fitted Q-iteration RL method to obtain a performance within 65% of a theoretical lower bound on the cost for a district heating network of 100 TCLs. This allows to be optimistic about RL application to optimisation of control that is applied to TCLs using other constraints, i.e. aiming demand-supply balance, not the cost optimisation.

To allow learning of the optimal control policy, interaction with a controlled system is required to gather experience for RL agent training. However, training an agent using a real power grid is not an acceptable option for an early stage of the development. Thus, the behaviour of a real power system is simulated with models developed in the Dymola environment using the Modelica language as an open access standard. The chosen power system model as a feeder of a number of TCLs represents the behaviour of interest that corresponds to properties of a real power grid. Specifically, being the most common type of load in the distribution grid thermostatically controlled loads (TCLs) can serve as means providing an ancillary service (Kirby and Hirst, 1999; Heffner, 2008; Meyn et al., 2015; Zhang et al., 2012). In (Tindemans et al., 2015) it was shown that an analytical approach to the controller development can be successful in demand response of thermostatic loads. One can achieve modulation of the power consumption of a heterogeneous set of TCLs according to a reference power profile.

In (Bogodorova et al., 2016), it was shown that voltage control-based ancillary services can be utilized using TCLs thermal capacity when regulating the power consumption using a voltage signal. However, the chosen constant control served as proof of concept. Therefore, more sophisticated control was mentioned as a future research direction. To expand this research, this paper focuses on the development of more complicated robust control. To find the required optimal voltage change-based control policy, reinforcement learning methods were applied.

The authors in (Moriyama et al., 2018) achieved 22%

[1]Experiment pipeline, procedure, full results, visualizations and analysis are available at https://github.com/OlehLuk/rl-power-control

improvement in energy consumption compared to a model-based control of the data centre cooling model using deep RL techniques. In (S.Mottahedi, 2017) the researcher applied Deep Reinforcement Learning to learn optimal energy control for a building equipped with battery storage and photovoltaics. In these cases, a reinforcement learning agent was trained and tested using power system model simulation developed using Modelica tools. Possibility of coupling reinforcement learning and Modelica models was validated in (Lukianykhin and Bogodorova, 2019), where the authors built a pipeline for RL agents that are training in environments simulated with Modelica-compiled FMUs.

In (Ruelens et al., 2016), the authors succeeded to reduce the total cost of energy consumption of the single electric water heater by 15% in a 40-days experiment. In addition, the authors emphasized the importance of proper state space discretization and dimensionality reduction, while using autoencoder for this purpose. Moreover, in the review (Vázquez-Canteli and Nagy, 2019) of applying mostly single agent methods and simplest algorithms such as Q-learning for demand response, the authors detected a tendency for better results, when action-state discretization and dimensionality reduction of action-state representations are applied. Although the authors investigated around 150 works, many of which applied control to TCLs, the control goal was mainly in cost reduction.

This paper describes an application of reinforcement learning to develop a voltage controller to manage power consumption of thermostatically controlled loads as part of ancillary services. The power system is modelled using Modelica that was integrated within ModelicaGym toolbox (Lukianykhin and Bogodorova, 2019) with Q-learning algorithm to train the controller. This integration was done using Python and allows utilization of SOTA RL algorithms via OpenAI Gym. The following project achievements are presented in the paper as contributions:

- An optimal control for the considered voltage controller design (see Section 2.1) was learnt using Q-learning algorithm. The algorithm's hyperparameters tuning was done to improve its performance. The achieved performance is comparable to optimal constant control chosen using a simulation of a whole time interval.

- The developed controller shows the capability to generalize: to perform well on testing intervals longer than training.

- The experiment setup configurations were tuned to investigate particularities of the developed controller. These include changes of length and start time of a simulation used for training and testing a controller; setting of reference power level profiles.

- Dependency of controller performance on different environment state discretization strategies was investigated. These smart discretization strategies account for problem formulation and gathered historical data. The optimal width of bin detection method was tested. The dependency of the controller performance on the number of bins in equal-width binning was investigated.

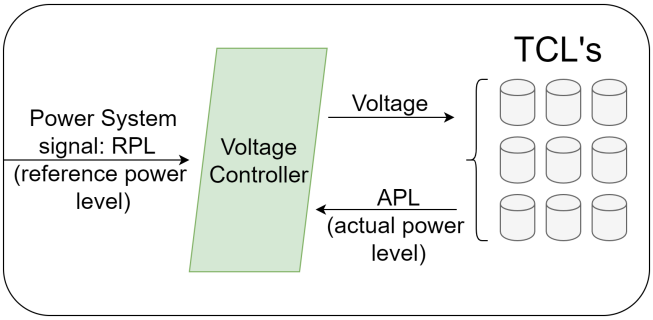


Figure 1. Schematic picture of the considered power system configuration and controller to be developed

The remainder of this paper is structured as follows. Section 2 formulates the research problem and describes a Modelica model for which the problem is formulated. Section 3 describes the approach to the controller development that was employed in experiments, including specific details of the utilized RL algorithm. It is followed by Section 4 that describes conducted experiments, results that are supplemented with discussion. Paper is finished by a short summary of the presented results together with conclusions and discussion of possible directions for future work in Section 5.

2 Problem Formulation

More specifically, the considered applied problem can be formulated as follows: given a grid model with multiple TCLs and means for voltage control, an optimal control strategy that controls power consumption at the point of common coupling should be found using RL (see Figure 1).

The controller is placed in the point of common coupling of TCLs and controls the voltage at the substation. This way, sufficient load change can be achieved by varying voltage in a small range when satisfying power grid constraints (Bogodorova et al., 2016). Detailed controller design description can be found in (Bogodorova et al., 2016).

The control goal is to make actual power consumption close to the reference power profile by changing the voltage on the bus. The voltage controller has one control parameter - proportional coefficient k. A reinforcement learning agent should choose its value to change the voltage, taking into account current values of actual and reference power levels. The control action is changed at discrete time steps of the considered time interval.

Mean squared error between actual and reference power levels is chosen as a measure of control strategy performance. This choice aimed to encourage control policies that avoid big differences between actual and reference power levels (APL and RPL, respectively). APL and RPL

DOI 10.3384/ecp202017431

are sampled with the same time steps as control action is changed.

2.1 Model

The considered power system model setup includes a feeder of 20 TCLs, tap changer, proportional controller that were modeled using Modelica language and Dymola environment (see Figure 2). FMUs were compiled in JModelica, while OpenModelica was used for model diagram visualization.

Each TCL has the same thermal resistance $R = 200°C/kW$, power consumption $P = 0.14$ p.u, ambient temperature $\theta_a = 32°C$, switching temperature range of $[19.75..20.25]°C$ and two variables: *switch* indicating if TCL is on (0) or off (1) and temperature θ. TCLs ($n = 20$) differ in value of thermal capacitance C. Value of C for i-th TCL is the i-th value in the array [2.0, 2.2286, 2.4571, 2.6857, 2.9143, 3.1429, 3.3714, 3.6, 3.8286, 4.0571, 4.2857, 4.5143, 4.7429, 4.9714, 5.2, 5.4286, 5.6571, 5.8857, 6.1143, 6.3429]. Since thermostats in the grid are heterogeneous in their characteristics and start their operation at random moment of time, they are modeled to be initialized both deterministically (Equation 1) or stochastically (Equation 2).

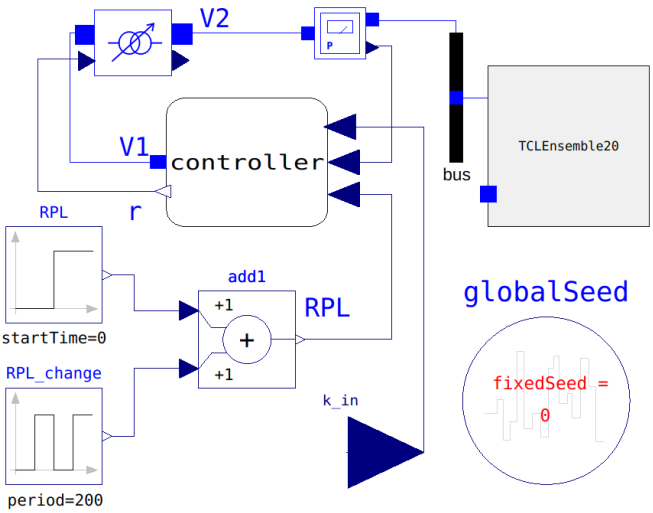


Figure 2. OpenModelica diagram of a simulated power system (20 TCLs)

For the deterministic case differential equation of an individual TCL:

$$\frac{d\theta}{dt} = \frac{-\theta_a + \theta + R \cdot P}{R \cdot C}, \tag{1}$$

For the stochastic case the stochastic component u is added as an input:

$$\frac{d\theta}{dt} = \frac{-\theta_a + \theta + R \cdot P}{R \cdot C + R \cdot u \cdot range}, \tag{2}$$

where u is a random number in $[0;1]$, $range = 4.5$ - a range of thermal capacitance of $[C, C + range]$.

At the beginning of the simulation TCLs 1-10 are on ($switch = 1$), TCLs 11-20 are off ($switch = 0$). Equation 3 describes *switch* value change. When TCL is on and temperature $\theta > \theta_{max}$ it switches off and when TCL is off and $\theta < \theta_{min}$ it switches on.

$$switch = \begin{cases} 1, & \theta < \theta_{min}, \\ 0, & \theta > \theta_{max} \end{cases} \tag{3}$$

Power consumption by each TCL:

$$P = switch \cdot g_0 \cdot v^2 \tag{4}$$

where v - voltage, g_0 - conductance

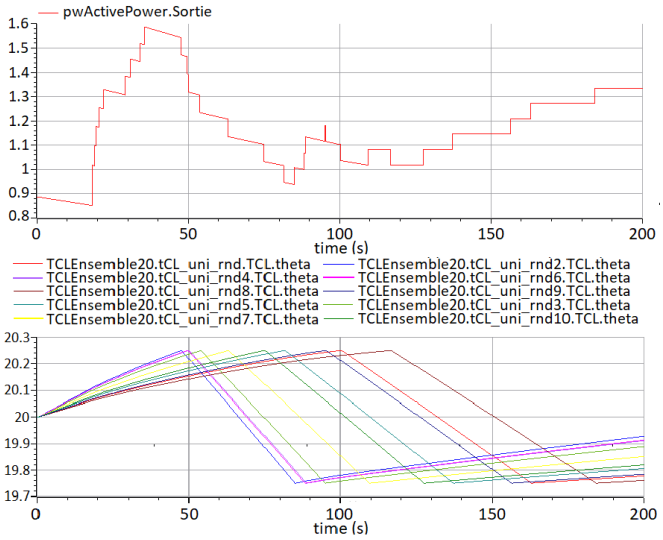


Figure 3. Output of a simulated power system (20 TCLs)

3 Approach to Solution

To investigate and assess opportunities for a voltage controller development using reinforcement learning, case studies should differ by their complexity. Therefore, to study RL controller's behaviour, deterministic TCL parameters initialization case and stochastic initialization were considered.

To verify the scalability and ability to generalize of the controller, several case studies were carried out:

- step down in reference power level (RPL) in the middle of the considered time interval;
- simulation interval for testing is longer than the one used for training;
- beginning of training of the controller at different start time for a simulation: including or excluding transition process at the beginning of a simulation.

Since the transition process is prominent for the particular model (Figure 3), it is valuable to investigate the influence of a transition process on a controller training

and its performance. A step down in RPL in the middle of the considered time interval ($t = 200s$) allows to verify an ability of the RL controller to follow the changing power consumption reference.

Most experiments focus on the case with constant RPL and stochastic TCLs' parameters initialization. This set up allows to validate the RL controller at the early development stage and has a possibility to scale to other model configurations, e.g. changes in RPL.

Q-learning algorithm with straightforward rewarding strategy was chosen to train an RL controller. In addition, binning was utilized as a basic discretization technique to handle continuous state and action spaces.

3.1 Q-learning for Modelica model

The Q-learning procedure that is utilized for RL controller training was presented in (Lukianykhin and Bogodorova, 2019). The procedure is a training of a Q-learning RL agent in an environment simulated with a Modelica model. It is implemented in Python and makes use of OpenAI Gym via ModelicaGym toolbox. It includes executing an action in the environment and processing corresponding feedback to learn optimal control with a Q-learning algorithm. The following hyperparameters have to be tuned for Q-learning algorithm: number of episodes of training, learning rate, exploration rate, exploration rate decay, discount factor, available actions and rewarding strategy. Size of a time step between two consecutive control signal changes is a hyperparameter of the experiment as well. The time step size is set when a simulation of the environment is initialized for the experimental procedure. Size and number of such steps define the length of the time interval simulated in the experiment for controller training and testing. Unless otherwise specified, the considered time interval length is equal to 200 seconds and time step is equal 1 second, the chosen number of training episodes is 100.

In addition, for all experiments, controller training was repeated to gather statistical data for more reliable estimates of a controller's performance and evaluation of corresponding system behaviour. Unless otherwise specified, the number of the repeated experiments equals 5. For the training, a trend in the controller's performance was estimated. To this end, the episode performances were smoothed with a window of size 20 for each repeat and averaged over repeats. The received average smoothed performance that depends on a training episode is a proxy for performance change evaluation during training. For the testing part of each repeat, the performance of the trained controller was estimated by repeated exploitation of a learnt RL-driven control strategy. Unless otherwise specified, the number of test episodes where performance is measured equals 50.

3.2 Binning of a continuous state

Q-learning, like most other RL algorithms, considers discrete environment states and actions available to an agent. On the contrary, real-world applications, like the problem considered in this paper, deal with continuous domains. Thus, there is a strong need in converting continuous spaces to reasonable discrete representation. The approach used for this purpose is called a discretization strategy.

As the considered problem is not high-dimensional, it was decided to focus on the application of a classic discretization strategy - binning. The main idea of binning is to divide continuous space into numbered intervals (bins) and encode data by an index of the bin corresponding to the data point. Different strategies for choosing bins edges can be utilized in binning.

Several strategies for choosing discretization bins were tested:

1. classic equal-width interval splitting;
2. optimal width estimation methods (analogy with histograms);
3. historical data quantiles as bin edges;
4. accounting for the reference power level, when choosing bin edges.

A basic approach to discretization of a continuous variable using bins is to split the space of possible values in bins of equal width. To account for values outside the interval, the most left bin can be half-opened from the left, the most right one can be half-opened from the right. The number of bins is a hyperparameter of this discretization strategy. Width of the bin is found by dividing the length of the interval of possible values by the number of bins.

The second strategy is when an optimal number of bins is estimated. For this purpose, optimal width detection methods were used from the problem of building a histogram. For example, the Freedman-Diaconis optimal bin width estimator (Freedman and Diaconis, 1981) shows robustly reasonable results for large datasets. Besides, historical data is required to use this optimal-width estimation approach.

In third, it was decided to account for the RPL, when discretizing space for APL, by making RPL an edge of a bin. This way, possible RL agent confusion is avoided because all values of APL higher than the RPL are encoded differently from values that are lower. Other bin edges in this method were chosen using the equal-width approach.

Historical data allows to apply one more method for choosing bins: use data quantiles as bin edges. In this case, the obtained bins are not of equal width. This way, less attention is paid to parts of the space with a low number of observations, while providing a detailed representation for regions with a high concentration of observations.

4 Experiments

First, case studies are described and their performance is evaluated. For the deterministic initialization case, both constant and step down in RPL, time steps of $t = 1s$ and $t =$

DOI 10.3384/ecp202017431

5s were considered. For the stochastic initialization results for constant RPL and time step of $t = 1s$ are presented.

Second, Q-learning was applied to find an optimal control policy. For the deterministic case with constant RPL and time steps of $t = 1s$ and $t = 5s$, hyperparameters are tuned, and the influence of parameters change are studied. For the stochastic case with constant RPL, time steps $t = 1s$ and $t = 5s$, previously chosen hyperparameters were utilized. Afterwards, tuning of the hyperparameters for time step $t = 1s$ was continued. The dependency of the controller's performance on RPL and simulation interval length was studied. In addition, step down in RPL for $t = 1s$ setup was used to validate the ability of the controller to generalize by testing it on longer time intervals than training.

Third, case study of different continuous state space discretization strategies is presented, including investigation of dependency of performance on the number of bins in equal-width binning and application of smart discretization strategies described in Section 3.2. In this case, only constant RPL and time step $t = 1s$ were considered.

4.1 Alternative approaches

Alternative approaches were determined to have a reference for the developed RL controller performance evaluation. Their performance was measured and used to evaluate the trained controller. These approaches are no controller in the system (baseline) and optimal constant control applied using the same controller design as for RL-learnt control (here and further called a competing approach).

Table 1. Performance summary (median, mean, std) for the baseline (no control) and the competing approach (optimal constant control) for stochastic TCLs parameters initialization, 1s control change, constant RPL of 1.2 (best results in bold)

Control param. k	Median	Mean	Std
Baseline	0.0546	0.0613	0.0196
0.5	0.0396	0.0421	0.0085
1	0.0427	0.0424	0.0094
2	0.444	0.0447	0.0098
3	0.0366	0.0379	0.0077
4	0.0263	0.0269	0.0053
5	0.0198	0.0208	0.0041
6	0.0154	0.0159	0.004
7	**0.0126**	**0.013**	**0.0043**

Resulted performance for both RPL configurations and time steps of $t = 1s$ and $t = 5s$ are presented in Table 2. Constant control with control parameter value $k = 0.5$ leads to the best performance. However, this case represents only one sample from the possible system realization space. Thus, a more generalized stochastic case should be considered. It was observed that higher values of the error metric for the non-constant RPL case, comparing to the constant one (see Table 2). It can be explained with the fact that applying constant control action is not the best strategy when the RPL is not constant.

For the stochastic initialization system behaviour is not the same under the same control. Therefore, trajectories were sampled and the corresponding performance metric was measured 50 times. Case of a constant RPL and time step of $t = 1s$ is presented in Table 1. However, for each different setup (RPL, time step or time interval) reference for the developed controller evaluation was determined in the same way as in the presented examples.

It is observed that best performing constant actions also correspond to the lowest variance of the measured performance. In addition, an optimal for this setup control parameter value $k = 7$ is different from the one for deterministic initialization case. The MSE for step down in RPL is higher compareded to the constant RPL case.

The received controller's performance is compared with the competing approach which is times better performing than the baseline. Distribution of performance is taken into account during each comparison, when setup with a stochastic initialization is considered.

4.2 Q-learning

4.2.1 Deterministic Case

As a first step, Q-learning application experiments were run on the deterministically initialized model. The optimal hyperparameters search was performed resulting in the parameters set that is used by default for all further experiments. The search was done by fixing all parameter values, but changing one or several parameters that are tuned. Optimal hyperparameters values for the deterministic case are given in Table 3. In some cases influence of a parameter on the performance was clear, while in exploration parameters variation experiment it wasn't obvious. The average smoothed episode performance is summarized in Figure 4. In Figure 4 small exploration rate with big exploration rate decay is an obstacle for training to converge. At the same time, difference between two other parameters

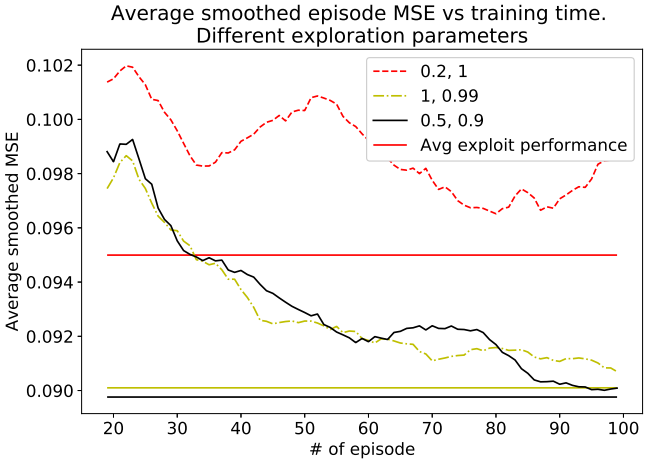


Figure 4. Experiment to change of exploration parameters. Trend in performance during training (average smoothed MSE vs training episode)

Table 2. Performance of the baseline and the competing approach for deterministic TCL parameters initialization case (best results in bold)

	Constant RPL(1.2 p.u.)		*Step down in RPL(1.4-1.1 p.u.)*	
Value of a control parameter k	Time step 1s, *MSE*	Time step 5s, *MSE*	Time step 1s, *MSE*	Time step 5s, *MSE*
Baseline	0.1465	0.1382	0.2055	0.1983
0.5	**0.0806**	**0.0767**	**0.1145**	**0.1138**
1	**0.0806**	**0.0767**	0.1147	0.116
2	0.0909	0.0896	0.1407	0.1494
3	0.0917	0.0789	0.1395	0.1299
4	0.0910	0.0962	0.1394	0.1531
5	0.09	0.0834	0.1405	0.1298
6	0.0913	0.1	0.1350	0.1418
7	0.0894	0.1	0.1334	0.1402

Table 3. Optimal hyperparameters for the deterministic case

Parameter	Chosen value
Learning rate	0.5
Exploration rate	0.5
Exploration rate decay	0.9
Discount factor	0.6
Actions	[0.1, 0.5, 1, 2, 7]
Reward	squared error scaled by -1000

configuration is hard to observe, because training process looks almost the same. The average performances of the controller during testing are close to each other, difference in average MSE is less than 0.001.

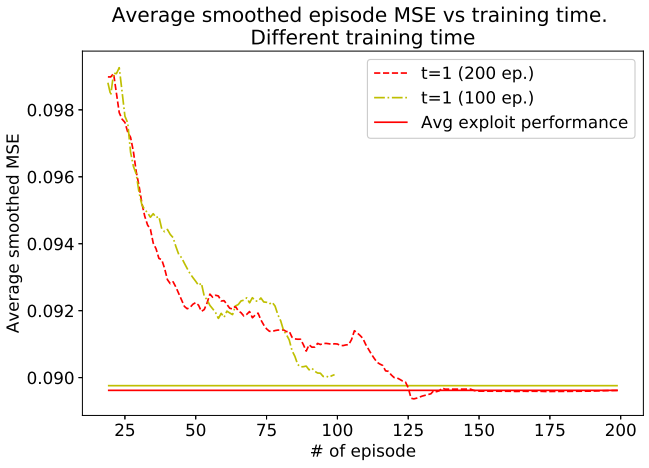


Figure 5. Optimal hyperparameters, deterministic initialization experiment. Trend in performance during training (average smoothed MSE vs training episode)

After choice of the optimal hyperparameters, experiment was run for longer training time - 200 episodes (see Figure 5). Longer training with the given parameters for time step of size $t = 1s$ and $t = 5s$ between control signal change is not leading to improvement in performance (see Figure 5). The RL agent converges to a suboptimal strategy, because performance is lower than for a constant control action $k = 0.5$. This can be caused by particularities of the system realization, as it is deterministicaly initialized being a single realization from the space of all possible system realizations. For generalization and effective agent's learning, experiments were continued on the stochastic case.

4.2.2 Stochastic Case

First, to introduce a stochastically initialized model, Q-learning experiments were performed for the constant reference power level. Second, step down in the reference power level was introduced at the half of considered time interval. For the both cases, experiments were run with the Q-learning utilizing optimal hyperparameters chosen on the deterministic case. These parameters served as an initial inference about optimal hyperparameters for the stochastic case. As a result, training of the controller converges (see Figure 6 for visualization for the constant RPL case).

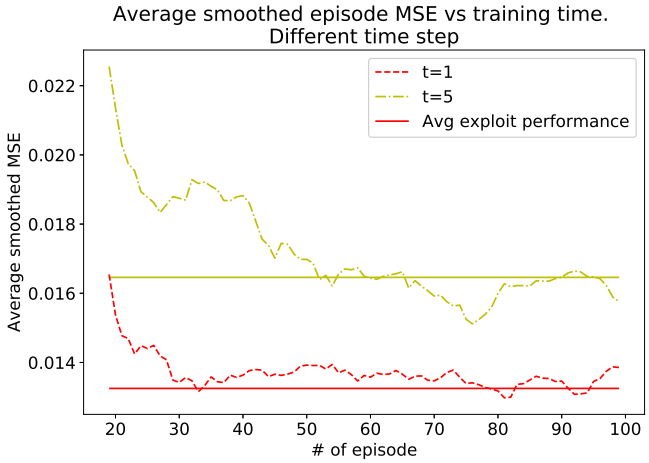


Figure 6. Inferred optimal hyperparameters, stochastic initialization experiment. Trend in performance during training (average smoothed MSE vs training episode)

DOI
10.3384/ecp202017431

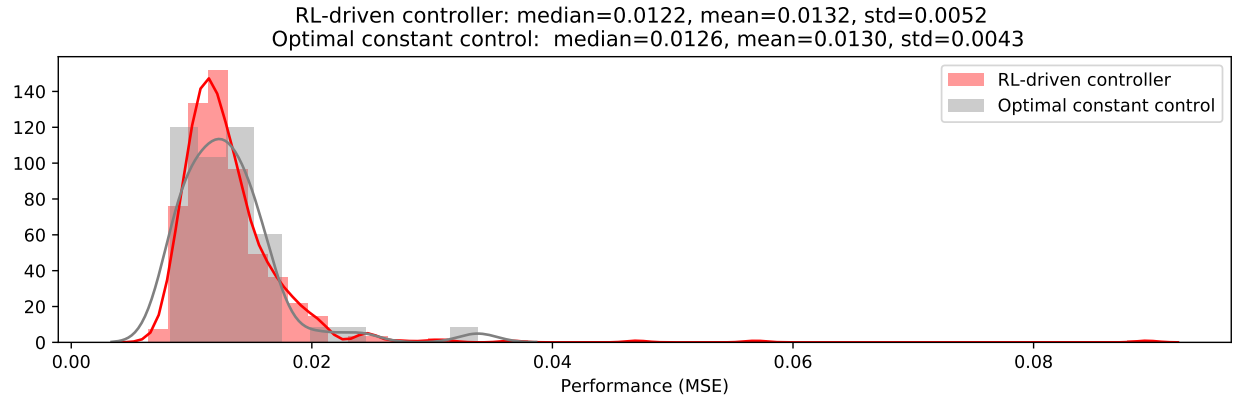


Figure 7. Distribution of controller and the competing approach performance for constant RPL of 1.2 p.u.

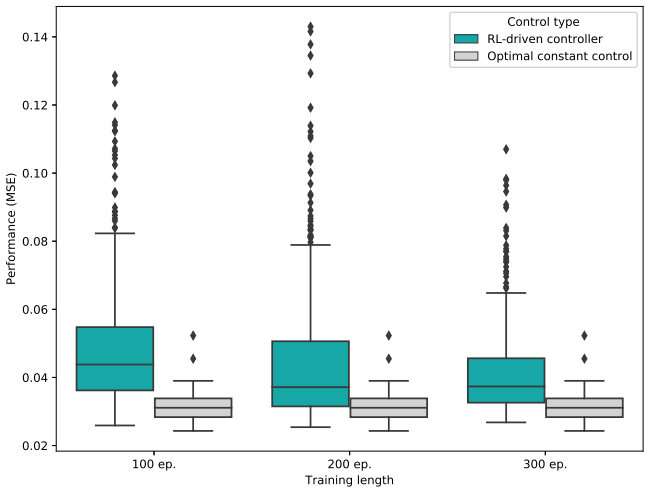


Figure 8. Distribution of controller and the competing approach performance for different training time

In Figure 7 in the case with RPL of 1.2 p.u. RL-driven controller performs slightly better than the competing approach. Figure 9 shows an example of system behaviour before and after controller training. It shown that APL approaches RPL after the RL-driven controller was trained, i.e. control goal is achieved at least to certain extent. In Figure 10 to investigate the dependence of performance on the RPL, experiments were conducted for RPL in the range of $[1.05..1.35]$ $p.u.$ For all constant RPL values the trained controller demonstrates performance comparable to the competing approach, except of RPL of 1.35 p.u.

In addition, distribution of RL-based controller is skewed having a long right tail - presence of outliers with high MSE (see Figure 7, 10). To check if longer training can improve results, agent was trained on the same setup twice longer - for 200 episodes instead of 100. The longer training did not show capability to significantly improve exploitation performance of a trained controller, while being efficient for reducing variance of the controller's exploitation performance. It can be concluded that during the training RL agent reaches a plateau and converges to a (sub)optimal policy.

As hyperparameters were chosen based on the experiments for the deterministic case and received results are not beating the competing approach, an attempt to find better hyperparameters combinations for the stochastic case was made. However, no parameter changes led to significant improvements in the performance measures, while in some cases leading to decrease in the performance. Thus, this combination of hyperparameters is considered as optimal.

When the learnt policy is good in the long run, in the short run the better performance of the competing approach may be explained by particularities of the chosen performance metric and transition process that is observed in the system at the beginning of the considered time interval. As MSE puts stronger emphasize on big differences, one large-amplitude oscillation in APL at the beginning of a simulation may significantly influence results. To test this hypothesis, experiment shown in Figure 11 was conducted on the time interval after transition process - $175 - 375$ seconds. It is observed, that there are no more outliers in controller performance. In this aspect, the performance of a trained RL-based controller is improved.

Q-learning controller training was done for the stochastic case with step down in RPL. In this case, to make a decision about a control action, controller takes into account both actual and reference power levels. The longer training did improved performance for control interval $t = 1s$, but didn't improve for $t = 5s$, so further investigation was done for the first option (see Figure 8). It is observed that longer training may improve the average performance. However, at some point an increase in training time does not lead to improvements in average performance anymore, but reduces variance of it.

The results indicate that even simple RL methods are capable to learn control. The learnt control is comparable with the the competing approach utilizing the simulation of the whole considered time interval. At the same time, trained controller is not strongly superior to the competing approach in sense of performance, while can be considered superior from the point of view of generalization.

To test a generalization capability of the RL-driven

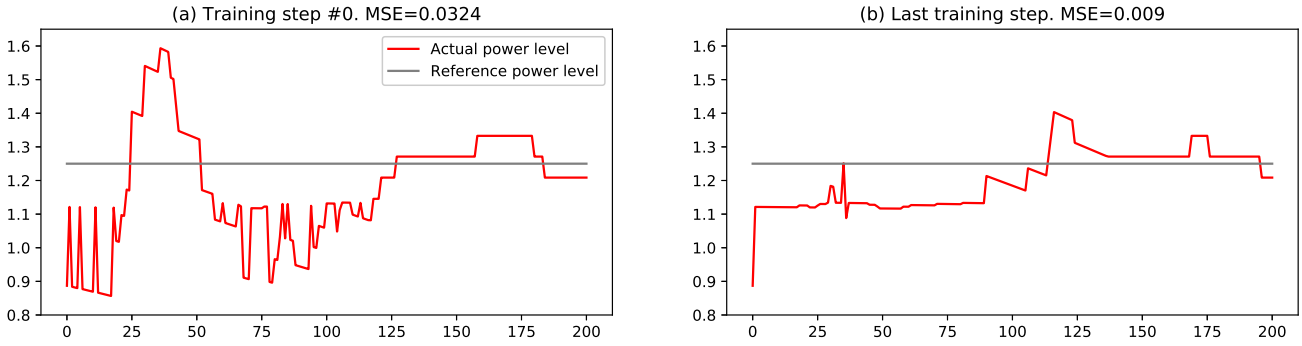


Figure 9. System behaviour before (a) and after (b) controller training

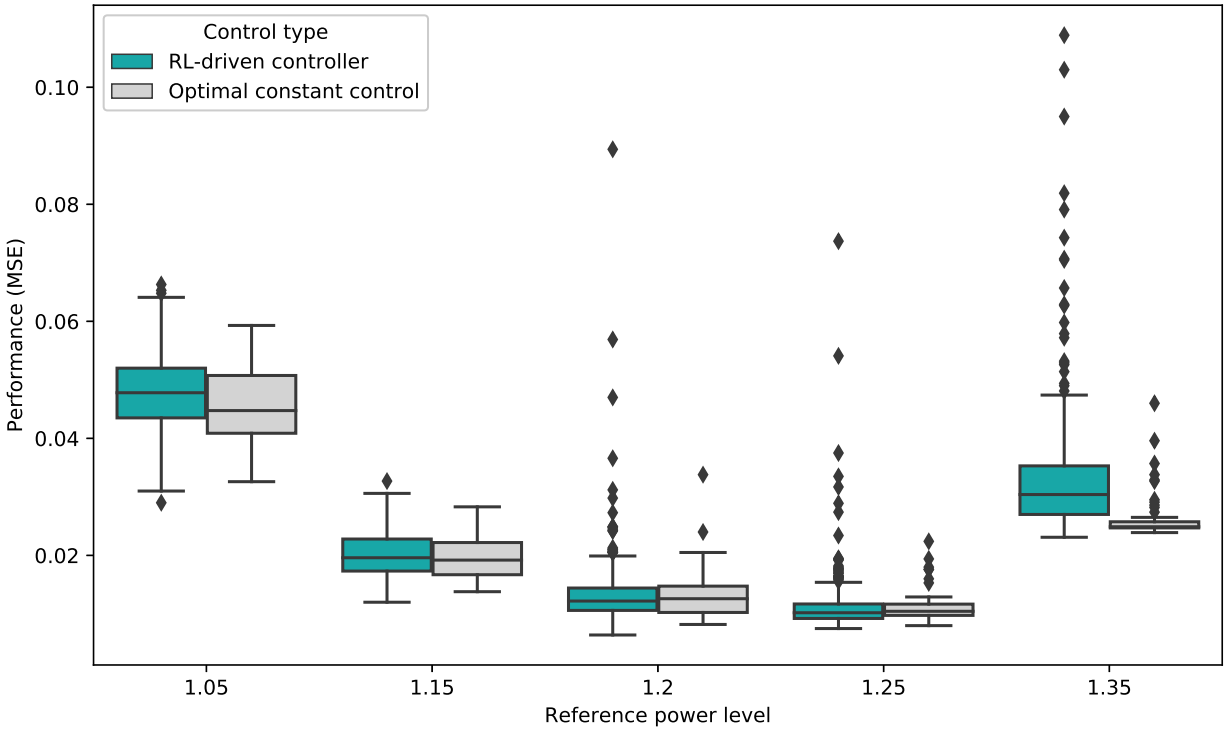


Figure 10. Distribution of controller and the competing approach performance depending on constant RPL

controller, training on $t = 200s$ interval with testing on $t = 400s$ was considered. When the transition process $[0..175]s$ at the beginning of an experiment is skipped, it induces intervals of $[175..375]s$ and $[175..575]s$ for training and testing respectively. In this case, optimal constant control action that was chosen based on the training period is not the optimal one for the testing period. At the same time RL-driven controller's policy allows the controller to slightly outperform the constant control

4.3 Smart Discretization

Dependency of the controller's performance on the number of bins in a fixed interval $[0.9; 1.7]$ for APL was investigated and presented in Table 4. Training converged to almost the same performance in all cases. However, a bigger number of bins corresponds to a higher variance of the performance. This is due to higher state space dimensionality that slows down convergence of an RL agent.

To apply optimal bin width estimation approach, historical data was collected in the experiments for the competing approach and in early stages of Q-learning experiments with other discretization strategy. Both options for collecting historical data lead to almost the same performance that is shown in Table 4. The estimated number of bins is around 60. Because of that, longer training is required to achieve good performance and reducing its variance.

Afterwards, an approach using quantiles of historical data as bins edges was applied using the same historical data. For the considered 10%-quantiles results didn't improve in comparison to the previous discretization strategies (refer to Table 4). Most likely there is no improvement, because the bins are too big.

Next, accounting for RPL in APL binning, when using its value as a bin's edge, was utilized. Total number of

DOI 10.3384/ecp202017431

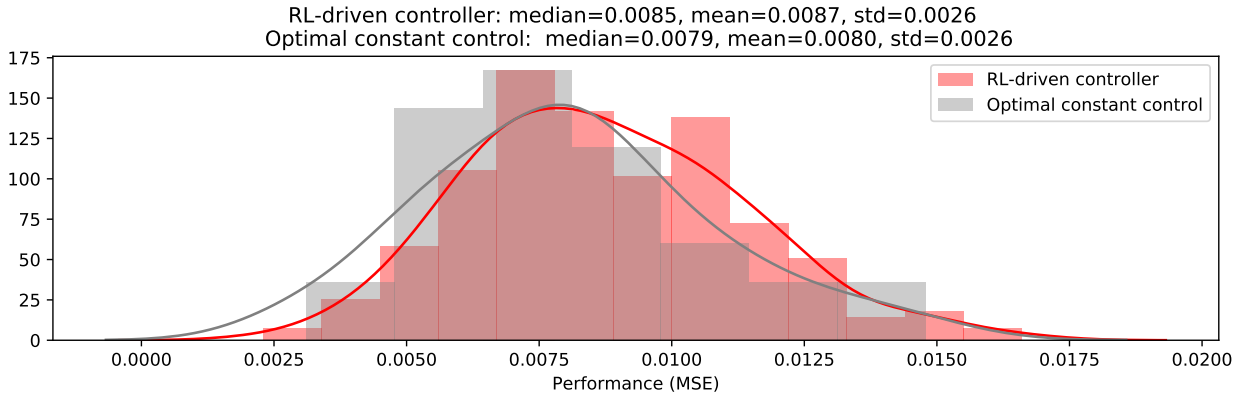


Figure 11. Distribution of controller and the competing approach performance for constant RPL of 1.2 p.u. and time interval 175-375 seconds

Table 4. Performance summary (median, mean, std) for Q-learning utilizing smart discretization strategies (HB - Histogram binning, CA - competing approach, HD - historical data, QL - Q-learning)

n	Experiment name	Median	Mean	Std
1	100 bins in [0.1;1.7]	0.0123	0.0132	0.0034
2	10 bins in [0.9;1.7]	0.0126	0.0133	0.0036
3	25 bins in [0.9;1.7]	0.0121	0.0131	0.0038
4	50 bins in [0.9;1.7]	0.0126	0.0132	0.0038
5	100 bins in [0.9;1.7]	0.0127	0.0142	0.0066
6	HB CA HD	0.0123	0.0134	0.0045
7	HB QL HD (32 episodes)	0.0126	0.0136	0.0042
8	HB, CA HD (200 episodes training)	0.0122	0.0131	0.0039
9	10 quantile bins, CA, HD	0.0128	0.0134	0.0039
10	10 quantile bins, QL, HD (32 episodes)	0.0126	0.0137	0.0049
11	10 bins, RPL as bin edge CA HD	0.0123	0.0131	0.0037
12	10 bins, RPL as bin edge QL HD (32 episodes)	0.0126	0.0132	0.0035

bins equals 10 and different historical data were tested (see Table 4). The latter was used to determine approximate interval for discretization using maximum and minimum historical values. A combination with historical data from the competing approach showed the best results among all smart descretization techniques.

In addition, the experiments with different discretization strategies were conducted for the skipping transition case, i.e. $[175..375]s$ simulation interval. In this case all discretization strategies have a comparable performance.

Results of testing the discretization strategies lead to conclusion that it is advised to make smaller bins in more dense regions. Although good results can be achieved by exploiting a high number of bins in equal-width binning, a bigger number of bins slows down training convergence and leads to inefficiency, as some parts of a Q-table of such an agent are updated extremely rarely. Thus, the best approach is to account for problem formulation and use historical data to utilize smart discretization strategy.

5 Conclusions & Future Work

The proof of concept developed in Python shows that RL is capable to learn efficient control policy to provide a voltage control-based ancillary service using TCLs. In addition, RL-based controller has shown the ability to learn efficient policies even without sophisticated tuning. Moreover, it shows the ability to generalize better than the competing approach of constant control. This way, a simulation of the whole considered interval can be avoided in practice. This is beneficial since such a simulation is not always available.

However, achieved performance, in general, is not significantly better than the performance of the competing approach. This may be explained by convergence to a sub-optimal policy or impossibility to learn an optimal policy by considering only one time point in the system trajectory for making a control change decision. I.e. it could be due to impossibility for an RL-agent to capture all the dynamics in the system when only values of APL and RPL at one time point are considered.

As the main direction of future work, authors consider work on improvement of the controller performance by applying other RL algorithms, such as deep RL and batch RL techniques, Bayesian approaches in RL. In addition, it may be beneficial to consider problem formulation that includes more information than just APL and RPL at the considered time point for making a decision about control change. In addition, performance may be improved by introducing some changes in controller design, such as relaxing constraints for output voltage.

As a side note, it would be useful to parallelize experiments to enhance a further development process. An experiment with the RL-driven controller (includes training and validation repeated 5 times) takes about 6 hours, while it takes approximately 3.5 hours for the optimal constant control experiment (simulation of the considered time interval repeated for each of 8 possible control parameter values). Machine parameters were: Intel-i7 7500U 2.7GHz (3 cores available), 12GB RAM. At the moment, parallelization attempts have not achieved significant speed up. Although parallel experiments make use of additional available CPU cores and RAM, the simulation tools could be a bottleneck.

Acknowledgements

Authors would like to thank ELEKS (`https://eleks.com/`) for funding the Machine Learning Lab at Ukrainian Catholic University and this research.

References

Miroslav Begovic, Aleksandar Pregelj, Ajeet Rohatgi, and Damir Novosel. Impact of renewable distributed generation on power systems. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pages 654–663. IEEE, 2001.

Tetiana Bogodorova, Luigi Vanfretti, and Konstantin Turitsyn. Voltage control-based ancillary service using thermostatically controlled loads. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2016.

Bert J Claessens, Dirk Vanhoudt, Johan Desmedt, and Frederik Ruelens. Model-free control of thermostatically controlled loads connected to a district heating network. *Energy and Buildings*, 159:1–10, 2018.

Damien Ernst, Mevludin Glavic, Florin Capitanescu, and Louis Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2008.

David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.

Grayson Heffner. Loads providing ancillary services: Review of international experience. 2008.

Ali Ipakchi and Farrokh Albuyeh. Grid of the future. *IEEE power and energy magazine*, 7(2):52–62, 2009.

BRENDAN Kirby and Eric Hirst. Load as a resource in providing ancillary services. *Lockheed Martin Energy Research, Oak Ridge National Laboratory. Oak Ridge, TN*, 1999.

Oleh Lukianykhin and Tetiana Bogodorova. Modelicagym: Applying reinforcement learning to modelica models. *arXiv preprint arXiv:1909.08604*, 2019.

O. Ma et al. Demand response for ancillary services. *IEEE Transactions on Smart Grid*, 4(4):1988–1995, Dec 2013. ISSN 1949-3053. doi:10.1109/TSG.2013.2258049.

Sean P Meyn, Prabir Barooah, Ana Bušić, Yue Chen, and Jordan Ehren. Ancillary service to the grid using intelligent deferrable loads. *IEEE Transactions on Automatic Control*, 60(11):2847–2862, 2015.

Takao Moriyama, Giovanni De Magistris, Michiaki Tatsubori, Tu-Hoa Pham, Asim Munawar, and Ryuki Tachibana. Reinforcement learning testbed for power-consumption optimization. In *Asian Simulation Conference*, pages 45–59. Springer, 2018.

Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, Jul 2009. ISSN 1573-7527. doi:10.1007/s10514-009-9120-4. URL `https://doi.org/10.1007/s10514-009-9120-4`.

Frederik Ruelens, Bert J Claessens, Salman Quaiyum, Bart De Schutter, R Babuška, and Ronnie Belmans. Reinforcement learning applied to an electric water heater: from theory to practice. *IEEE Transactions on Smart Grid*, 9(4):3792–3800, 2016.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL `http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html`.

S.Mottahedi. Battery energy management system using reinforcement learning, 2017. URL `https://github.com/smottahedi/RL-Energy-Management/blob/master/presentation.ipynb`.

Simon H Tindemans, Vincenzo Trovato, and Goran Strbac. Decentralized control of thermostatic loads for flexible demand response. *IEEE Transactions on Control Systems Technology*, 23(5):1685–1700, 2015.

José R Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied energy*, 235:1072–1089, 2019.

O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, et al. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, 2019. URL `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`.

Wei Zhang, Karanjit Kalsi, Jason Fuller, Marcelo Elizondo, and David Chassin. Aggregate model for heterogeneous thermostatically controlled loads with demand response. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE, 2012.

A Modelica-based solution for the simulation and optimization of microgrids

Stéphane Velut[1] Johan Andreasson[1] Jiri Navratil[1] Marcus Åberg[1]

[1]Modelon, Sweden, stephane.velut@modelon.com

Abstract

By integrating a high share of distributed generation units, microgrids can accelerate the shift to a more sustainable power grid. This transition is however not free from challenges. The variability and uncertainty of the renewable energy sources as well as the absence of large-scale dispatchable storage systems pose challenges for the integration and operation of this new type of power grid. Model-based engineering can provide valuable tools to develop design and control strategies that do not jeopardize the stability and reliability of the power supply. This paper presents elements of a Modelica-based workflow for the design and operation of microgrids. The framework allows for a multi-fidelity modeling approach and is therefore suitable for solving a large variety of engineering problems, from early component design to the verification of component and control design using detailed models. This paper illustrates the flexibility of the framework with respect to the user interface, the models and the analyses.

Keywords: microgrid, power systems, controls, dynamic optimization, renewables, battery

1 Introduction

Environmental considerations and increasing awareness of infrastructure sensitivity have led to reconsiderations of how energy system should best be configured. The current, highly centralized systems were developed for large production units such as nuclear and fossil power plants, and are not suitable for renewable, intermittent and distributed energy sources such as wind and solar (Fathima and Palanisamy, 2015).

A microgrid is a group of interconnected energy sources, loads and storage devices that can operate both connected with the surrounding electricity grid, and disconnected in islanding mode. It has the potential to offer increased self-sufficiency and reliability at low cost and reduced environmental impact (Eto et al, 2018). Microgrids typically include smaller production units such as photovoltaic arrays, wind turbines, microturbines and generators (combustion engines) and storage devices such as flywheels and batteries. Their comparably smaller investment cost makes them attractive to install in remote areas and their capacity for reducing transmission congestion makes them interesting for energy suppliers (Venkatraman and Khaitan, 2015).

In a previous paper (Windahl et al, 2019), the authors identified the need for a flexible and collaborative platform that allows different stakeholders to analyze microgrid systems, from the early system design based on site-specific data to the detailed verification of the complete system including controls. The authors considered Modelica as suitable technology candidate thanks to its ability to describe multi-physics systems with various fidelity levels and thanks to the openness of the standard that allows for model customization to fit specific application needs.

The goal of this paper is to share the latest development of Modelon's microgrid solution, especially for the detailed electrical transient analysis. Additional examples for optimal design and operation are also provided to further illustrate the flexibility of the approach.

2 Modelica based framework

In this section we briefly present the different components of the proposed Modelica solution for the analysis of microgrid systems. The framework consists of three elements: Modelica libraries with models of various fidelity levels, a computational engine for simulation and optimization over different time horizons and finally user interfaces to conveniently formulate and solve various engineering tasks on the microgrid models. As explained in (Windahl et al, 2019), the framework tightly integrates the components through a common architecture.

2.1 Modelica libraries

2.1.1 Low fidelity models

The low-fidelity models that describe the microgrid behavior in terms of energy and power flows are available in the Thermal Power Library, commercial product of Modelon. A brief overview is given in (Windahl et al, 2019). The component models are typically implemented as efficiency curves and are

suitable for simulation and optimization over long horizons, typically from one day to one year.

2.1.2 High fidelity models

The high-fidelity models are targeting mechanical and electrical transient analysis over short horizons, in the range of microseconds to minutes. The core libraries used for that purpose are two:

2.1.2.1 Electric Power Library

Electric Power Library by Modelon is a general-purpose library for the simulation of electric power systems. It contains all required components to describe various configurations of DC and AC grids – 1 phase, 3 phase, balanced and unbalanced: lines, loads, machines, and inverters. Concerning the converters, a core component for the integration of renewable energy sources, it exists in different versions to capture the fast dynamics of switching gates or just the average dynamics for slower transients. Examples of application using the library can be found in (Olenmark et al 2014), (Enerbäck et al, 2013), (Magnúsdóttir et al, 2017), (Pettersson et al, 2012). Figure 1 shows for instance how to describe the electro-mechanical transient of a wind power plant and its connection to the main grid using back-to-back converters and two controllers. The machine side converter maximizes the power drawn from the wind whereas the grid side controller maintains a stable DC voltage.

The library has recently been adapted to the specific needs of microgrid system modelling. This includes the implementation of typical blocks such as Phase-Locked-Loop, typical controllers implemented in the dq frame for inverter-interfaced distributed energy resources (grid forming, grid feeding and grid supporting). With these additions, it is possible to describe and simulate arbitrary DC and AC microgrids in either islanded or connected mode.

2.1.2.2 Electrification Library

Electrification Library by Modelon is a multi-physics Modelica library for the design, verification and control of electrified systems. The fidelity level of its component models is somewhat lower than the one of Electric Power Library, but it is possible to adapt the level of detail of the electrical, mechanical and thermal dynamics to cover time scales from fast electrical to slow thermal and even aging dynamics. With a large set of battery models, it is a good complement to Electric Power Library when modelling battery energy storage systems in microgrids.

Using a simple adaptor, Electrical Power and Electrification libraries can be combined on the DC side, as it is shown in Figure 2 that represents a battery – from Electrification Library – connected to the main grid through a grid-feeding inverter – from Electric Power library.

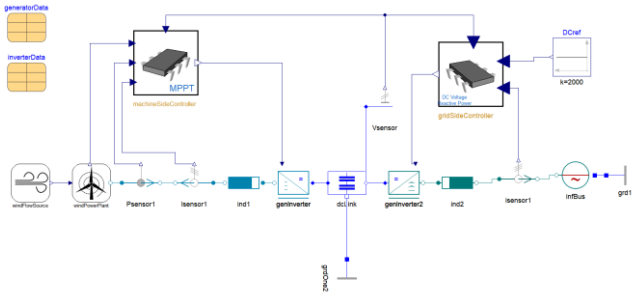


Figure 1. Electro-mechanical Modelica model of a wind power plant connected to an AC grid through a back-to-back inverter. The machine side controller is in power mode and maximizes the power extracted from the wind whereas the grid side controller maintains the voltage level at the DC link.

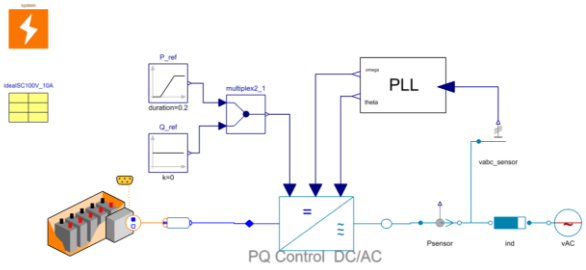


Figure 2. Model of a battery connected to a stiff AC grid via a grid-feeding inverter block, getting the reference phase and frequency from a Phase-Locked-Loop component.

.

2.2 Computational engine

Optimica Compiler Toolkit by Modelon is a Modelica compilation and execution environment with applications beyond dynamic simulation, most notably steady-state simulation and optimization. It supports the Optimica language (Åkesson et al, 2008) (Magnusson et al, 2015) which is an extention to Modelica to allow formulation of optimization problems. Applications include for example district heating (Schweiger et al, 2017).

The optimization framework has also been demonstrated in industrial applications as described in (Dietl et al, 2018) where the approach is applied to optimize in real-time the start-up of a gas combined cycle power plant.

In addition to the usage described here, it is used as Modelica engine in several system simulation tools on

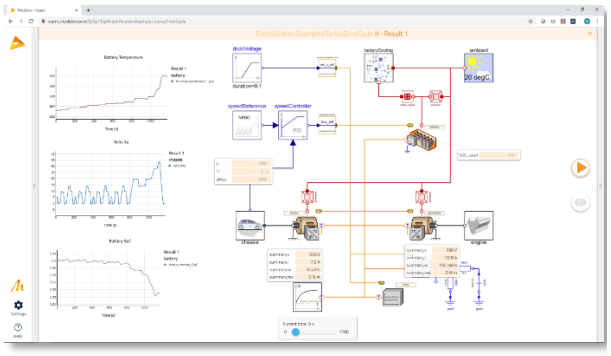
DOI 10.3384/ecp202017441

the market, including for example ANSYS Twin Builder.

2.3 User interface

Three types of user interfaces are available, each one being accessed from a web-browser and targeting a specific user profile with his own engineering tasks and access rights:

- Model centric view for building the microgrid model in a web-browser by drag and dropping components from the previously mentioned libraries. A dashboard view, as a simple interface for the deployment of the tool to less technical end-users. Figure 5 shows a web-app to setup, simulate and optimize a microgrid system. Results can be visualized in tables or as time trajectories.



- Figure 3 shows an electric vehicle model based on Electrification Library.
- Workflow centric view to implement advanced engineering tasks to perform various computations and exchange data between the different steps, see Figure 4 for an illustration.
- A dashboard view, as a simple interface for the deployment of the tool to less technical end-users. Figure 5 shows a web-app to setup, simulate and optimize a microgrid system. Results can be visualized in tables or as time trajectories.

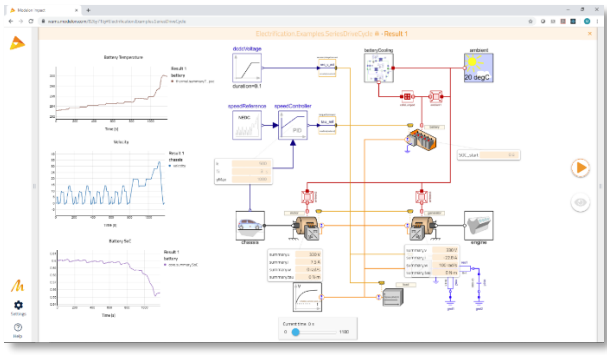


Figure 3. Model authoring interface to create and parameterize system models from a web-browser. Post-processing views are simply created by drag-and-drop to the canvas and can be saved to be applied on other simulation setups.

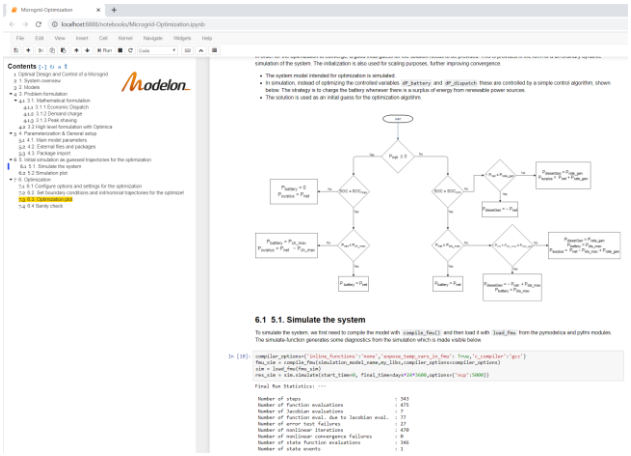


Figure 4. Jupyter notebook as interface for implementing and executing complex engineering workflow, also from a web-browser.

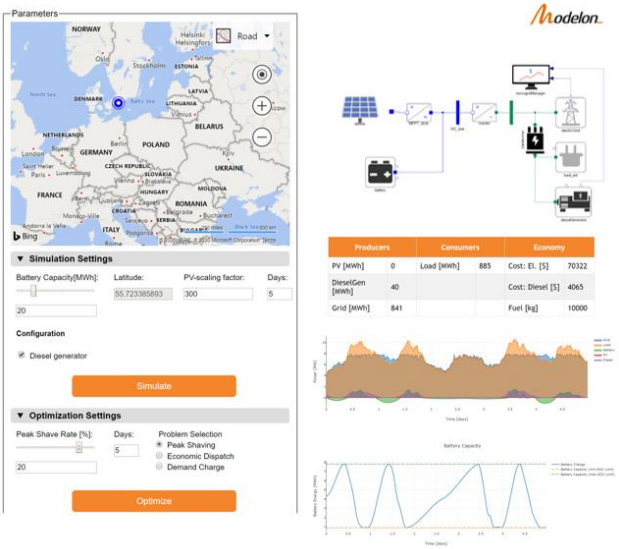


Figure 5. Web-app for deployment of microgrid analysis and design tools to less technical end-users. An open rest API makes it possible to access the main features of Modelon compiler, e.g. parameter settings, compilation, simulation, from customized html dashboards. A widget has here been integrated to conveniently set the microgrid location from an interactive map, which determines the power produced by the PV panels.

3 Use cases

3.1 Optimal design and operation

3.1.1 Platform capability

In (Windahl et al, 2019) and (Akhlagi, 2019), the authors list various types of microgrid optimization problems that can be formulated and solved using the Modelica platform. This includes optimal design problems such as battery or photovoltaics sizing and operational problems such as demand charge reduction or economic dispatch. It is also possible to solve hybrid problems in the sense that they combine physical

domains - e.g. microgrids with cogeneration plants and thermal loads - or that they combine design and operation problems. The peak shaving formulation is for instance looking for the minimal battery size (design problem) when optimally operating the remaining assets (control problem). For longer time horizons, Model Predictive Control can be applied, see (Axelsson et al, 2015) for the implementation details.

Compared to previous work, the optimization package has been improved as follows:

1. For an improved user-friendliness and readability, the complete optimization problem including constraints, cost function and the degrees of freedom is specified from the graphical user interface of the authoring tool instead of being added in a separate Optimica file and Python code.
2. CO2 emission is computed in the diesel engine component and environmental constraints in terms of overall allowed emission by the microgrid can be therefore introduced in the optimization problem.
3. An electrical vehicle component has been added to represent a car park charging station with forecast on the cars' parking schedule and battery charging requirements.

3.1.2 Optimization of a microgrid with multiple gensets

Let's now consider the economic dispatch problem of a fictive grid-connected system represented in Figure 6. The microgrid includes the following units:

1. A PV power station
2. Two diesel engines of 750 kW capacity and identical efficiency curve but different fuel properties
    a. Genset 1 with a fuel characterized by a cost of 0.33 USD/l and a CO2 emission of 2.5 kg/l
    b. Genset 2 with a cheaper fuel (0.25 USD/l) but emitting more CO2 (5 kg/l).
3. A load with its associated power forecast over the optimization horizon
4. A battery of size 40MWh with operational constraint in terms of max and min state of charge (0.1 and 0.9 respectively) and a maximal charge/discharge rate of 2 MW.

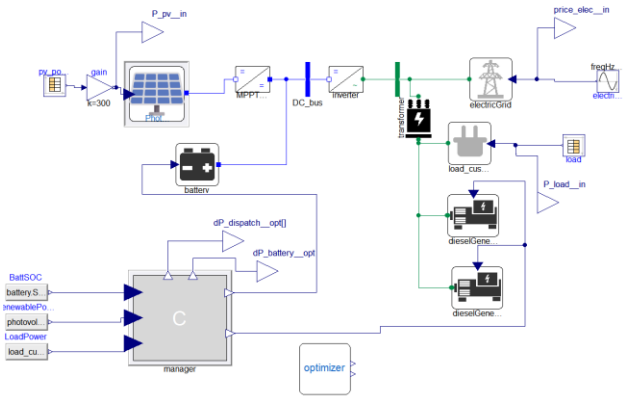


Figure 6. Microgrid system used for the economic dispatch formulation. It consists of two engines with their specific fuel and emission characteristics, a battery, a PV panel a load and the grid. The optimizer block contains the formulation of the cost function and the overall CO2 emission constraint.

The goal is to minimize the operational cost from the grid power import and the engine fuel usage while satisfying the following operational and environmental constraints:

- A maximum power grid import of 8MW
- A maximum overall CO2 emission of 25 tons

Forecast for electricity price, load and solar irradiation are given by 15 min sampled trajectories over the 2 days long optimization horizon.

The optimization workflow has been implemented as a jupyter notebook and the functionality has also been made available from the model authoring tool. It contains the following steps:

- Experiment setup by mainly specifying the files with the forecast data
- Compilation of the simulation model
- Initial simulation for generating the guess trajectories for the optimization,
- Setup and compilation of the optimization model
- Optimization
- Post processing for results visualization and comparison with reference case

The problem is initialized with a simulation using a simple control strategy aiming at charging the battery during overproduction and prioritizing the diesel engine with cheapest fuel.

The results are shown in Figure 7 and Figure 8. The optimal cost (not shown) is slightly lower than the reference one but the constraints in terms of maximal power and CO2 emission is fulfilled. Using the optimization framework, it is possible to investigate the trade-off between operational cost and environmental footprint.

DOI 10.3384/ecp202017441

The solution time for this problem is 25 s, using a standard laptop. Similar performance has been achieved for all considered problems with similar sampling time and optimization horizons.

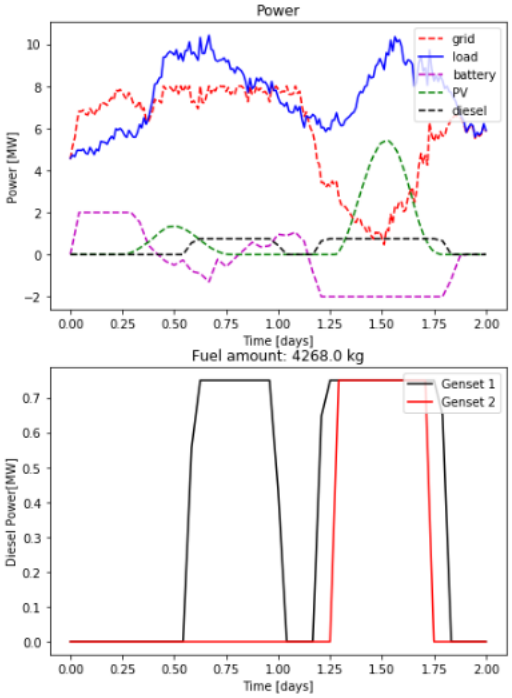


Figure 7. Economic dispatch results. The top figure shows the optimal power trajectories. Note that the grid import has been shaved to satisfy the maximal grid import of 8 MW. The bottom plot shows that the genset with more expensive fuel is prioritized due to its lower CO2 emission constraint. When electricity is expensive around 1.5 day (not shown), both engines are used.

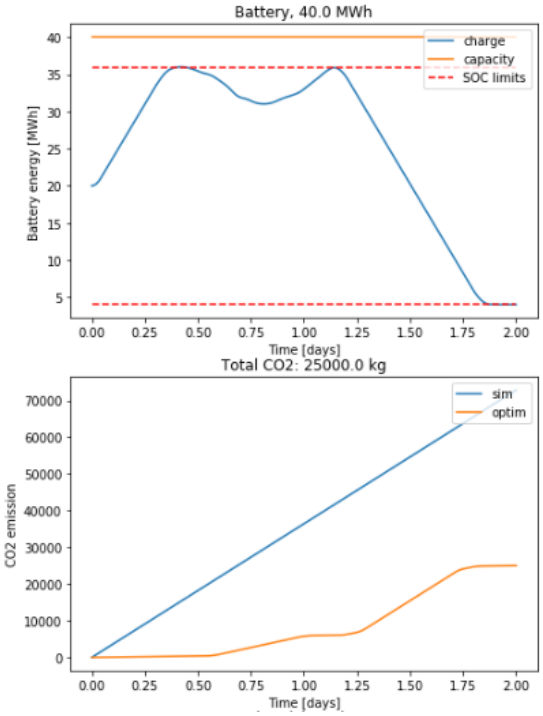


Figure 8. Economic dispatch results. The top plot shows the battery energy together with the operating limits. The battery usage follows the electricity price (not shown): it is charged at low tariffs and discharged at high ones. The lower plot shows the CO2 emission reduction compared to the reference case.

3.2 Electrical transients

As explained in Section 2.1.2, detailed electrical transients can be simulated within the platform by using off-the-shelf libraries. Some additional control components have lately been implemented to apply typical control strategies for inverter-interfaced energy sources. This comprises phase-locked-loops, low level current and voltage controllers to be used in grid feeding, grid forming and grid supporting inverters. Figure 9 shows for instance the synchronization block that is used to match phase and voltage of a microgrid in islanded mode before its connection to the main grid. The Modelica implementation is similar to the one in (Biswarup et al, 2019) and consists mainly in PID controllers, PLL and logical blocks.

To illustrate the libraries' ability to simulate detailed power transients, the low voltage, three phase AC system shown in Figure 10 has been modelled and simulated in a specific scenario. The system comprises

- a genset with a first order governor, a first order exciter and a third order generator model
- a renewable energy source operating in grid feeding mode and represented by an ideal DC source. The inverter switching dynamics is not considered here and it is described by an average model. The PQ controller is implemented in the

dq frame. Note that higher fidelity models could be used to describe the DC transients on the production side.
- an impedance load consuming a given active and reactive power at nominal voltage
- a stiff AC grid represented by a three-phase voltage source
- Impedance lines

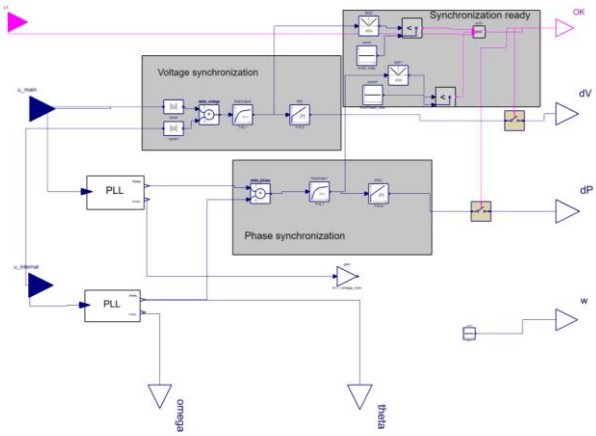


Figure 9. Modelica implementation of a grid synchronization block including phase-locked-loops to track the phase and frequency of the internal and main grids, and PID to adjust voltage and power level of the units.

