# THE AMERICAN MODELICA CONFERENCE 2020

MARCH 23–25 | UNIVERSITY OF COLORADO BOULDER | WWW.MODELICA.ORG

Modelica

## PROCEEDINGS OF THE AMERICAN MODELICA CONFERENCE 2020

Boulder, Colorado, USA, March 23-25, 2020

**EDITORS**

Michael Tiller, Hubertus Tummescheit, Luigi Vanfretti, Christopher Laughman, and Michael Wetter

**PUBLISHED BY**

Modelica Association and Linköping University Electronic Press

ISBN: 978-91-7929-900-2
Series: Linköping Electronic Conference Proceedings, No. 169
ISSN: 1650-3686
eISSN: 1650-3740
DOI: 10.3384/ecp20169

**ORGANIZED BY**

North America Modelica Users' Group
na.modelica-users.org

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

**CONFERENCE LOCATION**

University Memorial Center
University of Colorado Boulder
1725 Euclid Avenue
Boulder, CO 80309
USA

# MARK JENNINGS

Senior Technical Leader, Vehicle Energy Management & Propulsion Systems Analysis; Research & Advanced Engineering, Ford Motor Company

## ABSTRACT

### Automotive Electrified Vehicle & Powertrain System Modeling, Simulation & Optimization

In recent years, key external drivers and technology trends including societal concern for climate change and emphasis on improvement of air quality, rapid expansion of renewable energy sources and emergence of vehicle connectivity have greatly accelerated the move towards electrification for on road vehicle propulsion. In response, Ford has greatly increased its investments in the electrification of its product portfolio. Core to Ford's product portfolio is a set of electrified powertrain system architectures encompassing HEV, PHEV and BEV systems spanning a range of vehicle applications with both lateral (front wheel drive) and longitudinal (rear wheel drive) driveline configurations.

The aggressive move towards electrification has been accompanied by an equally aggressive move towards reliance on model-based system design, development and optimization. This has been driven by critical needs of the business: the need to reduce development costs; the need to reduce development time for delivery of new technologies to the marketplace; and the need to better understand, manage and optimize critical attribute trade-offs and system interactions in order to realize the full potential of powertrain electrification.

Critical areas of importance to Ford for further expansion of powertrain system modeling and simulation capabilities and methodologies are highlighted and discussed. These include:

- Better optimization of in-vehicle IC engine operation for HEV's
- Thermal management and climate system design optimization
- Energy efficiency trade-offs with drivability
- On road energy management and emissions including interactions with driver assistance technologies and automated driving systems
- Leveraging connectivity for improved powertrain attributes
- Establishment of an enterprise level virtual vehicle development process

## BIO

Mark Jennings is Senior Technical Leader for Vehicle Energy Management & Propulsion Systems Analysis for Ford Motor Company working in Ford's Research & Advanced Engineering organization on electrified powertrain systems. Over the past 20 years, he has been leading efforts to establish and apply model-based development and optimization methodologies towards the advancement of electrification technologies for vehicle propulsion. This work has covered a range of modeling & simulation approaches spanning hardware capability assessments leveraging optimal control methods through direct simulation and optimization of complex, coupled feedback control and physical hardware systems.

Mark's work on electrified powertrain systems has covered a range of system technologies encompassing mild/medium hybrid electric vehicle (HEV) systems, full HEV systems, plug-in HEV systems, battery electric vehicles and fuel cell electric vehicles. Throughout his years at Ford, he has led numerous efforts to define and assess new electrified powertrain system concepts. Through this work he has had played a significant role in defining Ford's powertrain electrification strategy.

Mark has pioneered the development of novel model based system engineering methods for electrified powertrain systems including the integration of SysML with highly detailed system simulation methodologies. Mark has led the development of model based calibration methods for HEV system controls. These methods have been successfully deployed in the development of Ford's production HEV systems and resulted in significant savings of engineering time and effort as well as reduction in the use of vehicle prototypes. Mark has also established novel model-based methods for early, pre-hardware development and optimization of new powertrain system technologies including assessing impact as and trade-offs with vehicle attributes.

Mark has BS, MS and PhD degrees in Mechanical Engineering. In addition to his experience in electrified powertrain systems and powertrain system simulation methodologies, Mark has extensive experience in multi-dimensional modeling & simulation of turbulent flows, heat transfer and combustion.

# CHRIS GEARHART

Director, Center for Integrated Mobility Sciences

## ABSTRACT

### Modeling and Simulation in Sustainability Research at NREL

Sustainable transportation research has evolved beyond vehicles and fueling systems to new, innovative, and integrated mobility strategies that have the potential to transform the movement of people and goods, enhance national energy security, boost the domestic economy, and save individuals and businesses both time and money. This can be achieved through an ecosystem that integrates technology advancements with a range of domestic energy resources, power grids and building systems, urban planning and efficient fleet operations. Modeling and simulation play a vital role in understanding these complex interactions. Dr. Gearhart will talk about the growing and evolving role of simulation in sustainability research at NREL's Center for Integrated Mobility Sciences.

## BIO

Chris Gearhart joined the National Renewable Energy Laboratory (NREL) in 2012 as director of the Hydrogen Technologies and Systems Center. In 2013 he was appointed to his current position as director of NREL's Center for Integrated Mobility Sciences (formerly known as the Transportation and Hydrogen Systems Center), where he manages a staff of more than 150 scientists and engineers dedicated to understanding the science and engineering needed to efficiently and sustainably move people and goods in a highly integrated system of systems.

Prior to joining NREL, Dr. Gearhart spent 16 years with Ford Motor Company. He started his career at Ford working in computer aided engineering developing finite element, CFD, and multi-body dynamics models. He went on to lead research and development teams in that company's Fuel Cell System, Stack, and Hydrogen Storage division as well as playing pivotal roles in product development, safety research, and reliability engineering.

He holds doctorate and master's degrees in physics from Washington University in St. Louis, where he used computational models to study heavy nuclei and neutron stars. He has bachelor's degree in physics and math from Drake University. Dr. Gearhart has also served on the faculty of Michigan State University and the School for Renewable Energy Science in Akureyri, Iceland.

## CONFERENCE CO-CHAIRS

Dr. Michael Tiller, Ricardo

Dr. Hubertus Tummescheit, Modelon

## PROGRAM CHAIR

Prof. Luigi Vanfretti, RPI

## CONFERENCE EXECUTIVE COORDINATOR

Dr. Christopher Laughman,
Mitsubishi Electric Research Laboratories

## LOCAL CO-CHAIRS

Prof. Wangda Zuo, Univ. of Colorado Boulder

Jessica Stershic, Univ. of Colorado Boulder

Dr. Michael Wetter,
Lawrence Berkeley National Laboratory

Paul Goosens, Maplesoft

Behnam Afsharpoya, Dassault Systèmes

## PROGRAM COMMITTEE

Johan Åkesson, Modelon AB

Scott Bortoff, Mitsubishi Electric Research Laboratories

Bernhard Bachmann, Fachhochschule Bielefeld

Albert Benveniste, INRIA

Christian Bertsch, Robert Bosch GmbH

Volker Beuter, VI-grade GmbH

Torsten Blochwitz, ESI ITI GmbH

David Blum, Lawrence Berkeley National Laboratory

Timothy Bourke, INRIA

Daniel Bouskela, EDF

David Broman, KTH Royal Institute of Technology

Massimo Cimmino, McGill University

Olaf Enge-Rosenblatt, Fraunhofer

Gianni Ferretti, Politecnico di Milano

Valentin Gavan, ENGIE Lab

Anton Haumer, OTH Regensburg

Dan Henriksson, Dassault Systemes

Yutaka Hirano, Toyota Motor Corporation

Jianjun Hu, Lawrence Berkeley National Laboratory

Moritz Lauster, Viessmann Group

Alessandro Maccarini, Aalborg University

Marek Matejak, Charles University in Prague

Alexandra Mehlhase, TU Berlin

Andreas Nicolai, TU Dresden, Institut für Bauklimatik

Henrik Nilsson, University of Nottingham

Thierry S Nouidui,
The United African University of Tanzania

Christoph Nytsch-Geusen, Universität der Künste Berlin

Hans Olsson, Dassault Systèmes

Zheng O'Neill, Texas A&M University

Martin Otter,
DLR, Institute of System Dynamics and Control

Kaustubh Phalak, Ingersoll Rand

Andreas Pillekeit, dSPACE

Adrian Pop, Linköping University

Lisa Rivalin, Facebook

Clemens Schlegel, Schlegel Simulation GmbH

Gerhard Schmitz,
Hamburg University of Technology (TUHH)

Stefan-Alexander Schneider,
University of Applied Sciences Kempten

Michael Sielemann, Modelon Deutschland GmbH

Martin Sjölund, Linköping University

Ed Tate, Exa

Wilhelm Tegethoff, TLK-Thermo GmbH

Matthis Thorade, Modelon

Jakub Tobolar, DLR – German Aerospace Center

Alfonso Urquia, UNED

Dietmar Winkler, USN

Stefan Wischhusen, XRG Simulation GmbH

Dirk Zimmer, DLR

## CONTENTS

### Paper Session 1/Room 1 **Power Systems I**

### Paper Session 1/Room 2 **Mechanical Systems I**

### Paper Session 2/Room 1 **Thermofluid Systems & HVAC**

### Paper Session 2/Room 2 **Mechanical Systems II**

## Paper Session 3/Room 1 **Power Systems II**

## Paper Session 3/Room 2 **Language & Tools**

## Paper Session 4/Room 1 **FMI**

## Paper Session 4/Room 2 **Electromechanical Systems**

# PAPERS

# Multi-market Optimization of a Data Center without Storage Systems

Yangyang Fu[1] Wangda Zuo[1,2] Kyri Baker[1,2]

[1]Department of Civil, Architectural and Environmental Engineering, University of Colorado Boulder, USA,
{yangyang.fu,wangda.zuo,kyri.baker}@colorado.edu
[2]National Renewable Energy Laboratory, USA

## Abstract

Data centers have numerous opportunities to participate in demand response programs considering their large capacities, flexible working environments and work loads, redundant design and operation, etc. Frequency regulation, as one service provided in demand response programs, can also benefits the data centers. This paper aims to develop a real-time multi-market optimization framework for a data center without storage systems to maximize their benefits from participating in both the energy market and the regulation market. Then a case study is conducted to numerically investigate the optimal bids at each hour by considering the energy cost, demand costs, and regulation revenues using a virtual data center located in PJM. Simulation results show that the proposed multi-market optimization framework can help data centers maintain minimum costs by getting maximum regulation revenues while satisfying energy and demand goals.

*Keywords: Frequency Regulation, Data Center, Multi-market Optimization*

## 1 Introduction

Data centers have numerous opportunities to participate in demand response (DR) programs considering their large energy capacities, flexible working environments and work loads, redundant design and operation, etc. For example, researches have shown that an optimized 30 MW data center is comparable to 7 MWh large-scale storage in providing DR service for the power grid (Wierman et al., 2014). Besides, some delay-tolerant data centers are allowed to have flexible work environment and workloads. What's more, the redundant design in data centers to meet reliability standards in order to guarantee their uptime and performance (Standards et al., 2005) can provide extra potentials to DR-related controls.

Frequency regulation (FR), as one type of DR, is an ancillary service that provides continuous, rapid, and automatic corrections for changes in electricity generation or use on a second-to-second basis in order to maintain the system frequency at its nominal value (e.g., 60 Hz in U.S.). Typically, FR resources are generators. FR uses certain amount of generators (e.g., about 1% of total generation) to continuously track the demand variations. The frequency must be strictly maintained within a very narrow range in order to comply with the control performance standards and the balancing authority area control error limit reliability criteria. Besides generators, fast-ramping demand side resources (DSRs) in buildings can also provide FR service to the grid by harnessing the demand flexibility provided by the modulating loads. Typical modulating loads on building side include energy storage systems such as flywheels, batteries and compressed-air energy system, electric boilers and heaters, and independent systems with variable frequency drivers (VFDs).

Recently, awareness of these potentials has drawn attention to the capabilities of data centers to participate in DR programs. A survey conducted by the Lawrence Livermore National Laboratory in 2015 shows that about 50% of the participating data centers have interest in smart pricing demand side programs, such as load shedding to avoid peak demand (Bates et al., 2015). However, data centers are reluctant to participate in fast demand response programs such as providing frequency regulation (FR) in ancillary service market, for multiple reasons. One reported concern is that data centers are still learning the process of providing FR and that providing grid services on such a fast timescale can be “outside of their visibility or control" (Bates et al., 2015). This concern is well-founded considering that these programs provide novel and relatively unexplored territory from the point of view of traditional data center control and operations.

This paper aims to explore data centers' ability of providing frequency regulation service to grids and maximize their benefits from participating regulation market and energy market as a whole. First, a synergistic control strategy together with a new regulation flexibility factor is proposed to enable the provision of regulation services in data center. Then, a real-time optimization framework is developed to maximize the data centers' benefits from participating in both the regulation market and the energy market. In Section 4, the optimization framework is evaluated in a Modelica-based environment for typical days in January and July.

# 2 Synergistic Strategy for Frequency Regulation

In this section, we propose a synergistic control strategy for data centers to provide FR service. This strategy is composed of four major parts. The first one is *Baseline Routine*, which predicts the baseline power usage when the data center provides no FR. The second one is *Bidding Capacity*, which is the capacity bid that the data center submits to the electrical market. The third one is *Server Power Management*, where an aggregator is adopted to represent the aggregated performance of servers in the data center. The clock frequency of the aggregator can be directly changed by a Proportional-Integral-Derivative (PID) controller in order to follow the regulation signal. Based on that, the desired frequencies for individual servers will be determined by a set of predefined assignment rules and then be propagated to all servers. The forth one is *Cooling Power Management*, which adjusts the chilled water supply temperature (CHWST) setpoint to respond to the regulation signal.

Figure 1 shows the workflow of the proposed synergistic control strategy. The *Baseline Routine* outputs the prediction of the overall power profile for the data center $P_{bas}$ when no FR service is provided. In this paper, the prediction is performed using detailed energy models, although many other methods such as machine learning techniques can also be used. The detailed energy models and baseline settings can be referred to Section 4.1. The *Bidding Capacity* is a module that can calculate the optimal capacity bid for the data center at each time step, and output raw regulation power $\Delta P_{reg,raw}$ based on the optimal capacity bid and received regulation signal $r$ from the electrical market. Then, the reference power $P_{ref}$ for the data center to track is the summation of the predicted baseline power $P_{bas}$ together with the raw regulation power $\Delta P_{reg,raw}$.

The *Server Power Management* first determines the number of required active servers in the aggregator $N_{act}$ based on the predicted workload $\lambda'$ in the next time step (e.g., one hour ahead). Then a closed-loop control using a PID controller is utilized to minimize the error between the measured total power usage $P_{mea}$ and the reference power $P_{ref}$ by adjusting the aggregated frequency of the server aggregator. Meanwhile, the *Cooling Power Management* applies an open-loop control to adjust the cooling system power usage by resetting the CHWST setpoint in response to the received regulation signal $r$.

The server aggregator receives the aggregated frequency $f_{agg}$ and the required number of active servers $N_{act}$ from the FR controller. Assuming there are $N_0$ number of servers in the data center, the server aggregator then calculates the CPU frequency $f_i$ for an individual server $i$ based on predefined assignment rules. The cooling system receives CHWST setpoint from the FR controller. Both the IT system and the cooling system respond in such a way that their total power $P_{mea}$ is adjusted to track the reference power $P_{ref}$.

For the aggregator, there are several assignment rules to control the individual server's frequency (Li et al., 2013; Wang et al., 2019). We can also represent the aggregated server power $P_{servers}$ of all servers under an assignment rule using a simplified model (Li et al., 2013) and this approach is adopted by this paper and detailed in Section 2.1. For the FR controller, more details are described in the rest of this section.

## 2.1 Server Power Management

The servers in the data center can be considered as an aggregator, which is characterized by the active number of servers $N_a$ and the aggregated frequency $f$. These two parameters can be determined based on the regulation signal $r$ and incoming workload $\lambda$. The aggregrated frequency can then be distributed to the single servers as $f_i$ using a predefined assignment algorithm. The relationship between $N_a$, $f$ and $r$, $\lambda$ is detailed in the rest of this section.

### 2.1.1 Server Aggregator Model

The IT equipment, especially the servers, are modelled as an aggregator, which can predict the total IT power usage and the server response time based on CPU frequency, workload arrival rate, and number of active servers (Li et al., 2013). Details are shown as follows.

$$P_{servers}(t) = \lambda(t)\sum_{0}^{r} b_i f(t)^i + \sum_{0}^{s} c_j N_a(t)^j, 0 \le i \le r, 0 \le j \le s \tag{1}$$

where $b_i$, $c_j$ are constant coefficients that can be obtained from curve fitting techniques, $\lambda(t)$ is the total arrival rate, $f$ is the aggregated relative frequency, ranging from 0 to 1, and $N_a$ is the active number of servers at current time. $f$ and $N_a$ can be optimally determined in order to minimize cost.

Here we use the average response time to quantify the service quality of a data center. The workloads are modeled as GI/G/m queues, which assumes a general distribution with independent arrival times and a general distribution of service times. The total time that a job spends in the queuing system is known as response time. The response time usually consists of two parts: waiting time, that is, the time that a job spends in a queue waiting to be serviced; and service time, that is, the time that a job needs to be executed. The average response time model is adopted from (Bolch et al., 2006). Details are shown as follows.

$$\mu(t) = kf(t) \tag{2}$$

$$t_s = \frac{1}{\mu(t)} \tag{3}$$

$$\rho(t) = \frac{\lambda(t)}{N_a(t)\mu(t)}, 0 \le \rho(t) \le 1 \tag{4}$$

$$P_m = \begin{cases} \frac{\rho(t)^m + \rho(t)}{2}, & \rho(t) \ge 0.7 \\ \rho(t)^{\frac{N_a(t)+1}{2}}, & \rho(t) < 0.7 \end{cases} \tag{5}$$

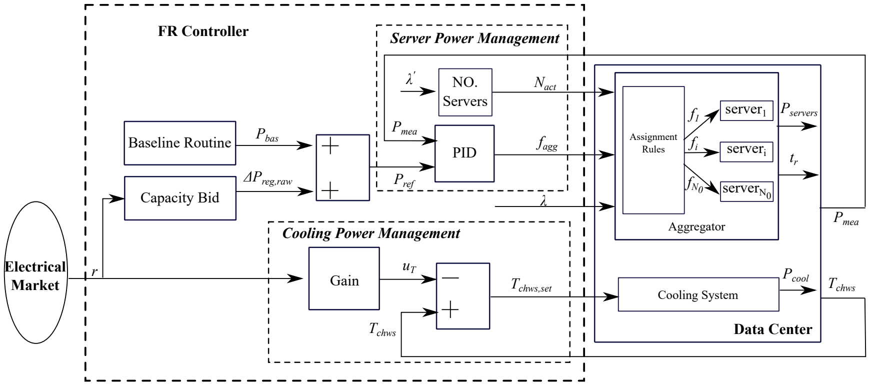


**Figure 1.** Data center frequency regulation control

$$t_w = \frac{C_A^2 + C_B^2}{2N_a(t)} \frac{P_m}{\mu(t)(1-\rho(t))} \tag{6}$$

$$t_r = t_s + t_w \tag{7}$$

In the above equations, $\mu$ is the mean service rate, $k$ is a constant parameter, assuming the service rate is proportional to the frequency, $\rho$ is the average utilization of the server, representing the fraction of occupied time, $P_m$ is approximated probability that an arriving job is queued, $C_A$ and $C_B$ are constant coefficients reflecting the variations of the inter-arrival times and request sizes, $t_r$, $t_s$, and $t_w$ are the average response time, service time and waiting time for the aggregator respectively.

### 2.1.2 Number of Active Servers

The number of servers in a data center needs to satisfy the following condition in order to ensure the stability of the queue. This condition means that the service rate in the data center should be greater than the arrival rate.

$$N_a(t)\mu(t) > \lambda(t) \tag{8}$$

Under design conditions, to guarantee reliability, a scaling factor $\gamma$ as defined in Eq.(9) is utilized here to describe the design redundancy of the servers. The $\gamma$ is set to greater than 1. If $\gamma = 1$, it means all the CPU clock frequencies need to set at maximum level just to serve the average workload, which limits the potential of FR. The $\gamma$ is described as

$$\gamma = \frac{\mu_0 N_0}{\lambda_0}, \tag{9}$$

where $\mu_0$ is the nominal service rate of a single server, $N_0$ is the nominal number of servers in a data center room, and $\lambda_0$ is the nominal arrival rate to be served by the data center.

When using a server aggregator model as described in Eq. (1), the $\gamma$ can then be rewritten as:

$$\gamma = \frac{kN_0}{\lambda_0} = \frac{kN_a(t)}{\lambda_{mean}(t)}, \tag{10}$$

where $k$ is a constant parameter, assuming the service rate is proportional to the aggregated frequency, $N_a$ is the number of active servers at current time step, and $\lambda_{mean}$ is the mean arrival rate at the current time step.

The number of active servers is calculated at an interval of 1 hour because the servers have relatively long wakeup time. The detailed formula is shown in Eq. (11), where the operator $\lceil x \rceil$ is the ceiling function which yields the smallest integer greater or equal to $x$.

$$N_a(t) = \lceil \frac{\gamma \lambda_{mean}(t)}{k} \rceil \tag{11}$$

By adding a FR flexibility factor $\beta$ during operation, we can determine the number of active servers based on the predicted coming arrival rate, as shown in Eq. (12). The greater $\beta$ is, the more servers are activated for a specific workload.

$$N_a(t) = \lceil \beta \frac{\gamma \lambda_{mean}(t)}{k} \rceil, N_a(t) \in [0, N_0] \tag{12}$$

### 2.1.3 Frequency Control

The aggregated frequency $f_{agg}$ is controlled by a PID controller to track the reference power $P_{ref}$ calculated from the electrical market. The reference power $P_{reg}$ is calculated as

$$\Delta P_{reg,raw}(t) = r(t) C_{reg} \tag{13}$$

$$P_{ref}(t) = P_{bas}(t) + \Delta P_{reg,raw}(t) \tag{14}$$

where $\Delta P_{reg,raw}$ is the raw power signal and $C_{reg}$ is the regulation capacity that the data center bids in the market.

The frequency $f_{agg}$ is then determined by the PID controller as follows.

$$f_{agg}(t) = K_p e(t) + K_i \int_0^t e(x)dx + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}, f_{agg}(t) \in [f_{min}, f_{max}] \tag{15}$$

$$e(t) = P_{ref}(t) - P_{mea}(t) \tag{16}$$

In the above equations, $K_p$, $K_i$, and $K_d$ denote the coefficients for the term P, I and D, respectively. $e$ is the error between the reference power $P_{ref}$ and the measured power

$P_{mea}$. The maximum aggregated frequency is 1, while the minimum frequency varies based on the number of active servers due to the constraints of Quality of Service (QoS). Details on how to determine $f_{min}$ are described in Section 2.1.4.

### 2.1.4 Minimum Aggregate Frequency

Using a service response time model shown in Eq. (2) and Eq. (7), we know that the response time of the servers depends on the aggregated frequency. If the frequency is low, then it takes relatively long time for the servers to respond to the arrival workload, which means the QoS of the data center is compromised. To enable FR and guarantee the QoS, the aggregated frequency should meet a minimum value. The minimum can be obtained by solving the following optimization problem.

$$\begin{aligned} \min \quad & f(\rho(t)) = \frac{\lambda(t)}{kN_a(t)\rho(t)} \\ s.t. \quad & 0 \le \rho(t) \le 1 \\ & t_r(t) \le t_{r,u} \end{aligned} \tag{17}$$

where $\rho(t)$ is the utilization rate as defined in Eq. (4), $t_r(t)$ is the service response time as calculated in Eq. (7) and $t_u$ is the maximum response time allowed by the data center.

Rearranging Eq. (2) to Eq. (7), we can get the response time $t_r(t)$ as a function of the utilization rate $\rho(t)$ as follows.

$$t_r(\rho(t)) = \frac{\rho(t)}{\lambda(t)}[N_a(t) + \frac{C_A^2 + C_B^2}{2(1-\rho(t))}P_m(\rho(t))] \tag{18}$$

It is easy to show that

$$\frac{\mathrm{d}t_r(\rho)}{\mathrm{d}\rho} > 0 \tag{19}$$

Thus, the above-mentioned optimization problem can be solved at each time step as:

$$f_{min}(t) = \frac{\lambda(t)}{kN_a(t)\rho^*(t)} \tag{20}$$

where $\rho^*(t)$ is the optimal utilization rate, and $\rho^*(t)$ should satisfy the nonlinear relationship shown as:

$$t_r(\rho^*(t)) - t_u = 0 \tag{21}$$

## 2.2 Cooling Power Management

The cooling system power is managed by resetting the chilled water supply temperature. The regulation signal from the electrical market is directly used to change the chilled water supply temperature setpoint $T_{chws,set}$ by Eq. (22).

$$T_{chws,set}(t) = T_{chws}(t) - \Delta T r(t) \tag{22}$$

where $T_{chws}$ is the chilled water temperature at current time step, $\Delta T$ is the user defined regulation range for the temperature, and varies based on the design supply temperature range of chillers. Here we set it to 2 °C. The negative sign at the right term means when regulation up is needed, the temperature setpoint is reduced, and vice versa.

# 3 Multi-market Optimization Framework

A real-time optimization framework is applied for optimizing the operation of the data center without thermal storage system in the presence of real-time (or day-ahead) energy prices, peak demand charges, and frequency regulation revenue. For each optimization time step, the overall objective can be described as:

$$\begin{aligned} \min \quad & J(C_{reg}) = E_{cost} + D_{cost} - R_{revenue} \\ s.t. \quad & 0 \le C_{reg}(t) \le C_{reg,max}(t) \\ & t_r(t) \le t_{r,u} \\ & S(C_{reg}) \ge S_l \end{aligned} \tag{23}$$

where $C_{reg}$ is the design variable, representing regulation capacity bid at each hour, $C_{reg,max}$ is the maximum capacity the data center can provide for regulation, $t_r$ is the response time of the data center service, $t_{r,u}$ is the allowable upper limit of the response time, $S$ is the regulation performance score defined by PJM as shown in Section 6.1, and $S_l$ is the lowest allowable performance score by PJM to participate in regulation market.

The cost function $J$ has three terms: energy cost $E_{cost}$, demand cost $D_{cost}$ and regulation revenue $R_{revenue}$. The energy cost is calculated by Eq. (24).

$$E_{cost} = \int_t^{t+\Delta t} p_{em}(t)P_{DC}(t)dt \tag{24}$$

where $p_{em}$ is the real-time price signals for energy use at time $t$, $P_{DC}$ is the total power consumption for the data center at time $t$. The calculation period starts from time $t$ and ends at $t+\Delta t$, where $\Delta t$ is the optimization step, and is set to 1 hour in this study.

The electric demand during the current optimization horizon is penalized by the demand price $p_{dm}$ as shown in Eq. (25).

$$D_{cost} = p_{dm} \cdot \max((P_{dm} - P_{dm,lim}), 0) \tag{25}$$

where $p_{dm}$ is the demand price, $P_{dm}$ is the power demand calculated as the average power for each 30-min interval, and $P_{dm,lim}$ is the limit of required demand. This function means if the demand in current step exceeds a predefined demand value, then the optimization cost function is penalized by the demand difference. Otherwise, no penalization is applied. Note that $p_{dm}$ and $P_{dm}$ are both utility specific, and may vary from this definition.

The revenues from regulation service is computed as follows.

$$R_{revenue} = \int_{t}^{t+\Delta t} p_{rm}(t)C_{reg}(t)dt \qquad (26)$$

where $p_{rm}$ is the real-time price signal from the regulation market, and $C_{reg}(t)$ is the regulation capacity bid for each time step.

The price signals such as $p_{em}$ and $p_{rm}$ need to be predicted one optimization step ahead, e.g. 1 hour in this study. Many researches have been conducted for this purpose. In this paper, historical prices of these two electrical markets are used, which means the hourly ahead prices are assumed to be perfectly predicted. The demand limit $P_{dm,lim}$ can also be predefined by the data center operators based on historical operation conditions. The maximum regulation capacity at each optimization step is set to 798.2 kW (20% of the nominal power). Note this maximum regulation capacity setting is not the feasible capacity the data center can provide at each hour, because the regulation capacity is related to data center operational conditions such as arrival rate, and weather conditions etc. This simplification has limited influence on the optimization results when the lower limit of performance score $s_l$ is set to a high value, because if the data center makes a bid that exceeds its capacity, it cannot track the reference signal, thus the regulation performance will be low. By setting $s_l$ to a high value can help data center make a reasonable bids when the regulation capacity is hard to predict. The optimization problem is solved using the pattern search algorithm in the optimization engine, GenOpt (Wetter et al., 2001).

# 4 Case Study

A data center as shown in Figure 2 is used to investigate the benefits from participating in different electrical markets. The data center is considered as a price taker only. This case study investigates the maximum benefits that data centers can obtain from both the real-time energy market and the regulation market in PJM. For the regulation service, only dynamic regulation is studied here, because its price is usually much higher than traditional regulation.

## 4.1 Case Description

The data center is located in Chicago, which is in ASHRAE Climate Zone 5A and within the PJM market territory. For the cooling system, there are two chillers and one integrated waterside economizer providing cooling to the data center room. This cooling system can operate in three modes: Free Cooling (FC) mode when only the WSE is enabled for cooling, Partial Mechanical Cooling (PMC) mode when the chiller and WSE are both triggered, and Full Mechanical Cooling (FMC) mode when only the chiller is activated. There are also two cooling towers, two constant-speed condenser water pumps, two variable-speed chilled water pumps, and one variable speed fan. The cooling system and its control are modelled using

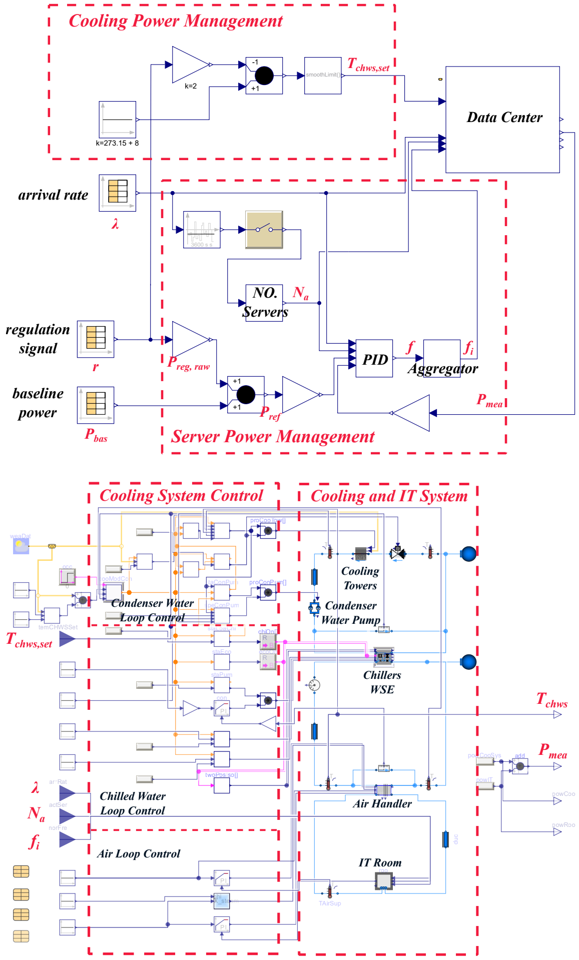


**Figure 2.** Modelica implementation of the studied data center for FR service: FR controller (top) and data center system (bottom)

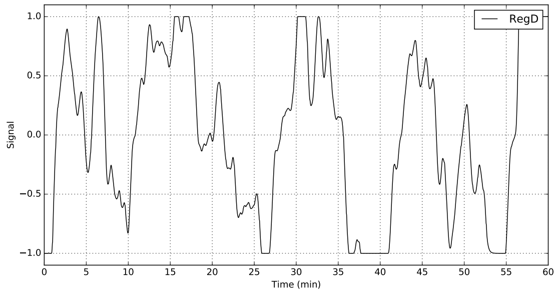


**Figure 3.** An example of one-hour historical RegD signal in January

an open-source equation-based Modelica environment (Fu et al., 2018, 2019a,c,b).

For the IT system, the design number of servers is 8000. The design factor $\gamma$ is set to 1.5 (Li et al., 2013). The total nominal electrical load is about 2700 kW. The calibrated coefficients for Eq. (1) are $b_0 = 0.0154$, $b_1 = 1.5837$, $b_2 = 0.1373$, $c_0 = -22.3540$ and $c_1 = 121.0212$ using the method mentioned in Ref. (Li et al., 2013). When not providing FR, the server aggregator operates at a frequency of 0.8 with a regulation flexibility factor of 1.0, and the CHWST setpoint is set to 8 °C. For the internet data center, the constants $C_A$ and $C_B$ are set to 1 as in Ref. (Li et al., 2013).

For the multi-market optimization, all the settings are the same as the baseline except that an additional FR controller as designed in Section 2 is used to provide regulation service for the grids by adjusting the CPU frequency and CHWST setpoint. The FR flexibility factor is set to 1.1 when providing regulation services. The QoS when providing regulation services is guaranteed by constraining the average response time of the data center service to 6 ms. The lower limit of the performance score in PJM to disqualify a regulation resource is 0.4 (LLC, 2019). Here we set it to a higher value, 0.9. The real-time optimization is performed at a one-hour interval for 2 days in both January ($1/20 \sim 1/21$) (when cooling system operates at FC mode) and July ($7/20 \sim 7/21$) (when cooling system operates at FMC mode).

The price signals of the real-time energy market and the regulation service market in January and July 2018 are posted in Ref. (PJM, 2019), and the price during the optimization period is plotted as shown in Figure 4. An example of one-hour historical RegD signal is plotted in Figure 3. A real-time web service in Wikipedia (Wang et al., 2019) is used as the workload arrival profile during optimization, which is shown in Figure 5.

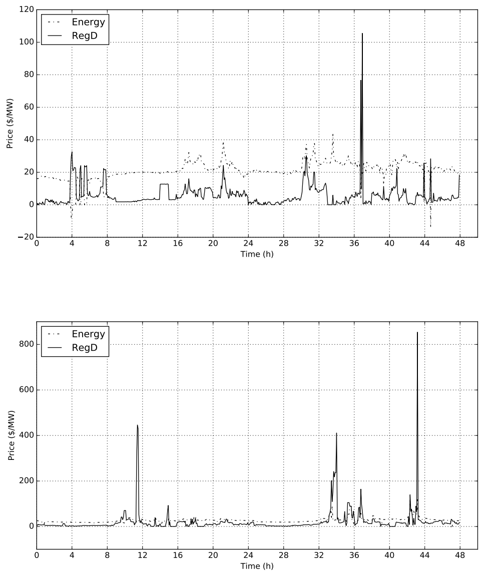


**Figure 4.** Historical real-time prices of PJM energy market and regulation market in January (top) and July (bottom)

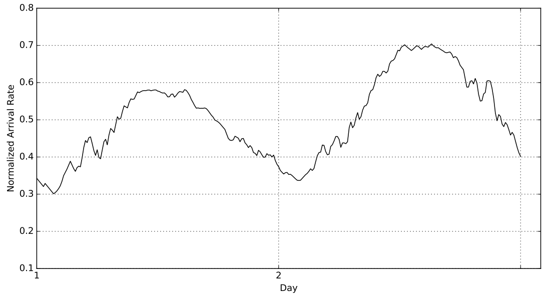


**Figure 5.** Two-day historical arrival rates in the data center

## 4.2 Results and Discussions

Table 1 compares the total cost of the data center in terms of baseline operation and multi-market optimization. The baseline system is denoted as *Base*, and the multi-market optimization is denoted as *OPT*. In both January and July, the data center without energy storage systems, using the proposed optimization framework, can benefit from participating in both energy market and regulation market. In the two days considered, *OPT* can save \$123.6 in July, while the saving is \$24.8 in January.

The savings mainly come from the revenues in the regulation market, and the cost for energy use and demand charge are almost the same in the *Base* and *OPT*. Because the sum of the RegD signal over a long time period (e.g. 1 hour) is almost 0, providing regulation service in the *OPT* leads to the similar energy use, thus similar energy cost compared with the *Base* where no regulation service is provided. By utilizing the demand cost defined in Eq. (25), the data center can provide regulation service without increasing monthly demand, thus no extra demand

**Table 1.** Multi-market optimization of data centers

| Costs | January | | July | |
|---|---|---|---|---|
| | *Base* | *OPT* | *Base* | *OPT* |
| Energy Cost ($) | 1043.3 | 1042.9 | 1591.4 | 1590.6 |
| Demand Cost ($) | 10459.3 | 10457.5 | 12063.9 | 12062.8 |
| Regulation Revenue ($) | | 22.6 | | 121.7 |
| Total Cost ($) | 11502.6 | 11477.8 | 13655.3 | 13531.7 |
| Total Savings ($) | | 24.8 | | 123.6 |

charge would be added to utility bills. The revenue from July is much higher than that in January because the price for dynamic regulation (RegD) resources is higher in July. As shown in Figure 4, during the studied two days, the average price from regulation market in July is about 21 $/MW, while that in January is only about 5.8 $/MW.

Figure 6 shows the hourly capacity bids in 1/21 and 7/21. The demand for each 30 minutes is denoted as the thin solid line. The demand limit used for demand cost as shown in Eq. (25) is denoted as the dashed line. The optimal capacity bid at each hour is denoted as the shaded area. At non-peak hours (e.g., 3:00 - 6:00), the optimal bid is mainly influenced by the price from energy market, price from regulation market and detailed shape of RegD signal. Because the demand is lower than the demand limit, the tradeoff between the energy cost and revenues from regulation market determines the optimal bid. The energy cost is highly influenced by the energy use, which is determined by the detailed shape of the RegD signal. If the sum of the RegD signal is larger than 0, then more energy would be consumed when providing frequency regulation service, thus the energy cost would increase. Although the energy cost increases in this case, the data center can get revenues from regulation market. If the sum of the RegD signal is no larger than 0, then at that hour, the data center can bid at their maximum capacity.

At peak hours (e.g., 12:00 - 16:00), the optimal bid is mostly influenced by the demand limit and the RegD signal. Figure 6 shows that at these hours, the bid is small so that the demand cannot exceed the required demand limit to avoid demand penalty. At 13:00, the bid is about 69 kW, but it is only about 5 kW at 14:00. The difference is caused by the detailed shapes of the RegD signals in these two hours. At 13:00, the sum of the RegD signal in first 30 minutes is slightly greater than 0, but in the second 30 minutes it is much smaller than 0. This means that regulation capacity bid in this hour can increase the demand in the first 30 minutes, but the demand in the second 30 minutes can be decreased compared with the same time in the baseline system. Therefore, at this hour, the data center can bid a large capacity as long as the demand in the first 30 minutes will not exceed the demand limit. The same situation happens at 14:00 but with a large sum of RegD signal at first 30 minutes. Also because the power at 14:00 is much closer to the demand limit, the data center can only bid a small capacity at this hour.

In summary, the proposed real-time optimization framework can help the data center without energy storage system harness the benefits from the energy market and the regulation market. However, the benefits are insignificant compared with the large baseline power in data centers. One of the reason is that data centers without energy storage system are difficult to limit their power demand during FR service, which contributes to a large portion of the utility bill. In the future, we will consider retrofit strategy (e.g., installing thermal storage energy system) in the data center to limit the power demand to maximize the benefit from the multi-markets

# 5 Conclusions

This paper developed a real-time multi-market optimization framework for the data center without storage systems to maximize their benefits from participating in both energy market and regulation market. Then, a case study was conducted to numerically investigate the optimal bids at each hour by considering the energy cost, demand costs and regulation revenues using a virtual data center located in PJM. Simulation results shows that using the proposed multi-market optimization framework can minimize the operational cost. Compared with the baseline system, providing frequency regulation service over the considered two days can save $24.8 in January and $123.6 in July.

# 6 Appendix

## 6.1 FR Performance Score

In the PJM market, new resources aiming to enter the regulation market need to pass an initial test by obtaining at least 0.75 for a defined performance score. The initial test signals of RegA and RegD are available at (PJM, 2019). The performance score is calculated as a composite score of accuracy, delay and precision, which are shown below (LLC, 2019).

$$c_{sig,res} = \frac{COV(reg, res)}{\sigma_{reg}\sigma_{res}} \tag{27}$$

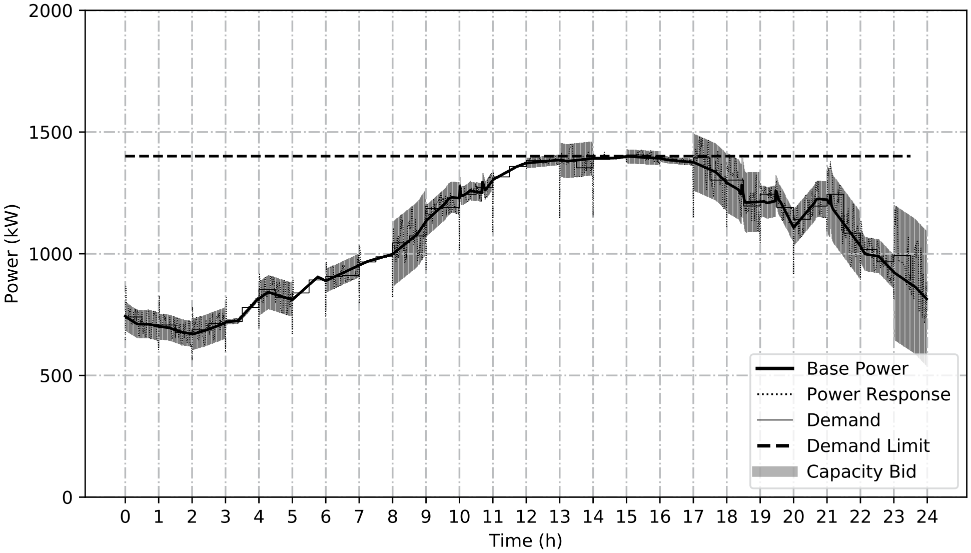
$$S_{accuracy} = \max_{\delta=0-5\text{ min}} (c_{reg,res(\delta)}) \tag{28}$$

$$S_{delay} = \left| \frac{5\text{ min} - \delta^*}{5\text{ min}} \right| \tag{29}$$

$$S_{precision} = 1 - \frac{1}{n} \sum \left| \frac{res - reg}{\overline{reg}} \right| \tag{30}$$

$$S = \frac{S_{accuracy} + S_{delay} + S_{precision}}{3} \tag{31}$$

In the above equations, *reg* represents the regulation signal the DSRs receive from the electrical markets, and *res* represents the response signal the DSRs generate after control actions. $c$, $COV$ and $\sigma$ are the correlation coefficient, covariance, standard deviation of these two signals. In PJM, the response signal *res* is recalculated with a time shift $\delta$ ranging from 0 to 5 minutes in an increment of 10 seconds, which leads to 31 response signals $res(\delta)$. The accuracy score $S_{accuracy}$ is the maximum correlation coefficient $c$ between *reg* and $res(\delta)$. The delay score $S_{delay}$ is calculated based on the delay time $\delta^*$ when the maximum accuracy score is obtained using Eq. (29). The precision score $S_{precision}$ is defined as the relative difference between regulation signal and response signal, where $n$ is the number of samples in the hour, and $\overline{reg}$ is the hourly average regulation signal. The final performance score $S$ in that hour is calculated as the weighted average of the three individual scores.



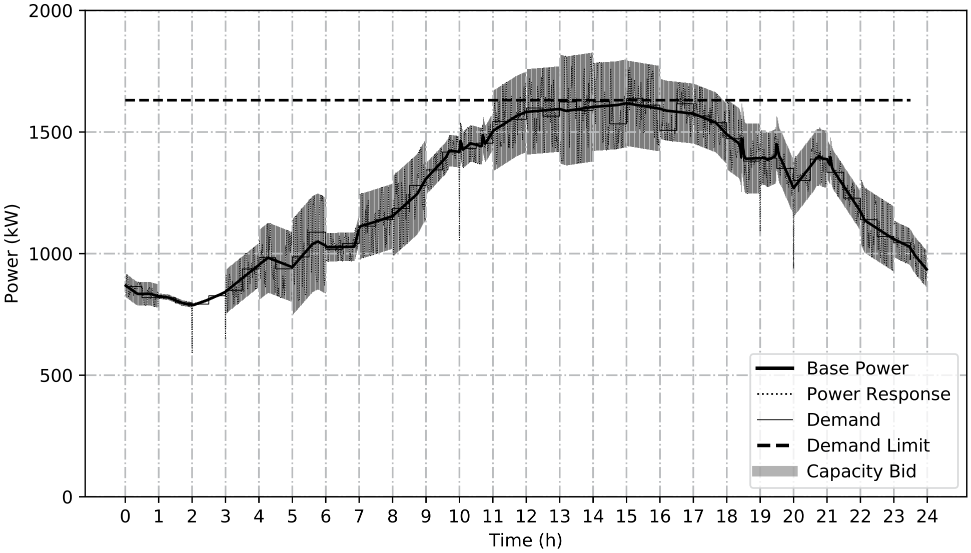


**Figure 6.** Optimal hourly regulation capacity bids in 1/21 (top) and 7/21 (bottom)

# References

Natalie Bates, Girish Ghatikar, Ghaleb Abdulla, Gregory A Koenig, Sridutt Bhalachandra, Mehdi Sheikhalishahi, Tapasya Patki, Barry Rountree, and Stephen Poole. Electrical grid and supercomputing centers: An investigative analysis of emerging opportunities and challenges. *Informatik-Spektrum*, 38(2):111–127, 2015.

Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

Yangyang Fu, Michael Wetter, and Wangda Zuo. Modelica models for data center cooling systems. In *2018 Building Performance Analysis Conference and SimBuild, Chicago, Illinois, United States of America*, 2018.

Yangyang Fu, Xing Lu, and Wangda Zuo. Modelica models for the control evaluations of chilled water system with waterside economizer. In *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4ñ6, 2019*, number 157, page 8. Link^ping University Electronic Press, Link^pings universitet, 2019a.

Yangyang Fu, Wangda Zuo, Michael Wetter, James W. VanGilder, and Peilin Yang. Equation-based object-oriented modeling and simulation of data center cooling systems. *Energy and Buildings*, 198:503 – 519, 2019b. ISSN 0378-7788. doi:https://doi.org/10.1016/j.enbuild.2019.06.037. URL `http://www.sciencedirect.com/science/article/pii/S0378778819307078`.

Yangyang Fu, Wangda Zuo, Michael Wetter, Jim W. VanGilder, Xu Han, and David Plamondon. Equation-based object-oriented modeling and simulation for data center cooling: A case study. *Energy and Buildings*, 186:108 – 125, 2019c. ISSN 0378-7788. doi:https://doi.org/10.1016/j.enbuild.2019.01.018. URL `http://www.sciencedirect.com/science/article/pii/S0378778818330573`.

Sen Li, Marco Brocanelli, Wei Zhang, and Xiaorui Wang. Data center power control for frequency regulation. In *2013 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2013.

PJM Interconnection LLC. Pjm manual 11: Energy & ancillary services market operations, 2019.

PJM. Ancillary services, 2019. URL `https://www.pjm.com/markets-and-operations/ancillary-services.aspx`.

Telecommunication Industry Association. Standards, Technology Dept, and American National Standards Institute. *Telecommunications Infrastructure Standard for Data Centers*. Telecommunication Industry Association, 2005.

Wei Wang, Amirali Abdolrashidi, Nanpeng Yu, and Daniel Wong. Frequency regulation service provision in data center with computational flexibility. *Applied Energy*, 251:113304, 2019.

Michael Wetter et al. Genopt-a generic optimization program. In *Seventh International IBPSA Conference, Rio de Janeiro*, pages 601–608, 2001.

Adam Wierman, Zhenhua Liu, Iris Liu, and Hamed Mohsenian-Rad. Opportunities and challenges for data center demand response. In *International Green Computing Conference*, pages 1–10. IEEE, 2014.

# Micro-grid Design and Cost Optimization using Modelica

Natesa MacRae[1] John Batteh[2] Stéphane Velut[3] William Skrivan[1]
Imran Khan[1] Darren Jang[1]

[1]National Research Council Canada, Canada, {Natesa.MacRae, Imran.Khan, William.Skrivan, Darren.Jang}@nrc-cnrc.gc.ca
[2]Modelon Inc., USA, {john.batteh}@modelon.com
[3]Modelon AB, Sweden, {stephane.velut}@modelon.com

## Abstract

This paper describes the modeling of the National Research Council of Canada (NRC) Micro-Grid Testing and Training Facility, which will be used to advance energy generation and storage technologies and optimize integrated system operations for a variety of micro-grid applications. First, integrated system models were assembled using the Microgrid Modelica library, developed by Modelon. Next, three use cases were defined based on a scaled down version of the Whale Cove, Nunavut micro-grid operating in: 'island mode' without renewables (present mode), island mode *with* renewables, and grid-connected mode. Various simulations, as well as design and economic optimizations were then performed. Through these analyses, it was shown that each parameter domain could be successfully assessed using this modeling framework, demonstrating the flexibility of both the modeling platform and the potential of the physical test facility to support in-depth analysis for different micro-grid configurations, technologies, and applications.

*Keywords: Micro-grid, remote area, distributed energy resources, renewables, model, optimization.*

## 1 Introduction

Canada has an estimated 280 remote communities and commercial sites where power is predominantly supplied by standalone diesel generators (NRCan 1, 2018). Because of their high reliance on diesel, these remote micro-grids generate significant emissions and are expensive to operate. Introducing renewable energy sources provides an opportunity to reduce both operating costs and emissions but, due to their variable energy output, they can add operational complexity with respect to maintaining the overall control and stability of the electrical system. New and emerging technologies are continuously being developed to address these and other challenges, but in order to reduce risk during deployment, it is essential that both the technologies and their applications are well understood. Even then, the successful integration of these technologies is not guaranteed, particularly if they require a high initial investment, have limited reliability, are not easy to operate, maintain, or repair, or if sufficient training is not provided.

Presently, a substantial amount of research is being directed at analyzing new and innovative micro-grid configurations and technologies. This includes evaluating micro-grids operating as a stand-alone system, as a building block in a flexible electric micro-grid 'network' consisting of various distributed energy resources and customer loads, or as a compliment to a centralized grid. Particularly, in the case of micro-grid / grid integration, challenges associated with switching between grid-connected and island modes, as well as reliability, power quality, and protection requirements, have received limited investigation (Ackeby, 2017). In all cases, these new and emerging components and micro-grid configurations will need to be analyzed and demonstrated to ensure that they are safe, reliable, and can meet the strict performance and operational requirements of the communities or applications they support. For this reason, the NRC is currently building a *physical* micro-grid testing and training facility as well as a complimentary *virtual* facility prototyping capability using the Microgrid Modelica library developed by Modelon. Although this initial study has focused on high level design and cost optimization, future model developments will include the use of Modelon's Electric Power Modelica library, for control, stability, and transient analysis (Modelon, 2019).

## 2 Background

### 2.1 Micro-Grid Facility

The NRC micro-grid testing and training facility has been designed to enable the analysis of the systems / technologies that support remote community micro-grids, grid-connected stand-by power plants, and off-grid residential, military, or commercial sites. This facility will allow the flexible integration of a range of power generation and storage technologies into an existing power network, to support their assessment under a variety of real-world conditions. Testing will include evaluating different power / energy configurations, simulating transient conditions and events, supporting micro-grid control system design and optimization, and performing accelerated lifetime testing to optimize the reliability of the system and components. Results of this testing will support the understanding, advancement, and deployment of micro-grid technologies, interfaces, and configurations, and will be used to inform new policies and safety regulations. The facility also offers a reduced-risk training environment for personnel to familiarize themselves with the operation and maintenance of new systems and provide practical feedback to technology developers.

The facility power network will consist of a unique set of distributed energy resources (NRC, 2019) including a:

- Biomass Combined Heat and Power (CHP) unit, ~40 kW of electricity ($kW_e$) and ~100 kW of heat ($kW_{th}$)
- Flat-panel (~10 kW) building-integrated solar photovoltaic (PV) system
- Concentrating mirror (~10 kW) Photovoltaic (CPV) system
- Diesel generator
- Energy storage system

The facility will also have the ability to operate in grid-connected or island mode (i.e., as an isolated micro-grid).

### 2.1.1 Biomass Combined Heat and Power (CHP) Unit

The Biomass CHP unit, shown in Figure 1, uses renewable biomass as an alternative to fossil fuels to generate both heat and power. The unit converts wood chips to synthetic gas (“syngas”), which is then cooled, filtered, and mixed with air before being supplied to an internal combustion engine. The engine is coupled to an electric generator to produce ~40 $kW_e$, as well as ~100 $kW_{th}$ as warm water (Volter, 2019).

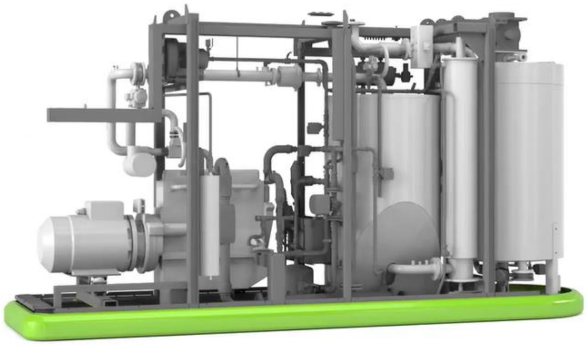


**Figure 1.** Biomass Combined Heat and Power Unit (Volter, 2019)

Biomass can be viewed as carbon neutral, in that the $CO_2$ the biomass removes from its environment during growth is equal to the $CO_2$ released during its combustion, resulting in ‘net-zero’ carbon emissions; however, because carbon storage in wood products only occurs gradually over a long period of time, compared with the rapid release of $CO_2$ that occurs when these feedstocks are processed, the environmental benefits of this approach are still the subject of much debate (Harvey, 2018). For this reason, only low value biomass feedstocks are typically used (e.g., wood chips or pellets generated from forestry waste streams).

### 2.1.2 Diesel Generator (DG)

The NRC micro-grid facility will include a variable speed diesel generator. Although not yet selected, a candidate 80 kWe diesel generator is shown in Figure 2.

**Figure 2.** Variable Speed 80 kW Diesel Generator (Caterpillar, 2010)

Conventional diesel generators are the most common and reliable systems for power generation; however, due to low ramp rates, fixed speed generators may struggle to adjust to the variable power output from renewables (e.g., PVs and wind turbines). Fixed speed generators also have a minimum loading requirement (typically 30-50% of maximum loading) that they cannot operate below without reducing the life of the generator, preventing these systems from being significantly ramped down to reduce fuel consumption and / or accommodate an increase in renewable energy production. In addition, the fuel efficiency of a fixed speed diesel generator drops significantly when the system is operating at or near minimum loading. (NRCan 1, 2018).

Although more complex than conventional fixed speed diesel generators, variable speed generators have the ability to ramp up quickly and efficiently, and operate more efficiently at lower minimum loads compared to fixed speed generators. They also provide greater flexibility for simulating different test conditions in the context of the micro-grid testing facility.

### 2.1.3 Solar Photovoltaic Systems

The solar PV arrays, shown in Figure 3, include both a concentrating mirror PV (CPV) system and a conventional building-integrated flat panel PV system. Where conventional PV cells generate electricity from both direct and diffuse radiation, CPV systems use stationary mirrors to concentrate a large area of direct sunlight onto a small area of (typically) higher capacity PV cells to generate electricity. Throughout the day, the solar cell collector moves based on the sun's position to maximize the direct sunlight that can be collected.

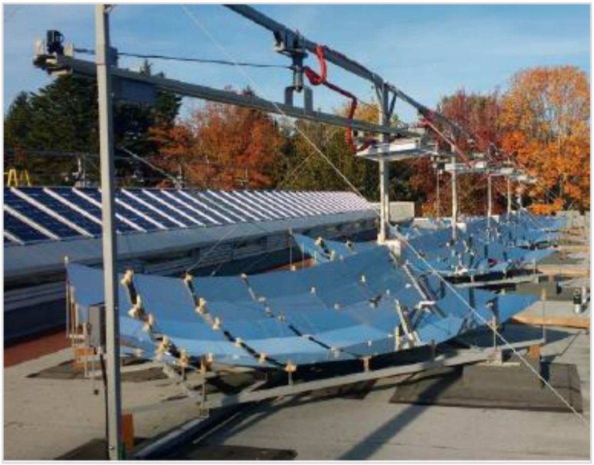


**Figure 3.** NRC concentrating mirror and flat panel solar PV systems

The advantages of CPV systems is that they require fewer expensive solar photovoltaic cells relative to flat panel PVs to generate the same amount of electricity. Disadvantages of CPV systems include the need for moving parts (compared with non-tracking PV systems) and large amounts of direct solar radiation. During cloud cover, CPVs will experience a significant drop in energy production, whereas a conventional PV system will still produce electricity from diffuse radiation (Kraemer, 2017). By including both types of PV systems in the facility power network, both approaches can be assessed to determine how each technology can be used to best support optimal micro-grid operation.

Challenges for all solar power generation systems include changes in the availability of sunlight that can occur seasonally (e.g., in northern communities, there is little to no sunlight during the winter). Under more favorable conditions, energy production can still vary on the order of seconds to minutes as the result of shading caused by cloud cover, resulting in momentary energy shortfalls that need to be made up by other energy sources. For a surplus of solar energy production, if using a fixed speed diesel generator, PV output may need to be curtailed or wasted because the generator is operating at minimum loading and cannot be ramped down further to accommodate the additional energy. This is where incorporating energy storage systems, such as batteries, can improve both the flexibility, performance, and reliability of micro-grid operation. (NRCan 1, 2018).

### 2.1.4 Energy Storage System (ESS)

The ESS can play a key role for micro-grid systems that incorporate renewable energy sources. For example, batteries have the ability to efficiently store surplus energy (from renewables as well as from diesel generator systems) and reliably provide energy during an energy shortfall at rates that a conventional fixed speed generator cannot. In this way, batteries can rapidly balance power generation with demand, resulting in improved power quality, system flexibility, and stability.

**Figure 4.** Battery Energy Storage System (single module shown) (EV Shop, 2019)

Although the ESS for the micro-grid facility has not yet been selected, for the purpose of this study, a battery system consisting of eight series-connected 12S1P modules with 120Ah lithium ion cells has been modeled (see Figure 4). Based on NRC's evaluation of these batteries, they have been specified to have an available discharge rate of up to 2C, a charge rate of 2C (between 20-80% state of charge (SoC)) and C/3 (>80% SoC), a nominal pack voltage of 352 V, and a total pack energy of 42.2 kWh.

## 2.2 Micro-Grid Use Cases

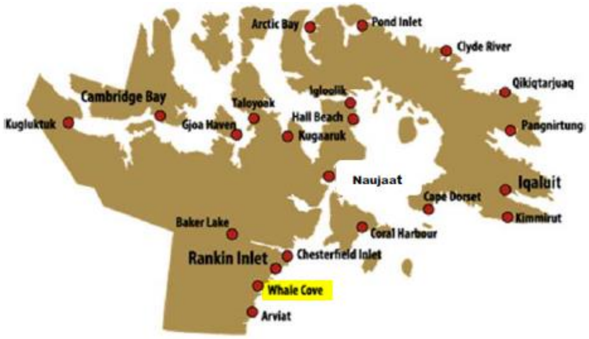


**Figure 5.** Qulliq Energy Corporation (QEC) partial service area map showing Whale Cove (QEC, 2018)

For the purposes of this study, three use cases were defined and analyzed based on a remote northern community: Whale Cove, Nunavut (see Figure 5):

1. 'Island mode' operation of the Whale Cove micro-grid without renewables (present mode).
2. 'Island mode' operation of the Whale Cove micro-grid *with* renewables (and energy storage).
3. Operation of the Whale Cove micro-grid including renewables and energy storage *with* a grid connection to the Manitoba Electrical Grid.

For the use cases that involve renewables, technologies that are currently integrated into the NRC facility power network were explored to see how they might benefit the existing Whale Cove micro-grid. Multiple design and economic optimizations were performed.

## 2.3 Micro-Grid Integrated System Model

Although there are multiple modeling tools available for micro-grid design, simulation, and optimization, the platform selected to model the micro-grid test and training facility was the commercial Microgrid Modelica library developed by Modelon, a web based modeling and simulation platform that uses the Optimica Compiler Toolkit for model simulation and optimization (Windahl, 2019). This platform was selected based on its flexible, multi-physics, and highly customizable modeling / optimization framework, its ability to accommodate models of varying levels of fidelity, its ability to provide physical (rather than only mathematical) representation of micro-grid components, and its ability to support acausal analysis. Note that a review of this and other modeling and optimization tools has been provided by (Windahl, 2019).

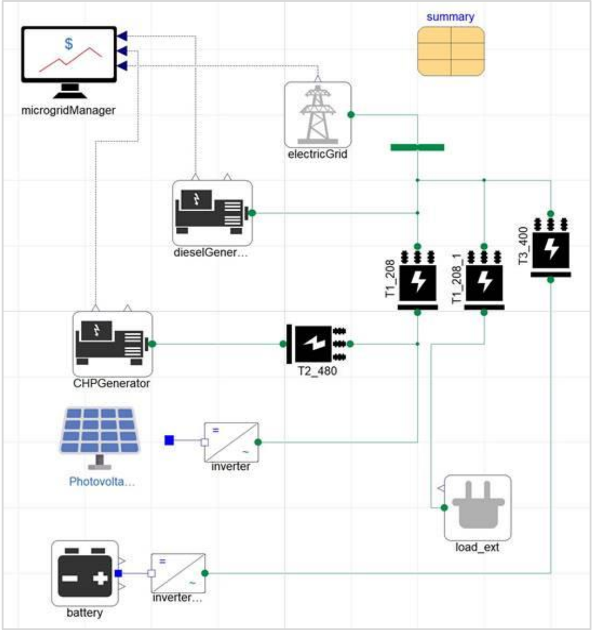


**Figure 6.** Micro-Grid Integrated System Model

Using the Microgrid library component models, an integrated system model (see Figure 6) was assembled including a diesel generator model, PV model, battery energy storage system model, a generator model for the biomass CHP system, and a simplified representation of a grid connection, conversion components, and a configurable resistive load.

### 2.3.1 Diesel Generator

The diesel generator model can be configured to represent both a fixed or variable speed diesel generator, as AC power is generated based on a control input signal and representative fuel consumption curve. The operational differences between the two types of generators can be defined as a function of acceptable ramp rates and load limits using the 'microgridManager' (see Section 2.3.8). For the purposes of this study, however, the diesel generator was modeled as a fixed speed generator (to best represent the generators currently operating at Whale Cove) and scaled to the maximum capacity of the NRC micro-grid facility.

Three generators currently provide power to Whale Cove: two 300 kW Caterpillar D3412 units, and one 150 kW Caterpillar D3406 unit (Nunavut, 2001). Although it is likely that at least one of these generators operates in standby at any given time, for simplicity, fuel consumption correlations from (Das, 2017) were used to compute the total fuel consumption (L/h) at max loading (100% capacity) and min loading (30% capacity) for all three generators. This value was then scaled to the rated capacity of the NRC micro-grid diesel generator to provide an equivalent fuel consumption correlation for the three generators operating as a single diesel generator (see Figure 7).

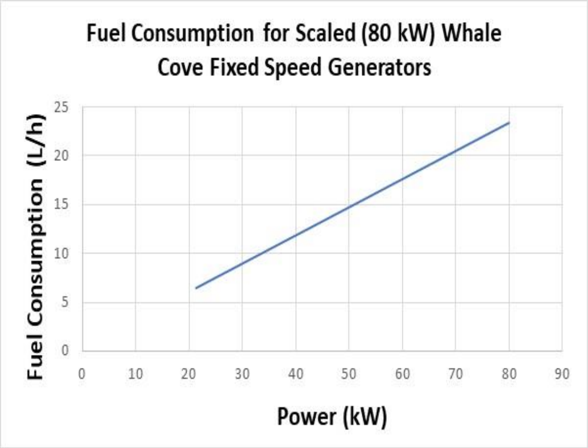


**Figure 7.** Diesel Fuel Consumption for Scaled (80 kW) Micro-grid Fixed Speed Generator

The parameters for the diesel generator model are summarized in Table 1. Although the peak load defined for Whale Cove is 402 kW, with the generator operating at a load factor of 56% (QEC, 2017), a scaled peak load was used based on an 80 kW generator operating at 56% loading (44.8 kW).

**Table 1.** Diesel Generator Parameters and Constraints

| *Whale Cove* | *Value* |
|---|---|
| Rated Capacity[2] | 750 kW |
| Remaining Capacity[1] | 718 kW |
| Max Load[1] | 402 kW |
| Load factor at Max Load[1] | 56% |
| *Micro-grid Test Facility* | *Value* |
| Max Capacity | 80 kW |
| Generator Min Loading | 30% |
| Max Load (at 56% load factor) | 44.8 kW |
| Min Load (at 30% gen. min loading) | 24.0 kW |
| Fuel Consumption at Max Capacity[2] | 23.4 L/h |
| Fuel Consumption at Max Load[2] | 13.2 L/h |
| Fuel Consumption at Min Load[2] | 7.2 L/h |

[1](QEC, 2017), [2](Nunavut, 2001) and (Das, 2017)

From (NRCan 1, 2018), the capacity to add renewables to this micro-grid was taken as the difference between the min loading output of the scaled diesel generator (24.0 kW) and the max scaled loading for Whale Cove (44.8 kW). Based on this, the resulting combined PV and CHP maximum power output was limited to ~ 21 kW.

For cost optimizations, the total cost of diesel based electricity generation was $0.827 per kWh (QEC, 2017).

### 2.3.2 Solar PV Model

The Micro-Grid solar PV model can be configured to represent both flat panel PV and CPV systems, using panel surface area, solar irradiance data, system capacity, and efficiency; however, for the purposes of this study, one system model was developed to represent the combined capacity of the PV and CPV systems. DC power estimates were used based on actual solar irradiance data from (NREL, 2019) for a flat panel solar installation at Rankin Inlet (located in close proximity to Whale Cove, see Figure 5) and scaled to produce the desired net PV output (see Figure 8).

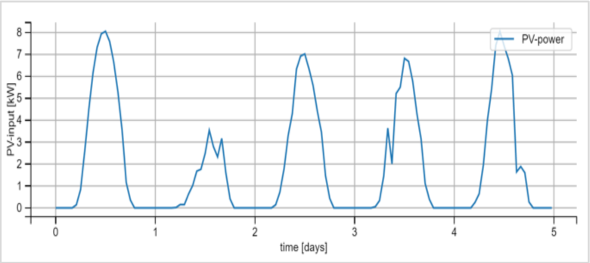


**Figure 8.** PV Power Output Approximation for Whale Cove (NREL, 2019) – Scaled for NRC Micro-Grid

Note that temperature effects on solar cell power generation were not considered.

### 2.3.3 Battery System Model

The battery system model was defined based on battery capacity, minimum and maximum state of charge, and maximum charge and discharge rates. These and other relevant battery system data are shown in Table 2. Note that a detailed thermal model was not included.

**Table 2.** Battery Model Parameters

| *Parameter Description* | *Value* |
|---|---|
| Nominal Capacity[1] | 120 Ah |
| Max SoC | 95 % |
| Min SoC | 20 % |
| Discharge C-Rate[1] | 2C |
| Charge C-Rate (20-80% SoC) [1] | 2C |
| Charge C-Rate (>80% SoC) [1] | C/3 |
| Total Pack Energy Capacity[1] | 42.2 kWh |
| Voltage[1] | 352 V |
| Configuration | 12S1P |

[1]Data based on the characterization of a real-world battery system performed at the NRC Vancouver battery test facility.

### 2.3.4 Biomass CHP Model

Parameters for the CHP system are shown in Table 3. For the purposes of this study, the biomass CHP model was approximated as an electric generator only (i.e., thermal energy production was not considered). The unit power output was scaled (constrained) to ~ 13.8 kW, 3 phase, 480 VAC power. Fuel consumption was taken as 1:1 with energy output (i.e., 1 kg/h biomass to generate 1 kWh electricity) with an efficiency of 20%. For cost optimizations, CHP based electricity generation was estimated at $0.20 per kWh.

**Table 3.** CHP Model Parameters

| *Micro-grid Test Facility* | *Value* |
|---|---|
| Rated CHP capacity[1] | 40 kW |
| *Scaled CHP system* | *Value* |
| Rated CHP capacity | 13.8 kW |
| Fuel consumption[1] | 13.8 kg / hr |
| Electric Efficiency[1] | 20 % |
| Fuel cost | $0.20 / kWh |

[1](Volter, 2019)

### 2.3.5 Ideal Grid Model

For the use case where the Whale Cove micro-grid includes a connection to the Manitoba electrical grid, an ideal grid model was used (constant voltage and frequency). The Manitoba grid capacity was constrained at 45% of the full power demand for Whale Cove, with a cost of $0.13 per kWh (see Table 4). Note that to install the 1,000 km of transmission lines needed to complete the grid connection to multiple Nunavut communities in the area, the estimated cost (in 2015) was ~ $900 million, with $40 million in diesel savings estimated per year. This cost is assumed to have been reflected in the $0.13 per kWh rate, given an estimated purchase price of $0.075 per kWh for QEC from Manitoba Hydro and the project's estimated 40 year lifetime to achieve a return on investment. (Karanasios, 2016).

**Table 4.** Grid Parameters

| *Manitoba Grid Connection* | *Value* |
|---|---|
| Rated Grid capacity | 45% |
| Grid energy costs[1] | $0.13 / kWh |

[1](Karanasios, 2016).

### 2.3.6 Load Model

The load for each simulation was modeled as a variable input following a real-world load profile derived from an NRC building on March 3, 2016 (see Figure 9). This load data was scaled to allow the reduced load capacity of the micro-grid diesel generator (~80 kW) to be used to provide the baseline for evaluating the impact of introducing consumables into a Whale Cove power grid.

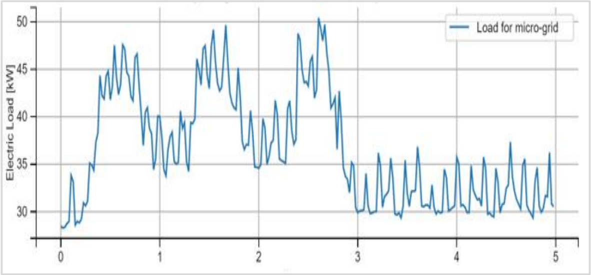


**Figure 9.** Load Profile for NRC-Van Building Mar, 2019

Note that the load data above has been used to represent 'scaled' daily use load variability only and does *not* reflect actual load profiles for Whale Cove, both in terms of daily use or seasonal load variations.

### 2.3.7 Transformer and Inverter Models

Transformers and inverters were characterized using efficiency based models, selectable for voltage or current type units. For this study, voltage units were used and all efficiencies were assumed to be 95%.

### 2.3.8 Micro-grid Manager

The micro-grid manager contains the control rules and constraints for the various controllable integrated micro-grid components. Using external sources, this manager was also used to support micro-grid optimization. Details related to configuration of the control algorithm for each simulation / optimization has been defined in each corresponding analysis below.

# 3 Micro-Grid Analysis

## 3.1 Use Case 1 Simulation

The first simulation performed evaluated the present diesel generator's ability to meet the power demand at Whale Cove. This simulation was performed as an initial validation of the integrated system model and to establish a baseline for cost and performance comparisons with subsequent use cases.

## 3.2 Use Case 2 Simulation

The second analysis was based on the present operating configuration at Whale Cove *with* renewables and energy storage added. This includes the biomass CHP system, flat panel PVs, CPVs, and batteries (with all components specified and scaled in relation to the NRC micro-grid test facility configuration).

The first analysis centered around the system's ability to match power generation and load curves based on the test facility configuration and a rule based simulation approach. For micro-grid reliability, the diesel generator was operated at minimum loading or higher, with the CHP system and combined PVs / CPVs meeting the remaining demand. Batteries were used to provide any additional power where there was power shortfall from all other sources. Once fully discharged, batteries were recharged from available sources.

The second analysis looked at the optimization of the micro-grid configuration with renewables and energy storage, to see how it could further improve upon the rule based simulation approach.

## 3.3 Use Case 3 Optimization and Analysis

The third use case included the addition of a grid connection (to the Manitoba electric grid). First an economic optimization was performed with electricity rates of $0.827 / kWh for diesel and $0.13 / kWh for the grid, and grid supply constrained to 20 kW. Using the same model, a design optimization was performed, where battery size was a degree of freedom.

# 4 Results & Discussion

## 4.1 Use Case 1 Simulation

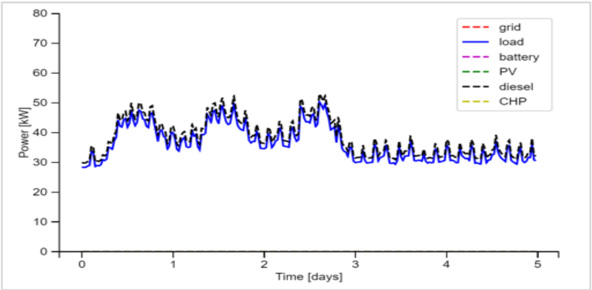


**Figure 10.** Use Case 1 – Whale Cove power generation using diesel generators only

The results for the use case 1 simulation are shown in Figure 10. Diesel power is shown to correctly track load over the course of the simulation, with a slightly higher power output level than demand (to cover transformer losses).

Based on the diesel electricity generation cost of $0.827 per kWh, the total baseline cost for (scaled) power using the current configuration is ~ $3800. Note that these costs are only defined for comparison purposes with other use cases (as the load data used is not representative of Whale Cove).

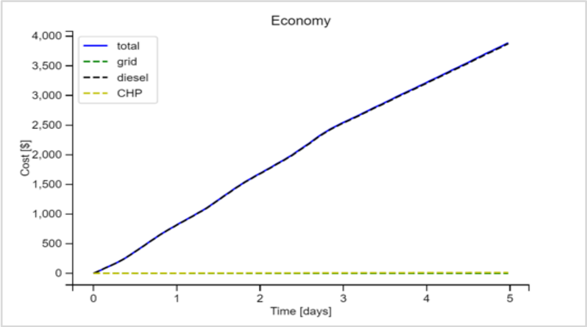


**Figure 11.** Use Case 1 Simulation – Baseline cost for diesel only power generation

## 4.2 Use Case 2 Simulation and Optimization

The results for the use case 2 simulation are shown in Figure 12.

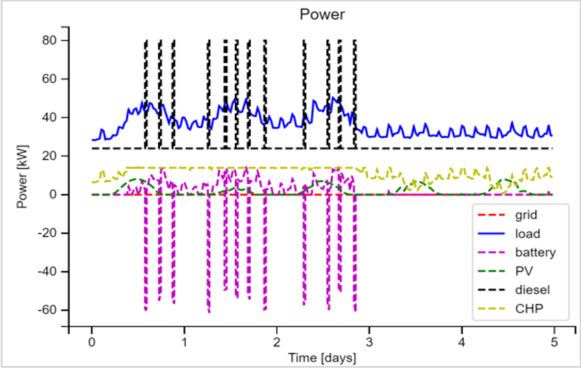


**Figure 12.** Use Case 2 Simulation – Whale Cove micro-grid with renewables and energy storage

The simulation above was configured to:

- Keep diesel generator operation to a minimum (not less than 30%, to prevent damage to the generator) to ensure the reliability of the remote community micro-grid.
- Use PV power directly, as generated.
- Engage the CHP and ESS systems, as required, to provide the remaining power.
- No contribution from the grid (island mode).
- Recharge the battery using available sources when the battery was discharged to its lower SoC limit (subject to C-rate constraints) – see Figure 13.

During high demand periods (Day 1 to 3), it can be seen that the battery is cycled much more than the on the weekend (Day 4 and 5), where the diesel generator, CHP, and PV systems are able to satisfy most of the demand. Also, where maximum PV and CHP capacity were required to satisfy the load during high demand periods, the diesel generator was required to ramp up to full capacity in order to recharge the batteries (where the battery power drops below zero, the battery is charging).

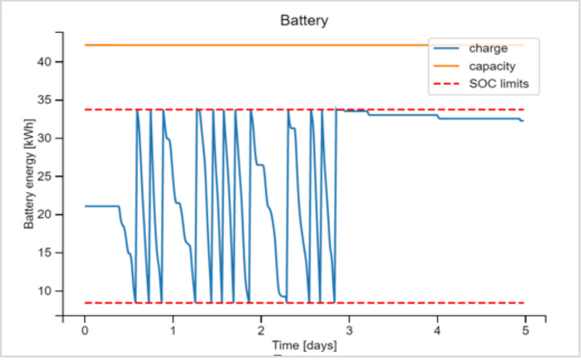


**Figure 13.** Use Case 2 Simulation - Battery charge and discharge cycles

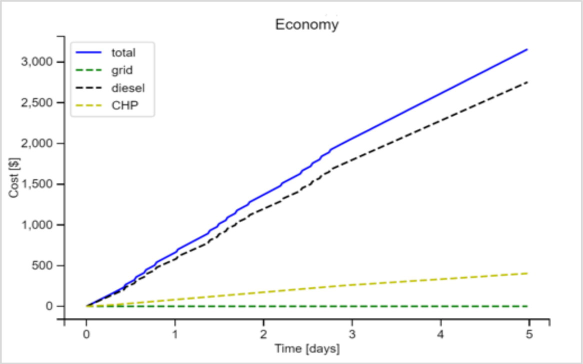
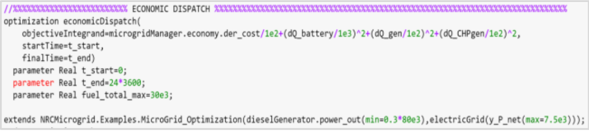


**Figure 14.** Use Case 2 Simulation – Cost of micro-grid operation with renewables and energy storage added

Based on the diesel generation cost of $0.827 / kWh and a CHP cost of $0.20 / kWh, the new *total* cost to satisfy the same demand dropped to $3200 (almost 16%). Note that this was a rule based simulation (not an optimization).

For the economic optimization of the use case 2 configuration, the objective function shown in Figure 15 was used:



```
//%%%%%%%%%%%%%%%%%%%%%%%%% ECONOMIC DISPATCH %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
optimization economicDispatch(
    objectiveIntegrand=microgridManager.economy.der_cost/1e2+(dQ_battery/1e3)^2+(dQ_gen/1e2)^2+(dQ_CHPgen/1e2)^2,
    startTime=t_start,
    finalTime=t_end)
  parameter Real t_start=0;
  parameter Real t_end=24*3600;
  parameter Real fuel_total_max=30e3;

extends NRCMicrogrid.Examples.MicroGrid_Optimization(dieselGenerator.power_out(min=0.3*80e3),electricGrid(y_P_net(max=7.5e3)));
```

**Figure 15.** Use Case 2 Optimization – Economic Objective Function (placeholder)

The results from the optimization of the micro-grid with renewables and energy storage are shown in Figure 16 (load balancing) and Figure 17 (economic analysis).

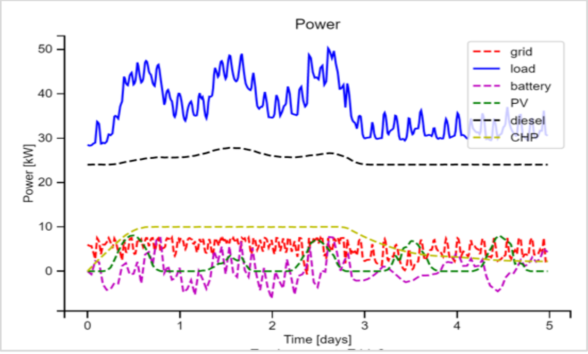


**Figure 16.** Use Case 2 Optimization (placeholder)

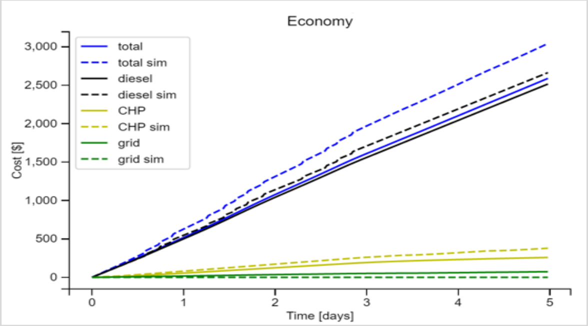
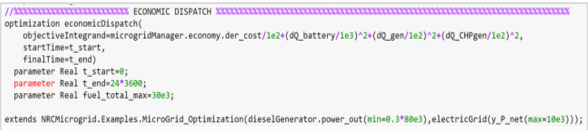


**Figure 17.** Comparative cost (placeholder) of Use Case 2 sim (dashed lines) and Use Case 2 optimization (solid)

It can be seen that the optimization reduces the total cost to $XX00 compared to the rule based simulation (~XX% compared to diesel only and ~XX% compared to the rule based simulation).

## 4.3 Use Case 3 Optimization and Analysis

The objective function for the use case 3 economic optimization is shown in Figure 18.



```
//%%%%%%%%%%%%%%%%%%%%%%%%% ECONOMIC DISPATCH %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
optimization economicDispatch(
    objectiveIntegrand=microgridManager.economy.der_cost/1e2+(dQ_battery/1e3)^2+(dQ_gen/1e2)^2+(dQ_CHPgen/1e2)^2,
    startTime=t_start,
    finalTime=t_end)
  parameter Real t_start=0;
  parameter Real t_end=24*3600;
  parameter Real fuel_total_max=30e3;

extends NRCMicrogrid.Examples.MicroGrid_Optimization(dieselGenerator.power_out(min=0.3*80e3),electricGrid(y_P_net(max=10e3)));
```

**Figure 18.** Use Case 3 Economic Optimization Objective Function (placeholder)

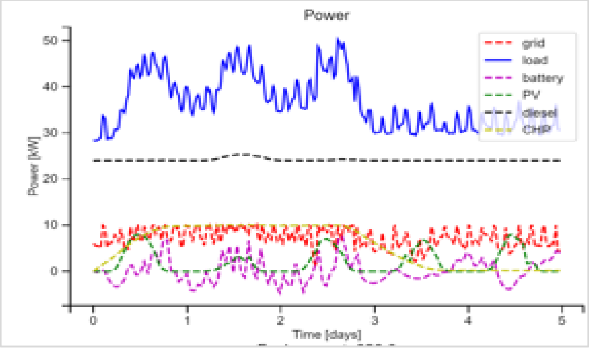


**Figure 19.** Use Case 3 Econ. Optimization – Whale Cove micro-grid with renewables, energy storage, and grid tie

The results from the economic optimization for use case 3 are shown in Figure 19. It can be seen that the micro-grid system is almost able to completely satisfy the demand using renewables, energy storage and the grid, without increasing the diesel generator operation beyond minimum loading (30%). Also, the CHP system operation is much less erratic (smooth ramp up and down).

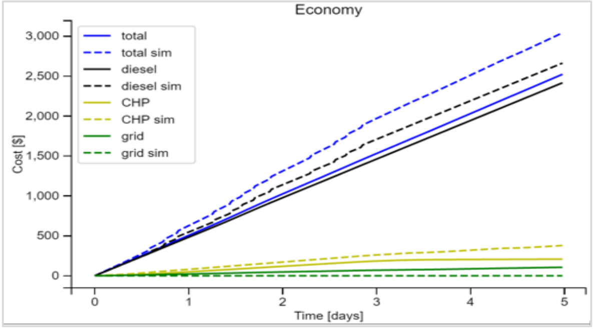
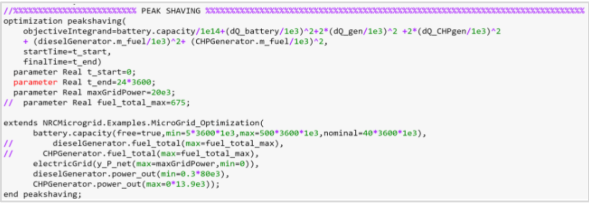


**Figure 20.** Comparative cost of Use Case 2 sim and Use Case 3 economic optimization

Figure 20 shows that the results of the optimization reduces the total cost to $2500 compared to the rule based simulation (~34% compared to diesel only and ~17% compared to the rule based simulation) with only 10 kW power supplied from the grid.

For the design optimization performed for use case 3, the objective function shown in Figure 21 was used. The results, shown in Figure 22, Figure 23, and Figure 24, show that with a grid connection present (with 20 kW max grid capacity), the original battery was oversized by 24 kWh for the same total energy cost.



```
//%%%%%%%%%%%%%%%%%%%%%%%%% PEAK SHAVING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
optimization peakshaving(
    objectiveIntegrand=battery.capacity/1e14+(dQ_battery/1e3)^2+2*(dQ_gen/1e3)^2 +2*(dQ_CHPgen/1e3)^2
    + (dieselGenerator.m_fuel/1e3)^2+ (CHPGenerator.m_fuel/1e3)^2,
    startTime=t_start,
    finalTime=t_end)
  parameter Real t_start=0;
  parameter Real t_end=24*3600;
  parameter Real maxGridPower=20e3;
//  parameter Real fuel_total_max=675;

extends NRCMicrogrid.Examples.MicroGrid_Optimization(
      battery.capacity(free=true,min=5*3600*1e3,max=500*3600*1e3,nominal=40*3600*1e3),
//          dieselGenerator.fuel_total(max=fuel_total_max),
//        CHPGenerator.fuel_total(max=fuel_total_max),
      electricGrid(y_P_net(max=maxGridPower,min=0)),
      dieselGenerator.power_out(min=0.3*80e3),
      CHPGenerator.power_out(max=0*13.9e3));
end peakshaving;
```

**Figure 21.** Use Case 3 Optimization – Economic Objective Function (placeholder for equation)

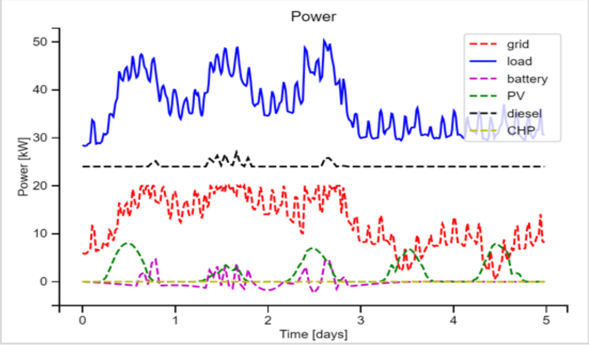


**Figure 22.** Use Case 3 Design Optimization with renewables, energy storage, and 20kW from grid

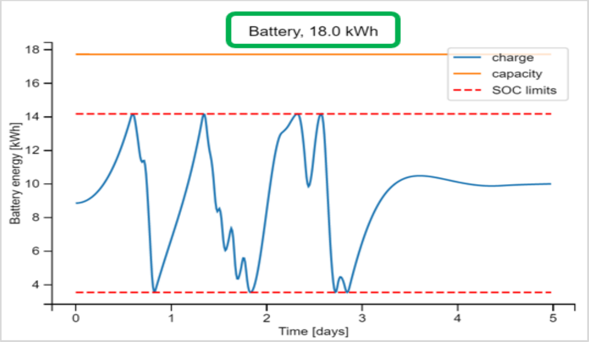


**Figure 23.** Use Case 3 Design Optimization

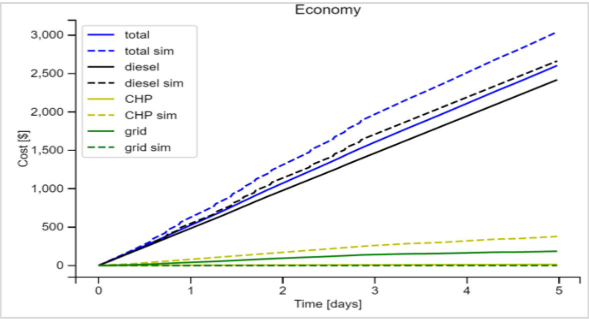


**Figure 24.** Comparative cost (placeholder) of Use Case 2 sim and Use Case 3 design optimization

# 5 Conclusions

The move towards increasing renewables for micro-grids operating in remote communities has provided the opportunity to investigate the potential to reduce fuel costs and emissions compared with traditional standalone diesel power plants. Additional work needs to be done to support the advancement and integration of new micro-grid technologies and to optimize overall integrated system design, performance, and overall costs. By using the Microgrid Modelica library models, many high level optimizations can be performed to answer initial questions on equipment sizing and economic dispatch. This, coupled with access to a physical micro-grid test environment, provides the added benefit of being able to eventually validate model results and develop higher fidelity models that can be integrated into a similar model framework.

## 5.1 Future Work

Plans for future work include:

- Generating data from the NRC Micro-grid Testing and Training facility using a configuration that mirrors this study in order to validate model results.
- Performing a design optimization of a combined PV and CPV system.
- Defining a more detailed co-generation model for the CHP system (i.e., with thermal energy generation considered).
- Integrating higher fidelity models from Modelon's Electric Power Modelica library into a comparable model framework to analyze the micro-grid's response to transient conditions on both the supply and demand side with respect to performance, stability, and reliability (e.g., analyzing stability during switching from grid connected to island mode, system performance and limitations during rapid demand changes, etc.).
- Using the above detailed integrated model framework to support micro-grid control system design and optimization.

# Acknowledgements

The authors would like to thank Johan Windahl and Hakan Runvik for their part in the development of the Modelon Micro-Grid modeling / simulation platform.

# References

Susanne Ackeby, Jonas Tjader, Caroline Bastholm. "The role of interaction of microgrids and centralized grids in development modern power systems." *8th Southern Africa Regional Conference, IEA-ISGAN,* Nov 14-17, 2017

Caterpillar (2010). "C4.4 Generator Set Electric Power." LEHE0867-00 CAN. www.Cat-ElectricPower.com

CBM (2019). Canadian Biomass Magazine – Canadian Biomass Pellet Map 2019. https://www.canadianbiomassmagazine.ca/wp-content/uploads/2019/05/CBM-PELLET-MAP-2019_HR.pdf

Indrajit Das, Claudio Canizares. "Feasibility Studies of Variable Speed Generators for Canadian Arctic Communities". Waterloo Institute for Sustainable Energy (WISE) for Innovus Power, 2017.

EV Shop (2019). https://evshop.eu/en/batteries/117-5300wh-bmw-i3-battery-module-45v-120ah-nominal.html

Chelsea Harvey, Niina Heikkinen. "Congress Says Biomass is Carbon Neutral but Scientists Disagree". *Scientific American E&E News Environment,* March 23, 2018.

Konstantinos Karanasios, Paul Parker. "Recent Developments in Renewable Energy in Remote Aboriginal Communities, Nunavut, Canada". *Papers in Canadian Economic Development – University of Waterloo.* Vol. 16, pp 54-64, 2016, DOI: http://dx.doi.org/10.15353/pced.v16i0

Susan Kraemer, *Solar Paces*. Oct 11, 2017. http://www.solarpaces.org/csp-competes-with-natural-gas-not-pv/

Natural Resources Canada (NRCan 1). "Towards Renewable Energy Integration in Remote Communities: A Summary of Electric Reliability Considerations." *Energy and Mines Minsters' Conference,* pp 1-23, Aug 2018. ISBN 978-0-660-27560-4

Natural Resources Canada (NRCan 2). "Photovoltaic and solar resource maps". Mar 20 2017 https://www.nrcan.gc.ca/18366

National Research Council Canada (NRC), (2019), "Microgrid testing and training facility."

https://nrc.canada.ca/en/research-development/nrc-facilities/microgrid-testing-training-facility

NEAS (2018). Sealift Rates for 2018 Season Nunavut (from Valleyfield, QC), March 30, 2018. https://neas.ca/wp-content/uploads/freight_nunavut.pdf

NREL (2019). Ver 6.1.3. https://pvwatts.nrel.gov/pvwatts.php

Nunavut Power. “Plant Status Report – Whale Cove 615, April 2001 https://web.archive.org/web/20040907175010/http://www.nunavutpower.com/communities/whalecove.html

Modelon (2019). Electric Power Library https://3pn0itd4zke1w5rdih3i1jfi-wpengine.netdna-ssl.com/wp-content/uploads/2018/10/Electric-Power-Library_EPL_Flyer.pdf

QEC (2018). http://www.qec.nu.ca/sites/default/files/2018-2019_qec_general_rate_application.pdf

Solar-Facts (2017). “Concentrating Photovoltaic Solar Panel Technology.” https://www.solar-facts.com/panels/concentrating-photovoltaic.php

Volter (2019). https://volter.fi/technology/

Johan Windahl, Hakan Runvik, Stéphane Velut. “Platform for Microgrid Design and Operation.” *Proceedings of the 13th International Modelica Conference*, pp 405-412, Mar 2019. DOI: 10.3384/ecp19157405

# Performance Benchmark of Modelica Time-Domain Power System Automated Simulations using Python

Sergio A. Dorado-Rojas   Manuel Navarro Catalán   Marcelo de Castro Fernandes   Luigi Vanfretti

Department of Electrical, Systems and Computer Engineering
Rensselaer Polytechnic Institute
Troy, NY, USA
{dorads, navarm2, decasm3, vanfrl}@rpi.edu

## Abstract

In this paper, a benchmark between solvers and Modelica tools for time-domain simulations of a power system model is presented. A Python-based approach is employed to automate Modelica simulations and compute performance metrics. This routine is employed to compare the performance of a commercial (Dymola) against an open-source (OpenModelica) simulation tool with different solver settings. Python scripts are developed to execute a dynamic simulation of a common model for power system studies with 49 states and 420 variables in three different scenarios. This degree of automation makes it easier to change solver settings and tools during execution. The performance of each of the tools is assessed through metrics such as execution time and CPU utilization. The quantitative comparison results provide a clear reference to the performance of the tools and solvers for the execution of time-domain simulations with a significant degree of complexity. The commercial tool offers better performance for variable-step solver, but the performance of the open-source software shows significantly faster results for fixed-step solvers.

*Keywords: Modelica, Python-Dymola Interface, Python-OpenModelica Interface, CPU performance.*

# 1 Introduction

## List of Acronyms and Definitions

### Definitions

**ST · Simulation Time**: the simulation time is understood as the time it takes for the program to translate, compile and integrate a model.

**ET · Execution Time**: the execution time is the elapsed time to complete the numerical integration of the compiled model. It also known as *integration time*. Note that execution time is included as a part of Simulation Time in the context of this paper.

**NMT · Normalized Minimum Execution Time**: performance metric taken as a function of the execution time of each of the tools. It is defined as

$$\mathrm{NMT}^{[\mathrm{solver}]} = \frac{\min(\mathrm{ET_D}, \mathrm{ET_{OM}})}{\mathrm{ET_{observed}}}$$

where $\mathrm{ET_D}$ are the execution times for Dymola and OM, respectively, and $\mathrm{ET_{observed}}$ is the corresponding integration time of each tool for a given solver obtained from the simulation log.

### Acronyms

ET · Execution Time
MSE · Mean Square Error
NAE · Normalized Absolute Error
OM · OpenModelica
OMPython · OpenModelica Python Interface
OpenIPSL · Open-Instance Power System Library
PDI · Python-Dymola Interface
ST · Simulation Time

## Motivation

Modeling and simulation of power systems have been a habitual practice in the energy industry since the 1960s. The complexity of a power system is steadily increasing to accommodate modern technologies into the existing grid. A more complex system leads to more elaborated models. High-complexity models are directly correlated with computationally expensive tasks (Milano, 2010). In this context, the Modelica language represents an accurate, equation-based, multi-domain solution modeling and simulation alternative. Numerous initiatives such as OpenIPSL have been taken to incorporate into the power system workflow the benefits of the Modelica language (Baudette et al., 2018).

On the other hand, the academic, scientific and industrial communities have come to acknowledge the intrinsic benefits of the Modelica language. An outcome of this trend is that the user base has increased significantly during the last years. This has led to the development of many libraries with users coming from a very wide domain spectrum. Nowadays, Modelica stakeholders include students, consulting firms, big laboratories, and industry agents.

Free tools such as OpenModelica are fundamental for learning the language at little to no cost and to set a reference for the Modelica language (Fritzson et al., 2006).

Commercial tools such as Dymola, SystemModeler or SimulationX provide advanced functionalities that satisfy particular requirements from the industry. However, there is no clear guidance for a user on how to select the tool-solver based on its simulation performance exclusively.

This paper intends to provide this guidance. It aims to compare the time-domain simulation performance of the solvers from both Dymola and OpenModelica when subjected to different solver settings (see (Braun et al., 2017) for a detailed analysis of the potential of OpenModelica to solve large-scale models). Since these tools do not have the same features and solvers, we have chosen some of the ones they have in common for benchmarking purposes.

## Contribution

This work is relevant to any user of the Modelica language. The tool performance analysis is based on the simulation of a power system model (IEEE 14 bus system), that serves as a representation of a dynamic nonlinear system. We consider three simulation scenarios: an initialization, a line-opening (one discrete event) and two bus faults (two discrete events). The paper reveals the difference in performance within the tools and helps users make an educated choice about the tools to use. The main contributions of the paper are the following:

- Quantitative evaluation of Dymola and OpenModelica simulation performance for time-domain simulation of complex dynamic systems (power systems).
- Benchmark of different solvers in a dynamic simulation with discrete events.
- Implementation of simple Python routines to automate Dymola and OpenModelica time-domain simulations.

## Paper Organization

The paper is broken down in the following sections: Section 2 describes the test system and the Modelica library employed to construct it. The experiment setup regarding hardware characteristics and software setup is described in Section 3. In Sections 4 and 5, we discuss performance results of each of the tools with respect to each solver and the corresponding performance metrics. Finally, Section 6 concludes the work.

## 2 Modelica Power System Model

The IEEE14 bus system[1] represents a part of the Midwestern USA American Electric Power System as of February of 1962. The single-line diagram of the system can be seen in Figure 1. This model was chosen because it is a widely used testing system for an initial assessment in power system dynamical studies since it has a significant number of variables and states (420 and 49, respectively) which makes it a common factor in such simulation-based studies (Milano, 2010). For this reason, its dynamic simulation a challenge to the tools and the CPU.

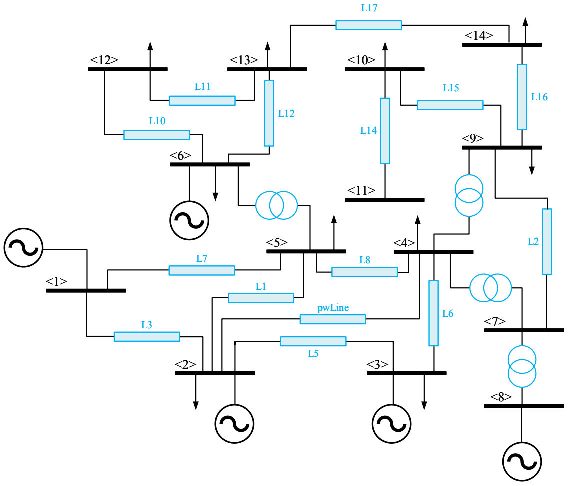


**Figure 1.** IEEE 14 bus model.

The test power system model (Figure 2) is built using the components from the open-source OpenIPSL library, a Modelica-based power system component library currently developed and maintained by ALSETLab at Rensselaer Polytechnic Institute. The library includes all the components to build a large power system model and perform dynamic analysis in time- and phasor-domain. The version of the library used in this paper is release 1.5.0[2].

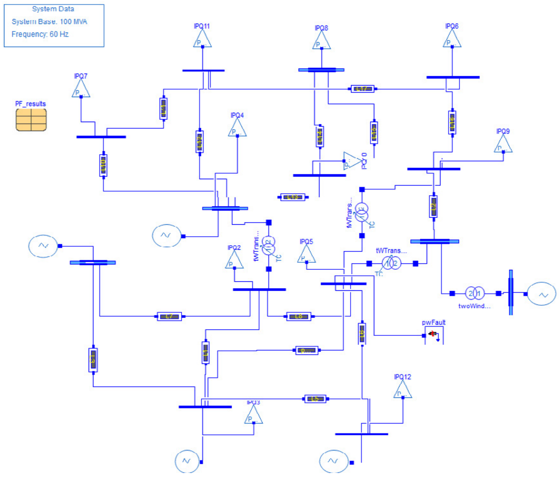


**Figure 2.** Implementation of the IEEE 14 bus model in Dymola using OpenIPSL.

## 3 Experiment Specifications

To make sure that the results are reproducible, this section details the conditions under which the experiments were performed regarding hardware setup and software characteristics.

[1] https://icseg.iti.illinois.edu/ieee-14-bus-system/

[2] The version of the library employed for this paper is included in the GitHub repository of the project. For the latest release of OpenIPSL, see: https://github.com/OpenIPSL/OpenIPSL

## 3.1 Hardware and Software Setup

The characteristics of the computer used to run the simulations are shown in Table 1.

| | Characteristic |
|---|---|
| **Operating System** | Ubuntu Server 18.04 LTS |
| **RAM** | 128 GB |
| **Processor** | Intel(R) Xeon(R) CPU E-1650 v4<br>12 Cores @ 3.60 GHz<br>15 MB Cache |
| **Storage** | 1 TB SSD |
| **Graphics Cards** | 4 x NVIDIA GTX 1080 Ti<br>(CUDA Capable)<br>11 GB GDDR5X (each) |
| **Dymola Distribution** | Dymola 2020x |
| **OpenModelica Distribution** | 1.14.0 |
| **Python Release** | 3.6.8 |
| **Dymola Compiler** | MinGW CC |
| **OpenModelica Compiler** | MinGW CC |

**Table 1.** Hardware characteristics and software specifications of the computer used to run the experiment.

To assess solver performance correctly, numerical integration must run in only one processor. While this is a default option in Dymola, we need to specify this option explicitly in OpenModelica before starting any simulation since it defaults to multicore execution. This is done thanks to the flag `setCommandLineOptions("-n=1")`.

## 3.2 Simulation Scenarios

To properly measure solver performance in diverse dynamic conditions, we will consider the following three scenarios of the IEEE14 bus model: system initialization, time-domain simulation with one line opening, and system response with two faults.

### IEEE14 System Initialization ($S_1$)

This scenario corresponds to a system with no disturbing events. The power flow of the model is modified so that the numerical simulation has problems during initialization. The provided initial conditions are such that the dynamic system is not initially at an equilibrium point, thus forcing the system to look for an acceptable steady-state condition at the beginning of the integration process. This increases the computational task and challenges the solver since the integration does not start with all state derivatives equal to zero.

### Line Opening ($S_2$)

Besides the aforementioned bad initialization condition, we introduce a line opening to disturb the system from steady-state and excite nonlinear dynamics. This kind of scenario is used to study system-wide stability when two sub-areas are disconnected from each other. The line opening corresponds to the connection between buses 2 and 4 (B2 and B4). The line will open from both ends at time $t = 60$ s and will re close at $t = 61.5$ s.

### Bus Faults ($S_3$)

In this case, the system will face two three-phase to ground faults at different times. This configuration is used to test the resiliency and stability of the system. By having two faults, the numerical complexity of the simulation increases, creating a more adverse scenario for the solvers to come up with a solution. Fault 1 occurs at bus 4 (B4) starting at $t = 20$ s and being removed at $t = 21.2$ s. Fault 2 takes place at bus 14 (B14) at $t = 80$ s, being cleared at $t = 81.2$ s. The parameters of the two faults are $R = 0$ pu and $X = 1 \times 10^{-5}$ pu.

## 3.3 Solver Selection

The performance of the time-domain simulation depends not only on the dynamic condition to be analyzed but also on the solver selection. In this regard, OpenModelica and Dymola contain a wide variety of different integration methods and three of them are going to be used and thus briefly described in this study. The Differential Algebraic System Solver (*dassl*) is an implicit, high-order, variable-step solver with time-step control. This solver is set as default solver in both OpenModelica and Dymola. The *Euler* method is another solver available in both software packages and it is an explicit (Forward Euler), first-order, fixed time-step solver. Finally, the last solver used in this study is the *runge kutta*. Dymola allows the user to chose between second, third and fourth order Runge-Kutta methods but in this work, only the fourth order is used since it is also available in OpenModelica. This solver is an explicit, fourth-order, fixed time-step solver. This paper will benchmark the performance of the tools with each of the mentioned solvers for the different scenarios of the test power system.

## 3.4 Time-step Selection

Since *dassl* is a variable-step solver with step-size control, there is no need to select a specific time-step for the solver. The selection of an adequate number of intervals is necessary to plot and analyze the results. For both tools, 5000 was found to be a reasonable number of simulation intervals. Moreover, to use the capabilities of a DAE solver to their full extent, we enable the newly incorporated DAEmode in Dymola by enabling the flag `Advanced.Define.DAEsolver = true` (Henningsson et al., 2019). In OpenModelica, to set similar settings we use the command `setCommandLineOptions("daeMode=true")`.

On the other hand, it is important to select an adequate step size $T_s$ for fixed-step solvers in order to guarantee that the algorithm is operating in its region of convergence. To get an upper bound for $T_s$, we performed a linear analysis of the system in Dymola employing the library `Modelica_LinearSystems2`. After determining the time constant of the fastest mode ($\tau \approx 1$ ms), we found that $T_s = 0.5$ ms was a reasonable value to capture the effects of the fastest mode, guaranteeing numerical convergence for both solvers, *Euler* and *Runge Kutta*. The selected time-step size implies that 240,000 simulation intervals are going to be needed for a simulation time of 120 s.

### 3.5 Benchmark Metrics

In order to understand and accurately compare the two tools the paper focuses on two simulation features to compare:

- *Simulation Time* (ST) corresponds to the time it takes for a program to complete all of the routines for each scenario comprising model translation, compilation and execution. The discussion of the results of the simulation time are found in Section 4.1, with special remark on Execution Time (ET).
- *CPU Utilization* is the percentage of central processing unit (CPU) that is being used at any time during the execution. Results for CPU utilization can be found in Section 4.2.

### 3.6 Code Structure

The complete code to perform the experiments and analyze the resulting data can be found in GitHub[3]. The execution of the simulations is automated through Python using the Dymola API (Python-Dymola Interface) and the OpenModelica Python Interface (OMPython) (Lie et al., 2018). The details of the Dymola routine can be seen in the file `dymola_simulation.py`. Likewise, the OpenModelica commands are included in the file `om_simulation.py`.

To measure performance we execute the routine in the script `measurement_performance.py`. It measures each of the performance metrics every 0.2 s while the code is running in a different parallel process. The main program is contained in the file `01_modelica_tool_performance_benchmark.py`.

## 4 Performance Results

Before presenting the performance results, we validate the simulation outputs of the three scenarios for Dymola and OpenModelica for all solvers. We employed the Normalized Absolute Error (NAE) and the Mean Square Error (MSE) defined in Equation (1) to quantify the numerical difference between the outcomes of each tool.

$$
\begin{aligned}
\text{NAE} &= \frac{|x_i - y_i|}{n} \\
\text{MSE} &= \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{n}
\end{aligned}
\tag{1}
$$

NAE shows how different the Dymola and OpenModelica results are throughout the simulation. MSE outputs a quantitative validation of the results of both tools (Devore and Berk, 2012). Full details can be seen in Table 3.

The numerical behavior of the simulation during initialization (*runge kutta* solver) can be observed in Figure 3 for the voltage magnitude signal at Buses 2 and 4. An initial transient behavior can be seen at the beginning of the integration time. This is not desired in a dynamic simulation since numerical convergence to a steady-state solution is not guaranteed given the fact that the solver starts from a guessing point with non-zero derivatives.

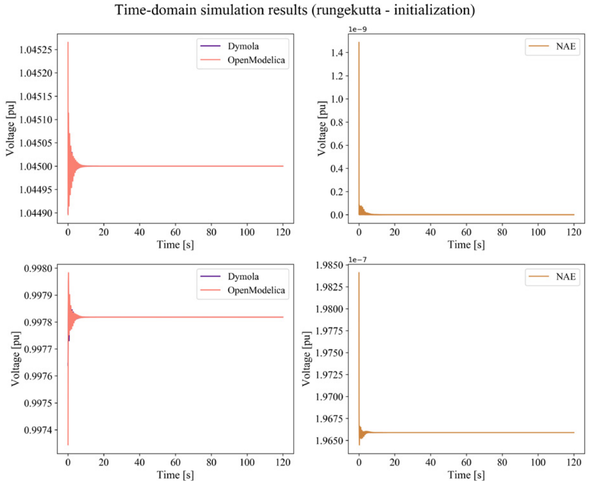


**Figure 3.** Comparison between Dymola and OpenModelica results for the initialization scenario using the *runge kutta* solver

The non-steady state behavior at the on-set of the simulation is due to the fact the initial guess used in the model (the so-called power flow) is not close enough to an equilibrium for the initialization routine to solve for a more precise set of initial values. A more complex initialization problem will better benchmark the capabilities of the tools. Despite this, Dymola and OM produce almost the same results, with an NAE in the order of $10^{-3}$.

Likewise, for the *runge kutta* solver, Figures 4 and 5 show the simulation results for the line opening (voltage magnitude at buses 2 and 4) and the double bus fault (voltage at affected buses 4 and 14) scenarios, respectively. Both Figures reveal how there is a minimal error between the results of both tools. Based on these results, it is concluded that fixed-step solvers can be applied to reduce discrepancies between different Modelica tools.

[3] https://github.com/ALSETLab/Time-Domain-Simulation-Performance-Benchmark

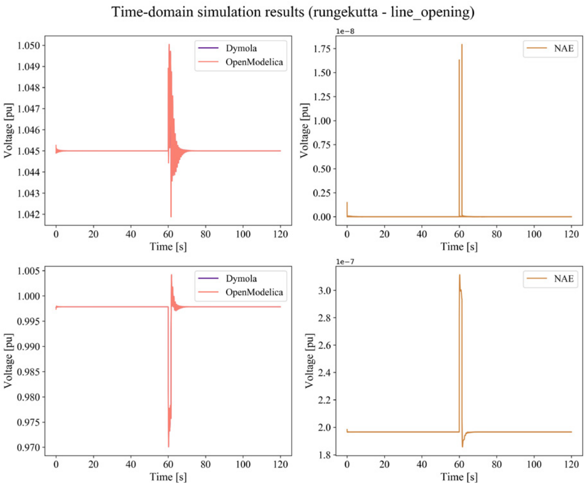


**Figure 4.** Comparison between Dymola and OpenModelica results for the line opening scenario using the *runge kutta* solver

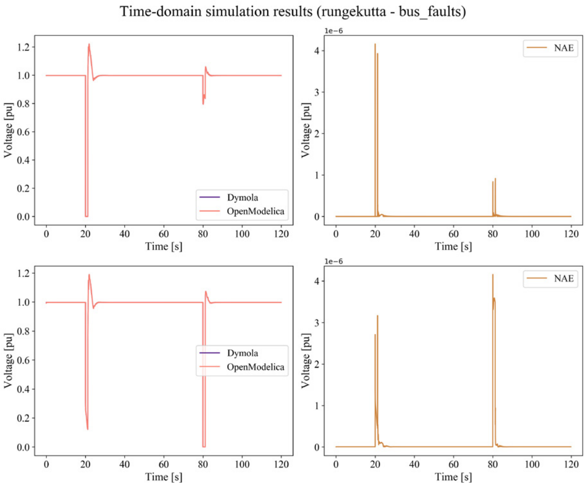


**Figure 5.** Comparison between Dymola and OpenModelica results for the double bus fault scenario using the *runge kutta* solver

The complete collection of plots for all solvers and simulation scenarios can be found online in the GitHub repository in the Notebook `02_Data_PostProcessing_SimulationResultPlotting.ipynb`.

## 4.1 Simulation Time

The information regarding simulation time is presented for all scenarios and solvers in Table 2. We must underline that simulation time includes compilation, translation and actual integration (execution time).

A clear conclusion from this information is that the variable-step solver is the most convenient for an initial analysis of the conditions of the system with an important amount of detail. Nevertheless, considering the information about MSE, a fixed-step solver shows advantages to reduce the numerical discrepancy between tools running the same model. The cost is a considerable increase in simulation time.

## 4.2 CPU Utilization

Since each instance of Dymola/OpenModelica was constrained to run only on one core, we expect exactly one processor to be responsible for numerical integration while a simulation is being carried out. The CPU usage of the assigned execution core is 100% due to the heavy numerical task of the simulation.

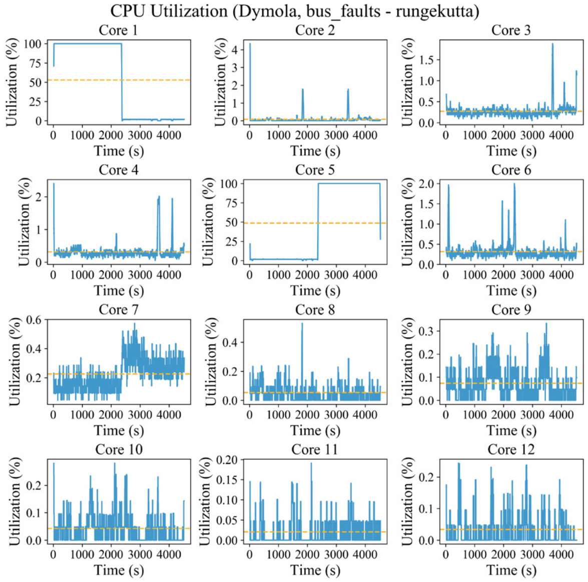


**Figure 6.** CPU Utilization for Dymola during bus fault scenario with *runge kutta* solver.

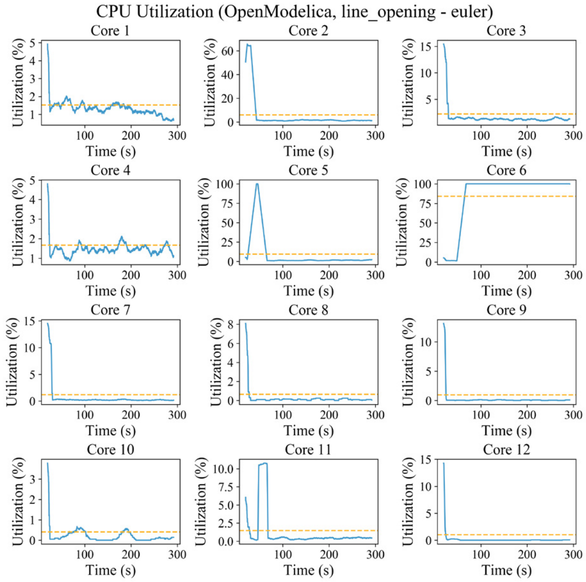


**Figure 7.** CPU Utilization for OpenModelica during line opening scenario with *euler* solver.

An interesting outcome of our experiments is that several CPUs are involved in the execution process but just one is performing the simulation tasks at a given time. We can detail this behavior in Figure 6 for a Dymola simulation using the *runge kutta* method of the bus fault scenario. Simulation starts in Core 1 where the CPU usage is at a 100% at the beginning of the running time. Afterward, it is delegated to Core 5. Finally, Core 10 completes the execution of the program. This behavior is due to a task scheduling routine in the processor level that dispatches to different cores the compilation, translation, and integration sub tasks.

Similar behavior happens with another solver and OpenModelica (Figure 7) in which the simulation started in Core 2, then was briefly assigned to Core 5 and was finished in Core 6. All the graphics can be detailed in the GitHub repository inside the Jupyter Notebook called `03_DataPostprocessing_CPU_Usage.ipynb`.

## 5 Performance Evaluation Metrics

A score was proposed to quantify the performance differences between the tools and the solvers. The score is obtained from the data generated for all simulations and solvers. This single metric makes it simpler to directly compare the performance of Dymola versus OpenModelica. From Table 2 the Execution Time (ET) for each scenario and solver were employed. These metrics were obtained directly from the program logs and measured in Python. Notice that the time registered using OMPython is slightly larger than the reported by the simulation log due to the communication interface between Python and OM. The translation and compilation time were not taken into account since this information is only available in the Dymola developer version, not in the release version.

The Normalized Minimum Execution Time score (NMT) of each scenario per solver is computed as

$$\mathrm{NMT}^{[\mathrm{solver}]} = \frac{\min(\mathrm{ET}_{\mathrm{D}}, \mathrm{ET}_{\mathrm{OM}})}{\mathrm{ET}_{\mathrm{observed}}} \tag{2}$$

where $\mathrm{ET}_{\mathrm{observed}}$ is the ET for a particular solver in Dymola or OpenModelica), and $\min(\mathrm{ET}_{\mathrm{D}}, \mathrm{ET}_{\mathrm{OM}})$ is the minimum execution time between both tools for a specific solver. Clearly, $\mathrm{NMT}^{[\mathrm{solver}]}$ lies between 0 and 1. The higher the NMT is, the faster the simulation will run for a particular selected solver. At a first glance, this metric might be counter-intuitive since a better solver/tool combination would reduce execution time. However, we propose an increasing score metric due to the fact that users are more familiar to higher scores for better performance. Therefore, the larger the NMT is, the faster a particular solver will run.

The NMT metric results are presented in Table 4. The performance of Dymola is remarkably better using *dassl*. Nevertheless, OM shows a smaller execution time than Dymola for fixed-step solvers (as can be seen from Table 2, the NMT scores and 5). This conclusion can be further detailed in Table 5 where a direct comparison between the execution time for the tools with the different solvers for each scenario is presented.

The NMT scores highlight that the performance of Dymola in terms of execution time is remarkably better for variable-step solvers. The relative advantage of selecting one tool with respect to the other can be computed from the NMT directly. For instance, Dymola runs 47.3x faster than OM for the first scenario using *dassl* which can be computed by a direct comparison of the ET listed in Table 5. The NMT score of OM for $\mathrm{S}_1$ is 0.0211 which is $1/0.0211 = 47.3$ times smaller than the corresponding Dymola metric reflecting the relative difference in execution time.

For the variable-step solver, the discrepancy between the tools can be attributed to the performance of the *dassl* solver in all simulations thanks to the aforementioned improvements for `DAEMode` inside Dymola (Henningsson et al., 2019).

The execution time of OM is faster than the one of Dymola for all scenarios when a fixed-step solver is used. The NMT scores show a relative advantage between 3.4 and 6.8 times favoring OM. We have contacted Dassault Systèmes about the performance of the simulations of the IEEE 14 Bus System using *runge kutta* methods (including *euler*) as integrator and GCC for compilation. Dassault reports that bug fixes have been made in the GCC runtime libraries, leading to CPU times are about 3 – 4 times faster, on par with the run times given when compiling with Visual Studio under Windows 10. Dassault informs that the updated libraries will be part of Dymola 2021.

We should point out that the scope on the potential optimization features has been limited to the use of the flag `Evaluate = true` in Dymola and `-d=evaluateAllParameters` in OM, which is standard practice when attempting to improve simulation performance.

The detailed step-by-step computations of the scores can be found in GitHub in the Notebook `05_BenchmarkMetrics.ipynb`.

