PROCEEDINGS

Asian Modelica Conference 2024

December 12-13

International Convention Center Jeju

Linköping Electronic Conference Proceedings Nr. 217

Proceedings of the Asian Modelica Conference 2024
Jeju, Korea, December 12 – 13, 2024

Editor
Dr. Andrea Neumayr, DLR, Germany

Published by
Modelica Association and Linköping University Electronic Press

Series: Linköping Electronic Conference Proceedings Nr. 217
ISBN: 978-91-8118-212-5
ISSN: 1650-3686
eISSN: 1650-3740
DOI: https://doi.org/10.3384/ecp217

Organized by
iVH Co., Ltd.
3F, 19, Yangjaecheon-ro 17-gil, Seocho-gu
Seoul, 06754, Korea

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

Conference location
International Convention Center Jeju, Korea

WELCOME TO ASIAN MODELICA CONFERENCE 2024

Welcome to the Asian Modelica Conference 2024 here in Jeju. It is a great honor to have you all join us for what promises to be an enlightening and inspiring event.

The concept of Modelica is revolutionizing industries, from manufacturing to healthcare, urban planning to energy management. Today, we gather to delve into the latest advancements, share innovative ideas, and explore the vast potential of digital twin technology.

This conference brings together a diverse group of experts, innovators, and enthusiasts from around the world. Each of you plays a crucial role in shaping the future of digital twins, and your presence here is a testament to the importance and impact of this technology.

Over the next 2 days, we will have the opportunity to engage in thought-provoking discussions, attend insightful presentations, and participate in hands-on workshops. These sessions are designed not only to educate but also to inspire, challenge, and drive forward our collective vision for the future of digital twins.

I would like to extend my heartfelt gratitude to our distinguished speakers, sponsors, and organizers. Your dedication and hard work have made this event possible, and we are deeply appreciative of your contributions.

As we embark on this journey together, I encourage each of you to take full advantage of the opportunities to connect, collaborate, and learn. Let's make this conference a catalyst for innovation and a platform for forging lasting partnerships.

Thank you, and once again, welcome to the Asian Modelica Conference 2024. Let's create a truly remarkable and impactful event.

Yongha Han
Conference Chair

Daeoh Kang
Program Chair

DOI
10.3384/ecp217

WELCOME TO ASIAN MODELICA CONFERENCE 2024

CONFERENCE BOARD MEMBERS

Conference Chair

Yongha Han	Name
Research Fellow, HMC DLR	Affiliation
Korea	Nationality

Program Chair

Daeoh Kang	Name
CEO, iVH	Affiliation
Korea	Nationality

Board Members

Name	Affiliation	Nationality
Prof. Martin Otter	Professor, DLR	Germany
Hubertus Tummescheit	Chief Solutions Officer, Modelon	Sweden
Woongcheol Choi	Professor, Kookmin university	Korea
YoonJei Hwang	LG Electronics	Korea
Byoungdoo Lee	Team leader, Hyundai E&C Technology Research Center	Korea
Dr. Rui Gao	RIGO TECH Co., Ltd	Japan
HyungSik Um	Samsung Electronics	Korea
EungSoo Kim	Professor, Seoul National University	Korea
Juneyoung Song	Hyundai Mobis	Korea
Prof. Martin Sjölund	Professor, Linköping University	Sweden

DOI
10.3384/ecp217

Reviewers

Prof. Martin Otter	DLR
Dr. Dirk Zimmer	DLR
Dr. Daeoh Kang	iVH
Dr. Rui Gao	Rigo Tech
Yongha Han	HMC
Prof. Francesco Casella	Politecnico di Milano
Dr. Wonyul Kang	iVH

SCOPE OF ASIAN MODELICA CONFERENCE 2024

Modelica is a freely available, equation-based, object-oriented language for convenient and efficient modeling of complex, multi-domain cyber-physical systems described by ordinary differential, difference and algebraic equations. The Modelica language and the companion Modelica Standard Library have been utilized in a variety of demanding industrial applications, including full vehicle dynamics, power systems, robotics, buildings and district energy systems, hardware-in-the-loop simulations and embedded control systems. The Functional Mock-up Interface (FMI) is an open standard for the tool-independent exchange of models and for co-simulation. It is supported by many Modelica and non-Modelica tools and is the key to utilizing Modelica models in non-Modelica environments. **Development in the Modelica Association is organized in Modelica Association Projects:**

LANG	Modelica Language
LIB	Modelica Libraries
FMI	Functional Mock-up Interface
eFMI	Functional Mock-up Interface for embedded systems
SSP	System Structure and Parameterization of Components for Virtual System Design
DCP	Distributed Co-Simulation Protocol

DOI
10.3384/ecp217

These projects collaborate to design and maintain a set of coordinated standards for modeling and simulation of complex physical systems.The Modelica conference will bring together people using Modelica and/or other Modelica Association standards for modeling, simulation, and control applications, such as Modelica language designers, tool vendors and library developers. The Modelica Conference provides Modelica users with the opportunity to stay informed about the latest language, library, and tool developments, and to get in touch with people working on similar modeling problems. **The conference will cover topics such as the following:**

Multi-engineering modeling and simulation with free and commercial Modelica libraries (mechanics, electrical, hydraulics, thermal, fluid, media, chemical, building, automotive, aircraft, ...)
Automotive applications
Thermodynamic and energy systems applications
Mechatronics and robotics applications
Medicine and biology applications
Other industrial applications, such as electric drives, power systems, aerospace, etc.
Large-scale system modeling
Real-time and hardware-in-the-loop simulation
Simulation and code generation for embedded control systems
Simulation acceleration by use of many CPU cores or GPU cores
Applications of Modelica for optimization and optimal control
Modelica modeling, simulation and design tools
Symbolic algorithms and numerical methods for model transformation and simulation

[KEYNOTE SPEAKERS] Speech 1

Yongha Han

Research Fellow
Hyundai Motor Group

Speaker Bio

YongHa Han is Research Fellow at Hyundai Motor Group. He joined Hyundai Motor Group in 1996 in Korea. He has been working in the areas of crash safety performance development and virtual technology development. He was appointed as a research fellow in charge of future core technologies in 2020 and is in charge of the Virtual Technology Innovation Research Lab.

(Hyundai Motor Group has been operating a research fellow system since 2009, and has been supporting top R&D experts to focus on their original research work, free from the burden of management.)

Recent research has mainly focused on three areas: ① Development of safety and NVH performance improvement solutions for electric vehicles based on virtual models of batteries and motors ② Development of high-accuracy models simulating new manufacturing processes such as giga casting, and development of solutions to map manufacturing effect on performance analysis model ③ Development of Innovative body and chassis mechanism solution and NVH active vibration control technology to increase the completeness of future mobility.

DOI
10.3384/ecp217

Abstract

How can we stay competitive continuously?
Virtual vehicle development strategy and use cases

Recently, three trends are buffeting R&D in the automotive industry :

① The transition from the combustion engine to electric vehicle technology

② Software-defined vehicles with increased customer centricity

③ Automated vehicles and the emergence of generative AI (gen AI).

Under these circumstances, the increasing complexity of development requires exponentially increasing costs and time, and it is essential to respond effectively through virtual development. Virtual vehicle development refers to all activities of developing a vehicle using a virtual model in the entire process from planning to sales. Recently, as a virtual vehicle development strategy, Model Based System Engineering (MBSE) has become a hot topic, and MBSE has the advantage of expressing system processes and characteristics as models, enabling smooth communication between stakeholders, and easily analyzing the overall system.

Hyundai Motor Group defines a practical MBSE from the perspective of vehicle development, which is to structure and systematize tasks using a descriptive model and efficiently implement (V-type) development using an executable model. In this presentation, the process, application cases, and issues related to the integration and operation of the virtual model, as well as several use cases related to virtual vehicle development with a focus on concept engineering and virtual reality verification will be given.