Note that the grid frequency has been arbitrarily lowered to 7 Hz to better follow the synchronization process when comparing the voltage signals in the inertial frame. The library allows for the simulation of networks operating at arbitrary frequency.

The scenario is defined as follows:
1. From t=0 s to t=1 s, the microgrid is operating in islanded mode. The REN source is increasing its load to a nominal value of 1MW whereas the genset takes care of frequency and voltage control.
2. The grid synchronization is activated at t=2s and the grid waits for the synchronization criteria to be satisfied to close the switch., which takes approximately 200 ms.
3. At t = 3.5 s, the load is increased by 30% in 200ms and the grid takes care of this disturbance
4. At t = 5 s, the grid is suddenly disconnected from the main grid as if a fault had occurred. The diesel engine quickly reacts to compensate for the power outage, the frequency and voltage levels stay under control.

4 Conclusion

An update of Modelon's web-based framework for microgrid design and analysis has been presented. Compared to its early version, focusing on long term analysis and on a limited set of optimal design and operation problems using low fidelity levels, the new version is able to simulate the fast current and voltage transients of various microgrids during islanded and grid-tied modes. The optimization package has also been upgraded for an improved user-friendliness and to include CO2 emission and a new component for electric car charging stations. Examples have been presented to illustrate the framework in two use cases: economic dispatch of a grid with PV, battery and gensets as well as detailed simulation of an AC grid during transition between islanded and connected modes.

5 Future work

Future work includes the support of the following features:
- More advanced control blocks for grid-supporting inverters with the implementation of droop controllers
- Implementation of a flexible grid architecture with templates and interfaces to allow for user-friendly reconfiguration
- Automatic Modelica code generation to easily define microgrid from specifications in a simple interface such as a web-app
- Optimization of microgrids in islanded mode and during mode transition

References

Dietl et al. Start up optimization of Combined Cycle Power Plants: Controller development and real plant test results, CoDIT 2018

Johan Åkesson. (2008). Optimica - An Extension of Modelica Supporting Dynamic Optimization. Proceedings of the 8th International Modelica 2008 Conference. Bielefeld, Germany.

Ramakrishnan Venkatraman and Siddharta Kumar Khaitan. A Survey of Techniques for Designing and Managing Microgrids. IEEE Power & Energy Society General Meeting, 2015

A. Hina Fathima and K. Palanisamy. Optimization in microgrids with hybrid energy systems – A review. Renewable and Sustainable Energy Reviews, 45, 431-446, 2015

J. Windahl, H. Runvik, S. Velut, Platform for microgrid design and operation, Proceedings of the 13th International Modelica Conference, 2019, Regensburg, Germany

Joseph H. Eto, Robert Lasseter, David Klapp, Amrit Khalsa, Ben Schenkman, Mahesh Illindala and Surya Baktiono. The CERTS Microgrid Concept, as Demonstrated at the CERTS/AEP Microgrid Test Bed, Energy Analysis and Environmental Impacts Division Lawrence Berkeley National Laboratory, 2018

 DOI 10.3384/ecp202017441

Jonas Enerbäck and Oscar Nalin Nilsson. Modelling and Simulation of Smart grids using Dymola/Modelica. Division of Industrial Electrical Engineering and Automation Faculty of Engineering, Lund University, 2013

Fredrik Magnusson and Johan Åkesson. Dynamic optimization in JModelica.org. Processes, 3(2):471–496, 2015

G. Schweiger et al., District heating and cooling systems – Framework for Modelica-based simulation and dynamic optimization, In Energy, 2017 , ISSN 0360-5442

A. Olenmark and J. Sloth. Flexible AC/DC Grids in Dymola/Modelica. Master of Sc. Thesis. Lund University. Faculty of Engineering, Lund, 2014

A. Magnúsdóttir and D. Winkler, Modelling of a Hydro Power Station in an Island Operation, Proceedings of the 12th International Modelica Conference, 2017, Prague, Czech Republic

J. Pettersson et al, Modeling and Simulation of a Vertical Wind Power Plant in Dymola/Modelica. Proceedings of the 9th International Modelica Conference, 2012, Munich, Germany

Modelon. Thermal Power Library. https://www.modelon.com/library/thermal-power-library/

Modelon. Electrification Library. https://www.modelon.com/library/electrification-library/

Modelon, Electric Power Library. https://www.modelon.com/library/electric-power-library/

Ali Akhlagi, A Modelica-based framework for modeling and optimization of microgrids, Master of Science Thesis, KTH School of Industrial Engineering and Management, Energy Technology TRITA-ITM-EX 2019:543, Stockholm, Sweden

M. Axelsson et al, A Framework for Nonlinear Model Predictive Control in JModelica.org. Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015

M. Biswarup and L. Vanfretti. Modeling of PMU-Based Automatic Re-synchronization Controls for DER Generators in Power Distribution Networks using Modelica and the OpenIPSL Proceedings of the 13th International Modelica Conference. March 4-6, 2019, Regensburg, Germany

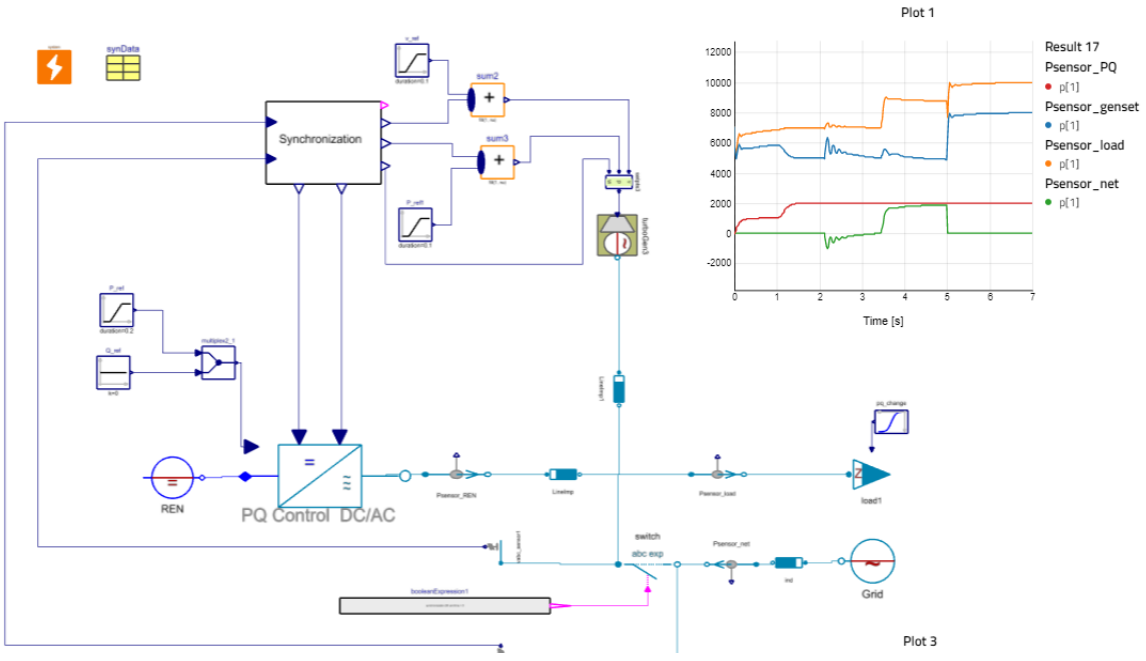


Figure 10. Three phase microgrid system implemented using Electric Power Library components. It is composed of 4 units: an ideal DC source controlled in PQ mode to represent a renewable energy source, a diesel engine, an ideal impedance load and an AC voltage source to represent the main grid. The microgrid is first operated in islanded mode. At t=2 s, the synchronization process is initiated, and synchronization occurs after 200ms. A load change occurs at t = 3.5 s. At t =5 s, the microgrid is suddenly disconnected from the grid and the diesel engine reacts to maintain the internal frequency and voltage levels.

DOI
10.3384/ecp2020174

Modelling and Control of Fast-Switching Solenoid Direct Injection Valves Using a New Magnetics Library

Julian Mühlenhoff[1] Emanuel Rauer[2] Tom Ströhla[1]

[1]Mechatronics Group, TU Ilmenau, Germany, {julian.muehlenhoff}@tu-ilmenau.de
[2]Woodward L'Orange GmbH, Stuttgart, Germany

Abstract

This paper deals with the special challenges in modelling and controlling fast-switching solenoid valves with Modelica. For this, a solenoid injector for application in direct injection combustion engines with switching times around 100 µs is being used as an example. The occurring eddy current losses as well as the local saturation phenomena based on magnetic field displacement require for detailed network models of the magnetic domain. Therefore, a new Modelica magnetics library based on a *consistent systems modelling analogy* is being proposed and implemented, which increases solvability of the injector models compared to the *Modelica Standard Library*. Additionally, different one-dimensional contact mechanic models for representation of the bouncing behaviour of switching-type solenoids are evaluated. The complete electro-magneto-mechanical model of the injector is then used for synthesis of novel closed-loop control schemes found by model inversion and parameter optimisation.

Keywords: solenoid injector, consistent magnetic analogy, FluxTubes, hysteresis damping models, needle trajectory control, model inversion, optimisation

1 Introduction

The past decades have been shaped by development of many alternative vehicle powering concepts, however the combustion engine still has its benefits when it comes to, e.g., refuelling or power density of the storage medium. On the other hand combustion engines are still an active field of research due to possible improvements in terms of pollution and carbon dioxide emissions.

Most of today's engines are based on the direct injection principle, by which the combustion can be controlled more precise due to possible pre- and post-injections as well as higher variability with the main injections per rotation (Bauer et al., 2004, pp. 146 – 147). The demands on injectors are thereby increasing; in particular the minimum switching times have to lie below 1 ms, while the injection pressures the valve has to deal with are also growing (Ströhla, 2012, p. 13). Due to higher manufacturing costs the use of piezo-actuated valves instead of solenoids is not an option for low-priced cars or engines.

These demands on solenoid injectors are met with a two-stage movement of the armature, which allows the latter to accelerate without opening the valve in the first stage. Once the armature has traversed this free stroke space and has accumulated some momentum, the valve seat is then opened by a needle, which gets picked up by the already moving armature (Denk, 2018, p. 17). Figure 1 shows such an injector with two stages of movement for the gasoline direct injection, on which this paper is based. Figure 2 illustrates the two-stage movement.

Figure 1. Solenoid valve for gasoline direct injection.

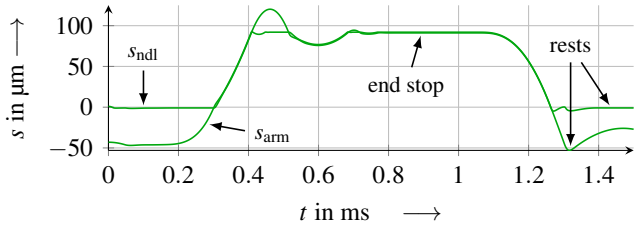


Figure 2. Exemplary movements of armature s_{arm} and needle s_{ndl} out of the resting positions during an injection cycle.

In order to minimise switching times and to overcome the time delaying influences of the coil inductance as well as the eddy currents, an electric excitation with a *boost phase* is required, as depicted in Figure 3. The boosting voltage of 65 V is used to "pump" huge amounts of energy within a short period into the magnetic system (Bauer et al., 2004, p. 153). After this, there are two phases during which the electric current is held at specified levels to provide different magnetic reluctance forces for catching and holding of the armature, respectively (Denk, 2018, p. 27).

The relatively large time delaying influence of the electromagnetic domain, compared to the switching times, limits the controllability of armature and needle positions and therefore of the injection process itself. The applied excitation scheme also leads to an overshoot in terms of armature velocity, which results in strong bouncing at the

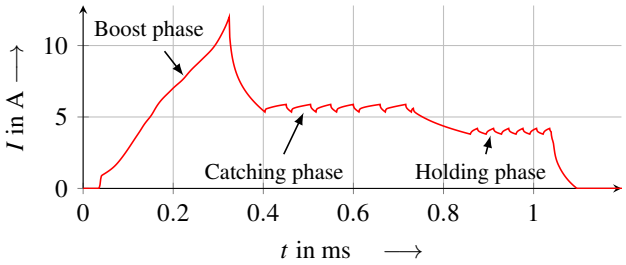


Figure 3. Common excitation scheme for solenoid injectors.

stop plate. This not only increases noise, but also reduces life span of the components due to mechanical wear. In addition, wear is also a major source for changing injection amounts during life time (Rauer et al., 2019). Another demand for the injection of smallest amounts of gasoline is the operation in so-called *ballistic mode*, in which armature and needle do not reach the stop plate (Denk, 2018, pp. 18 – 19). While this allows the engine to run in alternative combustion modes, the sensitivity of the output spray based on deviations in manufacturing and assembling or wear is still a problem.

Therefore, this paper addresses the challenges when modelling the electric, magnetic and mechanic domains of an injector in order to use this model for synthesis of novel control schemes. Fluid mechanics is neglected here, because lumped-parameter models are not suited for the complex behaviour of the injection fluid inside the valve, where shock waves and combustion pressures have to be taken into account. This domain could be included later by connecting *CFD* software via *FMI*, e.g.

2 Modelling the magnetic domain

2.1 Magnetic network analogies

For system simulation purposes it is common to transfer the usually as partial differential equations formulated physics into a network of lumped elements. As connection variables there can be chosen multiple sets of pairs consisting of a potential and a flow variable. For the magnetic domain usually the magnetic flux Φ gets utilised as the flow variable, for which reason the magnetic reluctances R_m are understood as "magnetic resistors". This is due to the fact, that component equations connecting potential and flow variable derivative-free with each other are commonly assumed to show *dissipative* behaviour like e.g. the electric resistor (Zimmer and Cellier, 2006).

However, this does not hold for magnetic reluctances, as they merely *store* energy in a *capacitive* way. To address this, the magnetic flow variable has to be changed to the time derivative of the magnetic flux $\dot{\Phi}$, which leads to the reluctance component equation

$$\dot{\Phi} = C_\mathrm{m}\left(V_\mathrm{m}\right)\,\Delta\dot{V}_\mathrm{m} \tag{1}$$

with the magnetic potential V_m and the reluctance capacity $C_\mathrm{m} \neq 1/R_\mathrm{m}$ (Littmann and Schiedeck, 2008). In demarcation against the *classic analogy* we want to call this

Table 1. Analogies describing magnetism by lumped elements.

	Classic analogy	**Consistent analogy**
Potential variable	V_m	V_m
Flow variable	Φ	$\dot{\Phi}$
Reluctance element	$\Delta V_\mathrm{m} = R_\mathrm{m}\Phi$ (dissipative)	$\Delta\dot{V}_\mathrm{m} = 1/C_\mathrm{m}\cdot\dot{\Phi}$ (capacitive)
Eddy current element	$\Delta V_\mathrm{m} = L_\mathrm{m}\dot{\Phi}$ (inductive)	$\Delta V_\mathrm{m} = R^*_\mathrm{m}\dot{\Phi}$ (dissipative)

principle the *consistent analogy* throughout this paper, as the (time-derived) potential and flow variables in the component equations indicate the correct energy-related behaviour. Additionally, the product of potential and flow variable is always a power value. Similarly to the reluctance, we can also rewrite the equations of eddy current elements – expressed in the magnetic rather than the electric domain – by introducing the parameter $L_\mathrm{m} \equiv R^*_\mathrm{m} \equiv 1/R_\mathrm{ec}$ as summarised in Table 1, with R_ec being the electric resistance of the eddy current path. A physically inductive element in the magnetic domain does not exist.

As a side effect, the coupling equations between electric and magnetic domain are also simplified, since both AMPÈRE's law (Kallenbach et al., 2017, p. 18)

$$\Delta V_\mathrm{m} = \Theta = N\,I \tag{2}$$

and FARADAY's law (Kallenbach et al., 2017, p. 30)

$$\Delta U = U_\mathrm{i} = -N\,\dot{\Phi} \tag{3}$$

are now defined via the pure flow variable of the counterpart domain (I and $\dot{\Phi}$) multiplied with the winding number N, respectively. Here, the electric domain is represented by potential U and flow I. This coupling principle can be observed in energy transformations of other physical domains and helps in teaching and understanding generalised mechatronic systems modelling (Grabow, 2013; Littmann and Schiedeck, 2008).

The classic analogy is widely used for network-based simulation purposes as well as for engineering-level rough design tasks (Birli et al., 2003; Ströhla, 2012). Also the `Modelica.Magnetics.FluxTubes` library bases upon this type of analogy (Bödrich and Roschke, 2005; Bödrich, 2008), which was extended by models for covering magnetic hysteresis phenomena (Ziske and Bödrich, 2012). FRANKE et al. tested the library with nonlinear magnetic networks, but did not include eddy current models (Franke, 2012). However, the `FluxTubes` library has not been tested yet on the literature with more complex magnetic networks consisting of eddy current and magnetic field displacement models, which are inevitable for all types of fast switching solenoids.

The present work has tried to apply the *MSL* `FluxTubes` library to medium sized magnetic networks with a total of 50 to 60 elements, containing nonlinear

DOI 10.3384/ecp202017449

ferromagnetic material behaviour as well as eddy current elements without the use of the PREISACH or TELLINEN hysteresis models. *Dymola* and *OpenModelica* failed either in determining the causality or in solving the resulting system of equations, depending on software version and magnetic network structure, until the eddy current elements were excluded from the network[1]. A reason for that could not finally be determined despite the fact, that the use of eddy current elements seems to be one trigger. Therefore, in the following section an approach based on the consistent magnetic analogy is being developed, with the aim of higher solvability when applied to complex eddy current enhanced networks.

2.2 A new Modelica magnetics library based on the consistent magnetic analogy

Implementation The `connector` class of the new, `FluxTubes_PhiD`-called library basically has to use the time derivative of the magnetic flux $\dot{\Phi}$ as the new flow variable. Even with this change of connection variables, the KIRCHHOFF current law (all flows sum to zero at every node) still applies, what can be shown by differentiation:

$$0 = \frac{d}{dt}\left(\sum_i \Phi_i\right) = \sum_i \dot{\Phi}_i. \tag{4}$$

On the other hand, the changed formulation of the reluctances introduces one new state for every reluctance element, which now in general needs an additional initial value for integration and hence complicates initialisation. When considering non-excited magnetic circuits with $\Theta = 0$ in steady state, it becomes clear that all magnetic potential differences $\Delta V_{\mathrm{m,j}}$ must be initialised with zero, as long as the magnetic circuit is unpolarised (Kallenbach et al., 2017, pp. 61 – 63). In every other non-trivial case the initialisation problem is not well defined exclusively via the consistent analogy, as the conditions for initialisation get lost due to the derivation of the magnetic flux in the connectors. An additional use of the classic analogy only for initialisation could be promising, however, this is left to future research. Therefore, simulations in this paper all start with a non-excited magnetic circuit.

Similar to this, the linked magnetic flux Ψ can not be calculated out of the momentarily total flux Φ by

$$\Psi = N\,\Phi, \tag{5}$$

but instead needs to be integrated from the flux derivative:

$$\Psi = \int_{t_0}^{t_\mathrm{f}} \dot{\Psi} = N \cdot \int_{t_0}^{t_\mathrm{f}} \dot{\Phi}. \tag{6}$$

Again, in steady state initialisation the initial value for the linked flux is assumed to be zero. Since the linked flux does only serve as an output variable and is not being used in other equations, this integration over the simulation period is not causing precision losses elsewhere. The same holds for the flux density B of each reluctance pathway l, which needs to be calculated via the magnetic permeability $\mu = \mu_0\mu_\mathrm{r}$ out of the field strength H:

$$B = \mu_0\mu_\mathrm{r}H = \mu_0\mu_\mathrm{r}\frac{\Delta V_\mathrm{m}}{l}. \tag{7}$$

As the consistent analogy does not *swap* potential and flow variable with each other, the topology of the network remains the same (Littmann and Schiedeck, 2008). Therefore in a third step only the *component equations* have to be reformulated according to Table 1. It should be noted though that in case of ferromagnetic materials, starting from HOPKINSON's law, the time derivative of the magnetic permeance $G_\mathrm{m} \equiv 1/R_\mathrm{m}$ can not be neglected in general:

$$\begin{aligned} \Phi &= G_\mathrm{m}V_\mathrm{m}, \\ \dot{\Phi} &= \dot{(G_\mathrm{m}V_\mathrm{m})} = \dot{G}_\mathrm{m}V_\mathrm{m} + G_\mathrm{m}\dot{V}_\mathrm{m} \\ \text{with } \dot{G}_\mathrm{m} &= \frac{dG_\mathrm{m}}{dt} = \frac{\partial G_\mathrm{m}}{\partial \Phi}\cdot\frac{\partial \Phi}{\partial t}. \end{aligned} \tag{8}$$

However, in case of linear, structure-invariant reluctances of constant pathway length the term $\dot{G}_\mathrm{m}V_\mathrm{m}$ becomes zero and hence the necessary iteration depth for the numerical solution decreases. Here, the `Evaluate = true` flag should be provided for efficient code generation, since the assumption $\dot{G}_\mathrm{m}V_\mathrm{m} = 0$ does only hold for special parameterisation cases.

If, in addition, the magnetic permeability correlation $\mu_\mathrm{r}(H)$ for ferromagnetic reluctances is inserted into the term $\partial G_\mathrm{m}/\partial\Phi$ and solved manually instead of being implemented into a causalised `function`, Equation (8) can be converted into the less implicit shape

$$\dot{\Phi} = C_\mathrm{m}(V_\mathrm{m})\,\dot{V}_\mathrm{m}, \tag{9}$$

where C_m is nonlinear in V_m, but the time derivative $\dot{G}_\mathrm{m}$ is disappearing. Herewith, the numerical integration of ferromagnetic reluctances can be accelerated due to fewer numerical root searches, compared to Equation (8). This has been implemented as a separate library called `FluxTubes_PhiD_fast` with the drawback of a non-interchangeable permeability function.

Validation The Modelica libraries described above have been tested against the *MSL* `FluxTubes` implementation as well as a commercial simulation program called *SESAM* (Birli et al., 2003) in order to determine possible errors in the new implementation. This has been done by comparing the calculated reluctance forces for the example network `SimpleSolenoid` of the `FluxTubes` library (Bödrich, 2008). Apart from numerical aberrations, no systematic deviations in the network calculations have been noticed. In the simulations described later, also no drift due to using the flux derivative has been noticed.

[1] This tests were done with *Dymola 5.3e*, *6.0b*, *2014 FD01* and *2015 FD01 64-bit* as well as *OpenModelica 1.9.1*, *1.9.2 Beta1* and different *pre-1.9.2* developer versions.

Table 2. Solvability and integration times (in seconds) of different implementation approaches of a magnetic library.

		Quasistatic problem		Dynamic problem	
Library	**Functionality**	**W/o eddy cur.**	**W/ eddy cur.**	**W/o eddy cur.**	**W/ eddy cur.**
`Modelica...FluxTubes` (v3.2.1)	classic analogy, $\mu_r(B)$ as function	*Dymola*: — *OM*: —	*Dymola*: — *OM*: —	*Dymola*: — *OM*: 8.6	*Dymola*: — *OM*: —
`Modelica...FluxTubes` (v3.2.2 pre-release)	classic analogy with magnetic hysteresis	*Dymola*: — *OM*: —	*Dymola*: — *OM*: —	*Dymola*: 58 *OM*: —	*Dymola*: — *OM*: —
`FluxTubes_MuRAsEq`	classic analogy, $\mu_r(B)$ as equation	*Dymola*: 0.16 *OM*: —	*Dymola*: — *OM*: —	*Dymola*: 2.3 *OM*: 12	*Dymola*: — *OM*: —
`FluxTubes_MuROfH`	classic analogy, $\mu_r(H)$ as equation	*Dymola*: 5.1 *OM*: —	*Dymola*: — *OM*: —	*Dymola*: 16 *OM*: 203	*Dymola*: — *OM*: —
`FluxTubes_PhiD`	consistent analogy, $\mu_r(H)$ as equation, cf. Equation (8)	*Dymola*: 0.89 *OM*: 0.97	*Dymola*: 1.4 *OM*: 1.6	*Dymola*: 14 *OM*: 19	*Dymola*: 25 *OM*: 27
`FluxTubes_PhiD_fast`	consistent analogy, $\mu_r(H)$ as equation, cf. Equation (9)	*Dymola*: 0.85 *OM*: 0.86	*Dymola*: 1.3 *OM*: 1.3	*Dymola*: 13 *OM*: 13	*Dymola*: 23 *OM*: 22

In a second step, several different implementations of magnetic libraries underwent an analysis w.r.t. the solvability of complex magnetic networks. For this, two different circuits – one with consideration of eddy currents and the other one without – were prepared based on the lumped-parameter model of the injector presented in Section 2.3. These networks were tested with quasistationary boundary conditions (fixed armature movement) and in dynamic configuration (free moving armature). Additionally to the consistent- and classic-based libraries described above, other implementation types of the same analogies were tested, which differ only in the way the magnetic permeability gets computed. This is due to the fact, that the iterative calculation of the nonlinear material characteristic $\mu_r(H)$ or $\mu_r(B)$ has been observed to be another crucial point for solving the network. Table 2 summarises all herewith tested libraries with a short explanation of the implementation details and provides the calculation timings, if solving the network was possible[2].

First of all, the two implementations of the consistent analogy described above – `FluxTubes_PhiD` and `FluxTubes_PhiD_fast` – lead to a solvable system of equations in all test cases and with both tested programs. The variant based on Equation (9) performs moderately better with speed advantages between 5 % and 30 %.

The *MSL* libraries based on the classic analogy seem to be harder to solve than the consistent analogy; the simulation process fails in most cases, either due to problems with causalisation or simulation. The few test cases, in which the simulation succeeds without complications, allow the conclusion that the consistent analogy is considerably more time-intensive to solve, which can be explained by the additional state variables $\dot{V}_m$ being introduced in the reluctances. Separate comparative simulations of small networks indicate the consistent analogy implementation performing almost an order of magnitude slower.

Inside the consistent analogy based libraries the permeability μ_r is expressed via the magnetic field strength H instead of the flux density B. Additionally, the correlation $\mu_r(H)$ is provided as an `equation` rather than a causalised `function`. In order to check, whether the definition of the permeance has an influence on the solvability, the two additional libraries `FluxTubes_MuRAsEq` and `FluxTubes_MuROfH` based on the classic analogy have been tested as well (c.f. Table 2). Herewith the solvability problems are still present, so that this approach seems also not promising. Furthermore, a separate test with the permeance expressed by an interpolation table resulted even worse, which is understandable due to more complicated differentiation of the look-up table output.

Hence, for the tested compiler versions of *Dymola* and *OpenModelica* the implementations based on the consistent analogy are assumed to be stable enough for modelling of complex magnetic circuits. The following simulations are all based on these libraries. It should be noted though, that Table 2 delivers only a snap-shot in terms of tested software, compiler versions and the specific magnetic network. E.g., in *OpenModelica 1.14.2* occasionally some problems with the `FluxTubes_PhiD_fast` library are occurring, whereas the library `FluxTubes_PhiD` shows no problems.

2.3 Deriving the lumped-parameter network

Field displacement In ferromagnetic materials usually eddy currents are evoked proportionally to the magnetic flux derivative $\dot{\Phi}$. Moreover, in axially permeated solids of revolution a field displacement effect similar to the *skin effect* can be observed, which leads to a magnetic field "invading" from the outermost to the innermost radius. This is caused by induced rotating electric currents, which again excite a magnetic potential ΔV_m against the initial direction. Herewith, the radially inhomogeneous field with local saturations of the ferromagnetic material leads to

[2]These and the following simulations were done on an *Intel Core i7-980X* processor. The applied software was *Dymola 2015 FD01 64-bit* and *OpenModelica r22929* (abbreviated with *OM* in the table). *DASSL* was used for integration with a tolerance of 10^{-7}. The given simulation timings were averaged out of 3 trials.

DOI 10.3384/ecp202017449

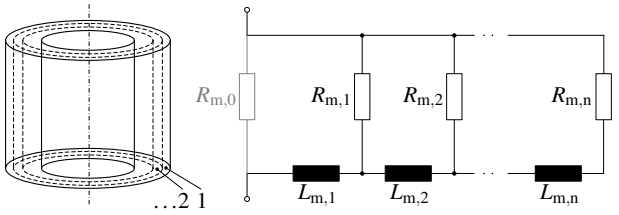


Figure 4. Discretised structure of the field displacement model for axially permeated hollow cylinders.

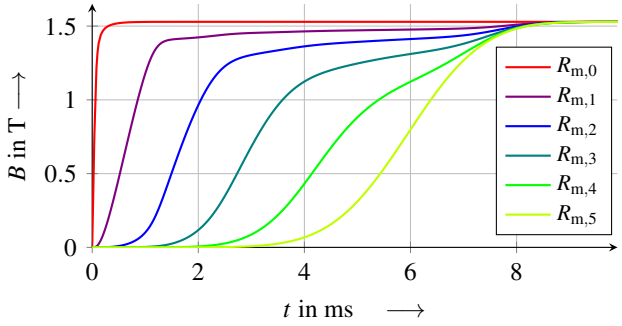


Figure 5. Flux density through the shells as a response to a rapid change in the field strength H with nonlinear material.

varying reluctance forces of the armature.

For this effect STRÖHLA found a substitute network with cascading reluctance and eddy current elements, as depicted in Figure 4 (Ströhla, 2002). Each pair of reluctance R_m and "magnetic inductance" L_m represents one shell of the body with constant cross sectional area (Kallenbach et al., 2017, p. 170). In order to prevent discontinuous coil currents, which can not be observed in reality, a small additional reluctance $R_{m,0}$ should be inserted in a way, that the overall reluctance magnitude stays the same. In reality, this fact is given by the variety of stray fields over non-conductive materials of the actuator. Depending on the wall thickness of the hollow cylinder a discretisation of $n \in [3,5]$ is sufficient for most applications, the result of which is demonstrated in Figure 5. The described model is implemented in the new libraries with variable discretisation depth n and propagation of the model parameters to the inner components, so that only the electric resistivity has to be provided additionally to regular reluctance elements.

Magnetic network With all basic elements and libraries being implemented, the lumped-parameter network can be derived from the injector geometry, the result of which is depicted in Figure 6. Like with all pot-shaped solenoids the reluctance pathways can be reduced to a two-dimensional, rotationally symmetric form, where the use of radially and axially permeated reluctances is sufficient. In case of highly saturated solenoids, stray fields over the coil area have to be taken into account (c.f. white elements in Figure 6), since the saturated ferromagnetic permeability converges to the vacuum permeability μ_0. The inclusion of different stray pathways usually needs an iterative procedure, while for the coil unified approaches are known (Ströhla, 2012, pp. 35 – 42). The more saturated parts of the solenoid exist, the less accurate the lumped-parameter approach will be.

The radially permeated reluctance elements in Figure 6 are modelled via an additional eddy current element, whose current pathway is assumed to be along the circumference of the mid radius. The majority of the remaining ferromagnetic pathways are represented by the field displacement models. However, one major drawback of this network topology is the fact, that the field displacement models of outer hollow cylinders do not influence the inner ones. Therefore, the total eddy current losses as well as the local saturation effects are presumably estimated too small in the model.

Validation In order to evaluate the accuracy of the magnetic network yet without other physical domains, a stationary comparison between a *finite element model* and the Modelica model is shown in Figure 7. Starting with the linked magnetic flux Ψ, the influence of saturations in the injector is obvious, as the model assumes the magnetic excitation as too small. This is a common aberration when transforming partial differential equations into a system of ordinary differential equations accompanied with lumped parameters in a network[3], since the simplification of magnetic pathways results in underestimating the total permeance of the system, as long as no pathways are mapped redundantly. On the other hand, the progression of the linked magnetic flux Ψ w.r.t. the air gap δ_m is well described by the Modelica model.

The comparison of the reluctance force F_m shows a similar result, but with higher divergence between the depicted electric excitations. Here, the discretisation of the working air gaps (c.f. Figure 6) shows a high sensitivity w.r.t. the shape of the reluctance force curve. The serial combination of each air gap model with a short ferromagnetic pathway for modelling local saturation effects is also fundamental.

Further increased accuracy could be received by optimising the network manually based on the magnetostatic results or by conducting a parameter identification with the help of numerical optimisation algorithms (Mühlenhoff et al., 2016). However, this would be beyond the scope of the network modelling approach, since a *finite element model* is more suitable in cases, where calculation times aren't a problem. Since the overall model aberrations in Figure 7 lie within 5 % to 10 %, the network based approach is sufficient for the desired purpose of optimising the injector control.

3 Modelling the mechanic domain

Having implemented electric and magnetic domains, the mechanical behaviour including contacts is left for re-

[3] E.g., the RAYLEIGH-RITZ-method for approximating the eigenfrequency in vibration mechanics leads to an estimation, which is always higher or the same as in the original system.

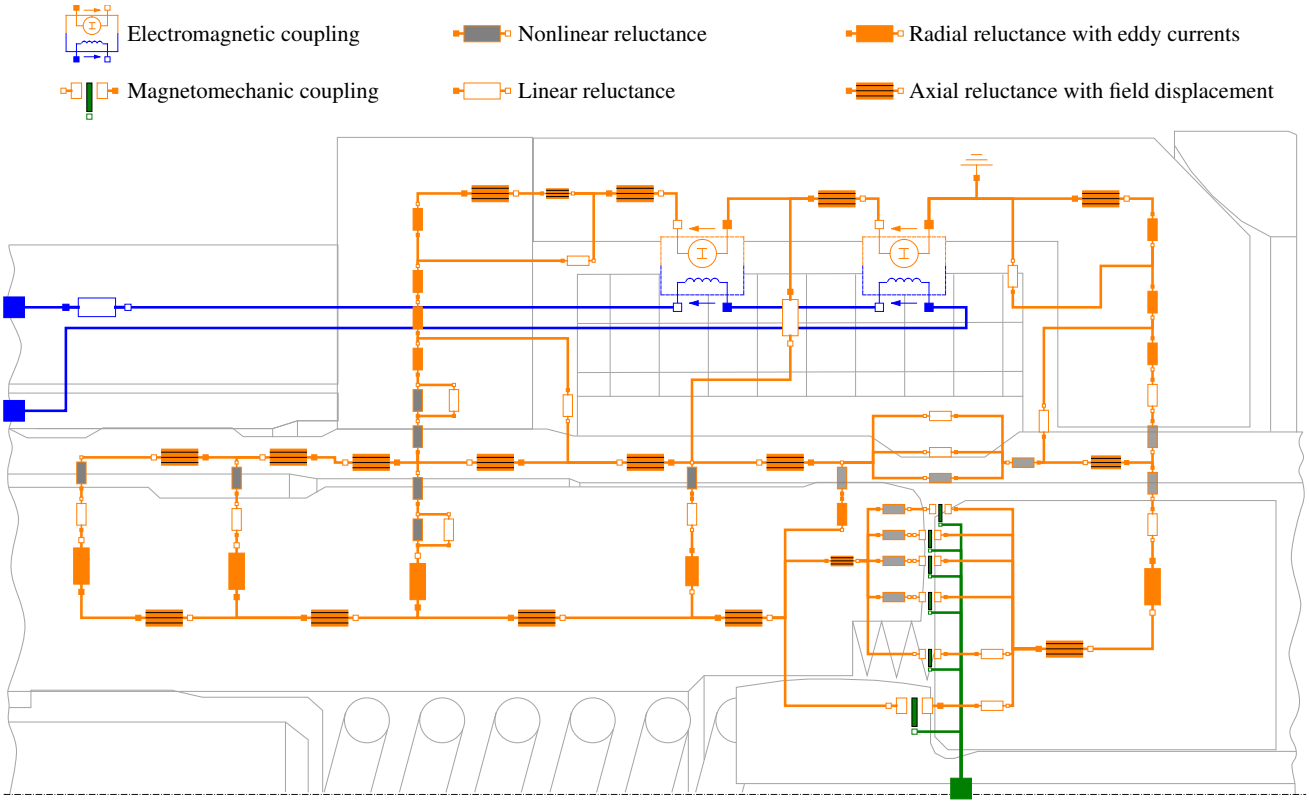


Figure 6. Lumped-parameter network for the magnetic domain of the injector with an underlying sketch of the internal parts.

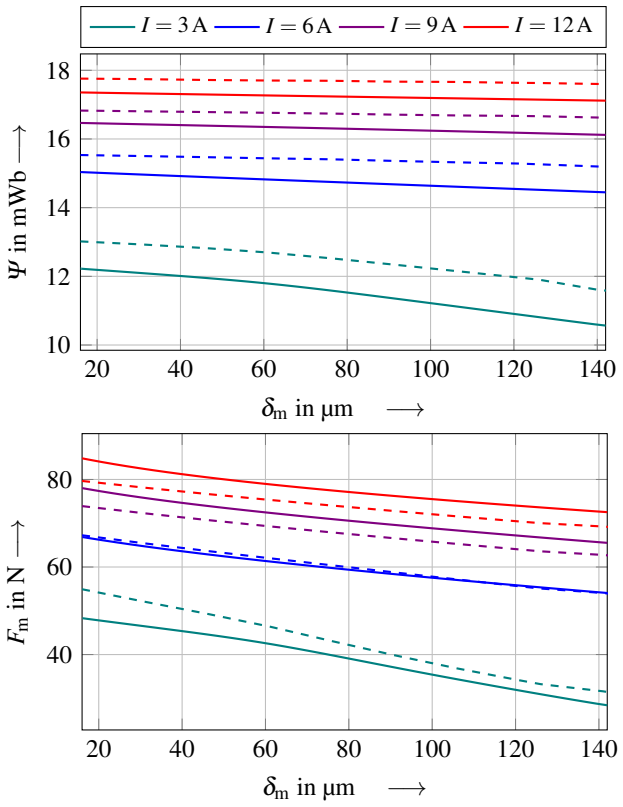


Figure 7. Quasistationary validation of the linked magnetic flux Ψ (top) and the reluctance force F_m (bottom) against a *finite element analysis* (dashed lines) at different electric excitations.

search. The movements of armature and valve-needle can easily be described by one-dimensional and translational equations of motion and do not require further investigation. On the other hand, contact problems in mechanics are known for their numerical stiffness and application-specific implementations (Tiller, 2001, pp. 162 – 166). Therefore, in this chapter different discrete and continuous contact models will be compared in terms of modelling effort and integration performance.

The injector to be modelled has a total amount of four end stops, with two of them acting in-between the bodies. Therefore, both masses are restricted in terms of position. Figure 8 shows the topology of the contacts neglecting other mechanical elements, such as acting forces.

3.1 Restitution-based contact models

One approach for modelling kinetic contacts is to reinitialise the state variables for the velocity v, whenever a previously defined boolean contact condition gets fulfilled. The principle of conservation of linear momentum is then used in order to determine the velocities v' *after* contact. Therefore, this approach is limited to simultaneous contact of only two masses. If a partial loss of energy is assumed by introducing a restitution coefficient $c_\mathrm{r} \in [0,1]$, the reinitialisation can be described by

$$v_1' := \frac{m_1 v_1 + m_2 v_2 - c_\mathrm{r} m_2 (v_1 - v_2)}{m_1 + m_2} \tag{10}$$

and vice versa (Gross et al., 2004, p. 93).

DOI 10.3384/ecp202017449

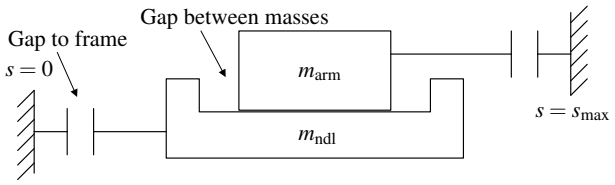


Figure 8. Topological structure of the four end stops between armature m_{arm}, valve-needle m_{ndl} and the mechanical frame.

This hybrid model is known for its ZENO-behaviour, which leads to infinite high computation effort to approximate a stationary contact (Heymann et al., 2005), since the static contact constraint of resting masses is not explicitly considered. Another numerical problem arises when bodies begin to fall into each other, caused by the reinitialised velocities being too small. This is often treated by detecting invalid velocities after reinitialisation and assigning the corresponding acceleration to zero for simulating "sticking" behaviour (Tiller, 2001, p. 164).

For the injector implementation, one has to use a state graph with a total of 19 each-exclusive boolean states and 41 transition conditions, in order to cover all possible combinations of contact situations. The state machine has been implemented by hand into the mechanics model without the use of specific state graph libraries. Though, the resulting model performed poor w.r.t. integration performance, as Table 4 shows[4]. In order to cover all these drawbacks and to enable reuse of the models in an object-oriented modelling fashion, the following section investigates different non-phenomenological models.

3.2 Force-based contact models

KELVIN-VOIGT models Force-based contact models share the basic principle of a contact force F_c being composed out of a capacitive and a dissipative part as functions of the indentation δ. The capacitive element can be modelled by a stiff linear or nonlinear spring force F_s, which represents the deformation of both contact surfaces. The dissipative part F_d is needed in order to consider energy losses. In case of a constant damping coefficient d and stiffness c, we get the linear KELVIN-VOIGT contact model (Machado et al., 2012):

$$F_c = F_s + F_d = \begin{cases} 0 & \delta < 0 \\ -c\delta - d\dot{\delta} & \delta \geq 0 \end{cases}. \qquad (11)$$

Because contact forces are intended to be always negative or zero – as long as no adhesion of the colliding bodies should occur – we can modify Equation (11) to

$$F_c = \min\left(0, F_s + F_d\right), \qquad (12)$$

which leads to substantially higher contact exit velocities. Figure 9 shows these resulting differences in the force-indentation-diagram. Nevertheless, both variants still exhibit the problem of discontinuous forces at contact start, resulting from the dissipative term in Equation (11), which is assumed to be unphysical (Machado et al., 2012). The `ElastoGap` model of the *MSL* handles this by multiplying the damping term $d\dot{\delta}$ with the indentation δ (c.f. Figure 9).

[4] *DASSL* was used as integration method with a tolerance of 10^{-5}. The failed simulations exhibited convergence problems at contact time points. The given simulation timings were averaged out of 3 trials.

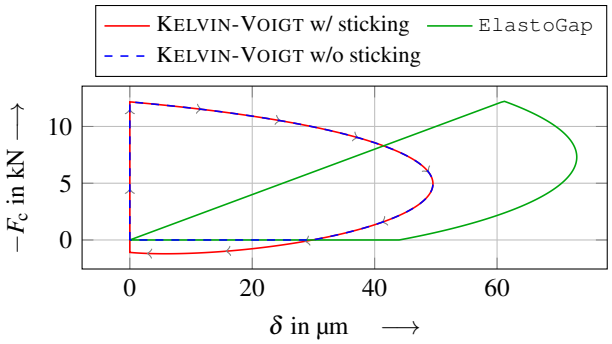


Figure 9. Trajectories of the contact force F_c versus the indentation δ for different KELVIN-VOIGT-based models.

However, the primary problem of the KELVIN-VOIGT-based models is the parameterisation of the damping coefficient d. While the stiffness rate c can be guessed reasonably via the surface geometries, there are no accurate approaches known for estimation of the damping constant as a function of the restitution coefficient c_r, which can be measured more easily and is less dependent on the initial impact velocity. In contrast, the damping constant d strongly depends on the relative velocities and therefore the model can not be deployed in varying boundary conditions with reasonable validity.

Hysteresis damping models This parameter identification problem can be handled by relating the damping term explicitly to the desired coefficient of restitution c_r (Machado et al., 2012). Starting from the purely elastic HERTZ'ian surface deformation force $K\delta^n$, a non-constant *hysteresis damping factor* χ is introduced for calculation of the damping part:

$$F_c = -K\delta^n - \chi\delta^n\dot{\delta} \quad \text{for } \delta > 0. \qquad (13)$$

In order to adapt the damping term to different contact situations and to receive a constant coefficient of restitution, the relative entrance velocity $\dot{\delta}^{(-)}$ of both masses at contact start is additionally used for the calculation of the hysteresis damping factor. Table 3 shows a selection of different definitions of the hysteresis damping factor on the literature (Machado et al., 2012).

In the *Modelica* implementation the hysteresis damping factor χ can be calculated by a conventional equation, as long as $\dot{\delta}^{(-)}$ is defined at all simulation times including initialisation. However, the initial impact velocity $\dot{\delta}^{(-)}$ has to be assigned within an algorithmic section at discrete time points whenever a new contact starts. Although this model still has discrete and algorithmic parts as well as requires for state events, the numerical integrations were stable, as long as the tolerances were set

Table 3. Definition of different hysteresis damping models.

Model	Damping factor
FLORES et al.	$\chi = \frac{8(1-c_\mathrm{r})}{5c_\mathrm{r}} \frac{K}{\dot{\delta}^{(-)}}$
GONTHIER et al.	$\chi = \frac{D}{c_\mathrm{r}} \frac{K}{\dot{\delta}^{(-)}}; 1 + \frac{D}{c_\mathrm{r}(1-D)} = e^{D(1+1/c_\mathrm{r})}$
HERBERT et al.	$\chi = \frac{6(1-c_\mathrm{r})}{(2c_\mathrm{r}-1)^2+3} \frac{K}{\dot{\delta}^{(-)}}$
HUNT et al.	$\chi = \frac{3(1-c_\mathrm{r})}{2} \frac{K}{\dot{\delta}^{(-)}}$
LANKARANI et al.	$\chi = \frac{3(1-c_\mathrm{r})}{4} \frac{K}{\dot{\delta}^{(-)}}$
ZHIYING et al.	$\chi = \frac{3(1-c_\mathrm{r}^2)e^{2(1-c_\mathrm{r})}}{4} \frac{K}{\dot{\delta}^{(-)}}$

Table 4. Calculation times of the total injector model with different types of contact models in two simulation setups.

	Injection cycle		Dyn. validation	
Contact model	**Dymola**	**OM**	**Dymola**	**OM**
Restitution-based	12.2 s	11.5 s	35.0 s	*failed*
`ElastoGap`	7.04 s	6.88 s	10.8 s	11.7 s
GONTHIER et al.	7.26 s	7.38 s	11.6 s	13.2 s

appropriate and an implicit integration method was chosen. With higher integration tolerances the model tended to show wrong dissipative behaviour as a result of the stiff and short-timed impact. Due to the ZENO-behaviour of the restitution-based model, the hysteresis damping approach can be solved faster, while the integration duration is on par with the `ElastoGap` model (c.f. Table 4).

Figure 10 shows the resulting trajectories of contact force versus indentation. The different definitions of the hysteresis damping factors lead to varying energy losses and therefore also varying exit velocities. For the injector, the model of GONTHIER et al. showed the best results in terms of the reached restitution coefficient, compared to the desired one ($c_{\mathrm{r,desired}} = 0.3$, $c_{\mathrm{r,reached.}} \approx 0.295$).

The specific benefit of all hysteresis damping models is a constant percentage of velocity losses during an impact, which does not vary when used in different impact situations with different masses and initial velocities. Furthermore, during contact the models are neither showing sticking behaviour nor uncontinuous forces. If one mass gets lifted out of a previously resting contact, the consideration of Equation (12) is still necessary, though.

3.3 Validation of the dynamic model

In order to validate the accuracy of the complete electro-magneto-mechanical *Modelica* injector model, Figure 11 shows a comparison against a measurement of the Ψ-I-characteristics of the injector. Due to the lower electric excitation frequency of 7 Hz in the measurement, the eddy current phenomena are not as significant as they are in the real injection cycle. Instead, the missing consideration of static magnetic hysteresis is apparent, however, the influence of which will get smaller the faster the electric excitation will be. On the other hand, the roundness and symmetric position of the central hysteresis loop indicates an acceptable quality of the combined effects of the total model.

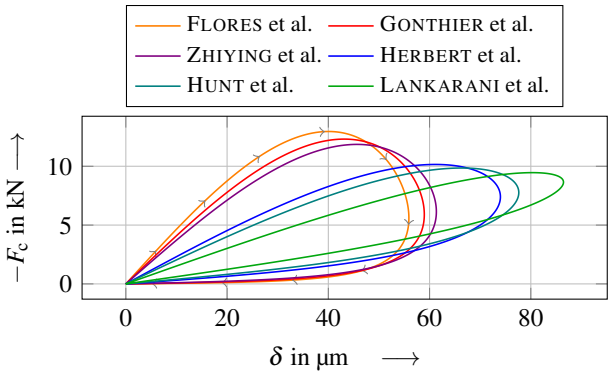


Figure 10. Trajectories of the contact force F_c versus the indentation δ for different hysteresis damping models with $c_\mathrm{r} = 0.3$.

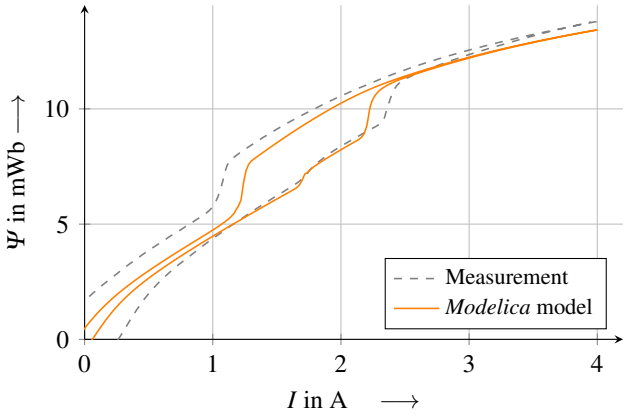


Figure 11. Dynamic validation of the *Modelica* model against a measurement basing on the Ψ-I-characteristic at $f = 7$ Hz.

4 Approaches for needle control

In order to avoid harmful bouncing of armature and valve-needle as well as injecting minimal amounts of fuel in ballistic mode, research on possible alternative control schemes of the injector needs to be done. The principle idea of controlling the trajectory of moving armatures is not new, though. KIRSCHBAUM investigated solenoid intake valves in resonant operating mode and applied dynamic optimisation methods to different types of models, in order to reach the end stop with a predefined velocity (Kirschbaum, 2001). Similar research on conventional intake valves was done by SCHIEDECK, where a heuristic optimisation method provided the parameterisation of a predefined control scheme (Schiedeck, 2003). With a similar aim, two *Modelica*-specific methods for synthesis of the injector control voltage are given below.

Model inversion Due to the acausality of *Modelica*, the simplest approach is to invert the actuator model by defin-

DOI 10.3384/ecp202017449

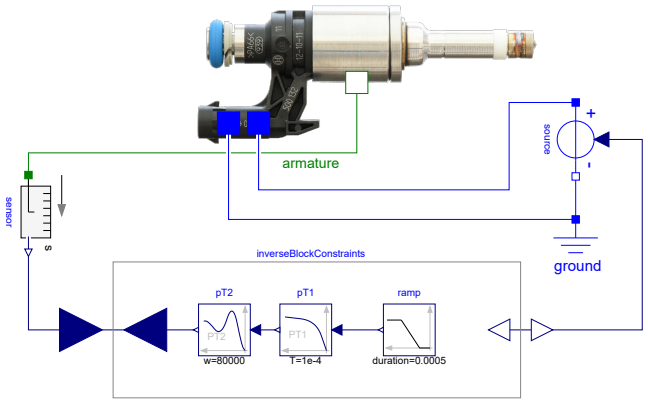


Figure 12. Structure of the inverted model.

ing the "output" movement and leaving the necessary voltage supply indefinite, as it is shown in Figure 12, thus making use of the automatic symbolic manipulation of *Modelica* compilers. The time delaying influence of the electrical (first order) as well as the mechanical system (second order) requires a three times differentiable signal for the armature position. Furthermore, a simplified magnetic model based on look-up tables as well as a modified hysteresis damping model with a constant damping factor is used in order to ease the numerical solution.

Figure 13 shows the results of the inverted solution with a ramp for the desired, yet unfiltered armature movement and the corresponding movement s_{arm} for opening the valve. While the acausality of *Modelica* enables such inverted solutions at low calculation times, the results in case of the examined injector are largely affected by bouncing effects related to the two-stage movement (c.f. Figure 13, $t \approx 0.61\,\text{s}$). Since the inverted model delivers only exact solutions in terms of the specific movement target and does not allow competing objectives, this method is more appropriate for conventional solenoids without body-to-body impacts.

Optimal control Optimisation techniques can handle such problems, as they are formulated as a minimisation of a scalar objective function J, which can account for several conflicting aims. A possible objective function for optimising the trajectory of one full injection cycle can be expressed by a combination of three criteria with their corresponding scaling factors k:

$$J = J_{\text{Penalty}} + k_{\text{Fuel}}\, J_{\text{Fuel}} + k_{\text{Impact}}\, J_{\text{Impact}}. \qquad (14)$$

In order to guarantee a fully opened cycle as an output of the optimisation, the term J_{Penalty} is used as a penalty, if the armature is not reaching the end stop. The mid term ensures that a certain amount of fuel gets injected; due to the lack of a fluid dynamics model of the injector, this has been approximated by determining a desired time for the valve being opened. The criterium J_{Impact} finally summarises all individual impact velocities $\dot{\delta}^{(-)}$, which take place during the full cycle.

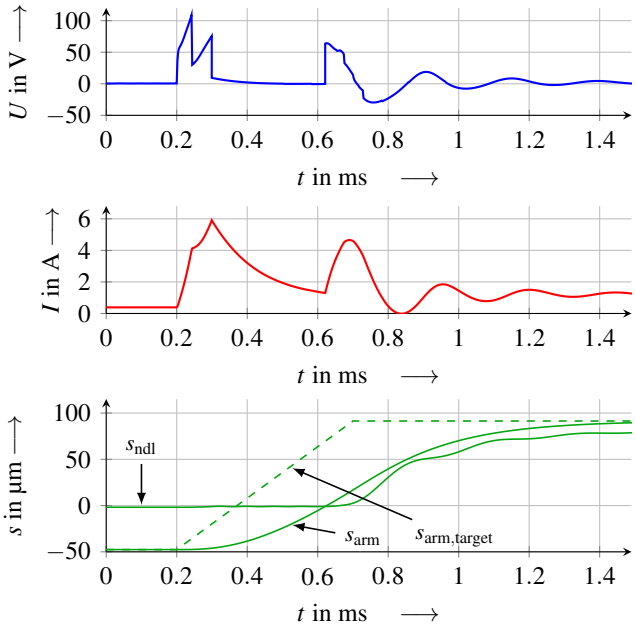


Figure 13. Controls and trajectories as a result of the inverted injector model: control voltage U (top), coil current I (mid) as well as the strokes of armature s_{arm} and valve-needle s_{ndl} with the unfiltered target of armature movement $s_{\text{arm,target}}$ (bottom).

In the present work, the complete *Modelica* injector model has been compiled to a *FMU* by the use of *OpenModelica* and then was imported to *JModelica.org*. For the optimisation, the control voltage U needs to be discretised over time and used as the variational input. The NELDER-MEAD heuristic optimisation algorithm (Gedda et al., 2012) can be used afterwards for the minimisation of Equation 14, which leads to the results depicted in Figure 14 with the desired smooth armature movement. With further modifications to the objective function, the optimal control problem can be adjusted to nearly every physical demand like, e.g., *safely capturing* the armature with a predefined first impact velocity. On the other hand the calculation times are considerably higher than the ones of the inverted model, with about 15 h on an 8 core processor[5]. However, many further improvements are imaginable, e.g., the use of a simpler model plus a refinement of the control with the complete model afterwards.

5 Conclusion

In this paper a new *Modelica* magnetics library has been presented, which uses an energy-consistent systems modelling analogy. Herewith, the solvability of complex electromagnetic networks including field displacement effects increases in *Dymola* and *OpenModelica* with a drawback of higher integration times. Though, future work should concentrate on the specific compilation problems as well as the differences across different compiler versions. In terms of contact mechanics, different models for simulat-

[5]The optimisations were done on two *Intel Xeon W5580* processors with a dual socket system.

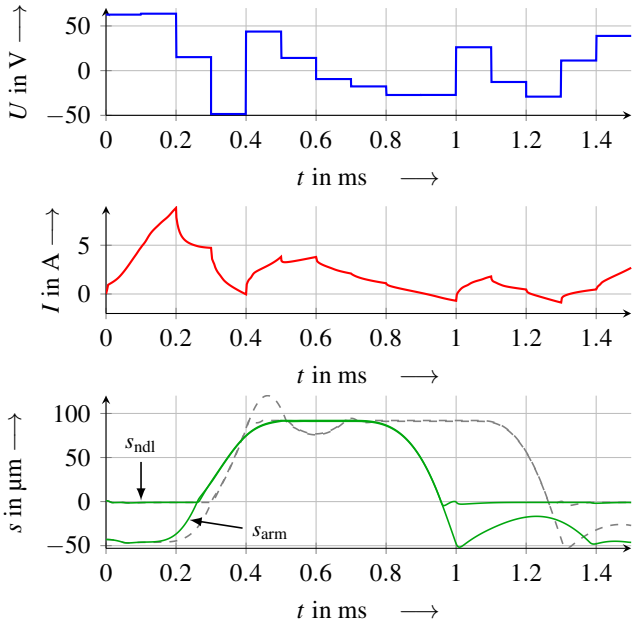


Figure 14. Controls and trajectories as a result of the *JModelica.org* optimisation: control voltage U (top), coil current I (mid) as well as the strokes of armature s_{arm} and valve-needle s_{ndl} with the standard control scheme in grey (bottom).

ing impacts have been implemented and compared, with the *MSL* `ElastoGap` model being less accurate. The hysteresis damping model of GONTHIER et al. showed the best compromise between integration times, ease of parametrisation and physical representation quality. The solenoid model was used for control synthesis, where especially the heuristic optimisation methods are found to be appropriate for the complex engineering demands in mechatronics and especially actuator development. The application and testing of the found control schemes on real solenoids and injectors are left to future work.

Open-source data

The magnetic and contact mechanic libraries are published by the Digital Library Thuringia *DBT*[6].

References

Horst Bauer, Karl-Heinz Dietsche, Thomas Jäger, and Sven A. Hutter, editors. *Gasoline-Engine Management*. Robert Bosch GmbH, 2nd edition, 2004.

Oliver Birli, Karsten Feindt, Eberhard Kallenbach, and Tom Ströhla. SESAM – a network-based design tool for developing electromagnetic actuators. In *Proc. of the Internat. Conf. on Mechatronics*, pages 53 – 59, 2003.

Thomas Bödrich. Electromagnetic actuator modelling with the extended Modelica magnetic library. In *Proc. of the 6th Internat. Modelica Conf.*, pages 221 – 227, 2008.

Thomas Bödrich and Thomas Roschke. A magnetic library for Modelica. In *Proc. of the 4th Internat. Modelica Conf.*, pages 559 – 565, 2005.

[6] `https://www.db-thueringen.de/receive/dbt_mods_00046032`

Frank Denk. *Elektromagnetische Einspritzventile für Ottomotoren*. PhD thesis, TU Ilmenau, 2018.

Marco Franke. *Multidisziplinäre Modellierung und Simulation eines Rolling Rotor Switched Reluktanz Antriebes*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2012.

Sofia Gedda, Christian Andersson, Johan Åkesson, and Stefan Diehl. Derivative-free parameter optimization of functional mock-up units. In *Proc. of the 9th Internat. Modelica Conf.*, pages 819 – 827, 2012. doi:10.3384/ecp12076819.

Jörg Grabow. *Verallgemeinerte Netzwerke in der Mechatronik*. De Gruyter, 2013. doi:10.1524/9783486719826.

Dietmar Gross, Werner Hauger, Walter Schnell, and Jörg Schröder. *Technische Mechanik 3. Kinetik*. Springer, 8th edition, 2004. doi:10.1007/3-540-34995-2.

Michael Heymann, Feng Lin, George Meyer, and Stefan Resmerita. Analysis of zeno behaviors in a class of hybrid systems. *IEEE Transactions on Automatic Control*, 50(3): 376 – 383, 2005. doi:10.1109/TAC.2005.843874.

Eberhard Kallenbach, Rüdiger Eick, Tom Ströhla, Karsten Feindt, Matthias Kallenbach, and Oliver Radler. *Elektromagnete. Grundlagen, Berechnung, Entwurf und Anwendung*. Springer Vieweg, 5th edition, 2017. doi:10.1007/978-3-658-14788-4.

Frank Kirschbaum. *Modellbildung und dynamische Optimierung schnellschaltender magnetomechanischer Aktoren*. PhD thesis, Universität-Gesamthochschule Siegen, 2001.

Walter Littmann and Florian Schiedeck. New mechatronic analogy in magnetics for modeling of electro-magento-mechanical systems. In *Forschung im Ingenieurwesen*, volume 72, pages 121 – 133. Springer, 2008. doi:10.1007/s10010-008-0074-z.

Margarida Machado, Pedro Moreira, Paulo Flores, and Hamid M. Lankarani. Compliant contact force models in multibody dynamics: Evolution of the hertz contact theory. In *Mechanism and Machine Theory*, volume 53, pages 99 – 121. Elsevier, 2012. doi:10.1016/j.mechmachtheory.2012.02.010.

Julian Mühlenhoff, Emanuel Rauer, and Tom Ströhla. Modellierung von Reluktanzantrieben mit Modelica. In *5. Ilmenauer Magnettag*, 2016.

Emanuel Rauer, Max Rotter, Michael Willmann, and Clemens Senghaas. Intelligent injectors – digitization in the injection technology of large engines. *MTZ worldwide*, 80(6):58 – 65, 2019. doi:10.1007/s38313-019-0038-8.

Florian Schiedeck. *Untersuchungen zur Flugregelung eines Magnetaktors als elektromagnetischer Ventiltrieb einer vollvariablen Ventilsteuerung*. Diploma thesis, TU Ilmenau, 2003.

Tom Ströhla. *Ein Beitrag zur Simulation und zum Entwurf von elektromagnetischen Systemen mit Hilfe der Netzwerkmethode*. PhD thesis, TU Ilmenau, 2002.

Tom Ströhla. *Vorteile, Potential und Grenzen einfacher Modelle im Entwurf elektromagnetischer Energiewandler im Umfeld leistungsfähiger Rechenhilfsmittel*. Habilitation thesis, TU Ilmenau, 2012.

Michael Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, 2001. doi:10.1007/978-1-4615-1561-6.

Dirk Zimmer and François E. Cellier. The modelica multi-bond graph library. In *Proc. of the 5th Internat. Modelica Conf.*, pages 559 – 568, 2006.

Johannes Ziske and Thomas Bödrich. Magnetic hysteresis models for Modelica. In *Proc. of the 9th Internat. Modelica Conf.*, pages 151 – 158, 2012. doi:10.3384/ecp12076151.

Simulating the Dynamics of a Chain Suspended Sub-sea Load Using Modified Components from the Modelica MultiBody Library

Savin Viswanathan[1] Christian Holden[1]

[1]Dept. of Mechanical and Industrial Engineering, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. {savin.viswanathan,christian.holden}@ntnu.no

Abstract

In this paper, the philosophy of the lumped-mass approach is adopted in specifying *components* so as to enable the *Modelica* compiler to formulate equations governing the motion of a chain-suspended sub-sea load, subjected to waves and current. The *discretized* simulation model of the chain-suspended load is built up using the components available in the *MultiBody* library of *OpenModelica*, after making necessary modifications. The combined wave and current loads acting on the segments are determined using the Morison equation, and applied as discrete external forces on the lumped segmental masses. The component model is developed and implemented using the *OMEdit* GUI, and the simulation results are then compared with those for a similar system modelled in the popular commercial ocean-engineering time-domain simulation software, *Orcaflex*, to demonstrate satisfactory agreement. Conclusions are drawn, and the simulation files are made available for public access.

Keywords: Modelica component-model for submerged cables, dynamics of sub-sea loads, OceanEngineering library

1 Introduction

The authors discussed the benefits of developing a *Modelica* standard library for ocean-engineering applications in (Viswanathan and Holden, 2019). In the above work, the *quasi-static* approach was adopted to specify the mooring forces at any given simulation time-step. However, it was noted that this led to the omission of the inertial and deflection effects of the mooring line, as discussed in detail by the authors in (Viswanathan and Holden, 2020a). Hence, steps in the direction of developing component-models capable of simulating the dynamic behaviour of mooring chains, as accurately as possible, were adopted by the authors. The present work, which is an offshoot of such efforts, brings to the *proposed* library, basic components to simulate the dynamics of a fully submerged, suspended sub-sea load.

The earliest reference to the application of the *lumped-mass* approach to sub-sea cables is traced to Walton *et. al* (Walton and Polachek, 1960), who prescribes the lumping of masses of straight segment lengths, and associated external forces, at nodal points which connect the adjoining segments, and thus arrive at equations of motions for the *discretized* mathematical model of the physically *continuous* cable. They further suggest a fixed time-step numerical scheme to obtain the cable dynamics. Other relevant works include (Nakajima et al., 1982), and (Thomas and Hearn, 1994).