## 6 Conclusions and Future Work

The paper presents a concrete analysis of the time-domain simulation performance of Modelica-based tools for different solvers in the context of large-scale nonlinear dynamic systems. The presented results can help a user to choose a tool depending on the final application, and lead to improvements in Modelica tools. The methodology of this benchmark can be extended to virtually any platform or Modelica tool.

We benchmarked the time-domain simulation performance of two popular Modelica tools, Dymola and OpenModelica, for a dynamic power system simulation using the IEEE 14 bus system. We considered several scenarios that challenge numerical solvers differently. Thanks to Python scripting, we were able to change automatically the simulation settings while directly measuring the performance of the computer instead of relying on simulation logs. Python functions also made it quicker to analyze straightforwardly the big set of data regarding simulation results and computer performance.

For the proposed heuristic score, we found out that OpenModelica performs better than Dymola in terms of execution time for fixed-step solvers while Dymola shows faster results when using a variable-step solver (see Table 5). Despite this, we must warn the reader that this conclusion is based upon only a particular system. Further research has to be done to include more test systems. Moreover, the use of a fixed-step solver has the main advantage of

The tool and solver benchmark results are expected to be reproduced in a larger system such as the Nordic 44 with $\approx$ 1300 states and 6300 variables. This system requires a considerable amount of RAM given its large number of states. Therefore, future work is related to the performance analysis in a 64-core machine with 512 GB of RAM for the N44 system considering different types of simulations and various initialization parameters.

## Acknowledgments

This work was funded in part by New York Power Authority (NYPA) and the New York State Energy Research and Development Agency (NYSERDA) through the Electric Power Transmission and Distribution (EPTD) High Performing Grid Program under agreement 137940.

The authors would like to thank Erik Henningsson from Dassault Systèmes for his valuable contributions for the improvement of the results of this paper.

## References

Maxime Baudette, Marcelo Castro, Tin Rabuzin, Jan Lavenius, Tetiana Bogodorova, and Luigi Vanfretti. OpenIPSL: Open-Instance Power System Library - Update 1.5 to iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations. *SoftwareX*, 7:34–36, jan 2018. ISSN 23527110. doi:10.1016/j.softx.2018.01.002.

Willi Braun, Francesco Casella, and Bernhard Bachmann. Solving large-scale Modelica models: new approaches and experimental results using OpenModelica. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic*, pages 557–563, jul 2017. doi:10.3384/ecp17132557. URL `http://www.ep.liu.se/ecp/article.asp?issue=132{%}26article=63`.

J. L Devore and K. N. Berk. *Modern Mathematical Statistics with Applications*. Springer-Verlag New York, 2012.

Peter Fritzson, Peter Aronsson, Adrian Pop, Hakan Lundvall, Kaj Nystrom, Levon Saldamli, David Broman, and Anders Sandholm. Openmodelica-a free open-source environment for system modeling, simulation, and teaching. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 1588–1595. IEEE, 2006.

Erik Henningsson, Hans Olsson, and Luigi Vanfretti. DAE Solvers for Large-Scale Hybrid Models. In *Proceedings of the 13th International Modelica Conference, Regensburg, Germany*, pages 491–502, feb 2019. doi:10.3384/ecp19157491. URL `http://www.ep.liu.se/ecp/article.asp?issue=157{%}26article=50`.

Bernt Lie, Sudeep Bajrachary, Alachew Mengist, Lena Buffoni, Arun Kumar, Martin Sjölund, Adeel Asghar, Adrian Pop, and Peter Fritzson. Api for accessing openmodelica models from python. In *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016*, pages 707–714. Linköping University Electronic Press, 2018.

Federico Milano. *Power System Modelling and Scripting*, volume 54 of *Power Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-13668-9. doi:10.1007/978-3-642-13669-6. URL `http://link.springer.com/10.1007/978-3-642-13669-6`.

# Appendix

In this appendix, all performance results of the different simulation experiments are presented. In Table 5, *runge-kutta* is abbreviated as rk.

| | | Simulation Time OpenModelica (OM) | | | | |
|---|---|---|---|---|---|---|
| | | Translation | Compilation | Execution | Total Time (OM log) | OMPython |
| $S_1$ | dassl | 2.3204 s | 6.6270 s | 7.8690 s | 16.8164 s | 19.3451 s |
| | euler | 2.5432 s | 6.5845 s | 277.5495 s | 286.6772 s | 289.2878 s |
| | rk | 2.3495 s | 6.6213 s | 783.0159 s | 791.9867 s | 794.6805 s |
| $S_2$ | dassl | 2.6079 s | 6.6411 s | 13.4004 s | 22.6494 s | 25.2542 s |
| | euler | 2.4023 s | 6.6437 s | 310.1061 s | 319.1521 s | 321.7222 s |
| | rk | 2.3489 s | 6.6591 s | 1086.3958 s | 1095.4040 s | 1098.0253 s |
| $S_3$ | dassl | 2.1952 s | 6.7301 s | 163.4884 s | 172.4137 s | 175.2962 s |
| | euler | 2.3248 s | 6.7801 s | 378.6069 s | 387.7118 s | 390.3140 s |
| | rk | 2.3960 s | 6.7332 s | 1344.6808 s | 1353.8100 s | 1356.2994 s |

| | | Simulation Time Dymola | | |
|---|---|---|---|---|
| | | Translation + Compilation | Execution | Measured Python |
| $S_1$ | dassl | 20.186 s | 0.1664 s | 20.3524 s |
| | euler | 24.7791 s | 1880.0109 s | 1904.7900 s |
| | rk | 21.2389 s | 4420.0125 s | 4441.2514 s |
| $S_2$ | dassl | 20.2363 s | 0.34082 s | 20.5772 s |
| | euler | 19.6561 s | 1850.0129 s | 1869.6690 s |
| | rk | 24.6567 s | 4410.0119 s | 4434.6686 s |

| | | Simulation Time Dymola | | |
|---|---|---|---|---|
| | | Translation + Compilation | Execution | Measured Python |
| $S_3$ | dassl | 20.2161 s | 14.4098 s | 34.6260 s |
| | euler | 16.4581 s | 1820.0119 s | 1836.47 s |
| | rk | 17.9956 s | 4590.0129 s | 4608.0085 s |

**Table 2.** Execution time for Dymola and OpenModelica for each simulation scenario using different solvers.

| | | Mean Squared Error (MSE) | |
|---|---|---|---|
| | | B2 | B4 |
| $S_1$ | dassl | $3.0011 \times 10^{-11}$ | $4.6482 \times 10^{-11}$ |
| | euler | $1.2894 \times 10^{-11}$ | $3.7950 \times 10^{-11}$ |
| | rk | $1.2853 \times 10^{-11}$ | $3.7828 \times 10^{-11}$ |
| $S_2$ | dassl | $1.1728 \times 10^{-8}$ | $1.1267 \times 10^{-7}$ |
| | euler | $2.3598 \times 10^{-10}$ | $3.2470 \times 10^{-9}$ |
| | rk | $2.3579 \times 10^{-10}$ | $3.2473 \times 10^{-9}$ |

| | | Mean Squared Error (MSE) | |
|---|---|---|---|
| | | B1 | B4 |
| $S_3$ | dassl | 0.0067 | 0.0002 |
| | euler | 0.0025 | 0.0002 |
| | rk | 0.0018 | 0.0002 |

**Table 3.** Mean Square Errors between voltage magnitude signals at different buses for each simulation scenario.

| | Dymola | | | OpenModelica | | |
|---|---|---|---|---|---|---|
| | $NMT^{[S_1]}$ | $NMT^{[S_2]}$ | $NMT^{[S_3]}$ | $NMT^{[S_1]}$ | $NMT^{[S_2]}$ | $NMT^{[S_3]}$ |
| dassl | 1 | 1 | 1 | 0.0211 | 0.0254 | 0.0880 |
| Euler | 0.148 | 0.168 | 0.208 | 1 | 1 | 1 |
| rk | 0.177 | 0.296 | 0.293 | 1 | 1 | 1 |

**Table 4.** Normalized Minimum Execution Time scores.

| | | Execution Time (ET) | | |
|---|---|---|---|---|
| | | OM | Dymola | Result |
| $S_1$ | dassl | 7.869 s | 0.1664 s | D > OM (47.3x) |
| | euler | 277.54 s | 4420.01 s | OM > D (6.8x) |
| | rk | 783.01 s | 1880.01 s | OM > D (5.6x) |
| $S_2$ | dassl | 13.40 s | 0.3408 s | D > OM (39.3x) |
| | euler | 310.10 s | 1850.01 s | OM > D (6.0x) |
| | rk | 1086.39 s | 4410.01 s | OM > D (4.1x) |

| | | Execution Time (ET) | | |
|---|---|---|---|---|
| | | OM | Dymola | Result |
| $S_3$ | dassl | 163.48 s | 14.40 s | D > OM (11.3x) |
| | euler | 378.60 s | 1820.01 s | OM > D (4.8x) |
| | rk | 1344.68 s | 4590.01 s | OM > D (3.4x) |

**Table 5.** Comparison between execution time in OM and Dymola for different solvers.

# A Modelica Library for Continuous and Discrete Extremum Seeking for Static and Dynamic Systems

Joscha Müller  Maxime Baudette  Daniel Arnold  Michael Sankur

Lawrence Berkeley National Laboratory
{joschamueller,baudette,dbarnold,msankur}@lbl.gov

## Abstract

Extremum Seeking (ES) is an optimization scheme that has become a popular tool for addressing decision-making problems in settings where system models are unavailable or inaccurate and communications are unreliable. This paper presents an open source Modelica Extremum Seeking library that introduces different continuous and discrete ES controllers and examples for possible ES control applications. The controllers are available for Modelica and in the Functional Mock-up Interface (FMI) standard, which allows the models to be used in a variety of different software environments.

## 1 Introduction

Black-box optimization methods are an important class of algorithms used in situations where objective functions are difficult or expensive to evaluate, or are, perhaps, unknown to the optimizer. Many black-box optimization techniques rely on a local exploration of the action space to determine the best action to take. Within this subclass of algorithms, Extremum Seeking (ES) approaches are particularly useful as they require no knowledge of the system over which they are optimizing.

This “model-free" property of ES has made it attractive in a variety of applications, including homogeneous charge compression ignition (HCCI) engine optimization (Killingsworth et al., 2009), maximum power point tracking (MPPT), wind turbine optimization (Krstic et al., 2014), and control of autonomous robots (Zhang et al., 2007). Additionally, several books have been written on the topic (Zhang and Ordonez, 2012), (Ariyur and Krstic, 2003). Many of the authors of this work have utilized ES to manage power injections of distributed generation devices in the smart grid (Arnold et al., 2018).

In the ES scheme the optimizer (e.g., mobile robot, power generation source, parameter to be tuned) injects a small sinusoidal perturbation into the local action space. This perturbation, in turn, introduces an oscillation in the objective function value. With proper filtering to extract the oscillatory component of the objective function, the decision-maker can extract the gradient of the control input with respect to the objective. No explicit knowledge of the structure of the objective function is needed to obtain the gradient information, only a time series of measurements of the objective.

Our past research has explored several extensions of the family of ES algorithms, such as the simultaneous control of two setpoints using a single controller (Arnold et al., 2018), and the introduction of a decaying probe (Sankur and Arnold, 2019). More recently we started studying the impacts of communication delays and missing data in the system's measurements on the performances of ES, following feedback from experimental implementation of the ES scheme in power system applications. This kind of studies requires a modeling environment that can easily implement more complex systems including both discrete and continuous dynamics. The Modelica language natively supports the modeling and simulation of complex cyber-physical systems and features many existing libraries containing models from many physical domains. It offers an attractive platform to further develop and study new variants of ES algorithms and their interaction with systems featuring more realistic characteristics (e.g., communication delays). This motivated the development of an ES library in Modelica. In this work, we present the library and examples of ES used to address several control/optimization problems. Models of different physical domains from the Modelica Standard Library were used to build a set of application examples, which are briefly highlighted in this paper.

The remainder of this paper is organized as follows. First, the working principle of ES control is introduced, followed by an overview of several examples of ES applied to domain-specific problems. Specifically, we discuss the optimization of a quadratic map, control of a spring-mass damper, speed control of a rolling wheel, and optimization of power injections in the smart grid. The paper concludes with a brief discussion of plans for future extensions to the ES Modelica library. The ES library was published as an open source project, and made available at `https://github.com/LBNL-ETA/ESL`.

Joscha Müller, Maxime Baudette, Daniel Arnold, and Michael Sankur are with the Grid Integration Group within the Energy Technologies Ares at the Lawrence Berkeley National Laboratory. This work was supported in part by the U.S. Department of Energy ARPA-E (DE-AR0000340) and Office of Energy Efficiency and Renewable Energy (DE-EE0008008 and DE-AC02-05CH11231).

# 2 ES Control Overview

In this section, we discuss the different types of ES controllers in our library. We provide an overview of a conventional ES controllers, and two types of fixed-step ES controllers (e.g. where the control action space is uniformly discretized).

The reader should note that the terms “continuous” and “discrete” can refer to continuous time or discrete time implementations of the controllers, and to refer to the possible values the controller setpoint and probe signal may take (i.e. action space). For clarity purposes, we will reserve the terms “continuous” and “discrete” to refer to continuous or discrete time. We will use “ES” or “conventional ES” for controllers using a continuous action space, and “fixed-step ES” for controllers using a discrete action space. Fixed-step ES controllers are inherently discrete controllers.

While the ES algorithm requires no knowledge of the objective function, it does require measurements of the objective function value. The reader should note that the mapping from input to objective function must be locally convex for the ES algorithm to converge to a minimizer, or locally concave to converge to a maximizer.

Without loss of generality, we consider the parallel operation of many ES controllers, and index each controller by the subscript $m$.

## 2.1 Conventional ES Controllers

Figure 1 shows a conventional ES controller within the dashed red rectangle. Consider the setpoint of the $m^{th}$ ES controller, $\hat{u}_m$. The controller adds a sinusoidal perturbation with amplitude $a_{u,m}$, frequency $\omega_{u,m}$, and phase $\theta_{u,m}$ to the setpoint $\hat{u}_m$, producing the controller output $u_m$. The controller output $u_m$, along with all other controller outputs, form the input to an unknown system. One or more entities take measurements of the system $y$, and a central entity computes objective function $\Psi(\boldsymbol{\mu})$.

The ES controller collects measurements of the objective function $\Psi(\boldsymbol{\mu})$. Objective function measurements $\Psi(\boldsymbol{\mu})$ pass through a high-pass filter, attenuating low frequency content of the objective function, such as content due to controller setpoints, and then pass through a low-pass filter, to attenuate measuerement noise and high frequency content, such as content produced by the discrete ES controller’s perturbations and setpoint updates, producing $\rho_{u,m}$.

The controller demodulates the signal $\rho_{u,m}$ through multiplication with a sinusoidal function with the same frequency $\omega_{u,m}$, amplitude $2a_{u,m}^{-1}$, and phase shift $\theta_{u,m} + \phi_{u,m}$, giving $\sigma_{u,m}$. The demodulated signal $\sigma_{u,m}$ passes through a low-pass filter, to attenuate high frequency content, producing a numerical estimate of the gradient of $\Psi(\boldsymbol{\mu})$ with respect to $\hat{u}_m$, $\hat{\xi}_{u,m}$. The gradient estimate $\hat{\xi}_{u,m}$ passes through an averaging operator (AO), producing $\overline{\hat{\xi}}_{u,m}$. One example of an AO is to average the gradient estimate over the last probe cycle, to further reduce high frequency content, such as probing frequency oscillations.

The averaged gradient estimate, $\overline{\hat{\xi}}_{u,m}$, is integrated with negative gain $-k_{u,m}$ to update the setpoint $\hat{u}_m$. The setpoint may be limited by a minimum and/or maximum value, which are typically set such that a complete probe cycle is feasible at any setpoint; this is not shown in Figure 1.

## 2.2 Single Fixed-Step ES Controller

Figure 1 shows a Single Fixed-Step ES controller (SFSES) within the dashed green rectangle. This controller operates on two timescales. The first timescale is that of perturbation and filtering. The second is the setpoint update timescale, which is slower than the perturbation and filtering timescale. We index each setpoint update period with the subscript $b$.

We start with the setpoint of the $m^{th}$ Single Fixed-Step ES Controller for the $b^{th}$ update period $\hat{v}_{m,b}$. The controller holds its setpoint $\hat{v}_{m,b}$ constant over period $b$. Throughout the $b^{th}$ update period, the $m^{th}$ SFSES adds a square wave perturbation with frequency $\omega_{v,m}$, zero-peak amplitude $a_{v,m}$, and phase $\theta_{v,m}$ to its setpoint to produce the controller output $v_{m,b}$. The controller collects measurements of the objective function $\Psi(\boldsymbol{\mu})$. Objective function measurements $\Psi(\boldsymbol{\mu})$ pass through a high-pass filter, to attenuate low frequency content of the objective function, such as those due to controller setpoints, producing $\rho_{v,m}$.

The controller demodulates the signal $\rho_{v,m}$, by multiplying with a square wave with the same frequency $\omega_{v,m}$, amplitude $a_{v,m}^{-1}$, and phase $\theta_{v,m} + \phi_{v,m}$, giving $\sigma_{v,m}$. This signal passes through a low-pass filter, to attenuate high frequency content, producing an estimate of the gradient of $\Psi(\boldsymbol{\mu})$ with respect to $\hat{v}_{m,b}$, $\hat{\xi}_{v,m}$.

At end of the $b^{th}$ update period, the gradient estimate is averaged over $b$ by the averaging operator (AO), giving $\overline{\hat{\xi}}_{v,m,b}$. The setpoint change is an integer multiple of the probe amplitude multiplied by the sign of the averaged gradient. The setpoint is then updated such that:

$$\hat{v}_{m,b+1} = \hat{v}_{m,b} - k_{v,m} a_{v,m} \operatorname{sgn}\left(\overline{\hat{\xi}}_{v,m,b}\right), \qquad (1)$$

and held constant over the update period $b+1$. The setpoint may be limited by a minimum and/or maximum value, which are typically set such that a complete probe cycle is feasible at any setpoint; this is not depicted in Figure 1.

## 2.3 Multiple Fixed-Step ES Controller

Figure 1 shows the Multiple Fixed-Step ES controller (MFSES) within in the blue dashed box. We start with the setpoint of the $m^{th}$ MFSES for the $b^{th}$ update period $\hat{w}_{m,b}$, which is held constant over the period. Throughout the $b^{th}$ update period, the controller adds a square wave perturbation with frequency $\omega_{w,m}$, zero-peak amplitude $a_{w,m}$, and phase $\theta_{w,m}$ to the setpoint to produce the controller output $w_{m,b}$. The controller collects measurements of the objective function $\Psi(\boldsymbol{\mu})$. Objective function measurements

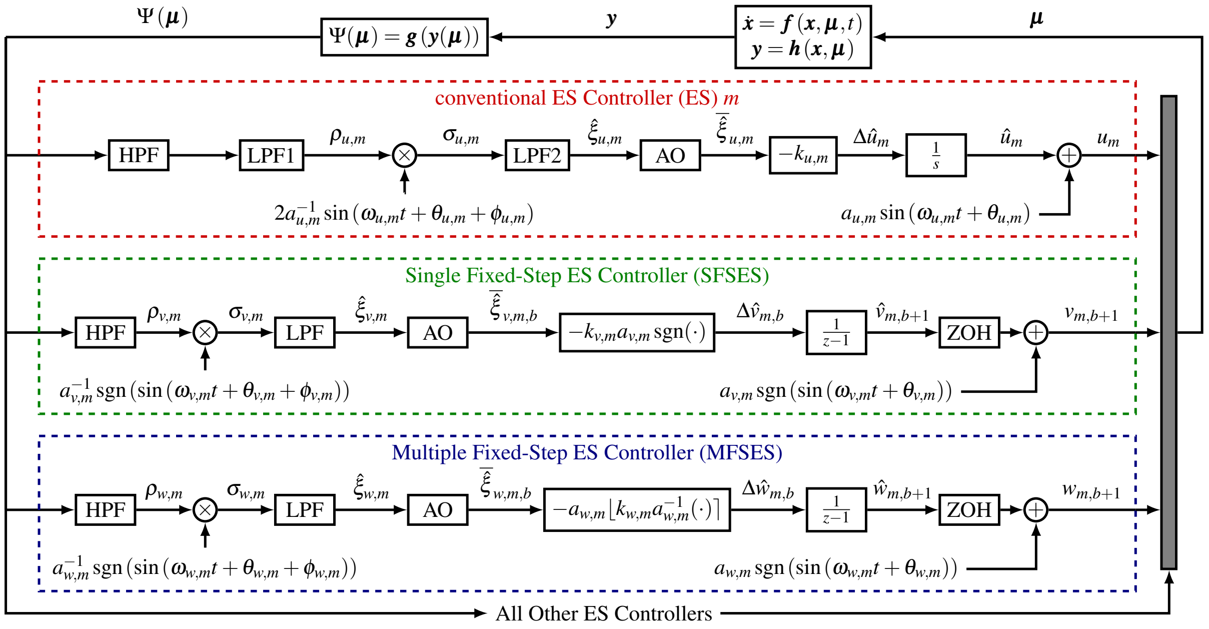

$\Psi(\boldsymbol{\mu})$
$\Psi(\boldsymbol{\mu}) = \boldsymbol{g}(\boldsymbol{y}(\boldsymbol{\mu}))$
$\boldsymbol{y}$
$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\mu}, t)$
$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\mu})$
$\boldsymbol{\mu}$
conventional ES Controller (ES) $m$
HPF
LPF1
$\rho_{u,m}$
$\sigma_{u,m}$
LPF2
$\hat{\xi}_{u,m}$
AO
$\bar{\hat{\xi}}_{u,m}$
$-k_{u,m}$
$\Delta\hat{u}_m$
$\frac{1}{s}$
$\hat{u}_m$
$u_m$
$2a^{-1}_{u,m}\sin(\omega_{u,m}t + \theta_{u,m} + \phi_{u,m})$
$a_{u,m}\sin(\omega_{u,m}t + \theta_{u,m})$
Single Fixed-Step ES Controller (SFSES)
HPF
$\rho_{v,m}$
$\sigma_{v,m}$
LPF
$\hat{\xi}_{v,m}$
AO
$\bar{\hat{\xi}}_{v,m,b}$
$-k_{v,m}a_{v,m}\,\mathrm{sgn}(\cdot)$
$\Delta\hat{v}_{m,b}$
$\frac{1}{z-1}$
$\hat{v}_{m,b+1}$
ZOH
$v_{m,b+1}$
$a^{-1}_{v,m}\,\mathrm{sgn}(\sin(\omega_{v,m}t + \theta_{v,m} + \phi_{v,m}))$
$a_{v,m}\,\mathrm{sgn}(\sin(\omega_{v,m}t + \theta_{v,m}))$
Multiple Fixed-Step ES Controller (MFSES)
HPF
$\rho_{w,m}$
$\sigma_{w,m}$
LPF
$\hat{\xi}_{w,m}$
AO
$\bar{\hat{\xi}}_{w,m,b}$
$-a_{w,m}\lfloor k_{w,m}a^{-1}_{w,m}(\cdot)\rceil$
$\Delta\hat{w}_{m,b}$
$\frac{1}{z-1}$
$\hat{w}_{m,b+1}$
ZOH
$w_{m,b+1}$
$a^{-1}_{w,m}\,\mathrm{sgn}(\sin(\omega_{w,m}t + \theta_{w,m} + \phi_{w,m}))$
$a_{w,m}\,\mathrm{sgn}(\sin(\omega_{w,m}t + \theta_{w,m}))$
All Other ES Controllers

**Figure 1.** Parallel operation of multiple ES Controllers of multiple types, to demonstrate the operational principles. For clarity, only one conventional ES controller, one Single Fixed-Step ES controller (SFSES), and one Multiple Fixed-Step ES controller (MFSES) are shown. However, this figure does not imply anything about the number or types of ES controllers that are present in an optimization or control scenario.

$\Psi(\boldsymbol{\mu})$ pass through a high-pass filter, to attenuate low frequency content of the objective function, such as those due to the setpoint, producing $\rho_{w,m}$.

The controller demodulates the signal $\rho_{w,m}$, by multiplying with a square wave with the same frequency $\omega_{w,m}$, amplitude $a^{-1}_{w,m}$, and phase $\theta_{w,m} + \phi_{w,m}$, giving $\sigma_{w,m}$. This signal passes through a low-pass filter, to attenuate high frequency content, producing an estimate of the gradient of $\Psi(\boldsymbol{\mu})$ with respect to $\hat{w}_m$, $\hat{\xi}_{w,m}$.

At end of the $b^{th}$ update period, the gradient estimate is averaged over $b$ by the averaging operator (AO), giving $\bar{\hat{\xi}}_{w,m,b}$. The gradient estimate is multiplied by gain $k_{w,m}$, and rounded to the nearest integer multiple of the probe amplitude, giving the setpoint change, such that:

$$\hat{w}_{m,b+1} = \hat{w}_{m,b} - a_{w,m}\lfloor k_{w,m}a^{-1}_{w,m}\bar{\hat{\xi}}_{w,m,b}\rceil, \qquad (2)$$

and held constant over the update period $b+1$. The setpoint may be limited by a minimum and/or maximum value, which are typically set such that a complete probe cycle is feasible at any setpoint; this is not shown in Figure 1.

## 2.4 Remarks on ES Controllers

In our library, the high-pass (HPF1) low-pass (LPF1) filter combination of the conventional ES controller typically takes two forms, depending on the use case. When only conventional ES controllers are used, the high-pass filter cutoff frequency is typically $1/10$ of the probe frequency, and the low-pass filter is omitted. When it is used in conjunction with one or more fixed-step ES controllers, the high-pass filter and low-pass filter cutoff frequencies are typically set to the probe frequency, and a rectification gain is applied such that the band-pass filter gain at the probe frequency is unity.

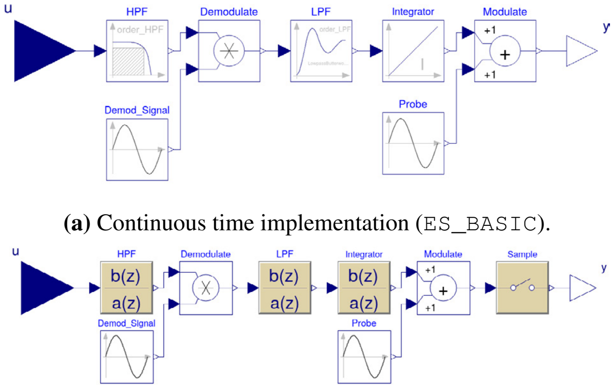
In our description of SFSES and MFSES, and in Figure 1, we have assumed that the probe zero-peak amplitude ($a_v$ for SFSES, and $a_w$ for MFSES) is the smallest possible discrete step. The reader should note that the probe amplitude may be an integer multiple of this value.

The reader should also note that the setpoint update of MFSES does not necessarily need to be rounded to the nearest integer multiple of $a_w$. Instead, a step size larger than $a_w$ can be applied for the setpoints, so that the setpoints will be rounded to the nearest integer multiple of $n_w a_w$ where $n_w$ can be chosen as any integer (e.g., if the stepsize is 5, the update size can be rounded to the nearest 15).

Finally, the reader should note that the setpoint updates of multiple discrete ES controllers are generally not synchronized.

## 3 Library Implementation

Our library includes several variants of the ES control algorithm. In our previous work, we considered a decentralized approach of the control scheme, where the objective is computed by a central entity, and the ES controller(s) use the common objective to manage their distributed resources. Therefore, we separated the objective function from the core ES control logic into a set of different blocks in this library. The control scheme combines a block for



**(a)** Continuous time implementation (`ES_BASIC`).

**(b)** Discrete time implementation (`discrete_ES_BASIC`).

**Figure 2.** Block diagrams for basic conventional ES Controllers.

the objective function and at least one ES logic block. contains several blocks for both the objective function and the ES logic. The objective function blocks allow the tracking of a target signal, or the regulation to keep the controlled variables within a defined interval. The different blocks also let the user include one or multiple signal(s) in the objective function. The ES logic blocks propose different implementations that are detailed in the upcoming sections. Finally, the library also features examples that are presented in Section 4.

### 3.1 Conventional Extremum Seeking Controller

The conventional Extremum Seeking controller (ES) is the most straightforward implementation of ES, as described in (Ariyur and Krstic, 2003). We built the block using the Modelica Standard Library, assembling the components as in similar fashion to Figure 1.

We have created several versions of this controller, implementing both continuous time and discrete time versions (see Figure 3). `ES_BASIC` is the basic version of ES in continuous time, shown in Figure 2a. `discrete_ES_BASIC` is a basic version of ES in discrete time, shown in Figure 2b. Both versions replace the bandpass filter with a high-pass filter, omit the gradient averaging operator, and omit setpoint limits.

`ES_ADV` is an advanced version of ES in continuous time, shown in Figure 3a. `discrete_ES_ADV` is an advanced version of ES in discrete time, shown in Figure 3b. Both `ES_ADV` and `discrete_ES_ADV` have a bandpass filter before demodulation, and a setpoint limiter. The user can choose to average the gradient in `ES_ADV`.

`ES_ADV_2D` is a 2-dimensional ES controller, in which two `ES_ADV` probe on the same frequency, with sinusoid phase offset by $\pi/2$ (Arnold et al., 2018; Sankur and Arnold, 2019).

The reader is invited to read (Choi et al., 2002) for an analysis of a discrete time implementation.

### 3.2 Single Fixed-Step Extremum Seeking Controller

`discrete_ES_SFS` is the Single Fixed-Step Extremum Seeking Controller (SFSES), and is shown in Figure 3c. We based our implementation on the Modelica Standard Library and developed a new square wave for the probe and demodulation signals.

The SFSES updates its setpoint by a fixed step size, with the direction determined by a sign function inside the `convergence` block. The `convergence` block contains thresholds, set by the user, for switching, such that if the magnitude of the input signal is too small, the sign function will return 0. The user can implement a minimum setpoint limit and/or a maximum setpoint limit.

### 3.3 Multiple Fixed-Step Extremum Seeking Controller

`discrete_ES_MFS` is the Multiple Fixed-Step Extremum Seeking controller (MFSES), and is shown in Figure 3d. We based our implementation on the Modelica Standard Library and developed a new square wave for the probe and demodulation signals.

The MFSES is similar to the SFSES, but its setpoint update is an integer multiple of the step size. The `round` block in the MFSES rounds its input, which is the gradient estimate multiplied by a gain, to the nearest integer multiple of the step size. The `round` block may round its input to zero, such that the setpoint is held constant. The user can implement a minimum setpoint limit and/or a maximum setpoint limit.

### 3.4 Functional Mock-up Units

Modelica also supports the Functional Mock-up Interface (FMI) standard. The FMI standard is an independent standard for model-exchange (ME) and co-simulation (CS). It establishes a standard binary format, the Functional Mock-up Unit (FMU), that can be imported into other FMI compliant software tools. It allows to combine models from different languages / tools in a single simulation (Blochwitz et al., 2012).

We exported the controller models as FMUs, to extend the outreach of the library to users outside of the Modelica community. Each FMU was exported for both CS and ME simulation modes that are defined in the FMI standard.

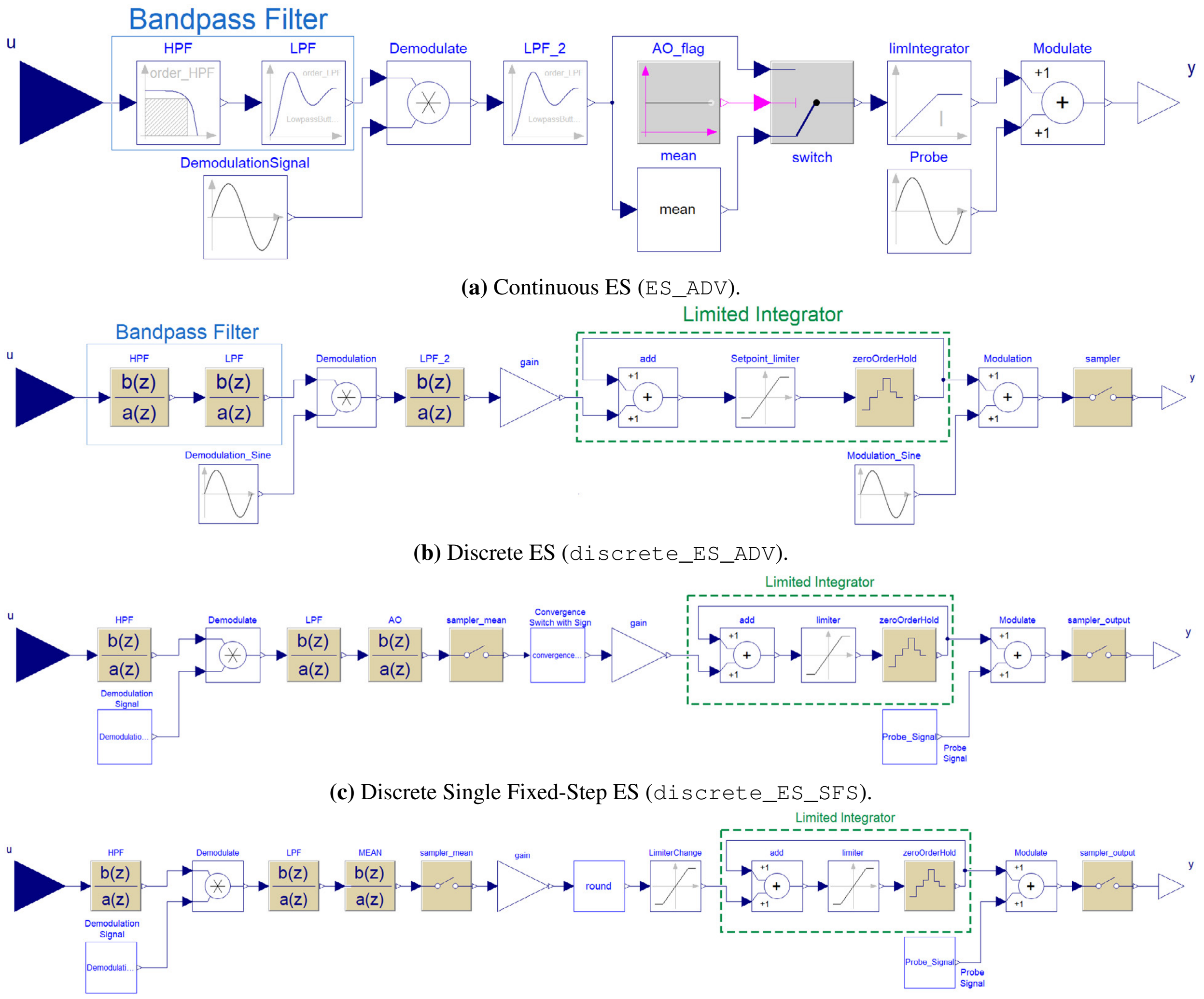
## 4 Examples

In this section, we present several examples in which one or more ES Controllers are employed to optimize an objective function, or implement control of an mechanical or electrical system.

### 4.1 Quadratic Objective Function minimized by a Single ES Controller

In our first example, a single ES controller optimizes its setpoint to minimize the following quadratic function:

$$\Psi(u) = (u-1)^2\,,$$



**(a)** Continuous ES (`ES_ADV`).

**(b)** Discrete ES (`discrete_ES_ADV`).

**(c)** Discrete Single Fixed-Step ES (`discrete_ES_SFS`).

**(d)** Discrete Multiple Fixed-Step ES (`discrete_ES_MFS`).

**Figure 3.** Block diagrams of several ES controllers of the library.

where $u$ is the output of the ES controller. We ran three scenarios: one where a conventional ES minimizes the quadratic function, one where SFSES minimizes the quadratic function, and one where a MFSES minimizes the quadratic function, as seen in Figure 4. The controllers do not have knowledge of the system or objective function, but receive measurements of the objective function value.

Figure 5 shows simulation results of the three scenarios. All three ES controllers optimize their setpoint to minimize the objective function. SFSES takes the longest time to converge, as it can only update its setpoint by one probe amplitude at a user defined rate, in this case 2 probe cycles. Its oscillatory behavior after reaching the optimal value is due to the setpoint update switch algorithm, and can be eliminated by implementing a switching threshold on the averaged gradient estimate.

## 4.2 Single Quadratic Objective Function Minimized by Three ES Controllers in Parallel

In this example, shown in Figure 6, one conventional ES controller, one SFSES controller, and one MFSES controller operate in parallel to minimize the following function:

$$\Psi(u,v,w) = (u-1)^2 + (v-2)^2 + (w-3)^3 \,,$$

where $u$ is the ES output, $v$ is the SFSES output, and $w$ is the MFSES output. The controllers do not have knowledge of the objective function, but receive measurements of its value. Simulation results are given in Figure 7. This example shows that multiple ES controllers, and multiple types of ES controllers, can detect their own impact on an objective and optimize themselves, as long as the frequency of each controller is not equal to, or a multiple of, the frequency of any other controller.

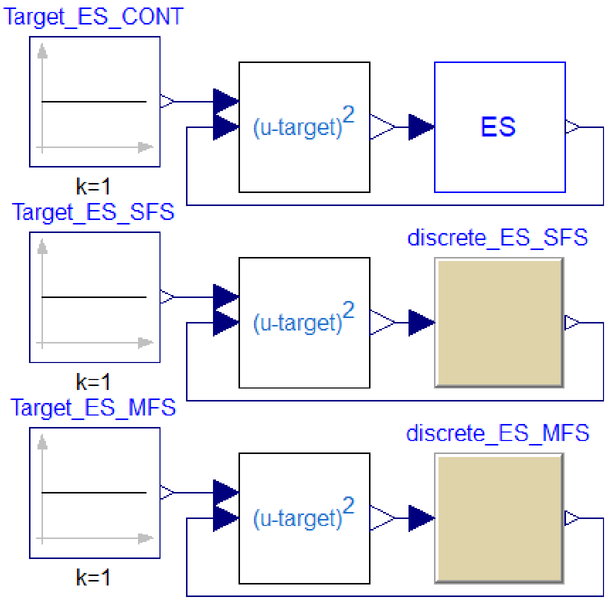


**Figure 4.** Block diagram for ES, SFSES, and MFSES controllers minimizing identical quadratic functions.

## 4.3 ES Control of Mass-Spring-Damper System

This example is an extension of the mass-spring-damper example from `Modelica.Mechanics.Translational.Examples.Damper` from the Modelica Standard Library. We extended the model with a position sensor, and a force applied to the mass and regulated by conventional ES, as in Figure 8. The objective function is:

$$\Psi = (y(u) - 5)^2,$$

where $y(u)$ is the position of the mass, and is a function of the ES regulated force $u$. Figure 9 shows that the controller converges to the optimal force value for the mass to be nearest to the desired position after roughly 100 seconds. The long rise time is due to the "slow" probe frequency, chosen to be a factor of 10 times slower than the natural frequency of the mass-spring-damper system, so as to avoid exciting the system at the natural frequency. ES convergence speed is dependent on several factors, one of which being probe frequency.

## 4.4 ES Control of Rolling Wheel Speed

The following example is an extension of the Modelica Standard Library model `Modelica.Mechanics.Rotational.Examples.RollingWheel`. The example features a wheel with an input torque, and a nonlinear friction torque, applied to it, as in Figure 10.

We replaced the `torqueStep` block with a controllable torque, regulated by ES. The translational speed of the wheel is measured by a speed sensor. The controller operates to minimize difference between the wheel's rotation speed and a target speed by controlling the torque as in:

$$\Psi(u) = (y(u) - 1)^2,$$

where $y(u)$ is the wheel speed, which is a function of the torque applied to the wheel $u$, regulated by ES.

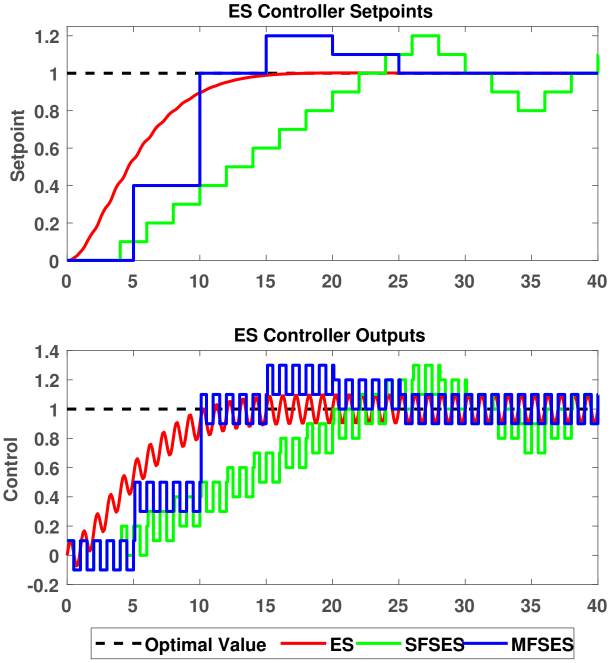


**Figure 5.** ES controller setpoints, and ES controller outputs for three separate simulations in which a single ES controller minimizes a simple quadratic function.

To compare the performance of the ES, SFSES, and MFSES controllers, we performed three separate simulations in which each type of ES controller optimized the torque applied to the wheel. Figure 11 shows the results of this experiment. All three types of controllers are able to regulate the torque to achieve the desired wheel speed. The conventional ES controller converged to the optimal torque as its setpoint can be updated by any size. The SFSES controller exhibited oscillatory behavior around the optimal torque value, as its setpoint update consists of a fixed step in either direction, depending on the sign of the gradient value. The MFSES controller overshot the optimal torque value, and eventually converged to a steady value near the optimal. This is due to the gradient value threshold for setpoint updates.

## 4.5 Power System: Distribution Feeder

In the last example, the ES library was used to build a power system example similar to our previous works (Sankur and Arnold, 2019), where the aim was to track a reference for the active (P) and reactive (Q) power injections at the feeder head. We used models from the OpenIPSL project (Baudette et al., 2018), in particular a three-phase implementation of the IEEE 13 test model.

The example was build to show the possibility to combine several ES resources into a single control scheme.

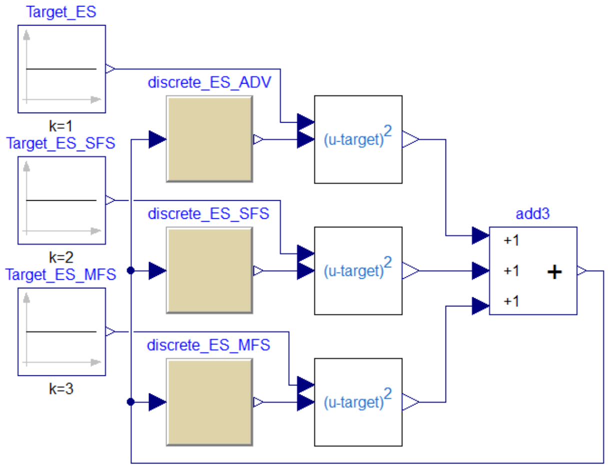


**Figure 6.** Block diagram of ES, SFSES, and a MFSES controllers operating in parallel to minimize a common quadratic objective function.

The modified model including the ES control scheme is shown in Figure 12. The scheme was designed arbitrarily to include ES resources throughout the feeder, connected to one, two, or three phases. In total, we added two three-phase resources (at nodes 632 and 680), one two-phase resource (at node 646), and one single-phase resource (at node 611) that were configured to be ES managed resources. These were connected to nine instances of the continuous 2D-ES (ESADV_2D) to manage both P and Q for each phase / resource Each individual ES was configured with the same gain corresponding to 0.2 kVA and a probing amplitude of 10 kVA. The probing frequencies were selected as $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}/2$, $\sqrt{7}/2$, $\sqrt{11}/3$, $\sqrt{13}/3$, $\sqrt{17}/4$, $\sqrt{19}/4$, $\sqrt{23}/4$.

The objective considered in this experiment was to reach different P and Q targets for each phase at the feeder head. We prepared a block to compute and extract the P and Q measurements of each phase, and placed it at the feeder head (node 650). The objective block was added and connected to constant targets that were set for P and Q for each phase respectively. The target values for $P_a$, $P_b$, $P_c$, $Q_a$, $Q_b$, and $Q_c$ are chosen arbitrarily to 1, 1.5, 2 MW and $-0.2$, $-0.3$, $-0.1$ MVAr respectively for the purpose of the example, yielding the following objective function:

$$\Psi = (P_a - 1)^2 + (P_b - 1.5)^2 + (P_c - 2)^2 \\ + (Q_a + 0.2)^2 + (Q_b + 0.3)^2 + (Q_c + 0.1)^2$$

The common objective was broadcast to every ES resources.

The example was initialized with all ES resources at zero The simulation was executed to reach time instant $t = 60s$ to let all the variables reach their target value. Figure 13 shows that the ES control sccheme is working as expected and all target values are reached. Note that the computation time of this particular example can be sharply reduced by toggling the *DAEsolver* mode in Dymola, which speeds up larger differential and algebraic equation systems, such as that of power system models

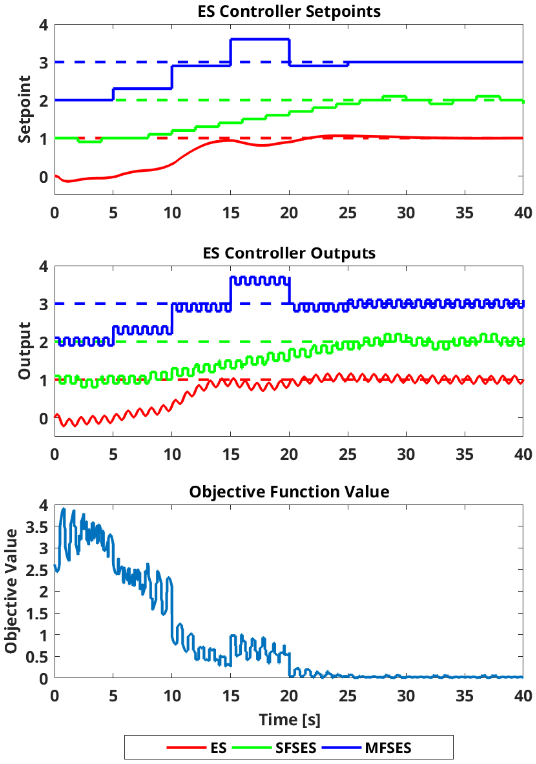


**Figure 7.** Controller setpoints, controller outputs, and objective function value for the scenario in which ES, SFSES, and MFSES controllers operate in parallel to minimize a quadratic objective function. Dashed lines represent the optimal value for the corresponding controller.

## 5 Conclusion

In this paper, we introduce an open-source Modelica package that implements several variants of Extremum Seeking (ES). ES offers an attractive solution for model-free optimization and control, and is applicable to a wide range of problems. In this paper we introduced an open-source package that implements several variants of ES stemming from the Author's previous work in grid applications. The implementation was carried out in Modelica, providing a flexible framework to model multiple types cyber-physical systems. The Modelica ES implementation allowed us to prepare examples for different domains that we included in the library. Thanks to Modelica's ability to combine continuous and discrete models, it was also possible to compare different ES variants in practical applications and its impact on the physical systems being controlled.

Our efforts focused on using a generic design to broaden the target audience to different domains. The ES control scheme was broken down into blocks that sepa-

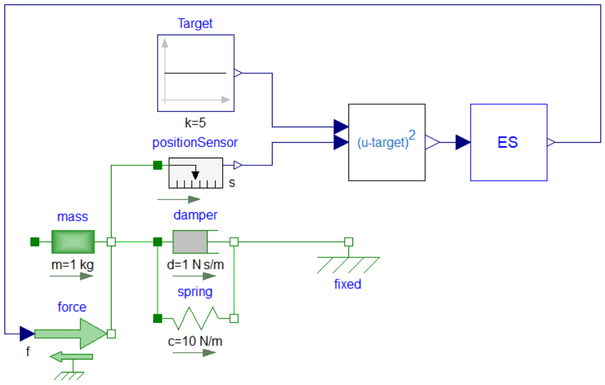


**Figure 8.** Block diagram of the Mass-Spring-Damper example.

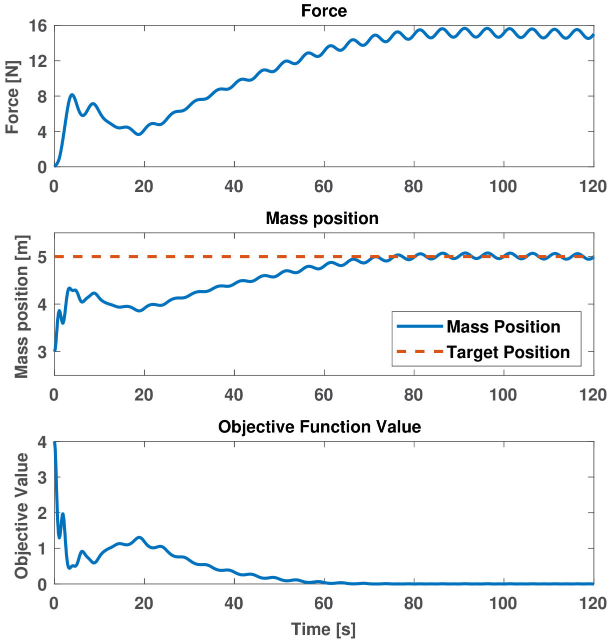


**Figure 9.** Force applied to the mass, mass position, and objective function value for the mass-spring-damper example.

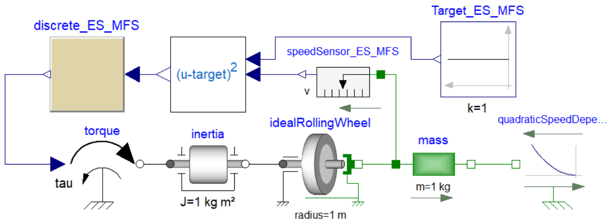


**Figure 10.** Block diagram of the rolling-wheel example.

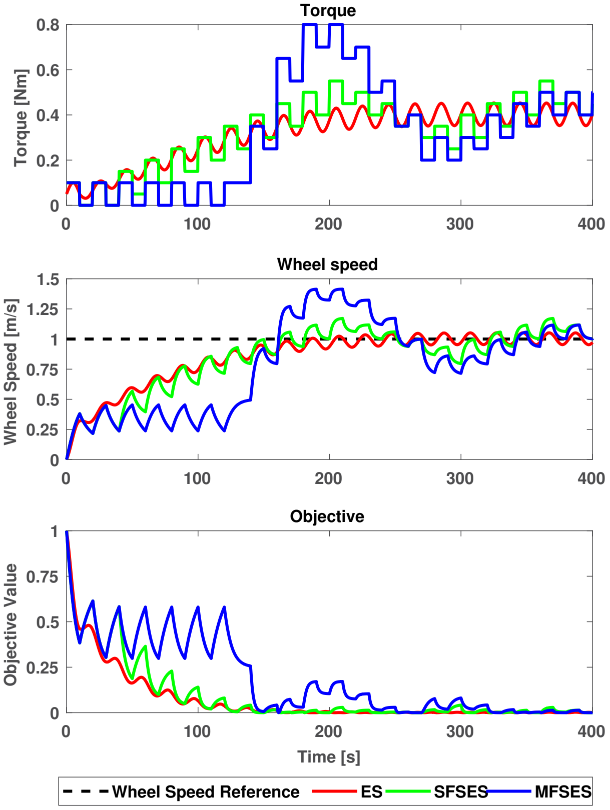


**Figure 11.** Torque, translational wheel speed, and objective function for the rolling wheel example, for three scenarios with ES, SFSES, or MFSES.

rate the objective function from the ES logic. This allows a flexible assembly of the inter-compatible blocks to suit the needs of different research communities of various domains. Finally, we included compiled versions of the ES blocks in the form of FMUs to further extend the possible applications of the library to any FMI compliant simulation tools.

We plan to extend and further develop our Modelica ES library in several key ways. We plan to add equilibrium based switching for probe amplitude decay, as in the work of several of this work's authors (Sankur and Arnold, 2019). A second key area is to add an estimator for the phase difference between controller output (system input) and objective function measurements, This is especially important when using ES to optimize or control dynamic systems. It also has applications for control systems that rely on digital network communications that introduce non deterministic delays. We also plan to implement more examples in different domains, including optimization of heat and fluid models.

## References

Kartik B Ariyur and Miroslav Krstic. *Real-time optimization by extremum-seeking control*. John Wiley & Sons, 2003.

D. B. Arnold, M. D. Sankur, M. Negrete-Pincetic, and D. Callaway. Model-free optimal coordination of distributed energy resources for provisioning transmission-level services. *IEEE Trans. Power Syst.*, 33(1):817–829, Jan. 2018. ISSN 0885-8950. doi:10.1109/TPWRS.2017.2707405.

Maxime Baudette, Marcelo Castro, Tin Rabuzin, Jan Lavenius, Tetiana Bogodorova, and Luigi Vanfretti. OpenIPSL: Open-Instance Power System Library–Update 1.5 to "iTesla Power

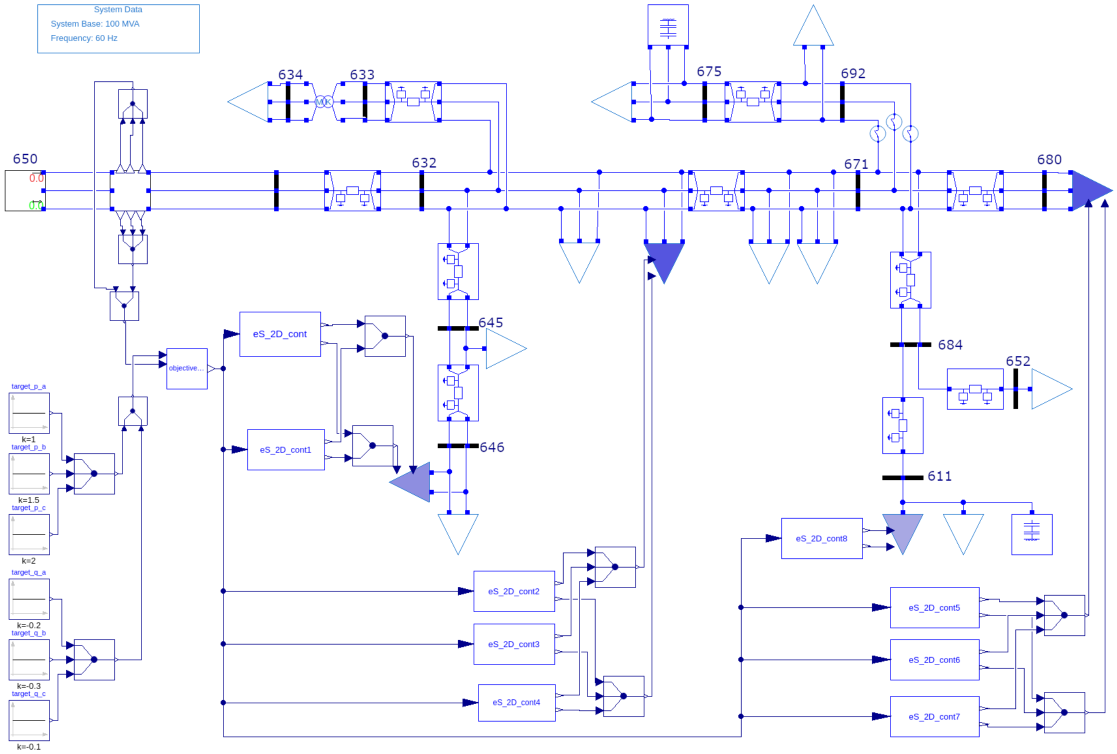


**Figure 12.** Block diagram of the IEEE 13 Example (ES resources highlighted in blue).

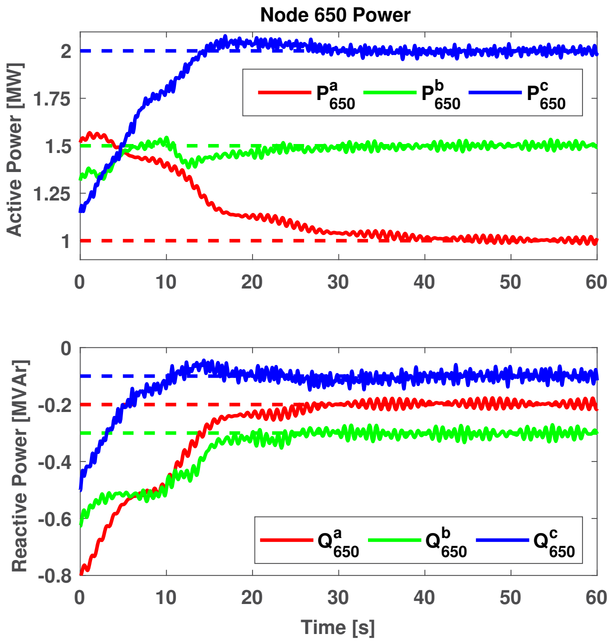


**Figure 13.** Active and Reactive power for each phase at Node 650 of the IEEE 13 node test feeder. Dashed lines represent the corresponding target value.

Systems Library (iPSL): A Modelica library for phasor time-domain simulations". *SoftwareX*, 7:34–36, 2018.

Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

Joon-Young Choi, Miroslav Krstic, Kartik B Ariyur, and Jin Soo Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on automatic control*, 47(2):318–323, 2002.

N. J. Killingsworth, S. M. Aceves, D. L. Flowers, F. Espinosa-Loza, and M. Krstic. HCCI engine combustion-timing control: Optimizing gains and fuel consumption via extremum seeking. *IEEE Transactions on Control Systems Technology*, 17(6):1350–1361, Nov 2009. doi:10.1109/TCST.2008.2008097.

M. Krstic, A. Ghaffari, and S. Seshagiri. Extremum seeking for wind and solar energy applications. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 6184–6193, June 2014. doi:10.1109/WCICA.2014.7053780.

Michael Sankur and Daniel Arnold. Extremum seeking control of distributed energy resources with decaying dither and equilibrium-based switching. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

Chunlei Zhang and Raul Ordonez. *Extremum-Seeking Control and Applications A Numerical Optimization-Based Approach*. 01 2012. ISBN 978-1-4471-2223-4. doi:10.1007/978-1-4471-2224-1.

Chunlei Zhang, Daniel Arnold, Nima Ghods, Antranik Siranosian, and Miroslav Krstic. Source seeking with nonholonomic unicycle without position measurement and with tuning of forward velocity. *Systems & control letters*, 56(3): 245–252, 2007.

# Modeling and Simulation of Filippov System Models with Sliding Motions using Modelica

Mohammed Ahsan Adib Murad[1] Luigi Vanfretti[2] Federico Milano[1]

[1]School of Electrical Engineering, University College Dublin, Ireland, mohammed.murad@ucdconnect.ie, federico.milano@ucd.ie
[2]ECSE, Rensselaer Polytechnic Institute, Troy, NY, USA, vanfrl@rpi.edu

## Abstract

This paper presents a generalized modeling formulation for implementation of dynamical system models that exhibit sliding behavior. The proposed formulation is based on Filippov theory, and is implemented using Modelica. The main advantage of the developed framework is that it effectively removes numerical chattering and trajectory deadlock. The robustness of the formulation is assessed considering three example system models: a stick-slip system, a relay feedback system and an anti-windup Proportional Integral (PI) controller in a power system application, i.e. an automatic voltage controller.

*Keywords: Discontinuity, non-smooth, Filippov, hybrid dynamics, chattering.*

## 1 Introduction

The Modelica Language is becoming an industry standard to model dynamical systems described by a set of Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs) with mixed continuous and discrete variables, known as hybrid systems (Fritzson, 2014). A subclass of hybrid systems referred to as Filippov systems (Filippov, 1988) are those where discontinuities appear in vector fields, i.e. in the right hand side of the model's equations. If the solution of Filippov systems enters into a constrained subset of the state space, known as *sliding*, the formalism given by Filippov (Filippov, 1988) allows defining a vector field on the sliding surface to properly handle discontinuities.

A Modelica implementation of the Filippov systems without considering Filippov formalism leads to chattering-Zeno-type deadlocks, which consists of infinitely many instantaneous switches of the discrete variables during time domain simulation (Aljarbouh and Caillaud, 2016). This significantly restrains the performance of the solvers of Modelica simulation tools and can lead to a simulation halt. More importantly, such chattering does not represent the complicated real physical system chattering (Levant, 2010). Therefore a generalized formulation is required for smooth continuation of trajectories. This paper fills this gap.

The authors in (Aljarbouh and Caillaud, 2016) and (Aljarbouh et al., 2016) discussed chattering problem of Filippov systems in Modelica, using OpenModelica. However, to solve this problem, a framework based on the Functional Mock-up Interface standard (FMI) and Acumen (Taha et al., 2015) is proposed. The authors in (Suski and Pytlak, 2017) presented a race car model based on the Filippov formalism implemented in OpenModelica, but did not provide implementation details and a generalized formulation. It is worth mentioning that there exist heuristic methods to solve the chattering issues (Bonilla et al., 2011). However, such methods lack a systematic generalization.

There exists several methods to formulate hybrid systems as smooth systems, for example are: linear complementarity description (Pfeiffer and Glocker, 1996); augmented Lagrangian approach (Leine and Nijmeijer, 2013); parameterized curve description (Otter et al., 1999) and then use a dedicated numerical method to solve time domain simulations. Such techniques require special treatment of numerical methods and do not model precisely discrete (non-smooth) events (Leine and Nijmeijer, 2013). Considering this, we propose a general purpose hybrid design for implementing Filippov systems using Modelica that allows smooth integration along chattering regions.

The main contributions of this paper are as follows:

- A generic formulation based on Filippov Theory (FT) for the implementation and direct numerical simulation of Filippov systems with one sliding surface using Modelica is proposed.
- A validation of the proposed formulation is performed comparing the results with a Matlab implementation and via simulation in two Modelica tools, namely OpenModelica and Dymola.

The remainder of the paper is organized as follows. Section 2 provides a background on FT. Section 3 presents the proposed generalized formulation for the implementation of Filippov system models using Modelica. Case studies are discussed in Section 4 where three examples are presented: a stick-slip system, a relay feedback system and an anti-windup PI controller in a power system voltage control application. Conclusions and future work directions are drawn in Section 5.

# 2 Filippov Systems

Filippov systems are dynamical system models with discontinuous right-hand side first-order ordinary differential equations (Filippov, 1988). Consider the following switched dynamical system of equations:

$$\dot{x} = f(x) = \begin{cases} f_1(x) & \text{when } h(x) < 0 \\ f_2(x) & \text{when } h(x) > 0 \end{cases} \tag{1}$$

where, the event function $h : \mathbb{R}^n \to \mathbb{R}$ and an initial condition $x(t_0) = x_0$ are known. The state space $\mathbb{R}^n$ is separated by a hyper-surface $\Sigma$ into two regions $R_1$ and $R_2$ as follows:

$$\begin{aligned} R_1 &= \{x \in \mathbb{R}^n \mid h(x) < 0\}, \\ R_2 &= \{x \in \mathbb{R}^n \mid h(x) > 0\}, \\ \Sigma &= \{x \in \mathbb{R}^n \mid h(x) = 0\}, \end{aligned} \tag{2}$$

such that $\mathbb{R}^n = R_1 \cup \Sigma \cup R_2$, assuming at $x \in \Sigma$ the gradient of $h$ never vanishes, i.e. $h_x(x) \neq 0$ for all $x \in \Sigma$.

The *Filippov convex method* (Filippov, 1988) states that the vector field on the surface of discontinuity is a convex combination of the two vector fields in the different regions of the state-space:

$$\dot{x} = f(x) = \begin{cases} f_1(x), & x \in R_1 \\ \overline{co}\{f_1(x), f_2(x)\}, & x \in \Sigma \\ f_2(x), & x \in R_2 \end{cases} \tag{3}$$

where, $\overline{co}(f_1, f_2)$ is the minimal closed convex set containing $f_1$ and $f_2$, i.e.

$$\overline{co}\{f_1, f_2\} = \{f_F : x \in \mathbb{R}^n \to \mathbb{R}^n : f_F = (1-\alpha)f_1 + \alpha f_2\}, \tag{4}$$

where $\alpha \in [0, 1]$.

Consider the trajectory starting at $t_0$ with $\dot{x} = f_1(x)$, with $x(t_0) = x_0$ reaches at $\Sigma$ in finite time $(t_k)$. Then at $t_k$ the trajectory can cross, slide or exit $\Sigma$. In such situation, the first order theory given by Filippov explains how to solve these equations as summarized in the following section.

## 2.1 Filippov first order theory

Filippov first order theory defines the vector field if the solution approaches the discontinuous surface. Let $x \in \Sigma$ and $n(x)$ is the unit normal to $\Sigma$ at $x$ i.e. $n(x) = \frac{h_x(x)}{\|h_x(x)\|}$ where, $h_x(x) = \nabla h(x)$ and $\nabla = \frac{\partial}{\partial x}$ ; the components of $f_1(x)$ and $f_2(x)$ onto the normal to the $\Sigma$ are $n^T(x)f_1(x)$ and $n^T(x)f_2(x)$ respectively.

### 2.1.1 Transversal crossing

If $x \in \Sigma$, then

$$(n^T(x)f_1(x)).(n^T(x)f_2(x)) > 0, \tag{5}$$

i.e. the trajectory leaves $\Sigma$. The system will return to $R_1$ with $f = f_1$, if $n^T(x)f_1(x) < 0$ or it will proceed to $R_2$ with $f = f_2$ (see Fig. 1[I]), if $n^T(x)f_1(x) > 0$.

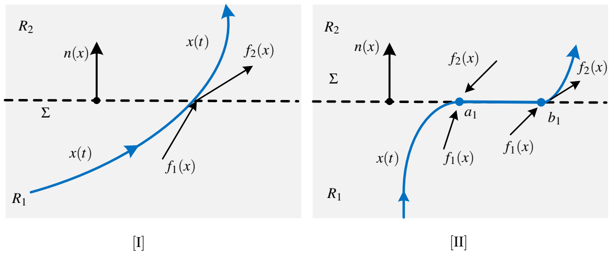


**Figure 1.** Different regions of the state space with [I] transversal and [II] sliding trajectory.

### 2.1.2 Sliding mode

Sliding occurs, at $x \in \Sigma$ if,

$$(n^T(x)f_1(x)).(n^T(x)f_2(x)) < 0\,. \tag{6}$$

An unique attracting sliding mode will occur if,

$$(n^T(x)f_1(x)) > 0 \quad \text{and} \quad (n^T(x)f_2(x)) < 0,\ x \in \Sigma\,, \tag{7}$$

and the solution does not leave $\Sigma$ (see $a_1$ in Fig. 1[II]). During sliding the time derivative $f_F$ is given by:

$$f_F(x) = (1-\alpha(x))f_1(x) + \alpha(x)f_2(x)\,, \tag{8}$$

where, $\alpha(x)$ is given by [proof, see (Filippov, 1988)]:

$$\alpha(x) = \frac{n^T(x)f_1(x)}{n^T(x)(f_1(x) - f_2(x))}\,. \tag{9}$$

If the signs are opposite in (7) a *repulsive sliding mode* will occur. In such a case, the solution is not unique, and thus, is not considered in this work.

### 2.1.3 Exit conditions

During sliding mode if one of the vector fields starts to point away, the solution continues above or below the sliding surface (see $b_1$ in Fig. 1[II]). The exit point is calculated by finding either the root $\alpha(x) = 0$ or $\alpha(x) = 1$ as appropriate. The following remarks are relevant:

- If $f_F(x) \neq f_1(x)$, $f_F(x) \neq f_2(x)$ such a solution is often called a *sliding motion*.
- If at the point of discontinuity, condition (6) becomes $\leq 0$ and $f_1(x) \neq f_2(x)$ then a continuous vector-valued function $f_F(x)$ is given which determines the velocity of motion $\dot{x} = f_F(x)$ along the discontinuity line. If $n^T(x)f_1(x) = 0$ then $f_F(x) = f_1(x)$; if $n^T(x)f_2(x) = 0$ then $f_F(x) = f_2(x)$.

# 3 Filippov Theory Based Formulation

Filippov systems can be implemented in a computer language considering *event driven* or *time stepping* approaches (Dieci and Lopez, 2012). The former method simulates the system model by detecting the actual event

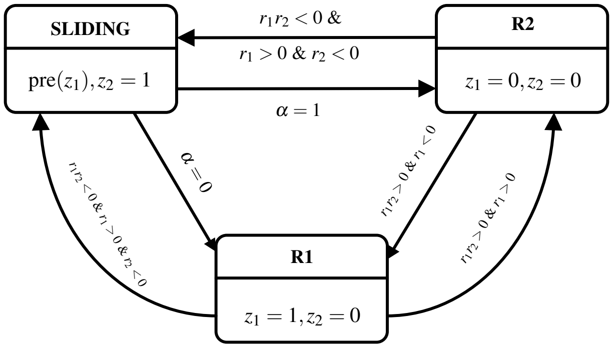


**Figure 2.** Generalized state transitions of Filippov systems.

time whereas the latter method without event detection (Piiroinen and Kuznetsov, 2008).

Modelica allows the conditional equations become activated or deactivated at the event instant (Fritzson, 2014) and the exact event instant can be detected during time domain simulation. Utilizing this Modelica language feature, in the following, we propose a formulation considering an accurate event detection method for the implementation of Filippov system models in Modelica.

## 3.1 General purpose design

According to FT, a system can have three states, the two states for $h(x) < 0$ (R1) and $h(x) > 0$ (R2), and a new state called SLIDING, characterized by $h(x) = 0$. For implementation using Modelica, it is convenient to introduce two discrete variables, say $z_1$ and $z_2$, into the differential equations, as follows:

$$\dot{x} = f_1(x)z_1(1-z_2) + f_2(x)(1-z_1)(1-z_2) + f_F(x)z_2\ . \tag{10}$$

Observe that in the above equation, depending on the values of $z_1$ and $z_2$ (e.g. 1 or 0), a proper vector field needs to be activated during time domain simulation. Lets define $r_1 = h_x^T(x)f_1(x)$ and $r_2 = h_x^T(x)f_2(x)$ and Fig. 2 shows the changes in $(z_1, z_2)$ for the three states and the conditions to move from one state to another. All these conditions are based on FT and are evaluated the moment at which the event function ($h(x)$) crosses zero.

In the SLIDING state the value of $z_2 = 1$. This automatically deactivates $f_1(x)$ and $f_2(x)$ (see (10)) without the need of changing the value of $z_1$. So the previous value (pre($z_1$)) is retained. The sliding vector field $f_F(x)$ is derived explicitly according to (8). The exit conditions are defined based on (9). In particular, $\alpha(x) = 0$ and $\alpha(x) = 1$ are the conditions that indicate to move to the R1 and R2 regions, respectively.

The case studies below show the steps of the proposed approach followed during time domain simulation in Modelica tools.

# 4 Case Studies

In this section we discuss the implementation and validation of the Filippov systems in Modelica considering our generalized formulation. The case studies are posted online: `https://github.com/ALSETLab/Modelica_Fillipov_Sliding_Models`.

## 4.1 Example 1: Stick-slip system

Consider the two-dimensional system (so-called stick-slip system) (Dieci and Lopez, 2009, 2012)

$$\dot{x} = f(x) = \begin{cases} f_1(x) & \text{when } h(x) < 0 \\ f_2(x) & \text{when } h(x) > 0 \end{cases}$$

with

$$f_1(x) = \begin{pmatrix} x_2 \\ -x_1 + \frac{1}{1.2-x_2} \end{pmatrix}, \qquad f_2(x) = \begin{pmatrix} x_2 \\ -x_1 - \frac{1}{0.8+x_2} \end{pmatrix},$$

and $h(x) = x_2 - 0.2$. This system has a single switching manifold with two dynamic states.

## 4.2 Direct implementation

A direct implementation of this system using Modelica is as follows:

```
model Stick_slip "Stick-Slip System"
  Real x1 (start=0);
  Real x2 (start=0);
  Integer z1 (start=1);
  Real h;
equation
  der(x1) = x2*z1 + x2*(1-z1);
  der(x2) = (-x1+(1/(1.2-x2)))*z1+(-x1-
      (1/(0.8+x2)))*(1-z1);
  h = x2 - 0.2;
  when h < 0 then
    z1 = 1;
  elsewhen h > 0 then
    z1 = 0;
  end when;
end Stick_slip;
```

OpenModelica and Dymola were used to simulate this example, however these tools halt when simulating this simple model. In OpenModelica, all solvers fail to simulate and report an error message: *Chattering detected around time 0.221654558425..0.221654756475 (100 state events in a row with a total time delta less than the step size 0.001).* On the other hand, Dymola's solver DASSL fails to continue the simulation. However some solvers for example: RkFix2 and Euler allows to continue simulation exposing chattering as shown in Fig. 3. Because of unnecessary chattering during the simulation, the results are not mathematically correct and it is not possible to understand the dynamic behavior of the real physical system.

### 4.2.1 Implementation using Filippov theory

The surface $\Sigma$ is defined by zero of $h(x) = x_2 - 0.2$. Here, $h_x(x) = [\frac{\delta h(x)}{\delta x_1} \quad \frac{\delta h(x)}{\delta x_2}]^T = [0 \quad 1]^T$, thus on $\Sigma$ (i.e. $x_2 =$

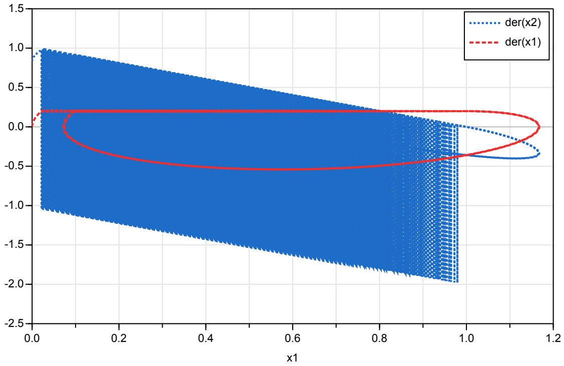


**Figure 3.** Time derivative of state variables $(\dot{x}_1, \dot{x}_2)$ of stick-slip system without Filippov sliding simulated in Dymola.

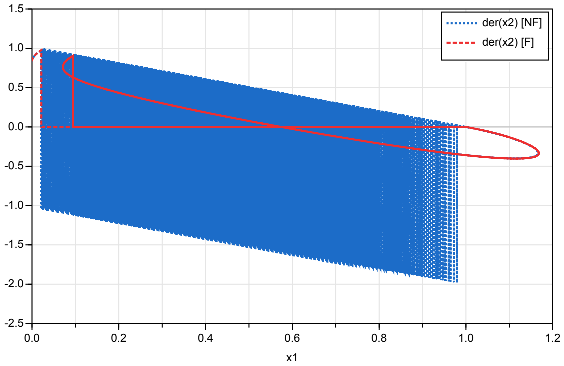


**Figure 4.** Time derivative of state variable $(\dot{x}_1)$ of stick-slip system without (NF) and with (F) Filippov theory simulated in Dymola.

0.2), calculating,

$$r_1 = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ -x_1 + \frac{1}{1.2 - x_2} \end{pmatrix} = -x_1 + 1 \,,$$

$$r_2 = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ -x_1 - \frac{1}{0.8 + x_2} \end{pmatrix} = -x_1 - 1 \,.$$

Therefore according to (6), $x_1 \in (-1, 1)$ there will be an attractive sliding mode on $\Sigma$. This sliding vector field on $\Sigma$ is calculated using equations (8,9):

$$\alpha(x) = \frac{n^T(x) f_1(x)}{n^T(x)(f_1(x) - f_2(x))} = \frac{-x_1 + 1}{2} \,,$$
$$f_F(x) = \begin{pmatrix} x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix} \,,$$

which means that on the sliding surface $\Sigma$ the $x_1$ state will grow linearly until reaching the value $x_1 = 1$, at which the point the trajectory will leave $\Sigma$ with vector field $f_1$ as for $x_1 = 1$ the $\alpha(x) = 0$.

Using the proposed FT based approach, the expressions derived above can be used to implement the model as follows (equation part of the model is given):

```
der(x1) = x2*z1*(1-z2) + x2*(1-z1)*(1-z2)
    + x2*z2;
der(x2) = (-x1+(1/(1.2-x2)))*z1*(1-z2) +
    (-x1-(1/(0.8+x2)))*(1-z1)*(1-z2);
h = x2 - 0.2;
r1 = -x1+1;
r2 = -x1-1;
a = (-x1+1)/2;
zeroCrossing.u = h;
zeroCrossing1.u = (-x1+1)/2;
zeroCrossing2.u = 1-a;
when zeroCrossing.y then
     if r1*r2 < 0 then
       if r1 > 0 and r2 < 0 then
       z1 = pre(z1);
       z2 = 1;
       else
       z1 = pre(z1);
       z2 = 0;
       end if;
      elseif r1*r2 > 0 then
        if r1 < 0 then
        z1 = 1;
        z2 = 0;
        elseif r1 > 0 then
        z1 = 0;
        z2 = 0;
        else
        z1 = pre(z1);
        z2 = 0;
        end if;
      else
        z1 = pre(z1);
        z2 = 0;
      end if;
elsewhen zeroCrossing1.y and pre(z2) == 1
    then
        z1 = 1;
        z2 = 0;
elsewhen zeroCrossing2.y and pre(z2) == 1
    then
        z1 = 0;
        z2 = 0;
 end when;
```

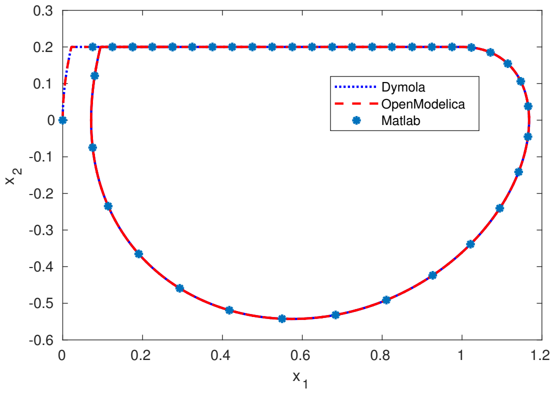


**Figure 5.** Periodic trajectories of the stick-slip system obtained in different simulation software tools.

Using this implementation the simulation of this system can be successfully carried out in both OpenModelica and

Dymola without numerical issues. Results are compared in Fig. 4. Observe that considering the Filippov sliding condition the simulation continues without any chattering. For further validation of the proposed generalized formulation and its implementation, the trajectories in Fig. 5 obtained using Modelica tools (OpenModelica and Dymola) and compared with the results obtained using Matlab utilizing the method in (Piiroinen and Kuznetsov, 2008).

### 4.3 Example 2: A relay feedback system

A relay feedback system with single-input and single-output is as follows (Piiroinen and Kuznetsov, 2008):

$$\begin{aligned} \dot{x} &= \boldsymbol{A}x + \boldsymbol{B}u \\ y &= \boldsymbol{C}x \\ u &= -\mathrm{sgn}(y) \end{aligned} \tag{11}$$

or

$$\dot{x} = \begin{cases} \boldsymbol{A}x + \boldsymbol{B}, & \text{when } \boldsymbol{C}x < 0 \\ \boldsymbol{A}x - \boldsymbol{B}, & \text{when } \boldsymbol{C}x > 0 \end{cases} \tag{12}$$

where,

$$\boldsymbol{A} = \begin{pmatrix} -(2\zeta\omega+1) & 1 & 0 \\ -(2\zeta\omega+\omega^2) & 0 & 1 \\ -\omega^2 & 0 & 0 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 1 \\ -2\sigma \\ 1 \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The state vector of this model is $x = [x_1, x_2, x_3]^T$ and the discontinuity surface $\Sigma$ is defined by $h(x) = x_1$. Re-writing the dynamical system according to FT,

$$\dot{x} = f(x) = \begin{cases} f_1(x) & \text{when } h(x) < 0 \\ f_2(x) & \text{when } h(x) > 0 \end{cases} \tag{13}$$

with

$$f_1(x) = \begin{pmatrix} -(2\zeta\omega+1)x_1 + x_2 + 1 \\ -(2\zeta\omega+\omega^2)x_1 + x_3 - 2\sigma \\ -\omega^2 x_1 + 1 \end{pmatrix},$$

$$f_2(x) = \begin{pmatrix} -(2\zeta\omega+1)x_1 + x_2 - 1 \\ -(2\zeta\omega+\omega^2)x_1 + x_3 + 2\sigma \\ -\omega^2 x_1 - 1 \end{pmatrix}.$$

Here, $h_x(x) = [1 \quad 0 \quad 0]^T$, thus on $\Sigma$ (i.e. $x_1 = 0$), calculating,

$$r_1 = -(2\zeta\omega+1)x_1 + x_2 + 1 ,$$

$$r_2 = -(2\zeta\omega+1)x_1 + x_2 - 1 .$$

The sliding vector field on $\Sigma$ obtained using equations (8,9):

$$\alpha(x) = (-(2\zeta\omega+1)x_1 + x_2 + 1)/2 ,$$

$$f_F(x) = \begin{pmatrix} 0 \\ b + 4(-(2\zeta\omega+1)x_1 + x_2 + 1)/2 \\ c - 2(-(2\zeta\omega+1)x_1 + x_2 + 1)/2 \end{pmatrix},$$

where $b = -(2\zeta\omega+\omega^2)x_1 + x_3 - 2\sigma$ and $c = -\omega^2 x_1 + 1$.

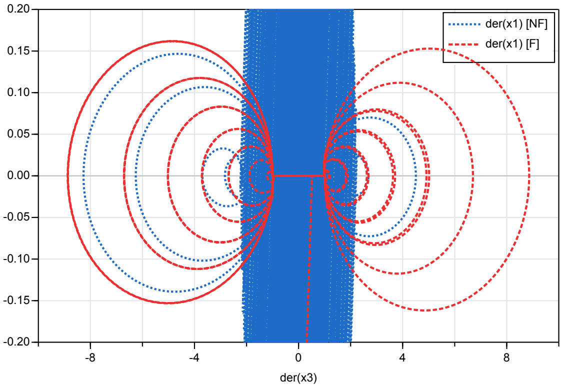


**Figure 6.** Time derivative of state variable ($\dot{x}_1$) of the relay feedback system model without (NF) and with (F) Filippov theory simulated in Dymola.

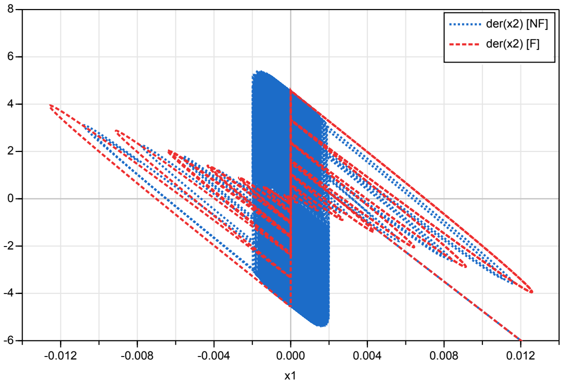


**Figure 7.** Time derivative of state variable ($\dot{x}_2$) of the relay feedback system model without (NF) and with (F) Filippov theory simulated in Dymola.

Similarly to the previous example, we have implemented both in direct form and using the proposed formulation. The parameters considered are: $\zeta = 0.05, \omega = 25$ and $\sigma = -1$. Using the Modelica language similar issues (chattering and trajectory deadlock) as in the previous example arise in OpenModelica and Dymola during time domain simulation without considering FT and the simulation results obtained using Dymola are shown in Figs. 6-7. Observe that chattering does not occur for the model implemented following the proposed approach based on FT. Finally the validation of the results of Modelica tools against the implementation in Matlab (Piiroinen and Kuznetsov, 2008) is shown in Fig. 8.

### 4.4 Example 3: Anti-windup PI controller

The IEEE Standard 421.5-2016 recommends an anti-windup (AW) or non-windup PI controller (IEEE, 2016) model for dynamic analysis of power systems. Mathematically, the model is:

$$\begin{aligned} &\text{If } y \geq w^{\max} : w = w^{\max} \text{ and } \dot{x} = 0 , \\ &\text{If } y \leq w^{\min} : w = w^{\min} \text{ and } \dot{x} = 0 , \\ &\text{Otherwise} : w = y = k_p u + x \text{ and } \dot{x} = k_i u . \end{aligned} \tag{14}$$

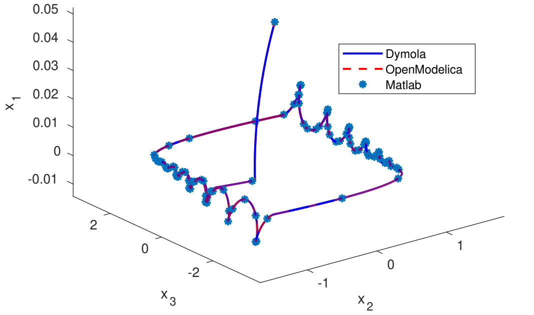


**Figure 8.** State space response for the relay feedback system obtained in different simulation software tools.

where $k_p$, $k_i$, are the proportional, integral gains of the controller, respectively; $x$ is the controller's state variable; and $u$, $y$ and $w$ are the control input, unconstrained output and constrained output respectively.

The discontinuous nature of the integral state variable of this model can lead to numerical issues such as trajectory deadlock and chattering. Only *ad hoc* approaches have been proposed to handle such issues (Hiskens, 2012; Fabozzi et al., 2017). In the authors preliminary results, it was shown that the formalism given by Filippov can effectively remove those numerical issues (Murad et al., 2019). In this work, the AW PI controller is utilized in a power system voltage control application namely an Automatic Voltage Controller (AVR), which is implemented using the proposed formulation in Modelica. In addition, the dynamic response is compared to a deadband (DB) based technique proposed in (Hiskens, 2012) and it is shown that the proposed approach achieves a smooth transient response.

### 4.4.1 Single machine infinite bus system

Consider a simple three bus power system network with a single machine shown in Fig. 9. The generator in bus 1 is equipped with an AW PI controller based AVR and a power system stabilizer as depicted in Fig. 10. The dynamics of this Single Machine Infinite Bus (SMIB) system is described by a set of DAEs in the following form (Milano, 2010),

$$\begin{aligned} \dot{x} &= f(x,y) \ , \\ 0 &= g(x,y) \ , \end{aligned} \tag{15}$$

where $x$ and $y$ are the vector of state and algebraic variables respectively.

Then generator in Fig. 9 (Gen) is modelled using a third order model (Milano, 2010), and the switching manifold for the maximum of the AVR, $h(x) = k_p v_a + x_i - v^{\max}$. When $h(x) < 0$, the differential equations of the SMIB system are:

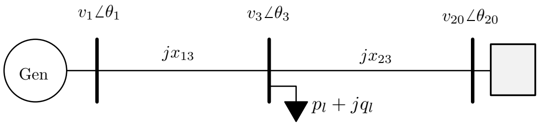


**Figure 9.** A single generator connected to an infinite bus.

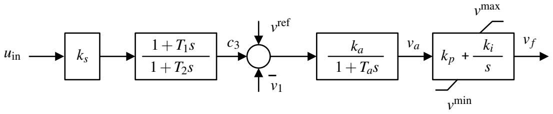


**Figure 10.** Scheme of AVR and power system stabilizer.

$$\dot{\delta} = \omega \tag{16}$$

$$\dot{\omega} = \frac{1}{M}(p_m - p_e - D\omega) \tag{17}$$

$$\dot{e}'_q = \frac{1}{T'_{d0}}(v_f - \frac{x_d}{x'_d}e'_q + \frac{x_d - x'_d}{x'_d}v_1\cos(\delta - \theta_1)) \tag{18}$$

$$\dot{v}_a = (k_a(v^{\text{ref}} + c_3 - v_1) - v_a)/T_a \tag{19}$$

$$\dot{x}_i = k_i v_a \tag{20}$$

$$\dot{s}_1 = \frac{1}{T_2}(c_2 - s_1) \ . \tag{21}$$

The notations and algebraic equations are given in (Murad et al., 2019). When the $v_f$ reaches to its max ($v^{\max}$) then (18) and (20) will be switched and all other state variables will remain same, as follows:

$$\dot{e}'_q = \frac{1}{T'_{d0}}(v^{\max} - \frac{x_d}{x'_d}e'_q + \frac{x_d - x'_d}{x'_d}v_1\cos(\delta - \theta_1)) \tag{22}$$

$$\dot{x}_i = 0 \tag{23}$$

### 4.4.2 Implementation using Filippov theory

We consider $f_1(x,y)$ is (16)-(21) and $f_2(x,y)$ is (16), (17), (22), (19), (23) and (21). Calculating, $h_x(x) = [\frac{\partial h(x)}{\partial x_1} \ \frac{\partial h(x)}{\partial x_2} \ \ldots \ \frac{\partial h(x)}{\partial x_6}]^T = [0 \ 0 \ 0 \ k_p \ 1 \ 0]^T$, and the normal to the switching surface is: $n^T(x) = [0\,0\,0\,k_p\,1\,0]$. On the switching manifold, calculating

$$\begin{aligned} h_x^T(x)f_1(x,y) &= k_p((k_a(v^{\text{ref}} + c_3 - v_1) - v_a)/T_a\ ) + k_i v_a \ , \\ h_x^T(x)f_2(x,y) &= k_p((k_a(v^{\text{ref}} + c_3 - v_1) - v_a)/T_a\ ) \ . \end{aligned}$$

Therefore according to (6), if an attractive sliding occurs on $\Sigma$, then using equation (8) $\alpha(x,y)$ is given by:

$$\alpha(x,y) = \frac{k_p((k_a(v^{\text{ref}} + c_3 - v_1) - v_a)/T_a\ ) + k_i v_a}{k_i v_a} \ .$$

According to (9), during the sliding requires (23):

$$f_F(x,y) = -k_p((k_a(v^{\text{ref}} + c_3 - v_1) - v_a)/T_a\ ) \ .$$

These expressions are used in the Modelica implementation.

**Table 1.** Parameters of the SMIB network

| Name | Values |
|---|---|
| Generator | $M=8$, $D=0$, $x'_d=0.25$, $x_d=1$,<br>$p_m=1$, $T'_{d0}=6$ |
| Line | $x_{13}=0.3$, $x_{23}=0.5$ |
| Load | $p_{l0}=0.7$, $q_{l0}=0.01$ |
| AVR | $k_a=2$, $T_a=0.005$, $k_p=5.5$, $k_i=35$,<br>$v^{\max}=1.6$, $v^{\min}=-1.5$, $v^{\text{ref}}=1$ |
| PSS | $k_s=1.5$, $T_1=0.23$, $T_2=0.12$ |

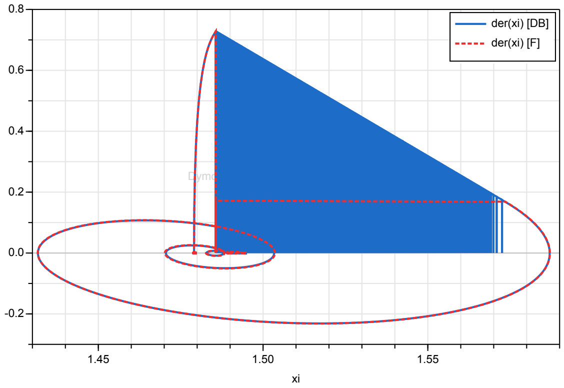


**Figure 11.** Time derivative of the integrator state variable ($\dot{x}_i$) in the AW PI controller with respect to the state variable ($\dot{x}_i$) using DB and Filippov (F) methods simulated in Dymola.

### 4.4.3 Simulation Results

The SMIB system is implemented considering the FT-based formulation and the DB based method in Modelica. The DB implementation is the same as in (Hiskens, 2012) and DB value used is 0.0001. The parameters of different components of the SMIB system are given in Table 1 and initial values for all variables are given in (Murad et al., 2019).

The SMIB system is simulated by increasing the voltage reference set-point ($v^{\text{ref}}=1.01$) and load ($p_{l0}=0.71$ pu, $q_{l0}=0.016$ pu) at $t=5$ s. Figs. 11 and 12 show the response of the time derivative of integrator state variable ($\dot{x}_i$) and field voltage ($v_f$) for both DB and FT based method's respectively. Following the disturbance, the integrator state variable ($x_i$) enters into a deadlock region and using the DB based implementation it shows chattering. Therefore the field voltage ($v_f$) shows numerous switching bounded by the DB (see zoom in Fig. 12). However, a smooth response is achieved using FT based method. In addition, except for the chattering the FT based model shows the same trajectories. It is important to mention that without DB or FT based techniques simulation in OpenModelica fails for all solvers while DASSL fails for Dymola.

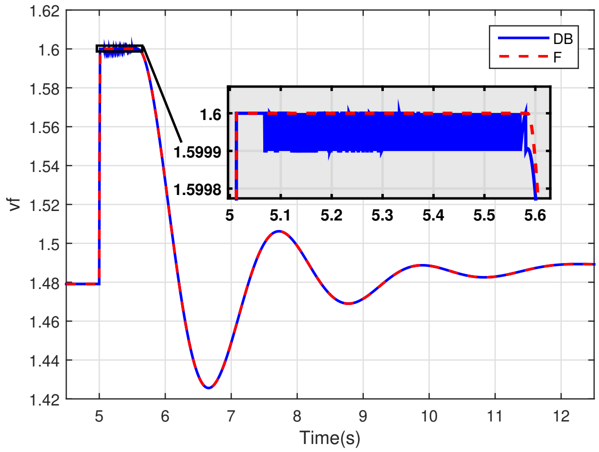


**Figure 12.** Trajectories of the field voltage ($v_f$) using DB and Filippov (F) methods simulated in Dymola.

# 5 Conclusions

A generic formulation to implement Filippov system models with sliding motion using Modelica is proposed. Three examples are presented considering such a general-purpose design with a single sliding surface. Simulation results in different Modelica tools indicate accurate dynamic response without any chattering or simulation halt.

Future work will extend the FT-based design for multiple discontinuity surface (Piiroinen and Kuznetsov, 2008). In addition a numerical performance of FT-based AW PI controller will be investigated considering the Modelica power system library: OpenIPSL (Vanfretti et al., 2016).

# 6 Acknowledgment

Mohammed Ahsan Adib Murad and Federico Milano are supported by the Science Foundation Ireland, under Investigator Programmed Grant No. SFI/15/IA/3074.

Luigi Vanfretti is supported in part by Dominion Energy, in part by the New York State Energy Research and Development Agency (NYSERDA) through the Electric Power Transmission and Distribution (EPTD) High Performing Grid Program under agreement number 137940, the Engineering Research Center Program of the National Science Foundation and the Department of Energy under Award EEC-1041877, in part by the CURENT Industry Partnership Program, and in part by the ECSE Department at RPI.

# References

Ayman Aljarbouh and Benoit Caillaud. Chattering-free simulation of hybrid dynamical systems with the function mock-up interface 2.0. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan*, number 124, pages 95–105. Linköping University Electronic Press, 2016.

Ayman Aljarbouh, Yingfu Zeng, Adam Duracz, Benoit Caillaud, and Walid Taha. Chattering-free simulation for hybrid dynamical systems semantics and prototype implementation.

In *2016 IEEE Intl. Conf. on Comp. Science and Eng. (CSE) and IEEE Intl Conf. on Embedded and Ubiquitous Computing (EUC)*, pages 412–422. IEEE, 2016.

J. Bonilla, L.J. Yebra, and S. Dormido. A heuristic method to minimise the chattering problem in dynamic mathematical two-phase flow models. *Mathematical and Computer Modelling*, 54(5):1549 – 1560, 2011. ISSN 0895-7177.

Luca Dieci and Luciano Lopez. Sliding motion in Filippov differential systems: Theoretical results and a computational approach. *SIAM Journal on Numerical Analysis*, 47(3):2023–2051, 2009.

Luca Dieci and Luciano Lopez. A survey of numerical methods for IVPs of ODEs with discontinuous right-hand side. *Journal of Computational and Applied Mathematics*, 236(16): 3967 – 3991, 2012. ISSN 0377-0427.

Davide Fabozzi, Stefan Weigel, Bernd Weise, and Fortunato Villella. Semi-implicit formulation of proportional-integral controller block with non-windup limiter according to IEEE Standard 421.5-2016. In *Bulk Power Systems Dynamics and Control Symposium (IREP)*, pages 1–7, 2017.

A. F. Filippov. *Differential Equations with Discontinuous Right-hand Sides*. Kluwer Academic Publishers, 1988.

FMI. Functional mockup interface, [Online]. URL `https://fmi-standard.org/`.

Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach*. John Wiley & Sons, 2014.

I. A. Hiskens. Dynamics of type-3 wind turbine generator models. *IEEE Transactions on Power Systems*, 27(1):465–474, Feb 2012. ISSN 0885-8950.

IEEE. IEEE recommended practice for excitation system models for power system stability studies. *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005)*, pages 1–207, Aug 2016.

Remco I Leine and Henk Nijmeijer. *Dynamics and bifurcations of non-smooth mechanical systems*, volume 18. Springer Science & Business Media, 2013.

Arie Levant. Chattering analysis. *IEEE transactions on automatic control*, 55(6):1380–1389, 2010.

F. Milano. *Power System Modelling and Scripting*. Power Systems. Springer Berlin Heidelberg, 2010. ISBN 9783642136689.

M. A. A. Murad, B. Hayes, and F. Milano. Application of Filippov theory to the IEEE standard 421.5-2016 anti-windup PI controller. In *2019 IEEE Milan PowerTech*, pages 1–6, June 2019.

Martin Otter, Hilding Elmqvist, and Sven Erik Mattsson. Hybrid modeling in modelica based on the synchronous data flow principle. In *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design (Cat. No. 99TH8404)*, pages 151–157. IEEE, 1999.

Friedrich Pfeiffer and Christoph Glocker. *Multibody dynamics with unilateral contacts*. John Wiley & Sons, 1996.

Petri T. Piiroinen and Yuri A. Kuznetsov. An event-driven method to simulate Filippov systems with accurate computing of sliding motions. *ACM Trans. Math. Softw.*, 34(3):13:1–13:24, May 2008. ISSN 0098-3500.

Damian Suski and Radosław Pytlak. Simulation of hybrid systems with sliding modes. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 419–424. IEEE, 2017.

Walid Taha, Adam Duracz, Yingfu Zeng, Kevin Atkinson, Ferenc A Bartha, Paul Brauner, Jan Duracz, Fei Xu, Robert Cartwright, Michal Konečnỳ, et al. Acumen: An open-source testbed for cyber-physical systems research. In *International Internet of Things Summit*, pages 118–130. Springer, 2015.

L. Vanfretti, T. Rabuzin, M. Baudette, and M. Murad. itesla power systems library (iPSL): A Modelica library for phasor time-domain simulations. *SoftwareX*, 5:84 – 88, 2016. ISSN 2352-7110.

# Modeling Contact and Collisions for Robotic Assembly Control

Scott A. Bortoff[1]

[1]Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, bortoff@merl.com

## Abstract

We propose an implicit, event-driven, penalty-based method for modeling rigid body contact and collision that is useful for design and analysis of control algorithms for precision robotic assembly tasks. The method is based on Baumgarte's method of differential algebraic equation index reduction in which we modify the conventional constraint stabilization to model object collision, define a finite state machine to model transition between contact and non-contact states, and represent the robot and task object dynamics as a single set of differential algebraic inequalities. The method, which is realized natively in Modelica, has some advantages over conventional penalty-based methods: The resulting system is not numerically stiff after the collision transient, it enforces constraints for object penetration, and it allows for dynamic analysis of the Modelica model beyond time-domain simulation. We provide three examples: A bouncing ball, a ball maze, and a delta robot controlled to achieve soft collision and maintain soft contact with an object in its environment.

*Keywords: robotics, control*

## 1 Introduction

To derive new control algorithms for robust robot assembly, it is important to construct system-level dynamic models of the robotic manipulator *and* the assembly task, including appropriate representations of object contact and collision *and also* the control algorithms themselves. Of course, Modelica is well-suited for all three of these domains. It is also important that such models should be useful for more than time-domain simulation. For example, we should be able to linearize the full model at various operating conditions, and to compute things such as a multi-variable frequency response for rigorous design and analysis of new types of feedback control algorithms. We also should be able to compute some model objects for purposes of real time model-based estimation and control.

Our long-term objectives are (1) to invent new control algorithms, especially for the delta robot, that make assembly processes robust with respect to uncertainty in the environment (especially uncertainty in the location of an object), and exploit new types of sensors, such as touch sensors, (2) to accelerate the process of experimental validation, and (3) to accelerate and simplify the process of controller parameter tuning that is done at commissioning time. Toward these ends, we not only seek to model and simulate robotic assembly, but we also desire a mathematical abstraction of robotic assembly that is consistent with the Modelica software representation, and is useful for synthesis and analysis of robust, hybrid feedback control algorithms. In addition, we intend to use our developing Modelica library of robot manipulators, assembly tasks and control algorithms together with the Modelica Device Drivers library (Thiele et al., 2017) to control the delta robot in our laboratory in various assembly experiments, without having to recode any aspect of the control algorithm, although this work is beyond our scope here.

In this paper, we propose an event-driven model of rigid body collision and contact that combines differential algebraic inequalities to represent rigid body motion with a Finite State Machine (FSM) to represent the state of contact and collisions. Rigid bodies in contact may be modeled as a set of differential equations coupled via a vector of Lagrange multipliers $\lambda$ to a set of algebraic equations that represent the contact, resulting in an index-3 Differential Algebraic Equation (DAE). Baumgarte's method of index reduction (Baumgarte, 1972, 1983) replaces the algebraic constraint equations with a linear combination of their derivatives with respect to time, resulting in an index-1 (or 0) set of DAEs (or ODEs). The contribution of this paper is to allow the structure of this DAE to change dynamically from an unconstrained state ($\lambda = 0$) to a constrained state ($\lambda > 0$) in order to model the physical process of collision and subsequent contact (and vis versa), and to observe that the dynamics introduced by Baumgarte's method can in fact model the transient associated with the collision. We introduce the FSM to represent the discrete state of contact, resulting in a hybrid DAE capturing collision, contact, and loss of contact. This mathematics can be represented in the Modelica language natively, and is consistent with other common models of robotic manipulators, allowing for integrated modeling, simulation and analysis of the robotic system and task.

The paper is organized as follows. In Section 2 we propose the contact and collision model in general terms, list its properties, advantages and disadvantages compared to other state-of-the-art methods, and provide a simple bouncing ball example. In Section 3 we present a second example: A ball maze. We design a feedback control algorithm to solve the maze and simulate the closed-loop system, which exhibits a sequence of collisions, using the proposed model. Finally, we show initial results for a delta robot assembly task (Bortoff, 2018, 2019), where control must achieve soft collision between the manipulator and a workspace object. Conclusions are drawn in Section 5.

10.3384/ECP2016954 DOI

## 2 Collision and Contact Model

Modeling collisions and contact among solid objects is a well-studied subject and we refer the reader to the vast literature on the subject, including works associated with the computer graphics community (Erleben et al., 2005). Collision models have been developed for the Modelica MultiBody library, notably (Engelson, 2000; Otter et al., 2005; Hofmann et al., 2011a), resulting in the third-party IdealizedContact library (Hofmann et al., 2011b). These references discuss several methods of contact detection and reaction, including impulsive methods and penalty-based methods. Several works have integrated Modelica with third-party software such as the Bullet Physics Library[1] or Gazebo[2], especially for collision detection but also to exploit their animation capability and potentially tap the large collection of robotic technologies represented in these tools such as advanced sensors (Bardaro et al., 2017). However, the collision detection algorithms used by these third party tools are intended primarily for animation, not dynamic analysis, and have limitations discussed in the literature e.g., requiring a non-zero collision margin or having difficulty with very large and very small shapes (Coumans, 2015).

In this paper we propose an event-triggered, implicit penalty (force) - based collision approach that is native to Modelica, and has some advantages (Pros) over conventional penalty - based methods:

**P1** It does not result in a stiff set of ODEs, even for stiff material properties,

**P2** It results in steady-state solutions without object penetration, although object penetration occurs during the transient collision phase, and

**P3** It is relatively easy to understand and implement, completely in Modelica, which allows representation of the complete physics of a problem in a single tool.

On the other hand, our proposed method has three disadvantages (Cons):

**C1** It does not conserve energy before and after elastic collisions,

**C2** It is event-driven, meaning it relies on the Modelica tool's ability to properly detect events, and

**C3** For large numbers of objects, it will certainly be less memory and time efficient than alternatives.

Despite these disadvantages, the method is useful for our purpose, and even **C1** is not significant except for very limited and well-defined circumstances that are generally outside our interest.

To begin, assume that the environment includes a number of other rigid bodies, which we denote as *task objects*, whose collective generalized position and velocity are denoted $q_c$ and $v_c$ respectively. For the task objects, the unconstrained Lagrangian equations of motion are assumed to be

$$\dot{q}_c = v_c \tag{1a}$$

$$M_c(q_c)\dot{v}_c + C_c(q_c, v_c) + D_c(v_c) + G_c(q_c) = f_c, \tag{1b}$$

where $M_c$ is the inertia matrix, $C_c$ represents Coriolis and centripetal forces and torques, $G_c$ represents forces and torques due to gravity, $D_c$ represents frictional forces and torques, $f_c$ represents external forces and torques, and the subscript $c$ denotes "constraint." In addition, we assume that all geometric constraints among the task objects can be expressed as the $N_c$-dimensional vector inequality

$$h_c(q_c) \geq 0, \tag{2}$$

and that all the geometric constraints between the task objects and the robotic manipulator can be expressed as the $N_r$-dimensional vector inequality

$$h_{rc}(q, q_c) \geq 0. \tag{3}$$

With this notation, two task objects are in geometric contact if at least one of the corresponding inequalities is zero; otherwise the task objects are not in geometric contact. The functions $h_c$ and $h_{rc}$ are, generally speaking, the distance between key features of the task objects and robotic manipulator, respectively.

For example, if the task objects were three solid cubes above a rigid surface, then (1) is the 36-dimensional rigid body equations for the blocks, with each body having three translational and three rotational degrees of freedom. The matrix $M_c$ is the $18 \times 18$ block-diagonal inertia matrix, $C_c$ represents Coriolis and centripetal forces and torques, $G_c$ represents forces and torques to to gravity, and $D_c$ represents frictional forces and torques on the collection of blocks. The vector $h_c$ includes distances from each vertex to the surface, and distances between the vertices and edges of each block in order to capture the constraint that every vertex edge of every cube must lie outside the volume of all other cubes.

In practice, $N_c$ and $N_{rc}$ may be large, and grow unfavorably as the number of objects increase. Physics engines such as Bullet reduce the dimension of inequalities that must be considered in the collision detection problem by eliminating from consideration those pairs of objects that are far away, i.e., $h_{ci}(q_c) >> 0$ for some $i$, using heuristics like bounding boxes, and by computing only the *minimum* distance between pairs objects e.g. (Gilbert et al., 1988), thereby reducing the number of inequalities in the collision detection problem. These aspects are important in typical computer graphics applications. In this paper we do not concern ourselves with efficiency, and concede that (2) and (3) may be high-dimensional, although for many specific assembly problems, these are reasonably sized.

[1] https://pybullet.org

[2] http://gazebosim.org

We further assume that the unconstrained (free of contact) robot manipulator is modeled in the usual manner:

$$\dot{q} = v \tag{4a}$$

$$M(q)\dot{v} + C(q, v) + G(q) = Bu. \tag{4b}$$

We now propose that Baumgarte's method of index reduction, which stabilizes a holonomic equality constraint $h(q) = 0$ when the constraint is active, can be extended and used to model both solid object contact and also collision, if we cause the Lagrange multipliers that correspond to object-to-object or object-to-robot contact to satisfy an inequality constraint, and optionally if we modify the coefficients and the structure of the constraint stabilization to represent stiff elastic collision. There are two parts to this model: The physics equations of constrained motion, and a finite state machine (FSM) that determines the state of contact between objects.

We first present the physics equations. Define $\lambda_c \in \mathbb{R}^{N_c}$ and $\lambda_{rc} \in \mathbb{R}^{N_{rc}}$ be Lagrange multiplier vectors of the same dimension as $h_c$ and $h_{rc}$, respectively. For now, denote the state of each FSM as $c_i \in \{0, 1, 2\}$, for $1 \leq i \leq N_c + N_{rc}$, with $c_i = 1$ corresponding to the contact state. Then we can define the equations of motion for the combined robot-task system as

$$\dot{q} = v \tag{5a}$$

$$M(q)\dot{v} + C(q, v) + G(q) = \lambda^T \frac{\partial h}{\partial q} + \lambda_{rc}^T \frac{\partial h_{rc}}{\partial q} + Bu \tag{5b}$$

$$\ddot{h} + \alpha_1 \dot{h} + \alpha_0 h = 0, \tag{5c}$$

and

$$\dot{q}_c = v_c \tag{6a}$$

$$M_c(q_c)\dot{v}_c + C_c(q_c, v_c) + G_c(q_c) = \lambda_{rc}^T \frac{\partial h_{rc}}{\partial q_c} + \lambda_c^T \frac{\partial h_c}{\partial q_c} + f \tag{6b}$$

and, for $1 \leq i \leq N_c$ and $1 \leq j \leq N_{rc}$,

$$\text{If } c_i = 1 \text{ then } \ddot{h}_{ci} + \alpha_1 \dot{h}_{ci} + \alpha_0 h_{ci} + \alpha_2 h_{ci}^3 = 0 \text{ else } \lambda_{ci} = 0 \tag{7a}$$

$$\text{If } c_j = 1 \text{ then } \ddot{h}_{cj} + \alpha_1 \dot{h}_{rcj} + \alpha_0 h_{rcj} + \alpha_2 h_{rcj}^3 = 0 \text{ else } \lambda_{rcj} = 0. \tag{7b}$$

In short, if a constraint is active, i.e., if two objects are in contact, then the corresponding Lagrange multiplier is an algebraic state-variable of the index-1 DAE and the corresponding stabilizing constraint equation is active. Otherwise, the Lagrange multiplier is set to zero, and the corresponding constraint stabilizing equation does not appear. This ensures that the number of equations and variables is the same, regardless of the constraint state.

The FSM for each constraint is required for a subtle reason. It might seem that the logic for constraint activation is that the constraint $i$ (or $j$) becomes active when $h_{ci} \leq 0$ (or $h_{cri} \leq 0$), and becomes inactive when $\lambda_{ci} < 0$ (or $\lambda_{rcj} < 0$), for $1 \leq i \leq N_c$, $1 \leq j \leq N_{rc}$, giving two well-defined states. (For this point, we drop the subscripts for notational simplicity.) However, in Modelica it is not good practice to have an activation condition (realized using either `when` or `if`), depend on two different variables. Further, it can occur that $\lambda < 0$ while $h < 0$, because of numerical errors when both are near zero, or because forces due to other objects cause $\lambda < 0$ while $h < 0$. In fact, this is common in practice. We therefore define the three-state FSM diagrammed in Figure 1, which ensures correct transition to and from contact. The Ballistic state is included for the case that $h < 0$ and $\lambda < 0$, to ensure that the constraint force can not be negative and pull objects together. Physically it must always be repulsive i.e., positive.

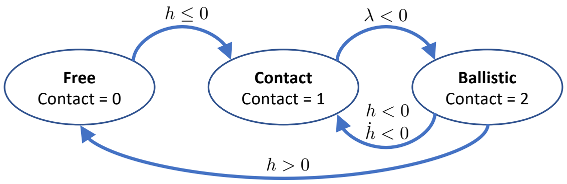


**Figure 1.** Finite State Machine (FSM) for contact.

Several remarks are in order. Note that the Lagrange multipliers have the physical interpretation as the force between two objects required to drive the constraint to zero, according to the second-order stabilized constraint equation (7). Also, the penetration between objects due to collision is governed only by corresponding second-order stabilized constraint equation (7), which is independent of the other dynamical equations by design. The parameters $\alpha_0$, $\alpha_1$ and $\alpha_2$ can be tuned for the specific material stiffness and damping properties. Importantly, we have added a nonlinear cubic term in order to capture nonlinear force behavior due to penetration (Hofmann et al., 2011a). This can be any odd-order polynomial or similar function, as long as (7) is stable.

This formulation has three advantages over more conventional penalty-type methods that explicitly compute a force between objects in contact as a function of penetration. First, the resulting dynamics (5)-(7) are not stiff after the collision transient due to (7) has transpired. If we choose values of $\alpha_0$ and $\alpha_1$ that are of the same time-scale as the rigid dynamics, while $\alpha_3$ may be large to account for stiff material properties, then we see that the Lagrange multiplier is precisely the force that will drive the constraint to zero, at which point the eigenvalues correspond to roots of $s^2 + \alpha_1 s + \alpha_0$, which by design is not stiff. In other words, we can model collisions between very stiff objects by making $\alpha_3 >> 0$, and after the transient from the collision transpires, assuming that the objects remain in contact, the dynamics will have eigenvalues at the roots of $s^2 + \alpha_1 s + \alpha_0$, so the system need not be stiff. Of course, the system will be stiff during the collision transient, necessitating small simulation time steps during this phase, but a variable step solver will be able to increase step size after contact is established between objects, and the transient due to (7) has transpired.

The second advantage is that the stabilized constraint equations (7) drive the constraint to zero when the constraint remains active. This means that, after the transient due to that specific collision transpires, the constraint $h = 0$ is exactly enforced and there is no penetration, aside from numerical error which is of the solver tolerance. The constraint $h = 0$ will be enforced even if the dynamic system continues to evolve: It does not need to be in steady-state. On the other hand, if we were to compute the force between objects using a "spring-damper" model, for example, then there would be nonzero penetration after the collision, it would remain nonzero due to the dynamic behavior of the rest of the system, even in steady-state. The penetration can be made small, but at the expense of using a stiff virtual spring, making the ODE stiff.

A bouncing ball is a good example, with Modelica code as follows.

```
model myBouncingBall

Real q(start=1.0), v(start=0.0), f;
Real h, hDot, hDotDot, lambda(start=0);
discrete Integer contact(start=0);
Boolean b1, b2, b3, b4;
parameter Real g=9.81, m=1.0;
parameter Real a0=100, a1=20, a2=1e6;
parameter Boolean linFlag = false;

algorithm
b1 := h <= 0;
b2 := lambda < 0.0;
b3 := h > 0.0;
b4 := h <= 0 and hDot < 0;
if edge(b1) and contact == 0 then
  contact := 1;
end if;
if contact == 1 and edge(b2) then
  contact := 2;
end if;
if contact == 2 and edge(b3) then
  contact := 0;
end if;
if contact == 2 and edge(b4) then
  contact := 1;
end if;

equation
if contact == 1 or linFlag then
  0 = hDotDot + a1*hDot + a0*h + a2*h^3;
else
  lambda = 0.0;
end if;

f = if linFlag then lambda
     else contact*lambda;

der(q) = v;
m * der(v) = -m * g + f;

h = q;
hDot = der(h);
hDotDot = der(hDot);

end myBouncingBall;
```

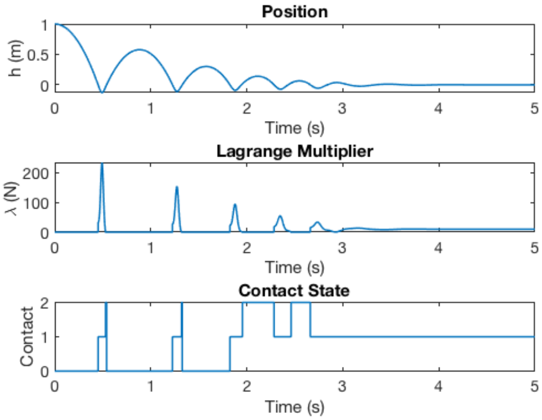


**Figure 2.** Bouncing ball simulation for stiff materials and low damping ($\alpha_1 = 100$, $\alpha_1 = 1$, $\alpha_2 = 1e6$, ). Note that the contact state passes through the "Ballistic" state, `contact = 2`, because $\lambda$ changes sign, and remains in that state if either $h > 0$ when $\lambda$ changed sign, or $h < 0$ and $\dot{h} > 0$ when $\lambda$ changed sign. After four bounces, the ball comes to rest, and the constraint $h \rightarrow 0$, and the Lagrange multiplier $\lambda > 0$.

The `algorithm` section computes the FSM. If its state is `contact == 1` then the Lagrange multiplier is active, and we include the stabilizing constraint equation with a stiff spring model of contact. If inactive, then we explicitly set `lambda = 0`, in order to balance the number of equations and states. We have included the flag `linFlag` in order to compute linearization when the constraint is active. It is necessary because Dymola will otherwise compute a mathematically incorrect linearization at the final time of simulation when the constraint is active, due to its numerical algorithm.

Figure 2 shows a Dymola simulation with a small positive value of damping. This is a typical situation in which this method is useful, although our typical uses have more damping than this. However, Figure 3 shows the situation when the damping is zero, ($\alpha_1 = 0$). Here we see the ball height increases over time, which means the model is adding energy to the system, which is not physical. This is for two reasons. First, the Lagrange multiplier, which is the reaction force due to collision, includes the steady-state gravity force $mg$ when the ball is in contact,

$$\lambda(t) = (\alpha_0 q(t) + \alpha_2 q^3(t) + g)m.$$

If we were to implement a mass-spring type of force during collision, this term would be absent and the simulation would conserve energy within the solver tolerance. So, during each contact our method adds a little extra energy that is not physical. This is a clear disadvantage of the method. The second reason that the energy increases is that the constraint is active beyond the point in time that $h$ crosses zero, by design of the FSM, which switches when $\lambda$ crosses zero, not when $h$ crosses zero. This is apparent in Figure 4. The reason the FSM is designed in this man-

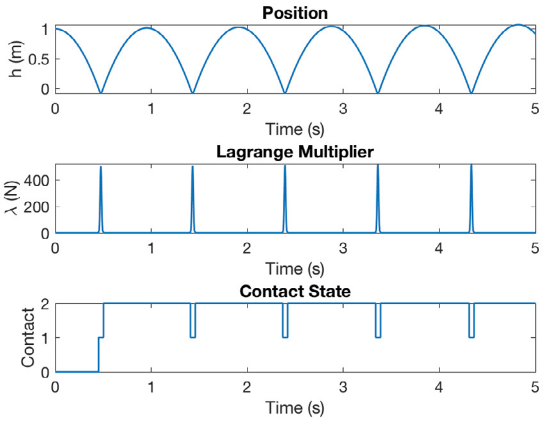


**Figure 3.** Bouncing ball simulation for stiff materials and zero damping ($\alpha_1 = 100$, $\alpha_1 = 0$, $\alpha_2 = 1e6$), showing the ball height growing over time.

ner is that the constraint equations cause $h \to 0$, and so small numerical errors in $h$ when it is near zero but slightly positive would cause a switch from contact to non-contact, even if $\lambda > 0$. This would cause a kind of undesirable and actually unphysical chatter in the system simulation. We prefer to switch when the constraint force changes sign, so that objects in contact will remain in contact as long as their contact force is positive.