[KEYNOTE SPEAKERS] Speech 2

Dr. Moritz Hübel

Industry Director for Energy & Process
Modelon

Speaker Bio

Moritz Hübel is Industry Director for Energy & Process at Modelon. He joined Modelon in 2019 in Hamburg, Germany and has been working with providing modelling and simulation solutions for customers from the Energy & Power industries and managing Modelon's global team of energy experts. Prior to joining Modelon, he has been working with customer and research projects for the Center of Combustion Engines and Thermodynamics (FVTR GmbH), managing the energy system simulation team. Moritz received his PhD in Thermodynamics & Power engineering from Rostock University in 2016. His PhD project was focused on flexibility optimization of large-scale thermal power plants using Modelica solutions for thermodynamic system simulation. He also holds a degree in mechanical engineering with a focus on energy systems and thermodynamics.

DOI
10.3384/ecp217

Abstract

Finding the right balance

What technologies can complement renewable energy to achieve true sustainability?

The global situation of energy supply today is dominated by uncertainties: existing energy systems need to be converted from conventional, mostly fossil generation towards more sustainable, low carbon-emission solutions. Energy resource supply chains on a global scale introduce political dependencies and complexities. Industries and societies opt for cost efficient solutions. Emerging technologies for renewable energy generation, storage and distribution are being developed with largely unknown future cost structure and scalability. In that context, investment decisions need to be made, especially in the energy industry usually require large investments and long payback periods. Engineers working with Modelica know how to tackle this challenge on a fundamental basis: the large set of unknowns needs to be addressed with an equally large number of equations.

The problem needs to be balanced with knowledge. While system simulation cannot give a single universally applicable answer to such problems, its usage helps to remove the unknows: energy balances are not negotiable and neither is the selection of cost competitive supply options under given legal, social and reliability constraints. The complexity of the problem can be reduced and broken down in a system model. Some uncertainties can be captured with empirical correlations or learning of data. In combination with the ability to quantify remaining uncertainty, e.g. future price data, robust technology decisions can be identified that are truly sustainable. In this presentation an overview of how Modelica can help make such choices will be given.

SPONSORS

Platinum Sponsors

Gold Sponsor

Silver Sponsor

DOI
10.3384/ecp217

Content

DOI
10.3384/ecp217

Combining Equation-based and Multibody Models

Andrea Neumayr[1] Martin Otter[1]

[1]German Aerospace Center (DLR), Institute of System Dynamics and Control (SR), Germany,
{andrea.neumayr,martin.otter}@dlr.de

Abstract

This article highlights the combination of equation-based modeling with multibody models. In other words, it combines the equation-based modeling language Modia and the multibody module Modia3D. The multibody system is defined in an object-oriented way and parts of it are defined by equations. Algebraic loops are treated that appear due to the connection of multibody and equation-based components. A new approach to variable structure systems are so-called predefined acausal components which consist of pre-compiled causal parts and acausal equations. To generalize the concepts for variable structure systems, the multibody module is defined as a predefined acausal component. As a result, the number of degrees of freedom of the multibody system can vary during simulation. This is demonstrated with a non-trivial example of a walking space robot from the MOSAR space project.

Keywords: Modia, Modia3D, Modelica, Julia, multibody, variable structure systems, segmented simulation

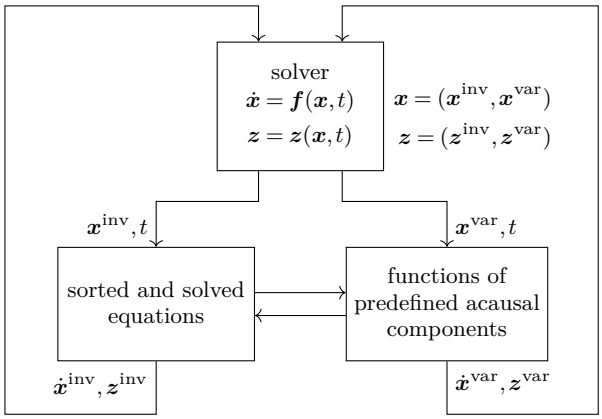


Figure 1. Predefined acausal component. Communication between the solver, the sorted and solved equations, and the functions of the predefined acausal components. The state vector $\boldsymbol{x}$ and the event indicators $\boldsymbol{z}$ are split into an invariant and a variant part: $\boldsymbol{x} = (\boldsymbol{x}^{\text{inv}}, \boldsymbol{x}^{\text{var}})$, $\boldsymbol{z} = (\boldsymbol{z}^{\text{inv}}, \boldsymbol{z}^{\text{var}})$. The variant parts consist of the states defined and used in the causal partitions of all predefined acausal components. The dimensions of the invariant parts are fixed before simulation begins. The dimensions of the variant parts can change at events during simulation.

1 Introduction

This publication is about generating Ordinary Differential Equations (ODEs) from equation-based models that are combined with variable structure multibody models. It summarizes, combines, and provides a deep insight into the findings of previous publications on symbolic transformations in Modia (Otter and Elmqvist 2017; Elmqvist et al. 2021), and iterative solving of multibody systems with Modia3D (Neumayr and Otter 2019), and variable structure systems (Neumayr and Otter 2023a; Neumayr and Otter 2023b). For this purpose, parts of previous publications are briefly repeated.

Modia is an equation-based modeling and simulation environment. It is inspired by the modeling language Modelica and has similar semantics. Modia is a domain-specific extension of the Julia programming language[1] (Bezanson et al. 2017). Modia3D is an open source multibody module with a modular and customizable component-based design pattern and is closely integrated with Modia. Furthermore, Modia supports a new approach to variable structure systems with predefined acausal components. Modia3D is one such component.

All current proposals for variable structure systems, e.g., (Mehlhase 2014; Mattsson, Otter, and Elmqvist 2015; Tinnerholm, Pop, and Sjölund 2022) require prior knowledge of all models and all modes in order to switch between these models during simulation. If this information is not available, and whenever the equation structure changes, the entire model is reprocessed and its code is regenerated and recompiled (or interpreted), e.g., (Zimmer 2010; Tinnerholm, Pop, and Sjölund 2022).

Neumayr and Otter (2023a) and Neumayr and Otter (2023b) introduce a new general concept for dealing with variable structure systems in which variables can appear and disappear during simulation. The two previous publications are briefly summarized below. There is no need to regenerate and recompile code when the number of equations and states changes at events. The method can be applied to declarative, equation-based modeling languages, such as Modia and Modelica. The transition between the modes,

[1]The pseudocode snippets in this publication are Julia-like.

called segments, is triggered by specific commands. Both the number of variables and the number of equations can vary from segment to segment.

The idea is to introduce predefined acausal components. Their equations are split into causal and acausal partitions. The causal partition is always evaluated in the same order, regardless of how the component is connected to components. This partition is sorted, explicitly solved for the unknowns, and implemented with one or more functions. The acausal partition is a set of equations that is sorted and solved. A large part of the variables in the causal partitions are hidden as local variables in functions and passed directly to the solver. This leads to the concept in Figure 1.

Based on this generic concept, this article shows how it can be applied to a class of multibody models implemented as a predefined acausal component.

2 Mathematical Descriptions

2.1 DAEs and ODEs

In equation-based modeling languages physical systems are described mathematically by Differential Algebraic Equations (DAEs) (1)

$$\boldsymbol{F}(\dot{\boldsymbol{x}}_{\mathrm{DAE}}, \boldsymbol{x}_{\mathrm{DAE}}, \boldsymbol{w}_{\mathrm{DAE}}, \boldsymbol{u}, t) = 0, \tag{1}$$

where $\boldsymbol{x}_{\mathrm{DAE}}(t)$ are variables appearing differentiated in the model, $\boldsymbol{w}_{\mathrm{DAE}}(t)$ are algebraic variables that are not differentiated, and $\boldsymbol{u}(t)$ are model inputs. These vectors depend on time $t \in \mathcal{R}$. $\boldsymbol{F}$ represents the equations of the system.