We, however, notice that such *time-step* dependent methods are inherently opposed to a fundamental philosophy behind *Modelica*, which is expressed by Dr. Michael Tiller in his words, (Tiller, 2013):

"The key point is that equations describing physical behavior cannot refer to time steps. This is because there is no timestep *in nature or the laws of physics, and so the response of a system cannot depend on it."*

The statement points to the fact that the *Modelica* user needs to specify only the differential algebraic equations governing the physics of the system, and solution methodology is best left to the *Modelica* compiler.

We, therefore, adopt the philosophy of Walton in modelling the cable/chain segments in *Modelica*, using components already available in the *Modelica.Mechanics.MultiBody* library, albeit with necessary modifications. Connecting these components enable the automatic generation of the coupled equations of motion by the *Modelica* compiler, which is then solved for obtaining the system dynamics.

The Morison equation is widely used in the ocean engineering domain to calculate fluid loads on slender structures. Numerous publications deal with the subject, and is described in detail, for e.g., in (Chakrabarti, 1987). In this work, the *Morison* equation is implemented as a *Block*, and the determined fluid drag and inertia loads are then applied as forces, along with buoyancy, at the lumped-mass points.

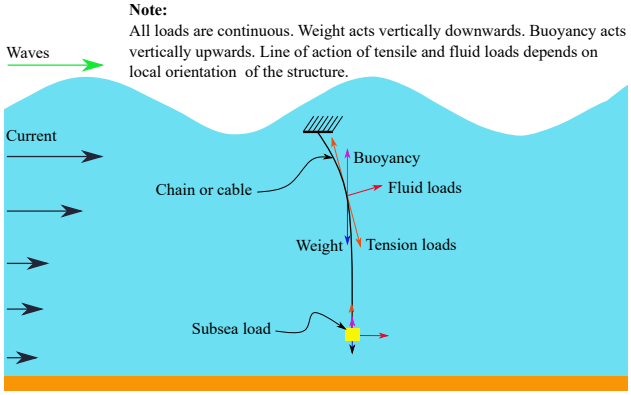
We, therefore, proceed by presenting a brief theoretical introduction to the *discretization* of the *continuous* cable/chain, along with the calculation of Morison loads. This is followed by a detailed description of system representation in *Modelica*. Simulation results are benchmarked using *Orcaflex*, and conclusions drawn. Both *Modelica* and *Orcaflex* simulation files are made available for public access at github.com/Savin-Viswanathan/Modelica2020Asia.

2 Theory

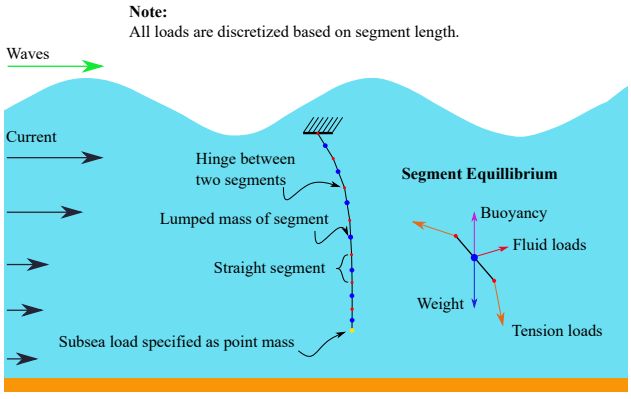
Figure 1a shows the forces acting on a chain suspended sub-sea load, and Figure 1b shows the discretized mathematical model for the same.

For simplicity, we consider:

- 2D motions in x and y directions only.
- Top end of the chain is fixed.
- Inelastic chain.
- Fully submerged chain and load at all times.
- End load has negligible drag area, and can be approximated as a point mass.



(a) Continuous physical model.



(b) Discretized mathematical model.

Figure 1. Discretization of the chain suspended sub-sea load.

The coupled equations of motion of the chain/cable segments based on the segment equillibrium may then be solved to determine the dynamic behaviour of the system.

Proper translation of the *discretized* model into a *Modelica* system-model effects the automatic generation of the coupled equations of motion governing the dynamic behaviour of the system. Details of modelling are described in detail in the next section.

Considering the j^{th} segment,

$$W_j = l_j \mu g \tag{1}$$

$$B_j = \frac{\pi D_b^2}{4} l_j \rho_w g. \tag{2}$$

Here, W_j [N] is the weight of the segment, B_j [N] is the buoyancy force experienced by the segment, l_j [m] is the length of the segment, μ [kg/m] is the specific linear mass of the chain/cable, D_b [m] is the diameter based on which buoyancy is calculated, ρ_w [kg/m^3] is the density of seawater, and g [m/s^2] is the acceleration due to gravity.

In evaluating the fluid loads, we make use of the Morison equation for combined wave and current loads on an inclined oscillating cylinder. See p. 188 of (Chakrabarti, 1987).

Experimental values for drag coefficient C_D and inertial coefficient C_M are scarce when structures are inclined. Hence, in determining these loads, we evaluate the fluid loads along the normal and tangential directions of the chain segment and then sum up their horizontal and vertical components. The advantage of this approach is that it enables the specification of separate drag (C_D) and inertia (C_M) coefficients for the normal and tangential directions. See p. 205 of (Chakrabarti, 1987). The normal and tangential components of the Morison force per unit length of the segment are thus given as

$$\begin{aligned} M_F^n =& C_M^n \rho \frac{\pi}{4} D^2 a_w^n - C_A^n \rho \frac{\pi}{4} D^2 a_l^n \\ &+ C_D^n \frac{1}{2} \rho D \mid v_w^n \pm U^n - v_l^n \mid (v_w^n \pm U^n - v_l^n). \end{aligned} \tag{3}$$

$$\begin{aligned} M_F^t =& C_M^t \rho \frac{\pi}{4} D^2 a_w^t - C_A^t \rho \frac{\pi}{4} D^2 a_l^t \\ &+ C_D^t \frac{1}{2} \rho D \mid v_w^t \pm U^t - v_l^t \mid (v_w^t \pm U^t - v_l^t). \end{aligned} \tag{4}$$

Here, superscripts n and t denote the normal and tangential directions, and subscripts w and l denote the water-particle and the mooring-segment respectively. Further, a [m/s^2] refers to acceleration, v refers to velocity, U [m/s] is the magnitude of the current velocity, and D [m] is the line drag diameter.

The current velocity, and wave induced water-particle velocities and accelerations, at the segment lumped-mass points, are to be considered in (3) and (4).

For a linear wave, the following are defined:

$$\omega^2 = gk\tanh(kd) \tag{5}$$

$$\eta = H/2\cos(kx - \omega t), \tag{6}$$

$$u = \frac{\pi H}{T} \frac{\cosh k(z+d)}{\sinh(kd)} \cos(kx - \omega t) \tag{7}$$

$$w = \frac{\pi H}{T} \frac{\sinh k(z+d)}{\sinh(kd)} \sin(kx - \omega t) \tag{8}$$

$$\dot{u} = \frac{2\pi^2 H}{T^2} \frac{\cosh k(z+d)}{\sinh(kd)} \sin(kx - \omega t) \tag{9}$$

$$\dot{w} = -\frac{2\pi^2 H}{T^2} \frac{\sinh k(z+d)}{\sinh(kd)} \cos(kx - \omega t). \tag{10}$$

DOI 10.3384/ecp202017459

Here, ω [rad/s] is the wave frequency, η [m] is the sea surface elevation, u and w [m/s] are the horizontal and vertical components of the wave-induced water particle velocities, the overdot denotes time derivative, H [m] is the wave height, T [s] is the wave period, k [m^{-1}] is the wave number, x and z [m] are the horizontal and vertical co-ordinates of the evaluation point, d [m] is the water depth, and t [s] is the simulation time. See pp. 51–52 of (Chakrabarti, 1987).

Figure 2 gives the expression for the normal and tangential components of the wave-induced water particle velocities associated with a segment inclined at angle θ to the horizontal. Similar expressions may be obtained for the relevant current, and segment kinematics.

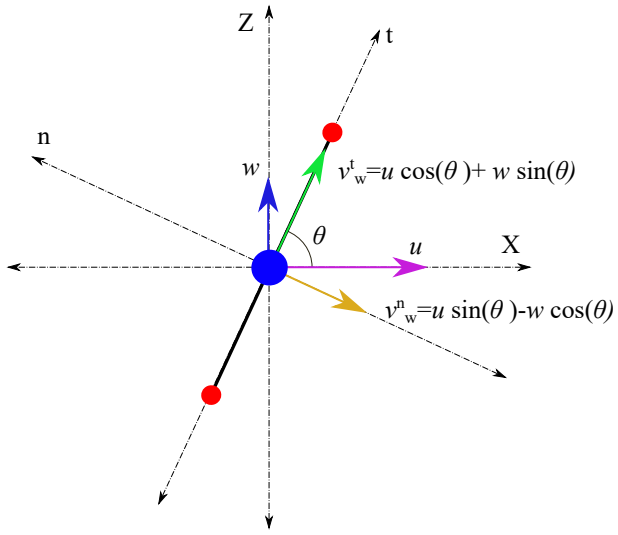


Figure 2. Normal and tangential components.

The horizontal and vertical components of the Morison loads on the segment may thus be determined as:

$$M_F^x = M_F^t \cos\theta + M_F^n \sin\theta \quad (11)$$

$$M_F^y = M_F^t \sin\theta - M_F^n \cos\theta. \quad (12)$$

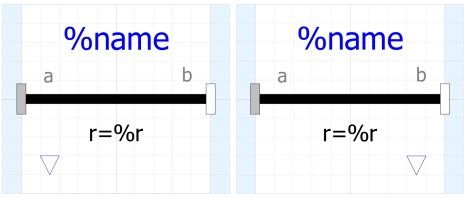
The problem is implicit in the sense of the interdependency between line orientation, tension and fluid loading.

3 Building the Modelica Model

Representation of the *discretized* model in *Modelica* is realized through the use of components already available in the Multi-Body-System (MBS) library of *Modelica*, with some modifications to meet the problem requirements.

The segmental lumped mass, and the suspended load, are represented by **PointMass** components, the massless lengths of segments lying on either side of its lumped-mass are represented by **FixedTranslation** components, the point of suspension of the top end is specified by a **Fixed** component, and the hinge connection between the segments are represented by **Revolute** joint components, all of which are available in the MBS library.

In the determination of fluid loads, we require the orientation of the segment at any given simulation time step, and hence a modification is effected to the **FixedTranslation** component by specifying a *RealOutput* interface to transmit the coordinates of the flanges. Two variants of the **FixedTranslation** components are specified, the icon representations of which are shown in Figure 3.



(a) UP_Seg. **(b)** LO_Seg.

Figure 3. *Icon* views of the modified *FixedTranslation* components.

UP_Seg is modified such that its *RealOutput* interface transmits the coordinates of its *flange_a*.

LO_Seg is modified such that its *RealOutput* interface transmits the coordinates of its *flange_b*.

The *diagram view* of the simplest sample system showing all used components is shown in Figure 4.

The segment model is built up by connecting the appropriate flanges of upper segment **UP_Seg**, a **PointMass**, and a lower segment **LO_Seg**. The interconnection between two segments, and of a segment with the point of suspension, can be effected through a **Revolute** joint. The point of suspension of the top end is specified by a **Fixed** component, and a **PointMass** component is used to specify the suspended load. Drag calculations are carried out by **DnB** blocks.The computed drag and buoyancy values are transformed to world forces by a **WorldForce** component, and applied as loads to the flanges of the lumped-masses. Gravity is included by the specification of the **World** component.

The environment, and cable/chain parameters, are specified inside the **DnB** block. The parameters specified are:

General: water depth d, water density ρ, ramping period for waves and current T_{rmp}.

Regular Wave: wave height H, period T.

Current: *vector* of depths at which the profile is defined z_{cg} and fully developed magnitudes of current at these depths U_f.

Cable: drag diameter D, buoyancy diameter D_b, normal and tangential added mass coefficients C_A^n and C_A^t. normal and tangential drag coefficients C_D^n and C_D^t.

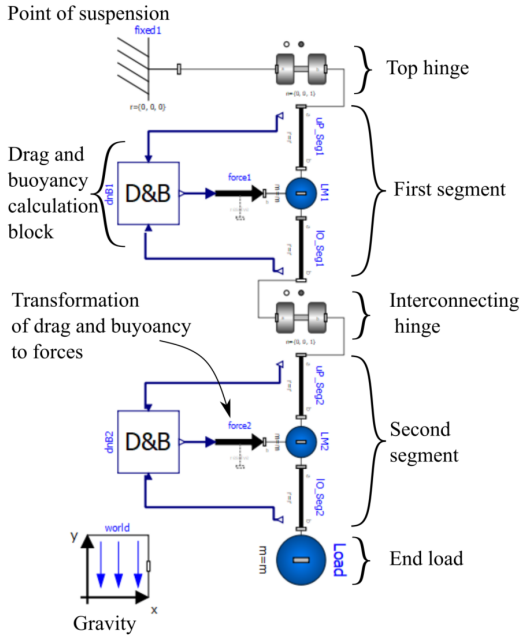


Figure 4. Modelica representation of a suspended subsea load system.

The wave number is computed by *function* **waveNumberIterator**, by iteration of the dispersion relation (5), as described in (Viswanathan and Holden, 2020b). The segment lengths, and instantaneous location of *lumped-mass* points are calculated based on the real outputs of the **UP_Seg**, and **LO_Seg**, associated with each segment.

The sea surface elevation (SSE) at the x co-ordinate of the *lumped-mass* point is calculated using (6), and the wave and current kinematic profiles are moved with the SSE as described in (SINTEF, 2014). The current velocity at the y coordinate of the *lumped-mass* point is then interpolated for using the **linearInterpolatorSV** *function*, and the wave-induced water-particle velocities and accelerations are calculated using (7)–(10).

The velocities and accelerations of the *lumped-mass* points at the current time step being provided by *Modelica*, the instantaneous drag may be determined using equations (3), (4), (11), and (12).

Drag and buoyancy forces on the end load may also be specified by using a similar **DnB** block, but has been omitted here for simplicity.

4 Results

We discuss the simulation results of a system with parameters shown in Table 1:

Figure 5 shows the diagram view of the above system in *Modelica*.

Figure 6a compares the line configurations in *Modelica* and *Orcaflex*, at $t = 100$ [s], when subject to a uniform current profile defined by $z_{cg} = \{-50,-25,-10,0\}, U_f = \{1,1,1,1\}$. Figure 6b compares the same for a current profile defined by $z_{cg} = \{-50,-25,-10,0\}, U_f = \{0,0.5,1,2\}$. In both cases, the wave height $H = 0$ [m].

Parameter	Value
Depth of suspension point below water surface	2.5 [m]
Chain length	30 [m]
Chain specific mass	10 [kg/m]
Discretization segment length	5 [m]
Chain buoyancy diameter	0.04 [m]
Chain drag diameter	0.04 [m]
Chain drag coeff. (normal)	1 [-]
Chain drag coeff. (tangential)	0.25 [-]
Chain added mass coeff. (normal)	1 [-]
Chain added mass coeff. (tangential)	0.5[-]
End load mass	100 [kg]
Ramp time for waves and current	10 [s]
Water depth	50 [m]
Water density	1025 [kg/m^3]
Current profile	*variable*
Regular wave parameters	*variable*

Table 1. System parameters

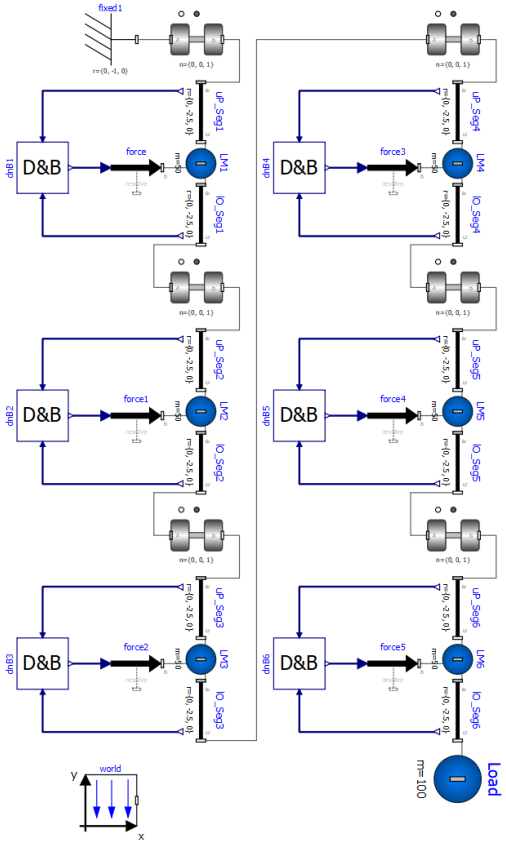
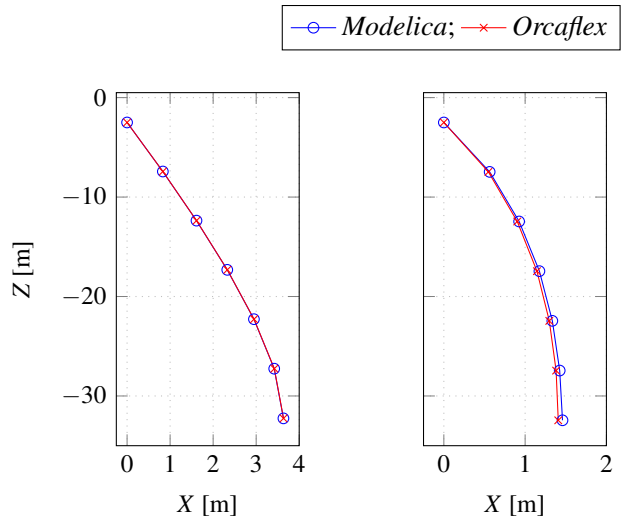


Figure 5. Diagram view of a subsea suspended load *system*.

Figure 7 compares the top end tensions for both the above cases.

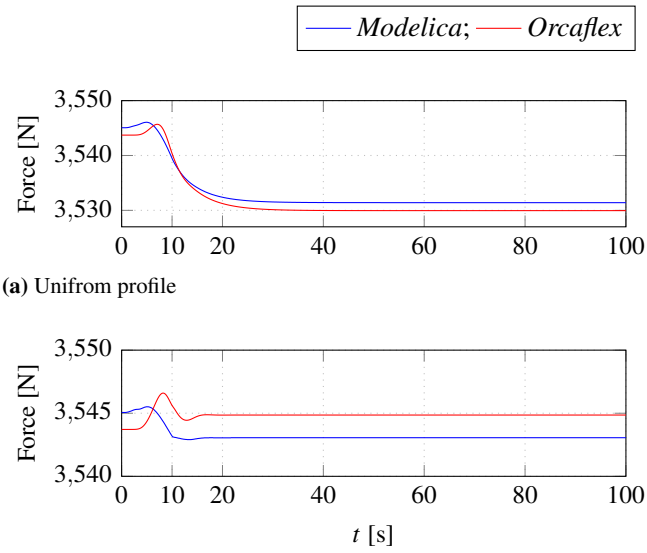
Figures 8a and 8b compare the horizontal and vertical response of the suspended load to regular waves of $H = 5$ [m] and $T = 10$ [s], in both *Modelica* and *Orcaflex*, while Figure 8c compares the top end tensions. Current loading is set to zero by specifying $z_{cg} = \{-50,-25,-10,0\}, U_f = \{0,0,0,0\}$.

DOI 10.3384/ecp202017459



(a) Unifrom profile **(b)** Non-uniform profile

Figure 6. Line configuration for different current profiles.



(a) Unifrom profile

(b) Non-uniform profile

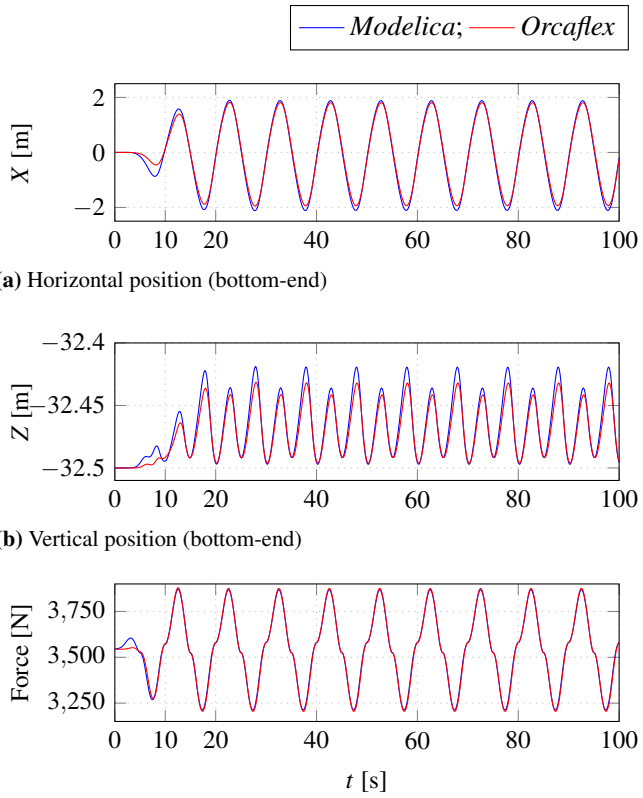
Figure 7. Line top end tensions for different current profiles.

Figure 9a and Figure 9b compares the horizontal and vertical response of the suspended load to regular waves of $H = 5$ [m] and $T = 10$ [s] in the presence of a current with profile defined by $z_{cg} = \{-50, -25, -10, 0\}, U_f = \{0, 0.5, 1, 2\}$, in both *Modelica* and *Orcaflex*, while Figure 9c compares the top end tensions.

5 Result Discussion

From the above figures, we observe a general agreement between *Modelica* and *Orcaflex* responses. To quantify the degree of agreement, we present the percentage variation between them in Table. 2.

In most cases, we observe good agreement with <5% variation. On examining the values with higher % variation, we infer that the numerical significance is quite low,

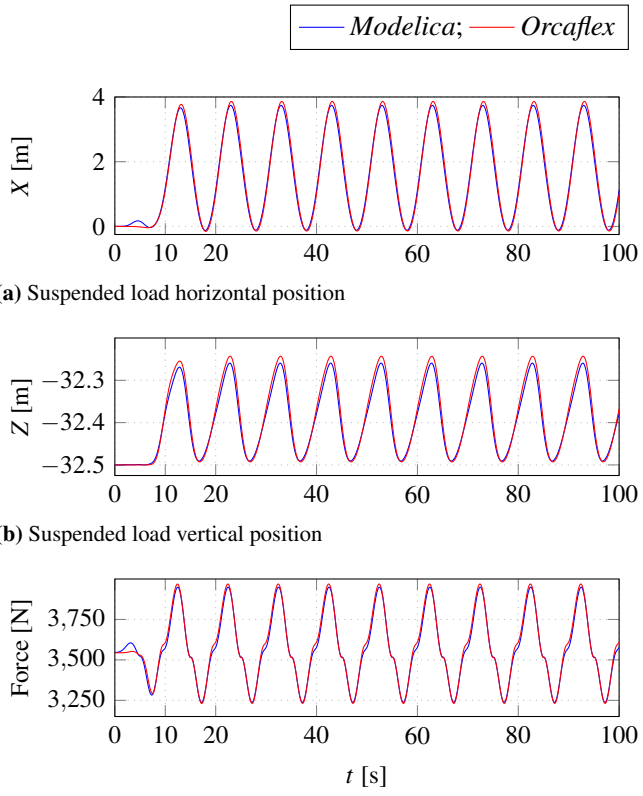


(a) Horizontal position (bottom-end)

(b) Vertical position (bottom-end)

(c) Top end tension

Figure 8. Regular wave response.



(a) Suspended load horizontal position

(b) Suspended load vertical position

(c) Top end tension

Figure 9. Combined wave–current response.

as demonstrated below for the highest variation of 21.9%, corresponding to the vertical displacement of the suspended load, as depicted in Fig. 8b.

Numerically, the *Modelica* and *Orcaflex* responses are $-32.4193 - (-32.4970) = 0.0777$ [m], and $-32.4324 - (-32.4960) = 0.0636$ [m], indicating a difference of 0.014 [m], which is quite insignificant when we consider that this variation of 1.4 cm is for the motion of the tip of a chain that is 30 [m] long. Similar inferences can be arrived at for all other values.

These variations could be due to the fact that we use *moved* kinematic profiles, while *Orcaflex* uses *Wheeler stretching* of wave and current kinematics.

Larger variations observed during the ramp-up time $T_{rmp} = 10$ [s], in all cases, is attributed to the fact that we use a sinusoidal ramping function while *Orcaflex* uses an in-built ramping function with a different ramp curve.

The reason for the variation in initial top-end tension and tension response to currents as observed in Figure 7, though insignificant, has not yet been understood.

Variable Description	%variation
Horizontal position of end load in uniform current (Fig. 6a)	-0.07
Horizontal position of end load in profile current (Fig. 6b)	3.72
Vertical position of end load in uniform current (Fig. 6a)	0.00
Vertical position of end load in profile current (Fig. 6b)	-0.03
Top end tension in uniform current (Fig. 7a)	0.04
Top end tension in profile current (Fig. 7b)	-0.05
Horizontal response in waves (Fig. 8a)	6.47
Vertical response in waves (Fig. 8b)	21.9
Top end tension response in waves (Fig. 8c)	-2.34
Horizontal response in waves and current (Fig. 9a)	-3.62
Vertical response in waves and current (Fig. 9b)	-7.31
Top end tension response in waves and current (Fig. 9c)	-11.11

Table 2. Variation between *Modelica* and *Orcaflex* results.

6 Conclusion

The work presented in this paper introduces a novel method for specifying fluid loads on a mass discretized subsea cable using components already available in the *Modelica MultiBody* library, with minor modifications.

Based on the agreement between *Modelica* and *Orcaflex* simulation results, it is concluded that the model exhibits satisfactory representation of structural and fluid inertia effects, and accurate modelling of the drag loads on a cable structure.

The only traceable reference to an attempt to use *Modelica* in a similar scenario, by other researchers, is in the modelling of the station keeping system of an offshore wind turbine in (Leimeister and Thomas, 2017), where limitations included the inability to account for:

- Relative accelerations in wave load calculation.
- Current loads on submerged structures.

These limitations have been successfully mitigated in the present model.

It may also be noted that the authors are relatively new to *Modelica*, and the results presented here are for a work in progress. The code presented along with this work may show instances of under-utilization of advantages offered by *Modelica*, for e.g., the use of the *array* concept in implementing the lumped mass philosophy. The main focus of the present stage of the authors' research is to build a general framework for simulation of ocean engineering systems in *Modelica*. Code refinement is planned for the next stage of the project.

Extension of the modelling philosophy presented in this work is expected to open the window towards the development of *Modelica* component models for catenary as well as taut moorings. Inclusion of linear and torsional spring elements is expected to enable *Modelica* representation of flexible structures with elasticity and bending stiffness viz. risers, elastic moorings, and umbilicals, in the future.

The further development of this work, coupled with the development of component models for waves and currents as described in (Viswanathan and Holden, 2020b), and for non-diffracting floating objects as described in (Viswanathan and Holden, 2020a), followed by the development of component models for diffracting objects in the future, would thus enable the integrated simulation of multiphysical ocean-engineering systems, in their entirety, using *Modelica*.

Presently, the authors are developing an *open-source* code for determining the hydrodynamic coefficients which appear in the equation-of-motion of diffracting floating-objects. The initial results look promising, and the subject will be dealt with in a future publication.

7 Acknowledgements

The research in this paper has received funding from the Research Council of Norway, SFI Offshore Mechatronics, project number 90034210.

References

Subratha Kumar Chakrabarti. *Hydrodynamics of Offshore Structures*. Computational Mechanics Publications, and Springer-Verlag, Dorchester, Great Britain, 1987. ISBN 0-905451-66-X.

Mareike Leimeister and Philipp Thomas. The onewind modelica library for floating offshore wind turbine simulations with flexible structures. In *Proceeding of the 12th International Modelica Conference*, pages 633–642, 05 2017. doi:10.3384/ecp17132633.

Toshio Nakajima, Seizo Motora, and Masataka Fujino. On the dynamic analysis of multi-component mooring lines. In *Proceedings of the Offshore Technology Conference*, 1982. doi:10.4043/4309-ms.

SINTEF. *Handbook on Design and Operation of Flexible Pipes*. 2014. URL sintef.no/en/latest-news/updated-handbook-on-design-and-operation-of-flexible-pipes/.

D.O Thomas and G.E. Hearn. Deepwater mooring line dynamics with emphasis on seabed interference effects. In *Proceedings of the Offshore Technology Conference*, volume OTC-7488-MS, May 1994. doi:10.4043/7488-MS.

Michael Tiller. Response to questions on determining variable values of previous time steps in modelica. In *Stackoverflow*, 2013. URL https://stackoverflow.com/questions/16558587/how-to-determine-value-from-previous-time-step-during-simulation-in-modelica.

Savin Viswanathan and Christian Holden. Towards the development of an ocean engineering library for openmodelica. In *Proceedings of the ASME 2019 38th International Conference on Ocean, Offshore and Arctic Engineering*, volume 7B:Ocean Engineering. OMAE, November 2019. doi:10.1115/OMAE2019-95054.

Savin Viswanathan and Christian Holden. Modelica component-models for non-diffracting floating objects and quasi-static catenary moorings. In *Proceedings of the American Modelica Conference*, 2020a. The conference has been postponed due to the prevailing COVID-19 situation. A link to the referred publication will be provided along with the download files for the referring publication, as and when such a link becomes available.

Savin Viswanathan and Christian Holden. Modelica component-models for oceanic surface-waves and depth varying current. In *Proceedings of the American Modelica Conference*, 2020b. The conference has been postponed due to the prevailing COVID-19 situation. A link to the referred publication will be provided along with the download files for the referring publication, as and when such a link becomes available.

Thomas S. Walton and Harry Polachek. Calculation of transient motion of submerged cables. *Mathematics of Computation*, 14(69):27–46, 1960. doi:10.1090/S0025-5718-1960-0116470-5.

DOI
10.3384/ecp2020174

Collaborative Development and Simulation of an Aircraft Hydraulic Actuator Model

Clément Coïc Johan Andreasson Anand Pitchaikani Johan Akesson Hemanth Sattenapalli

Modelon, {name.surname}@modelon.com

Abstract

This paper discusses the typical engineering workflow of an aircraft hydraulic actuator development – enhanced using Modelon Hydraulics library and the new Modelon Impact environment. Modelon Impact – consistent with Modelon's philosophy – enables having a more collaborative development and maintains more continuity between different phases of design cycle as well as democratizes models by making it available in different forms including dedicated apps.

In order to cover the scope of the engineering design workflow, several steps are discussed:

- Specification of the system requirements through test scenarios
- Sizing of the system based on a steady-state design point
- Re-use of the sized parameters for simulation of the model
- Performing design verification, based on dynamic simulation scenarios
- Export of the model using the Functional Mock-Up Interface standard, e.g, for co-simulation with flight control laws

As most realistic engineering workflows involve several teams and, therefore, several tools, this paper discusses the openness and connectivity of the proposed Modelon solution.

Keywords: Hydraulic actuation system, Model-based Design, Collaborative Design

1 Introduction

In a Model-Based System Engineering (MBSE) workflow, virtual prototypes (models) are used to facilitate the design and testing of the system under study. The terminology Model-Based Design (MBD) is also often used to emphasize that system sizing is performed with the help of models. While the complete MBSE design process would cover many types of different model, this paper focuses on the behavioral modeling of the system that is covered by the Modelica Language.

The Requirement, Functional, Logical, and Physical (RFLP) formalism would include additionally, *requirement* management, UML/SYSML models for the *functional* definition of the system or 3D/CFD/FEA models for the detailed *physical* design.

Figure 1 gives an overview of how Modelica models fits in the over-all design flow, using example applications from the Vehicle Dynamics Library: Starting from the left, lower fidelity models are used to explore large sets of potential configurations before gradually narrowing down the design. As the design work continues, the fidelity is increased. Beyond natively providing a large fidelity range, the models can also be evaluated against formal requirements (references to requirements library and doors?), connect to other models (FMI, external object), and/or reduced order models from tools for detailed physical design.

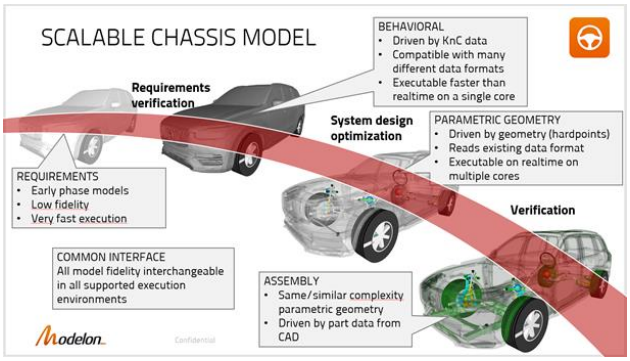


Figure 1. The Modelica Language allow for the design of scalable models, where fidelity can be adapted to the different stages of Modeli-Based Systems Engineering (MBSE).

The behavioral model-based design of a system can assist several tasks in the entire MBSE workflow. These are typically: system architecture definition, system pre-sizing, component pre-sizing, detailed behavioral modeling and simulation, unitary and integration verification, real-time simulation for Hardware-in-the-Loop (HiL) simulation [DRE14], etc.

Unfortunately, most of current engineering workflows are decoupled between these tasks. The main reason is often that each task is performed by a different person, often in a different team or company and the tools used are different.

Modelon's approach is to deploy Modelica, FMI and other open standards to promote collaboration. This is done in three ways:

1) Modelon Library Suite[1] is provided for capable Modelica tools. The suite consists of libraries covering several industries and applications. This enables models to be built, shared, and executed in collaborative efforts involving multiple tools.
2) The Optimica Compiler Toolkit[2] is a Modelica engine for vendors that desire to include Modelica support in their offerings, and it is embedded in ANSYS Twin Builder®[3] and Siemens Simcenter Amesim®[4]. On top of that they can add the Modelon Library Suite and/or libraries from other providers. In addition, Modelon's FMI tools[5] provide key elements for linking together FMI-based toolchains.
3) Modelon Impact is our new end-user product that encapsulates our experience into one offering for collaborative product design. It combines the functionalities of our portfolio, including libraries and Modelica engine, with a web-based architecture and a novel user interface.

This paper discusses how Modelon provides streamlined MBSE workflows, connecting the broken streams. We will use, as an example, the development of an aircraft hydraulic actuator. The actuator is developed based on the Modelon Hydraulics Library that is available on multiple platforms.

The role of the aircraft actuator in his context is presented in Section 2. Section 3 describes the different phases of the model-based design of the actuator. Section 4 addresses each of these phases and emphasizes the collaborative aspects of our solution. A summary and conclusions are provided in Section 5.

2 Aircraft hydraulic actuator

2.1 An aircraft hydraulic actuation system

The actuation system of an aircraft performs a safety critical function that ensures controllability. Although today's trend is focused on developing electrical actuation systems, current aircrafts and new developments still require hydraulic fluid to power (most of) the actuators.

In this paper, the aircraft actuations system refers to the set of actuators, operated in closed-loop, that drives the control surfaces and the associated hydraulic systems. Assuming the use of hydraulic power, the main functions covered by the aircraft actuation systems are to [COI16]:

- Store hydraulic fluid, typically implemented by reservoirs, pressurized or non-pressurized.
- Generate hydraulic power – typically ensured by mechanical pumps, electric pumps, hand pumps, ram air turbine (RAT) and power transfer units (PTU).
- Store hydraulic power, mostly implemented by accumulators, and also by the capacitance effect of the lines.
- Distribute hydraulic power, mostly ensured by rigid lines and hoses as well as dedicated valves to distribute the fluid.
- Convert hydraulic power to mechanical power, typically performed by linear or rotary actuators. This function can be split into two main sub-functions:
  - Meter hydraulic power, typically ensured by proportional control valves.
  - Transform hydraulic power into mechanical power, usually performed by hydraulic cylinders (linear or rotary).

The example used throughout this paper is an aircraft hydraulic actuator that fulfills the functions "convert hydraulic power". In order to add realism to this paper, an A320-like aileron actuator is used as example.

2.2 The role of the aileron actuator

The roll control of an A320 aircraft is mainly driven by the aileron pitch angles, assisted by the external spoilers at low speed. Each aileron is actuated by two actuators. Each of these actuators is qualified as *simplex*, as each has only one hydraulic cylinder. The loss of the hydraulic power connected to one actuator would result in the loss of the actuator itself. Therefore, for a given aileron, both actuators are each fed by different hydraulic circuits to ensure redundancy.

The command and monitoring functions of the aileron actuators are realized by the Elevator and Aileron Computer (ELAC). In order to avoid force-fighting between the two actuators of an aileron and to reduce the overall wear and fatigue on the actuators, the ELAC commands one actuator to be "active" while the other is set to "stand-by".

Such an "active/stand-by" configuration is common on aircrafts. The active actuator drives the control surface while the stand-by one is driven by the surface motion – and generates some damping. In the event of the active actuator failure or loss, the ELAC would invert the roles so the surface could still be actuated by the functioning actuator

[1] https://www.modelon.com/products-services/modelon-library-suite-modelica-libraries/
[2] https://www.modelon.com/products-services/modelon-creator-suite/optimica-compiler-toolkit/
[3] https://www.ansys.com/en/products/systems/ansys-twin-builder
[4] https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-amesim.html
[5] https://www.modelon.com/products-services/modelon-deployment-suite/

DOI 10.3384/ecp202017467

2.3 Architecture description of the A320 aileron actuator

Figure 2 [MAR18], depicts the architecture of an A320 aileron actuator. The numbering included in the description of the modes, below, are referring to the elements of this figure.

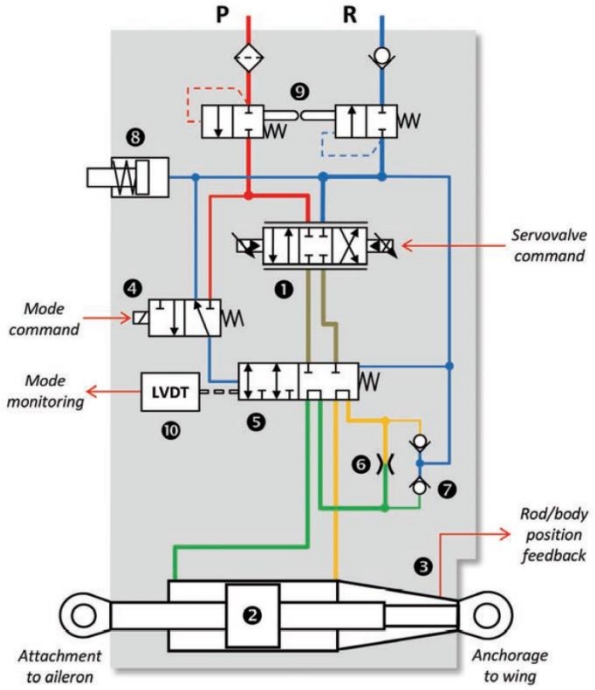


Figure 2. The A320 Aileron Architecture.

In active mode, the aileron actuators' main function is to *convert hydraulic power* into mechanical power. Such a function can be split into two sub-functions: *meter* and *transform hydraulic power*. The components dedicated to performing these two sub-functions are respectively the electro-hydraulic servo-valve (1) and the symmetrical hydraulic cylinder (2); the servo-valve meters the hydraulic power to the cylinder chambers, thus enabling the rod extension or retraction – which is measured by a LVDT sensor (3).

In stand-by mode, both chambers of the cylinder are connected through a hydraulic restriction (6), ensuring the damping of the actuator. In order to avoid the creation of gas in the chambers, anti-cavitation valves (7) are placed so that the return line can provide pressure to the chambers when their pressure becomes too low.

Switching between the two modes is made possible by the mode valve (5), whose position is dependent on its pressurization, controlled by a solenoid valve (4) and measured by a LVDT sensor (10).

Additionally, two mechanically coupled isolation valves (9) are included to keep the fluid inside the actuator in case of pressure supply loss. A fluid compensator (8) enables the storage of low-pressure fluid to compensate for the variation of fluid volume (due to thermal or compressibility effects).

3 Model-based development workflow of the actuator

3.1 Isolation of the actuator sizing

When breaking down the actuation system function into subsystems, requirements are propagated and refined in each subsystem. A common approach for aircraft manufacturer to do so is to, first, define an aircraft control strategy that distributes the maneuverability requirements to the different flight control surfaces. This applies also for secondary functions, such as load alleviation. It is thus a good first step to assume the actuation of each control surface as isolated and define the performance of each one of these.

The overall problem is obviously more complex and aircraft manufacturers are driven by a mass-reduction objective which requires global optimization of the actuation system. This is beyond the scope of this paper, and we will therefore focus here on a conservative design of an actuator based on known boundary conditions. This will yield a realistic design, not too far from an optimum. This is also a good way to obtain appropriate start values for solving a global optimization problem of the entire system.

A conservative design would assume 70% of the effective supply pressure available at the actuator hydraulic connectors [SAE12a]. Doing so, the actuator can be sized separately from the rest of the hydraulic system and shall provide back the maximum flow required in order to size the power generation and distribution systems.

3.2 Actuator requirements and sizing

Requirement specifications of aerospace actuators contain hundreds of requirements involving performance, constraints, safety, stress, maintainability etc. In order to keep the presentation concise and simple, only a few performance requirements are addressed herein.

The typical characteristics found in the literature [SAE12b] are the maximum load (stall load) of the actuator and maximum rates of the piston (no load speed), in both extension and retraction. These are usually not requirements issued from realistic scenarios but computed based on the design point as these are easy to test on test benches for performance acceptance. For an A320 aileron actuator, these parameters as well as the supply pressure are summarized in Table 1.

In line with the engineering workflow, we choose here to size the actuator directly based on a design point which consists of moving a given load at a given speed. This point is however not available in the literature and thus arbitrarily selected by the authors in order to meet the literature characteristics. This design point is added into Table 1.

Last but not least, control laws require the actuator to have a low phase at low frequencies for small amplitude

commands. This is covered by the bandwidth requirement of Table 1.

Table 1 Performance requirement of A320 aileron actuator

A320 hydraulic system		
Supply pressure	3 000 psi	207 Bar
A320 aileron actuator		
Stroke	1.7 in	43 mm
No load speed	3.5 in/sec	89 mm/s
Stall load – Extension	10 200 lb	45.4 kN
Stall load - Retraction	10 200 lb	45.4 kN
Design point		72 mm/s 20 kN
Bandwidth		~3 Hz (small amplitudes)

3.3 Design verification and simulator

Once all parameters have been sized, these are set in the model and simulations are run to verify the current design. These simulations, often called *off-design simulations*, receives as boundary conditions real command and loading scenarios and compare the output with the expected actuator behavior.

In a following step, the actuator model is typically interfaced with the entire hydraulic system and other consumers, including other actuators, landing gears, etc., and the overall actuation system response is studied.

Finally, the model complexity is reduced, maintaining a satisfactory accuracy in order to export the model for real-time applications – such as flight simulators, for example, for tuning the with flight control laws.

4 Collaborative development

In this section, the development of the actuator model described in Section 3 is addressed step by step and collaborative aspects of this process are emphasized. A custom web application developed utilizing the openness of the Modelon Impact tool is also presented. This web application is made to cover the entire development workflow detailed in Section 3.

4.1 Actuator model assembly

4.1.1 Active mode

In terms of modeling the actuator in active mode, the main components are the symmetric hydraulic cylinder and the electro-hydraulic servo-valve. Both these components are off-the-shelf components in the Modelon Hydraulics Library, which is the library being used to construct the model. The parameters for the models will come from the sizing performed in Section 4.2.

The hydraulic cylinder model is available as a library component: *DoubleActingDualRod*. This is a library model that has two variable hydraulic chambers along with the required mechanics such as end stops and mass of the modeled piston. The friction between the cylinder and piston is also modeled. The main required parameter values are the dimensions of the cylinder and piston. While the stroke is a known parameter, the piston and rod areas remain to be defined, at this stage.

The electro-hydraulics servo-valve is modeled using a functional representation which models flow areas between different ports of the servo-valve as variable orifices. The parameters to be provided are the areas of the flow ports.

The supply and return pressures are connected to the servo valve as source and sink. A simple controller is built around the actuator. The controller can command a required position of the piston by controlling the spool position of the servo-valve, as shown in Figure 3.

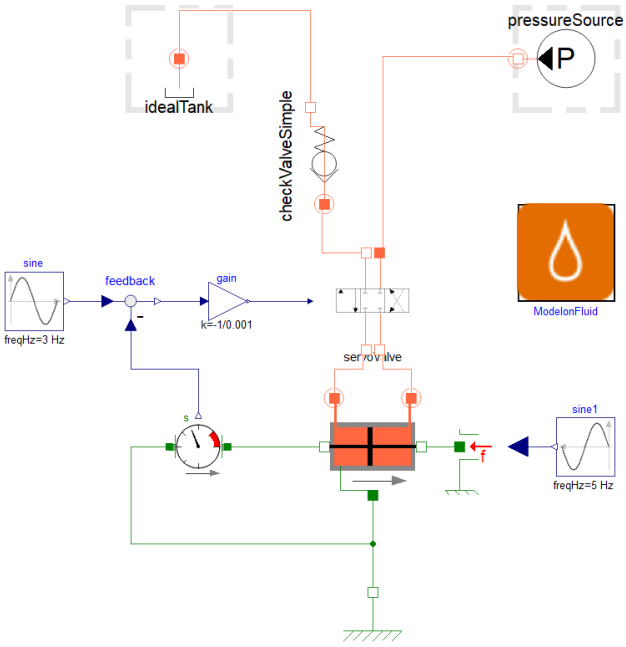


Figure 3. A simple hydraulic actuator model.

4.1.2 Stand-by mode: the mode valve

Until now, the A320 aileron actuator model was assembled with all components necessary for simulating the active mode, i.e., in this mode the actuator follows the commands from the Actuator Control Electronic. The next step for increasing the model realism is to add the stand-by (damping) mode and to switch from one mode to the other. This switch is performed by the mode valve.

For the A320 aileron actuator, the mode valve is a six port, two position valves. For out analysis, the anti-

DOI 10.3384/ecp202017467

cavitation valves are removed and thus the mode valve can be simplified to a four ports only. It is positioned between the servo valve and the actuator. Two of the ports are connected to the control ports of the servo valve while the other two are connected to the cylinder chambers.

In active mode, the mode valve establishes connection between the servo valve control ports and the cylinder ports enabling nominal functioning of the system. In stand-by mode, the mode valve isolates the control ports of the servo valve and connects both cylinder chambers through a damping orifice and anti-cavitation valves – refilling the chambers in case the pressure goes below return pressure.

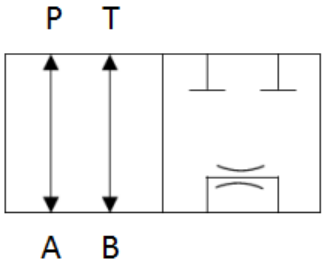


Figure 4. A320 aileron actuator-like mode valve, in a simplified ISO1219 view.

Figure 4 shows an ISO1219 view of this valve, omitting the anti-cavitation valves. Figure 5 shows its location in the actuator model.

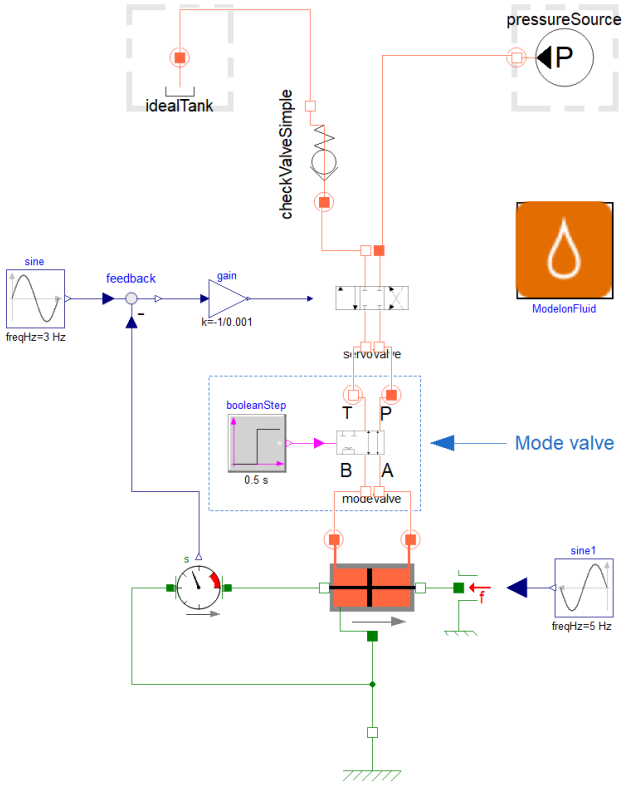


Figure 5. Position of the mode valve in a A320 aileron actuator-like model.

Since all combinations of the directional control valves (DCV) cannot be provided off-the-shelf, Modelon Hydraulics Library provides a framework in which the user can easily custom-create DCVs of their choice. For the mode valve, a generic DCV with 4 ports and 2 positions can be extended and easily parametrized to fit our needs.

Users need to provide parameters that represents the mapping between the continuous normalized spool position and the normalized flow path opening. In our implementation, the spool position was moved from 0 to 1. The flow path area (normalized) also varies from 0 to 1 where 1 corresponds to the area indicated by nominal flow and pressure drop value provided by the user.

The valve, being generic, assumes all six possible paths – i.e. PA, PB, AT, BT, PT and AB. For the A320 aileron mode valve, the only possible paths are PA, BT and AB. The relationship between the spool position and flow path areas can be provided per Figure 6.

Such mapping between spool position and flow paths can be written in the model through the below parameter assignments. The paths that are not connected do not need to be specified, as their corresponding values default to zero:

- open_A_B = [0, 0, 0.5, 1, 1]
- open_B_T = [1, 1, 0.5, 0, 0]
- open_P_A = [1, 1, 0.5, 0, 0]
- spool_x_axis = [0, 0.45, 0.5, 0.55, 1]

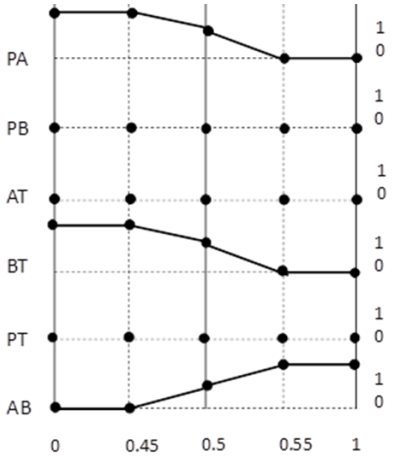


Figure 6. Normalized flow paths v/s spool opening for a mode valve similar to the one of an A320 aileron actuator.

Note that the mode valve is controlled by a unique solenoid valve. This solenoid would provide either the supply or return pressure to the spool surface, moving it to one position or another. In this paper, we decided to only control the valve with a Boolean input to keep it simple.

4.2 Actuator sizing

The sizing of the actuator is achieved using the power of Modelica a-causality: boundary conditions are over constrained during initialization while the unknown parameters are relaxed. This way, the model remains

fully balanced and parameters definition results from the initialization problem solving.

The requirement specification, presented in Table 1, specified steady-state values, exception made of the bandwidth requirement. Consequently, the steady-state initialization, embedded in Modelon Hydraulics library, is selected.

Finally, these performance requirements assume a maximum opening of the valve, which is thus set accordingly in the model.

4.3 Actuator design verification

4.3.1 Speed v/s force characteristic

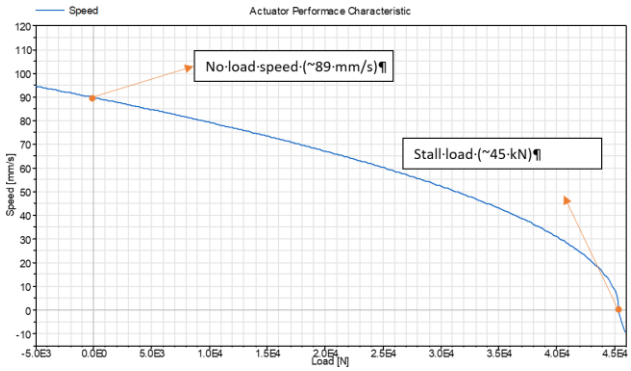


Figure 7. Actuator performance characteristic: speed v/s force.

The design achieved previously can be verified by plotting the speed versus load characteristic of the modeled actuator, we can see that the model meets the actuator requirements of Table 1, in terms of maximum speed and load. A more detailed view is also shown in Figure 11.

4.3.2 Frequency response

The frequency domain verification of the actuator is done linearizing the model shown in Figure 3. The Bode plot shown in Figure 8 highlights a close correlation of the bandwidth mentioned in Table 1.

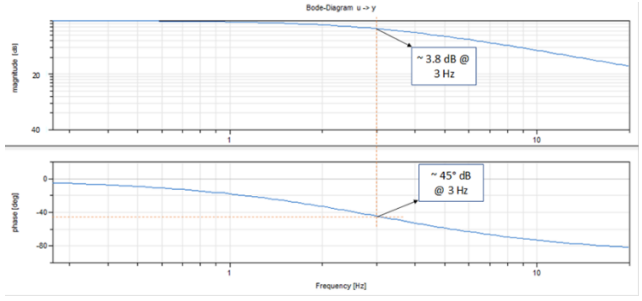


Figure 8. Bode plot showing cut-off frequency.

To achieve these simulations, the parameters were not required to be manually entered by the user as there were computed from on-design simulation and automatically set to the model as off-design parameters – as explained in section 4.2.

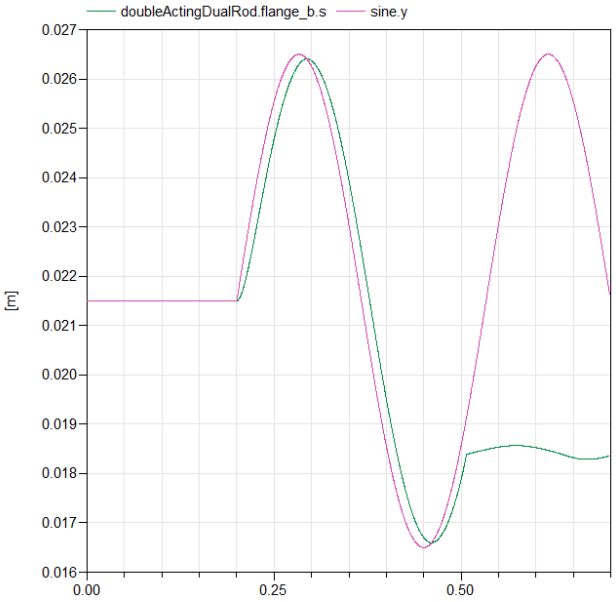


Figure 9. Active/stand-by switch mode at time 0.5.s

4.3.3 Time and fault response

In order to illustrate the mode switching, a dynamic simulation scenario is run where the actuator is initially in active mode and at time 0.5 it switches to the damping mode. The actuator is exposed to a sine antagonist load which is easily faced by the hydraulic actuator in active mode but that drives the actuator in by-pass mode.

4.4 Web Application

Modelon Impact enables users to create and run their own web-applications. Through Python REST API, Modelon Impact allows communication between the web-app and the models. To provide even more flexibility, Python or JavaScript scripting is allowed. This gives the possibility to implement complex workflow into an easy flowing webpage that can be used by many engineers who do not want to get into the details of modeling parts.

A dedicated web-application was developed to summarize the entire workflow presented until now in this paper.

4.4.1 Performance requirements

The performance requirements of the actuator are fed into the app through discrete data points that the actuator must be satisfied in terms of actuator speed and load points. A specific Python script was developed and integrated in the web-application to determine the more constraining points and draw an enclosing curve that will be the envelope for the actuator performance as shown in Figure 10. Points below this enclosing curve would no be relevant to investigate for sizing purpose.

DOI
10.3384/ecp202017467

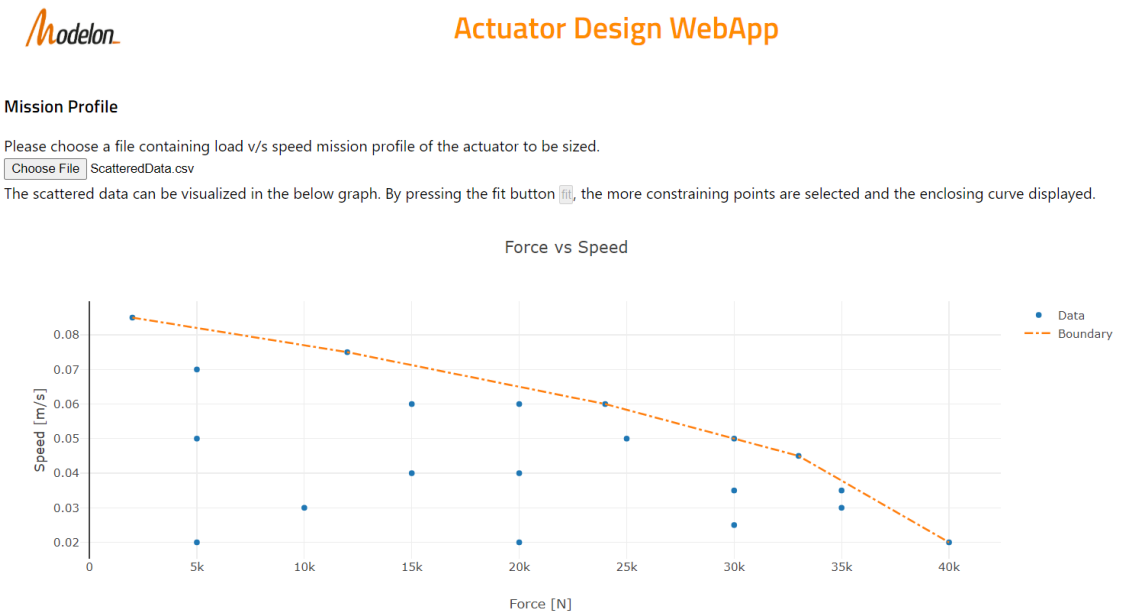


Figure 10. WebApp – view of the performance requirements with the enclosing curve.

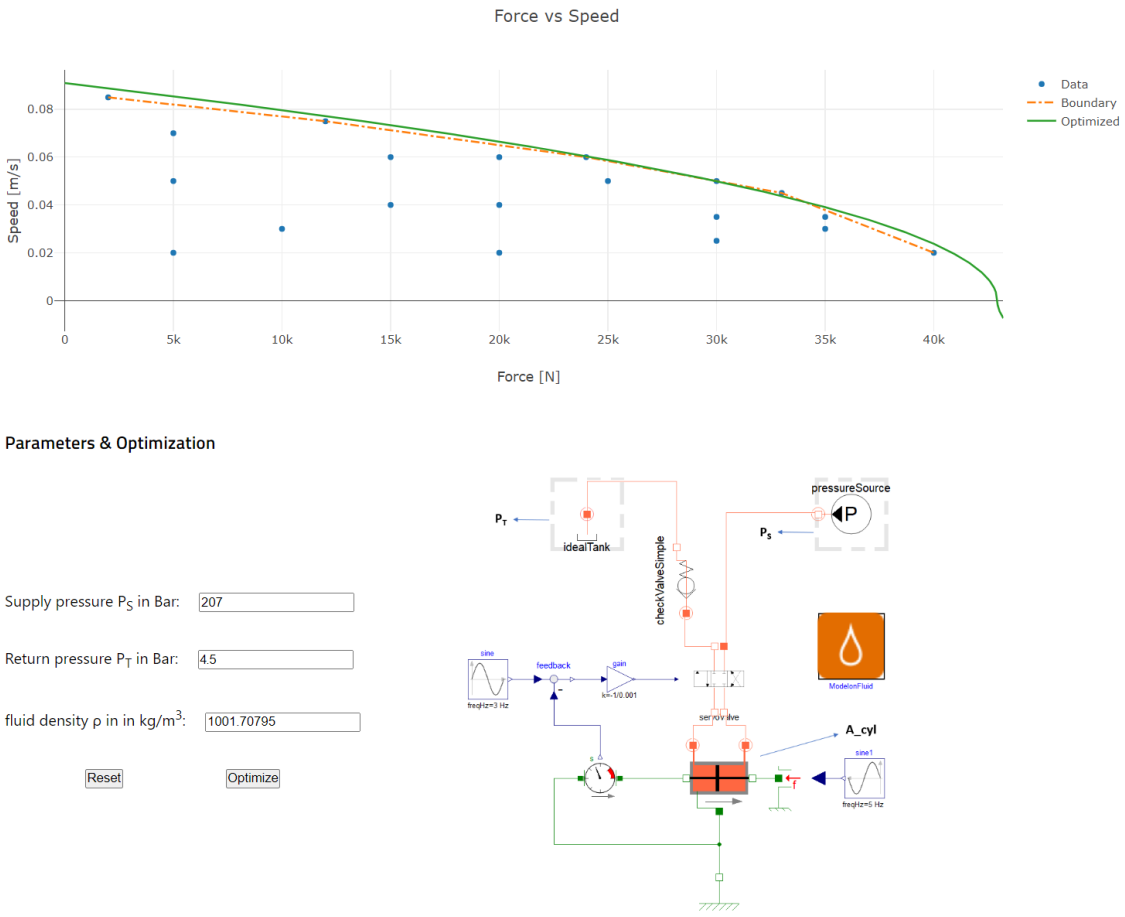


Figure 11. WebApp – optimized design based on parameter inputs for boundary conditions.

4.4.2 Actuator sizing

The next step is to provide the boundary conditions of the actuator system under which the desired performance must be achieved. These are the system boundary pressures and hydraulics fluid density. A simple model is used by the WebApp to run the different designs (actuator cylinder area) through an optimization routine that provides a design which satisfies the enclosing curve of the envelope. This is shown clearly in the Figure 11.

4.4.3 Actuator analysis

The final step of this workflow is to apply the design to the complex dynamic model of the actuator system and perform frequency analysis. The parameters computed previously are applied to the complex model automatically (action performed by the web-application through Modelon Impact API) and state space representation is extracted by the WebApp. This in turn is used to get the bode plots for the actuator control system in totality as shown in Figure 12.

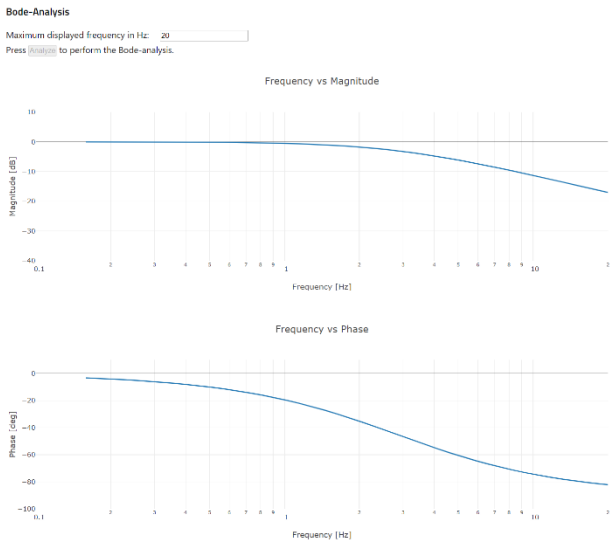


Figure 12. WebApp showing the Bode plot of the actuator control system.

4.4.4 Different user types

Modelon Impact brings a lot of value to a company, even more, in combination with web-applications. While with the graphical user interfaces, the extensive set of libraries, the state of the art compiler and FMI support Modelon Impact opens the door to simulation to many users, the web-applications enable all remaining persons to consume the models, based on a given workflow.

4.5 Real-time simulation

During the development of a Fly-by-Wire actuator and its control electronics, tuning the actuator control laws is of particular importance. While most can be performed by simulation on a standard computer, one relevant step is to test the control laws when executed on a real-time target, which in turn controls the physical plant model, i.e., the hydraulic actuator, inputs. These two models are usually not coming from the same software.

It is critical that the plant model can be simulated in real-time i.e., the computation of the model outputs consumes less time than the corresponding simulated time step. If this is not the case, overruns result which will render incorrect results of the simulation. This type of real-time integration enables validating the actual actuator control laws under more realistic conditions.

In the following, we will illustrate how the actuator model developed in Section 5.1 is integrated in a Speedgoat target and runs real-time.

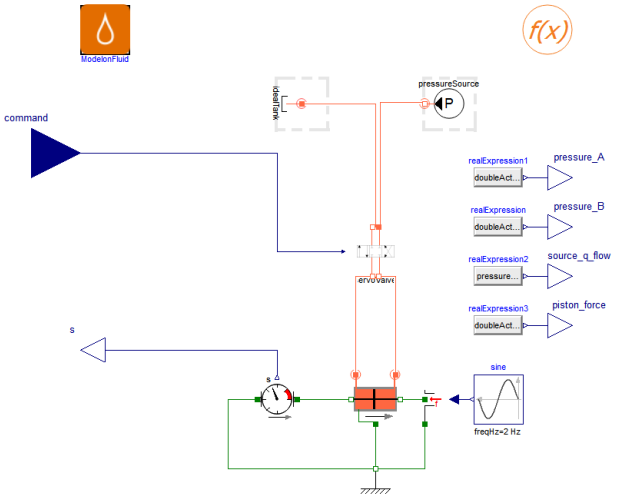


Figure 13. The actuator with interfaces for FMU export.