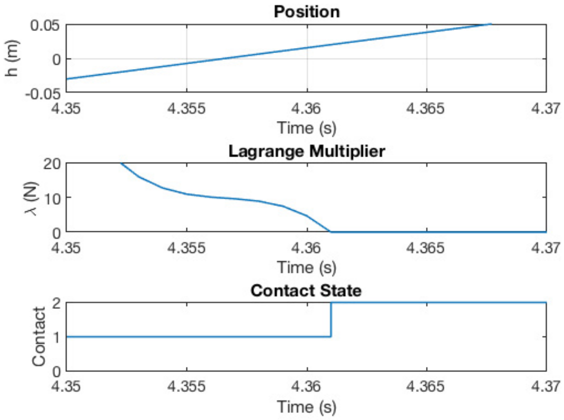


**Figure 4.** Close-up of the transition between contact and non-contact in Figure 3, which occurs when $\lambda$ changes sign near $t = 4.262$s. Note that when $h$ crosses zero, near $t = 4.356$s, $\lambda = 9.81$, so additional positive force is applied to the ball for $4.262 < t < 4.356$, adding energy to the ball.

We emphasize that the situation when $\alpha_1 = 0$ is not representative of a typical use of this method, and discussed here only to be clear about the method's limitations. Indeed, in this case, the constraint is not stabilized. Most real problems involving assembly have significant damping, and in these cases the energy increase due to the method is negligible.

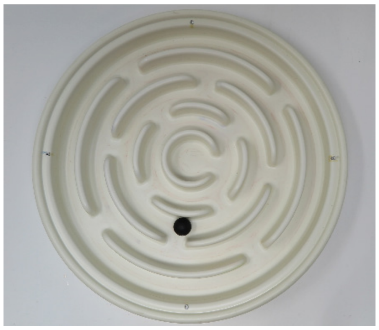


**Figure 5.** Rolling ball maze toy (van Baar et al., 2019).

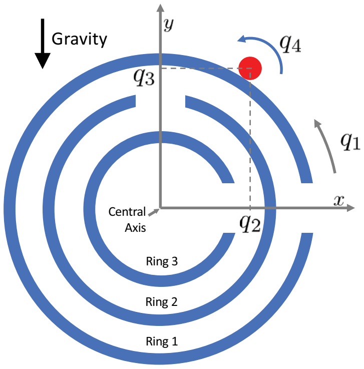


**Figure 6.** Ball maze diagram.

# 3 Ball Maze Example

A ball maze exhibits contact and non-contact physics that is effectively modeled using the proposed approach. Referring to Figure 5, the objective of the game is to manipulate the maze orientation in a way to cause the ball to roll into the center ring. The maze has been used to demonstrate learning-type control (van Baar et al., 2019), and typically the manipulation is "tip-tilt" in nature, causing the ball to roll due to gravity. The objective is challenging because after the ball passes through a gate, it collides with a maze ring and once in contact, the configuration is unstable, so the ball will roll one way or another, frustrating the player.

Here we will solve the maze using an alternative approach. Instead of tipping and tilting the maze, we stand it on end, so that the central axis (of symmetry) is orthogonal to the gravity vector, and we rotate the maze about the central axis, so that the maze itself has only one degree of freedom, as shown in Figure 6. The strategy is to use feedback to stabilize the ball when it is in contact with a ring, and then rotate the closed-loop maze system so that the ball falls through a sequence of gates, eventually reaching the center. Therefore we need a model that is appropriate for control system design, and also is appropriate for the simulation of the ball motion as it comes into and out of contact with the sequence of rings.

The ball maze is modeled as

$$M\ddot{q}+G=H^T(q,i)\lambda+Bu \tag{8a}$$
$$\ddot{h}(q,\dot{q},\lambda,i)+\alpha_1\dot{h}(q,\dot{q},i)+\alpha_0 h(q,i)=0 \tag{8b}$$

where $q\in\mathbb{R}^4$, $q_1$ is the angular displacement of the maze, $q_2$ and $q_3$ are the Cartesian coordinates of the ball center-of-mass, $q_4$ is the angular displacement of the ball relative to the base frame,

$$M=\mathrm{diag}(J_m,m_b,m_b,J_b)\,, \tag{9a}$$
$$G=[0\;0\;gm_b\;0]^T, \tag{9b}$$
$$B=[1\;0\;0\;0]^T, \tag{9c}$$
$$H(q,i)=\frac{\partial h(q,i)}{\partial q}\,, \tag{9d}$$

$J_m$ and $J_b$ are the rotational inertia of the maze and ball, respectively, $m_b$ is the ball mass, $g$ is the acceleration due to gravity, $u$ is an input torque, and $\lambda\in\mathbb{R}^2$ is the Lagrange multiplier vector. When the ball is in contact with maze ring $i$ of radius $r_{mi}$, $1\le i\le 4$, the two holonomic constraints

$$h_1(q,i)=q_1^2+q_2^2+(r_{mi}+r_b)^2 \tag{10a}$$
$$\begin{aligned}h_2(q,i)&=r_b(q_4-q_{40})+r_{mi}(q_1-q_{10})\\&\quad+r_{mi}\left(\mathrm{atan}(q_2/q_3)-\mathrm{atan}(q_{20}/q_{30})\right)\end{aligned} \tag{10b}$$

are active so that $\lambda_1$ and $\lambda_2$ are time-varying algebraic states. In (10), $h_1$ is the distance constraint, $h_2$ is the rolling constraint, $r_b$ is the ball radius, and $q_0=[q_{10}\;q_{20}\;q_{30}\;q_{40}]^T$ is $q(t)$ at the time of collision $t=t_0$ with ring $i$. When the ball is not in contact with any rings, i.e., it is passing through a gate, then $\lambda_1=\lambda_2=0$.

We design a stabilizing feedback controller for the case when the ball and ring $i$ are in contact using the root locus method, assuming the ball position along the $x$-axis, $y_1=q_2$, and the maze orientation, $y_2=q_1$, are measured outputs, and splitting the input as $u=u_1+u_2$. Model (8) and (10) is coded in Modelica, from which we compute a linearization at the open-loop unstable equilibria $q_2=0$, and a pole-zero plot of the system from $u_1$ to $y_1$. This shows four pole-zero cancellations at $s=-5$, corresponding to the stabilized constraint (8b) with $\alpha_1=10$ and $\alpha_0=25$, two pole-zero cancellations at the origin, one pole at $s=-1.1$, and one unstable pole at $s=1.4$. This system can be stabilized using two lead-type compensators, one for each output. The first stabilizes the ball position $y_1=q_2$, with zero at $s=-0.6$, pole at $s=-10$ and positive feedback gain, as shown in the root locus in Figure 7 (top). This moves the unstable pole into the left half plane, but leaves the two poles at the origin. After closing this loop, we next compute the pole-zero map for the system from $u_2$ to $y_2$. This shows the two poles at the origin and a non-minimum phase zero at $s=0.65$. This can also be stabilized with a lead compensator with zero at $s=-0.05$, pole at $s=-5.0$, and gain $k_2=0.035$, as shown by the root locus in Figure 7 (bottom). The second loop has an upper limit on its gain, due to the non-minimum phase zero, and has a slower response, while the first loop has a lower limit on its gain, due to the unstable pole, and has a faster response. The closed-loop system is realized in Modelica as shown in Figure 8. Note that a single set of controller gains is effective for any of the rings $1\le i\le 4$, but this must be checked by computing a linearization for each ring radius.

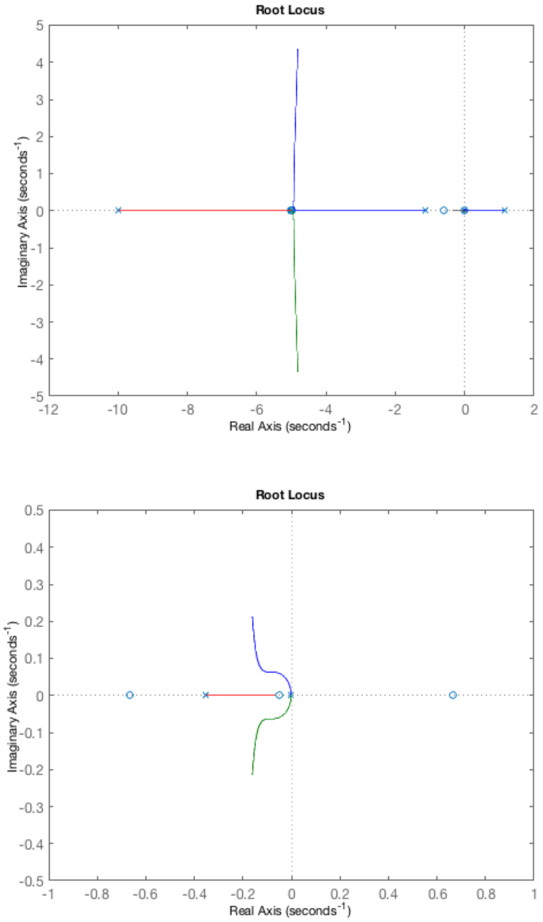


**Figure 7.** Root locus showing a lead compensator stabilizing the ball position $y_1=q_2$ (top), and lead compensator stabilizing the maze rotation $y_2=q_1$ (bottom).

To simulate the ball maze, we require a FSM to switch the Lagrange multipliers from active to inactive, similar to Figure 1. However for the ball maze, we have sufficient damping to prevent bouncing, and the system will move one-way through the maze, simplifying the logic. Essentially the constraints become active when the ball contacts a ring, i.e., when $h_1$ changes sign, for the sequence of rings (note that $h_1$ depends on ring radius $i$), and become inactive i.e., the ball falls through a gate, when the rotation of the maze moves the gate under the ball. We can define the

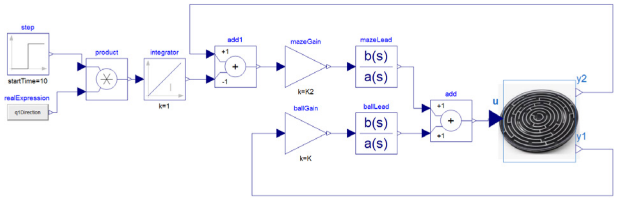


**Figure 8.** Feedback Controller for Ball Maze.

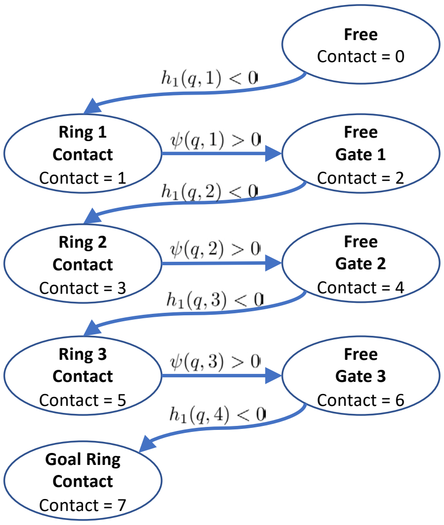


**Figure 9.** Ball maze FSM.

location of the gates by defining

$$\psi(q,i) = \begin{cases} q_1 + r_{m1}\text{atan}(q_2/q_3) - \pi/2 & \text{for} \quad i = 1 \\ -q_1 - r_{m2}\text{atan}(q_2/q_3) & \text{for} \quad i = 2 \\ q_1 + r_{m3}\text{atan}(q_2/q_3) - \pi/2 & \text{for} \quad i = 3 \\ -q_1 - r_{m4}\text{atan}(q_2/q_3) & \text{for} \quad i = 4 \end{cases} \tag{11}$$

so that $\psi(q,i)$ changes sign as the ball moves "over" open gate $i$. Then the FSM is as shown in Figure 9, and is straightforward to implement as a Modelica algorithm.

Figure 10 shows results of a Dymola simulation of the closed-loop system. The ball is initialized in the free state, and falls toward the outer ring, making contact at about $t = 0.1s$. The maze turns counterclockwise beginning at $t = 10s$ and rotates until the ball falls through gate 1. As it falls through the gate, in the contact $= 2$ state, $\lambda = 0$ and the ball moves freely, until it collides with ring 2 at about $t = 27$s. The collision causes the large spike in $\lambda$, representing the elastic collision, after which the contact $= 3$ state is maintained as the maze is rotated clockwise until gate 2 is below the ball. Thus, as the maze rotates, the system switches between the constrained contact state and an unconstrained free state, although the number of dynamic states remains constant (8) throughout. The ball maze continues to rotate until the ball gets to the inner ring, at which point the controller is turned off.

Figure 11 shows a sequences of screen captures from a Dymola animation of the closed-loop system. Note that we show only three rings here for simplicity. Space constraints prohibit a full listing of the Modelica code, which is available from the conference website or by contacting the author directly. Note that it would be simpler to rotate the maze in a single direction. However, changing the direction of $\dot{q}_1$ allows for the maze to retain previously collected balls in the center. It also shows the slower and non-minimum phase response of the maze angle $q_1$ to the reference.

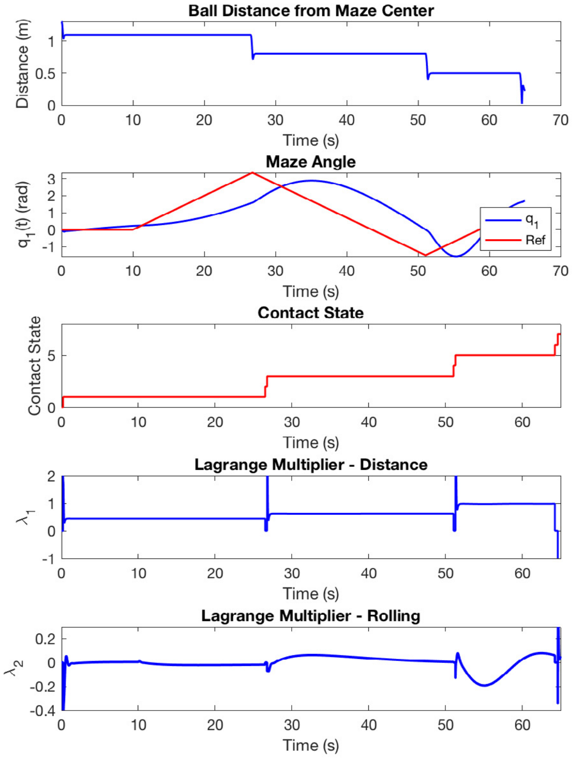


**Figure 10.** Ball maze simulation. Note the brevity of the free-motion states, contact $= 2,4,6$, when the ball passes through a gate, and when $\lambda = 0$.

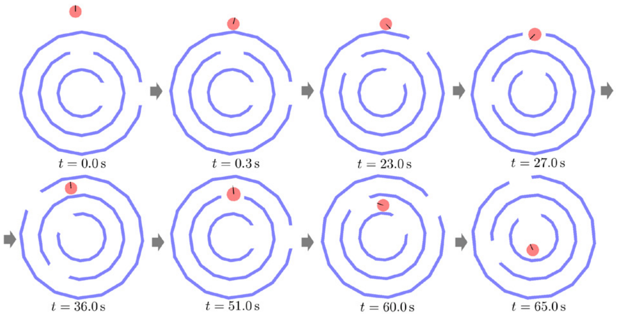


**Figure 11.** Sequence of ball maze configurations as the closed-loop system drives the ball toward the goal.

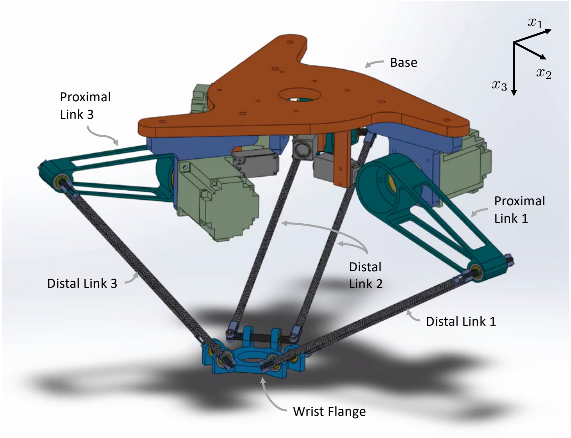


**Figure 12.** Delta robot.

## 4 Delta Robot Soft-Touch Control

The delta robot (Clavel, 1990) shown in Figure 12 consists of three (or more) identical under-actuated arms, arranged symmetrically about the $x_3$-axis (pointing down). Each arm consists of a proximal link, rigidly attached to the servomotor shaft at the base, and a pair of parallel distal links that are attached to the proximal link by universal joints. The six distal links are in turn attached to the wrist flange by universal joints, so that the two distal links associated with each arm remain parallel during operation. The configuration provides three translation DOFs of the wrist flange within the robot's reachable workspace, while the orientation of the wrist flange remains fixed. This beneficial feature of the delta robot decouples the translational kinematics and dynamics of the robot from the rotational kinematics and dynamics associated with a wrist that is mounted below the wrist flange (not shown in Figure 12). The servomotor angles are directly measured, while the universal joint angles are not measured, but can be computed from the servomotor angles via the forward kinematics.

A formulation and Modelica realization of the delta robot translational dynamics was derived previously (Bortoff, 2018, 2019) by defining the dynamics for each unconstrained arm, and then adding the holonomic coupling constraint representing the connections to the wrist flange. The resulting index-3 differential-algebraic equation (DAE) is stabilized using Baumgarte's method (Baumgarte, 1972, 1983), giving an index-1 DAE.

The objective here is to derive and simulate a soft-contact control algorithm for the delta robot. In this case the task object is a Lego brick on a surface, and the task is to pick up the brick with a gripper that is mounted to the wrist flange without sliding the brick along the surface, despite uncertainty in its location on the surface. Such a control algorithm would be used in practice to grasp very fragile objects. To accomplish this, the gripper frame is moved by the robot servos so that the left finger is close to the block surface, and then it approaches the block with low velocity and low impedance so that it does not transfer energy sufficient to cause motion of the block. For this particular simulation, the gripper servo is not used.

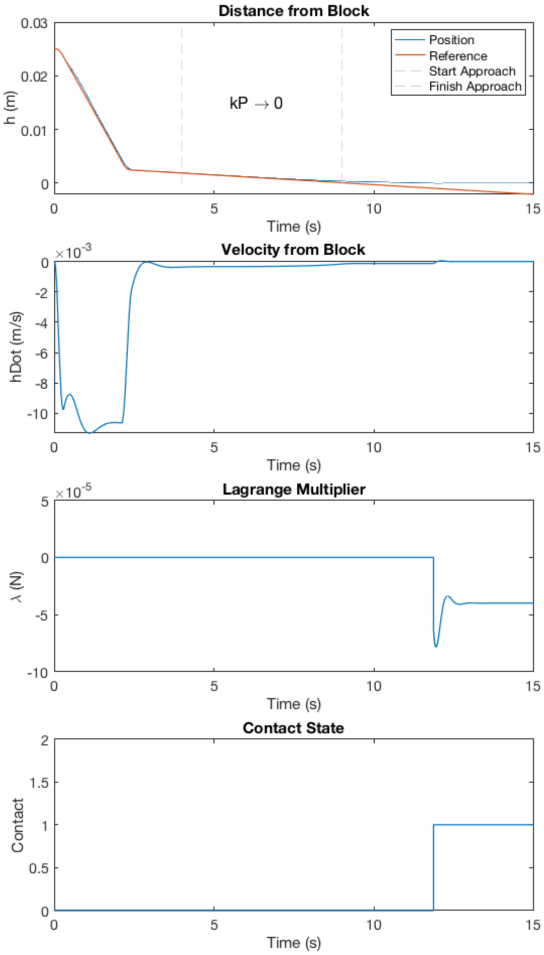


**Figure 13.** Simulation of soft collision and contact. The distance to the object (top), is commanded by a smooth reference to be within 5mm of the object in the first 2.5s. Then it approaches at low velocity of 0.2mm/s, and the position gain $k_p$ is reduced continuously to zero until $t = 9$s. At this point the robot is in velocity control mode. Contact is made at $t = 12$s, and the force of impact peaks at 90mN, below what would cause the block to move. Contact is maintained with a force of 40mN, and there is no motion of the block or bouncing.

Figure 14 is a block diagram in Dymola showing the feedback controller. The reference generator computes smooth reference trajectories for position, velocity and acceleration in task space within constraints on maximum values for velocity, acceleration and jerk, using cubic splines. In this specific case, the trajectory is such that the approach velocity is 0.2mm/s. The PDFF controller block is a digital PD controller with feedforward

$$u(k) = k_p(k)(r(k) - y(k)) + k_d(k)(\dot{r}(k) - \dot{y}(k)) + \ddot{r}(k) + f(k) \tag{12}$$

where $\dot{y}(k)$ is computed by filtering $y(k)$ to approximate the derivative, the feedback gains $k_p$ and $k_d$ are time-varying, set by a higher-level controller, and $f$ is an ex-

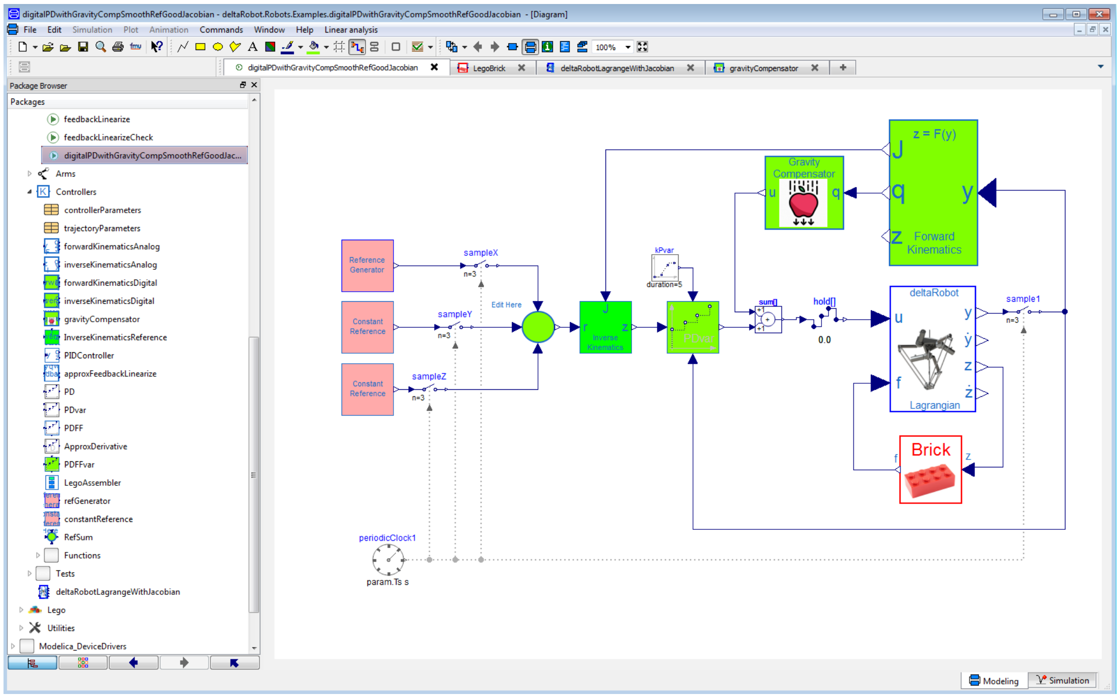


**Figure 14.** Feedback control model for soft contact. Blocks in green are discrete-time control blocks that use the Modelica Synchronous Library and compute forward and inverse kinematics, PD control with variable gains, and gravity torque compensation. The pink blocks at left compute smooth reference trajectories. The red Brick block computes the dynamics of a block in contact with a surface, which comes in contact with the delta robot end effector during simulation. The input $f$ into the delta robot model is the force applied to the end effector by the brick, which is a Lagrange multiplier internal to the Brick model. Our deltaRobot library is open at left, showing some of the control system components.

ternal input from the left touch sensor, which is not used here. The Forward and Inverse Kinematics blocks, and the Gravity Compensator block, compute the forward and inverse kinematics, and also a torque to cancel the effect of gravity on the robot. These functions cannot be computed analytically for the delta robot, and use Newton's method to solve a set of implicit functions, described in (Bortoff, 2018).

To achieve soft contact, the $k_p$ gain in the direction of travel is continuously reduced to zero as it approaches the brick, putting the robot effectively in velocity control mode and reducing its impedance at the moment of contact. Closed-loop stability is maintained if

$$0 \leq k_p(k) \leq k_d^2, \tag{13}$$

for fixed $k_d$, which follows from the Circle criteria (Vidyasagar, 1993). Importantly, there is no heuristic switch from a “position control mode” to a “velocity control mode.” Rather it is a single controller that is continuously adjusted between these two extremes along the reference trajectory as the end effector approaches the task object. Further, there is no switch from “motion control mode” to a “force control mode” when contact is detected. In fact, the robot lacks a force - torque sensor. Rather, there is a single feedback controller that can achieve soft contact without any switch. This is important because we want to eliminate tuning and commissioning effort, and we desire a control architecture amenable to robustness analysis.

Figure 13 shows a simulation result of the end effector approaching and contacting the brick. The contact between the block and surface, and between the block and robot is modeled as in Section 2. In this simulation, a smooth trajectory commands the left finger to approach at low velocity, while the impedance is reduced by continuously until $k_p = 0$ before impact. From this point forward it is effectively in a velocity control mode. Contact is made, and the Lagrange multiplier becomes active at $t = 12$s. There is no bounce and the force imparted is sufficiently small to maintain the friction contact with the surface, so the block does not move. The small force is maintained after the contact.

## 5 Conclusions

We have presented a useful model of contact and collision intended to support development of model-based control design and analysis for robotic assembly. The advantages and disadvantages of the modeling approach were discussed. In particular, this method will be effective in situations involving low numbers of contacts, where it is important to enforce penetration constraints after the collision transient occurs, and in cases where dynamic analysis is required, not just time-domain simulation. Three examples illustrate the method, with the ball maze providing a control design and simulation use case.

There are several important issues that are not addressed in this paper. First, friction at the contact is not included, although it would not be difficult to augment the approach to include it. Second, and more importantly, redundant constraints, where the Jacobian $H$ is non full row rank, and therefore the Lagrange multipliers are not unique, is also not considered. Additional logic or a way to regularize the problem, i.e., by adding an additional constraint, might work. But despite these and other disadvantages, we intend to use this method to develop a range of control algorithms for robotic assembly, in addition to modeling other mechatronic problems in which contact and collisions are central to the problem.

# References

Gianluca Bardaro, Luca Bascetta, Francesco Casella, and Matteo Matteucci. Using Modelica for advanced multi-body modelling in 3D graphical robotic simulators. In *Proceedings of the 12th International Modelica Conference*, pages 887–894, 2017.

J. W. Baumgarte. Stabilization of constraints and integrals of motion in dynamic systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

J. W. Baumgarte. A new method of stabilization for holonomic constraints. *ASME Journal of Applied Mechanics*, 50:869–870, 1983.

Scott A. Bortoff. Object-oriented modeling and control of delta robots. In *IEEE Conference on Control Technology and Applications*, pages 251–258, 2018.

Scott A. Bortoff. Using Baumgarte's method for index reduction in Modelica. In *Proceedings of the 13th International Modelica Conference*, pages 333–342, March 2019.

R. Clavel. Device for the movement and positioning of an element in space. U.S. Patent 4, 976, 582, Dec. 11 1990.

Erwin Coumans. Bullet 2.83 physics SDK manual. https://github.com/bulletphysics/bullet3/tree/master/docs, 2015.

Vadim Engelson. *Integration of Collision Detection with the Multibody System Library in Modelica*. PhD thesis, Link oping University, 2000.

Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlmann. *Physics-Based Animaation*. Charles River Media, 2005.

Elmer Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for compting the distance bewteen complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

Andreas Hofmann, Lars Mikelsons, Ines Gubsch, and Christian Schubert. Simulating collisions within the Modelica multibody library. In *Proceedings of the 10th International Modelica Conference*, pages 949–957, 2011a.

Andreas Hofmann, Lars Mikelsons, Ines Gubsch, and Christian Schubert. Modelica idealized contact library. https://github.com/modelica-3rdparty/IdealizedContact, 2011b.

Martin Otter, Hilding Elmqvist, and José Díaz López. Collision handling for the Modelica MultiBody library. In *Proceedings of the 4th International Modelica Conference*, pages 45–53, March 2005.

Bernhard Thiele, Thomas Beutlich, Volker Waurich, Martin Sjölund, and Tobias Belmann. Towards a standard-comform, platform-generic and feature-rich Modelica device drivers library. In *Proceedings of the 12th International Modelica Conference*, pages 713–723, 2017.

Jeroen van Baar, Alan Sullivan, Radu Cordorel, Davesh Jha, Diego Romeres, and Daniel Nikovski. Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

M. Vidyasagar. *Nonlinear Systems Analysis: Second Edition*. Prentice-Hall, 1993.

Miomir Vukobratovic, Veljko Potkonjak, and Vladimir Matijevic. *Dynamics of Robots with Contact Tasks*. Kluwer, 2003.

# Fast Simulations of Air Conditioning Systems Using Spline-Based Table Look-Up Method (SBTL) with Analytic Jacobians

Lixiang Li[1] Jesse Gohl[1] John Batteh[1] Christopher Greiner[2] Kai Wang[2]

[1]Modelon Inc, USA, {lixiang.li, jesse.gohl, john.batteh}@modelon.com
[2]Ford Motor Company, USA, {cgreiner, kwang37}@ford.com

## Abstract

Refrigerant property calculation has a significant impact on the computational performance of vapor compression cycle simulations. In a previous publication, the authors have described the Modelica implementation of a Spline-Based Table Look-Up Method (SBTL) for fast calculation of refrigerant properties. This implementation demonstrated significant improvement in computational speed for a range of complex air conditioning system models. This paper describes further development of the SBTL method to allow the generation of analytic Jacobians. The new implementation with analytic Jacobian capability is tested on a range of air conditioning system models and demonstrates significant further improvement of computational speed when compared to the original SBTL model.

*Keywords: Refrigerant Properties, Equation of State (EOS), Analytic Jacobians, Thermodynamic Modeling, Vapor Compression Cycle, Air Conditioning, Spline Interpolation, Computational Performance*

## 1 Introduction

Dynamic simulations of vapor compression cycles typically require significant numbers of function evaluations to calculate properties of the working fluid. There are a number of different approaches for working fluid calculations. These calculations are typically performed using reference Helmholtz energy (multi-parameter) equation of state (EOS) (Tillner-Roth et al, 1994; Richter et al, 2011) to achieve high accuracy. Short formulation (Span et al, 2003) of Helmholtz energy EOS improves the computational performance, but it does not cover all popular refrigerants, in particular the commonly used R1234yf. In Modelon's Air Conditioning Library (Modelon, 2019), both the reference Helmholtz EOS and short Helmholtz EOS are implemented for a wide range of refrigerants.

While typically accurate, these Helmholtz approaches have a complicated multi-parameter functional form that is very computationally expensive. In addition, Helmholtz energy EOS uses density and temperature as thermodynamic states, but system models typically use pressure and enthalpy as dynamic states. Thus, internal iteration is needed when property calculations are needed in vapor compression cycle system simulations.

To address performance requirements for complex vapor cycle system simulations, a new property model based on the Spline-Based Table Look-Up Method (Kunick et al, 2015) was implemented in Modelica (Li et al., 2018). This implementation uses external C functions for fast spline evaluations and inversion. A full overview of the SBTL method and implementation is provided in the previous publication (Li et al., 2018) from the authors and thus not repeated here. The SBTL property model was validated against the existing highly accurate Helmholtz energy EOS models. Accuracy and computational performance were tested starting from single function calls and then increasing in system complexity to heat exchanger tests and then full system models of vapor compression cycles using Modelon's Air Conditioning Library and the associated library regression testing suite. The SBTL property model was also tested with a series of system models from Ford Motor Company including drive cycle simulations, air conditioning pulldown tests with air loop, and a shutdown-startup test. Testing demonstrated significant improvements in computational speed without sacrificing accuracy over the range of tests conducted. Complex system models demonstrated 2x speedups using the new SBTL property model as compared to those using the Helmholtz energy EOS. The SBTL property model and implementations for R134a and R1234yf were added to the 2018.2 release (version 1.17) of Modelon's Air Conditioning Library.

While the SBTL method has demonstrated significant improvements in computational efficiency for vapor cycle models, there are additional opportunities for efficiency improvement. In particular, the SBTL property model as initially implemented in Modelica and used with Dymola (Dassault Systemes, 2019) results in numerical Jacobians when used in system simulations. The computational benefits of analytic Jacobians with Modelica models has been well-documented (Braun et al., 2011; Jorissen et al., 2015). In particular, Modelon has experienced

10.3384/ECP2016964 DOI

speedups of up to 100x in very complex thermal power models after the development of a high precision water property model in Thermal Power Library (Modelon, 2019) that is capable of generating analytic Jacobians with Dymola.

This paper describes the extension of the SBTL method to allow the generation of analytic Jacobians. A short overview of the use of analytic Jacobians in system simulation is provided in Section 2. A brief overview of the SBTL method and implementation is provided in Section 3. Section 4 provides an overview of the development and implementation effort to enhance the SBTL method to allow analytic Jacobian generation. Section 5 shows a series of different models that were tested to validate the accuracy and computational efficiency improvement of the SBTL method with analytic Jacobian capability.

## 2 Analytic Jacobians in System Simulation

One key advantage of Modelica modeling is that the equations are typically accessible to the Modelica compiler. Depending on the sophistication of the Modelica compiler, it is often possible for the compiler to utilize the differential-algebraic equations to provide additional information required for the numerical solver. In particular, there are two critical places where the equations and their derivatives can be leveraged to improve the computational efficiency of the numerical solver:

- Nonlinear equation solution using Newton-Raphson techniques
- Numerical integration scheme

In both these cases, the Jacobian of the system of equations typically needs to be calculated.

While many Modelica compilers apply automatic differentiation to construct Jacobians, there are many ways in which automatic differentiation can fail to generate an analytic Jacobian. Common issues include external code without derivatives provided via the appropriate annotations, functions without appropriate information for the tool to perform automatic differentiation at the level required to generate an analytic Jacobian, or even equations and equation structure which make it difficult for the tool to perform automatic differentiation. In cases where an analytic Jacobian cannot be generated, numerical Jacobians are calculated. While the analytic Jacobian is not an approximation but is the true symbolic derivatives of the equation, a numerical Jacobian is an approximation and typically calculated via finite differences.

### 2.1 Nonlinear Equation Solution

It is common for the differential-algebraic equations that result from a Modelica model to involve the solution of a nonlinear system of equations. Consider the following representation of a nonlinear system of equations:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{1}$$

Newton's method is a typical iterative technique for the solution of non-linear equation systems. The iteration scheme for Newton's method is as follows:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \boldsymbol{J}(\boldsymbol{x}_n)^{-1}\,\boldsymbol{f}(\boldsymbol{x}_n) \tag{2}$$

where the Jacobian is as follows:

$$\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{\nabla f}(\boldsymbol{x}) \tag{3}$$

Thus, it can be seen that the Jacobian is fundamental for solving nonlinear equation systems. The use of analytic Jacobians increases the accuracy of the derivatives used in the iterative solution of nonlinear equation systems, leading to a reduction in the number of iterations required for convergence and potentially also increase the robustness of the solution of the nonlinear equation system.

### 2.2 Numerical Integration

Many numerical integration schemes require the calculation of the Jacobian of the system of equations (Braun et al., 2011; Jorissen et al., 2015). For example, implicit integration schemes require Jacobian computation and require iterations for convergence. These iterations lead to multiple function evaluations for the calculation of the Jacobian when numerical Jacobians are used. Increased accuracy from analytic Jacobians results in fewer iterations to convergence.

The Jacobian is related to the number of states in the model. Complex models with more states lead to a larger Jacobian as the number of elements in the Jacobian matrix is equal to the square of the number of states. Thus, it is expected that complex models with a larger number of states will exhibit greater computational improvement with analytic Jacobians by elimination of the large number of finite differences that would be required for the approximate of the Jacobian numerically.

## 3 Spline-Based Table Look-Up Method (SBTL) Overview

In recent years, different kinds of interpolation-based methods have been explored for fast calculation of refrigerant properties (Laughman et al, 2012; Schulze, 2013; Aute et al, 2014). SBTL method is one of them with proven accuracy, performance, and robustness. A detailed description of the SBTL method can be found in the reference (Kunick et al, 2015). This section focuses on its distinct features and derivative derivation.

## 3.1 Key features of SBTL method

By using a specific type of quadratic/biquadratic spline (Späth, 1995), the SBTL method (Kunick et al, 2015) possesses the following distinct features:

- Equidistant grid
- Continuous first derivatives
- Analytic inverse
- Consistent phase boundary definition

These features make the method a good fit for system simulation of vapor compressor cycle, where both speed of function evaluations and consistency are key requirements. To further tailor the implementation for modeling vapor compression cycle in Modelica, the authors made several design choices in the previous work (Li et al., 2018):

- Performed an overall fit to cover the whole domain instead of fitting over several sub-divided domains, balancing accuracy and data loading time at model initialization.
- Limited the use of grid transformation, trading-off between accuracy and the cost of derivative evaluation and inversion.
- Used external C functions for the spline evaluation, inversion, and derivatives to maximize the speed.

The benefit of the above choices carries over to the implementation of analytic Jacobians with a fine balance of speed and simplicity.

## 3.2 Derivative of the splines with grid transformation

To illustrate the impact of grid transformation on derivative calculation, we will look at the example of 1D saturation temperature splines. Derivatives of 2D spline follow the same principle.

The SBTL method uses equidistant grid to avoid searching. To enhance the accuracy of the interpolation, particularly for fitting highly non-linear functions, grid transformation can be used on both the independent and dependent variables. The chain rule must be applied properly when calculating derivatives for the transformed variables. Take the 1D saturation temperature spline as an example:

$$\bar{p} = \log(p) \tag{4}$$

$$T_{s\{i\}} = a_{i1} + a_{i2}(\bar{p} - \bar{p}_i) + a_{i3}(\bar{p} - \bar{p}_i)^2 \tag{5}$$

where $\bar{p}_i$ is the $i$th transformed pressure node and $a_{ik}$ are the spline coefficients in the $i$th interval of the spline. The derivative in the $i$th interval is given as:

$$\frac{dT_{s\{i\}}}{dp} = \frac{dT_{s\{i\}}}{d\bar{p}}\frac{d\bar{p}}{dp} = \frac{dT_{s\{i\}}}{d\bar{p}}\frac{1}{p} \tag{6}$$

where

$$\frac{dT_{s\{i\}}}{d\bar{p}} = a_{i2} + 2a_{i3}(\bar{p} - \bar{p}_i) \tag{7}$$

The 2nd-order derivative can be written as:

$$\begin{aligned}\frac{d^2T_{s\{i\}}}{dp^2} &= \frac{d}{dp}\left(\frac{dT_{s\{i\}}}{d\bar{p}}\frac{1}{p}\right) \\ &= \left(\frac{d^2T_{s\{i\}}}{d\bar{p}^2} - \frac{dT_{s\{i\}}}{d\bar{p}}\right)\frac{1}{p^2}\end{aligned} \tag{8}$$

where

$$\frac{d^2T_{s\{i\}}}{d\bar{p}^2} = 2a_{i3} \tag{9}$$

As we can see, more advanced grid transformation can result in complicated expression for derivative function due to chain rule. Therefore, model developers need to be careful when balancing the cost of derivative calculation and model accuracy.

# 4 Implementation of Analytic Jacobians with SBTL

In this section, we discuss the implementation of analytic Jacobians in SBTL medium models. Our first step towards analytic Jacobians was to identify what derivative functions were needed by the compiler to construct the Jacobians. We then derived the derivative functions analytically based 1D and 2D splines. In the SBTL method, dew and bubble enthalpy functions were implemented as inverse function of the 2D spline of temperature (Kunick et al, 2015). Therefore, we had to pay close attention to consistency between function evaluation and derivative calculation when deriving derivatives of the saturation property functions. Additional derivatives might also be needed in some component models to obtain analytic Jacobians for the full system model.

## 4.1 Diagnostics of required derivative functions

Two advanced flags in Dymola were used to identify the derivatives required for the analytic Jacobians:

- Advanced.GenerateAnalyticJacobian
- Advanced.PrintFailureToDifferentiate

With the first flag set to true, Dymola will try to construct analytic Jacobians during compilation of the model. When Dymola fails to do so, the user can set the second flag to true to find out what derivative functions are missing.

Consider the twin evaporator cycle model (see Figure 4) from the Air Conditioning Library as an example, Using the flags above, we were able to find out that we needed to provide derivatives (with respect to time) of the following functions:

- Partial derivatives of density $\left.\frac{\partial\rho}{\partial h}\right|_p$, $\left.\frac{\partial\rho}{\partial p}\right|_h$
- Saturation properties: $\frac{dh_l}{dp}$, $\frac{dh_v}{dp}$, $\frac{d\rho_l}{dp}$, $\frac{d\rho_v}{dp}$
- Isentropic enthalpy $\mathrm{h}_{isen}(p, s, X)$

- Spline functions for compressor efficiencies

In this example, not only derivatives of the medium properties are required, some specific component models (e.g. compressor) also need derivatives. We suggest model developers perform this kind of diagnostics on selected system models that can well represent the causality and complexity of their typical use cases.

### 4.2 Derivatives of saturation properties

While it is difficult to derive analytic derivatives for fluid property models based on Helmholtz energy EOS, it is relatively straight forward to do so for spline-based models, especially in the single-phase region. SBTL method, despite its advantage in speed and consistency, does require additional work when it comes to derivative functions of saturation properties. First, grid transformation needs to be taken care of as discussed in Section 3.2. More importantly, we need to keep the derivatives consistent with the spline fit, i.e. all derivatives are obtained from the 1D and 2D spline and no extra information or fit shall be used.

Dew and bubble enthalpy functions are implemented as inverse of the 2D spline for temperature in SBTL method. They can be written as ${h_l}^{inv}(p,\, T_s(p))$ and ${h_v}^{inv}(p,\, T_s(p))$. This feature ensures consistent definition of the phase boundary, but it is more complicated to get derivatives of saturation properties in SBTL method than an implementation of $h_l$ and $h_v$ as 1D splines of pressure. As summarized in (Thorade and Saadat, 2013), we can write the first-order derivatives (with respect to pressure) as

$$\frac{\mathrm{d}h}{\mathrm{d}p} = \left.\frac{\partial h}{\partial p}\right|_T + \left.\frac{\partial h}{\partial T}\right|_p \frac{\mathrm{d}T_s}{\mathrm{d}p} \tag{10}$$

where

$$\left.\frac{\partial h}{\partial p}\right|_T = -\frac{\partial T/\partial p|_h}{\partial T/\partial h|_p} \tag{11}$$

$$\left.\frac{\partial h}{\partial T}\right|_p = \frac{1}{\partial T/\partial h|_p} \tag{12}$$

Similarly, we can get

$$\frac{\mathrm{d}\rho}{\mathrm{d}p} = \left.\frac{\partial \rho}{\partial p}\right|_T + \left.\frac{\partial \rho}{\partial T}\right|_p \frac{\mathrm{d}T_s}{\mathrm{d}p} \tag{13}$$

where

$$\left.\frac{\partial \rho}{\partial p}\right|_T = \left.\frac{\partial \rho}{\partial p}\right|_h + \left.\frac{\partial \rho}{\partial h}\right|_p \left.\frac{\partial h}{\partial p}\right|_T \tag{14}$$

$$\left.\frac{\partial \rho}{\partial T}\right|_p = \frac{\partial \rho/\partial h|_p}{\partial T/\partial h|_p} \tag{15}$$

The partial derivatives $\left.\frac{\partial T}{\partial p}\right|_h, \left.\frac{\partial T}{\partial h}\right|_p, \left.\frac{\partial \rho}{\partial p}\right|_h, \left.\frac{\partial \rho}{\partial h}\right|_p$ can be evaluated as derivatives of 2D splines $T(p,\, h)$ and $\rho(p,\, h)$ along the dew and bubble lines.

Second-order derivatives of saturation properties are also needed when calculating the derivative of $\left.\frac{\partial \rho}{\partial h}\right|_p$ and $\left.\frac{\partial \rho}{\partial p}\right|_h$ in the two-phase region. Derivation details are outlined in a Jupyter Notebook on GitHub of CoolProp, hence not to repeat in this paper.

## 5 System simulations using SBTL method with analytic Jacobians

In this section, we demonstrate the speedup of system simulations using SBTL method with analytic Jacobians by comparing with those using Helmholtz energy EOS or SBTL method without analytic Jacobians. We first look at a simple heat exchanger test and a system model of vapor compression cycle in the Air Conditioning Library. We then move on to a complex three-branch air conditioning (AC) system and an AC system coupled with a cooling loop, both developed at the Ford Motor Company, to evaluate their computational performance in drive cycle simulations. Lastly, we study the scalability of the SBTL method with analytic Jacobians by using higher discretization for the heat exchangers in the three-branch model. All the simulations are performed with the same computer configuration shown in Table 1.

**Table 1.** Configuration of the computer used for testing

| | |
|---|---|
| *Model* | Dell XPS 8700 Desktop |
| *Processor* | Intel® Core™ i7-4770 CPU |
| *RAM* | 16.0 GB |
| *System* | 64-bit, x64 based, Windows 10 Pro |
| *Software* | Dymola 2018 FD01 |
| *C compiler* | Visual Studio 2015 Express Edition |
| *Solver* | Dassl |
| *Tolerance* | 1e-6 (to ensure mass conservation) |

### 5.1 Comparison of heat exchanger test results and performance

Heat exchangers modeled by finite volume method usually have the most property function calls in system simulations of vapor compressor cycles. Therefore, a heat exchanger test is representative of how well the property model is going to perform in full system simulations. We used an evaporator test bench from the Air Conditioning Library, shown in Figure 1, to test the SBTL property model with analytic Jacobians for R134a. The two-layer evaporator had 18 discretized volumes and 56 dynamic states (pressure and enthalpy in each refrigerant volume and temperature in each wall element). The test was run for 20 s with constant boundary conditions except for the refrigerant mass flow rate which ramps from 0.028 to 0.038 kg/s during 5s to 7s.

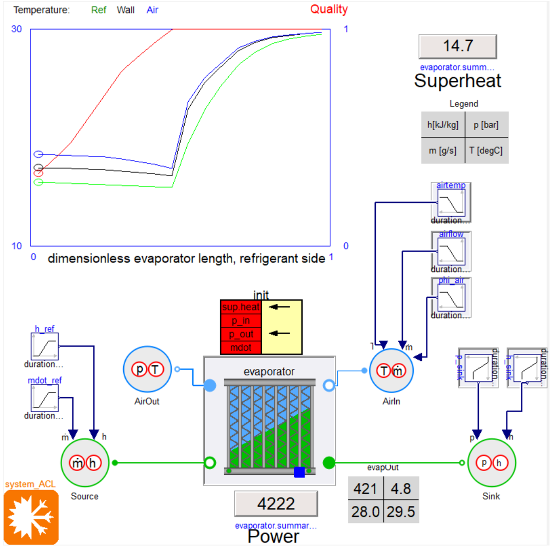


**Figure 1.** Evaporator test bench in Modelon's Air Conditioning Library.

Trajectories of evaporator cooling power and air outlet temperature matched very well between the SBTL model with analytic Jacobian and its data source, the short formulation of Helmholtz energy EOS, as shown in Figure 2. The maximum deviation in cooling power is less than 0.03%. The computation performance of different property models is vastly different. CPU time of the three cases can be found in Figure 3. The use of analytic Jacobians cuts the time by 17% compared with the original SBTL model. The heat exchange tests provide a preview of the great potential of using the SBTL method with analytic Jacobians to speed up system simulations.

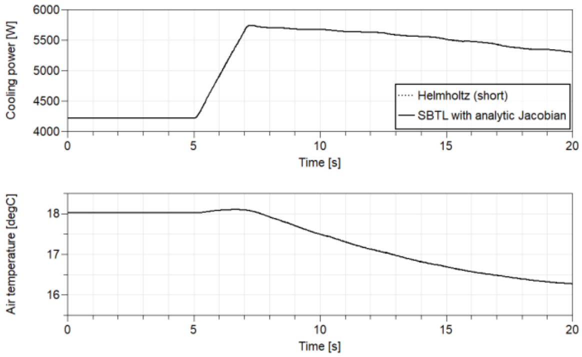


**Figure 2.** Comparison of cooling power and air outlet temperature.

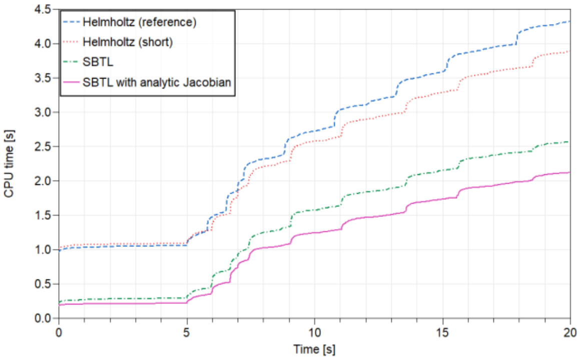


**Figure 3.** Comparison of CPU time running the evaporator test bench.

## 5.2 Comparison of system models in the Air Conditioning Library results and performance

The twin evaporator model from the Air Conditioning Library (Figure 4) was simulated for 180s to further evaluate the speedup of a complex system model brought by analytic Jacobians. The model has 131 dynamic states and has a 10°C ramp in the incoming air temperature at the condenser after 90 s.

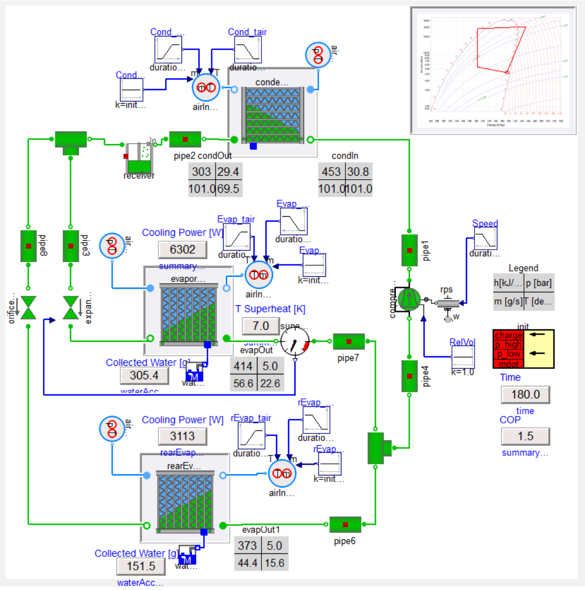


**Figure 4.** Twin evaporator cycle model in Modelon's Air Conditioning Library.

As shown in Figure 5, simulation with analytic Jacobians ran at 1.7x the speed of the one without them. The speedup can mostly be explained by a decrease in the number of function evaluations from 2987 to 1000 when analytic Jacobians were used. This agrees with findings in (Jorissen et al., 2015).

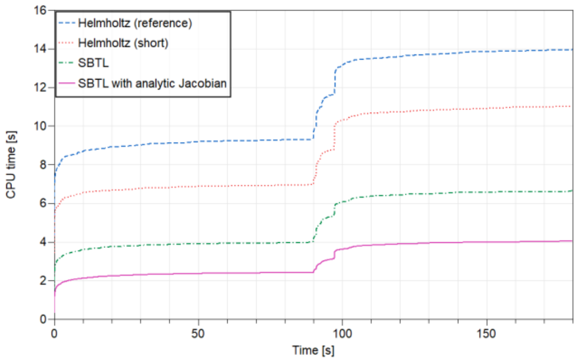


**Figure 5.** CPU time comparison of the twin evaporator cycle simulations. From top to bottom: Blue – Reference Helmholtz EOS, Red – Short formulation of Helmholtz EOS, Green – SBTL, Magenta – SBTL with analytic Jacobians.

We then moved on to a pull-down test example with 91 dynamic states in the Air Conditioning Library, as depicted in Figure 6. A comparison of the CPU time can is shown in Figure 7. The simulation ran 2x faster with analytic Jacobians than without.

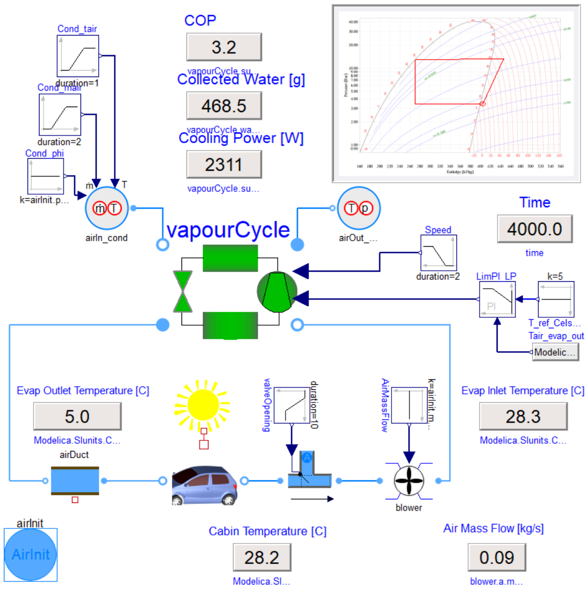


**Figure 6.** Pull-down test example in Modelon's Air Conditioning Library.

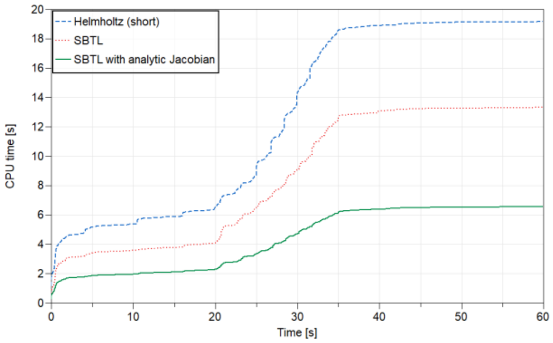


**Figure 7.** CPU time comparison of the pull-down simulations.

## 5.3 Comparison of Ford air conditioning system models results and performance

In this section, we show the results from two complex automotive air conditioning system models, developed at Ford Motor Company, to illustrate the speedup by the SBTL method with analytic Jacobians in drive cycle simulations. There are three distinct features in these simulations compared with the ones in previous sections:

- The working fluid in both systems is R1234yf. Short formulation of Helmholtz energy EOS was not available for it and reference state Helmholtz EOS had to be used. The original SBTL method offered an even greater speed-up in these cases (Li et al., 2018).
- Both models run into low flow conditions in the chiller, which was known to challenge the performance and robustness of the model.
- Tabulated boundary conditions or compressor shut down are imposed on the model, inducing more fast dynamics.

By running these simulations, we can get a better understanding of the performance and robustness of the SBTL method with analytic Jacobians in complex system models.

Figure 8 depicts a three-branch air conditioning system. Two evaporators and one chiller are connected in parallel, and all of them are superheat controlled by the thermal expansion valves (TXVs). The condenser model consists of a condenser section, a subcool section, and an internal receiver. The system model has 183 dynamic states. Apart from the tabulated boundary conditions, we deliberately introduced fast dynamics by rapidly closing the valve upstream of the chiller TXV and introducing low flow rate condition in the chiller.

Drive cycle (SC03) simulations were performed for 598s using different R1234yf medium models. Figure 9 shows a good match in results. As seen in Figure 10, the simulation finished in 112.6s when using the SBTL medium with analytic Jacobians, just 19% of the real-time (598s). This is more than 4x the speed of the original SBTL medium and 10x the speed of the reference state Helmholtz energy EOS.

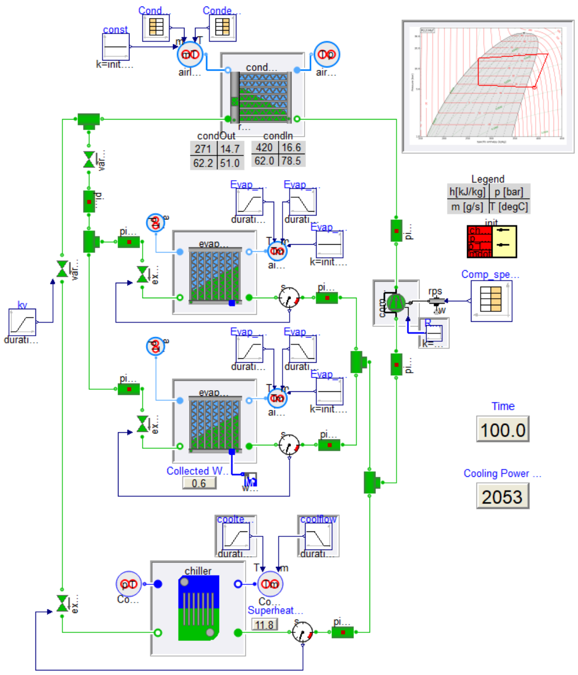


**Figure 8.** Air conditioning system with two evaporators and one chiller connected in parallel in the loop.

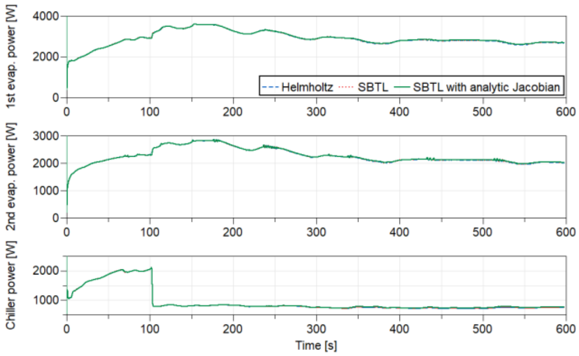


**Figure 9.** Cooling power of the evaporators and the chiller using different property models.

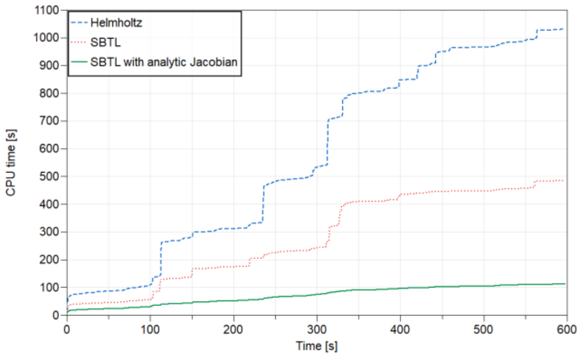


**Figure 10.** CPU time comparison of the three-branch air conditioning system model.

Figure 11 is a diagram of a vapor compression cycle coupled with a battery cooling system. The refrigerant loop was initialized at certain operating conditions (defined by pressures, chiller superheat and condenser) The compressor was off from 0s, simulating cycle shutdown. The temperature of the battery kept rising afterward and hit a certain threshold. The compressor was then turned back on (at about 122s) to cool the coolant and eventually drive the battery temperature down. Fast dynamics introduced by the shutdown and startup of the compressor posed serious challenges to the performance and robustness of the model.

Compressor speed and refrigerant mass flow rate are plotted in Figure 12. Simulations using different R1234yf medium models predicted the same mass flow trajectory during the fast transients. As shown in Figure 13, the SBTL medium with analytic Jacobians offered significant speedup (14x) compared to the original SBTL medium. If we focus on the zero-flow period from 22s to 122s, it only took 1.34s of CPU time using analytic Jacobians, a 125x speedup compared to the simulation using the SBTL medium without analytic Jacobians.

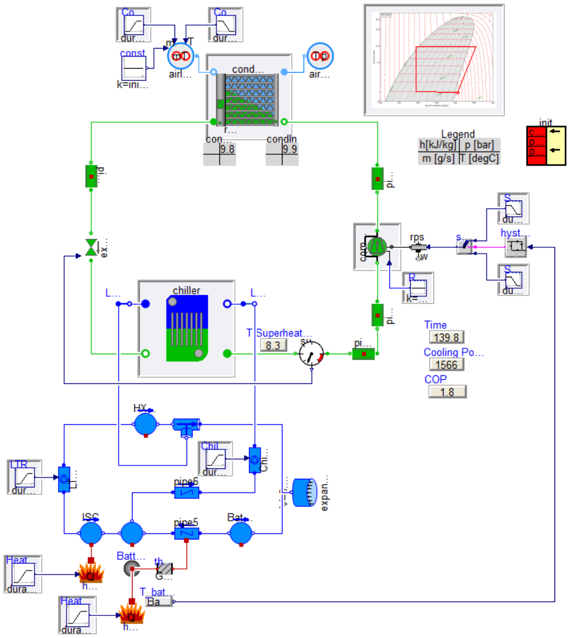


**Figure 11.** R1234yf vapor compression cycle with a chiller connected to a battery cooling loop.

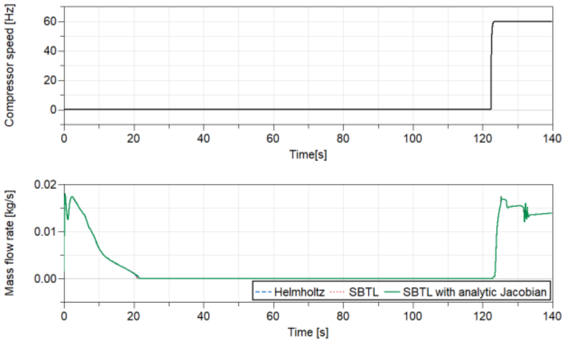


**Figure 12.** Compressor speed and refrigerant mass flow rate.

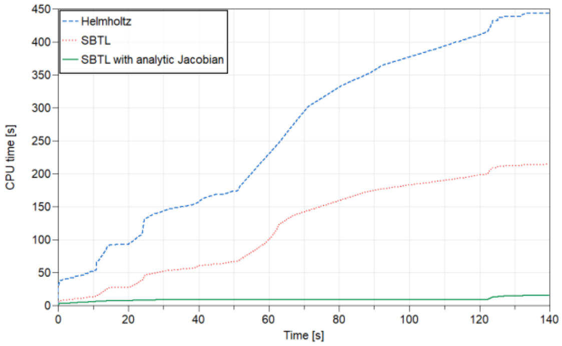


**Figure 13.** CPU time comparison for the model shown in Figure 11.

## 5.4 Scalability of SBTL method with analytic Jacobians

To better understand how the performance of the SBTL method with analytic Jacobians scales with the number of states, we simulated the three-branch air conditioning system, shown in Figure 8, with higher discretization on the refrigerant side of the heat exchanger. Non-sparse DASSL solver was used for the study. Simulation setup and CPU time are summarized in Table 2. As the total number of states increases, the CPU time grows as $O(N^{1.6})$.

**Table 2.** Setup and CPU time of 598s drive cycle simulations of the three-branch air conditioning system with different discretizations.

| Number of finite volumes | | | Total number of states | CPU time [s] |
|---|---|---|---|---|
| Condenser | Each evaporator | Chiller | | |
| 12 | 12 | 5 | 183 | 112.65 |
| 24 | 24 | 10 | 311 | 327.01 |
| 36 | 36 | 15 | 439 | 472.23 |
| 48 | 48 | 20 | 567 | 730.11 |
| 60 | 60 | 25 | 695 | 1014.87 |

## 6 Conclusions

This paper summarizes the further development of the SBTL method for fast calculation of refrigerant properties in Modelica to allow the generation of analytic Jacobians. By expanding the SBTL implementation to provide the additional derivatives needed for analytic Jacobians, Dymola can generate analytic Jacobians for complex vapor cycle system models with Air Conditioning Library. The new SBTL model capable of analytic Jacobians was tested on a suite of models ranging from a heat exchanger bench test to full closed vapor cycle loop systems, including several complex system models from Ford Motor Company.

The computational improvements from the SBTL models with analytic Jacobians are significant. While the original implementation of the SBTL method provides improvements in computational speed on the order of 33% - 50% for complex system models when compared to the baseline Helmholtz property models, the computational improvements from SBTL with analytic Jacobians are even greater when compared to the original SBTL implementation without analytic Jacobians. Reductions in computational speed ranging from to 2-4x are obtained on complex system models for SBTL with analytic Jacobians versus SBTL without analytic Jacobians. Even greater improvement was observed in models under low or zero flow conditions. In addition, the computational improvement could be even larger for more complex models with even more states.

The new SBTL models with analytic Jacobians capability will be available in an upcoming release of Modelon's Air Conditioning Library.

## References

Tillner-Roth, R. and Baehr, H.D. An international standard formulation of the thermodynamic properties of 1,1,1,2-tetrafluoroethane (HFC-134a) for temperatures from 170 K to 455 K at pressures up to 70 MPa. *Journal of Physical and Chemical Reference Data*, 23(5), 657-729, 1994. DOI: https://doi.org/10.1063/1.555958

Span, R. and Wagner, W. Equations of state for technical applications. I. Simultaneously optimized functional forms for nonpolar and polar fluids. *International Journal of Thermophysics*, 24(1), 1-39, 2003. DOI: https://doi.org/10.1023/A:1022390430888

Modelon AB, *Air Conditioning Library*, https://www.modelon.com/library/air-conditioning-library/, Lund, Sweden, 2019.

Richter, M., McLinden, M.O., and Lemmon, E. W. Thermodynamic Properties of 2, 3, 3, 3-Tetrafluoroprop-1-ene (R1234yf): Vapor Pressure and p–ρ–T Measurements and an Equation of State. *Journal of Chemical & Engineering Data*, 56(7), 3254-3264, 2011. DOI: https://doi.org/10.1021/je200369m

Kunick, M., and H. J. Kretzschmar. Guideline on the fast calculation of steam and water properties with the spline-based table look-up method (SBTL). *Technical report, The International Association for the Properties of Water and Steam*, Moscow, Russia, 2015. URL: http://www.iapws.org/relguide/SBTL.html

Li, L., Gohl, J., Batteh, J., Greiner, C., and Wang, K., Fast Calculation of Refrigerant Properties in Vapor Compression Cycles Using Spline-Based Table Look-Up Method (SBTL). *Proceedings of the 1st American Modelica Conference*, p. 77-84, 2018. DOI: http://dx.doi.org/10.3384/ecp1815477

Dassault Systemes, Dymola, https://www.3ds.com/products-services/catia/products/dymola/, 2019.

Braun, W., Ochel, L., and Bachmann, B., Symbolically Derived Jacobians Using Automatic Differentiation – Enhancement of the OpenModelica Compiler. *Proceedings of the 8th International Modelica Conference*, DOI: 10.3384/ecp11063495, p. 495-501, 2011.

Jorissen, F., Wetter, M., and Helsen, L., Simulation Speed Analysis and Improvements of Modelica Models for Building Energy Simulation. *Proceedings of the 11th International Modelica Conference*, p. 59-69, 2015. DOI: http://dx.doi.org/10.3384/ecp1511859

Modelon AB, *Thermal Power Library*, https://www.modelon.com/library/thermal-power-library/, Lund, Sweden, 2019.

Laughman, C., Zhao, Y., and Nikovski, D. Fast Refrigerant Property Calculations Using Interpolation-Based Methods. *International Refrigeration and Air Conditioning Conference*. Paper 1344, 2012. URL: https://docs.lib.purdue.edu/iracc/1344/

Schulze, C. W. A contribution to numerically efficient modeling of thermodynamic systems. *PhD Thesis*, 2013. URL: http://www.digibib.tu-bs.de/?docid=00057492

Aute, V. and Radermacher, R. Standardized Polynomials for Fast Evaluation of Refrigerant Thermophysical Properties. *International Refrigeration and Air Conditioning Conference*. Paper 1499, 2014. URL: https://docs.lib.purdue.edu/iracc/1499/

Späth, H. One dimensional spline interpolation algorithms. AK Peters/CRC Press, 1995.

Späth, H. Two dimensional spline interpolation algorithms. AK Peters, Ltd., 1995.

Tummescheit, H. Design and implementation of object-oriented model libraries using Modelica. *PhD Thesis*, 2002. URL: http://lup.lub.lu.se/record/20836

Thorade, M. and Saadat, A Partial derivatives of thermodynamic state properties for dynamic simulation. *Environmental earth sciences*, 70(8), pp.3497-3503, 2013. DOI: 10.1007/s12665-013-2394-z

Jupyter Notebook on the GitHub of CoolProp, URL: https://github.com/CoolProp/CoolProp/blob/master/doc/notebooks/Saturation.ipynb

# Data-driven Prediction of Occupant Presence and Lighting Power: A Case Study for Small Commercial Buildings

Jing Wang[1], Wangda Zuo[1], Sen Huang[2], Draguna Vrabie[2]

[1]Department of Civil, Environmental, and Architectural Engineering, University of Colorado Boulder, USA,
{jing.wang, wangda.zuo}@colorado.edu

[2]Pacific Northwest National Laboratory, USA, {huang875, draguna.vrabie}@pnnl.gov

## Abstract

Commonly used deterministic methods are unable to capture the randomness in occupant behavior and its impact on electric power consumption. In this paper, we propose a new data-driven model to capture occupant behavior in a stochastic manner. Unlike existing models and prediction tools, this new model does not require occupant presence data and can learn occupants' arrival and departure time based on lighting power consumption data, which is more readily available than occupant presence data. We applied this occupant behavior model to lighting power consumption prediction and implemented the entire prediction process in Modelica. We then validated the Modelica model by comparing the predicted daily, weekly and monthly peak lighting power with measurements from two small commercial buildings. The results suggest that the prediction matches the measurement within acceptable deviations of 7%. The results also indicate that the proposed stochastic model performs better for long-term prediction of lighting power (monthly and weekly) than the short-term (daily).

*Keywords: Occupant behavior modeling, occupant presence prediction, lighting power prediction, regression model, stochastic simulation*

## 1 Introduction

The increasing penetration of renewable energy is introducing more variability within the power grid (J. Wang et al. 2018). To better balance generation and consumption, the power demand side needs to become more flexible and even more controllable. Some studies focus on estimating building load flexibility by controlling thermostatically controllable loads (TCLs) such as HVAC systems and water heaters in buildings (Wu et al. 2018; Zhao et al. 2017). Compared to TCLs, the lighting system has the advantage of shorter response time which makes it more suitable for faster demand response mechanisms (e.g., shimmy).

The stochasticity of occupant behavior and its impact on power and energy consumption presents a challenge to accurate real-time estimation of building electric loads. Traditional building energy modeling tools use static hourly schedules both for occupant presence and building equipment. This leads to discrepancies between the simulated power shape and the actual consumed power (Luo et al. 2017; Kim et al. 2017), especially for short-term prediction scenarios such as those needed for fast demand response. Limited data availability is a second challenge, as due to privacy reasons, occupant sensor data is often unavailable. These challenges must be accounted for in theoretical and model-based studies on occupant behavior and its related impacts on the power consumption and flexibility characterization of the built environment.

For commercial buildings, existing occupant presence prediction models have been developed mainly on single office rooms. Wang et al. used exponential distribution to predict the vacancy intervals of single offices (D. Wang, Federspiel, and Rubinstein 2005). Small commercial buildings have not gained enough attention concerning occupant behavior studies.

Lighting prediction models have been investigated over the past 40 years, and the research points to strong correlation between occupants' presence and the lighting status in a zone. The first published study for occupants' light switching behavior in office buildings found that switching mainly takes place when entering or vacating a space and the switch-on probability on arrival exhibits a strong correlation with minimum daylighting illuminance in the working area (Hunt 1980). Manual switch-off probability of lights is strongly correlated with the expected length of absence (Pigg, Eilers, and Reed 1996). Later, this research was expanded by the study of correlations between intermediate switch-on/-off behavior and illuminance levels (Reinhart and Voss 2003).

In this paper we propose a methodology for occupant presence and lighting power prediction based on minute-level power meter data. We apply the methodology for two small commercial buildings use cases (one bakery and one ice cream shop) and validate the prediction performance with real data collected from building sites. Here we present only the prediction of occupant presence and lighting power. In future work we will extend the methodology to other loads driven by occupant behavior.

DOI 10.3384/ECP2016973

The innovation of this work lies in: (1) The proposed method can be applied to occupant presence prediction without occupancy sensor data and it has been validated against real power meter data. (2) The method can be used for sub-hourly power demand prediction within acceptable deviations of 7%. (3) The method could be applied to other building systems and the Modelica model is extensible and scalable. The rest of the paper is organized as follows: Section 2 presents the methodology. Section 3 discusses the results. Section 4 concludes this paper with future work and limitations.

## 2 Methodology

Our method is based on the assumption that the usage of the lighting system and its associated power consumption is strongly determined by the presence of the occupants in the building spaces. This assumption allows us to extract occupant presence schedules from lighting power data. We then use the extracted presence data to train logistic regression models that predict people's arrival and departure times. The trained probability models are then implemented in Modelica language to reproduce building occupancy patterns. The lighting power is then predicted by multiplying the occupant presence value (0 or 1) with the observed nominal lighting power. We then extend the model to address realistic scenarios of multi-stage lighting power. To validate our model, we compare the simulation results with the lighting power data collected at two building sites and evaluate the model performance with respect to several statistical metrics.

The following flowchart (Figure 1) shows the research workflow for the results presented in this paper.

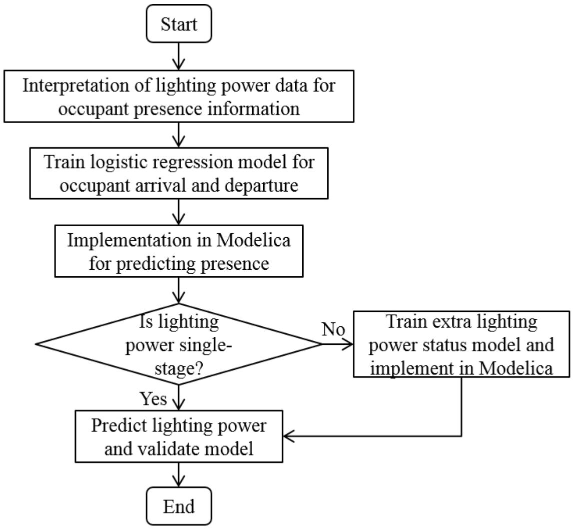

Start
Interpretation of lighting power data for occupant presence information
Train logistic regression model for occupant arrival and departure
Implementation in Modelica for predicting presence
Is lighting power single-stage?
No
Train extra lighting power status model and implement in Modelica
Yes
Predict lighting power and validate model
End

**Figure 1.** Research and modeling workflow.

### 2.1 Determine Occupant Presence

In this section, we discuss the extraction of occupant presence information from the lighting power data. As indicated in the literature review, occupant arrival time and departure time has a strong correlation with the lighting power utilization: According to Hunt's work (Hunt 1980), the action of turning on the lights depends on the minimum illuminance level on the working plane upon arrival and people tend to leave the lights on until the space is fully empty. This is consistent with our observation on the lighting power data in the two studied buildings (C2: ice cream shop and F1: bakery). As plotted in Figure 2 and Figure 3, once the lights are turned on, they will remain on for the whole day until all the people leave the space. This means that in this case the illuminance level is not a strong driver for the light utilization. In our preparation work where we used regression of lighting power based on indoor illuminance levels, prediction accuracy was relatively low. In this paper, we will assume that people in the two studied buildings are not sensitive to the illuminance levels and will turn on the lights once they enter the space and will keep the lights on while they are there. In other words, lights are not switched on to increase work-place illuminance levels, but rather to show potential customers that the store is open. Based on this assumption, we extract the occupant presence information from the lighting power data and regard it as the ground truth.

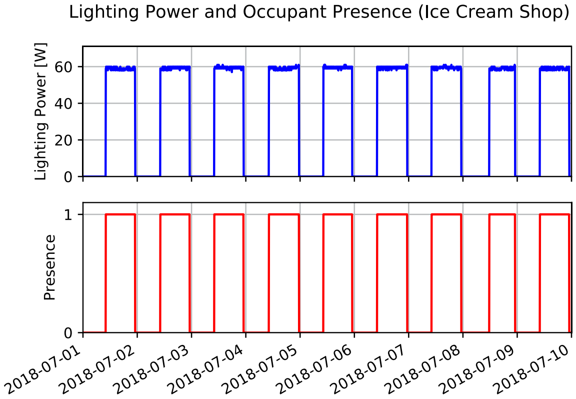


**Figure 2.** Lighting power and occupant presence (C2: ice cream shop).

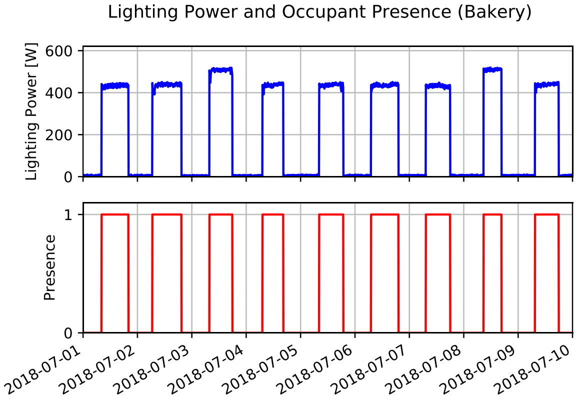


**Figure 3.** Lighting power and occupant presence (F1: bakery).

To convert the lighting power data into occupant presence information, we first cleaned the power meter data by removing obvious outliers such as values that are extremely large for lighting systems. Then, we selected the threshold for determining occupant presence (e.g., 0 for absent; 1 for present) to avoid oscillations in presence status. For instance, the threshold for the ice cream shop is 50 W; for the bakery is 350 W. Any power value above this threshold is converted to 1 and below this threshold into a 0. Because the power data has 1-minute resolutions, we will make the assumption that presence or absence of 1 minute can be neglected and we will filter out two consecutive changes of occupant presence to eliminate frequent oscillations in the resulted presence data.

The lighting power shapes shown in Figure 2 and Figure 3 indicate the different characteristics of the two buildings. For the ice cream shop, only one power value occurs every day regardless of weekday or weekend. However, for the bakery, two distinct levels are observed in the power shape. Hence, for his case, we divide the power shape into two parts namely base lighting power (Figure 4) and additional lighting power (Figure 5) and we model them separately. This two-stage lighting behavior is probably caused by zoning of the lighting system. The expression for multi-stage lighting power can be described with Eq. 1.

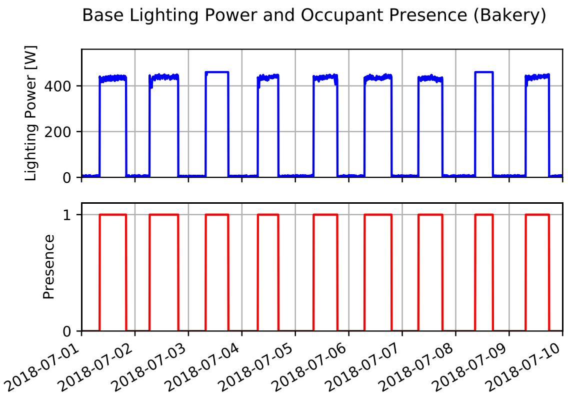


**Figure 4.** Base lighting power and occupant presence in F1 bakery.

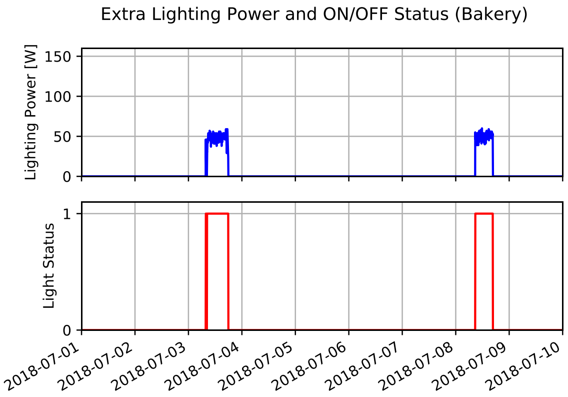


**Figure 5.** Extra lighting power and lighting status in F1 bakery.

$$P(t) = a_0(t)P_{base} + a_1(t)P_{extr,1} + \cdots + a_{n-1}(t)P_{extr,n-1} \tag{1}$$

$P$ is the lighting power; $a_i$ is the binary variable that indicates the status of base or extra lighting; n is the number of stages. Both $P$ and $a_i$ are time dependent. Here, $a_0$ indicates the building occupancy and the rest of them indicates the on/off of extra lighting devices. $a_i$ is predicted with logistic regression models introduced in Section 2.2. $P_{base}$ and $P_{extr,i}$ are the average power value of each stage. For C2, $n = 1$; for F1, $n = 2$.

## 2.2 Train Logistic Regression Models

The prediction of occupant presence could be viewed as a classification problem. As discussed before, the arrival and departure behavior in the two studied buildings follows the same pattern for weekdays and weekends regardless of the indoor illuminance level. Hence, the main feature for classifying occupant presence is the time of the day. We chose logistic regression as our model for the training because: (1) it is a linear classifier and is easy to train; (2) it can reach the same level of accuracy as non-linear classifiers; (3) it is easy to implement in Modelica. We divided the arrival and departure behavior into two models and trained them separately as they have opposite trends along time of the day.

To rule out the impact of seasonal change in the occupant behavior, the training and validation datasets were selected from the summer of 2018. June and July data were used for the training and August data was used for the validation. The accuracy is defined as the rate of classifying the data point into the right group. The confusion matrices for the test datasets of all the regression models are shown in Table 1. The format of the confusion matrices follows the pattern in Table 2.

**Table 1.** Confusion Matrices for Classification Performance.

<table>
<tr><td rowspan="6"><i>C2 Arrival</i></td><td rowspan="3">3693</td><td rowspan="3">44</td><td rowspan="4"><i>F1 Arrival</i></td><td rowspan="2">2736</td><td rowspan="2">132</td></tr>
<tr></tr>
<tr><td rowspan="2">118</td><td rowspan="2">1406</td></tr>
<tr><td rowspan="3">31</td><td rowspan="3">624</td></tr>
<tr><td rowspan="4"><i>F1 Departure</i></td><td rowspan="2">1797</td><td rowspan="2">260</td></tr>
<tr></tr>
<tr><td rowspan="6"><i>C2 Departure</i></td><td rowspan="3">283</td><td rowspan="3">60</td><td rowspan="2">273</td><td rowspan="2">2062</td></tr>
<tr></tr>
<tr><td rowspan="4"><i>F1 Extra On</i></td><td rowspan="2">16</td><td rowspan="2">0</td></tr>
<tr><td rowspan="3">4</td><td rowspan="3">1849</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">0</td></tr>
<tr></tr>
</table>

**Table 2.** Example Confusion Matrix (C2 Arrival).

| | *Predicted No* | *Predicted Yes* |
|---|---|---|
| *Actual No* | 3693 | 44 |
| *Actual Yes* | 31 | 624 |

The accuracy of the classifier is then calculated with Eq. 2.

$$Accuracy = \frac{No.\ of\ correctly\ classified\ points}{No.\ of\ total\ data\ points} \qquad (2)$$

For building F1, the lighting power is divided into the base power and the extra power. The base part reflects occupants' arrival and departure and is regressed in dependence on time of the day. The frequency (i.e., number of total times) of extra lights on of F1 in 2018 is plotted in bars (Figure 8). From the figure, we can see that the status of the extra lighting has a correlation with day of week. Hence, the feature for this part is chosen as day of week. Also, from the figure, we see that the total frequency of extra lights on in 2018 is only 8.8%. To deal with the imbalance in the training dataset, we adopted the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al. 2002), which made the minority (extra lights on) class equal to the majority class (extra lights off) by creating synthetic samples of the minority class. The logistic regression parameters for each model are listed in Table 3. The probability

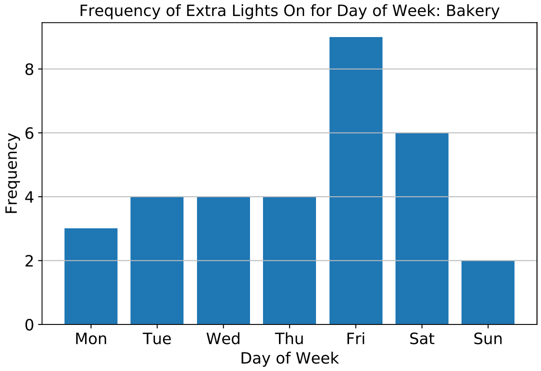


**Figure 8.** Extra lights on frequency for day of week in F1 bakery (2018).

**Table 3.** Logistic Regression Parameters.

| | | Accuracy | β0 | β1 | β2 | β3 | β4 | β5 | β6 | β7 |
|---|---|---|---|---|---|---|---|---|---|---|
| C2 | Arrival | 0.98 | -27.1983 | 0.0447 | N/A | | | | | |
| | Departure | 0.97 | 34.6877 | -0.0249 | | | | | | |
| F1 | Arrival | 0.94 | -11.9311 | 0.0254 | N/A | | | | | |
| | Departure | 0.88 | 13.7769 | -0.0125 | | | | | | |
| | Extra On | 0.84 | -0.8309 | -0.4829 | 0.4967 | -0.2586 | 0.4967 | -0.1171 | -0.4829 | -0.4829 |

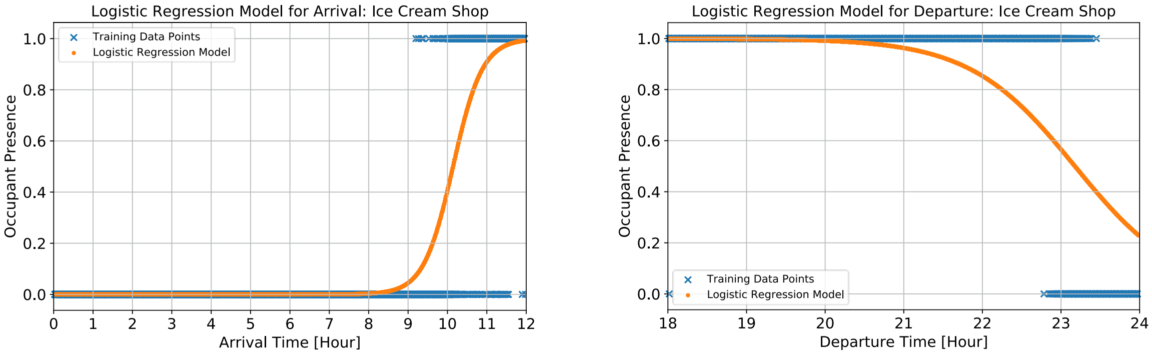


**Figure 6.** Logistic regression model for arrival (left) and departure (right) in C2 ice cream shop.

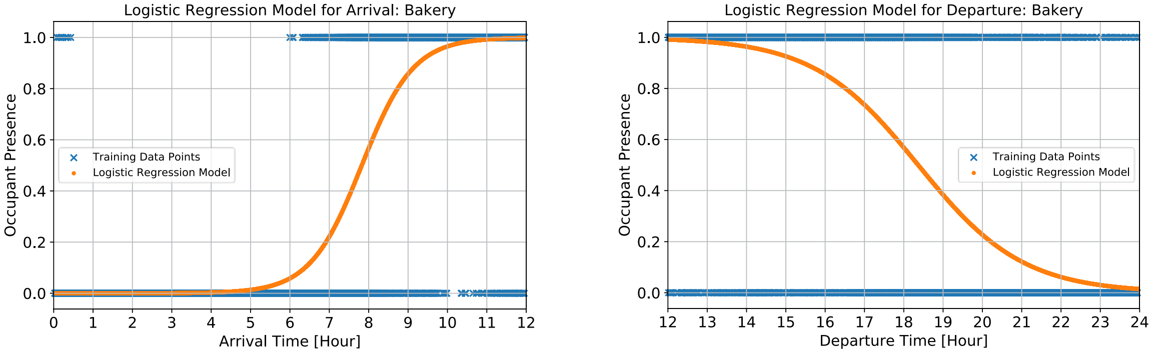


**Figure 7.** Logistic regression model for arrival (left) and departure (right) in F1 bakery.

function is expressed in Eq. 3, where $p$ represents the probability of occupant present or extra lights on; $e$ is the natural log base; $\beta$ is the regression intercept and coefficients; $m$ refers to the number of logistic regression independent variables. The accuracy of all the models are above 84%. Table 4 lists the probability of extra lights on for day of week in building F1.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m)}} \tag{3}$$

**Table 4.** Probability of Extra Lights On for Day of Week from Logistic Regression.

| | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| Proba bility | 0.21 | 0.42 | 0.25 | 0.42 | 0.28 | 0.21 | 0.21 |

Figure 6 and Figure 7 visualize the training data points and the logistic regression models for arrival and departure in C2 and F1. Based on our observations, occupants will arrive before 12 pm and leave after 12 pm. Hence, the arrival models are trained with data points before 12 pm and the departure models with points after 12 pm. For the ice cream shop departure model, people tend to leave very late. To increase the prediction accuracy, we used data after 6 pm to train this model.

### 2.3 Implement in Modelica

The implementation of the presence model and the extra lighting status model is adapted from Buildings.Occupants.Office.Lighting.Hunt1979Light in Modelica Buildings library (Wetter et al. 2014). The model is implemented as a stochastic simulation model. Every two minutes, a binary variable generator will randomly generate a binary number. The probability of this number being 1 equals the calculated probability of the occupant being present at that time of day based on the logistic regression model. Similarly, in the extra light status model, the probability of the random number being 1 equals the probability of the extra light being on at the simulated day of week.

Figure 9 depicts the layout of the two-stage lighting power prediction model for F1. The presence models generate binary signals which will be multiplied with the nominal power of each stage. The nominal powers are the calculated mean values of the lighting power in each stage. The sum of the lighting power of all stages are then compared with the actual lighting power data to validate the performance of the stochastic simulation models. An assumption is made in this model that the extra light will only be on when both of the following conditions are satisfied: (1) The extra light should be on for that day of week; (2) There are occupants in the building. The simulation was run for the whole month of August 2018 and the time step was set as 10 minutes. The actual time step was picked by Dymola to be 2 minutes due to the stochastic events.

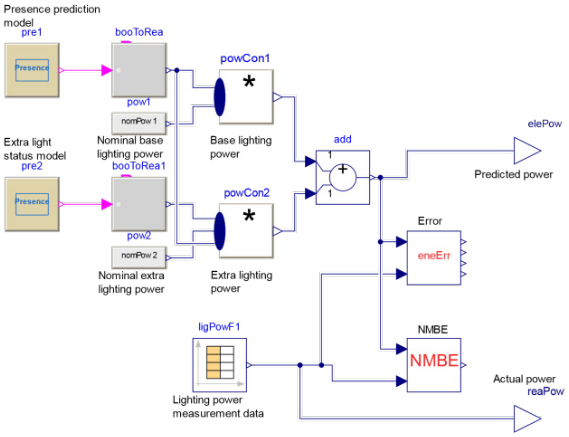


**Figure 9.** Modelica layout of the two-stage lighting power prediction model.

## 3 Results and Discussions

We evaluate both the occupant presence prediction performance and the lighting power prediction performance in this section. The presence models are evaluated with the root mean squared error (RMSE) and the coefficient of variation of RMSE (CVRMSE) of the probability distribution model. The lighting power prediction performance is evaluated with the relative error of the peak power and normalized mean bias error (NMBE). The error in the lighting power prediction is dependent on the presence prediction error as well as the error of nominal power estimation.

ASHRAE Guideline 14-2002 has requirements for whole building energy calibration (ASHRAE 2002). The smaller the time scale, the more tolerant the criteria. For example, the criteria for monthly NMBE is 5%, monthly CVRMSE is 15%, and the criteria for hourly NMBE is 10%, hourly CVRMSE is 30%. Though only the lighting system is calibrated in our work, the principle for different time scales should apply.

### 3.1 Occupant Presence Prediction

RMSE represents the standard deviation of the errors and CVRMSE is the ratio of the standard deviation to the mean of the dependent variable. They both describe how concentrated the data is around the line of its best fit. Large errors are especially noticed in these metrics. The equations for calculating the two metrics are listed below. $x_{o,i}$ is the original value of the predicted variable, $x_{f,i}$ is the forecasted value, N is the number of total data points. Table 5 lists the RMSE and CVRMSE of the occupant and extra lighting status prediction models. The CVRMSE for the occupant presence models are below 25%. The CVRMSE for extra lighting prediction is 125%. This is caused by the imbalance of the training data. The probability of the extra lights being on is much lower than the probability of them

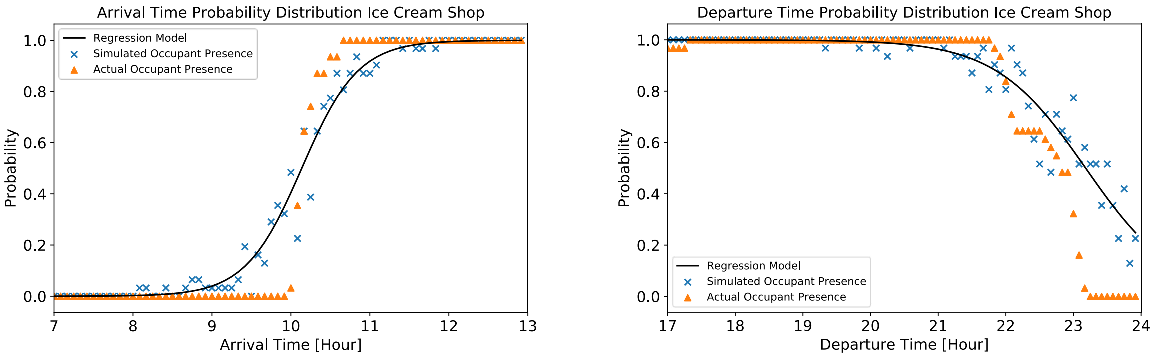


**Figure 10.** Arrival and departure time probability distribution (C2: ice cream shop).

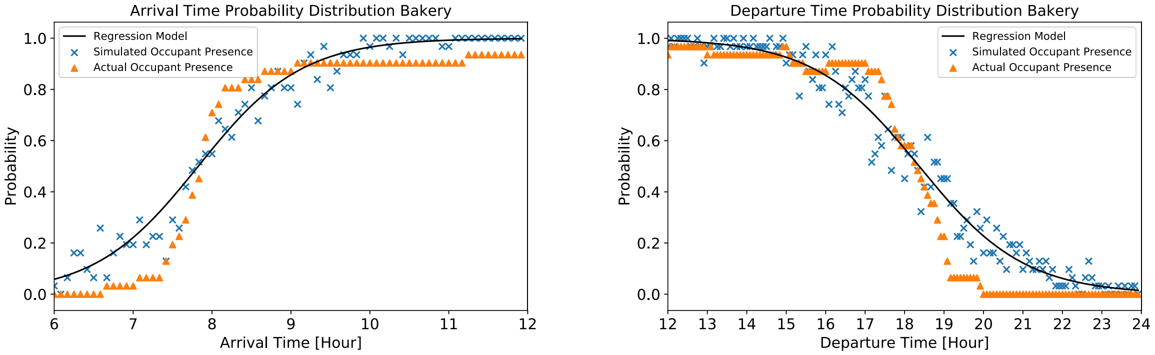


**Figure 11.** Arrival and departure time probability distribution (F1: bakery).

being off. Hence, the mean value $\overline{x_o}$ is very small and small errors could cause a large CVRMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(x_{f,i} - x_{o,i}\right)^2}{N}} \qquad (4)$$

$$CVRMSE = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_{f,i} - x_{o,i}\right)^2}}{\overline{x_o}} \qquad (5)$$

**Table 5.** RMSE and CVRMSE of Occupant Presence and Lighting Status Prediction Results.

| | *C2* | *F1* | |
|---|---|---|---|
| | *Occupant Presence* | *Occupant Presence* | *Extra Lights* |
| *RMSE* | 0.108 | 0.101 | 0.153 |
| *CVRMSE* | 20.9% | 25.0% | 125% |

Figure 10 and Figure 11 plot the regression model, simulated probability distribution and the actual probability distribution of arrival and departure in the two buildings. From the figure, we see that the simulated probability distribution aligns with the regression model very well. The actual probability distribution deviates from the regression model especially during the transitional periods in the middle (e.g., 9 to 11 for C2 arrival, 17 to 21 for F1 departure). This could have been caused by the inappropriate selection of the training data. The high accuracy of the classifiers shown in Table 3 is partially because more data points are located outside the transitional period. The classifier can distinguish those points easier. Another reason could be that only one feature is used to predict occupant presence. This could have limited the shape of the logistic regression model to further fit the actual curve. More features should be explored in the future.

Table 6 compares the probability of extra lights on in F1 calculated from the simulated results and the actual data. From the table, we see that the simulated and actual results deviate on Tuesday and Wednesday. For other days, the simulation results reproduced the actual probability well.

**Table 6.** Comparison of Simulated and Actual Probability of Extra Lights On for Day of Week.

| | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| Simulated | 0 | 0.29 | 0.29 | 0.14 | 0.29 | 0.29 | 0.14 |
| Actual | 0 | 0 | 0 | 0.14 | 0.29 | 0.29 | 0.14 |

## 3.2 Lighting Power Prediction

To evaluate the lighting power prediction performance of the models, peak power prediction relative error and NMBE are calculated on a monthly, weekly and daily basis. In this way, the lighting power prediction performance is evaluated for different time scales. As the models in this paper are mainly designed for shorter-time demand response scenarios, annual energy consumption is out of scope. Table 7 summarizes the peak power prediction accuracy. For C2, the errors are all below 2.36%. For F1, which is two-stage prediction, the errors are larger, but all stay below 6.9%. Hence, the multi-stage method performs well in predicting peak power.

**Table 7.** Peak Power Prediction Accuracy.

| | *Monthly Peak Power* | *Weekly Peak Power* | *Daily Peak Power* |
|---|---|---|---|
| *C2* | 2.36% | 2.36%~2.36%<br>(avg: 2.36%) | 0.73%~2.36%<br>(avg: 1.99%) |
| *F1* | 6.90% | 2.15%~6.90%<br>(avg: 5.34%) | 1.05%~6.90%<br>(avg: 2.42%) |

To further evaluate the fitness of the power curve to the real power curve, the NMBE metric is adopted, which describes the average bias in the model. NMBE is determined with Eq. 6. By definition, it is the sum of error over the sum of the actual values. This metric evaluates the fitness of the model over the whole simulation horizon.

$$NMBE = \frac{\sum_{i=1}^{N}\left(x_{f,i} - x_{o,i}\right)}{N \times \overline{x_o}} \qquad (6)$$

Table 8 summarizes the daily, weekly and monthly NMBE of the lighting power. The lighting power obtained by multiplying the ground truth occupancy data with nominal power is set as the baseline for better comparison. From the table, the two-stage prediction generally has larger errors than the single-stage model. For the single-stage lighting power (C2), the monthly, weekly and daily NMBE are all within 5%, which indicates a high accuracy for power demand predictions. For the two-stage lighting power (F1), the monthly and weekly average errors are within 10%, which is still acceptable. However, we see a big deviation in the daily NMBE, and this leads to a high average value for daily NMBE. This high deviation could have been caused by an uncommon data record on Aug. 19 (see Figure 12) when the lights are only on for a short time period but the model simulated it just as usual.

**Table 8.** NMBE of Lighting Power Prediction.

| | | *Baseline* | *Model* |
|---|---|---|---|
| *Monthly NMBE* | *C2* | 0.061% | 3.92% |
| | *F1* | -0.55% | 8.28% |
| *Weekly NMBE* | *C2* | -0.27%~0.44%<br>(avg: 0.060%) | -0.25%~9.84%<br>(avg: 4.07%) |
| | *F1* | -2.84%~1.30%<br>(avg: -0.68) | 0.33%~20.4%<br>(avg: 7.92%) |
| *Daily NMBE* | *C2* | -0.56%~0.72%<br>(avg: 0.057%) | -2.59%~23.72%<br>(avg: 4.03%) |
| | *F1* | -12.9%~50.9%<br>(avg: 0.39%) | -21.6%~807%<br>(avg: 44.1%) |

Additionally, as the models are simulated in a stochastic manner and the occupant presence was determined every 2 minutes, we see an obvious oscillation in lighting power in Figure 12. This feature of the model leads to that the longer the simulation time, the closer the expectation of the simulation results will be to the actual data. This explains why the model shows a better performance concerning monthly NMBE. However, short-term accuracy of the model still needs some improvement.

## 4 Conclusion

This paper proposed a methodology for occupant presence learning and reproducing based on lighting power metering data. The method was validated against real data. The results show that the proposed multi-stage lighting power prediction method can predict daily peak power with 2.42% relative error. The monthly and weekly NMBE of lighting power are on average below 8.28%.

Through the training and validation process of this work, we found that logistic regression models are sensitive to the quality of the training data. Ideally, the dataset should be more focused on the transitional region (i.e., where the value turns from 0 to 1 or vice versa) of the model and the two classes should be well balanced. Further, increasing the number of independent features should help improve the fitness of the probability model. The stochastic simulation results show that stochastic models can be very accurate for

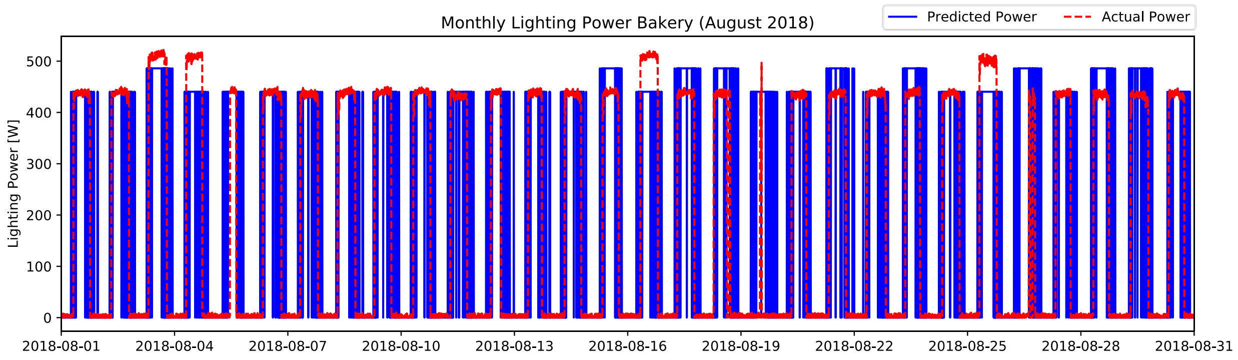


**Figure 12.** Monthly predicted and actual lighting power in F1 bakery.

long-term predictions. However, they cannot predict uncommon events, and this can lead to large short-term prediction errors.

This work has the limitation of not having the ground truth data for occupant presence. The presence generated from lighting power can be delayed when people arrived and did not turn the lights on. This can be cross validated with other appliance usage data in the future. In the best-case scenario, occupant surveys should be conducted to know their preferences and habits, and occupant sensors should be installed.

## Acknowledgements

This research was supported by the U.S. Department of Energy, Energy Efficiency and Renewable Energy, Building Technologies Office, under Contract No. DE-AC05-76RL01830. This research was also supported by the National Science Foundation under Awards No. IIS-1802017 and IIS1633363. BIGDATA: Collaborative Research: IA: Big Data Analytics for Optimized Planning of Smart, Sustainable, and Connected Communities.

## References

ASHRAE. 2002. *ASHRAE Guideline 14-2002: Measurement Of Energy And Demand Savings*.

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. “SMOTE: Synthetic Minority over-Sampling Technique.” *Journal of Artificial Intelligence Research* 16: 321–57.

Hunt, D R G. 1980. “Predicting Artificial Lighting Use-a Method Based upon Observed Patterns of Behaviour.” *Lighting Research & Technology* 12 (1): 7–14.

Kim, Yang-Seon, Mohammad Heidarinejad, Matthew Dahlhausen, and Jelena Srebric. 2017. “Building Energy Model Calibration with Schedules Derived from Electricity Use Data.” *Applied Energy* 190: 997–1007.

Luo, Xuan, Khee Poh Lam, Yixing Chen, and Tianzhen Hong. 2017. “Performance Evaluation of an Agent-Based Occupancy Simulation Model.” *Building and Environment* 115: 42–53.

Pigg, S., Mark Eilers, and John Reed. 1996. “Behavioral Aspects of Lighting and Occupancy Sensors in Private Offices: A Case Study of a University Office Building.” *ACEEE 1996 Summer Study on Energy Efficiency in Buildings*, no. 8: 161–70.

Reinhart, C F, and K Voss. 2003. “Monitoring Manual Control of Electric Lighting and Blinds.” *Lighting Research & Technology* 35 (3): 243–58. https://doi.org/10.1191/1365782803li064oa.

Wang, Danni, Clifford C. Federspiel, and Francis Rubinstein. 2005. “Modeling Occupancy in Single Person Offices.” *Energy and Buildings* 37 (2): 121–26. https://doi.org/10.1016/j.enbuild.2004.06.015.

Wang, Jing, Wangda Zuo, Landolf Rhode-Barbarigos, Xing Lu, Jianhui Wang, and Yanling Lin. 2018. “Literature Review on Modeling and Simulation of Energy Infrastructures from a Resilience Perspective.” *Reliability Engineering & System Safety*.

Wetter, Michael, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. 2014. “Modelica Buildings Library.” *Journal of Building Performance Simulation* 7 (4): 253–70.

Wu, D, H Hao, T Fu, and K Kalsi. 2018. “Regional Assessment of Virtual Battery Potential from Building Loads.” In *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, 1–5. https://doi.org/10.1109/TDC.2018.8440225.

Zhao, L, W Zhang, H Hao, and K Kalsi. 2017. “A Geometric Approach to Aggregate Flexibility Modeling of Thermostatically Controlled Loads.” *IEEE Transactions on Power Systems* 32 (6): 4721–31. https://doi.org/10.1109/TPWRS.2017.2674699.

# Development and Verification of Control Sequences for Single-Zone Variable Air Volume System Based on ASHRAE Guideline 36

Kun Zhang, David H. Blum, Milica Grahovac, Jianjun Hu, Jessica Granderson, Michael Wetter
Building Technology and Urban Systems Division
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
{kunzhang,dhblum,mgrahovac,jianjunhu,jgranderson,mwetter}@lbl.gov

## Abstract

This paper presents work on the development and verification of ASHRAE Guideline 36-2018 control sequences for single-zone variable air volume air-handling unit (AHU) systems. The Control Description Language, a subset of the Modelica Language, is used to implement those advanced control sequences. The sequences address control for components such as the economizer, supply air temperature setpoint reset, fan speed control, and zone heating/cooling states determination. Each component sequence is validated in open-loop tests and then used to compose a single comprehensive controller. This controller is also first validated in open loop and then tested in closed loop with an AHU system and building envelope model constructed using the Modelica Buildings library. The Guideline 36 controller is compared with a conventional control strategy applied to the same AHU and building model. Annual simulations show that the Guideline 36 control sequences yield 17.3 % of annual HVAC energy savings against the conventional control strategy in this case study.

*Keywords: Control, VAV, ASHRAE Guideline 36, Buildings, HVAC*

## 1 Introduction

The Heating, Ventilation and Air Conditioning (HVAC) control industry has not yet had a standard for expressing control sequences of HVAC systems (Pang et al, 2017). The controllers in the market can be generally divided into two types: configurable and programmable. The first type of controller is pre-programmed; it is therefore easy to install and commission. However, the embedded control logic is often overly simplistic, resulting in a compromise of thermal comfort and energy efficiency required by evolving energy standards and building codes. The second type is fully programmable (Hydeman, Taylor, & Eubanks, 2015). Yet, due to a lack of standard high-efficiency sequences, the implemented control scheme of the HVAC system is often project-specific. Therefore, significant resources are required to engineer, specify, program and commission each project. It is also common that the implemented control sequences are sub-optimal and error-prone, which leads to varied building operational efficiency and performance (Hydeman et al, 2015).

The American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) has initiated projects related to high-performance control sequences for HVAC systems through its Research Projects 1455 (Taylor Engineering, 2014) and 1711. A first version of ASHRAE Guideline 36-2018 (G36) (ASHRAE, 2018) was published upon completion of the Project 1455. The control sequences included in the G36 are based on the best-in-class industry practices. The guideline aims at reducing energy consumption and improving thermal comfort and indoor air quality of buildings. It also provides potential to reduce the time of the engineering, specification, programming and commissioning process (ASHRAE, 2018).

Implementing the advanced control scheme as described in the Guideline 36 does present its own challenges, due to the complexity of the sequences. The English language description can be ambiguous, and its interpretation to implement the sequences in a programming language is not straight-forward.

The G36 2018 version includes control sequences for the air distribution for single-zone and multi-zone variable air volume systems. The multi-zone system has been implemented in the Control Description Language (CDL) and reported in (Wetter et al, 2018). This paper focuses on the implementation of the G36 control sequences for Single Zone Variable Air Volume (SZVAV) systems using CDL (Wetter et al, 2018a). CDL is a subset of Modelica with its own set of data types and elementary blocks. It intends to allow for implementation of control sequences in computer code that can be used in real buildings, assessed through explicit simulation, and reused for verification tests during the commissioning process. CDL was developed under the OpenBuildingControl (OBC) project (Wetter et al, 2018b). Both CDL and the implemented sequences for OBC are incorporated in the master branch of the Modelica Buildings library version 7.0.0 (Wetter et al, 2014). Another closely-related project called “Spawn of EnergyPlus” (Wetter et al, 2015) aims at enhancing the EnergyPlus simulation engine (EnergyPlus Development Team, 2019) by integrating the Modelica Buildings library so it can simulate these sequences using EnergyPlus envelope models and Modelica HVAC and control models.

Related work has indicated the potential of control strategies of HVAC systems to impact their energy consumption. For example, Pang et al. (2017) found that energy consumption can vary up to 66% from various control strategies for multi-zone VAV systems, while Fernandez et al. (2017) found that this variation could be up to 60% for various HVAC control and commissioning cases. For the multi-zone G36 sequence, Wetter et al. (2018a) found potential energy savings of up to 30%.

As of writing this paper, the authors are unaware of publications related to implementing the G36 SZVAV control sequences in a programming language and evaluating the performance of those advanced sequences in a simulation environment. Therefore, the objective of this paper is to describe the process of implementing and verifying the control sequences using CDL, as well as the performance evaluation of the sequences compared with a conventional controller. Then, the sequences will be available for further performance evaluation and eventual real building implementation through the process developed in the OBC project.

Section 2 gives an overview of the control sequences and their implementation with the main components of the sequences described in the subsections separately. Section 3 presents the case study, where a baseline controller is also developed to help evaluate the performance of the G36 control sequences and emphasize key advantages over those found in practice. Section 4 presents the comparison results with analysis. The paper closes with discussion and conclusion.

## 2 Guideline 36 control sequences of single-zone VAV system

A SZVAV air handling unit (AHU) system is often applied to medium to large single-floor spaces such as small retail stores, classrooms, and auditoriums. It is usually composed of a variable-speed supply fan (additionally with or without an exhaust fan), a cooling coil, a heating coil and a mixing box for controlling the ratio of recirculated zone return air and outside air (see Figure 1). The delivery of additional outside air when conditions are appropriate is called economizer operation. Control sequences of the SZVAV system in the G36 specification include supply fan speed control, supply air temperature control, minimum outdoor air control, economizer control, zone state, freeze protection and alarms.

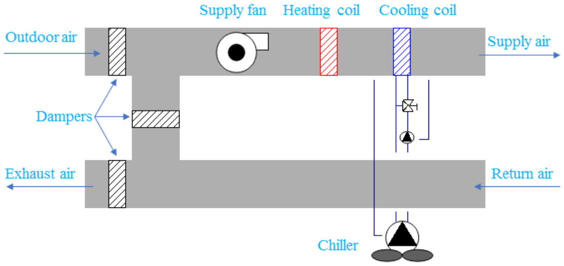


**Figure 1.** Single-zone VAV air handling unit system.

The goal of the G36 control strategy is to maximize free cooling and to avoid excess energy consumption to run fans and provide mechanical heating and cooling. The essential ideas of the strategy are to vary the fan speed, reset the supply air temperature setpoints (both heating and cooling) in different conditions, and adjust the outdoor air damper position according to the supply fan speed.

Elementary CDL blocks were used to implement composite blocks representing subsets of the control sequences. These composite blocks were then integrated into a single controller block. Figure 2 shows the tree view of the single-zone VAV control sequences package in the Modelica Buildings library version 7.0.0.

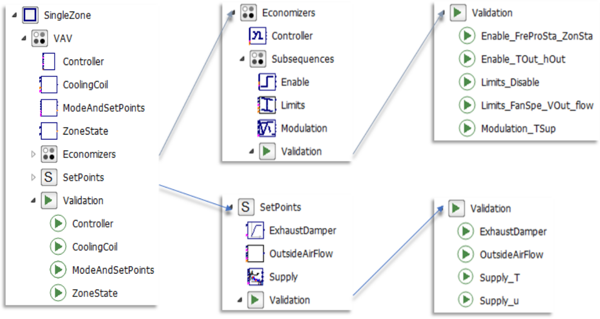


**Figure 2.** Structure of the single zone VAV package in the Modelica Buildings library 7.0.0.

The implementation of the sequences is modular. It allows users to customize the sequences for their needs primarily by simple parameter selection, such as whether there exists an enthalpy sensor to direct economizer operation, but also through easy access to certain parts of the control sequences. The CDL implementation also provides information on the control objective and functionality of each control block in the form of html info sections.

The implementation also considered numerical integration error and/or sensor noise which may cause chattering of the control, though this is beyond the scope of the G36 itself. CDL blocks for hysteresis or timers were therefore added for the part of the control sequences that use continuous-time semantics.

Each composite block in the package is validated in open-loop simulations. As can be seen from Figure 2, each sub-package includes a Validation package. The validation models are not only to verify whether the control sequences satisfy the control intent under a wide range of preset input conditions, but also intended to provide utilization examples for the library users.

The following sections explain the key components of the control sequences and how they were implemented and validated in the Buildings library.

## 2.1 Setpoints for supply air temperatures and fan speed

There are two separate supply air temperature (SAT) setpoints in the G36 sequences: 1) SAT for heating, which is used to control the heating coil and economizer dampers, and 2) SAT for cooling, which is used to control the cooling coil. The two temperature setpoints are reset at different rates but controlled using the same temperature sensor. The supply fan speed is also reset using the same control loops as the SAT for heating and cooling. These two control loops correct for the error between measured zone air temperature and the heating and cooling temperature setpoints respectively.

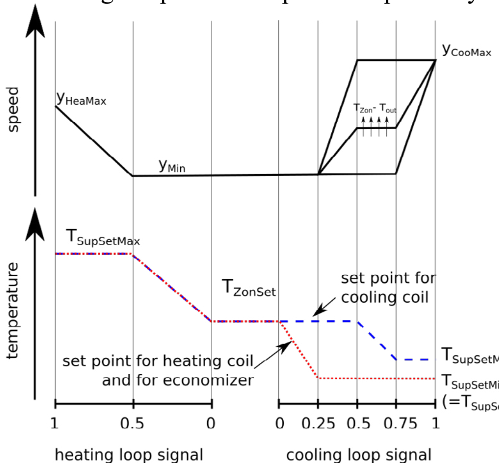


**Figure 3.** Control diagram for single zone VAV control logic: fan speed and supply air temperature setpoints as a function of control loop signals.

Figure 3 shows the G36 control diagram of the heating and cooling SAT setpoints and fan speed trajectories under heating and cooling control loop signals. The control logic requires that the heating SAT setpoint $T_{SupSet}$ increases linearly when the heating coil valve control signal, i.e. the heating loop signal, increases from 0 to 0.5; and stays at the maximum heating SAT $T_{SupSetMax}$ when the heating signal is within 0.5 and 1 (see the red dotted curve). When the system is in the deadband (neither heating nor cooling state), the heating SAT is the same as the zone temperature setpoint $T_{ZonSet}$ and the fan speed remains at the minimum $y_{min}$. The fan speed stays at $y_{min}$ when the heating loop signal is within 0 and 0.5; and it increases linearly to the maximum fan speed $y_{HeaMax}$ when the heating loop signal is within 0.5 and 1 (see the solid black curve in the left part of the upper plot).

In cooling mode, the SAT setpoint (see the blue dotted curve) is reset in a similar linear modulation logic as the heating SAT setpoint. The fan speed is varied continuously based on the difference between inside and outside air temperatures and cooling loop signal, as shown in the upper right portion of Figure 3.

### 2.1.1 Implementation in CDL

The block Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.SetPoints.Supply implements the functionalities of the SAT reset and supply fan speed control as indicated in the G36. The block allows users to set the maximum SAT setpoint for heating and the minimum SAT setpoint for cooling. For the fan speed control, the parameters that can be changed are the maximum fan speed for heating, and the minimum and maximum fan speeds for cooling.

### 2.1.2 Open-loop verification

Figure 4 presents the fan speed control validation results as a function of the cooling control loop signal from the validation simulation of the VAV controller. In the validation model, instances of the controller are configured identically, but the input signal for zone temperature differs in order to validate that the fan speed is increased correctly. It can be seen that Figure 4 is a representation of the upper right part of Figure 3 as required in the G36.

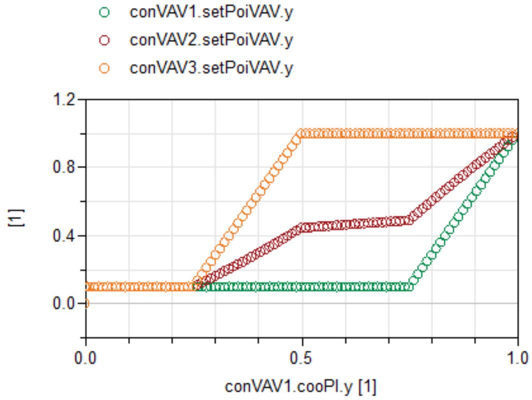


**Figure 4.** Fan speed control as a function of the cooling control loop signal.

Similarly, Figure 5 presents the lower right part of Figure 3. It shows how the SAT setpoint for heating and economizer, and the SAT setpoint for cooling are modulated in different cooling control signals. Validation models such as these confirm that our controller implementation behaves according to the prescribed control sequences of the G36.

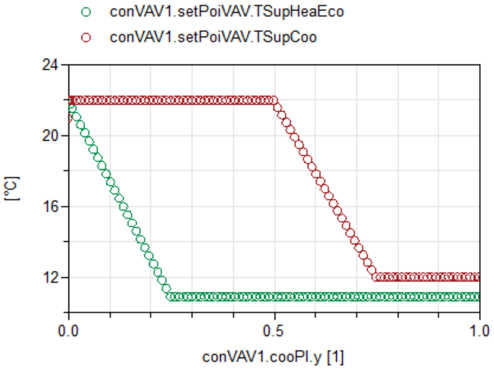


**Figure 5.** Heating and cooling supply air temperature as a function of the cooling control loop signal.

## 2.2 Economizer

The single-zone AHU economizer control according to the G36 comprises the SAT, outdoor air (OA) damper and economizer lockout control.

The economizer SAT control loop has an intent to maintain the SAT at its heating setpoint by modulating the heating coil and both OA and return air (RA) damper positions. The dampers are complementary, meaning that a single actuator controls both dampers.

The logic assumes a single OA damper for both the economizer and the minimum OA functionality. The minimum OA damper position, which aims at satisfying the outdoor airflow requirement, is reset based on both current outdoor airflow requirement and fan speed.

The economizer is locked out (CDL implementation uses the term disabled) as the outdoor air condition, i.e., dry bulb temperature and/or enthalpy depending on the sensors installed, exceeds the climate and energy code specific setpoints.