On the one hand, DAEs (1) can be solved numerically with DAE solvers such as DASSL (Brenan, Campbell, and Petzold 1996) or IDA from the Sundials suite (Hindmarsh, Serban, and Collier 2015). This approach has some limitations. For this reason, there are solvers for DAEs with a particular structure (Arnold 2017) that have much better numerical properties.

On the other hand, a system of DAEs $\boldsymbol{F}$ (1) can be transformed into Ordinary Differential Equations (ODE) in state-space form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \tag{2}$$

and solved with ODE solvers. The non-trivial transformation from an implicit DAE system to an explicit ODE system can be performed symbolically and automated by any compiler of equation-based modeling languages. If the structure of the physical system changes during simulation – known as a variable structure system – so does its underlying mathematical description represented by DAEs and its corresponding ODEs. Therefore, it would be required to execute the computationally expensive transformation and compilation from DAEs to ODEs again.

2.2 Multibody Equations

The equations of motion of a multibody system with kinematic loops are described as follows, see e.g., (Arnold 2017):

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} \\ \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{G}^T(\boldsymbol{q},t)\boldsymbol{\lambda} + \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) &= \boldsymbol{\tau} \\ \boldsymbol{0} &= \boldsymbol{g}(\boldsymbol{q},t), \end{aligned} \tag{3}$$

where $\boldsymbol{q}$ are the generalized coordinates of the joints of the spanning tree (such as the angle of a revolute joint), $\boldsymbol{v}$ are the derivatives of $\boldsymbol{q}$, $\boldsymbol{\tau}$ are the generalized forces in the joints of the spanning tree (such as the driving torque of a revolute joint), $\boldsymbol{\lambda}$ are the generalized forces/torques in the cut-joints, $\boldsymbol{M} = \boldsymbol{M}^T$ is the positive definite mass matrix, $\boldsymbol{g}$ are the kinematic constraint equations of the cut-joints on position level, $\boldsymbol{G} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}$ are the partial derivatives of the constraint equations with respect to $\boldsymbol{q}$ and has full row rank, and $\boldsymbol{h}$ are applied generalized forces. This DAE with index 3 gives rise to numerical problems when integrating it directly. Instead, with the method of Gear, Leimkuhler, and Gupta (1985) and Gear (1988) it can be transformed to a DAE with index 1 (4), see (Otter and Elmqvist 2017; Neumayr and Otter 2019) with much more beneficial numerical properties:

$$\begin{aligned} \boldsymbol{0} &= \dot{\boldsymbol{q}} - \boldsymbol{v} + \boldsymbol{G}^T(\boldsymbol{q},t)\dot{\boldsymbol{\mu}}_{int} \\ \boldsymbol{0} &= \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{G}^T(\boldsymbol{q},t)\dot{\boldsymbol{\lambda}}_{int} + \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) - \boldsymbol{\tau} \\ \boldsymbol{0} &= \boldsymbol{g}(\boldsymbol{q},t) \\ \boldsymbol{0} &= \boldsymbol{G}(\boldsymbol{q},t)\boldsymbol{v} + \boldsymbol{g}^{(1)}(\boldsymbol{q},t), \end{aligned} \tag{4}$$

where:

1. The derivative of the constraint equations $\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{q},t)$ are added as new equations.
2. New unknowns $\dot{\boldsymbol{\mu}}_{int}$ are introduced to stabilize the DAE.
3. The generalized constraint forces $\boldsymbol{\lambda}$ are replaced by $\dot{\boldsymbol{\lambda}}_{int}$ the derivatives of its integral.

In the following, the focus is on the special case of tree-structured multibody systems where (3) and (4) simplify to the index 1 DAE

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} \\ \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) &= \boldsymbol{\tau}. \end{aligned} \tag{5}$$

This equation can be transformed into the ODE

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{M}^{-1}(\boldsymbol{q},t)\left(\boldsymbol{\tau} - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t)\right) \\ &= \boldsymbol{f}_{\mathrm{mbs}}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{\tau},t). \end{aligned} \tag{6}$$

Conceptually, it is easy to define this multibody model as a predefined acausal component: The ODE

DOI
10.3384/ecp21717

(6) is part of the sorted and solved equations. The function $\boldsymbol{f}_{\text{mbs}}$ is part of the functions of the predefined acausal component, in Figure 1. However, this approach has serious drawbacks. Therefore, it is handled differently in Modia/Modia3D. The following issues are discussed in detail in the following sections:

1. *Object-oriented definition of multibody system.*
   A multibody model consists of various components, such as bodies, joints, and force elements. Users expect to drag and combine these elements individually with equation-based components. An example is shown in Figure 2 as a Modelica object-diagram. The multibody components `body, rev, world` are combined with components from equation-based libraries. The corresponding Modia/Modia3D model is in Listing 1. In section 3 is explained, how to treat the multibody components as specially marked parameters. These are used to inject equations before symbolic processing begins.

2. *Algebraic loops between multibody and equation-based models.*
   Algebraic loops can occur between multibody systems (5) and equation-based components. For example, if $\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{q}, \boldsymbol{v}, \dot{\boldsymbol{v}}, t)$ due to the connection structure. Figure 2 is an example that contains an algebraic loop due to the connection of the rotational components `motorInertia, gear` to the flange of the revolute joint `rev`. Modia/Modia3D treat such algebraic loops efficiently, see section 4. For example, code-size grows linearly with the number of iteration variables.

3. *Variable structure multibody systems.*
   In Modia3D the structure of the multibody system and its degrees of freedom can vary during simulation. A non-trivial example explains how to generalize the newly introduced concepts, in section 5.

3 Object-Oriented Definitions of Multibody Systems

A Modia/Modia3D model[2] of a one-arm robot with a drive train is sketched in Listing 1 to briefly recap the object-oriented definitions of multibody systems. Parts are already published in Elmqvist et al. (2021). A corresponding Modelica object-diagram is shown in Figure 2.

Listing 1. Modia/Modia3D model of a one-arm robot with motor, ideal gear and cascaded P-PI controller that drives the flange of a revolute joint.

```
Servo = Model3D(
  world = Object3D(feature=Scene()),
  body  = Object3D(feature=Solid(...)),
  rev   = RevoluteWithFlange(
    obj1=:world, obj2=:body, axis=3,
    phi=Var(init=0.0), w=Var(init=0.0)),

  ramp          = Ramp,
  ppi           = Controller,
  wSensor       = UnitlessSpeedSensor,
  motorInertia  = Inertia,
  gear          = IdealGear,
  connect = :[
    (ramp.y, ppi.refGain)
    (gear.flangeB, rev.flange)
    ...])

servo = @instantiateModel(Servo)
simulate!(servo, stopTime=...)
```

A Modia model is defined with the predefined dictionary `Model`. All parts of the model are declared with name/value pairs. Parameters are defined with the predefined dictionary `Par`[3]. A Modia3D model is defined with the predefined dictionary `Model3D`. It may contain Modia components, see Listing 1. The instances `world, body, rev` of multibody components are individually defined and combined with instances `ramp, ppi, wSensor, motorInertia, gear` of equation-based Modia components.

Multibody components, such as `Object3D, RevoluteWithFlange`, are defined as very simple Modia components, see Listing 2 and Listing 3. They contain enough information to transform an instance of such a component into acausal and causal partitions before symbolic processing begins. This is a generic Modia approach for predefined acausal components and not specific to multibody systems.