4.5.1 Model adaptations for HiL simulation

The changes in order to achieve real-time performance are the following:

- The mode valve is removed. The intent of the HiL simulation is to observe the actuator response to commands, meaning the stand-by mode would bring only complexity that is not needed.
- Isolation of the plant model, since the control logic will be in the real-time target implemented in Matlab Simulink®.
- Addition of a outputs monitoring the pressures, flow rate and force of the plant model, to provide easy access and and plotting of these.

Figure 13 displays the final version of the model.

4.5.2 Integration on the Real-Time target

The integration of the model in the Real-Time target consists of a few steps:

- Encoding in Simulink® of the control/command blocks for the actuators.

DOI 10.3384/ecp202017467

- Inclusion of the Modelica model made from Modelon Hydraulics, exported as an FMU, using FMI Toolbox for MATLAB/Simulink[6].
- Adding Speedgoat-specific blocks to monitor the variables of interest.
- Build the real time executable using the Simulink Real-Time™ target.

Figure 14 shows the result of the first three steps listed above, prior to generation of the executable.

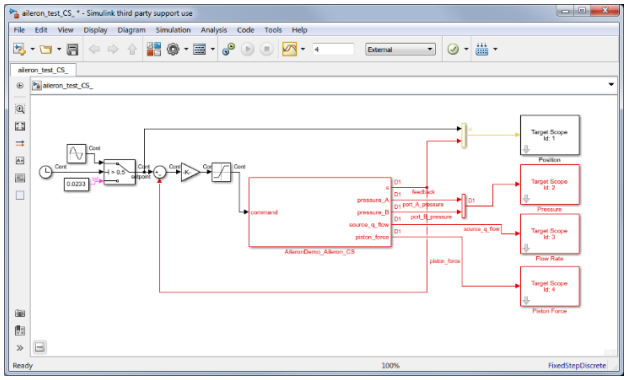


Figure 14. The actuator plant model, implemented in Simulink for import to the Speedgoat real-time target.

In this application, the plant model has been exported with its own solver (FMU for co-simulation) so that both the control logic and the plant model have separate solvers and the models communicate at each time step.

4.5.3 Real-time simulation of the integrated system

Once the model is integrated and imported on the real-time target, it satisfies hard real-time requirements. The main variables set as outputs previously are plotted in Figure 15:

- Top left: actuator command and response. We observe an initial position of the actuator on-purpose different from the command leading to a fast transient at the beginning of the simulation. After reaching the setpoint, the actuator reacts in a satisfactory dynamic behavior, slightly influenced by the antagonist loads.
- Bottom left: the flow exiting the ideal pressure source is plotted. The dynamic observed is a redressed sine similar to the actuator position response. The frequency is doubled as the flow is provided in both positive and negative speeds of the actuator.
- Top right: the pressure in both chambers are observed. These are oscillating around an average point, in phase with the force oscillation shown at the bottom right.

Note: to avoid having a larger black picture, contrast on Figure 15 have been changed.

[6] https://www.modelon.com/products-services/modelon-deployment-suite/fmi-toolbox/

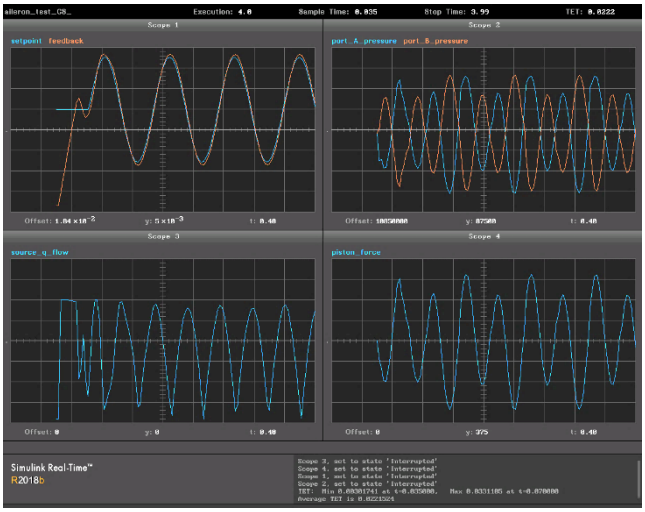


Figure 15. Plots of the main variables of interests in the Speedgoat GUI.

5 Summary and conclusions

This paper addressed the engineering workflow of model-based development of an aircraft hydraulic actuator. In order to be realistic, the focus was led on the need to enable a collaborative development – which involved several users and several tools. Modelon Impact was introduced as one of these and brought a new dimension when spreading the model usage to a relatively broad audience through relevant web-apps. Another key aspect for this collaborative development were the availability of Modelon Hydraulics library in several platform and Modelon tools to support FMI standard.

References

[COI16] Coïc, C., *Model-Aided Design of a High-Performance Fly-by-Wire Actuator, Based on a Global Modelling of the Actuation System using Bond-graphs.* PhD dissertation, INSA Toulouse/Airbus Helicopters, 2016.

[DRE14] Drenth, E., Törmänen, M., Johansson, K., Andersson, B.-A., Andersson, D., Torstensson, I. & Åkesson, J., *Consistent Simulation Environment with FMI based Tool Chain,* The 10th International Modelica Conference, Lund, Sweden, 2014.

[MAR18] Maré, J.-C., *Aerospace Actuators 3: European Commercial Aircraft and Tiltrotor Aircraft.* ISTE-Wiley, 2018.

[SAE12a] SAE Aerospace, *AIR994 – Recommended Practice for the Design of Tubing Installations for Aerospace Fluid Power Systems.* USA, 2012.

[SAE12b] SAE Aerospace, *AIR4253B – Description of Actuation Systems for Aircraft With Fly-By-Wire Flight Control Systems.* Aerospace Information Report, 2012.

DOI
10.3384/ecp2020174

Real-Time Simulation of an Aircraft Electric Driven Environmental Control System for Virtual Testing Purposes

Dirk Zimmer[1], Niels Weber[1], Peter Eschenbacher[1]

[1] DLR German Aerospace Center, Insitute of System Dynamics and Control,
Münchener Strasse 20, 82234 Oberpfaffenhofen, Germany,
dirk.zimmer@dlr.de

Abstract

A new object-oriented approach for the robust modeling of thermofluid systems has revealed to be also favorable for hard real-time simulation. This paper presents a complex and challenging application: an electric driven environmental control system architecture for potential use in future aircraft. The system is challenging to simulate in real-time because it also contains a computational expensive vapor cycle model and reconfigures because of switching by-passes. The system is complex to control due to its high degree of integration and interdependencies. Goal of this work is to integrate the model of this system for the virtual testing into an overall energy management for the aircraft on-board systems.

Keywords:Thermal fluids, Thermal process systems, real-time simulation, environmental control systems

1 Motivation and Background

1.1 Virtual testing of an Energy Management Algorithm

Algorithms for energy management are often a key design factor for the overall system design. This statements also holds true for our research field which is the system design of future More-Electric (or even All-Electric) Aircraft (MEA or AEA respectively). An intelligent energy management optimizes the use of power from different sources and thereby increases overall system efficiency. More important for an aircraft: the energy management can reduce peak loads and thereby help to reduce the weight of components such as generators or cables. This weight reduction in turn increases overall aircraft efficiency.

However, aircraft applications are subject to high safety standards and hence an energy management algorithm also has to be robust against failures or degradation of components. It is therefore necessary to test the interaction of the energy management algorithm with the corresponding subsystems.

A part of this testing has to be performed already at an early design stage when the systems are not yet available. Hence virtual testing is the only choice: here the energy management algorithm is executed on micro-controller architectures and the system response is generated by real-time simulation on conventional desktop PCs.

Especially interesting and challenging is the interaction of the energy management algorithm and the air generation unit (denoted as pack) of the environmental control system. It is interesting because the pack is the second largest power consumer on board only (vastly) surpassed by propulsion.

For the modelling and simulation community, this is challenging because neither the real-time simulation of such a pack nor the internal (prototypical) control are trivial tasks.

To gain a better understanding, let us first look at a Modelica model diagram of a potential electrical pack architecture and then study how this pack can interact with the energy management.

1.2 An Electric-Driven Environmental Control System Architecture

Figure 1 depicts a potential architecture for an electric vapour cycle pack (EVCP). It is one possible variant of the pack presented in (Golle, 2016), (Zimmer, 2018) and (Bender 2018). To gain a better understanding, let us remind that the function of a pack is to:

- provide a sufficient flow of fresh air (at different altitudes) to pressurize the cabin and keep air quality high.
- reach a desired pack discharge temperature (at different environmental conditions) to heat or cool the cabin.
- prevent fogging by ensuring an upper limit of the water content of the air.

We can now see these tasks being realized by the depicted architecture in the following way:

- Two compressors, a main compressor (5) and a smaller base compressor (3) provide the power for airflow generation, based on outside air (1). The base compressor can be bypassed (2) when not needed (as in operation near ground level). The primary heat-exchanger (4) between these two compressors prevents a too high outlet temperature of the main compressor.

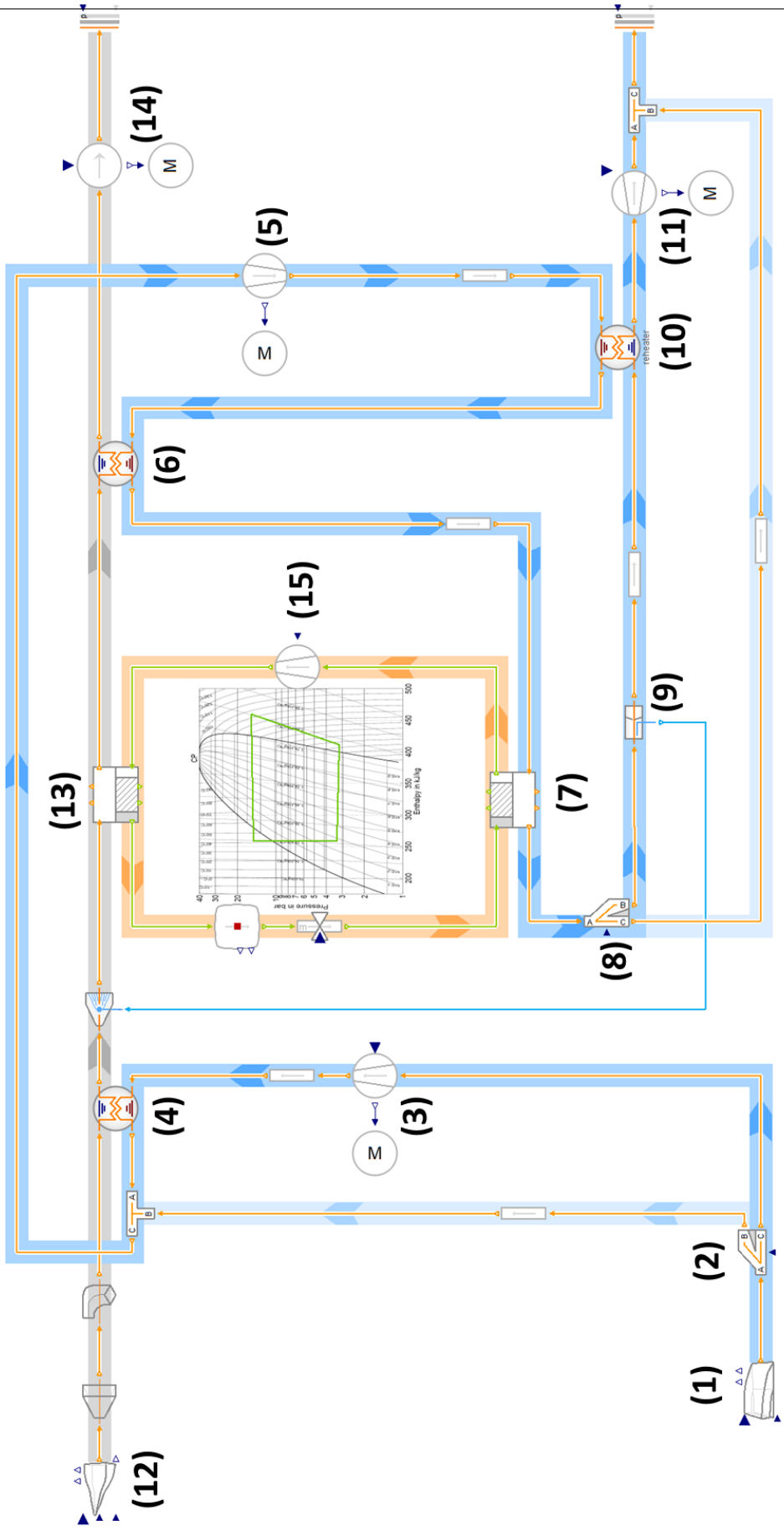


Figure 1: Modelica Diagram of an electric vapour cycle pack architecture (cleaned-up and freed from the control aspects)

DOI 10.3384/ecp202017477

- Cooling of the air is achieved by using the main heat exchanger (6) and the condenser (13) and evaporator (7) of the electric driven (15) vapor cycle. Heating can be achieved by closing the nozzle that is part of the turbine (11) and thereby expanding the air less efficiently than it has been compressed before.
- The reduction of the water content is performed by a water extractor (9) at the coldest point of the whole cycle. To prevent icing of the turbine (11), a reheater is used (10). On high altitudes (or in dry weather), the dehumidification is not needed and can be bypassed (8).

The heat sink for this thermodynamic process is the ram-air flow. This is either driven by the pressure of the ram-air door (12) in flight or actively propelled by fan (14) on ground.

For the sake of clarity, the control elements have been removed from this diagram. Even the prototypical control design for such a plant model with its non-linearity and reconfigurations is far from trivial. It will be quickly addressed in Section 4 but is essentially out of the scope of this paper. For the moment, let us focus on the interactions between the pack and the overall energy management.

1.3 Interaction between the Energy Management and the EVCP Pack.

Being the largest consumer of electrical power (and the second largest of overall power), the EVCP Pack offers substantial optimization for load management.

In case of overload scenarios (for instance when the airplane is operating with a failed generator), the Energy Management Algorithm may choose to reduce the power demand of the pack. To achieve such a desired demand reduction, the energy management manipulates the set points for the EVCP control and of the cabin pressure control. It may

- reduce the amount of fresh air supplied
- increase or lower the pack discharge temperature
- reduce the cabin pressure

How effective each of these actions will be with respect to load reduction strongly depends on the current flight and environmental conditions. Also each of these actions will (slightly or even severely) lower the passenger comfort. The energy management algorithm tries to maximize the lowering in power demand while it tries to minimize the lowering of passenger comfort. It does so based on a simplified model of the aircraft ECS system built from first-principles. The first-principle approach has the advantage that it is independent of the detailed technical realization of the EVCP Pack and also more robust in case of local failures or other forms of degradation.

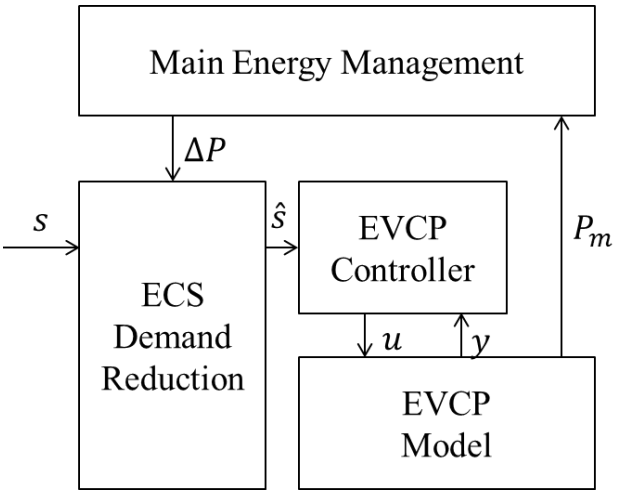


Figure 2: The main energy management issues a power reduction ΔP which is causing the ECS Demand Reduction to modify the set-point s of the EVCP controller to $\hat{s}$. The actual reaction in terms of power P_m is then fed back to the Main Energy Management.

However, this implies that the effective reaction of the EVCP Pack with its in-built controller may deviate from the expectations of the energy management algorithm. The energy management does take this into account by measuring the actual power demand of the pack and making it part of the control loop. Depending on the packs reaction, the energy management algorithm may hence strengthen or weaken its actions.

Understanding this interaction (see Figure 2) between energy management and pack is hence vital and as long as such packs are not available for testing, the test has to be performed virtually.

1.4 The Main Challenge

This paper focusses on setting up the EVCP pack model as hard real-time model so that it can be coupled with the inputs and outputs of the energy management algorithm running on a microcontroller.

This is a challenging task because such complicated models typically involve large non-linear equation systems and stiffness. Both prevent hard real time applications. The iterative solution of non-linear equation systems may fail to reach convergence within a fixed time frame. The stiffness of the system often demands stiff-stable ODE solvers which are implicit solvers and share the above problem of indefiniteness in finding a solution.

Fortunately, we can build on prior work (Zimmer, 2019) that is using a new approach for modeling fluid streams. This new approach avoids non-linear equations systems and provides means to manipulate eigendynamics of the mass flow rates without changing its major thermodynamic properties.

2 New Methods for Modeling Streams of Thermofluids

In 2018, we proposed a different format to organize the equations for thermodynamic fluid streams in an object-oriented way (Zimmer 2018, 2020) and implemented this in a Modelica Library. This has been further applied in (Zimmer 2019) and (Otter 2019). The reader is advised to study these papers to gain more insight. Yet we repeat the essential idea and its implications in this section.

2.1 Fundamentals

The new approach centers on the decomposition of pressure p into:

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

The term r is thereby defined as *inertial pressure*:

$$\Delta r = -L\frac{d\dot{m}}{dt}$$

Where L is the *inertance* of a fluid and follows straight from the geometry of the one-dimensional flow and (more importantly) is independent from the thermodynamic state of the fluid:

$$L = \int \frac{1}{A} ds$$

where s is the length in flow direction and A is the cross section area. This means that given all values of $\hat{p}$, and the current state of mass flow rates $\dot{m}$, we can compute $d\dot{m}/dt$ and all values of r in an arbitrary system by a linear equation system.

Now, what about $\hat{p}$? It is denoted as *steady mass-flow pressure* because evidently $p = \hat{p}$ for $d\dot{m}/dt = 0$. It is possible to compute all $\hat{p}$ for non-cyclical flows in explicit form from source to sink if we are willing to assume that the thermodynamic properties are a function of steady-mass flow pressure $\hat{p}$ and not of p.

This approximation can be justified for a wide array of applications since:

- it is only a transient phenomena, the error at steady-state will be zero.
- for gases, r is typically very small
- for liquids, the thermodynamic properties are often insensitive to r
- many formulas assume steady flow conditions anyway.

Cyclical flows have to be torn-apart by volume elements. Here the volume inlet represents structurally a sink and the outlet a source. If applied correctly the Block-Lower Triangular (BLT) form of complete thermofluid system will have the structure depicted in Figure 3.

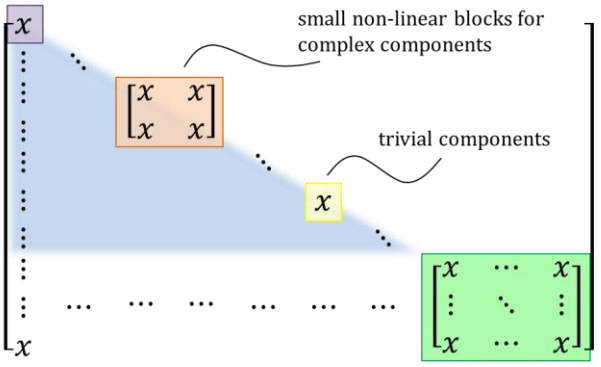


Figure 3: Illustration of the BLT Stucture using the new approach, taken out of (Zimmer, 2018). Green are linear equations. Everything else is potentially non-linear.

2.2 Implications for Real-time Systems

The BLT structure already reveals one major implication for real-time system: all non-linear equations form an explicit downstream computation if all individual components are put into explicit form for downstream direction. This is a feasible task and means that no non-linear equation system has to be solved at run-time by an iterative method.

What can we say about the stiffness of the resulting system? Is it possible to apply an explicit solver, like Runge-Kutta 3?

In general, this is not practically feasible. Using natural parameters, the mass-flow dynamics are often very fast with corresponding eigenvalues being strongly negative. The application of explicit solvers would thus demand for an excessively small step width and prevent real-time performance.

The goal is hence to manipulate these mass-flow dynamics so that a sufficiently large step-width can be applied but without impacting the remaining thermodynamic behavior of the system.

Fortunately the mass flow dynamics is largely subject to the linear equation system that is independent of the thermodynamic state and composed out of the law for the inertial pressure. Typically the corresponding matrix M (gree in Figure 3) consists in integer coefficients and values for the inertance. We now have three options to manipulate the fast mass-flow dynamics:

1. Manipulate the values for the inertance in M
2. Introduce artificial terms at the 0-entries of M
3. Restructure the matrix by using alternative equations.

It depends on the concrete application which of these options is best to apply. The paper (Zimmer, 2019) outlined the choice for a simplistic bleed-driven environmental control system. Let us repeat this exercise in the following section.

DOI 10.3384/ecp202017477

3 Mass Flow Dynamics in the EVCP

Despite the complexity of the pack, there are actually only 4 types of mass-flow dynamics of interest. These are:

- the mass flow dynamics within the ram-air channel,
- the mass flow dynamics of the refrigerant in the vapor cycle model,
- the mass flow dynamics of the main fresh-air ducting,
- The mass flow dynamics of the two by-passes.

3.1 The Ram-Air Flow

The control of the ram-air is actually an intricate issue. It is either controlled by a fan and/or by the ram-air door. The latter is thereby highly non-linear dependent on the current flight conditions especially the true air speed.

Fortunately, none of this is of concern. Simply manipulating the inertance of the ram-air flow does the trick (Method 1). Applying a factor of 50 is large enough to suppress too fast dynamics and yet the flow remains nicely controllable.

A solution in line with the next two sections could be designed as well but it was not necessary.

3.2 The Refrigerant Flow

The compressor of the vapour cycle (15) is power controlled. Typically it adapts the pressure raise for the current mass flow rate of refrigerant. So the tuple of power P and mass-flow rate $\dot{m}$ determines the pressure raise Δp

$$(P, \dot{m}) \rightarrow \Delta p \rightarrow \left(\frac{d\dot{m}}{dt}, r\right)$$

where $\dot{m}$ is a state whose derivative is dependent on Δp. Typically the time constant is very fast.

For a real-time solution, we hence choose to restructure the equation system (Method 3). Now we define the pressure raise as a state and prescribe the mass-flow rate such that the desired power level is reached. Its derivative and the corresponding inertial pressure directly follow from this.

$$(P, \Delta p) \rightarrow \left(\dot{\text{m}}, \frac{d\dot{m}}{dt}, r\right) \rightarrow \left(\frac{\Delta p}{dt}\right)$$

We determine the derivative of Δp by stating

$$\frac{\Delta p}{dt}\,\tau = r$$

We can now nicely control the time constant τ of the pressure raise and avoid too fast dynamics. This technique has also previously been outlined for valves in (Zimmer, 2019).

3.3 The Fresh-air Flow

The main compressor (5) shall control the mass flow to the desired set-point $\dot{m}_{set}$. As such it is even more straight forward than the vapor cycle compressor. We can restructure the dependencies straight forward to

$$(\dot{m}_{set}) \rightarrow \left(\dot{\text{m}}, \frac{d\dot{m}}{dt}, r\right) \rightarrow \left(\frac{\Delta p}{dt}\right)$$

and again apply

$$\frac{\Delta p}{dt}\,\tau = r$$

to keep the dynamics in check

3.4 The By-pass Flows

The natural dynamics of opening and closing by-pass valves is highly nonlinear. Mostly this is because it not only involves flow control valves but also check-valves to prevent potential counter flow. Check-valves are very non-linear (ideally even discrete) in their behavior. In this application, we can however avoid these troubles by simply stipulating the mass-flow split in the splitter (2) and (8).

In a classic free splitter (A being inlet, B and C being outlet), the mass flow balance is upheld and all inertial pressures are set equal:

$$\begin{gathered} m_A + m_B + m_C = 0 \\ r_B = r_A \\ r_C = r_A \end{gathered}$$

If we choose to prescribe the mass flow split by a parameter $0 < s < 1$, we gain one constraint equation and in turn have to remove the equality of all inertial pressures:

$$\begin{gathered} m_A + m_B + m_C = 0 \\ s \cdot m_A = -m_B \\ r_A = s\, r_B + (1 - s) r_C \end{gathered}$$

For the last equation, we assume that the flow is split under equal density. Then the forces acting on the corresponding cross-section areas of both sides have to be equal to each other (illustrated in Figure 4). This implicates the additional assumption that kinetic energy needed to change the mass flow rate can be exchanged between the two bypasses.

Such an assumption can be made (to a limited degree) for bypasses where one flow also propels its

bypass flow such as in a dedicated gap in a heat exchanger or the surrounding of a fan. For the bypasses as in (13) or (14) this assumption is actually invalid.

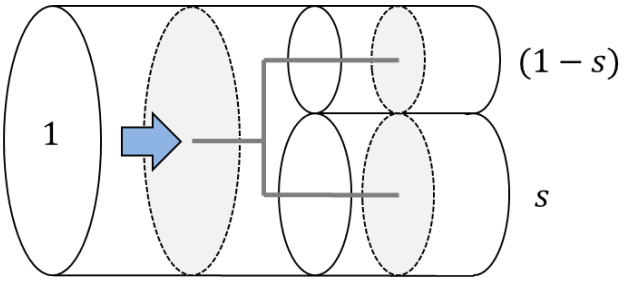


Figure 4: Illustration of a splitter with an enforced flow regime. A corresponding thought experiment illustrates a mechanic construction by 3 rigidly connected discs. The inertial pressures on these discs have to match resulting in the corresponding equation.

Fortunately we can still apply this method here because the bypass is either fully open or fully closed. At these extremes, this yields valid results and at the moment, the transient behavior is of no major interest.

A word of warning: for bypasses that are really meant to regulate something by controlled mixing (like bypasses for a high-frequency temperature control or trim-air valves), this method should definitely not be applied.

For the moment, we are seeing this solution as an intermediate solution to be replaced with better alternatives in the future.

4 Control of the EVCP

To reach the desired set-point (in terms of temperature, mass flow and maximum water content) a prototypical controller needs to be developed. Furthermore, a decision strategy for the opening and closing of the bypasses is needed.

All this is far from trivial and exceeds the scope of this paper. However, for the general understanding we briefly outline how a prototypic controller for the EVCP can be designed in functional terms.

4.1 Fresh Air Flow Control

The control of the desired amount of fresh air is performed by powering the main compressor (5). The main compressor may be supported by the base compressor. The decision whether to switch on or off the support the bypass is done based on the pressure ratio between air inlet pressure (1) and main compressor outlet pressure (5). The switch is implemented using hysteresis. This pressure ratio may also be influenced by the bypass to the turbine (8).

4.2 Temperature Control

For the cooling case, the pack discharge temperature is controlled by the flow of ram-air which is in turn controlled by the ram-air door or subsequently (at or near) ground by the ram-air fan. The gain factor of the control inputs is thereby highly non-linearly dependent on the flight conditions. Temperature control is also relatively slow due to the heat capacity of the heat-exchangers and the latency of the refrigerant in the vapour cycle.

For the heating case, the ram-air flow needs to be limited to a minimum and the variable nozzle of the turbine (11) lowers its efficiency. The inefficient expansion of previously compressed air enables heating.

4.3 Humidity control

The effectiveness of water separation is essentially influenced by the temperature at the water separator (9). In combination with the overall pack discharge temperature control this is influenced by the variable nozzle of the turbine (11). An inefficient expansion requires a lower temperature at the inlet of the reheater (10) at hence enables a more efficient water extraction.

In case the fresh air has too little water content in the first place, the energy expensive water extraction can be bypassed (8) unless heating is required. Also this discrete decision is implemented by hysteresis.

Humidity control only extracts water. Water is not added in case the air is too dry (unfortunately).

4.4 Remaining Degrees of Freedom.

There remain two degrees of freedom that can be used to optimize the performance during operation as part of the overall thermal management.

One is the powering of the compressor of the vapour cycle (that again contains its own sub-controllers and potential bypasses). This can be used to regulate the temperature gradient in the ram-air channel so that each heat-exchanger is used effectively.

The other degree of freedom is the turbine pressure ratio. This enables a trade-off between electric power needed by the EVCP Pack (high pressure ratio) or more additional drag caused by the ram-air channel opening (low pressure ratio).

In summary the overall control implementation embraces tasks from high-level thermal management down to internal pack reconfiguration and is hence a very complex task.

DOI 10.3384/ecp202017477

5 Simulation Results

The final model including the controller includes 44 states and more than 1500 time varying variables.

A few of the states are needed for the design of the controllers but most of them describe time-constants of the heat-exchangers and some of them have been introduced in order to enable an explicit computation of the non-linear equations within a component (Zimmer, 2013).

After the manipulations (as sketched in Section 3), the dynamics of all states is slow enough to enable a real-time simulation. Runge-Kutta 3 with a step-width of 0.1s was chosen as integration method.

With these settings, the model performs 5 times faster than real-time on a conventional desktop PC including writing the simulation results on the hard disk. The computation for the vapour cycle thereby dominates the computational demand (>85%). This is because of the complex media model and because of the complex moving boundary heat-exchanger models.

Figure 5 shows the simulation result for a simplistic ascent of the airplane from ground to flight altitude. This scenario changes the environmental conditions, the cabin pressure requirements and the required pack-discharge temperature. Some of these transients are relatively quick.

The three curves in Figure 5 show the pack discharge temperature by 3 different curves: the desired set point (the control target) and the actual resulting pack discharge temperature from the original model and the modified real-time model. The occurring differences demand explanation.

For both models there is a noticeable difference between the set-point and actual value indicating a relatively poor control quality. There are good reasons for that: First of all, the implemented controller is prototypical and certainly not optimal. Second the opening and closing of bypasses that happens during ascent, causes major disturbances that are difficult to compensate without preemptive action. Last but not least, the control authority is limited also for a real system. It simply takes time to react. Hence the actual system would also include further bypasses for higher-frequency temperature control. These are omitted here because we currently focus on the main energetic behavior and not on comfort. However, the gain of control authority by dedicated temperature control valves has its energetic penalty as well and hence they will be included in future versions of our work.

The difference between the original model and its real-time derivative is essentially due to the disturbance on the control due to the reconfigurations: turbine bypass opens around 33 min, compressor bypass closes around 50 min. As outlined in Section 3.4, the quality of the bypass regulation is changing during the transition period and consequently also the quality of the disturbance. Hence the two models (who share their control scheme) react differently.

For the moment, this difference between the models is acceptable for us because the general model uncertainty (also regarding the control design) with respect to reconfigurations is very high. However, this is something that requires more effort in the future and is currently regarded as an intermediate solution.

Also these results indicate that such reconfigurations may pose difficulties in interaction with an energy management and hence a closer investigation of this topic may be needed.

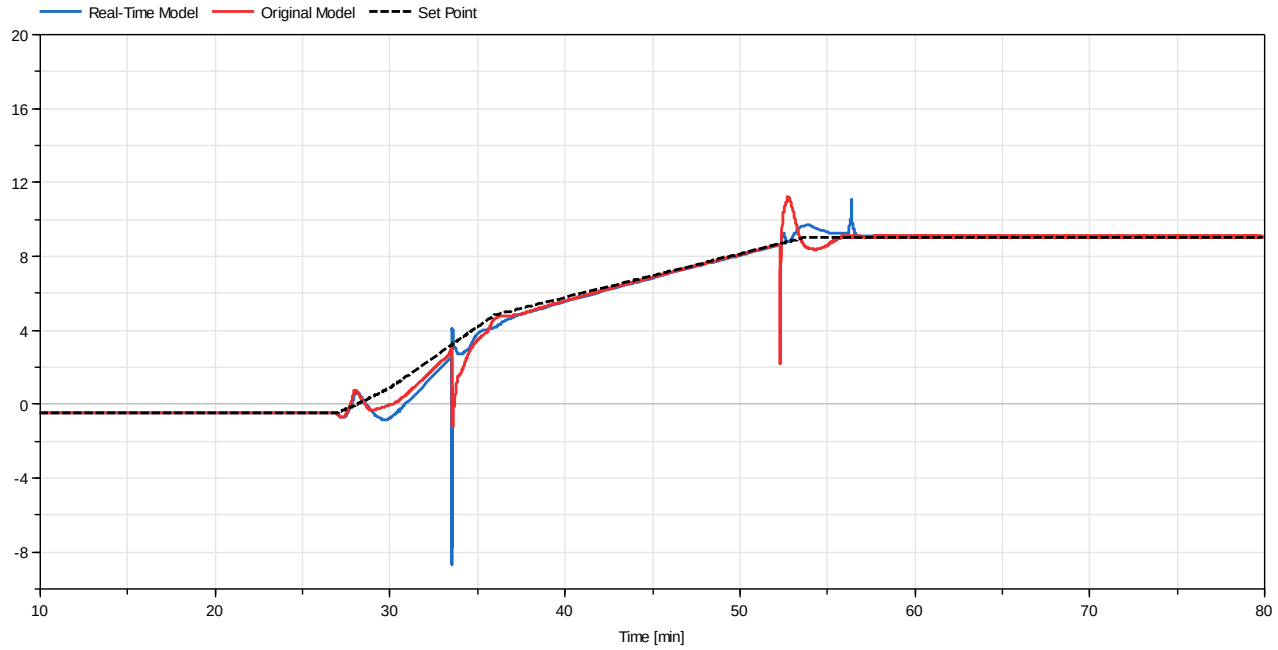


Figure 5: Pack discharge temperatures at flight ascent. This graph shows the control quality w.r.t to the setpoint and the difference between the real-time simulation and the original model.

6 Conclusions

6.1 Summary

The provided example of an EVCP successfully demonstrates that hard real-time simulation of even complex thermofluid systems becomes a fairly well achievable task with Modelica. The example contains almost all what is typically regarded as challenging: vapor cycles (see also Schulze, 2011), complex heat-exchanger models, bypasses, compressors, turbines and complex boundary conditions. Also it handles 3 different kinds of media: moist air, water, and a refrigerant. This gives us confidence for future challenges and larger systems.

The main remaining challenge is a better solution for bypasses with check-valves. The proposed solution here is good enough as long as the transition period is not important but the reactions of the control system reveal noticeable differences after all.

Finally, the hard real-time simulation of such thermodynamic processes should not be such an issue in the first place. Typically all time-constants of interest are well above the desired communication interval. Only large non-linear equation system and fast mass-flow dynamics oppose a quick simulation. The new approach (Zimmer, 2018, 2019), (Otter, 2019) based on the inertial pressure allows to prevent the former and isolate the latter.

A final remark (because it often gets forgotten): Hard real-time simulation is actually a very slow form of simulation because it solely focusses on a guaranteed response time. Without doing all of the work outlined in this paper but simply using implicit solvers with variable step-width such as ESDIRK23 (Jørgensen, 2018) on the original model, the systems simulates much faster than real-time, just not with a guaranteed response time.

6.2 Outlook on future work

This paper focused on setting up the main EVCP model for hard real-time simulation. The main task, which is the coupling with overall management algorithm, has yet to be conducted.

This will reveal how well the energetic behavior of a complex system such as the EVCP can be handled by an overall energy management. Another questions is how sudden events such as the reconfiguration of the system impact the energy management in return.

Acknowledgements

The work is receiving funding from European Union's Horizon 2020 for the Clean Sky 2 Joint Technology Initiative under grant agreement H2020-CS2-CFP03-2016-01 GAN 737792.

References

Bender, Daniel (2018) *Exergy-Based Analysis of Aircraft Environmental Control Systems and its Integration into Model-Based Design*. Dissertation, Technische Universität Berlin.

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

John Bagterp Jørgensen, Morten Rode Kristensen, Per Grove Thomsen (2018) A Family of ESDIRK Integration Methods In: *Numerical Analysis*

Otter, Martin et al., (2019) Thermodynamic Property and Fluid Modeling with Modern Programming Language Constructs. In: *Proceedings of 8th International Modelica Conference. 13th International Modelica Conference*, Mar 04-06, 2019, Regensburg, Germany.

Schulze, Christian (2011) Real-Time Simulation of Vapour Compression Cycles. In: *Proceedings of 8th International Modelica Conference*., 20.-22. Mar 2011, Dresden.

Zimmer, Dirk (2020) Accepted for publication: Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks In: *MCMDS Journal*.

Zimmer, Dirk (2019) Towards Hard Real-Time Simulation of Complex Fluid Networks. In: *Proceedings of the 13th International Modelica Conference*, 04.-06. Mär. 2019, Regensburg, Deutschland

Zimmer, Dirk and D. Bender, A. Pollok (2018) Robust Modeling of Directed Thermofluid Flows in Complex Networks. In: *Proceedings of the 2nd Japanese Modelica Conference*, Tokyo, Japan

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

Dynamic Modeling and Simulation of Reformed methanol Fuel Cell System Using Modelica

Xin Yao Tian[1] Lin Lin Yang[1] Rui Gao[2]

[1] Dalian Institute of Chemical Physics, Chinese Academy of Sciences, Dalian 116000, Liaoning
{tianxinyao,yanglinlin}@dicp.ac.cn

[2] Modelon K.K. Roppongi 1-10-3-209, Minato ku, Tokyo, 106-0032, Japan
rui.gao@modelon.com

Abstract

In this paper, a dynamic model of a reformed methanol fuel cell (RMFC) system is developed. The base fuel cell model used in the modeling work was provided by Modelon K.K., using Modelica language. RMFC is a coupled multi-physics system involving fluid mechanics, heat & mass transfer, chemical & electrochemical reaction. Therefore, the dynamic modeling and simulation analysis of RMFC is valuable for optimizing operation parameters, as well as system controller design. The component model of methanol reformer, methanol catalytic burner and high temperature polymer electrolyte fuel cell are developed and integrated into a system model using Dymola®. And the models have been verified by comparison with the experimental data. This model is expected to provide a base for developing the optimal control strategy for the RMFC.

Keywords: Reformed methanol fuel cell, High temperature polymer electrolyte fuel cell, Dynamic modeling, Modelica,

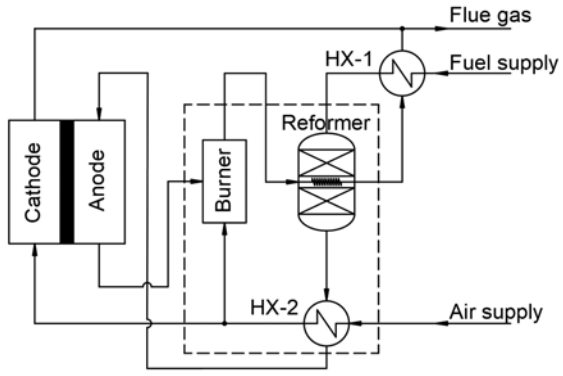
1 Introduction

Polymer electrolyte membrane fuel cells (PEMFC) are receiving considerable attention, since they convert chemical energy directly into electrical energy with high efficiency and low emission of pollutants. However, PEMFC have a problem with hydrogen storage, transportation and distribution, either under high pressure or on liquid form cooled down to below -253 ℃. One possible solution to this problem is to use a liquid fuel as a hydrogen carrier and reform it into a hydrogen rich gas as it is needed (Justesen K K, 2015). Methanol is a compact form of liquid hydrogen at ambient conditions. On a volume basis, methanol has 40% more hydrogen than liquid hydrogen, and 140% more hydrogen than hydrogen compressed at 700 bar (Choon F S, 2018).

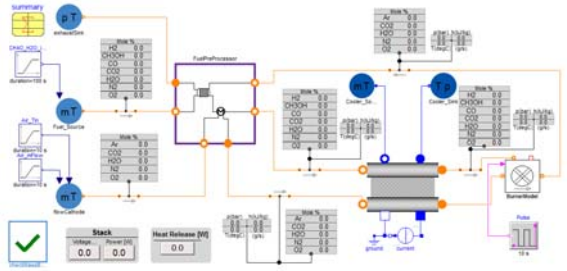
A typical reformed methanol fuel cell system consists of three main subsystems: a fuel processor, a high temperature polymer electrolyte membrane (HTPEM), fuel cell power system, and a heat recovery system, As shown in Figure 1.

The methanol reformer in combination with an HTPEM (operating at 120 ~ 180℃) fuel cell leads to simpler system structure compared to the LTPEM (operating at 60 ~ 80 ℃). Because HTPEM have strong CO resistance up to 30000ppm (10ppm for LTPEM), eliminating CO cleanup stage in the fuel processor subsystem (Jannellia E, 2005).

Due to the complex configuration, dynamic modeling and optimization are essential to improve the performance and efficiency of the RMFC system. Most of the fuel cell system dynamic model are developed with Simulink, a block-based environment for multidomain simulation. In this paper, object-oriented equation based language, Modelica, is used for modeling RMFC, which can be easy to modify and reuse. Modelon's commercial Fuel Cell Library (FCL) contains predefined reactors for fuel preprocessing and internal reforming, and predefined stack structures. Besides, FCL provides powerful media model to capture the property of fuel, air and reaction gas. The main contribution of this paper is to create a RMFC model using the Modelon Fuel Cell Library and validate the model with experimental results.



(a)



(b)

Figure 1.(a)Reformed methanol fuel cell system process flow chart(b)The fuel cell system model in Dymola

2 Model

2.1 Fuel processor subsystem

The dashed box in Figure 1 outlines the fuel processor subsystem. A typical fuel processor may consist of a catalytic burner, a reformer and a heat exchanger (HX-2).

Catalytic combustion reactions include oxidative combustion of methanol, carbon monoxide, and hydrogen. The burner model has a volume model of air and fuel inlets and air outlets. It also has an energy connector for heat to the environment and a Boolean input for ignition signals. According to the stoichiometric matrix, the fuel-to-fuel ratio is specified and the ignition signal is true to ignite the burner.

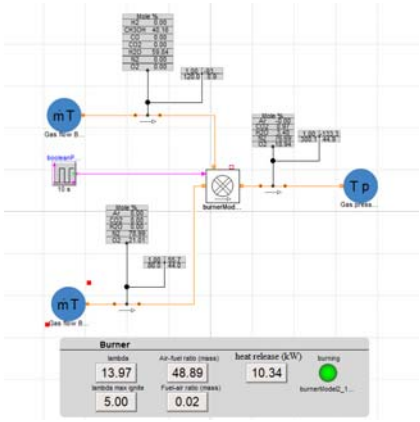


Figure 2. The catalytic burner model

Methanol steam reforming for hydrogen production includes reforming (MSR), cracking (MC) and shift conversion (WGS) (Andreasen S J, 2014). The reformer model is shown in Figure 3. The reformate is cooled to the operating temperature of the stack and enters the anode of the stack as fuel for the fuel cell. As shown in Figure 4.The reaction is as follows:

MSR: $CH_3OH + H_2O \rightarrow CO_2 + 3H_2$ (1)

MC $CH_3OH \rightarrow CO + 2H_2$ (2)

WGS $CO + H_2O \leftrightarrow CO_2 + H_2$ (3)

The water gas shift reaction (3) is ignored in the model because the CO concentration is very low, less than 1vol%, at the working temperature (250~280℃) of the reformer. The reforming reactor model in this paper includes methanol fuel gas inlet, reformed gas outlet and catalytic burner tail gas as heat exchange medium.

Reaction kinetic equation (O.P. Klenov, 2015):

DOI 10.3384/ecp202017485

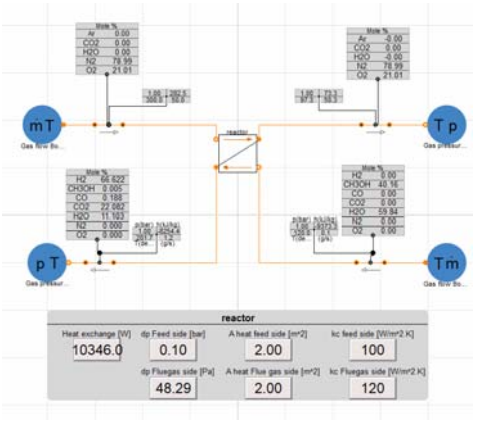


Figure 3. The reformer model

The reformate is cooled to the operating temperature of the stack and enters the anode of the stack as fuel. The fuel processor subsystem As shown in Figure 4.

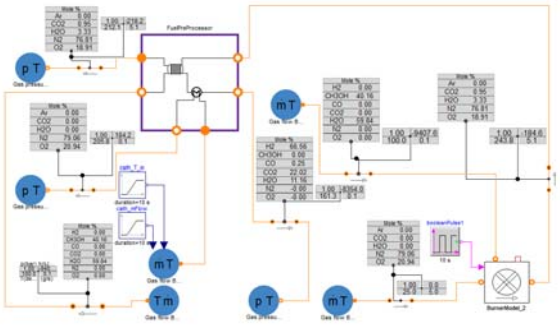


Figure 4. The fuel processor subsystem model

$$CH_3OH + H_2O \overset{k_1}{\rightarrow} CO_2 + 3H_2 \tag{4}$$

$$CH_3OH \overset{k_2}{\rightarrow} CO + 2H_2 \tag{5}$$

$$r_{CH_3OH} = -k_1 C_{CH_3OH} - k_2 \tag{6}$$

$$r_{H_2O} = -k_1 C_{CH_3OH} \tag{7}$$

$$r_{CO_2} = k_1 C_{CH_3OH} \tag{8}$$

$$r_{CO} = k_2 \tag{9}$$

$$r_{H_2} = 3k_1 C_{CH_3OH} + 2k_2 \tag{10}$$

$$k_1 = \frac{[A_1 + B_1 \ln(\frac{S}{M})]e^{(\frac{-E_1}{RT})}}{P^{D_1}} \tag{11}$$

$$k_2 = \frac{A_2 e^{(\frac{-E_2}{RT})}}{P^{D_2}} \tag{12}$$

2.2 HTPEM fuel cell stack

Due to the irreversible loss of the HTPEM fuel cell during the reaction, the actual cell potential is less than the equilibrium potential. The irreversible loss of an actual fuel cell is often referred to as a polarized overpotential or overvoltage. There are three main types of polarization that cause irreversible losses: ① activated polarization; ② ohmic polarization; ③ concentration polarization. These losses cause the fuel cell voltage to be less than the ideal potential (Daniel, 2016).

The output voltage V_{cell} of a proton exchange membrane fuel cell. It is expressed as follows:

$$V_{cell} = E_{Nernst} - \eta_{act} - \eta_{ohm} - \eta_{con} \tag{13}$$

In the formula, E_{Nernst} is thermodynamic electromotive force, η_{act} is activation overvoltage, η_{ohm} is ohmic overvoltage, and η_{con} is concentration overvoltage.

For N_{cell} fuel cell units connected in series to form a fuel cell stack, the voltage can be expressed by the formula(14).

$$V_{stack} = N_{cell} \cdot V_{cell} \tag{14}$$

(1) Thermodynamic potential

According to the equation of the hydrogen / oxygen fuel cell, the thermodynamic electromotive force can be obtained.

$$E_{Nernst} = \frac{\Delta G}{2F} + \frac{\Delta S}{2F}(T_{stack} - T_{ref}) \tag{15}$$

In the formula (15), ΔG is the change value of Gibbs free energy; F is the Faraday constant (96485C); ΔS is the change value of entropy; R is the gas constant; T_{stack} is the temperature of the stack.; T_{ref} represents the reference temperature. The simplified expression is as follows:

$$E_{Nernst} = 1.180 - 2.453 \times 10^{-4}(T_{stack} - 323.15) \tag{16}$$

(2) Activation loss

The activation loss mainly manifests when the electrode surface is about to activate the electrochemical reaction, because the electrode cannot overcome the phenomenon of slow rate of activation energy required for the charge transfer process in the electrochemical reaction. Generally, both the anode and the cathode

generate an activation overpotential, but the reaction rate of the anode is much faster than that of the cathode. Therefore, the activation overpotential is generally determined by the reaction conditions of the cathode. According to the Tafel equation, the activation loss η_{act} in this paper is expressed as follows:

$$\eta_{act} = \frac{RT_{stack}}{(2\alpha_a F)} ln \frac{i}{i_a^0} + \frac{RT_{stack}}{(4\alpha_c F)} ln \frac{i}{i_c^0} \quad (17)$$

$$i_a^0 = i_{a\_ref}^0 \cdot \alpha_a (\frac{C_{H_2}}{C_{H_2 ref}})^{\gamma_a} \cdot e^{(-1400(\frac{1}{T_e} - \frac{1}{T_{ref_a}}))}$$
$$i_c^0 = i_{c\_ref}^0 \cdot \alpha_c (\frac{C_{O_2}}{C_{O_2 ref}})^{\gamma_c} \cdot e^{(-7900(\frac{1}{T_e} - \frac{1}{T_{ref_c}}))} \quad (18)$$

(3) Ohmic loss

The ohmic resistance is mainly due to the ionic resistance of the proton exchange membrane. The high temperature membrane ohmic loss η_{ohm} is expressed by the formula (20), estimated using Ohm's law.

$$R_{ohm} = 0.4025 - 0.0007 T_{stack} \quad (19)$$

$$\eta_{ohm} = i_{stack} \cdot R_{ohm} \quad (20)$$

(4) Concentration loss

Mass transfer affects the concentration of hydrogen and oxygen at the surface of electrocatalyst. Poor mass transport leads to a significant concentration reduction of the oxygen and hydrogen within the catalyst layer. It can be shown, by considering the Nernst and Tafel equation, that concentration has an effect on the fuel cell voltage .The concentration overvoltage η_{con} is expressed as formula (21)~(25).

$$\eta_{con} = \left(-\frac{RT_{stak}}{\alpha_a F} \ln\left(1 - \frac{f(\lambda_a) \cdot i}{3000}\right)\right) + \left(-\frac{RT_{stak}}{\alpha_c F} \ln\left(1 - \frac{f(\lambda_c) \cdot i}{2000}\right)\right) \quad (21)$$

$$f(\lambda_a) = -\lambda_a \cdot \ln(1 - \frac{1}{\lambda_a}) \quad (22)$$

$$f(\lambda_c) = -\lambda_c \cdot \ln(1 - \frac{1}{\lambda_c}) \quad (23)$$

$$\lambda_a = \frac{2FN_{H_2}}{i} \quad (24)$$

$$\lambda_c = \frac{4FN_{O_2}}{i} \quad (25)$$

(5) Single cell equivalent model

The circuit is shown in the Figure 5. In the figure, R_{ohm} is the ohmic loss, and the Voltage V already includes activation loss and concentration loss. When the demand current changes, the corresponding voltage changes accordingly.

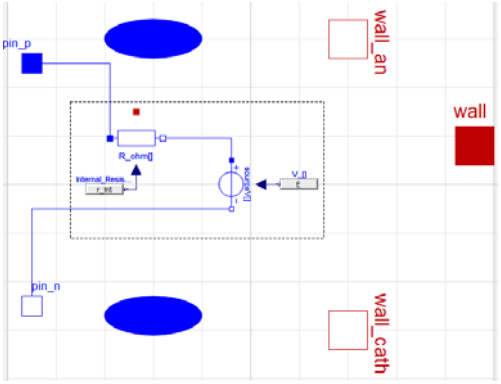


Figure 5. Single cell Model

(6) Stack model

A stack model made of N_{cell} cells connected in series includes an anode gas flow channel, a cathode gas flow channel, a mass connector, an energy connector, and an electrical component connector. The hydrogen fuel gas required for the anode is provided by the reforming reaction of methanol, not pure hydrogen, so the fuel cell voltage must also consider the voltage drop caused by Carbon monoxide poisoning.

The voltage drop caused by carbon monoxide poisoning can be expressed as:

$$dV_{poi} = 2.5 \times 10^4 e^{(-0.0424 T_{stack})} \cdot i + 7.48 \times 10^3 e^{(-0.0436 T_{stack})} \cdot i \cdot \ln(\frac{y_{CO}}{y_{H_2}}) \quad (26)$$

Combined with the above equivalent model, the stack voltage V_{stack} at steady state can be expressed as:

$$V_{stack} = N_{cell} \cdot (E_{Nernst} - \eta_{act} - \eta_{ohm} - \eta_{con}) \quad (27)$$

The output power P_{stack} and consumed power $P_{consumed}$ of the stack are shown in the following formula (28)~(29) The efficiency of the stack η is shown in the formula (30).

$$P_{stack} = V_{stack} \cdot i \quad (28)$$

$$P_{consumed} = N_{cell} \cdot (\eta_{act} + \eta_{ohm} + \eta_{con}) \cdot i \quad (29)$$

$$\eta = \frac{V_{stack} \cdot i}{N_{cell} E_{Nernst} \cdot i} = \frac{V_{cell}}{E_{Nernst}} \quad (30)$$

DOI 10.3384/ecp202017485

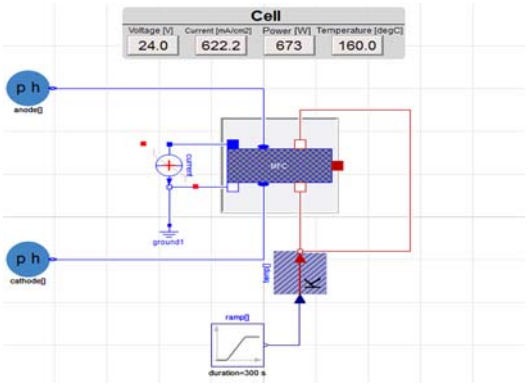


Figure 6. HTPEM stack model

2.3 Heat recovery subsystem

As shown in the reformed methanol fuel cell system, the high temperature fuel gas exiting the reformer is used to preheat the air entering the cathode of the stack. As introduced in Section 2.1. This is also conducive to cooling the reforming outlet gas to a temperature suitable for the anode of the stack. The waste heat of the flue gas at the burner outlet is further recovered, and this part of the heat is used for vaporizing methanol.

3 Results and Discussion

The remaining parameters of the system model are shown in the following table.

Table 1. Parameters of system model.

Parameter symbol	***Value***
Active cell area	45cm^2
Reformer Fuel molar fraction	
H_2O	59.84%
CH_3OH	40.16%
Anode Fuel molar fraction	
H_2	66.62%
CO	0.19%
CO_2	22.08%
H_2O	11.11%
CH_3OH	0.0049%
Air oxygen fraction	21.01%

According to the single cell model built earlier, and the operating parameters in the Table 1, the simulation of the i-V curve is performed. Set the temperature of the stack to 120 ° C, 150 ° C, and 180 ° C, respectively. The i-V curves are shown in Figure 7 (a) (b) (c), and the model fit well with the experimental data.

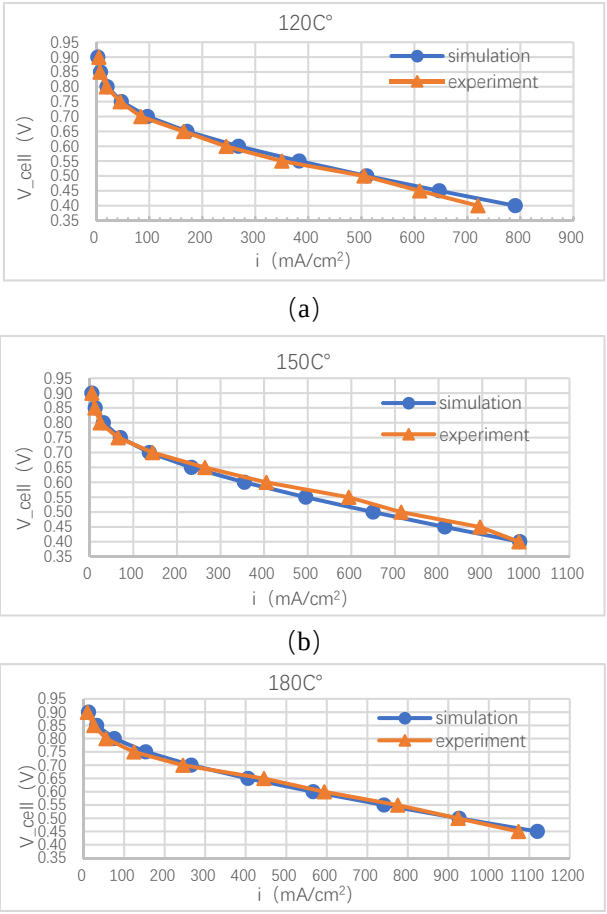
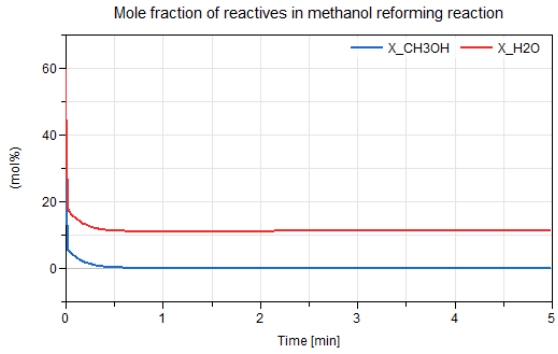


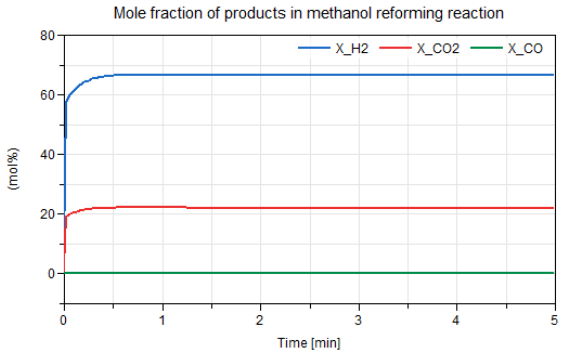
Figure 7. Comparison between experimental and computational results of HTPEM fuel cell performance

Figure 8 and Figure 9 shows the dynamic response of each component in the reformer and catalytic burner over time. The reformer outlet is the fuel of the anode The reaction quickly reaches its steady state value, and then there is almost no change. This composition is in line with similar simulation results in related literature.

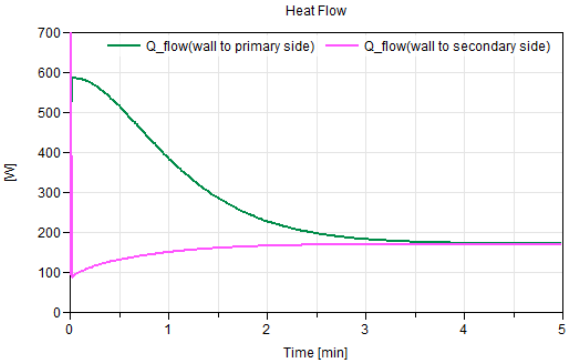
The heat exchanger HX-1 uses residual heat to vaporize a 0.147g/s methanol aqueous solution, recovers about 27% of flue gas waste heat. And HX-2 preheats cold air from 25℃ to 120℃. This part of the heat is provided by the reformer outlet gas.



(a)

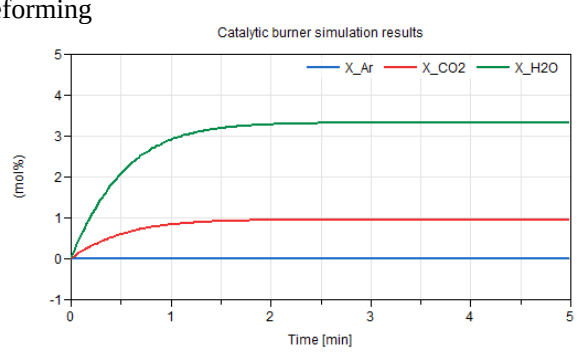


(b)

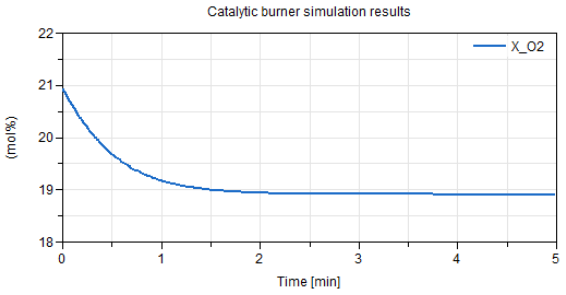


(c)

Figure 8. Simulation results for: (a) methanol,water (b) hydrogen CO, CO_2 content (c) heat required for reforming



(a)



(b)

Figure 9. Catalytic burner simulation results for: (a) CO_2, H_2O content (b) O_2 content

4 Conclusion

It can be concluded that Dymola and Modelica are very convenient tools for modeling fuel cell systems. Modelica is an equation-based language that offers object-oriented, structural, and multi-domain coupled modeling possibilities and flexibility.

This paper introduces a simple reformed methanol fuel cell system includes a fuel processor subsystem, HTPEM fuel cell stack and heat recovery subsystem. It can provide a basic model for RMFC system analysis and design.

Nomenclature

T	Temperature
P	Pressure
R	Universal gas constant
r	Reaction rate
k_i	Reaction rate coefficient
A_i	Parameter in Eq. (12-13)
B_i	Parameter in Eq. (12)
D_i	Parameter in Eq. (13)
$\frac{S}{M}$	Ratio of methanol to water
F	Faraday constant
V	Voltage
i	Current density
y	Molar fraction
λ	Stoichiometry
α	Charge transfer coefficient
γ	Pre-exponential factor

DOI 10.3384/ecp202017485

References

Justesen K K, Andreasen S J, Pasupathi S, et al. Modeling and control of the output current of a Reformed Methanol Fuel Cell system[J]. International Journal of Hydrogen Energy, 2015, 40(46):16521-16531.

Choon Fong Shih, Tao Zhang, Jinghai Li , et al. Powering the Future with Liquid Sunshine[J]. Joule 2, 1925–1949, October 17, 2018 1927.

Jannellia E, Minutilloa M, Pernab A. Analyzing microcogeneration systems based on LT-PEMFC and HTPEMFC by energy balances. Am Control Conf 2005; 108: 82-91.

Andreasen S J, Ashworth L , Sahlin S , et al. Test of hybrid power system for electrical vehicles using a lithium-ion battery pack and a reformed methanol fuel cell range extender[J]. International Journal of Hydrogen Energy, 2014, 39(4):1856-1863.

O.P. Klenov, L.L. Makarshin, A.G. Gribovskiy, et al. CFD modeling of compact methanol reformer[J]. Chemical Engineering Journal, 2015, 282: S1385894715004817.

Daniel, Ikhu-Omoregbe, Myalelo, et al. Parametric Analysis of a High Temperature PEM Fuel Cell Based Microcogeneration System[J]. International journal of chemical engineering, 2016.

DOI
10.3384/ecp2020174

Coupling of Modelica and Biochemical Simulator, SUMO, by Using C-API

Satomi Nishida[1] Takayuki Ohtsuki[2]

[1]Modelon KK, Japan, satomi.nishida@modelon.com
[2]Kurita Water Industries Ltd., Japan, t.ootsuki32@kurita-water.com

Abstract

Wastewater treatment engineering field has its own domain specific simulation needs and especially for the biological treatment process, which is the main process of municipal wastewater treatment, large scale dynamic models have been heavily used since the late 80's along with its specific model description method. However, most of wastewater treatment process simulators focus on modelling of the treatment reaction performance of the treatment process units involved and lacks capability to analyze related mechanical components such as valves, pumps and blowers in detail. On the other hand, need for mechanical component performance analysis has been growing from the cost management perspective, for example, minimization of electric energy consumption and maintenance interval estimation because a large part of these cost happens at the mechanical components in the system. Operation improvement should be achieved without deterioration of the treated effluent water quality, therefore, the simulator coupling based on system analysis approach utilizing existing process models and Modelica based mechanical component models can be promising option.

This paper reports the results of coupled simulation framework development for a biological wastewater process simulator, SUMO, and Modelica. Biological treatment process and its related mechanical components were simulated by SUMO and Modelica, respectively. Dynamic simulation was conducted to evaluate the influence of controller parameter settings for the aeration system in a specific biological process. Specifically, the impact on the effluent quality, on the electric energy consumption and on the cycle frequency of the air flow control valves. It was also demonstrated that Modelica can be used with domain specific models for industries where Modelica and related model interface technology like FMI is not common, as far as such domain simulator supports a simple low-level C language interface.

Keywords: Modelica, SUMO, Wastewater Treatment, Biological process, Simulator coupling, C-API

1 Introduction

Main focus of wastewater treatment system is always effluent water quality stability, and it requires keeping some allowance on operation conditions to meet influent wastewater load fluctuation which is the main cause of the system disturbance for the treatment system; this way of system management often causes energy and cost loss. On the other hand, recent days even in the wastewater treatment field, energy reduction and cost reduction are gaining more attention and this situation requires operation management that takes into account energy consumption reduction as well as water quality stability. Energy consumption in such process happens typically at mechanical systems attached to the water treatment process such as pumps and blowers. This is the main motivation to conduct system simulations that allow analysis of the biological process and mechanical component performance simultaneously.