### 2.2.1 Implementation in CDL

The package Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.Economizers, as illustrated in the upper middle part of Figure 2, provides a CDL implementation of the SZ G36 economizer. It comprises a main economizer Controller block, a package with three subsequences: Limits, Enable and Modulation, and the corresponding validation models for both the Controller block and the subsequences. The Limits sequence implements the minimum OA damper position reset. The Enable sequence resets the OA and RA damper position limits based on the economizer lockout conditions, for example outdoor air temperature, and equipment and building status, such as fan enable status and whether the zone requires any heating or cooling. The Modulation sequence implements the SAT control. When the economizer is disabled, the modulation sequence keeps evaluating the SAT control loop, but the RA damper position is fixed to a fully open position and the OA damper is fixed to the minimum open position to meet outdoor airflow, as specified by the outputs of the Enable sequence.

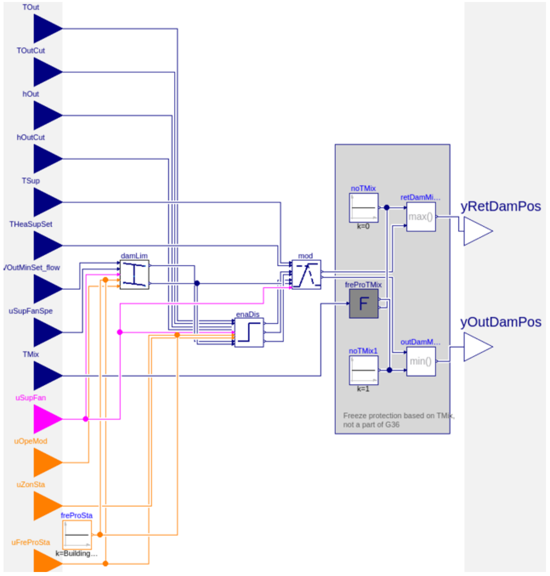


**Figure 6.** Block diagram of the CDL implementation of the Economizer controller in the Buildings library. It comprises four subsequences: Enable, Limits, Modulation and Freeze Protection**.**

The economizer Controller block is illustrated in Figure 6. The block takes as inputs the setpoints, such as the SAT and its heating setpoint, minimum outdoor airflow setpoint, measured quantities such as outdoor air temperature and/or enthalpy, supply air fan speed, mixed air temperature, status variables such as the supply fan on/off status, operating mode, zone state and freeze protection status. The block outputs the OA and RA damper positions to be sent to actuators. In addition to the G36 definition the controller implements a custom freeze protection block based on the mixed air temperature tracking for reasons elaborated in (Wetter et al, 2018).

### 2.2.2 Open-loop verification

All control blocks contained in the Economizer package, including the top-level Controller block and the subsequences, have at least one corresponding validation model. Here we present a validation model of the Modulation block.

Figure 7 shows the validation results of the Modulation block performance. The plot (b) shows the PI controller output signal over time when the SAT rises from below to above its setpoint, as shown in plot (a). With these inputs the control signal drops monotonically from 1 to 0. Plot (c) illustrates how the OA and RA damper and cooling valve positions behave as a function of the SAT control loop signal. The heating coil signal, which rises from 0 to 1 at the far-right side of plot (c), is mapped to the SAT control signal such that only the upper portion of the signal, starting at a value at which the OA damper is fully closed and the RA damper is fully open, is used for the heating valve control.

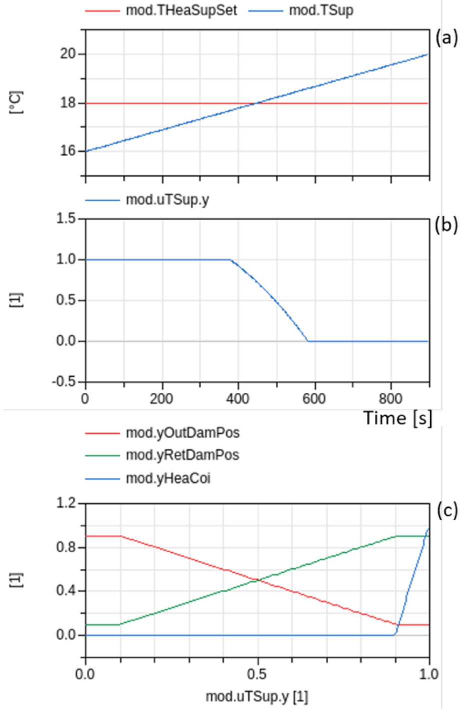


**Figure 7.** Modulation block validation results for a simulation duration of 15 minutes show the time series of (a) test values of SAT (in blue) and SAT setpoint (in red) that yield the (b) PI controller signal as a response. (c) Control diagram with OA (in red), RA damper (in green) and cooling coil valve (in blue) positions as a function of the SAT control signal.

## 2.3 Outdoor airflow control

The control of minimum outdoor airflow rate setpoint complies with the ventilation rate procedure of ASHRAE 62.1 (ASHRAE, 2016). It adjusts the setpoint according to the zone operation mode, zone status (heating, cooling or standby), and window status if it has any operable window.

### 2.3.1 Implementation in CDL

The block Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.SetPoints.OutsideAirFlow outputs the minimum outdoor airflow rate setpoint as specified in the G36. Figure 8 shows the implementation of the block in the Buildings library.

There are three steps to specify the setpoint. First, it finds the minimum breathing zone outdoor airflow rate, which is the sum of the rate specified according to area and the rate specified according to occupant population. The number of occupants could be retrieved directly from an occupancy sensor, if present. Otherwise, the default occupant density is used to calculate the outdoor air requirement according to (ASHRAE, 2016). Second, the sequence selects warm-air or cool-air distribution effectiveness depending on the zone heating or cooling status, as specified in ASHRAE 62.1. Finally, it sets the minimum outdoor airflow setpoint for the zone when it is in occupied mode with the window (if there is one) being closed. When the zone is not in occupied modes or the window is open, the setpoint becomes zero.

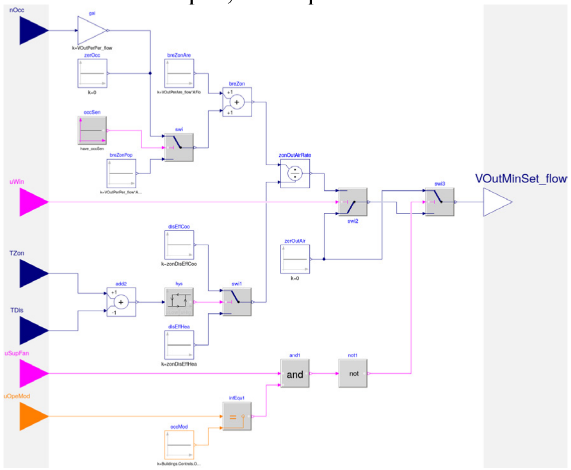


**Figure 8.** Block diagram of the CDL implementation of specifying minimum outdoor airflow setpoint.

### 2.3.2 Open-loop verification

Figure 9 shows the results of validating the sequence by giving the inputs of increasing occupancy and the change of zone state from heating to cooling. It illustrates that the minimum output airflow setpoint increases when there are more occupants in the zone. Also, the sequence can choose different air distribution effectiveness depending on the zone state. The zone state is decided based on the temperature difference between the zone and the supply air, with a hysteresis being applied to avoid chattering.

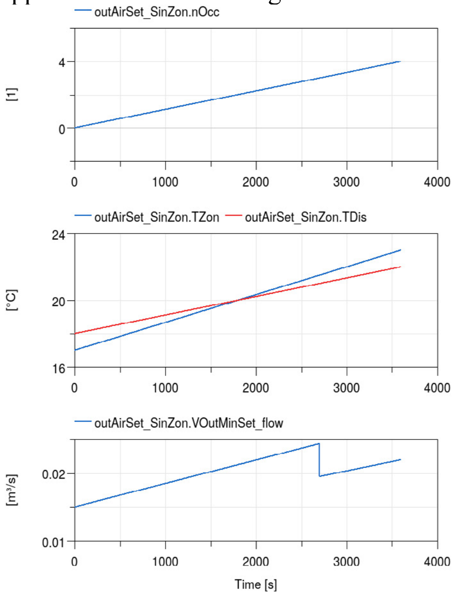


**Figure 9.** Validation results for block of specifying the minimum outdoor airflow rate setpoint show that the setpoint changes along with the changes of occupancy and

zone state (at the time 2700s the zone state is changed from heating to cooling).

## 3 Case study

To test the controller in a closed-loop scenario, a model is created to integrate the controller with a SZVAV AHU system and a single-zone building envelope model. Measurements of the building air temperature, supply air temperature, return air temperature, and mixed air temperature are fed back to the controller to close the control loop (see Figure 10). Other important parts of the model include the weather data and occupancy schedules.

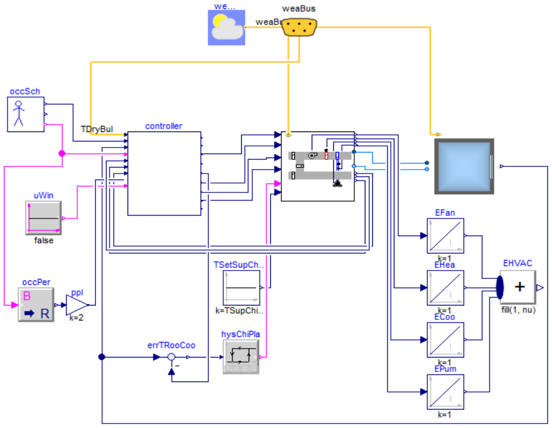


**Figure 10.** Closed-loop control model with the building, AHU and G36 controller.

The details of the building and AHU model are illustrated in the subsections below, and the performance of the G36 controller is compared with a conventional Baseline controller in Section 4.

### 3.1 Building envelope model

The building envelope model used for this case study is from the model Buildings.Air.Systems.SingleZone.VAV.Example.BaseClasses.Room. It uses an instance of Buildings.ThermalZones.Detailed.MixedAir to model the transient heat conduction within the building constructions and longwave radiation heat exchange between the surfaces (walls, roof and windows etc.). The heat convection and radiation between the ambient (indoor and outdoor air) and the envelope is also modeled at each time step.

The information of the envelope such as geometry and materials are from the BESTEST case 600, and derived from the EnergyPlus validation project (Henninger & Witte, 2004).

The weather data used in the case study is the DRYCOLD weather data included in the Buildings library, which is the weather used for the BESTEST case studies. It is from the weather station of Denver-Stapleton in Colorado, USA. The occupancy schedule for the building is assumed to be from 8am to 6pm daily, which means that the system is operated based on this schedule. The internal heat gains are modelled as constant gains when the zone is occupied.

### 3.2 Air handling unit system model

The air handling unit system model from the class Buildings.Air.Systems.SingleZone.VAV.ChillerDXHeatingEconomizer contains a variable-speed supply fan, a heating coil, a water-based cooling coil, and an economizer. The cooling coil is assumed to be served by an air-cooled chiller. The model assumes that pressure drops through the system are lumped into a single component and the cooling coil is a dry coil. The mass flow of chilled water through the cooling coil is controlled by a three-way valve to maintain the cooling supply air temperature setpoint. The cooling coil mixing valve and the economizer dampers are modeled as ideal, i.e., they exactly control a specified ratio of fluid flow through contributing branches.

The fan and pump models are idealized to exactly track the set point for the mass flow rate, and they are from the model Buildings.Fluid.Movers.FlowControlled_m_flow. The details about the fan/pump model are described in (Wetter, 2013).

The design airflow rate for the AHU system is 0.625 $m^3/s$. The minimum outdoor airflow rate is 0.0144 $m^3/s$ and the design outdoor airflow rate is 0.025 $m^3/s$. The calculation of the ventilation requirement for the building model is based on ASHRAE Standard 62.1 (ASHRAE, 2016) for an office with reference occupancy density. Note that the G36 controller is capable to adjust the outdoor airflow rate between the minimum and design outdoor airflow based on whether there are occupants in the zone; while the Baseline controller is configured to provide the design outdoor airflow.

The chiller model is Buildings.Fluid.Chillers.ElectricEIR. It is a model of an electric chiller, based on the DOE-2.1 chiller model and the EnergyPlus chiller model Chiller:Electric:EIR (Hydeman et al, 2002). Its nominal Coefficient of Performance (COP) is 5.5. The heating plant is not modelled and we assume it is a geothermal heat pump with a constant COP of 4.0.

### 3.3 Baseline controller model

The Baseline controller is based on the commonly used single-maximum VAV control with dry-bulb economizer control. During cooling, the fan speed is controlled to maintain the room temperature at the cooling setpoint temperature using a P controller, between a minimum and maximum fan speed. Flow through the cooling coil is controlled to maintain a constant supply air temperature setpoint. During heating, the fan speed is constant at the minimum speed while the heating coil is controlled to maintain the room temperature at the heating setpoint using a P controller. The minimum position of the outdoor air damper ensures enough ventilation flow to meet ASHRAE 62.1

at minimum fan speed. If the outside air dry-bulb temperature is lower than the return air dry-bulb temperature, the economizer opens the damper further to provide cooling of the mixed air to the supply air temperature setpoint as much as possible. During unoccupied times, the zone heating and cooling setpoints are set back and the minimum outdoor air damper position is set to zero.

## 4 Results comparison

The performance of these G36 and base controllers was compared using identical models for the building envelope, AHU system and weather. Overall, the G36 controller saves 17.3 % HVAC electric energy compared with the Baseline case. The heating energy use for both controllers is nearly equal, with most of the energy savings of the G36 controller associated with the cooling energy. The pump electricity use is minimal for both cases and the G36 controller uses slightly more electricity for the supply fan. Figure 11 shows the breakdown of the monthly energy use for the two controllers with left bars indicating the Baseline controller and right bars indicating the G36 controller. It can be clearly seen that G36 requires less energy use for cooling throughout all the months. In the winter months (December, January and February) G36 consumes 2.6% more heating energy than the Baseline. This small increase could be due to two factors. The first is when near the end of an occupancy period the internal loads are decreasing and the zone switches from cooling mode to deadband. This mode switch increases the SAT setpoint according to Figure 3. For the remaining time the fan is supplying outside air, and the temperature is low outside, heating is briefly used to heat the supply air to the setpoint. This is shown in Figure 14. The second factor is the small amount of increased outside air the G36 control sequence provides during morning heat up, as shown in the upper plot of Figure 13, which adds a small amount of heating load to the coil.

Figure 12 shows the zone temperature profiles during a winter and a summer week along with the zone heating and cooling setpoint. We can see that the zone temperatures are maintained within the heating and cooling setpoint bands by both controllers during these two extreme weeks. In addition, we can see that the zone temperatures are very close to each other in both cases. We actually find that both controllers deliver very similar zone temperatures all year around, with temperature difference within 0.5 K, the same magnitude as the temperature hysteresis settings in the controllers. Using the zone temperature as the thermal comfort indicator, we can conclude that both controllers maintain the thermal comfort in the zone equally close. This means that the G36 controller does not compromise thermal comfort while yielding energy savings.

Figure 13 shows the outdoor airflow during the same two weeks. We can see that the outdoor airflow profiles of the two cases are very similar to each other in winter. During this winter week, the outdoor air temperature is very low as shown by the lime curve in Figure 12, so the controllers restrict the outdoor air fraction to the minimum required for ventilation during heating, as seen in the mornings of each day and use the economizer if any cooling is needed, as seen during the other afternoons in the week. In the summer week, the G36 is capable of lowering the outdoor airflow rate by adjusting the minimum outdoor air damper position based on the fan speed. This reduces excess load on the cooling coil when the outdoor air temperature is higher than the zone temperature. As the Baseline controller assumes no active reset on the minimum outdoor air damper position, excess outdoor air is brought in when the fan speed increases for space cooling.

In Figure 13 we also find that there are sudden jumps in the outdoor airflow profiles, for example, on the afternoon of August 1st (more significant airflow increases for the G36 controller). During that period, the outdoor air temperature becomes lower than the zone temperature setpoint (see the lower plot of Figure 12); both controllers therefore increase the OA damper opening to use more outdoor air to cool down the building. However, the G36 controller simultaneously resets the cooling SAT setpoint up (see the green curve in Figure 15). This results in an increase of the supply airflow rate in order to meet the zone cooling load; however, the G36 controller does so by use of more outside air and without use of any mechanical cooling. On the other hand, the Baseline controller maintains a constant SAT for cooling (see the blue curve in Figure 15), so mechanical cooling is still required to reach the

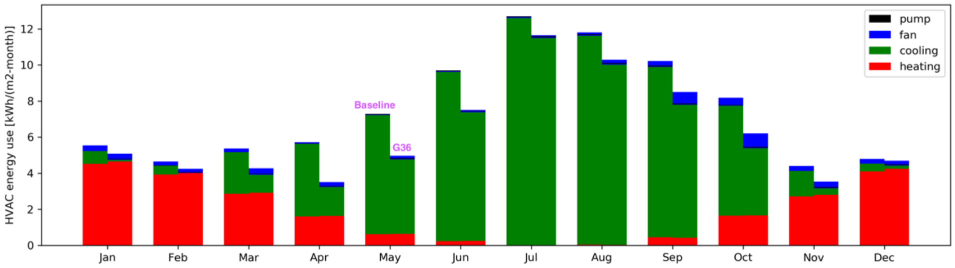


**Figure 11.** Site HVAC electricity use for each month (Left bars: Baseline; Right bars: G36).

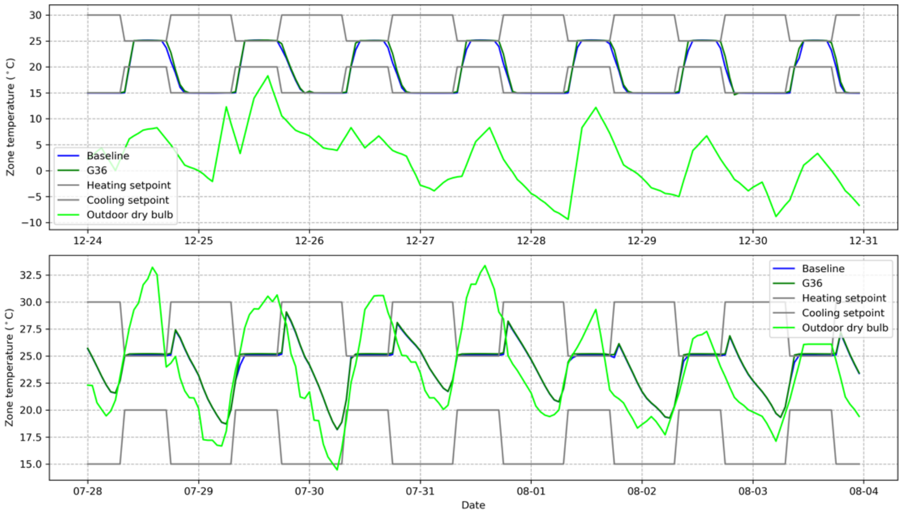


**Figure 12.** Zone temperature profiles during a winter (top) and a summer (bottom) week

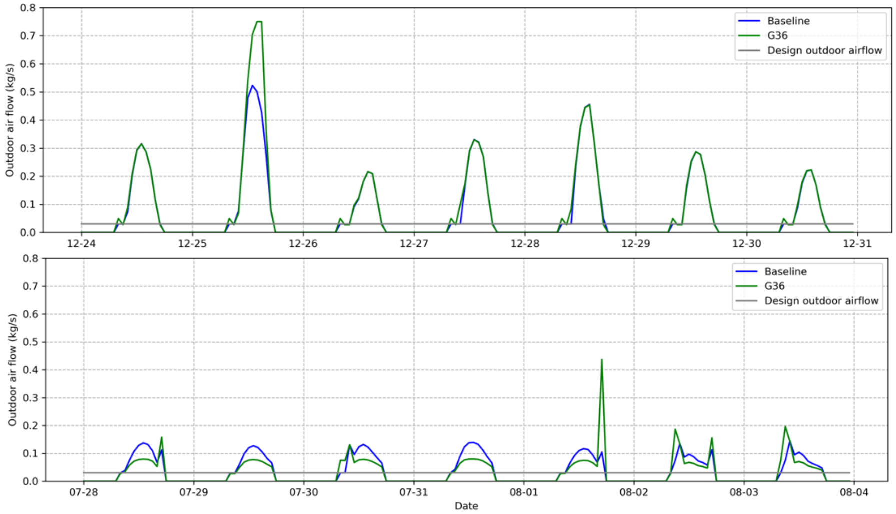


**Figure 13.** Outdoor airflow rate during a winter (top) and a summer (bottom) week.

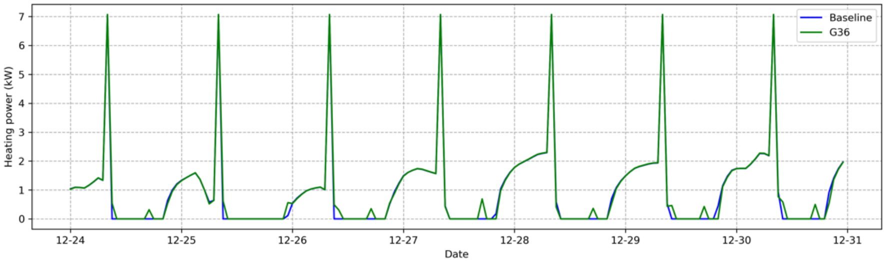


**Figure 14.** Heating power demand during a winter week.

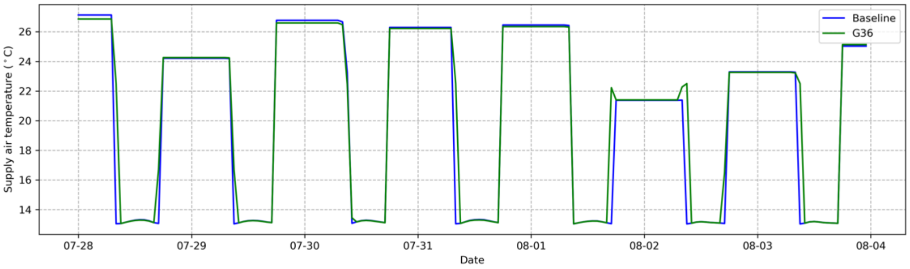


**Figure 15.** Supply air temperature for cooling in a summer week.

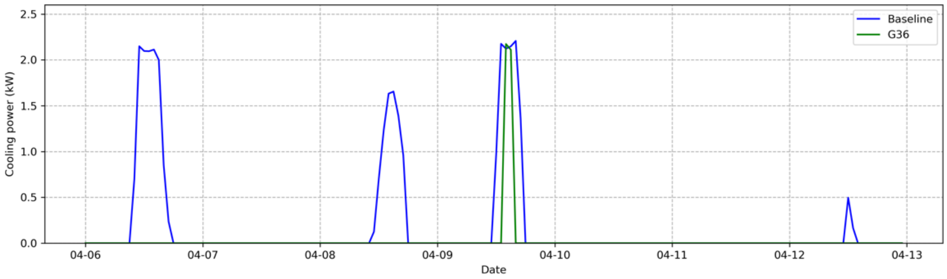


**Figure 16.** Cooling power demand in a week of shoulder season.

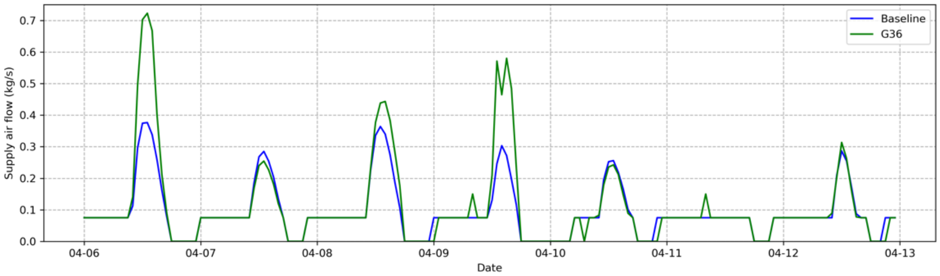


**Figure 17.** Supply airflow rate in a week of shoulder season.

lower cooling SAT setpoint, even though the economizer is enabled. This shows how the advanced control sequences of the G36 take even more advantage of available free cooling by coordinating the SAT setpoints reset and the economizer operation. This strategy of coordinating the SAT setpoint reset and the economizer operation is the main reason why the G36 controller consumes less cooling energy than the Baseline. The strategy is particularly useful during shoulder seasons. Figure 16 shows the cooling power demand in a week of the shoulder season. We can see that the G36 uses much less cooling energy than the Baseline for this week. Figure 17 shows the supply airflow rate in the same week of the shoulder season as in Figure 16. We can see that the G36 controller has higher supply airflow than the Baseline for the whole week. This is because the G36 controller engages the economizer more often to increase the outdoor airflow to utilize free cooling than the Baseline. This explains why the G36 does not save fan energy as shown in Figure 11.

Finally, it should be noted that the simulation time for each controller was similar, with the Baseline controller at 507 seconds and the G36 at 526 seconds. The simulations were run on a Linux operating system with a 16-core processor (Intel Xeon® CPU X5650 @2.67GHz) and a 32GB memory. In general, the simulation time with different control strategies can be largely dependent on the number of events generated through mode or on/off switching.

## 5 Discussions and conclusions

This paper presented the work on implementation, validation and application of ASHRAE Guideline 36-2018 control sequences for single-zone variable air volume air-handling unit systems. Those advanced control sequences address control for AHU system components such as the economizer, supply air temperature setpoints reset, fan speed control, and zone heating and cooling states.

The control sequences were implemented using the Control Description Language in a modularized approach, which therefore allows the users to customize the sequences for their needs. Each component sequence was validated in open-loop tests and then used to compose a single comprehensive controller. This controller was firstly validated in open loop and then tested in closed loop with an AHU system and building envelope model constructed using the Modelica Buildings library.

The Guideline 36 control sequences were compared with the conventional control strategy based on single-maximum VAV control. Both controllers were applied to the same AHU and building system in a case study.

Annual simulations show that the Guideline 36 control sequences yield 17.3 % of annual HVAC energy savings against the conventional control strategy. The G36 control scheme can take advantage of free cooling by adjusting the economizer dampers and resetting supply air temperature setpoints; the G36 controller therefore has reduced energy consumption due to cooling. Verification of annual zone temperatures show that both controllers maintain the zone temperature very closely to each other within thermal comfort bands. This shows that the energy savings of the G36 control sequences do not compromise thermal comfort while delivering the energy savings.

It should be noted that the percentage of energy savings shown by the G36 controller in this paper is specific to the selected case study. The energy savings potential is subject to variables such as climate zones, internal heat gains assumption and the baseline control sequences. Future work of this study includes investigating the impact of those variables on the G36 control sequences performance. Validation of the control sequences with measurement data is also important to further verify the implementation of the sequences.

## Data availability

All the models and components used in this paper are open-source and can be downloaded from the Github repository https://github.com/lbl-srg/modelica-buildings. The Modelica Buildings branch for the models used in this study is issue1608_compareSZVAV (commit 7c939c0). Table 1 lists the Modelica path of the two closed-loop system models in the case study and the SZVAV package in the Buildings library.

## Acknowledgements

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231, and the California Energy Commission's Electric Program Investment Charge (EPIC) Program. We thank Taylor Engineering for their funding of this work through the California Energy Commission's (CEC) Electric Program Investment Charge (EPIC) Program.

**Table 1. Models and package used in the paper from the open-source Modelica Buildings library**

| *Name* | *Modelica Path* |
|---|---|
| Baseline system model | Buildings.Air.Systems.SingleZone.VAV.Examples.ChillerDXHeatingEconomizer.mo |
| G36 system model | Buildings.Air.Systems.SingleZone.VAV.Examples.Guideline36.mo |
| SZVAV package | Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV |

## References

ASHRAE. (2016). *Ventilation for acceptable indoor air quality*. *ASHRAE Standard*. Atlanta, GA: American Society of Heating Refrigeration and Air-Conditioning Engineers, Inc.

ASHRAE. (2018). *ASHRAE Guideline 36-2018 High-Performance Sequences of Operation for HVAC Systems* (Vol. 8400). Atlanta, GA: American Society of Heating Refrigeration and Air-Conditioning Engineers, Inc.

EnergyPlus Development Team. (2019). *EnergyPlus engineering reference: The reference to EnergyPlus calculations*. EnergyPlus Version 9.2. US Department of Energy.

Fernandez, N., Xie, Y., Katipamula, S., Zhao, M., Wang, W., & Corbin, C. (2017). *Impacts of Commercial Building Controls on Energy Savings and Peak Load Reduction*. Richland, WA: Pacific Northwest National Laboratory Technical Report 25985.

Henninger, R. H., & Witte, M. J. (2004). *EnergyPlus testing with ANSI/ASHRAE standard 140-2001 (BESTEST)*. Washington, D.C.: U.S. Department of Energy.

Hydeman, M., Gillespie, K. L., & Dexter, A. L. (2002). Tools and techniques to calibrate electric chiller component models. *ASHRAE Transactions*, *108 PART 1*, 733–741.

Hydeman, M., Taylor, S. T., & Eubanks, B. (2015). Control sequences & controller programming. *ASHRAE Journal*, *57*(3), 58–62.

Pang, X., Piette, M. A., & Zhou, N. (2017). Characterizing variations in variable air volume system controls. *Energy and Buildings*, *135*, 166–175. https://doi.org/10.1016/j.enbuild.2016.11.031

Taylor Engineering. (2014). *ASHRAE RP-1455: Advanced Control Sequences for HVAC Systems Phase I.* Alameda, CA: Taylor Engineering.

Wetter, M. (2013). Fan and pump model that has a unique solution for any pressure boundary condition and control signal. In *Proceedings of BS 2013: 13th Conference of the International Building Performance Simulation Association, Chambery, France, August 26-28* (pp. 3505–3512).

Wetter, M., Grahovac, M., & Hu, J. (2018a). Control Description Language. *Proceedings of The American Modelica Conference 2018, October 9-10, Somberg Conference Center, Cambridge MA, USA*, *154*, 17–26. https://doi.org/10.3384/ecp1815417

Wetter, M., Hu, J., Grahovac, M., Eubanks, B., & Haves, P. (2018b). OpenBuildingControl: Modelling Feedback Control as a Step Towards Formal Design, Specification, Deployment and Verification of Building Control Sequences. In *2018 Building Performance Modeling Conference and SimBuild, Chicago, IL, September 26-18, 2018* (pp. 775–782).

Wetter, M., Nouidui, T. S., Lorenzetti, D., Lee, E. A., & Roth, A. (2015). Prototyping the next generation energyplus simulation engine. *Proceedings of Building Simulation 2015:14th Conference of International Building Performance Simulation Association, Hyderabad, India, December 7-9*, 403–410.

Wetter, M., Zuo, W., Nouidui, T. S., & Pang, X. (2014). Modelica Buildings library. *Journal of Building Performance Simulation*, *102*(1), 253–270. https://doi.org/10.1080/19401493.2013.765506

# Modelica Component Models for Oceanic Surface Waves and Depth Varying Current

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 theory of progressive ocean-surface gravity-waves is discussed, followed by the concept of the representation of the irregular sea-state by a sea-spectrum. Fourier series decomposition of the irregular sea-surface into its constituent regular waves and the method of realizing unique time-records of the sea-surface-elevation from commonly used sea-spectra is described. A detailed description of the development of *Modelica* component-models to generate regular as well as irregular waves, and depth-varying current, with an eye on the requirements imposed by probable integrated simulation scenarios, is then presented and the results discussed.

*Keywords: regular wave, irregular wave, sea-spectrum, Modelica ocean-engineering library.*

## 1 Introduction

The advantages of developing an *OpenModelica* ocean engineering library populated with domain-specific *component-models* and *functions* to carry out the integrated simulation of multi-pyhsical ocean engineering systems was demonstrated by the authors (Viswanathan and Holden, 2019). This earlier work:

1. Gives a brief description of the simulation of systems based on the hydrodynamic response of catenary-moored non-diffracting floating objects in the presence of waves and current,

2. Demonstrates the satisfactory agreement of the *Modelica* simulation results with those obtained using a popular ocean-engineering commercial software (*Orcaflex*), and

3. Brings out the advantages of using a component-model based simulation approach.

The voluminous nature of the earlier work precluded the possibility of delving into the theoretical and implementational details of the various *Modelica* component-models of the ocean-engineering library proposed by the authors, the preliminary version of which is available for download at `github.com/Savin-Viswanathan/OELib_OMAE2019`.

The present work which deals with the development of *Modelica* component-models for simulating the kinematics and dynamics of regular and irregular waves, and depth varying current, is the first among a series of two papers which will fill in such gaps in theory and implementation.

## 2 Theory

The theory presented here upto Section 2.4.2 is a brief summary of that given in (Dean and Dalrymple, 2001) .

### 2.1 The Fundamentals

The application of the conservation of mass to a reference fluid volume yields the continuity equation:

$$\begin{aligned}\frac{1}{\rho}\left(\frac{\partial\rho}{\partial t}+u\frac{\partial\rho}{\partial x}+v\frac{\partial\rho}{\partial y}+w\frac{\partial\rho}{\partial z}\right)\\+\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0.\end{aligned}\tag{1}$$

Here, $\rho$ [kg/m$^3$] is the fluid density, $t$ [s] is time, and $u,v,w$ [m/s] are the fluid velocities in the $x,y,z$ directions.

Disregarding the effects of surface tension and elasticity, the application of the translational equation of motion to a fluid particle yields the Navier-Stoke's equations:

$$\frac{\mathrm{D}u}{\mathrm{D}t}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\frac{1}{\rho}\left(\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{yx}}{\partial y}+\frac{\partial\tau_{zx}}{\partial z}\right)+X,\tag{2}$$

$$\frac{\mathrm{D}v}{\mathrm{D}t}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\frac{1}{\rho}\left(\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial\tau_{yy}}{\partial y}+\frac{\partial\tau_{zy}}{\partial z}\right)+Y,\tag{3}$$

$$\frac{\mathrm{D}w}{\mathrm{D}t}=-\frac{1}{\rho}\frac{\partial p}{\partial z}+\frac{1}{\rho}\left(\frac{\partial\tau_{xz}}{\partial x}+\frac{\partial\tau_{yz}}{\partial y}+\frac{\partial\tau_{zz}}{\partial z}\right)+Z.\tag{4}$$

Here, $\frac{\mathrm{D}}{\mathrm{D}t}$ is the material derivative, $p$ [N/m$^2$] is the fluid pressure, $\tau$ [N/m$^2$] is the shear stress where the first subscript refers to the surface perpendicular and the second subscript refers to the direction of the stress, and $X,Y,Z$ [N] are body forces along the $x$, $y$, and $z$ directions.

### 2.2 Assumptions and the Governing Equation

The following assumptions are made:

- Incompressible fluid ($\rho$ =constant).
- Inviscid fluid ($\tau=0$).
- Irrotational flow ($\frac{\partial w}{\partial y}=\frac{\partial v}{\partial z}$, $\frac{\partial w}{\partial x}=\frac{\partial u}{\partial z}$ and $\frac{\partial v}{\partial x}=\frac{\partial u}{\partial y}$).

DOI 10.3384/ECP2016991

- Low wave steepness, i.e., ($H \ll L$).

- Long crested waves (2D flow, in $x$ and $z$ directions only).

- Horizontal and time-invariant bottom boundary.

Assuming incompressible fluid and long crested waves,

$$\text{Eqn. (1)} \rightarrow \frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0. \tag{5}$$

Assumption of inviscid fluid gives the Euler equations:

$$\text{Eqn. (2)} \rightarrow \frac{\mathrm{D}u}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial x}, \tag{6}$$

$$\text{Eqn. (4)} \rightarrow \frac{\mathrm{D}w}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial z} - g. \tag{7}$$

The assumption of irrotational flow makes it possible to define a scalar velocity potential $\phi(x,y,z,t)$ [m$^2$/s] such that its directional derivative gives the fluid velocity in that direction. i.e.,

$$u = \frac{\partial \phi}{\partial x}, \qquad w = \frac{\partial \phi}{\partial z}. \tag{8}$$

Thus, for an incompressible, irrotational flow in the $x$ and $z$ directions, the integrated form of the Euler equation yields the Bernoulli equation for unsteady potential flow,

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}\left\{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial z}\right)^2\right\} + \frac{p}{\rho} + gz = C(t), \tag{9}$$

where $C(t)$ is the Bernoulli term and is a constant for steady flows.

With $\mathbf{u} = [u,w]^\mathrm{T}$, (5) may be expressed in vector form as $\nabla.\mathbf{u} = 0$. From (8), $\mathbf{u}= \nabla\phi$, so we have

$$\nabla.\nabla\phi = \nabla^2\phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = 0. \tag{10}$$

Equation (10) is the well-known Laplace equation, and consitutes the governing differential equation which is valid throughout the fluid domain. Our interest is in determining the velocity potential which satisifies the Laplace equation, which then makes it possible to determine the fluid velocities at any point in the fluid domain.

The Bernoulli equation relates the fluid velocities to the the fluid pressure, and the integration of the fluid pressure along the surface of any submerged/floating object gives the force that the fluid exerts on the object, which is, in most cases, the element of interest in wave-body interaction problems.

## 2.3 The Boundary Conditions

In seeking a solution for the velocity potential in (10), we make use of the following physical conditions which must be satisfied by the fluid velocity and pressure, at the domain boundaries:

1. *The Kinematic Free-Surface Boundary-Condition* (KFSBC) stemming from the fact that there cannot be any fluid flow across the interface between the liquid domain and the atmosphere at the free surface of the fluid.

2. *The Bottom Boundary-Condition* (BBC) stemming from the fact that there cannot be any fluid flow across the sea floor.

3. *The Dynamic Free-Surface Boundary-Condition* (DFSBC) stemming from the fact that 'free' surfaces such as the air-water interface cannot support pressure variations across it, and hence must be capable of responding in order to maintain the pressure continuity across the liquid and gaseous domains. This displacement of the free surface means that the position of the upper boundary is not known a priori in the water-wave problem. For small-amplitude waves, this condition is given by the requirement that the pressure on the free surface is uniform along the wave form.

4. *The Spatial and Temporal Periodicity Condition at the Lateral Surfaces* (LPBC) stemming from the fact that the solution we seek is the velocity potential associated with a wave which is periodic in both space and time.

Mathematical expressions for the kinematic boundary conditions may be derived from the equation of the form $\mathrm{F}(x,y,z,t) = 0$, describing the boundary surface. For a temporally varying surface, the total time-derivative of the surface is zero, on the surface. Hence, for a 2D wave surface-profile,

$$\frac{\mathrm{DF}}{\mathrm{D}t} = \frac{\partial \mathrm{F}}{\partial t} + u\frac{\partial \mathrm{F}}{\partial x} + w\frac{\partial \mathrm{F}}{\partial z} = 0 \text{ on } \mathrm{F}(x,z,t) = 0. \tag{11}$$

Or, rearranging and using vector notation,

$$-\frac{\partial \mathrm{F}}{\partial t} = \mathbf{u}.\nabla\mathrm{F} = \mathbf{u}.\mathbf{n}|\nabla\mathrm{F}|. \tag{12}$$

Here, $\mathbf{n}= \frac{\nabla\mathrm{F}}{|\nabla\mathrm{F}|}$ is the unit normal to the surface, and $|\nabla\mathrm{F}| = \sqrt{\left(\frac{\partial \mathrm{F}}{\partial x}\right)^2 + \left(\frac{\partial \mathrm{F}}{\partial z}\right)^2}$.

The kinematic boundary condition is thus expressed as

$$\mathbf{u}.\mathbf{n} = -\frac{\partial \mathrm{F}/\partial t}{|\nabla\mathrm{F}|} \text{ on } \mathrm{F}(x,y,z,t) = 0. \tag{13}$$

At the free surface, $\mathrm{F}(x,z,t) = z - \eta(x,t) = 0$, where $\eta(x,t)$ is the displacement of the free surface about the horizontal plane. Equation (13) gives

$$\mathbf{u}.\mathbf{n} = \frac{\partial\eta/\partial t}{\sqrt{\left(\frac{\partial\eta}{\partial x}\right)^2 + 1}} \text{ on } z - \eta(x,t) = 0. \tag{14}$$

Taking

$$\mathbf{n} = \frac{-\frac{\partial\eta}{\partial x}\mathbf{i} + \mathbf{k}}{\sqrt{\left(\frac{\partial\eta}{\partial x}\right)^2 + 1}}, \tag{15}$$

(14) gives the KFSBC as

$$w = \frac{\partial\eta}{\partial t} + u\frac{\partial\eta}{\partial x} \text{ on } z = \eta(x,t). \tag{16}$$

Assuming a horizontal, time-invariant bottom at $z = -h$, $\mathrm{F}(z) = z + h = 0$. Equation (13) gives $\mathbf{u}.\mathbf{n} = 0$ at $z = -h$. Here, $\mathbf{n} = \mathbf{k}$, and hence the BBC can be expressed as

$$w = 0 \text{ on } z = -h. \tag{17}$$

By specifying a uniform pressure ($p_\eta$ =constant) along the wave form in the Bernoulli equation at the free surface, the DFSBC may be expressed mathematically as

$$\frac{\partial\phi}{\partial t} + \frac{1}{2}\left\{\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial z}\right)^2\right\} + \frac{p_\eta}{\rho} + gz = \mathrm{C}(t) \text{ on } z = \eta(x,t). \tag{18}$$

The LPBCs may be expressed as

$$\phi(x,t) = \phi(x+L,t), \tag{19}$$

$$\phi(x,t) = \phi(x,t+T). \tag{20}$$

Here, $L$ [m] is the wave length and $T$ [s] is the wave period.

## 2.4 Solution of the Boundary Value Problem

The BVP to be solved is thus the Laplace equation (10) subject to: 1. the KFSBC in (16), 2. the DFSBC in (18), 3. the BBC in (17), and the LPBC in (19) and (20). The diagrammatic representation of the problem is shown in Figure 1.

### 2.4.1 Manipulation of the Free Surface Boundary Conditions

On carrying out a non-dimensional analysis of the terms in the KFSBC and the DFSBC, under the assumption of low wave steepness, i.e., $H/L \ll 1$, we notice that $u\frac{\partial\eta}{\partial x} \ll \frac{\partial\eta}{\partial t}$, $u\frac{\partial\eta}{\partial x} \ll \frac{\partial\phi}{\partial z}$ and $\left(\frac{\partial\phi}{\partial x}\right)^2, \left(\frac{\partial\phi}{\partial z}\right)^2 \ll \frac{\partial\phi}{\partial t}$.

Further, the KFSBC and DFSBC are to be evaluated at $z = \eta(x,t)$, which is a priori unknown. However, on taking a Taylor series expansion of the BCs about the mean free surface at $z = 0$, we notice that the second-order and subsequent higher-order terms can be neglected; hence, we can safely assume the validity of the BCs at the mean free surface instead of the actual free surface. Details about linearization and shifting can be found in (Techet, 2005).

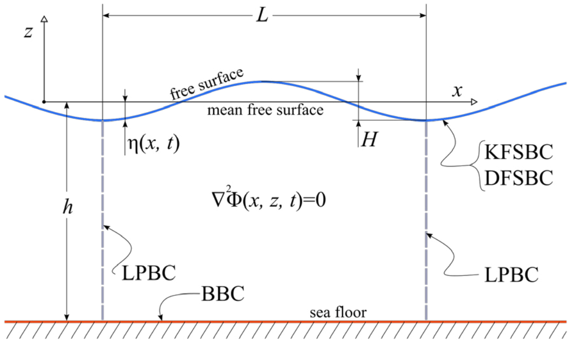


**Figure 1.** Boundary value problem for the velocity potential. Adapted from (Dean and Dalrymple, 2001).

Taking the pressure at the free surface as the constant atmospheric pressure, we can eliminate the Bernoulli constant and the pressure term in (9) as demonstrated in (Andersen and Frigaard, 2011). Thus, the modified BCs are:

$$\text{KFSBC:} \quad \frac{\partial\phi}{\partial z} = \frac{\partial\eta}{\partial t} \quad \text{on} \quad z = 0, \tag{21}$$

$$\text{DFSBC:} \quad \frac{\partial\phi}{\partial t} + g\eta = 0 \quad \text{on} \quad z = 0. \tag{22}$$

Differentiating (22) w.r.t. $t$ and using (21), we can combine both the BCs to give the Combined Free-Surface Boundary-Condition (CFSBC) as:

$$\text{CFSBC:} \quad \frac{\partial^2\phi}{\partial t^2} + g\frac{\partial\phi}{\partial z} = 0 \quad \text{on} \quad z = 0. \tag{23}$$

### 2.4.2 Complex Exponential Form of the Velocity Potential

It is often mathematically advantageous to use the complex form of the velocity potential; see p. 4 of (Chakrabarti, 1987). Since the solution we seek is related to a progressive sinusoidal wave, we may express the velocity potential as

$$\phi = \varphi(z)e^{i(kx-\omega t)}. \tag{24}$$

The LPBCs were utilized in the formulation of the above equation, the real part of which represents the velocity potential of a sinusoidal wave progressing in the positive x-direction; see pp. 2, 12 of (Krogstad and Arntsen, 2000).

The BVP is now given by:

$$\text{Laplace eqn:} \quad \frac{\partial^2 \varphi}{\partial z^2} - k^2 \varphi = 0. \tag{25}$$

$$\text{CFSBC:} \quad -\omega^2 \varphi + g\frac{\partial \varphi}{\partial z} = 0 \text{ on } z = 0. \tag{26}$$

$$\text{BBC:} \quad \frac{\partial \varphi}{\partial z} = 0 \text{ on } z = -h. \tag{27}$$

Assuming a solution of the form

$$\varphi(z) = \mathrm{C}_1 e^{-kz} + \mathrm{C}_2 e^{kz}, \tag{28}$$

$$\text{Eqn. (27)} \rightarrow \quad \mathrm{C}_1 k e^{-kh} - \mathrm{C}_2 k e^{kh} = 0, \tag{29}$$

$$\text{Eqn. (26)} \rightarrow \quad (gk - \omega^2)\mathrm{C}_1 - (gk + \omega^2)\mathrm{C}_2 = 0. \tag{30}$$

The above homogenous equation system has non-trivial solutions only when the determinant is zero. This gives the dispersion relation

$$\omega^2 = gk \tanh(kh). \tag{31}$$

Now, setting $\mathrm{C}_1 = \frac{1}{2}\mathrm{B}e^{kh}$ and $\mathrm{C}_2 = \frac{1}{2}\mathrm{B}e^{-kh}$,

$$\text{Eqn. (28)} \rightarrow \quad \varphi(z) = B\cosh[k(z+h)], \tag{32}$$

$$\text{Eqn. (24)} \rightarrow \quad \phi(x,z,t) = B\cosh[k(z+h)]e^{i(kx-\omega t)}, \tag{33}$$

$$\text{Eqn. (22)} \rightarrow \quad \eta = \frac{-1}{g}\frac{\partial \phi}{\partial t}\bigg|_{z=0}. \tag{34}$$

Considering that that highest value of $\eta$ is the wave amplitude $A = H/2$ [m], from (33) and (34), we have

$$B = \frac{-igA}{\omega}\frac{1}{\cosh(kh)}. \tag{35}$$

The complex exponential form of the velocity potential may now be expressed as

$$\phi(x,z,t) = \frac{-igA}{\omega}\frac{\cosh k(z+h)}{\cosh(kh)}e^{i(kx-\omega t)}. \tag{36}$$

## 2.5 Kinematics and Dynamics of Regular Waves

Considering the real part of the velocity potential in (36),

$$\phi = \frac{gH}{2\omega}\frac{\cosh k(z+h)}{\cosh(kh)}\sin(kx-\omega t), \tag{37}$$

$$\eta = H/2\cos(kx-\omega t), \tag{38}$$

$$u = \frac{\pi H}{T}\frac{\cosh k(z+h)}{\sinh(kh)}\cos(kx-\omega t), \tag{39}$$

$$w = \frac{\pi H}{T}\frac{\sinh k(z+h)}{\sinh(kh)}\sin(kx-\omega t), \tag{40}$$

$$\dot{u} = \frac{2\pi^2 H}{T^2}\frac{\cosh k(z+h)}{\sinh(kh)}\sin(kx-\omega t), \tag{41}$$

$$\dot{w} = -\frac{2\pi^2 H}{T^2}\frac{\sinh k(z+h)}{\sinh(kh)}\cos(kx-\omega t), \tag{42}$$

$$\delta_x = -\frac{H}{2}\frac{\cosh k(z+h)}{\sinh(kh)}\sin(kx-\omega t), \tag{43}$$

$$\delta_z = \frac{H}{2}\frac{\sinh k(z+h)}{\sinh(kh)}\cos(kx-\omega t), \tag{44}$$

$$p = \rho g\frac{H}{2}\frac{\cosh k(z+h)}{\cosh(kh)}\cos(kx-\omega t). \tag{45}$$

Here, $\dot{u}$ and $\dot{w}$ [m/s$^2$] are the water particle accelerations, $\delta_x$ and $\delta_z$ [m] are water particle displacements from their mean position, and $p$ [N/m$^2$] is the dynamic pressure. (Chakrabarti, 1987).

## 2.6 Kinematics and Dynamics of Irregular Waves

The process of linearization carried out in Section 2.4.1 implies the validity of the superpostion principle with regards to the quantities expressed in Section 2.5. This, in turn, justifies the representation of the irregular wave parameters as the summation of the parameters of constituent regular waves; or, in other words, as a Fourier series, as represented in Figure 2.

The way of describing the sea-state is linked to the energy content in waves. Linear theory gives the wave energy per unit area of the sea surface due to a regular wave as

$$E = \frac{1}{2}\rho g \zeta_{0i}^2. \tag{46}$$

Here, $\zeta_{0i}$ [m] is the amplitude of the regular wave under consideration; see p. 97 of (Dean and Dalrymple, 2001).

The spectrum for the irregular wave process is defined such that the area of the wave spectrum $S_\eta(\omega)$ within the frequency interval $\Delta\omega$ represents the wave energy for the same frequency interval. Hence, from a known spectrum function, we can find the amplitude $\zeta_{0i}$ [m] of the harmonic wave component which represents the wave energy for a given frequency resolution using (47); see p. 23 of (Faltinsen, 1999), and p. 122 of (Chakrabarti, 1987):

$$\zeta_{0i} = \sqrt{2S_\eta(\omega_i)\Delta\omega}. \tag{47}$$

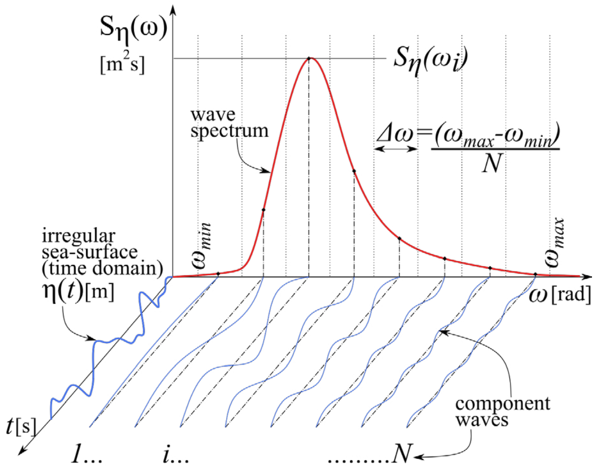


**Figure 2.** Concept of Fourier series representation of irregular waves and the wave spectrum. Adapted from (Faltinsen, 1999).

Once the amplitudes corresponding to each of the component regular waves is known, randomness may be introduced by the inclusion of an arbitrary phase difference $\varepsilon_i$ [rad]. The property of the irregular wave may now be expressed as the summation of the property of the component waves of specified frequencies with random phase. For e.g., the sea surface elevation (SSE) at a given $x$ coordinate is expressed by equation (48); see p. 123 of (Chakrabarti, 1987):

$$\eta(x,t) = \sum_{i=1}^{N} \zeta_{0i} \cos(k_i x - \omega_i t - \varepsilon_i). \tag{48}$$

Here, $N$ is the total number of wave components (frequency bands), $\zeta_0$ [m] is the component wave amplitude, $\omega$ [rad/s] is the wave angular frequency, and $\varepsilon$ [rad] is the phase. Subscript $i$ refers to the number of the component wave under consideration. The wave number $k$ [rad/m] is to be determined from the dispersion relation given in (31). $g$ [m/s$^2$] is the acceleration of gravity and $d$ [m] is the water depth. $S_\eta(\omega_i)$ [m$^2$s] is the energy spectral density and $\Delta\omega$ [rad/s] is the width of the frequency bands dividing the total wave spectrum.

The even distribution of component frequencies will cause the resultant wave to be periodic with a period of of $2\pi/\omega_{min}$ [s], and thus not truly irregular. Hence, the component frequency within each frequency interval is selected based on a unifrom random distribution, as advised in p. 209 of (Fossen, 2011).

# 3 Modelica Implementation

## 3.1 General Considerations

Most simulation problems envisaged would require, in one way or the other, the determination of wave/current forces acting on structures with varying degrees of restraint as illustrated in Figure 3. Since the wave forces vary both temporally and spatially, and since the location information is contained in the component-model for the body in a wave-body interaction problem, it was decided that a wave component-model that generates all the required parameters that allows for the determination of the various quantities given in the equations under Section 2.5, at any location ($x$, $y$, $z$) within the problem domain, at any specified simulation time $t$, would be the best approach.

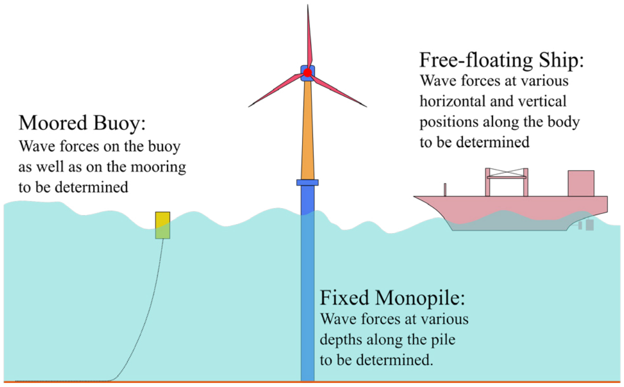

Free-floating Ship:
Wave forces at various horizontal and vertical positions along the body to be determined
Moored Buoy:
Wave forces on the buoy as well as on the mooring to be determined
Fixed Monopile:
Wave forces at various depths along the pile to be determined.

**Figure 3.** Expected simulation scenarios.

Once the wave parameters such as component frequencies, corresponding amplitudes and phases are determined, they would have to be made available to the body component-model for determination of wave properties at the desired location. Towards this end, an information bus holding the required data is to be specified from which the body component-model may then access this data.

Considering the above general requirements, the system model for integrated simulation may be represented by the block diagram in Figure 4. While the wave, current, and data bus are common components for any ocean engineering simulation, the other components may vary depending upon the scope of the simulation. The rest of this paper is dedicated to the implementation of the component-models for waves and current.

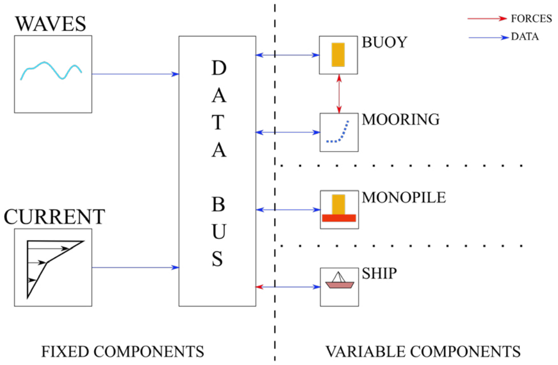


**Figure 4.** General block diagram for integrated simulation of an ocean engineering system.

Flow-charts provided in the following sub-sections have been prepared with ocean engineers, most likely to be unfamiliar with *Modelica*, in mind, and some elements might appear superfluous to the *Modelica* savvy reader.

## 3.2 Regular-Wave Component-Model

The height of the regular wave $H_r$ [m], time period $T_r$ [s], water depth $d$ [m], water density $\rho_w$ [kg/m$^3$], ramp time $T_{rmp}$ [s], delay time $T_{del}$ [s] and the number of frequency components ${}_n\omega_i = 1$ are specified as *parameters* in the **Regular_Airy_Wave** component model. $T_{sim}$ [s] is the required duration of simulation.

$T_{rmp}$ is used to ramp the wave height in order to prevent impulse wave loads at the start of the simulation, while $T_{del}$ maybe used to start the waves at a specified time into the integrated simulation.

The wave angular frequency $\omega = 2\pi/T$ [rad] and $d$ are passed on as parameters to the function **waveNumberIterator**, which iterates for the wave number based on the dispersion relation given in (31), and returns the final value to **Regular_Airy_Wave**.

A data connector **WaveDataConnector** transmits $d$, $\rho_w$, $\omega$, $T$, $k$, $\varepsilon$, $\zeta_{0i}$, and $SSE_{X0}$ to the data bus which is an *expandable connector* named the **EnvironmentBus**. Here, $\varepsilon$ [rad], the phase difference is redundundant for the case of a regular wave and is set to zero, while $SSE_{X0}$ is the sea surface elevation calculated at $x = 0$ using (38).

The algorithm for generation of regular wave parameters is depicted in the flow chart given in Figure 5, and the flow chart for the *function* **waveNumberIterator** is given in figure 6. The first value for the wave number iteration is taken to be $k_0 = \frac{2\pi}{L_0}$, where $L_0 = \frac{gT^2}{2\pi}$ [m] is the deep-water wave length as given on p. 66 of (Dean and Dalrymple, 2001).

Equations (37)–(45) can then be used to calculate the wave properties at the required position coordinates, contained in the body component-model, at any required simulation time $t$ [s].

## 3.3 Irregular-Wave Component-Model

The generation of component wave parameters based on the Pierson-Moskowitz spectrum is considered for detailed description. The algorithm for the irregular-wave component-model **IRW_PM_RDFCWI** is shown in Figure 7.

The water depth $d$ [m], significant wave height $H_s$ [m], the ramp time $T_{rmp}$ [s], the lower cut-off frequency $\omega_{min}$ [rad/s], the upper cut-off frequency $\omega_{max}$ [rad/s], and the number of frequency components to be considered ${}_n\omega_i$, are specified as *parameter* inputs.

The frequency resolution $\Delta\omega = (\omega_{max} - \omega_{min})/{}_n\omega_i$ [rad/s] is determined. The component frequency within each frequency interval $\Delta\omega$ is then selected based on a uniform random distribution by the *function* **frequencySelector**.

To generate a vector of random numbers, a *function* **randomNumberGenerator** based on the *Modelica.Math.Random.Generators.Xorshift64star* random number generator, with a *for* loop included, to return a vector of random numbers of specified size, corresponding to the number of frequency components ${}_n\omega_i$, is called. The **frequencySelector** function is a simple function that shifts the component frequencies randomly within the associated frequency interval based on the generated random numbers $rnd\_shft[{}_n\omega_i]$.

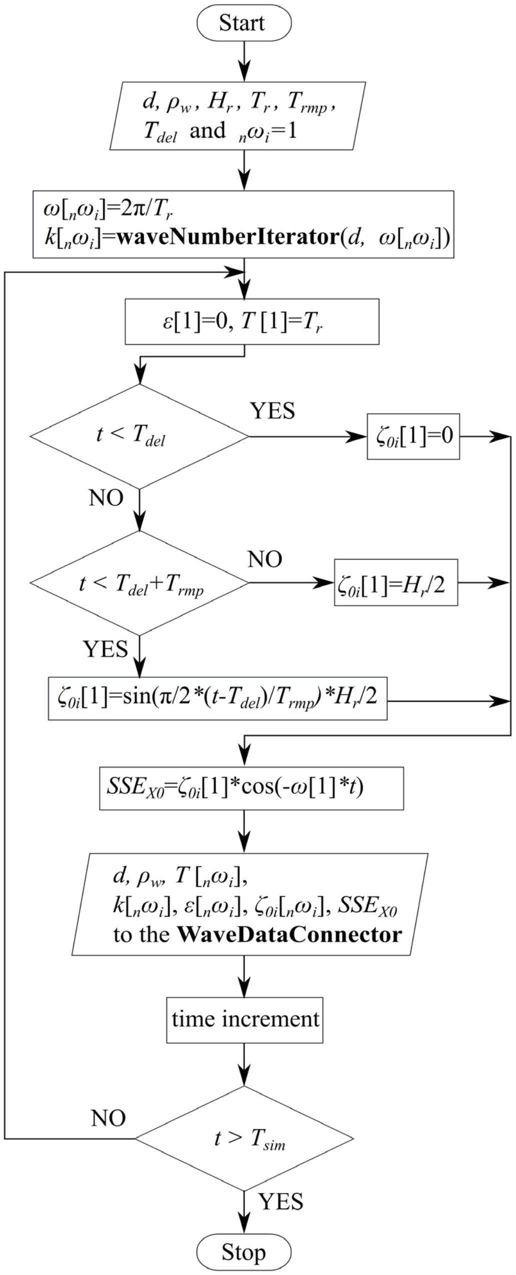


**Figure 5.** Flow chart for regular-wave component-model.

Once the component frequencies are identified, the corresponding spectral values are determined by calling the *function* **spectrumGenerator_PM** which calculates the

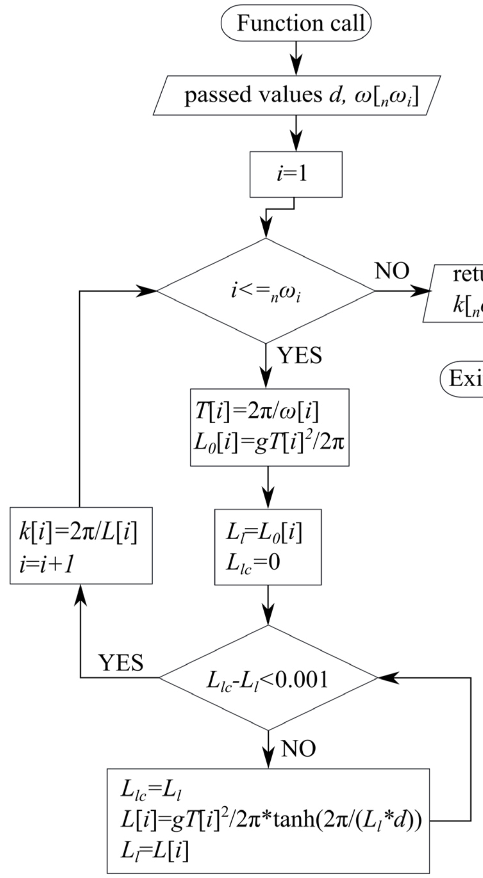


**Figure 6.** Flow chart for iteration of the wave number.

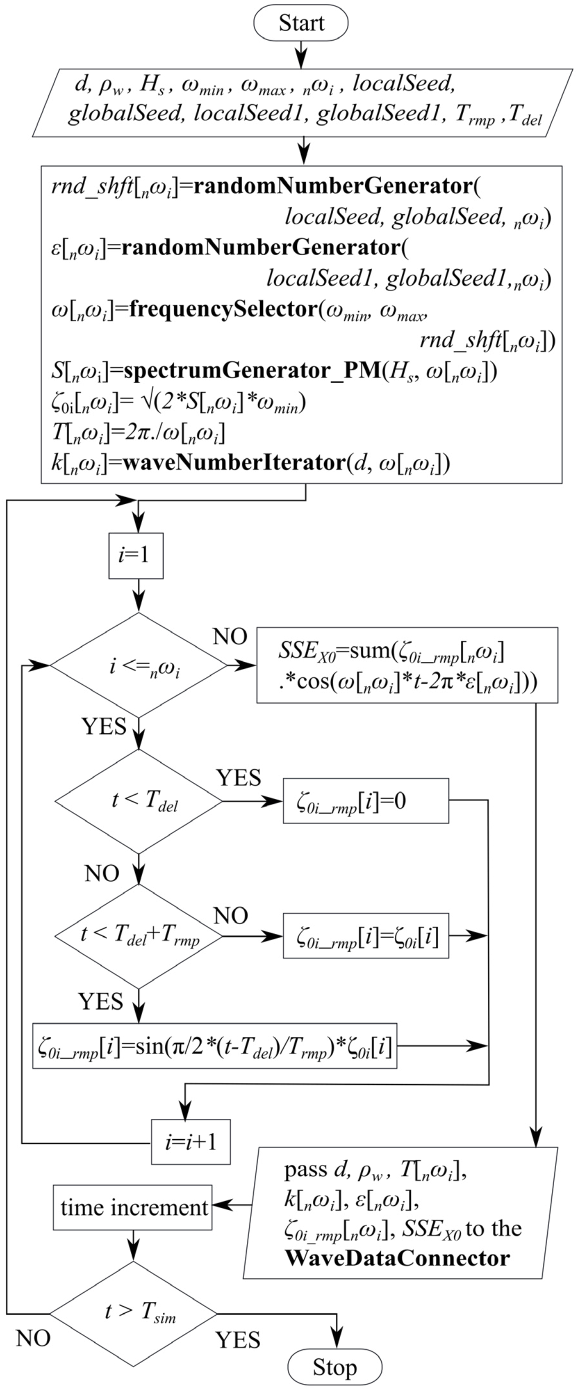

Start
$d$, $\rho_w$, $H_s$, $\omega_{min}$, $\omega_{max}$, $_n\omega_i$, localSeed, globalSeed, localSeed1, globalSeed1, $T_{rmp}$, $T_{del}$
rnd_shft[$_n\omega_i$]=**randomNumberGenerator**( localSeed, globalSeed, $_n\omega_i$)
ε[$_n\omega_i$]=**randomNumberGenerator**( localSeed1, globalSeed1,$_n\omega_i$)
ω[$_n\omega_i$]=**frequencySelector**($\omega_{min}$, $\omega_{max}$, rnd_shft[$_n\omega_i$])
S[$_n\omega_i$]=**spectrumGenerator_PM**($H_s$, ω[$_n\omega_i$])
$\zeta_{0i}$[$_n\omega_i$]= √(2*S[$_n\omega_i$]*$\omega_{min}$)
T[$_n\omega_i$]=2π./ω[$_n\omega_i$]
k[$_n\omega_i$]=**waveNumberIterator**(d, ω[$_n\omega_i$])
$i$=1
$i$ <=$_n\omega_i$
NO
$SSE_{X0}$=sum($\zeta_{0i\_rmp}$[$_n\omega_i$] .*cos(ω[$_n\omega_i$]*t-2π*ε[$_n\omega_i$]))
YES
$t$ < $T_{del}$
YES
$\zeta_{0i\_rmp}[i]$=0
NO
$t$ < $T_{del}$+$T_{rmp}$
NO
$\zeta_{0i\_rmp}[i]$=$\zeta_{0i}[i]$
YES
$\zeta_{0i\_rmp}[i]$=sin(π/2*($t$-$T_{del}$)/$T_{rmp}$)*$\zeta_{0i}[i]$
$i$=$i$+1
pass $d$, $\rho_w$, T[$_n\omega_i$], k[$_n\omega_i$], ε[$_n\omega_i$], $\zeta_{0i\_rmp}$[$_n\omega_i$], $SSE_{X0}$ to the **WaveDataConnector**
time increment
$t$ > $T_{sim}$
NO
YES
Stop

**Figure 7.** Flow chart for the irregular-wave component-model.

spectral density values based on the empirical formula

$$S_\eta(\omega_i) = \frac{5\pi^4 H_s^2}{T_p^4 \omega_i^5} \exp\left(\frac{-20\pi^4}{T_p^4 \omega_i}\right). \qquad (49)$$

Here, $T_p$ [s] is the peak period of the spectrum, and is related to $H_s$ through the relations $T_p = \frac{2\pi}{\omega_p}$, and $\omega_p^2 = \frac{0.161g}{H_s}$. $\omega_p$ [rad/s] is the peak angular frequency; see pp. 105–107 of (Chakrabarti, 1987).

In the future, generation of wave records based on other commonly used sea-spectra may be incorporated by defining the correspoding spectrum generating functions.

The amplitudes of the component waves $\zeta_{0i}$ are then determined using (47), and corresponding wave numbers $k_i$ are determined using the *function* **waveNumberIterator** described in Section 3.2. The randomly distributed phases are determined by a second call to the *function* **randomNumberGenerator**. This function call returns a vector $\varepsilon[_n\omega_i]$ of uniformly distributed random numbers in (0,1] and hence, the associated phase difference is expressed as $2\pi\varepsilon$ [rad] .

Having determined all the required parameters, the sea surface elevation at $x = 0$, $SSE_{X0}$ [m], is then calculated using the formula given in (48). The values are then linked to the *expandable connector* EnvironmentBus using the **WaveDataConnector** as described in Section 3.2.

## 3.4 Component-Model of Depth-Varying Current

The component-model for current is a simple block which produces as its output two vectors $zcg[n]$ and $Ucg[n]$. $zcg$ contains the co-ordinate information and $Ucg$ contains the corresponding current velocities. The *parameters* specified are the $zcg[n]$ which is a vector containing the $n$ depth positions where current velocities are defined, $Uf[n]$ which is a vector containing the fully developed current values, and the ramp time $T_{\text{rmp}}$ [s]. $Ucg[n]$ holds the instantaneous value of the ramped current. A sinusoidal ramping function is used for smooth ramping. A **CurrentDataConnector** links the $zcg$ and $Ucg$ values to the *expandable connector* **EnvironmentBus**. The current velocity at any location may now be computed by the different body component-models by interpolation.

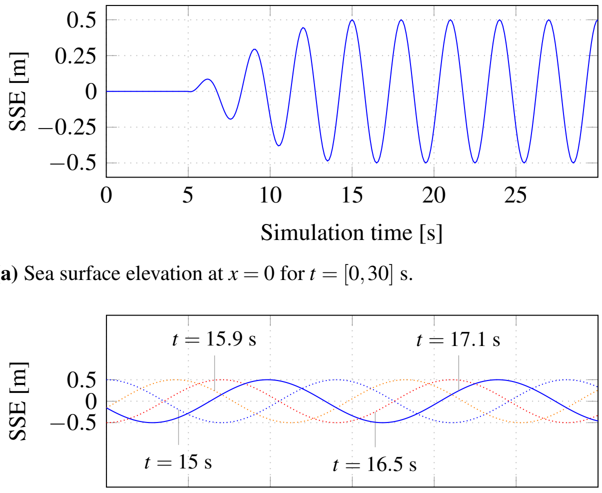
# 4 Results

All results presented below are based on outputs of the above component models. Simulation files are available for download at `github.com/Savin-Viswanathan/Modelica2020-a`.

## 4.1 Regular Wave

The simulation model **Check_RegularWave** under the *sample simulations* in the above link calculates the wave properties based on the parameters generated by the **Regular_Airy_Wave** component-model.

Figure 8a shows a sample sea surface elevation at $x = 0$ [m] with $T_{del} = 5$ [s], $T_{rmp} = 10$ [s], $H_r = 1$ [m], $d = 10$ [m] and $T_r = 3$ [s], for a simulation interval of 0–30 [s], while Figure 8b shows the progressive wave profile for the same wave in the spatial interval 0–30 [m] for different simulation times.



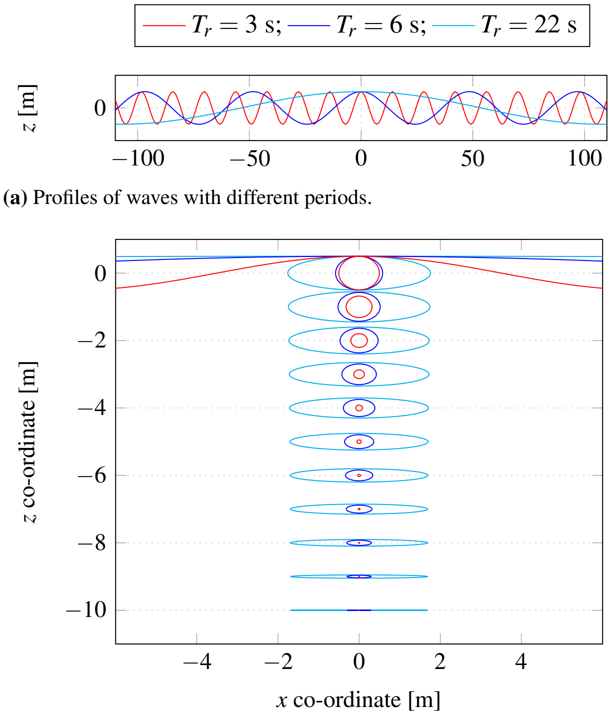
**(a)** Sea surface elevation at $x = 0$ for $t = [0, 30]$ s.

**(b)** Wave profile at different simulation time steps for $x = [0, 30]$ m.

**Figure 8.** Sea surface elevation and the progressive wave profile.

Figure 9a shows the wave profiles at $t = 0$ [s] for different $T_r$, and Figure 9b shows the trajectory traced by water particles with different mean positions during a complete wave cycle, at different depths, for the different wave periods, in a water depth $d = 10$ [m]. We observe that,

- For $T_r = 3$ [s], $k = 0.447414$ [m$^{-1}$], and $kd > \pi$. The wave is in *deep water* and the trajectories are circular. The displacements in the vertical and horizontal directions decay exponentially with depth and the particles near the bottom boundary have no horizonatal or vertical displacements.
- For $T_r = 6$ [s], $k = 0.129834$ [m$^{-1}$], and $\frac{\pi}{10} < kd < \pi$. The wave is in *intermediate water* and the trajectories are elliptical. The displacements in the vertical and horizontal directions decay with depth and the particles near the bottom boundary have only horizontal displacements.
- For $T_r = 22$ [s], $k = 0.029246$ [m$^{-1}$], and $kd < \frac{\pi}{10}$. The wave is in *shallow water* and the trajectories are elliptical. The displacements in the vertical direction decay linearly with depth, while the horizontal displacement is near constant at all depths.



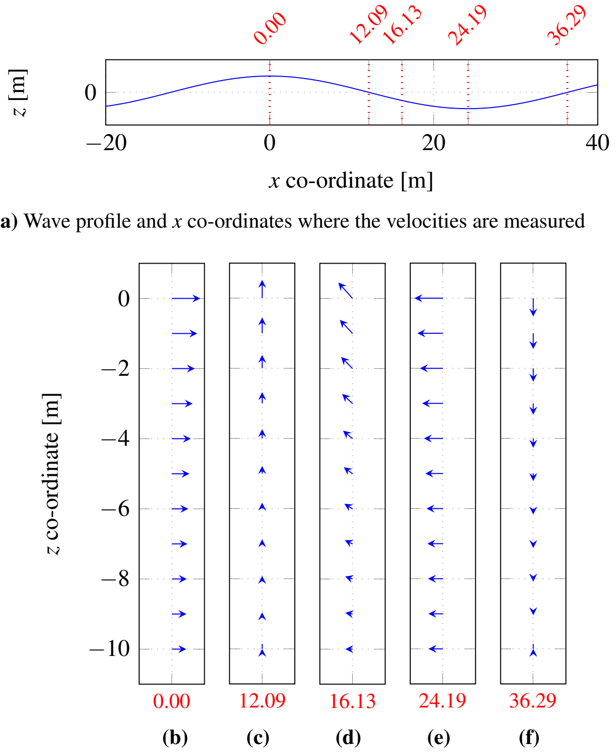
**(a)** Profiles of waves with different periods.

**(b)** Water particle trajectories of waves with different wave periods.

**Figure 9.** Wave profiles and water particle trajectories.

Figure 10a shows the intstantaneous wave profile for a regular progressive wave with $T_r = 6$ [s], $H_r = 1$ [m], in a water depth $d = 10$ [m], when there is a crest at $x = 0$ [m]. Figure 10b–10f shows the quiver plots for the instantaneous velocities of water-particles with different mean $z$ co-ordinates, under different $x$ co-ordinates.

An important consideration to keep in mind is that the linearization of the boundary conditions in the derviation of the velocity potential has the effect that the water particle kinematics derived from such a potential does not



**(a)** Wave profile and $x$ co-ordinates where the velocities are measured

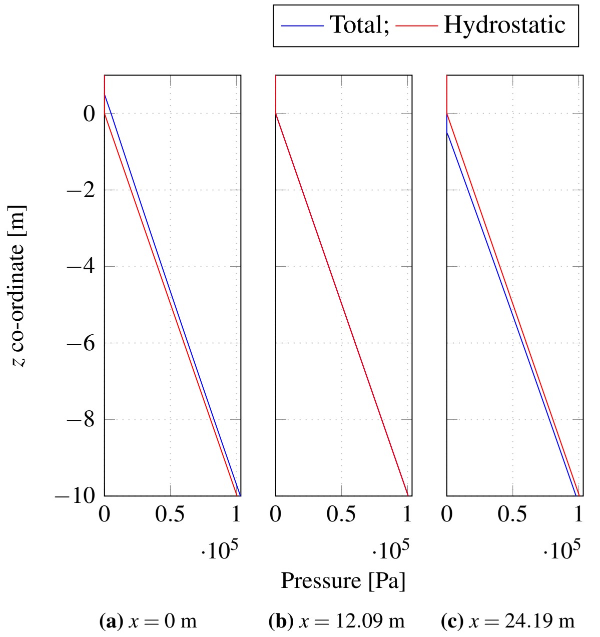
**Figure 10.** Wave profile (a), water-particle velocities (b)–(f).

account for the change of position of the particle within the fluid, and hence the theory cannot give a proper description for flow velocity and acceleration in the region between the still water level and the wave crest and in the void to the wave trough. Extrapolation of the values is, in general, not recommended since the wave forces will be overestimated. A better way is to apply *Wheeler stretching* or *move* the profile for velocity and acceleration to the instantaneous sea surface; see p. 221 of (SINTEF, 2014).

Figure 11 shows the pressure distribution at various $x$ co-ordinates for the same wave as above. The dynamic pressure above $z = 0$ [m] has been calculated using a truncated Taylor series for small positive distances, as given on p. 84 of (Dean and Dalrymple, 2001).

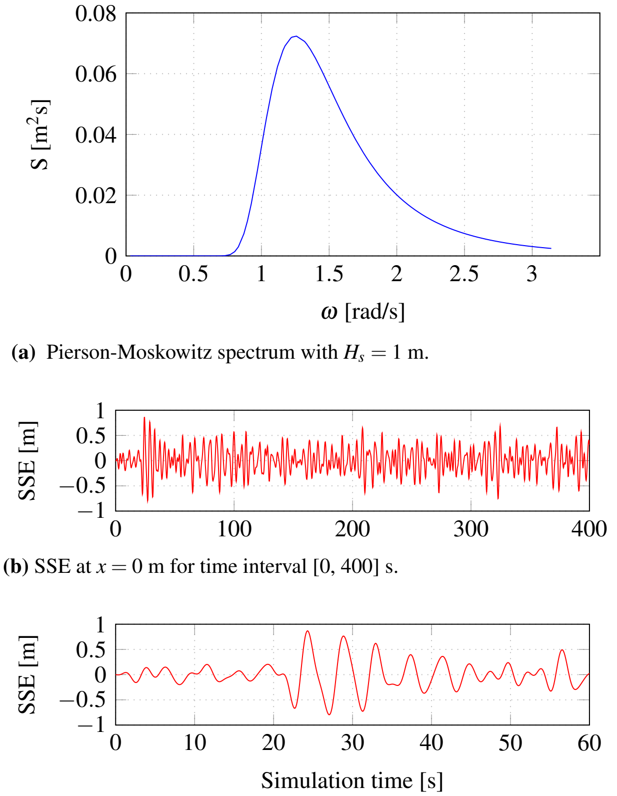
## 4.2 Irregular Wave

Figure 12a depicts a Pierson-Moskowitz spectrum of $H_s =$ 1 [m] generated by the **spectrumGenerator_PM** function, while Figure 12b depicts the sea surface elevation for an irregular wave record with 100 frequency components, generated from the spectrum by the **IRW_PM_RDFCWI** irregular-wave component-model with $T_{rmp} = 10$ [s], $T_{del} = 0$ [s], $\Delta\omega = \omega_{min} = 0.03141$ [rad/s], and $\omega_{max} =$ 3.141 [rad/s]. Figure 12c shows an expanded view of the same wave record in a shorter time interval, for clarity.



**(a)** $x = 0$ m **(b)** $x = 12.09$ m **(c)** $x = 24.19$ m

**Figure 11.** Pressures beneath the wave crest, down-crossing, and trough.



**(a)** Pierson-Moskowitz spectrum with $H_s = 1$ m.

**(b)** SSE at $x = 0$ m for time interval [0, 400] s.

**(c)** SSE at $x = 0$ m for time interval [0, 60] s.

**Figure 12.** Irregular waves.

## 4.3 Depth-varying Current

Figure 13 depicts the instantaneous profile for a depth varying current which is based on the output of the **CurrentProfile_4pt** component-model. The current is ramped up to full value using the parameter $T_{rmp} = 5$ [s].

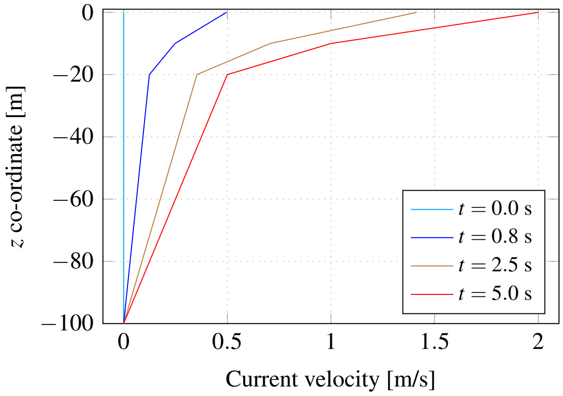


**Figure 13.** Current profile

# 5 Conclusion

General considerations to be kept in mind while formulating a framework for carrying out integrated simulation of ocean-engineering systems is presented and the algorithms for development of *Modelica* component-models for the generation of regular and irregular waves are described. The implementation of a simple component-model for generation of depth varying current is also presented.

Graphical representation of the wave kinematics and dynamics based on the output of the component-models for regular waves are then presented to show satisfactory agreement with general results discussed in (Dean and Dalrymple, 2001). A sample sea-surface-elevation based on the output of the component-model for irregular waves is presented. Since, within the assumption of linearity, the properties of the irregular wave are a linear combination of the properties of the consitutent regular waves, it is deemed that the output of the irregular wave component-model is satisfactory. Graphical representation of the output of the component-model for depth varying current is then presented.

For a better understanding of how these component models perform within an integrated simulation scenario, readers may refer to (Viswanathan and Holden, 2019). The present paper fills in for the lack of theoretical and implementational details for the wave and current component-models in the above work.

Theory and implementation of component-models for non-diffracting floating objects and for mooring forces based on the quasi-static catenary approach, used in (Viswanathan and Holden, 2019), is discussed in (Viswanathan and Holden, 2020), along with comparison of results for the same system modelled in the commonly used ocean-engineering software Orcaflex. Satisfactory agreement of surge/heave responses, and of Morison forces under various combinations of wave and current loading is demonstrated in (Viswanathan and Holden, 2020), and these maybe taken as proof for the correct representation of wave-current kinematics by the component models discussed in this work.

# 6 Acknowledgements

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

# References

Thomas Lykke Andersen and Peter Bak Frigaard. *Lecture Notes for the Course in Water Wave Mechanics*. Department of Civil Engineering, Aalborg University. DCE Lecture notes, No.24, 2011. URL vbn.aau.dk/en/publications/lecture-notes-for-the-course-in-water-wave-mechanics(69731932-7a17-47ea-b557-6b9e0c81050f).html.

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

Robert G. Dean and Robert A. Dalrymple. *Water Wave Mechanics for Engineers and Scientists*. Allied Publishers Limited, Mumbai, India, 2001. ISBN 81-7764-195-6.

Odd M. Faltinsen. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press, 1999. ISBN 0-521-45870-6.

Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Chichester, United Kingdom, 2011. ISBN 97-8111-999-1496.

Harald E. Krogstad and Oivind A. Arntsen. *Lecture Notes on Linear Wave Theory- Part A- Regular Waves*. Norwegian University of Science and Technology, Trondheim, February 2000. URL folk.ntnu.no/oivarn/hercules_ntnu/LWTcourse/.

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/.

Alexandra H. Techet. *Free Surface Waves- Handout*. Massachusetts Institute of Technology, 2005. URL web.mit.edu/2.016/www/handouts/Free-Surface-Waves.pdf.

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, OMAE2019-95054, June, 2019. URL doi.org/10.1115/OMAE2019-95054.

Savin Viswanathan and Christian Holden. Modelica component-models for non-diffracting floating objects and quasi-static catenary moorings. *Proceedings of the American Modelica Conference*, March, 2020. The referring paper and the referred paper are part of the proceedings of the same conference.

# Modelica Component Models for Non-diffracting Floating Objects and Quasi-static Catenary Moorings

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 theory behind determining the hydrodynamic response of a floating object in the presence of waves is discussed, followed by a simplification for the case of wave-transparent objects. The Morison equation is introduced as a means to estimate lateral wave and current loads on slender bodies. The quasi-static catenary approach to determine mooring forces is then discussed. Development of *Modelica* component-models to simulate the hydrodynamic response of free-floating and catenary-moored non-diffracting objects, in the presence of waves and depth varying current, is then dealt with in detail, and the results dicussed.

*Keywords: hydrodynamics of non-diffracting floating objects, quasi-static catenary mooring, Modelica ocean-engineering library.*

## 1 Introduction

The advantages of developing an *OpenModelica* ocean-engineering library populated with domain-specific *component-models* and *functions* to carry out the integrated simulation of multi-pyhsical ocean engineering systems was demonstrated by the authors (Viswanathan and Holden, 2019). This earlier work:

1. Gives a brief description of the simulation of systems based on the hydrodynamic response of catenary-moored non-diffracting floating objects in the presence of waves and current,

2. Demonstrates the satisfactory agreement of the *Modelica* simulation results with those obtained using a popular ocean-engineering commercial software (*Orcaflex*), and

3. Brings out the advantages of using a component-model based simulation approach.

The voluminous nature of the earlier work precluded the possibility of delving into the theoretical and implementational details of the various *Modelica* component-models of the ocean-engineering library proposed by the authors, the preliminary version of which is available for download at `github.com/Savin-Viswanathan/OELib_OMAE2019`.

(Viswanathan and Holden, 2020) gives a detailed description of the development of *Modelica* component-models to simulate regular as well as irregular waves and depth-varying current. This present work elaborates on the theoretical and implementational details of the component-models for non-diffracting floating objects, and catenary mooring based on the quasi-static approach.

## 2 Theory

### 2.1 Hydrodynamics

Considering the steady-state interaction of a floating object with a regular wave, the loads acting on the body may be considered to be comprised of:

- The fluid pressure loads due to the *incident* wave acting on the body, which is assumed fixed at its mean position.

- The fluid pressure loads due to the *scattered/diffracted* wave from the body, which is assumed fixed at its mean position.

- The fluid pressure loads due to the *radiated* wave system set up by the body as it oscillates in its six Degrees-of-Freedom (DoF) in calm water.

An illustration of the *Diffraction-Radiation* problem is given in Figure 1.

The loads due to the incident wave are referred to as *Froude-Krylov* loads, and those due to the scattered wave are referred to as *diffraction* loads in p. 39 of (Faltinsen, 1999), and this is the convention followed in this work. Another widely used convention is to refer to the combined incident and scattered wave-problem as the *diffraction* problem as in p. 288 of (Newman, 1989).

The Froude-Krylov and diffraction loads taken together constitute the wave excitation loads and may be determined by integrating the incident and diffracted wave dynamic pressures over the mean wetted hull surface. The integration of the dynamic component of the radiation wave pressures give the associated hydrodynamic loads commonly referred to as *added-mass* and *damping*, while the integration of the hydrostatic component gives the *restoring* loads. The added-mass loads are in phase with the body acceleration and the damping loads are in phase with the body velocity.

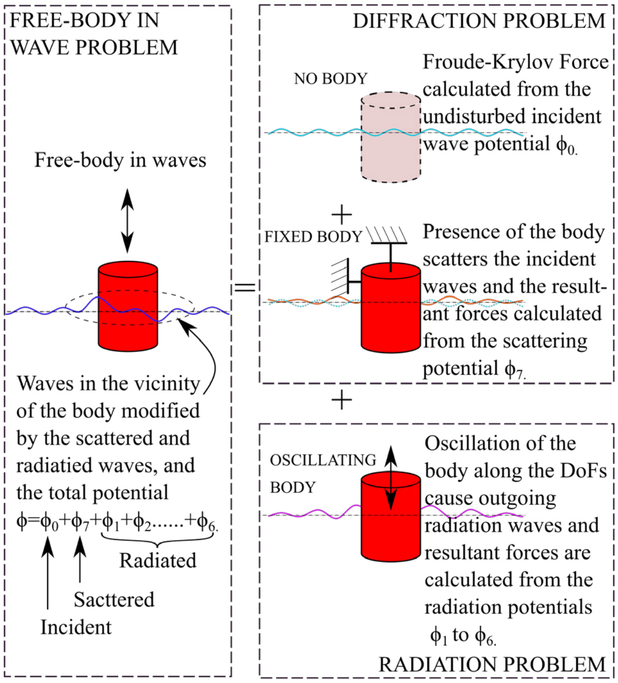

FREE-BODY IN
WAVE PROBLEM
Free-body in waves
Waves in the vicinity of the body modified by the scattered and radiatied waves, and the total potential
$\phi=\phi_0+\phi_7+\phi_1+\phi_2......+\phi_6.$
Radiated
Sacttered
Incident
=
DIFFRACTION PROBLEM
NO BODY
Froude-Krylov Force calculated from the undisturbed incident wave potential $\phi_0$.
+
FIXED BODY
Presence of the body scatters the incident waves and the resultant forces calculated from the scattering potential $\phi_7$.
+
OSCILLATING BODY
Oscillation of the body along the DoFs cause outgoing radiation waves and resultant forces are calculated from the radiation potentials $\phi_1$ to $\phi_6$.
RADIATION PROBLEM

**Figure 1.** The diffraction-radiation problem. Adapted from (Faltinsen, 1999)

The assumption of inviscid, incompressible fluid and irrotational flow implies the existance of a velocity potential which may be determined by solving the associated linearized boundary value problem (BVP). The solution to the BVP for determining the incident wave potential is discussed in (Viswanathan and Holden, 2020). Each of the wave systems above may thus be described by its respective velocity potential, and within the assumption of linearity, the total potential $\phi$ [m$^2$/s] in the vicinity of the body is given by

$$\phi = \phi_0 + \phi_7 + \sum_{j=1}^{6} \dot{\eta}_j \varphi_j. \tag{1}$$

Here, $\phi_0$ and $\phi_7$ [m$^2$/s] are the incident and diffracted wave velocity potentials, respectively, and the summation term represents the radiation wave potential, where $\dot{\eta}_j$ [m/s] is the body velocity along the respective DoF, and $\varphi_j$ [m] is the spatial component of the complex velocity-potential due to the body oscillation with unit velocity in the corresponding DoF; see (Faltinsen and Michelsen, 1974).

Numerical solutions to the diffraction-radiation problem may be sought through the Boundary Element Method or through the Harmonic Polynomial Cell method, to determine the hydrodynamic coefficients. See (Newman and Lee, 2002) and (Shao and Flatinsen, 2014).

### 2.1.1 Simplifications in the case of a small-volume structure

When the size of the structure is large, the diffraction forces are significant, and hence, one must solve the diffraction-radiation problem to determine the wave loads. However, when the structure is relatively small compared to the incident wave-length, depending on the relative significance of the inertia and drag forces, one may utilize the Froude-Krylov theory or the Morison equation to determine the wave loads. The Froude-Krylov theory is applicable for a relatively small structure when drag forces are small compared to the inertia forces. When the drag forces are significant, one has to use the Morison equation; see p. 168 of (Chakrabarti, 1987).

Considering the case of a vertical cylindrical buoy, drag forces are not signifcant for motions in the vertical plane, and hence the Froude-Krylov theory can be used to calculate the vertical wave-loads. However, for motion in the horziontal plane, drag forces become significant and hence, the Morison equation should be used to determine the horizontal wave-loads.

**2.1.1.1 The Froude-Krylov Force** For small-volume upright cylindrical structures, the long wave approximation is applicable for $L > 5D$, where $L$ [m] is the wavelength and $D$ [m] is the diameter of the cylinder. Considering the translational DoFs, i.e., $i = 1,2,3$, the force on the relatively small body may then be expressed as:

$$\mathbf{F} = \mathbf{i}F_1 + \mathbf{j}F_2 + \mathbf{k}F_3 \tag{2}$$

$$\text{where } F_i = -\iint_{S_{0B}} pn_i \, \mathrm{d}s + A_{i1}a_1 + A_{i2}a_2 + A_{i3}a_3. \tag{3}$$

Here, $p$ [N/m$^2$] is the undisturbed incident wave pressure, and $\mathbf{n}$ = ($n_1$, $n_2$, $n_3$) is the unit normal vector to the body surface, defined to be positive into the fluid. The integral is over the average wetted surface of the body. Furthermore, $a_1$, $a_2$, and $a_3$ [m/$s^2$] are the acceleration components along the $x$, $y$, $z$ directions of the undisturbed wave field, and are to be evaluated at the geometrical mass centre of the body. $A_{i1}, A_{i2}, A_{i3}$ [kg] are added-mass terms. **i, j, k** are unit vectors along $x$, $y$, $z$. We also note that the wave generation capability of the body is very small when the long-wave approximation holds true, and hence, the potential damping terms may be neglected; see pp. 60–61 of (Faltinsen, 1999).

The first term of (3) is the Froude-Krylov force, the vertical component of which is approximated as

$$F^z_{FK} \approx \rho g A_{wp} \eta. \tag{4}$$

Here, $\rho$ [kg/m$^3$] is the water density, $g$ [m/s$^2$] is the acceleration due to gravity, $A_{wp}$ [m$^2$] is the water-plane area, and $\eta$ [m] is the sea surface elevation (SSE) about the mean sea level. See (Techet, 2005).

The sea surface elevation $\eta$, at any $x$ co-ordinate, may be expressed as:

$$\eta(x,t) = \sum_{i=1}^{N} \zeta_{0i} \cos(k_i x - \omega_i t - \varepsilon_i). \tag{5}$$

Here, $N$ is the total number of wave components (frequency bands), $\zeta_{0i}$ [m] is the component wave amplitude,

$\omega_i$ [rad/s] is the wave angular frequency, and $\varepsilon_i$ [rad] is the phase. Subscript $i$ refers to the number of the component-wave under consideration. The wave number $k_i$ [rad/m] is to be determined from the dispersion relation. For details, see (Viswanathan and Holden, 2020).

The SSE for a regular wave can be expressed by taking $N = 1$ in (5).

**2.1.1.2 The Morison Equation** Though initially formulated to calculate the horizontal wave-forces on fixed, surface-piercing vertical piles, where $D \ll L$, and the drag forces significant, the Morison equation has since been adapted to determine wave loads on oscillating slender structures. A thorough treatment of the subject can be found in Chapter 6 of (Chakrabarti, 1987).

The horizontal wave and current loads per unit length, on a cylindrical object free to move in presence of waves and current, may be determined from

$$\begin{aligned} M_F^x =& C_M^x \rho \frac{\pi}{4} D^2 \dot{u} - C_A^x \rho \frac{\pi}{4} D^2 \ddot{x} \\ &+ C_D^x \frac{1}{2} \rho D \mid u \pm U - \dot{x} \mid (u \pm U - \dot{x}). \end{aligned} \tag{6}$$

Here, $M_F^x$ [N] is the Morison force, $C_M^x$ [-] is the inertia coefficient, $\rho$ [kg/m$^3$] is water density, $D$ [m] is the body diameter, $\dot{u}$ [m/s$^2$] is the wave-induced water-particle acceleration along $x$, $C_A^x$ [-] is the added-mass coefficient , $\ddot{x}$ [m/s$^2$] is the body acceleration along $x$, $C_D^x$ [-] is the drag coefficient, $u$ [m/s] is the wave-induced water-particle velocity along $x$, $U$ [m/s] is the current velocity along $x$, and $\dot{x}$ [m/s] is the body velocity along $x$. $C_M^x$ and $C_D^x$ are available from numerous field and laboratory tests, e.g., (Yeung, 1981), which allows the designer to choose appropriate values; see p. 172 of (Chakrabarti, 1987). Also, $C_M^x = 1 + C_A^x$; see p. 178 of (Chakrabarti, 1987). The wave kinematics are given by (7) and (8); see pp. 48–52 of (Chakrabarti, 1987).

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

$$\dot{u} = \frac{2\pi^2 H}{T^2} \frac{\cosh\,[k(z+d)]}{\sinh\, kd} \sin(kx - \omega t). \tag{8}$$

Here, $H$ [m] is the wave height, $T$ [s] is the wave period, $k$ [rad/m] is the wave number, $z$ [m] is the vertical coordinate of the point at which the wave kinematics are to be calculated, $d$ [m] is the water depth, $x$ [m] is the horizontal coordinate of the point at which the wave kinematics are to be calculated, and $\omega$ [rad/s] is the angular frequency of the wave.

## 2.2 Catenary Mooring

When a floating object is moored by a slack mooring line, the line assumes the shape of a half catenary; see p. 9 of (Chakrabarti, 1987).

For simplicity, we consider a mooring line that acts in the $x$-$z$ plane. At the point of suspension, the chain tension has a horizontal and vertical component, the magnitudes of which depend on $\psi$ [rad], the angle made by the tangent to the caternary with the horizontal, at the point of suspension. The horizontal component of this force prevents the drifting of the floater in the direction away from the anchor position. In the absence of other external forces, the floater drifts to a position such that the suspended length of the mooring line is vertical. $X$ [m] is the distance from the anchor point to the fairlead on the buoy, and $x$ [m] is the distance from the touch- down-point (TDP) to the fairlead. As the buoy drifts away from the anchor, the TDP moves towards the anchor, and vice-versa, as shown in Figure 2. The total chain length is represented by $l_c$ [m] and the suspended length of the chain is represented by $l_s$ [m]. The length of the chain lying on the seafloor is thus $l_f = l_c - l_s$.

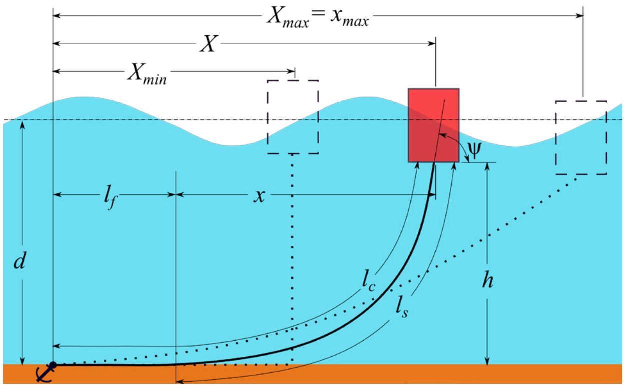


**Figure 2.** The mooring half-catenary.

From (Tatum, 2004) we have the following relations:

$$a = \frac{T_H}{w} \tag{9}$$

$$z = a \cosh\left(\frac{x}{a}\right) \tag{10}$$

$$l_s = a \sinh\left(\frac{x}{a}\right) \tag{11}$$

$$z = a \sec(\psi) = a + h \tag{12}$$

$$z^2 = l_s^2 + a^2. \tag{13}$$

Here, $a$ [m] is the catenary parameter, $x$, $z$ [m] are catenary co-ordinates, $T_H$ [N] is the horizontal tension, and $w$ [N/m] is the submerged specific weight of the catenary.

From (MIT, 2011), we have:

$$T_H = \frac{xw}{\cosh^{-1}\left(1 + \frac{wh}{T_H}\right)} \tag{14}$$

$$l_s = h\sqrt{\left(1 + \frac{2T_H}{wh}\right)}. \tag{15}$$

Considering that we are dealing with the response to linear waves of small-volume structures with small draughts and even smaller variations in draughts, and that the water depth is very large when compared to the draught, we note that when $X = X_{min} \approx (l_c - d)$, $x \approx 0$, and when $X = X_{max} \approx \sqrt{l_c^2 - d^2}$, $x = x_{max}$.

Equation (14) can be iterated to get values of $T_H$ for $x \in \{0, 0.1, 0.2........x_{\max}\}$. We may then use the relation

$$X = l_c - l_s + x \tag{16}$$

to generate a look-up table of horizontal tension values for different horizontal positions of the floater w.r.t. the anchor position. The horizontal tensions for intermediate positions may then be determined by interpolation.

Once the instantaneous horizontal tension values are determined, we may use (10) and (12) to calculate the instantaneous suspended length of the chain, the submerged weight of which is the vertical tension at the fair lead.

The mooring line is also subject to Morison forces, both in the horizontal and vertical directions. To determine these forces, the chain is discretized into a number of segments, and the horizontal and vertical Morison loads are calculated at the mid-points of such segments, and summed up to get the loading on the entire chain. It is assumed that the catenary shape is not affected by such loads, and the sole effect of the fluid loading is a modification in the horizontal and vertical tension values for a given configuration.

The Morison loads per unit length in the normal and tangential directions to a segment, at its mid point, is calculated using equations (17) and (18). This method allows for the separate specification of $C_D$ and $C_M$ values, experimental values of which are scarce when the structures are inclined; see p. 205 of (Chakrabarti, 1987). This method is widely used, and is referred to as the *cross-flow* principle in p. 166 of (Orcina, 2010).

$$\begin{aligned} M_F^n =& C_M^n \rho \frac{\pi}{4} D^2 a_w^n - C_A^n \rho \frac{\pi}{4} D^2 a_l^n \\ &+ C_D^n \frac{1}{2} \rho D \mid v_w^n \pm U^n - v_l^n \mid (v_w^n \pm U^n - v_l^n). \end{aligned} \tag{17}$$

$$\begin{aligned} M_F^t =& C_M^t \rho \frac{\pi}{4} D^2 a_w^t - C_A^t \rho \frac{\pi}{4} D^2 a_l^t \\ &+ C_D^t \frac{1}{2} \rho D \mid v_w^t \pm U^t - v_l^t \mid (v_w^t \pm U^t - v_l^t). \end{aligned} \tag{18}$$

Here, superscripts $n$ and $t$ denote the normal and tangential directions, and subscripts $w$ and $l$ denote the water-particle and the mooring-segment. Further, $a$ [m/s$^2$] is the acceleration, $v$ is the velocity, $U$ [m/s] is the magnitude of the current velocity, and the equivalent-line diameter is $D = 1.8 d_{cw}$ [m], where $d_{cw}$ [m] is the diameter of the chain wire. See p. 303 of (Orcina, 2010).

The determination of the position of the link mid-points would require the approximation of the quasi-static catenary shape from the horizontal tension at each time step. This is effected by discretizing the mooring length $l_c$ into a number of segments, and determining the position of each node connecting the segments using (10)–(12). Once the mid-point positions at each time step are located, the link velocity and accelerations can be expressed as time derivatives of the displacement.

The wave-induced water-particle kinematics at the link mid-point maybe calculated from (7), (8), (19), and (20); see pp. 48–52 of (Chakrabarti, 1987):

$$w = \frac{\pi H}{T} \frac{\sinh[k(z+d)]}{\sinh(kd)} \sin(kx - \omega t) \tag{19}$$

$$\dot{w} = -\frac{2\pi^2 H}{T^2} \frac{\sinh[k(z+d)]}{\sinh(kd)} \cos(kx - \omega t). \tag{20}$$

The current velocities at the required positions can be interpolated from the specified current profile. The wave-induced kinematics and the current profile are moved with the SSE. See p. 221 of (SINTEF, 2014).

Once $M_F^n$ and $M_F^t$ are determined for each link, they are resolved into their horizontal and vertical components, and summed up, to get the total horizontal and vertical forces acting on the mooring chain at each time step. These are then summed up with the vertical and horizontal mooring tension values to get the modified values with fluid loading as $F_M^x$ and $F_M^z$ [N].

### 2.3 The Equations of Motion

Having determined the loads acting on the cylindrical buoy, we may express the equations of motion (EoM) in the horizontal and vertical directions as:

$$M^x \ddot{x} + C^x \dot{x} + K^x x = M_F^x + F_M^x \tag{21}$$

$$M^z \ddot{z} + C^z \dot{z} + K^z z = F_{FK}^z + m_a^z \dot{w}_{cb} + F_M^z \tag{22}$$

Here $M$ [kg] is mass, $C$ [Ns/m] is the damping, $K$ [N/m] is the restoring force, $M_F$ is the Morison load on the buoy, $F_M$ [N] is the mooring load, $m_a$ [kg] is the added-mass, and $\dot{w}_{cb}$ [m/s$^2$] is the vertical wave induced water particle acceleration evaluated at the vertical centre of buoyancy of the buoy. Superscripts $x$ and $z$ denote the horizontal and vertical directions. In (21), the added mass load is included in the Morison force term given by (6).

## 3 Modelica Implementation

Flow-charts in the sub-sections that follow have been prepared with ocean engineers, most likely to be unfamiliar with *Modelica*, in mind, and some elements might appear superfluous to the *Modelica* savvy reader.

The general considerations in the implementation of *Modelica* component-models for integrated simulation of ocean engineering systems, the implementation of component-models to simulate waves and depth-varying current, and the method of linking the generated outputs to a universal data bus have been discussed in (Viswanathan and Holden, 2020).

The following data are thus available at the **EnvironmentBus**:

- $\omega[{}_n\omega_i]$, vector of component wave frequencies.
- $T[{}_n\omega_i]$, vector of component wave Time periods.
- $k[{}_n\omega_i]$, vector of component wave numbers.

- $\varepsilon[{}_n\omega_i]$, vector of component wave phases.
- $\zeta_{0i}[{}_n\omega_i]$, vector of component wave amplitudes.
- $zcg[n]$, vector of $z$ co-ordinates where current velocities are provided.
- $Ucg[n]$, vector of current velocities at above $z$ co-ordinates.

## 3.1 Non-diffracting Floating Cylinder Component-model

The water depth $d$ [m], water density $\rho_w$ [kg/m$^3$], density of the mooring line material $\rho_c$ [kg/m$^3$], specific mass of the mooring line in air $m_a$ [kg/m], cylinder radius $r$ [m], height $h$ [m], structural mass $m_s$ [kg], ballast mass $m_b$ [kg], vertical centre of gravity position w.r.t. the keel $z_{KG}$ [m], added-mass coefficients $C^x_{ma}$, $C^z_{ma}$ [-], drag coefficients $C^x_D$, $C^z_D$, and damping $C^x$, $C^z$ [kg/s] are specified as parameters. $m_{flg}$ is a parameter to specify if the buoy is free-floating or moored. It is set to 0 in case of a free-floating buoy, and to 1 if moored.

A composite connector **Fairlead[2]**, having two flanges of type *Modelica.Mechanics.Translational.Interfaces.Flange_a*, is specified at the centre of the bottom surface of the buoy, to transfer the horizontal and vertical mooring loads.

A data connector **ebdc** is specified to link the wave and current data from the **EnvironmentBus** to the buoy component-model.

Figure 3 shows the flow-chart for the component-model. Here, $K$ [N/m] is the stiffness, $M$ [kg] is the dry mass of the buoy, $A_{wp}$ [m$^2$] is the water-plane area of the buoy, $z_s$ [m] is the static draught, $z_m$ [m] is the draught considering the mooring line length, $z_{cb}$ [m] is the $z$ co-ordinate of the centre of buoyancy, $z_{fb}$ [m] is the calm-water $z$ co-ordinate of the body CG. $z_1$, $z_2$, $z_3$ [m] are the instantaneous $z$ co-ordinates of the body CG, top surface and bottom surface, respectively. $z$ [m] is the instantaneous vertical displacement from $z_{fb}$ and $x$ [m] is the displacement in the horizontal direction of the body CG, both of which are to be determined from the equations of motion. $v^x$ and $v^z$ [m/s] are the body velocities in the $x$ and $z$ directions, while $a^x$ and $a^z$ [m/s$^2$] are corresponding accelerations. $SSE_x$ [m] is the instantaneous sea surface elevation at the $x$ co-ordinate of the body CG, $a_w$ [m/s$^2$] is the vertical component of the wave-induced water-particle acceleration, calculated at the instantaneous $(x, z_{cb})$ position, and $M^x_F$ [N] is the instantaneous wave-current Morison loading on the buoy. Fairlead[1].$f$ and Fairlead[2].$f$ [N] are the horizontal and vertical components of the mooring load, at the the fairlead. $t$, $T_{sim}$ [s] are the current and total simulation times, respectively.

$SSE_X$ is calculated using (5). The function **wave_awCalculator** returns the value of $a_w$ calculated using (8), with consideration of the *moved* kinematic profile. The function **morisonForceCydlBuoy** returns the value of $M^x_F$. The EoMs along the $x$ and $z$ directions are then solved with the specified initial conditions to determine the body response.

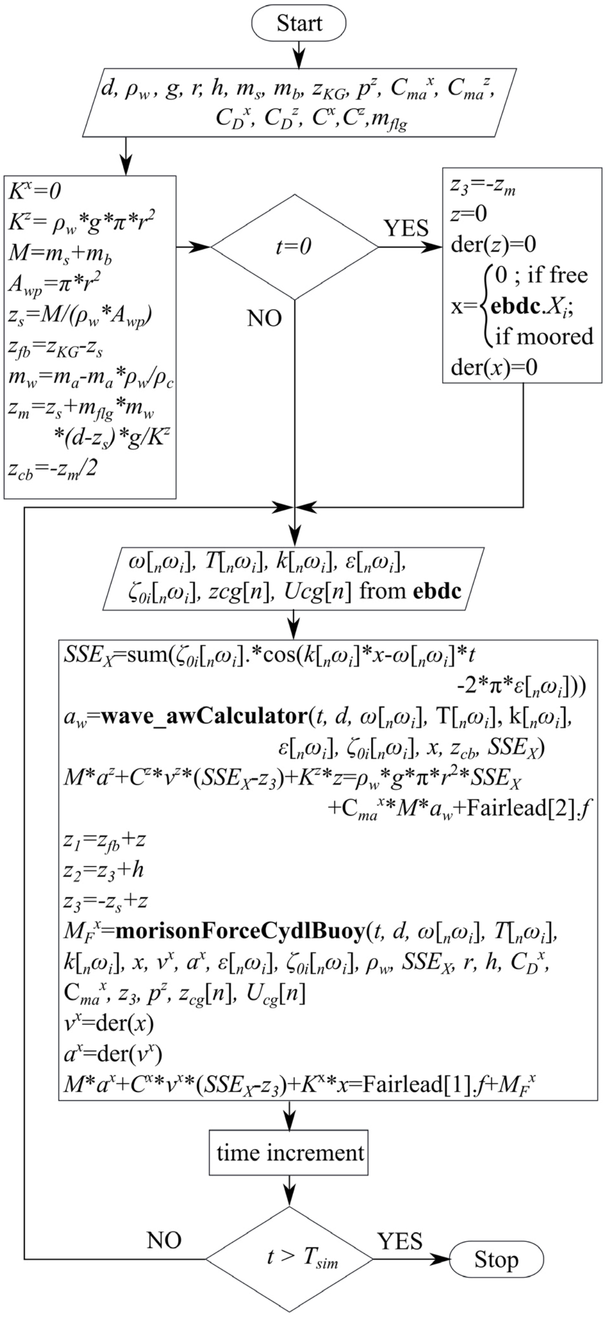

Start
d, ρw, g, r, h, ms, mb, zKG, p^z, Cma^x, Cma^z, CD^x, CD^z, C^x, C^z, mflg
K^x=0
K^z= ρw*g*π*r^2
M=ms+mb
Awp=π*r^2
zs=M/(ρw*Awp)
zfb=zKG-zs
mw=ma-ma*ρw/ρc
zm=zs+mflg*mw *(d-zs)*g/K^z
zcb=-zm/2
t=0
YES
NO
z3=-zm
z=0
der(z)=0
x= 0 ; if free ebdc.Xi; if moored
der(x)=0
ω[nωi], T[nωi], k[nωi], ε[nωi], ζ0i[nωi], zcg[n], Ucg[n] from ebdc
SSEX=sum(ζ0i[nωi].*cos(k[nωi]*x-ω[nωi]*t -2*π*ε[nωi]))
aw=wave_awCalculator(t, d, ω[nωi], T[nωi], k[nωi], ε[nωi], ζ0i[nωi], x, zcb, SSEX)
M*a^z+C^z*v^z*(SSEX-z3)+K^z*z=ρw*g*π*r^2*SSEX +Cma^x*M*aw+Fairlead[2].f
z1=zfb+z
z2=z3+h
z3=-zs+z
MF^x=morisonForceCydlBuoy(t, d, ω[nωi], T[nωi], k[nωi], x, v^x, a^x, ε[nωi], ζ0i[nωi], ρw, SSEX, r, h, CD^x, Cma^x, z3, p^z, zcg[n], Ucg[n]
v^x=der(x)
a^x=der(v^x)
M*a^x+C^x*v^x*(SSEX-z3)+K^x*x=Fairlead[1].f+MF^x
time increment
NO
t > Tsim
YES
Stop

**Figure 3.** Flow-chart for the floating cylinder component-model.

## 3.2 Quasi-static Catenary Mooring Component-Model

Considering space limitations, the flow chart for the component-model **Catenary_Mooring_Mf0**, which does not take into consideration the wave and current loads on the mooring line itself, is given in Figure 4.

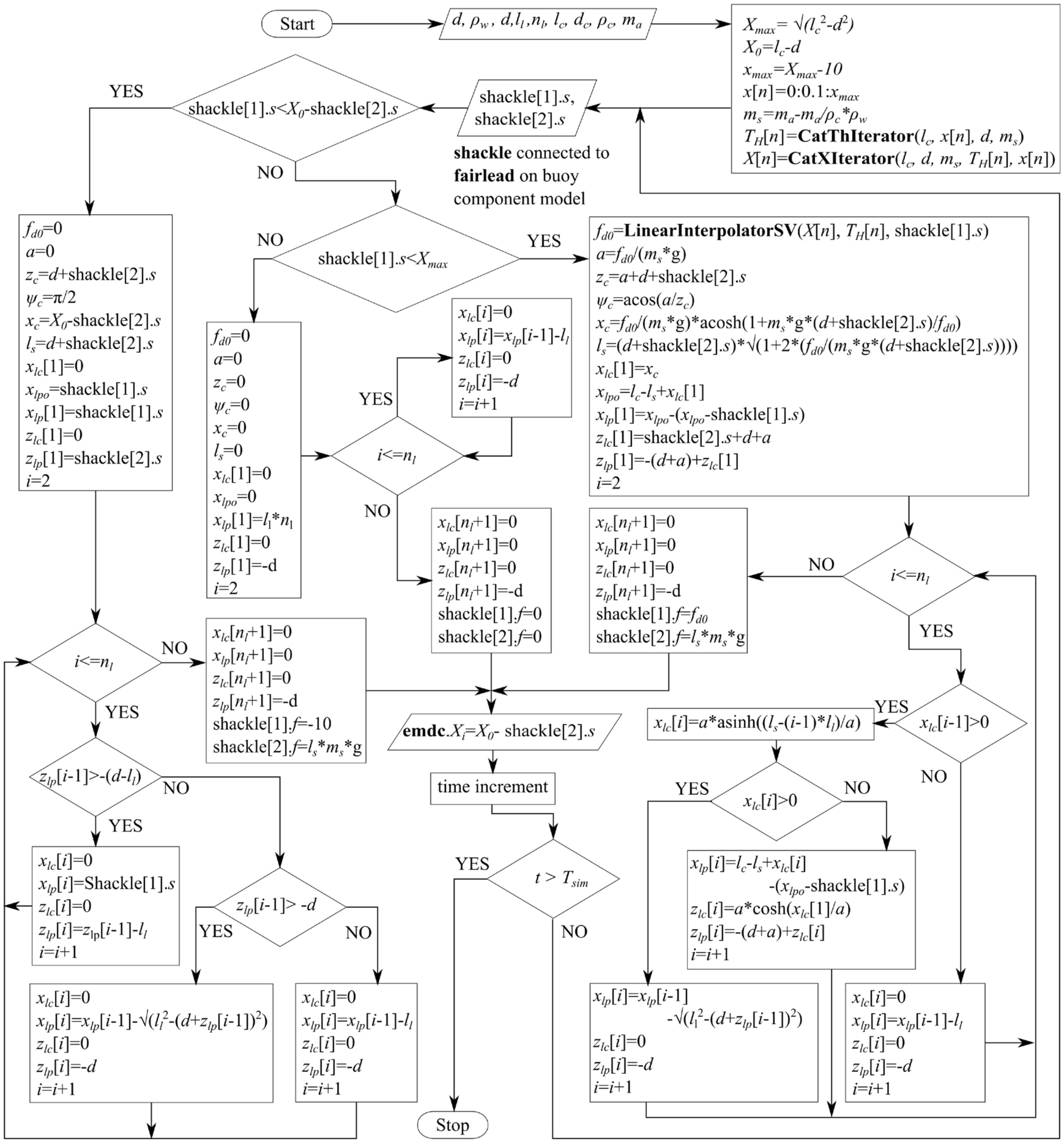


**Figure 4.** Flow-chart for the quasi-static catenary component-model.

A composite connector **Shackle[2]**, having two flanges of type *Modelica.Mechanics.Translational.Interfaces.Flange_b*, is specified at the top end of the catenary to transfer the horizontal and vertical mooring loads. The parameters defined are the water depth $d$ [m], water density $\rho_w$ [kg/m$^3$], the number of segments into which the mooring is discretized $n_l$, the length of each segment $l_l$ [m], the density of the mooring material $\rho_c$ [kg/m$^3$], and the dry specific mass of the chain $m_a$ [kg/m]. Since the **Shackle** is connected to the **Fairlead**, the corresponding positional data are also available.

$X_{max}$, $X_0$, $x_{max}$ [m] are calculated along with the submerged specific mass of the mooring $m_s$ [kg/m]. The vector containing the $x$ positions, where the horizontal mooring load is to be iterated, is defined as $x[n]$. A *function* **CatThIterator** returns the vector $T_H[n]$, containing the corresponding horizontal tension values, calculated based on (14). Another *function* **CatXIterator**, returns the vector $X[n]$ containing the $X$ position corresponding to the $x$ position, as defined in Figure 2.

A data connector **emdc** is specified to link the wave and current data from the **EnvironmentBus** to the mooring. In addition, the initial $x$ co-ordinate of the top end of the mooring line $X_i$ [m], is transmitted to the universal data bus for utilization by the buoy component-model to specify its initial condition.

$f_{d0}$ [N] is the horizontal mooring load, for the given $x$

co-ordinate of the top end of the mooring line, $a$ is the catenary parameter, $z_c$ [m] is the $z$ co-ordinate of the top end of the catenary, in the local co-ordinate system of the catenary, the origin of which lies at a distance of $a$ [m] below the bottom-most point of the catenary, as described in (Tatum, 2004). $\psi_c$ [rad] is the slope of the top-most catenary segment, $x_c$ [m] is the $x$ co-ordinate of the top-most point of the catenary, and $l_s$ [m] is the suspended length of the catenary.