Listing 2. Definitions of multibody components as special parameters.

```
Object3D(; kwargs...) = Par(; kwargs...,
  _constructor = :(Modia3D.Object3D))

Solid(; kwargs...) = Par(; kwargs...,
  _constructor = :(Modia3D.Solid))
```

The components in Listing 2 are defined as Julia functions with keyword arguments. All provided keyword arguments are collected by variable `kwargs...`. The function body consists of one constructor `Par`. It creates a dictionary that defines a parameter consisting of the specified keyword arguments `kwargs...`, and the additional keyword argument `_constructor = <name>`. Before a model is symbolically processed, all parameter definitions that contain a `_constructor` keyword are replaced by a reference to a Julia object. It is generated with `_constructor` and all keyword arguments of the parameter. For example, in a

[2] Modia3D.jl, v0.12.2, test/Robot/ServoWithRampAndRevolute.jl

[3] For more details, see e.g., Elmqvist et al. (2021, section 2) and the Modia tutorial https://modiasim.github.io/Modia.jl/stable/tutorial/Tutorial.html.

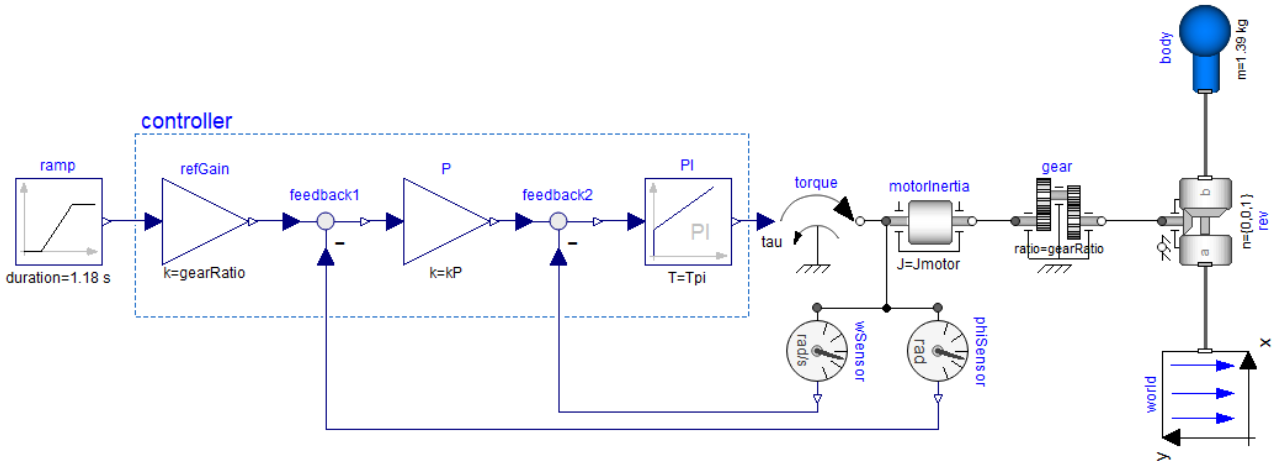


Figure 2. A single revolute joint of a manipulator rotates around the z-axis and is driven by a servo motor via an ideal gear. The revolute angle is controlled by a cascaded P-PI controller, that tracks the reference ramp.

first step `Object3D(feature = Solid())` is replaced by `Modia3D.Object3D(feature = Modia3D.Solid())`. In a second step, this constructor is executed and returns a reference to a Julia object that is associated with key `body`. This can be regarded as a generalization of the concept of External Objects in Modelica.

Listing 3. Definition of multibody components as Modia Models.

```
Flange = Model(phi=Var(potential=true),
               tau=Var(flow=true))
RevoluteWithFlange(; obj1, obj2, axis=3,
    phi=Var(init=0.0), w=Var(init=0.0)) =
  Model(;
    _constructor = Par(value =
      :(Modia3D.Joints.Revolute),
      _jointType = :RevoluteWithFlange),
    obj1    = Par(value = obj1),
    obj2    = Par(value = obj2),
    axis    = Par(value = axis),
    flange  = Flange,
    phi     = phi,
    w       = w,
    equations = :[
      phi = flange.phi
      w   = der(phi)])
```

The multibody component `RevoluteWithFlange` in Listing 3 is defined as a Modia `Model`. It consists of parameters `obj1, obj2, axis`, local variables `phi,w` (that are initialized with zero), an instance `flange` of a rotational flange, and two equations `phi = flange.phi` and `w = der(phi)`. These equations are the acausal part of a revolute joint. The causal part is defined with parameter `_constructor` together with all parameters (defined with keyword `Par`).

During instantiation of a Modia model (before its equations are symbolically processed), all parameter definitions are evaluated. For example, if a parameter `p` is defined with an equation `p = 2*Lx + 3`, assuming that `Lx = 4` is defined as a parameter, then this expression is replaced by `p = 11`.

Listing 4. Constructor generated for `RevoluteWithFlange`.

```
ref = Modia3D.Joints.Revolute(obj1, obj2,
      axis=3)
```

During the parameter evaluation, a special action is taken for parameters with name `_constructor`: A constructor call is assembled from the constructor name and any defined parameters. For example, the `RevoluteWithFlange` definition of Listing 3 results in the constructor call of Listing 4. This constructor is called on the fly resulting in an instance of Julia struct `Revolute`. The call returns a reference `ref` to the created instance. A statement like `rev = RevoluteWithFlange()` in Listing 1 is a key/value pair with the key `rev` and the value is an instance of a `Model` dictionary. This value is replaced by an instance of a parameter dictionary, resulting in `rev = Par(value = ref)`. So, the generated instance of the revolute joint is stored as a parameter. The evaluated parameters are displayed with e.g., `simulate!(logEvaluatedParameters = true)`.

The keys of other instances are referenced in the argument list, e.g., `RevoluteWithFlange(obj1 = :world)`. During parameter evaluation, symbols like `:world` are searched for on the left side of the equal signs. They are then replaced by the corresponding value of this keyword. For example, `:world` is replaced by the Julia reference created by the constructor call `Modia3D.Object3D(feature = Modia3D.Scene())`. Once all parameters are evaluated, all keyword arguments of multibody components contain a reference to the instantiated Julia objects.

A multibody model inside a Modia model is defined with dictionary `Model3D`, see Listing 1. This dictionary is a `Model` dictionary with two additional parameters `_buildFunction` and `_initSegmentFunction`, see

DOI
10.3384/ecp21717

Listing 5. Before symbolic processing begins, a model is recursively inspected. For each subdictionary containing the parameter `_buildFunction`, the function defined with `functionName` is called with the subdictionary as an argument. The items returned by this function call, are added to the subdictionary.

Listing 5. Definition of Model3D model.

```
# predefined acausal component
Model3D = Model(
  # called once before symb. processing
  _buildFunction = Par(functionName
    = :(buildModel3D!)),
  # called before each new sim. segment
  _initSegmentFunction = Par(functionName
    = :(initSegmentModel3D!))
)
```

Furthermore, the entire model hierarchy is flattened. Alias variables are eliminated. The set of all equations is generated, as sketched in Listing 6 for the model of Listing 1.