Although a number of simulators for chemical and biochemical processes are used in the water treatment process industry, they often lack capabilities to analyze the performance of mechanical components used. Modelica could be one of choices for such mechanical component simulation, however, technologies of Modelica and FMI are not widely used in the water treatment process industries at this moment.

SUMO is a commonly used simulator for biological wastewater treatment. It has a C language interface called C-API and can be invoked from Modelica by using External Object.

In this paper, a biological wastewater treatment system was modeled. A simulator coupling framework was implemented using the C-API. The case study with various conditions was conducted, and the effect of conditions on both the biological process and the mechanical component operation was evaluated. Dymola (Dassault Systems) is used as our Modelica platform.

2 Water Treatment Background

2.1 Overview

Water treatment processes are mainly classified into physicochemical methods and biological methods (Sperling and Chernicharo, 2005). Physicochemical methods separate pollutants by size, specific gravity, and chemical characteristics, *e.g.* sedimentation and filtration. Biological methods treat water by using the absorption and decomposition by microorganisms, *e.g.* aerobic activated sludge process and anaerobic digestion process.

Biological wastewater treatment processes are commonly used for decomposition of organic pollutants. The adaptability of biological consortia to wide range of organic contaminants and the high energy efficiency of biological catalytic reactions have made it a main treatment process in municipal wastewater treatment processes and also in industries which generate organic wastewater. Aerobic wastewater treatment processes require oxygen for microorganism activity, and blowers to supply air consume most of the electric energy of the treatment process.

Wastewater treatment systems for industries are generally composed of various types of water treatment processes to achieve certain level of effluent quality. But still the biological wastewater treatment process is the key process for organic contaminant removal. And its blower is known to be the main energy consumer of the system. This report focused on a Moving Bed Biofilm Reactor (MBBR) as one of the typical aerobic activated sludge biological treatment processes used in industry.

2.2 Moving Bed Biofilm Reactor System

An MBBR was originally developed by Kaldnes Miljøteknologi. It consists of aeration tanks with moving media where biofilm grows. The media are continuously moved by vigorous liquid flow driven by aeration (Sidek *et al.*, 2016). MBBR is widely used for various wastewater treatment mainly because it allows one pass treatment which makes its operation easier compared to conventional suspended growth type activated sludge system without internal media which requires sludge return to keep sufficient biomass concentration in its main reactors. On the other hand, it is also known that the oxygen dissolution efficiency of MBBR is inferior to that of conventional activated sludge process because the media enhances coalescence phenomena of microbubbles injected by blowers and air diffusers which effectively reduces the surface area for the oxygen gas transfer to liquid phase. The electric energy consumption for aeration of MBBR becomes larger due to this poor oxygen dissolution efficiency. Therefore, especially in an MBBR process, aeration control strategy is important to minimize this larger energy requirement.

Figure 1 shows the example configuration of an MBBR system, which is modeled in this report. This system has three reactors, A, B and C. Wastewater is supplied to the tanks A and B equally, and then cascades into the tank C with an expectation of polishing effect. The header pipe pressure is kept constant by controlling the blower motor rotation speed through an inverter. Dissolved oxygen (DO) concentration sensors are installed at each tank. The airflow to the tanks A and B is controlled by multiple on/off valves to keep DO concentration at its target value. Air flow to the reactor C is kept almost constant.

To supply air to the reactors, MBBR is equipped with aeration unit. This example configuration has several Roots-type blowers. Each blower is equipped with an inverter, and the blower outlet pressure is kept constant by the blower speed control. If the outlet pressure becomes too high or low because of excessive or insufficient air flow at the minimum or maximum blower speed, the number of blowers running is reduced or increased by the on/off controller.

2.3 SUMO

Dynamic simulation models of biological processes in the wastewater treatment domain have been widely used since 1987 when the mathematical modelling task group of International Water Association (IWA formally called as IAWPRC) proposed a general base framework of activated sludge process model called ASM1 (Henze *et al.*, 1987). The wastewater treatment process is inherently dynamic as the influent to the treatment system is always fluctuating and the biological consortia, usually called as activated sludge in this domain, is adapting to changing conditions. Therefore, the process model needs to be dynamic. This model has been expanded and enhanced by the task group as ASM2, ASM2d and ASM3 (Henze *et al.*, 2000). The first commercial implementation of these model and their enhanced models named GPS (and soon after its successor GPS-X) were put into market around 1989 (Patry and Chapman, 1989).

SUMO is a relatively new platform for wastewater treatment modelling and simulation (Russell, 2019), which is fundamentally based on ASM model concepts, but vastly expands the model scope to cover much more detailed biological reactions, ionic species equilibrium for pH estimation and gas transfer phenomena typically linked with aeration system. The supported unit process models include the conventional suspended growth activated sludge system and other wide variety of biofilm processes including MBBR. It supports both dynamic and steady-state simulations.

Some of the commercial implementations of these models use a tabular model description format called Gujer matrix, introduced by Henze *et al.* (1987) for the explanation of biokinetic reaction models. This format is now widely accepted as one of the standard methods

DOI 10.3384/ecp202017493

of representing biokinetic models by research communities (Rieger *et al.*, 2012). SUMO extends this tabular model representation format. Proprietary model description language called SumoSlang (Kovács *et al.*, 2013) has a highly optimized syntax for the specific needs of water treatment process model descriptions in couple of table formats. Not only biokinetic reactions, but also equilibrium chemistry equations and algebraic declarative models are expressed in a two-dimensional table format in Excel. These models are automatically translated into C++ programs by the custom lexical analyzer and parser which use Excel model descriptions as source codes, and these C++ programs are further linked to customized initial steady-state solver and DAE solver settings. In the biological treatment domain, many modelers find this Gujer-matrix type format easy to understand, which is one of the reasons why they do not use general simulation languages such as Modelica. These features of the model description make SUMO and other wastewater simulators alike unique as domain-oriented simulation system which is quite different from the generic nature of Modelica.

There is a Modelica library (Reichl, 2003) of original models of ASM series, which is a part of the scope of SUMO. However, they have not been maintained to cover the latest biological model enhancements, and it is generally assumed that their model implementations are not easy to understand for modelers in the water treatment industry without the familiar Gujer matrix type tabular model representations. In the author's opinion, this is one of the main reasons why generic simulation languages are not so popular in the water treatment industry, especially in biological treatment engineering fields.

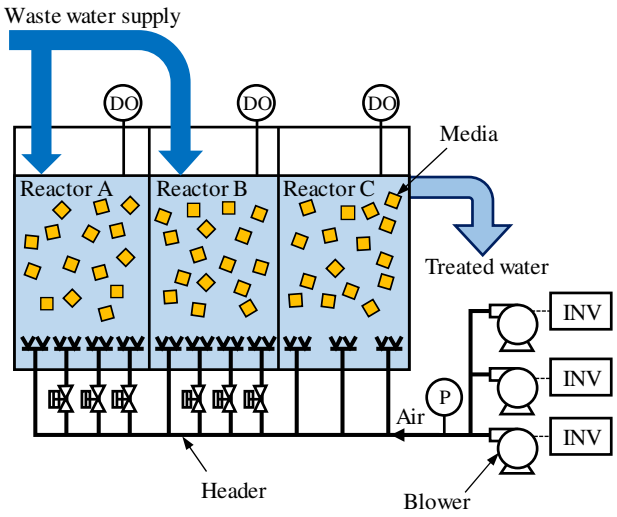


Figure 1. An example of an MBBR wastewater treatment system.

Figure 2 shows the MBBR process modeled by SUMO. This process consists of three reactors in series and each reactor model includes a layered one-dimensional three-compartment model for the biofilm reactions with diffusion phenomena and a fully mixed balk water model for the outside of the biofilm. The number of state variables for this biological system was 1236. The detail of the base kinetic model was described by SumoSlang (Kovács *et al.*, 2013). This model was exported as a DLL file. SUMO has C-interface to SUMO solver which allows Modelica to control SUMO simulation through the External Object.

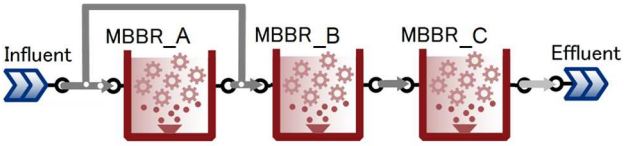


Figure 2. MBBR reactors modeled by SUMO.

3 Framework for Simulator Coupling

3.1 Overview

Modelica and SUMO simulate the mechanical component dynamics and biochemical process dynamics respectively, as shown in Figure 3. Modelica sets the load of wastewater and calculates the air flow rate to each reactor based on valve manipulation strategy for keeping DO constant combined with header pipe pressure control as previously described. SUMO estimates the water quality including DO based on given load and air flow rate provided by the Modelica model. Both simulator solvers use variable time step algorithms. Figure 4 shows a flow chart of the coupled simulation. The quasi-dynamic coupling method is used; each solver exchanges their simulation results before the beginning of each simulation step and uses the results of the previous time step. Therefore, the received data change discretely. A communication step size between Modelica and SUMO was determined as sufficiently shorter interval than the DO change response time to air

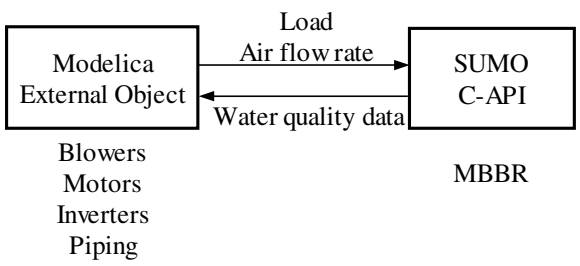


Figure 3. Simulator coupling.

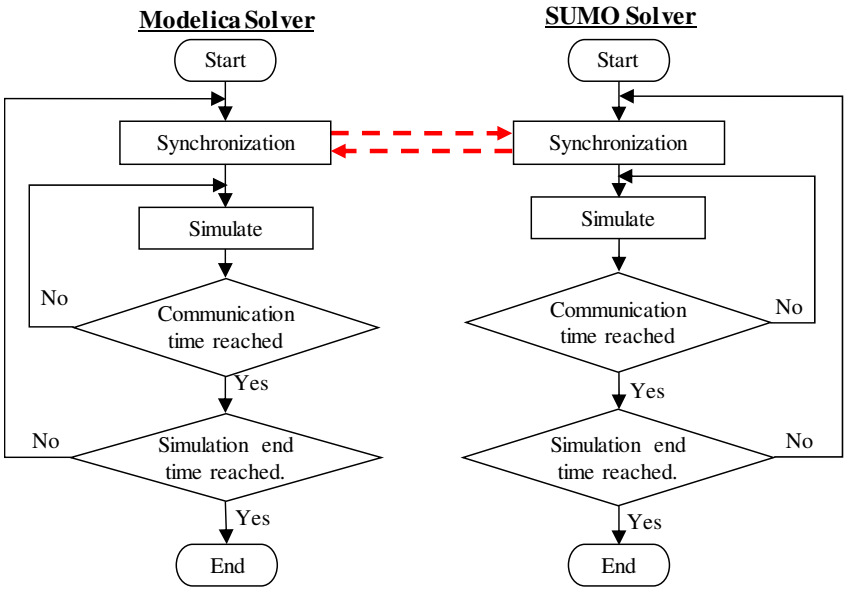


Figure 4. Flow chart of coupled simulation between Modelica and SUMO.

flow changes. Assuming oxygen transfer rate and DO sensor response time around several tens of seconds, 5 second communication interval is assumed to be sufficiently short. In biological wastewater treatment units, water quality other than DO has much longer time constants compared to this communication step size. It should be more than few minutes taking into account the hydraulic retention time calculated from rector size and influent flow rate which is typically more than hours. From this reason biological system can be assumed quite stable between these communication steps. Aeration system has much more shorter time constant than this communication interval in nature typically blower pressure control response time should be sub second order, which means mechanical models in Modelica need to take much shorter integration steps compared to SUMO biological process model in between the communication steps.

3.2 Implementation in Modelica

The coupling simulation framework described in the previous section was implemented as a Modelica model by using External Object. SUMO solver is started in a constructor, the process is initialized at the initialization process of Modelica, and the process of SUMO is destroyed by a destructor at the end of simulation.

Before the coupled simulation study is conducted, to commence the coupled simulation study under stable condition of biofilm states in MBBR which requires months of continuous operation, steady state of MBBR model is calculated by SUMO only unlinked with Modelica model exposing one year of constant wastewater load and aeration condition. After that, coupled simulation study with Modelica is conducted.

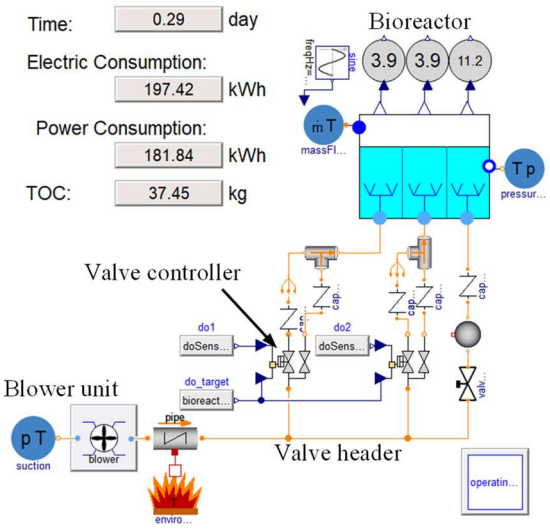


Figure 5. Biological wastewater treatment system simulator coupling implementation on Modelica.

The results of the biofilm initialization process can be exported from SUMO as an XML file which can be reloaded at the initialization step of coupled simulation study to shorten the initialization time.

4 Simulation Results

4.1 System Model

Figure 5 shows an MBBR wastewater treatment system modeled using Modelica. The components of the Modelon Base Library (MBL), which is a commercial library developed by Modelon, are used for the blower and piping components. The performance

DOI 10.3384/ecp202017493

curves of the blowers, motors, and inverters are implemented in the corresponding component models to calculate a required shaft power and electric power consumption.

The TOC (total organic carbon) concentration of the influent wastewater is set in the water flow source by using the media ExtraProperty. The TOC represents the total concentration of the organic contaminants in the influents, and the SUMO model requires more detailed concentration profile information of these contaminants. Therefore, the fractional ratios of these contaminants in the TOC are set as the model parameters for the influent water. These ratio parameters are also specified by Modelica model and passed to the SUMO solver. Bioreactor model contains the External Oject to handle the SUMO solver.

This system model contains two on/off controllers for automatic valves, controller for aeration header pressure controller, and controller to manage the number of blowers on operation. Table 1 and Table 2 show the actions of the controllers. The valve controllers control the air flow rate to keep DO concentration of the reactor A and B based on the intermittent sampling of DO concentration. The blower speed is controlled to keep the header pressure constant. The air flow is estimated based on the Cv value settings of manual valves automatic on/off valves manipulated by control logic. These valves are connected in parallel, as shown in Figure 1. The number of blowers in operation is controlled based on the intermittent sampling of time-averaged blower speed. If the averaged blower speed is low, that is the blower load is low, and if multiple blowers are running, the controller stops one blower.

4.2 Case Study

The system parameters of the target system are shown in Table 3. The TOC (total organic carbon) concentration of the influent wastewater was assumed to be constant, and flow rate and DO setpoint was assumed to follow sinusoidal pattern with a period of 1 day. Parameter study was conducted to evaluate the effect of different open/close action interval for the automatic valves.

Figure 6 shows the DO setpoint assumed, DO concentration estimated by SUMO and number of open valves of each reactor determined by controller. If the action interval is set to 1 minute or 2 minutes, DO concentrations were kept close to its target values. If the action interval is set to 5 minutes, DO concentration deviation from its target value became significant compared to other shorter interval cases. The action interval of 5 minutes was judged to be too long compared to the time constant of oxygen transfer process in the biological process.

Figure 7 shows TOC concentration of each reactor. The TOC concentration in the reactor A and B tended to increase when the influent load is increased. On the other hand, the TOC concentration of the last reactor (C) is almost stable. There was no significant difference in these TOC concentration values between different automatic valve control intervals. The maximum TOC concentration values of effluent are 1.51 mgC/L, 1,46 mgC/L, and 1.52 mgC/L for the interval of 1 min., 2.min and 5 min., respectively. These values are essentially same as the values determined from the minimum achievable concentration assumed in the influent characteristic assumptions and the leakage of unbiodegradable matter from biomass through its decay process.

Table 1. Action of valve controller.

DO concentration	*Controller Action*
> Target value	Closes one opened valve. If all on/off valves are open, do nothing.
< Target value	Opens one closed valve. If all on/off valves are closed, do nothing.

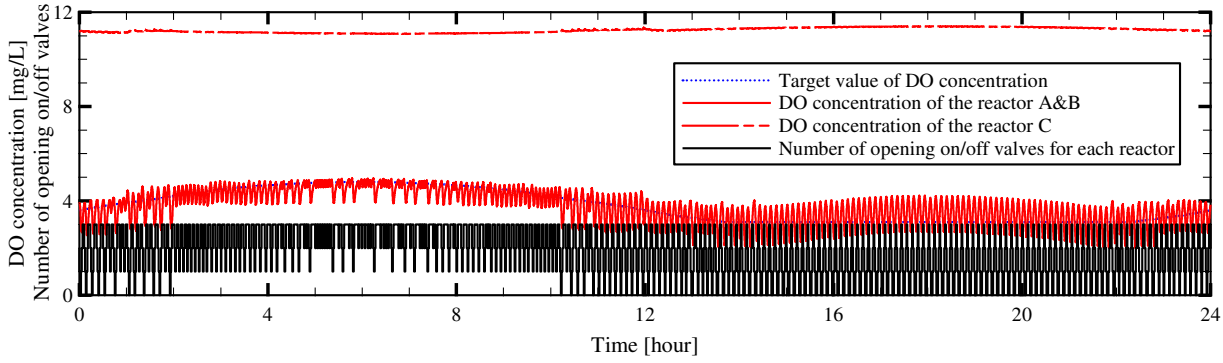
Table 2. Action of blower unit controller.

Averaged blower speed (60min)	*Controller Action*
< Low threshold	Stops one stopped blower.
> High threshold	Starts one running blower.

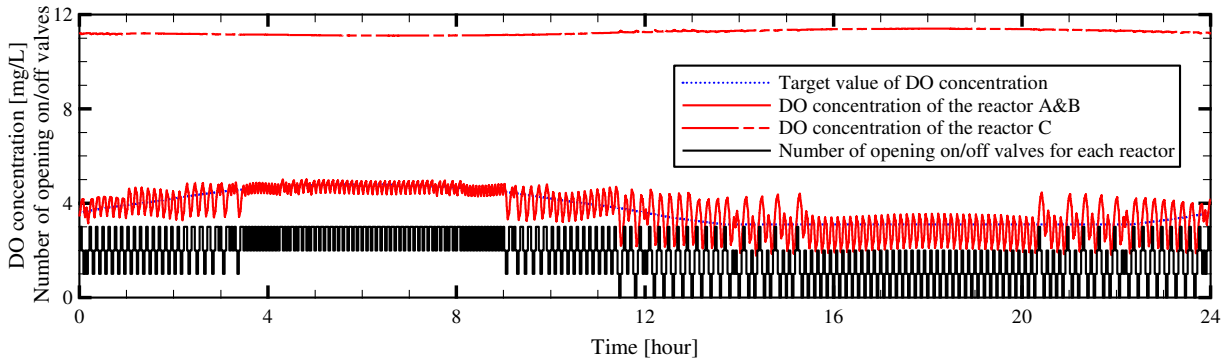
Table 3. Conditions for the case study.

Model Parameters	*Conditions*
Influent wastewater TOC concentration	150 mgC/L
Raw water flow rate	Max: 8.54 m^3/h Min: 2.56 m^3/h
Number of reactors	3
Volume of each tank	400 m^3
Number of blowers	3
Number of on/off valves	3
Header pressure	1.519 barA
Sampling interval of blower unit controller	1 hour
Open/close action interval for the automatic valves	1 min./2 min./5 min.

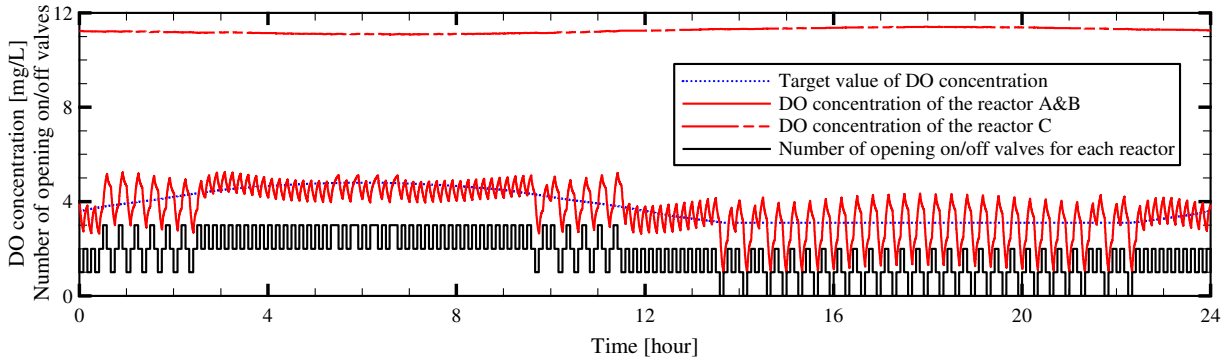
The last reactor could be judged to be effective for the final polishing task as its designer expected. This also means aeration control setting does not have any effect on the effluent quality as far as we tested. We can focus on the energy efficiency and mechanical action difference in these cases. These effluent quality results



(a) 1 minute.



(b) 2 minute.



(c) 5 minute.

Figure 6. DO concentration and number of open valves of each reactor.

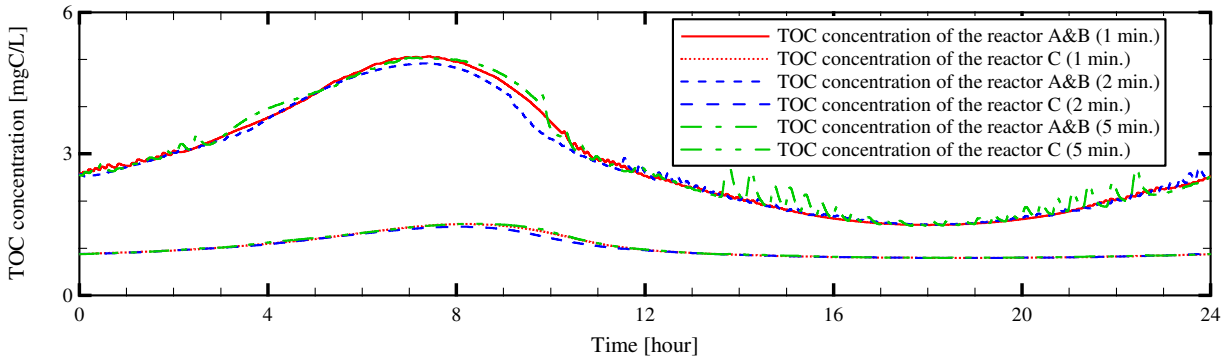


Figure 7. TOC concentration of each reactor.

DOI 10.3384/ecp202017493

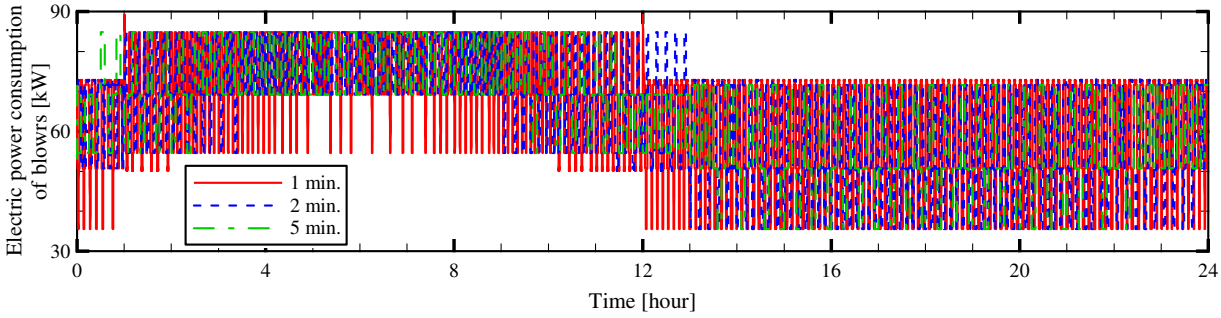


Figure 8. Electric power consumption of blowers.

of the simulation were obtained from a theoretical understanding of the process, and they were consistent with the observed TOC values of the effluent under the sufficient aeration intensity, ranging around 3 to 5 mgC/L.

Figure 8 shows the electric power consumptions of the system. The power consumptions are almost proportional to the number of open valves, or air flow rate. It is confirmed that the necessary air flow is almost determined by the load, and the effect of the valve on/off interval was negligible.

Total electric energy consumption and the number of times the valve opened are shown in Table 4. It is estimated that the total energy consumption is almost determined by the total amount of the load as is the case with the instantaneous value, and the electric energy consumption values for all cases are almost same. From the perspective of maintenance costs, a smaller number of valve openings is preferable in order to avoid unnecessary wear of the installed valves. In this aspect, 5 minutes interval is obviously preferred for the expected longer valve life. The simulator coupling could provide the designers with the information necessary to check expected water quality, power consumption, and equipment maintenance interval.

Table 4. Simulation results.

	1min.	*2 min.*	*5 min.*
Electric energy Consumption	1557 kWh	1554 kWh	1542 kWh
Number of times the valve opened	554	336	141

5 Conclusions

In this report, the coupled simulation for the system analysis of wastewater treatment is demonstrated where water treatment process is simulated by domain-oriented simulator and the mechanical components related to aeration is simulated using Modelica via Dymola. These solvers are successfully linked through the C interface of SUMO and the Modelica External Object. The system level effect implication of the different aeration control scenarios can be investigated using the developed system model. Specifically, effect on the MBBR biological treatment process performance, the electric energy consumption of blowers and cycle frequency on the automatic valves can be evaluated. That is, it could provide the necessary information for water treatment performance evaluation, the operation cost and maintenance cost planning.

There are many domain-specific simulators which does not rely on Modelica and FMI technologies. But, Modelica tools can still be used under the simulator coupling framework developed if external simulators can provide a low-level manipulation interface such as a C language API. The authors expect the practical usefulness of the coupled simulation of domain-specific simulators and Modelica models, as shown in this paper, will be a driving force to promote the utilization of the wide range of existing models based on Modelica hopefully through the application of more convenient model integration methods such as FMI to the domain.

References

M. Henze, W. Gujer, T. Mino and M. v. Loosdrecht, 2000. *Activated sludge models ASM1, ASM2, ASM2d and ASM3*. IWA Publishing.

M. Henze, L. G. Jr., W. Gujer, G. v. R. Marais and T. Matsuo, 1987. *Activated Sludge Model No.1*. IAWPRC Publishing.

R. Kovács, I. Takács and J. D. Benke. Facilitating biofilm reactor modelling with an easy-to-use spreadsheet-based tool designed for process engineers. *IWA Biofilm Conference*, 2013.

G. G. Patry and D. T. Chapman, 1989. *Dynamic Modeling and Expert Systems in Wastewater Engineering*. Lewis Publishers, Inc.

G. Reichl. WasteWater a Library for Modelling and Simulation of Wastewater Treatment Plants in Modelica. *3rd International Modelica Conference*: 171-176, 2003.

L. Rieger, S. Gillot, G. Langergraber, T. Ohtsuki, A. Shaw, I. Tak´cs and S. Winkler, 2012. *Guidelines for Using Activated Sludge Models*. Scientific and Technical Report Series, 11. IWA Publishing.

D. L. Russell. *Practical Wastewater Treatment, Second Edition*. WILEY, pp. 264-265, 2019.

L. M. Sidek, H. A. Mohiyaden, G. Hayder, A. HusseinS, H. Basri, A. F. Sabri and M. N. Noh. Application of Moving Bed Biofilm Reactor (MBBR) and Integrated Fixed Activated Sludge (IFAS) for Biological River Water Purification System: A Short Review. *IOP Conference Series Earth and Environmental Science*, 2016.

M. V. Sperling and C. A. d. L. Chernicharo, 2005. *Biological Wastewater Treatment in Warm Climate Regions Volume I*. IWA Publishing.

 DOI 10.3384/ecp202017493

Power and Temperature Prediction for Computer System Power Optimization

Koji Nishi[1] Shota Takada[1] Takamichi Kaneda[1]

[1]Department of Innovative Engineering, Ashikaga University, Japan, nishi.koji@v90.ashitech.ac.jp

Abstract

This paper investigates desktop computer system power optimization with Modelica. Firstly, microprocessor power equation is modeled with voltage and temperature dependency. Secondly, steady state heat transfer path is modeled as a variable thermal resistance with fan power change and is connected to microprocessor power model to discuss optimum working point of fan which minimizes power consumption of the power rail for the microprocessor on the target computer system under CPU-intensive workload condition. After that, transient heat transfer path modeling is explored as a Cauer thermal network creation.

Keywords: Computer system, Microprocessor, Power Optimization

1 Introduction

Computer systems are vital in the modern world. Higher performance is still explored not only for high performance computing systems and server systems but also for consumer computer systems such as desktop computers, notebook computers and slate devices. Also, IoT (Internet of Things) and M2M (Machine-to-Machine) infrastructure has been rapidly developed in a few years and computer systems for these use will be increased rapidly. While, energy consumption needs to be eliminated in the sustainable development of modern society and power consumption by computer systems is one of important subject to be solved.

The latest microprocessors incorporate power management features which control trade-off of computing performance and power consumption. With these features, microprocessor's power efficiency has been improved drastically in the past decade, however, system level power optimization is still on the way because system level power optimization needs to consider power consumption of not only microprocessor itself but also peripherals and power supply circuit for them. And power consumption of the microprocessor is highly dependent on temperature, however, current system-level temperature control methodology is to maintain junction temperature of microprocessor within operational temperature range and not to minimize system-level power consumption.

To minimize system-level power consumption, power modeling methodology with practical accuracy is required. There are many efforts and researches on electrical characteristics on CMOS (Complementary Metal-Oxide-Semiconductor) circuits. And microprocessors incorporate power management features, while system-level power optimization is still on the way. Thus, author investigates microprocessor power equation to conduct both power and thermal control as a part of previous study (Nishi, 2012). In the study, microprocessor power equation with voltage and temperature dependencies is introduced and is employed in three-dimensional heat conduction simulation. However, three-dimensional simulation costs too much from a computational load standpoint and takes time to create accurate thermal simulation model. To enable fast simulation environment, three-dimensional simulation needs to be replaced by another methodology which ensures high speed calculation with practical accuracy. Also, model creation effort should be minimized as much as possible.

As another methodology to conduct thermal simulation with reduced computational load, there are thermal network methodology and model order reduction (MOR). Thermal network methodology has long history for temperature prediction. As for thermal network application to semiconductor devices, there already exist some researches in 1980s. For example, thermal network is utilized for temperature prediction of multi-chip module (Ishizuka and Fukuoka, 1986). Traditional way to create thermal network is to determine thermal resistance and heat capacitance values from dimensions and thermophysical properties of each component along heat transfer paths. In recent years, curve fitting technique to determine thermal resistance and heat capacitance values from measurement data is also sometimes employed. On the other hand, MOR is a technique to reduce computational complexity of simulation model and is sometimes utilized with Modelica. For example, MOR is applied to FEM (Finite Element Method) model and reduced order model is implemented as FMU (Functional Mockup Unit) (Gödecke *et al*, 2012). Also, MOR based technique is proposed for thermal simulation as well (Codecasa *et al*, 2014). MOR is useful if three-dimensional simulation model or dimensions and

thermophysical properties of each component along heat transfer paths are available.

This research aims to establish fast simulation environment for computer system power optimization with Modelica. In this paper, power equation is modeled as a Modelica component. Secondly, steady state heat transfer path is modeled as a Modelica component with a variable thermal resistance which expresses fan speed change of heat sink fan over the microprocessor. Thirdly, they are connected each other, and system-level power optimization is investigated. After that, transient heat transfer path modeling is explored as a Cauer thermal network creation, to establish transient control optimization environment.

2 Target Desktop Computer System

2.1 Hardware Configuration

The target computer system in this paper is a desktop personal computer, Sycom Radiant GZ2650X470A (Fig. 1 (a)). It employs a AMD Ryzen 5 1500X processor, which is fabricated with 14nm process technology and has 4 CPU cores with totally 8-thread execution capability. Its thermal design power (TDP) is 65 W. The base frequency of the microprocessor is 3.5 GHz, however, the microprocessor supports boosted frequency up to 3.7 GHz. Under lower temperature conditions, it runs over 3.5GHz with higher power consumption than TDP. Thus, a large heat sink fan prepared by microprocessor manufacturer is attached over the microprocessor package (Fig. 1 (b)). Table 1 shows major hardware components of the target computer system. The microprocessor consumes more than half of the computer system's power under CPU-intensive workload scenario and this paper focuses on microprocessor power.

2.2 Key Software Components

Key software components of the target computer system are listed in Table 2. The system is based on Ubuntu 16.04 (64-bit). Prime 95 is an application software and is utilized as a CPU-intensive workload in this paper. It processes FFT (Fast Fourier Transformation) continuously. Prime 95 has some test modes and "small FFT" is utilized in this paper. AMD μProf is utilized as a monitoring software tool which can log CPU temperature, estimated power consumption and so on.

(a) System outlook (b) Heat sink fan

Figure 1. Desktop personal computer outlook and heat sink fan for microprocessor.

3 Power Modeling

3.1 Microprocessor Power Equation

3.1.1 Power Equation of CMOS digital circuits

Power consumption of CMOS digital circuits, including the latest microprocessor, is due to electrical charge and discharge to/from load capacitance C_{load} and power loss by leakage current (Hiramoto *et al*, 2009). Therefore, its power consumption can be modeled as the equation below (Kunimine *et al*, 2011):

$$P_{CMOS} = aC_{load}V_{DD}^2 f_{op} + I_{leak}V_{DD} \quad (1)$$

Here, a is activation rate, C_{load} is load capacitance, V_{DD} is power supply voltage, f_{op} is operational frequency and I_{leak} is leakage current. Unfortunately, it is known that C_{load} has voltage dependency (Uchida *et al*, 2001; Uchida *et al*, 2002; Watanabe *et al*, 2007). Also, there are several types of leakage current (Hiramoto *et al*,

Table 1. Major hardware components of target desktop computer system.

Function	*Component*
Microprocessor	AMD Ryzen 5 1500X (65W TDP)
Motherboard	MSI B350 TOMAHAWK
Memory	DDR4-2400 (16GB ×2)
Graphics card	NVIDIA GeForce GT710 (1GB)
HDD	Toshiba DT01ACA100 (1TB)
Power supply	Antec NeoECO NE650C (650W)

Table 2. Major software components of target desktop computer system.

Function	*Component*
Operating system	Ubuntu 16.04 (64-bit)
Software utilized as CPU workload	Prime 95 v29.5 (Linux version 64-bit)
Processor information gathering	AMD μProf v1.2.275.0 (Linux version 64-bit)

DOI
10.3384/ecp2020174101

2009). Many of them have voltage dependency. Moreover, subthreshold current, which is a dominant factor of leakage current has not only voltage dependency but also temperature dependency.

3.1.2 CPU Core Power Equation

For microprocessor CPU cores, V_{DD} needs to be changed when f_{op} is changed to increase computational performance or lower power consumption. Thus, C_{load} is changed when f_{op} is changed. Author investigated the way to express C_{load} for microprocessor silicon die temperature prediction and concluded linear approximation of V_{DD} works well because V_{DD} range of CPU core is not so much large during normal operation as a part of previous study (Nishi, 2012). As for leakage current, subthreshold current is modeled as exponential function of temperature for precise discussion (Amelifard, 2008). However, exponential function is not suitable for high speed electro-thermal simulation, especially for transient simulation, from a computational load standpoint. Also, microprocessor is utilized under temperature range from room ambient to around 100 deg C and it is relatively small range compared to other semiconductor devices such as power semiconductor. Thus, Author investigated the way to express leakage current for microprocessor silicon die temperature prediction and concluded linear approximation of V_{DD} and quadratic approximation of silicon die temperature T work well as a part of previous study (Nishi, 2012). Therefore, power consumption of a CPU core can be expressed as the equation below:

$$P_{CPU} = (d_1 V_{DD} + d_2) V_{DD}^2 f_{op} + (s_1 V_{DD} + s_2)(T^2 + s_3 T + s_4) V_{DD} \quad (2)$$

Here, d_1, d_2 are coefficients of the first term, called as dynamic power term, and s_1 to s_4 are coefficients of the second term, called as static power term, on the right in eq. (2).

3.1.3 Microprocessor Power Equation

Microprocessor has several power rails and sum of their power consumption equals to microprocessor power consumption. Since this paper focuses on CPU intensive workload case only and microprocessor power consumption other than CPU cores is relatively small and can be modeled as constant. Therefore, microprocessor power consumption in the case that all CPU cores run the same workload is

$$P_{Microprocessor} = nP_{CPU} + P_{Other} \quad (3)$$

Here, n is number of CPU cores and P_{Other} is power comsumption by circuits other than CPU cores in microprocessor silicon die. Applying junction temperature T_J, which is reported as Die SoC temperature in AMD µProf, as silicon die temperature for P_{CPU} and assuming P_{other} is constant, the microprocessor power equation is obained finally from eq. (2) and (3),

$$P_{Microprocessor} = (c_1 V_{DD} + c_2) V_{DD}^2 f_{op} + (c_3 V_{DD} + c_4)(T_J^2 + c_5 T_J + c_6) V_{DD} + c_7 \quad (4)$$

Here, c_1 to c_7 are coefficients of microprocessor power consumption. By determining these coefficients, microprocessor power consumption can be calculated.

3.2 Power Consumption of Microprocessor Power Rail

On the latest desktop computer system, EPS12V power rail works as microprocessor dedicated power supply rail from power supply unit to motherboard (Fig. 2). Since EPS12V has power loss by voltage regulators which convert 12V voltage level to several lower voltage levels for the microprocessor, EPS12V is thought as the key power rail for the target desktop computer system under CPU-intensive workload. Thus, not only microprocessor power but also EPS12V power are collected and modeled in this paper. Since EPS12V power is sum of microprocessor power and power loss of DC-DC conversion by voltage regulators of microprocessor power rails, it should take the same form as eq. (4) with different coefficient values from microprocessor.

3.3 Power Modeling of The Target Desktop Computer System

Microprocessor and EPS12V power are sensitive to application workload and usual application software changes activation rate in eq. (1) and eq. (2) dynamically. To collect consistent and accurate voltage and

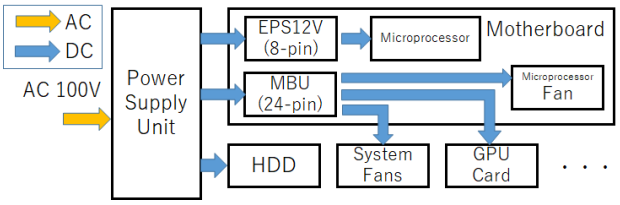


Figure 2. Power supply structure of target desktop computer system.

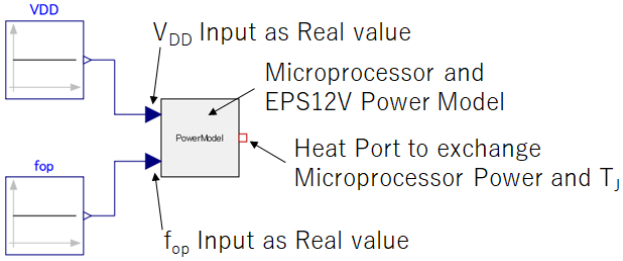


Figure 3. Microprocessor and EPS12V power model with V_{DD} and f_{op} inputs.

temperature dependent power data, constant workload should be added. In this paper, M172031, one of small FFT execution in "Torture Test" of Prime 95, is utilized and is added to all CPU threads. Totally eight M172031 modules run on target desktop computer system during mesurement.

AMD μProf tool is utilized to read junction temperature and estimated microprocessor power. EPS12V power rail current and voltage are measured by current probe and data logger, respectively.

OpenModelica v1.13.2 is utilized to create Modelica models and conduct simulation in this paper. Figure 3 shows a created Modelica component model which calculates microprocessor and EPS12V power and T_J. Coefficients are obtained as parameters and only V_{DD} and f_{op} are externally obtained via connectors. A heat port is prepared to exchange microprocessor power and T_J with external component. Two Eq. (4), one for microprocessor and another for EPS12V, are expressed as Modelica equations. Figure 4 shows Modelica simulation result in the case that an ambient temperature component is connected to heat port of the power model and ambient temperature value is sweeped from 40 deg C to 70 deg C.

4 Heat Transfer Path Modeling

To obtain power consumption under real environment, junction temperature also needs to be calculated. Thus, heat transfer path is modeled as the thermal resistance from junction to ambient, $R_{th,\,JA}$, which can be utilized in steady state simulation (Nishi *et al*, 2019).

$$R_{th,JA} = \frac{T_J - T_A}{P_{Microprocessor}} \tag{5}$$

Here, T_A is ambient temperature.

Measurement is conducted with the same condition described in section 3.3 and thermal resistance is modeled as a function of fan power.

Figure 5 shows a created Modelica component model which calculates $R_{th,\,JA}$ from fan power input value. At this time, $R_{th,\,JA}$ is modeled as polynomial which is obtained by curve fitting of measurement result. Figure 6 shows thermal resistance variation by fan power.

5 Power Optimization Under Steady State

To obtain both microprocessor power and junction temperature, the equation below needs to be solved:

$$T_J = R_{th,JA} \cdot P_{Microprocessor} + T_A \tag{6}$$

In Modelica, eq. (6) is solved by connecting created Modelica component models in Fig. 3 and Fig. 5 as shown in Fig. 7.

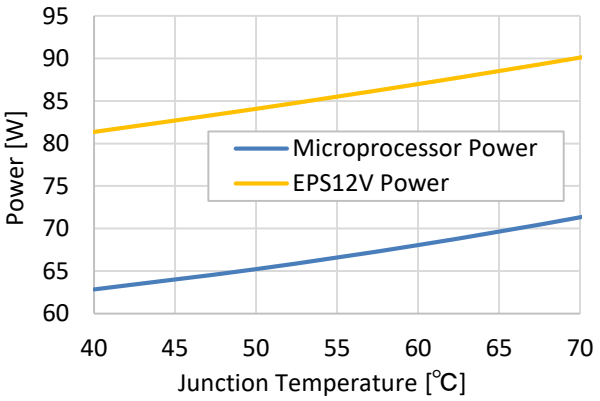


Figure 4. Relationship between junction temperature and power consumption.

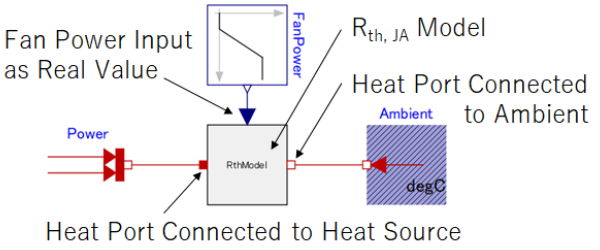


Figure 5. Thermal resistance model with fan power input and connections with heat source and ambient.

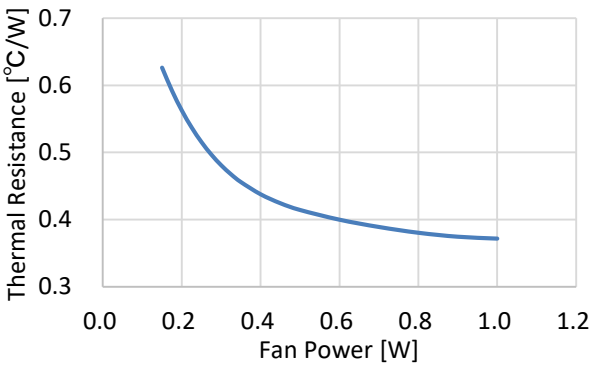


Figure 6. Relationship between fan power and thermal resistance from junction to ambient.

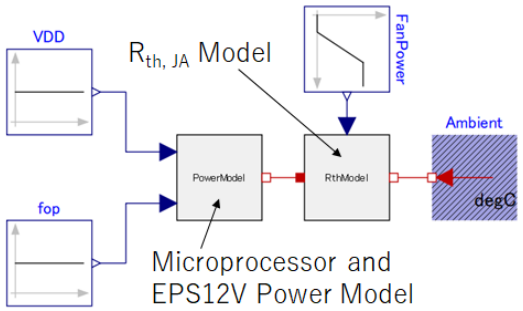


Figure 7. Modelica model connection to calculate both power and junction temperature.

DOI
10.3384/ecp2020174101

Figure 8 shows junction temperature variation by fan power, changing ambient temperature. As fan power becomes higher, junction temperature becomes lower, which results in microprocessor power decrease. Figure 9 shows relationship between power consumption, junction temperature and fan power under 30 deg C ambient as an example. Not only microprocessor power but also EPS12V power decrease monotonically by fan power increase.

EPS12V power is dominant and occupies more than half of total power which PSU (Power Supply Unit) provides. To consider system-level power optimization, not only EPS12V power but also fan power should be taken into account. Thus, sum of EPS12V power and fan power (hereafter, target power) is checked as shown in Fig. 10. Fan power is relatively small, however, target power has the minimum value. Figure 11 shows derivative of target power. Target power becomes minimum when its derivative becomes zero. Optimum fan power which shows minimum target power is ~0.84 W at T_A = 10 deg C, ~0.93 W at T_A = 50 deg C, respectively. This means an optimum working point moves by ambient condition. Since application workload with usual application software varies time by time, fan needs to be controlled time by time. Optimum working point under lower workload scenario can be obtained by applying the same methodology.

6 Transient Simulation

Under steady state condition, optimum fan power is obtained easily with a few Modelica components. Instead, fan control simulation is required to obtain optimum target power under transient state condition because temperature doesn't change immediately and temperature variation is delayed by the effect of heat capacitance along heat transfer paths. Thus, heat transfer path modeling should be carefully for transient simulation and is discussed in this chapter.

Transient heat transfer path can be obtained by extending thermal resistance to RC network. There are two RC network topologies; Cauer and Foster. Cauer RC thermal network is employed in this paper (Fig. 12). To determine R_{th} and C_{th} values, measurement is conducted in the case that microprocessor starts to execute eight M172031 modules from idle state on the target computer system and fitting is executed to minimize the equation:

$$\boldsymbol{eval} = \sqrt{\frac{1}{t}\int_0^t \left(T_{J,measured} - T_{J,estimated}\right)^2 dt} \qquad (7)$$

Here, t is the total execution time (250 sec), $T_{J,\,measured}$ is junction temperature obtained from measurement and $T_{J,\,estimated}$ is junction temperature obtained from Cauer

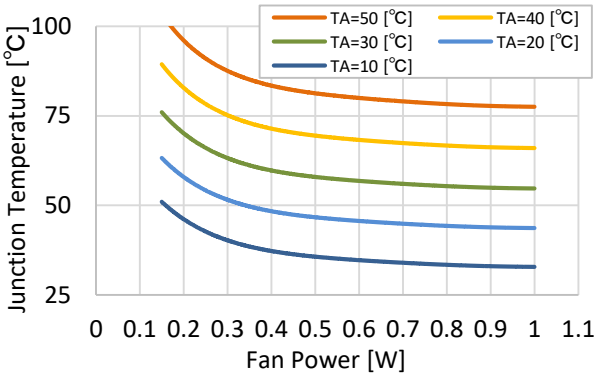


Figure 8. Relationship between microprocessor's junction temperature and fan power.

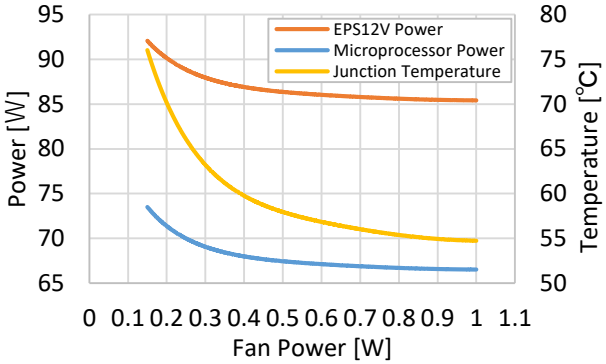


Figure 9. Relationship between microprocessor related power consumption and fan power under 30 deg C ambient.

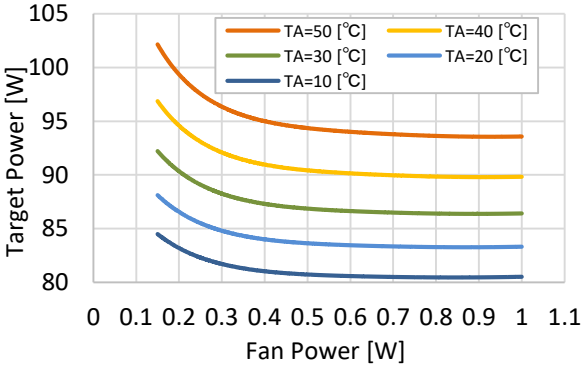


Figure 10. Relationship between sum of "EPS12V power and fan power" and fan power.

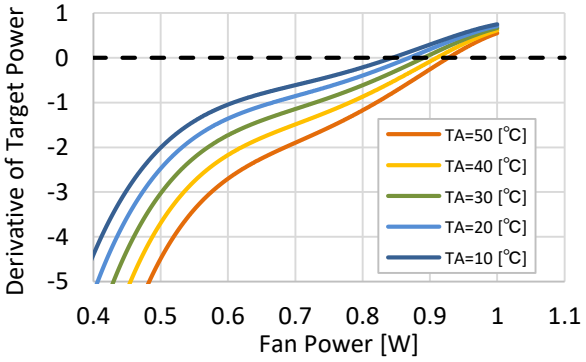


Figure 11. Derivative of sum of "EPS12V power and fan power" curve.

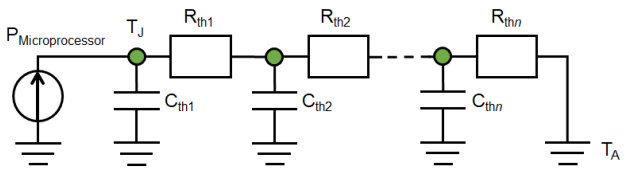


Figure 12. Cauer RC thermal network.

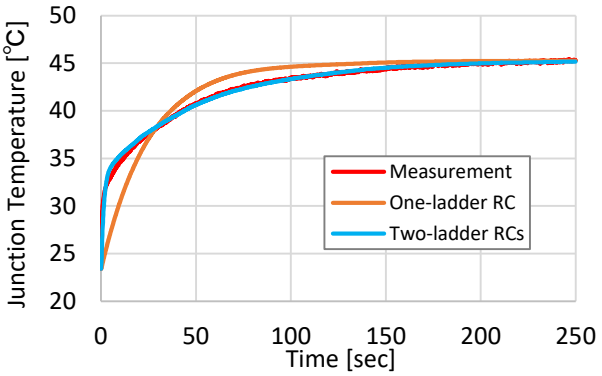


Figure 13. Transient curve of microprocessor junction temperature.

RC thermal network.

Figure 13 shows fitting results with one-ladder and two-ladder Cauer RC network utilizing OpenModelica. One-ladder RC network shows obvious difference from measurement, while two-ladder RC network fits well compared to one-ladder RC network. By adding more ladders, estimated junction temperature will fit more with measurement. Transient fan control optimization with Cauer RC network will be the future work of this research.

7 Conclusions

This paper investigates power optimization of microprocessor dedicated power rail, EPS12V, targeting a desktop computer system under a constant CPU-intensive workload condition. Microprocessor and EPS12V power models and heat transfer path model as a variable thermal resistance by fan power are created as Modelica components based on measurement data. By connecting them each other, and optimum fan control condition is explored. It is found that an optimum fan working point exists to minimize sum of EPS12V and fan power. By extending a variable thermal resistance to Cauer RC thermal network, transient heat transfer path is modeled. Transient fan control optimization with Cauer RC network will be the future work of this research.

References

AMD µProf,
URL https://developer.amd.com/amd-uprof/ (Accessed: 2020-02-10)

AMD Ryzen™ 5 1500X Processor,
URL https://www.amd.com/en/products/cpu/amd-ryzen-5-1500x (Accessed: 2020-02-10)

B. Amelifard, F. Fallah and M. Pedram. Leakage minimization of SRAM cells in a dual-Vt and dual-Tox technology, *IEEE Trans. on VLSI Systems* 16(7): 851-860, July 2008.

L. Codecasa, V. d'Alessandro, A. Magnani, N. Rinaldi, P. J. Zampardi. Fast novel thermal analysis simulation tool for integrated circuits (FANTASTIC), Proceedings of THERMINIC 2014, September 2014. doi: 10.1109/THERMINIC.2014.6972507

A. Gödecke, M. Mühlbauer, J. Nieveler, I. Vittorias, T. Vontz. FEM models in System Simulations using Model Order Reduction and Functional Mockup Interface, *Proceedings of the 9th International Modelica Conference*: 565-570, September 2012. doi: 10.3384/ecp12076565

T. Hiramoto, K. Uchida, N. Sugii and K. Takeuchi. Integrated Nanodevice, Maruzen (ISBN 978-4-621-08208-9): 131, 143-147, November 2009 (In Japanese).

M. Ishizuka and Y. Fukuoka. Transient Temperature Rise for Multi-Chip Packages, *Transactions of the Japan Society of Mechanical Engineers. B* 52(476): 1772-1776, April 1986. doi: 10.1299/kikaib.52.1772 (In Japanese)

N. Kunimine, *et al*. The Newest Thermal Design Technique and Electronics Cooling Technology, CMC Publishing (ISBN 978-4-7813-0510-3): 60-61, December 2011. (In Japanese)

K. Nishi. Transient Heat Conduction Simulation around Microprocessor Die, *Thermal Science & Engineering* 20(2):27–34, April 2012. doi: 10.11368/tse.20.27 (In Japanese)

K. Nishi and T. Kaneda. Microprocessor Electrical-Thermal Co-Simulation Model Creation for Low Power Conservation of Computer Systems, *OpenCAE Symposium 2019*, B-3, December 2019. (In Japanese)

OpenModelica,
URL https://www.openmodelica.org (Accessed: 2020-02-10)

Prime 95,
URL https://www.mersenne.org/download/#stresstest (Accessed: 2020-02-10)

K. Uchida, J. Koga, R. Ohba, T. Numata, and S. Takagi. Experimental Evidences of Quantum-Mechanical Effects on Low-field Mobility, Gate-channel Capacitance, and Threshold Voltage of Ultrathin Body SOI MOSFETs, *Technical Digest of International Electron Devices Meeting (IEDM)*: 633-636, December 2001.

K. Uchida, H. Watanabe, A. Kinoshita, J. Koga, T. Numata, and S. Takagi. Experimental Study on Carrier Transport Mechanism in Ultrathin-body SOI n- and p-MOSFETs with SOI Thickness less than 5 nm, *Technical Digest of International Electron Devices Meeting (IEDM)*: 47-50, December 2002.

H. Watanabe, K. Uchida, and A. Kinoshita. Numerical Study of C-V Characteristics of Double-Gate Ultrathin SOI MOSFETs, *IEEE Transactions of Electron Devices* 54(1): 52-58, January 2007.

DOI
10.3384/ecp2020174

A Health Monitoring Study of Multiple-Unit Train Braking System using Sample Identification Approach

Bo Wang[1] Yang Ji[1] Bohui Liu[1] Feng Gao[2] Weijun Yang[2] Dunwen Gan[2]

[1]Shenyang Yuanda Simtek Co., Ltd., China, {bo.wang,yang.ji,bohui.liu}@cnydsimtek.com
[2]Beijing Zongheng Electro-Mechanical Technology Development Co. , China,
{gaofeng,yangweijun,gandunwen}@zemt.cn

Abstract

The development of the Chinese high-speed railway has experienced considerable dynamics in recent years. Further development of the trains depends to a large extent on the optimization of the key subsystems and monitoring of reliability and safety. In this paper the sample identification based health monitoring method for fault diagnosis and health monitoring by use of a Modelica based model of the train system is addressed. In this study the proposed method sovled the issue of generating enough fault samples for health monitoring and be validated by the train operation realtime data. The paper demonstrates the applicability of the method and modeling concept by means of diagnosis and monitoring of the state of health of the braking system.

Keywords: high speed train, braking system, health monitoring

1 Introduction

1.1 Development of Chinese railway and recent activities in high speed train development

The Chinese high speed railway (HSR) was first introduced in 2007 designed for the speeds of 250-350 km/h. In 2008, the world's first high speed rail with a designed speed of 380 km/h between Beijing and Shanghai was in operation. In mid-2018, the HSR has extended to 30 provincial-level administrative divisions and reached 27000 km in total length **(author?)** (1). Since 2008, China has developed CRH trains (CRH1, CRH2, CRH3 and CRH5) with a maximum velocity of 300-350 km/h. In 2010, a new generation of CRH trains with a top speed of 380km/h were developed and entered service on the Shanghai-Hangzhou High Speed Railway. The newest trains CR400AF and CR400BF called "Fuxing Hao" in figure 1 can reach the top speed of 400km/h narrowing the commuting time from 6 hours to 4.5 hours between Beijing and Shanghai.

1.2 Motivation

As the fast development of the Chinese high-speed trains, the reliability and safety of the high-speed trains are critical for multiple-unit train (EMU) further development. However, there is no efficient fault detection and health monitoring methods which are really applicable in EMU trains due to the lack of failure data samples. And the high cost and difficulty of the experiment test of failure makes the health monitoring of EMU more difficult to achieve. In this paper, a modelica-based model is developed to generate fault samples for health monitoring.

Figure 1. Fuxing Hao

1.3 Overview of the paper

As depicted in figure 2, the Chinese high-speed train consists of two parts: plant system containing traction system, braking system, auxiliary system and so on; and control system including network system, safety detection system. etc. In this paper, the braking system is used for demonstrating the sample identification method. First, we give a short background of the Chinese railway development and the motivation for the new health monitoring method proposed in this paper. Then the failure of the braking system of EMU was analyzed in the motivational example. Afterwards, the theroy of the sample identification method was explained in the technical background of fault diagnosis. Besides the theory of the proposed method, a demonstration of fault sample generation and health monitoring was also dicussed to verify the sample identification method, explicitly. In the presented study, a braking system with braking control unit (BCU), electromagnetic valve, reverse valve, double check valve and distribution valve were implemented with Modelica in Dymola. The health monitoring for the pressure and delay time of the reverse valve in the relay valve has been performed to illustrate the proposed approach.

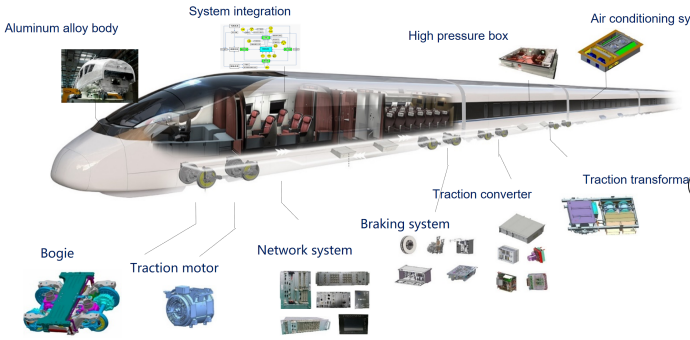


Figure 2. The high speed train and the components of subsystems

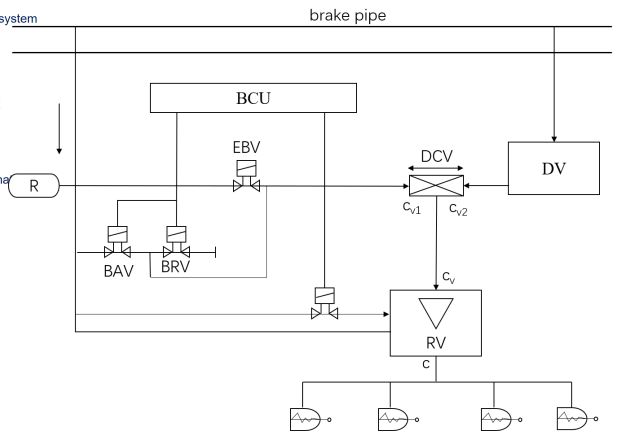


Figure 3. Control system of braking system

2 Motivational Example

2.1 Introduction of the EMU braking system

Braking is divided into two forms: dynamic braking and friction braking. Dynamic braking transforms the kinetic energy into electrical energy by changing the traction motor into a generator in braking. Disk braking is the most widely used in the friction braking (2). The EMU braking is achieved through the combination of electric braking and pneumatic braking, because the kinetic energy is too large for electric braking to consume in the limited time. The braking system can realize multiple functions such as service braking, emergency braking, urgency braking and so on. This paper takes focus on the urgency braking in which only pneumatic braking is applied to the train. The EMU braking system consists of three subsystems: braking control system, air supply system and braking devices. For the pneumatic braking, control system is the critical part that directly influence the braking performance of the train. As shown in figure 3, the braking control system consists of braking control unit (BCU), electromagnetic valve (BAV, BRV and EBV), reverse valve (RV), double check valve (DCV) and distribution valve (DV). The electromagnetic valve activates the electro-pneumatic braking and the distribution valve activates the pneumatic braking. The electromagnetic valve and distribution valve both produce the pre-controlled pressure C_v, the double check valve can ensure the transformation from C_v to actual braking force. The braking cylinder pressure C was adapted by reverse valve according to the C_v value and accommodated the air volume of cylinder.

2.2 Description of the failure to be analyzed

In the emergency braking mode, a failure of changeover on high and low speed occurred occasionally, resulting in a low braking force during the braking, thereby prolonging the braking distance. Through the preliminary analysis of the fault performance and braking system principle, it is considered that relay valve of the braking control module causes the fault, and the reason is that the switching resistance of the relay valve is too high. In the braking control system, the relay valve (shown in figure 4)

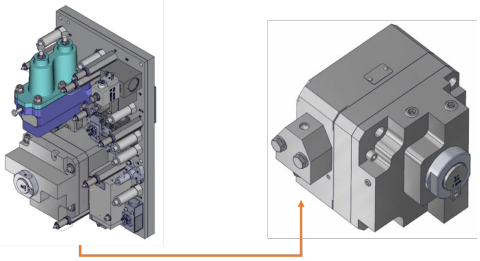


Figure 4. Relay valve

outputs the braking cylinder pressure complying with the requirements of the brake command by high-low speed changeover signal. After further analyzing the working principle of the relay valve, the cause of the fault is located in the reverse valve of the relay valve responsible for controlling the switching of the high/low speed braking force, as shown in figure 5. The reverse valve works as the following principle: when the train is applied brake at high running speed, compressed air pushes the changeover piston to right side to charge air into relay valve's pressure adjusting chamber, and relay valve outputs a lower braking pressure, which means lower braking deceleration of the train. While the running speed is decreased, compressed air in left side's chamber of changeover piston is cut off and vented, and the piston is pushed by return spring's force so that compressed air in pressure adjusting chamber of relay valve is vented, and control the relay valve to output a higher braking pressure, and of course, higher braking deceleration to the train. Failure

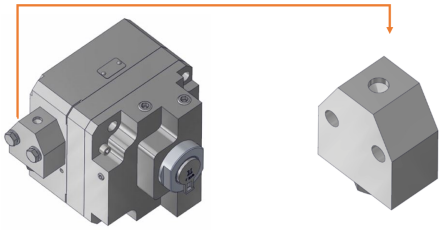


Figure 5. Reverse Valve

DOI 10.3384/ecp2020174109

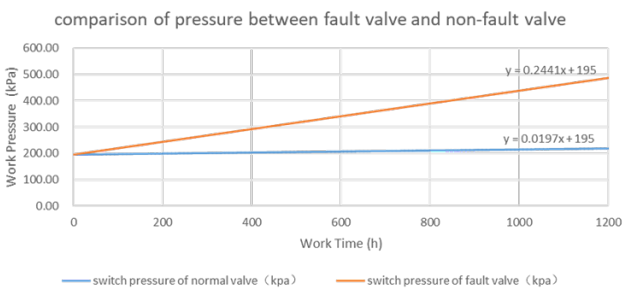


Figure 6. Pressure of fault valve and non-fault valve

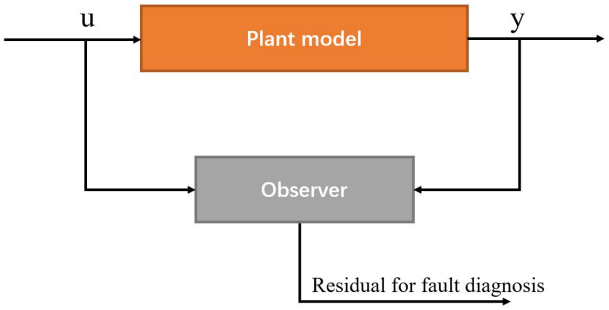


Figure 7. Schematic of model-based fault diagnosis

phenomenon indicates that the piston can not be restorable by right side spring when the piston's left side pressure is vented. The reasons include: the excessive piston movement resistance; spring failure; small spring force. Because the failure is accidental, the piston can be finally restorable by right side spring in most cases, which indicates that this phenomenon is not caused by spring failure. So it should be caused by excessive resistance of valve stem. Finally, it was confirmed by experiments that the compatibility of the piston's rubber articles and lubricating grease was poor, resulting in excessive expansion of the outer dimensions, which further increased the friction force and verified the previous analysis. It can be seen from the experimental results that as the number of use time changes, the pressure required for switching gradually increases, and the increase amount in the fault valve is significantly higher than that of the normal valve, as shown in figure 6.