$x_{lc}$ and $z_{lc}$ are vectors holding the $x$ and $z$ co-ordinates of the end points of the segments, in the local co-ordinate system of the catenary. $x_{lp}$ and $z_{lp}$ are the vectors holding the $x$ and $z$ co-ordinates of the end points of the segments, in the global coordinate system. $x_{lpo}$ is the plot correction parameter to account for the minor difference between the actual catenary shape with its top end $z$ coordinate corresponding to the instantaneous position of the buoy keel, and the catenary shape which is back-calculated based on the horizontal tension value from the look-up table, which is in turn based on the $z$ co-ordinate of the top end of the catenary lying at the sea-surface.

If the $x$ co-ordinate of the shackle is less than $X_{min}$ [m], then a small force in the positive $x$ direction is applied to restore the buoy to a region where the mooring model is valid. The vertical mooring load is then the weight of the vertically suspended length of the mooring. For the rare cases when shackle[1].$s < X_{min}$, the plot of the mooring shows a vertically suspended-length instead of the actual shape.

When the $x$ co-ordinate of the shackle is between $X_{min}$ and $X_{max}$, the horizontal mooring load is the corresponding value interpolated from the lookup-table using a *function* **linearInterpolatorSV**, and the vertical load is based on the suspended length back-calculated from the horizontal load. The plot of the mooring line shows the catenary shape, back-calculated from the horizontal load, and corrected using $x_{lpo}$.

If the loads on the buoy exceed the capacity of the mooring line, then the $x$ co-ordinate of the shackle exceeds $X_{max}$, the catenary is assumed to be detached from the buoy, and would lie extended on the sea-floor.

### 3.2.1 Current and Wave Loads on the Mooring Line

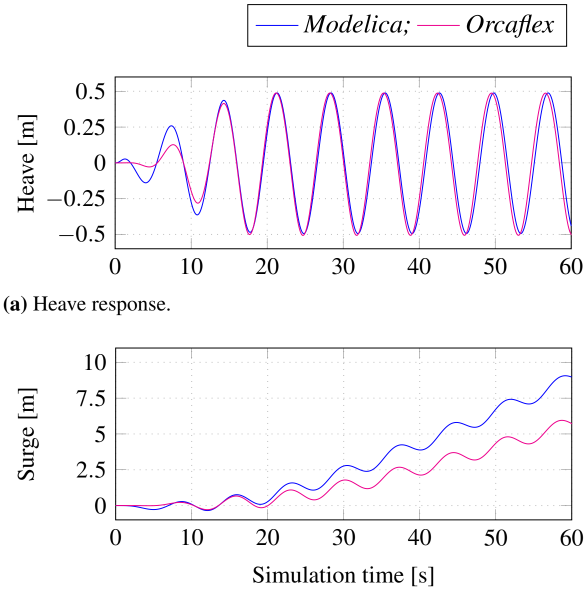
Simulation of current and wave Morision loads on the mooring line is based on the theory given in Section 2.2. The methodology is similar to the one represented by Figure 4, with additional loops for determining fluid and structure velocities and accelerations, and is easily discernible from the code. **Catenary_Mooring_MfC** considers the Morison loads due to current and mooring velocities, while **Catenary_Mooring_MfCW** considers the loads due to current, wave, and mooring line velocities and accelerations.

## 4 Results

The simulation files for all results discussed below are available at `github.com/Savin-Viswanathan/Modelica2020-b`. Comparison results using *Orcaflex* are also presented here.

Figure 5a shows the heave response of a cylindrical buoy of $r = 0.6$ [m], $h = 2$ [m], $m_s = 350$ [kg], $m_b = 500$ [kg], in a water depth of $d = 50$ [m], when subjected to a regular wave with $H_r = 1$ [m] and $T_r = 7$ [s], with $T_{del} = 0$ [s], and $T_{rmp} = 20$ [s]. We have assumed $C^x_{ma} = C^z_{ma} = 1$, $C^x = 0$ [kg/s], and $C^z = 3100$ [kg/s].

Figure (4) in (Viswanathan and Holden, 2019) had shown the same results for heave, and we had noticed a slight discrepancy between the *Modelica* and *Orcaflex* results. The cause was identified to be an error in the treatment of the added mass term in (4) of (Viswanathan and Holden, 2019), and has been corrected based on the theory described here in Section 2.1.1.1. Figure 5b shows the surge response.



**(a)** Heave response.

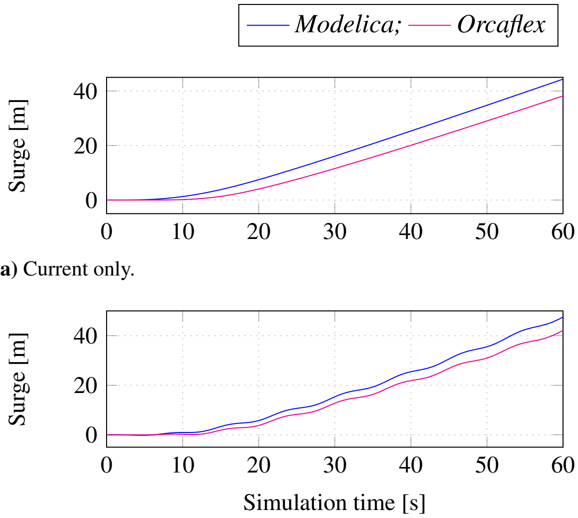
**(b)** Surge response.

**Figure 5.** Unmoored cylindrical buoy in waves.

Figure 6a shows the surge response of the above buoy, in the presence of a uniform current of 1 [m/s] in the $x$ direction, while Figure 6b shows the same in presence of both the above wave and current.

Figure 7a shows the horizontal Morison loads on a fixed buoy with same properties as the earlier one, but with a draught of 1 [m], when subjected to a uniform current of 1 [m/s]. Figure 7b shows the surge Morison loads when only a regular wave, with the same parameters as above, acts on the fixed buoy, and Figure 7c shows the Morison loads when both the current and the wave acts on the buoy.

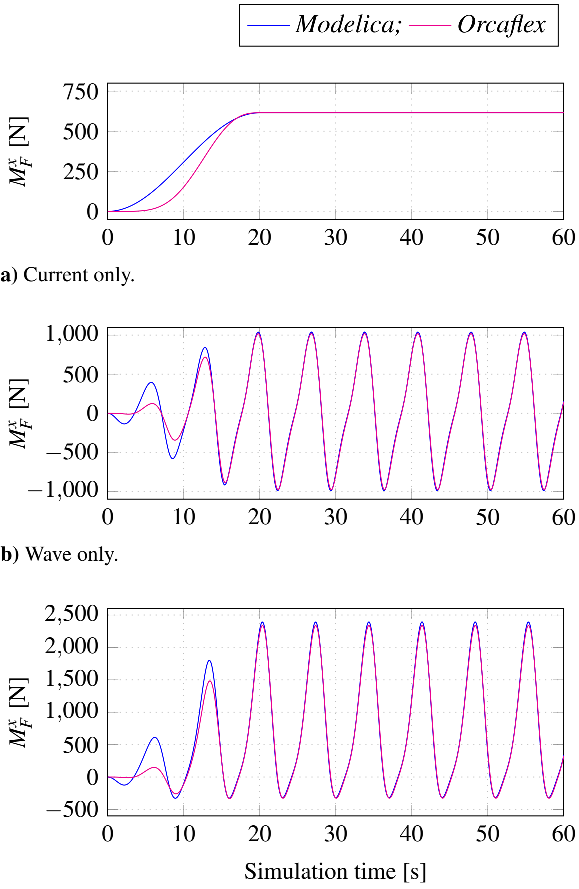
From Figure 7, we observe that the Morison loads are a close match, and hence, the difference between *Modelica* and *Orcaflex* values in Figures 5b, 6a, and 6b, can be attributed to the difference in the way in which the loads are ramped up during the start of simulation, as evid-



**(a)** Current only.

**(b)** Wave and current.

**Figure 6.** Surge response of an unmoored cylindrical buoy.



**(a)** Current only.

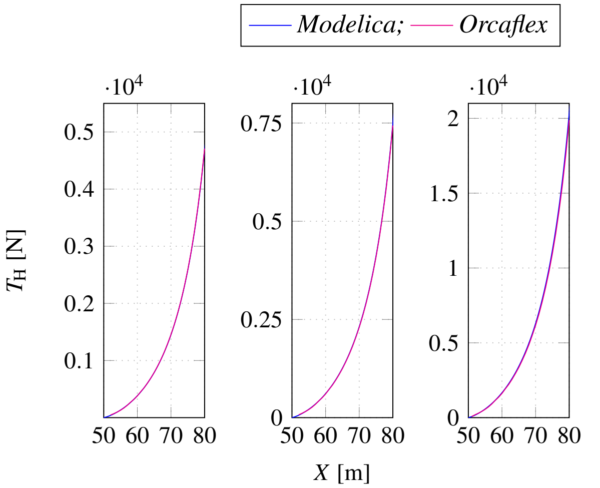
**(b)** Wave only.

**(c)** Waves and current.

**Figure 7.** Morison loads on a fixed cylinder.

ent from Figure 7a. *Orcaflex* uses *vertical-stretching* of the water-particle kinematics, while the present *Modelica* model employs *moved* kinematic-profiles, and this could be the cause of the minute difference in peak values of the Morison loads in Figures 7b and 7c.

*Orcaflex* uses a lumped-mass and spring-damper model for the mooring lines, while mooring forces in the present work are based on the quasi-static catenary theory. Figure 8 shows the horizontal tension values given by *Orcaflex* and *Modelica* models for different $X$ positions, for moorings of different specific masses. The horizontal tensions are a close fit, with *Modelica* giving slightly higher values than *Orcaflex*. e.g., for a chain with specific mass of 16 [kg/m], at $X = 70$ [m], $T_H$ in *Modelica* is $2,336$ [N] and $2,299$ [N] in *Orcaflex*. The mooring horizontal tensions from *Orcaflex* were determined by placing the top end of the line at different $X$ positions along the free surface manually and a small error in the values had occured in Figure 10 of (Viswanathan and Holden, 2019).



**(a)** $m_a = 10$ kg/m **(b)** $m_a = 16$ kg/m **(c)** $m_a = 43.5$ kg/m

**Figure 8.** Horizontal tensions for mooring chains with different specific masses ($m_a$).

Figure 9 shows the shape of a mooring line, with specific mass 10 [kg/m], in *Modelica* and *Orcaflex* when the top end is placed at $X$=60 [m], and at 80 [m] with $z$= 0 [m]. When a uniform current of 1 [m/s] is applied across the full depth of the water-column, the *Orcaflex* line, based on the lumped mass model, deflects under the influence of the current. The deflection is not expected to be large enough to cause considerable difference in the fluid loading experienced by the mooring line. However, it is evident that the static catenary model would not capture the forces that result as a consequence of the dynamics of the mooring line itself.

Figure 10 shows the surge and heave response of the above free-floating buoy, when moored with a mooring line of specific mass 10 [kg/m], under different conditions of wave and current. It was noticed that the model failed to simulate when the acceleration forces due to the fluid and the motion of the chain were considered, using the **CatenaryMooring_MfCW** component model. Hence the

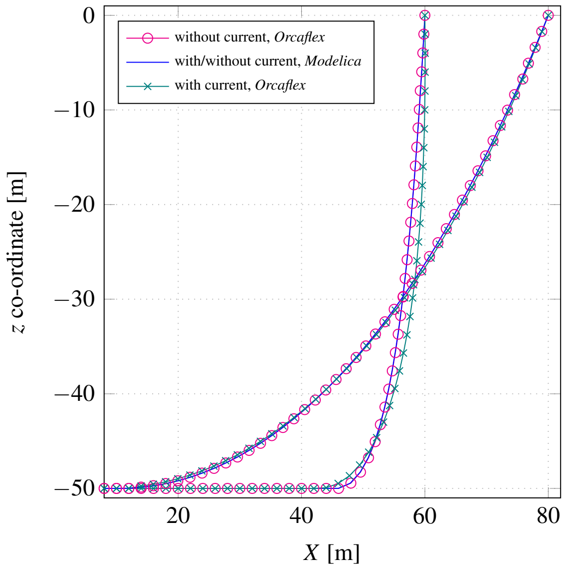


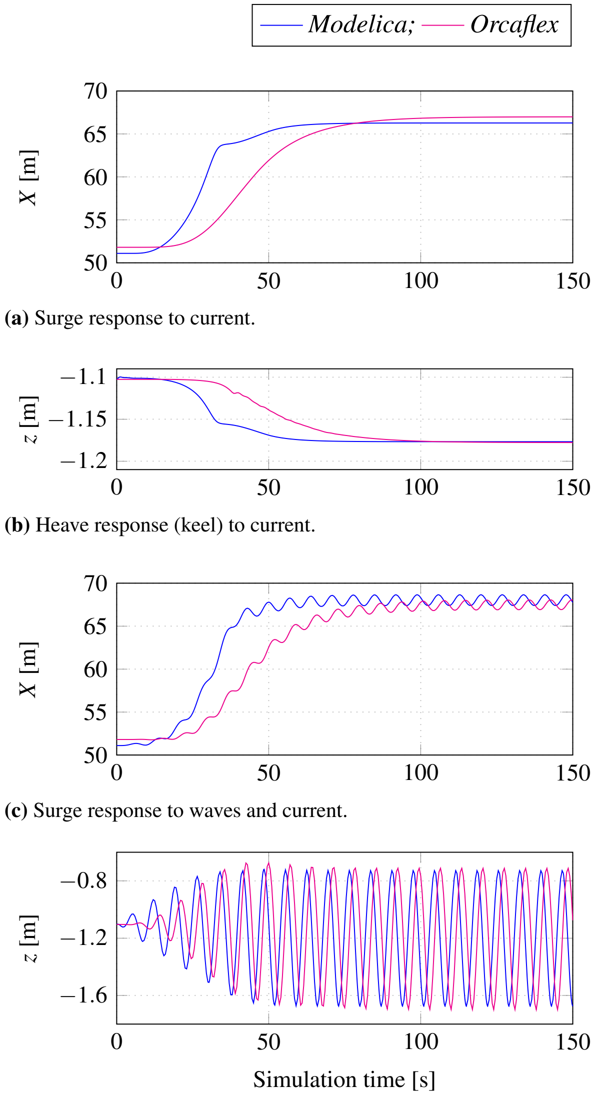
**Figure 9.** Shape of the mooring line.

results shown are with the **CatenaryMooring_MfC** component model. The combined wave-current velocity and accleration loads, and inertial loads due to the structural response of the mooring chain, are inherent in the *Orcaflex* model, while they are not accounted for by the present *Modelica* model. The phase difference in the response between the models may be because of this difference.

When the current velocity is reduced to 0.5 [m/s], **CatenaryMooring_MfCW** could be used for simulation, and the results are shown in Figure 11. However, the model is sluggish with many warnings for non-convergence. This could be due to the discontinuities in the accelerations of the segment mid-points, calculated based on the instantaneous static catenary shape. An example of the vertical component of the acceleration of the mid-point of the second segment from the top-end of the mooring, is shown in Figure 11c.

## 5 Conclusion

Implementation of the theory to develop *Modelica* component-models for a non-diffracting cylindrical object, and for a quasi-static mooring catenary, is described in detail. Simulations to determine the hydrodynamic response of a free-floating cylinder are carried out, and the results compared with a smilar model in *Orcaflex*. It is observed that the heave responses in both cases are in satisfactory agreement. Minor differences in the surge responses are reconciled based on the comparison of Morison forces on a fixed cylinder, under various loading scenarios, and it is concluded that these differences are a result of the differences in the ramp-up functions used in this *Modelica* model and in *Orcaflex*, and hence, do not constitute *errors* in the simulation results.



**(a)** Surge response to current.

**(b)** Heave response (keel) to current.

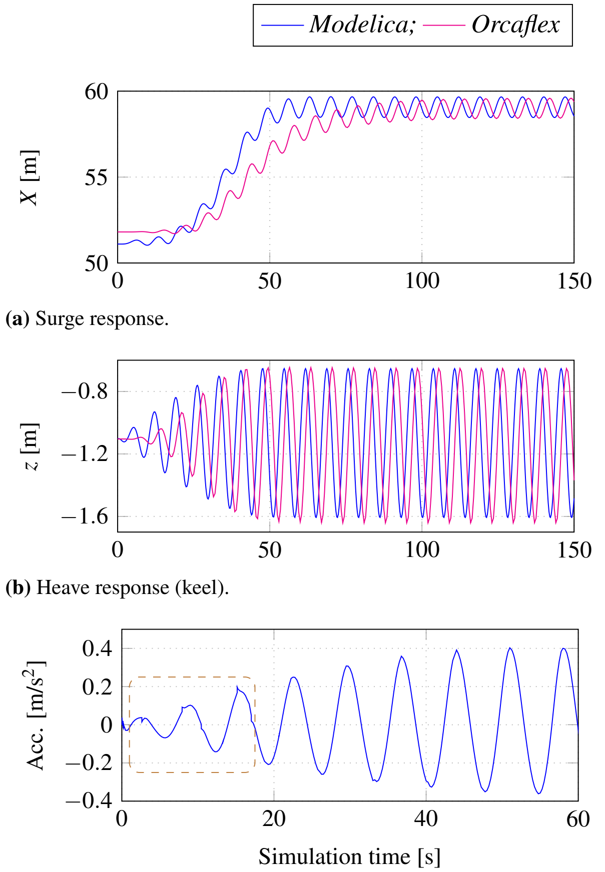
**(c)** Surge response to waves and current.

**(d)** Heave response (keel) to waves and current.

**Figure 10.** Hydrodynamic response of a moored cylindrical buoy.

The static mooring loads, based on the catenary theory in the present *Modelica* model, and those based on the lumped-mass spring-damper system of the *Orcaflex* model, are demostrated to have satisfactory agreement. The comparison between mooring configurations under different loading scenarios points out the probability that differences in the fluid loading of the mooring line, as a result of the deviation of the mooring line from the catenary shape, might not be significant, compared with the contributions from the dynamics of the mooring line itself, which the present *Modelica* model does not capture.

The simulations of a moored floating cylinder further demostrate the satisfactory agreement between this *Modelica* model and a similar *Orcaflex* model. The simulations bring out the deficiencies caused by the assumptions of the quasi-static catenary, which is not in agreement with the actual physics of the system.



**(a)** Surge response.

**(b)** Heave response (keel).

**(c)** Vertical acceleration of the second chain-link from the fairlead.

**Figure 11.** Moored cylindrical buoy in waves and reduced current.

To overcome the deficiency of not being able to account for the mooring line dynamics, and to mitigate issues as seen in Figure 11c, a *Modelica* mooring component-model based on the lumped-mass spring-damper approach is being developed, along with a frequency-domain hyrdodynamic analysis component-model, which would enable the generation of hydrodynamic parameters for diffracting objects. The initial results appear promising and will be the topic of discussion in a future work.

## 6 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.

O.M. Faltinsen. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press, 1999. ISBN 0-521-45870-6.

O.M. Faltinsen and F.C. Michelsen. Motion of large structures in waves at zero froude numbers. *Read at the International Symposium on the Dynamics of Marine Vehicles and Structures in Waves, London*, 1974.

MIT. *Lecture Notes on Mooring Dynamics-II*. Massachusetts Institute of Technology, 2011. URL `ocw.mit.edu/courses/mechanical-engineering/2-019-design-of-ocean-systems-spring-2011/lecture-notes/MIT2_019S11_MD2.pdf`.

J. N. Newman and C. H. Lee. Boundary-element methods in offshore structure analysis. *Journal of Offshore Mechanics and Arctic Engineering*, 124:81–89, May 2002. doi:10.1115/1.1464561.

J.N Newman. *Marine Hydrodynamics*. The MIT Press, Cambridge, Massachusetts, 1989. ISBN 0-262-14026-8.

Orcina. *Orcaflex Manual- Version 9.1a*. Orcina, 2010. URL `citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.721&rep=rep1&type=pdf`.

Yan-Lin Shao and O.M. Flatinsen. A harmonic polynomial cell (hpc) method for 3d laplace equation with application in marine hydrodynamics. *Journal of Computational Physics*, (274):312–314, 2014.

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/`.

Jeremy B. Tatum. *Lecture Notes on the Catenary*. University of Victoria, 2004. URL `astrowww.phys.uvic.ca/~tatum/classmechs/class18.pdf`.

A.H. Techet. *Lecture Notes on the Design Principles for Ocean Vehicles*. Massachusetts Institute of Technology, 2005. URL `ocw.mit.edu/courses/mechanical-engineering/2-22-design-principles-for-ocean-vehicles-13-42-spring-2005/readings/r10_froudekrylov.pdf`.

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, OMAE2019-95054, June, 2019. URL `doi.org/10.1115/OMAE2019-95054`.

Savin Viswanathan and Christian Holden. Modelica component-models for oceanic surface-waves and depth varying current. *Proceedings of the American Modelica Conference*, March, 2020. The referring paper and the referred paper are part of the proceedings of the same conference.

Ronald W. Yeung. Added mass and damping of a vertical cylinder in finite-depth waters. *Applied Ocean Research*, 3(3): 119–133, 1981.

# The Rotorcraft Aerodynamics Library: A Modelica Library for Simulation of Rotorcraft Aerodynamics and Whirl Flutter

Cory Rupp Nicolas Reveles

ATA Engineering, Inc., USA, {cory.rupp, nicolas.reveles}@ata-e.com

## Abstract

Rotor aeromechanics is a multidisciplinary branch in the field of rotorcraft that involves performance, loads, vibrations, stability, and noise. Aeromechanic analysis in this field is a complex task typically requiring specialized software tools to simultaneously resolve the coupled structural, mechanical, and aerodynamic solutions that contain motion (both rigid and elastic) and forces. One drawback of these tools is that they are often overspecialized and do not lend themselves well to the analysis of the latest rotorcraft advances and concepts, including those utilizing distributed electric propulsion. In this paper, a Modelica library is presented that provides an extensible platform for performing analysis and design studies of current and future rotorcraft. Several examples are presented that illustrate how the library can be used to perform aerodynamic and whirl flutter stability analysis as well as control system design for multi-rotor aircraft.

*Keywords: Modelica, rotorcraft, aeromechanics, aerodynamics, whirl flutter, flight control*

## 1. Introduction

Analytic methods for analyzing rotors are frequently capable of adequately characterizing the basic aerodynamics of rotor blades but have shortcomings in addressing more-complex configurations, as such methods are not easily extendable to adequately describe complex geometry, elastic blades, or physics such as time-resolved rotor wakes. In addition, detailed hub mechanics are often missing that can significantly affect rotor behavior and stability. Therefore, numerical methods are frequently preferred for analyzing real-world designs. Several comprehensive rotor analysis tools exist (e.g., CAMRAD-II, RCAS) with varying capabilities intended to overcome the limitations of analytical methods.

Whirl flutter is a phenomenon that principally affects propeller and proprotor aircraft that encounter high rotor inflow velocities and involves the tip path plane precessing (“whirling”) in the form of a dynamic instability. The precession stems from the gyroscopic motion of the rotating blades. This otherwise stable motion is destabilized under certain conditions by aerodynamic forces introduced by changes in the local blade inflow angles due to the precession (Reed, 1967). Analytic methods exist, such as the Houbolt-Reed method (Houbolt and Reed, 1962; Reed and Bland, 1961), that are capable of quantifying stability; however, they are limited by the same factors since general rotor analysis and software is typically used once a basic rotor design is established to capture additional aerodynamic or mechanical effects.

Existing software tools for performing rotor aerodynamics and whirl-flutter analyses are generally tailored specifically to rotor analysis and therefore are limited, whether by the available connections to external tools, the rotor control schemes, or the ability to model effects such as electromechanical interaction for rotors driven by electric motors. Emerging rotorcraft designs, however, are increasingly moving toward multi-rotor and electrified rotor concepts for which these tools typically lack capability to analyze. An example is the NASA X-57 distributed electric propulsion aircraft, which consists of 14 rotors distributed along the wings intended to enhance performance under certain flight conditions. Modelica has previously been used to analyze this complex multi-physics problem (Batteh et al., 2018); however, explicit rotor dynamics and stability were not considered. Existing rotor analysis tools are not designed to handle the coupled multi-rotor aero-electrical-mechanical problem, so alternative analysis approaches are necessary.

This paper presents a new Modelica library, the Rotorcraft Aerodynamics Library (*RotorAeroLib*), that provides a rotor aerodynamics modeling capability and rotor mechanical templates that can be used to analyze various rotor configurations such as those found on helicopters as well as fixed wing, tilt-rotor, and distributed electric propulsion aircraft. By leveraging the capabilities and extensibility of Modelica, this library will enable engineers to perform sophisticated analyses beyond what is capable in existing comprehensive rotorcraft tools.

## 2. The Rotorcraft Aerodynamics Library

The Rotorcraft Aerodynamics Library mixes new modeling capabilities of rotor blade aerodynamics with existing Modelica Standard Library models to provide a

DOI 10.3384/ECP20169111

toolset for coupled aerodynamic analysis and design. The rotor hub topologies used in rotorcraft vary significantly, so it is imperative to have a flexible and extensible modeling technology such as Modelica. The goal of the library is to make building aerodynamic models of rotors easier by providing a set of common building blocks.

The Rotorcraft Aerodynamics Library is organized into aerodynamic models and template implementations of rotor components, including, for example, the rotor blade as well as several example models. The top-level view of the library is shown in Figure 1. The most relevant of the models will be described in the following sections. Some of the template rotor models, e.g., the *RotorBlade* model, make use of the freely available Modelica Deployable Structures Library (Rupp and Schweizer, 2018) (available via the Modalica Association website), which contains an appropriate flexible beam formulation that can be used for rigid for elastic rotor analysis.

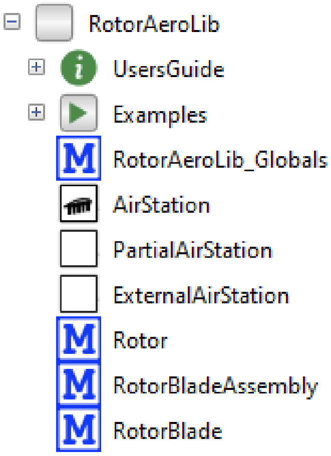


**Figure 1.** Top-level view of the rotorcraft aerodynamics library.

## 2.1. AirStation

At the top level of *RotorAeroLib* is a Modelica model that implements the aerodynamic equations for the rotor problem. This model, the *AirStation*, computes localized aerodynamic lift and drag sectional forces imposed upon a rotor blade as a function of local blade position, velocity, angle of attack, lift and drag coefficients, as well as global terms such as freestream velocity and overall rotor properties as described by blade element theory (BET) (Stepniewski and Keys, 1984).

In BET (also known as strip theory or lifting line aerodynamics), the rotor blade is assumed to act as a one-dimensional beam and is discretized along its length where the aerodynamic equations of each blade cross-section (i.e., an airstation) are solved independently considering local conditions (Leishman, 2006). The resultant forces are then summed along the blade length to derive total thrust and rotor torque.

Following the variable definitions for the airstation aerodynamics shown in Figure 2, the inflow angle $\phi$ is calculated via the perpendicular $U_P$ and tangential $U_T$ air flow speeds via

$$\phi = atan\left(\frac{U_P}{U_T}\right) \tag{1}$$

which is related to the geometric/collective angle $\theta$ and ultimately the angle of attack $\alpha$ via

$$\alpha = \theta - \phi. \tag{2}$$

Lift $\Delta L$ and drag $\Delta D$ forces at an airstation can then be computed from

$$\Delta L = \frac{1}{2}\rho U^2 c C_l \Delta r \tag{3}$$

$$\Delta D = \frac{1}{2}\rho U^2 c C_d \Delta r \tag{4}$$

where

$$U^2 = {U_P}^2 + {U_T}^2 \tag{5}$$

is the total cross-sectional airstream speed (neglecting radial flow), $\rho$ is the density of air, $c$ is the blade chord length, $\Delta r$ is the lengthwise blade discretization size (i.e., the width of the airstation), and $C_l$ and $C_d$ are the airfoil coefficients of lift and drag, respectively, defined here by

$$C_l = C_{l,0} + C_{l,\alpha} * \alpha + C_{l,\alpha^2} * \alpha^2 \tag{6}$$

$$C_d = C_{d,0} + C_{d,\alpha} * \alpha + C_{d,\alpha^2} * \alpha^2. \tag{7}$$

Such analytic forms of sectional lift coefficients are typically obtained from wind tunnel measurements and have been relatively common in rotor aerodynamic analyses since the early days of aviation (Knight and Hefner, 1937). An alternative approach involves defining large tables of sectional aerodynamic data, known as C-81 tables, after the name of the computer software originally developed by Bell Helicopter (Harris, 2012). The C-81 tables typically include sectional lift, drag, and pitching moment as a function of angle of attack, Reynolds number, and Mach number and therefore are nontrivial to construct even with the large collection of experimental data for common airfoil sections available (Abbott and von Doenhoff, 1959). While these tables are undoubtedly more complete, the accessibility of the above analytic forms offer a straightforward implementation in *RotorAeroLib* that could, if desired, be expanded to use a lookup table.

The resulting lift $\Delta L$ and drag $\Delta D$ forces are then transformed into the local *AirStation* frame, which is oriented at the blade geometric angle (Figure 2), using

$$\Delta F_z = \Delta L\, cos(\alpha) + \Delta D\, sin(\alpha) \tag{8}$$

$$\Delta F_y = \Delta L\, cos(\alpha) - \Delta D\, sin(\alpha). \tag{9}$$

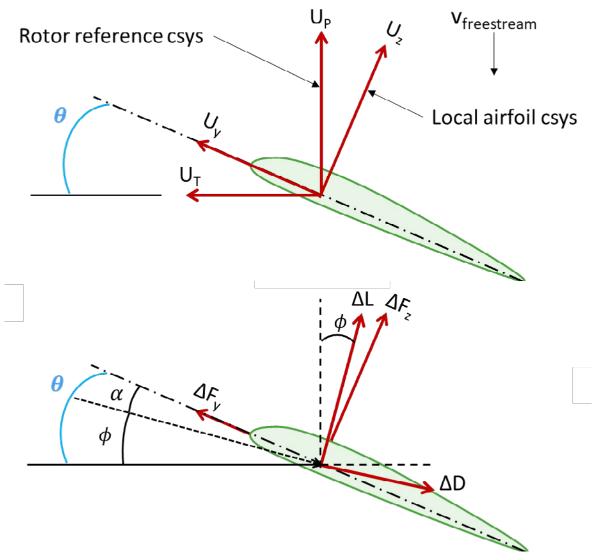


**Figure 2.** Variable definitions for the *AirStation* aerodynamics.

For the *AirStation*, these equations are solved in a reference coordinate system that is assumed to be oriented with X along the nominal blade direction and Z perpendicular to the nominal rotor plane. Thus, this coordinate system will rotate with the blade azimuth, but it is unaffected by flap or lead-lag blade motion. The geometric angle of an *AirStation* is then computed as the relative rotation about the X-axis between the *AirStation* frame and the relative coordinate system. This angle continuously changes as the blade flaps, lead-lags, and feathers and is thus tightly coupled to the blade motion and aerodynamics.

As in BET, it is the intent that several *AirStation* models are distributed along each rotor blade in a discretized fashion so that as the number of *AirStation* models increases, the solution converges to the asymptotic limit. *RotorAeroLib* provides a discretized rotor blade model that serves this purpose, which is described in a later section.

The *AirStation* model extends the *PartialAirStation* model, which contains the basic input/output quantities necessary to provide an interface to an arbitrary aerodynamic solver. Whereas the *AirStation* model provides a pure Modelica implementation of the rotor aerodynamic equations, the *ExternalAirStation* model does the same but provides an external C function interface for implementing aerodynamic loading from an external source such as a computational fluid dynamics solver or one of several high-fidelity rotor aerodynamics software packages. Work utilizing *ExternalAirStation* is ongoing, and the model will be updated within *RotorAeroLib* as functional interface codes are implemented and validated.

In addition to the standard BET equations, the *AirStation* model can optionally include various aerodynamic correction terms to improve the accuracy of the solution. These terms include corrections for induced airflow, Mach effects, tip loss, aspect ratio, unsteady circulatory aerodynamics, and unsteady noncirculatory aerodynamics. Boolean parameters controlled by an outer *RotorAeroLib_Globals*model (to be described next) turn these contributions on and off. Some of the computations for these corrections, e.g., unsteady noncirculatory aerodynamics, will significantly increase computational cost due to the introduction of additional states and nonlinear terms, but some flight conditions require them to ensure accuracy of the results. The following subsections describe each of the aerodynamic terms available in the *AirStation* model.

### 2.1.1. Aerodynamic Parameter useInflowCorrection

One of the shortfalls of BET is that it uses an assumed inflow condition. In the case of the *AirStation* model, the inflow is assumed to be uniform and independent of collective or blade twist, which is inaccurate for many flight conditions. An improvement for axial flight corrects the inflow using blade element momentum theory (BEMT) (Leishman, 2006) in which inflow ratio $\lambda$ and perpendicular airflow speed are corrected at each airstation via the equations

$$\lambda^2 + \left(\frac{\sigma C_{l,\alpha}}{8} - \lambda_c\right)\lambda - \frac{\sigma C_{l,\alpha}}{8}\theta\tilde{r} = 0 \tag{10}$$

$$U_P = V_r + \lambda V_{tip} \tag{11}$$

where $\sigma$ is the rotor solidity, $C_{l,\alpha}$ is the lift curve slope, and $\lambda_c$ is the inflow due to axial climb rate. With BEMT, the inflow ratio is now a function of collective and blade twist, which leads to more accurate solutions.

Although BEMT provides an improved estimate of the rotor disk inflow conditions for axial flight, the inflow is assumed to respond instantaneously to changes in rotor operating conditions. These assumptions are typically appropriate for performance prediction as well as certain stability analyses that involve high inflow conditions (e.g., propellers in forward flight). Although not implemented within the current version of *RotorAeroLib*, more advanced models (Leishman, 2006), such as dynamic inflow and free wake models, exist for the purpose of extending applicability to a broader problem set.

### 2.1.2. Aerodynamic Parameter useAspectRatioCorrection

The rotor aspect ratio is the ratio of the blade length to its mean chord length, defined as

$$A_R = 0.75\frac{R}{c} \tag{12}$$

where $R$ is the blade length (Bland and Bennett, 1963). The 0.75 factor comes from the aspect ratio at the three-quarters radius. The aspect ratio correction factor $F_{AR}$ is defined as

$$F_{AR} = \frac{A_R}{A_R+2} \tag{13}$$

and accounts for tip loss (Bland and Bennett, 1963). This correction factor is applied directly to the *AirStation* lift value.

### 2.1.3. Aerodynamic Parameter useMachCorrection

As a blade rotates, the tangential, and thus total, air speeds it encounters are a function of radial position along its length. Full-scale vehicles commonly have tip Mach numbers in excess of 0.6, warranting the use of the Prandtl-Glauert compressibility correction. The compressibility correction factor takes into account these effects individually at each *AirStation* since each will have a different Mach number (Bland and Bennett, 1963). With the speed of sound $c_s$, the Mach number defined as

$$M = \frac{U}{c_s} \tag{14}$$

feeds into the Mach correction factor $F_M$ as

$$F_M = \frac{1}{\sqrt{1-M^2}} \tag{15}$$

which then gets applied directly to the *AirStation* lift value.

### 2.1.4. Aerodynamic Parameter useTipLossCorrection

Finite lifting wings encounter a phenomenon termed "tip loss," which is caused by the high-pressure air beneath the airfoil sections rolling over the blade tip to the low-pressure side above the airfoil, resulting in a loss of lift. Accounting for these effects in rotary wing aerodynamics is naturally more challenging than fixed wing aerodynamics due to the unsteady aerodynamics. There exists numerous empirical tip loss models (Leishman, 2006; Stepniewski and Keys, 1984) to incorporate these effects into simplified aerodynamic computations. The implementation within *RotorAeroLib* utilizes a tip loss model similar to that used in the Dymore comprehensive solver (Bauchau, n.d.). In the Modelica model, blade section lift is modified by the correction factor

$$F_{tiploss} = tanh\left(\frac{1-\frac{r}{R}}{1-\mu}\right) \tag{16}$$

where $r$ is the radial distance along the blade of the airstation and $\mu$ is a user-prescribed tip loss coefficient, typically selected between 0.95 and 1.0. The resulting tip loss factor is approximately unity near the rotor hub and increases significantly near the blade tip where tip losses are at their greatest. This correction factor is applied directly to the *AirStation* lift value.

### 2.1.5. Aerodynamic Parameter useUnsteadyAero

Rotor aerodynamics tend to be dominated by unsteady effects, especially at the low inflow flight conditions typically encountered by conventional helicopter main rotors. This is a consequence of several rotor phenomena and is largely attributed to a combination of cyclic controls (i.e., commanded blade pitch becomes a function of rotor azimuth), tip vortex trajectories (both periodic and aperiodic), and blade structural dynamics (including pitch, flap, and lead-lag motion, along with local aeroelastic deformations).

One method for incorporating unsteady aerodynamics involves application of Wagner's function. The value of the lift coefficient $C_l$ at angle of attack $\alpha$ can be represented by a state space system approximation of the Wagner function (Leishman and Nguyen, 1990) as

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.01375\left(\frac{2U}{c}\right)^2 & -0.3455\left(\frac{2U}{c}\right)\end{bmatrix}\begin{Bmatrix}x_1\\ x_2\end{Bmatrix} + \begin{Bmatrix}0\\ 1\end{Bmatrix}\alpha(t) \tag{17}$$

$$C_l(t) = C_{l,\alpha}\begin{bmatrix}0.006825\left(\frac{2U}{c}\right)^2 & 0.10805\left(\frac{2U}{c}\right)\end{bmatrix}\begin{Bmatrix}x_1\\ x_2\end{Bmatrix} + \frac{C_{l,\alpha}}{4}\alpha(t) \tag{18}$$

where $C_{l,\alpha}$ is the lift curve slope and $x_1$, $x_2$ are state variables. If selected, this set of equations fully replaces the linear equation for the lift coefficient (eq. 6).

### 2.1.6. Aerodynamic Parameter useUnsteadyNoncircAero

Unsteady noncirculatory terms (also termed "apparent mass" terms) are a result of unsteady blade accelerations. The unsteady blade motion requires a certain mass of air be moved to accommodate the blade's physical position. The acceleration of this mass of air gives rise to the noncirculatory terms.

This introduces a correction $C_{l,nc}$ to the *AirStation* lift coefficient defined as

$$C_{l,nc} = \pi b\left(\frac{\dot{\theta}}{U} + \frac{\ddot{h}}{U^2} - \frac{ba\ddot{\theta}}{U^2}\right) \tag{19}$$

where $b$ is the semi-chord, $\theta$ is the geometric angle, $h$ is the vertical plunge distance from the rotor plane, and $a$ is distance from the center of blade rotation to the mid-chord (Hodges and Pierce, 2002). This correction term is added to the lift coefficient from unsteady aerodynamics and is only used if the useUnsteadyAero parameter is true.

## 2.2. RotorAeroLib_Globals

The *RotorAeroLib_Globals* model uses a Modelica *outer* declaration to define a globally scoped class containing definitions of whole-rotor design parameters, variables, and the aerodynamic correction term usage flags. One *RotorAeroLib_Globals* model is intended to be used with each rotor, so in multi-rotor configurations, the user must take care to properly encapsulate the model within an instance of a rotor to ensure a one-to-one correspondence between a rotor's *RotorAeroLib_Globals* model and its set of *AirStation* models.

Within the *RotorAeroLib_Globals* model are parameters for the rotor blade radius, root cutout radius, chord length, solidity, and tip loss factor. Also included are the air density, speed of sound, and free-stream velocity. To calculate certain aerodynamic values, it is necessary to know the nominal blade tip speed, which cannot be calculated at any given *AirStation* due to its purely local definition. Therefore, the *RotorAeroLib_Globals* model contains an input port for the rotor shaft speed from which the nominal blade tip speed is calculated. This value is then used in an *AirStation* model which contains an *inner RotorAeroLib_Globals* model.

Also defined within the *RotorAeroLib_Globals* model are the parameter definitions for the various *AirStation* correction terms. Each of these is turned on and off via a Boolean parameter setting, which then gets referenced within the *AirStation* model.

### 2.3. RotorBlade

A rotor blade typically has properties that vary along its length, including cross-sectional and material properties which are usually defined in terms of EI, GJ constants. Typically, there is also a cutout region near the blade root where there is no airfoil, beyond which the blade twists along its axis to account for variation in aerodynamics due to changes in transverse speed as a function of radius. The *RotorBlade* model provides a template with which to include these effects as parameters.

The *RotorBlade* model consists of a series of flexible beam members from the *DeployStructLib* library attached end to end with a discretization level set by a parameter. Between each beam segment is a *FixedRotation* model that progressively implements the twist of the beam along its length. The twist rate is set by a parameter. *AirStation* models are attached to a frame located at the quarter chord of each beam, rotating with the beam as it moves and imparting aerodynamic forces onto the beam. The *AirStation* is also connected to a reference coordinate system, which is always oriented with the blade nominal position within the rotor plane to provide a reference for the computation of the aerodynamic equations.

### 2.4. RotorBladeAssembly

A rotor can have any number of blades, so the *RotorBladeAssembly* model is parameterized to create multiple instances of a *RotorBlade* model and automatically set up the appropriate relative and reference orientations for the blades and their inputs. Most parameters are passed through this model and applied to the underlying *RotorBlade* models.

The *RotorBladeAssembly* model provided in this library uses a configuration where the blades are fixed together in the same plane, which would be representative of fixed rotor or gimballed rotor topologies. Other rotor topologies can be created by modifying this model and incorporating the appropriate rotor mechanisms (e.g., pitch links, pitch horns, and joints). An example of a gimballed rotor topology is provided in *RotorAeroLib.Examples.GimballedRotor*.

### 2.5. Rotor

The *Rotor* model is a template for properly combining a *BladeAssembly* and a *RotorAeroLib_Globals* model to ensure that the reference coordinate system is correctly set up for the *RotorBlade* models and is isolated from any motion due to the rotor topology mechanisms. This is important because the aerodynamic equations require a reference coordinate system fixed in the nominal blade orientation. With the exception of a fixed rotor, the flapping and lead-lag motion of the blades are perturbations upon this nominal blade orientation, so it is important that the reference coordinate system is properly isolated at a point that is outside of such motion.

The *Rotor* model also contains an implementation of the rotor motor or more simply the shaft speed control, which notionally could be of any type and complexity. In some cases, it is sufficient to set the rotor speed to a set value without concern about coupled motor dynamics, as is the case with the *Rotor* model. With the advent of electric aircraft propulsion and multi-rotor hobby or maintenance copters, electric motors are of increasing interest. An example of an electric motor coupled into the rotor drive train is provided in *RotorAeroLib.Examples.ElectricRotor*.

## 3. Examples

We now present three examples of rotor analysis using the *RotorAeroLib* to perform different types of rotorcraft analyses. The first example validates the basic BEMT implementation of the aerodynamic equations. Presented in the second example is an evaluation of the rotor whirl-flutter stability boundary and comparison to theory and test data. The final example demonstrates the altitude control of a quadcopter model, including the modeling of electric motors.

### 3.1. Aerodynamics Validation

The Modelica implementation of the aerodynamic equations in *RotorAeroLib* was exercised to both verify and validate the strip theory aerodynamics by replicating the Knight and Hefner five-bladed 1937 rotor experiment (Knight and Hefner, 1937). The results were also compared to an analytic model of the BEMT equations (Reveles et al., 2019). The model was discretized with eleven airstations on each rigid blade and assumed to have no tip loss. Unsteady aerodynamic terms were also turned off. The analytic model utilized a tip loss model with 100 airstations per blade. Figure 3 shows that the Modelica model has excellent agreement with both the analytic method and the test data for thrust, while Figure 4 displays the same trends for rotor power. The use of a tip loss model on the analytic curve is observed to slightly reduce thrust and nominally increase the predicted power required, both of which are in the direction of the experimental measurements.

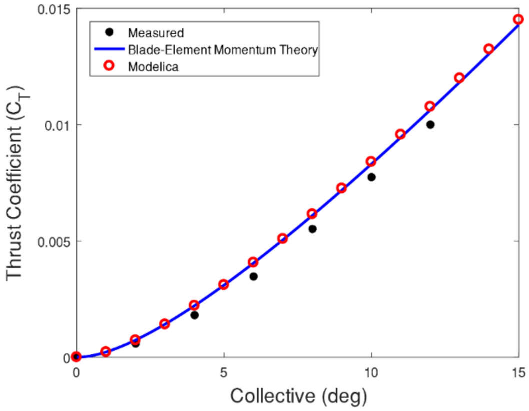


**Figure 3.** Thrust coefficient as a function of collective for five-bladed Knight and Hefner experiment.

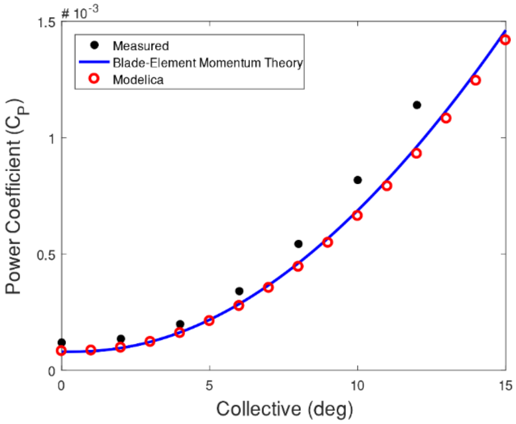


**Figure 4.** Power coefficient as a function of collective for five-bladed Knight and Hefner experiment.

## 3.2. Whirl Flutter

To test the ability of using the *RotorAeroLib* to accurately model the physics of whirl flutter, a model of the Bland and Bennett whirl-flutter experiment (Bland and Bennett, 1963) was developed. The model, shown in Figure 5, consists of a fixed rotor connected to a pylon that has pitch and yaw about its base with characteristic stiffness and damping, which represents the rotor's connection to an aircraft wing. In this model, the rotor is free to rotate and is driven only by the aerodynamics induce by the freestream air velocity and not by a motor nor by any other power source. The rotor was created with six airstations per blade, which was observed to have little difference (outside of computational speed) from a twenty-airstations-per-blade model for this problem. Built-in twist was linearized and a sectional lift coefficient of 5.7/rad was utilized without any in-plane drag coefficients. A visualization of this configuration is shown in Figure 6. The propeller was trimmed to a windmilling state (zero torque) and results compared with analytic theory. Figure 7 shows a comparison of advance ratio results, revealing good correlation between the model, analytic theory, and measurements taken from reference (Bland and Bennett, 1963).

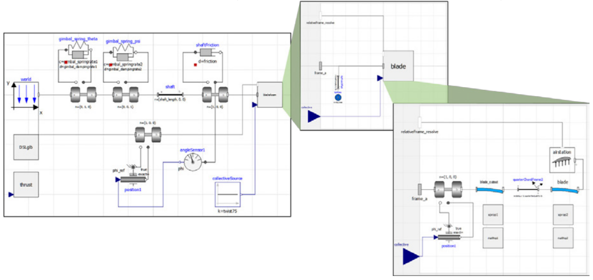


**Figure 5.** Block diagram for the Modelica-based whirl flutter model.

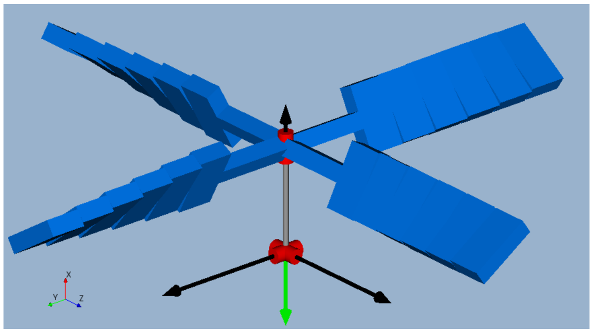


**Figure 6.** Modelica model of the Bland and Bennett propeller.

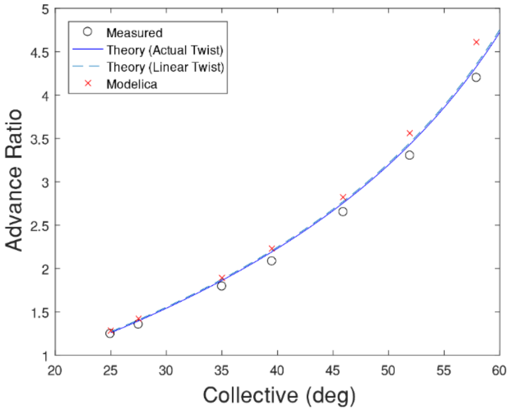


**Figure 7.** Advance ratio for the propeller windmilling state as a function of collective.

To solve the whirl-flutter stability problem, the Modelica model is initialized into a windmilling state (steady-state and zero torque) at which point a linearization is performed to obtain a state-space representation of the system. From this, the eigenvalues of the matrix $\boldsymbol{A}$ relating system states and their derivatives are retrieved, yielding the aerodynamic damping and frequency as the real and imaginary parts of the eigenvalues, respectively. A negative real part corresponds to a stable system, whereas a positive real

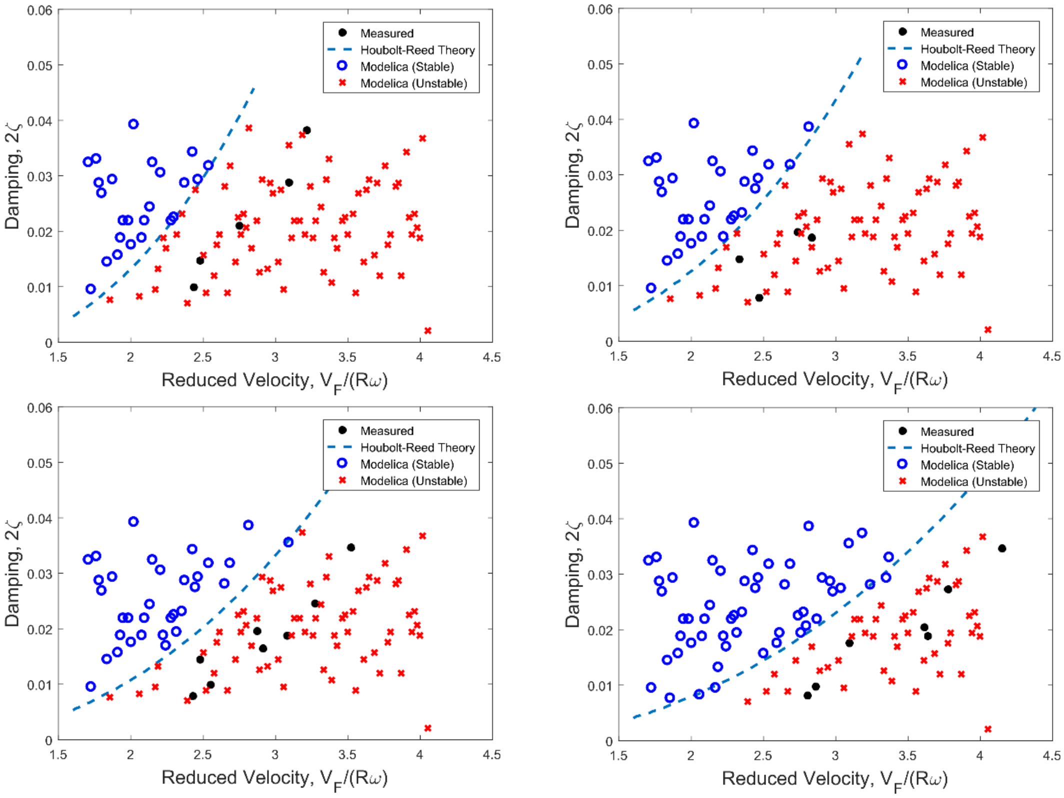


**Figure 8.** Comparison of whirl-flutter boundary between Modelica (blue circles are stable, red X's are unstable) and the present Houbolt-Reed analysis with only quasi-steady aerodynamic terms. Blade angles are 25° (top left), 35° (top right), 46° (bottom left), and 58° (bottom right).

part corresponds to an unstable system. By sampling the parameter space of gimbal damping rates and freestream velocity, the whirl-flutter stability boundary can be characterized by the sign of the eigenvalue real part.

The Modelica model was used to predict the whirl-flutter stability boundaries using quasi-steady aerodynamics for four different collective angles: 25°, 35°, 46°, and 58°. Figure 8 plots the stable and unstable test points from a parametric sweep of the operating envelope along with the Houbolt-Reed analytical model (Reveles et al., 2019) and experimental data from reference (Bland and Bennett, 1963). There exists excellent agreement between the two analysis methods, where all Modelica results align with the Houbolt-Reed stability boundaries for 25° and 35° collectives, and only subtle deviances are observed at the higher collective angles of 46° and 58°. The small deviations at higher collectives are attributed to weak nonlinearities that are automatically resolved in the Modelica model. It is observed that the Modelica model results at higher collectives subtly shift the results closer to the observed measurements as compared to the Houbolt-Reed results, although without the use of unsteady aerodynamics, an accurate reconstruction of the stability boundary is not expected. It is also evident that the analytical methods show better correlation at higher collectives than at lower collectives. This may be due to the high inflow velocities encountered in the windmilling state that induce large local angles of attack, beyond which small angle assumptions may not necessarily apply. Additional deficiencies in thin airfoil theory are likely exacerbated by the relatively lower Reynolds numbers encountered at model scale that tend to induce earlier separation than those encountered at full scale.

### 3.3. Quadcopter Flight Control

The final example demonstrates the use of the *RotorAeroLib* on the flight control of a quadcopter model. Each rotor of the aircraft consists of a fixed rotor rigidly attached to a spar connected to the quadcopter body, which is represented by a lumped mass. For this simplified example, all four rotors rotate in the same plane and operate in a quiescent environment with zero freestream velocity. As is typical of these vehicles, two rotors rotate clockwise, while the other two rotate counterclockwise to provide a means to maintain rotational inertia balance. A control system simultaneously controls the input voltage to four rotor electric motors that drive the rotors; hence, the rotor speed is the same for all four rotors. The control system is used to set the altitude of the quadcopter and consists of a PID controller with maximum and minimum motor voltage limits. Figure 9 shows a visual representation of the quadcopter.

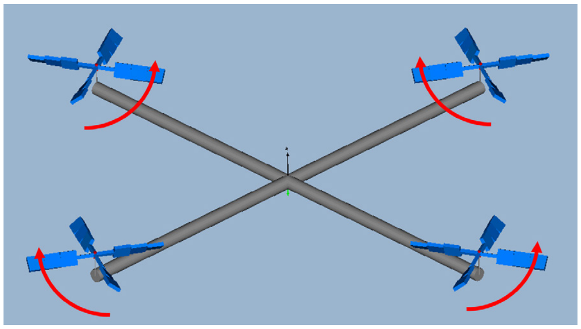


**Figure 9.** Visualization of the quadcopter model with rotor rotation directions identified.

A demonstration of the quadcopter model is performed by changing the altitude set point of the controller to 5 m after two seconds of hover at 0 m, which is also its initialization state. Figure 10 plots the set point and quadcopter altitude, showing a small overshoot in altitude and then settling to the set point. Figure 11 shows the motor voltage input and rotor speed as the maneuver takes place, from which the coupled electro-aero-mechanical dynamics and controller voltage limits are apparent. Figure 12 plots the motor current, which further shows the electro-mechanical coupling of the system. The sizing of the rotors, motors, and other aspects of this demonstration model are somewhat arbitrary, which is clear in the very high motor current peaks, although the steady-state current is approximately 1.1 Amp.

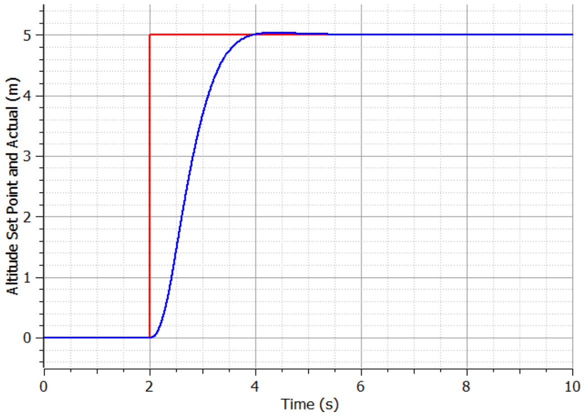


**Figure 10.** Quadcopter altitude set point (red) and actual (blue).

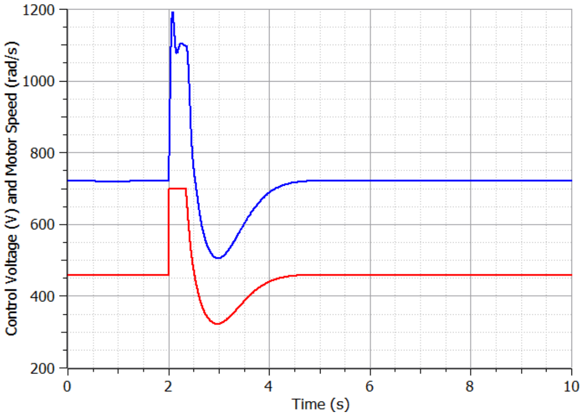


**Figure 11.** Quadcopter motor voltage (red) and rotor speed (blue).

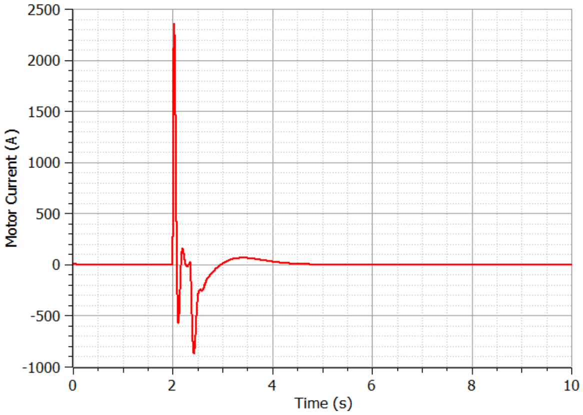


**Figure 12.** Quadcopter motor current.

## 4. Conclusions

This paper presents a Modelica library, *RotorAeroLib*, for the analysis of rotorcraft aerodynamics, whirl flutter, and rotorcraft control. It provides models useful for simulating the coupled motion of a rotor, the rotor blades, and the aerodynamic forces imparted upon the blades as they move through a fluid. The library may also be used for performing whirl-flutter analysis of rotors connected to flexible structures where dynamic instabilities may manifest.

The Rotorcraft Aerodynamics Library is open source and available via the GitHub page https://github.com/ata-engineering/RotorAeroLib which will likely be linked to via the Modelica Association website at https://modelica.org/libraries. While the library should work for any Modelica implementation per the Modelica standard, it was developed using OpenModelica and has not been tested using other software. External contributions and bug fixes or reports are encouraged.

## References

I.H. Abbott and A.E. von Doenhoff. *Theory of Wing Sections*. Dover Publications, New York, 1959.

J. Batteh, J. Gohl, M. Sielemann, P. Sundstrom, I. Torstensson, N. MacRae, and P. Zdunich. Development and Implementation of a Flexible Model Architecture for Hybrid-Electric Aircraft. In *Proceedings of the 1st American Modelica Conference*. Cambridge, MA, USA, October 2018. doi: 37 10.3384/ecp1815437.

O.A. Bauchau. Dymore User's Manual [Computer software] [Online]. University of Maryland, n.d. http://dymoresolutions.com.

S.R. Bland and R. M. Bennett. Wind-Tunnel measurement of Propeller Whirl-Flutter Speeds and Static-Stability Derivatives and Comparison with Theory. NASA TN D-1807, 1963.

F.D. Harris. *Introduction to Autogyros, Helicopters, and Other V/STOL Aircraft. Volume II: Helicopters.* Ames Research Center, Moffett Field, CA, 2012.

D.H. Hodges and G.A. Pierce. *Introduction to Structural Dynamics and Aeroelasticity*. Cambridge University Press, New York, NY, 2002.

J.C. Houbolt and W.H. Reed. Propeller-Nacelle Whirl Flutter. *Journal of the Aerospace Sciences*, Vol. 29, No. 3, March, 1962.

M. Knight and R.A. Hefner. Static Thrust Analysis of the Lifting Airscrew. NACA TN No. 626, December, 1937.

J.G. Leishman. *Principles of Helicopter Aerodynamics, 2nd-edition*. Cambridge University Press, New York, NY, 2006.

J.G. Leishman and K.Q. Nguyen. State-Space Representation of Unsteady Airfoil Behavior. *AIAA Journal*, Vol. 28, No. 5, May 1990, pp. 836-844.

W.H. Reed. Review of Propeller-Rotor Whirl Flutter. NASA TR R-264, 1967.

W.H. Reed and S. R. Bland. An Analytic Treatment of Aircraft Propeller Precession Instability. NASA TN D-659, January 1961.

N. Reveles, J. Schoneman, C. Rupp, and E. Blades. Utilizing Analytic and Direct Methods for the Verification and Validation of Whirl Flutter Analyses. In *Proceedings of the AIAA SciTech Forum*. San Diego, CA, USA, January 2019. doi: 10.2514/6.2019-1865.

C. Rupp and L. Schweizer. The Deployable Structures Library. In *Proceedings of the 1st American Modelica Conference*. Cambridge, MA, USA, October 2018. doi: 10.3384/ecp18154187.

W.Z. Stepniewski and C.N. Keys. *Rotary-Wing Aerodynamics*. Dover Publications, New York, 1984.

# Hierarchical Multi-Level Electric Power System Simulation with Smart Photovoltaic Systems using the Functional Mock-up Interface on the Lawrencium Computing Cluster

Christoph Gehbauer[1] Joscha Müller[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA, {cgehbauer,joschamueller}@lbl.gov

## Abstract

The adoption of distributed photovoltaics (PV) with smart inverters was one of the first large-scale deployment of grid-interactive, customer owned assets. This paper introduces a co-simulation platform for future scenarios of Distributed Energy Resources (DER), in the context of large-scale deployment, to assess the local and global impact on the electric power grid. The co-simulation platform utilizes the Functional Mock-up Interface (FMI) industry standard to couple 80,851 individual simulators. For this purpose a Modelica package named SCooDER was developed, which includes models for various DERs. The simulation was conducted at the Lawrencium high performance computing cluster. It included a hierarchical structure of multi-level electricity grids (i.e., transmission, medium-voltage distribution, and low-voltage distribution), and PV with smart inverters and time-varying load profiles at 80,000 load buses, in representation U.S. state sized electric power grid. With the flexibility of the simulation framework and the agreed-on industry standard for simulation model exchange, future applications can be very broad by coupling multi-domain simulators.

## 1 Introduction

Power grids world wide are undergoing big changes due to the increasing amount of distributed energy resources (DERs) such as photovoltaics (PV), behind-the-meter battery storage (BTM-BS), and electric vehicles (EVs) (Bayod-Rújula, 2009). Deploying large amounts of DER can result in unintended and potentially critical conditions on the grid. Simulations offer one means of conducting a contingency analysis to identify critical scenarios and to support the planning and installation of DERs. However, large-scale, detailed simulations are complicated to setup and solve. As a result, detailed simulations are often only conducted for a specific application due to the cost and labor intensive setup. They are also often only conducted for specific events in subsections of the grid like in (Leou et al., 2013), (Godfrey et al., 2010), or (Stetz et al., 2012). Simulations of larger grids often do not include detailed models of single DERs; instead they use pre-computed power flows (i.e., positive and negative active and reactive loads) for discrete time steps.

Another difficulty for simulations is the large variety of devices that are connected to the power grid and the resulting use cases of the simulation. In addition to the variety of DER listed above, applications typically also include buildings and commercial or industrial facilities. The challenge, therefore, is that different devices are modeled with different, domain-specific tools. These tools are often proprietary, which makes it difficult to extend or interconnect them. One solution to couple a wide variety of different domain-specific models is the Functional Mock-up Interface (FMI). Initiated by the European automotive industry, it was developed to standardize the exchange and co-simulation capabilities of models from different vendors. This standard can be used to export simulation models as a Functional Mock-up Unit (FMU), which in turn can be used in other software tools which support the FMI standard (Nouidui et al., 2019; Blochwitz et al., 2012). The source code of FMUs can be hidden, so proprietary models can be shared without revealing sensitive information.

The FMI standard is currently supported to different extents by 134 software tools (Modelica Association, s.a.). Relevant to the electric power system are the tools EMTP-RV, EcosimPro, Matlab/Simulink, and ESI SimulationX which directly support the FMI standard. Other tools such as Cymdist, PowerFactory, PSCAD, DSATools, PSS/E, Pandapower, GridDyn, MATPOWER, and OpenDSS can be supported indirectly by wrapping their Application Programming Interfaces (APIs) into FMUs using a tool called SimulatorToFMU. (Nouidui and Wetter, 2017)

In this paper we apply the FMI standard to power systems by conducting a large-scale grid simulation, which involves 80,851 FMUs in total, representing 80,000 individual customers with co-located PV systems and smart inverters connected to the grid. Smart inverters are designed to regulate the reactive power output in respect to locally observed grid voltages, to mitigate the impact of distributed PV generation on the grid. The simulation is partitioned in multiple subsections of the grid which are representative of the different voltage levels. The subsections are again encapsulated in FMUs and connected with other FMUs to form a large grid simulation. To decrease simulation time, the parallelism of subsections was applied and scaled across different compute nodes

10.3384/ECP20169120 DOI

at the Lawrencium High Performance Computing Cluster (HPCC) at Lawrence Berkeley National Laboratory (LBNL). This paper describes the developed framework and discusses its scalability and flexibility for large-scale simulations.

## 2 Overview

The central part of this paper is the leverage of the FMI standard, which is used to create simulation models for a large-scale simulation of an electricity grid. The scenario includes 100 percent PV penetration (i.e., peak PV production is equal to peak load demand) and voltage-dependent smart inverter controls at each of the 80,000 individual customers. The transmission grid model is representative of a large power grid, the nominal size of a state in U.S. (i.e., Illinois).

### 2.1 Functional Mock-up Interface

The FMI standard describes a set of standardized functions to export and link simulation models for the use in co-simulations. A model which is exported in compliance with FMI is called an FMU. It consists of the files below which are zipped as a zip file with the ending $.fmu$:

- An Extensible Markup Language (XML) file describing parameters, inputs, outputs, and dependencies of the model.
- Compiled C-code with standardized FMI functions to evaluate the model.
- Resource data which can contain additional information such as documentation, dependency files that are required by the simulation, or graphical illustrations.

Using the FMI standard as the back-end of the simulation offers some benefits, which include (a) leveraging an agreed-upon industry standard which is well maintained, updated, and improved by industry, (b) the ability to utilize a large variety of model libraries for different domains that are built and maintained by large industry and research institutions, (c) the flexibility to couple models of different domains (e.g., battery chemistry model with EV drive-train model coupled with the electric power grid) to form a single multi-domain co-simulation, (d) industry developed wrappers for the FMI standard in different programming languages (e.g., Python, MATLAB, Modelica, Java) that are well maintained and documented, and (e) a large community working on the FMI and related standards.

The FMI standard supports two types of simulation model export. The first mode is the co-simulation (CS) mode. In *CS* mode, every model contains its own numerical solver and provides the result for the next timestep, i.e., start time of the model plus step size, as an output. When a simulation is run in *CS* mode, the individual models are simulated with a defined timestep, and forced to advance time for the step defined. The execution time and order is managed by an orchestrator, where one iteration completes when all FMUs are evaluated once. The other mode is the model exchange (ME) mode. *ME* models do not contain their own solver and instead provide the evaluated system of equations as output. In case of a first-order differential equation the output would be the state derivative, while *CS* would output the integrated state. An orchestrator coordinates the evaluation of models and a global solver solves the coupled models as a network. This way, the models are run repeatedly, and with a variable timestep, until inputs and outputs converge to an equilibrium (Blochwitz et al., 2012). Since *CS* can only advance in time, it is often desirable to utilize *ME* FMUs for simulation, especially when algebraic loops are present. Using *ME* FMUs, the solver can iterate back and forth in time until it finds a solution, while the *CS* FMU has to advance time. In case of a step function or other rapid change in output, the *CS* would miss the event, while *ME* allows to go back in time to find the exact event. A common workaround with *CS* FMUs are sufficiently small timesteps which, on the other hand, increase solving time. All the models in this paper are exported as FMUs and simulated by the FMI standard. The FMUs contain both types of export, depending on the hierarchy level.

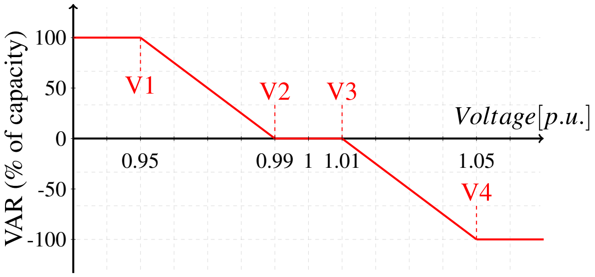


**Figure 1.** Volt/Var control curve for smart inverters.

### 2.2 Smart Inverter

To control and guide the implementation of DERs, the IEEE 1547 international interconnection standard (Basso et al., 2015) was established. In addition, some U.S. states implemented additional rules for DERs to comply with. For example, the California Public Utilities Commission (CPUC) established Rule 21 (Commission et al., 2014) for California. With respect to IEEE 1547 and Rule 21, PV inverters connected to the power grid have to provide advanced inverter functionalities. These include the capability of advanced control features like the reactive power droop control, commonly referred to as Volt/Var control. The Volt/Var control is illustrated in Figure 1.

With an active Volt/Var control, the inverter controls its reactive power generation or consumption (on the y-axis) depending on the local voltage (on the x-axis). If the local voltage exceeds a threshold, V2 or V3, the inverter starts to consume or generate reactive power in a linear response, depending on whether voltages are too low or too high. It saturates (i.e., reaches the maximal reactive power absorption or generation) at V1 or V4 accordingly. In electrical power systems the generation of reactive power results in an increase in system voltage, and the absorption of reactive power results in a decrease in system voltage.

A Modelica package called Smart Control of DER (SCooDER) was developed by LBNL to facilitate the testing and simulation of such devices (Gehbauer et al., 2019). It contains models of PV generators, inverters, batteries, sensors, controllers, and other models relating to DER and power system simulations. The models are intended for detailed DER simulations and can all be exported as FMUs. This study used the PV generator and smart inverter control models, which are exported as FMUs using JModelica.

## 2.3 JModelica

JModelica is an open-source platform supporting simulation, optimization, and analysis of complex dynamic systems in the Modelica language. It provides tools for exporting and simulating FMUs (JModelica.org, s.a.). All the simulations in this paper were conducted with components of the JModelica package. PyFMI is a Python based tool for loading and executing FMUs and is part of this package. It supports simulations in either *CS* or *ME* mode by providing a simple interface to run simulations with the FMI standard in an open source environment (Andersson et al., 2016). PyFMI relies on third-part numerical solvers (i.e., CVODE as default) to solve the system of equations. CVODE is a C-based solver that can be used to solve systems of stiff and non-stiff ordinary differential equations (Cohen et al., 1996).

## 2.4 Pandapower

For the power grid part of the simulations, Pandapower, an open-source Python-based tool for simulating and analyzing power systems, was used. Pandapower provides power flow and optimal power flow (OPF) capabilities, which are based on PYPOWER, which in turn is based on the MATPOWER tool. Pandapower ships with a large library of pre-configured electric grid models that can be easily loaded and simulated (Thurner et al., 2018). The next section describes a tool for automated export of Pandapower as an FMU, allowing users to leverage Pandapower's power systems modeling capabilities.

## 2.5 SimulatorToFMU

SimulatorToFMU is a Python package developed by LBNL to wrap the high-level Python API of a third-party simulator in a Python function, which can then be exported as an FMU (Nouidui and Wetter, 2017). The utility auto-generates Modelica code that contains a model to communicate with the simulation tool through its Python API. It then invokes a Modelica translator to compile the model and export it as an FMU. It is built to leverage third-party Modelica compilers (i.e., JModelica, Dymola, or OpenModelica) to export the model. This helps to ensure the forwards and backwards compatibility with new versions of FMI. SimulatorToFMU requires an XML file, which specifies the input and the output of the simulator, as well as the Python function which interacts with the simulator. The export is implemented as a Python function call.

## 2.6 Lawrencium HPC Cluster

For the large-scale simulation of many FMUs, the HPCC at LBNL was used. It currently includes four clusters with a total of 924 compute nodes, ranging from 16 to 32 central processing units (CPUs) and between 64 to 128 gigabytes (GB) of random access memory (RAM) per compute node (LBNL, s.a.). It utilizes large parallel network file storage to enable fast data transfers between compute nodes. An overview of the four active stages of Lawrencium is given in Table 1.

| Name | Nodes<br>Total | Cores<br>per Node | RAM [GB]<br>per Node |
|---|---|---|---|
| L6 | 215 | 32 | 96 |
| L5 | 187 | 20 or 28 | 64 or 128 |
| L4 | 142 | 24 | 64 |
| L3 | 380 | 16 or 20 | 64 |

**Table 1.** Active Lawrencium HPCC stages.

The simulations conducted in this study require a high number of CPU cores for efficient scale-up, and a moderate number of compute nodes. LR6 was chosen as the target cluster for this application. Note that all clusters are connected to a common private network which is utilized in this work to distribute work across multiple nodes and clusters. The framework for this work-load distribution was written in Python and exported as FMU. This FMU augments the HPCC as a single model to be used in the co-simulation.

# 3 Setup

This study's objective was to demonstrate the scalability and versatility of the FMI standard for power system applications. The setup was therefore focused on power grids and the impact of high penetration of PV equipped with smart inverters. What made this setup challenging was the large amount of controllable PV whose active and reactive power output impacts one another through the coupling of the electric power grid. The local grid voltage as input to the smart inverter and the reactive power as output forms a feedback loop, which is challenging to solve at large scale.

To simulate a realistic power grid, prototypical example network models of different voltage levels (i.e., 115 kilovolt (kV) transmission, 20 kV medium-voltage distribution, and 0.4 kV low-voltage distribution) were utilized to form a whole power grid. The three network models are shown in Fig. 2, Fig. 3, and Fig. 4. Fig. 2 shows the 42 bus transmission network which is based on IEEE Illinois Case 57 (Christie and Dabbagchi, 1993), Fig. 3 shows

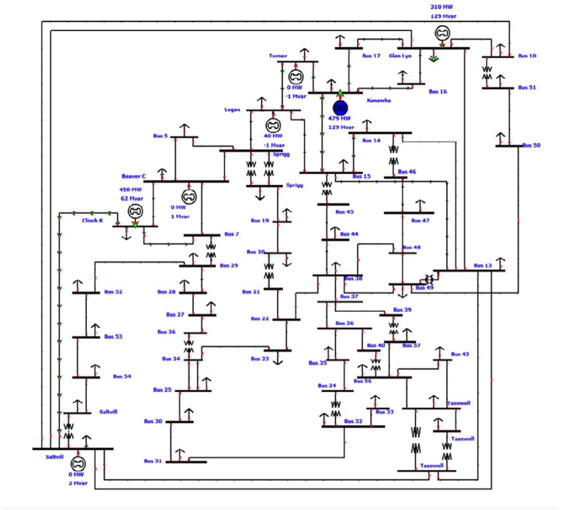


**Figure 2.** IEEE 57 Transmission network.

the medium-voltage distribution network which is based on the 13 bus CIGRE MV network (Rudion et al., 2006), and Fig. 4 shows a 146 bus low-voltage distribution network based on the Kerber LV network (Kerber, 2011). All networks are taken from the Pandapower example models. Each of the 80,000 load buses in the low-voltage distribution networks is representative of a single customer with a PV system, smart inverter, and time-varying base load. The base loads are taken from the U.S. Department of Energy (DOE) statistical reference of building types. It defines 17 load profiles for the U.S. (Department of Energy, s.a.). For this study pre-computed files for the historic weather data of San Francisco, California were used.

## 3.1 Time-Varying Load Profiles

The 17 DOE load profiles were clustered using the kMeans algorithm (Krishna and Murty, 1999) with the objective to establish grouping of data samples by minimizing the centroid distances. The optimal number of clusters, defined as where the loss of the algorithm is less than 20 percent, was found to be 5 clusters. In a next step one representative load profile of each cluster was manually chosen, which reduced the total number of individual load profiles to 5. This helped to reduce the overhead of assigning load profiles of similar type (e.g., *quickservicerestaurant* and *fullservicerestaurant*). The individual profiles are shown in Fig. 5 where all load profiles are scaled by its peak power demand of the selected day, which is June 1st for this study. The manually selected profiles are highlighted as a solid line, while the other profiles associated with the cluster are shown as dotted line in the same color. It can be seen that some profiles, such as the orange one which centers around the *fullservicerestaurant* profile consists of six DOE load profiles (i.e., *outpatient*, *smallhotel*, *largeoffice*, *quickservicerestaurant*, and *supermarket*) while others, such as the violet one only consist of one DOE load profile (i.e., *residential*). In either case, the grouping of DOE profiles was determined by the kMeans clustering. In order to realistically assing the profiles to load buses on the feeder model, a statistical distribution was defined, based on data of 50 individual feeder models from a California utility company.

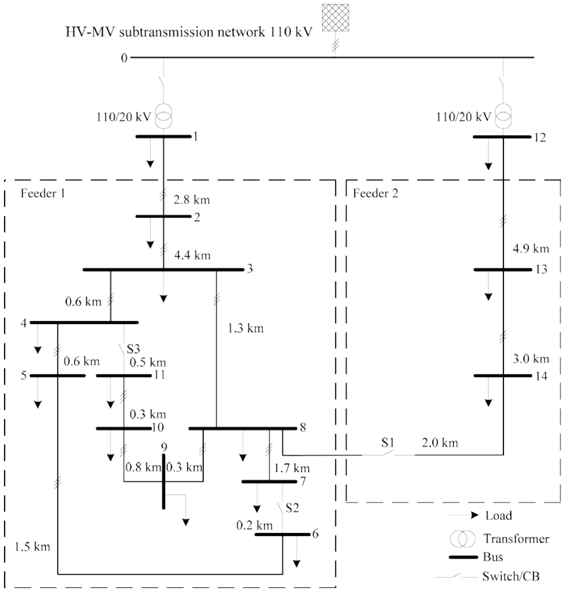


**Figure 3.** CIGRE Medium-Voltage network.

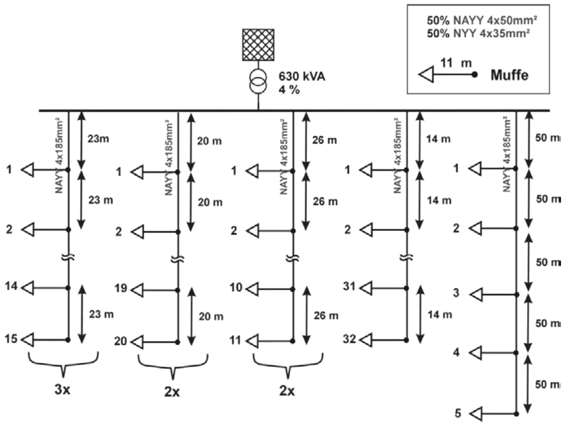


**Figure 4.** Kerber Low-Voltage network.

The distribution of load types is shown in Fig. 6 where the data for 50 individual feeder models is plotted as blue, orange, and green crosses for residential, industrial and commercial customers accordingly. The data is ranked by the share of residential customers. Two linear fit functions were established with the residential share as independent variable, and industrial and commercial share as dependent variables. The evaluated fit is shown as the solid line in the same color as the base data. For this study, a distribution based on the mean residential share, which is 46 percent, was chosen. The five load profiles were manually assigned to one of the three categories, and the total number of each load profile occurrence was computed. The

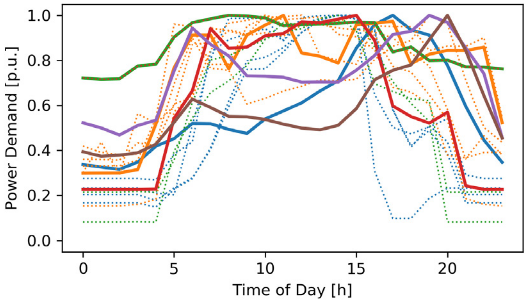


**Figure 5.** Overview of clustered DOE load profiles for June 1st.

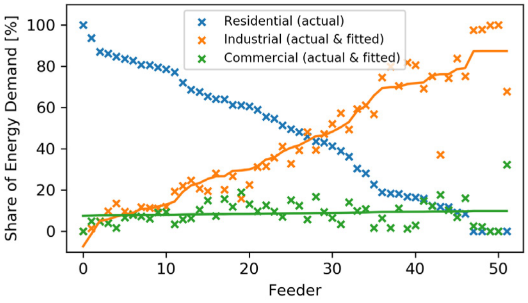


**Figure 6.** Load type distribution on 50 feeders of a Californian utility company.

computed mix of load profiles was randomly assigned to feeder load buses.

Since the networks are only coupled by the voltage at the interconnection point as input and the resulting active and reactive power as output, it was possible to implement scaling factors to scale single customer load demand and PV generation as aggregated power flow to the nominal of the example network model. This step avoids the over- or undersizing of customers, and results in a unique scaling factor for every load bus and feeder. This allows for a wide variety of control actuation by the smart inverter across the simulated power grid.

## 3.2 Functional Mock-up Units

The architecture to conduct simulations at the Lawrencium HPCC involves seven distinct FMUs which are loaded multiple times and parameterized differently, to result in a coupled system of 80,851 FMUs in total. The FMUs use both the *ME* or *CS* mode, based on the hierarchy level. The architecture is described in Section 3.3 and Fig. 8. The FMUs are described here:

- PV FMU: This FMU computes the PV generation based on historic weather data as input. It is a derivative of the PV model introduced by the Modelica Buildings Library (MBL) (Wetter et al., 2014) and modified by the SCooDER package. It uses a detailed sky model to compute incident solar irradiation on a tilted surface. The PV generation was calculated with the total irradiance with a 10 degree tilt towards south. It was exported as an *ME* FMU using JModelica. The path to the Modelica model is *SCooDER.Components.Photovoltaics.Model.PV andWeather_simple* or *Buildings.Electrical.AC. OnePhase.Sources.PVSimpleOriented* for the base version.

- Smart Inverter FMU: The function of the smart inverter FMU is to regulate the reactive power output based on locally observed grid voltages, introduced as Volt/Var control in Section 2.2. It is a linear response feedback control, based on the voltage at the interconnection point, with a deadband (±0.01 p.u.) around the control setpoint (1.0 p.u.), with saturation (0.95 and 1.05 p.u.). The maximal reactive output was set to 30 percent of the inverter size. It was implemented in the SCooDER package and exported using the *ME* API with JModelica. The path to the model is *SCooDER.Components.Controller.Model.voltVar_ param_simple_firstorder*. It also includes a first-order response to reflect the internal control delays of the smart inverter. The response time was validated with measurements taken at LBNLâĂŹs FLEXGRID facility. A positive side-effect of the control delay is the separation of algebraic loops in the co-simulation.

- Feeder Model FMU: The electrical feeder network relies on a custom Python function which calls Pandapower (a) at instantiation to create the electrical model from the example networks, and (b) on runtime to execute a power flow analysis which computes the nodal voltages based on active and reactive power flow at the load buses. The base load in the example network was scaled based on a time-varying demand profile, derived from the DOE reference buildings, as described earlier. To export the function with the SimulatorToFMU tool, another Python wrapper function was developed. To make it possible to scale-up simulations across many compute nodes at Lawrencium HPCC, the two Python functions are coupled through a socket communication, illustrated in Fig. 7. Hereby the Pandapower wrapper was encapsulated in a simple web-server and called by the SimulatorToFMU wrapper using Hypertext Transfer Protocol (HTTP) requests. Information is exchanged through JavaScript Object Notation (JSON) sanitized objects. The advantage of this implementation is the potential for large scale-up by parallelization across many compute nodes. Both Python functions were optimized for fast computation. This results in large reductions of computation time, because these functions are called at every iteration step of the feeder model. While a first implementation took up to 15 seconds to complete a single timestep, the finally implemented version completed within an average of

200 milliseconds. Further results regarding the scalability are shown in the Results section. The FMU was exported with SimulatorToFMU using the *ME* mode.

- Distribution Model FMU: The electrical distribution network also uses Pandapower to provide and solve the electric network model, and it uses the same architecture as the Feeder Model FMU. It was exported with SimulatorToFMU using the *CS* mode.

- Coupled Feeder FMU: The partitioning at Lawrencium HPCC using multiple compute nodes requires that the FMUs be executed in parallel. This is implemented in the Coupled Feeder FMU, which is a time-discrete wrapper to a coupled system of one Feeder Model FMU and PV FMU, and one Smart Inverter FMU at each of the 146 load buses. While PV generation is also present at each load bus, only one PV FMU was loaded, and the generation was scaled to 1 per unit (p.u.), which resulted in a univariate PV profile for all buses. This is a simplification made to keep the number of FMUs and connections within the coupled system low, and to allow for improved solving times. Each load bus also included the time-varying demand profile which is embedded in the Feeder Model FMU. The system of FMUs was coupled using the $CoupledFMUModelME2$ function provided by PyFMI. The Python script was wrapped using the described web-server approach and once-again exported using the *CS* API with SimulatorToFMU. While the exported Coupled Feeder FMU appears to raise state events, the actual implementation of the underlying C function does not support this functionality. However, the wrapped coupled feeder system does support the indication and handling of state events (e.g., when control loops saturate). To simplify this assumption, the Coupled Feeder FMU can be seen as discrete, while the embedded coupled feeder system is continuous with state event handling by the CVODE solver of PyFMI. The tolerances for CVODE were modified as 1e-3 for $atol$ to meet the accuracy of Pandapower.

- Transmission Model FMU: This FMU encapsulates the transmission model in Pandapwer. It is similarly structured to the Distribution Model FMU and also exported as FMU using SimulatorToFMU with the *CS* mode.

- Coupled Distribution FMU: The highest level of aggregation is the Coupled Distribution FMU, which encapsulates one Distribution Model FMU, 13 Coupled Feeder FMUs, and one custom orchestrator. It was exported using SimulatorToFMU with the *CS* mode.

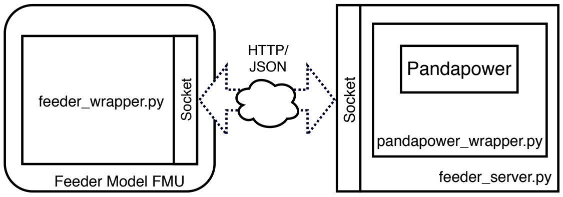


**Figure 7.** Illustration of the socket-based communication between the FMU wrapper and the simulator. The simulator can be hosted locally or remotely across the Lawrencium HPCC.

### 3.3 Architecture

The architecture for computation reflects the electrical network by decoupling the system at the different voltage levels. As described previously, each load bus of the transmission model is connected to one coupled distribution network. Each distribution model is connected to 13 coupled feeder networks, which are then in turn connected to the PV, smart inverter, and time-varying load at each feeder load bus. The architecture is illustrated in Fig. 8.

The architecture combines the Feeder Model FMU, PV FMU, and Smart Inverter FMU into a coupled system of FMUs, all exported using the *ME* mode. This system is solved for a specific time step using the $CoupledFMUModelME2$ function provided by PyFMI, and is again wrapped as an FMU to form the Coupled Feeder FMU. The Coupled Feeder FMU reflects a fully populated feeder where state events of models (e.g., deadband or saturation of the smart inverter control) are considered and solved by PyFMI. The timestep within the Coupled Feeder FMU is variable based on the solver of PyFMI. One level higher on the medium voltage distribution level, the Distribution Model FMU is coupled with the Coupled Feeder FMU by a custom orchestrator which employs the *CS* mode of the FMUs. These FMUs are discrete in time and are invoked based on the high-level time step. This system of FMUs was again wrapped as another FMU, namely Coupled Distribution FMU. It was loaded for each load bus of the Transmission Model FMU and execution was handled by a custom orchestrator that utilizes PyFMI.

The partitioning of the FMUs on the Lawrencium HPCC was determined by a total of 546 Coupled Feeder FMUs, which each were solved in parallel. It was most practical to assign five Coupled Feeder FMUs to one compute node at HPCC. The LR6 Lawrencium HPC cluster, which provides 32 logical CPU cores per compute node, was used for this study. The Transmission Model FMU consisted of 42 load buses, each connected to one Coupled Distribution FMU. This translates to a total of 12 LR6 compute nodes for the simulation. With 215 compute nodes available at LR6, this would scale to a maximum of 1,075 Coupled Distribution FMUs. With 1,898 load buses per Coupled Distribution FMU, the maximal number of individual customers would be about 2 million. However, solving clock-time could be traded against scaleup to fur-

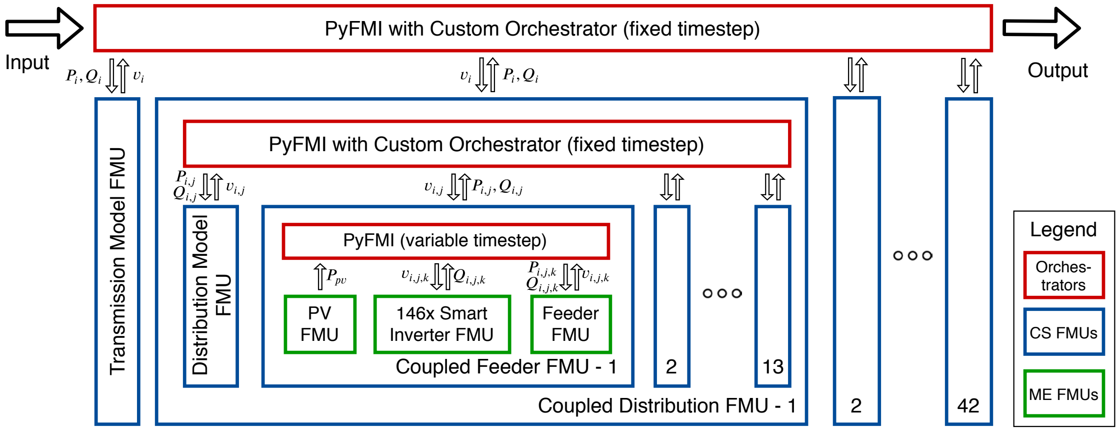


**Figure 8.** Hierarchical multi-level simulation architecture with high-level inputs and outputs for each FMU. The exchange variables are active power, $P$, reactive power, $Q$, and voltage, $v$. The three hierarchical voltage layers are transmission, denoted by $i$ with 42 load buses, medium-voltage distribution, denoted by $j$ with 13 load buses, and low-voltage distribution, denoted by $k$ with 146 load buses.

ther increase the number of individual loads per node.

## 3.4 Orchestrator

The co-simulation of multiple independent models requires the models to be in a standardized format, in this case the exported FMUs, and an orchestrator to coordinate the execution of FMUs and to solve the system of equations. The Coupled Feeder FMU and Coupled Distribution FMU are special cases because they wrap a whole co-simulation to form a new FMU. In case of the Coupled Feeder FMU it already includes an orchestrator which in this case is PyFMI. However the top-level coordination of the Coupled Distribution FMU and transmission level is implemented with a custom algorithm which controls (a) the primary timestep of the model, and (b) the convergence of the model. While the timestep coordination is implemented as a simple $for$ loop, the convergence is more involved by requiring an iteration algorithm and convergence objective and criteria. The implemented algorithm is limited to a maximal number of 10 iterations, with a convergence criteria of voltage derivative less than 5e-3 per unit. The iteration sequence starts with the initialization of the grid model FMU whose computed bus voltages represent the inputs to the coupled FMU. In order to support the solver in its action, the voltage derivatives are dampened by a first-order delay, until an equilibrium is reached. This delay dampens the control action of the decentralized control loops of the underlying Smart Inverter FMUs, to avoid oscillatory behavior. Note that this delay does not affect the physical behavior of the system, and is solely implemented as part of the solving algorithm. The coupled FMU is evaluated in parallel, and the resulting active and reactive power at the feeder heads are fed directly into a last evaluation of the grid model FMU. The bus voltage derivatives are computed, and the system is checked for convergence. If it did not converge and if it also did not reach the maximal number of iterations, another iteration is initiated. Otherwise the iteration is terminated and the result returned.

One drawback with coupling the FMUs exported with the *CS* mode is the timestep control. *CS* FMUs can only advance time which is problematic when seeking the convergence of a system at a defined simulation time. The workaround are sufficiently small timesteps where dynamics within the FMU are close to constant. In this case the internal timestep for an iteration was set to 10 milliseconds.

# 4 Results

The co-simulation was conducted for one sunny day with an hourly timestep on the transmission level. As described earlier, the low-level Coupled Feeder FMU, which encapsulates the PV generation, time-varying base load, and smart inverter, was solved with a variable timestep. The Fig. 9 provides an overview of the smart inverter actuation and solving time for all of the 80,000 individual customers.

The first subplot shows the statistics of the time-varying base load as boxplot normalized to each customers nominal active power demand, for each hour of the simulation. The statistics are based on each of the 80,000 individual customers. The secondary axis of the first subplot, in red, shows the univariate PV profile which is applied to all customers, also normalized to the customer size. The second subplot shows the distribution of smart inverter control actuation as reactive power scaled to the nominal reactive power of each customer. The third subplot shows the distribution of feeder head voltage, as determined by the transmission and medium-voltage distribution network. It is scaled to the nominal feeder voltage. The fourth subplot shows statistics of the solving time, as boxplots on the primary axis, and maximal number of iterations, in red on the secondary axis. Both statistics are based on the 546 Coupled Feeder FMUs. The mean simulation time

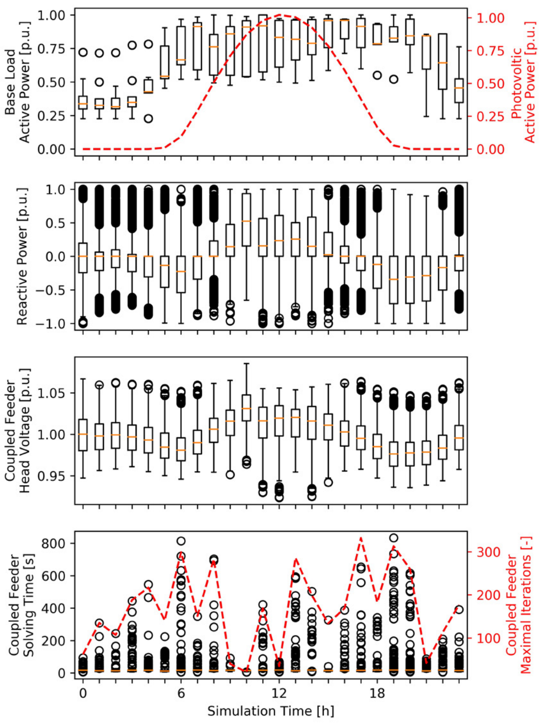


**Figure 9.** Overview of full-scale grid results with 80,000 individual customers.

per iteration is 25 seconds across all timesteps, whereas the standard deviation ranges between 25 to 175 seconds, depending on the timestep.

## 5 Discussion and Future Work

This paper successfully demonstrated the distributed co-simulation of various components of an electric power grid. This includes time-varying load and PV generation, feedback control loop of a smart inverter with Volt/Var control, and electrical coupling through multiple voltage levels.

While the setup simplified the complexity of electric distribution networks, customer load variation, and site-dependent PV generation, it demonstrated the capabilities of the developed SimulatorToFMU tool and the FMI overall to solve complex large-scale cross-platform co-simulation setups. The setup included individually validated models from publicly available Modelica libraries, in case of the Smart Inverter FMU and PV FMU, or Pandapower package, in case of the grid models. The coupled system of FMUs was solved using the PyFMI package which was developed and validated by its developer Modelon. While the whole system results could not compared to other simulators, mainly due to lack of availability to couple those individual models, which was the motivation of this paper, the project team believes that the performed validations are sufficient.

The introduced simulation setup exploited the potential of reduced complexity for high-level aggregated models, while keeping a full detail on the lower-level models. An aggregation of 80,000 individual customers was used to represent the interaction on a U.S. state sized electric power grid. The level of scale, i.e., about 50 times scaleup in this example, was achieved by reducing the number of feeders per substation to one, as well as the number of medium-voltage distribution circuits per transmission node to one. One example where the low-level detail of individual customers is important would an over-voltage event to trip smart inverters which in turn would suddenly reduce the voltage, and could end-up with a cascading effect triggering neighboring substations or feeders. In particular the setup of a coupled co-simulation system being wrapped as an FMU is important to capture such state events of smart inverters. The aim of this wrapped system was to simplify the algorithm and to facilitate a effective separation of low-level FMUs (i.e., PV FMU and Smart Inverter FMU), mid-level FMUs (i.e., Distribution Model FMU and Coupled Feeder FMU), and high-level FMUs (i.e., Transmission Model FMU and Coupled Distribution FMU) with the objective of reducing solving time. The complexity with this setup was the need to evaluate the coupled systems in parallel, while the coupling on higher levels required a serial evaluation. In addition the partitioning on compute nodes on the Lawrencium HPCC required the discrete separation of the models on compute nodes. One drawback of the current implementation is the limitation that state events cannot be propagated to higher-level FMUs. In case of the cascade effect described earlier, a higher-level distribution or transmission system would not recognize the event until the next timestep was reached. This limitation originates in the implementation of the developed SimulatorToFMU tool to export Python scripts as FMUs. As described in Section 3.2, the Coupled Feeder FMU appears to raise state events, but the actual implementation of the underlaying C function does not support this functionality. This is a recognized limitation of SimulatorToFMU, and it is being discussed for future implementation by providing a zero-crossing function to project and raise such state events.

Another simplification taken in this paper was the export of the high-level FMUs with the *CS* API, while lower-level FMUs use the *ME* API. As described in the Introduction, *ME* allows to extract derivatives and roll-back time, which are important features to solve coupled systems. The *ME* model is therefore generally preferred for the application in power systems, especially when algebraic loops (e.g., from feedback controllers) are present. However *ME* requires an external solver and handling of events, whereas *CS* internally solves this problem. It is therefore easier to interface with *CS* FMUs, to build a customized orchestrator. In addition, the added functionality of state events would not be exploited because of the previously discussed limitation of SimulatorToFMU.

For future work, simulations with real load profiles could be done. Due to the lack of open source load profiles based on real measurements, this study used the simulated OpenEI load profiles for DOE reference buildings. This results in less variety of loads and therefore less realistic scenarios. Nevertheless, realistic and authentic data are not necessary to proof the functionality of the large scale simulation itself. The connection between all levels of FMUs was proven in this paper. For more detailed real world scenarios simulated with the developed platform, a higher variety of real load profiles would be preferable.

In future it also might be possible to upgrade the architecture with the FMI 3.0 standard (alpha version released). The proposed functionalities include ports and icons, array variables, clocks and hybrid co-simulation, binary data type, intermediate output values, and source code FMUs (FMI 3.0, 2018). Especially the clocks and hybrid co-simulation functionality, which allows the triggering of events in co-simulation mode, can have promising applications in large scale simulations. Longer timesteps can be used, without missing events in single FMUs and the resulting effects those effects would have on other FMUs. Because of the industry based nature of the FMI standard and the growing support of it, further developments to improve the usability and capabilities of the FMI can be expected.

# 6 Conclusion

The results of this paper demonstrate that the FMI standard offers a promising way to create large scale simulations. It is an easy and convenient way to combine a wide variety of models on a large scale, but to still simulate them with a high level of detail. The developed SimulatorToFMU tool enables users to further increase the number of available model libraries and simulation tools spanning into the power systems domain. While this paper focused on PV systems with smart inverters, many future scenarios involving multi-domain models could be conducted. Examples are detailed building simulations to replace simple load profiles, electric vehicles with individual availability and constraints, or advanced control systems such as Model Predictive Control for building heating, ventilation, and air conditioning systems to assess load shifting capabilities on a large scale.

Simulating large parts of the power grid with multi-level and high-fidelity resources provide a more realistic result compared to plain simulations of single voltage level and aggregated generation. By connecting many distribution feeders with a transmission grid model, the impact of feeders on one another is taken into account, which helps to reveal global grid challenges such as steep ramping demand in a high-PV deployment scenario.

This paper also demonstrated the ability of FMI and other tools to scaleup to about eighty thousand individual FMUs evaluated in a co-simulation. The multi-level hierarchical partitioning of the simulation into many parallel models, which run independently on different compute nodes at the Lawrencium HPCC, greatly reduces the simulation time and allows for a fidelity which is often not achievable on a single computer.

# Acknowledgment

This work was supported by the U.S. Department of Energy SunShot National Laboratory Multiyear Partnership (SuNLaMP) program award number 31266 and the California Energy Commission (CEC) Electric Program Investment Charge (EPIC) solicitation PON-14-303.

# References

Christian Andersson, Johan Åkesson, and Claus Führer. *Pyfmi: A python package for simulation of coupled dynamic models with the functional mock-up interface*. Centre for Mathematical Sciences, Lund University, 2016.

Thomas Basso, Sudipta Chakraborty, Andy Hoke, and Michael Coddington. IEEE 1547 Standards advancing grid modernization. In *2015 IEEE 42nd Photovoltaic Specialist Conference (PVSC)*, pages 1–5. IEEE, 2015.

Angel A Bayod-Rújula. Future development of the electricity systems with distributed generation. *Energy*, 34(3):377–383, 2009.

Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

Rich Christie and Iraj Dabbagchi. IEEE 57-bus system. 1993.

Scott D Cohen, Alan C Hindmarsh, and Paul F Dubois. CVODE, a stiff/nonstiff ODE solver in C. *Computers in physics*, 10(2): 138–143, 1996.

California Energy Commission et al. Rule 21 Smart Inverter Working Group Technical Reference Materials. *California Energy Commission*, 2014.

Department of Energy, s.a. Commercial and Residential Hourly Load Profiles for all TMY3 Locations in the United States, s.a. https://openei.org/doe-opendata/dataset/commercial-and-residential-hourly-load-profiles-for-all-tmy3-locations-in-the-united-states, Accessed: 2019-09-24.

FMI 3.0, 2018. FMI 3.0-alpha feature list. https://fmi-standard.org/news/2018/05/30/fmi-3-0-alpha-feature-list.html, 2018. Accessed: 2019-10-27.

Christoph Gehbauer, Joscha Mueller, Tucker Swenson, and Evangelos Vrettos. Photovoltaic and Behind-the-Meter Battery Storage: Advanced Smart Inverter Controls and Field Demonstration. https://github.com/LBNL-ETA/SCooDER, 2019.

Tim Godfrey, Sara Mullen, David W Griffith, Nada Golmie, Roger C Dugan, and Craig Rodine. Modeling smart grid applications with co-simulation. In *2010 first IEEE International conference on smart grid communications*, pages 291–296. IEEE, 2010.

JModelica.org, s.a. JModelica.org. `https://jmodelica.org`, s.a. Accessed: 2019-09-26.

Georg Kerber. *Aufnahmefähigkeit von Niederspannungsverteilnetzen für die Einspeisung aus Photovoltaikkleinanlagen*. PhD thesis, Technische Universität München, 2011.

K Krishna and Narasimha M Murty. Genetic K-means algorithm. *IEEE Transactions on Systems Man And Cybernetics-Part B: Cybernetics*, 29(3):433–439, 1999.

LBNL, s.a. High Performance Computing at Berkeley Lab. `https://sites.google.com/a/lbl.gov/lrc/home`, s.a. Accessed: 2019-09-27.

Rong-Ceng Leou, Chun-Lien Su, and Chan-Nan Lu. Stochastic analyses of electric vehicle charging impacts on distribution network. *IEEE Transactions on Power Systems*, 29(3):1055–1063, 2013.

Modelica Association, s.a. List of tools supported by FMI. `https://fmi-standard.org/tools/`, s.a. Accessed: 2019-10-27.

Thierry Nouidui and Michael Wetter. SimulatorToFMU v0. 1. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2017.

Thierry Nouidui, Jonathan Coignard, Christoph Gehbauer, Michael Wetter, Jhi-Young Joo, and Evangelos Vrettos. Cyder–an fmi-based co-simulation platform for distributed energy resources. *Journal of Building Performance Simulation*, 12(5):566–579, 2019.

Krzysztof Rudion, Antje Orths, Zbigniew A Styczynski, and Kai Strunz. Design of benchmark of medium voltage distribution network for investigation of DG integration. In *2006 IEEE Power Engineering Society General Meeting*, pages 6–pp. IEEE, 2006.

Thomas Stetz, Frank Marten, and Martin Braun. Improved low voltage grid-integration of photovoltaic systems in Germany. *IEEE Transactions on sustainable energy*, 4(2):534–542, 2012.

Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. Pandapower–An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, 2018.

Michael Wetter, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. Modelica buildings library. *Journal of Building Performance Simulation*, 7(4):253–270, 2014.

# Modeling future heat pump integration in a power radial

Konstantin Filonenko[1] Mikkel Copeland[2] Klaus Jespersen[2] Christian Veje[1]

[1]Center for Energy Informatics, University of Southern Denmark, Denmark, {kfi,veje}@mmmi.sdu.dk
[2] EWII Energi A/S, Denmark, mico@teknologisk.dk, klj@slukefter.dk

## Abstract

This paper considers integration of heat pumps in Danish low-voltage residential power distribution network, hereafter referred to as "radial", providing a data-driven case study for the electrical package of the Modelica Buildings library. The loads in the distribution grid for year 2030 are estimated based on the requirement of fully sustainable energy system by the year 2050. Combined with local consumption, the total system model is validated by measuring mains transformer signal at the chosen radial. The maximum cable capacity is compared to future current flow estimations for a 2030 grid, simulated based on a specific official Danish scenario. The study shows, that there is no threat to the network cables, if only heat pumps are integrated. However, when maximum load is applied to the grid, the values of cable currents are relatively close to the limit, which may complicate integration of other technologies. The results point at recommendations for safety measures to protect electric lines during periods of rapid technological development.

*Keywords: Buildings Library validation, Danish electric grid, heat pumps, photovoltaics, sustainable energy, distributed generation, electric power distribution*

## 1 Introduction

### 1.1 Motivation

The Danish Energy Agency expects the heat production from heat pumps to reach 2265 GWh in 2030 equivalent to 23% of the total residential heat demand with almost linear increase over time (Energistyrelsen, 2018; Energistyrelsen, 2018a). Integration of heat pumps in the existing traditional electrical system is supposed to increase energy efficiency compared to fossil fuel-based heating sources (Bloess, Schill, et al., 2017), e.g. boilers in Denmark (Petrović and Karlsson, 2016; Nielsen, Morales et al., 2015).

However, new installations may compromise grid safety, if the increased load exceeds capacity of distribution lines designed at the previous technological stage. Similarity of heat pump control settings in different households can lead to cumulative dynamic effect of simultaneous heat consumption resulting in high peak loads in the electricity grid. Transients will, therefore, play an important role in determining cable capacity limit which should be maintained to avoid damages.

On the other hand, heat pumps can increase flexibility of the distribution grid by postponing heat production when utilizing the heat storage in buildings (Fischer and Madani, 2017). As a result, one consumer's production can be shifted in time to match another consumer's consumption, thereby reducing the overall peak. Real-time model predictions may play, in this case, an important role for controlling power flows in the distribution grid and mitigating simultaneity effect.

This imposes requirements on model complexity/accuracy and its computational efficiency. Simultaneous production and consumption require that the bidirectional power flows are considered. Optimized control requires that the model is either black-box or lower-order grey-box to minimize the response delay. It is however desirable to have a validated white-box model available for comparison to ensure that no overfitting occurs when estimating the reduced-order model parameters and that it can be applied outside the training data range.

### 1.2 State of art

Modelica Buildings library has functionality for developing computationally efficient grey-box models of multi-energy systems (Lawrence Berkeley National Laboratory, 2019). The electrical package is suitable for simulating networks with bidirectional power flows and for optimization and control combining different energy dynamics in the same model (Bonvini, Wetter, and Nouidui, 2014).

Other Modelica libraries, capable of simulating electric distribution grids are IDEAS, PowerSystems, IPSL, OpenIPSL, ModPowerSystems, ObjectStab, PNlib, Electric Power Library (Winkler, 2017; Franke and Wiesmann, 2008). However, only Buildings and IDEAS contribute to the IBPSA library, which aims to standardize district energy simulation and optimization and integrate modeling with GIS data handling.

On the other hand, commercial Modelica libraries (Electric Power Library, Wind Power Library) and non-Modelica proprietary tools (DIgSILENT PowerFactory, POWERSYS EMTP-RV, PSCADTM, Siemens

PSS/ER®) restrict exchange of validated models and complicate their use for model predictive control.

For this reason and due to availability of an easily comprehensible example (Wetter, Bonvini, et al., 2015), the electrical package from Buildings is chosen here to build a validated multi-energy white-box model with bidirectional power flow between prosumers and the main transformer. This model reuses components from Buildings.Electrical with minor modifications (Copeland and Jespersen, 2019).

### 1.3 Aim of research

The models of the building electrical systems from IDEAS package including converters and adapters are somewhat more detailed for needs of the local distribution system operator (DSO) when estimating the consumption and testing network capacities. Buildings library, on the other hand, does not include a component representing an internal electrical system of the consumer. Additionally, no system-level experimental model validation of electrical package is known to the authors. Few examples considered in the above library descriptions did not present any time-varying experimental validation. Therefore, the three goals set in this work are to

1. Develop a component responsible for the interaction of DSO and electricity consumer (not available in literature) combining loads for traditional household appliances and new technologies (heat pumps) with production from local energy sources (photovoltaics).
2. Compare the output of a data-driven model of the radial in a Danish residential area to measurements of the mains transformer signal obtained from the DSO during March 2019.
3. Apply the model to estimate the future current flow as a result of future network development, when more and more heat pumps are integrated to achieve the energy sustainability goals for 2050 established by Danish government.

## 2 Cable box model

### 2.1 Concept

This paper considers an important aspect of interaction between the DSO and residential consumer electrical subsystem through an interface called here a cable box (kabelskab in Danish electric sector). The cable box is a simple cabinet with a busbar, providing certain services to the consumer (ABB, 2020). It usually stands by the side of the pavement close to the households it serves and includes fuses and cable connectors. To understand the purpose of the cable box, it should be considered within the context of the overall power distribution system.

The transformer maintained by the DSO converts electric power from 10 kV to 0.4 kV to be distributed to several districts each via its own supply cable owned by the DSO. To clarify this structure, the term radial is used for each network emphasizing that it extends from the transformer along with other radials attached to the same transformer. The radial may be spilt several times and have any topology accepted for electric distribution networks. Several cable boxes are located along the radial and terminate the supply to the consumers. DSO is responsible for delivering the power only to the cable boxes, so it could be said that the cable box encapsulates the part of electrical system, for which consumers are solely responsible.

### 2.2 Implementation

In the considered case, a cable box connects consumers (typically, 3 to 4 households per cable box in Denmark) to a distribution line aggregating their power flows in either direction. For this reason, the cable box model seems to be the most appropriate building block for the Danish low-voltage distribution network model.

It is implemented here as a class extending classes from electrical package of Buildings Modelica Library constructed in the way similar to the Power Systems Library (Franke and Wiesmann, 2014) and Electric Power Library (Modelon, 2019). The package relies on overdetermined connectors to represent bidirectional power flows (Franke and Wiesmann, 2014). The replaceable phase system within a connector is analogous to the replaceable medium used by Modelica.Fluid library to model generic fluid flow (OpenModelica, 2019).

The implementation of the cable box component is shown in Figure 1. Both traditional and heat pump loads shown in the figure with their time tables extend *Buildings.Electrical.AC.ThreePhasesBalanced.Loads*, while the photovoltaic (PV) class instantiated three times extend from *Buildings.Electrical.AC.OnePhase.Sources.PVSimpleOriented*. The instances represent different orientations of the PV cells with respect to direct, diffuse and reflective irradiation. Both components are connected to the distribution grid via the one-phase terminal.

The parameter interface of the cable box shown in Figure 2 includes the names of the files containing time series for traditional and heat pump loads and the parameters of the PV cell: area and orientation. The PV model for each orientation is based on the total PV surface area A, active area fraction f, as well as module and DC/AC conversion efficiencies, η and ηc, and calculates the total electric power produced by the PV:

$$P = I_s A\, f\, \eta \eta_c \qquad (1)$$

The total irradiation $I_s$ is supplied as an input to the component and is composed of three time series corresponding to the direct, diffuse and reflected irradiation. The traditional and heat pump electric systems are approximated by the linearized loads with fixed power factor of 0.98 and variable power supplied externally as a time series discussed in the next section.

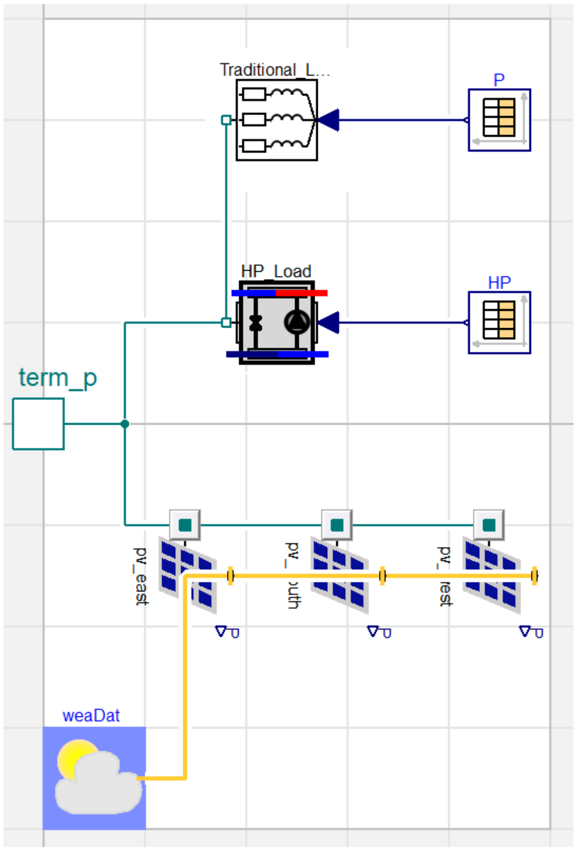
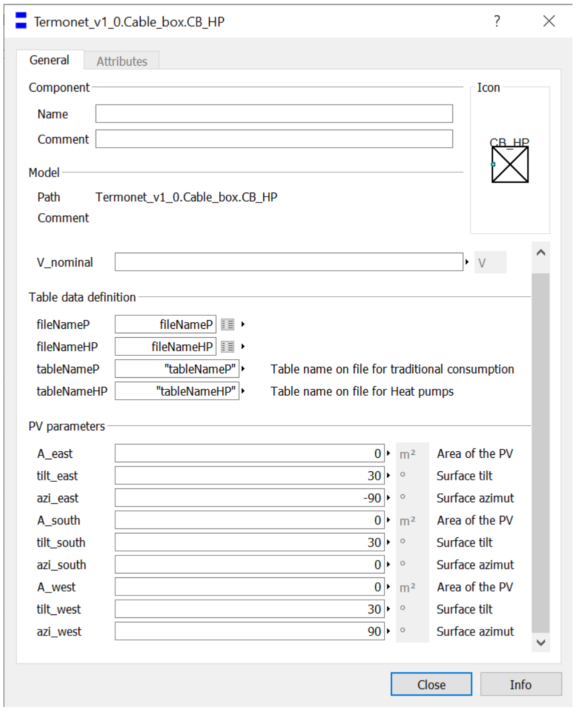


**Figure 1.** Modelica model of a residential cable box



Termonet_v1_0.Cable_box.CB_HP

Component

Name

Comment

Model

Path Termonet_v1_0.Cable_box.CB_HP

Comment

Icon

CB_HP

V_nominal V

Table data definition

| | | |
|---|---|---|
| fileNameP | fileNameP | |
| fileNameHP | fileNameHP | |
| tableNameP | "tableNameP" | Table name on file for traditional consumption |
| tableNameHP | "tableNameHP" | Table name on file for Heat pumps |

PV parameters

| | | | |
|---|---|---|---|
| A_east | 0 | m² | Area of the PV |
| tilt_east | 30 | ° | Surface tilt |
| azi_east | -90 | ° | Surface azimut |
| A_south | 0 | m² | Area of the PV |
| tilt_south | 30 | ° | Surface tilt |
| azi_south | 0 | ° | Surface azimut |
| A_west | 0 | m² | Area of the PV |
| tilt_west | 30 | ° | Surface tilt |
| azi_west | 90 | ° | Surface azimut |

**Figure 2.** Parameter dialog of the cable box model

## 2.3 Verification

To ensure, that the sub-models provide reliable representation of the real cable box, they are verified in Figure 3 and Figure 4 against Matlab computations relying on analytical formulas and the following relation between the total current I, power factor pf and nominal grid voltage U (Copeland and Jespersen, 2019):

$$I = \frac{P}{pf\ U} \qquad (2)$$

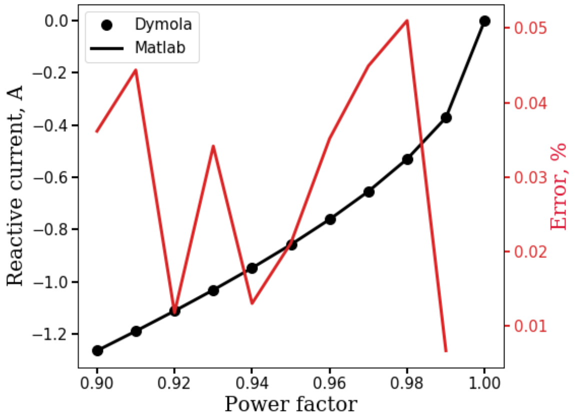


**Figure 3.** Verification of the load component.

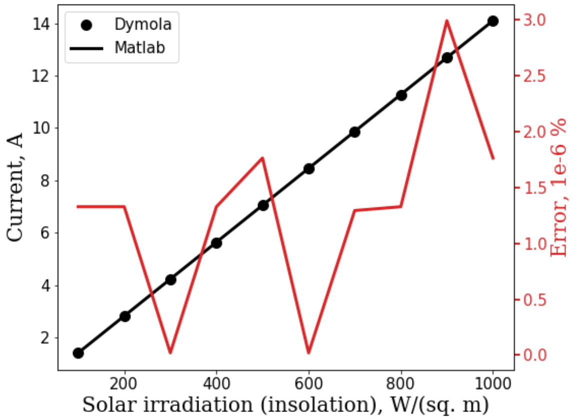


**Figure 4.** Verification of the PV component

Load calculations are compared in Figure 3, where the verification of the load component is done by varying the power factor in the range from 0.9 to 1 and calculating the reactive current from the total current and the power factor:

$$I_r = I \sin(\text{acos}(pf)) \qquad (3)$$

To calculate $I$, $P$ is set equal to the nominal power of the grid in Eq. (1). The load performs as expected with an error magnitude of order 1E-2%.

The objective of the PV verification shown in Figure 4 is to ensure that the output current from the component *Buildings.Electrical.AC.OnePhase.Sources.PVSimple* is as expected for a set of standard PV parameters. The active power in Eq. (2) is found from Eq. (1), and the irradiation base for the calculation is selected to be from 100 W/m$^2$ to 1000 W/m$^2$ to imitate real operation conditions. The error in this case is on the order of 1e-6%.