Listing 6. Flattened model equations with equations injected by `buildModel3D!`.

```
# Equation from RevoluteWithFlange
rev.w = der(rev.phi)

# Equations injected with buildModel3D!
mbs1 = Modia3D.openModel3D!(model, _x,
    time)
mbs2 = Modia3D.setStatesRevolute!(mbs1,
    rev.phi, rev.w)
mbs3 = Modia3D.setAccelerationsRevolute!(
    mbs2, der(rev.w))
genForces = ...

success =
    Modia3D.setHiddenStatesDerivatives!(
    model, mbs3, ...)
```

Function `openModel3D!` creates an instance of the multibody system. It contains, e.g., the generated instance of the revolute joint. The instantiated top level `model` is passed as an argument. So, the function `openModel3D!` has access to the complete model definition. Function `setStatesRevolute!` stores the current values of the angles and angular velocities of all revolute joints of the multibody model. These variables are states in the Modia equations, due to their definition in Listing 3. Function `setAccelerationsRevolute!` stores the angular accelerations of all revolute joints in the multibody model. Further function calls basically construct (5) in residue form. So, $\boldsymbol{f}_{\text{gen}} = \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) - \boldsymbol{\tau}$. The set of flattened equations is processed symbolically, i.e., equations are differentiated, sorted and simplified. The result is stored as a Julia function. It is compiled into binary code that is called by the `simulate!` function.

4 Symbolic Transformations

In Modia, models are symbolically transformed to ODEs (2) in state-space form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{p},t), \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_0), \tag{7}$$

where $\boldsymbol{x}(t)$ is the state vector, $\boldsymbol{p}$ is a hierarchical dictionary of parameters, and t is the time. The algorithms and the symbolic transformations are described in Otter and Elmqvist (2017) and Elmqvist et al. (2021). After the symbolic processing a Julia function called `getDerivatives` is generated and compiled to calculate the derivatives $\dot{\boldsymbol{x}}$.

Physical models often lead to linear equation systems. Modia generates very compact code to solve them numerically during execution of the model.

Assume, a nonlinear equation system

$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{w},\boldsymbol{u}), \tag{8}$$

with unknown local variables $\boldsymbol{w}$ and known variables $\boldsymbol{u}$ are identified by structural analysis. With the tearing algorithm of Otter and Elmqvist (2017) this equation system can be transformed to

$$\boldsymbol{w}_1 = \boldsymbol{g}_1(\boldsymbol{w}_{\text{eq}},\boldsymbol{u}) \tag{9}$$

$$\boldsymbol{0} = \boldsymbol{g}_{\text{eq}}(\boldsymbol{w}_1,\boldsymbol{w}_{\text{eq}},\boldsymbol{u}). \tag{10}$$

where the unknowns and equations are split into an explicitly evaluable part $\boldsymbol{w}_1$ and $\boldsymbol{w}_{\text{eq}}$ is solved implicitly.

If $\boldsymbol{g}$ is linear in the unknowns $\boldsymbol{w}$, it is possible to rearrange (conceptually) equation (10) into a linear equation system

$$0 = \boldsymbol{A}(\boldsymbol{w}_1,\boldsymbol{u})\boldsymbol{w}_{\text{eq}} - \boldsymbol{b}(\boldsymbol{w}_1,\boldsymbol{u}). \tag{11}$$

In (11), $\boldsymbol{A},\boldsymbol{b}$ are functions of the explicitly solved variables $\boldsymbol{w}_1$ and the known variables $\boldsymbol{u}$. The equation has to be solved for variables $\boldsymbol{w}_{\text{eq}}$. In the worst case, $\boldsymbol{A}$ would have n^2 elements ($n = \dim(\boldsymbol{w}_{\text{eq}})$). Therefore, the size of the rearranged code would be $O(n^2)$. So, the code size would increase quadratically with the number of iteration variables n.

Listing 7. Conceptual implementation of linear equation iteration.

```
# Initialize memory m (m.w_eq = 0, ...)
while true
  w_eq = m.w_eq
  w_1 = g_1(w_eq, u)
  m.r = g_eq(w_1, w_eq, u)
  if lEqIteration(m); break; end
end
```

Instead, the concept is to generate the code in Listing 7 and `lEqIteration` in Listing 8. Together they construct and solve the linear equation system (11). Residues $\boldsymbol{r}$ are computed and stored in the memory m. The linear equation system is solved to compute $\boldsymbol{w}_{\text{eq}}$ and $\boldsymbol{w}_1$ from this solution. The code size of this approach is $O(n)$.

Listing 8. Linear equation iteration `lEqIteration`.

```
function lEqIteration(m)
  n = length(m.w_eq)
  if m.mode == QUIT
    return true
  elseif m.mode == COMPUTE_B
    # compute b with w_eq = 0
    # r = A*0 - b => b = -r
    m.b = -copy(m.r)
    m.j = 1
    m.w_eq = e_1
    m.mode = COMPUTE_A
  else # m.mode == COMPUTE_A
    # compute column j of A with w_eq =
        e_j
    # r = A*e_j - b => A[:,j]
    m.A[:,j] = m.r + m.b
    if m.j != n
      m.j += 1      # j+1
      m.w_eq = e_j # j+1-th unit vector
    else
      # solve linear equation system
      # A*w_eq = b
      m.w_eq = m.A \ m.b
      m.mode = QUIT
    end
  end
  m.r = zeros(n)
  return false
end
```

The function `lEqIteration` in Listing 8 is called in a while loop from Listing 7. It iteratively computes vector $\boldsymbol{b}$, matrix $\boldsymbol{A}$, and finally $\boldsymbol{w}_{\text{eq}}$, depending on the actual mode (`COMPUTE_B`, `COMPUTE_A`, `QUIT`). All vectors $\boldsymbol{b}, \boldsymbol{r}, \boldsymbol{w}_{\text{eq}}$, matrix $\boldsymbol{A}$, column counter j, and the actual mode are stored in a memory m, and are updated when needed. To compute vector $\boldsymbol{b}$, the first mode is `COMPUTE_B`. The residues $\boldsymbol{r}$ are computed with $\boldsymbol{w}_{\text{eq}} = \boldsymbol{0}$. This allows to set $\boldsymbol{b} = -\boldsymbol{r}$. To compute matrix $\boldsymbol{A}$, the next mode is `COMPUTE_A`. To iteratively calculate the columns of $\boldsymbol{A}$, the residues are computed with $\boldsymbol{w}_{\text{eq}} = \boldsymbol{e}_j$ that is the j-th unit vector from $j = 1, \ldots, n$. When the n-th column of $\boldsymbol{A}$ is computed, so $\boldsymbol{A}$ is known, the linear equation system is solved for $\boldsymbol{w}_{\text{eq}}$. One final iteration of the while loop is needed to evaluate $\boldsymbol{w}_1$.

Moreover, symbolic processing analyses if $\boldsymbol{A}$ is a function of the parameters $\boldsymbol{p}$, so it does not change after initialization. In this case, the LU-decomposition of $\boldsymbol{A}$ is computed once at initialization and stored in the memory m. During simulation, only a (cheap) backwards solution is applied to compute the solution. If the size of the residual equation is one, a simple division is done, instead of using a linear equation solver. These special cases are not shown in Listing 8 to keep the description simple.

Modia uses the linear equation solver of the Julia package RecursiveFactorization.jl[4] with the left-looking LU-algorithm of (Toledo 1997) for dimensions up to $n = 500$ by default. Benchmarks show a large speed-up compared to the linear standard solver based on OpenBLAS[5] which is otherwise used.

The ODE and DAE solvers of Julia package DifferentialEquations.jl[6] (Rackauckas and Nie 2017) are used for the generated `getDerivatives` function. The `getDerivatives` function is called (automatically) as required by the interface of the selected solver.

One powerful technique for DAE solvers increases the simulation speed enormously. It is applicable when the size n of a linear system of equations exceeds a certain limit ($n \geq 50$), and the unknowns $\boldsymbol{w}_{\text{eq}}$ are a subset of the derivatives of the DAE states. The relevant DAE state derivatives are used as solutions $\boldsymbol{w}_{\text{eq}}$ of the linear system of equations. The residuals $\boldsymbol{r}$ are used for the DAE solver. For each model evaluation, the residuals of the linear equation system are calculated only once instead of solving a linear equation system. At events (including initialization), the linear equation system is constructed and solved, and providing consistent initial conditions for the DAE solver.