3 Technical background of fault diagnosis

3.1 Traditional methods

A fault is defined as an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition (3). In order to improve the reliability and safety of systems, fault diagnosis is introduced to monitor, locate and identify the faults. Traditional model-based method includes unknown-input observers approach (4), parity relation approach (5), stable factorization approach (6), Kalman-filter-based residuals (7), parameter estimation method (8), distributed fault diagnosis filtering method (9), etc. The observer based method is widely used in fault diagnosis and health monitoring as shown in figure 7.

3.2 Sample identification based approach

This study addressed a new health monitoring method based on the sample identification. First step is to build the system models. After analyzing the reasons for the failures, a system model with no faults was built including the components in which failure occurred and other subsystems concerned in the fault diagnosis. Concerning the high accuracy demand of the fault diagnosis and the complexity of the plant model, the modeling tool should be able to cover multiple physical domains and realize multi-level modeling with fast simulation and promised accuracy. Modelica is a technology modeling the dynamic behavior of technical systems consisting of components from, e.g., mechanical, electrical, thermal, hydraulic, pneumatic, fluid, control and other domains in a convenient way. Dymola is a tool based on Modelica with fast simulation and high accuracy which is used in this study. The model should be simplified as much as possible without decreasing the accuracy of fault diagnosis. And the healthy models should be validated for ensuring the accuracy of fault samples. At the same time, the corresponding fault models should be developed and be able to substitute the non-fault models in the healthy system models. Then, by inject the fault models into the healthy system models, the fault samples can be generated. As modelica can realize the multi-level modeling conveniently, it's very easy to change the healthy models by fault models. Next, distract the character value of the fault models which indicating the health degrees of the system concerned. In this study, the character is the delay time of braking pressure and the pressure of the reverse valve, the delay time defines the healthy degrees (0~1) and pressure determines if the train is healthy (1) or failure (0). Finally, the fault samples were stored in the health monitoring server for computing the health degrees of the train based on the realtime detected data during the operation of the train. The algorithm for health monitoring is to compare the realtime data with the fault samples and determines the healthy degrees of the train or related subsystems.

The relationship of health degree with braking pressure P and delay time T can be generated by the considerable samples, as shown in equation 1.

$$H = g(T) \tag{1}$$

The braking signals and health degrees can be detected in real time, and analyzed to obtain the health state of the train. There are many algorithms for the health detectors, in which a simple one is utilized in this study: the delay time is the difference of braking operation time T_1 and signal output time T_2, as shown in equation 2:

$$T = T_2 - T_1 \tag{2}$$

The command is the braking boolean signal in the model

$$T_1 = if\ command \text{ then } time \tag{3}$$

The measured braking pressure is P, and $\tilde{P}$ was obtained after low pass filter, if the braking pressure was controlled to the desired pressure with no waves, the pressure should be above the threshold a and the difference of the pressure should be below threshold b, as shown in equation 4:

$$T_2 = \text{if } j \text{ then } time \tag{4}$$

Where j is defined in equation 5:

$$j = \text{if } (\tilde{P} > a \text{ and } \text{der}(\tilde{P}) < b) \tag{5}$$

and the relationship of braking pressure P and delay time T with health degrees H is shown in equation 6:

$$H = J(P,T)g(T) \tag{6}$$

Where $J(P,T)$ is described in equation 7:

$$J(P,T) = \text{if } (command \text{ and } \tilde{P} > a) \tag{7}$$

As the description above, the health monitoring was implemented and demonstrated in the following chapter.

4 Case study

4.1 Modeling and validation of braking system

The Chinese EMU is composed of 8 car units, divided into 2 traction units, each includes 2 driven cars and 2 trail cars, every driven car has a traction control unit (TCU) and every car unit has a braking control unit (BCU). BCU adopts the electrical-pneumatic braking controlled by TCU and BCU. The electrical braking is preferred and pneumatic braking is applied when the braking force is insufficient. BCU sends commands to air supply system according to the braking level set by driver, actual velocity of the train and pressure value detected by the braking system. A BCU model is shown in figure 8. Technically, drivers can apply the braking without level set on the screen, but it's difficult to find appropriate operation position which is important to establish the reflection of driver. So braking is divided into service braking with 7 levels (B1 ~ B7), emergency braking (EB) and urgent braking (UB) which applies mechanical braking only. In practice, the signals received by braking system are non-level continuous signals. The mechanical braking system is modeled based on the PneumaticLib developed by Modelon AB, as shown in figure 9, consisting of parking control unit, braking control unit, air supply and air spring control unit, pressure switches and braking executing unit. The braking execution unit of the driven car consists of four wheel discs

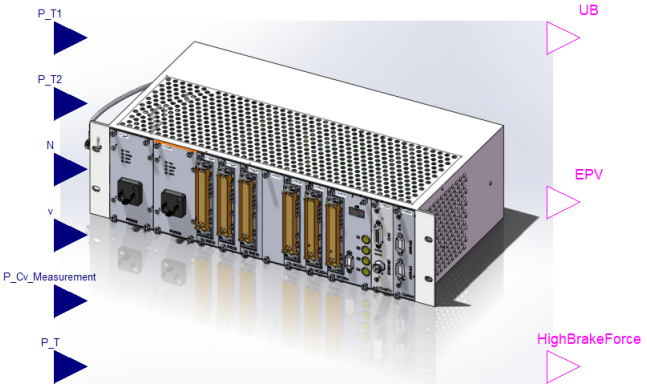


Figure 8. BCU model

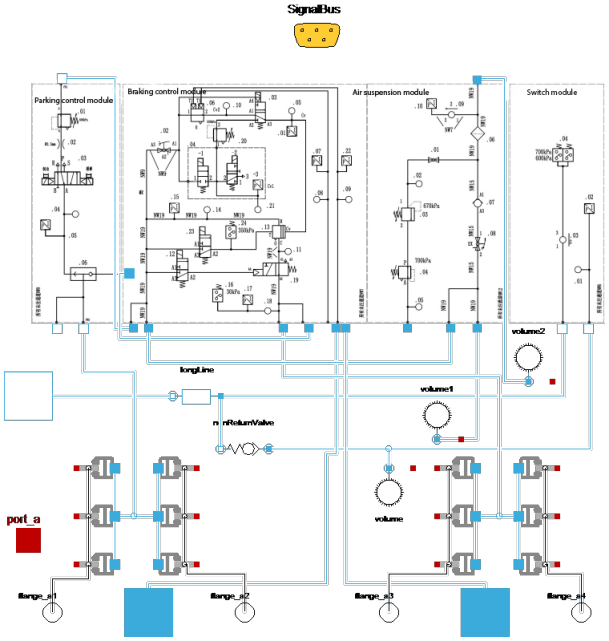


Figure 9. Braking system of the trail car TC01

per wheel set, and for the trail car, the braking execution unit is six hub discs. The main working unit is BCU. The braking system of the car TC01 is shown in figure 9. The BCU is as shown in figure 10, in which there are 3 special valves: EP valve used for adjusting and stabilizing the pressure quickly, empty and load valve used for controlling and adjusting the pressure according to the loads, relay valve used for switching between service braking and urgent braking. The models are described as follows: The details of EP valve model in figure 10 is shown in figure 11, consisting of 2 direction control valves (DCV_2_2). The pressure between the two valves is adapted by the switching operation of the direction control valve. The empty and load valve was depicted in figure 12, whose main function is to receive the air spring pressure signals of 2 bogies and output the braking control pressure that is linear to the load pressure value based on the leverage effect, so that the braking cylinder control pressure varies with the load in the braking mode. The mechanism of the

DOI 10.3384/ecp2020174109

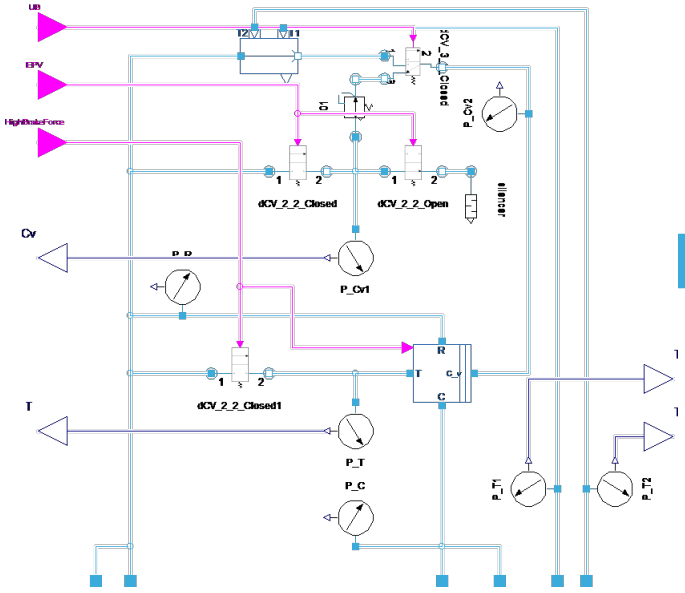


Figure 10. Braking control unit

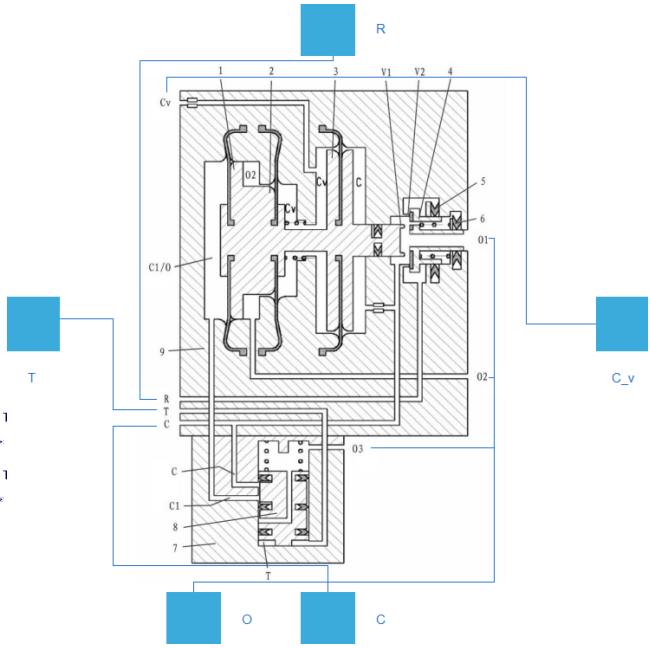


Figure 13. Mechanism of relay valve

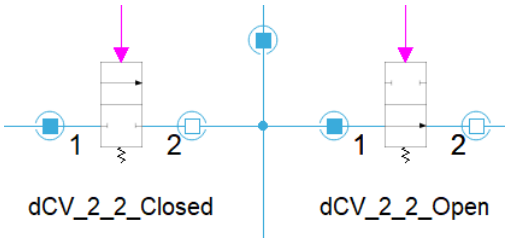


Figure 11. EP valves

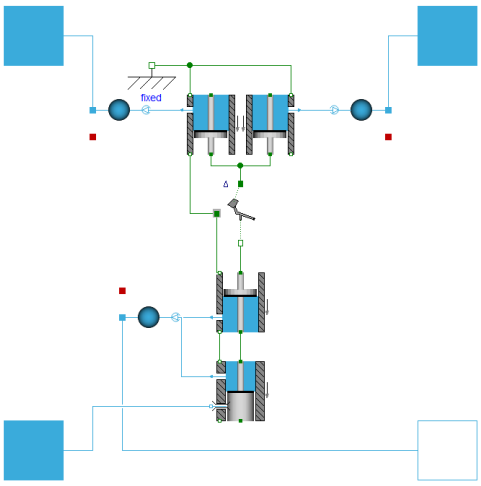


Figure 12. Empty and load valve model

relay valve is shown in figure 13, whose main function is to increase the flow rate and ensure the output pressure C varies with pressure C_v. When the train is running at high speed, reverse valve is closed, relay valve output low braking pressure; when the train is running at low speed, reverse valve is open and relay valve outputs high braking pressure. The reverse valve in normal operation is shown in figure 14. The model is validated by experiment data, as shown in figure 15. The input of BCU is the control signal from driver and actual speed of the train. Given that the driver operated in levels and the input signal of the BCU is continuous, the calculating results were revised and validated its consistency with test data. The formula of the validation is shown in equation 8, where T_C is the sample period, X_n is revised value, X_{meas} is test data and $X_{simulation}$ is calculating results. The validation results is shown in figure 16 indicating that the model is accurate enough for fault diagnosis and health monitoring.

$$\frac{1}{T_c}\int_t^{t+T_c}\frac{|X_{meas}(u)-X_{simulation}(u)|}{|X_{meas}(u)|+0.01X_n}du \leq 0.05\,\forall t \qquad (8)$$

4.2 Fault injection and failure sample generation

The fault caused by the reverse valve of the relay valve is performed as time delay of the switch between high and low pressure. Specifically, the reason is that the coefficient of friction is too large for the moving valve. The fault models of reverse valve are injected into the healthy model through changing the coefficient of friction for the fault diagnosis modeling. The fault models of reverse valve is shown in figure 17. The same control signal is applied to the models with fault injection, and with different coefficient of friction, different braking pressure is obtained, the

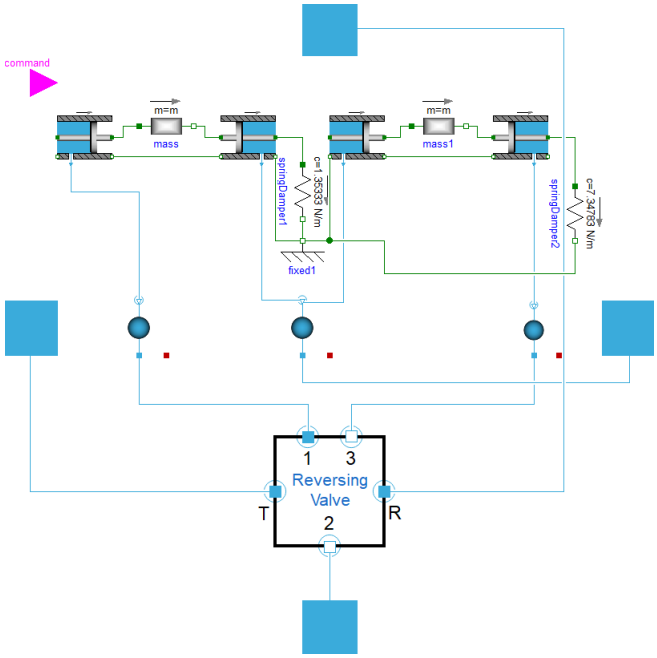


Figure 14. Relay valve model in normal operation

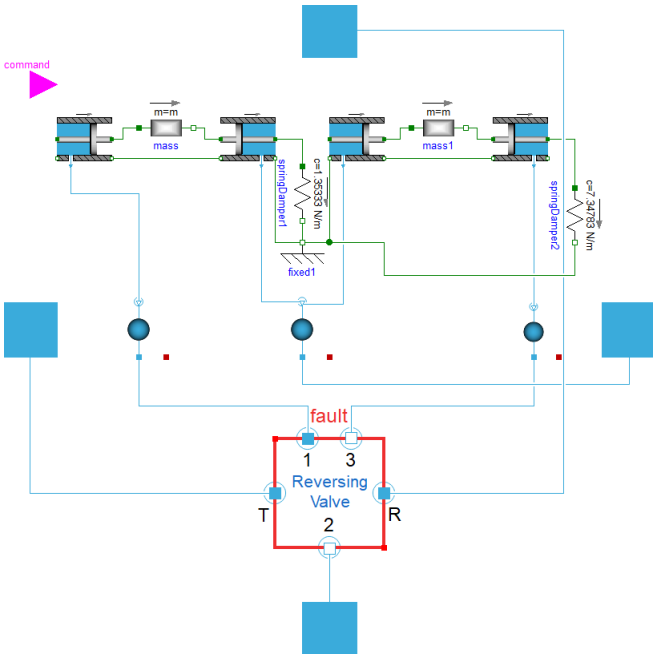


Figure 17. Relay valve with faults

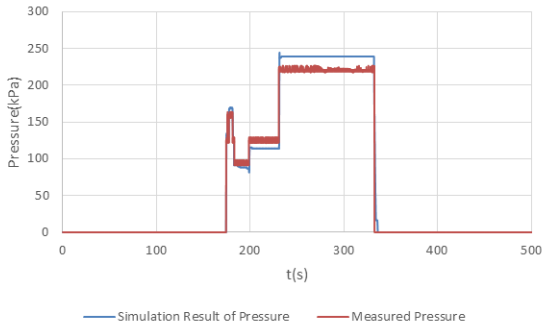


Figure 15. Validation results of the braking pressure

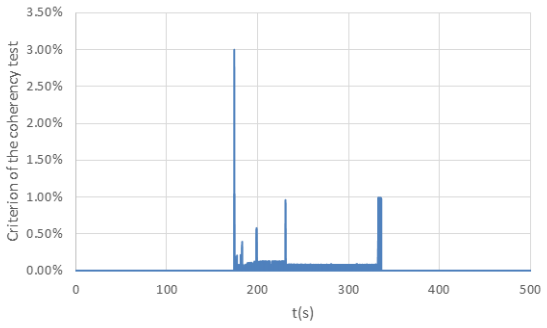


Figure 16. Relative error of the coherency validation

time of the reverse valve to complete the switching operation is delayed, or even can't switch normally. Specifically manifested in three forms as follows: smooth switching between high and low pressure, slow switching between high and low pressure, failed switching between high and low pressure. When there are switching delays or failures that reverse valve can not be switched properly, the braking distance of the failed train is longer than that of the fault-free train, as shown in figure 18.

4.3 Application of sample identification based health monitoring

The faults in the train have bad impacts on the performance, life cycle and stability of braking system. The unstable performance and component broken manifest immediately, and performance degradation is difficult to notice which may cause severe consequences once the failure occurred. It's critical to monitor the system states online in real time ensuring the normal operation in healthy condition. The fault distribution is assumed as normal distribution. The healthy degree is 1 when the coefficient of friction is 0 in which the probability is biggest. And the health degree and probability decrease with coefficient of friction increases, as shown in figure 19. However, considerable samples are essential in the process of the monitoring online system development, which are generated by injecting different coefficient of friction into models with the same pneumatic switch signals, as shown in figure 19, 20. The figure 21 depicts different braking cylinder pressure and abstract the delay time of signals T and braking pressure P to analyze and decide the health degree. The simulation results of the test case by applying the proposed sample identification based approach is shown in figure 22. More sophiscated control algorithms for fault diagnosis can be

DOI
10.3384/ecp2020174109

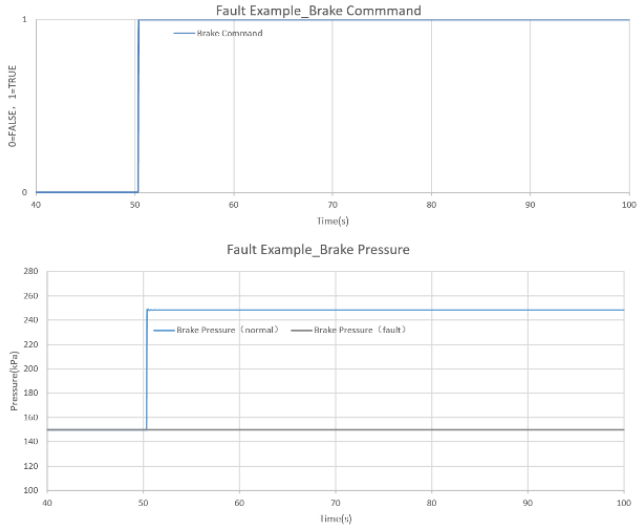


Figure 18. Simulated results when the high and low pressure failed to switch

developed based on the considerable samples generated by the models, and applied in the health monitoring system development.

5 Conclusion and future work

In this paper, a sample identification method for the fault diagnosis and health monitoring is introduced. The models of braking system in a chinese high speed train with high accuracy and the corresponding fault models have been implemented. The faults were injected into the healthy models to generate a large amount of fault samples. The new health monitoring method has been demonstrated by a predefined failure in the braking system.

Acknowledgment

The authors gratefully acknowledge ZEMT for their help of providing the braking system test data. SIMTEK also cooperated with ZEMT on the modeling of other EMU subsystems such as traction system and control system and integrated all the key subsystems as a train for power consumption estimation and other studies.

References

[1] K. Barrow, “Ten years, 27,000km: China celebrates a decade of high-speed,” International Railway Journal. August 2, 2018.

[2] WS Guan, JJ Wang, HS Wang, “The Braking Mode Simulation and Analysis for High-speed Railway”, 4th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, China: Beijing, 2011, pp.683–686.

[3] Z WG, C Cecati, et al, “A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches”, IEEE transactions on industrial electronics, vol.62, 2015, pp. 3757-3767.

[4] J. Chen and H. Zhang, “Robust fault detection of faulty actuators via unknown input observers,” , vol.22, Int. J. Syst. Sci., 1991, pp.1829–1839.

[5] E. Chow and A. Willsky, “Analytical redundancy and the design of robust detection systems”, IEEE Trans. Autom. Control, vol.29,1984, pp.603–614.

[6] N. Viswanadham, J. Taylor, and E. Luce, “A frequency-domain approach to failure detection and isolation with application to GE-21 turbine engine control systems”, Control, Theory Adv. Technol., vol.3, 1987, pp. 45–72.

[7] S. Helm, M. Kozek, and S. Jakubek, “Combustion torque estimation and misfire detection for calibration of combustion engines by parametric Kalman filtering”, IEEE Trans. Ind. Electron., vol.29, 2012, pp. 4326–4337.

[8] A. Akhenak, E. Duviella, L. Bakoa, and S. Lecoeuchea, “Online fault diagnosis using recursive subspace identification: Application to a damgallery open channel system”, Control Eng. Pract., vol.21, 2013, pp. 797–806.

[9] C. Keliris, M. Polycarpou, and T. Parisini, “A distributed fault detection filtering approach for a class of interconnected continuous-time nonlinear systems”, IEEE Trans. Autom. Control, vol.58, 2013, pp. 2032–2047.

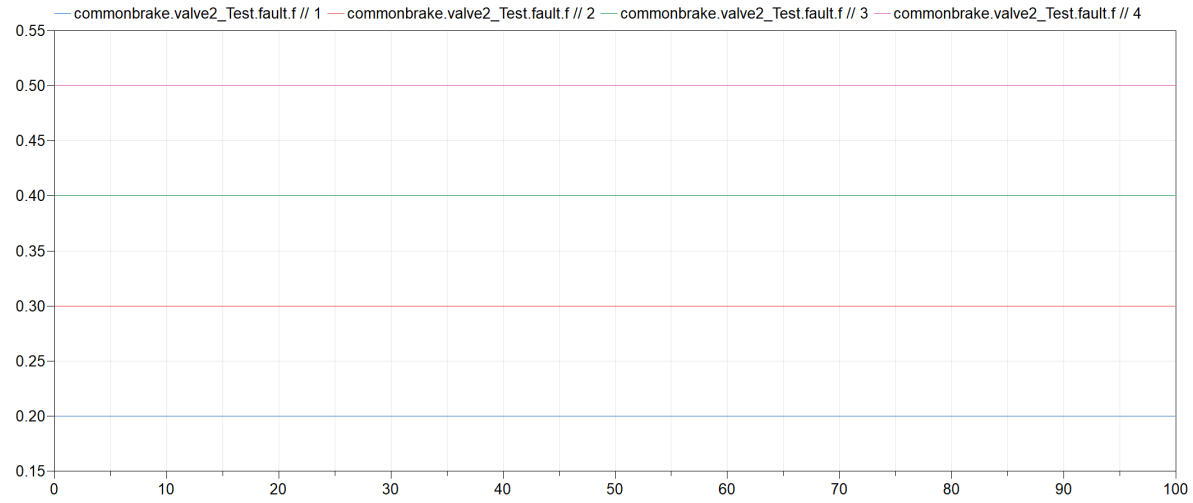


Figure 19. Coefficient of friction when injecting faults

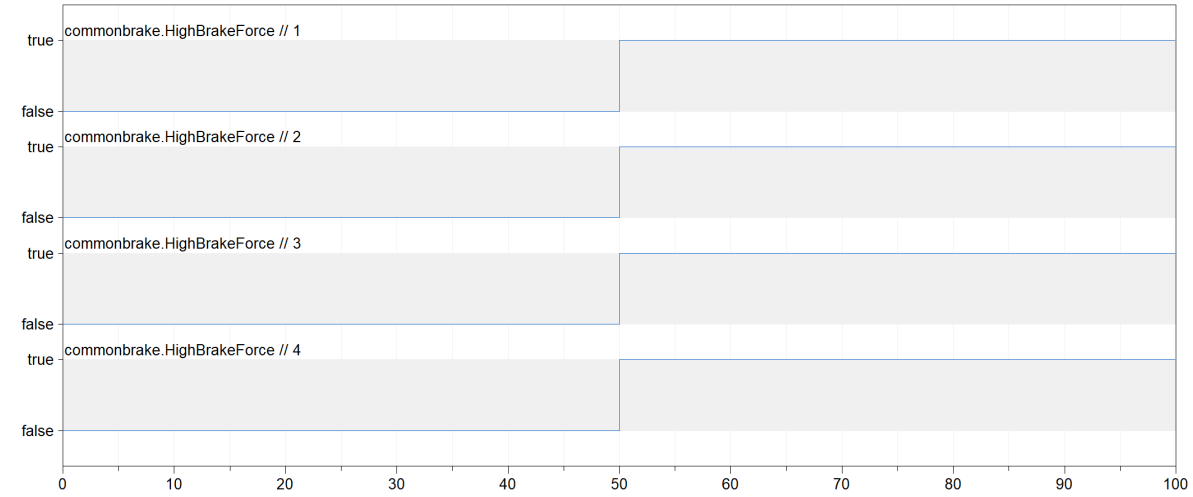


Figure 20. Pneumatic switch signals

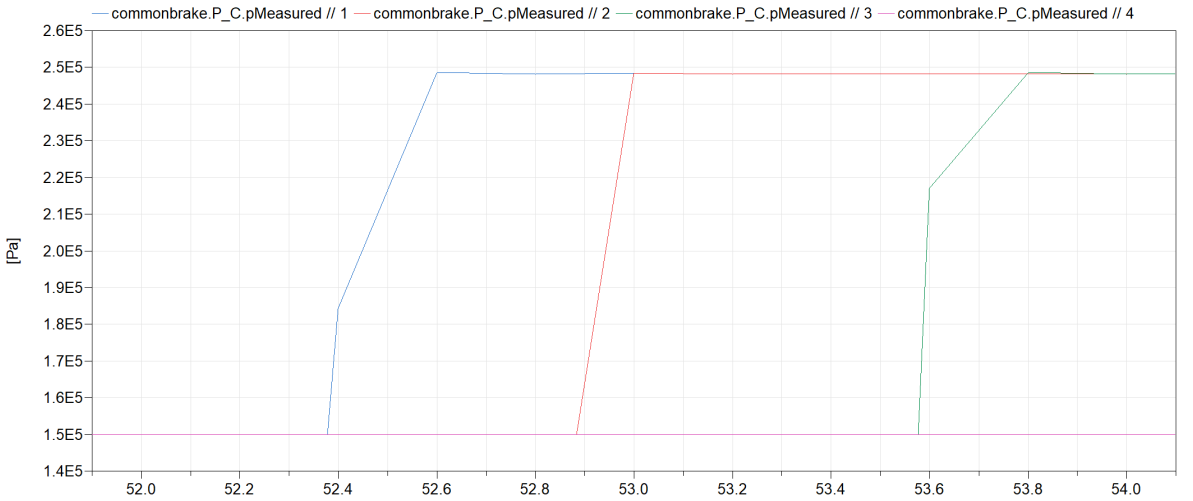


Figure 21. Braking pressrue relative to different coefficients of friction

DOI 10.3384/ecp2020174109

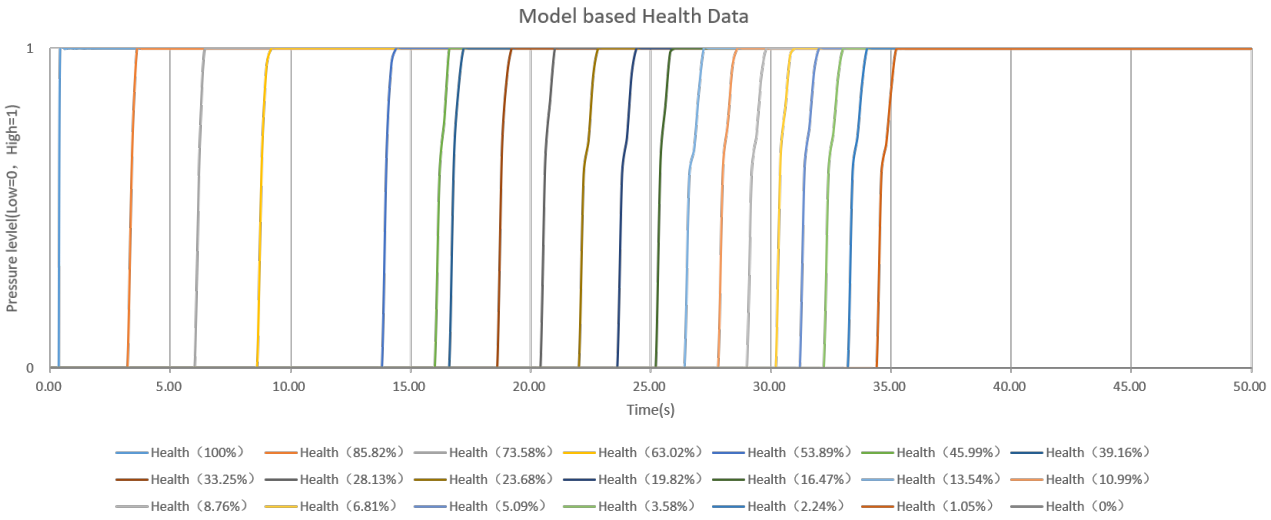


Figure 22. Different health degrees decided by multi samples generated by fault injection

DOI
10.3384/ecp2020174

Relay System Model with Contact Bounce and Flexible Beam

Rui Gao[1] Ivar Torstensson[2]

[1]Modelon K.K. Roppongi 1-10-3-209, Minato ku, Tokyo 106-0032, Japan,
rui.gao@modelon.com

[2]Modelon, AB, Sweden, Anders Carlssons Gata 14 SE-417 55 Göteborg, Sweden
ivar.torstensson@modelon.com

Abstract

In this paper, an electromagnetic relay's dynamic modeling problem is discussed. The moveable part of the relay is modeled as Euler-Bernoulli beam, while the impact phenomenon between the moveable and stationary poles is captured by 2D contact model using polygon to represent the contact pole profile. For the completeness of the system, the electromagnetic driving force is modeled using electrical and magnetic components from Modelica Standard Library. The simulation results show that the model reproduces the electromagnetic relay's physical behavior under normal operation condition. In addition, the model can be used to investigate the effect of abnormal scenario such as the contact surface profile change under material erosion. This application illustrates that the proposed method and model can be applied to the electromagnetic relay's simulation-based design as one-stop tool.

Keywords: electromagnetic relay, contact bounce, polygon, flexible beam

1 Introduction

The electromagnetic relay is one of the most widely used devices in industries and daily life. Electromagnetic relays provide a well proven solution to switching loads in a variety of applications. However, relays are known for their limited reliability due to mechanical wear of internal switching elements, essentially the life of the relay may be determined by the life of the contacts. The contact bounce phenomenon between the stationary and moveable contact poles has been studied from various perspectives. The research results show that it is the main reason for electric abrasion and material erosion of the contact poles, which affects the switching reliability.

Many studies have been focused on the characteristics of the contact bouncing mechanism through both experimental and analytical investigation, like the bounce time, first bounce amplitude, factors to affect contact bounce (McBride et al, 1992; Kondo et al, 2019);

In the dynamic modeling and simulation perspective, researchers are mainly concentrated on FEM method and tools. MATLAB/Simulink is used to model a simplified relay model represented by ordinary differential equation to capture the contact bounce (Nouri et al, 1997), indicating that lumped model is as effective as FEM in modeling relay system. Combined approach of distributed and lumped modeling is reported (Xiong et al, 2009), where finite difference method for reed system dynamic analysis with 1D spring mass system for the contact force calculation is studied. However, the friction force is not considered.

Modelica is a suitable language for multi-discipline modeling and simulation and has been widely used across industries. However, Modelica's traditional network model of component module encounters challenge in electromagnetic relay modeling. The difficulties stem from the innate features of relay where the moveable contact pole mounted on the flexible beam is driven by electromagnetic force, and reaction force from impact is discontinuously acted upon. Especially, the simulation performance under this specific method to capture discontinuous contact phenomenon has not yet been established with common acceptance.

There is a commercial library for flexible multibody modeling (Hackerman et al, 2006); Claytex developed another commercial library Flexbody. Both libraries need Finite Element model's results and importing them into Modelica environment after pre-processing. This dependency to the pre-processor makes the Modelica users difficult to use these solutions. A motorbike swingarm modeling and analysis application using Modelica multibody and parameters obtained through FEM package is reported (Gianni et al 2014), where authors try to alleviate the dependency to FEM software. On the other hand, an efficient Modelica modeling method that is independent with FEM code is investigated for flexible beam (Ericsson et al. 2015). We will use the model created based on this approach in the relay modeling in this paper.

Another difficulty in relay mechanical modeling is how to handle the discontinuous contact phenomenon. Modelica Standard Library (MSL) provides a 1D model WithStopAndFriction to capture the slip and stick, where hard stops are included using mode transition. This proves to be a practical method for simple 1D application. However, the collision occurred beyond the translation direction cannot be applied. An early effort to handle the contact modeling under multibody

paradigm is reported where an external C package is used for collision detection (Otter et al. 2005). A free Ideal Contact Library is developed in Modelica multibody framework. It is easy to use but limited to the cases that the contact point location is predictable beforehand (Oestersötebier et al, 2014). A generic Modelica framework for multibody contacts and Discrete Element Method has been proposed (Elmqvist et al 2015). Component-based 3D modeling for dynamic systems inspired by game software is studied recently (Neumayr and Otter, 2018, 2019) where the limitation of the current Modelica specification is analyzed and a new way of 3D modeling technique implemented by Julia language with Modia environment is illustrated, e.g. Modia3D prototype.

Though the latest research results are quite promising to improve the multi-body system contact modeling with variable step solver, the deformation of flexible body has not been mentioned. To address electromagnetic relay's modeling issues like how to develop an efficient, light-weight package that covers flexible beam, contact bounce, and electromagnetic driving in order to realize simulation-based design is of wide interest to industries. This is the main focus of this paper.

The rest of the paper is organized as follows. Section 2 describes the relay system modeling. Section 3 illustrates the simulation results for normal case and abnormal case. A conclusion is summarized at the Section 4.

2 Relay Modeling

A relay is an electromagnetic switch used to switch high voltage or current using low power circuits. Its typical structure is shown in Figure 1, containing an electrical coil, magnetic core, yoke, armature, moveable and stationary contact.

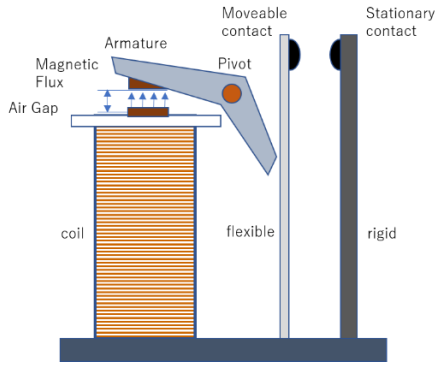


Figure 1. Typical electromagnetic relay structure.

An electromagnetic relay can always be separated into two subsystems (electromagnetic and mechanical) which interact with each other. The electromagnetic subsystem is governed by Maxwell's equations whereas the mechanical subsystem obeys Newton's law.

2.1 Flexible beam

The moveable beam in Figure 1 can be equivalently modeled as cantilever beam in Figure 2. This is one challenge to model continuum with Modelica without relying on FEM software, while the simulation performance is expected to be at 1D simulation tool level.

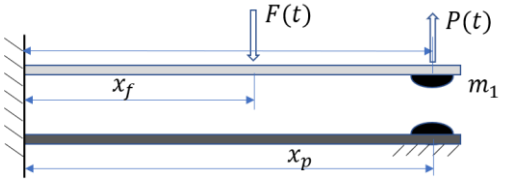


Figure 2. Mechanical part of the relay

Since the cross-section area of the beam is constant, the linear stress strain law can be applied to get motion equation as below (Xiong et al 2009).

$$\left[\rho A + m_1\delta(x - x_p)\right]\frac{\partial^2 y}{\partial^2 t} + EI(1 + ig)\frac{\partial^4 y}{\partial^4 x} = F(t)\delta(x - x_f) - P(t)\delta(x - x_p) \quad (1)$$

where E is the elastic modulus of the beam, ρ the density, A and I are respectively the area and the moment of inertia of the beam cross-section, g is the hysteretic damping ratio where (1+ig) is complex number, $\delta(x)$ Dirac function and $P(t)$ the contact force during the impact. Cantilever beam using Euler-Bernoulli theory is typical suitable to the beams for which the length is much larger than the thickness.

For the moveable beam in Figure 2, the ratio of length over thickness is over several dozens. Euler-Bernoulli beam implemented in Modelon Base Library (MBL) is used in the successive modeling in this paper. The implementation is based on Craig-Bampton method, which is a reduction technique where complex deformable structures are divided into an assemblage of substructures (Ericsson et al 2015). Each substructure has its own mass and stiffness matrix associated with a set of generalized coordinates. The set involves two forms of generalized coordinates, boundary generalized coordinates and internal generalized coordinates. The boundary generalized coordinates prescribe the displacements and rotation at the boundaries while the internal generalized coordinates are related to the free vibration modes of the substructure with completely restrained boundaries. Consequently, the two sets of generalized coordinates become decoupled at the boundaries of the substructure and coupled internally.

2.2 Contact object

In this paper, we chose a polygon-based contact solution referring to the previous study result (Elmqvist et al,

DOI 10.3384/ecp2020174119

2015). To have a reasonable computation performance, the 2-dimension contact model is built and equipped with Multibody connectors, which can be used seamlessly with rigid multibody system model.

The contact detection is usually composed of broad and narrow detection phase, while in the relay modeling application the latter phase only is good enough. The contact pole surface profile is defined by polygon, as an external object to Modelica. The polygon vertex coordinates array has to be set up by the order of counterclockwise.

Another external object is defined for contact force. The contact model is built by arranging two potential colliding polygons together in pair. Internally, these two bodies have no connector for interaction. Instead, an algorithm is used to monitor polygon objects' relative positions and detect the collision. When the collision occurs, contact force object will be used to compute the contact forces. Externally, this polygon pair has two multibody connectors working as interface to interact with outside multibody system.

The mechanical parts of the relay system are modeled by multibody system at 3D space while the contact model works at 2D space. The mechanical model with ideal driving force is shown in Figure 3**.** The body with red color represents the flexible beam, body with blue color is rigid body; The component with icon of two colliding polygons is the 2D contact model handling the two contact bodies.

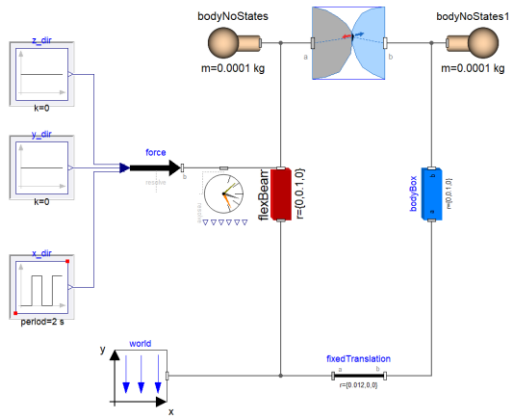


Figure 3. Relay mechanical model using MultiBody and 2D contact model

At the parameter setup diagram of the 2D contact model, the profiles of the contact pole surface can be specified through coordinates array. Table 1 shows the main mechanical parameters of the relay system.

Table 1. Relay mechanical parameters

Parameters	*value*	*unit*
Length of the fixed and moveable beam	100	mm
Dist. between the fixed & moveable beams	12	mm
Pos. of driving force	100	mm
Flexible beam		
Density	8.8	g/cm^3
Young's modulus	109 x 10^9	Pa
Shear modulus	109 x 10^9/2.6	Pa
Poisson's ratio	0.3	1
Contact poles		
Young's modulus	200 x 10^9	Pa
Poisson's ratio	0.3	1
Radius of the pole	5	mm

2.3 Contact normal force

The contact force is calculated in term of the intersection status between two collision bodies. The force value is volume dependent as in Eq. (2) for polyhedron (Elmqvist et al 2015)

$$F = Ek\sqrt{Vd} \tag{2}$$

where the E is the Young's modulus of the objects, V is the volume of the overlapping region of polyhedron (here, the area of overlapping between polygons), d is the penetration depth and k is calculated in Eq. (3),

$$k = \frac{4}{3\sqrt{\pi}} \tag{3}$$

This is a generalization of the Hertz model. The force is applied at the centroid of the overlapping region.

2.4 Contact friction force

The contact surface profile in this study is of spherical shape. The moveable part is flexible beam, the vibration from the beam will make the contact bounce phenomenon even more complicated since the contact point location may change at each time. To address the slip during tangential contact, the contact model needs to be able to calculate the friction force during the impact process.

A common practice to integrate friction force into multibody system is to approximate the law of friction using a continuous function (Popov, 2010). A tangential velocity dependent function will be used to model the friction force. The function is defined as

$$F_f = \left((2\mu_{s*} - \mu_k)\frac{x^2}{x^4+1} + \mu_k - \frac{\mu_k}{x^2+1} \right) F_n \tag{4}$$

$$\mu_{s*} = \mu_s \left(1 - 0.09 \left(\frac{\mu_k}{\mu_s}\right)^4\right) \quad (5)$$

$$x = \frac{v_t}{v_s} \quad (6)$$

where the coefficients μ_s and μ_k are the static and kinetic friction coefficient respectively, F_n is the contact normal force, v_t is the tangential velocity and v_s is the transition velocity from static to kinetic friction. Figure 4 shows the friction as a function of tangential velocity (Elmqvist et al 2015).

Keeping a purely kinetic friction function means that there will never be a completely "stuck" mode. A small tangential force that would not influence a system with Coulomb friction at rest. This friction function will give rise to a tangential speed until a force equilibrium is met.

The transition velocity v_s decides how quickly the friction force increases. A smaller value will approximate Coulomb friction better.

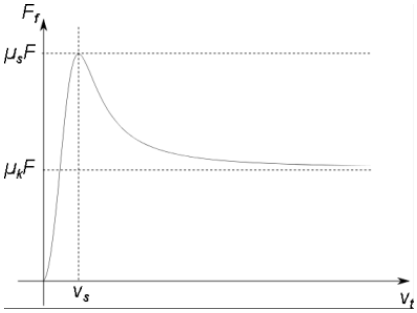


Figure 4. Friction as function of tangential velocity

2.5 Electromagnetic circuit

The electromagnetic part is modeled by an equivalent circuit using flux tubes and air-gaps network. Saturation of the magnetic material and dynamical effects can be introduced to the model.

In the current study, the model from Magnetic Library of MSL is used in Figure 5.

The interaction between the subsystems is twofold: firstly, the electromagnetic forces and torques applied on the mechanical subsystem produce deformations and accelerations; Secondly, the motion affects the electromagnetic subsystem by induced electromotive forces and by variations of reluctance.

In this paper, we focus on the former one, that is on the dynamics of the mechanical system driven by force from electromagnetic subsystem. The thrust F developed by a translational electro-magneto-mechanical actuator is calculated by following Eq. (7) in the lumped magnetic network models.

$$F = \frac{1}{2} \sum_{i=1}^{n_{linear}} V_{mi}^2 \frac{dG_{mi}}{dx} \quad (7)$$

where *nlinear* is the number of flux tube elements with constant relative permeability that change its permeance $G_{m\,i}$ with armature position (index i), $V_{m\,i}$ the magnetic voltage across each respective flux tube and $dG_{m\,i}/dx$ the derivative of the respective permeances with respect to armature position.

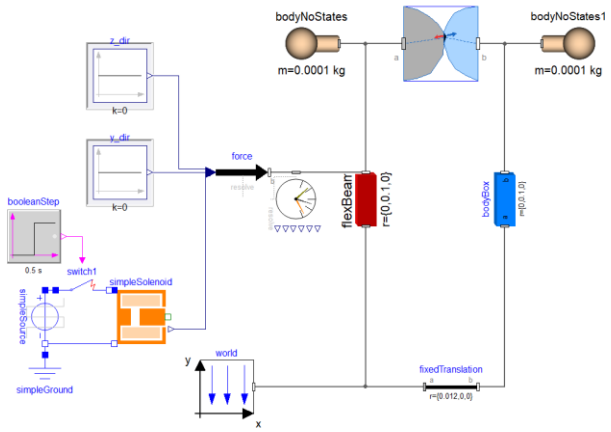


Figure 5. Electromagnetic relay model

The armature rotational dynamic is not considered. Instead a translational solenoid model in MSL.Magnetic with minor customization is used to generate the corresponding thrust force.

3 Simulation

3.1 Contact bouncing suppression

Eliminating the contact bouncing is of interest in relay design. In Figure 6, the upper plots are the moveable contact position in horizontal direction; the lower plots are the Boolean signal for contact.

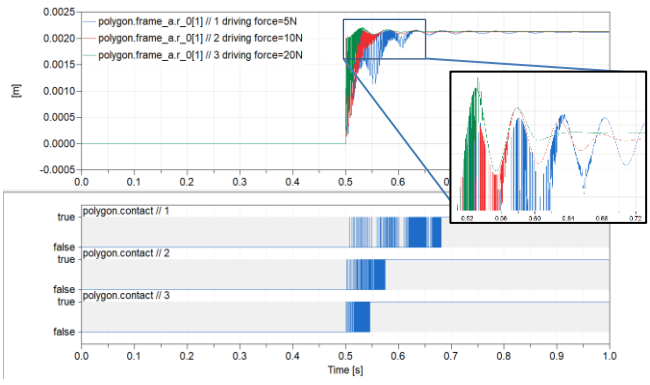


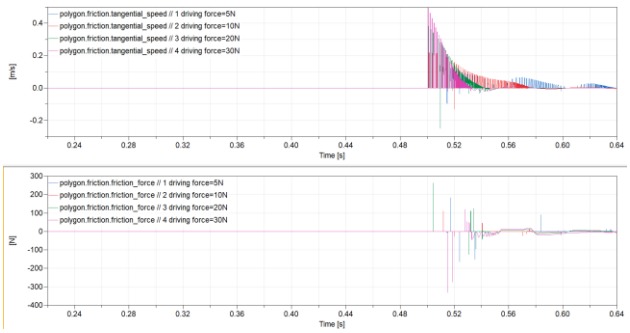
Figure 6. Contact point position horizontal variation (upper); Contact Boolean signal (lower)

The simulation results show that when the driving force applied has different amplitude as 5N, 10N, 15N, the higher the driving force, the less contact bounce in contact make. This result is consistent with the result in previous studies [McBride W. 1992].

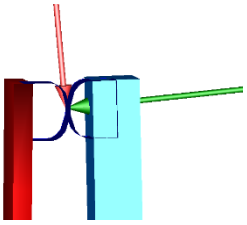
However, when increasing the driving force to 30N, it is observed that the tangential speed will grow quickly and the friction force correspondingly (Figure

DOI 10.3384/ecp2020174119

7). This can lead to the unstable contact that should be avoided.



(a) Tangential speed and friction force of the contact under different electromagnetic driving forces



(b) Normal force and friction forces in contact

Figure 7. Speed dependent friction force and normal force

3.2 Flexible beam vibration

To identify the effect of moveable beam to the relay contact bounce, the pulse signal for electromagnetic force is set up as period of pulse is 2s, width of pulse is 50%, start time=0.15s and the amplitude is 5N. Then the simulation result of the moveable contact pole position in horizontal direction is shown in Figure 8. Continuous vibration is observed after the first contact release in 1.15s and the second contact make is under the mixed beam vibration and contact bounce. This result is regarded as a new observation through relay system simulation that has not been reported at the early simulation research.

In the conventional relay system, there is a return spring and the moveable beam is rigid. The relay structure discussed in this paper has no return spring which makes the structure simple and has high structure reliability.

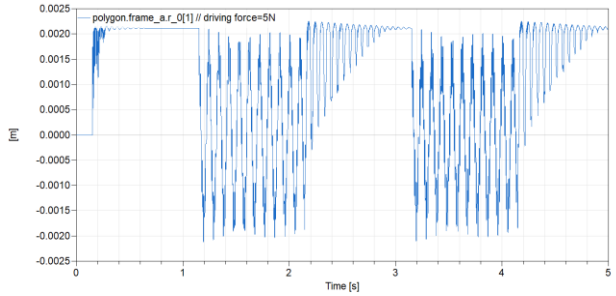


Figure 8. Vibration of the successive contact make

However, to achieve the high robustness in contact make is still a design problem through which to optimize the parameters of the relay system.

In the current simulation study, we simple increase the electromagnetic driving force from 5N to 20N, Figure 9 shows that it is effective to suppress the contact bounce as well, though the number of the contact bounce after the first time will be bigger.

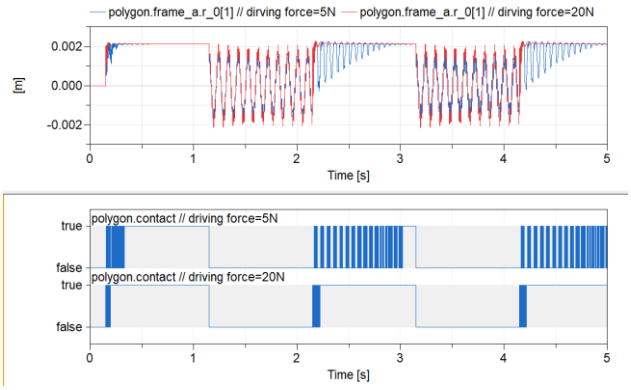


Figure 9. Contact bounce suppression by increasing driving force amplitude from 5N to 20N

3.3 Contact surface profile change

In practical, on a worn electrical contact the radius is expected to be dominated by the local roughness rather than the initial radius of the contacts.

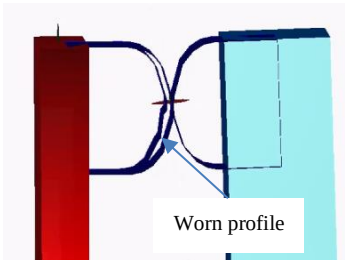


Figure 10. An example of contact pole worn surface profile

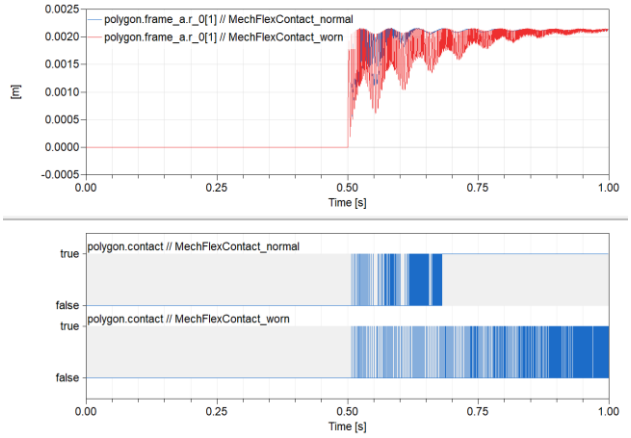


Figure 11. Comparison of the contact bounce between normal profile and worn profile

It is noticed that the contact bounce with a worn contact pole profile in Figure 10 will be continued longer than a normal case as shown in Figure 11.

Though the mechanism of electric erosion to the contact surface and its effect to the contact dynamics need further research, the model built in the paper is expected to provide simulation support to the study.

3.4 Electromagnetic response

For the completeness, the ideal driving force is generated from electromagnetic circuit. The typical responses of the electromagnetic system are shown in Figure 12.

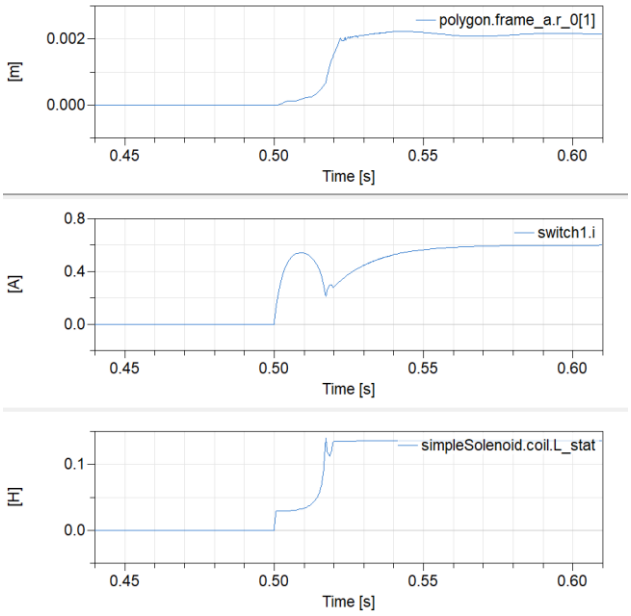


Figure 12. Moveable contact position(upper); Electrical switch current (middle); Magnetic circuit static inductance (lower)

3.5 Simulation performance

Using the created model including flexible beam, 3D multibody and 2D contact model, it takes less than 8s to simulate a use case in Figure 9. i.e. 3 contact make process within 5s (Figure 13).

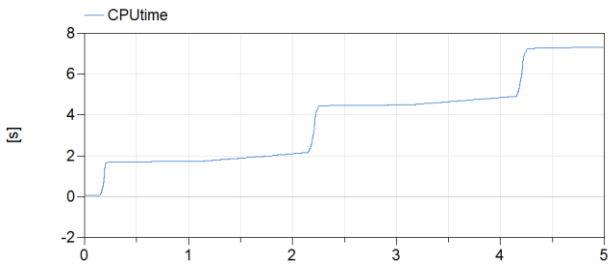


Figure 12. CPU time and event for a 5s simulation

4 Conclusions

A multi-physics electromagnetic relay model is created where the mechanical part is under multibody framework and electromagnetic part is by equivalent circuit. The main effort is to use the flexible beam represent the relay moveable part and 2D contact model to capture the impact between the moveable and stationary poles. Simulation results show the built model reproduces the physical behavior of the relay, especially the contact bounce phenomenon. Furthermore, the model is easily extended to a simulation for a worn contact surface case, which is expected to help improve the relay design by simulation analysis.

The simulation performance of the created model shows at the similar level with rigid multibody system even though the flexible beam and contact elements are included.

For the future work, the model validation using practical measurement to improve the prediction accuracy is an important step towards practical engineering application. Applying the latest 3D modeling technique is another topic of interest to investigate efficient modeling approach with performance comparison.

Acknowledgements

The polygon-based collision function is supported by Oskar Åström and Hilding Elmqvist based on the earlier work (Elmqvist *et al*, 2015).

References

A. Heckmann, M. Otter, S. Dietz and J. D. López. The DLR FlexibleBody library to model largemotions of beams and of flexible bodies exported from finite element programs. In:5th International Modelica Conference. Wien, 2006, pp. 85 –95.

A. Ericsson, A. Kjellander Efficient Modeling of a Flexible Beam in Dymola using Coupled Substructures in A Floating Frame of Reference Formulation, *Master Thesis*, Department of Mechanical Engineering, Lund University, 2015

A. Neumayr, M. Otter. Component-Based 3D Modeling of Dynamic Systems *Proceedings of the 1st American Modelica Conference*, pp. 175–186, Oct. 2018, Cambridge, Massachusetts, USA. doi:10.3384/ ECP18154175

A Neumayr, Martin O. Algorithms for Component-Based 3D Modeling *Proc. of the 13th International Modelica Conference*, pp. 383–392, Mar. 2019, Regensburg, Germany. doi:10.3384/ ECP19157383

F. Oestersötebier, P. Wang, A. Trächtler A Modelica Contact Library for Idealized Simulation of Independently Defined Contact Surfaces, *Proceedings of the 10th International Modelica Conference, pp929-937, March 10-12, 2014, Lund, Sweden*, doi: 10.3384/ECP14096929

FlexBody Library, https://www.claytex.com/products/

Gianni F. Bruno S. Andrea R. Multibody Model of a Motorbike with a Flexible Swingarm *Proc. of the 10th International Modelica Conference*, pp. 273-282, *March 10-12, 2014, Lund, Sweden*, doi: 10.3384/ECP14096273

H. Elmqvist, A. Goteman, V. Roxling, and T. Ghandriz. Generic Modelica Framework for MultiBody Contacts and Discrete Element Method. In Peter Fritzson and Hilding Elmqvist, editors, *Proc. of the 11th International Modelica Conference*. LiU Electronic Press, Sept. 2015. URL http://www.ep.liu.se/ ecp/118/046/ecp15118427.pdf.

H. Nouri., N. Larsen, T. S. Davies. Contact Bounce simulation using MATLAB. *IEEE Xplore*, 9(2):284–288, November 1997. doi: 10.1109/HOLM.1997.638054

John W. McBride, Suleiman M. Sharkh. Electrical Contact Phenomena During Impact, *IEEE Transactions on Component, Hybrids, and Manufacturing Technology*. Vol., 15, No. 2, April 1992.

M. Otter, H. Elmqvist, and J. Diaz Lopez. Collision Handling for the Modelica MultiBody Library. In Gerhard Schmitz, editor, *Proc. of the 4th International Modelica Conference*, March 2005. https://modelica.org/events/Conference2005/online_proceedings/Session1/Session1a4.pdf.

S. Kondo, Y. Yokote, H. Tanaka Simulation Technology to Predict Dynamic Motion of Relay (In Japanese) , *OMRON Technics*. Vol. 51.021JP, 2019.4.

Valentin L. Popov. Contact Mechanics and Friction, Physical Principles and Applications, *Springer*. e-ISBN 978-3-642-10803-7. doi:10.1007/978-3-642-10803-7.

Xiong J., He J. Zang C. Dynamic Analysis of Contract of Aerospace Relay Based on Finite Difference Method. *Chinese Journal of Aeronautics*, 22 (2009):262-267. doi:10.1016/S10000-9361(08)60097-7.

DOI
10.3384/ecp2020174

Modeling and Simulation of SSPC based on Dymola Software and Modelica Language

Yufeng Wang[1] Yufei Tao[1] Qinzhou Lin[1] Weilin Li[1]

[1]College of automation, Northwestern Polytechnical University, China, wyfnwpu@mail.nwpu.edu.cn, taoyufei0412@163.com, shforest@hotmail.com, liweilin907@126.com

Abstract

Solid State Power Controller (SSPC) is one of the key components of aircraft power distribution system. It is a switch device composed of solid-state semiconductor devices, which is used to switch on/off the circuit and realize the control and protection of power distribution bus. It is one of the important components of aircraft automatic power distribution technology. Compared with traditional mechanical switches, the SSPC does not cause mechanical wear, has a low failure rate, and has high reliability, which is especially suitable for aviation applications. Compared with smart contactors, SSPC is small in size and high in integration, meeting the streamlined requirements of advanced aircraft electrical systems for power distribution devices. The Dymola simulation software based on Modelica language has the advantages of simple operation, high integration, and convenient for collaborative simulation. This paper first introduced the basic principle of SSPC, then combined with Modelica language and Dymola simulation software to model and verify the function of current limiting, overcurrent protection and short circuit protection of SSPC, numerical simulation results were also provided.

Keywords:SSPC, Dymola modeling, Modelica language.

1 Introduction

Avionics, weapons, and flight control systems are becoming more and more complete with the rapid development of aeronautical technology, aircraft performance has been greatly improved, and more and more airborne electrical equipment has been used. The aircraft distributed power distribution system has become the future development direction. Traditional mechanical switches can no longer meet the needs of circuit protection. Solid-state power controllers that can be integrated in power distribution systems are an important part of distributed power distribution systems.

The SSPC is an intelligent switching device composed of semiconductor devices. As the core component of Secondary Power Distribution Assembly, it is used to switch on and off the circuit, realize circuit protection, receive the control signal of the preceding computer and upload the working status to it [1]. Its function is similar to the traditional combination of circuit breakers and relays in series or other control protectors. It is used to replace the relay's conversion function and the protection function of circuit breaker, the functional equivalent diagram of SSPC is shown in Figure 1. It is much better than these traditional devices in performance and function. SSPC uses solid-state semiconductor devices as switching devices, which can quickly turn on and off the circuit with extremely short delay. Because solid-state switching devices have no contacts or moving parts. the SSPC does not cause mechanical wear, has a low failure rate, and has high reliability, which is especially suitable for aviation applications [2]. Compared with smart contactors, SSPC is small in size and high in integration, which meets the requirements for simplified power distribution devices of advanced aircraft electrical systems [3].

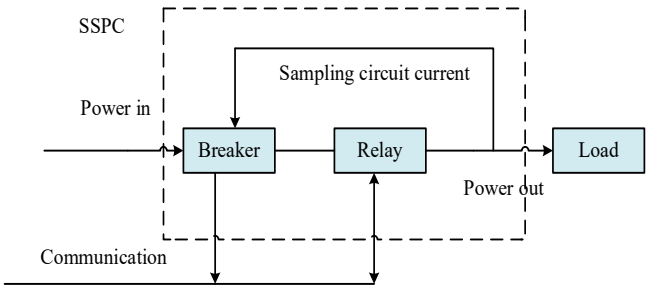


Figure 1. Functional equivalent diagram of SSPC

Dymola simulation software is suitable for modeling various physical systems. It supports a hierarchical model structure, a truly reusable component library, wiring terminals and composite acasual connections. The model library can be applied in various engineering fields [4-5].

Dymola uses a new object-oriented and equation-based modeling approach. The automatic processing of equations replaces the manual conversion of equations to block diagrams in the traditional sense. Other highlights of Dymola are as follows:

- Handle large, complex, multi-engineering models.
- Faster modeling by building graphical models.
- Faster simulation --symbol preprocessing.
- Open user-defined model elements.
- Open interface to other programs.

- 3D animation effect.
- Real-time simulation.

In this paper, the Dymola simulation software based on Modelica language is selected to build and simulate a 28V DC SSPC model, including overvoltage protection, overcurrent protection, short circuit protection and other functions.

2 SSPC Principle Analysis

As can be seen from Figure 2, the SSPC is mainly composed of logical control, MOSFET driver, power switch tube, sampling resistor, overvoltage protection module, overcurrent protection module and so on. The logical control module uses intelligent chips (or simple D flip-flops and NAND gates) to judge the voltage and current signals, and uses the result as the input of the MOSFET driving circuit to drive the MOSFET. The sampling resistor connected between the switch tube and the airborne test equipment is used to sample the current and voltage.

According to the above analysis of SSPC principle, it can be seen that it is different from the conventional thermal protection method used to achieve circuit protection in the power distribution system. Instead, it detects the current and voltage to drive the MOSFET. The performance is far superior to thermal protection devices, and its damage characteristics can be designed to be lower than the damage characteristics of on-board test equipment, achieving a more perfect protection function for the distribution network.

On the other hand, due to the fast response speed, SSPC can quickly cut off the power supply line when a short-circuit fault occurs, preventing a short-circuit from causing a sharp drop in the grid voltage and causing power interruption. Combined with the reverse-current diode, it can prevent the voltage change of the power supply bus bar and realize the uninterrupted power supply of the DC power supply.

3 Modeling various parts of SSPC

3.1 Overvoltage protection module

The overvoltage protection mainly completes the overvoltage protection function. Here, the operating point voltage of the overvoltage protection is set to 28 * 2V. When the output voltage of the channel exceeds the operating point voltage, this channel is protected and the DC contactor it controls is disconnected. When the output voltage returns to normal, the DC contactor should be switched on again.

According to the design requirements of the overvoltage protection module, a model of the overvoltage protection module is established, as shown in Figure 3.