# 3 System model

The model of the radial shown in Figure 5 is constructed from the verified cable box (Section 2) and distribution components (Section 3.1) connected to the grid voltage source – a 10kV to 0.4kV transformer. It uses the actual geographical layout to retain the visual information for future references. At the top level, the average load profiles and cable parameters are set by redeclaring the suitable replaceable records.

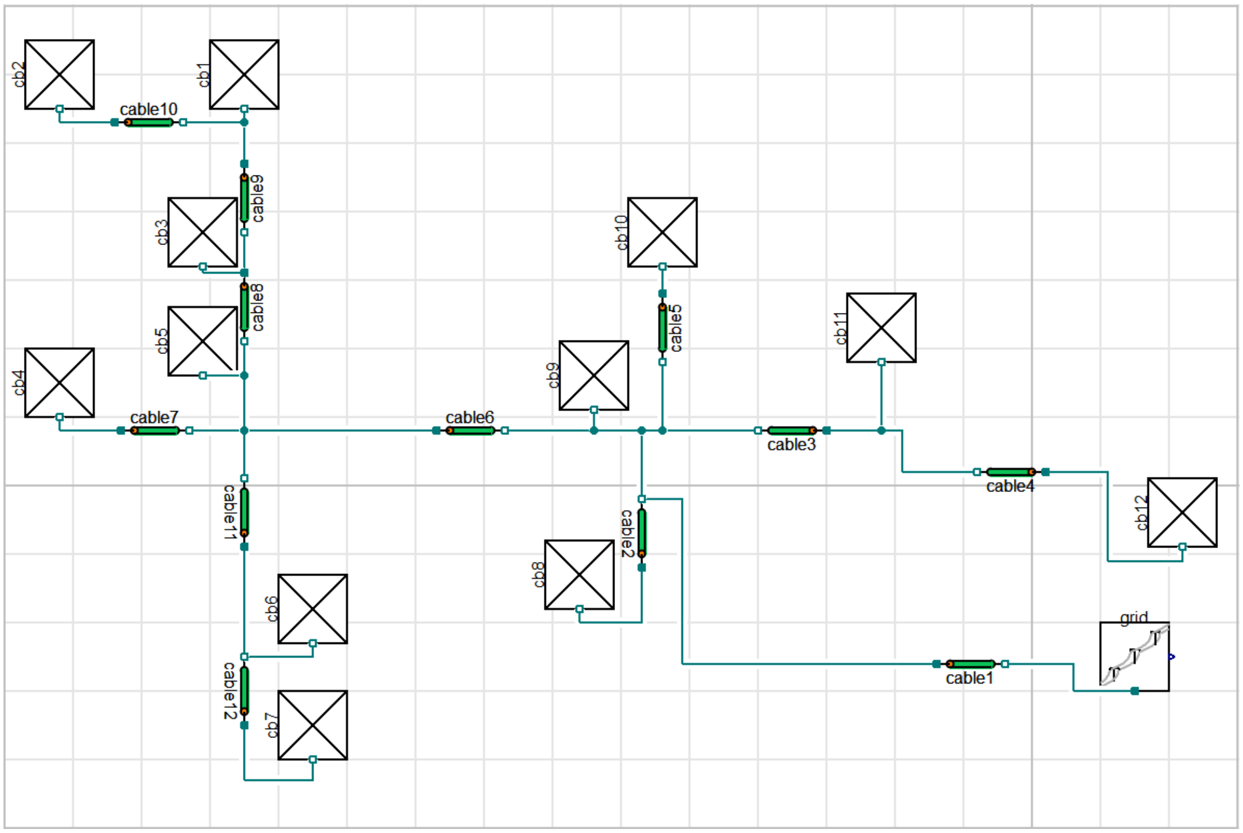


**Figure 5.** Model of the Danish radial (distribution cables: green cylinders, cable boxes: squares with diagonals drawn, grid: square with transmission lines drawn)

## 3.1 Distribution

To create a distribution line model, the class *Buildings.Electrical.AC.OnePhase.Lines.TwoPortRLC* is extended from Buildings library. The impedance of the distribution cables is calculated based on characteristic resistances and reactances taken from data sheet provided by the manufacturer. Both voltage and power losses are calculated in the model, however, the temperature dependence of the resistance is not accounted for in order to simplify the analysis.

The distribution line model is verified in the range of the power factors from 0.9 to 1 in Figure 6. The figure compares the voltage loss over the line segment obtained in Modelica simulation and the same voltage loss calculated in Matlab by using the formula

$$\Delta U_f = I\, l_c\, (r\, pf\ + x\, \sin(\arccos(pf))) \qquad (3)$$

where $l_c$ is the length of the cable and $r$ and $x$ are characteristic resistance and reactance found in manufacturer datasheet. To calculate $I$ from Eq. (1), P has been set equal to the nominal active grid power.

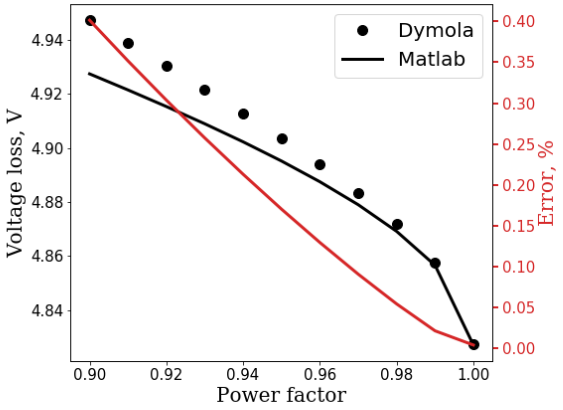


**Figure 6.** Verification of distribution line component

The error in Figure 6 goes up to half of the percent for power factors close to 0.9, but decreases below 0.1% in the range around 0.98, which is the value mainly encountered in the actual system. It can be concluded that the model is accurate enough for the considered power factor values.

## 3.2 Radial network topology and time series

The model of the radial shown in Figure 5 accurately reproduces the topology of the DSO installation and includes two main nodes (1) connecting cable box cb5 to the distribution cables 6, 7, 8, 11 and (2) connecting cable box cb9 to the distribution cables 1, 2, 3, 5, 6. Cable 1 serves as a supply cable from the main transformer.

To ensure that the radial is accurately represented in simulations, the seasonal consumption profiles for both traditional and heat pump loads were provided by the Danish DSO. The average of these electricity consumption time series for the traditional load is shown in Figure 7.

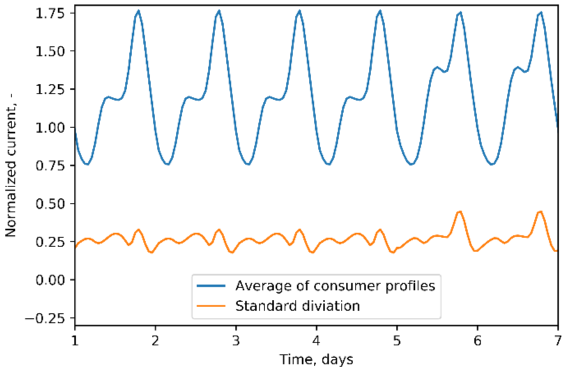


**Figure 7.** Average and standard deviation of the electrical consumption profiles used in modeling.

Each of the 30 individual profiles, which average is calculated in the figure, accounts for lighting, electronic devices, cooking, etc., but exclude domestic heating, since the radial is located in a district heating area. These profiles are intended for capturing a typical consumption pattern and are not used directly for a specific consumer in the model. The corresponding cable box loads are calculated as average values of the profiles of the consumers attached to the same cable box for each time moment.

The average takes into account the probability of each profile as corresponding to the common behavior pattern. Since the original consumption profiles represent real households, they are expected to be accurate with regards to the system construction and objectives of the model. The profiles, however, have limitations in terms of resolution since single occurrence peak values are leveled out by the mean calculation. From the analysis of the daily and varying seasonal profiles, the consumption showed the expected behavior of lower consumption in the summer compared to the winter. This consumption change depends on temperature and social behavior pattern in using lighting and electronic devices.

The heat pump electrical consumption W is defined by the domestic hot water (DHW) and space heating (SH) demands:

$$W = (DHW + SH)/COP \qquad (4)$$

where the heat pump coefficient of performance (*COP)* is calculated from the following regression formula obtained from manufacturer data:

$$COP(T) = (-4.2E\text{-}5)T^3 + (4.9E\text{-}4)\,T^2 + (6.9E\text{-}2)T + 2.4 \qquad (5)$$

Correspondingly, the heat demand curve in Figure 8 is generated from two profiles with SH part produced by the heating degree day model

$$SH = -0.16\,T + 2700 \qquad (6)$$

and the DHW part calculated based on the $800kWh$/year/person usage for 24 hour period and repeated throughout the year.

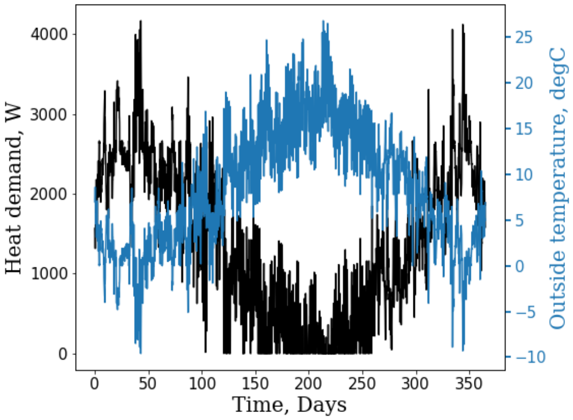


**Figure 8.** Modelled yearly heat pump demand profile

The time series for PV irradiation supplied to each PV orientation through the information connectors in Figure 1 are produced as a sum of the three contributions:

$$I = I_{dir}\,R_{dir} + I_{dif}\,R_{dif} + I_{ref}\,R_{ref} \qquad (7)$$

where the direct, diffuse and reflected irradiation terms are governed by the following geometric relations (Copeland and Jespersen, 2019):

$$R_{dir} = cos(\theta_i)/cos(\theta_z) \qquad (8)$$

$$R_{dif} = A_I R_{dir} + (1 - A_I)\,(1 + cos(\beta))/2 \qquad (9)$$

$$R_{ref} = (1 - cos(\beta))\,\rho/2 \qquad (10)$$

with incident, zenith and horizontal tilt angles denoted by $\theta_i$, $\theta_z$ and $\beta$ and the surface reflectiveness and anisotropic index denoted by $\rho$ and $A_I$.

## 3.3 Validation

The model of a cable box is validated in Figure 9 against the current measured at the transformer station supplying the radial with power (grid component in

Figure 5). The current, voltage, power factor, etc. of the system were continuously monitored in the period from March 8 to March 19, 2019 by the sensor located between the grid and cable 1 (the measurement data outside of this time period is not available). Figure 9 shows the rolling average of the current measurements over the 1-hour window (orange curve), the total current simulated in Modelica (blue curve) and the root mean square error (green curve) between the two curves. Deviation of the simulated curve from the measured curve can be explained by the difference in assumed and actual ambient conditions during March 2019 resulting in a distorted PV production pattern. Additionally, the outliers seen in the figure are impossible to predict, since they depend on which appliances are connected to the cable box. In terms of the short-time load exertion on the transmission line, the peak has no real influence since the cables can handle much higher short-time current than the limit set by the providers.

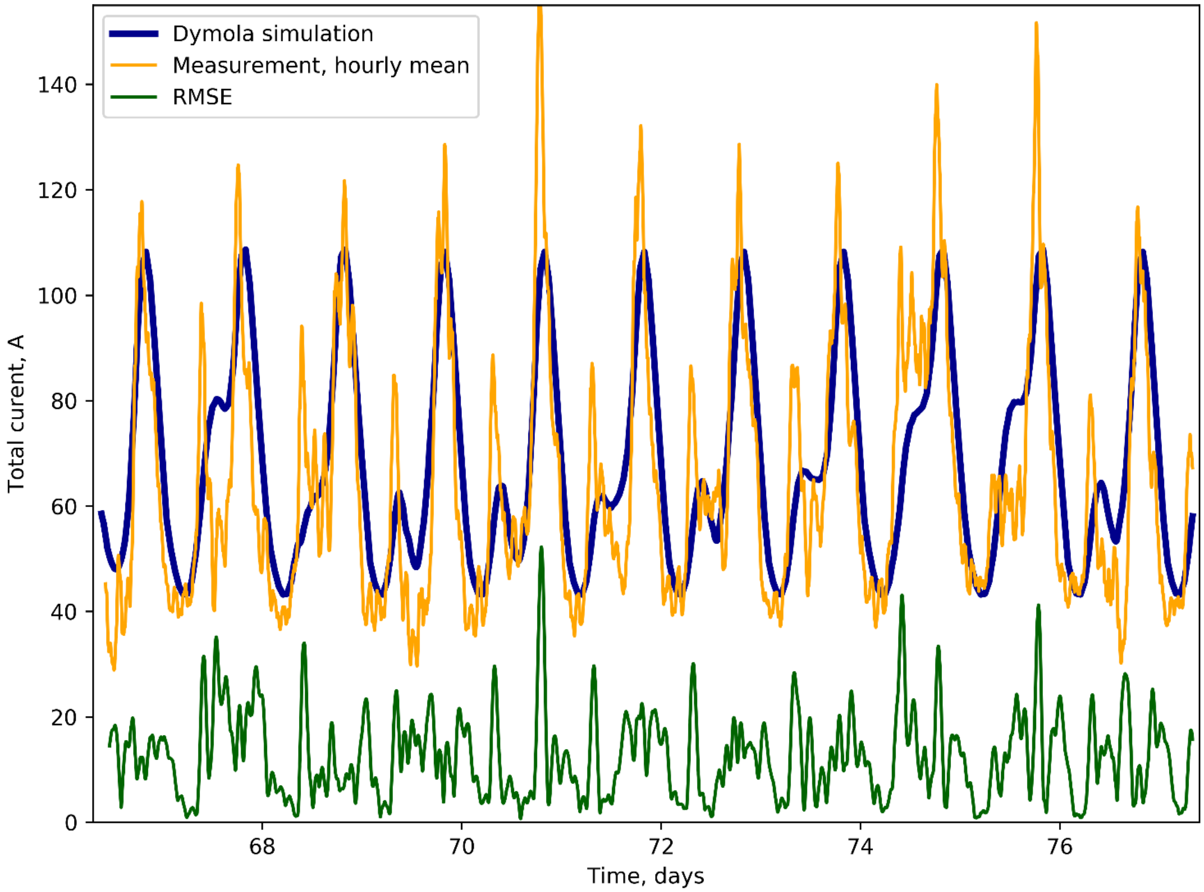


**Figure 9.** Validation of the radial model: comparison of the electric current values calculated in Modelica (blue curve) and the measured transformer current (orange curve) over two weeks of March 2019 and the root mean square error between the measured and calculated values (green curve).

## 4 Results

The heat pump integration scenario simulated here is proposed by Danish Energy Agency (Energistyrelsen, 2019), which did not anticipate considerable increase in either the traditional demand or the number of PV cells.

It is therefore assumed that other loads and PV production remain unchanged and only the impact of heat pump integration is considered according to the three scenarios summarized in Table 1: *Static scenario* is based on 0% forecast for the heat pump integration rate before the year 2030, which is not very probable, but will serve as a reference for other projections. *Moderate increase* scenario is based on DEA 23% projection for the electric heat pump units installed in Danish electrical system. To probe the system safety under the full heat pump penetration, the third, *Maximum increase* scenario is based on the 100% projection, corresponding to the case where heat pumps supply heat to all consumers on the radial.

**Table 1.** Heat pump integration scenarios.

| *Scenario* | *Projection* |
|---|---|
| Static | 0% |
| Moderate increase | 23% |
| Maximum increase | 100% |

In Figure 10, the total currents through the supply cable 1 and the peripheral cable 8 (colored curves) calculated as $i_{tot} = \sqrt{Re(i)^2 + Im(i)^2}$ are drawn for all three scenarios. These cables were chosen because they carry the largest current for their cable sizes. As could be expected, the currents through the cable 1 are larger by absolute value than the corresponding currents through cable 8, due to the fact that the supply line is exposed to all loads. Additionally, cable 1 has 3 PV installations of 5+3 kW, 5+3 kW and 5 kW while cable 8 has none. The capacities of cable 1 and cable 8 are shown in the figure as a dotted and a solid line, respectively.

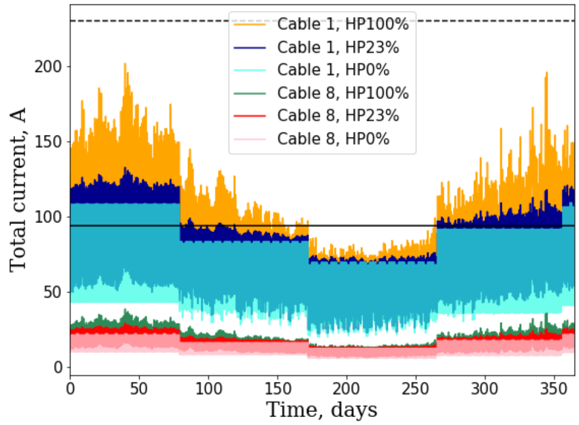


**Figure 10.** Heat pump penetration scenarios (year 2030). Cable capacities: cable 1 - 230 A, cable 8 - 94 A. HP0% curves are made semi-transparent.

In the wintertime corresponding to the right quarter of the figure, the current in the reference static scenario has stable amplitude, changing only during the transition periods from colder to the warmer time. During these periods, the light availability for PV generation increases, thereby reducing the current required to power the cable boxes. The sharp drops in the figure occur, because the profiles for each new season are recalculated from previous datasets, which is a normal practice to be able to forecast and plan consumption. When the heat pump load is partially included within the Moderate increase scenario, this transition effect is superposed by the variation of the amplitude within each season due to direct influence of the increased heat pump capacity on the electric system.

Further explanation can be drawn from Figure 11, which includes the same currents having the same color codes as in Figure 10, but in February of the same year.

Larger heat demand in coldest period explains the global current peak in February for Moderate and Maximum increase scenarios. However, in the latter case, the system is more sensitive to the temperature variations, and consumption is increased due to reduced heat pump COP at lower ambient temperatures and larger installed heat pump capacity. The more tangible amplitude modulation can be attributed to the effect of coincidence of the heat demand events in response to systematic reduction of the ambient temperature. This effect becomes stronger, when larger number of consumers chose the same heating strategy in the cold period. Since in present study, the simplest strategy is implemented for all active consumers, the increase of their number leads to an unequivocal effect of current fluctuations following the consumer heat consumption.

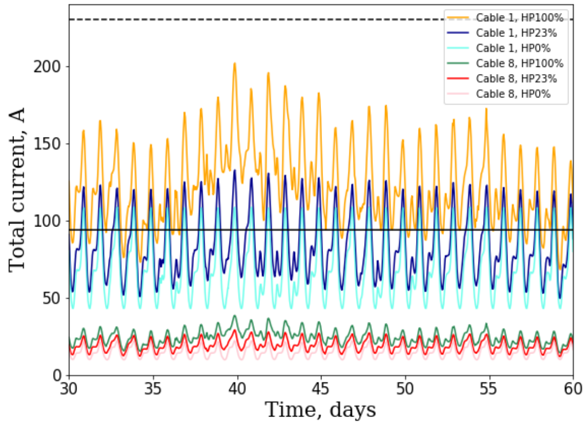


**Figure 11.** Total current in February 2030. The legend from Figure 10 applies.

In summer, on the other hand, the current values shown for all three scenarios in Figure 12 are close to each other, since less heating is needed, and the major part of the installed capacity is not used. The analysis of the peripherical lines (only Cable 8 is included in Figures) shows similar results with an increased current flow where heat pumps are installed and is discussed in the next section.

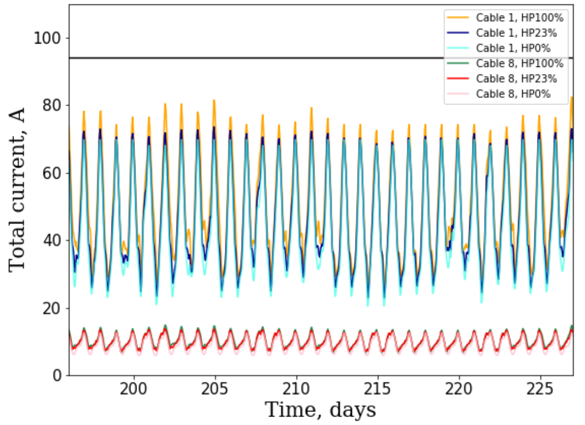


**Figure 12.** Total current from mid-July to mid-August 2030. Legend from figure 10 applies.

## 5 Discussion and outlook

Based on the obtained results, the simulated radial will perform satisfactorily for the 2030 projection offered by DEA, if only heat pumps are integrated. The supply cable 1 has the maximal mean current flow in winter equal to the half of the cable capacity in the Moderate increase scenario. The currents through peripheral distribution cable 8 and, therefore, all other peripheral cables are much less than its capacity limit.

If the radial is located in the area with 100% penetration of heat pumps (Maximum increase scenario), the current flow experiences significant seasonal fluctuations and the February peak exceeds 200 A. This and other peaks are relatively close to the capacity limit. Therefore, the integration of other technologies, such as electrical vehicles, may lead to an electrical signal exceeding the cable capacity.

The authors propose the following remedies, if the distribution currents exceed the installed capacities, which constitute the subject of future research on the topic:

1. Reduce simultaneity of consumption through load shifting and peak shaving
2. Introduce centralized model predictive control of heat pump's set points in different households to decrease the overall load on the system
3. Use flexibility of the thermal storage in buildings.

## 6 Conclusion

In this paper, the cable box component has been developed and validated against measurements at Danish residential radial. The component is applied to investigation of the integration of heat pumps in a typical Danish electric power distribution radial providing a data-driven case study for the electrical package of the Modelica Buildings library.

The maximum cable capacity is compared to future current flow estimations for 2030 grid simulated based on a specific Danish officials' scenario. Three simulations were performed to investigate the limit to which the distribution lines are stressed, if (1) no heat pumps are integrated in the households, (2) only 23% of the households adopted heat pumps and (3) if all households have heat pumps.

Based on the output data from Modelica and the subsequent analysis of the result, it can be concluded that there is no immediate threat to the cable capacity of the transmission lines in the radial. However, in the Maximum increase scenario, the values of cable currents are relatively close to the limit, which may complicate integration of other technologies. Therefore, to ensure the grid safety, the three solutions are proposed. The research and the model can be further modified to help the electric distribution system operators to estimate the grid load under known consumer behavior and the rapidly changing electric grid infrastructure.

Additionally, the use of Modelica as a modeling tool in the considered context offers fast and physically sound models of energy conversion systems like heat pump providing a potential benefit for dynamic simulation of sector coupling. The described model is purely electrical but can be relatively easily converted to the multi-energy system by adding relevant mechanical, hydraulic and/or thermal components. This will allow to study the influence of heat pump control and heat storage flexibility in buildings on power grid performance, which is left for future work.

## Acknowledgements

This manuscript emerged from the IBPSA Project 1, an international project conducted under the umbrella of the International Building Performance Simulation Association (IBPSA). Project 1 will develop and demonstrate a BIM/GIS and Modelica Framework for building and community energy system design and operation. The work was financed (research part) by the Danish Ministry of Energy, Utilities and Climate for the project DK Energy Live Lab – Vejle Nord (No. 2017-3451) and (participation) by Danish Energy Agency (Det Energiteknologiske Udviklings- og Demonstrationsprogram, EUDP) for the project IBPSA Project 1 CEI - SDU Participation (64018-0518).

## References

ABB. Kabelskabe Combi-Line. Web-page containing the product description, 2020. url: https://new.abb.com/dk/om-abb/vores-forretning/electrification-products/udendoers-kapslinger/combi-line

Andreas Bloess, Wolf-Peter Schill, Alexander Zerrahn. Power-to-heat for renewable energy integration: A review of technologies, modeling approaches, and flexibility potentials. *Applied Energy*, No 9, pp. 1611-1626, 2018. doi: https://doi.org/10.1016/j.apenergy.2017.12.073.

Marco Bonvini, Michael Wetter, and Thierry S. Nouidui. A Modelica package for building-to-electrical grid integration. *BauSIM 2014 Conference, Aachen, Germany, September 2014*.

Jochen Cremer, Marco Pau, and Ferdinanda Ponci. Optimal Scheduling of Heat Pumps for Power Peak Shaving and Customers Thermal Comfort. *In proceedings of 6th International Conference on Smart Cities and Green ICT Systems – SMARTGREENS 2017, 22nd-24th April 2017, Porto, Portugal*, pp. 23-34.

Mikkel Copeland and Klaus Jespersen. Modeling the future distributed electricity grid. *Master thesis, June 2019, University of Southern Denmark*.

Energistyrelsen. Analyseforudsætninger til Energinet. 2018. url: https://ens.dk/sites/ens.dk/files/Analyser/analysefoudsaetninger_til_energinet_2018.pdf.

Energistyrelsen, Basisframskrivning 2018; Energi- og klimafremskrivning til 2030 under fravær af nye tiltag. 2018a. url: https://ens.dk/sites/ens.dk/files/Analyser/basisfremskrivning_2018.pdf.

Rüdiger Franke and Hansjürg Wiesmann. Flexible modeling of electrical power systems – the modelica powersystems library. *Proceedings of the 10th International Modelica Conference*, 4(2):137–147, 2014.

David Fischer and Hatef Madani. On heat pumps in smart grids: A review. *Renewable and Sustainable Energy Reviews*, No 70, pp. 342-357, 2017. doi: https://doi.org/10.1016/j.rser.2016.11.182.

Lawrence Berkeley National Laboratory, Modelica Buildings library, 2019. https://simulationresearch.lbl.gov/modelica/

Modelon. Electric Power Library, 2019. url: https://www.modelon.com/library/electric-power-library/

Maria G. Nielsen, Juan M. Morales, Marco Zugno, Thomas E. Pedersen, Henrik Madsen. Economic valuation of heat pumps and electric boilers in the Danish energy system, *Applied Energy*, No 167, pp. 189-200, 2016. doi: https://doi.org/10.1016/j.apenergy.2015.08.115.

Hans Olsson, Martin Otter, Sven E. Mattson and Hilding Elmqvist. Balanced Models in Modelica 3.0 for Increased Model Quality. *Proceedings Proc. of the 7th Modelica Conference, Bielefeld, Germany, March 2008.* pp. 21-33.

OpenModelica. Modelica.Fluid.UsersGuide.Overview, 2019. url: https://build.openmodelica.org/Documentation/Modelica.Fluid.UsersGuide.Overview.html

Stefan Petrović and Kenneth Karlsson. Residential heat pumps in the future Danish energy system. *Energy*, No 114, pp. 787-797, 2016. doi: https://doi.org/10.1016/j.energy.2016.08.007.

Michael Wetter, Marco Bonvini, Thierry S. Nouidui, Wei Tian, and Wangda Zuo. MODELICA BUILDINGS LIBRARY 2.0. *Proceedings of BS2015: 14th Conference of International Building Performance Simulation Association, Hyderabad, India, Dec. 7-9, 2015*, pp. 387-394.

Dietmar Winkler. Electrical Power System Modelling in Modelica - Comparing Open-source Library Options. *Proceedings of the 58th Conference on Simulation and Modelling (SIMS 58) Reykjavik, Iceland, September 25th – 27th, 2017.* No. 138, pp. 263-270. doi: https://doi.org/10.3384/ecp17138263.

# Parameter Estimation of User-Defined Control System Models for Itaipú Power Plant using Modelica and OpenIPSL

Meaghan Podlaski[1] Luigi Vanfretti[1] Marcelo de Castro Fernandes[1] Jonas Pesente[2]

[1]Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, USA, {podlam,vanfrl, decasm3}@rpi.edu
[2]Itaipú Binacional, Brazil, pesente@itaipu.gov.br

## Abstract

The power industry heavily relies on power system modeling to understand system operations, perform system planning studies, and identify and correct problems that arise within the system. By minimizing the error between the models and actual physical system, it can be ensured that the models provide representation of both the existing and future the power system. Many of these models in the system are user-defined, i.e. they are specialized representations of a specific system component in the system. It is important that these customized models produce an accurate response. However, maintaining such models is costly, so it is of value to determine if those models can be replaced with a generic model. Phasor measurement data can be used to calibrate model parameters and reduce error. In this paper, this process is automated for a generator in the Itaipu power plant using RaPId, a MATLAB toolbox that integrates measurements, models using Modelica/FMI standards, and optimization routines. This is achieved by using a combination of particle swarm optimization (PSO) algorithm and classical gradient optimization routines to calibrate the model parameters. In this paper, the calibration of a generic model of a synchronous generator, automatic voltage regulator, and power system stabilizer are estimated and compared to the user-defined models for an automatic voltage regulator, power system stabilizer, and turbine governor.

*Keywords: Modelica, FMI, synchronous generator parameter estimation, PSO, system identification*

## Glossary

| | |
|---|---|
| **GENSAE** | Salient pole generator with exponential saturation |
| **AVR** | Automatic voltage regulator |
| **PSS** | Power system stabilizer |
| **TG** | Turbine governor |
| **SMIB** | Single machine infinite bus system |
| **Model description** | Source of the definition of the model via a standard or another document |
| **OpenIPSL model** | Model in Modelica available in the OpenIPSL library adapted from the model description |
| **Efd** | Field exciter voltage |
| **PMU** | Phasor measurement unit |
| **P** | Active power |
| **Q** | Reactive power |
| **Pmech** | Mechanical power |

## 1 Introduction

### 1.1 Motivation

New advancements in renewable power generation and control systems creates a resilient power grid and limits our negative impact on the environment. Simulation-based studies are helpful in determining which technologies have the highest benefit for the grid and the potential impacts of integrating a resource with the bulk electric grid.

Highly accurate dynamic power system models are necessary, especially in cases where user-defined models are used to simulate system conditions. Many renewable resources and control systems utilize user-defined models in their simulations, creating the burden of maintaining multiple models for system operators. It is necessary to determine the accuracy of these models and investigate if these models can be replaced by a generic, standardized model. Low confidence in the parameters of these models leads to more conservative and possible erroneous assessments of their responses to an event. There is also inherent uncertainty due to changes in the system parameters due to wear and aging of system components. Limited opportunities exist to test the physical power system because the existing system cannot be compromised for experimentation; building a new system for testing is not be a viable option, as it would be too costly (L. Vanfretti, W. Li, T. Bogodorova and P. Panciatici, 2013). The results of this paper expand upon (M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi, 2019), to show alternate ways to model power systems and derive user-defined model parameters with accuracy using Modelica and FMI. Previously, the IEEE standard models were used to represent the system. These models were calibrated using the same algorithms and methods used to calibrate the user-defined models. These results also compare the performance of the user-defined power system models to IEEE standard power system models.

Real-world measurement data from phasor measurement units (PMU) can be used to improve power system models. By implementing these models with Modelica and interfacing with other softwares using FMI, the parameters can be calibrated for various power system components. Using a set of measurements obtained from PMUs attached at the terminal bus of Itaipu Binacional, the world's second-largest renewable hydro-electric dam, the parameters of different components of the power generation can be calibrated. This particular generator studied in this work produces 700 megawatts (Itaipu Binacional), an amount of power capable of supplying a city of 1.5 million people.

## 1.2 Related Works

Previous studies for power system model calibration using PMU measurements focus on using different solver methods and standardized models. PMU measurements have been used for dynamic model validation and calibration using various methods such as extended Kalman filter techniques (Z. Huang, P. Du, D. Kosterev, and S. Yang, 2013). Existing conventional and renewable plants are calibrated in (J. Chen, P. Shrestha, S. Huang, N. D. R. Sarma, J. Adams, D. Obadina, and J. Ballance, 2012) using PMU data to help determine the cause of faults within the system. The dynamic parameter identification uses a combination of particle swarm optimization (PSO) and sensitivity analysis for a system consisting of a wind turbine, its reactive power support, and step up and step down transformers. The parameter identification produces good results for an undamped oscillation under weak grid conditions. The calibrated models helped operators find problems with the AVR of the plant for the fault studied, allowing for improved operations under weak grid conditions in the future. This is especially important for a plant like Itaipú, which provides such a large amount of power.

Modelica and the FMI standard have also been used extensively in power system model calibration. In (Andersson and Strömner, 2013), a multi-domain model for a wind turbine is calibrated using synthetic and real-life measurement data. The model calibration follows a sequential approach in calibrating system components similar to the one outlined in this paper. The calibration process outlined in (Andersson and Strömner, 2013) utilizes the optimization features in the Modelica Design Library in Dymola rather than exporting the models as FMUs to utilize optimization routines in other software.

## 1.3 Paper Contribution

This paper contributes a study focused on the calibration of a standardized generator model and user-defined models for its control system. Both a classical gradient optimization method and a particle swarm optimization method are utilized in model calibration. FMI Toolbox and RaPId in MATLAB were used to calibrate the models discussed in this study.

## 1.4 Paper Organization

This paper is organized as follows. Section 2 outlines the setup of the models using Modelica/FMI as well as the software-to-software validation. Section 3 outlines the optimization problem defined in MATLAB that will be solved using RaPId. Section 4 shows the results of the optimization using the methods outlined in the previous sections, comparing the fitness of each calibration method.

# 2 Creating Power System Models Using Modelica and OpenIPSL

The power system model was implemented in Modelica using Dymola, the OpenIPS library, as well as the Modelica standard library to create the user-defined models from Itaipu. The Itaipu generators are salient pole generators with exponential saturation, which is modeled using the GENSAE model, as derived in (Kundur). The turbine-governor (TG), automatic voltage regulator (AVR), and power system stabilizer (PSS) models are user-defined models that were originally implemented in Anatem, the simulation software Itaipu uses for their plant. The voltage measurements from the PMU are injected into the system at the terminal bus to calibrate the generator, TG, AVR, and PSS.

## 2.1 Modelica model

### 2.1.1 Modelica model overview

The model was implemented in Dymola using in the Modelica language using the OpenIPSL (M. Baudette, L. Vanfretti, J. Rabuzin, M. Murad) and Modelica Standard (MSL) libraries. The Itaipu generators are salient pole generators with exponential saturation, so the IEEE standard GENSAE model is used to create the system model. The equations used to model the generator are included in the Appendix. User-defined models from Itaipu's modeling software, Anatem, were implemented using Modelica for the AVR, PSS, and TG. These user defined models feature functions from both the MSL and OpenIPSL used to model the components. The components are modeled using transfer functions and behaviors specific to the Itaipu plant. In the cases where the IEEE controller models are studied, the models' behaviors and transfer functions are derived from the IEEE standard for excitation system models (IEEE, 2016). These models are configured in a manner such that we can export them as FMUs to be used in MATLAB for model calibration.

The power system model was developed using the Modelica language and Dassault's Dymola software as shown in Figure 1. The components are labeled as follows:

A. Tables containing the PMU data for the active and reactive power measurements.

B. System data contains frequency and base power for the system. The machineData block contains parameter data stored in a record, which is propagated to all system components. A record exists with the results of every parameter calibration test run.

C. User-defined turbine governor model from Itaipu.

D. GENSAE generator model [1], a salient pole generator with exponential saturation.

E. User-defined AVR model from Itaipu.

F. User-defined PSS model from Itaipu.

G. Active and reactive power outputs to be used in the FMU.

H. Controllable voltage source component.

I. Tables containing real and imaginary voltage components from the PMU measurements.

The names and details of the parameters calibrated from the components listed above are listed in the Appendix. Figure 2 shows the relationships between the components shown in Modelica model in Figure 1. The generator outputs an active (P) and reactive (Q) power, as well as speed derivation ($\Delta\omega$), mechanical power ($P_{mech}$), and electrical power ($P_{elec}$). The PSS uses $P_{elec}$ as an input to obtain an additional tracking signal ($V_{OTHSG}$) for the AVR's input. The AVR also needs the generator's exciter field voltage ($E_{fd0}$) and terminal machine voltage ($E_{comp}$) to determine what adjustments need to be made to $E_{fd}$. The governor, which is a speed controller for the generation unit, utilizes $\Delta\omega$ and the reference mechanical power ($P_{mech0}$) to control the turbine behavior in terms of speed and mechanical power. The turbine will then provide a mechanical power signal to the generator. These components are also mapped to a one line diagram of an SMIB system to provide understanding of where these components would be in a power system.

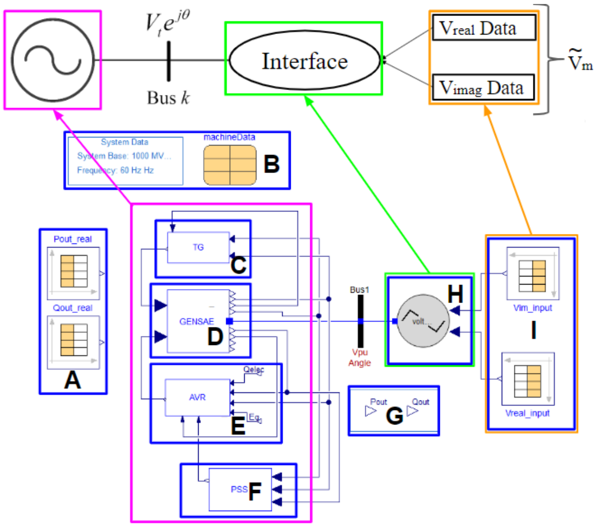


**Figure 1.** Modelica model of the generator, TG, AVR, and PSS in Dymola. This is compared to an SMIB one line diagram where all of the relationships are shown using magenta boxes.

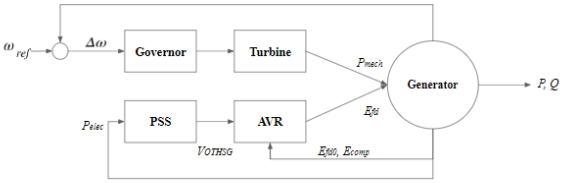


**Figure 2.** Relationship between system components.

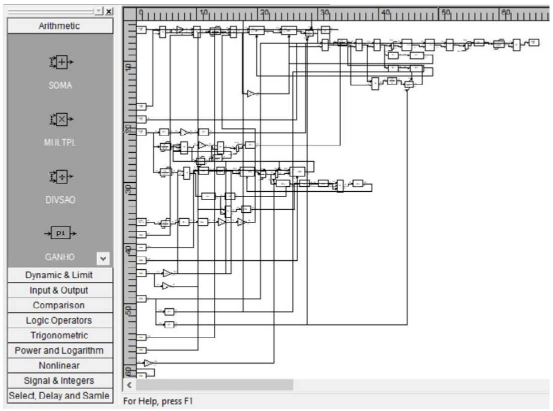


**Figure 3.** Itaipu User-Defined AVR model in CduEdit Software.

### 2.1.2 Re-implementation of User-defined Models

The user-defined models have been developed from models provided by Itaípu's engineers. They were created using the modeling software CduEdit. An example of how the models are set up in CduEdit is shown in Figure 3.

Some additional functions were created for the Anatem-Modelica equivalent user-defined models, such as the 'pulso' function (meaning 'pulse' in English), which was used in the modeling of the PSS and AVR:

```
model Pulso
  parameter Real p1 = 0;
  parameter Real p2 = 1;
  parameter Real p3 = 1e10;
  parameter Real p4 = 1;
  Modelica.Blocks.Interfaces.RealInput u;
  Modelica.Blocks.Interfaces.RealOutput y;
equation
  if u<p1 then y = 0;
  elseif (u>p1 and u<p3) then y = p2;
  else y=p4;
  end if;
end Pulso;
```

The Modelica implementation for the AVR model is shown in Figure 4. It includes a main AVR loop, overexcitation limiter, underexcitation limiter, VHz limiter, and a scaling factor. The main loop contains the transfer functions to regulate Efd, the field voltage. The scaling factor scales the initial Efd of the generator to the initial output of the AVR to ensure that the models are operating on the same base. The definition for each of the parameter abbreviations are located in the Appendix.

### 2.1.3 Brazilian software

The engineers at Itaipu used industry-specific software to implement their models. We re-implemented these models

[1] In previous work, the Itaipú plant has been modeled with a simpler generator model with parameters $X_d$, H, $R_a$, and $X'_d$ (M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi, 2019). The order of parameter selection was derived from guidelines in (Kundur). The equations used to model the generator are also obtained from (Kundur).

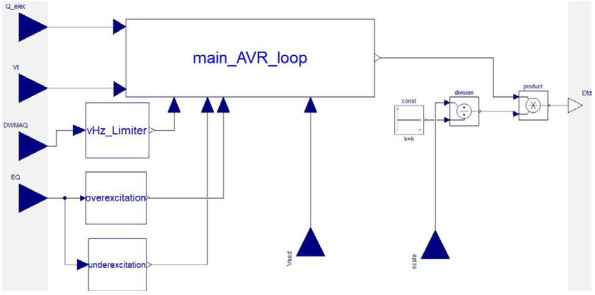


**Figure 4.** User-defined AVR model functions implemented in Modelica.

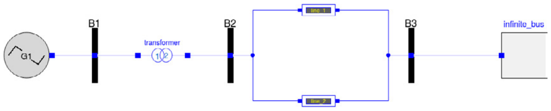


**Figure 5.** SMIB system set up in Dymola. Equivalent system is also implemented in Anatem.

in Dymola. The software CduEdit (Cepel, c), which is pictured in Figure 3, is used to create control system diagrams for components. It is proprietary software used in Brazil in the utility sector for engineers to maintain a database of user-defined controller models for power systems (Cepel, c). It is a graphical interfaces used to create and edit user defined controls (CDUs). The CDUs can be simulated in ANATEM (Cepel, b), which is an industry-specific tool used for electromechanical transient analysis. These softwares also interact with ANAREDE (Cepel, a), which is a program that assists in the analysis of power system networks, such as power flow, network equivalents, and contingency analysis. These tools are limited in analysis capability, so we must re-implement the CDUs for the Itaipu plant in Dymola for parameter calibration.

### 2.1.4 Software-to-software verification with Anatem

The verification will be carried out by simulating the same system in both software packages and comparing the obtained results, with the goal of showing that the models are equivalent in both software programs. This is necessary because it is challenging to prove to the users of domain-specific tools that they can obtain the same results as those tools in Modelica as long as the models are correctly re-implemented. This brings confidence to the models in Modelica that the results are going to be just as good as, if not better than, the domain-specific tool.

The system chosen to be implemented is the single-machine infinite-bus (SMIB) system, due to its simplicity. The SMIB system implemented using OpenIPSL is shown in the Figure 5.

It is important to mention that some parameter conversion need to be carried out. For example, the saturation curves need to be converted from an exponential representation to a polynomial one. In addition, exciter and power system stabilizers were adapted. The per unit parameters from the circuit were converted to ANAREDE in the Anatem software for the power-flow calculation, resulting in the adequate initial guess values to the system.

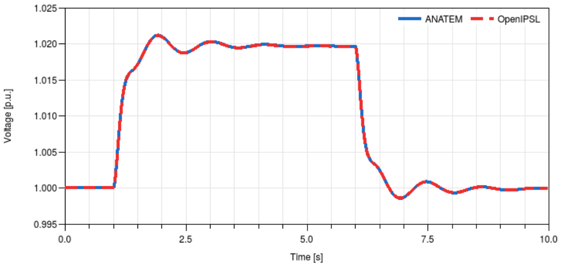


**Figure 6.** OpenIPSL vs Anatem voltages at bus B1 for software validation.

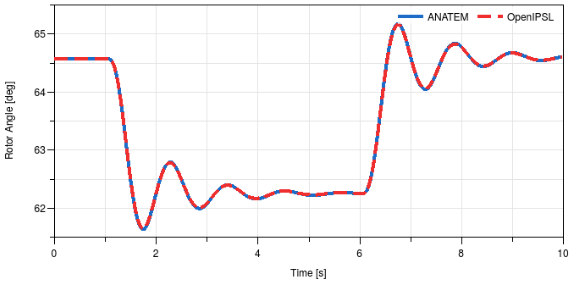


**Figure 7.** OpenIPSL vs Anatem rotor angle at bus B1 for software validation.

The event tested is a step change in the terminal voltage reference for the exciter system. The reference increase by 0.02 at time t=1s and it decreases, also as a step change, back to the original value at instant t=6s. The results from OpenIPSL and ANATEM are displayed in figures for easy comparison. The resultant voltage at bus B1 given by both software packages is displayed in Figure 6. It is possible to observe that the curves from ANATEM and OpenIPSL overlap throughout the entire simulation.

The rotor angle behavior given by both software packages is shown in Figure 7. Again, it is possible to observe the superposition of the results given by both software packages.

### 2.1.5 Using PMU data with Modelica

The voltage data obtained from the PMUs is injected into the system at the site of the infinite bus, shown in Box I in Figure 1. Inside of the 'combiTimeTable' blocks in Block I, the real and imaginary components of the voltage are listed in tables over the period of the system event. Those voltage signals are then converted to flow variables (real and imaginary currents) to be injected at the machine's point of interconnection. Those flow variables then control the generator power output according to the voltage input.

Block A in Figure 1 contains the tables of data for the active and reactive power. The purpose for including these measurements in the model is to observe the fit of the simulation to the measurements in the plotting window.

## 2.2 Preparing models for RaPId

The models were optimized using RaPId (L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, F. Jose Gomez-Lopez, 2016), a MATLAB toolbox used for parameter validation, calibration, and opti-

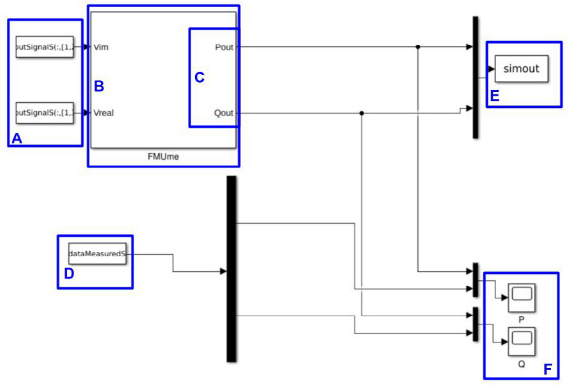


**Figure 8.** Itaipu system in Simulink using the FMI Toolbox for use in RaPId.

mization that uses models exported using FMI standard for Model Exchange, called Functional Mock-up Units (FMUs). RaPId uses both Simulink and MATLAB functions from the FMI Toolbox for MATLAB (Modelon, 2018) to simulate and perform the computations with the model. The FMUs need to be loaded and configured in a Simulink block from the FMI Toolbox library; in this paper, the block for model exchange from the FMI Toolbox was used to simulate the model. A MATLAB script is used to specify measurements, define an optimization problem, and to provide an initial guess of the desired parameters. The complete system used for all experiments discussed in this paper is configured in Simulink is shown in Figure 8, labeled as follows:

A. Input voltage measurements split into a real and imaginary component. Measurements are from PMUs.
B. FMU containing the Modelica model.
C. Output of the FMU (created from system in Figure 1); simulated P and Q.
D. Measurements of P and Q for graphical comparison, used in software for validation.
E. Output P and Q results to the work space. This is updated every iteration.
F. Scopes to monitor the simulated response against the measurements during each simulation run.

These models were calibrated using the toolbox with the methods outlined in Section 3. This was necessary to determine the change in the parameters due to the aging components in the system.

# 3 Parameter Identification and Calibration Methods

The `fmincon` and particle swarm optimization (`PSO`) solvers are used in the parameter estimation experiments for the user defined models. The `PSO` solver is used to find a global solution for the parameters when calibrating each component in the model. Once a global solution is found, `fmincon` is used to optimize the local solution.

In Equation (1), the lower and upper limits to the optimization problem are defined as $p_{min}$ and $p_{max}$.

$$\min f(x) \text{ such that} \left\{ \; p_{min} \leq x \leq p_{max} \; \right\}. \tag{1}$$

During each optimization run using `PSO` and `fmincon`, the estimated parameter vector $\bar{p}$ is continuously optimized and updated to simulate the response of the system. The absolute difference between the simulation and measurements is calculated at each time step, as follows:

$$\varepsilon_1 = \begin{bmatrix} P_{out}^{simulated} - P_{out}^{reference} \\ Q_{out}^{simulated} - Q_{out}^{reference} \end{bmatrix}^T \tag{2}$$

where $P_{out}^{simulated}$ is the output from the simulation of the FMU for the active power, $P_{out}^{reference}$ is the active power measurement from the PMU data; $Q_{out}$ is the reactive power and follows the same process. The objective function is then determined by computing the Frobenius norm from the mismatch $\varepsilon_1$ between the simulation and PMU measurements for active and reactive power of the generator. The sum of mismatches is calculated from the norms of the measurement/simulation pair at each time step, returning the fitness of the simulated model to the measurements. The fitness of the active and reactive power are weighted equally when optimizing the parameters by minimizing:

$$f(x) = \Sigma_{i=1}^{m} \Sigma_{j=1}^{n} (\varepsilon_{ij} * \varepsilon_{ij}) \tag{3}$$

This is repeated for each individual parameter calibrated in this paper listed in the Appendix.

The model shown in Figure 1 was exported from Dymola as a Functional Mock-up Unit (FMU) (Dassault, 2018) to calibrate the models' parameters using MATLAB's optimization solver, `fmincon` (Mathworks), a particle swarm optimization (PSO) solver, and Simulink. The fmincon optimization runs were executed for up to 5000 iterations using an error tolerance of 1 x $10^{-5}$. The PSO optimization runs were executed for up to 200 iterations with an error tolerance of 1 x $10^{-3}$. The parameter values were changed each iteration with the goal of providing the best optimization fit to the reference measurements using RaPId (L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, F. Jose Gomez-Lopez, 2016), a MATLAB toolbox for rapid parameter identification.

The optimization process used in this paper for the comparison with the generic IEEE power system models is described in (M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi, 2019). Both the user-defined and generic models are calibrated using the same process.

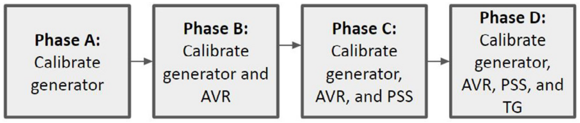

Phase A: Calibrate generator
Phase B: Calibrate generator and AVR
Phase C: Calibrate generator, AVR, and PSS
Phase D: Calibrate generator, AVR, PSS, and TG

**Figure 9.** Process for calibrating user-defined components.

### 3.1 Sequential Parameter Estimation Methodology for User-Defined Models

The optimization process used in this paper is shown in Figure 9. First the generator parameters are calibrated without any of the control system included in the model. All of these parameters are calibrated simultaneously using `PSO`. The results from the `PSO` are then used as the initial guess to calibrate the parameters sequentially. The sequential parameter estimation process follows the method shown in Algorithm 1.

**Algorithm 1** Sequential parameter calibration using heuristic solver, then gradient solver

```
Use PSO algorithm:
x = vector for all parameters to be calibrated
$x_{min}$ = vector of lower limit of all parameters calibrated
$x_{max}$ = vector of upper limit of all parameters calibrated
x = $\min f(x)$ such that $x_{min} \leq x \leq x_{max}$
Use fmincon solver:
p = empty vector
$p_{min}$ = empty vector
$p_{max}$ = empty vector
For parameter in x:
p = p.append(x) % Use results of PSO as starting guess x
$p_{min} = p_{min}$.append($x_{min}$)
$p_{max} = p_{max}$.append($x_{max}$)
$\min f(x)$ such that $p_{min} \leq p \leq p_{max}$
```

This method follows the sequence defined in Figure 9. After calibrating the generator parameters, the final solution of the generator `fmincon` optimization is used as the initial guess of generator parameters for calibrating the AVR. Similarly to the calibration of the generator model, Algorithm 1 is used to calibrate the parameters of the generator and AVR model. A PSO routine is run for all of the parameters in the generator and AVR, then its solution is used as the initial guess for the calibration of the individual AVR parameters with the fmincon solver. This process is then repeated again for the PSS and TG, adding them into the system sequentially.

### 3.2 User-defined models vs. Generic Models

In this paper, two different models for calibration are studied: generic and user-defined. The generic models are re-implemented from the model description provided in IEEE's Recommended Practice for Excitation System Models for Power System Stability Studies (IEEE, 2016). These models in (IEEE, 2016) are also implemented in other power system analysis tools, including the OpenIPSL library. In this study, the GENSAE synchronous generator, SEXS AVR, and STAB3 PSS are used to model the Itaipu system for the generic model studies. The generic TG model is the IEEE HYGOV model (NEPLAN AG, b), which follows the model definition for an IEEE standard turbine-governor in a hydro plant. The control diagrams and equations defined by IEEE to create these models are include in the Appendix.

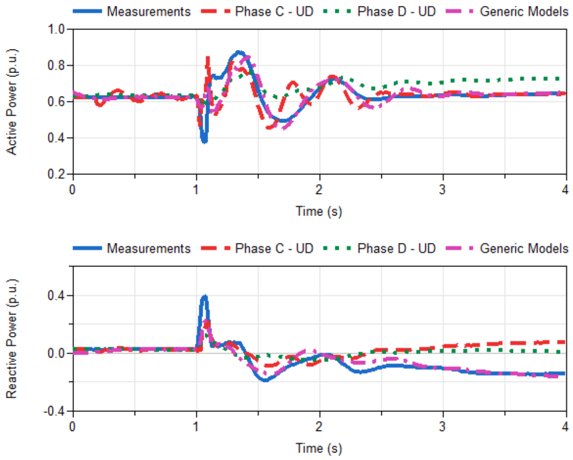


**Figure 10.** Calibration results for User-defined AVR/PSS calibration (red), AVR/PSS/TG (green), and IEEE generator AVR/PSS/TG (purple) for September 22, 2015 event.

The user-defined models used to model the Itaipu system were created by the plant engineers in CDUEdit, which is explained in Section 2.1.3. These models in Dymola are the exact same as the model description in the CDUEdit software. They have the same performance in both ANATEM and in Dymola as shown in Section 2.1.4.

## 4 Case Studies

Data from two different fault events occurring at a generator at Itaipu were used for this model validation and calibration process. The models with calibrated parameters were simulated in Dymola with a variable time step solver, Dassl with a tolerance of $10^{-3}$.

### 4.1 Results - Case 1: September 22, 2015

The model was calibrated using both user-defined components and IEEE generic components (M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi, 2019) for the AVR, PSS, and TG models. Figure 10 shows the results of the AVR and PSS calibration for this data set using both the user-defined and IEEE generic models. The generic models show graphically that they perform better at modeling the fault response than the user-defined models. The Euclidean norm according to Equation 3 of the user-defined models is 1.2; the IEEE generic models have a Euclidean norm of 1.1017. The IEEE generic models have a slightly better fit than the user-defined models. The results for all of the calibration steps are shown in Table 1.

The AVR/PSS calibration shows a better fit than the AVR only calibration according to Figure 11. The accu-

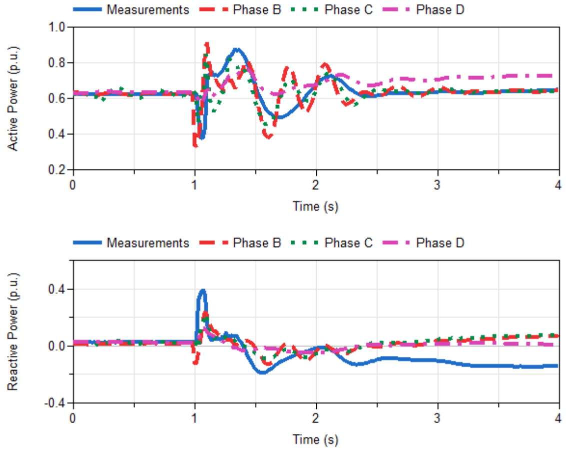


**Figure 11.** All calibration phases for user-defined models compared for September 22, 2015 data set.

**Table 1.** ||x|| fitness of results per model for September 22, 2015 and November 2, 2016 events

| Model Setup/Date | 9/22/2015 | 11/2/2016 |
|---|---|---|
| GENSAE generator | 1.9548 | 1.0884 |
| Generic AVR | 1.5767 | 2.2833 |
| Generic AVR/PSS | 1.1017 | 1.1017 |
| Generic AVR/PSS/TG | 1.3192 | 1.3288 |
| UD AVR | 2.2875 | 2.2827 |
| UD AVR/PSS | 1.2 | 1.1055 |
| UD AVR/PSS/TG | 2.0944 | 5.6683 |

racy of the model decreases when the TG is added to the system, causing distrust in that model. The TG model significantly damps the active power output from the generator and provides too much reactive support. The reference signals, such as reference voltages, were not calibrated in the TG and may need to be adjusted in the future to develop a more accurate model.

### 4.2 Results - Case 2: November 2, 2016

The model was calibrated using both user-defined AVR, PSS, and TG models and IEEE generic AVR and PSS models (M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi, 2019). Figure 12 shows the results of the AVR and PSS calibration for the data set using the generic and user-defined control system models. The user-defined models have a better fit with the measurements than the generic models.

Figure 15 shows the results of the calibration at each step. When the TG is added to the model, the simulation creates a poor fit to the measurements. The TG damps the power output from the generator and causes a large dip in reactive power when the system returns to steady state around 1.5 seconds.

The user-defined TG model has an issue similar to the results of the previous data set where the power from the generator is significantly damped. The addition of the TG in the model causes an over correction of reactive power after the fault, as shown in Figure 15. After the fault, the TG does not let the system return to a steady state; there is a slow oscillation most evidently seen in the reactive power response of the model. When the mechanical power of the generator is used to control the input (shown in Figure 1 with block C removed), the system recovers to near steady state within 1-2 seconds after the fault, as shown in Figure 16. The mechanical turbine power is held constant due to the absence of the turbine-governor.

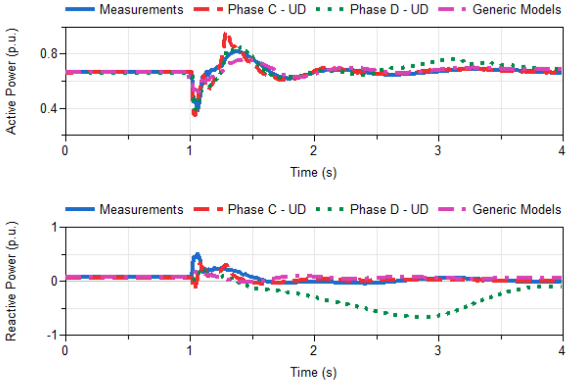


**Figure 12.** Calibration results for User-defined AVR/PSS calibration (red), AVR/PSS/TG (green), and IEEE generator AVR/PSS/TG (purple) for for November 2, 2016 event.

The fitness of the model calibration to the measurements are shown in Table 1. The fitness of the AVR/PSS user-defined model and the generic model are comparable, but the fitness significantly decreases when the TG is included in the model.

### 4.3 Errors with the User-Defined TG Model

The user-defined TG causes such a high error. Figures 13 and 14 shows the results for the power system containing models of the re-implemented user-defined AVR and PSS. The TG was varied between a calibrated model for the user-defined TG and the IEEE HYGOV TG model. The IEEE model is more accurate than the user-defined model in this case, showing that the user-defined TG model has some error that causes the machine to absorb large amounts of reactive power. Figure 14 shows a drastic change between the user-defined Itaipu TG and the IEEE HYGOV TG response. There seems to be an error in the transfer function of the user-defined TG that causes the machine to consume a large amount of reactive power under certain conditions instead of going back to steady-state like the actual system response.

## 5 Discussion

While carrying out this work, it was expected that the user-defined model description would produce a more accurate result than the generic models. The model of the Itaipu

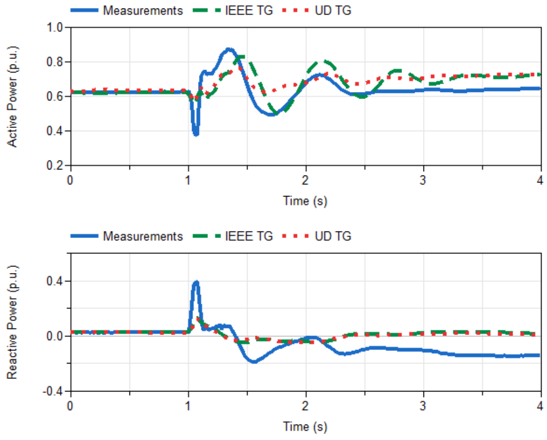


**Figure 13.** Final phases for user-defined AVR and PSS models with both a user-defined and IEEE standard TG model compared for September 22, 2015 data set.

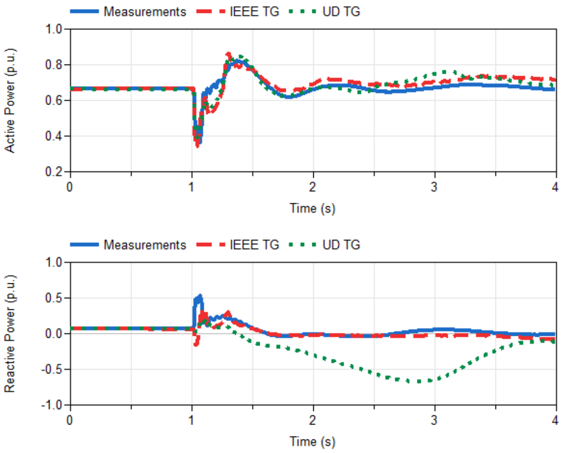


**Figure 14.** Final phases for user-defined AVR and PSS models with both a user-defined and IEEE standard TG model compared for November 2, 2016 data set.

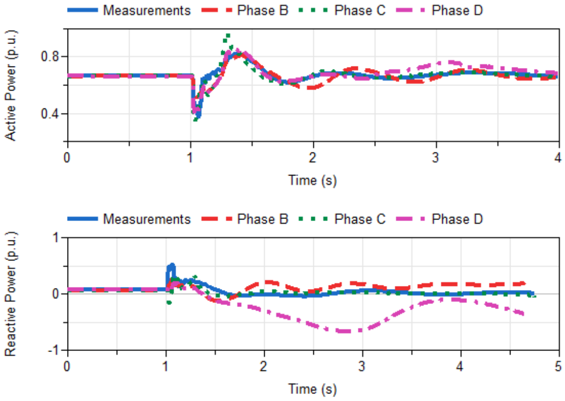


**Figure 15.** All calibration phases for user-defined models compared for November 2, 2016 data set.

plant was most likely originally described in the 1990's. There have been changes in the models since, but they have not updated to be reflected in the model. The user-defined models were created back before the IEEE standard models were originally defined. Although the IEEE standard models are simplified, they have a broader application scope and are able to capture the actual system response and dynamics with better accuracy.

These observations raise the importance of model maintenance and model validation. Thanks to the availability of PMU measurements, this is becoming more possible. This was not possible before, where the only measurements were taken during commissioning tests that do not fully reflect the entire spectrum of the system response.

## 6 Conclusion

The user-defined models did not perform as well as the generic models for the control system of the plant after calibrating the parameters for the two faults analyzed. Although the fitnesses of the two modeling methods are comparable when the AVR and PSS are included in the system, the generic models are consistently more accurate than the user-defined models. In both cases studied, the fitness of the model increases when the user-defined AVR is added to the system from the basic generator only model; however, in the generic case, the models show an improvement in fitness to the PMU measurements after each parameter is calibrated. In the future, the models used for the Itaipu power plant need to be corrected to better to fit the actual response of the plant to the simulation.

These results show that the models currently used for power systems cannot be blindly trusted without the type of analysis shown in this paper. The approximations of the models do not capture all of the behaviors in the physical system, causing distrust in the models. For example, there is a 20 Hz oscillation seen in the measurements that the models cannot replicate, as shown in Figure 17. This implies for both the user-defined and generic models that more detailed representations of certain components need to be developed. This implies that the models need to be revisited to be able to capture the behavior, but also the traditional modeling approach may be insufficient.

## Acknowledgements

This work was funded in whole or part by by the National Aeronautics and Space Administration through the University Leadership Initiative award for the Center for Cryogenic High-Efficiency Electrical Technologies for Aircraft (CHEETA), the New York State Energy Research and Development Agency (NYSERDA) through the Electric Power Transmission and Distribution (EPTD) High Performing Grid Program under agreement number 137940, the Engineering Research Center Program of the National Science Foundation and the Department of Energy under Award EEC-1041877, in part by the CURENT Industry Partnership Program, in part by Dominion En-

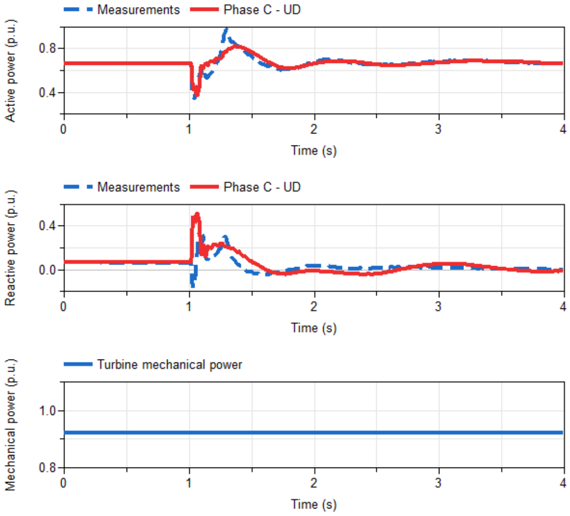


**Figure 16.** Response of the system consisting of generator, AVR, and PSS during the event on November 2, 2016.

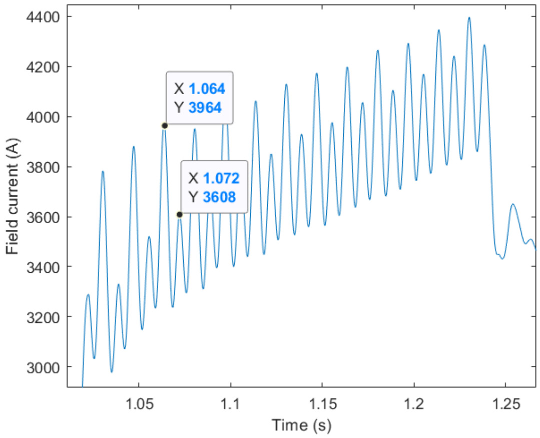


**Figure 17.** 20 Hz oscillation in field current from measurements.

ergy, and in part by the ECSE Department at RPI.

The first author was supported by Rensselaer Polytechnic Institute through the Rensselaer Graduate Fellowship 2019-2020.

## References

Stefan Andersson and Jonatan Strömner. Model calibration of a vertical wind power plant using Dymola/Modelica. 2013.

Cepel. ANAREDE - Network Analysis Program, a.

Cepel. ANATEM - Analysis of Electromechanical Transients, b.

Cepel. CDUEdit - User-Defined Controller Editor, c.

Dassault. Dymola user manual vol 2. 6.10. Technical report, Dassault Systemes AB, March 2018.

IEEE. IEEE Recommended Practice for Excitation System Models for Power System Stability Studies. Technical report, IEEE Power and Energy Society, 2016. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7553421`.

Itaipu Binacional. Generating units.

J. Chen, P. Shrestha, S. Huang, N. D. R. Sarma, J. Adams, D. Obadina, and J. Ballance. Use of synchronized phasor measurements for dynamic stability monitoring and model validation in ERCOT. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, July 2012. doi:10.1109/PESGM.2012.6345152.

P. Kundur. *Power System Stability and Control*. McGraw-Hill.

L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, F. Jose Gomez-Lopez. RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models. 2016. doi:10.1016/j.softx.2016.07.004.

L. Vanfretti, W. Li, T. Bogodorova and P. Panciatici. Unambiguous power system dynamic modeling and simulation using modelica tools. *IEEE Power Energy Society General Meeting*, pages 1–5, 2013. doi:10.1109/PESMG.2013.6672476.

M. Baudette, L. Vanfretti, J. Rabuzin, M. Murad. iTesla Power Systems Library (iPSL): A Modelica Library for Phasor Time-Domain Simulations. doi:10.1016/j.softx.2016.05.001.

M. Podlaski, L. Vanfretti, J. Pesente and P. H. Galassi. Automated parameter identification and calibration for the itaipu power generation system using modelica, fmi, and rapid. *7th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6, 2019. doi:10.1109/MSCPES.2019.8738793.

Mathworks. fmincon Interior-Point Algorithm with Analytic Hessian. Technical report.

Modelon. FMI Toolbox User's Guide 2.6.4. Technical report, Modelon AB, July 23 2018.

NEPLAN AG. EXCITER MODELS: Standard Dynamic Excitation Systems in NEPLAN Power System Analysis Tool. Technical report, NEPLAN AG, a. URL `https://www.neplan.ch/wp-content/uploads/2015/08/Nep_EXCITERS1.pdf`.

NEPLAN AG. TURBINE-GOVERNOR MODELS: Standard Dynamic Turbine-Governor Systems in NEPLAN Power System Analysis Tool. Technical report, NEPLAN AG, b. URL `https://www.neplan.ch/wp-content/uploads/2015/08/Nep_TURBINES_GOV.pdf`.

NEPLAN AG. POWER SYSTEM STABILIZER MODELS: Standard Dynamic Power System Stabilizers in NEPLAN Power System Analysis Tool. Technical report, NEPLAN AG, c. URL `https://www.neplan.ch/wp-content/uploads/2015/08/Nep_PSSs.pdf`.

P. Pourbeik, G. Chown, James Feltes, F. Modau, S. Sterpu, R. Boyer, K. Chan, L. Hannett, D. Leonard, L.T.G. Lima, W. Hofbauer, L. Gerin-Lajoie, S. Patterson, J. Undrill, and F. Langenbacher. Technical report, IEEE, 01 2013.

Z. Huang, P. Du, D. Kosterev, and S. Yang. Generator dynamic model validation and parameter calibration using phasor measurements at the point of connection. *IEEE transactions on power systems*, 28(2):1939–1949, 2013.

# APPENDIX: Variables and Parameters

| Parameter | Details |
|---|---|
| | Generator - IEEE GENSAE Model |
| T'$_{d0}$ | d-axis transient open circuit time constant |
| T"$_{d0}$ | d-axis sub transient open circuit time constant |
| T"$_{q0}$ | q-axis sub transient open circuit time constant |
| H | Inertia constant |
| D | Speed damping |
| X$_{d}$ | d-axis reactance |
| X'$_{d}$ | d-axis transient reactance |
| X"$_{d}$ | d-axis sub transient reactance |
| X"$_{q}$ | q-axis sub transient reactance |
| X$_{q}$ | q-axis reactance |
| X$_{l}$ | leakage reactance |
| | AVR - Itaipu User-Defined Model |
| K$_{v}$ | AVR integrator gain |
| K$_{ei}$ | AVR gain |
| K$_{min}$ | Underexcitation limiter gain |
| K$_{point}$ | AVR gain |
| T$_{i}$ | Overexcitation limiter time constant |
| T$_{a}$ | Overexcitation limiter time constant |
| T$_{b}$ | Overexcitation limiter time constant |
| T$_{ai}$ | Overexcitation limiter time constant |
| | PSS - Itaipu User-Defined Model |
| K$_{f}$ | |
| K$_{f1}$ | AVR time constant |
| T$_{f}$ | AVR gain |
| T$_{p}$ | PSS time constant |
| K$_{1}$ | PSS time constant |
| T$_{1}$ | PSS time constant |
| K$_{2}$ | PSS gain |
| T$_{2}$ | PSS gain |
| | TG - Itaipu User-Defined Model |
| T$_{n}$ | Accelerometer time constant |
| NT$_{v}$ | Adjustment of accelerometer time constant |
| T$_{d}$ | Integrator time constant |
| T$_{f1}$ | Time of closing distributor fast part |
| T$_{f2}$ | Time of closing distributor slow part |
| T$_{v}$ | Equivalent time of distributor valve |
| T$_{w}$ | Water staring time |
| T$_{ya}$ | Time of opening of the distributor |

# APPENDIX: Equations for IEEE standard components

## IEEE HYGOV Turbine Governor

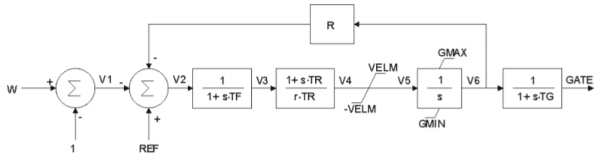


**Figure 18.** Control diagram for IEEE HYGOV TG (NEPLAN AG, b)(Pourbeik et al., 2013)

## GENSAE generator

$$K_{1d} = \frac{(X'_d - X"_d)(X_d - X'_d)}{(X'_d - X_l)^2}$$

$$K_{2d} = \frac{(X'_d - X_l) * (X"_d - X_l)}{(X'_d - X"_d)}$$

$$K_{3d} = \frac{X"_d - X_l}{X'_d - X_l}$$

$$K_{4d} = \frac{X'_d - X"_d}{X'_d - X_l}$$

$$\frac{dE_{pq}}{dt} = \frac{1}{T_{pd0}(E_{fd} - X_{ad}I_{fd})}$$

$$\frac{d\Psi_{kd}}{dt} = \frac{1}{T"_{d0}(E'_q - \Psi_{kd} - X'_d - X_l) * id}$$

$$\frac{d\Psi"_q}{dt} = \frac{1}{T"_{q0}(-\Psi"_q + (X_q - X"_q)) * id}$$

$$\Psi_{d"} = E'_q + K_{3d} + \Psi_{kd}K_{4d}$$

$$\Psi_d = \Psi"_d - X"_d * i_d$$

$$\Psi_q = -\Psi"_q - X"_q * i_q$$

$$X_{ad}I_{fd} = K_{1d} * (E'_q - \Psi_{kd} - (X'_d - X_l) * i_d)$$
$$+ (X_d - X'_d) * i_d + (SE_{exp} + 1) * E'q$$

$$T_e = \Psi_d * i_q - \Psi_q * i_d$$

$$u_d = (-\Psi_q) - R_a * i_d$$

$$u_q = \Psi_d - R_a * i_q$$

## SEXS AVR

$$V_{REF} = E_{fd0}/K + E_{COMP0}$$

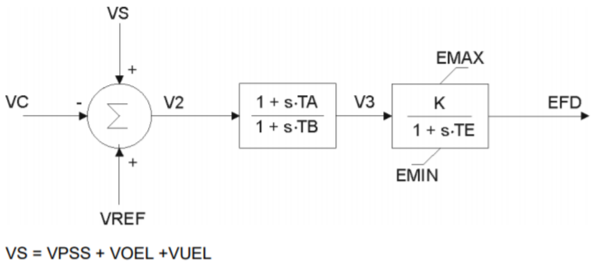


**Figure 19.** Control diagram for SEXS AVR(IEEE, 2016)(NEPLAN AG, a)

## STAB3 PSS

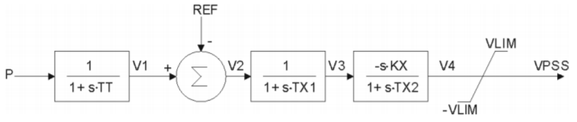


**Figure 20.** Control diagram for STAB3 PSS(IEEE, 2016)(NEPLAN AG, c)

# Application of Model-Based Testing to Dynamic Evaluation of Functional Mockup Units

Cláudio Gomes[1] Romain Franceschini[1,4] Nick Battle[2] Casper Thule[3] Kenneth Lausdahl[3]
Hans Vangheluwe[1] Peter Gorm Larsen[3]

[1]University of Antwerp, Belgium,
{claudio.gomes,romain.franceschini,hans.vangheluwe}@uantwerp.be
[2]Independent, United Kingdom, nick.battle@acm.org
[3]Aarhus University, Denmark, {casper.thule,lausdahl,pgl}@eng.au.dk
[4]University of Corsica, France

## Abstract

A successful co-simulation standard is crucial in applying co-simulation in large scale distributed development processes. A factor that affects the success of a standard is how easily a vendor can implement it. In this paper, we describe an approach to facilitate the implementation of the Functional Mock-up Interface standard. In particular, we propose the use of model-based testing for evaluating the tools that export Functional Mock-up Units (FMUs). This has the benefit that the model used as documentation to describe the possible behaviors of an FMU, can also be used to test it. These principles are embodied in a tool, which is open source, and available online. We then use this tool to evaluate the FMUs available in the FMI Cross-check repository.

*Keywords: model-based testing, functional mock-up interface standard, co-simulation*

## 1 Introduction

Multi-paradigm modeling is a natural response to the challenges posed by the development of complex systems (Vangheluwe; Vangheluwe et al.). These challenges arise not only from essential system complexity (e.g., many interacting, heterogeneous, components), but also from the ensuing development process (e.g., concurrency and distribution) (Tomiyama et al.). Model integration is the means by which multiple models, constructed in different tools and formalisms, can be integrated to answer questions about the system these models represent.

Co-simulation is a technique where the models are integrated through their corresponding simulators (Kübler and Schiehlen, a,b; Gomes et al., f; Hafner and Popper; Palensky et al.). Each simulator, given inputs to the model, is capable of producing outputs, both function over time. Therefore simulators cooperate in producing the overall behavior of the system.

While co-simulation can only be used to answer questions about a system's behavior, it has the advantage that the contents of each model need not to be disclosed, as the model and solver can be encapsulated in a black box. It is therefore a suitable technique to address the challenges arising from concurrent and distributed development processes, where many tools/formalisms might be used and external suppliers may play a role.

Co-simulation standards are crucial enablers. These prescribe the interfaces with which inputs/outputs/parameters can be set/obtained and, optionally, the interaction protocol that each simulator abides to. For example, the Discrete Event System (DEVS) specification that prescribes the integration protocol between simulators (Zeigler; Gomes et al., f; Van Tendeloo and Vangheluwe). On the other hand, the Functional Mock-up Interface (FMI) Standard for co-simulation (FMIv2.0; Blochwitz et al., a,b) prescribes the interfaces, but under-specifies the interaction protocol. In this paper, we focus on the FMI version 2.0. In the FMI terminology, the simulators are referred to as Functional Mock-up Units (FMUs).

Past research (Schweiger et al., a,b), and the co-authors' experience, have shown that there are some ambiguities in the FMI standard (see, e.g., Section 5.2), which lead to not only technical difficulties (e.g., co-simulations crash), but also numerical difficulties (e.g., instabilities. For instance, the second and third most eminent barriers in the adoption of the FMI standard are: "Lack of transparency in features supported by FMI tools" and "insufficient documentation and a lack of examples, tutorials, etc." (Schweiger et al., b). Other somewhat barriers include: "It is difficult to implement FMUs", and "There is a lack of tools that sufficiently support FMI" (Schweiger et al., a). In the same empirical study, the authors identified the most experienced issue to be "Difficulties in practical aspects, like IT-prerequisites in cross-company collaboration."

**Goal.** We aim at supporting the community in rooting out possible ambiguities and improving the conformance to the standard. We propose an approach to the development of an evaluation tool for a co-simulation standard. Ideally, such a tool would never be necessary, as the ideal standard would allow for automated synthesis of co-simulation interfaces. However, we recognize that it is difficult to rigorously specify a standard to the level required by automated synthesis. In particular, we propose the use of Model-Based Testing (MBT) for the evaluation. This has the benefit that the model used to describe the possible behaviors of a simulator can also be used to test it, and can be applied with minimal setup. We describe a tool that embodies these principles and tests co-simulation Functional Mock-up Units (FMUs) exported by tool providers. The tool does not test the numerical performance of an FMU (e.g., numerical error, freedom from stability problems, etc.). This tool is open source and available online with documentation and examples (Gomes et al., 2019). It is our goal that it helps in the development of FMUs and that our approach inspires standardization bodies to adopt the same technique.

**Structure.** In the next section, we survey prior work. Then, in Section 3, we describe our contribution. In Section 4 we describe the application of our contribution to empirically evaluate 169 FMUs, downloaded from the FMI Cross-check repository (Modelica Association, 2019a). Section 5 summarizes the results, lessons learned, and discusses the limitations of our approach. Section 6 concludes.

## 2 Related Work

Our goal is similar to other researchers that have helped identify ambiguities and omissions in the FMI standard.

In particular, the FMI development committee makes available an FMU Compliance Checker on its website (Modelica Association, 2019b). It verifies the consistency of the FMU metadata, and runs a co-simulation with user-defined input data using a fixed communication step size, appropriate to the FMU under test. This tool, along with the FMU-SDK (qTronic GmbH, 2019), available at the FMI web site, play an important role in improving the conformance of FMUs to the FMI standard. Moreover, the work in (Bertsch et al.) focuses on the testing of FMU importing tools, by the use of reference FMUs, build using the FMU-SDK.

Battle et al. has produced a tool that focuses on the conformance of the FMU metadata. It has been applied to the FMI Cross-check repository (Modelica Association, 2019a), and has yielded important lessons:

- Roughly 17% out of 692 FMUs do not follow the rules regarding the InitialUnknowns field of the model structure (FMIv2.0, Section 2.2.8, p. 60).
- About 18% have inconsistencies related to the derivatives declared. For instance, the derivative indexes do not match the set of real scalar variables with a derivative field declared; there are derivatives, but no real scalar variables with a derivative field declared; or there are real scalar variables with derivative field set, but no derivatives declared.

Our work aims at complementing the existing work by modeling all possible interactions with an FMU allowed by the FMI standard. In this sense, it obviously differs from the tool described in Battle et al.. However, our approach also differs from the FMU Compliance Checker, which runs a co-simulation with inputs prescribed by the FMU. While our tool also uses the information declared by the FMU, it may also stop a co-simulation, reset the FMU, initialize some variables while leaving others uninitiated, etc, as long as these operations are allowed by the standard.

Finally, we highlight the DCP-Test-Generator tool, made available in (Leibniz University Hannover, 2019). This tool was created with the same goal as ours, except it targets the Distributed Co-simulation Protocol (DCP) standard (Baumann et al.; Krammer et al., b,a). The DCP standard focuses on Hardware-in-the-loop and real-time co-simulations. It complements the FMI standard by encompassing information that is crucial for communication over a variety of transport protocols. At the heart of the standard is a state machine that dictates how the co-simulation progresses. This state machine was reused to generate test cases, providing anecdotal evidence that our approach has the benefit of leveraging any state machine diagrams used for documentation, leading to an inexpensive way of adopting any standard.

## 3 Model-Based Testing FMUs

In this section, we introduce Model-Based Testing (MBT) and then we explain the language we created to develop the FMU model, used in the MBT process.

### 3.1 Model-Based Testing

We adopt the terminology in Roy Awedikian and define Model-Based Testing (MBT) as the use of a model of the System-Under-Test (SUT) in order to guide test case generation. Such model can be represented as a state machine. Figure 1 shows an example that captures some of the possible interactions with an FMU.

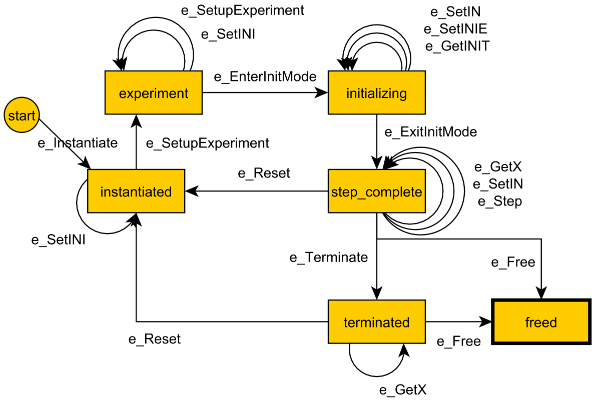


**Figure 1.** Example state transition system – Simplified FMU interaction model. The initial state is called "start", represented as a circle, and the final state is "freed" (outlined rectangle).

Any finite path accepted by the state transition system constitutes a possible test case of the SUT, and each test case must be associated with an oracle that dictates whether the SUT has passed the test or not. For instance, in Figure 1, an accepted path could be:

1. Start-e_Instantiate->instantiated
2. instantiated-e_SetupExperiment->experiment
3. experiment-e_EnterInitMode->initializing
4. initializing-e_ExitInitMode->step_complete
5. step_complete-e_Free->freed

where each edge corresponds to procedures that invokes the corresponding operations on the FMU. Some of these operations are detailed in Section 3 and all are available in the online code (Gomes et al., 2019).

If a test case fails, the MBT tool cleans up allocated resources, records the log of the SUT, and provides enough information to reproduce the test case (e.g., a seed value).

MBT is applied to SUTs as a black box, i.e., affecting only the SUT's inputs, making an ideal technique to test black box FMUs, and there is a large body of research on how to optimize the generation of paths to discover problems quickly (Li et al.; Peleska).

We have extended the Modbat (Artho et al., 2019; Artho et al.) to perform MBT on a finite state machine-based language that we developed. The language uses the GraphML syntax (Brandes et al.), for which the graph editor yEd (yWorks, 2019) can be used.

In the following sub-section, we describe the syntax and semantics of the proposed language.

## 3.2 Simulator Environment Language

We propose a language to describe all possible simulator operations. Our language is largely based on non-deterministic Extended Finite State Machines (EFSM), with constructions that make it easier to deal with large numbers of possible test paths.

Figure 1 shows an example of a traditional EFSM. Albeit a simple example, one can readily see the some of the problems associated with using a traditional EFSM for the description of simulator environments:

- There can be many edges between the same pair of states (e.g., edges from step_complete onto itself);
- There are operations that can be performed in almost all states (e.g., e_Free and e_Reset);
- There are many operations whose execution should not be repeated, or should be repeated a fixed number of times;
- Adding more interactions will quickly make the EFSM unreadable;

As a result, we propose the following extensions that are implemented in our language. These are purely syntactic, as they do not add to the expressive power of EFSMs, and their implementation can be done by reduction to a more complex traditional EFSM.

### 3.2.1 Edge-Or

This extension allows one to compactly represent multiple possible operations in the same edge. Its syntax and example reduction are described in Figure 2.

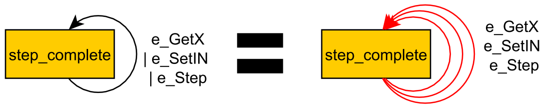


**Figure 2.** Edge-or syntactic extension and example reduction. Each edge-or (left) gets expanded to a traditional EFSM edge (right, in red).

### 3.2.2 State-Or

This extension allows one to compactly represent an operation between many possible states. Its syntax and example reduction are described in Figure 3.

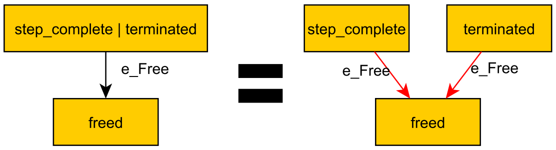


**Figure 3.** State-or syntactic extension and example reduction. Each State-or (left) gets expanded to a state (right), preserving the edges leaving/arriving at that state (in red).

### 3.2.3 Bounded Repetition

This extension allows one to compactly represent a self-loop edge that should be repeated only a finite number of times while in the same state. Its syntax and example reduction are described in Figure 4.

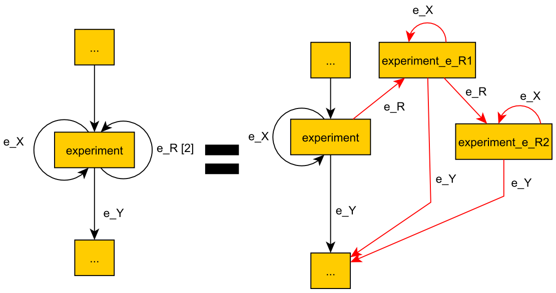


**Figure 4.** Bounded Repetition syntactic extension and example reduction. A bounded repetition edge (left) gets expanded to a counter of the number of times that edge has been executed (right). Notice that upon attaining the maximum number of edge executions, the state machine can only execute other edges.

### 3.2.4 Edge and State Merge

This extension, in combination with the previous ones, allows one to split the specification of the environment into multiple models. The merging of multiple descriptions is done by merging states with the same name, and taking the union of their edges. Figure 5 illustrates this operation. These reductions can be applied until none is applicable.

### 3.2.5 Edge Implementations

Each edge corresponds to a method, implemented in a class, as exemplified in Figure 6.

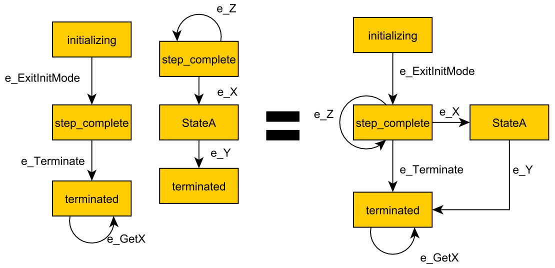


**Figure 5.** Edge and State Merge operation example. the two descriptions on the left get merged into one, on the right.

When setting/getting variables, we followed the definitions in (FMIv2.0, Fig. 11). For example, in Figure 6, the set `INIE` is defined as follows:

```
INIE = vars.filter(v =>
        v.causality==Causality.Input ||
        (v.variability != Variability.Constant
          && v.initial == Initial.Exact))
```

### 3.2.6 Tess Success Criteria

Whether a test passes or not is decided when the code corresponding to an edge is executed: any uncaught exception, including an assertion on a false statement, represents a test failure.

## 3.3 Model-Based Testing Tool for FMUs

The main use case and the structure of our tool are summarized in Figure 7.

# 4 Empirical Evaluation of FMUs

In this section, we describe the application of our tool to the FMUs made available for the FMI Cross-check repository (Modelica Association, 2019a).

## 4.1 Methodology

This goal of this study is to measure how well the FMUs tolerate sequences of operations that are valid with respect to the FMI standard.

The model used to generate test cases was created following (FMIv2.0, Fig. 11), and is partially illustrated in Figure 9. The full model, and edge implementations, can be consulted in the tool's repository (Gomes et al., 2019). Most operations are mapped directly to the FMI Standard operations. The following are the noteworthy exceptions:

- The `e_GetINIT` and `e_GetX` operations select at random one of the variables in the corresponding set `INIT` and `X`.
- The `e_SetINI`, `e_SetIN`, and `e_SetINIE`, operations select at random one of the variables in the corresponding set `INI`, `IN`, and `INIE`. The value to set the variable with is computed either from the declared nominal value, or is equal to 1.
- The `e_SetupExperiment` operation chooses, at random, whether the stop time, equal to 1s, is defined or not.

```
class FMIGraphModel extends ModBatGraphModel {
  // SUT and Time
  var instance: IFmiComponent
  var t = 0
  ...

  // Edge Methods
  def e_Instantiate() = {
    instance = instantiate(fmu, getGuid(fmu))
  }
  def e_SetINIE() = {
    // Pick a random var from the
    //  INIE set of variables
    // Pick a value (e.g., nominal value), and
    //  invoke the corresponding
    //  instance operation.
    setVar(getRandomElement(INIE))
  }
  def e_Terminate() = {
    val s = instance.terminate()
    assert(s == Fmi2Status.OK)
  }
  def e_Step() = {
    // Choose a step size according to
    // FMU Capabilities
    var H = chooseH()
    // Execute the step, and
    // check if the step was carried out
    val res = instance.doStep(t, H, true)
    assert(res == Fmi2Status.OK)
    t = t+H
  }
  def e_Free() = {
    instance.freeInstance()
  }
  ...
}
```

**Figure 6.** Pseudocode exemplifying implementation of some of the operations introduced in Figure 1. The full code is available online (Gomes et al., 2019).

- The `e_Step` operation will either pick a random step size (from the set $\{0.001, 0.01, 0.1\}$) if the FMU supports adaptive step size, otherwise 0.01 will be used.
- The `e_GetFmuState` and `e_SetFmuState` are only executed if the FMU supports the corresponding operations, and the state is set with the most recent state recorded.

Each operation is accompanied by a simple assertion checking whether it ran successfully. The sets of variables used are defined in Figure 8.

Our tool was applied to each FMU sequentially, with the following parameters: • Number of Random Walks=1000; • Self-Loop Execution Limit=10. The later refers only to the self-loops that are not already constrained by the bounded repetition operator. Both parameters are forwarded to Modbat (Artho et al., 2019).

For each FMU, the 1000 tests are executed in sequence. Before the first test is run, the FMU is loaded. Each test creates a new FMU instance and, regardless of the test result, that instance is always freed before the next test begins. After the 1000-th test is concluded, the FMU is

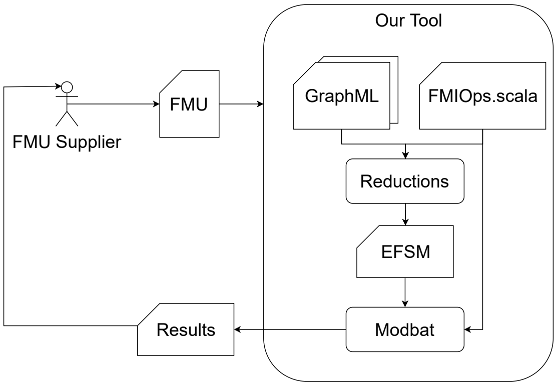


**Figure 7.** Main use case and structure of our tool. The FMU Supplier provides an FMU. The GraphML files, created with the simulator environment language (Section 3.2), along with the implementation of the edges, are part of our tool. These are loaded and transformed into a configuration that Modbat can use to run the MBT.

unloaded. This method avoids any concurrency problems within instances of the same FMU and different FMUs.

Each failed test is logged, along with the output log of the FMU instance and the sequence of operations that were executed from the instantiation until the failure.

Failed tests are grouped into equivalence classes. Two test failures are considered equivalent if they occurred in the same operation on an instance of the same FMU. For example, two tests failures on the `e_Reset` operation of an instance of the same FMU are considered equivalent, even if the first test failure happened just after the instance was instantiated, and the second test failure happened after a step was completed.

To analyze the results, we selected a test from each equivalence class and inspected the logs. The results are summarized in the next sub-section.

## 4.2 Results

The data used to write the results in this section can be obtained by contacting the authors. We do not mention any concrete FMU failures because we recognize that the responsible companies are working to improve their FMUs. Instead, we show aggregate statistics, and present a summary of the failures and the size of the respective equivalence class.

The aggregate statistics are: • FMUs passing all tests=77; • FMUs failing at least one test=55; • FMUs with process crash=37; • Total FMUs tested (sum of previous items)=169; • Total Tests=169000; • Total Failed Tests=14733; • Test Equivalence Classes/Analyzed Failures=102. When a process crash occurs, no other test on the FMU that caused the crash is run. Since this should never happen, we do not count such FMU as failing at least one test even though it did fail one test. Rather, we count it in the "FMUs with process crash" category.

The following is the list of the test failures:

```
INI = vars.filter(v =>
  (v.variability != Variability.Constant &&
    (v.initial == Initial.Approx ||
     v.initial == Initial.Exact)) ||
  (ders.contains(v) &&
    v.`type`.`type` == Types.Real &&
    v.causality == Causality.Input))

IN = vars.filter(v =>
  v.causality==Causality.Input ||
  (v.causality != Causality.Parameter &&
    v.variability == Variability.Tunable))

INIE = vars.filter(v =>
  v.causality==Causality.Input &&
  (v.variability != Variability.Constant &&
    v.initial == Initial.Exact))

INIT = vars.filter(v =>
  v.causality==Causality.Output ||
  ders.contains(v) ||
  derMap.containsValue(v))

X = vars.filter(v => v.causality==Causality.Output)
```

**Figure 8.** Definition of the sets used in the get and set operations. The full code is available online (Gomes et al., 2019).

**Failure 1.** The FMU does not recognize the value reference for a variable declared in its model description. Such a variable is set during initialization mode, and belongs to the set `INI`, as defined in Figure 8.

**Failure 2.** After an FMU is terminated, it fails when a variable belonging to the set `X` (Figure 8) is queried. Recall FMIv2.0, State "terminated", Fig. 11.

**Failure 3.** During stepping mode, a tunable parameter (i.e., a scalar variable with causality="parameter" and variability="tunable") is changed. The FMU then logs a message that it cannot be changed, and returns an error. Recall FMIv2.0, State "step Complete", Fig. 11.

**Failure 4.** The `getRealStatus` (invoked as part of the `e_GetLST` in Figure 9) operation is not supported after an instance is terminated. Recall FMIv2.0, State "terminated", Fig. 11.

**Failure 5.** URI has multiple possible formats for the absolute path of a file, and some FMUs only support one. This causes a failure in the instantiation of the FMU.

**Failure 6.** The outputs are queried after a change in the inputs, without a `doStep` in-between, causing the FMU to return an error. We investigate this issue in Section 5.

**Failure 7.** The `reset` operation is not supported.

**Failure 8.** Some FMUs do not isolate instances in the sense that one failed operation in an instance of an FMU will affect the outcome of other operation calls in a different instance of the same FMU.

**Failure 9.** A variable was set with a value that is outside the scope of an FMU.

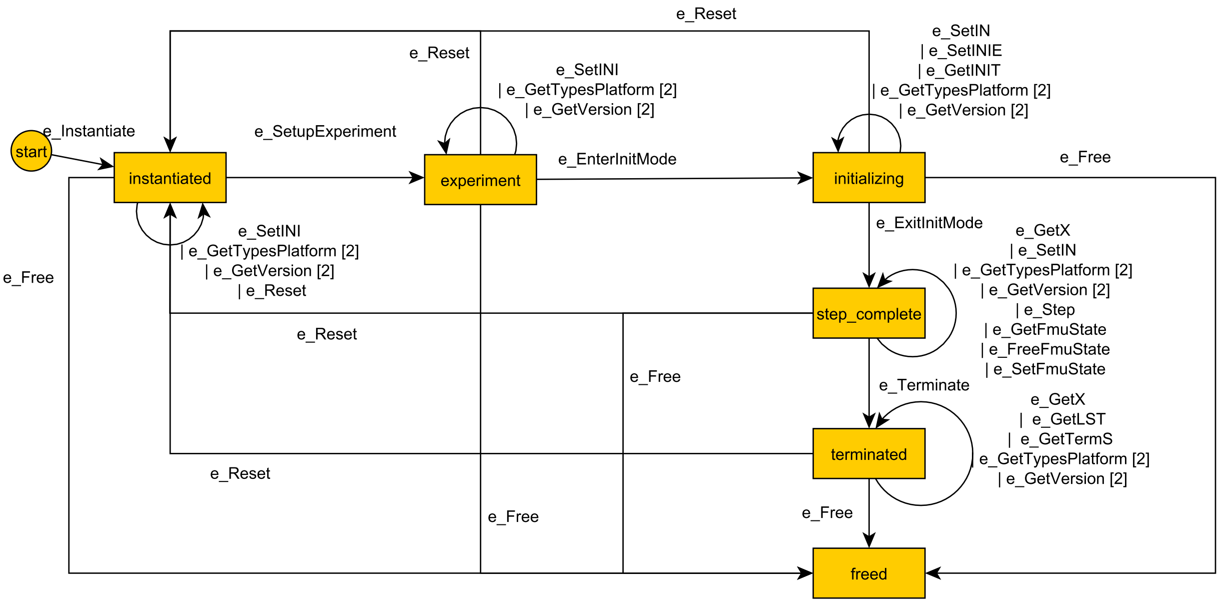


**Figure 9.** Partial model used to generate test cases. We did not consider asynchronous FMUs.

The majority of the process crashes are caused by invocation to the `reset` operation.

Figure 10 summarizes the number of occurrences of each failure, and relates some of them to the failures described above.

# 5 Discussion

In this section, we start by making explicit the limitations of this study, and how to mitigate them. Then, we discuss the lessons learned.

## 5.1 Limitations

An FMU that passes the tests provided in this tool does not constitute a proof that the FMU is correct. Currently, our tool uses the Modbat tool (Artho et al., 2019) to generate random walks in the graph, from start to finish. The later prints coverage information regarding the transitions taken by the model. We chose the number of tests per FMU to be 1000 because it allows us to cover about 95% of the possible choices in the model of Figure 9. Note that the coverage also includes the random choices made inside each edge operation (e.g., the choice of the scalar variable to set), as exemplified in Figure 6, but it depends on the number of scalar variables declared in each FMU. The self-loop limit is chosen to be a small value (10) because our goal is not to run co-simulations to the end. The performance of the tool was not a factor hindering our study. For example, on the example FMU that is shipped with the tool, it takes about 12 seconds to complete these tests.

An FMU failing a test does not necessarily mean that the FMU does not conform to the FMU standard. For example, it is acceptable that an FMU returns an error when the invocation of the `e_Reset` operation is made, but our tools will nevertheless signal a failed test. This enables us to measure the capabilities of the FMUs, which is important in the configuration of co-simulations involving those FMUs.

We have used Scala to implement our tool, and the INTO-CPS FMI library to load and interact with the FMUs. It is possible that some of the failed tests and/or process crashes are caused not by the FMU, but by the library. For the failures analyzed however, we did not find any evidence of this.

The choices made in the implementation of each operation are largely due to the co-authors' experience, and therefore can be a cause for error. Indeed, from the analyzed failures, only one crash was caused by the choice of parameters to run the co-simulation. The size of the equivalence class for this failure is 14 (out of 14733 tests). Whenever possible, we used the model description given by the FMU (e.g., default experiment, nominal/max/min values for variables). However, many FMUs do not provide such information.

Regarding the dataset used, it is worth noting that companies are continuously improving their FMI support and therefore these results will likely become outdated soon. However, the methodology and principles are embodied in an easy to use tool (Gomes et al., 2019).

Regarding the failure analysis, while there is some anecdotal evidence that test failures from the same equivalence class (as defined in Section 4.1) have the same cause, this was not exhaustively checked. Hence, it is possible that some test failures are not being reported here.

Regarding the model used for MBT, detailed in Figure 9, it does not yet cover all permissible operation sequences described in the FMI standard. In particular, we did not consider asynchronous stepping and state serialization, assumed that state get/set operations can only be called while in the stepping mode, and that there's only one setup experiment (hence the `experiment` state). Moreover, a state based model may not be the best representation to specify more complex test scenarios. For example, a sound property of a rollback operation is that, under the same input signals, an instance that has been rolled back should have the exact same behavior as before the rollback. We are convinced that such property is more easily expressed in Linear Temporal Logic, rather than as a state machine.

Finally, we considered the FMI Standard version 2.0, and not the most recent (version 2.0.1), because, at the time of this study, most tools in the FMI cross check repository implement version 2.0.

## 5.2 Lessons Learned and Ambiguities

Despite the above limitations, we have extracted interesting insights and questions from these experiments.

Some FMUs have parameters that refer to numerical properties (such as internal solver step size). Since these are not model parameters, should they be standardized? If not, should the FMU avoid disclosing these parameters and, instead, attempt to infer appropriate values for these parameters, from the information provided by the master algorithm? There is some evidence that practitioners have difficulties configuring co-simulations (Schweiger et al., b), leading us to believe that fewer parameters translates to easier configuration. However, having such parameters exposed can enable the implementation of adaptive master algorithms (Gomes et al., b,a), or automated configuration techniques (Gomes et al., d; Holzinger and Benedikt)

Some FMUs have internal limitations on the values that variables can take, but fail to declare these constraints in their model description. When values on those variables violate these constraints, the FMU causes the co-simulation to fail, and the only way to know what caused the co-simulation to fail is to inspect the logs of the FMUs. Recognizing that there is a trade-of between demanding more standardization in error reporting, and having a wider adoption of such standardization, we recommend that the log messages of the FMUs be made clear regarding validity range violations.

We now focus on failure 6. We believe that such failure is a actually an intended feature of version 2.0 of the FMI standard (FMIv2.0). However, as we show next, this is not so clear, and only one tool reported failure 6. In particular, (FMIv2.0, Page 104) contains "There is the additional restriction in *slaveInitialized* state that it is not allowed to call fmi2GetXXX functions after fmi2SetXXX functions without an fmi2DoStep call in between". However, this is contradicted by the fact that the standard supports *feed-through* dependencies, which induce algebraic dependencies between inputs and outputs. To quote the standard:

- "*output*: The variable value can be used by another model or slave. The *algebraic* relationship to the inputs is defined via the dependencies attribute of `<ModelStructure><Outputs><Unknown>`.", page 45.
- "Attribute dependencies defines the dependencies of the outputs from the knowns [...] at the current Communication Point (CoSimulation).", page 58.

Since even the simplest mechanical systems, such as mass-spring-dampers, when coupled with a power bond in a co-simulation, exhibit such feed-through effect (Gomes et al., e), it is not clear that failure 6 is a failure of the standard.

In order to investigate how this ambiguity is being interpreted in practice, we devised a simple test, that follows the formal definition of feed-through: the input $u_c$ feeds through to output $y_c$ when there exist two different values $v_1, v_2$ and FMU state $s_c$ that lead to two different output values (Gomes et al., c):

$$\mathtt{get}_c(\mathtt{set}_c(s_c, u_c, v_1), y_c) \neq \mathtt{get}_c(\mathtt{set}_c(s_c, u_c, v_2), y_c). \tag{1}$$

The test tries to find two values for which the above holds. If such values are found, the FMU is said to implement feed-through.

Out of 113 FMUs, 7 implement feed-through as defined in Equation (1). Moreover, from the authors' experience with the tool OpenModelica (Open Source Modelica Consortium, 2019), FMUs from the latest version (1.13.2) implement feed-through, whereas FMUs from the versions 1.12.X do not, further highlighting the uncertainty surrounding this feature.

Acknowledging that different versions of the same tool have different degrees of support for the FMI standard, we argue that the tools listed in FMI standard website should disclose which version is being used in the FMI Crosscheck. This helps users to determine to which degree their version supports the standard.

Finally, we remark that, in the definition of the sets of variables `INI`, `IN`, `INIE`, `INIT`, and `X` (recall Figure 8 and (FMIv2.0, Figure 11)), the authors had difficulties understanding whether the `Set*` operations cannot be made on variables whose causality is output, and whether the `Get*` operation cannot be made on variables whose causality is input. For instance, the definition of `INIE`, is "any variable with variability ≠ 'constant' and with initial='exact'" (FMIv2.0, Figure 11), and does not explicitly state that causality='input' should be the case. Instead, elsewhere in the same page, "[In Initialization Mode] Variables with initial = 'exact' , as well as variables with variability = 'input' can be set" (FMIv2.0, Page 103). We suspect that failures 1 and 2 might be caused by misinterpretation of the definition of these sets.

# 6 Summary and Future Work

With the goal of rooting out possible ambiguities and improving the conformance to the FMI standard, we describe an approach to apply MBT to evaluate FMUs. We have contributed with a state machine based language that can be used to describe test cases for FMUs, and we package it in an easy to use tool, available online (Gomes et al., 2019). We then used this tool to empirically evaluate existing FMUs and discussed the results. Our approach is easy to apply in existing and subsequent version of co-simulation standards. For the test failures observed, we are currently contacting the companies responsible, with the necessary information to ensure they can reproduce the failure.

In the future, we intend to extend the state machine that is used to generate tests. In particular, we will consider FMU soundness properties such as rollback and reset contracts, and step rejection (Broman et al.) and tunable parameter properties. Our language might have to be extended to specify such soundness properties. One possible extension is to implement linear temporal logic monitors. This will separate the property specification from the system model, and the former should guide the model-based testing of the latter.

## Acknowledgments

We are grateful to the Research Foundation - Flanders (File Number 1S06316N) and to the Poul Due Jensen Foundation, for the financial support. Additionally, we thank the developers of the INTO-CPS Library and Modbat tools, on which our contribution builds upon. Finally, we thank Martin Krammer, Martin Benedikt, and the reviewers, for the information on the related work and throughout feedback on this work.

## References

Cyrille Valentin Artho, Armin Biere, Masami Hagiya, Eric Platon, Martina Seidl, Yoshinori Tanabe, and Mitsuharu Yamamoto. Modbat: A Model-Based API Tester for Event-Driven Systems. In *Hardware and Software: Verification and Testing*, volume 8244, pages 112–128. Springer International Publishing. ISBN 978-3-319-03076-0 978-3-319-03077-7. doi:10.1007/978-3-319-03077-7_8.

Cyrille Valentin Artho, Armin Biere, Masami Hagiya, Eric Platon, Martina Seidl, Yoshinori Tanabe, and Mitsuharu Yamamoto. Modbat repository, 2019. `https://github.com/cyrille-artho/modbat`, accessed 25th September 2019.

Nick Battle, Casper Thule, Cláudio Gomes, Hugo Daniel Macedo, and Peter Gorm Larsen. Towards a Static Check of FMUs in VDM-SL. In *17th Overture Workshop*, page to be published.

Peter Baumann, Martin Krammer, Mario Driussi, Lars Mikelsons, Josef Zehetner, Werner Mair, and Dieter Schramm. Using the distributed co-simulation protocol for a mixed real-virtual prototype. In *2019 IEEE International Conference on Mechatronics (ICM)*, volume 1, pages 440–445. IEEE Industrial Electronics Society.

Christian Bertsch, Awad Mukbil, and Andreas Junghanns. Improving Interoperability of FMI-supporting Tools with Reference FMUs. pages 533–540. doi:10.3384/ecp17132533.

Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International Modelica Conference*, pages 173–184. Linköping University Electronic Press, a. doi:10.3384/ecp12076173.

Torsten Blochwitz, Martin Otter, Martin Arnold, C. Bausch, Christoph Clauss, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, M. Monteiro, T. Neidhold, Dietmar Neumerkel, Hans Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th International Modelica Conference*, pages 105–114. Linköping University Electronic Press; Linköpings universitet, b. doi:10.3384/ecp11063105.

Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. *Graph Markup Language (GraphML)*.

David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of FMUs for co-simulation. In *Eleventh ACM International Conference on Embedded Software*, page Article No. 2. IEEE Press Piscataway, NJ, USA. ISBN 978-1-4799-1443-2.

FMIv2.0. Functional Mock-up Interface for Model Exchange and Co-Simulation. URL `https://fmi-standard.org/downloads/`.

Cláudio Gomes, Romain Franceschini, Nick Battle, Casper Thule, Kenneth Lausdahl, Hans Vangheluwe, and Peter Gorm Larsen. FMIMOBSTER repository, 2019. `https://msdl.uantwerpen.be/git/claudio/FMIMOBSTER`, accessed 25th September 2019.

Cláudio Gomes, Benoît Legat, Raphaël Jungers, and Hans Vangheluwe. Minimally Constrained Stable Switched Systems and Application to Co-simulation. In *IEEE Conference on Decision and Control*, pages 5676–5681, a. doi:10.1109/CDC.2018.8619223.

Cláudio Gomes, Benoît Legat, Raphaël M. Jungers, and Hans Vangheluwe. Stable Adaptive Co-simulation: A Switched Systems Approach. In *IUTAM Symposium on Co-Simulation and Solver Coupling*, volume 35, pages 81–97. Springer, Cham, b. doi:10.1007/978-3-030-14883-6_5.

Cláudio Gomes, Levi Lucio, and Hans Vangheluwe. Semantics of Co-simulation Algorithms with Simulator Contracts. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 784–789. IEEE, c. doi:10.1109/MODELS-C.2019.00124.

Cláudio Gomes, Bentley James Oakes, Mehrdad Moradi, Alejandro Torres Gamiz, Juan Carlos Mendo, Stefan Dutre, Joachim Denil, and Hans Vangheluwe. HintCO - Hint-Based Configuration of Co-Simulations. In *International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 57–68, d. ISBN 978-989-758-381-0. doi:10.5220/0007830000570068.

Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art, e. URL http://arxiv.org/abs/1702.00686.

Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: A Survey. 51(3):Article 49, f. doi:10.1145/3179993.

Irene Hafner and Niki Popper. On the terminology and structuring of co-simulation methods. In Dirk Zimmer and Bernhard Bachmann, editors, *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 67–76. ACM Press. ISBN 978-1-4503-6373-0. doi:10.1145/3158191.3158203.

Franz Holzinger and Martin Benedikt. Optimal Trigger Sequence for Non-iterative Co-simulation:. In *Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 80–87. SCITEPRESS - Science and Technology Publications. ISBN 978-989-758-381-0. doi:10.5220/0007833800800087.

R. Kübler and W. Schiehlen. Two Methods of Simulator Coupling. 6(2):93–113, a. ISSN 1387-3954. doi:10.1076/1387-3954(200006)6:2;1-M;FT093.

R. Kübler and W. Schiehlen. Modular Simulation in Multibody System Dynamics. 4(2-3):107–127, b. ISSN 1384-5640. doi:10.1023/A:1009810318420.

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*, page No. 1. Society for Computer Simulation International, a. doi:10.22360/summersim.2018.scsc.001.

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*, volume 157, pages 87–96. Modelica Association, b. doi:10.3384/ecp1915787.

Leibniz University Hannover. Dcp-test-generator repository, 2019. https://github.com/modelica/DCPTestGenerator, accessed 25th September 2019.

Wenbin Li, Franck Le Gall, and Naum Spaseski. A Survey on Model-Based Testing Tools for Test Case Generation. In Vladimir Itsykson, Andre Scedrov, and Victor Zakharov, editors, *Tools and Methods of Program Analysis*, volume 779, pages 77–89. Springer International Publishing. ISBN 978-3-319-71733-3 978-3-319-71734-0. doi:10.1007/978-3-319-71734-0_7.

Modelica Association. FMI cross-check repository, 2019a. https://github.com/modelica/fmi-cross-check/tree/master/fmus/2.0/, accessed 25th September 2019.

Modelica Association. FMI standard website, 2019b. https://fmi-standard.org, accessed 25th September 2019.

Open Source Modelica Consortium. Openmodelica, 2019. https://openmodelica.org/, accessed 25th September 2019.

Peter Palensky, Arjen A. Van Der Meer, Claudio David Lopez, Arun Joseph, and Kaikai Pan. Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling. 11(1):34–50. ISSN 1932-4529. doi:10.1109/MIE.2016.2639825.

Jan Peleska. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. 111:3–28. ISSN 2075-2180. doi:10.4204/EPTCS.111.1.

qTronic GmbH. FMU software development kit, 2019. https://github.com/qtronic/fmusdk, accessed 25th September 2019.

Bernard Yannou Roy Awedikian. *Practical Model-Based Testing*. ISBN 978-0-12-372501-1.

Gerald Schweiger, Cláudio Gomes, Georg Engel, Irene Hafner, Josef Schoeggl, Alfred Posch, and Thierry Nouidui. Functional Mock-up Interface: An empirical survey identifies research challenges and current barriers. In *Proceedings of the American Modelica Conference*, pages 138–146. Linköping University Electronic Press, Linköpings Universitet, a. ISBN 978-91-7685-148-7. doi:10.3384/ecp18154138.

Gerald Schweiger, Cláudio Gomes, Georg Engel, Irene Hafner, Josef-Peter Schoeggl, Alfred Posch, and Thierry Nouidui. An empirical survey on co-simulation: Promising standards, challenges and research needs. 95:148–163, b. ISSN 1569190X. doi:10.1016/j.simpat.2019.05.001.

T. Tomiyama, V. D'Amelio, J. Urbanic, and W. ElMaraghy. Complexity of Multi-Disciplinary Design. 56(1):185–188. ISSN 00078506. doi:10.1016/j.cirp.2007.05.044.

Yentl Van Tendeloo and Hans Vangheluwe. An Introduction to Classic DEVS.

Hans Vangheluwe. Foundations of Modelling and Simulation of Complex Systems. 10. doi:10.14279/tuj.eceasst.10.162.148.

Hans Vangheluwe, Juan De Lara, and Pieter J. Mosterman. An introduction to multi-paradigm modelling and simulation. In *Proceedings of the AI, Simulation and Planning in High Autonomy Systems Conference*, pages 9–20. Society for Computer Simulation International.

yWorks. yed graph editor website, 2019. https://www.yworks.com/products/yed, accessed 25th September 2019.

Bernard P. Zeigler. *Theory of Modelling and Simulation*. New York, Wiley. ISBN 0-471-98152-4.

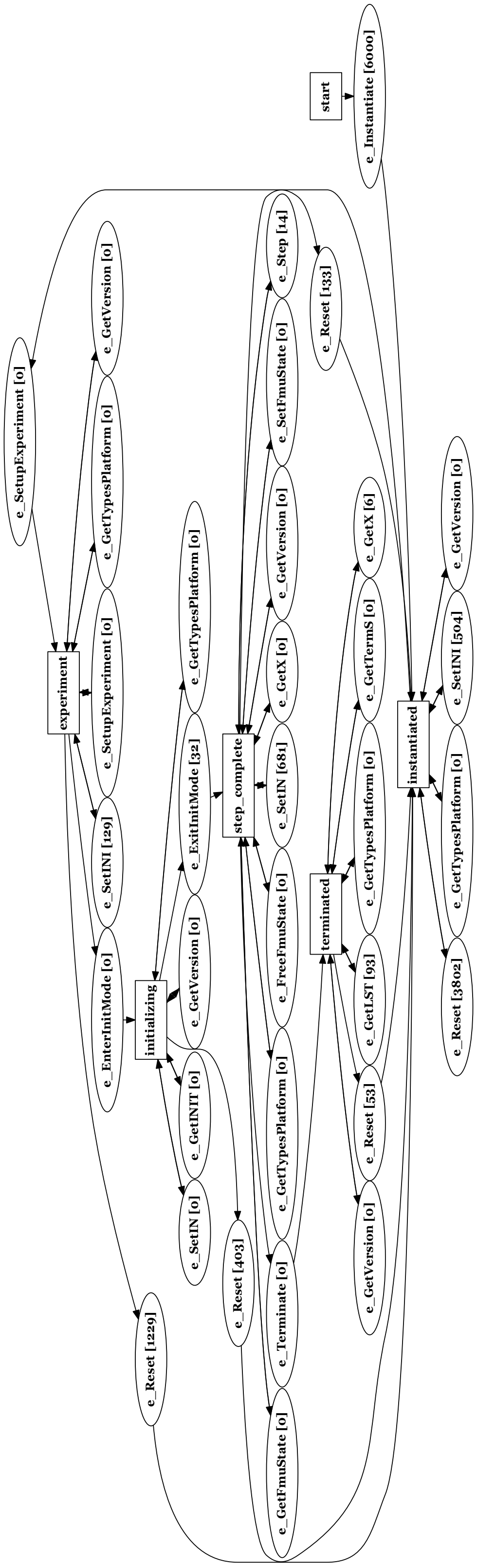


**Figure 10.** Number of occurrences of ea ch failure. Each failure leads to a trace. All traces were joined and counts were taken of the edges that caused the failure. The size equivalence class of each failure can be computed by summing the occurrences for each edge corresponding to an operation. Many tests failed at instantiation because of failure 5. Some of the test failures when entering and exiting initialization mode are due to failure 9.

# Contributions to the Efficient and Parallel Jacobian Evaluation and its Application in OpenModelica

Willi Braun[1] Martin Schroschk[2] Vitalij Ruge[3] Andreas Heuermann[1] Bernhard Bachmann[1]

[1]University of Applied Sciences Bielefeld, Germany, w-braun@posteo.de,
{bernhard.bachmann,andreas.heuermann}@fh-bielefeld.de
[2]Center for Information Services and High Performance Computing, TU Dresden, Germany
martin.schroschk@tu-dresden.de
[3]Siemens AG, Energy Sector, Erlangen, Germany vitalij.ruge@siemens.com

## Abstract

Many algorithms related to Modelica-based simulations heavily rely on the efficient provision of Jacobian matrices. Besides the accuracy of the derivative information, the performance of the derivative evaluation is also of great interest, since it can have a large share in the total simulation time. In this paper, we propose two complementary approaches basing on identification of constant parts and parallelization to accelerate Jacobian evaluation. Furthermore, the implementations of these techniques in the open-source Modelica tool OpenModelica are discussed. The gained speedup in Jacobian evaluation is demonstrated on benchmark models of the ScalableTestSuite.