To demonstrate the outlined approach, the model in Listing 1 resp. Figure 2 is symbolically processed resulting in the `getDerivatives` function of Listing 9. In the first statements of this function, all used parameters are inquired. The states $\_x$ provided by the solver are assigned to the corresponding model variables. Afterwards, all explicitly solved equations are present. To solve the algebraic loop present in the sorted equations, a new memory m is allocated and its stored data is initialized with zero values before entering the while loop. The while loop computes the residues iteratively, to solve the multibody equations (6), the equations of components `motorInertia`, and `gear` with `lEqIteration` in Listing 8 for the iteration variable $\boldsymbol{w}_{\text{eq}}$.

Listing 9. Generated function for model in Listing 1.

```
# _x states vector from solver
function getDerivatives(_x, model, time)
  < get parameters: startTime, duration,
      kRefGain, gearRatio, ...>

  # states
  rev.phi = _x[1]
  rev.w = _x[2]
  ppi.PI.x = _x[3]

  # explicitly solved equations
  # f1 from eq (6)
  der(rev.phi) = rev.w
  ppi.refGain.u =
    ramp(time, startTime, duration)
  ppi.refGain.y =
    kRefGain * ppi.refGain.u
  motorInertia.phi = gearRatio * rev.phi
```

[4] https://github.com/YingboMa/RecursiveFactorization.jl

[5] https://www.openblas.net/

[6] https://github.com/SciML/DifferentialEquations.jl

DOI
10.3384/ecp21717

```
wSensor.flange.phi = motorInertia.phi
ppi.P.u =
  ppi.refGain.y - wSensor.flange.phi
ppi.P.y = kP * ppi.P.u
der(motorInertia.phi) =
  gearRatio * der(rev.phi)
der(wSensor.flange.phi) =
  der(motorInertia.phi)
wSensor.w = der(wSensor.flange.phi)
ppi.PI.u = ppi.P.y - wSensor.w
der(ppi.PI.x) = ppi.PI.u / Tpi
motorInertia.flangeA.tau =
  kpi * (ppi.PI.x + ppi.PI.u)
motorInertia.w =
  der(motorInertia.phi)

# open 3D model
mbs1 = openModel3D!(model, _x, time)

# set states in revolute joints
mbs2 = setStatesRevolute!(mbs1,
    rev.phi, rev.w)

begin
# new memory m: m.A=zeros(1,1),
# m.b=zeros(1), m.w_eq=zeros(1),
# m.r=zeros(1), m.j=0
# m.mode = COMPUTE_B
m = initlEqIteration(model)

while true
  # explicitly solved equations
  der(rev.w) = m.w_eq[1]
  der(der(rev.phi)) = der(rev.w)
  der(der(motorInertia.phi)) =
    gearRatio * der(der(rev.phi))
  der(motorInertia.w) =
    der(der(motorInertia.phi))
  motorInertia.a =
    der(motorInertia.w)
  gear.flangeA.tau =
    -Jmotor * motorInertia.a +
      motorInertia.flangeA.tau
  gear.flangeB.tau =
    -gearRatio * gear.flangeA.tau

  # set acceleration in joints
  mbs3 = setAccelerationsRevolute!(
    mbs2, der(rev.w))

  # f2 from eq (6): compute generalized
  # forces in joints from position,
  # velocity, acceleration, collisions
  genForces = computeGeneralizedF(mbs3)

  # compute residue vector
  if m.mode != QUIT
    m.r[1] =
      genForces[1] + gear.flangeB.tau
  end
  if lEqIteration(m); break; end
end

# report derivatives to solver
model.der_x[1] = der(rev.phi)
model.der_x[2] = der(rev.w)
model.der_x[3] = der(ppi.PI.x)
return nothing
end
```

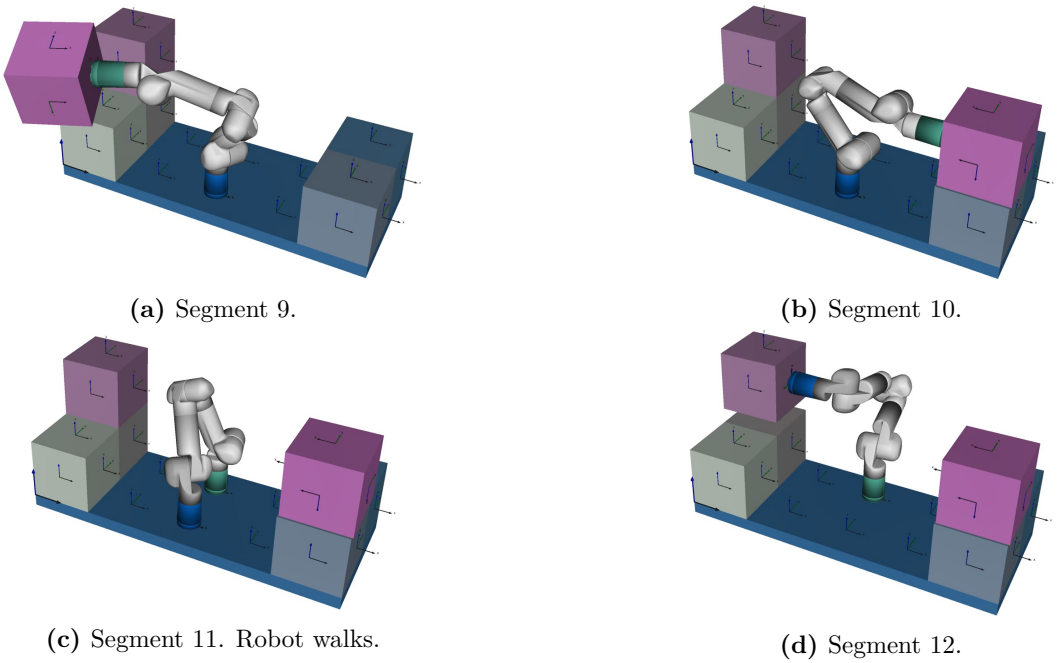
5 Variable Structure Systems: Relocatable Space Robot

Modia3D is designed as a predefined acausal component of Modia. It offers invariant and variant joints. The latter ones can be changed during simulation of variable structure systems. Currently, the category of variant joints consists of a joint type that rigidly fixes two Object3Ds and a joint type that allows a free motion between two Object3Ds. The second joint type can be replaced exclusively by another joint from this category with action commands e.g., `actionAttach`, `actionReleaseAndAttach`, `actionRelease`, `actionDelete`.

In this article, a sophisticated application with a new action command `ActionFlipChain` is discussed. This new action command allows flipping a kinematic chain with segmented simulation. It is demonstrates with a relocatable space robot (Deremetz et al. 2020). The symmetric, 7 DoF robotic manipulator belongs to the MOSAR project (Modular and Re-Configurable Spacecraft, see (Letier et al. 2019)). The robotic manipulator consists of one arm with 7 joints, and two end effectors. One end effector is colorized blue while the other is colorized green, see Figure 3. It enables the detection, manipulation and positioning of spacecraft modules. The robot relocates itself on the interfaces of the spacecraft or on the modules. The visualization data and trajectory for each joint of the robot are taken from Reiner (2022). The drive of each joint has gear dynamics that is modeled by a spring/damper pair with Modia. The 3D mechanics is modeled with Modia3D.

The described behavior above is simulated with the upcoming model. A robot places two modules and walks on a platform, such as a spacecraft. The robot uses the end effectors of its arm in Figure 3. The model shows the robot's ability of gripping the modules with either one of its end effectors and to alternate between attaching of its end effectors to the platform. This allows the robot to walk. In doing so, the kinematic chain of the robot's joints across its span of arm must be reversed. This means that the parent-child relationship between the Object3Ds is flipped. Special treatments of the joints are required to appropriately implement this.