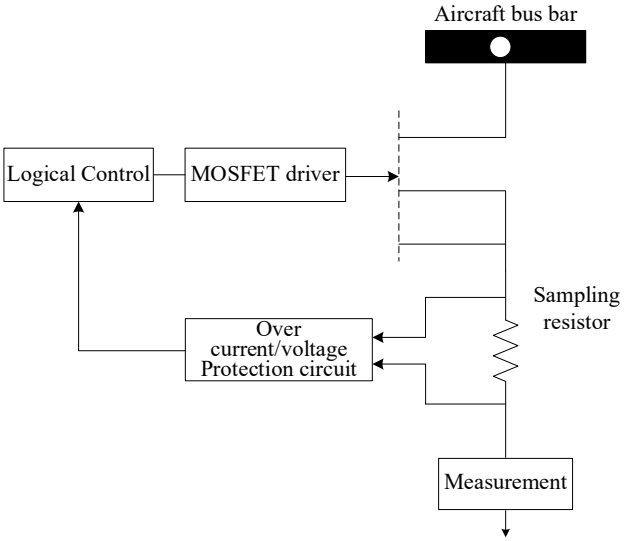


Figure 2. SSPC principle block diagram

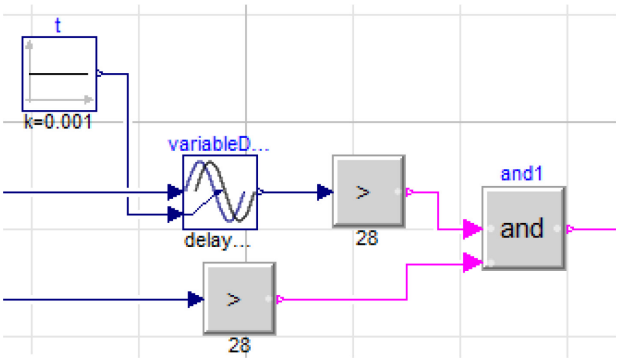


Figure 3. Overvoltage protection module

3.2 Overcurrent protection module

The difficulty of SSPC is the protection processing when the line is overloaded. This paper uses the I^2t protection curve to protect the line from overload. This is because the power loss of the line is equal to the product of the line resistance and the square of the current, and the temperature of the line is determined by the time the line consumes. For power distribution lines, I^2t is a constant value and exceeding this value may cause damage to the power distribution lines. Therefore, SSPC uses I^2t protection curves for the protection of key lines [6].

Inverse time-limit overcurrent protection of transmission lines generally adopts the following three standard inverse time-limit characteristic equation:

（1）General inverse time limit

$$t = \frac{0.014T_p}{(I/I_P)^{0.02} - 1} \tag{1}$$

（2）Very inverse time limit

$$t = \frac{1.35T_p}{I/I_P - 1} \tag{2}$$

（3）Extreme inverse time limit

DOI 10.3384/ecp2020174127

$$t = \frac{8T_p}{(I/I_P)^2 - 1} \quad (3)$$

In general, t is the operating time; T_p is the time constant; I is the effective value of the rated current; I is the effective value of the current at the time of the fault and I_p is the effective value of the rated current. In this paper, the very inverse time limit method is used, and its inverse time protection curve is shown in Figure 4.

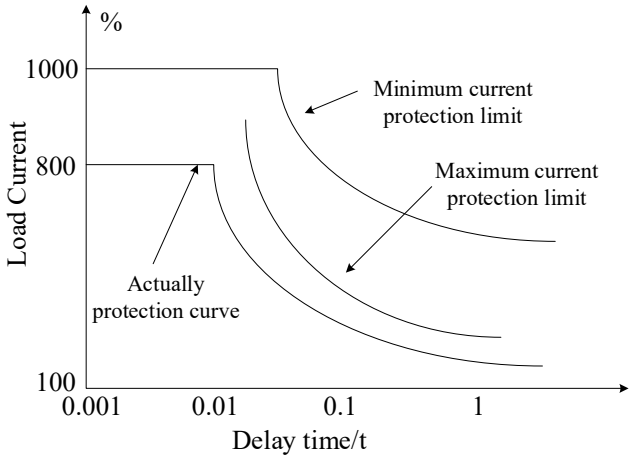


Figure 4. Typical inverse time limit curve of SSPC

The overcurrent protection mainly completes the overcurrent protection function. When the control protector detects the overcurrent fault at the output, the connection contactor of the local system and the DC contactor of the channel should be disconnected according to the anti-delay protection characteristic curve.

According to the requirements of overcurrent protection function, when the current is greater than the typical value in the overcurrent protection delay curve, the module needs to realize the reverse delay protection and the state output retention module mentioned above. The module construction is shown in Figure 5.

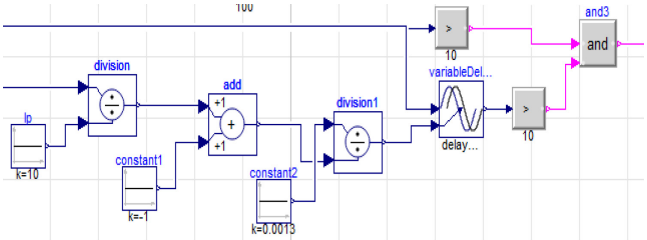


Figure 5. Overcurrent protection model

Because the overcurrent protection cannot be restored automatically, the output holding module is required to hold its output. When outputting a shutdown signal, the shutdown signal should be kept from disappearing. Therefore, an S-R trigger is added after the fault signal to achieve this function.

3.3 Short circuit protection module

The instantaneous value of the short-circuit current can reach tens or even hundreds of times of the rated current. Therefore, if the ordinary reverse delay protection strategy is used, it will cause irreparable damage to the main circuit switch tube and the diode, and even damage the main circuit device by breakdown [7]. A high current surge will occur at the moment the circuit is turned on, but this current is a normal generated current with a time in the microsecond range, so it will not cause circuit damage. Delay protection measures need to abandon this state [8].

The specific requirements of the short-circuit protection function are as follows, when the control protector detects a short-circuit fault at the output terminal, it should delay the output of the short-circuit protection signal by 0.001s and disconnect the contactor.

According to the requirements of the overcurrent protection function, this module needs to realize the anti-delay protection and the state output holding module described above when the current is greater than the typical value in the overcurrent protection delay curve, and the model building is shown in Figure 6.

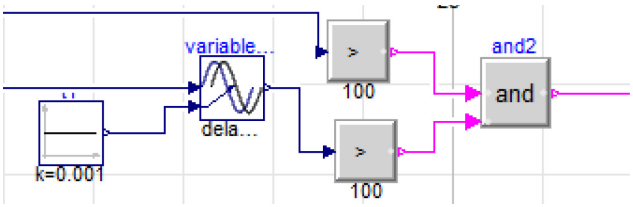
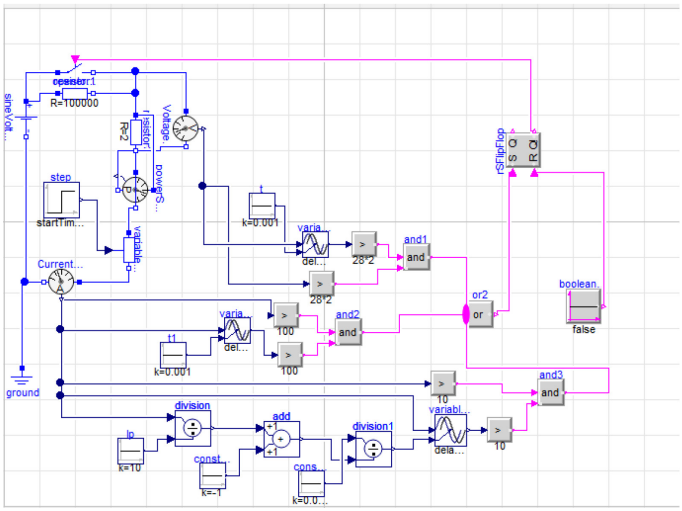


Figure 6. Short-circuit protection model

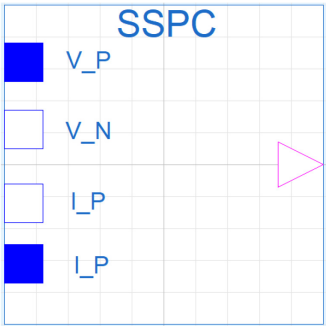
4 System modeling and simulation

4.1 System modeling

Connect the overvoltage protection, overcurrent protection, and short-circuit protection model modules that have been previously established and package them, as shown in Figure 7.



(a) Internal diagram of DC control protector model



(b) SSPC model encapsulation

Figure 7. Overall protection model

4.2 Simulation Results

Set up the test model shown in Figure 8, input voltage 28V, connect to the SSPC package model, and simulate and verify the overvoltage protection, overcurrent protection and short circuit protection functions of the SSPC. In the test model, the variable resistance plays the role of partial voltage. When 0.5s occurs, the resistance value of the variable resistor becomes 0. At this point, the voltage at both ends of the load increases, resulting in overvoltage or overcurrent, thus turning off the switch to achieve protection. The simulation results are shown in figure 9 below.

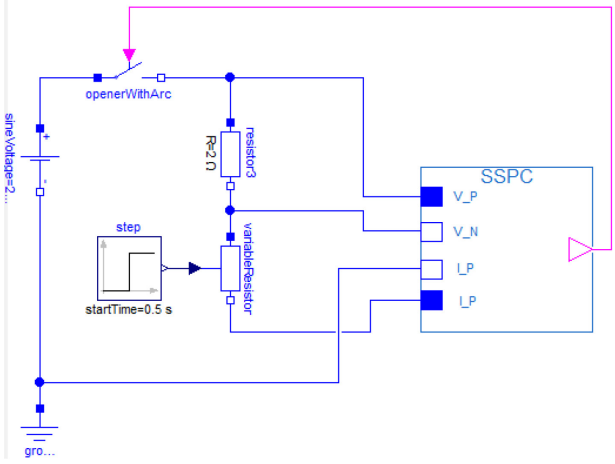
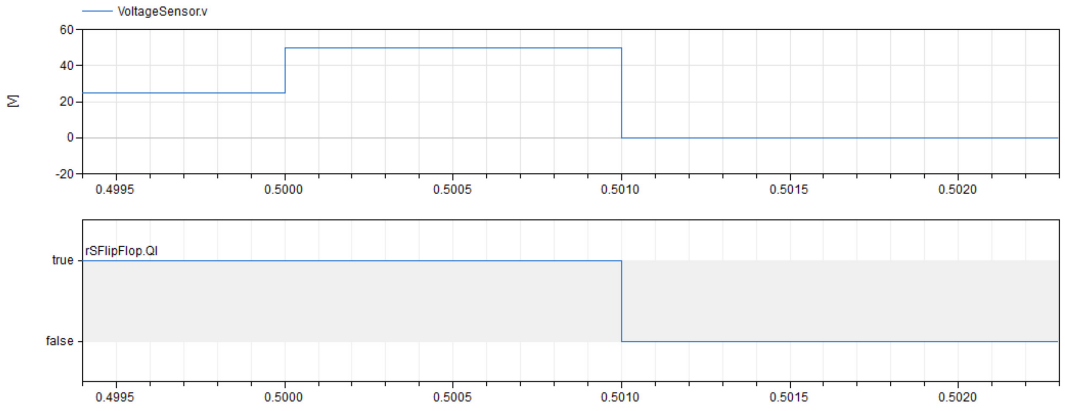
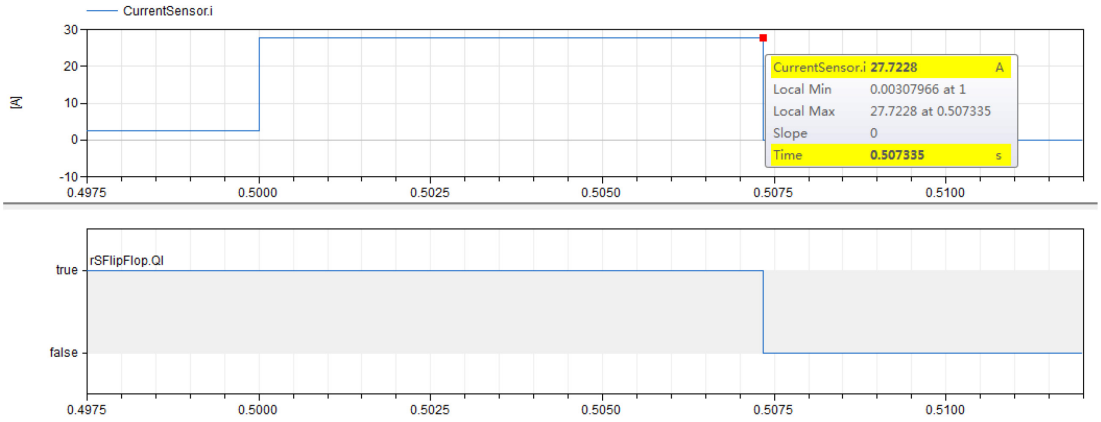


Figure 8. SSPC model test



(a) Simulation results of overvoltage protection



(b) Over-current inverse delay protection simulation results

DOI 10.3384/ecp2020174127

(c) Short circuit protection simulation results

Figure 9. Simulation results

From the simulation results, it can be seen that when the overvoltage protection and short circuit protection, the switch control signal delay 0.001s becomes a low signal, which meets the requirements, when the load current becomes 27.7A, and the control signal delay is 0.0073s to become a low signal, which realizes overcurrent reverse delay protection, as shown in Figure 9(b). Through the test of the building model, the correctness of the model is verified.

5 Conclusion

This paper builds a SSPC model based on the Dymola simulation software, and simulates its function of overvoltage protection, overcurrent protection and short-circuit protection. The correctness and accuracy of the model are verified. Compared with other simulation software, the model built in the Dymola simulation software is simpler and more convenient, which well demonstrates the characteristics of SSPC. Secondly, this model is universal and can be applied to different scenarios by modifying the corresponding parameters.

References

[1] Barrado A , Izquierdo D , Sanz M , et al. Behavioural Modeling of Solid State Power Controllers (SSPC) for Distributed Power Systems[C]// Applied Power Electronics Conference and Exposition, 2009. APEC 2009. Twenty-Fourth Annual IEEE. IEEE, 2009. doi:10.1109/APEC.2009.4802897.

[2] Izquierdo D , Barrado A , Raga C , et al. Protection Devices for Aircraft Electrical Power Distribution Systems: State of the Art[J]. *IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS*, 2011, 47(3):1538-1550. doi: 10.1109/taes.2011.5937248.

[3] Devinda A. Molligoda, Pradip Chatterjee, Chandana J. Gajanayake. Review of design and challenges of DC SSPC in more electric aircraft[C]// 2016 IEEE 2nd Annual Southern Power Electronics Conference (SPEC). IEEE, 2016. doi: 10.1109/SPEC.2016.7846117.

[4] Machado, José, Seabra, Eurico, Soares, Filomena. A new Plant Modelling Aproach for Formal VerificationE Purposes[J]. *Ifac Proceedings Volumes*, 40(9):167-172. doi: 10.3182/20070723-3-pl-2917.00027.

[5] Qiong-Zhong Chen,, Yu-Feng Mo,, Guang Meng,. Dymola-based modeling of SRD in aircraft electrical system[J]. Aerospace & Electronic Systems IEEE Transactions on, 42(1):220-227. doi: 10.1109/TAES.2006.1603417.

[6] M. W. Stavnes, A. N. Hammoud，Assessment of Safety in Space Power Wiring Systems[J]，*IEEE Aerospace and Electronic Systems Magazine*，1994，9：21-27. doi:

[7] Ahmed M M R, Mawby P A. Design specification of a 270 V 100 A solid-state power controller suitable for aerospace applications[C]// 2009.

[8] Liu Z Z, Gayowsky T J, Fuller R J, et al. SSPC technology incorporated with thermal memory effects to achieve the fuse curve coordination[J]. 2010. doi: US7706116 B2.

DOI
10.3384/ecp2020174

A Protocol-Based Verification Approach for Standard-Compliant Distributed Co-Simulation

Martin Krammer[1] Christian Kater[2] Clemens Schiffer[1] Martin Benedikt[1]

[1]Virtual Vehicle Research GmbH, Co-Simulation and Software Group, Austria,
{martin.krammer,clemens.schiffer,martin.benedikt}@v2c2.at
[2]Comsysto Reply GmbH, Germany, c.kater@reply.de

Abstract

The Distributed Co-Simulation Protocol (DCP) is a platform and communication system independent application level communication protocol. It is designed to integrate models or real-time systems into simulation environments. The specification document defines the structure and behaviour of slaves. It is advantageous to verify and validate a slave before integrating it into a larger co-simulation scenario. This is especially true if tests are performed on large rigs, where availability and time are typically limited. Until now, no systematic procedure for design, verification and validation of slaves is available.

In this paper, we introduce a process for design, verification and validation of DCP slaves. The process is used to systematically encapsulate models or real-time systems into slaves. For verification, the DCP state machine and protocol definitions are used to derive sequences of protocol data units (PDU) from any given DCP slave description. To demonstrate the feasibility of our approach, we show a use case from the automotive engineering domain. It includes a slave representing an engine, which is embedded using the Functional Mock-Up Interface (FMI). We also introduce the *DCP test generator* and *DCP tester* tools to automate the verification steps. With these contributions, slaves can be developed systematically and more efficiently. The introduced software is available under an open-source license.

Keywords: co-simulation, simulation, test, real-time, distributed

1 Introduction

Co-simulation-based methodologies have evolved significantly during the last decade. Nowadays co-simulation is a major enabler for holistic cross-domain or system simulations. It allows integration of simulation models, tools, and solvers from different sources.

The functional mock-up interface (FMI) (Blochwitz et al., 2011) represents an important software standard for co-simulation in several industry sectors. It was proposed to solve the need for interoperability between models, solvers and tools. FMI was developed in the MODELISAR project, starting in 2008. The FMI specification is standardized as a Modelica Association Project (MAP). Its most recent specification version is 2.0.1 which was released in 2019. The FMI specification document defines an interface for model exchange and co-simulation. Today more than 100 software tools support the FMI[1]. For distributed simulation environments, network communication technologies are frequently used in practice. However, "the definition of this communication layer is not part of the FMI standard" (Modelisar Consortium and Modelica Association Project "FMI", 2019, p.95).

The Distributed Co-Simulation Protocol (DCP) fills this technology gap. It was developed in the ACOSAR project (Krammer et al., 2016). The DCP is an application-level communication protocol designed to integrate models or real-time systems into simulation environments. It enables exchange of simulation related configuration information and data by use of an underlying transport protocol (such as UDP, TCP, or CAN). At the same time, the DCP supports the integration of tools and real-time systems from different vendors. The DCP is intended to make simulation based workflows more efficient and reduce the overall system integration effort. It was designed with FMI compatibility in mind, i.e., it follows a master-slave communication principle, uses an aligned state machine implementing an initialization mechanism, and defines an overall integration process which is driven by standardized XML file formats. Version 1.0 of the DCP specification document was released as an open-access Modelica standard in early 2019 (Krammer et al., 2018a, 2019).

2 Motivation

The DCP specification document describes the design of a slave only. A master is required to control a co-simulation scenario, which includes at least one slave. A slave encapsulates a model or real-time system. It therefore represents a simulation subsystem providing standardized access capabilities. The subsequent paragraphs motivate testing for slaves.

[1]http://fmi-standard.org/tools/

Development Support The initial need to automate protocol-based verification emerged out of the ACOSAR project. The specification document was engineered using a requirements-based approach (Krammer et al., 2018b). During development of the DCP new sets of technical requirements led to new specifications, which in turn raised demand for testing. First approaches included a tester tool able to send and receive predefined sequences of protocol data units (PDU) to and from a single slave. This fulfilled initial requirements for simple and configurable tests. Nevertheless that approach still demanded manual development of test cases, which caused high effort while suffering from low test coverage.

Protocol Implementation For full standard compatibility, DCP implementations must behave as specified and meet the standard requirements. The DCP is a platform and programming language independent specification. So it makes sense to test implementations at protocol level.

Application Specific Adaptation If a DCP slave encapsulates a real-time system, functional integrity is very important. On one hand, a DCP slave needs to behave as intended, e.g., connect to specific variables at the given communication step size. On the other hand, missing or delayed data packets, broken physical connections, etc. need to be handled properly if no other mechanisms are in place to avoid damage to equipment and operators. To handle these issues, the DCP features several mechanisms, like separate states for error handling and recovery. However, the criteria for transitioning to these states are application dependent and must be developed accordingly. Furthermore, all possible real-time system hazards must be analyzed, and potential safety measures implemented. Therefore, adaptations and safety measures are subject to test, in order to assure their functionality during operation.

Limited Access Access to industrial models or real-time systems might be limited, due to high capacity utilization or high rental cost. Typical examples include physical appliances like various kinds of automotive test beds, roller rigs, brake dynamometers, or driving simulators, to name a few. If DCP slaves encapsulate such systems, or are used in connection with such systems, it seems reasonable to test the DCP slave prior to scenario integration, in order to save time and money.

Economy For aforementioned reasons it seems reasonable to test DCP slaves with the goal to fix as many defects as early as possible. Delivering a reliable DCP slave is not only beneficial for the integrator, but also for the provider. From an economic perspective the overall number of development cycles is reduced and time-to-market is improved.

This paper contributes in two ways. Our goal is to describe a systematic development process for DCP slaves and how created DCP slaves can be tested prior to scenario integration. Therefore we introduce (1) a generic DCP slave development process and (2) a methodology and two coordinated tools for test of DCP slaves. Section 3 recapitulates related work. Section 4 describes the process main phases. Section 5 defines the main concepts of protocol based verification. Section 6 introduces the DCP Test Generator and Tester tools. Section 7 describes a use case from the automotive domain. Finally, Section 8 concludes this paper.

3 Related Work

A survey of communication protocol testing is presented in (Lai, 2002). It focuses on test sequence generation methods, test coverage, fault models and prediction, test tools, and experience reports. (Bochmann and Petrenko, 1994) provide a good overview of methods and their relevance for software testing. (Sidhu and Leung, 1989) give four formal methods for protocol testing by developing test sequences. (Linn, 1989) introduces a conformance evaluation methodology for protocol testing.

In the context of FMI, many efforts for testing functional mock-up units (FMU) have been made. The FMI Cross-Check repository[2] contains a large number of test FMUs provided by different vendors. The repository holds exported FMUs and results for imported FMUs of the tools that take part in this initiative. Therefore the contained FMUs can be used to test and improve the interoperability of FMI compatible import- and export-capable tools. The FMI Compliance Checker[3] is intended for FMU validation. It checks for FMI 1.0 and 2.0 compliance to the standard specification. Its basic features include XML model description checking and validation of binary FMUs. For the latter, it is able to load a binary module and check for the availability of all required functions. For an FMU for model exchange, the checker tool tests for explicit Euler numeric integration capability. For an FMU for co-simulation, the checker tool tests for fixed step-size calculation capability. Furthermore, the checker tool is able to provide numerical input data and log computed solution outputs. FMPy[4] is a Python library by Dassault Systèmes to validate and simulate FMUs. It has a graphical user interface, compiles C code FMUs and generates CMake projects for debugging.

To our best knowledge, a dedicated DCP test approach or tool is not available at this point in time.

[2] https://github.com/modelica/fmi-cross-check

[3] https://github.com/modelica-tools/FMUComplianceChecker

[4] https://github.com/CATIA-Systems/FMPy

DOI 10.3384/ecp2020174133

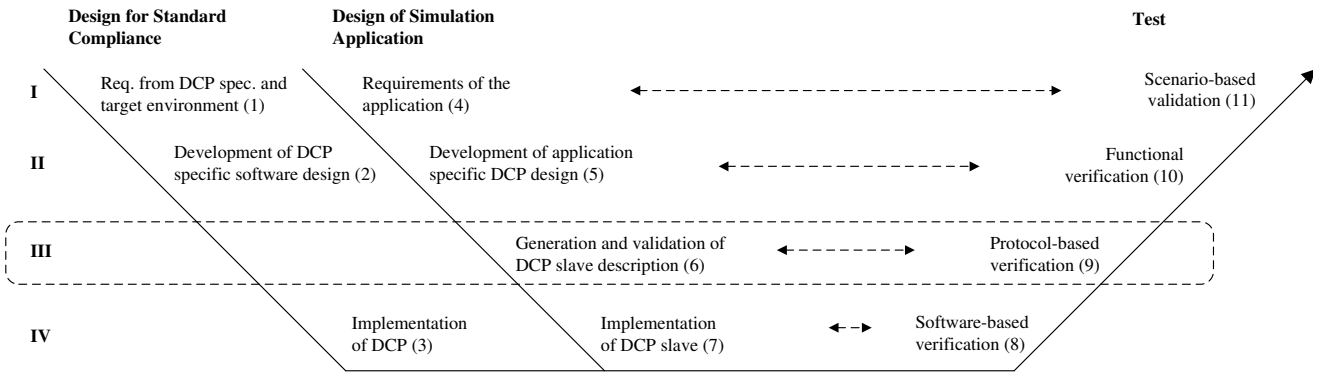


Figure 1. V-model for development, verification and validation of a DCP slave.

4 DCP Slave Development Process

4.1 Overview

We propose a V-model for development, verification and validation of DCP slaves. The classical V-model describes activities in development projects. It is traversed over time. Activities on the left-hand side are related to specification and design, whereas activities on the right-hand side are related to test. Each specification activity corresponds to one test activity. Figure 1 shows a V-model for design and test of a DCP slave. The V-model has four levels, ranging from requirements engineering and scenario (I), development and functions (II), data model and protocol (III), to implementation and software (IV). Its left side is split into two branches. Activities (1)-(3) are related to *Design for Standard Compliance*. They are used to implement the DCP according to the specification document, without any concrete simulation application. Activities (4)-(7) are related to *Design of Simulation Application*. They are used to tailor an existing DCP implementation according to the required simulation application. Its right side is dedicated to *Test*. Activities (8)-(11) are performed subsequently after implementation, to verify and validate the resulting slave.

Due to the structure of the V-model, it can be tailored to various needs. It can be used to implement the protocol first, and add the application specific parts afterward. The DCPLib represents an example of this approach. Alternatively, it might be desirable to have a monolithic implementation, where the simulation application directly implements standard-compliant behaviour. Examples include electronic control units (ECU) or microcontrollers providing DCP access. Subsequently, each of the V-model activities are described in detail.

4.2 Design for Standard Compliance

4.2.1 Requirements from DCP Specification and Target Environment

The DCP standard consists of a data model, a finite state machine, and a communication protocol including a set of protocol data units. The specification document defines how these parts interact with each other. These requirements are independent from an actual simulation application, and are mandatory for implementation. Additionally, requirements of the target environment will affect subsequent development. This includes e.g., the target operating system, preferred programming languages, available software modules or libraries, communication systems, and transport protocols.

4.2.2 Development of DCP Specific Software Design

This activity should clarify how to implement the DCP. The main result is a software architecture. DCP PDUs may be implemented using object-oriented concepts of classes and inheritance, in contrast to using pre-defined arrays. For send and receive functionality, an external application programming interface might be used. Depending on the target platform, concurrency can be exploited. For example, threads can be used to send and receive data at regular intervals.

4.2.3 Implementation of DCP

During this activity the DCP is implemented. The resulting work product of this activity is a compiled library or software module. It is not able to run on its own, as simulation application-specific information is missing.

4.3 Design of Simulation Application

4.3.1 Requirements of the Simulation Application

In this activity, the requirements of the simulation application are determined. This includes the identification of the simulation model or real-time system

for encapsulation, and the identification of quantities for simulation data exchange.

4.3.2 Development of Application Specific DCP Design

In this activity, several slave internal properties are defined. First, this includes the definition of variables and their causality (inputs, outputs, parameters and tunable parameters), mapping of these variables to variables of the encapsulated model or real-time system, and variable properties, like step size in combination with the slave's specified time resolution. Second, the finite state machine must be adapted to the behaviour of the encapsulated model or real-time system. Third, according to the specification, each state of `Normal Operation` requires a well-defined transition to the `Error Handling` state, where appropriate actions for recovery must be defined. Finally, if the intended slave should support starting simulations from a non-trivial initial condition, then an initialization/synchronization strategy must be elaborated.

4.3.3 Generation and Validation of DCPX

The DCP slave description (DCPX) is an XML (Extensible Markup Language) file which describes one single DCP slave. It contains all static information related to one specific DCP slave. Its structure is defined by a normative XML XSD (XML Schema Definition) file (Krammer et al., 2019). It not only defines the required structures of elements and attributes, but also supplementary assertions and constraints. Assertions and constraints are well suited for expressing logical relationships between elements and attributes.

Assertions are expressed in the `xs:assert` tag using the XML Path Language (XPath). They are a feature of XSD version 1.1. Assertions constrain the existence and values of related elements and attributes. Furthermore, `xs:unique`, `xs:key` and `xs:keyref` tags are used to express constraints. Typical examples of application include the verification of uniqueness of names and the verification of cross-referenced key values.

In the context of the DCP specification, assertions and constraints provide strong formalisms which can be used for automated validation, whenever a DCPX file is generated or imported.

4.3.4 Implementation of DCP Slave

In this activity the slave is implemented. As a typical resulting work product, an executable is created out of production code. A DCP slave is defined as either a simulation model or a real-time system on a ready-to-run execution platform that is accessible via DCP over a supported communication medium. Therefore the slave software might be deployed on its target platform.

4.4 Test

4.4.1 Software-based Verification

This activity is dedicated to software verification approaches. Procedures and processes for software verification are covered in (Rakitin, 2001; Myers et al., 2011). For example, inspections can be used to monitor the design for standard compliance.

4.4.2 Protocol-based Verification

This activity is situated between software-based and function-based verification. Details are presented in Section 5.

4.4.3 Function-based Verification

If protocol-based verification is successfully passed, testing may continue with function-based verification. Function-based verification refers to the functionality of the connection to the underlying model or real-time system. For that purpose, a dedicated master shall be able to configure and stimulate a slave's inputs and parameters by sending PDUs. This master must also observe the slave's response by monitoring its outputs. If defined, all variables must be within their thresholds, like minimum or maximum values. Furthermore, available dependency information can be used for testing as well. The outputs of a DCP slave may depend on its inputs and parameters.

4.4.4 Scenario-based Validation

In this activity, the developed slave is instantiated and integrated into the overall co-simulation scenario. A co-simulation scenario is defined as the integration of multiple DCP slaves to perform a common simulation task. A capable master is required, to perform registration, configuration, initialization, synchronization and the actual simulation.

5 Protocol-based Verification

5.1 Finite State Machine Based Approach

The main idea of protocol-based verification is to check if a DCP slave behaves according to the standard's specification document. This includes the communication protocol, the state machine and its transitions, and the slave's configuration. To check a slave for protocol compliance, the concept of a tester was developed. The tester should connect to a slave and stimulate it by sending PDUs. However, sending plain, pre-defined sequences of PDUs to slaves for protocol-based verification is not reasonable. A DCP slave may expose non-deterministic behaviour, due to e.g., network delay, affecting the protocol and state machine (Modelica Association Project DCP, 2019, p. 21). Figure 2 shows an example of such a situation. After a state change request from state CONFIGURED to state INITIALIZING, a slave acknowledges the request and performs the transition (steps 16 to

DOI 10.3384/ecp2020174133

19). In INITIALIZING it performs the necessary calculations, performs a self-triggered state transition to INITIALIZED and sends a notification including the new current state identifier (step 19 to step 20). Another possibility would be that the master decides to stop during the Initialization superstate. It sends STC_stop to the slave (step 19 to 21). The slave may acknowledge this request, and perform the state transition to STOPPING as requested (steps 21, 22, 23). However, if computation for initialization ends before STC_stop can be processed, the slave transitions to INITIALIZED and sends a corresponding notification. A negative acknowledgment including an error code follows (steps 21, 24, 25).

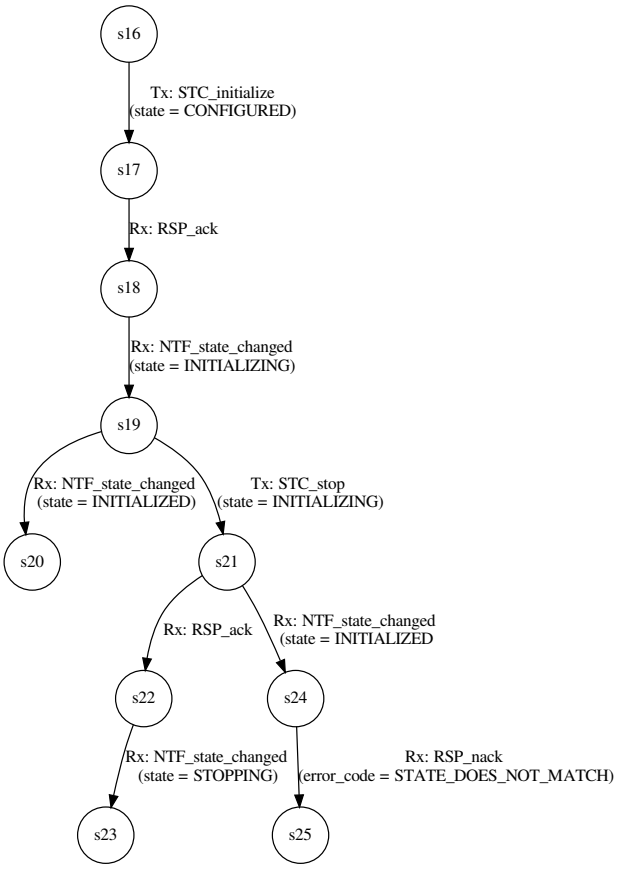


Figure 2. Example for multiple options to pass through the DCP state machine.

Protocol-based verification must consider such cases and react accordingly. That includes scalable time frames for request and response PDUs.

5.2 DCP Test Procedure

To circumvent the previously explained problem and to cover all possible situations, a DCP test procedure is defined. It is based on a finite automaton, which defines *steps* for protocol verification. Each step has a successor step. To proceed from step to step, transitions are defined. A transition can be triggered by either sending or receiving a protocol data unit (PDU). The DCP uses the concept of PDUs which are exchanged between DCP slaves and the DCP master. Since the DCP not only covers the exchange of simulation data (e.g., inputs/outputs, parameters), but also the set up (e.g., the configuration of inputs/outputs) and control (e.g., start and stop commands) of a co-simulation scenario, 34 different PDUs are organized in PDU families. A taxonomy is shown in (Krammer et al., 2018a). Certain PDU types are only meant to be sent one-way, from the sender to the receiver. For example, PDUs of the configuration family are only sent from the DCP master to the slave. Other PDU types may be sent in both directions. For example, PDUs of the data family may be used to exchange simulation data between slaves, but also between the master and a slave. The finite automaton of the DCP test procedure allows to define arbitrary PDU exchanges. Additionally it includes the concept of time to define a period after which a transition should be performed. Every finite automaton expressed using the DCP test procedure has defined steps for entry and exit. A valid step to exit the finite automaton is denoted here as an *accepting step*. A test procedure is successfully executed if the finite automaton is left through such an accepting step.

Formally, a DCP test procedure is defined by the following 6-tuple $(MaxStep, F, C, M, \delta, \lambda)$, where

- $MaxStep$ is the upper bound of steps in the procedure. The set of steps S is defined as

  $$S = \{n \mid n \in \mathbb{N}_0 \wedge n < MaxStep\}$$

  The initial step number equals to zero.

- F is the set of accepting steps. The test procedure is successful if it finalizes in an accepting step.

- C is a set of time points (unit: seconds) at which a transition from one step to another is performed.

- M is a set of PDUs according to the DCP specification which are sent or received within the test procedure.

- δ is a transition function between the steps of the test procedure:

  $$\begin{aligned}\delta &\subseteq S \times \Sigma \to S \\ \Sigma =&(\{Receive\} \times M) \cup \\ &(\{Send\} \times (C \cup \{-\}) \times M)\end{aligned}$$

- λ defines whether a transition should be considered in the statistical evaluation of the procedure:

  $$\lambda = S \times \Sigma \to \{true, false\}$$

5.3 Test Procedure Extension

Generating a generic test procedure is not feasible, because different slaves have different features which need to be considered. Furthermore, some parts of the test procedure will appear repeatedly. A typical example is to test if a slave correctly repudiates incorrect requests for state changes, in all of its states. Therefore extensions to DCP test procedures may be defined. An extension contains a description on how to extend a given test procedure, together with the DCP slave description of the slave under test. The main idea is to take a basic sequence through the state machine, modeled as a test procedure, and extend this sequence with slave specific aspects and repetitive parts. The main control elements to define an extension are:

- *ExtensionSet* contains a sequence of operations which has to be executed on every step which has transition with an `NTF_state_changed` PDU as predecessor, given by a certain state. The emerging transitions will be put in between the considered step and its successor. Algorithm 1 shows how this can be implemented in a generator for test extensions.
- *AddTransition* adds a transition between two steps. The transitions needs to be bound to a receiving or sending PDU, as well as the information that the occurrence of this transition shall be logged. The fields of the PDU can be specified by:
  - $Value$: a specific value for the field.
  - *Random*: a random, invalid value.
  - $Variable$: the value of the field is read from a variable.
  - *Invalid*: a invalid value for this field.
- *Update* overwrites the value of a given field of the PDU inside a transition. This can be used to update specific fields, which need to be valid in a transition. For example, the `STC_register` PDU is needed in a basic path through the state machine. But the UUID of the slave can not be known upfront. To solve this, *Update* can be used.
- $UpdateMaxStep$ Increases the MaxStep attribute by a given value.
- If allows conditional execution of control elements.
- $ForEach$ loops over a given set of items, like the states of the DCP protocol.

In addition to these control elements a test procedure extension can also access the DCP slave description to use and check against specific properties of the tested slave.

5.4 Test Procedure Algorithm

An algorithm to operate on the formal model introduced in Section 5.2 is given in pseudo-code in Algorithm 2. It is defined in two parts, intended to be run in parallel.

The first part is responsible for receiving PDUs. It waits for a PDU to arrive. If this PDU represents a valid transition to the next step of the finite automaton, then this transition is performed. If the next step is an accepting step, the algorithm successfully returns. A critical section is defined using the concept of mutual exclusion, in order to avoid interference between the sending and receiving parts. If the received PDU does not represent a valid transition to the next step, the algorithm aborts.

The second part of the algorithm is responsible for sending PDUs. If valid transitions to the next step of the finite automaton exist having a clock time assigned, the minimum remaining clock time is selected. If no clock time is assigned, a transition is selected at random. After that, the PDU corresponding to the selected transition is sent. The sending part also contains a critical section protected by mutual exclusion. If the current step represents an accepting step, the algorithm returns successfully.

6 Implementation Details

In this section we present two tools and supporting data formats, to enable protocol-based verification within the introduced DCP slave development process. This entire tool chain supports Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), both over Internet Protocol version 4. However, the introduced finite automaton, test procedure and data formats may be used in connection with other DCP-supporting transport protocols in the future as well.

6.1 Workflow

Figure 3 shows a workflow for the DCP test generator and DCP tester tools. The DCP test generator consumes the test scenario template, the test scenario extension, and the slave description for the slave-under-test (1). It produces a test scenario (2), which is handed over to the DCP tester. The DCP tester communicates with the slave-under-test by using a communication system and a chosen transport protocol. It performs the specified test procedure (3). The tester logs all transitions together with timestamps and corresponding PDUs (4). It generates logs that may be used for development or statistical evaluations. The slave-under-test may also generate a report.

6.2 Data Structures

To represent a test procedure an XML schema definition is used. Figure 4 shows a graphical represen-

DOI 10.3384/ecp2020174133

input: state: State after which entry the ExtensionSet shall be executed
foreach x *holds* $x \in S \wedge x \notin F \wedge \exists n \in \mathbb{N} : (n, (Receive, NTF\_state\_changed(state)), x) \in \delta$ **do**
  predX $= \{t | t \in \delta \wedge \exists n \in \mathbb{N} : t.from = n \wedge t.to = x \wedge t.\Sigma = (Receive, NTF\_state\_changed(state))\}$;
  sucX $= \{t | t \in \delta \wedge \exists n \in \mathbb{N} : t.from = x \wedge t.to = n \wedge t.\Sigma = (Receive, NTF\_state\_changed(state))\}$;
  entry = MaxStep;
  <execute sub elements>;
  foreach t *in* $predX$ **do**
    t.to = entry;
  end
  foreach t *in* $sucX$ **do**
    t.from = MaxStep - 1;
  end
end

Algorithm 1: Execution of a ExtensionSet

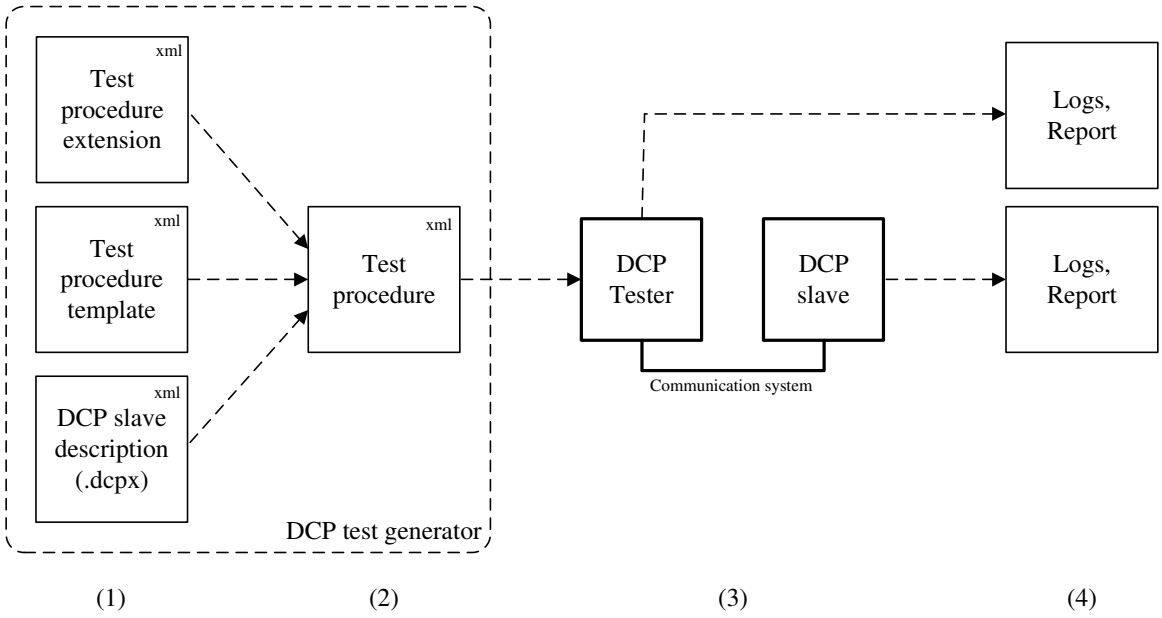


Figure 3. Workflow for DCP test generator and DCP tester tools

tation of this schema definition. It implements the introduced formal model from Section 5.2. A root element `DcpTestProcedure` is defined, which includes the set of sending and receiving `Transitions`. Furthermore, it also contains information for network `Drivers`, i.e., IP and port information for the tester tool.

6.3 Test Generator and Tester as Open-Source Software

The DCP test generator is written in Java. It is implemented as a command-line tool. Its main output is the test procedure. The DCP tester is written in C++. It is implemented as a command-line tool. The test generator[5] and the tester[6] are provided as open-source software, licensed under a BSD 3-clause license. The DCP specification document is maintained as a Modelica Association[7] Project (MAP). The Modelica Association is a non-profit, non-governmental organization with members from

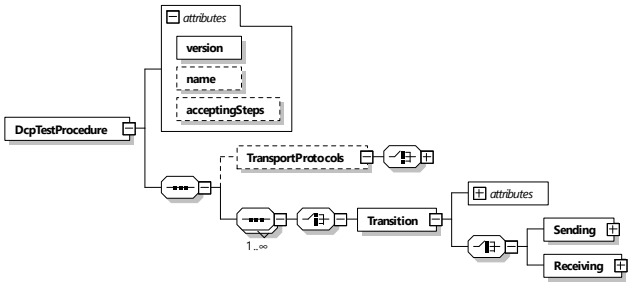


Figure 4. Test procedure data structure as logical view of an XML schema definition.

[5] https://github.com/modelica/DCPTestGenerator
[6] https://github.com/modelica/DCPTester
[7] http://www.modelica.org

```
1  step=0;
2  lastAction=0;
3  lastCheck=now();
4  foreach clocks for every sending transition with defined times stamp in test procedure do
5  |  clock(transition.sending) = transition.sending.numerator / transition.sending.denominator;
6  end
7  do in parallel
8  |  while wait for received PDU AND Receive PDU pdu do
9  |  |  lock();
10 |  |  if ∃ transition ∈ successor(step): transition.receiving = pdu;
11 |  |   then
12 |  |  |  if pdu.log then
13 |  |  |  |  Add PDU to statistics;
14 |  |  |  end
15 |  |  |  step = transition.to;
16 |  |  |  unlock();
17 |  |  |  if isAccepting(step) then
18 |  |  |  |  return 0;
19 |  |  |  end
20 |  |  else
21 |  |  |  unlock();
22 |  |  |  return 1;
23 |  |  end
24 |  end
25 end
26 do in parallel
27 |  while !isAccepting(step) do
28 |  |  lock();
29 |  |  <Update clock time of sending transitions by subtracting time between now and last update>;
30 |  |  transition = null;
31 |  |  if ∃ transition ∈ successor(step): transition.sending is defined AND
   |  |   transition.sending.clockTime is defined;
32 |  |   then
33 |  |  |  transition = <choose transition with min remaining clock time>;
34 |  |  |  if clock(transition.Sending) < 0 then
35 |  |  |  |  clock(transition.Sending) = transition.Sending.numerator /
   |  |  |  |   transition.Sending.denominator;
36 |  |  |  else
37 |  |  |  |  transition = null;
38 |  |  |  end
39 |  |  end
40 |  |  if transition = null AND ∃ transition ∈ successor(step): transition.sending is defined AND
   |  |   transition.sending.clockTime is not defined;
41 |  |   then
42 |  |  |  transition = <choose transition randomly>;
43 |  |  end
44 |  |  if transition != null then
45 |  |  |  <send PDU of transition>;
46 |  |  |  step = transition.to;
47 |  |  end
48 |  |  unlock();
49 |  end
50 |  return 0;
51 end
```

Algorithm 2: DCP tester algorithm

DOI 10.3384/ecp2020174133

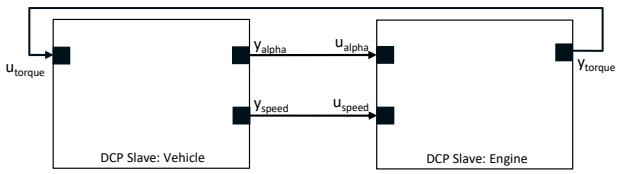


Figure 5. Co-simulation scenario implemented using DCP

Europe, North America, and Asia. Since 1996, its simulation experts have been working to develop the open standard Modelica and the open-source Modelica Standard Library. Today it aims at coordinated standardization, development of software technology, and corresponding methods in the fields of cyber-physical systems and systems engineering. The DCP specification is available as an open-access standard[8] licensed under a Creative Commons BY-SA 4.0 license.

7 Industrial Use Case

7.1 Description

In order to demonstrate the methods and algorithms outlined in this paper, a use case from the automotive domain was used. It consists of a co-simulation scenario including a vehicle simulation and an engine on a testbed. Both were implemented as separate DCP slaves, where the engine model was used as the slave-under-test. This slave has two input variables. The first input represents the accelerator pedal angle measured in degrees, the second represents the shaft speed in revolutions-per-minute. This slave has one output variable representing the shaft's torque in newton metres. It was implemented using standard compliant DCPLib. The intended co-simulation scenario is shown in Figure 5. Our goal is to perform a protocol based test of the DCP slave representing the engine (process level III), before it is functionally tested (process level II) and actually used in context of the shown co-simulation scenario (process level I).

7.2 Results

For this evaluation of protocol-based verification we consider the suggested test procedure template, several different test procedure extensions, and the DCP slave description file of the slave-under-test. First we executed a basic procedure without any extensions at all. The generated procedure passes straight through the different phases of the DCP state machine. Hence, this test procedure can be applied to any DCP slave. Second, the base template was extended to include more protocol tests. The generated procedure tests the slave's capability to deal with valid and invalid values for configuration. Third, a comprehensive extension set including configurations for data exchange was generated. However, all data PDUs for one unique `data_id` are counted only once. This was done to prevent that the same transition in the procedure is verified multiple times. Finally, the third procedure was augmented by including the heartbeat feature (Modelica Association Project DCP, 2019, p.62).

The DCP test generator was used to generate these procedures. The DCP tester was used to execute the test procedures. Table 1 shows the results. All test procedures were successfully evaluated and finalized in an accepting step. The column δ indicates the number of transitions contained in the test procedure, and *LoC* refers to the lines of code in the generated test procedure. Furthermore, the numbers of transitions are divided into numbers of corresponding sent and received PDUs. Based on the execution of the test procedures, we analyzed the numbers of *actually* sent and received PDUs. The last column shows the test procedure execution time, measured on a standard laptop device. All times measured include 4 seconds of simulation time in soft-real-time (SRT) mode. For the communication system, a local UDP socket-based configuration was used.

8 Conclusion

With protocol-based verification we introduce a novel test methodology for distributed co-simulation according to the DCP standard. In the associated DCP slave development process it fits between software- and function-based verification. Continuous protocol-based verification performed on a library designed for standard compliance, e.g., DCPLib, can potentially improve the quality of code over time. Both DCP slave providers and integrators can benefit from protocol-based verification, as they can shift verification activities to earlier phases of system development. The generation of test procedures using steps and transitions is advantageous over linear script based approaches. Extension sets for test procedures provide a scalable way to include new and more specific tests. The DCP test generator and the DCP tester are freely available under open source licenses.

The approach outlined in this paper is not intended as a means for exhaustive testing. A positive test result of protocol-based verification does not confirm that a DCP slave is fault-free and will never violate the specification document. Instead, it provides confirmation that the DCP slave-under-test is able to handle the actually executed sequence of sent and received PDUs. Technology-specific aspects like scheduling, network delay, jitter, etc. may cause non-deterministic behaviour during protocol-based verification.

Some extensions might also depend on the design of the DCP slave-under-test, indicated by e.g., capability flags. So a more modular approach for selection of extension sets could be desirable.

[8] http://www.dcp-standard.org

Test Procedure	δ	LoC	PDUs sent			PDUs received			Time [s]
			Specified	Actual	Ratio	Specified	Actual	Ratio	
Basic (no extensions)	62	325	16	15	0.94	46	37	0.80	10.19
Configuration (no data)	1925	11173	813	673	0.83	1112	695	0.63	10.30
Configuration (including data)	2369	13201	897	722	0.80	1472	747	0.51	12.70
Heartbeat (including data)	9053	49700	2764	742	0.27	6289	773	0.12	14.63

Table 1. Test procedures, their characteristics, and application results.

Acknowledgments

The work reported in this paper was conducted in the course of the ITEA3 Project ACOSAR (N°14004). It was partially funded by the Austrian Competence Centers for Excellent Technologies (COMET) program, the Austrian Research Promotion Agency (FFG), and by the German Federal Ministry of Education and Research (BMBF) under the support code 01lS15033A. The published versions of the DCP test generator and DCP tester tools were contributed by the Simulation & Modeling Group of Leibniz Universität Hannover.

References

Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, Christoph Clauss, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, Manuel Monteiro, Thomas Neidhold, Dietmar Neumerkel, Hans Olsson, Jörg-Volker Peetz, and Susann Wolf. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference*, pages 105–114, 03 2011. ISBN 978-91-7393-096-3. doi:10.3384/ecp11063105.

Gregor V. Bochmann and Alexandre Petrenko. Protocol testing: Review of methods and relevance for software testing. *Proceedings of the 1994 ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 1994*, pages 109–124, 1994. doi:10.1145/186258.187153.

Martin Krammer, Nadja Marko, and Martin Benedikt. Interfacing Real-Time Systems for Advanced Co-Simulation - The ACOSAR Approach. In Catherine Dubois, Francesco Parisi-Presicce, Dimitris Kolovos, and Nicholas Matragkas, editors, *STAF 2016 Doctoral Symposium and Projects Showcase*, pages 32–39, Vienna, Austria, 2016.

Martin Krammer, Martin Benedikt, Torsten Blochwitz, Khaled Alekeish, Nicolas Amringer, Christian Kater, Stefan Materne, Roberto Ruvalcaba, Klaus Schuch, Josef Zehetner, Micha Damm-Norwig, Viktor Schreiber, Natarajan Nagarajan, Isidro Corral, Tommy Sparber, Serge Klein, and Jakob Andert. The distributed co-simulation protocol for the integration of real-time systems and simulation environments. In *Proceedings of the 50th Computer Simulation Conference*, SummerSim '18, pages 1:1–1:14, San Diego, CA, USA, 2018a. Society for Computer Simulation International. URL `http://dl.acm.org/citation.cfm?id=3275382.3275383`.

Martin Krammer, Nadja Marko, and Martin Benedikt. Requirements engineering for consensus-oriented technical specifications. In *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, pages 315–324, Banff, Alberta, Canada, 2018b. ISBN 9781538674185. doi:10.1109/RE.2018.00039.

Martin Krammer, Klaus Schuch, Christian Kater, Khaled Alekeish, Torsten Blochwitz, Stefan Materne, Andreas Soppa, and Martin Benedikt. Standardized Integration of Real-Time and Non-Real-Time Systems: The Distributed Co-Simulation Protocol. In *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4-6, 2019*, volume 157, pages 87–96, Regensburg, Germany, Feb. 2019. Modelica Association. doi:10.3384/ecp1915787. URL `http://www.ep.liu.se/ecp/article.asp?issue=157%26article=9`.

R. Lai. A survey of communication protocol testing. *Journal of Systems and Software*, 2002. ISSN 01641212. doi:10.1016/S0164-1212(01)00132-7.

Richard J. Linn. Conformance Evaluation Methodology and Protocol Testing. *IEEE Journal on Selected Areas in Communications*, 7(7):1143–1158, 1989. ISSN 07338716. doi:10.1109/49.44561.

Modelica Association Project DCP. *DCP Specification Document, Version 1.0*. Modelica Association, Linköping, Sweden, 2019. URL `http://www.dcp-standard.org`.

Modelisar Consortium and Modelica Association Project "FMI". Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0.1, 2019.

Glenford J. Myers, Corey Sandler, and Tom Badgett. *The Art of Software Testing*. Wiley Publishing, 3rd edition, 2011. ISBN 1118031962.

Steven R. Rakitin. *Software Verification and Validation for Practitioners and Managers, Second Edition*. Artech House, Inc., USA, 2nd edition, 2001. ISBN 1580532969.

Deepinder P. Sidhu and Ting Kau Leung. Formal Methods for Protocol Testing: A Detailed Study. *IEEE Transactions on Software Engineering*, 15(4):413–426, 1989. ISSN 00985589. doi:10.1109/32.16602.

Towards an Open-Source Modelica Compiler in Julia

John Tinnerholm[1] Adrian Pop[1] Martin Sjölund[1] Andreas Heuermann[1] Karim Abdelhak[2]

[1]Department of Computer and Information Science, Linköping University, Sweden, {first.last}@liu.se
[2]Faculty of Engineering and Mathematics, Bielefeld University of Applied Sciences, Germany, {first.last}@fh-bielefeld.de

Abstract

Recently the Julia language has become an option for scientific computing. As of 2020, efforts exist to provide libraries that emulate the equation-based modeling features provided by Modelica or otherwise provide such functionality in Julia. The issue with these approaches is that investment in standardization and libraries would be lost unless standard-compliance is guaranteed. We believe that it is possible to combine features from both by implementing such a compiler in Julia. We argue that this approach would open additional opportunities. One such being the handling of variable structure systems (VSS) within the framework of a Modelica standard-compliant compiler. The other being a proposed compiler architecture reminiscent of LLVM for equation-based object-oriented languages. Using the OpenModelica Compiler as a baseline, we verified the fidelity of our implementation by simulating a selected set of models. While there are performance penalties, we argue that improvements to the frontend would mitigate these issues. *Keywords: Modelica, OpenModelica, Compilers, Applied computing, Julia, Variable Structure Systems*

1 Introduction

Cyber-physical Systems (CPS) are becoming increasingly complex every year. This trend increases the workload for modelers, and subsequently, requirements on associated tooling. Thus, tools need to scale and be flexible enough to handle increased complexity. These new requirements call for interdisciplinary cooperation between applied mathematics and computer science. One open problem in the area of modeling and simulation is how to handle specific categories of VSS (Utkin, 1977) in a way that adheres to the requirements of standard compliance and performance required by industry.

To our knowledge[1], Modelica only supports a limited subset of VSS through the usage of if-equations. Thus the set of VSS Modelica supports are limited to scenarios that can be described *a priori*. However, when modeling a system, it is not always possible to account for all changes with sufficient fidelity. An important but not a necessary component for a computational framework to support VSS is a JIT (Just-in-time) compiler.

Tinnerholm (2019) demonstrated that there is room for improving both the compile-time and runtime performance of OMC (the OpenModelica Compiler) and the possibility of JIT integration. Due to the similarities between Julia and MetaModelica (Fritzson et al., 2019b), a MetaModelica to Julia translator was developed (Tinnerholm et al., 2019) to examine alternatives to achieve JIT functionality. One such option is to use LLVM (Lattner and Adve, 2004), which was also investigated in (Tinnerholm, 2019). The other option was instead to generate Julia code. While automatically generated, Julia code in some cases had subpar performance compared to generating the LLVM intermediate representation (IR)[2] directly, the ease of code generation and the possibility of software reuse was promising. The problem of providing a standard-compliant compiler to deal with complex future requirements from industry motivated our continued investigation of Modelica-Julia integration.

The structure of this paper is as follows: First, we present the background of VSS in section 2. This section is followed by a discussion on providing a standard-compliant computational framework for VSS support section 3. This is followed by a technical overview together with initial verification of our Julia based Modelica implementation in section 4 and section 5. Related work is presented in section 6 and conclusions followed by future research directions in section 7.

2 Variable Structure Systems

The definition of VSS varies in literature since it defines a rather large class of systems. When the term system is used to discuss situations in which a model has a separate mode of functionality, *Multi-Mode DAE Models* (Benveniste et al., 2019) is sometimes used as a more precise description. The term used in (Höger, 2017) names variable structured systems with possible infinite sets of modes *Implicit Hybrid Automata*. The rationale being that there exist both explicit hybrid systems, which reefers to systems where the variability is specified *a priori* explicitly. When this behavior is not specified, the variability is implicit. Another term that captures the dynamics of these systems is *Structurally Dynamic Systems* (Giorgidze and Nilsson, 2009).

[1]As of August 7, 2020

[2]LLVM IR, the common currency used between components that constitute the LLVM Compiler Infrastructure.

In this paper, the term VSS is encompassing all classes of variable structured systems. We will refer to VSS with fixed variability (Pepper et al., 2011) as FVSS. It follows that support for general VSS includes FVSS. The compiler architecture we are proposing in this paper is meant to support the subset of VSS that may change without *a priori* information. We are thus attempting to enable future support for implicit hybrid automata through our proposed computational framework.

An early treatment of VSS was given by Utkin (Utkin, 1977). Utkin illustrates a simple variable structured system with the set of equations: eq. (1), eq. (2) and eq. (3).

$$\ddot{x} = -\psi x \tag{1}$$

$$\psi = \begin{cases} \alpha_1^2 \text{ where } x\dot{x} > 0 \\ \alpha_2^2 \text{ where } x\dot{x} < 0 \end{cases} \tag{2}$$

$$\alpha_1^2 > \alpha_2^2 \tag{3}$$

Systems such as these can be modeled in Modelica by the use of if-equations, see listing 1. However, although the structure varies, it is an FVSS. The behavior is statically encoded. Changes to the structure of the equations by means such as conditional declarations are, to our knowledge, not yet possible in standard Modelica (Zimmer, 2010; Höger, 2019).

Listing 1. The Modelica if-equation an example of a FVSS.

```
model M
  equation
  if <Condition> then
    <Equation 1>
  else
    <Equation 2>
  end if;
end M;
```

The handling of VSS is an active area of research and has been treated in (Neumayr and Otter, 2019; Elmqvist et al., 2017; Pepper et al., 2011; Höger, 2014; Mattsson et al., 2015; Höger, 2017; Giorgidze and Nilsson, 2009). A further summary is presented in section 6.

3 Towards a standard-compliant computational framework for Variable Structured System support

We have previously investigated the possibility of providing JIT support via the LLVM compiler infrastructure and compared that approach with using Julia generated code. In this section, we will discuss our initial findings when experimenting with LLVM in section 3.2. We will also present recent advances using the Julia programming language in section 3.3.

3.1 Architectural principles

There exists several computational frameworks to handle VSS (Zimmer, 2010; Höger, 2019; Elmqvist et al., 2017). While they differ somewhat in implementation, they all allow the compiler to be self-recursive, either through interpretation or JIT. In this section we briefly elaborate on two previously investigated alternatives, LLVM in section 3.2 and Julia in section 3.3

3.2 LLVM in the OpenModelica Compiler

LLVM (Lattner and Adve, 2004) is a compiler infrastructure currently employed both in the development of tools for static analysis and as a component for full-fledged compilers such as the Rust Compiler and Clang. The Julia project uses it for efficient code generation and JIT compilation. Recently, OpenModelica was extended with a new LLVM backend (OMLB) using an LLVM based JIT (Tinnerholm, 2019). While this effort did not cover all aspects of the Modelica language, it demonstrated possible benefits the OpenModelica environment would achieve by targeting LLVM, both in terms of compile-time and runtime performance.

3.3 The Julia language and extensions for equation-based modeling

The Julia language (Bezanson et al., 2012) is a new language for numerical computing. The Julia ecosystem supports many features required by a Modelica compiler, such as symbolic processing and interaction with other programming languages via its foreign function interface.

Modia is an example of a language extension to provide equation-based modeling in Julia to support VSS (Elmqvist et al., 2017). As noted by Fritzson et al. (2019b), an automatic translation of the OMC would provide the OpenModelica environment with these facilities without the need of re-implementing them. The disadvantage being that the development of MetaModelica would stall. Thus it would reduce the future capabilities of using Modelica to model the syntax and semantics of programming languages, a feature that is provided to Modelica with the MetaModelica extension (Fritzson et al., 2019a).

However, a Julia implementation would allow developers of OpenModelica access to the Julia ecosystem. We believe that such access would increase cooperation between the Modelica and Julia communities, aiding the Modelica effort by profiting from contributions to numerical computing from the Julia community and vice versa.

4 A Modelica frontend in Julia

The MetaModelica to Julia translator (Tinnerholm et al., 2019) was implemented in Susan (Fritzson et al., 2009). The translator (Tinnerholm et al., 2019) has been extended and has now been used to translate the oldest compiler frontend in OMC. In addition, a Julia based runtime has been developed along with a parser capable of parsing the

DOI 10.3384/ecp2020174143

Modelica Standard Library[3].

The parser uses the ANTLR3[4] based Modelica parser from OpenModelica and the external foreign function interface (FFI) that Julia provides to create and call Julia values and functions from C. The Julia representation of the Modelica abstract syntax tree[5] can be instantiated and translated to the existing representation of differential-algebraic equations.

The runtime consists of preprocessing constructs to ease translation, using Julia MetaProgramming, which is further expanded upon in section 4.1. It also re-implements the external runtime library that is used in OMC.

4.1 MetaModelica constructs via Julia-MetaProgramming

When writing a translator, it is key to make sure that the translated code is still readable. It is also desirable to handle specific language constructs existing in the target language.

Thus, for such a venture to be successful, it is vital that the languages are similar in paradigm and possess the same constructs. While there are some differences between MetaModelica and Julia (Fritzson et al., 2019b), Julia possesses constructs to mitigate these issues. The Julia macro system provides the users with the necessary tools to introduce new syntactical constructs to the language during compilation time. This means that even though Julia does not support inheritance[6], it is possible to implement it.

This property simplifies the process of automatically translating from MetaModelica to Julia since less logic is needed in the translator. The reason is that we can extend the Julia language with MetaModelica specific constructs and thus avoid generating extra code during the translation process. An example of how metaprogramming is employed for these purposes can be seen in listing 2. The listing depicts a translation of MetaModelica uniontype inside OMC see listing 3.

4.2 Performance of the existing frontend compared with the translated

The work described in this paper is a capable, but still an experimental prototype. Thus performance is not yet on par with MetaModelica based compiler even if the functionality (for the frontend) is the same. Currently, the memory usage of the abstract data structures (ADS's) remains approximately the same. However, translation time

Listing 2. Julia representation of Modelica equations in the Absyn IR.

```
@Uniontype Equation begin
 @Record EQ_IF begin
   ifExp::Exp # conditional expression
   equationTrueItems::IList # then
   elseIfBranches::IList # elseif
   equationElseItems::IList # else
 end

 @Record EQ_EQUALS begin
   leftSide::Exp # left hand side
   rightSide::Exp # right hand side
 end
 ...
end
```

Listing 3. MetaModelica metamodel of equations in OMC

```
uniontype Equation
  record EQ_IF
    Exp ifExp "Conditional expression" ;
    list<EquationItem> equationTrueItems "
        true branch" ;
    list<tuple<Exp, list<EquationItem>>>
        elseIfBranches "elseIfBranches" ;
    list<EquationItem> equationElseItems "
        equationElseItems Standard 2-side
        eqn" ;
  end EQ_IF;

  record EQ_EQUALS
    Exp leftSide "leftSide" ;
    Exp rightSide "rightSide Connect stmt"
        ;
  end EQ_EQUALS;
...
end Equation
```

when translating models from Absyn to SCode[7] into what is sometimes called flat Modelica[8] is currently far from optimal.