*Keywords: Jacobian Evaluation, Symbolic Differentiation, Derivatives Computation, Coloring, Sparsity, Parallelization, Modelica, OpenModelica*

## 1 Introduction

Solving computational problems numerically often rely on the access of derivative information, e.g. Jacobian matrices. In the Modelica context this includes algorithms used for solving algebraic loops, implicit integration methods as well as optimization algorithms. Both the speed of the derivatives generation and their accuracy can have a significant impact on such algorithms w.r.t. runtime and robustness. From a mathematical point of view there are several techniques to provide derivatives. A very common numerical method are finite differences, which approximate derivatives by difference quotients. For implicit integration methods with stepsize control, like DASSL or Sundials/IDA, the accuracy of the Jacobian is subordinated, since it is dominated by the current stepsize. The numerical approach using finite differences is very reasonable for this class of algorithms and therefore default in OpenModelica. So accuracy is not an argument for the contrary symbolic differentiation (cf. (Braun, Gallardo Yances, Link, and Bachmann 2012)) in case of implicit integration. But performance might be. The computation time of the Jacobian matrices can have a major share in the total simulation time of a model. To emphasize this statement we consider the model DistributionSystemLinear from the ScalableTestSuite (Casella 2015): In Table 1 the execution time for the simulation and Jacobian evaluation using the symbolical derivative module of OpenModelica, as well as the total number of Jacobian evaluations over the simulation time is stated for three different model sizes $N$. The timings were obtained on an Intel Xeon E5-2680 v3 processor using OpenModelica in version 1.12. Since these models contain one large linear system, which makes the compression futile, the Jacobian evaluation takes a significant amount of the total simulation runtime. Therefore, the right hand side for the Jacobian has to be evaluated significantly more often. Accelerating the evaluation of Jacobians would lead to a sizeable reduction of the total simulation runtime. Furthermore, the Jacobian evaluation does not make use of the available parallel hardware resources today's multi-core processors, like the used Intel Haswell processor, offer. This example serves as a representative for models having evaluation expensive right hand sides which are dominating the large algebraic loop.

**Table 1.** Timings for overall simulation $t_{total}$ and Jacobian evaluation $t_{jac}$ for sequential execution and the total number of Jacobian evaluations over a simulation run for three different model sizes $N$.

| $N$ | Dim. of J | Evals. of J | $t_{total}$ [s] | $t_{jac}$ [s] |
|---|---|---|---|---|
| 14 | $196 \times 196$ | 15 | 3.8 | 2.7 |
| 20 | $400 \times 400$ | 14 | 12.7 | 10.3 |
| 28 | $784 \times 784$ | 15 | 47.3 | 41.5 |

In previous work, it was shown how directional derivatives are computed in OpenModelica within the symbolic derivative module (Braun, Ochel, and Bachmann 2011). These techniques were combined with coloring approaches in (Braun, Gallardo Yances, Link, and Bachmann 2012) and become extensively used by other algorithms inside of the OpenModelica Compiler as for algebraic loops, optimization and FMI (cf. (Braun and Bachmann 2014), (Ruge and Bachmann 2014), (Åkesson, Braun, Lindholm, and Bachmann 2012), (Franke, Walther, Worschech, Braun, and Bachmann

2015)). Algorithms for optimization and solving algebraic loops strongly rely on fast calculation of accurate derivatives. Therefore, the focus of this paper is on efficient evaluation of directional derivatives necessary for evaluating symbolical Jacobians. We present two complementary techniques to improve the performance of Jacobian evaluation and apply them to the OpenModelica Compiler. The first technique identifies recurring calculations with constant parts in the directional derivatives. These constant parts need to be evaluated only once and the results will be reused in the construction of the Jacobian matrix. The second approach introduces a parallelization approach for the Jacobian evaluation and sketches the implementation in the C simulation runtime of OpenModelica.

The paper is organized as follows: First, the context, generation and evaluation of Jacobian matrices in Modelica compilers is described in Section 2. Based on that two complementary techniques to accelerate the Jacobian evaluation are presented in Section 3. In Section 4 the proposed algorithms and improvements are evaluated using Modelica models. Finally, the presented work is summarized and an outlook for further improvements is given.

## 2 Jacobians in Modelica Models

A Modelica model is typically translated to a basic mathematical representation of differential and algebraic equations (DAEs) and transformed to ordinary differential equations (ODEs) before being able to simulate the model. The result of the so-called flattening process is the equation system

$$\begin{aligned} F(x(t),\dot{x}(t),y(t),u(t),p,t) = 0, \quad t \in [t_0,t_f] \\ x(t_0) = x_0, \end{aligned} \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$ are the potential states, $\dot{x}(t) \in \mathbb{R}^{n_x}$ are the potential state derivatives, $y(t) \in \mathbb{R}^{n_y}$ are the algebraic variables and $u(t) \in \mathbb{R}^{n_u}$ are the inputs. The simulation time is given by $t \in [t_0,t_f]$. For simplicity, the initial conditions of the DAE states at start time $t_0$ are given by $x_0$. Introducing $z = (\dot{x}\ y)$, denoting the unknown variables, and $v = (x\ u\ p)$, denoting the known variables, the DAE can be re-written as

$$F(z,v,t) = 0. \tag{2}$$

Next, the causalization process to get an ordering of the unknown variables $z(t)$ is applied, which enables to solve them sequentially

$$z = G(v,t) \in \mathbb{R}^{n_x+n_y}. \tag{3}$$

The general form of the causalized system consists of a sequence of $k$ assignment statements including implicit systems of equations. These assignments and so-called algebraic loops can be stated as

$$0 = g_i(z_i,z_1,z_2,\ldots,z_{i-1},v,t) \in \mathbb{R}^{n_i}, \quad i = 1,\ldots,k, \tag{4}$$

with

$$(z_1,\ldots,z_k) := z, \quad z_i \in \mathbb{R}^{n_i} \quad \text{and} \quad \sum_{i=1}^{k} n_i = n_x + n_y.$$

Each function $g_i$ assigns values to $z_i$ by utilizing previously computed values for $z_1,\ldots,z_{i-1}$.

The Jacobian of a function of vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m, x \mapsto f(x)$ is defined as

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \tag{5}$$

An important tool when computing Jacobians are directional derivatives. The directional derivative of a vector valued function $f(x)$ is defined by

$$df = \frac{\partial f}{\partial x} \cdot dx, \tag{6}$$

where $dx \in \mathbb{R}^n$ represents the direction in which the directional derivative $df \in \mathbb{R}^m$ is evaluated. The vector $dx$ is also referred to as a *seed vector*. In the following, directional derivatives will be used extensively to construct Jacobians. A straight forward, although naive, approach to construct a Jacobian from directional derivative evaluations is as follows: Using the identity matrix $I \in \mathbb{R}^{n\times n}$, and the unit vectors $e_1,\ldots,e_n \in \mathbb{R}^n$ it holds

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial x} \cdot I \\ &= \frac{\partial f}{\partial x} \cdot \begin{pmatrix} e_1 & \ldots & e_n \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial f}{\partial x} \cdot e_1 & \ldots & \frac{\partial f}{\partial x} \cdot e_n \end{pmatrix}. \end{aligned} \tag{7}$$

Thus, a Jacobian with $n$ columns may be constructed from $n$ evaluations of directional derivatives along the $n$ unit vectors. Applying the concept of directional derivatives on the DAE (2) yields the relation

$$\frac{\partial F}{\partial z} dz + \frac{\partial F}{\partial v} dv = 0, \tag{8}$$

where $dv$ is the input seed vector and $dz$ works as the directional derivative of the relation (3) with respect to the direction $dv$. By solving the system of equations (8) for a particular seed vector $dv$, the directional derivative of the DAE is obtained. Technically this system of equations is represented in OpenModelica as a symbolic equation system, like the original equation system (3). This system contains the desired partial derivatives $dz$ as unknowns, the seed vector $dv$ and all other variables from the original system are considered as known and needs to be transformed like the original system into an explicit form. This can be achieved by applying so-called block lower triangular algorithms resulting in

$$dz = -\left[\frac{\partial F}{\partial z}\right]^{-1} \cdot \frac{\partial F}{\partial v} \cdot dv =: H(z,v,t) \cdot dv. \tag{9}$$

This form can be further passed to the code generation to enable the evaluation of the directional derivative at simulation time. It is important to note that the system of equations (9) is linear in the unknown variables $dz$ and the application of linear solvers is sufficient.

# 3 Accelerate Jacobian Evaluation

In this section we present two different but not incompatible approaches to accelerate the Jacobian evaluation to motivate the use of the symbolic differentiation module of OpenModelica for simulation and optimization.

## 3.1 Reuse of Constant Parts

Analyzing system (9), it is obvious, that the calculation of $H(z,v,t)$ is independent from the seed vector. If no algebraic loop is involved, the calculation is symbolically realized sequentially as

$$A = H(z,v,t) \tag{10}$$

$$dz = A \cdot dv\,. \tag{11}$$

Therefore, the calculation of equation (10) can be evaluated once for each Jacobian matrix construction. The implementation within OpenModelica is realised on expression level and basing on the `wrapFunctionCall` module to identify time-consuming function calls as well as common subexpressions and separate them inside the code generation phase of the OpenModelica Compiler backend. In order to emphasize the approach Example 3.1.1 is outlined in the following.

**Example 3.1.1.** A simple Modelica model with some sort of expensive function $foo$ is considered:

```
model reuseConstantParts
  Real x_1(start=1,fixed=true);
  Real x_2(start=0,fixed=true);
  Real y_1, y_2;
equation
  y_1 = sin(x_1)*foo(x_2);
  y_2 = sin(x_1)*cos(y_1);
  der(x_1) = y_1*y_2;
  der(x_2) = y_1 + y_2;
end reuseConstantParts;
```

Let $bar$ be the derivative of $foo$ and $dv$ the seed vector with respect to $v := (x_1, x_2)$. Then, differentiation will lead to

$$\frac{\partial\, y_1}{\partial v_i} = \cos(x_1)\cdot dv_1 \cdot foo(x_2) + \sin(x_1)\cdot bar(x_2)\cdot dv_2$$

$$\frac{\partial\, y_2}{\partial v_i} = \cos(x_1)\cdot dv_1 \cdot \cos(y_1) - \sin(x_1)\cdot \sin(y_1)\cdot \frac{\partial\, y_1}{\partial v_i}$$

$$\frac{\partial\, der(x_1)}{\partial v_i} = \frac{\partial\, y_1}{\partial v_i}\cdot y_2 + \frac{\partial\, y_2}{\partial v_i}\cdot y_1$$

$$\frac{\partial\, der(x_2)}{\partial v_i} = \frac{\partial\, y_1}{\partial v_i} + \frac{\partial\, y_2}{\partial v_i}.$$

OpenModelica will generate the additional common subexpressions `cse4=cos(x_1)`, `cse5=bar(x_2)`, and `cse6=sin(y_1)` independent of the seed vectors to be reused. Those are needed for the Jacobian evaluation but not the ODE-evaluation and only computed once for each integrator step and reused for each seed vector. The subexpressions `cse1` and `cse2` are determined during the ODE-evaluation, and will be reused for the Jacobian evalution. Finally, OpenModelica will generate C source code for the Jacobian evaluation similar to:

```
/* Constant equations */
cse4 = cos(x_1);
cse5 = bar(x_2);
cse6 = sin(y_1);
/* Dynamic equations */
y1.pDER = cse4*dv_1*cse2 + cse1*cse5*dv_2;
y2.pDER = cse4*dv_1*cse3 - cse1*cse6*y1.pDER;
der(x1).pDER = y1.pDER*y2 + y1*y2.pDER;
der(x2).pDER = y1.pDER + y2.pDER;
```

Furthermore, if the equation system (3) contains (non-) linear algebraic loops, an additional optimization of the Jacobian matrix generation can be achieved as follows: Implicit equation systems of the form

$$g_i(z_i, z_1, \ldots, z_{i-1}, v, t) = 0 \tag{12}$$

are differentiated straightforwardly equation by equation in order to compute the directional derivative $dz_i$. This yields into

$$\frac{\partial g_i}{\partial z_i} dz_i + \sum_{k=1}^{i-1} \frac{\partial g_i}{\partial z_k} dz_k + \frac{\partial g_i}{\partial v} dv = 0 \tag{13}$$

and can be solved as a linear system

$$dz_i = -\left[\frac{\partial g_i}{\partial z_i}\right]^{-1} \left(\frac{\partial g_i}{\partial v} dv - \sum_{k=1}^{i-1} \frac{\partial g_i}{\partial z_k} dz_k\right), \tag{14}$$

for $dz_i$. A LU factorization of the matrix

$$\frac{\partial g_i}{\partial z_i} = L \cdot U$$

is performed allowing to obtain the solution directly by forward and backward substitution. At this, the determination of the lower triangular matrix $U$ and the upper triangular matrix $L$ is the most computational expensive step. Since the matrix is independent of the seed variables $dv$, the LU factorization is constant for every Jacobian matrix construction. Up to now, the LU factorization is performed for each Jacobian matrix evaluation in OpenModelica. We implemented the reuse of the LU factorization so that the LU factorization is computed only for the first evaluation of equation (11) and than reused for the subsequent evaluations. The performance benefits of this improvement is quite huge for the OpenModelica implementation as depicted in Section 4.

## 3.2 Parallelization of Jacobian Evaluation

From a computer architecture view, the overall performance of a system is driven nowadays by increasing core count due to the fact that further scaling of single core frequency has reached physical limits w.r.t. overheating and power consumption (Kirk and Wen-Mei 2016). Thus, algorithms and software need to be parallelized to benefit from today's multi-core chips (Sutter and Larus 2005), (Olukotun and Hammond 2005). In this section, we deduce a parallelization using multithreading for evaluation of Jacobians basing on representation (7).

For simplicity and without limiting the generality the sparsity and coloring is omitted in the following. The algorithms and results presented in this paper are directly applicable for colored Jacobians with compressed columns.

The basic idea is to enable concurrent evaluations of the independent columns (i.e. the directional derivatives) in the Jacobian matrix. In Listing 1 the main Jacobian construction loop of OpenModelica is depicted: The Jacobian columns are elemtwise evaluated within two nested `for` loops by calls to the function `directional_der_JacA`. The later corresponds to the directional derivative function in equation (9) and the main work here is to achieve a thread-safe version of it. This means that all writable variables and structures, as linear algebraic loops, need thread local data to enable simultaneous execution.

**Listing 1.** Main Jacobian construction

```
for(int i = 0; i < jac->columns; i++) {
  jac->seedVars[i] = 1.0;
  callbacks->directional_der_JacA(data, jac);

  for(int j = 0; j < jac->rows; j++)
    matrixA[i][j] = jac->resultVars[j];

  jac->seedVars[i] = 0.0;
}
```

Since the Jacobian can contain linear algebraic systems it is necessary to deal with non-thread-safe solvers in parallel regions. In general each thread has to solve the same linear system for different seed vectors simultaneously. Figure 1 provides a schematic overview. We decided for our first implementation to provide a duplication of the structure of all linear algebraic systems to every thread. Each thread works (read and write) exclusively on its own copy and is thus thread-safe. This approach is not the most memory economical and will be changed in the final implementation in OpenModelica. To switch from a global to a thread local data structure is a major change in the OpenModelica Compiler backend and its C simulation runtime. The thread local Jacobian data structure is depicted in Listing 2.

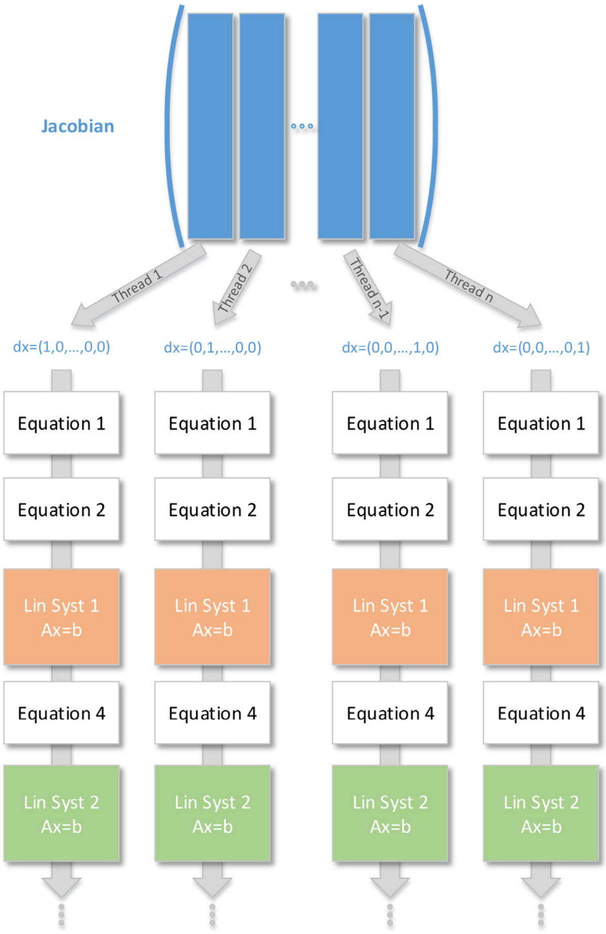


**Figure 1.** Evaluate Jacobian columns with linear systems in parallel

**Listing 2.** Thread local data structure for Jacobians.

```
typedef struct LINEAR_SYSTEM_THREAD_DATA {
  void *solverData[2]; /* Private date for
      external linear solvers */
  modelica_real *x;          /* Solution x */
  modelica_real *A;          /* Matrix A */
  modelica_real *b;          /* Vector b */
  ANALYTIC_JACOBIAN *jacobian; /* Jacobian */
  ...
} LINEAR_SYSTEM_THREAD_DATA;
```

The OpenMP API (*OpenMP Application Programming Interface* 2018) for shared-memory parallelization is used to implement the presented parallel Jacobian evaluation in the C runtime of OpenModelica. We choose OpenMP over possible alternatives, like Pthreads and Intel TBB, for two significant reasons:

*Availability:* The OpenMP specifications are corporately developed by hardware, software and compiler manufactures since 1997 and it has become a de-facto standard for shared-memory parallelization of C/C++ and Fortran applications. Thus, it is widely supported by all major compiler collections as well as available and applicable on most systems.

*Suitability:* Since OpenMP heavily supports the fork-join model by providing work sharing constructs, like parallel `for` loops and parallel sections. This fits the need for alternating single- and multithreaded computing stages in our presented approach. Parallelization is introduced through so-called pragmas, which are ignored if the compiler does not support OpenMP or no OpenMP support is requested. This leads to the huge advantage, that existing sequential code can be parallelized, but still remains sequentially executable if OpenMP is not supported.

The developed parallelization approach makes use of the fork-join model in the sense that the columns of the Jacobian are computed in parallel while the remaining simulation is sequentially executed. The $n$ columns of the Jacobian matrix are evaluated in the existing sequential implementation within $n$ iterations of a `for` loop. For the parallel implementation, this `for` loop is preceded by a work sharing `#pragma omp for` clause, which distributes the several loop iterations to the available threads at runtime. The implementation is free of explicit barriers and critical sections, i.e., the spawned threads run fully concurrently in the parallel region. The maximal number of independent chunks is barely the total number of columns of the Jacobian. Consequently, the approach can be scaled up to a maximum of $n$ threads for a fixed model. In practice, the speedup gained from using an increasing number of threads will saturate because of the ever increasing overhead due to thread management will predominate the computations. The actual mapping of the columns to the threads is performed by a scheduling algorithm. The OpenMP API offers several scheduling algorithms like static, guided and dynamic scheduling. Which scheduler fits best depends among other things on the number of columns, the system's architecture and the containing equations. The Section 4.3 holds studies on using different schedulers.

## 4 Benchmarks

In the following, we study the characteristics and possible accelerations of the proposed enhancements w.r.t. Jacobian evaluation within OpenModelica using models from the ScalableTestSuite. We choose the ScalableTestSuite over other Modelica libraries for benchmarking for the following reasons:

*Scalability:* All contained models are scalable by setting one or more integer parameters. This is essential for the analysis of the scaling abilities of the approaches and their implementations, since the size of the Jacobian scales with the model size.

*Significance:* There are several models with a quite time consuming Jacobian evaluation.

*Open-source*: The ScalableTestSuite is freely available under the BSD 3-Clause License at GitHub[1]. Thus, the models can be inspected, and the results can be retraced and reproduced, respectively.

### 4.1 Hardware and Software Stack

In order to provide transparent, comparable and reproducible results, we describe the hardware and the relevant software stack on which the benchmarks are evaluated. The software configuration is as follows: The GNU Compiler Collection (GCC, 7.0.1) was used as C and C++ compilers with the optimization flags `-O2 -march=native`. The open-source Lapack implementation OpenBlas in version 0.2.20 is linked for basic linear algebra routines. We implemented the proposed algorithms and the parallelization of the Jacobian evaluation within a branch basing on OpenModelica 1.13-dev. The OpenModelica compiler needs to be invoked with `--generateSymbolicJacobian` to generate symbolical Jacobians for a model and the simulation flag `-jacobian=symbolical` must be passed for execution.

All benchmarks are evaluated on the High Performance Computing system (HPC) Taurus at TU Dresden (*Online Documentation for HPC System Taurus* 2018). In particular, we choose an Intel Haswell node comprising of two Intel Xeon E5-2680 v3 CPUs with 12 cores each, and 64 GB RAM in total. In order to obtain reproducible benchmark results the Hyper-threading technology as well as the turbo mode is disabled and the benchmarks run with the CPU‘s base frequency of 2.5 GHz. The resources are exclusively allocated, i.e., no noise from other users has to be considered.

Unless otherwise stated, all benchmark configurations are repeated five times and the average time over the runs is determined and used for the analysis. Using the average is reasonable due to the fact that the minimal and maximal timing values for all benchmarks are very close to the corresponding mean value.

### 4.2 Reuse of Constant Parts

The thermal models HeatingSystem_N ($N = 20, 40, 80$) and electric models PowerSystemStepLoad_N_64_M ($M = 4, 8, 16$) from ScalableTestSuite are used to analyze the proposed reuse of constant parts.

The HeatingSystem model represents a district heating system with $N$ heated units, supplied by a heat distribution system. We noticed that for all $N$ between 14 and 15 % of the total number of equations for the Jacobian evaluation is recognized as constant equations regarding the seed vector and therefore computed only once per time step.

The PowerSystemStepLoad model assembles a power system with a linear topology, obtained by connecting $N$ power generators with each $M$ finite volumes in a linear network with equal transmission lines, and with a load connected to each generator. Similar to the model above OpenModelica with reuse of constant parts is able to recognize between 22 and 23 % of the Jacobian equations to be reusable for different seed vectors.

The constant equations in these models consists entirely of trigonometric functions, depending only on state variables. While they are not especially expensive, there

[1] `https://github.com/casella/ScalableTestSuite`

are a lot of them. For example the PowerSystemStepLoad_N_64_M models contain each 4032 constant equations and up to 14210 equations dependent of the seed vector, while the Jacobian is of dimension up to $1537 \times 1537$.

The Tables 2 and 3 present the achieved timings for the HeatingSystem and PowerSystemStepLoad models using colored symbolical Jacobians. The columns with heading *No RcP* refer to the previous implementation without any reusage of constant parts in Jacobian evaluation. In contrast, the timings provided by the improved implementation, which considers constant parts, are stated within the columns *RcP*. For completeness and comparison reasons, the timings obtained from the default numerical technique for Jacobian evaluation are also given. The best timings for the Jacobian evaluation are highlighted in bold face.

As can be seen from Table 2, the reusage of constant parts significantly accelerates the Jacobian evaluation for symbolical Jacobians by a factor of 1.8 up to 2.0.

For this model, it would be advantageous to use dense symbolical Jacobians, since the number of colors found by the used heuristic is equal to the dimension of the Jacobian. Overall, the best timings are obtained for all model sizes using symbolic Jacobians in conjunction with reusage of constant parts. Furthermore, the symbolical Jacobian evaluation is considerably faster than the default numerical method. For example, the time spent to evaluate the Jacobian matrices drops from 316.61 s to 62.96 s for model size $N = 80$.

The benchmark results for the model PowerSystemStepLoad_N_64 are depicted in Table 3. For this model, the gained speed-up from the reusage of constant parts while Jacobian evaluation is about 1.7. The presented approach speeds up the Jacobian evaluation to such an extent, that it now outperforms the numerical evaluation.

The effect of the reusage of constant parts technique depends on the ratio between function calls that can be gathered as common sub expressions and then extracted with `RemoveSimpleEquations` and the total number of equation for the Jacobian evaluation.

## 4.3 Parallel Jacobian Evaluation

The model DistributionSystemLinear_N_M ($N = 14, 20, 28,\ M = N$) is used to demonstrate the application of the parallel Jacobian evaluation. This model represents an AC current distribution system, where the amount of segments can be scaled by the two parameters $N$ for the primary distribution line and $M$ for each secondary distribution line. The considered model sizes and the corresponding attributes are depicted in Table 4. Since the columns of the Jacobian matrix can be evaluated completely independently from each other and the implementation is free of explicit synchronization constructs (e.g., barriers and critical clauses), we expect near linear scalability for the parallelization approach. Full linear scalability cannot be expected, because the columns evaluated by the various threads need to be written to the global Jacobian matrix which is in shared memory. And on the other hand, the OpenMP loop scheduling will introduce some parallelization overhead as well. Figure 2 shows the strong scaling of the parallel Jacobian evaluation for models of size $N = 14$, $N = 20$ and $N = 28$, whereas the speedups refer to execution time with one thread. The black doted curve refers to the ideal (i.e, linear) speedup. As can be seen, the achieved speedup curves of all three models are nearly identical and the scalability is very close to linear speedup. This is a very good result and shows that the Jacobian evaluation can be significantly accelerated by parallelization. We also like to stress that the parallel Jacobian evaluation scales up to almost the full system size for all model sizes. Because the synchronization on OS level using the full system with 24 threads dominates the small amount of calculations w.r.t. thread-local Jacobian evaluations, the scaling slumps for model sizes $N = 14$ and $N = 20$ at this point. For model size $N = 28$ the computational portions overlay the overhead due to OS synchronization.

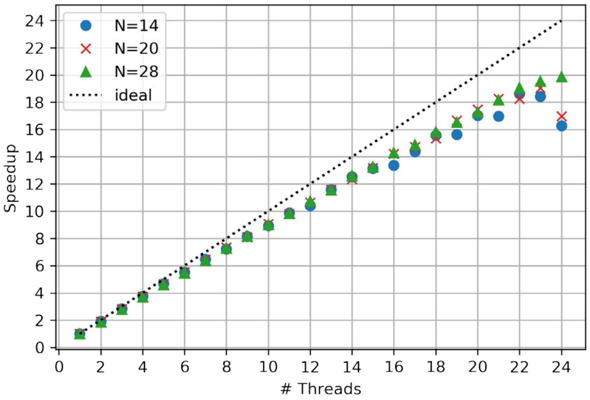


**Figure 2.** Strong scaling of parallel Jacobian evaluation for models with size $N = 14$, $N = 20$ and $N = 28$.

Next, we study the impacts of the four loop scheduling algorithms that the OpenMP API offers. Loop scheduling can have an serious impact on the scaling behavior and execution time of a parallel loop. This is especially true if not all iterations of a loop are equal in terms of computational effort. Such disequilibrium is called load imbalance. With respect to the considered parallelization approach for the Jacobian evaluation we do no expect load imbalances. But, eventually the good scaling behavior can be continued to full system size even for small models. The available scheduling algorithms are an in short:

*Static:* Without specification of the chunk size, the loop is divided into approximately equal chunks. They are assigned to the available threads.

*Dynamic:* The iterations are distributed to threads in the team in chunks. Each thread executes a chunk of iterations, then requests another chunk, until no chunks remain to be distributed. The chunk size defaults to 1.

**Table 2.** Comparison of timings (in seconds) for total simulation and Jacobian evaluation for the HeatingSystem models achieved by using numerical and symbolical Jacobian evaluation without and with reuse of constant parts.

| | Numerical | | | Symbolical colored | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | *No RcP* | | *RcP* | | |
| N | t_total | t_jac | #Jac Eval | t_total | t_jac | t_total | t_jac | #Jac Eval |
| 20 | 11.70 | 6.20 | 54780 | 8.87 | 2.91 | 7.49 | **1.51** | 54647 |
| 40 | 67.82 | 43.64 | 102521 | 52.71 | 21.21 | 43.76 | **11.59** | 102548 |
| 80 | 451.65 | 316.61 | 189576 | 404.59 | 153.70 | 324.56 | **75.98** | 190012 |

**Table 3.** Comparison of timings (in seconds) for total simulation and Jacobian evaluation for the PowerSystemStepLoad_N_64_M models achieved by using numerical and symbolical Jacobian evaluation without and with reuse of constant parts.

| | Numerical | | | Symbolical colored | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | No RcP | | RcP | | |
| M | t_total | t_jac | #Jac Eval | t_total | t_jac | t_total | t_jac | #Jac Eval |
| 4 | 4.29 | 0.62 | 17 | 4.46 | 0.84 | 4.11 | **0.49** | 21 |
| 8 | 4.76 | 0.78 | 20 | 4.92 | 0.85 | 4.57 | **0.50** | 21 |
| 16 | 5.25 | 0.83 | 19 | 5.26 | 0.84 | 4.89 | **0.49** | 19 |

**Table 4.** Model size $N$ and the corresponding values for number of variables, states and algebraic loop size and columns in Jacobian matrix.

| $N$ | Vars | States | Loops | Cols. in $J$ | Evals. of J |
|---|---|---|---|---|---|
| 14 | 12364 | 196 | 1621 | 196 | 15 |
| 20 | 25096 | 400 | 3277 | 400 | 14 |
| 28 | 49016 | 784 | 6381 | 784 | 15 |

*Guided:* The iterations are assigned to threads in the team in chunks. Each thread executes a chunk of iterations, then requests another chunk, until no chunks remain to be assigned. For a chunk size of 1, the size of each chunk is proportional to the number of unassigned iterations divided by the number of threads in the team, decreasing to 1. It defaults to 1.

*Auto:* The decision regarding scheduling is delegated to the compiler and/or runtime system. The programmer gives the implementation the freedom to choose any possible mapping of iterations to threads in the team.

The OpenMP API provides the *runtime* clause, via which the schedule type can be specified at runtime. Thus, recompilation of the entire project is not necessary. In Figures 3 and 4 the speedups of the four different scheduling types for model size $N = 14$ and $N = 28$, respectively, are diplayed. The speedup is calculated with respect to the sequential execution time of the default scheduling type, which is dynamic for GNU libgomp (*Online Documentation for GNU libgomp: OMP_SCHEDULE* 2018). The speedups for the different schedulers are very tightly bunched together for 1 to 23 threads. There is a meaningful difference in performance using 24 threads for the smaller model. At this point, the computational parts become to small compared to the overhead introduced by the thread managment. The scalibility of the approach is not limited to 23 threads, but depends on the number of Jacobian columns. The larger model scales up to the full system using dynamic and guided scheduling. Overall, the default scheduler (i.e., dynamic with chunk size of 1) provides the best results. Hence, no adjustments from the users will be necessary.

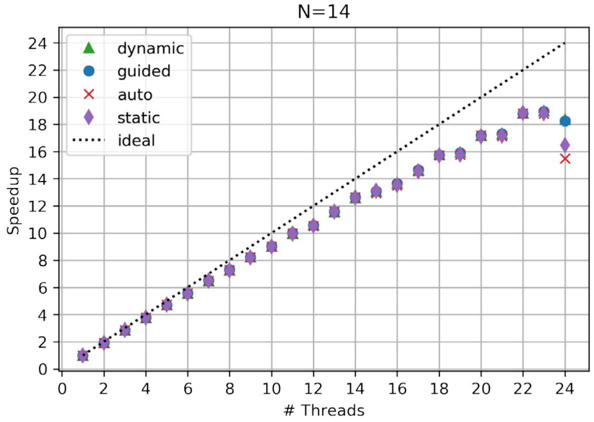


**Figure 3.** Speedup obtained for the scheduling types *static*, *dynamic*, *guided* and *auto* for model size $N = 14$.

## 5 Conclusion and Outlook

The presented work contributes to the efficient and parallel Jacobian evaluation using symbolic differentiation. For that, two complementary techniques are proposed to accelerate the computations:

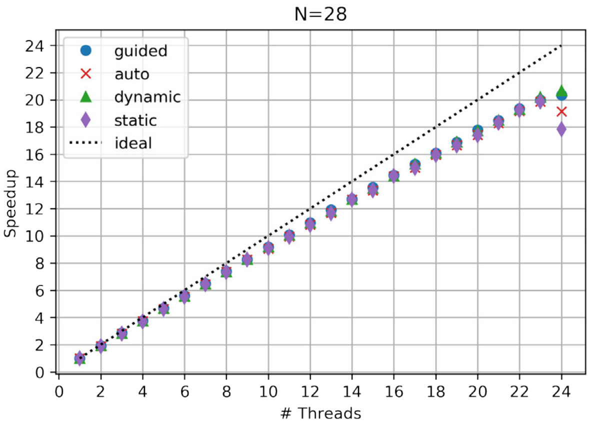


**Figure 4.** Speedup obtained for the scheduling types *static*, *dynamic*, *guided* and *auto* for model size $N = 28$.

### Reuse of Constant Parts

While the speedup for many models from the ScalableTestSuite are enormous there are still examples for which OpenModelica currently is not able to find all constant parts. For example for the SteamPipe models no constant parts are found, despite big potential. The problem is, that `removeSimpleEquation` currently can't handle array equations and equations with function calls inside function calls. We plan to extend the described implementation in this way and expect a significant improvement for all examples inside ScalableTestSuite using Modelica.Fluid and Modelica.Media models.

### Parallel Jacobian Evaluation

The parallelization approach allows for concurrent evaluation of the several columns of a Jacobian, which are independent. The OpenMP API is used for the implementation in OpenModelica. As discussed from a theoretical point of view and as shown by benchmarks, the parallel Jacobian evaluation provides near linear scalability. Simulations where Jacobian evaluations have a large share and optimization algorithms can greatly benefit from this. Although, only GCC was considered as compiler set, the scaling property of the approach and the implementation should be independent from the particular tool chain and will also hold for smaller shared-memory systems.

Furthermore the same approach can be adapted to the numerical and colored Jacobian evaluation in a similar way and should speed up the default simulation with OpenModelica significantly.

### Sparse Rows Evaluation

As published in (Braun, Gallardo Yances, Link, and Bachmann 2012) the sparsity pattern of the Jacobian matrix can be calculated in OpenModelica and used to compress the matrix by combining columns with no shared non-zero elements in the same rows. This process is well-know as coloring and applied by replacing the identity matrix $I$ in (7) with a compressed matrix, where all non-zero elements of the same color are combined into one compressed column as depict in Figure (5). Although the speed-up by compressing Jacobian matrices is enormous, still in every column the full directional derivative (9) is evaluated. This also includes all rows that are structurally zero.

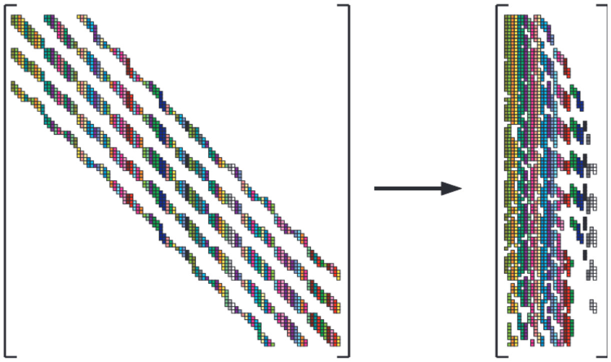


**Figure 5.** A matrix and its compressed representation from (Gebremedhin, Manne, and Pothen 2005).

To address this issue we propose to exploit the sparsity pattern further by attributing every equation whether it needs to be evaluated for the current column or not. To make this decision the sparsity pattern is used to mark the output rows of every column. In the next step an algorithm is applied to detect the minimal equation set that is needed to evaluate the marked output rows. The developed algorithm is based on Tarjan's algorithm as proposed in (Manzoni and Casella 2011). A necessary input for this algorithm is the directed graph, which is based on matching of the system (9). The output of the algorithm is mapped to every equation, so that at runtime every equation of (11) includes the dependency characteristic.

Since a large portion of the costly function calls can be reused as shown in section sec:ConstantParts it is not clear how big the remaining impact of this approach is. A first implementation of the described approach, but without the reuse of constant parts, leads to no clear results and needs further study.

### Combination of Described Techniques

When writing this paper, the presented techniques are implemented on different branches of OpenModelica. Both branches base on a recent OpenModelica version and are very close to the mainline development branch and should be included into the master branch in the near future and be part of the next release.

## Acknowledgment

The presented work is part of the PARADOM project funded by the Federal Ministry of Education and Research (BMBF) under support code 01IH15002.

## References

Braun, Willi and Bernhard Bachmann (Feb. 2014). *Generic Differentiation Module and its Application within the OMC Backend*. Annual OpenModelica Workshop, Linköping Universitet, Sweden. Open Source Modelica Consortium. PDF.

Casella, Francesco (2015). "Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. 118. Linköping University Electronic Press, Linköpings universitet, pp. 459–468.

Franke, Rüdiger, Marcus Walther, Niklas Worschech, Willi Braun, and Bernhard Bachmann (2015). "Model-based control with FMI and a C++ runtime for Modelica". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. 118. Linköping University Electronic Press, Linköpings universitet, pp. 339–347.

Gebremedhin, Assefaw Hadish, Fredrik Manne, and Alex Pothen (Apr. 2005). "What Color Is Your Jacobian? Graph Coloring for Computing Derivatives". In: *SIAM Rev.* 47.4, pp. 629–705. ISSN: 0036-1445. DOI: 10.1137/S0036144504444711. URL: http://dx.doi.org/10.1137/S0036144504444711.

Kirk, David B and W Hwu Wen-Mei (2016). *Programming massively parallel processors: a hands-on approach*. Morgan kaufmann.

*Online Documentation for GNU libgomp: OMP_SCHEDULE* (June 2018). Accessed 2018-06-03. GNU. URL: https://gcc.gnu.org/onlinedocs/libgomp/OMP_005fSCHEDULE.html#OMP_005fSCHEDULE.

Manzoni, Vincenzo and Francesco Casella (Mar. 2011). "Minimal Equation Sets for Output Computation in Object-Oriented Models". In: *Proceedings 8th International Modelica Conference*. Ed. by C. Clauss. Modelica Association. Dresden, Germany, pp. 784–790. ISBN: 978-91-7393-096-3. DOI: 10.3384/ecp11063784. URL: http://www.ep.liu.se/ecp/063/088/ecp11063088.pdf.

Braun, Willi, Lennart Ochel, and Bernhard Bachmann (Mar. 2011). "Symbolically Derived Jacobians Using Automatic Differentiation - Enhancement of the OpenModelica Compiler". In: *Proceedings of the 8th International Modelica Conference*. Ed. by Christoph Clauß. Dresden, Germany: Linköping University Electronic Press. DOI: 10.3384/ecp11063.

Braun, Willi, Stephanie Gallardo Yances, Killian Link, and Bernhard Bachmann (Sept. 2012). "Fast Simulation of Fluid Models with Colored Jacobians". In: *Proceedings of the 9th International Modelica Conference*. Ed. by Martin Otter and Dirk Zimmer. Munich, Germany: Linköping University Electronic Press. DOI: 10.3384/ecp12076.

Åkesson, Johan, Willi Braun, Petter Lindholm, and Bernhard Bachmann (Sept. 2012). "Generation of Sparse Jacobians for the Function Mock-Up Interface 2.0". In: *Proceedings of the 9th International Modelica Conference*. Ed. by Martin Otter and Dirk Zimmer. Munich, Germany: Linköping University Electronic Press. DOI: 10.3384/ecp12076.

Olukotun, Kunle and Lance Hammond (2005). "The future of microprocessors". In: *Queue* 3.7, pp. 26–29.

*OpenMP Application Programming Interface* (June 2018). Accessed 2018-06-01. OpenMP Architecture Review Board. URL: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf.

Ruge, Vitalij and Bernhard Bachmann (Feb. 2014). *Efficient Built-in Dynamic Optimization Features of OpenModelica*. Annual OpenModelica Workshop, Linköping Universitet, Sweden. Open Source Modelica Consortium. PDF.

Sutter, Herb and James Larus (Sept. 2005). "Software and the Concurrency Revolution". In: *Queue* 3.7, pp. 54–62. ISSN: 1542-7730. DOI: 10.1145/1095408.1095421. URL: http://doi.acm.org/10.1145/1095408.1095421.

*Online Documentation for HPC System Taurus* (June 2018). Accessed 2018-09-11. TU Dresden. URL: https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/HardwareTaurus.

# Traceability in the Model-based Design of Cyber-Physical Systems

Christian König[1] Alachew Mengist[2] Carl Gamble[3] Jos Höll[1] Kenneth Lausdahl[4] Tom Bokhove[5] Etienne Brosse[6] Oliver Möller[7] Adrian Pop[2]

[1]TWT GmbH Science & Innovation, Stuttgart, Germany, christian.koenig@twt-gmbh.de
[2]Department of Computer and Information Science, Linköping University, Sweden, alachew.mengist@liu.se
[3]School of Computing, Newcastle University, Newcastle upon Tyne, United Kingdom
[4]Department of Engineering - Aarhus University, Finlandsgade 22, Aarhus N,Denmark
[5]Controllab Products, Enschede, The Netherlands
[6]Softeam Research & Development Division, Paris, France
[7]Verified Systems International Bremen, Germany

## Abstract

Design, development, and analysis of complex Cyber Physical Systems (CPSs) using models involves a collaboration of expertise from different engineering domains. Heterogeneous artefacts are generated, often using different lifecycle modeling languages and simulation tools. Capturing the traceability information among these artefacts can be used to support several activities such as requirements tracing, impact analysis of change requests, verification, validation, and documentation. However, creating trace links among these heterogeneous artefacts is challenging as different tools in the development lifecycle are usually disparate and there is no precise semantic in the terminology used between requirement engineers, verification engineers, and system modelers. In this paper, we present a linked data-based approach to capture traceability information and create trace links that relate heterogeneous artefacts in the model-based design process of CPSs through a standardized interface and format using OSLC. This enables artefacts from different tools to be connected and queried through a standardized interface and format. A practical prototype system for supporting traceability is designed through integration with the INTO-CPS toolchain of CPS design. The traceability data is stored in Neo4j graph database which can be queried for generating various reports such as impact analysis, variant handling, etc.

*Keywords: Traceability, Trace links, Linked data, Tool integration, OSLC, Open Service for Lifecycle Collaboration, Model Based Design, Cyber-Physical-Systems*

## 1 Introduction

Cyber-Physical Systems (CPSs) are complex systems that typically consist of computing elements, physical elements and communication channels (Song et al., 2016). CPSs can for example be found in the aerospace domain, in automotive engineering, building automation or robotics, where they often have safety-relevant features. Therefore, exhaustive testing of the systems is necessary. To minimize cost and development time, much of the testing needs to be done virtually, while the actual device is not yet fully ready. To allow this in an efficient manner, model-based systems engineering (MBSE) methods are being applied to the design of CPSs, to develop and test those systems in a time and cost-efficient manner. The MBSE approach requires that parts and features of the system can be related to the design requirements in an automated fashion, supported by appropriate tools. This is the requirements traceability challenge, where the different steps of the implementation and validation of requirements need to be traceable to the original requirements. Only then is it possible to reliably demonstrate that all requirements have been taken into account in the design of the CPS. The software tools that are used to develop and test such complex systems are often heterogeneous.

One important challenge for MBSE of CPS therefore is to enable the engineers to trace the different elements of a CPS along its development cycle back to the requirements. For the tools that are used for the development, this means that they require a common approach to provide and process the traceability-relevant data, and to have a common syntax and semantics for generating and transmitting the data.

During the past decade, the Open-Services for Lifecycle Collaboration (OSLC) specifications (Open-services.net, 2008) have emerged for integrating development lifecycle tools (e.g., modeling tools, change management tools, requirements management tools, quality management tools, configuration management tools) using Linked Data (Heath and Bizer, 2011) approach. The goal of OSLC is to make it easier for tools to work together by specifying a minimum amount of protocol without standardizing the behavior of a specific tool. The OSLC specifications use the Linked Data method to enable integration at the data level via links between artefacts. The artefacts information is represented as Resource Description Framework (RDF) (Manonla and Miller, 2004) resources identified by HTTP URIs. OSLC also provides a common protocol for manipulating those RDF resources through standard RESTful (Richardson and Ruby, 2007) web services such as creation (HTTP POST) and retrieval (HTTP

10.3384/ECP20169168 DOI

GET), update (HTTP PUT) and delete (HTTP DELETE) operations on RDF resources.

The main contribution of this paper includes the followings. First, we demonstrate a Linked Data-based approach for traceability in the model-based design process for CPSs, and the implementation of this traceability approach in the respective tools. This enables recording and establishing the traceability links of model elements (e.g., requirements, activities, artefacts, modeling tools,) through a standardized interface and format using OSLC. Second, we build a flexible trace links ontology of artefacts between requirements, simulation models, FMUs, simulation results, and test results. Third, we validate the ontology through an example workflow with heterogeneous artefacts in systems engineering.

## 2 Background

While there have been significant efforts dedicated to the introduction of traceability, there are several challenges that prevent traceability from being used in all the cases where such an application would be beneficial.

First, for traceability to be accepted in industrial use, the overhead that is created by it, must be minimal. Therefore, most functions need to be automated, and well integrated into different tools. Due to the heterogeneity of software tools that are used in systems engineering, all tools need to be equipped with interfaces that exchange data in a unified format, so that the syntax and semantics of the traceability data are compatible. Different "traceability information models" (TIM) have been developed that define the data model used to describe the traceability links between different entities (Mustafa and Labiche, 2017). However, up to now, no universal TIM was established.

For the usability of traceability, considering the representation of the essential traceability data depending on the context is important (Li and Maalej, 2012). Here, visualisation of the data as matrices, lists or trees was explored.

This paper describes the concept and following implementation of traceability in the INTO-CPS project, that was running from 2015 until 2017 (Larsen et al., 2016a). In this project, an integrated tool-chain for model-based design of CPS was developed. On the tooling side, INTO-CPS covers abstract modelling in SysML with the Modelio[1] tool. Detailed modelling of the different parts of a CPS is done in the tools Overture[2], 20-sim[3] and OpenModelica[4]. System simulation is executed through a newly created Co-Simulation Orchestration Engine (COE) based on the FMI standard (Blochwitz et al., 2012). The FMI standard allows creation of executable units, called Functional Mock-up Units (FMUs) that contain a simulation model and its solver. Such an FMU can be controlled with a standardized API, and its input and output signals are described with an XML file, which is called the `modelDescription.xml`. Test automation and model checking are performed by the RT-Tester toolsuite[5]. INTO-CPS enables a smooth workflow between the different tasks of CPS development, and the corresponding tools. Regarding traceability, the different tools initially had no interface for exchanging traceability data.

License rights for distribution and usage of INTO-CPS application, along with the COE and the traceability daemon and the SysML profile for Modelio, which also contains the traceability features, reside with the INTO-CPS association [6] under open source. The traceability functions for Overture and OpenModelica belong to the respective organisations and are also available under an open source license.

## 3 Traceability in the INTO-CPS project

### 3.1 Scope

The primary scope for traceability in INTO-CPS is demonstration of the basic traceability tasks across the tool-chain. This includes mostly tracing of the requirements and connecting them with the models, the simulation results, the produced code and test results. Furthermore, analysing the impact of changes (e.g. in requirements) to the overall system can be supported through traceability. Traceability is meant to create as little overhead for the user as possible, so that most actions should occur automatically without the need for user interaction. The openness of the INTO-CPS tool-chain shall be maintained, by defining open interfaces and formats, such as the ones described in this paper.

However, it is not in the primary scope of the traceability work in INTO-CPS to be able to trace back all steps of the development and revert to each step in the development's history. For this, other parts of the INTO-CPS project and tools are seen as more appropriate, such as versioning tools (SVN, Git) or functionality of the INTO-CPS Application. Therefore, as will be described below in Section 5, some of the tools support Git for versioning.

### 3.2 Ontology

The traceability ontology for INTO-CPS uses concepts from the PROV working group of the World Wide Web Consortium (W3C)[7]. PROV deals with three objects: *entity*, *activity* and *agent*. Entities can include requirements, models, simulation configuration files or simulation results. Activities can be saving of a model, running a cosimulation or more. An agent is a user that performs these activities.

[1] see `https://www.modelio.org/`

[2] see `http://overturetool.org/`

[3] see `http://www.20sim.com/`

[4] see `https://openmodelica.org/`

[5] see `https://www.verified.de/products/`

[6] see `http://into-cps.org/`

[7] see `https://www.w3.org/TR/prov-overview/`

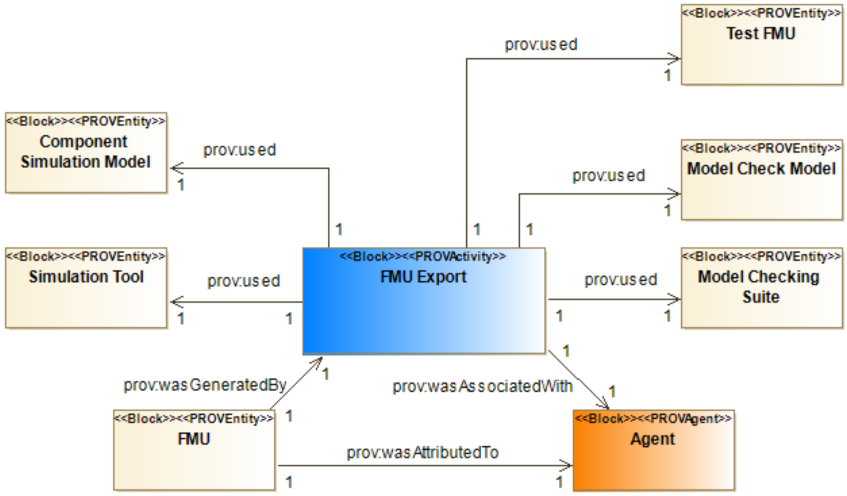


**Figure 1.** Block Definition Diagram showing the *FMU Export* activity.

In the context of traceability, these objects are connected by relations. PROV proposes relations such as "used", "was generated by" or "was derived from". In addition to these relations, further relations are used for the ontology of INTO-CPS, to express the relevant relations. These are taken from the OSLC[8] definitions. Furthermore, custom relations are proposed. The complete list of relations is described in the following table:

| | |
|---|---|
| prov:used | one entity used another one |
| prov:wasAttributedTo | attribution of entities |
| prov:wasAssociatedWith | association of activities |
| prov:wasGeneratedBy | one entity is generated from another |
| prov:wasDerivedFrom | one entity is derived from another |
| prov:hadMember | one entity has one or more members |
| oslc:elaborates | an entity that elaborates on a requirement |
| oslc:satisfies | an entity that satisfies a requirement |
| oslc:verifies | an entity that verifies an assumption |
| into:doesNotVerify | an entity that does not verify an assumption |
| into:violates | an entity that violates an assumption |

A number of activities are defined which shall create traces, using the objects and relations shortly described above. To illustrate this concept, a single activity is described in the following Figure 1 with its SysML Block Definition Diagram. The Activity "FMU Export" is associated with an agent and related to a number of entities. It generates an FMU (e.g. a file) and uses a simulation tool and a Component Simulation Model, and may in addition use a Test FMU, a Model Check Model and a Model Checking Suite, if the FMU is derived from a Model Checking activity. Finally, the FMU Export activity is associated with an Agent, and the FMU itself is attributed to this agent.

## 3.3 Architecture

This section introduces an outline of the tool and file system elements that support the traceability activities in the INTO-CPS tool-chain, and this outline is shown in Figure 2.

The central element of the traceability architecture is the *traceability daemon*, along with an interface (a REST-API[9]) that can be used in two ways: The tools (e.g. 20-Sim, OpenModelica, Overture, Modelio, RT-Tester, the COE, the INTO-CPS application) write data to the interface (e.g. they send traceability information from actions that happened within the tools to the daemon), the INTO-CPS application queries the interface (e.g. for retrieving information from the database). The daemon acts as front-end to the database (both for write and read operations, as explained later).

A schema [10] for the traceability messages is developed, which defines the format of the messages, and restricts their content. The schema defines the valid syntax of data submissions from the various tools. The tools need to implement the correct format, and the daemon validates if the tool has done it properly. This makes sure that the database contains only information that follows the same format, and therefore can be queried easily. Crucially, since the schema is machine-readable, the validation is done automatically. Furthermore, since the schema is public, traceability can be easily implemented in other

[8] see http://open-services.net/

[9] REST: Representational State Transfer; API: Application Programming Interface

[10] see https://github.com/INTO-CPS-Association/into-cps-application/tree/development/src/resources/into-cps/tracability/schemas

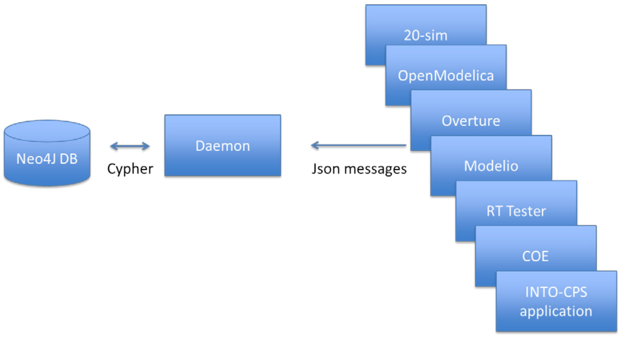


**Figure 2.** Schematic architecture of the traceability-related tools.

tools (e.g. tools from vendors outside the INTO-CPS consortium, for instance from the INTO-CPS association) so that these can send valid traceability messages to the daemon. In the schema, all allowed traceability-related entities, such as activities, artefacts or tools are contained. Relations between entities, such as *prov:wasGeneratedBy*, *oslc:satisfies* and more are also described in the schema. It is therefore important in such a tool-chain-wide approach as with traceability, that all the tools comply with the schema and that the whole ontology (see Section 3.2) is covered by it. The process of generating messages, sending them to the daemon and validating them, is shown in the Figure 3 below. The user action, which in this example is the export of a `modelDescription.xml` file from the Modelio tool, triggers the generation of a message with the file ending *.dmsg, which shall comply to the Schema. This message is sent via HTTP to the daemon, who validates the message according to the schema. If the message is valid, the daemon adds the content of this message to the global database and saves the *.dmsg file in the project folder.

## 4 Example Workflow

To illustrate the methods that have been described in the previous sections, we introduce an example. Figure 4 illustrates on the left side a complete workflow for the process from System Modelling, Behaviour Modelling and Co-Simulation, here performed using the tools Modelio, OpenModelica and the INTO-CPS Application..

The user starts with System Modelling, where the system is described by using SysML blocks including their ports and attributes. Requirements are written down and linked to the different SysML blocks that shall implement them. From a single block, a `modelDescription.xml` file can be exported, which is then imported into a modelling tool such as OpenModelica, to give the frame in which the behaviour will be implemented. This model will be saved, and finally exported to an FMU. Multiple of these FMUs will be connected in the INTO-CPS Application to form a Multi-model. The Co-Simulation will be set up with parameters

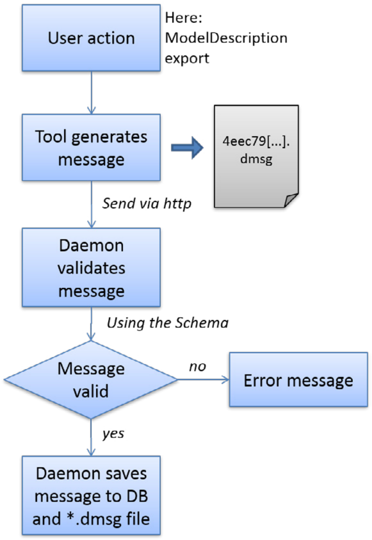


**Figure 3.** Schematic process of generating and saving a traceability message. For readability, the message content is omitted.

such as time-stepping or simulation duration. Finally, the Co-Simulation is run to give the simulation traces.

The right hand side of Figure 4 shows the main artefacts, from the requirements over the model files to the simulation results, and their relations in the Prov or OSLC notation (see Section 3.2), as they are stored in the traceability database. Note that for readability, we show only the most relevant artefacts here, and omit those that are created by the tools to represent authors, activities, tools, timestamps or additional information. The figure shows that all the artefacts are linked, so that it is for instance possible to see that a particular requirement has been implemented in a system element, and later its behavior has been simulated. As we will discuss in Section 6, it is also possible with the usage of a test automation tool, to add information about the verification or violation of requirements.

This example only shows a very simple workflow, without advanced activites such as Model Checking or Design Space Exploration. It does, however, illustrate the relation between the engineering workflow, the traceability data and its representation. The different steps in the development cycle of a CPS that are depicted in Figure 4 are usually performed by multiple people or organisations, using different tools.

## 5 Implementation in the INTO-CPS tools

This section describes the implementation of the traceability features in the different tools, which are shown in Figure 2. OSLC is chosen as an implementation specification to manage traceability information from different lifecycle tools used in the model-based design of Cyber-Physical Systems. In our prototype, artefacts and their

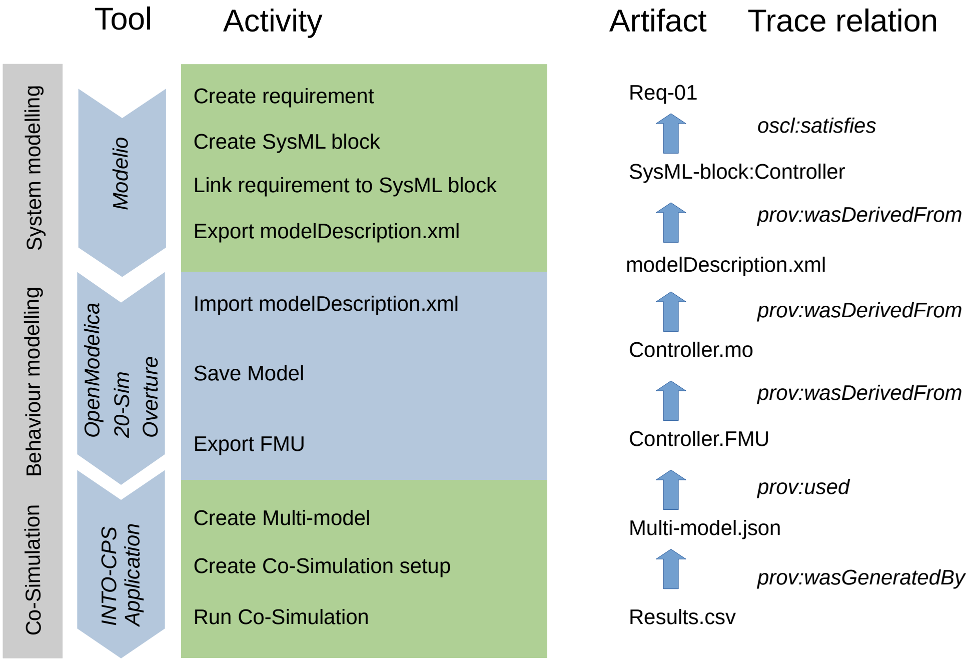


**Figure 4.** Example workflow (left), and most relevant artefacts and their trace relations (right).

relationships are described using an RDF/JSON format. The structure of the RDF consists of RDF triples with a Uniform Resource Identifier (URI) grouped into graphs: artefacts(subjects), relations (Predicates) and what it belongs to (objects). For instance, "Model (Subject) satisfies (Predicate) a requirement (Object)". The type of link (i.e., the predicate used in the link triple) defines the semantics of the link thus providing traceability between artefacts.

## 5.1 Traceability Daemon

The *traceability daemon* is essentially the core of the traceability tool support. In our implementation, it is launched or terminated by the INTO-CPS Application. The daemon's primary function is to create an OSLC compliant HTTP port and listen for the POST and GET actions. The different tools send data to an IP address at which the daemon is running (which, in our implementation, must be known to the tools). It stores the data sent via a POST request and will return a suitable response to a GET request. The requests are sent from the different tools, and the received data is stored in the Neo4J database. The daemon provides an interface that allows to retrieve the required data. This interface basically passes Cypher (see Section 6) queries to the Neo4J database.

The database is stored in a binary format, which causes problems when it is versioned (e.g. in a Git or SVN system) and changed by multiple users. To solve this, a step is added between receiving of the traceability messages and storing them in the Neo4J database. Each message the daemon receives is saved as plain text into a single file (with .dmsg file ending) in the project folder. The content of one such file is indicated in Figure 3. At the startup of the INTO-CPS Application, the daemon builds the database from these single files. This allows multiple users to work on the project simultaneously. Each user generates traceability messages by the different actions he/she performs. These messages are stored in the project folder. After completion of a task, the user pushes the files to the global repository. Merging of the database is then done by combining the .dmsg files. After an update of the project folder, each user has access to the whole database. The schematic process is shown in Figure 5 below.

In addition, the daemon is validating the messages it receives from the tools with respect to the schema, to make sure that only those messages that comply with the schema are written into the database. Only when all tools use the same message format will the queries (see Section 6) return meaningful information.

## 5.2 Modelio

The requirements definition part is handled only by Modelio. Modelio represents the *Architecture Modeling* activity in the INTO-CPS workflows. Modelio records the following traceability actions: Architecture creation, Architecture modification, ModelDescription export, Co-Simulation configuration export, Requirements generation and linking to SysML blocks.

Consequently, these actions are traced[11]. The Architecture creation / modification captures the generation and modification of a SysML block in Modelio. The genera-

[11] In this context, "traced" means that messages are generated and sent to the daemon

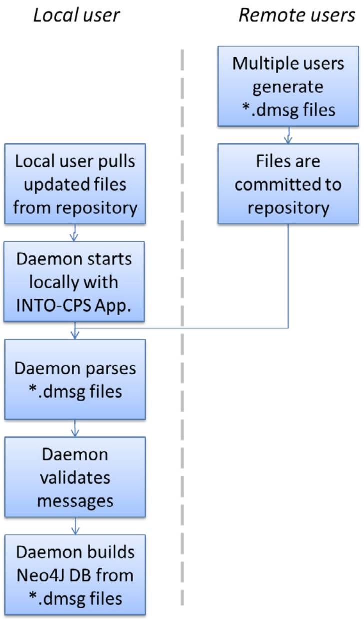

Local user
Remote users
Multiple users generate *.dmsg files
Files are committed to repository
Local user pulls updated files from repository
Daemon starts locally with INTO-CPS App.
Daemon parses *.dmsg files
Daemon validates messages
Daemon builds Neo4J DB from *.dmsg files

**Figure 5.** Schematic process of merging multipe messages from different users and building the Neo4J database from them.

tion of `modelDescription.xml` files from a SysML block is the next step in the workflow. Exporting a co-simulation configuration from a SysML connections diagram, which can be transformed in the INTO-CPS application into a Multi-model, is also traced. Generation of requirements, and association of these requirements with SysML blocks is traced. This association can either be of the type *oslc:verifies* or *oslc:satisfies*.

In addition to the ad-hoc generation of traceability messages, which are created when the related action is being performed, Modelio also offers the option to convert the Git history of a Modelio project into traceability messages. This is particularly useful for situations, where traceability was not used from the very beginning.

## 5.3 Modeling tools

While Modelio is the starting point for requirement tracing, the modeling tools OpenModelica, 20-sim and Overture capture specific functional and non-functional aspects of the system design. They are described briefly in this section. Generation of models, either from scratch or from an imported `modelDescription.xml` file (e.g. coming from Modelio in the previous step) is the next step in the workflow, and consequently traced, together with their modification. FMUs can be imported from other tools, to include them in the native models. Exporting an FMU is the next step in the workflow, and is consequently also traced. OpenModelica, 20-sim and Overture record the following traceability actions: Model creation, model modifiation, FMU export, FMU import, modelDescription.xml import.

**20-sim** 20-sim is a modeling and simulation tool for mechatronic and control systems. Because Git is needed for the INTO-CPS traceability daemon, it is not possible to only enable the traceability daemon without enabling Git version control. If both options (for Git version control and for communication with the traceability daemon) are enabled, every traceable action in 20-sim will store a copy of its data in the indicated Git repository. If the model itself is already in a Git repository, this will also make sure to commit the changes to this repository automatically. There is an additional option named “Write custom save messages”, which will ask the user to write a custom message whenever a traceable action is performed. This message will be stored in Git as the Git commit message.

The “Model creation” action is a “Save as” action, which is the moment when the user officially saves a new model to disk. In the same line of reasoning, a “Model modification” action is a “Save” action in 20-sim, because the user modifies an existing model on disk. 20-sim has no support for deleting a model from within its user interface, therefore there is no traceability query to delete a model from 20-sim. 20-sim also has support to export and import an FMU and to import a `modelDescription.xml` file. These three actions are also traced. The exported or imported FMU or the imported `modelDescription.xml` file will also be placed under version control in the Git repository. Currently 20-sim does not support tracing the export of a tool-wrapper FMU.

**OpenModelica** OpenModelica is an open-source modeling and simulation tool which uses the Modelica language (Fritzson et al., 2018). Traceability support in OpenModelica is very similar to the one implemented in 20-sim. After an initial configuration of the Git repository and traceability daemon, the actions for saving a model, import of a `modelDescription.xml` file and export of an FMU are traced without further user interaction. Traceability in OpenModelica is described in more detail in (Mengist et al., 2017).

**Overture** In Overture, which is a modeling tool using the Vienna Development Method (Larsen et al., 2010), traceability is implemented as an additional package (as a `.jar` file), that can be downloaded from the GitHub page [12]. This package extracts traceability information from the Git repository, where the current Overture project is stored in. It can be either triggered manually, or simply added to a Git post-commit hook, to send new traces to the daemon after the user commits the changes to the model to the repository. Similar to Modelio, this way of extracting traceability messages from the Git repository is useful if traceability has not been used since the start of the project.

[12] see `https://github.com/overturetool/intocps-tracability-driver/releases`

### 5.4 RT-Tester

RT-Tester is a tool for test automation and model checking. It records the following traceability actions: Define test model, define test objectives, run test, define model-checking model, define continuous time abstraction, run model-checking query.

There is no need for the user to configure these operations, because per default valid settings (for the INTO-CPS Application) are used.

### 5.5 INTO-CPS Application

The INTO-CPS Application is the graphical user interface for configuring and running co-simulation scenarios, design-space exploration, test automation and model checking (Larsen et al., 2016b). It records the following traceability actions: Multi-model creation, Co-Simulation configuration creation, run simulation.

These actions are automatically recorded once the user creates a multi-model from an exported SysML configuration diagram, generates a Co-Simulation configuration from a multi-model, or modifies these configurations. Finally, the start of a simulation run is also recorded.

## 6 Queries and Visualisation

In order to bring a benefit to the user, the traceability data not only needs to be recorded, but also analysed and presented in a way that is helpful to the user. The tools therefore must have a way of querying the database, for specific information, such as relations between requirements, models, test results, users or simulation results.

The results from these queries are displayed within the INTO-CPS Application as lists, separated between different categories (FMUs, Users, Simulations, Requirements), as discussed below in Section 6.2. These categories can be extended and minimized, to present a neatly arranged view to the user. Additionally, for expert users that have a good understanding of the underlying structure, and that are proficient in generating queries to the database, it is possible to manually enter queries to search the traceability database, using the Cypher query language (see Section 6.1 below).

While there is plenty of research on traceability in software or systems engineering, only few industry standard tools implement traceability. One of them, IBM Doors Next Generation, is among the most popular tools (Winkler and Pilgrim, 2010), which displays traceability relations between requirements on different levels (e.g. high-level requirements and their refinements) as trees or lists [13]. Another common way of displaying traceability relations is the matrix view, which shows the relations between different artefacts in a 2-dimensional table. However, due to the heterogeneity of the different artefact types (requirements, models / FMUs, simulation results, configuration files etc.), the matrix view is not implemented in the context of INTO-CPS. Another standard way of presenting links is the graph view, where the different artefacts and their relations are shown in a graph. This is possible using the built-in Neo4J interface which allows browsing the complete graph. In principle, however, the openness of the INTO-CPS tool-chain allows for creation of new views, if they are required by a specific use-case.

### 6.1 Cypher query language

The Neo4J database uses a query language called *Cypher*. This language uses ASCII syntax to represent nodes and relations. Nodes are surrounded by parentheses "(" and ")", and relationships are identified by square brackets "[" and "]". More information can be found on [14]. In a study by Rath et at, graph query languages such as Cypher were also identified as suitable for requirements traceability (Rath et al., 2017).

### 6.2 Implementation in the INTO-CPS application

For representation of the traceability links to the users, pre-defined queries were integrated to the INTO-CPS Application. They allow the user to search for different artefacts and relations between artefacts. The user interface is identical to the rest of the INTO-CPS Application, which lowers the entry barriers for users. The search queries generate lists of items, which can be minimized to keep an overview of all the presented data.

The following queries are implemented in the INTO-CPS Application to allow for an easy usage.

1. FMUs: Query the database for all requirements that are related to a specific FMU.

2. Users: Query the database for all activities and artefacts that are related to a specific user.

3. Simulations: Query the database for all the Co-Simulation results that are associated with a multi-model.

4. Requirements: Query the database for test results that are linked to requirements.

Right-clicking on the "Traceability" button on the left-hand side of the window opens a context menu (see Figure 6). Clicking on "Trace Objects" shows the overview of the different queries, that can then be extended and minimized.

**FMU and requirements** This query is ran in two steps. The first query lists all the FMUs that are stored in the database (e.g. after FMU export in Overture, 20-sim or OpenModelica, see Section 5.3.)

In the Cypher language (see previous Section), the first query is achieved with the following command:

[13] see also `https://jazz.net/library/article/88104`

[14] `https://neo4j.com/developer/cypher-query-language/`

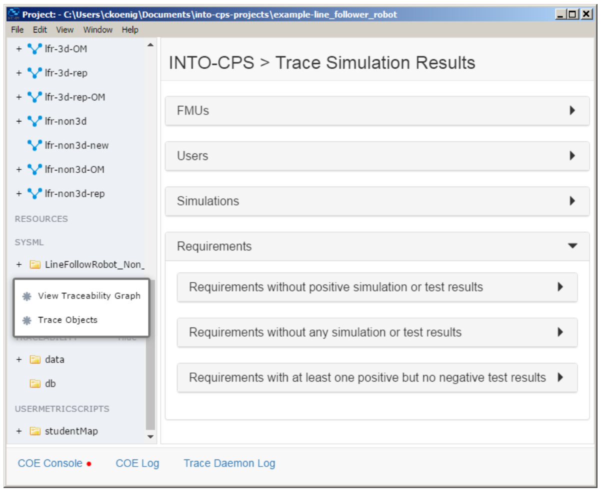

Project: - C:\Users\ckoenig\Documents\into-cps-projects\example-line_follower_robot
File
Edit
View
Window
Help
lfr-3d-OM
lfr-3d-rep
lfr-3d-rep-OM
lfr-non3d
lfr-non3d-new
lfr-non3d-OM
lfr-non3d-rep
RESOURCES
SYSML
LineFollowRobot_Non_
View Traceability Graph
Trace Objects
data
db
USERMETRICSCRIPTS
studentMap
INTO-CPS > Trace Simulation Results
FMUs
Users
Simulations
Requirements
Requirements without positive simulation or test results
Requirements without any simulation or test results
Requirements with at least one positive but no negative test results
COE Console
COE Log
Trace Daemon Log

**Figure 6.** Overview of the traceability queries in the INTO-CPS Application.

```
match(n{type:’fmu’})
return n.uri, n.path
```

In the next step, all requirements that are related to a specific FMU (<FMU_name>) are queried by the following command (note that "act" denotes an activity, and "elem" denotes an element):

```
match (act)<-
  [:Trace{name:"prov:wasGeneratedBy"}]-
  ({uri:’<FMU_name>’})
  -[:Trace{name:"oslc:satisfies"}]->
  (elem)
return
  elem.uri, elem.hash, act.time, elem.type
order by act.time desc
```

This returns all the requirements that are linked by the *oslc:satisfies* relationship to the particular FMU.

**Users, artefacts and activities** The INTO-CPS project aims explicitly the collaborative modelling, which means that multiple people are typically involved in the process. To support this, all the users and their actions can be traced. First, all users are queried from the database by using the following command:

```
match (usr{specifier:’prov:Agent’})
return usr.name, usr.uri
```

Next, all the artefacts that were influenced by a particular user (here identified by the URI, which contains the e-mail address *Agent.user@mail.com*) can be found by:

```
match (usr{uri:’Agent.user@mail.com’})<-
  [:Trace{name:’prov:wasAttributedTo’}]-
  (entity)
return entity.uri, entity.type
```

These artefacts are for example simulation results, FMUs, model description files or simulation configurations. A complete list of activities can be found in the schema under the enumeration for `ArtefactType`. In addition, all the activities performed by this user can be traced by:

```
match (usr{uri:’Agent.user@mail.com’})<-
  [:Trace{name:’prov:wasAssociatedWith’}]-
  (entity)
return entity.uri, entity.type
```

The activities are for example architectureModelling, modelDescriptionExport, simulationModelling and so forth. A complete list of activities can be found in the schema under the enumeration for `ActivityType`. Those activities reflect the activities as described in the ontology (see Section 3.2).

**Simulation results and files** To show which resources were used to generate simulation results, all the simulation results are queried first by the following command:

```
match (n{type:’simulationResult’})-
  [:Trace{name:"prov:wasGeneratedBy"}]->(m)
return n.uri, m.time, m.type
```

In the next step, all files that were used (i.e. that have the relation *prov:used*) to produce a particular simulation result (<Result_file>) are queried by the following command:

```
match({uri:’Entity.<Result_file>’})-
  [:Trace{name:"prov:wasGeneratedBy"}]->
  (simulation)-
  [:Trace{name:"prov:used"}]-
  (entity)
return entity.uri, entity.path, entity.hash
```

This query lists the FMUs, the configuration files and the log files which are related to this particular simulation result.

**Requirements and Test results** To relate requirements with the test results from RT-Tester (see Section 5.4), three different queries were implemented. Requirements without positive simulation or test results are queried by:

```
match (req{type:’requirement’})
  where not (req)<-
    [:Trace{name:"oslc:verifies"}]-()
return req.uri
```

This query indicates to the user all those requirements that have not been validated yet. Requirements without any simulation or test result are queried by the following command:

```
match (req{type:’requirement’})
  where not (req)<-
    [:Trace{name:"into:violates"}]-()
  and not (req)<-
    [:Trace{name:"oslc:verifies"}]-()
return req.uri
```

This query finds all requirements that have not yet been tested, and therefore were neither found to violate a requirement, nor to verify one. And finally, requirements with at least one positive but no negative test result are queried by:

```
match (req{type:'requirement'})
  where (req)<-
    [:Trace{name:"oslc:verifies"}]-()
  and not (req)<-
    [:Trace{name:"into:violates"}]-()
return req.uri
```

This finds those requirements that have been tested positively and can be seen as fulfilled, since no counter-example was found.

### 6.3 Evaluation of the current tool-chain

While the presented implementation in the different tools is mainly a proof of concepts, some conclusions for a general evaluation of the concepts can be drawn. The open architecture that relies on OSLC allows different tools, independent of their platform, to exchange data through well-established standards (such as HTTP or JSON). The format of the messages is published in a JSON schema file, which allows any other tool to verify its messages and connect to the workflow. The overhead in terms of the amount of data that is being sent and stored is considered to be little, as the JSON files are lightweight, and only relevant activities are recorded. The Neo4J database with its Cypher querying language allows flexible querying of the database, so that additional activities or tools could easily be integrated. The current solution with the traceability daemon running locally with the INTO-CPS application should in future be replaced with a centralised traceability service, which is however easy to implement. Therefore, the tool prototypes currently implement no dedicated error handling if no connection to the daemon is available. Furthermore, no mechanism to handle user authentification was implemented in our prototype. In summary, we consider our approach to traceability to be simple and yet powerful enough, due to its openness and little overhead. Especially for larger project in safety critical domains, where documentation of the relation between requirements, tests and validation results is required, the presented approach is promising.

## 7 Related Work

There are several existing techniques that aim to model traceability among heterogeneous domains. However, in a systematic literature review conducted in (Mustafa and Labiche, 2017), existing traceability approaches are either limited to a specific domain and problem, or they lack to specify traceability link semantics. For example, a Traceability Information Model (TIM) has been proposed in (Taromirad et al., 2013) for capturing heterogeneous artefacts in the context of the safety-critical systems domain where the TIM is defined on top of multi-domains using Ecore (Steinberg et al., 2009) metamodeling language. This model, however, lacks to specify how source and target artefacts can be linked and it is not clear how to classify a trace link or an artefact.

In (Hung Le Dang and Hubert Dubois and Sébastien Gérard, 2008), the authors proposed a traceability model for tracing heterogeneous artefacts (requirements model, design artefacts, and verification and validation model) in automotive systems. This solution only supports specific types of trace links and cannot be extended to support a new traceability link and various modeling languages.

The Traceability Metamodeling Language (TML) is also presented in (N. et al., 2009) for defining the syntax and semantics of traceability metamodels. It can support any type of traceability links including the newly created ones. However, only artefacts from Meta Object Framework (MOF)-based models can be traced and linked.

ModelBus (Hein et al., 2009) is a tool integration framework in the domain of system engineering, which uses traceability data such as timestamps for artefact creation and modification, and information about the creator of a specific artefact. But, it rather uses a one to one transformation for the integration of two tools since there is no common data format. As with ModelBus, our approach also builds the integration based upon Web Services. However, the usage of common data format and OSLC in our approach makes the integration more up to date to the latest industrial standard of managing traceability data in the whole development lifecycle in the model-based design of CPSs.

Compared to the above existing approaches, we presented a linked data-based approach to handle standardized traceability links for heterogeneous artefacts from different lifecycle modeling languages and simulation tools. The integration is based upon the standardized defined schema to ensure that all tools use the same format for sending their data, and an ontology was defined to describe the data that is collected at different events.

## 8 Conclusion

This paper presents the results of the traceability and model management efforts in the INTO-CPS project. A common architecture for traceability was designed, using a central database as repository for all traceability information, and a daemon to receive data from the different tools. A message schema was defined that ensures that all tools use the same format for sending their data, and an ontology was defined to describe the data that is collected at different events. Collaborative work involving multiple users is supported. All tools record the relevant actions, and the whole workflow of INTO-CPS is covered, with respect to traceability data. Collection of data is automated as far as possible, minimizing overhead for the users. Queries were implemented in the INTO-CPS application to return meaningful data to the user.

While the INTO-CPS tool-chain is well covered with respect to traceability, external tools are not supported. For example, if FMUs were generated in other tools, this is not listed in the traceability database. Therefore, methods for covering these artefacts coming from external tools, could be developed in the future. Since interface, ontology and format for the messages are public, support for external tools can easily be integrated by their developers. In principle, traceability should be used since the beginning of a project, such as CPS design. However, parsing of the Git repository, as it is enabled by Overture or Modelio, enables users to take advantage of traceability even though it was not used from the very beginning.

In the context of INTO-CPS, we enabled requirements traceability in the whole tool-chain of CPS design, from requirements collection, systems modeling, through physical and cyber modeling, down to co-simulation and test automation. This presents an important step in the true integration of the different tools that are used in CPS design.

## Acknowledgment

Support for the INTO-CPS project through the European Commission's Horizon2020 funding scheme (Grant Number 644047) is gratefully acknowledged.

## References

T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. The Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *Proceedings of the 9th International Modelica Conference*, Munich, Germany, September 2012.

Peter Fritzson, Adrian Pop, Adeel Asghar, Bernhard Bachmann, Willi Braun, Robert Braun, Lena Buffoni, Francesco Casella, Rodrigo Castro, Alejandro Danós, Rüdiger Franke, Mahder Gebremedhin, Bernt Lie, Alachew Mengist, Kannan Moudgalya, Lennart Ochel, Arunkumar Palanisamy, Wladimir Schamai, Martin Sjölund, Bernhard Thiele, Volker Waurich, and Per Östlund. The openmodelica integrated modeling, simulation and optimization environment. In *Proceedings of the 1st American Modelica Conference*. Modelica Association and Linköping University Electronic Press, October 2018. doi:10.3384/ecp18154206.

Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space (1st edition). Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool, 2011. doi:10.2200/S00334ED1V01Y201102WBE001.

Christian Hein, Tom Ritter, and Michael Wagner. Model-driven tool integration with modelbus. In *Proceedings of the 1st International Workshop on Future Trends of Model-Driven Development*, pages 35–39. SciTePress, May 6-10 2009. doi:10.5220/0002174800350039.

Hung Le Dang and Hubert Dubois and Sébastien Gérard. Towards a traceability model in a marte-based methodology for real-time embedded systems. *Innovations Syst Softw Eng*, 4 (3):189–193, 2008. ISSN 1614-5054.

Peter Gorm Larsen, Nick Battle, Miguel Ferreira, John Fitzgerald, Kenneth Lausdahl, and Marcel Verhoef. The Overture Initiative – Integrating Tools for VDM. *SIGSOFT Softw. Eng. Notes*, 35(1):1–6, January 2010. ISSN 0163-5948. doi:10.1145/1668862.1668864. URL `http://doi.acm.org/10.1145/1668862.1668864`.

Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, Peter Fritzson, Joerg Brauer, Christian Kleijn, Thierry Lecomte, Markus Pfeil, Ole Green, Sylianos Basagiannis, and Andrey Sadovykh. Integrated tool chain for model-based design of cyber-physical systems: The into-cps project. In *2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data)*, Vienna, Austria, April 2016a. IEEE. http://ieeexplore.ieee.org/document/7496424/.

Peter Gorm Larsen, Casper Thule, Kenneth Lausdahl, Victor Bandur, Carl Gamble, Etienne Brosse, Andrey Sadovykh, Alessandra Bagnato, and Luis Diogo Couto. Integrated Tool Chain for Model-Based Design of Cyber-Physical Systems. In Peter Gorm Larsen, Nico Plat, and Nick Battle, editors, *The 14th Overture Workshop: Towards Analytical Tool Chains*, pages 63–78, Cyprus, November 2016b. Aarhus University, Department of Engineering. ECE-TR-28.

Yang Li and Walid Maalej. Which traceability visualization is suitable in this context? a comparative study. In Regnell B. and Damian D., editors, *Requirements Engineering: Foundation for Software Quality. REFSQ 2012*, volume 7195 of *Lecture Notes in Computer Science*, pages 194–210. Springer, Berlin, Heidelberg, 2012.

Frank Manonla and Eric Miller. RDF Primer. W3C Recommendation. World Wide Web Consortium. `https://www.w3.org/TR/2004/REC-rdf-primer-20040210`, 2004. [Online; accessed 10-January-2020].

Alachew Mengist, Adrian Pop, Adeel Asghar, and Peter Fritzson. Traceability support in openmodelica using open services for lifecycle collaboration (oslc). In *Proceedings of the 12th International Modelica Conference*. Modelica Association and Linköping University Electronic Press, May 2017. doi:10.3384/ecp17132823.

Nasser Mustafa and Yvan Labiche. The need for traceability in heterogeneous systems: A systematic literature review. In *41st International Computers, Software and Applications Conference*. IEEE, July 4-8 2017. doi:10.1109/COMPSAC.2017.237.

Drivalos N., Kolovos D.S., Paige R.F., and Fernandes K.J. Engineering a dsl for software traceability. In Gašević D.and Lämmel R. and Van Wyk E., editors, *Software Language Engineering*, volume 5452 of *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, pages 151–167. Springer, Berlin, Heidelberg, 2009.

Open-services.net. Open services for lifecycle collaboration – lifecycle integration inspired by the web. `http://open-services.net`, 2008. [Online; accessed 10-January-2020].

Michael Rath, David Akehurst, Christoph Borowski, and Patrick Mäder. Are graph query languages applicable for requirements traceability analysis? In *REFSQ Workshops*, 2017.

Leonard Richardson and Sam Ruby. *RESTful Web Services (First ed.)*. O'Reilly, 2007.

Houbing Song, Danda B. Rawat, Sabina Jeschke, and Christian Brecher. *Cyber-Physical Systems: Foundations, Principles and Applications*. Academic Press, Inc., Orlando, FL, USA, 2016. ISBN 0128038012, 9780128038017.

David Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley, Boston, MA, 2009.

Masoumeh Taromirad, Nicholas Matragkas, and Richard F. Paige. Towards a multi-domain model-driven traceability approach. In *7th International Workshop on Multi-Paradigm Modeling co-located with 2013 ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 27–36. MPM@MoDELS, September 2013.

Stefan Winkler and Jens Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling (SoSyM)*, 9(4):529–565, 2010.

# Parameter Estimation Methods for Fault Diagnosis using Modelica and FMI

Ahmad Alsaab[1] Morgan Cameron[2] Colin Hough[1] Purna Musunuru[3]

[1]ESI UK, UK, {ahmad.alsaab, colin.hough}@esi-group.com
[2]ESI Group, France, morgan.cameron@esi-group.com
[3]ESI US R&D, USA, purna.musunuru@esi-group.com

## Abstract

We compare a number of different methods for estimating model parameters based on external stimuli. We examine the trade-offs between the different methodologies with respect to the modelling effort necessary to implement them and the granularity of the estimation obtained. In implementing these methods, we utilize Modelica and FMI.

As an application we show how these methods can be combined with component fault modes to provide effective real-time estimates of the health of a physical asset based on thermal sensor data. In particular, we contrast the effectiveness of the different estimators in predicting the degree and location of fault.

*Keywords: Parameter estimation, parameter tuning, fault diagnosis, Modelica, FMI*

## 1 Introduction

Parameter estimation methods are applicable to scenarios that are describable by a physical or data-driven models. Such scenarios include virtual testing, virtual commissioning, fault identification and control optimization. Parameter estimation is also a vital method for tuning parameters in control systems in order to produce robust control (Astrom, 1994).

Typically, physical products are designed to meet predefined specifications. Prototypes of these products are built and tested in an effort to confirm their behavior in standardized, well-defined conditions. Today, by building a virtual prototype, (virtual) testing can be done sooner in the design cycle and quicker. In this way, virtual testing supports faster iteration of the re-engineering of the physical product.

Beyond the design cycle, it is becoming increasingly important to leverage these methods in order to make informed decisions about the condition of the product in-operation. As the digital representation of the physical asset is updated in real-time using sensor data from instrumented components it becomes possible to optimize the asset's performance and identify possible sources of failure.

Likewise, we can imagine a scenario in an instrumented manufacturing plant where application of these methods would allow flexible adjustment of operating parameters to optimize production. In the same way, faults could be detected by comparing estimated operating parameters to expected (nominal) values corresponding to normal operating conditions.

We can divide methods used for condition estimation of a physical asset into two broad categories: those that use only data, and those that utilize a physical model of the system.

Data-driven methods are applicable when there is an abundance of labelled historical sensor data.

Model-based methods, such as state estimators, are particularly applicable when there is a paucity of available labelled historical sensor data. Furthermore, as they relate the fault to a physical component in the model, they allow us to identify more easily the root cause. The location of the fault in the model can be directly associated with a component. In contrast, with a purely data-driven approach we can detect the presence of a fault, but not its location.

In this paper, we present a comparison of different parameter estimation methods and exemplify how they can be implemented using Modelica and FMI.

## 2 Estimation methodology

A problem that is often encountered when employing standard parameter estimation methods is that they may require a mathematical model. With models constructed using Modelica, for example, the model of the physical system is typically constructed using connected components. In this context, using standard parameter estimation methods, requires that the mathematical equations of the system are modified to directly compute parameter values. Often this is difficult, especially if the user does not have access to such a model or the strong domain background to construct one. Moreover, some components may be black boxes, in which case this kind of modification is impossible.

Using a different approach such as the particle filter method (Liu, 1998), we can extend the scope to black box-type models without requiring knowledge of the mathematical details of the system. To avoid being required to modify a Modelica model in order to estimate its parameters, for example, we can use standardly-available tools to convert the model to an

FMU and then apply these methods to perform estimations.

Other approaches each have their own disadvantages. Kalman filters, for example, require knowledge of the governing equations of the system. Furthermore, the approach is limited to linear systems (Kalman, 1960). Extended Kalman filters do not require the system to be linear, but do require knowledge of the mathematical model/matrix of the system (Julier, 2004; Cocho *et al,* 2017).

Particle filters do not have these disadvantages. They can operate with nonlinear systems without knowledge of the governing equations and can be used to estimate parameters of the system without prior knowledge.

We can distinguish between the applicability of estimators by categorizing them as either “global” or “local”. We use the term “local estimator” for methods that are applied inside the model and require knowledge of the mathematical equations of the components inside the system. Furthermore, each physical component in the system for which you want to estimate its parameters requires an instance of the estimator. In contrast, we can use the term “global” estimator to refer to a method such as a particle filter that is applied outside the model and can estimate all required parameters of the system, treating the model as a black box.

A number of studies have already been undertaken to develop parameter estimation methodologies using Modelica or FMI. An optimization library Ceres (Agarwal, 2012) was used to develop a parameter-estimation framework for FMUs, exemplified with the estimation of parameters of a delta robot. By posing the problem as a non-linear least squares problem depending on the error between the measured output and estimated output of a simulated FMU. An Extended Luenberger Observer (Bortoff, 2014) was implemented for estimating heat flows and heating load using FMUs. In both these studies, the estimation method required knowledge of the Jacobian of the system, unlike some of the methods described in the present paper, which require no a priori knowledge of the Jacobian

Similarly, FMI has been used in conjunction with the unscented Kalman filter (UKF) as a nonlinear parameter estimator. Unlike the non-linear least squares optimization methods, the UKF estimator does not require knowledge of the Jacobian of the model. Nevertheless, the UKF approximates the Gaussian distribution of the state by using a set of points called sigma points, while, the particle filter algorithm can approximate any arbitrary distribution (Bonvini, 2014).

In (Videla, 2008; Brembeck *el at*, 2014; Brembeck, 2019), three different variants of the Kalman filter (extended Kalman filter, unscented Kalman filter and ensemble Kalman filter) were implemented to estimate the system parameters of a Modelica model. In all cases, the estimators are “global” in nature, utilizing a stand-alone executable version of the model was used to represent the model.

A number of open source tools have been developed for parameter estimation using FMI such as ModestPy (Arendt *et al*, 2018), RaPId (Vanfretti *et al*, 2016) and Optifmus (Bonilla et al, 2017). These tools are configurable to allow the user to select the desired estimation algorithm, either by using one of the provided algorithms, or by incorporating new, user-implemented estimation algorithms. In the present paper, we do not focus on a particular toolkit, rather we describe a set of methods appropriate for state estimation. We note that the extensible nature of those toolkits means that any of the estimation methods described here could be incorporated into those toolchains.

An extension to RaPId was developed (Bogodorova *et al*, 2017) to implement an extended particle filter method. That method takes a similar approach to the “global” particle filter algorithm we present here. However, the architecture of the RaPId toolbox, isolates the estimation algorithm from the Modelica model itself, thereby precluding the implementation of a “local” variant of the algorithm such as the one we present in this paper.

## 2.1 Local estimators

In the present context, we use the term “local estimator” to refer to a method that can be implemented directly inside an individual Modelica component, without requiring the use of any other software tool for implementation. In a local estimator, the estimation takes place over the course of a single simulation.

Focusing on a single element of the whole system is easy because of the componentized nature of Modelica models; we can estimate parameters for a single component without having to construct an estimator for the whole system.

In Modelica, parameters cannot change over the course of a simulation. It is therefore desirable to restructure the model such that the parameters to be estimated are replaced by (unknown) variables. In turn, these variables can be defined to have a dependence on (time-varying) inputs to the system. In this way, the estimated parameters can be driven by external stimuli (e.g. sensor data). This, in turn, though, means that to implement the estimator it is necessary to modify the mathematical description of the individual components for which we would like to estimate parameters. For this reason, this technique is more suited to individual components rather than the whole system. Moreover, because of the correlations between components induced by connections in Modelica, it is difficult to implement a single estimator for connected sets of components.

One attraction of Modelica-based approaches is that we can imagine developing libraries of self-tuning (individual) components where parameters will adjust automatically based on data from other physical components.

## 2.2 Global estimators

Sometimes we may want to estimate multiple parameters (in multiple components) simultaneously, for example when estimating the degree of fault in multiple faulty components. In this case, we use the term "global estimator" to refer to any method which takes into account the whole system and does not require modifications to the governing equations. In fact, in some "global" estimation algorithms, no knowledge of the equations is even necessary.

A global estimator can run a simulation multiple times, taking as input the result of a previous iteration for initial values of state variables or parameters.

# 3 Tuning parameters

When a fault occurs in a physical system it can be interpreted as a change in the parameters of the model. The fault can be recognized by monitoring the difference between the output of the model and measured data. The value or the degree of fault can be determined by retuning the parameters of the system so that the new parameters give the same output response as the fault. In this work, we consider three kinds of parameter-tuning methods, specifically:

1. Direct Calculation
2. Least Square Method
3. Particle Filter Estimator

## 3.1 Direct calculation

In a Modelica model, the parameters define values which stay constant during the solution (simulation) of the model (Modelica Association, 2012). To solve the model, the number of equations must be equal to the number of (time-varying) variables. As an example, let us take a system of two springs and two masses (Figure 1).

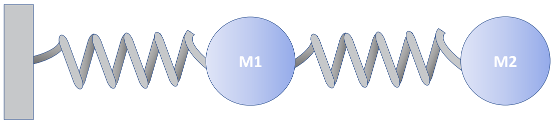


**Figure 1.** Two springs and masses system.

The model of the system is described by four parameters and six states or variables. The model parameters are the two masses M1 and M2 and the stiffnesses of the two springs K1, K2 respectively. The system states are the displacements of the masses $x_1$ and $x_2$, velocities $v_1$and $v_2$, and accelerations $a_1$and $a_2$. The outputs of the system are the displacement of the masses.

If the displacement of the M2 is measured, M1 can be calculated by declaring M1 as a time-varying state variable instead of a parameter and changing the declaration of $x_2$ to an input. In this way, the number of variables is kept equal to the number of the equations.

To test this method, a simulation was run from t=0s to t=10s, in which the value of the mass M1 was changed from 5kg to 2kg at t=5s (Figure 3). It can be seen from (Figure 2) that, as the vibration signal changes, the estimator was able to capture the change in the system parameter.

This estimator is very easy to implement, but it is noted that by construction, there needs to be a connected reference signal/external stimulus (e.g. sensor data source) per parameter to be estimated. Furthermore, it is very sensitive to noise in these signals.

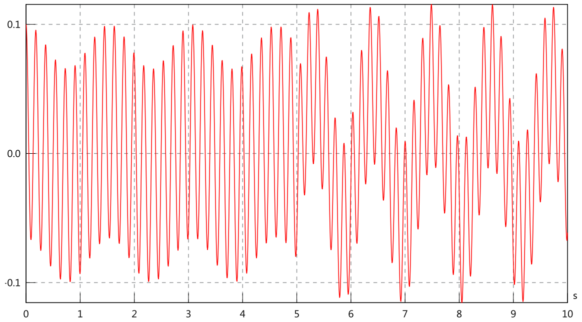


**Figure 2.** Displacement of M2.

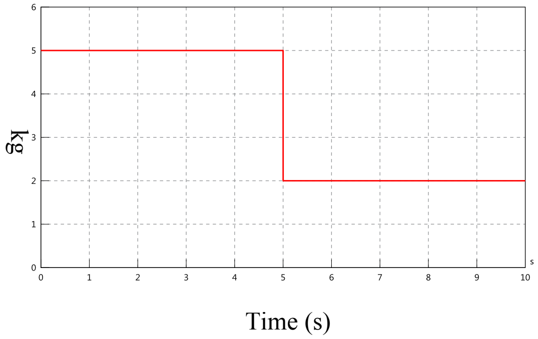


**Figure 3.** Mass of M2 as a function of time (reference signal).

## 3.2 Recursive Least Squares Estimation

The recursive least squares method (RLS) can be used on-line and off-line to estimate parameters of static and dynamic linear systems (Watson, 1967) such as the two masses and two springs system seen above. If there is a linear system described as follows:

$$y = X^T A = a_1 x_1 + a_2 x_2 \dots \dots + a_n x_n \qquad (1)$$

where y is the output of the system, x is the state of the system and A is the parameters of the system.

The parameters of the system can be estimated in RLS using the following set of equations:

$$K_j = P_{j-1}X_j\left(1 + X_j^T P_{j-1}X_j\right)^{-1} \tag{2}$$

$$P_j = P_{j-1} - K_j X_j^T P_{j-1} \tag{3}$$

$$\hat{A}_j = \hat{A}_{j-1} - K_j\left(X_k \hat{A}_{j-1} - y_j\right) \tag{4}$$

Where j the current sample time, $K$ is the estimator gain, $P$ is the covariance matrix, $\hat{A}$ is the parameters of the system. The subscripts $j$ denotes the value of that variable at that (time) iteration.

To estimate the value of the mass, when a connector class is used, the mass has two connectors *ctr1* and *ctr2* to connect with the springs. The connector *ctr1* has four variables: the force *F*, a flow variable between components, and the displacement *x*, a potential variable, as well as the velocity *v* and the acceleration *a*.

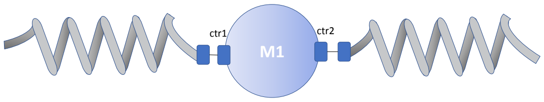


**Figure 4.** Mechanical connectors of mass.

$$a = \frac{d(v)}{dt} \tag{5}$$

$$v = \frac{d(x)}{dt} \tag{6}$$

$$ma = (F1 + F2) \tag{7}$$

So, to estimate the model mass of the object M1 using the RLS the relevant part of the model can be written in Modelica as:

```
ctr2.x=ctr1.x; // Displacement of ctr1 is
equal to displacement of ctr2
x=ctr1.x; // Displacement of the mass is
equal to displacement of ctr1
F=ctr1.F+ctr2.F; // Force on the mass is
equal to sum of the forces at connectors.

when sample(0, tSample) then
   P = pre(P) - pre(P)^2*ctr1.a^2(1+
pre(P)*ctr1.a^2)^-1;
   K=pre(P)*ctr1.a*(1+ pre(P)*ctr1.a^2)^-
1;
   m=pre(m)-1*K*(pre(m)*ctr1.a-F);
end when;
```

Where x is the displacement, F is the force, P is the covariance in (3), and ctr1 and ctr2 are the mechanical connectors shown in Figure **4** and tSample is a (Real) parameter representing the sample rate at which to apply the estimation.

Figure 5 shows the estimated value of the mass, where the RLS algorithm took approximately two seconds to compute the real value of the mass.

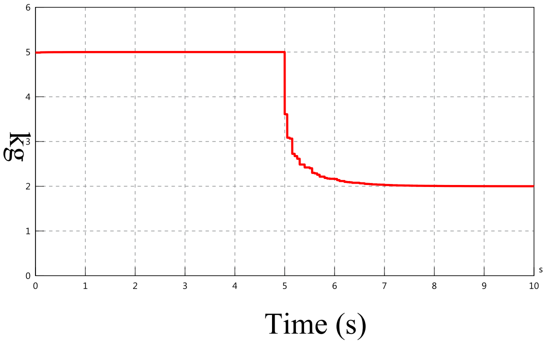


**Figure 5.** Estimated mass of M2 as computed by RLS algorithm.

This method is easy to implement, and is robust against sensor noise, but is only applicable to linear systems.

## 3.3 Particle Filter Estimation

As well as estimating the states of a system, the particle filter algorithm can also be used to estimate its parameters. Unlike the recursive least squares estimator, this algorithm is suitable for estimating the parameters of both linear and nonlinear systems (Moral *et al,* 2012; Gordon *et al,* 1993).

The particle filter uses the results of multiple concurrent simulations to produce an instantaneous estimate of the value(s) of parameters.

In this scheme, selection criteria are used that weight the uncertainty in the model versus the uncertainty in a reference signal. These selection criteria are used to weight the outputs of each of the concurrent simulations. These outputs are in turn used to run another set of simulations, updating parameter values and initial conditions accordingly.

The particle filter algorithm mainly consists of four steps namely initialization, prediction, updating and resampling. In the initialization step, the set of particles is created, where each particle includes initiation guess of the parameters. In the prediction step, for each particle, the FMU is run after assigning the parameters of the system with the particle's parameters. The updating step uses the residual, i.e. the error between the outputs of the summation and the measurement, to give an importance weight to each particle. In the last step, the resampling step, the parameters stored in each particle are redistributed depending on their weights. This algorithm is represented as a flow chart in Figure 6.

These stages were implemented in Modelica for the mass-estimation example previously described.

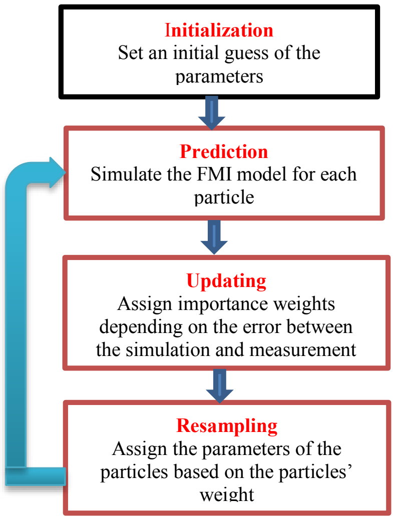

Initialization
Set an initial guess of the parameters
Prediction
Simulate the FMI model for each particle
Updating
Assign importance weights depending on the error between the simulation and measurement
Resampling
Assign the parameters of the particles based on the particles' weight

**Figure 6.** Particle filter algorithm.

To test this implementation, the particle filter algorithm was executed using 50 particles. The results of the estimation of the mass are shown in Figure 7. It was observed that the particle filter took less than one second to adapt the estimate to the correct value of the mass.

It is noted that the particle filter algorithm is an iterative method involving multiple simulations of the whole model. In the case of large models, or indeed large parameter sets, this could result in CPU-intensive calculations that could limit the method's applicability in real-time applications. A detailed investigation of this aspect is considered to be outside of the scope of this paper.

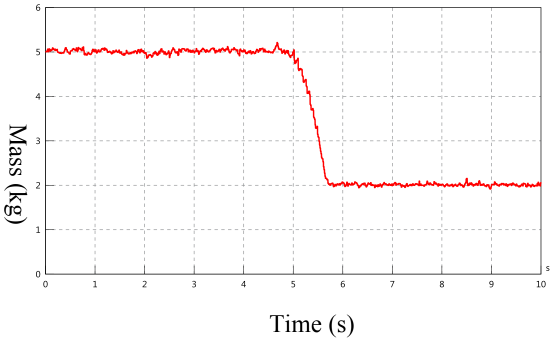


**Figure 7.** Estimated mass of M2 as computed by the particle filter algorithm

So far, the three tuning methods considered were implemented using Modelica by explicitly changing the mathematical model of the component and replacing it with the respective estimator algorithm.

While the Modelica implementation of the particle filter algorithm was seen to work well for simple models, we observed that the computation time increased significantly for more complex models. There is not a standardized framework defined in the Modelica specification for the parallel execution of Modelica models. This makes the parallelization of models a proprietary solution, that is tool-dependent (Modelica Association, 2012). For this reason, we developed an alternative implementation in Python that utilizes the easy conversion of a Modelica model to an FMU, to allow execution of simulations concurrently. Our implementation makes use of the PyFMI framework to manage interaction with the FMUs (Andersson *et al*, 2016). An additional advantage of utilizing FMI, is that we can apply the implementation to FMUs generated by non-Modelica tools. Furthermore, with this implementation, we realize a global estimator that can be used to estimate parameters without changing the mathematical structure of the system.

To demonstrate this approach, we consider the model of a thermo-mechanical system (Figure 8). Here, the input mechanical energy, *Pin,* is converted to heating energy according to a loss rate *U*, while the heating energy is converted to an (effective) temperature through the heating capacity of the component material. The heating energy is in turn exchanged with the cooling system *Tc* subject to a thermal resistance. The model has four parameters, namely: loss rate *U*, heating capacity *C*, thermal resistance *Rth* and cooling temperature *Tc*. In order to emulate the effect of a fault in the real system, an artificial error/fault was produced by changing the loss rate. In this example, measured mechanical power from a real mechanical system, is connected to the model via *Pin.* Readings from a temperature sensor in the real system are used as a reference signal in order to measure the validity of the estimated value.

To test the implementation of the global particle filter estimator, we converted the Modelica model in (Figure 8) to an FMI 2.0 Co-Simulation FMU using SimulationX. Table 1 show the quantities that were exported as tunable parameters in the FMU.

**Table 1.** Parameters of the thermal model

| *Parameter Name* | *Value* |
|---|---|
| Thermal Capacitor (C) | 0.5 $J/K$ |
| Loss Rate ($U$) | 0.73 |
| Thermal Resistance ($R_{th}$) | 28.011 $K/W$ |
| Cooling Temerature ($T_c$) | $36^0$ |

Subsequently, the particle filter was applied to tune the parameters, thus quantifying the degree of fault in the system.

Figure 9 shows that without tuning of the loss rate parameter *U*, there is a significant difference between measured data and the output of the model.

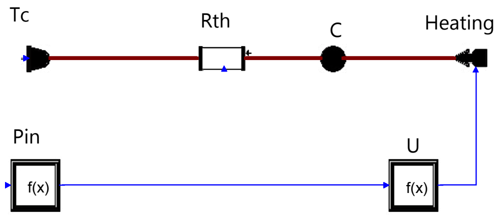


**Figure 8.** Thermal System. Pin is the input power, U, loss rate, C, Rth and Tc are the heating capacity, heating resistance and cooling temperature respectively.

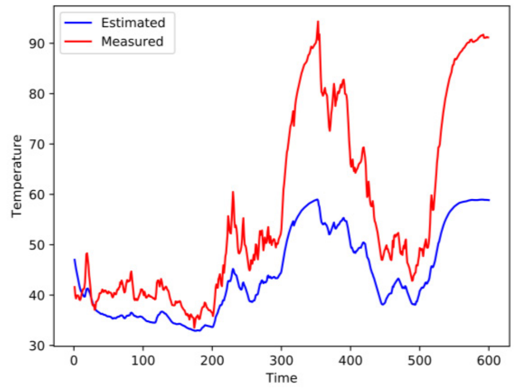


**Figure 9.** Estimated temperature of thermal model without tuning via particle filter (blue) compared to reference thermal signal (red)

When applying parameter tuning, the particle filter was seen to correct the estimated temperature to the measured temperature (Figure 10) by changing the value of the loss rate from 0.70% to 0.93% (Figure 11).

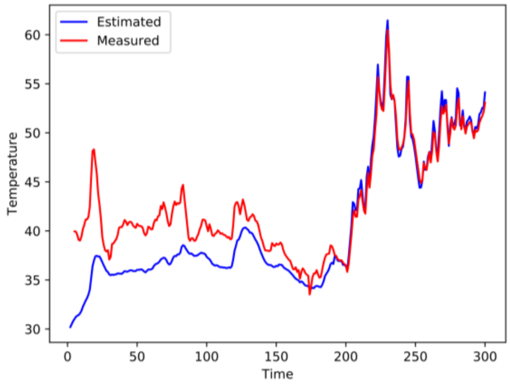


**Figure 10.** Estimated temperature of thermal model with tuning via particle filter (blue) compared to reference thermal signal (red)

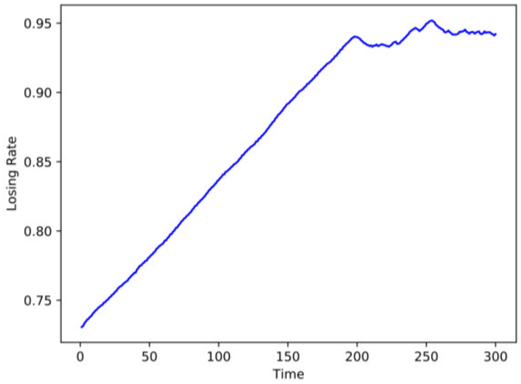


**Figure 11.** Tuned thermal loss factor, representing degree of fault in thermal system

## 4 Conclusions

In this paper we considered a number of different parameter estimation algorithms and how they could be implemented using Modelica and FMI. We examined the applicability of two different types of estimation methodology (local and global). In each case, we detailed the level of knowledge of the system necessary to implement the estimation method. We investigated the tradeoffs between the modelling effort and granularity of the estimation obtained.

We demonstrate how these methods can be applied to use external stimuli, in this case thermal sensor data, to obtain an estimate of the health of a real system.

## Acknowledgements

Funding for this work, through the Innovate UK WindTwin project, is gratefully acknowledged by the authors.

## References

S. Agarwal, K. Mierle, et al. Ceres solver. http:// ceres-solver.org, 2012.

C. Andersson, J. Åkesson, C. Führer. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface, volume LUTFNA-5008-2016 of Technical Report in Mathematical Sciences. Centre for Mathematical Sciences, Lund University, 2016.

K. Arendt , M. Jradi, M. Wetter, C. Veje. ModestPy: An Open-Source Python Tool for Parameter Estimation in Functional Mock-up Units. In Proceedings of the American Modelica Conference, 2018.

K.J. Astrom, B. Wittenmark. Adaptive Control (2nd. ed.). Addison-Wesley Longman Publishing Co., Inc., USA. 1994.

T. Bogodorova, L. Vanfretti, V. S. Perić and K. Turitsyn, "Identifying Uncertainty Distributions and Confidence Regions of Power Plant Parameters," in IEEE Access, vol. 5, pp. 19213-19224, 2017.

J. Bonilla, J. A. Carballo, L. Roca, M. Berengue. Development of an open source multi-platform software tool for parameter estimation studies in FMI models. Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, 2017.

M. Bonvini, M. Wetter, M.D. Sohn. An FMI-based framework for state and parameter estimation. In: Proceedings of the 10th International Modelica Conference, Lund, Sweden, pp. 647–656. 2014.

S. A. Bortoff, C. R. Laughman. An Extended Luenberger Observer for HVAC Application using FMI. Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019.

J. Brembeck, A. Pfeiffer, M. Fleps-Dezasse, M. Otter, K. Wernersson, H. Elmqvist. Nonlinear State Estimation with an Extended FMI 2.0 Co-Simulation Interface. In Proceedings of the 10th International Modelica Conference. Lund, Sweden, pages 53–62, 2014.

J. Brembeck. A Physical Model-Based Observer Framework for Nonlinear Constrained State Estimation Applied to Battery State Estimation. Sensors (Basel). 2019.

M. G. Cocho, O. Salgado, J. Croes, B. Pluymers, W. Desmet. Model-based virtual sensors by means of Modelica and FMI, Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, pp 337-344, May 15-17, 2017.

N. J. Gordon, D. J. Salmond and A. F. M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proceedings F - Radar and Signal Processing, vol. 140, no. 2, pp. 107-113, April 1993.doi: 10.1049/ip-f-2.1993.0015

S.J. Julier, J.K. Uhlmann. Unscented filtering and nonlinear estimation. Proceedings of the IEEE. 92 (3): 401–422, 2004. doi:10.1109/jproc.2003.823141.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME—Journal of Basic Engineering. 82(D): pp 35-45, 1960.

J.S. Liu, R. Chen. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association. 93 (443): pp1032–1044, 1998.

P. D. Moral, A. Doucet, A. Jasra. On Adaptive Resampling Procedures for Sequential Monte Carlo Methods. Bernoulli. 18 (1): 252–278, 2012. doi:10.3150/10-bej335.

Modelica Association. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, Version 3.3. Modelica Association, May 2012. URL: https://www.modelica.org/documents/ModelicaSpec33.pdf

L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, and F. J. Goméz-López. RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models. SoftwareX, Volume 5, Pages 144-149, 2016.

J. I. Videla, B. Lie. "Using Modelica/Matlab for parameter estimation in a bioethanol fermentation model". In: Proceedings of the 6th International Modelica Conference. Bielefeld, Germany, Mar. 3–4, pp. 287–299, 2008.

G. S. Watson. Linear Least Squares Regression. Ann. Math. Statist. 38 no. 6, pp 1679—1699, 1967.

# Nonlinear State Estimation with FMI: Tutorial and Applications

Christopher R. Laughman    Scott A. Bortoff

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

## Abstract

One of the key uses enabled by the functional mockup interface (FMI) standard is the ability to combine Modelica models governed by differential-algebraic equations with measurement data to systematically estimate unmeasured quantities in physical systems. While it is clear how this might be done in theory, many implementation challenges can make this difficult in practice. This paper provides a tutorial connecting the mathematical formulation of two different estimators, the extended Kalman filter (EKF) and the ensemble Kalman filter (EnKF), to an FMI-based Modelica implementation of these estimators. The efficacy of these methods are demonstrated on an example of a small motor model and a larger thermodynamic model of a building, and some of the advantages and disadvantages of this FMI-based approach to estimation are discussed, as well as limitations of FMI associated with constraint management for these estimation methods. The code for the motor example is publicly available and is attached to this publication.

*Keywords: observers, state estimation, extended Kalman filter, ensemble Kalman filter, functional mockup interface (FMI)*

## 1 Introduction

Information about unmeasured physical quantities is often desired when designing complex engineered systems to improve system control, implement performance monitoring, or perform fault detection and analysis. Data about such variables may not be available for a variety of reasons, such as sensor cost or the fact that quantities of interest may be very difficult or impossible to measure directly. As an example, measurements of the amount of heating or cooling energy delivered by an HVAC system to an occupied space in a building would be useful in the design of improved air temperature controllers, but accurate estimates of this variable are difficult to obtain because they depend on local temperature differences and airflow rates that cannot be easily characterized.

This problem is commonly addressed by combining sensor data with a model of the system, thereby leveraging knowledge about the system structure which cannot be directly inferred from a given set of sensor data. Though a wide variety of modeling approaches can be used depending on the type of system under study, models of physical systems often benefit from the use of equation-oriented tools such as Modelica (Modelica Association, 2017) because their physics-based structure allows the internal states to have a meaningful interpretation, and their first-principles construction tends to produce good extrapolative performance over high-dimensional range of potential operating conditions.

A variety of estimation and observer-based techniques, including the range of Kalman filters and particle filters, have been developed since the mid-twentieth century to solve the problem of estimating unknown system quantities of interest given a model and a set of observations. These methods can be more precisely described by considering a system model described by a set of ordinary differential equations

$$\dot{x} = \mathcal{M}(x,u,t;\mu) + \mathcal{W} \tag{1}$$

$$y = \mathcal{H}(x,u,t;\mu) + \mathcal{V} \tag{2}$$

$$z = \mathcal{G}(x,u,t;\mu), \tag{3}$$

where $\mathcal{M}$ is the forward model operator, $\mathcal{H}$ is the observation operator, $\mathcal{G}$ is the map from the states and inputs to the performance variables of interest, $x$ represents the system state, $u$ represents the system input including both the control inputs and the unmeasured or measured disturbances, $\mu$ represents the system parameters (which we will assume for the present work are known), and $\mathcal{W}$ and $\mathcal{V}$ represent the process and observation noise with covariances $Q$ and $R$, respectively. We assume that the physical system is governed by the model $(\mathcal{M},\mathcal{H})$, while the noise terms are included in our approximate model representation to describe model and measurement errors. In this context, these methods are designed to produce an estimate $x$ that minimizes a metric related to the error between the model output $y$ and the measured system output $y_m$, with the expectation that the estimated system state is sufficiently close to that of the plant that the performance variables will accurately describe the variables of interest.

Unfortunately, the Modelica language is not designed to implement these state estimation methods, as it cannot readily update the state vector for the compiled model to incorporate additional measurement information. The functional mockup interface (FMI) standard (Modelica Association, 2019) was thus created, in part, to enable the use of Modelica models in this setting. The co-simulation interface provides an efficient method to evaluate the right-hand sides of Equations 1-3 at a given time step and correct the state vector to assimilate updated sensor data, which enables the implementation of these state estimation methods on top of the existing Modelica models.

Early work in using FMI 1.0 to implement state estimation methods was presented by Brembeck et al. (2011), which demonstrates the use of weighted least-squares and Kalman filters to estimate the state-of-charge for a lithium-ion battery. This work was significantly extended by Brembeck et al. (2014) to cover a draft implementation of FMI 2.0 standard for co-simulation, and test implementations of an extended Kalman filter and a moving horizon estimator are demonstrated on an electric vehicle application. Brembeck (2019) further develops this work and implements a number of practical refinements on the state-of-charge estimator. Other recent work done by Vytvytskyi and Lie (2019) compares the performance of unscented Kalman filters and ensemble Kalman filters for state estimation in hydropower plants and finds that these methods work well in this application. The present paper is an extension of the work presented by Bortoff and Laughman (2019), in which we demonstrated the use of a model-exchange functional mockup unit (FMU) to construct an extended Luenberger observer (ELO) with output injection to estimate the heat capacity of a water-cooled fan coil unit in a building.

While this prior work demonstrates the potential and some of the capabilities of using FMI to construct estimators based on Modelica models, an engineer with an interest in implementing these methods is faced with the daunting task of understanding the details of the FMI standard, as well as the API for the standard in a given language (e.g., Modelica, Python, or C++), before an estimator can be implemented for a given FMU. Moreover, a host of practical issues must be addressed in the process of this implementation that depend on the type of estimation method that is used. Larger Modelica models (tens to hundreds of states) are also associated with a variety of challenges that are not encountered for smaller models.

In this paper, we provide a tutorial-oriented description of two different types of estimators: an extended Kalman filter (EKF) (Simon, 2006) and a stochastic ensemble Kalman filter (EnKF) (Evensen, 2009a), so that a reader might obtain an improved understanding of the theory and process behind the implementation of these methods using the FMI. The EKF represents a traditional Kalman filter formulation for a nonlinear model, while the EnKF represents a particle-based approach which is similar to the EKF, but seeks a reduced computational effort by avoiding the integration of the covariance matrix. Both of these methods are first implemented in this paper on a simple model of an electrical machine, and then on a much larger model that describes the thermofluid dynamics of a building application, to provide an indication of the differences inherent in using them for practical Modelica models. While we could conceivably instantiate the FMU via any available API (e.g., pyFMI, FMPy, or other tools), we choose to instantiate the FMUs back into Modelica using the Dymola 2020 compiler (Dassault Systemes, 2019) because of the extensive Modelica support of the FMI interface and because this framework can easily be used to interface these estimators to other Modelica models.

The remainder of this paper is organized as follows: in Section 2, we will describe the theory of both the EKF and the EnKF in the context of the simple motor example, and connect the mathematical description of these estimators to the code listings provided in this paper. A complete implementation of this motor example is also available and is attached to this paper. Issues relating to the practical implementation of these estimators, as well as the simulation results from their application to the motor problem, will be discussed in Section 3. These same estimation methods will then be used in Section 4 to estimate the cooling capacity on a larger building model. Conclusions and directions for future development and work will briefly be discussed in Section 5.