The platform program for the robot and six modules is sketched in Listing 10. Hereby, segments 9–12 correspond to Figure 3a – Figure 3d. At initialization, the robot and the six modules are not rigidly attachted to the platform. In segment 2, the blue end effector is rigidly attached to the platform. In segments 3–8, the six modules are rigidly attached to the



(a) Segment 9. **(b)** Segment 10. **(c)** Segment 11. Robot walks. **(d)** Segment 12.

Figure 3. Walking robot on a platform. One end effector of the arm is fastened to the platform while the other one is able to place one of the two modules or it can walk on the platform.

platform. In segment 9 and 10, the green end effector is moving and replacing a module. In segment 11, the robot is walking. This means that the attachment to the platform alternates between the blue and the green end effector. The blue one is released and the green one is attached. The kinematic chain spanned between the end effectors is reversed. In segment 12, the blue end effector is gripping a module.

The relocatable space robot places two modules and walks on the platform. This scenario lasts 86 s and the simulation is performed in 2.2 s. This is much faster than real-time, since collision handling with point contacts is neglected. Moreover, it is impossible to represent collisions between two parallel surfaces with a collision algorithm that computes point contact like the Minkowski Portal Refinement (MPR) algorithm (Snethen 2008; Neumayr and Otter 2017).

Listing 10. Platform program for relocatable robot.

```
function platformProgram(actions)
  # segment 1 (from initialization)
  # segment 2
  # attach blue end effector to platform
  ActionAttach(actions,
    "blueEnd", "platform.X2Y2")
  EventAfterPeriod(actions, 1e-10)
  # segment 3 - 8
  # attach 6 modules to platform
  ActionAttach(actions,
    "boxX1Y1Z1.Zneg", "platform.X1Y1")
  ...
  EventAfterPeriod(actions, 7.0)

  # segment 9
  # attach box to green end effector
  ActionReleaseAndAttach(actions,
    "boxX1Y1Z2.Xpos", "greenEnd")
  EventAfterPeriod(actions, 17.0)

  # segment 10
  # release box off green end effector,
  # attach box to other box
  ActionReleaseAndAttach(actions,
    "boxX1Y1Z2.Zpos", "boxX5Y1Z1.Zpos")
  EventAfterPeriod(actions, 6.0)

  # segment 11
  # attach green end effector to platform
  # flip kinematic chain between blue and
  # green end effector
  ActionFlipChain(actions, "greenEnd",
    "platform.X2Y2", "blueEnd")
  EventAfterPeriod(actions, 14.0)

  # segment 12
  # attach box to blue end effector
  ActionReleaseAndAttach(actions,
    "boxX1Y2Z2.Xpos", "blueEnd")
  EventAfterPeriod(actions, 23.0)

  # segment 13
  # release box off blue end effector,
  # attach box to other box
  ActionReleaseAndAttach(actions,
    "boxX1Y2Z2.Zpos", "boxX5Y2Z1.Zpos")
  ...
end
```

This application demonstrates that by introducing new features and combining them with existing ones, the new approach for variable structure systems is relatively easy to extend.

DOI
10.3384/ecp21717

6 Conclusion

In this article, equation-based modeling and multibody modeling are combined using the example of a one-armed robot. It shows how to integrate multibody equations, equation-based Modia components and a combination of both. Therefore, Modia3D's multibody components are defined as Modia parameters. In addition, Modia3D components are defined as Modia models with an equation section. All of this is processed to generate code that is solved iteratively. The iterative solution method is discussed in detail. The multibody tree is also set up during initialization, and it is processed to calculate the generalized forces needed to solve the generated code. When dealing with variable structure systems, parts of the multibody tree are rebuilt when a new segmented is initialized.

References

Arnold, Martin (2017). "DAE Aspects of Multibody System Dynamics". In: *Surveys in Differential-Algebraic Equations IV*. Cham: Springer International Publishing, pp. 41–106. DOI: 10.1007/978-3-319-46618-7_2.

Bezanson, Jeff et al. (2017). "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1, pp. 65–98. DOI: 10.1137/141000671.

Brenan, Kathryn Eleda, Stephen L Campbell, and Linda Ruth Petzold (1996). *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. Vol. 14. SIAM. ISBN: 0-89871-353-6.

Deremetz, Mathieu et al. (2020). "MOSAR-WM: A relocatable robotic arm demonstrator for future on-orbit applications". In: *71st International Astronautical Congress, IAC 2020*. IAF. URL: https://elib.dlr.de/139962/.

Elmqvist, Hilding et al. (2021). "Modia - Equation Based Modeling and Domain Specific Algorithms". In: *Proceedings of the 14th International Modelica Conference*. LiU Electronic Press, pp. 73–86. DOI: 10.3384/ecp2118173.

Gear, Charles William (1988). "Differential-Algebraic Equation Index Transformations". In: *SIAM Journal on Scientific and Statistical Computing* 9.1, pp. 39–47. DOI: 10.1137/0909004.

Gear, Charles William, Ben Leimkuhler, and Gopal K Gupta (1985). "Automatic integration of Euler-Lagrange equations with constraints". In: *Journal of Computational and Applied Mathematics* 12, pp. 77–90. DOI: 10.1016/0377-0427(85)90008-1.

Hindmarsh, A.C., R. Serban, and A. Collier (2015). *User Documentation for IDA v2.8.2*. Tech. rep. UCRL-SM-208112. Lawrence Livermore National Laboratory.

Letier, Pierre et al. (2019). "MOSAR: Modular spacecraft assembly and reconfiguration demonstrator". In: *15th Symposium on Advanced Space Technologies in Robotics and Automation*.

Mattsson, Sven Erik, Martin Otter, and Hilding Elmqvist (2015). "Multi-mode DAE systems with varying index". In: *11th International Modelica Conference*, pp. 89–98. DOI: 10.3384/ecp1511889.

Mehlhase, Alexandra (2014). "A Python framework to create and simulate models with variable structure in common simulation environments". In: *Mathematical and Computer Modelling of Dynamical Systems* 20.6, pp. 566–583. DOI: 10.1080/13873954.2013.861854.

Neumayr, Andrea and Martin Otter (2017). "Collision Handling with Variable-step Integrators". In: *8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*. EOOLT'17. ACM, pp. 9–18. DOI: 10.1145/3158191.3158193.

Neumayr, Andrea and Martin Otter (2019). "Algorithms for Component-Based 3D Modeling". In: *13th International Modelica Conference*. LiU Electronic Press. DOI: 10.3384/ecp19157383.

Neumayr, Andrea and Martin Otter (2023a). "Modelling and Simulation of Physical Systems with Dynamically Changing Degrees of Freedom". In: *Electronics* 12.3. DOI: 10.3390/electronics12030500.

Neumayr, Andrea and Martin Otter (2023b). "Variable Structure System Simulation via Predefined Acausal Components". In: *Proceedings of the 15th International Modelica Conference*. LiU Electronic Press. DOI: 10.3384/ecp204.

Otter, Martin and Hilding Elmqvist (2017). "Transformation of Differential Algebraic Array Equations to Index One Form". In: *Proceedings of the 12th International Modelica Conference*. LiU Electronic Press. DOI: 10.3384/ecp17132565.

Rackauckas, Christopher and Qing Nie (2017). "DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia". In: *Journal of Open Research Software* 5.1. DOI: 10.5334/jors.151.

Reiner, Matthias J. (2022). "Simulation of the on-orbit construction of structural variable modular spacecraft by robots". In: *Proceedings of the American Modelica Conference*. LiU Electronic Press. DOI: 10.3384/ECP2118638.