The reason for performance issues is due to differences between the covariant type system of MetaModelica and the invariant type system of Julia. This difference results in redundant creation and deletion of abstract data structures such as *list* and *arrays*.

Another reason is due to exception handling. The reason being that certain parts of the control flow are directed by the *matchcontinue* construct (Pop et al., 2019). In the compiler runtime[9], this construct is implemented as a macro that wraps the match macro in an exception handling block.

In MetaModelica, this is implemented with

[3] Used MSL v3.2.3
On Github: github.com/modelica/ModelicaStandardLibrary

[4] ANTLR3: ANother Tool for Language Recognition
www.antlr3.org

[5] The abstract syntax tree in the OMC is a hybrid between a concrete and abstract syntax tree. Absyn preserves parts of the original textual representation, which the SCode-representation lack.

[6] As to our knowledge; Julia version v1.3.1 (Dec 30, 2019)

[7] The SCode preferred internal representation before DAE IR generation.

[8] Flat Modelica is an intermediate stage in the compilation process, before symbolic manipulation of the equation system occurs. It is a set of DAE's with additional hybrid language constructs also remaining.

[9] MetaModelica.jl

`setjmp`/`longjmp` and a custom pass that removes them in contexts where they are not needed, or replaces a `longjmp` with a `goto` when jumping in a local context. At the time of writing, these kinds of optimizations are not provided.

4.3 Verification of OMFrontend.jl

To verify the fidelity of our Julia-based translated frontend, we simulated three Modelica models using the same settings. The same models were then simulated using the original OpenModelica environment (using the existing frontend, which the translated Julia one was based on). The result of these experiments is elaborated on further in section 5.2. From these experiments, we have reasons to believe that the translated frontend behaves correctly. We have also manually compared the DAE produced by the original OMC frontend and the Julia based one for the small HelloWorld model.

5 OMCompiler.jl

In section 4, we discussed how a Modelica frontend in Julia by means of automatic translation was provided. We briefly discussed the current capabilities and reasoned about the fidelity to OMC, which will be further elaborated upon in this section. While the performance of the new Julia based frontend is not on par with the existing frontend, we believe that these issues are possible to mitigate when the issues we brought up in section 4 are addressed. In this section, we will present the new open-source Modelica compiler `OMCompiler.jl`[10]. It consists of the various components listed in table 1, one of these being the previously discussed `OMFrontend.jl`. It is also supported by additional components such as `MetaModelica.jl` [11], which provides the compiler runtime. An overview of the inter-dependencies between them is illustrated in fig. 1.

In addition to the work that is presented here, we have already started the process of separating the different components that make up the compiler into different packages. In that way, we hope that future users of this work are able to utilize parts of our compiler as software components tailored for their specific use case. Further implications of this component-based design are discussed in section 5.4. An example of how a separate package can be used to simulate the exported output of the compiler design proposed here is provided in section 5.3.

5.1 An initial Modelica backend in Julia, OMBackend.jl

The compiler design proposed has the implication that a possible backend can be developed and used in separation. To assess the initial simulation characteristics of our proposed framework, a proof of concept backend `OMBackend.jl`[12] was developed, which is capable of simulating rudimentary DAE's. In this section, we will give a short introduction to a typical Modelica backend. We will also present our initial experiments to verify the results of both frontend and backend operations in section 5.2.

A Modelica compiler performs the following four steps on a flattened DAE provided by the frontend (Cellier and Kofman, 2006).

1. Pre-Optimization
2. Causalization
3. Post-Optimization
4. Code generation

5.1.1 Layout of OMBackend.jl

The purpose of the backend is to transform a given IR into a form suitable for simulation. The procedure to transform the given DAE IR into this form consists of:

1. Lower frontend into backend DAE IR:
   - Translate equations and variables to backend structure.
   - Detect and mark state variables.
   - Convert equations to residual equations.
2. Lower backend to simulation code IR:
   - Order variables into three categories:
     1) Derivatives of states
     2) States and algebraic variables
     3) Time-invariant parameters and constants
   - Index variables and store them in a hash table.
3. Generate Julia code by targeting `DifferentialEquations.jl`

To verify the correctness of the DAE IR generated by the frontend, three examples are presented, which can be studied in sections 5.2.1 to 5.2.3. While Julia currently remains the sole target language, it is possible to generalize and aim for different languages such as C (necessary for FMI).

5.2 Verification

To verify the behavior of `OMFrontend.jl` and `OMBackend.jl`, three examples are provided. The simulations of these examples where run both using the Julia based Modelica Compiler, `OMCompiler.jl` and using OMC. The result of simulating them are presented in sections 5.2.1 to 5.2.3.

[10] OMCompiler.jl
On Github: https://github.com/JKRT/OMCompiler.jl

[11] MetaModelica.jl
On Github: https://github.com/OpenModelica/MetaModelica.jl

[12] OMBackend.jl
On Github: https://github.com/JKRT/OMBackend.jl

DOI 10.3384/ecp2020174143

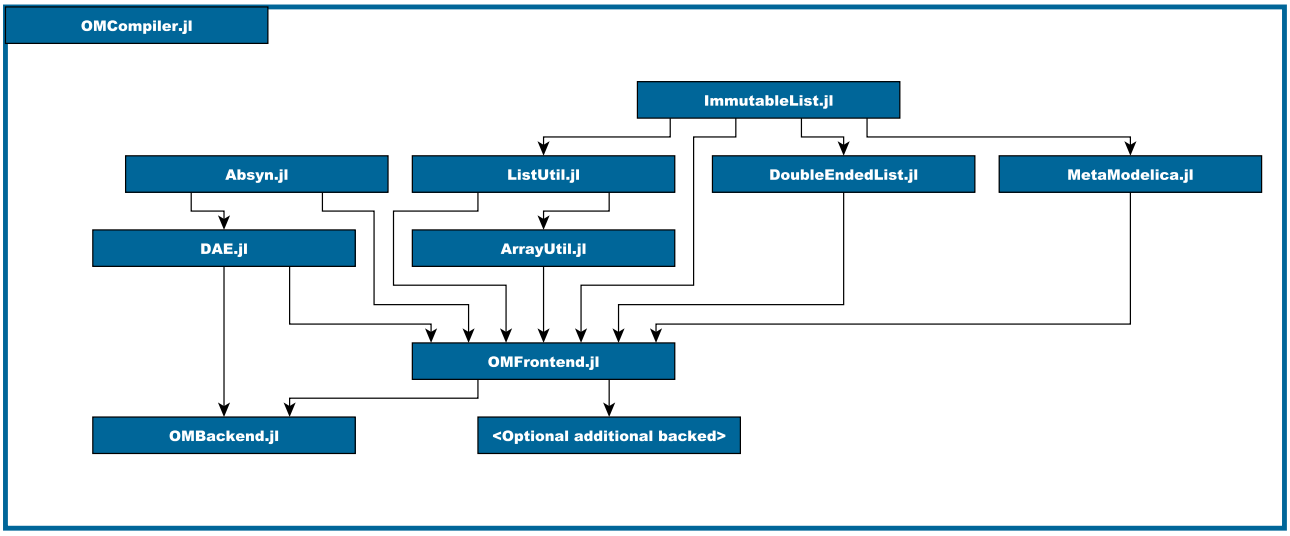


Figure 1. Overview of the inter-dependencies between packages that make up the proposed Julia based Modelica compiler.

Table 1. A brief description of the different components that make up `OMCompiler.jl`

Julia packages	Description
`Absyn.jl`	The concrete syntax tree
`ArrayUtil.jl`	Various utility functions for arrays
`DoubleEnded.jl`	Mutable linked list
`ImmutableList.jl`	Linked list support
`ListUtil.jl`	Various utility functions for lists
`MetaModelica.jl`	Compiler runtime
`Mutable.jl`	Mutable support
`OMBackend.jl`	Simulation and code generation
`OMFrontend.jl`	Lowers Modelica into DAE IR
`OpenModelicaParser.jl`	Standard compatible parser
`SCode.jl`	Implements SCode IR

5.2.1 Example: Hello World in Modelica

Our first example is the HelloWorld model presented in (Fritzson, 2015d) see listing 4 and listing 5 for the generated Julia code.

Listing 4. The Hello World model in Modelica (Fritzson, 2015a).

```
model HelloWorld
  Real x(start = 1);
  parameter Real a = 1;
equation
  der(x) = - a * x;
end HelloWorld;
```

Listing 5. Code generated for listing 4 by `OMCompiler.jl`.

```
function HelloWorldDAE_equations(res, dx,
    x, p, t)
  res[1] = ((dx[1]) -
    (((- (p[1] ))) * (x[1])))
end
```

5.2.2 Example: The Van der Pol Oscillator

This example demonstrates the results when compiling the Van der Pol oscillator (Fritzson, 2015d). The model shown in listing 6 was compiled using the different components of `OMCompiler.jl` with final code generation being conducted by `OMBackend.jl` and generates Julia code for simulation, as shown in listing 7. A plot illustrating the fidelity of the simulation with respect to OMC can be studied in fig. 3.

Listing 6. The Van der Pol oscillator model (Fritzson, 2015c)

```
model VanDerPol
  Real x(start = 2);
  Real y(start = 0);
  parameter Real lambda = 1;
equation
  der(x) = y;
  der(y) = lambda*(1 - x*x)*y - x;
end VanDerPol;
```

5.2.3 Example: The Lotka-Volterra equations

In this example, we present the result of generating code and simulating the Lotka-Volterra predator and prey model (Fritzson, 2015b). It models the relationship between prey and predators with respect to time. The result of simulating this using `OMCompiler.jl` and OMC can be seen in fig. 4.

5.2.4 Summary of experiments

As illustrated in figs. 2 to 4 the frontend together with the backend, can compile and simulate standard Modelica within the computational framework of Julia. Although

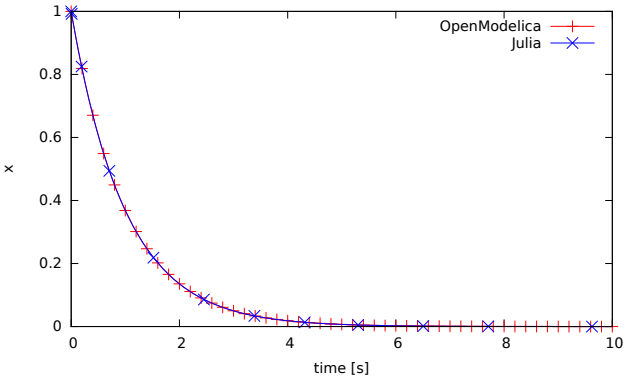


Figure 2. Julia and the corresponding OpenModelica simulation of listing 4 for 10 seconds using IDA.

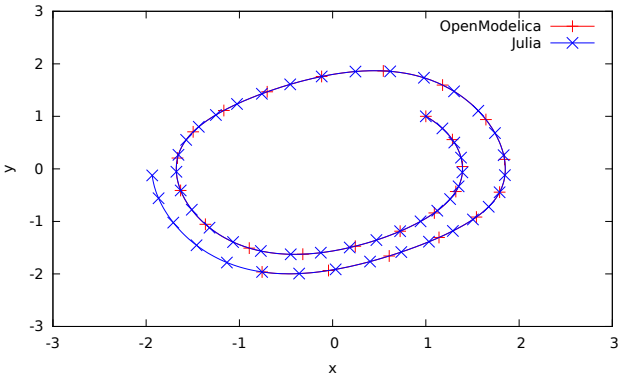


Figure 3. Julia and the corresponding OpenModelica simulation of listing 6 during 10 seconds using IDA.

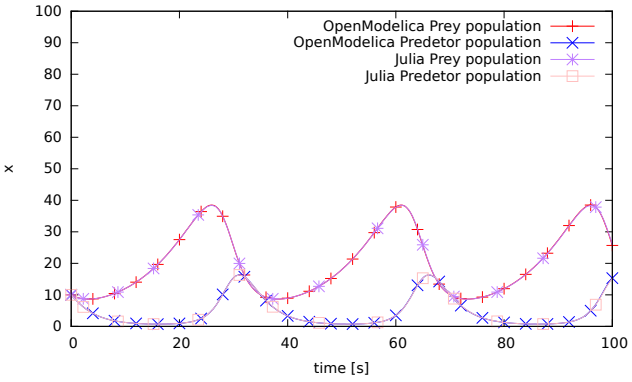


Figure 4. Julia and the corresponding OpenModelica simulation of the Lotka-Volterra model (Fritzson, 2015b) during 100 seconds using IDA

DOI
10.3384/ecp2020174143

Listing 7. DAE residual equation in Julia

```
function VanDerPolDAE_equations(
    res #= Residual vector =#,
    dx  #= State derivatives =#,
    x   #= States & alg. variables =#,
    p   #= Parameters =#,
    t   #= time =#)
  res[1] = dx[1] - x[2]
  res[2] = dx[2] - (p[1] * ((1.0
      - x[1] ^ 2.0 ) * x[2]) - x[1])
end
```

not all elements of Modelica, such as hybrid system support is as of this writing not supported, we believe that the flattened DAE generated by the frontend presented here is correct. This can be seen by inspecting the resulting graphs. We conclude that the models are simulating within acceptable numeric error tolerances.

Providing hybrid system support along with VSS support, however, would require extensions to the backend. The possibility of adding such support in Julia has already been demonstrated in the work concerning *Modia* (Elmqvist et al., 2017) and within *Haskell* (Giorgidze and Nilsson, 2009).

5.3 Performance of OMBackend.jl

Comparing the current state of `OMBackend.jl` with regards to memory and time during translation is not feasible. The reason being that the current capabilities of the backend are rudimentary. Thus, fewer operations are performed compared to more complete backends such as the one present in OMC. The same holds for the simulation runtime. While both OpenModelica and `OMBackend.jl` are using IDA, the event handling and capabilities to catch asserts of OpenModelica C runtime are more advanced and will, therefore, consume more memory and time. It would thus be hard to draw general conclusions from such an experiment, and it would, in some respect, give our work an unrealistic advantage.

5.4 A non-monolithic compiler

One of the major benefits of the LLVM compiler framework (Lattner and Adve, 2004) is that it is possible for developers to make use of the different components on their own.

Inspired by the LLVM effort, we decided to follow a design approach reminiscent of LLVM by instead of opting for a monolith compiler, to separate the different parts including intermediate representations such as `Absyn.jl` [13] and `SCode.jl` [14] into different components. We elaborate on the consequences of this design in section 5.5.

[13] Absyn.jl
On Github: github.com/OpenModelica/Absyn.jl

[14] SCode.jl
On Github: github.com/OpenModelica/SCode.jl

5.5 Benefits of component-based compiler

The benefits of providing a component-based Compiler are many. First and foremost the intermediate representations of the OMC along with the parser will be available as separate packages. This means that developers can use the parser and the different intermediate representations of the OMC for experimentation without having to change and modify the current monolithic compiler. This opens up many possibilities for increased cooperation and improved tool support. For instance, the backend is not necessarily restricted to be used for one language. By utilizing the DAE IR, it would be possible to devise a specification for another equation-based language and utilize the same backend for simulation purposes, see fig. 5. Analogously it would be possible to support several backends using the frontend presented here.

6 Related work

Several authors have proposed computational frameworks for the treatment of VSS within the context of equation-based languages. One of the first systems to successfully deal with structural variability was Mosilab (Nytsch-Geusen et al., 2005).

However, according to (Höger, 2019), Mosilab did not get traction and had certain drawbacks such as a combinatorial explosion of modes. Zimmer proposed a language *Sol* inspired by Modelica that handles structural changes to the model through symbolic processing at runtime via interpretation (Zimmer, 2010). Höger has a similar approach a few years later but additionally provides a method for dynamic index reduction that improves performance during structural changes (Höger, 2014). Höger (2017) claims that a few language extensions are sufficient to add VSS support to Modelica successfully. Höger (2019) presents a compiler prototype that handles variable structured systems with many modes.

What differs between the framework proposed in this paper and the computational framework suggested by Höger is two things. We base our prototype on an existing compiler, OMC, and we propose a component-based approach. The implication being that the compiler itself is not necessarily coupled to one language.

The Modia language extension to Julia also supports structural changes to the model at runtime (Elmqvist et al., 2017; Benveniste et al., 2019). The main difference between Modia and our approach is that Modia does not aim to be compliant with the Modelica standard nor evolving with it being an embedded domain-specific language for Julia rather than a compiler constructed using Julia.

An approach similar to Modia from a few years earlier is Hydra by Giorgidze and Nilsson (2009). Like Modia Hydra is embedded in a host language, in the case of Hydra the language is Haskell. Similarly to Modia Hydra does not implement Modelica. While both Hydra and the work presented in this paper internally make use of both IDA and LLVM. Hydra is a host language realized within

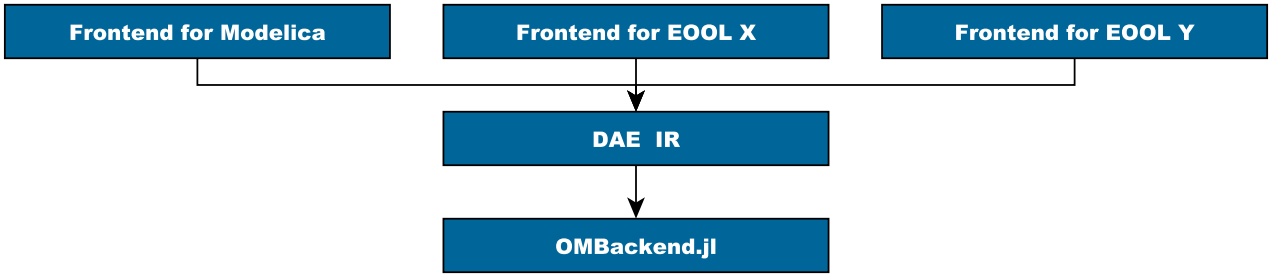


Figure 5. Separate frontends for equation-based languages utilizing the same backend.

Haskell and not a compiler, thus not as flexible in terms of providing support for new syntax and semantics. Still, Hydra provides support for dynamic hybrid systems, whereas at the time of writing `OMCompiler.jl` does not. A similar approach has also been proposed in Broman and Siek (2012). Consequently, the work presented in this paper can be seen as a complement to previous research, see how techniques pioneered before would fare when applied to existing Modelica models used in the industry.

At the time of this writing, the most extensive handling of differential equations in Julia is provided by `DifferentialEquations.jl` (Rackauckas and Nie, 2017). While not a standardized programming language comparable to Modelica, it is a capable package for simulating differential equations. `DifferentialEquations.jl` was selected as our final backend target.

7 Conclusions

In this paper, we presented an initial prototype of a Julia based Modelica compiler. The purpose was to provide flexible foundations for a computational framework capable of VSS support for Modelica while also adhering to existing standards. Thus, we propose an architecture reminiscent of LLVM for equation-based languages utilizing the Julia environment in combination with an automatically translated frontend to support this effort.

While there are performance issues, especially during the frontend phase of compilation, we believe that it is possible to overcome these issues. What we propose is an automatic translation of the new high-performance Modelica frontend in OMC Pop et al. (2019), which relies less on backtracking and outperforms the old frontend on which the frontend here is based.

We plan to investigate the results of Höger (2017), Benveniste et al. (2019), Giorgidze and Nilsson (2009), and Zimmer (2010) to evaluate how VSS support could be incorporated into this framework and consequently how VSS support would work in practice in a Modelica compiler. This raises further directions for future work such as examining what simplifications are possible in existing industrial-grade models.

Acknowledgements

This work has been supported by SSF in the LARGEDYN project. This work has also been supported by Vinnova in the ITEA EMPHYSIS project and the EMISYS project. Support from the Swedish Government has also been received through the ELLIIT project.

OpenModelica development is supported by the Open Source Modelica Consortium. Many students, researchers, engineers have contributed to the OpenModelica system. There is no room here to mention all these people, but we gratefully acknowledge their contributions. Additionally, we would like to thank Mahder Gebremedhin for providing feedback and advice concerning the backend design and the anonymous reviewers that provided additional feedback to this paper.

References

Albert Benveniste, Benoît Caillaud, Hilding Elmqvist, Khalil Ghorbal, Martin Otter, and Marc Pouzet. Multi-mode dae models-challenges, theory and implementation. In *Computing and Software Science*, pages 283–310. Springer, 2019.

Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

David Broman and Jeremy G Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-173*, 2012.

François E Cellier and Ernesto Kofman. *Continuous system simulation*. Springer Science & Business Media, 2006.

Hilding Elmqvist, Toivo Henningsson, and Martin Otter. Innovations for future modelica. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 693–702. Linköping University Electronic Press, 2017.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley-IEEE Press, 2 edition, April 2015a. ISBN 978-1-118-85912-4. pg. 20.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*.

Wiley-IEEE Press, 2 edition, April 2015b. ISBN 978-1-118-85912-4. pg. 841-842.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley-IEEE Press, 2 edition, April 2015c. ISBN 978-1-118-85912-4. pg. 22-23.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley-IEEE Press, 2 edition, April 2015d. ISBN 978-1-118-85912-4.

Peter Fritzson, Pavol Privitzer, Martin Sjölund, and Adrian Pop. Towards a text generation template language for Modelica. In Francesco Casella, editor, *Proceedings of the 7th International Modelica Conference*, pages 193–207. Linköping University Electronic Press, September 2009. ISBN 978-91-7393-513-5. doi:10.3384/ecp09430124.

Peter Fritzson, Adrian Pop, Martin Sjölund, and Adeel Asghar. MetaModelica – A Symbolic-Numeric Modelica Language and Comparison to Julia. In Modelica'2019. doi:10.3384/ecp19157289.

Peter Fritzson, Adrian Pop, Martin Sjölund, and Adeel Asghar. Metamodelica–a symbolic-numeric modelica language and comparison to julia. In *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*, number 157. Linköping University Electronic Press, 2019b.

George Giorgidze and Henrik Nilsson. Higher-order non-causal modelling and simulation of structurally dynamic systems. In *Proceedings of the 7th International Modelica Conference; Como; Italy; 20-22 September 2009*, number 043, pages 208–218. Linköping University Electronic Press, 2009.

Christoph Höger. *Compiling Modelica : about the separate translation of models from Modelica to OCaml and its impact on variable-structure modeling*. Doctoral thesis, Technische Universität Berlin, Berlin, 2019. URL `http://dx.doi.org/10.14279/depositonce-8354`.

Christoph Höger. Dynamic structural analysis for daes. In *Proceedings of the 2014 Summer Simulation Multiconference*, SummerSim '14, pages 12:1–12:8, San Diego, CA, USA, 2014. Society for Computer Simulation International. URL `http://dl.acm.org/citation.cfm?id=2685617.2685629`.

Christoph Höger. Elaborate control: variable-structure modeling from an operational perspective. In *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 51–60, 2017.

Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*, page 75. IEEE Computer Society, 2004.

Sven Erik Mattsson, Martin Otter, and Hilding Elmqvist. Multimode dae systems with varying index. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 89–98. Linköping University Electronic Press, Linköpings universitet, 2015. doi:10.3384/ecp1511889.

Modelica'2019. *Proceedings of the 13th International Modelica Conference*, March 2019. Modelica Association and Linköping University Electronic Press.

Andrea Neumayr and Martin Otter. Algorithms for component-based 3d modeling. In *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*, number 157, page 10. Linköping University Electronic Press, Linköpings universitet, 2019.

Christoph Nytsch-Geusen, Thilo Ernst, André Nordwig, Peter Schneider, Peter Schwarz, Matthias Vetter, Christof Wittwer, Andreas Holm, Thierry Nouidui, Jürgen Leopold, et al. Mosilab: Development of a modelica based generic simulation tool supporting model structural dynamics. In *Proceedings of the 4th International Modelica Conference TU Hamburg-Harburg*, volume 2, 2005.

Peter Pepper, Alexandra Mehlhase, Ch Höger, and Lena Scholz. A compositional semantics for modelica-style variable-structuremodeling. In *Proceedings of the 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools; Zurich; Switzerland; September 5; 2011*, number 056, pages 45–54. Linköping University Electronic Press, 2011.

Adrian Pop, Per Östlund, Francesco Casella, Martin Sjölund, and Rüdiger Franke. A New OpenModelica Compiler High Performance Frontend. In Modelica'2019. doi:10.3384/ecp19157689.

Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

John Tinnerholm. An LLVM backend for the Open Modelica Compiler. Master's thesis, Linköping University, Department of Computer and Information Science, 2019. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-154291`.

John Tinnerholm, Martin Sjölund, and Adrian Pop. Towards introducing just-in-time compilation in a modelica compiler. In *Proceedings of the 9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools*, pages 11–19, 2019.

Vadim Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic control*, 22(2):212–222, 1977.

Dirk Zimmer. *Equation-based modeling of variable-structure systems*. ETH Zurich, 2010.

DOI
10.3384/ecp2020174

The DLR Robots library – Using replaceable packages to simulate various serial robots

Tobias Bellmann[1] Andreas Seefried[1] Bernhard Thiele[1]

[1]Institute of System Dynamics and Control, German Aerospace Center (DLR), Germany,
{firstname.lastname}@dlr.de

Abstract

In order to simulate different kinds of serial robots, the implementation of functionalities such as the calculation of their direct and inverse kinematics, visualization, collision behavior, etc. is necessary. However, providing these functionalities in robot specific models leads to additional modeling overhead in cases where one would like to switch between several different robot models. The DLR Robots library demonstrates an implementation of all robot specific components as replaceable Modelica packages, allowing for an user-friendly way to exchange robot models without modifying the general structure of the overlying model.

Keywords: robots, replaceable packages, path-planning, inverse kinematics, LUA scripts

1 Introduction

The simulation of robotic systems is a great example for the multi-domain versatility of Modelica, combining multi-body mechanics with controllers, electric drives and algorithms, e.g. path-planning. A multitude of scientific works uses Modelica to simulate a specific robot, providing models for the mechanics and all other components exclusively for the simulated robot model (Kazi et al., 2002; Hirzinger et al., 2005; Dwiputra et al., 2014; Brossog et al., 2014). Other approaches use parameter sets to simulate more than one robot model within a given structure (Reiner, 2011). However, both approaches lack flexibility regarding switching between different robot types in a Modelica model, e.g. if the number of axes is changing. In this case, instead of changing a parameter value, the complete model structure has to be altered and adapted for the new robot model. In this paper, a new approach to model robots in Modelica is presented. By separating the robot functionalities (e.g. visualization, dynamics, path-planning, etc.) from the model-based description of the robot itself, and wrapping the latter in a replaceable package, it becomes possible to switch between entirely different robot models without the need for changing the structure of the main model. This approach is inspired by the Modelica Media library (Casella et al., 2006), where different media provide their own functions and models, also wrapped in a replaceable package, enabling the user to switch easily between different media in a model.

2 Structure of the library

The library is structured into the following sub-packages shown in Figure 1. The functional blocks e.g. for the calculation of direct or inverse kinematics, visualization functionalities, dynamic models, path-planning or collision detection can be found under `Robots.Blocks`. The sub-package `Robots.RobotModels` mainly contains the base class for the replaceable package `baseRobotModel` defining all common properties, functions and records. The implementation of two fictive robot models demonstrates the usage of this base class. A virtual robot controller is available in the `Robots.Controllers` sub-package, allowing to interpret robot programs and generate reference trajectories for the robot models. The `Utilities` and `Functions` sub-packages provide helper models used in the blocks and examples.

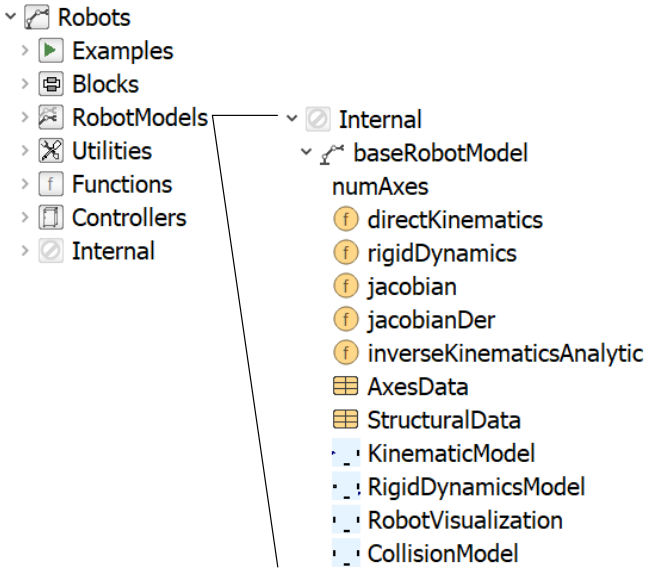


Figure 1. Overview over the DLR Robots library and the partial baseRobotModel package

2.1 Structure of a robot package

All specific robot models extend from the aforementioned base package `baseRobotModel`. This partial package defines the basic sub-models and their interfaces which are required for a robot to function within this library. Figure 1 shows the base package components to be overloaded

and defined by the user implemented specific robot model:

Constant `numAxes`: The integer constant `numAxes` defines the number of axes and is an important parameter for all other blocks, defining their input / output dimensions.

Overloaded functions: For certain functionalities in algorithms (e.g path-planning algorithms), a robot must provide several Modelica functions: the `directKinematics` function calculates the Cartesian pose consisting of `r[3]` (position) and `T[3,3]` (orientation matrix) of the robots end-effector from its joint angles `q[numAxes]`:

```
(r,T) = directKinematics(q)
```

The `rigidDynamics` function returns the torques `tau[numAxes]`, the mass inertia matrix `M[numAxes,numAxes]` and the additional torques generated by the Coriolis and gravitational forces `tau_aux[numAxes]`:

```
(tau,M,tau_aux) =
    rigidDynamics(q,qdot,qddot)
```

In case a linearization of the direct kinematics is necessary, the Jacobian and its derivative have to be provided as well:

```
J = jacobian(q)
J_der = jacobianDer(q,q_dot)
```

In some cases, the inverse kinematics of a serial robot can be calculated analytically. The function `inverseKinematicsAnalytic` can be used to define it:

```
q = inverseKinematicsAnalytic(r,T)
```

Parameter records: Every robot model has to provide two records to define its basic mechanical and dynamic parameters. The record `AxesData` contains all axes related data, e.g. minimum and maximum joint angles/extensions, maximum joint velocities and accelerations, as well as torque limits or gear transmission ratios. The `StructuralData` record should be used to define the mechanical dimensions of the robot, e.g. the distances between the robot joints as well as the position of the centers of gravity of the single robot components. The `StructuralData` record provides no strict naming convention to be overloaded, but is intended to be used freely by the robot model designer to hold all structural information of the robot.

Overloaded models: Every user defined robot model package extending the `baseRobotModel` package also includes a number of models to be used by the functional blocks. The `KinematicModel` model uses standard MultiBody components as joints, rotations and translations to build the kinematic chain between the robots base and its tool center point (TCP). The coordinate frames of every robot axis are provided as an array of MultiBody frames ()`frame_axes`).

The `RobotVisualization` model encapsules the visualization components of the DLR Visualization library used for the real-time visualization of robot systems. It does not contain a kinematic structure, but attaches a visualizer block to every frame of every axis, provided by the `frame_axes` input.

The `RigidDynamicsModel` model uses standard Modelica MultiBody components to model a rigid dynamics model of the robot, including masses and inertias of the robot arm. Similar to the `KinematicModel`, MultiBody joints are used to build the kinematic, but this model also describes the robot components' masses (See Figure 2) and therefore allows the calculation of the dynamic forces acting them.

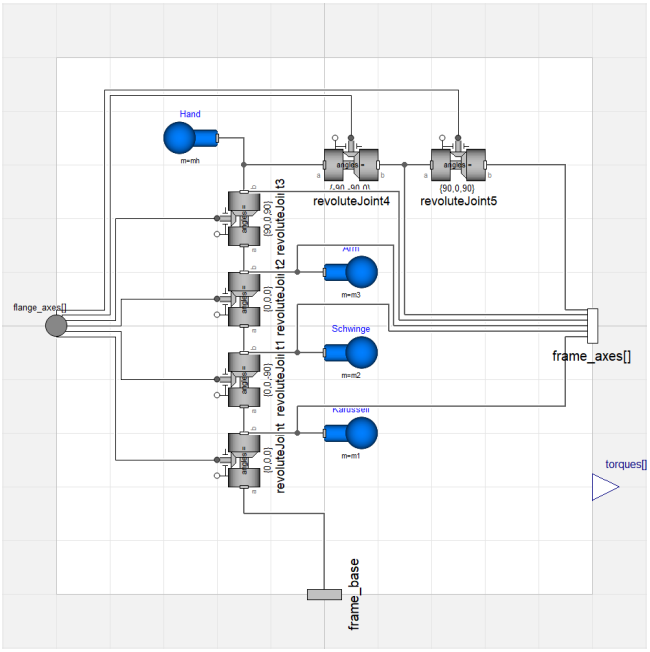


Figure 2. Example for a simple rigid dynamics model for a six axes industrial robot

The `CollisionModel` provides an interface to the DLR ContactDetection library, used to calculate collisions between the robot, itself and other components in its reach. Like the `RobotVisualization` block, the `CollisionObject` blocks modeling the shape of the robot are attached via the connector array `frame_axes`.

2.2 The robotChoice partial model

In order to exchange the robot package in a functional model, the base class `Internal.robotChoice` is provided as an interface definition to parameterize the robot model:

```
partial model robotChoice
"A base class providing the dialog option
    to choose the robot model."
replaceable package robotModel =
    Robots.RobotModels.BelloBot
    constrainedby
Robots.RobotModels.Internal.baseRobotModel
    "Robot model to be simulated"
annotation (choicesAllMatching=true);
...
end robotChoice;
```


DOI 10.3384/ecp2020174153

Within the functional block extending this partial interface definition, the robot package can now be chosen and used with the parameter `robotModel` (see Figure 3).

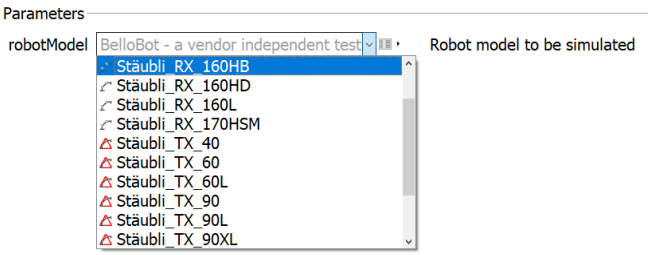


Figure 3. Dymola parameter dialog showing the selection of the robot model

For example, if the user requires access to the maximum axis acceleration, this could be achieved by the following code:

```
extends Robots.Internal.robotChoice;
Real q_ddot[robotModel.numAxes];
...
q_ddot = robotModel.AxesData.q_ddot;
```

3 Utilizing functional blocks to build models with serial robots

The DLR Robots library provides a multitude of robot model independent functionalities. These functional models from the subpackage `Robots.Blocks` utilize the aforementioned `robotChoice` parameter to select the robot model to be used and therefore allow the user to design robot model independent simulation models. Figure 4 shows some of the available blocks of this sub-package.

Visualizer blocks: The `Visualizer` blocks allow the user to visualize the simulated robot with the DLR Visualization library.The DLR Visualization library is used to provide real-time visualization of the robot and its surroundings.

Direct kinematics blocks: Blocks from the package `DirectKinematics` enable access to the axes limits (`AxesLimits` model) or provide models wrapping the robots direct kinematics, Jacobian and its derivative. The user can define additional TCP transformations in order to account for different tool center points.

Inverse kinematics blocks: The `InverseKinematics` package holds several algorithms to calculate the robots inverse kinematics. If an analytical inverse kinematics is provided by the selected robot model package, the `AnalyticalInverseKinematic` model can be used, which simply wraps the function `robotModel.inverseKinematicsAnalytic`. The `DampedLeastSquares` block calculates the inverse kinematics numerically via the well-known damped least squares algorithm (Wampler, 1986). It uses the

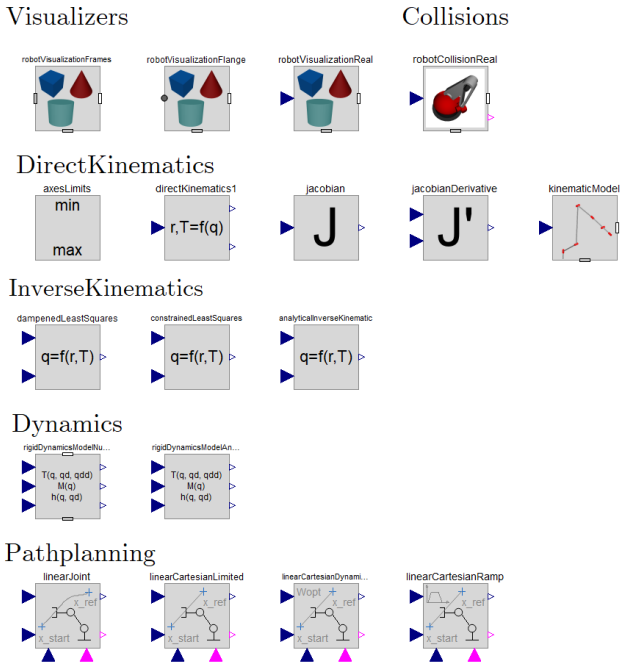


Figure 4. A selection of functional block, to be parameterized with the desired robot package

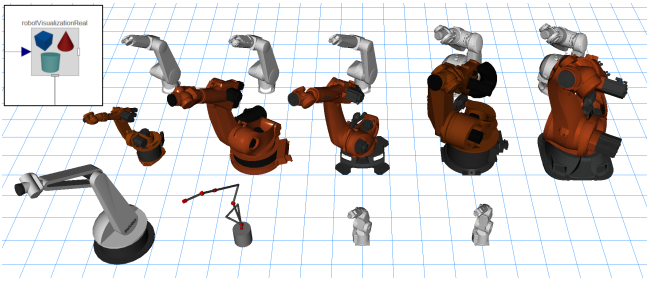


Figure 5. The same visualization block is used with different robot packages selected, visualizing Stäubli, KUKA, Mitsubishi and fictional robots

`directKinematics` and `jacobian` functions from the selected robot package to calculate the joint angles to a given end-effector position via an iteration loop. The `ConstrainedInverseKinematics` model implements the algorithm from (Bellmann et al., 2011b) to generate the joint trajectory resulting in a Cartesian movement of the end-effector, where the reference pose is followed as closely as possible. Here, joint limitations such as minimum and maximum joint angles, velocities, accelerations and torques are considered. In every time step of the calculation, the optimal change of the joint angles Δq is calculated via a local optimization step. The optimization criterion demands a minimization of the pose error, but can also contain additional sub-criteria, for example the desired configuration of redundant robot kinematics with more than six axes.

Robot dynamics blocks: The `Dynamics` subpackage contains models to calculate the robot joint torques/forces, mass inertia matrices and forces act-

ing on the robot structure, either by encapsuling the `robotModel.rigidDynamics` function or utilizing the `RigidDynamicsModel` to calculate them numerically.

Path-planning blocks: The DLR Robots library provides several path-planning algorithms intended to generate e.g. point-to-point (PTP) trajectories between two poses. The `LinearJoint` and the `LinearJointAxes` blocks generate time-optimal PTP trajectories between two poses by interpolating between them in joint space. Maximum joint velocities and accelerations from `robotModel.AxisData` are also considered. The `LinearCartesianLimited` and `LinearCartesianDynamicProgramming` blocks also generate PTP trajectories between two poses, but always on a straight line, interpolated in Cartesian space. The first one is a very fast two-pass algorithm considering the velocity and acceleration limits, but the resulting trajectory can exceed them in the vicinity of numerical singularities ($||J|| < eps$). The second one implements a Dynamic Programming approach to calculate the joint trajectories in a time optimal way, always considering the axes' dynamical limits.

The `RealtimeMovement` block is basically an inverse kinematics block to be used for sensor guided reference trajectories, where a reference pose trajectory is given and the joint angles have to be calculated accordingly and checked for feasibility.

The `TeachBox` block calculates simple and slow movements around the Cartesian axes, in order to implement the possibility to control the robot by a classical teaching interface.

All PTP path-planning blocks can be triggered via a Boolean input. Upon activating the trigger, the current reference pose `(r_ref, T_ref)` is then stored as the target pose, and the movement from the current robot configuration `q_robot` starts. After finishing the movement, the Boolean output `finished` becomes true, indicating the completion of the movement.

4 Using LUA scripts as robot programs

In the real world, a robot program defines a sequence of robot operations, for example different kind of movements, activation of tools, etc. Robot programs must be able to react to external signals, e.g. from a high-level programmable logic controller, or sensors. Nearly every manufacturer provides its own language for their robots, for example KUKA KRL or Mitsubishi MELFA Basic. In order to program a simulated robot in Modelica, different methods can be thought of, for example using Modelica state machines, Modelica functions generating reference positions over time or Modelica models to be parameterized with the reference trajectories and operation activities. However, all these pure Modelica implementations generate significant overhead and require a recompilation of the model if the program has to be changed. In the DLR Robots library, a different approach has been chosen. By utilizing the Modelica/C interface and the external object mechanism, a LUA[1] interpreter can be coupled with the Modelica robot model, using a LUA script as robot program. The flexibility of LUA allows the user to exploit all aspects of a high-level scripting language such as variables, mathematical operations or a feature-rich command-set, but also the usage of custom designed commands tailored to the needs of robot programming, e.g. for PTP movement commands.

4.1 Coupling the LUA interpreter with Modelica

The components of the LUA/Modelica conglomerate are shown in Figure 6. One of the advantages of LUA is the very sleek implementation of its interpreter in C, consisting only of a handful .c/.h C-files. The Robots library provides a small C++ library integrating the following components:

- the interface to the Modelica External Object (see Table 1),
- the LUA interpreter,
- a data core to hold the robot program states like reference position, current command, etc.,
- helper functions to load and execute the LUA script, and
- custom LUA function definitions to be used in the LUA script to control the robot (see Table 2)

The LUA interpreter runs in a separate thread and therefore does not block the execution of the simulation process in Modelica. The Modelica External Object interface to said C++ library comprises of several Modelica functions listed in Table 1.

To enable the user to program robot operations in LUA, a set of custom LUA functions must be provided. Table 2 shows the available commands, which can be used in a LUA script to control the robot in Modelica.

4.2 The robot controller in Modelica

The DLR Robots library provides a robot controller model (`Robots.Controllers.ControllerSixAxes`) suited for the control of six axes industrial robots. Figure 7 shows the interfaces of this model. The inputs and outputs can be utilized by the LUA robot program with the I/O commands from Table 2, allowing the controller to react to signals from the Modelica model. The robot joint sensor input `q_robot` is used by the controllers' path-planning in order to determine the start point for planned trajectories and whether the trajectory is finished. Said trajectory is provided via the robot reference joint angle output `q_ref` and can be subsequently used as an

[1] `https://www.lua.org/`

DOI 10.3384/ecp2020174153

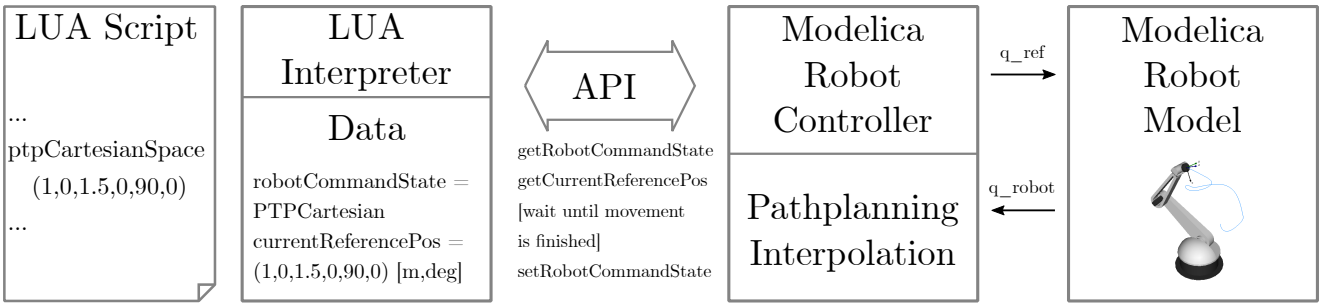


Figure 6. Overview of the LUA/Modelica integration. The LUA script is executed by the LUA interpreter. The Modelica model communicates via the external object API with the LUA Interpreter and handles the physics simulation of the LUA program commands.

Table 1. Modelica functions utilized by the External Object (con-/destructor omitted)

Function	Description
`open`	Opens a LUA script
`run`	Runs the loaded LUA script in a parallel thread
`getRobot↩ CommandState`	Returns the current command state of the robot program
`setRobot↩ CommandState`	Sets the current command state of the robot program
`setNumAxes`	Defines the number of axes in programs
`getReference↩ CartesianPosition`	Returns the current TCP reference position in Cartesian space
`getAnalogOutputs`	Returns the value of the analog outputs to Modelica
`setAnalogInputs`	Sets the value of the analog inputs in the LUA interpreter
`getDigitalOutputs`	Returns the value of the digital outputs to Modelica
`setDigitalInputs`	Sets the value of the digital inputs in the LUA interpreter
`getOverrides`	Retruns the override values to set a speed scale factor for movements

input for the axes controllers. Internally, the controller consists of three sub-components (see Figure 8). The interpreter encapsules the API from Table 1 and provides the sample clock for the robot controller. The path-planning block contains the trajectory generator models from `Robots.Blocks.Pathplanning.TrajectoryGenerators`, and switches between the different path-planning algorithms depending on the robot command state (e.g. idle, PTP in joint space or Cartesian space, etc.). The interpolator block takes the sampled reference trajectory from the path-planning block and

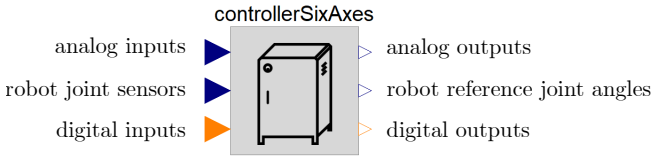


Figure 7. Interfaces of the `ControllerSixAxes` model

interpolates it with various filters (e.g. moving average) to create a smooth reference joint trajectory output.

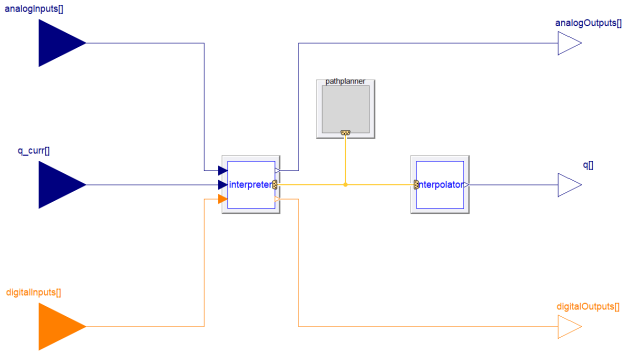


Figure 8. Sub-components of the `ControllerSixAxes` model

4.3 The robot command state

The main arbiter between the LUA interpreted program calls and the Modelica model is the Variable `RobotCommandState`. It determines the current state of the robot and can be set to one of the path-planning command states (*PTPJOINT_CARTESIAN, PTPJOINT_JOINT, PTPCARTESIAN, TRAJECTORY, TEACH*) by the LUA interpreter. Additionally, it can be reset by the Modelica model to *IDLE* by the path-planning block at the end of a movement. Figure 9 shows this alternating interaction between the two executed threads: the LUA interpreter and the Modelica simulation.

Table 2. Custom LUA functions to be used in robot programs

Function	Description
`ptpCartesianSpace` ($\underline{r}_{ref}, \underline{\varphi}_{ref}$)	Commands a linear PTP movement in Cartesian space
`ptpJointSpace`↩ `Angles`($q_1,..,q_n$)	Commands a linear PTP movement in Joint space (reference position in joint angles)
`ptpJointSpace`↩ `Position`($\underline{r}_{ref}, \underline{\varphi}_{ref}$)	Commands a linear PTP movement in Joint space (reference position in cartesian coordinates)
`setDigitalOutput` (*index, value*)	Sets the digital output *index* to *value*. This value can then be used in Modelica to control parts of the model.
value = `getDigitalInput` (*index*)	Returns the *value* of the digital input from Modelica to be used in the LUA script
`setAnalogOutput` (*index, value*)	Sets the analog output *index* to *value*. This value can then be used in Modelica to control parts of the model.
value = `getAnalogInput` (*index*)	Returns the *value* of the digital input from Modelica to be used in the LUA script
`wait`(*time*)	Pauses the robot program execution for *time* seconds
`print`(`string`)	Outputs *string* to the console window.

4.4 Program example - positioning during a welding process

In this example, a LUA robot program is used to position a robot in predefined welding positions and to activate the welding gun. The following listing shows a part of the LUA robot program:

```
print("Welding Program 1")
setDigitalOutput(1,0); //deactivate
    welding gun
wait(1);
ptpJointSpaceAngles(0,-90,45,0,45,0)
ptpCartesianSpace
 (0.37,-1.78,1.21,-42.03,117.47,-35.89)
ptpCartesianSpace
 (0.37,-1.78,1.18,-42.03,117.47,-35.89)
setDigitalOutput(1,1);  //activate welding
    gun
wait(1);  // wait for completion of
    welding process
setDigitalOutput(1,0);  //deactivate
    welding gun
ptpCartesianSpace
 (0.375,-1.78,1.21,-42.03,117.47,-35.89)
 ...
```

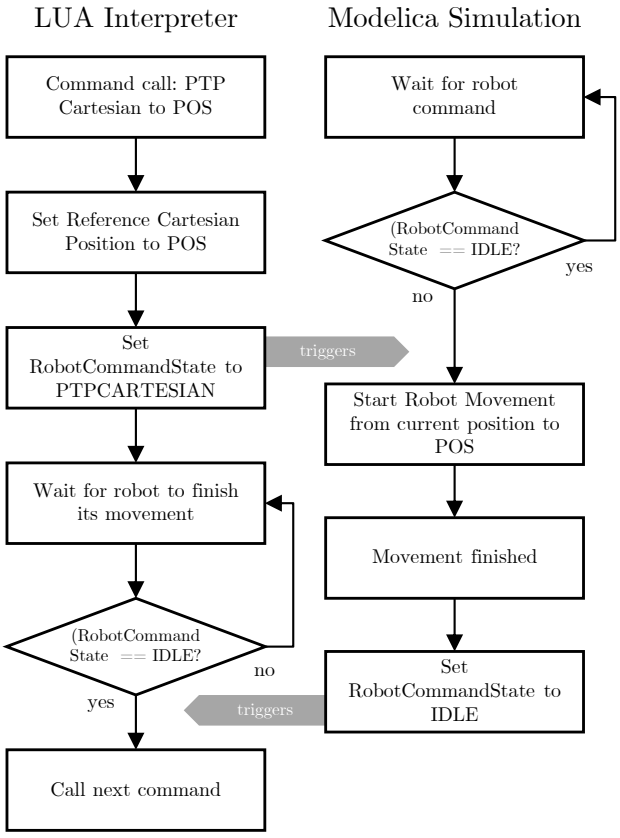


Figure 9. Exemplary interaction between the LUA interpreter thread and the Modelica simulation thread performing a Cartesian PTP movement to the position POS

Figure 10 shows the Modelica model with the aforementioned `ControllerSixAxes`, the robot drive train, kinematics and the robot visualization. The selected robot model is a KUKA Quantec robot, equipped with a welding gun operating on a car frame. The first digital output of the robot controller is connected with the welding gun to control their actuators and to activate the welding process. The robot uses the PTP path-planning blocks of the library to reach the predefined welding positions in the program. After a position is reached, the digital output is activated triggering the closing of the welding gun and ultimately the start of the welding. After waiting for the welding process to finish, the robot moves to the next welding position.

5 Applications

As a base library for robotic research, the DLR Robots library can be used in various simulation and control applications. Some projects utilize it to analyze robotic systems, while others are making use of the real-time capable path-planning algorithms to control real-world robots. All following examples are basically using the same functional blocks introduced in Section 3, but with different robot model packages.

DOI 10.3384/ecp2020174153

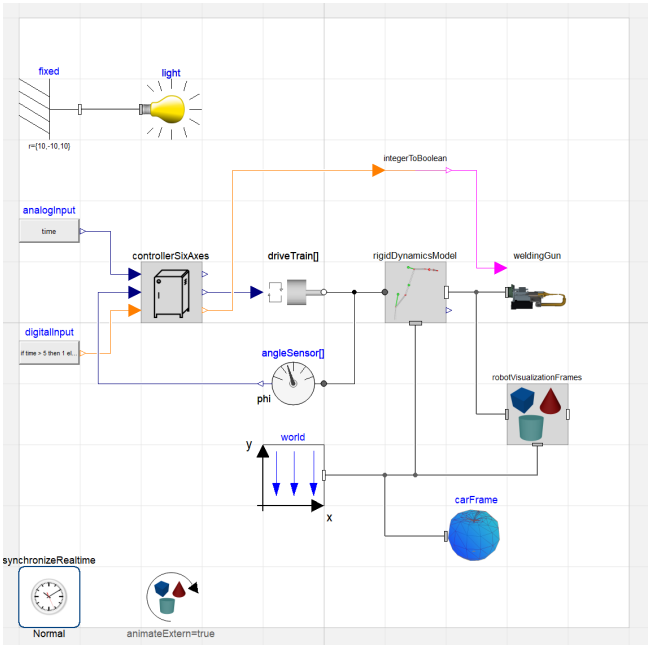


Figure 10. Modelica model utilizing the DLR Robots library to simulate the robot movements of a welding process

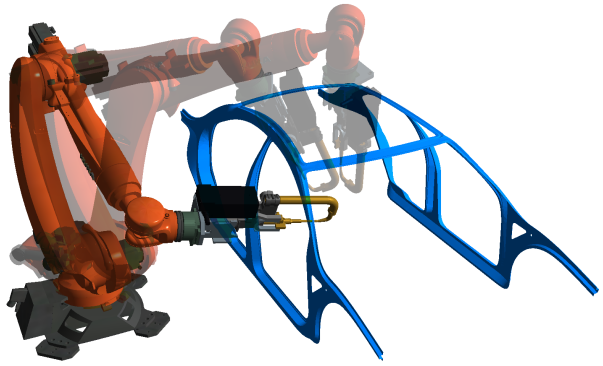


Figure 11. Positioning for multiple welding processes during execution of the LUA robot program

5.1 DLR Robotic Motion Simulator

The DLR Robotic Motion Simulator is a modified KUKA KR500TÜV industrial robot carrying a multi-purpose simulator cockpit (see Figure 12). The system can be used to perform interactive driving and flight simulations (Bellmann et al., 2011a). During the simulation run, the driver/pilot controls the virtual vehicle and the robot moves accordingly in real-time to simulate the movements of the vehicle. In this use-case, the DLR Robots library is used to provide the real-time path-planning of the robot joint trajectories while considering the hardware limits of the system.

5.2 DLR Terramechanics Robotic Locomotion Lab

The DLR Terramechanics Robotic Locomotion Lab (TROLL) is a robotic test bed for automated wheel/soil

Figure 12. The Robotic Motion Simulator (RMS) is a flexible, industrial robot based flight/driving simulator. The path-planning algorithms and kinematic functions of the DLR Robots library are used to control the Robotic Motion Simulator in real-time.

contact tests (Buse et al., 2018). An industrial robot is pressing a space rover wheel into a soil surface, measuring the resulting forces, slip of the wheel, deformation of the soil or even the movement of soil particles via camera based particle image velocimetry. The DLR Robots library has been used to simulate the complete system as part of a feasibility study (see Figure 13), and is also used to control the robot during operations.

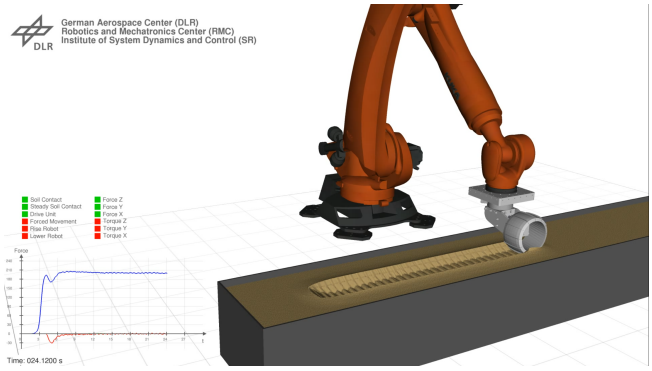


Figure 13. Feasibility study of the DLR Terramechanics Robotic Locomotion Lab - A KUKA KR3100 Quantec presses a rover wheel into a simulated soil surface, while following the rotating wheel with constant speed over the ground

5.3 Active Space debris removal with a robot arm

The increasing density of large debris objects in Low Earth Orbit poses a growing problem as the probability for collisions increases. In order to de-orbit large objects, several approaches are actively researched, such as Active Debris Removal (ADR) utilizing a robot arm. As part of an ESA project, this approach has been simulated utilizing the DLR Robots library to model the robots' arm behavior in a combined control GNC simulation (Reiner, 2018). The chaser satellite first synchronizes with the tumbling

movement of the target satellite (here: Envisat). Next, the robot arm with three axes grabs the docking ring and enables a physical connection between the chaser satellite and the target. Thereafter, the chaser satellite actively steadies the tumbling target and subsequently initiate the controlled de-orbiting. In this use-case, the DLR Robots library was used to simulate the kinematics and dynamics of the robot arm, whereas the drive-trains have been simulated with additional detailed models.The GNC simulation tool based on the object-oriented DLR SpaceSystems (Reiner and Bals, 2014) library and DLR Environments library (Briese et al., 2017) is used to design and simulate the control algorithms, satellite dynamics including flexible elements such as the solar panel, kinematics as well as the robot arm control.

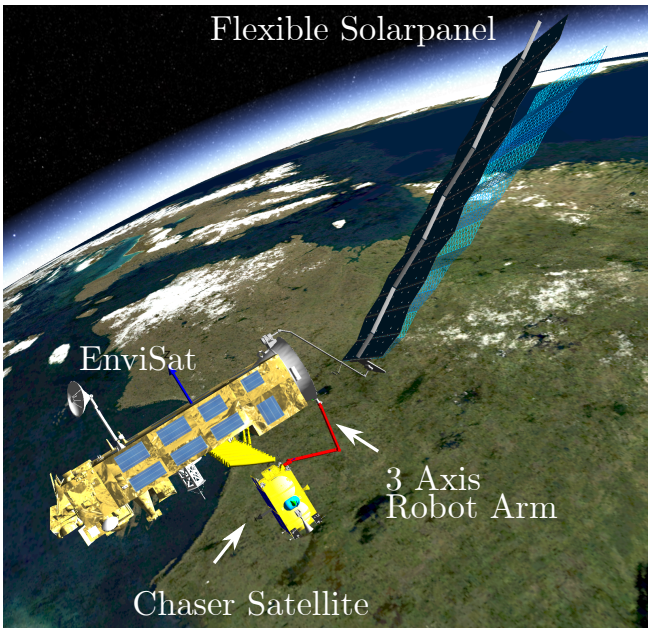


Figure 14. De-orbit scenario simulation of the nonfunctional EnviSat. The capture satellite grasp the tumbling target with a three axes robot arm.

5.4 Analyzing grasping and placement processes during rover missions

For the research project ROBEX (Robotic Exploration of EXtreme Environments) an analog (on earth) mission scenario has been designed, where a rover performs autonomous pick-up and placement of sensor packages (Hellerer et al., 2016; Wedler et al., 2015). In order to analyze the forces and torques acting on both the arm and the rover, the DLR Robots library has been used. Additionally the complete mission scenario has been simulated including robotic operations (see Figure 15).

6 Conclusions

The replaceable package mechanism in Modelica provides the user with great flexibility, as it allows parameterizing models not only with parameters or functions but also

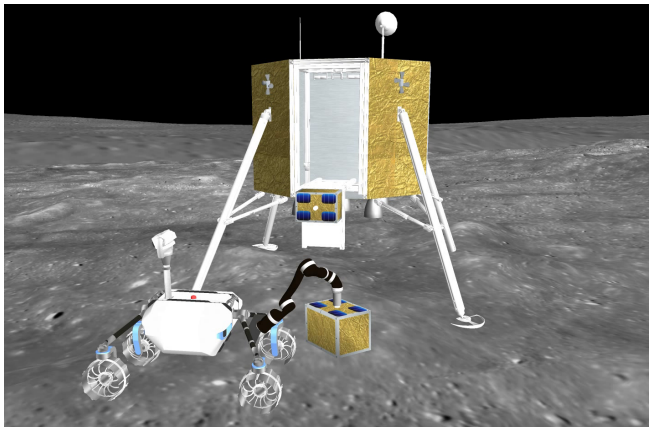


Figure 15. Rover equipped with Jaco Arm lifting a sensor package during the ROBEX mission scenario

complete sets of functionalities and even structural components. This is used in the DLR Robots library to ease the workload of system-modelers and to increase the reusability of Modelica models utilizing robotic systems. The combination of Modelica with the scripting language LUA is a promising method to provide flexible and higher-level control over simulation procedures, such as robot operations. In the future, a more generic LUA library will be developed in order to enable this potential in other domains beyond robotics.

7 Acknowledgments

The authors would like to thank Matthias Reiner and Fabian Buse (DLR) for providing additional example material for this paper. Additionaly we would like to thank Mehran Assanimoghaddam, Stefan Hartweg, Matthias Reiner and Robert Reiser (DLR) for their contributions to the library and valuable discussions.

References

Tobias Bellmann, Johann Heindl, Matthias Hellerer, Richard Kucher, Karan Sharma, and Gerd Hirzinger. The DLR Robot Motion Simulator Part I: Design and setup. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4694–4701. IEEE, Mai 2011a.

Tobias Bellmann, Martin Otter, and Gerd Hirzinger. The DLR Robot Motion Simulator Part II: Optimization based path-planning. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4702–4709. IEEE, Mai 2011b.

Lale Evrim Briese, Andreas Klöckner, and Matthias Reiner. The DLR Environment Library for Multi-Disciplinary Aerospace Applications. In *12th International Modelica Conference*, Mai 2017. URL `https://elib.dlr.de/112971/`.

Matthias Brossog, Johannes Kohl, Jochen Merhof, Simon Spreng, Jörg Franke, et al. Energy Consumption and Dynamic Behavior Analysis of a six-axis Industrial Robot in an Assembly System. *Procedia Cirp*, 23:131–136, 2014.

Fabian Buse, Tobias Bellmann, Roy Lichtenheldt, and Rainer Krenn. The DLR Terramechanics Robotics Locomotion Lab. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Juni 2018. URL https://elib.dlr.de/121796/.

Francesco Casella, Martin Otter, Katrin Proelss, Christoph Richter, and Hubertus Tummescheit. The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-fluid Pipe Networks. In *Proceedings of the 5th international modelica conference*, pages 631–640, 2006.

Rhama Dwiputra, Alexey Zakharov, Roustiam Chakirov, and Erwin Prassler. Modelica Model for the Youbot Manipulator. In *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 096, pages 1205–1212. Linköping University Electronic Press, 2014.

Matthias Hellerer, Martin J. Schuster, and Roy Lichtenheldt. Software-in-the-Loop Simulation of a Planetary Rover. In *The International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Juni 2016. URL https://elib.dlr.de/104934/.

Gerd Hirzinger, Johann Bals, Martin Otter, and Johannes Stelter. The DLR-KUKA Success Story: Robotics Research improves Industrial Robots. *IEEE Robotics & Automation Magazine*, 12(3):16–23, 2005.

Arif Kazi, Günther Merk, Martin Otter, and Hui Fan. Design Optimisation of Industrial Robots using the Modelica multi-physics Modeling Language. In *33rd International Symposium on Robotics*, pages 347–352, Oktober 2002. URL https://elib.dlr.de/11898/. LIDO-Berichtsjahr=2002,.

Matthias Reiner and Johann Bals. Nonlinear inverse models for the control of satellites with flexible structures. In *10th International Modelica Conference 2014*, Linköping Electronic Conference Proceedings, pages 577–587. LiU Electronic Press, 2014. URL https://elib.dlr.de/92164/.

Matthias J. Reiner. *Modellierung und Steuerung von strukturelastischen Robotern*. PhD thesis, Technische Universität München, 2011.

Matthias J. Reiner. Modelling And Combined Control Of A Satellite With A Robot Arm For Active Debris Removal. In *69th International Astronautical Congress*, 2018. URL https://elib.dlr.de/123349/.

Charles W Wampler. Manipulator Inverse Kinematic Solutions based on Vector Formulations and Damped Least-Squares Methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, 1986.

Armin Wedler, Mathias Hellerer, Bernhard Rebele, Heiner Gmeiner, Bernhard Vodermayer, Tobias Bellmann, Stefan Barthelmes, Roland Rosta, Caroline Lange, Lars Witte, Nicole Schmitz, Martin Knapmeyer, Alexandra Czeluschke, Laurenz Thomsen, Christoph Waldmann, Sascha Flögel, Martina Wilde, and Yuto Takei. ROBEX – Components and Methods for the Planetary Exploration Demonstration Mission. In *13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, ASTRA. ESAWebsite, 2015. URL https://elib.dlr.de/98242/.