# 2 Tutorial: EKF & EnKF with FMI

## 2.1 Extended Kalman Filter

As described in (Simon, 2006), the Kalman filter updates the state of a system model whenever measurements are available, and is the optimal variance-minimizing algorithm for linear systems with Gaussian process and measurement noise. While this state estimation method is thoroughly documented in many textbooks, it is helpful to review it because it is closely related to both the EKF and the EnKF. Assume a linear system model

$$x_k = Ax_{k-1} + Bu_{k-1} + w_k \tag{4}$$

$$y_k = Cx_k + v_k, \tag{5}$$

where $w_k$ and $v_k$ are the zero-mean, independent, and identically distributed Gaussian process and measurement noise with covariances $Q$ and $R$. As a result we can characterize the statistical properties of the state by its mean and variance, e.g.,

$$\hat{x}_k = \mathbf{E}[x_k] \tag{6}$$

$$P_k = \mathbf{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T]. \tag{7}$$

The Kalman filter proceeds in two phases to update and correct the state at time $k$ given the state at $k-1$ and a set of measurements $y_k$. The first of these phases is referred to as the *forecast* step, which predicts the state at the next time step without the knowledge of any additional measurements, and the second phase is the *analysis* step, which corrects the state forecast using the received measurements. In the forecast step, we update the state from $k-1$ at time $k$ by propagating forward the state and the covariance via the original system model, e.g.,

$$\hat{x}_k^f = A\hat{x}_{k-1}^a + Bu_{k-1} \tag{8}$$

$$P_k^f = AP_{k-1}^a A^T + Q, \tag{9}$$

where the covariance matrix $P$ can be derived by explicitly calculating the variance from Equation 7. Now that the state of the system has been propagated forward during the forecast step, we can correct the state during the analysis step using the measurements. This formulation minimizes the trace of the estimation error covariance at each time step.

$$\hat{x}_k^a = \hat{x}_k^f + K_k(y_k - C\hat{x}_k^f) \tag{10}$$

$$P_k^a = (I - K_k C)P_k^f \tag{11}$$

$$K_k = P_k^f C^T (CP_k^f C^T + R)^{-1} \tag{12}$$

The state is corrected at each time step $k$ to optimally tradeoff between predictions of the model forecast and the information obtained from the measurements. The state will be biased towards the forecasts if the errors in the measurements are large, while the state will be corrected towards the measurements if the errors are small.

Because this estimator is designed for linear systems, it must be modified to work on the nonlinear models often studied in Modelica. The EKF represents perhaps the most straightforward modification of the basic Kalman filter equations for this purpose, which is to linearize the nonlinear system model at each time step, and correct the state based upon the predictions from the linearized model. Given the original system description in Equations 1-3, we can linearize the system at each time step $k$,

$$\dot{x} = \mathcal{M}_{(x,k)}x + \mathcal{M}_{(u,k)}u \tag{13}$$

$$y = \mathcal{H}_{(x,k)}x \tag{14}$$

where $\mathcal{M}_{(x,k)} = \partial\, \mathcal{M}/\partial x|_k$, $\mathcal{M}_{(u,k)} = \partial\, \mathcal{M}/\partial u|_k$, and $\mathcal{H}_{(x,k)} = \partial\, \mathcal{H}/\partial x|_k$. This model can then be discretized for a system with sample time $T_s$ using the matrix exponential, e.g., $A = e^{\mathcal{M}_{(x,k)}T_s}$, which yields the linear system at this operating point represented by Equations 4 and 5. Once in this discrete-time representation, the previous Kalman filter equations (8-12) can be implemented.

In order to describe the implementation of the EKF as explicitly as possible, we present a simple nonlinear model of a two-phase permanent magnet machine from Simon (2006) to provide context for the derivation of these modifications, and to provide a concrete example for which we can develop the EKF and EnKF code listings of Figures 1 and 2. This model consists of four coupled ODEs that describe the electrical and mechanical dynamics of the machine, e.g.,

$$L\frac{di_a}{dt} = -Ri_a + \omega\lambda\sin\theta + v_a \tag{15}$$

$$L\frac{di_b}{dt} = -Ri_b - \omega\lambda\cos\theta + v_b \tag{16}$$

$$J\frac{d\omega}{dt} = \frac{3}{2}\lambda\left(-i_a\sin\theta + i_b\cos\theta\right) - B\omega \tag{17}$$

$$\frac{d\theta}{dt} = \omega, \tag{18}$$

where $i$ represents electrical current, $v$ represents voltage, $R$ is the winding resistance, $L$ is the winding inductance, $\lambda$ is the flux linkage of the coil, $J$ is the rotational inertia of the shaft, and $B$ is the damping factor of the load.

In general, electrical variables can be observed easily and reliably, whereas the mechanical variables are more expensive to measure. We consequently assume that we have observations of the the input voltages $v_a$ and $v_b$, as well as the currents $i_a$ and $i_b$, but want to obtain estimates of the shaft speed $\omega$. This model can thus be written down in the form of Equations 1-3, where the state $x = [i_a\ i_b\ \omega\ \theta]^T$ and the input $u = [v_a\ v_b]$. Equations 15-18 are straightforward to implement as a model in Modelica; we chose to avoid the specification of the phase voltages as explicit inputs, and instead defined them as time-varying real variables.

We create an estimator from a Modelica model on the basis of the above theory by first exporting the underlying Modelica model (representing $(\mathcal{M},\mathcal{H})$) from a Modelica tool as an FMU, reimporting the same FMU back into Modelica as a co-simulation FMU, and then implementing the EKF by modifying the Modelica code that is autogenerated upon reimportation. The co-simulation format is needed because of the discrete-time formulation of the EKF. This autogenerated code contains a wide variety of helper functions (often encapsulated in their own package, such as `fmiFunctions` in Dymola) that are needed to interface with the FMI API and run the FMU in the Modelica tool, and which are the building blocks from which the estimators are built. These helper functions do not necessarily have the same API as that which is described in the FMI standard because they provide an interface between the specific tool and the functions defined by the standard. However, the functions used in the estimator are relatively straightforward, and would be expected to be found in most complete FMI implementations.

Two aspects of this reimported FMU are of particular note. First, the variables in the Modelica-instantiated FMU are referred to by 9-digit integer labels, rather than by their original names. The `modelDescription.xml` file included in the FMU lists the correspondence between these labels and their names, but use of particular variable names in the estimator requires explicitly re-establishing this correspondence in the estimator code (Brembeck et al., 2014). Second, though the derivative variables are separately enumerated in the XML file in the element `modelStructure`, the FMU does not maintain an easily parsed list of the integer labels corresponding to the state variables. The set of state variable integer labels can instead be obtained by importing the FMU in the model-exchange format for this express purpose, though this model-exchange FMU must be unloaded so that it can be reimported in the co-simulation format to create the estimator code. Once loaded, the initialization section of the model-exchange FMU contains a list of the state variables selected during compilation as well as their associated integer labels. Because the process of reading and correcting the state in the EKF requires the manipulation of these integer labels (which are often sequential for the state variables), this is of major importance when implementing the estimator.

The essential code implementing the EKF is provided in Figure 1, while the complete code is provided in an attachment to this paper. To make this code more readable, certain simplifications were adopted with the hope that they do not obfuscate the overall intent. Some boilerplate code, such as variable definitions, was eliminated when the information about the variable could be inferred from its context. In addition, some sections of code that were autogenerated during the FMU import were also commented out. Finally, integer labels were shortened to improve readability.

This code excerpt begins in lines 1-13 with the definition of the fmiFunctions section (`fmiF`), which is autogenerated upon FMU import, and the user-specified definition of a number of important variables for the EKF, such as the number of states and outputs `Nx` and `Ny`, the process noise covariance matrix $Q$, and the measurement noise covariance $R$. We also create a new set of inputs and outputs `y` and `yhat` (lines 15-16), since we will be reading the measurements of the actual plant that will be assimilated by the estimator in variable `y`, and we want to study the performance variables `yhat`. Once these variables are set up, the FMU enters a `when` loop that will step through the stimulation and is initialized in an autogenerated block of code represented by line 21.

Lines 23-41 create of the linearized forward model operator $\mathcal{M}_{(x,k)}$ and linearized observation operator $\mathcal{H}_{(x,k)}$. The `fmiGetDirectionalDerivative` function takes the directional derivative of the integer labels corresponding to the derivatives of the state variables (in the array of the second argument) with respect to the dot product of the integer labels corresponding to the state variables (the third argument) and the column of the identity matrix $I^{N_x \times N_x}$ corresponding to the number of the column (the fourth argument). While this function could be used to take the mixed derivative with respect to multiple variables, the array `IdentityMatrixA` is used in this case to construct the Jacobian by computing the gradient of the state vector with respect to each individual state variable. Note that $\mathcal{M}_{(x,k)}$ is of size $N_x \times N_x$, whereas $\mathcal{H}_{(x,k)}$ is of size $N_y \times N_x$, and the entries of the output vector in lines 37-38 are the same as the state vector because the states $i_a$ and $i_b$ are assumed to be measured.

Once these linearized Jacobians have been created for the current time step, we apply the correction terms calculated at the end of the last time step and step the simu-

```
1  import MSL.Math.Matrices.exp;
2  import MSL.Math.Matrices.inv;
3
4  package fmiF
5  ...
6  end fmiF;
7  public
8  parameter Integer Nx=4, Ny=2;
9  parameter Real Hz[1,Nx]=cat(2,
10   {{1}}, {{1}}, zeros(1,2));
11 parameter Real Q[Nx,Nx]=transpose(Hz)*Hz;
12 parameter Real R[Ny,Ny]=identity(Ny);
13 // {Other variable declaration code}
14
15 MSL.Blocks.Interfaces.RealInput y[2];
16 MSL.Blocks.Interfaces.RealOutput yHat[2];
17
18 algorithm
19 when {initial(),
20       sample(startTime, stepSize)} then
21 // {Initialization/slave-mode code}
22
23 for i in 1:Nx loop
24   dz := fmiF.fmiGetDirectionalDerivative(
25     fmi,
26     {560, 561, 562, 561},
27     {432, 433, 434, 435},
28     IdentityMatrixA[i,:]);
29   dF[:,i] := dz;
30 end for;
31
32 F := exp(dF*stepSize);
33
34 for i in 1:Ny loop
35   dh := fmiF.fmiGetDirectionalDerivative(
36     fmi,
37     {432, 433},
38     {432, 433, 434, 435},
39     IdentityMatrixC[i,:]);
40   H[:,i] := dh;
41 end for;
42
43 PPred := F*PCorr*transpose(F) + Q;
44 K := PPred*transpose(H)*inv((H*PPred*
      transpose(H)) + R);
45
46 // Apply state correction after 5 steps
47 if time >= (startTime + 5*stepSize) then
48   fmiF.fmiSetReal(fmi,
49     {432, 433, 434, 435}, xCorr);
50 end if;
51
52 // {Step forward with fmiDoStep()}
53
54 xPred := fmiF.fmiGetReal(
55   fmi, {432, 433, 434, 435});
56 yPred := fmiF.fmiGetReal(
57   fmi, {432, 433});
58 xCorr := xPred + K*(y - yPred);
59 PCorr := (identity(4) - K*H)*PPred;
60
61 // {Variable allocation code}
62 equation
63 yHat[1] = ’x[1]’;
64 yHat[2] = ’x[2]’;
```

**Figure 1.** Extended Kalman filter (EKF) code.

lation forward. The Kalman gain matrix and the forecast for the covariance matrix are calculated in lines 43-44, and the state correction is applied to the state integer labels in lines 46-50 by using `fmiSetReal`. The `if` statement in line 47 is used to allow the values of the covariance matrix to accumulate before applying the corrections to the state variables. Once the state vector has been corrected, the FMU is stepped forward to compute the forecast step using `fmiDoStep()` auto-generated code. After this step, the corrections to the state vector $\hat{x}$ and the covariance matrix $P$ for the analysis step are computed in lines 54-61 for application to the next cycle. Finally, we need to add two equations in line 63-64 to assign the outputs to be the state estimates that we want to examine.

## 2.2 Ensemble Kalman Filter

While the EKF is a popular estimation technique because it embodies a logical extension of the Kalman filter to nonlinear systems, it has been shown to have a few drawbacks. First, the entire covariance matrix $N_x \times N_x$ matrix $P$ must be integrated at each time step; while this does not present a burden for small systems, the quadratic growth of the size of this matrix with the length of the state vector poses significant computational barriers for large-scale problems involving thousands of states. Second, the development of the EKF as an extension of the linear KF uses a linearized equation to describe the propagation of the error covariance matrix. This may not be a valid assumption, and can result in unbounded linear instabilities of the error evolution (Evensen, 2009b).

An alternative approach created to address the size limitations of the EKF is called the ensemble Kalman filter, or EnKF (Evensen, 2009b). Rather than directly propagate all of the covariances forward at each time step, the EnKF uses a sequential Monte Carlo approach to propagate forward an ensemble of statistically sampled state vectors, or particles, through the system dynamics and directly estimate the covariance matrix from this distribution. This avoids the computation of the Jacobians for the forward model and the observation operator. This algorithm is summarized with a range of variants by Vetra-Carvalho et al. (2018), and will be briefly presented here.

As indicated above, the essential distinction between the EKF and the EnKF can be seen by examining the formulation of the covariance matrices. Whereas the EKF formulates the covariance matrix $P$ as

$$P_k = \mathbf{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T], \tag{19}$$

the EnKF uses the fact that it propagates forward an ensemble of particles to define the ensemble covariance matrices around the ensemble mean rather than the true mean, e.g.,

$$P_k = \mathbf{E}\left[(\hat{x}_k - \mathbf{E}[\hat{x}_k])\,(\hat{x}_k - \mathbf{E}[\hat{x}_k])^T\right]. \tag{20}$$

This allows us to approximate the error covariance matrices by using simple multiplications, rather than integrating the full set of differential equations for the covariance matrix forward at every time step. This method has been successfully used in numerical weather prediction models with more than $10^6$ states, which would not be generally feasible with an EKF.

Because the ensemble of particles evolves forward with time, it is also necessary that we formulate the problem in terms of the ensemble mean and the ensemble perturbations. Taking $X^a$ as $N_e$ particles sampled from a state space of $N_x$ so that $X^a$ has dimension $N_x \times N_e$, we construct the analysis mean and perturbation ensemble by

$$X^a = \overline{X}^a + X'^a. \tag{21}$$

As a result, the covariance matrix $P^a$ can be written as

$$P^a = \frac{X'^a (X'^a)^T}{N_e - 1}. \tag{22}$$

We summarize the construction of the filter as follows, emphasizing the similarity to the linear Kalman filter. The ensemble forecast is constructed by propagating the nonlinear forward operator and observation operator by a step, e.g. integrating forward the model equations

$$X_k^f = \mathcal{M}(X_{k-1}^a, u_{k-1}, t_{k-1}; \mu) \tag{23}$$

$$HX_k^f = \mathcal{H}(X_k^a, t_k; \mu) \tag{24}$$

where $HX_k^f$ indicates the integation of the nonlinear observation operator forward on the ensemble by one time step. We then calculate the output error covariance $HPH_k$ and the innovation term $C_k$ by computing

$$HX_k'^f = HX_k^f - \overline{HX}_k^f \tag{25}$$

$$HPH_k = \frac{HX_k'^f (HX_k'^f)^T}{N_e - 1} \tag{26}$$

$$C_k = (HPH + R)^{-1}\left(Y_k - HX_k^f\right) \tag{27}$$

We construct the ensemble perturbation matrix $X_k'^f = X_k^f - \overline{X}_k^f$ and compute the correction ensemble

$$X_k^a = X_k^f + \frac{1}{N_e - 1} X_k'^f (HX_k'^f)^T C_k \tag{28}$$

which completes the calculation of the analysis step for the EnKF.

As was the case for the EKF, much of the implementation of the EnKF using the FMU is relatively straightforward. Many of the specific modifications needed to implement the EnKF by using the autogenerated Modelica code pertain to the generation and propagation of the particles used to characterize the estimator's statistical properties, as well as the computation of the ensemble mean. While random number generation code from the Modelica Standard Library (MSL) (included in `Modelica.Math.Random` and `Modelica.Math.Distributions`) was used in the creation of the perturbations, separate implementations of the noise generation functions were required because the models in `Modelica.Blocks.Noise` could not be included in the algorithm sections of the FMU.

The EnKF code described in Figure 2, and which is also available in its entirety in the attachment, bears many similarities to the EKF code discussed previously. Many of the variables are first defined in lines 1-21, with some of the variable definitions commented out for the sake of brevity. The random number generation code imported from the MSL is abbreviated as `MSL*` for similar reasons, though these functions can be quickly found through a search. Two variables (`initState` and `nextState`) were also required to maintain and evolve the state of the random number generator. The inputs and outputs are defined in lines 20-21, and the model is initialized using the autogenerated code in lines 23-26.

The initial ensemble of particles is generated in lines 28-35 during the initialization phase of simulation by using the `MatrixPerturbation` function. This function takes a set of initial guesses for the state variables and creates $N_e$ perturbed instances of this $N_x \times 1$ state vector by adding zero-mean noise with a standard deviation based upon the order of magnitude of the initial guesses. This scaling of the particle perturbations is important, as perturbations that are too small will not provide sufficient diversity to estimate the covariance, while perturbations that are too large could result in incorrect model behavior or otherwise reduce the information provided by these initial guesses.

```
1  import distribution=MSL*.quantile;
2  import generator=MSL*.Xorshift128plus;
3  import MSL*.impureRandomInteger;
4  package fmiF
5  ...
6  end fmiF;
7  public
8  parameter Integer Nx=4, Ne=5, Ny=2;
9  parameter Real R[Ny,Ny]=[0.01, 0; 0, 0.01];
10 parameter Real[Nx] x_init={0, 0, 0, 0};
11 parameter Real[Nx,Ne] X_init=[x_init,
       x_init, x_init, x_init, x_init];
12 parameter Real mu=0, sigma=0.1;
13
14 MSL.Blocks.Noise.GlobalSeed globalSeed;
15 parameter Integer actualGlobalSeed=
       globalSeed.seed;
16 final parameter Integer localSeed=
       impureRandomInteger(
       globalSeed.id_impure)
17 Integer initState[generator.nState];
18 Integer nextState[generator.nState];
19 // {Other variable declaration code}
20 MSL.Blocks.Interfaces.RealInput y[Ny];
21 MSL.Blocks.Interfaces.RealOutput yHat[Ny];
22
23 algorithm
24 when {initial(),
25        sample(startTime, stepSize)} then
26 // {Initialization/slave-mode code}
27
28 if initial() then
29   // Create particle distribution
30   initState := generator.initialState(
31     localSeed, actualGlobalSeed);
32   (X, nextState) := MatrixPerturbation(
33     X_init, initState);
34   initState := nextState;
35 end if;
36
37 if time >= startTime + (5*stepSize) then
38   X := XCorr;
39 end if;
40
41
42 for i in 1:Ne loop
43   fmiF.fmiSaveFMUState(fmi);
44   fmiF.fmiSetReal(fmi,
45     {432, 433, 434, 435}, X[:,i]);
46   // {Step ensemble member with fmiDoStep}
47   X[:,i] := fmiF.fmiGetReal(fmi,
48     {432, 433, 434, 435});
49   HX[:,i] := fmiF.fmiGetReal(fmi,
50     {432, 433});
51   if i<Ne then
52     fmiF.fmiRestoreFMUState(fmi);
53   else
54     continue;
55   end if;
56 end for;
57
58 // EnKF computations
59 X_bar := MatrixMean(X,1);
60 HX_bar := MatrixMean(HX,1);
61
62 for i in 1:Ne loop
63   X_prime[:,i] := X[:,i]-X_bar;
64   HX_prime[:,i] := HX[:,i]-HX_bar;
65 end for;
66
67 HPH:=1/(Ne-1)*HX_prime*transpose(HX_prime);
68 A := HPH + R;
69 (Y, nextState) := MatrixPerturbation(
70   [y, y, y, y, y], initState, sigma=0.01);
71 initState := nextState;
72
73 D := Y - HX;
74 C := MSL.Math.Matrices.solve2(A,D);
75 E := transpose(HX_prime)*C;
76 XCorr := X + (Ne-1)*X_prime*E;
77 yHat := HX_bar;
78
79 // {Variable allocation code}
80 equation
81 yHat[1] = ’x[1]’;
82 yHat[2] = ’x[2]’;
```

**Figure 2.** Stochastic ensemble Kalman filter (EnKF) code.

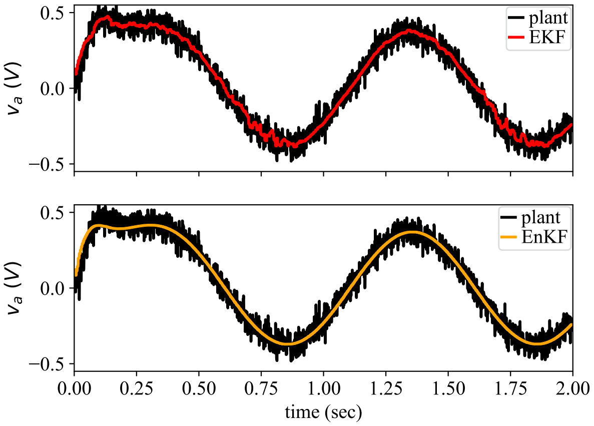


**Figure 3.** Estimated and measured phase A currents for the EKF and EnKF experiments.

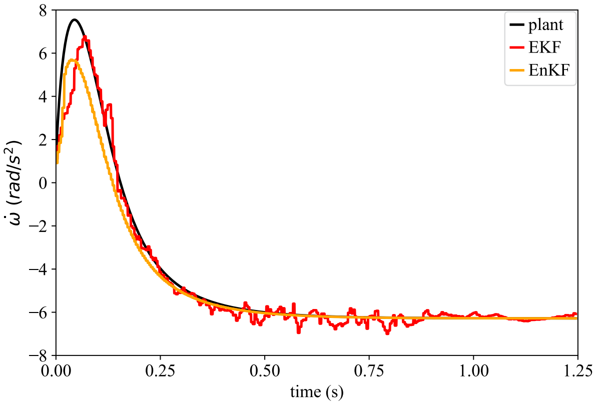


**Figure 4.** Shaft speed.

After creating this ensemble, the corrected state vector is applied to the system in lines 37-39; this is performed after 5 time steps as was the case for the EKF. Lines 42-56 then use this ensemble and the corrected state vector during model integration to evolve each particle forward according to the dynamics of the nonlinear model. This is accomplished by

1. Saving the state of the FMU using `fmiSaveFMUState`,
2. Setting the integer labels of the state variables to the values of the current particle,
3. Stepping the system dynamics forward with this particle using `fmiDoStep()`,
4. Saving the results of the integration step to the variables `X` and `HX`, and
5. Resetting the state of the FMU using `fmiRestoreFMUState` so the next particle can be integrated forward.

Note that the evolution of the model does not depend on the evaluation of the Jacobians, which can potentially reduce the amount of computation required. However, the requirement that each particle be individually stepped forward in time can also present significant computational requirements, depending on the implementation of the integrator and the relative scales of the state variables.

The analysis phase of the EnKF is completed in lines 59-77 of the EnKF example. This implementation follows Vetra-Carvalho et al. (2018) closely, and culminates in the computation of the corrected state vector in line 76. The outputs of interest are then assigned in lines 81-82 to complete the construction of this estimator.

## 3 Simple Motor System

The construction of these estimators was based upon a Modelica model of the simple motor described in Equations 15-18, with model parameters set to the values provided in Table 1. After generating a co-simulation FMU for this model, we saved the model into a separate folder that was expressly set up for the storage of FMUs, as FMUs cannot be stored within the package directory structure. This imported Modelica model was then duplicated and modified to create the EKF and the EnKF estimators, which remained distinct from the original imported FMU. Note that the path to the FMU DLL is stored as a text string in the autogenerated Modelica code, and must be changed manually if the FMU is moved after it is reimported.

We evaluated these estimators by creating a test model that included both the original Modelica motor model and the new estimator. Zero-mean noise with a standard deviation of 0.05 A was added to the motor current observations before feeding these signals into the estimator to evaluate the robustness of the estimates. In addition, the states of the initial motor model were each initialized to zero, but the states of the estimator model were each initialized to 1 to test the estimator's robustness to initial state errors. The EnKF was configured to use 5 particles in its ensemble; we found that the performance and convergence of this filter was dependent upon the number of particles used.

One minor observation that merits a highlight is that the simulation time is set in the FMU when it is reimported, and is not set by the annotations usually used to capture information in Modelica models. As such, if the user wants to change the start or end time of a simulation, or the number of time steps used, this must be done both in the simulation window, as usual, and by setting the appropriate parameters of the FMU.

Figure 3 demonstrates the ability of both estimators to construct good estimates of state $i_a$ from noisy observa-

| $\mu$ | Value | $\mu$ | Value |
|---|---|---|---|
| $v_a$ | $\sin(2\pi t)$ | $\lambda$ | $0.1\ V \cdot s$ |
| $v_b$ | $\cos(2\pi t)$ | $J$ | 1.8e-4 $\mathrm{kg \cdot m^2}$ |
| $R$ | 1.9 Ω | B | 1e-3 $\mathrm{kg \cdot m/s}$ |
| $L$ | 3 mH | | |

**Table 1.** Motor parameters.

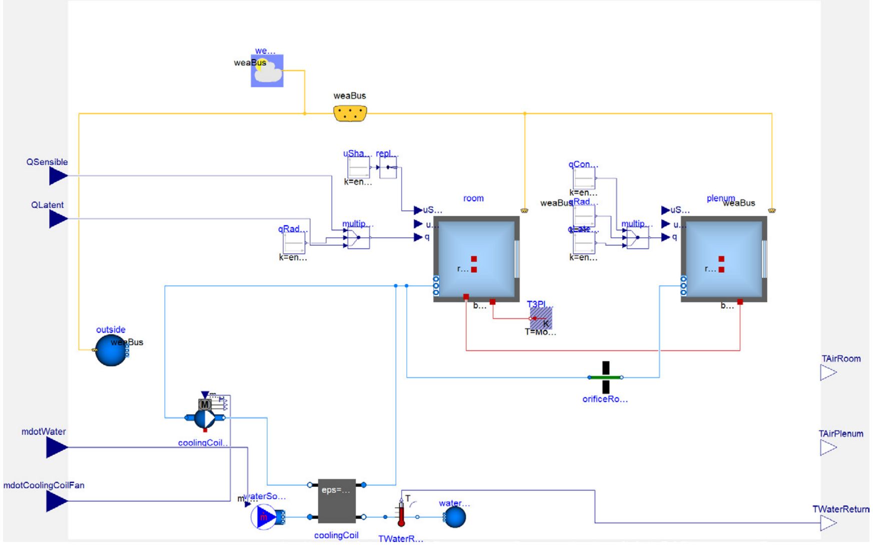


**Figure 5.** Two zone (room/plenum) model including a fan coil unit.

tions of the original motor model. The upper plot in this figure shows the performance of the EKF, while the lower plot shows the performance of the EnKF; the state estimate for both systems can be seen to be close to the mean of the input signal.

Figure 4 illustrates the estimates of the motor speed state $\omega$ as well as the estimates produced by the EKF and EnKF. It is clear from a comparison between the original plant signal and the estimates that both filters can produce good estimates of this state without its measurement. It is interesting to note that the estimate for this state from the EKF is somewhat noisier than those estimates from the EnKF, but that the mean of both of these estimates is quite close to the state of the original plant.

## 4 Building System

Though we can gain significant insights into the process of designing and implementing estimators by using the previous simple motor model, our primary interest in this technology lies in the ability to leverage large-scale Modelica system models for use in estimation problems. As a case study of this application, we constructed a two-zone model of one floor of a commercial office building, including both the occupied space and a plenum above, by using the models provided by the Modelica Buildings library (Wetter et al., 2014) with the objective of estimating the sensible heat load in the occupied space. This model is very similar to that which was used by (Bortoff and Laughman, 2019).

A schematic diagram of this building model is illustrated in Figure 5, which shows the two zones as well as a water-source dry-coil fan coil model that is used to manage the room air temperature. The use of the dry coil model is somewhat atypical, and will be elaborated later in this section. This fan coil is connected to a fixed temperature water source (10 °C) and sink (16 °C), and the room temperature is regulated by a PI controller that adjusts the mass flow rate of water through the coil to maintain the set point. The area of the floor is 415 $m^2$, while the room height is 2.6 m and the plenum height is 1.3 m; there is also a total of 83.6 $m^2$ of window area on the external walls. The heat load varies between 1.66 kW (4 $W/m^2$) between the hours of 7pm and 8am, and 4.15 kW (10 $W/m^2$) between the hours of 8am and 6pm, with smooth ramps during the transition hours. The Tokyo TMY3 weather file is used to provide the ambient conditions, and this simulation is run for 5 days from June 12-17 to study its behavior over a practical duration of time.

The connections between the building model and the estimator are illustrated in Figure 6, which illustrate the use of the original Modelica plant model and the FMI-based estimator. This estimator is based on four measurements

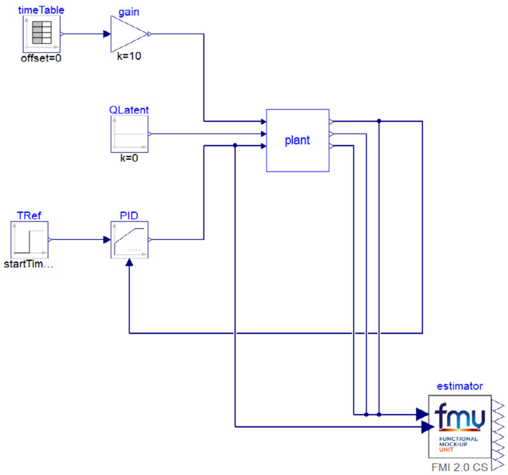


**Figure 6.** Estimator.

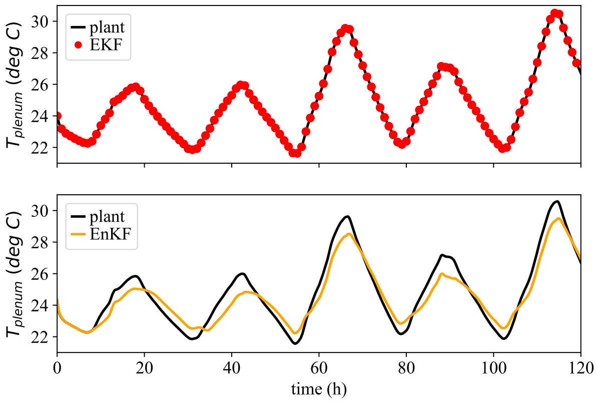


**Figure 7.** Estimated and measured plenum air temperatures.

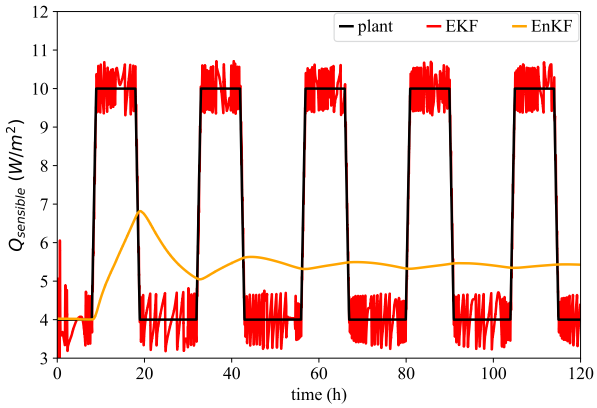


**Figure 8.** Imposed and estimated heat loads.

from the plant: a measurement of the water flow rate provided by the control signal for the fan coil unit, the room air temperature, the plenum air temperature, and the return water temperature. We then designed an EKF and an EnKF to estimate the heat load on the room. This objective necessitated a small modification of the model used for the estimator; while the sensible load is customarily specified as an input to the model, we added an integrator driven by a constant value to the sensible load input for the estimator model. By incorporating this integrator, we force the compiler to include the heat load as a state which can then be estimated. The scales of the states also required specific attention in the construction of the EnKF. In general, the states in these thermodynamic models are poorly scaled; for example, the internal energy of the water is on the order of $10^7$ J/kg, while the humidity ratio of the room is on the order of $10^{-3}$ kg w/kg da. Perturbations of the state vector therefore must be carefully designed so they do not dominate its mean, which tends to be constructed carefully in consideration of physical intuition.

We tested the EKF by adding 0.01 °C of quantization noise to the measured outputs of the plant; while this amount of quantization is small, it was chosen to ensure that the observed variations in room and return water temperatures (which were small due to the large coil capacity) would not be hidden by the quantization. In using these quantized inputs, we examined the ability of the EKF to properly estimate both the states characterized by the inputs as well as the heat load. The upper plot of Figure 7 illustrates the measured and estimated plenum temperatures for the EKF, and it is clear that the estimator can accurately capture the dynamics of the original plant. Moreover, Figure 8 demonstrates the rapid convergence of the estimates to the imposed heat load, suggesting that the EKF is effective for solving this estimation task.

In comparison, the estimates of both the plenum air temperature and the sensible heat load from the EnKF are relatively poor. While the EnKF captures the general shape of the plenum air temperature dynamics, there are significant errors; moreover, the errors in the heat load estimates are substantial, and completely miss the time-varying behavior of the heat load towards the end of the simulation.

The poor performance of the EnKF is fundamentally related to interactions between the ensemble perturbations and the state and output constraints of the model. Many models of thermodynamic systems have limits of validity on their variables; for example, the dry air model used in the Buildings library is only accurate down to temperatures of 200 K, while the humidity ratio in a space cannot physically go below zero. However, the ensemble perturbations in the EnKF are not designed to satisfy these constraints in a statistically valid manner. We found many circumstances in which small perturbations in the ensemble led to constraint violations either of the state variables directly (e.g., humidity ratio), or of algebraically related output variables (e.g., temperatures), causing the simulations to crash. The need to place stringent limits on the dispersion of the particles thus prevented the covariance matrix from accurately characterizing the system dynamics and producing an accurate estimate of the heat load.

This behavior was apparent in all of the experiments with the EnKF. We found that there were effectively no useful perturbations that could be applied to a wet-coil fan coil model that would not violate the constraint that the humidity ratio needs to be positive, and that even the room with the dry coil model often had problems in which the the mixed air humidity ratio would tend toward zero. While the EnKF does demonstrate promise for problems in which a suitably large ensemble can be used, it appears to have significant limits when used in an FMI-based context with constraints on state and output variables.

While the design of the FMI standard is suited to describe sets of differential equations for the purpose of simulation as well as estimation using Kalman filters and EKFs, the fact that it does not provide a direct association between variables and their constraints imposes a crucial limitation in the practical implementation of constrained estimators (Simon, 2010) or particle filters for large-scale problems (Van Leeuwen et al., 2019). For example, an estimator designed for the system

$$\frac{dx_1}{dt} = f_1(x_1, x_2, x_3; \mu) \quad (29)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, x_3; \mu) \quad (30)$$

$$x_3 = f_3(x_1, x_2; \mu) \quad (31)$$

$$a_1 \leq x_1 \leq b_1 \quad (32)$$

$$a_3 \leq x_3 \leq b_3 \quad (33)$$

must take the state and inequality constraints imposed by Equations 32 and 33 into account when generating the perturbation ensembles for the state vector $[x_1\ x_2]$. While these constraints could be incorporated manually into an estimator for a small system when modifying the Modelica code generated upon import of the co-simulation FMU, the application of these estimation methods would be much more practical if the FMU included the infrastructure needed to systematically associate the variables with their constraints.

## 5 Conclusions & Discussion

While the basic infrastructure used to construct FMI-based system models might not always have the most intuitive interface, their capabilities make them excellent candidates for use in a variety of estimation problems. We found that the implementation of the extended Kalman filter is relatively straightforward once the FMI API is fully understood, and that these filters demonstrated good performance on both a small test problem and a larger estimation problem that utilized the capabilities of Modelica for building models of complex physical systems. We also found that the FMI interface also enabled the construction of other types of estimators, such as the ensemble Kalman filter, suggesting that there is potential in the further investigation of other FMI-based interfaces for estimation applications.

However, this study also revealed some important limitations of FMI for the application of constrained estimation approaches, such as ensemble Kalman filters or other particle filtering methods. The lack of a direct association between a state variable and its constraints posed significant difficulties in the implementation of ensemble-based methods. Future work on systematically accommodating such constraints in FMI could have a significant impact on FMI's use on the range of estimation problems, especially for the large-scale applications to which Modelica models are so well-suited.

## References

S.A. Bortoff and C.R. Laughman. An extended Luenberger observer for HVAC application using FMI. In *Proceedings of the 13th International Modelica Conference*, pages 149–155, 2019. doi:10.3384/ecp19157149.

J. Brembeck. A physical model-based observer framework for nonlinear constrained state estimation applied to battery state estimation. *Sensors*, 19, 2019. doi:10.3390/s19204402.

J. Brembeck, M. Otter, and D. Zimmer. Nonlinear observers based on the Functional Mockup Interface with applications to electric vehicles. In *Proceedings of the 8th International Modelica Conference*, pages 474–483, 2011. doi:10.3384/ecp11063474.

J. Brembeck, A. Pfeiffer, M. Fleps-Dezasse, M. Otter, K. Wernersson, and H. Elmqvist. Nonlinear state estimation with an extended FMI 2.0 co-simulation interface. In *Proceedings of the 10th International Modelica Conference*, pages 53–62, 2014. doi:10.3384/ECP1409653.

Dassault Systemes. Dymola 2020, 2019.

G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2 edition, 2009a.

G. Evensen. The ensemble Kalman filter for combined state and parameter estimation. *IEEE Control Systems Magazine*, pages 83–104, 2009b. doi:10.1109/MCS.2009.932223.

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

Modelica Association. Functional Mockup Interface for Model Exchange and Co-Simulation, Version 2.0.1, 2019. URL `www.fmi-standard.org`.

D. Simon. *Optimal State Estimation: Kalman, H-∞, and Nonlinear Approaches*. John Wiley & Sons, 2006.

D. Simon. Kalman filtering with state constraints: A survey of linear and nonlinear algorithms. *IET Control Theory and Applications*, 4(8):1303–1318, 2010. doi:10.1049/iet-cta.2009.0032.

P. J. Van Leeuwen, H. R. Künsch, L. Nerger, R. Potthast, and S. Reich. Particle filters for high-dimensional geoscience applications: a review. *Quarterly Journal of the Royal Meteorological Society*, 145(723):2335–2365, 2019. doi:10.1002/qj.3551.

S. Vetra-Carvalho, P.J. Van Leeuwen, L. Nerger, A. Barth, M.U. Altaf, P. Brasseur, P. Kirchgessner, and J.-M. Beckers. State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems. *Tellus A*, 70:1–43, 2018. doi:10.1080/16000870.2018.1445364.

L. Vytvytskyi and B. Lie. Combining measurements with models for superior information in hydropower plants. *Flow Measurement and Instrumentation*, 69, 2019. doi:10.1016/j.flowmeasinst.2019.101582.

M. Wetter, W. Zuo, T. Nouidui, and X. Pang. Modelica buildings library. *Journal of Building Performance Simulation*, 7(4): 253–270, 2014. doi:10.1080/19401493.2013.765506.

# Enhanced Steady-State in Modelon Jet Propulsion Library, an Enabler for Industrial Design Workflows

Clément Coïc[1] Moritz Hübel[1] Matthis Thorade[1]

[1]Modelon Deutschland GmbH, Germany,
{clement.coic,moritz.hubel,matthis.thorade}@modelon.com

## Abstract

This paper communicates on the implementation of Physics-based Solving in the Modelon Jet Propulsion library, driven by requirements from industrial jet engine design workflows. On- and off-design simulation modes are typically sequential and iterative steps in a model-based design process of jet engines. The solution Modelon provides – based on the Jet Propulsion Library, Optimica Compiler Toolkit, FMI Toolbox and pyFMI – enables performing a robust design of a gas turbine for a design point satisfying relevant constraints of typical off-design scenarios. This paper illustrates this workflow with component and system level examples.

*Keywords: Jet Propulsion, Physics-based Solving, Steady-State, On-Design, Off-Design, Optimization*

## 1 Introduction

The sizing of a gas turbine is typically performed over a set of different scenarios. The design point would be the first step in identifying basic parameters as it would be the most constraining scenario for most components. Computing the component parameters from the boundary conditions on this point will be qualified as on-design simulation. For a jet engine, this would typically be the cruise mode at the top of climb. Other scenarios will be run to validate the design for conditions that are relevant for the expected operation such as takeoff or landing. In addition to the validations, iterative tuning of variables (e.g., cooling flow fraction) can be included – these would be named off-design simulations.

Running these scenarios in a disconnected fashion would be tedious and error prone. Model Based System Engineering applied to the design of gas turbines provides a relevant workflow that is addressed in this paper and serves as source of requirements for augmenting the Jet Propulsion library with additional features. Section 2 of this paper discusses such workflows, the resulting requirements and the associated implementation within Modelon Jet Propulsion Library. Section 3 provides some key benefits of using Modelica language and Modelon Optimica Compiler Toolkit to address this problem. Section 4 presents two component examples and discusses their specific on- and off-design behaviors. Section 5 illustrates all these topics within a system level example: the cycle model design of a jet engine.

## 2 Discussion on Model-Based System Engineering

### 2.1 Simulation Modes in a MBSE Design Cycle

In an industrial Model-Based System Engineering (MBSE) development, the customer needs, system goal, and purpose would define the system requirements (R from the RFLP acronym). The system would be split into several subsystems – most likely through a function breakdown structure process (F from the RFLP acronym) – and requirements would be propagated and incremented, with traceability and rationale, to the subsystems. This step would be reproduced as many times as necessary to reach a level of subsystems that can be assigned to a given company department – typically organized by physical domains or main product functions.

Based on a requirement specification and the functions and scenarios it shall fulfill – assuming correctness, completeness and consistency –a subsystem can be designed. In a model-based design approach, lumped parameter modeling and simulation (L for logical from the RFLP) can be used to define a viewpoint of the subsystem architecture, select component technologies (assisting trade-off studies) and size each of them. At this stage, it is relevant to highlight that some efforts exist in verifying that models satisfy the requirements using the Modelica language (see for example [OTT15] or [BOU18]). Finally, the detail design would be performed using 3D drawings, Finite Element Analysis, Computational Fluid Dynamics, etc. – this would be the P for physical from the RFLP.

One of the main added values of using model-based development is the possibility to run virtual tests during the design cycle, prior to manufacturing any physical prototype. This enables iterating on the design in a cost- and time-effective manner.

This statement enables deriving two different simulation modes, based on the different steps of the design process:

10.3384/ECP20169196 DOI

1. On-design: the boundary conditions (design point scenarios) are known and the aim of the simulation is to compute the parameters of the system (sizing).
2. Off-design: the parameters are all known at this stage and the system and experiments can be run based on some boundary conditions. The outcomes will be the system behavior.

It is quite intuitive based on the above description to see these two modes as sequential and probably iterative. A user would first run the on-design simulation on a design point and then run the simulation and validate his design. The iteration process would come from the fact that the design point might not be unique – e.g., it might be different for each component or subsystem of the system – and thus convergence of the system design can only be achieved by iterating on the on-design simulations.

Figure 1 presents the double-V-model from VDI 2206 [VDI04] standard, displaying the RFLP steps, on which the on- (1) and off-design (2) simulations and potential design iterations (3) have been drawn onto in orange.

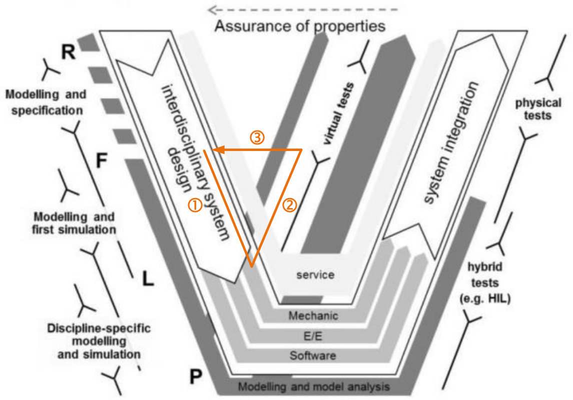

Assurance of properties
R
Modelling and specification
F
Modelling and first simulation
L
Discipline-specific modelling and simulation
P
interdisciplinary system design
virtual tests
system integration
physical tests
hybrid tests (e.g. HIL)
①
②
③
service
Mechanic
E/E
Software
Modelling and model analysis

**Figure 1.** Simulation modes on a design cycle, based on [EIG12]

## 2.2 Requirements for models

The notion of system and subsystem is a question of perspective. When changing viewpoint, one system might become a subsystem and vice-versa – e.g., when sizing an aircraft, the jet engine is a subsystem; however, when sizing the jet engine, the engine is the system while the turbine might be a subsystem. Models in a MBSE development are a "key tool" for the design. For a model developer, the model shall be considered as a system. A good practice is thus to have requirements for the model development, in the same way that we have requirements driving the physical system itself.

The authors already contributed in:

- adding requirements when it comes to model development [COI16], quoting:
  - R1 Realism. The model architecture shall enable incremental modelling to progressively increase realism regarding the key physical effects that impact performance.
  - R2 Genericity. As far as possible, the model shall be made of a combination of generic sub-models that can be re-used for other modelling purposes.
  - R3 Interfacing. The model shall have standard interfaces that are conserved throughout modelling levels in order to ensure models' replaceability.
  - R4 Balancing. The model shall be balanced (mechanically, energetically, etc.).
  - R5 Ageing and faults. The model shall enable ageing effects or faults to be simulated.
  - R6 Causality. The model shall be developed to admit various causalities, including for inverse simulation.
- developing model architecture that fits the main functions to fulfill [COI18].

It is relevant here to note that the Modelica language enables developing models that fulfill all these requirements. The Modelon Jet Propulsion library is a great example of that. In [SIE17], Sielemann et al. introduced this library dedicated to modeling and simulation of jet engines. This paper shows how the model architecting enables selecting different levels of realism of the subsystems (R1), based on the use of generic models (R2) and interfaces (R3) – see Figure 2. Balancing (R4) is ensured in the library in Modelon implementation of physical laws and following Modelica specification for model numerical balancing [OLS08]). The a-causality of Modelica enables satisfying the last requirement (R6). While ageing and faults (R5) are perfectly feasible with Modelica language, this topic is out of the scope of this library and communication, for now.

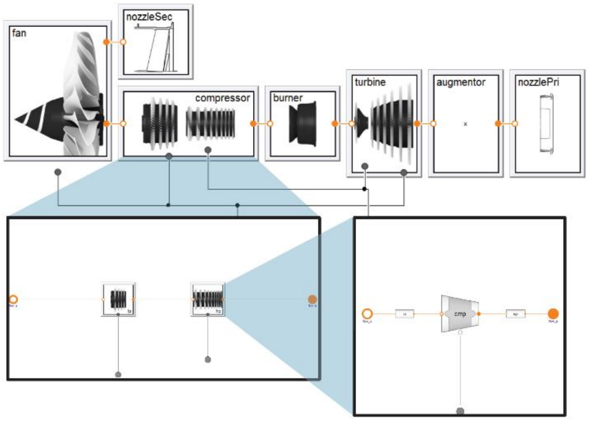


**Figure 2.** Top-level turbofan model breakdown shown on the top, compressor break-down on the lower left, high pressure compression section break-down on the lower right – from [SIE17]

## 2.3 Additional requirements for the simulation modes in Steady-State

Unfortunately, there is still a lack of standards when it comes to requirements for model development. In this paper, the authors would like to contribute on this topic by adding model requirements to fit the workflows of model-based design and more specifically for the simulation modes discussed in §1.

Typically, the technical requirements needed to follow the design workflow are:

- In terms of processing:
  - R7. The user shall be able to switch between on- and off-design without changing model structure or realism.
  - R8. The user shall be able to re-use the outputs from the on-design simulation as input to the off-design simulations.
  - R9. The user shall access the model execution through convenient user interface and scripting tools.
- In terms of user friendliness:
  - R10. Switching between simulation modes should not create execution overhead.
  - R11. Switching between simulation modes should be performed by changing a single parameter.
  - R12. The model shall have a robust steady-state embedded capability.

## 2.4 Implementation of Steady-State simulation modes into Modelon Jet Propulsion Library

To meet these requirements, Modelon developed a vendor-specific language construct, supported by the Modelon Optimica Compiler Toolkit (OCT), to support Physics-based Solving of systems. The Physics-based Solving implemented in our libraries relies on Modelon insights about the physical properties of components and systems to achieve a structure of the system of equations that yields vastly superior numerical properties as compared to traditional tearing algorithms.

Tearing is a symbolic substitution technique well-established in general system simulation. For an introduction in the context of Modelica, see [ELM94]. In most Modelica tools, the selection is fully automatic and based on heuristics. These heuristics are based on the structure of the equation system and code implementation, but not on physical insight. This typically works well for nonlinear algebraic equation systems with a small to medium size of iteration variables per block. However, when using automatic tearing, iteration variables can change unexpectedly for the user with small model changes (e.g., adding more components to the system, changing the type of a model, or with a structural parameter change). Then, hand-tuned start values are lost. Additionally, automatic tearing may choose iteration variables for which the user has less intuition concerning suitable start values or bounds based on engineering insight.

With the Physics-based Solving, the above challenges were solved, and the authors were able to implement this engineering in the library. Instructions embedded in component models guide the compiler and solver on which variables and equations should be selected respectively as iteration variables and residuals for the steady-state simulation. The Physics-based Solving enables a robust steady-state simulation (R12). This language construct enables changing the iteration variables and residuals based on Boolean parameters, without the need for recompilation. The information is stored in an object-oriented fashion, such that modelers can assemble systems graphically, and the desired solving can be deduced from the model topology (model instances and connections). In the discussed application, changing the simulation mode parameter from on- to off-design would change the set of iteration variables and residuals the solver would use. This feature was used to meet requirements R7, R10 and R11.

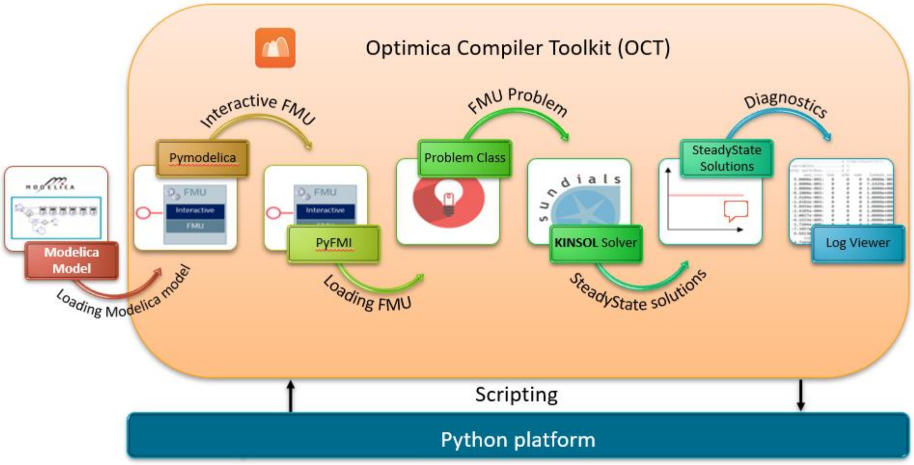


**Figure 3.** Typical workflow using OCT with Python interface

Generally accepted choices for iteration variables and residual equations are documented in many textbooks and scientific papers from the gas turbine community (see Walsh and Fletcher [WAL04], Oates [OAT97], for instance). They are commonly referred to as "thermodynamic matching" by this community. All that was required for this work was encoding the information in the Modelica language through the above-mentioned language construct, and then testing and validating physical and numerical behavior.

The Modelica compiler OCT generates Functional Mock-up Units (FMUs) – following the FMI standard. Using pyFMI or the FMI Toolbox from Modelon, it is easy to interact with the FMU, and thus requirements R8 and R9 are both met using Python or Matlab scripting into Jupyter notebooks. Figure 3 shows the typical workflow using OCT with Python interface for steady-state simulation.

# 3 Steady-state simulation using OCT – a robust solution

## 3.1 Modelica Compiler capabilities

The Modelica language, being equation-based, enables the compiler to have more degrees of flexibility. All Modelica compilers are able to rearrange the acausal equations into causal algorithms, suitable for simulation. Modelica compilers are also usually able to automatically derive the sensitivities relevant for solving optimization problems.

While these features are generic to most Modelica compilers, Modelon compiler (OCT) has the additional capability to hide residual equations and iteration variables from the solver in a compiled model. This feature is key to enable switching between simulation modes without recompiling the model.

## 3.2 Solver Improvements

Modelica models and FMUs usually use variables expressed in SI units. Variable values may therefore differ by several orders in magnitude [SIE12]. This can yield to ill-conditioned systems which are difficult to converge. In order to handle ill-conditioned systems, the solver supports scaling of both the residual equations and of the iteration variables, which is used in the criteria for a successful solution. Note that while for the Newton method the step is scaling-independent (i.e., direction and length do not change with scaling), for the steepest descent search direction as well as solver termination criteria, results are dependent on the scaling factors applied.

Iteration variables are normally scaled based on the nominal values provided by the modeler. This is done using the nominal attribute $x_{nom_j}$, according to $D_x^j = 1/|x_{nom_j}|$. If the nominal attribute for a particular iteration variable is not set, then the corresponding scaling factor $D_x^j$ is set to one.

Residual scaling factors utilized by the solver are evaluated based on the Jacobian and follow ideas similar to Jacobian equilibration. The automatic scaling is chosen as $D_x^j = 1/\left\|\bar{J}^k(x_0)\right\|_\infty$, where $\bar{J}$ is a scaled Jacobian calculated as $\bar{J}(x) = \nabla_x F(x) D_x^{-1}$ and where $x_0$ is the initial guess.

OCT steady-state solver relies on a combination of residual and step norm as exit criterion. Newton step norm-based criterion is typically used as the primary convergence indicator. If iterations fail to converge for that criterion an additional check on residual criterion is done and the solution is accepted if residuals are sufficiently small.

## 3.3 Debugging capability

The OCT solver generates log messages in XML format, enabling automated post-processing and debugging of non-convergence. OCT provides a dedicated Python package that facilitates parsing of the log and extraction of the most relevant information.

To further facilitate interactive non-convergence debugging, the framework includes a Python package that enables user interaction with the equation system from the Python console. The interactive features include temporary elimination of some of the iteration variables and residuals from the equation system, local residual and Jacobian analysis, etc. The interactive framework facilitates equation debugging and enables localization of the problematic residual equation (e.g., with local extrema or discontinuity).

# 4 Component level examples

## 4.1 Combustor

The combustor or burner is a key component for gas turbine simulation. It models the injection of a fuel stream into a gas stream, and its partial or complete combustion. It is usually modeled as a two-port or three-port component, based on whether the fuel supply shall be modeled in a physical way (e.g., solving mass flow rates from pressure differences and correlations of wall friction) or simplified. In case of the latter, no physical fluid connector is used for the fuel supply. Instead, parameters or signal inputs are used to specify fuel supply information. In the case of the former, a physical fuel connector with complete information on convective transport quantities such as pressure, mass flow, composition, and specific enthalpy is included, and physical pipe or boundary condition models are connected to the burner. Independently of the modeling abstraction of the fuel supply, the gas inlet and outlet of the combustor are always normally modeled with physical connectors in the Jet Propulsion Library.

Engineering activities call for some flexibility in the modeling of such combustors. One aspect relates to the prescription of fuel flow. The most basic ways of prescribing the latter are to prescribe either the dimensional fuel flow (in units kg/s or a non-SI counterpart) directly, or the non-dimensional fuel-to-air ratio ("FAR"). Following the "thermodynamic matching" principle, upon evaluation of the combustor model equations, a guessed or correct air flow rate entering the combustor will be known. Thus, it is straightforward to compute the dimensional mass flow rate of fuel from FAR in case of the latter, and both are equivalent from a computational order principle. "Thermodynamic matching," then, prefers the usage of non-dimensional FAR over dimensional fuel flow rate because of its higher robustness against variations in gas turbine requirements/size. (As the gas turbine becomes larger, the air flow rate increases. To compensate, an increase in fuel flow is required to maintain otherwise unchanged conditions, which is already accounted for when using FAR as iteration variable and not accounted for when using dimensional fuel flow as iteration variable.).

However, the user typically wants to prescribe fuel flow or FAR indirectly via combustor outlet temperature (so-called "T4"), net thrust, etc. With this implementation, it is possible to switch between these prescriptions while iterating on commonly preferred FAR by switching out residual equations (that enforce equality of T4, net thrust, etc.).

To enable switching between different equation systems for on-and off-design, similar functionality was implemented in the compressor.

## 4.2 Compressor

The compressor usually receives mechanical power from a shaft and converts it into thermofluidic power by compressing the entering gas – as its name indicates. Following the air flow path, it is typically located before the burner and the turbine and is aimed at bringing the gas to a higher pressure. It is modeled in the Jet Propulsion Library as a two-port component – it can also include vectorized bleed ports if desired – and uses maps to define the compressor performance. While the compressor is modeled in a physical way, its map is purely an algebraic abstraction of the performance that enables solving the compressor models in a more robust way.

For the compressor, the library includes a model of the R-line map. R-lines are sets of curves that can be parallel to the surge line and evenly spaced among each other. The use of such an artificial interpolation to coordinate R-lines (sometimes also called argument lines or beta lines) ensures unique results in the regions of low corrected air flow, where pressure ratio is almost a constant and regions of constant air flow towards the highest air flow region for a given speed line (avoiding table look-up along vertical or horizonal tangents).

The performance map is a good example of how on- and off-design sets of equations are different and how the Physics-based Solving is well suited to this workflow:

- For on-design simulation, the scaling factors of the map are computed based on the design point performances.
- For off-design simulation:
    - These scaling factors shall now be fixed parameters and therefore are held constant during the simulation, and the equations that defined them are hidden to the solver.
    - The off-design operating points and surge margin are an output of the map and are computed based on the off-design point definition.

For both simulation modes, the R-line is an iteration variable that is associated to a residual equation that enforces the corrected air flow from the map to be equal to the physical airflow computed in the compressor.

# 5 Optimization of a jet engine – a system example

## 5.1 System under study

In this part, the optimization of a jet engine (also described as cycle model in the literature – due to the importance of thermodynamics effects) is discussed. The architecture selected and objectives for the parameter values are based on [SIE19] in which the technology investigated corresponds to an Entry-into-Service in year 2035 for geared turbofan engine.

A geared turbofan is shown in Figure 4. It usually includes one inlet fan that can be divided into a center part that supplies the flow for the core region (fanCore) and a coaxial outer section that supplies air for the bypass (fanByp). The center part of the engine includes two compressor stages (ipc and hpc), the combustion chamber or burner (brn), and the turbine consisting of a high-pressure section (hpt) and a low-pressure section (lpt). There are two nozzles at the outlet, one for the bypass (nozByp) and one for the center part (nozCore). A gearbox (gear) between the turbine and the fan allow different rotational speed (e.g., for optimized efficiency of each section). It enables a design in which the speed of the inlet fan with its large blades is reduced and the compressor and turbine can be operated at higher speeds.

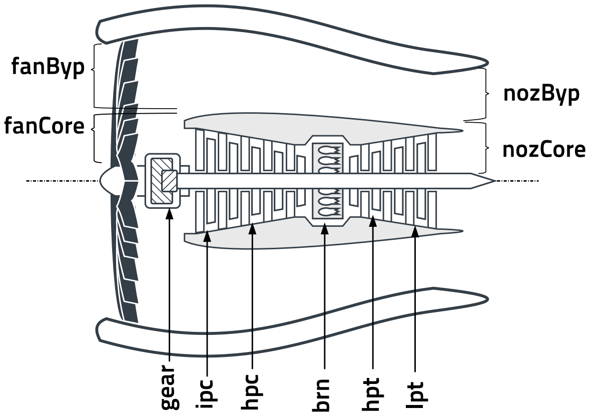


**Figure 4.** Cross-section view of a geared turbofan design

Figure 5 presents the associated model of this geared turbofan developed using Modelon Jet Propulsion library.

On the left of the schematic, the air flow enters the gas turbine via an inlet. Eventually, it is split into the core flow (through combustor and turbines) and bypass flow. The bypass flow terminates, after the bypass side of the fan and a duct model, with the bypass nozzle. The core flow passes through the usual set of compressors, the burner, and the usual set of turbines. The connections on the top of the compressor and turbine components represent the bleed air flow. Air is extracted from the high-pressure compressor both for customer uses (routed to boundary custBld) and to cool the high-pressure and low-pressure turbines. At the bottom of the compressors and turbines, the mechanical shaft connections are made. The high-pressure spool and the low-pressure spool with fan gear are shown. Eventually, the core flow is disposed of via the core nozzle. Analysis overview

In order to achieve a proper design of the cycle, a set of simulations on several operating points has been performed. The scheme involves the following steps:

- Converging a simulation for the design point
- Extracting the gas turbine sizing and initialization data for use in off-design points (see [BEC15] for more details on this)
- Creating a simulator instance for all off-design points (e.g., cruise and take-off)

We exemplify the need for iteration described in Figure 1 via the definition of cooling flows to be extracted from the high pressure compressor to the high pressure turbine and low pressure turbine (see above cycle description). Typically, they must be set to some approximate value on the design point, and then the resulting turbine blade metal temperatures must be computed in all off-design cases (so-called “uniform blade temperatures” in gas turbine parlance). In order to enforce a maximum temperature across all cases, the gas turbine designer must iterate on the cooling flow fractions on the design case until all temperature constraints are met.

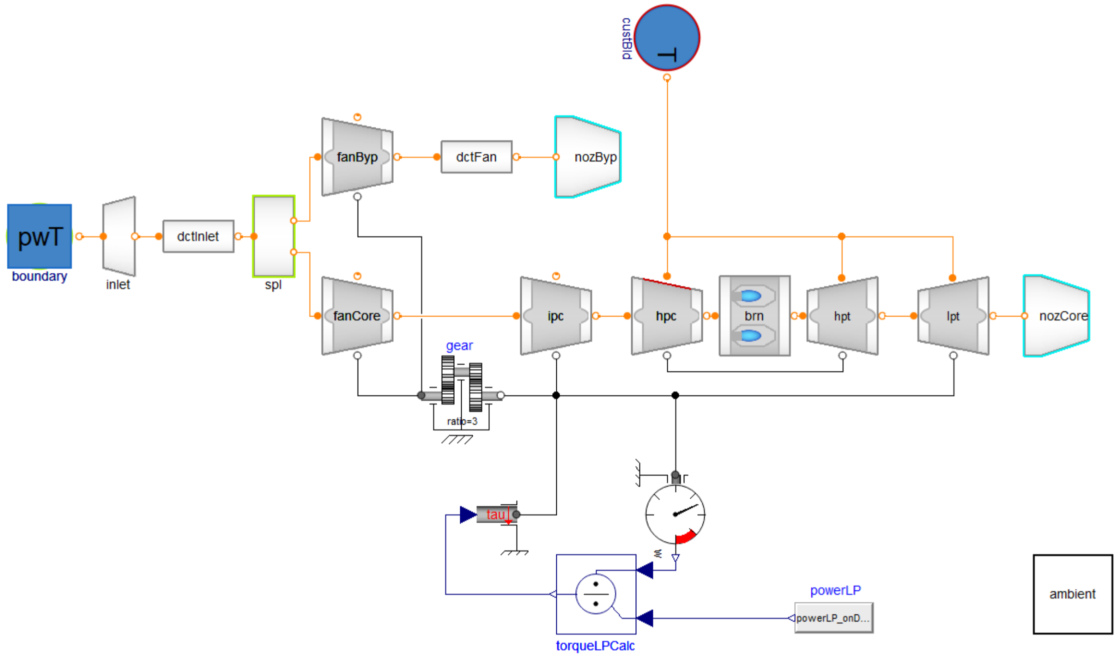


**Figure 5.** Architecture of the geared turbofan to optimize

## 5.2 Cycle missions and simulation modes

In [ZHA19], a slight variant of the above-shown geared turbofan model was investigated for a A320-200 type aircraft. The model setup was replicated for this effort. [SIE19] provides data on the accuracy of the resulting data match. The general missions presented in this communication are listed in Table 1, below.

| | Top of climb | Cruise | Take-off |
|---|---|---|---|
| Thrust | 24 kN | 18 kN | 92.5 kN |
| Day type | ISA | ISA | Hot day (ISA+15 K) |
| Altitude | 35000 ft | 35000 ft | 0 ft |
| Mach Number | 0.78 | 0.78 | 0.25 |
| Type | On-Design | Off-Design | Off-Design |

**Table 1.** Cycle design missions

The design point in the missions under study is the “Top of Climb” – i.e., it is aerodynamically the most constraining mission point for the sizing of the gas turbine, optimizing its specific fuel consumption (SFC), while keeping the uniform blade temperature (UBT) below 1240°C. Note that the SFC corresponds to the ratio between the fuel consumption and the thrust of the jet engine. Nevertheless, the other missions are relevant to include as they represent the points with highest thrust requirement (take-off) and longest duration (cruise, thus driving overall mission block fuel consumption).

## 5.3 Cycle design, results and discussions

For the geared turbofan under study and modeled with Modelon Jet Propulsion library, the overall pressure ratio (OPR) technology variable was set to 55, which is a relevant order of magnitude for a 2035 technology. Overall pressure ratio is the product of all fan and compressor pressure ratios, and a key technology variable as it increases the thermal efficiency of the cycle. A manual convergence was achieved by varying the above-mentioned cooling flows on the design point, and all temperature criteria were eventually met for the different mission profiles.

In a second step, in a small design space exploration exercise, the OPR was varied down to 50 and up to 60 (however, the manual convergence of cooling flows and temperatures is not included in the analysis). Figure 6 shows a key cycle design result: the specific fuel consumption (SFC) versus the OPR.

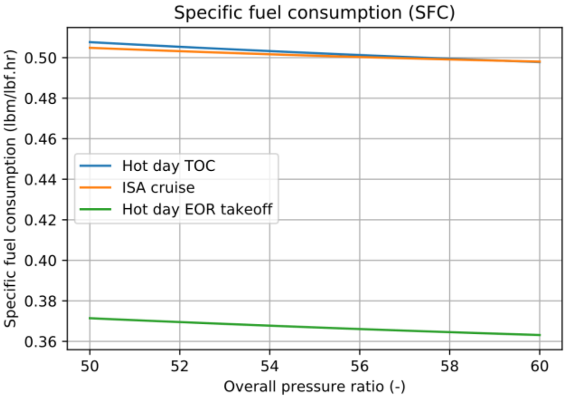


**Figure 6.** Design exploration, SFC v/s OPR

Figure 6 shows how the specific fuel consumption improves with increasing OPR. The most relevant line is the orange one, which is weighed highly in the total mission block fuel. From this plot alone, we would be tempted to directly move to even higher OPR values to leverage the fuel consumption improvement. However, upon reviewing results on other key variables (the above-described temperatures), we see that further manual convergence work is required to yield the complete picture.

In Figure 7 we see the corresponding results. Based on material and cooling technology constraints, the uniform blade temperature (UBT) shall remain below 1240 K. The manual convergence was applied at OPR 55 and yields exactly these or lower values for all operating conditions and turbines. Figure 7 shows how the UBT varies within the design exploration.

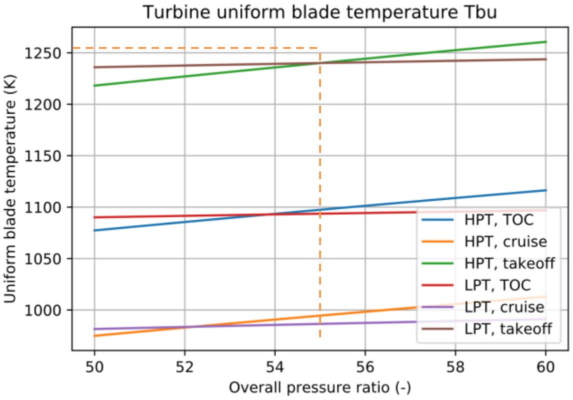


**Figure 7.** Design exploration, UBT v/s OPR

Keeping the cooling flow fractions constant results in violation of temperature constraints on turbine blade metal temperatures. This convergence would have to be achieved not only for an OPR of 55 but for all the OPR values to avoid dropping with UBT below acceptable temperatures at lower OPR (leaving unused efficiency potential on the table) and going beyond acceptable UBT at higher OPR (and thus resulting in an infeasible design). This plot thus illustrates the need for iterations on the flow fractions when sizing a cycle model.

To illustrate the challenge further, the variation in velocity ratio and specific thrust are plotted respectively in Figure 8 and Figure 9. To yield a balanced and efficient design, the gas turbine designer intends to maintain these at specific values, which lead to high efficiency [KYP17]. These values would have to be included in the manual convergence, too, and illustrates the complexity of the required analysis work.

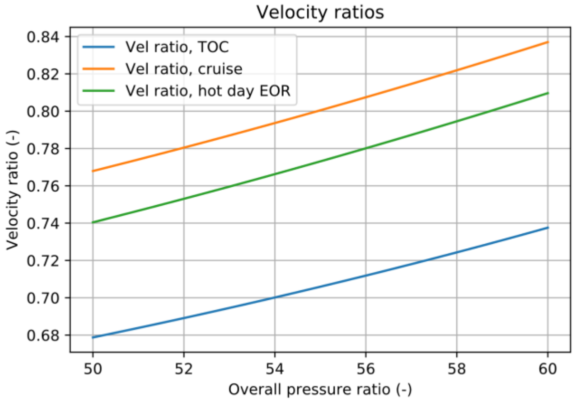


**Figure 8.** Design exploration, VR v/s OPR

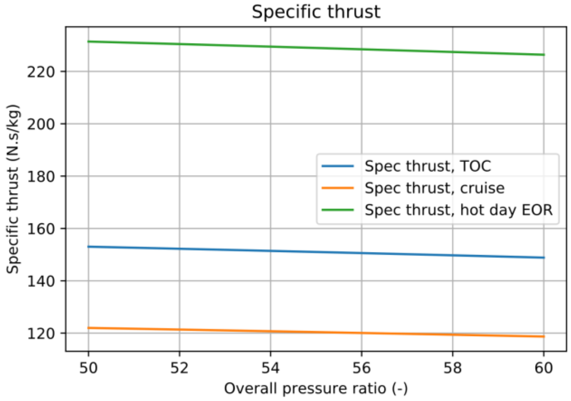


**Figure 9.** Design exploration, ST v/s OPR

Figures 5 to 8 illustrate how robustly the steady-state cycle analysis problems can be solved with the new Physics-based Solving functionality. This analysis is enabled by the functionality introduced above, such as for the extraction of solutions from one simulation for the initialization of another. At the same time, the figures also illustrate how a full optimization of the cycle design for given objectives requires observing and satisfying constraints, which, based on problem complexity, can be tedious. This observation indicates a potential next step in our work – to implement automatic multi-point design techniques to solve the above-described consistency via the already-used equation solver across the design point and all the off-design points. This is an ongoing development that is beyond the scope of this paper, and it will be described in a future publication.

## 6 Conclusion and perspectives

This paper discussed how the Modelon products enable suiting the design workflow of a typical cycle model. This simulation was made possible by properly addressing the technical requirements for following such a workflow and developing a dedicated Physics-based Solving design in the library to meet these requirements. Additionally, Modelon Optimica Compiler Toolkit as well as pyFMI and FMIT products are enablers for this workflow. Illustrations of this statement were performed in component and system level examples.

While this paper addresses the complexity of gas turbine design and illustrates the benefit of Modelon dedicated products, it also touches the multi-point design problem without covering it in the examples.

## 7 References

[BEC15] Becker, R.-G., Bolemant, M., Krause, D, Peitsch, D., *An automated process to create start values for gas turbine performance simulations using neural networks and evolutionary algorithms*, https://ieeexplore.ieee.org/xpl/conhome/8365669/proceeding International Gas Turbine Congress, Tokyo, Japan, 2015.

[BOU18] Bouskela, D. & Jardin, A., *ETL: A New Temporal Language for the Verification of Cyber-Physical Systems*, 2018 Annual IEEE International Systems Conference (SysCon), Vancouver, Canada, 2016.

[COI16] Coïc, C., Fu, J. & Maré, J.-C., *Bond Graphs Aided development of Mechanical Power Transmission for Aerospace Electromechanical Actuators*, International Conference on Bond Graph Modelling, Montréal, Canada, 2016.

[COI18] Coïc, C. & Biéron, M., *An Evolutive Bond Graph Modeling of Aerospace Hydraulic Reservoirs and its Modelica Implementation*, International Conference on Bond Graph Modelling, Bordeaux, France, 2018.

[ELM94] Elmqvist, H., Otter., M., *Methods For Tearing Systems Of Equations In Object-Oriented Modeling. European Simulation Multiconference*, Barcelona, Spain, 1994.

[EIG12] Eigner, M., Gilz, T. & Zafirov, R., *Proposal for functional product description as part of a PLM solution in interdisciplinary product development.* International Design Conference, Dubrovnik, Croatia, 2012.

[KYP17] Kyprianidis, G. K., Dahlqvist, E., *On the trade-off between aviation NOx and energy efficiency. Applied Energy, volume 185, pages 1506-1516,* Elsevier, 2017.

[OAT97] Oates, G. C., *Aerothermodynamics of Gas Turbine and Rocket Propulsion.* AIAA Education Series, 1997.

[OLS08] Olsson, H., Otter, M., Mattsson, S. E. & Elmqvist, H., *Balanced models in Modelica 3.0 for increased model quality.* Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 2008.

[OTT15] Otter, M., Nguyen, T., Bouskela, D., Buffoni, L., Elmqvist, H., Fritzson, P., Garro, A., Jardin, A., Olsson, H., Payelleville, M., Schamai, W., Thomas, E. & Tundis, A., *Formal Requirements Modeling for Simulation-*

*Based Verification*, Proceedings of the 11$^{th}$ International Modelica Conference, Versailles, France, 2015.

[SIE12] Sielemann, M., *Device-Oriented Modeling and Simulation in Aircraft Energy Systems Design.* PhD Dissertation, Munich, Germany, 2012.

[SIE17] Sielemann, M., Pitchaikani, A., Selvan, N. & Sammak, N., *The Jet Propulsion Library: Modeling and simulation of aircraft engines.* Proceedings of the 12th International Modelica Conference, Praha, Czech Republic, 2017.

[SIE19] Sielemann, M., Thorade, M., Nguyen, A., Zhao, X., Sahoo, S. & Kypriandis, K., *Modelica and Functional Mock-Up Interface: Open Standards for Gas Turbine Simulation.* ASME TurboExpo, Phoenix, USA, 2019.

[WAL04] Walsh, P. P. & Fletcher, P., *Gas Turbine Performance.* John Wiley & Sons, 2004.

[VDI04] VDI GUIDELINE 2206, *Entwicklungsmethodik für mechatronische Systeme – Design methodology for mechatronic systems*, Beuth, 2004. (in German).

[ZHA19] Zhao, X., Sahoo, S., Kyprianidis, K., Sumsurooah, S., Valente, G., Rashed, M., Vakil, G., Hill, C. I., Jacob, C., Gobbin, A., Bardenhagen, A., Prössl, K., Sielemann, M. Rantzer, J. & Ekstedt, E., *A Framework for Optimization of Hybrid Aircraft.* ASME TurboExpo, Phoenix, USA, 2019.

# Object Oriented Modeling and Control Design for Power Electronics Half-Bridge Converter using Modelica

Giuseppe Laera[1] Luigi Vanfretti[1] Kyle Thomas[2] Matthew Gardner[2]

[1]ECSE, Rensselaer Polytechnic Institute, Troy (NY), {laerag,vanfrl}@rpi.edu
[2]Dominion Energy, Richmond (VA), {kyle.thomas,matthew.gardner}@dominionenergy.com

## Abstract

In this paper the focus is on a particular type of converters that is the two-level VSC (DC/AC Voltage Source Converter). In this category the averaged model and the switching model of an half-bridge converter are considered. The half-bridge is a building block for multiphase and multilevel converters.

The implementation of the two models of half-bridge converter using Modelica language is described with the structure of the package developed in Dymola. Different control strategies are introduced showing different behavior of the models in the simulations. The goals of this paper are several. First of all the modeling choice was to use Modelica for this type of work, that is traditionally carried out with domain specific tools, to show that it is possible to perform implementation and studies in the same field where traditional commercial softwares have been commonly and extensively used, with additional benefits that the language provides. In addition to that, this paper shows that the control design studies for an half-bridge converter, typically performed using averaged value models, can result in a set of control parameters values that are not successfully applicable to switching models of the same power electronic device.

*Keywords: VSC converters, half-bridge, averaged model, switching model, STATCOM, Modelica, Dymola, PI control, lead compensator, lag compensator, resonant control*

# 1 Introduction

## 1.1 Motivations

With the enhancements of power semiconductors the application of power electronics has been extended from traditional domestic and industrial applications to electric power systems. The power-electronic converters for power systems are used especially for power compensation and power filtering. Such converters consist of a power circuit realized through different configurations of switches and passive components coupled with a control system (Yazdani and Iravani, 2010).

In the past, the applications of power-electronic converter systems in electric power systems were limited to the HVDC (High Voltage Direct Current) transmission systems and static reactive power compensators like SVC (Static Var Compensator). With time the application areas for generation, transmission and distribution of electric energy have been extended for several reasons. The main incentives are represented by:

- Continuosly growing development of power electronics technology for electric power systems
- Development of signal processing and control strategies
- Issues with power line congestions
- Growing of energy consumption leading to the utilization of the existent electric infrastructure at its technical limits (stability issues)
- Increasing penetration of renewable energy sources due to the economic feasibility and to environmental concerns

The use of power-electronic converters in power systems is also motivated by the need to improve the efficiency and reliability of the existent electric infrastructure for integrating large scale renewable energy sources and storage systems.

The main applications of power-electronic converters in power systems are:

- *Active filters*: they synthesize and inject specific components of current or voltage into the grid to improve the power quality (Rashid, 2017)
- *Compensation*: the aim is to improve the power transfer capability of the lines, the voltage stability, the power quality. In this category we have the STATCOM (Static Synchronous Compensator). The function of the STATCOM is to act as power compensator by injecting or absorbing reactive power from the grids (see Figure 1).
- *Power conditioning*: the idea is to allow for a power exchange between two electrical systems under control to meet specific requirements like frequency, voltage magnitude, etc. (Maza-Ortega et al., 2017)

This paper focuses on the implementation and design of a simple power electronic DC/AC converter, the half-bridge. The first modeling choice is to use Modelica for

DOI 10.3384/ECP20169205

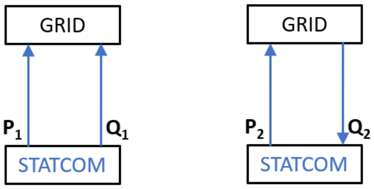


**Figure 1.** Behavior of the STATCOM as compensator (reactive power Q1 injection and reactive power Q2 absorption).

this type of work that is traditionally carried out with domain specific tools. In contrast with the current practice, the use of a modern object-oriented equation based programming language like Modelica allows to show that it is possible to perform implementation and studies in the same field where traditional commercial softwares have been commonly and extensively used, with additional benefits that the language provides. The models used in this paper have been chosen on purpose from the literature (referenced in the bibliography) to show that it is possible to obtain comparable results with this alternative tool instead of the classical PSCAD, etc.; and so that a reader from the power engineering community can quickly realize the value of the Modelica approach.

## 1.2 Previous Works

Modeling of power electronic converters in Modelica has been described in previous studies like (Haumer and Kral, 2011). It is a very informative work and it provides a very thorough description for an active front end converter with PWM (three phase AC/DC converter), however, from the authors point of view it requires that the reader is already quite familiar with Modelica. The paper also deals with an example more appropriate for engineers in the area of machine drives. The aspect of the synchronization of the control of the inverter to the grid voltage, discussed in (Haumer and Kral, 2011), would require the implementation of a PLL (Phase-Locked Loop) component that is part of the on-going work of the authors for a three-phase full STATCOM model. The main objective of this paper is to use the application illustrated as the starting point for the implementation of more complex power electronics models and it is focusing more on the differences of a couple of output current control strategies and tuning depending on the considered model of the half-bridge converter if averaged or switching.

In power system analysis studies, different levels of modeling detail are used to represent the converter switches. This is because when simulating a large number of switches, for example in modular multilevel converters, even ideal switches will lead to very large simulation time. Hence, different modeling approaches to represent each module (i.e. switches) are used (Saad et al., 2016). In this paper the switching behavior of the switches is considered ideal. This means that they can be turned off or on instantaneously. In reality a switch, combination of a transistor and its anti-parallel diode, cannot be turned on and off instantaneously. Those processes require a so called *dead time* or *blanking time* during which the signal to the gate of the switch is set to zero to reach the complete on or off condition. In other words the switching devices have finite turn-on and turn-off delays (Zammit et al., 2016). This is necessary because if we consider the leg of the half-bridge converter, as in this paper, if one switch has not yet completed the turn-off process and, at the same time, the other switch is turned-on then a *shoot-through* failure (short-circuit current through the leg) can happen damaging the converter (De Doncker et al., 2010). The dead time introduces a non-linearity which causes distortion in the output voltage and current (Zammit et al., 2016).

Another example of half bridge converter implementation using Modelica is given in (Winter et al., 2015) where a DC/DC converter, extensively used in the automotive industry, is considered. An averaged model of the converter is taken into account and the study covers aspects like converter losses in addition to a validation versus a SPICE model.

In (Olenmark and Sloth, 2014) an investigation about the implementation of different control strategies for power electronic converters has been performed in Modelica with the software tool Dymola, the same used in this paper. However, a different tuning method for PI controllers is implemented in this paper. It is the lambda method described in (McMillan et al., 1999), (Pruna et al., 2017) and usually applied in several industrial processes.

## 1.3 Contributions

In this paper a simple power electronic system has been chosen as the basis for more complex power electronics devices such as a three-phase STATCOM. One of the goals of the authors is to illustrate to the power engineering community the value of adopting Modelica, and consequently our approach attempts to make emphasis on how Modelica can be used for typical control design tasks considering that the reader is not so familiar with Modelica. For example, the simple use of records serves to illustrate how to use them for control design and not only provide parameter data. Because the community in North America that uses Modelica is still developing, the authors believe that this paper can provide a good instructional value to the readers, especially since we are making the models available on GitHub [1].

The modeling approach was based on using the Modelica Standard Library to build all the components with diagram blocks similarly to a tool like PSCAD and Simulink. One of the advantages of using Modelica and the Modelica Standard Library is that the developed models can work in different software environments like OpenModelica, Dymola, Wolfram SystemModeler, SimulationX, etc. without the need to load external third party libraries. This

[1] https://github.com/ALSETLab/Modeling-Control-Design-Power-Electronics-Half-Bridge-Converter-Modelica

is impossible with the current tools used by the power engineering community.

## 2 Half-bridge converter applications and modeling

In this section the authors present the typical modeling approaches used to represent the half-bridge converter, which will be implemented in the next section. This converter can be seen as basic building block used to model more complex power electronics systems. As an example consider a STATCOM connected to the AC grid as shown in Figure 2. The whole system includes the DC/AC converter, the control system, the transformer and the heat exchanger. The DC/AC converter and the control system in this STATCOM are built from a basic half-bridge topology. Hence, in this paper, the main goal is to derive an implementation of the basic topology needed to build a full STATCOM converter (see Figure 3).

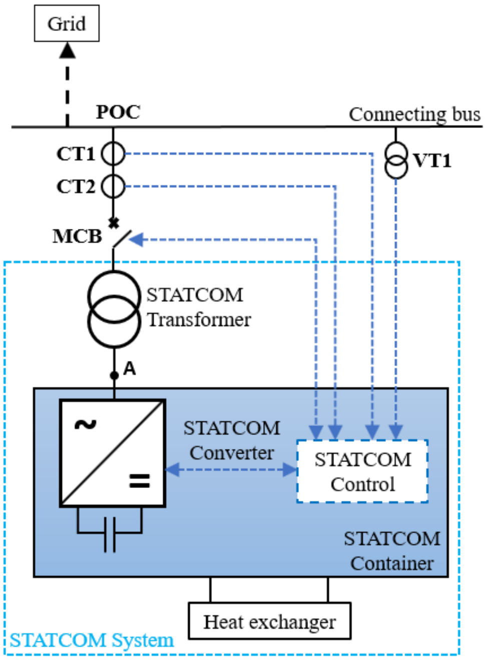


**Figure 2.** Connection of the STATCOM to the AC grid.

In other words, the system considered in this project is the first step in developing a 3-phase, three-wire, two-level VSC converter that can represent a STATCOM. The half-bridge is the basic building block needed to implement either three phase VSC's through parallel combination (see Figure 3) or multimodule VSC converters through parallel/series combination.

The two models of the half-bridge converter of this paper are represented in Figure 4 and Figure 5. They also include a resistance in series to each switch (current generator in Figure 4 and transistor in Figure 5) representing the non ideal effect of the on-state resistance of the switches.

The two models have been implemented using Model-

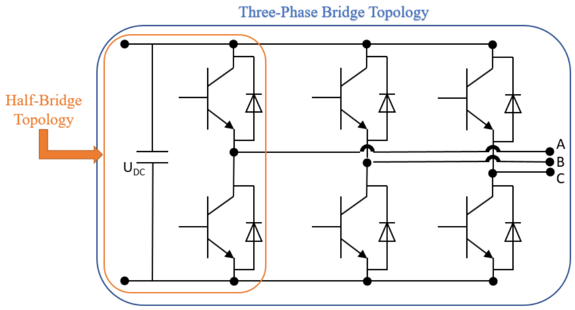


**Figure 3.** Half-bridge converter as building block of a three phase VSC converter.

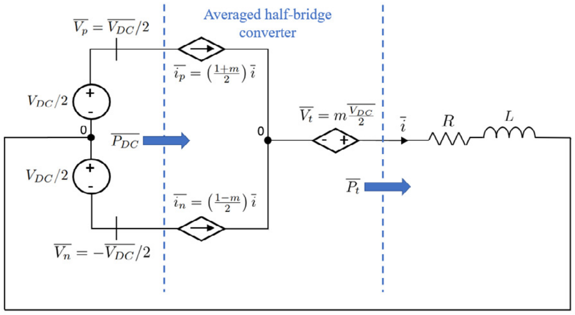

$\overline{V_p} = \overline{V_{DC}}/2$
Averaged half-bridge converter
$\overline{i_p} = \left(\frac{1+m}{2}\right)\overline{i}$
$V_{DC}/2$
+
-
0
$\overline{P_{DC}}$
0
$\overline{V_t} = m\frac{\overline{V_{DC}}}{2}$
$\overline{i}$
$R$
$L$
$V_{DC}/2$
$\overline{P_t}$
$\overline{i_n} = \left(\frac{1-m}{2}\right)\overline{i}$
$\overline{V_n} = -\overline{V_{DC}}/2$

**Figure 4.** Averaged equivalent circuit of the half-bridge converter.

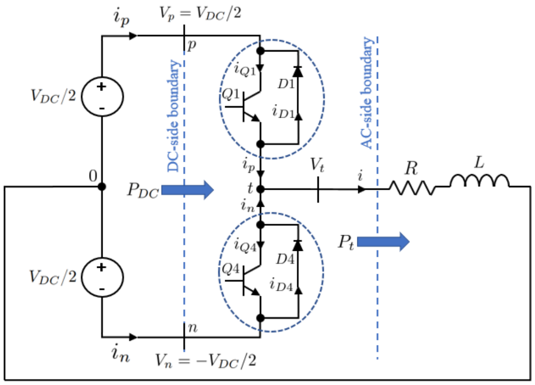


**Figure 5.** Switching equivalent circuit of the half-bridge converter.

ica language in the software environment Dymola. Modelica is an open-source object oriented language that is suitable for multidomain modeling of physical systems (Vanfretti et al., 2013).

If we look at Figure 4, the dynamics of the AC current $i$, that is the output of the system, can be described by Equation (1).

$$L\frac{di}{dt} + (R + r_{on})i = V_t \tag{1}$$

In Equation (1) $R$ and $L$ are the parameters of the AC system to which the converter is connected to and $r_{on}$ is the on-state resistance of each switch of the converter. A switch is represented in this case by an equivalent current source (see Figure 4) that is not ideal because of the parameter $r_{on}$. For the analysis of this system $r_{on}$ is lumped with the resistance $R$. From Equation (1) we can consider the AC current $i$ as state variable and $V_t = mV_{DC}/2$ as our control input since it can be changed by varying $m$, the modulating signal.

A block diagram of the system represented by Equation (1) is given in Figure 6 (Yazdani and Iravani, 2010).

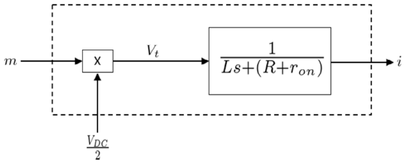
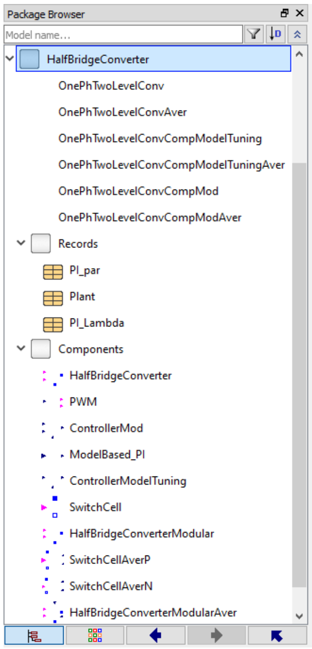


**Figure 6.** Block diagram of the AC side of the half-bridge converter.

# 3 Modelica model design and implementation

The objective of this paper is to implement the two models of the half-bridge converter in Modelica. The structure of the package created to model all the components and systems is given in Figure 7. The different averaged and switching models with and without controls are listed at the top of the package *HalfBridgeConverter*. The first two models of the list are the switching and averaged models without controls, the second couple of models represents those with a PI controller and the third couple of models represents those with a modified control strategy. A record package is used for tuning the PI controller through the lambda method. Given the input parameters of the system in the *Plant* record, the calculations of the PI parameters are performed by the formulas in the *PI_Lambda* record and the results are stored into the *PI_par* record used to set the values of the PI controller parameters. The package *Components* contains all the components used in the models listed at the top of the *HalfBridgeConverter* package. The components are created by using only the Modelica Standard Library.


Package Browser

Model name...

HalfBridgeConverter
- OnePhTwoLevelConv
- OnePhTwoLevelConvAver
- OnePhTwoLevelConvCompModelTuning
- OnePhTwoLevelConvCompModelTuningAver
- OnePhTwoLevelConvCompMod
- OnePhTwoLevelConvCompModAver
- Records
  - PI_par
  - Plant
  - PI_Lambda
- Components
  - HalfBridgeConverter
  - PWM
  - ControllerMod
  - ModelBased_PI
  - ControllerModelTuning
  - SwitchCell
  - HalfBridgeConverterModular
  - SwitchCellAverP
  - SwitchCellAverN
  - HalfBridgeConverterModularAver

**Figure 7.** Modelica package for the models of this paper.

Another task of this paper regards the regulation of the output current $i$ of the open-loop system in Figure 6 to a specified reference signal.

## 3.1 Modelica implementation

The first step consisted of modeling the system in the two versions without any control. The averaged model of the half-bridge converter in Figure 4 is represented in Figure 8.

The switching model of the half-bridge converter in Figure 5 is represented in Figure 9.

Each switch encapsulates another block diagram (see Figure 10 and Figure 11).

It should be noticed that for the averaged model the gates and their signals in Figure 8 and Figure 10 are not in use but they are already implemented for the switching model of the half-bridge converter in Figure 5. This will make easy to change the type of converter in the model of the whole system by simply changing the class of that component using the Modelica feature of replaceable.

The whole systems are then modeled as in Figure 12 and in Figure 13. The same systems with a PI controller are given in Figure 14 and in Figure 15. The output current ripple of the switching models in Figure 13 and Figure 15 can be reduced by introducing a capacitor in parallel to the series of resistance and inductance on the AC side of the converter.

The block of the controller in Figure 14 and Figure 15 is represented in Figure 16 and it includes the PI compensator in Figure 17.

Finally the systems with a more elaborated control strategy still look like as in Figure 14 and Figure 15 but in this case the controller block is represented in Figure

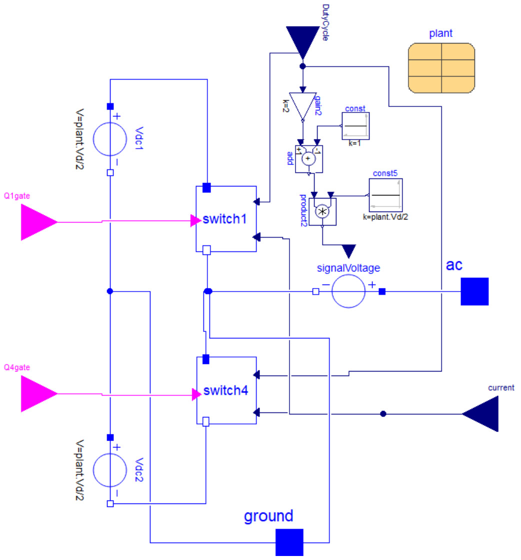


**Figure 8.** Block diagram of the averaged model of the half-bridge converter.

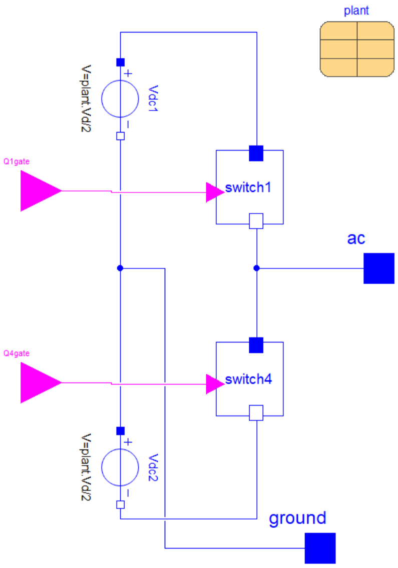


**Figure 9.** Block diagram of the switching model of the half-bridge converter.

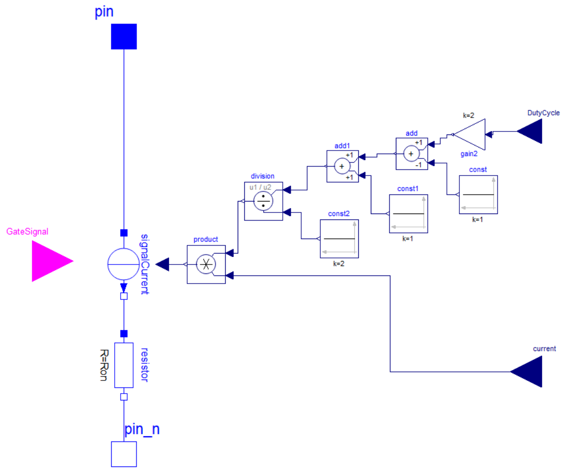


**Figure 10.** Block diagram of *switch1* in Figure 8.

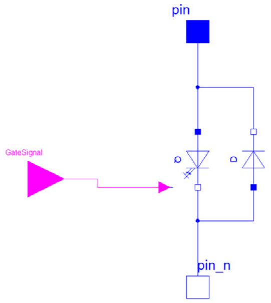


**Figure 11.** Block diagram of *switch1* in Figure 9.

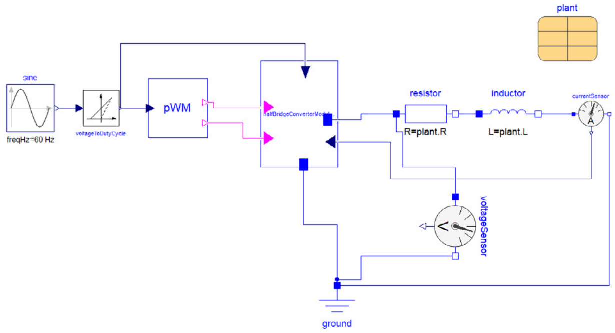


**Figure 12.** Block diagram of the system with the averaged model of the half-bridge converter without any control.

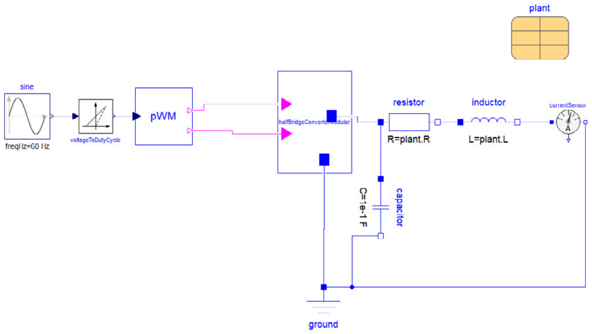


**Figure 13.** Block diagram of the system with the switching model of the half-bridge converter without any control.

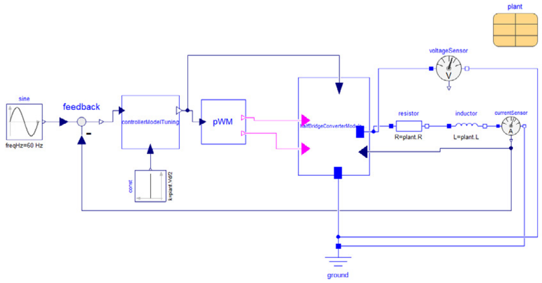


**Figure 14.** Block diagram of the system with the averaged model of the half-bridge converter with PI control.

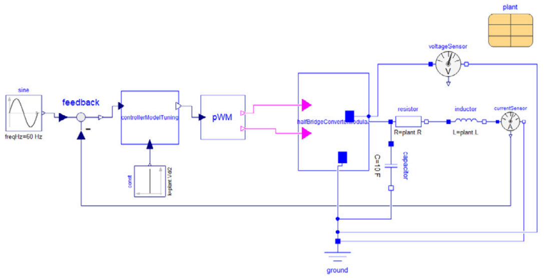


**Figure 15.** Block diagram of the system with the switching model of the half-bridge converter with PI control.

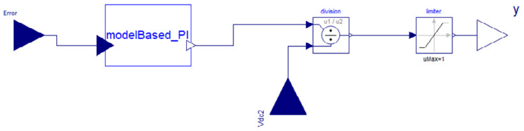


**Figure 16.** Block diagram of the controller in Figure 14 and Figure 15.

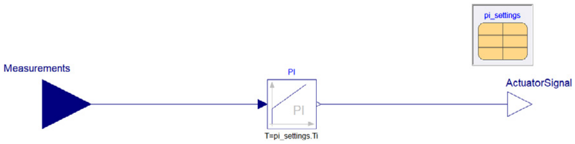


**Figure 17.** Block diagram of the PI compensator contained in the controller block of Figure 16.

18.

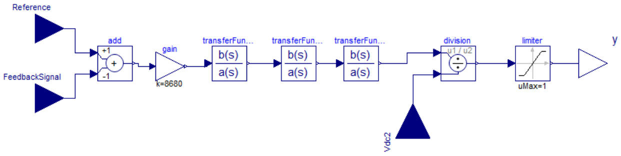


**Figure 18.** Block diagram of the controller with modified strategy for both the averaged and switching model.

In Figure 18 there is a gain and three blocks of transfer functions that have been added for this new control strategy. The three transfer functions correspond to a lead compensator, a lag compensator and a resonant control. This modification of the controller has been introduced because with only a PI controller it is not possible to track a sinusoidal reference waveform without error but only a step reference signal (Yazdani and Iravani, 2010).

## 3.2 Control design of the PI for the averaged model of the half-bridge converter

The control of the output current can be obtained by creating a closed-loop including a controller that takes as input an error signal generated by the difference between the output current $i$ and a reference current $i_{ref}$. The block diagram of the described closed-loop system is given in Figure 19.

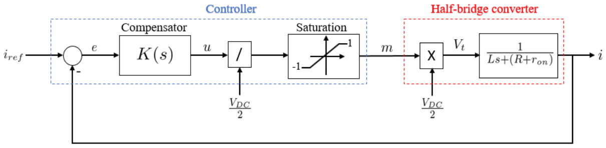


**Figure 19.** Block diagram of the closed-loop system with the half-bridge converter and a controller.

From Figure 19, the compensator $K(s)$ takes as input the error signal $e$ and gives as output the signal $u$ that is divided by $V_{DC}/2$ to compensate for the voltage gain of the converter since $V_t = mV_{DC}/2$. In addition to that, the modulating signal $m$ must satisfy $|m| \leq 1$ so the *Saturation* block has been introduced in the block diagram.
Depending on the reference signal and the performance to match different compensators may be used. If the reference signal $i_{ref}$ is a step function then a proportional-integral (PI) compensator can be considered for control purposes. Its generic descriptive expression is:

$$K(s) = \frac{k_p s + k_i}{s} \tag{2}$$

From Equation (2) and Figure 19 the open-loop transfer function of the system can be derived:

$$G_{open}(s) = K(s)G(s) = \left(\frac{k_p}{Ls}\right)\left(\frac{s + \frac{k_i}{k_p}}{s + \frac{R+r_{on}}{L}}\right) \tag{3}$$

From Figure 6 it is possible to see that the open-loop system has a stable pole at $p = -(R + r_{on})/L$. In general, considering the typical values for $R$, $r_{on}$ and $L$, the pole $p$ is quite close to the origin giving a slow natural response. So to improve the open-loop frequency response the pole $p$ can be cancelled by the zero of the PI compensator. Then, from Equation (3), it is possible to choose for the compensator $k_i/k_p = (R + r_{on})/L$ and $k_p/L = 1/\tau_i$ where $\tau_i$ is the desired time constant of the closed-loop system. With the previous choices the transfer function of the closed-loop system becomes:

$$G_{closed}(s) = \frac{K(s)G(s)}{1 + K(s)G(s)} = \frac{i}{i_{ref}} = \frac{1}{\tau_i s + 1} \qquad (4)$$

The time constant $\tau_i$ should be small enough to have a fast response of the current control. But it has also to be considered that $1/\tau_i$ should be smaller than the switching frequency of the half-bridge converter. Considering the mentioned aspects $\tau_i$ is usually in the range $0.5 - 5ms$ and it can vary depending on the specific application of the converter and its switching frequency (Yazdani and Iravani, 2010).

## 4 Simulations

The analysis of results can start by considering as reference input of the half-bridge converter a step function with a step of amplitude $50A$ applied at $0.1s$. From Figure 12 we can just replace the sinusoidal source with a step block that is used to calculate the duty cycle $d$ necessary for the modulation factor $m$ ($m = 2d - 1$) that is one of the inputs of the block *halfBridgeConverter*. The output of the *halfBridgeConverter* is connected to the AC system characterized by the components $R$ and $L$. The current sensor in series with $R$ and $L$ is used to measure the AC output current $i$ that is fed back as input of the *halfBridgeConverter* and that will be controlled in the following steps.

As application example, we can consider the following parameters values (Yazdani and Iravani, 2010):

- $L = 690\mu H$
- $R = 5m\Omega$
- $r_{on} = 0.88m\Omega$
- $\tau_i = 5ms$

Applying these parameters to the system in Figure 12 we can run a simulation and plot the step input and the output AC current of the system. The results are in Figure 20.

As we can see from Figure 20 the AC output current does not track the step input. The final value of the output AC current is much higher than the step input.
Then a PI compensator is introduced. From the parameters previously defined, we can derive $k_p = L/\tau_i = 0.138$ and

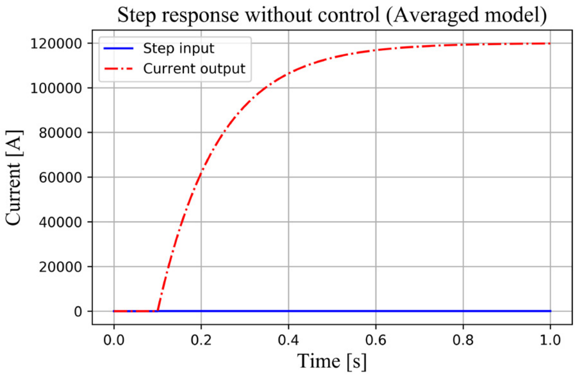


**Figure 20.** Plot of the step input (blue curve) and AC output current (red curve) of the system with averaged model without controls. Due to the difference of magnitudes between the two curves the step of $50A$ cannot be seen in the figure.

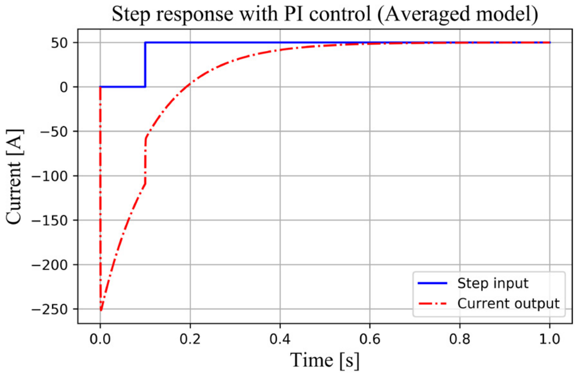


**Figure 21.** Plot of the step input (blue curve) and AC output current (red curve) of the system with averaged model with PI controller.

$k_i = k_p(R + r_{on})/L = 1.176$. We can run a new simulation and the results are illustrated in Figure 21.

From Figure 21 we can see that now the system is able to settle at a final value corresponding to the step input value by using a PI controller. The initial negative overshoot of the current is due to the initial conditions of the system.

If we now consider the same conditions for the system with the switching model of the half-bridge converter, applying the same step without controls we get the results in Figure 23. Introducing the same PI control of the previous case we get the results in Figure 24. The value of the capacitance C at the output of the switching model of the converter can affect the output current ripple. There are also other requirements that need to be met, for example, the admissible voltage ripple on the load and a reasonable value of the capacitance C. An example is shown in Figure 22 where the switching model of the half-bridge converter without any control has been used. It shows the impact of the variation of the capacitance on the output current.

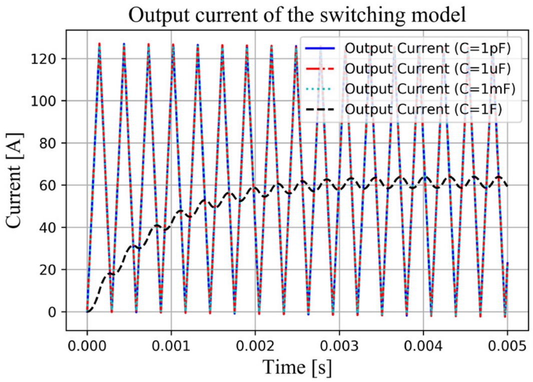


**Figure 22.** Plot of the AC output current of the system with switching model without controls for different values of the output capacitance C.

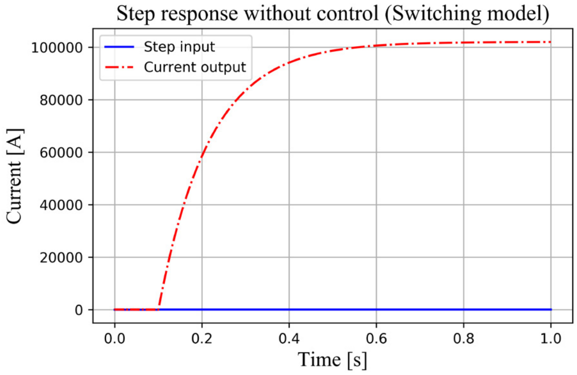


**Figure 23.** Plot of the step input (blue curve) and AC output current (red curve) of the system with switching model without controls. Due to the difference of magnitudes between the two curves the step of $50A$ cannot be seen in the figure.

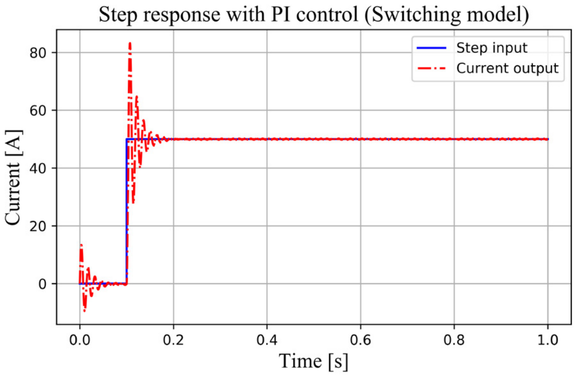


**Figure 24.** Plot of the step input (blue curve) and AC output current (red curve) of the system with switching model with PI controller.

From Figure 20 and Figure 23 we can see that without any control the output current is not able to track the step reference either with the averaged model or with the switching model. Introducing a PI controller, from Figure 21 and Figure 24, the results are better even if we have a large negative overshoot for the system with the averaged model of the half-bridge converter and some oscillations of the output current at the initialization of the system with the switching model when the step is applied.

The next test consists of changing the source of the reference signal with a sinusoid of amplitude 1000 and frequency $60Hz$ starting at $t = 0.07s$. Running a simulation we get the results in Figure 25 for the system with the switching model without any control.

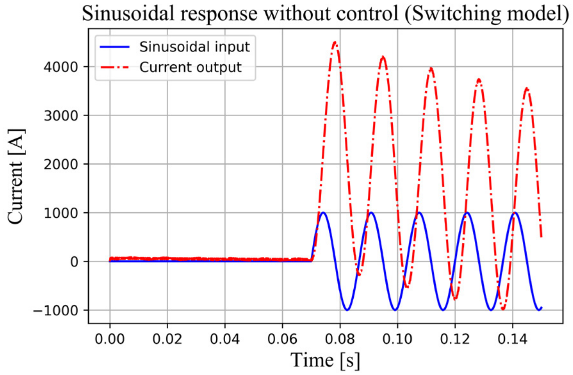


**Figure 25.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the switching model without controls.

For the system with the averaged model without any control we get the results in Figure 26.

From Figure 25 and Figure 26 the behavior of the systems with the switching model and the averaged model without any control looks very similar but not able to track the reference signal.

When introducing the same PI controller with the sinu-

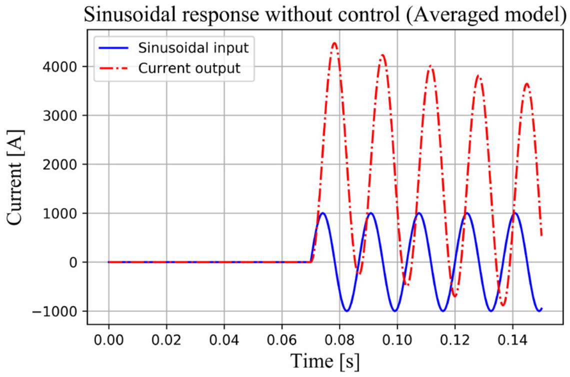


**Figure 26.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the averaged model without controls.

soidal reference signal we get the results in Figure 27 for the system with the switching model and the results in Figure 28 for the system with the averaged model.

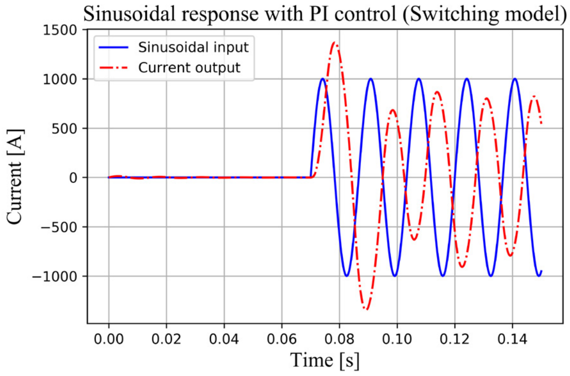


**Figure 27.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the switching model with PI controller.

From Figure 27 and Figure 28 we can see that the behavior of the system is better with the averaged model of the half-bridge converter than with the switching model. They both present errors in amplitude and phase compared to the reference but the averaged model looks like more aligned with the reference.
Then introducing a more elaborated control strategy described in 3.1 a new simulation with the sinusoidal reference can be performed. The results for the system with the switching model are in Figure 29 and for the system with the averaged model in Figure 30.

From Figure 29 and Figure 30 the behavior of the system with the switching model is better than the one with the averaged model since the tuning of the control parameters has been performed on the switching model.
In order to get a similar behavior of the system with the averaged model, like in Figure 31, the gain of the controller has been increased from a value of 8680 to 450000.

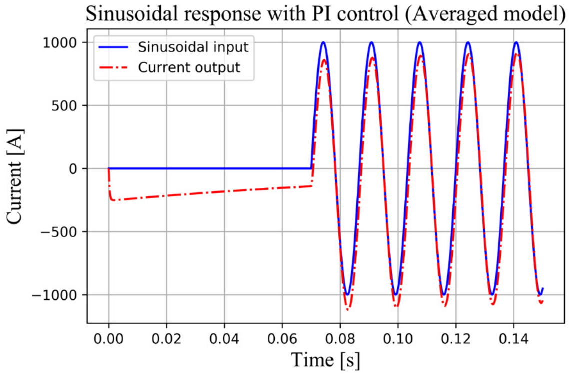


**Figure 28.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the averaged model with PI controller.

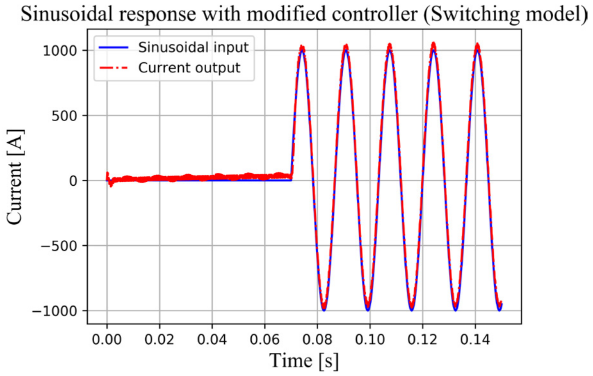


**Figure 29.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the switching model with modified controller.

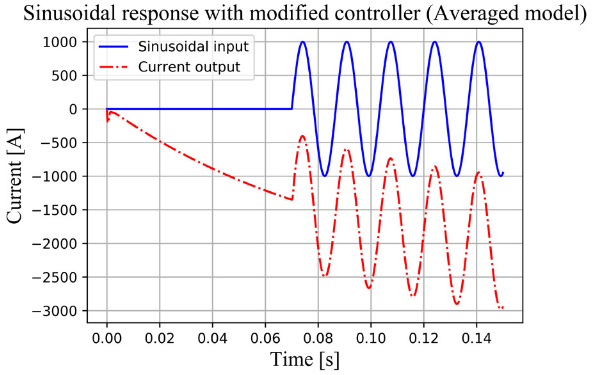


**Figure 30.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the averaged model with modified controller.

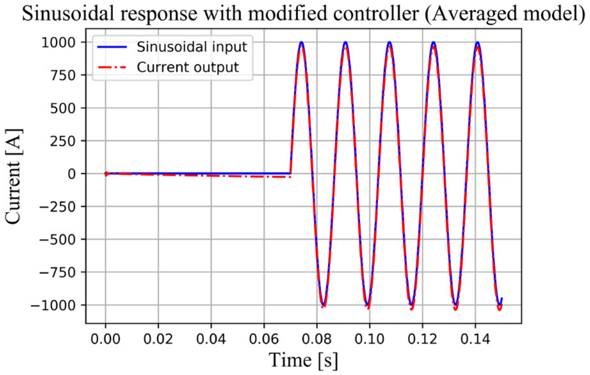


**Figure 31.** Plot of the sinusoidal input (blue curve) and AC output current (red curve) of the system with the averaged model with modified controller.

The voltage distortion due to voltage drop at grid's R-L due to non-sinusoidal currents of switching converter has not been shown because the focus of the paper is on the control of the output current of the half-bridge converter. There are also other aspects that can be analyzed starting from what has been presented like voltage distortion, THD content, other control strategies and so on. They can be studied in a future work.

## 5 Further Work and Conclusions

The modularity of blocks of components and records allows for an easy reuse in different models. This allowed to run simulations very quickly. One of the objectives of this paper is to communicate to the power system analysts the benefits of Modelica language features, such as the replaceable and redeclare features. In most power system tools, the user would have to set up different system models when using different representations. In future work, the authors plan to illustrate how we can use these features by changing from detailed component models, to the ideal switches, to the averaged value model. Simplifying model development and management.

The simulations show that the introduction of a feedback control improves the reference tracking. A specific combination of control parameters can work for a system with the switching model of the half-bridge converter but not for the one with the averaged model or viceversa. This will also be the case if we consider a more detailed or less detailed model of the switches, which will be illustrated in a future work. So for the tuning of the controls other tools can be considered for the estimation of their parameters. Other control strategies can be analyzed, for example, to reduce the initial overshoot of the output current of the system with the averaged model of the half-bridge or the initial oscillations of the system with the switching model. They represent an additional stress for the components of the converter.

The implementation of a three-phase converter will follow this work that is an initial step. So different control strategies can be taken into account for this more complex case.

## References

Rik De Doncker, Duco WJ Pulle, and André Veltman. *Advanced electrical drives: analysis, modeling, control*. Springer Science & Business Media, 2010.

Anton Haumer and Christian Kral. Modeling a mains connected pwm converter with voltage-oriented control. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 63, pages 388–397. Linköping University Electronic Press, 2011.

JM Maza-Ortega, E Acha, S García, and A Gomez-Exposito. Overview of power electronics technology and applications in power generation transmission and distribution. *Journal of Modern Power Systems and Clean Energy*, 5(4):499–514, 2017.

Gregory K McMillan, Douglas M Considine, et al. *Process/industrial instruments and controls handbook*, volume 7. McGraw Hill, 1999.

Andreas Olenmark and Jens Sloth. Flexible ac/dc grids in dymola/modelica-modeling and simulation of power electronic devices and grids. *CODEN: LUTEDX/TEIE*, 2014.

Edwin Pruna, Edison R Sasig, and Santiago Mullo. Pi and pid controller tuning tool based on the lambda method. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6. IEEE, 2017.

Muhammad H Rashid. *Power electronics handbook*. Butterworth-Heinemann, 2017.

Hani Saad, Sébastien Dennetière, and Jean Mahseredjian. On modelling of mmc in emt-type program. In *2016 IEEE 17th Workshop on Control and Modeling for Power Electronics (COMPEL)*, pages 1–7. IEEE, 2016.

Luigi Vanfretti, Wei Li, Tetiana Bogodorova, and Patrick Panciatici. Unambiguous power system dynamic modeling and simulation using modelica tools. In *2013 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2013.

Michael Winter, Sascha Moser, Stefan Schoenewolf, Julian Taube, and Hans-Georg Herzog. Average model of a synchronous half-bridge dc/dc converter considering losses and dynamics. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 479–484. Linköping University Electronic Press, 2015.

Amirnaser Yazdani and Reza Iravani. *Voltage-sourced converters in power systems*, volume 34. Wiley Online Library, 2010.

Daniel Zammit, Cyril Spiteri Staines, and Maurice Apap. Compensation techniques for non-linearities in h-bridge inverters. *Journal of Electrical Systems and Information Technology*, 3 (3):361–376, 2016.

Modelica