Snethen, Gary (2008). "Xenocollide: Complex collision made simple". In: *Game Programming Gems 7*. Course Technology. Charles River Media, pp. 165–178. ISBN: 978-1-58450-527-3.

Tinnerholm, John, Adrian Pop, and Martin Sjölund (2022). "A Modular, Extensible, and Modelica-Standard-Compliant OpenModelica Compiler Framework in Julia Supporting Structural Variability". In: *Electronics* 11.11, p. 1772. ISSN: 2079-9292. DOI: 10.3390/electronics11111772.

Toledo, Sivan (1997). "Locality of Reference in LU Decomposition with Partial Pivoting". In: *SIAM Journal on Matrix Analysis and Applications* 18.4, pp. 1065–1081. DOI: 10.1137/S0895479896297744.

Zimmer, Dirk (2010). "Equation-based modeling of variable-structure systems". PhD thesis. ETH Zurich. DOI: 10.3929/ethz-a-006053740.

DOI
10.3384/ecp21717

The DLR Cables Library

Tobias Bellmann[1] Andreas Seefried[1] Thomas Bernhofer[1]

[1]Institute of System Dynamics and Control, German Aerospace Center, Germany,
{firstname,lastname}@dlr.de

Abstract

The DLR Cables Library can be used to simulate steel cables with nonlinear stiffness such as steel wire cables with and without non-metal core or coatings. The cable itself is simulated as discrete element component, and can easily be connected to Modelica Multibody components.
Additional components like winches and pulleys are included in the library. With those, complex cable systems with multiple interconnected pulleys become possible in Modelica. Two applications of the library are demonstrated, a construction crane and a cable robot used to compensate gravitation in a space robotic application.
Keywords: Modelica, Cables, Pulleys, Winches, Ropes

1 Introduction

In technical applications, cables oftentimes play important roles e.g. in load handling, transportation, robotics, etc. While the Modelica Standard library contains a feature-rich set of multi-body components (Otter, Elmqvist, and Mattsson 2003), the simulation of cable systems with pulleys and winches right now is only possible with additional libraries (e.g. (Wu et al. 2023), (Börner 2016), (Berger and Heinrich 2011)). In this paper, we would like to present a pure Modelica language based implementation of nonlinear cables, pulleys and winches (see example in figure 1). The components can be coupled with the standard Modelica Multibody frame connectors, enabling the integrated simulation or optimization of complex mechanical systems. The implementation of the cables is focused on simulation speed, allowing complex models, e.g. a construction crane, to be simulated in real-time. However, this library is mainly intended to model the (nonlinear) stretching behaviour of steel or composite cables both of free cables under load and cables wound up on a winch. The bending behaviour of the cables is simulated, but due to the used modeling approach and the highly nonlinear nature of the inner cable mechanics, this can result in sub-par accuracy concerning bending stiffness, damping, eigenmodes, etc.

1.1 Features of the library

The DLR Cables library provides models for cables, winches and pulleys, as well as models to connect cables to multi-body frames. The structure of the library can be seen in Figure 2. A cable consists of `n` discrete cable elements with flexible length.

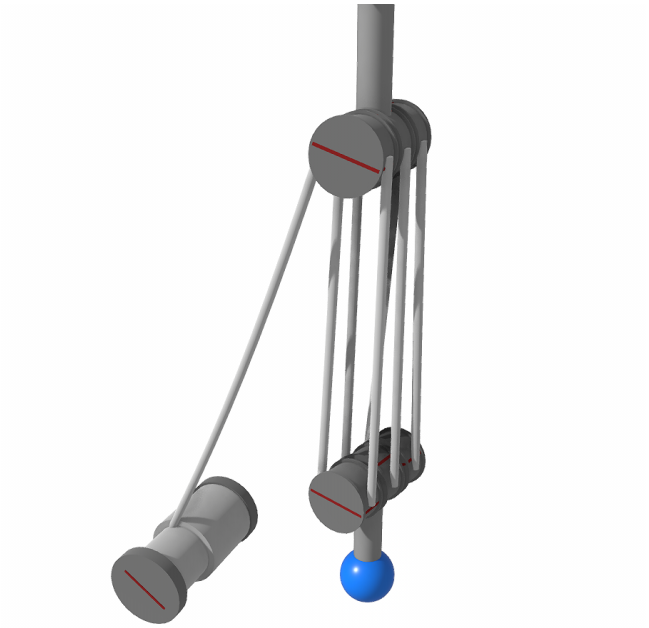


Figure 1. A triple block and tackle system to lift heavy masses, simulated with the DLR Cables Library.

Each cable element has two connectors, a `cableFrame_a` and `cableFrame_b`. In `cableFrame_b`, the normalized direction of the current cable segment is stored and available for the next cable element to calculate the bending of two consecutive cable elements. The cable stiffness is defined by a forces table: by lengthing the cable, a counterforce between `cableFrame_a` and `cableFrame_b` is applied. Thus, a nonlinear stiffness behaviour can be easily implemented. The table has two columns: first the lengthening of a 1m cable, second the force that is acting against that lengthening. In addition to the stiffness force, a damping force acts against the velocity of the cable element along its direction. The damping coefficient is constant. For bending, a constant stiffness and constant damping coefficient can be set. It is possible to tear a cable. The implementation is as follows: the force between `cableFrame_a` and `cableFrame_b` is set to zero. So there is still a complete cable, but one cable element can be streched without any counterforce. For visualization, the cable's transparency element that is 'broken' is set to 100%.

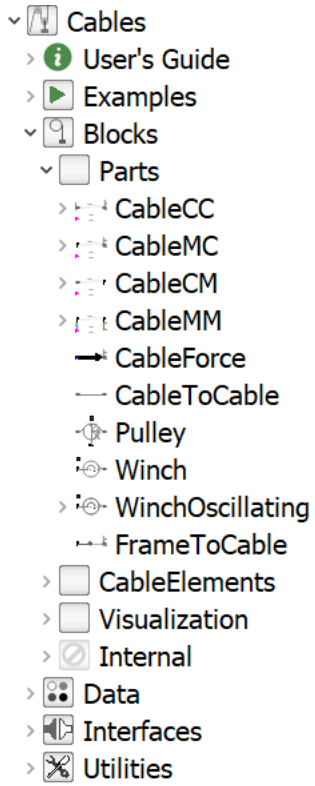


Figure 2. Package overview, showing the main components of the DLR Cables Library.

1.2 Cable models

The Cable models utilises a FEM approach to model a cable with tensile and bending stiffness / damping. Torsion forces are not yet implemented. Three cable models are provided, differing only in their interfaces (Cable-to-Cable, Multi-Body-to-Cable, Cable-to-Multi-Body). The cable stiffness can be defined via a table to reflect non-linear stiffness characteristics of e.g. cables with a non-metal core or coatings. The bending stiffness depends on the modulus of elasticity and the diameter of the cable. A simplified model is used to reduce the computational effort.

1.3 Winch models

The winch model can be used to simulate a winch connected to a multi-body frame. There are two different models available. An ideal winch that provides as much cable as needed and a more realistic oscillating winch where the cable is wound up and unwound in a meandering manner. Both convert the rotation of the winch flange into an elongation or shortening of the attached cable. It is possible to store the cable stresses during the windup of the cable to model the varying elongation of the cable under dynamic loads. When unwinding the winch, this elongation is restored.

1.4 Pulley model

The pulley model allows the connection of two cables to model a pulley system. If one cable is elongated, the other one is shorted by the same length. Effects of asymmetrical stresses and elongations can be considered. For instance, if an elongated cable under load is winded up the cable and the rotational angle of the pulley are large than the transported un-elongated cable. If it is unwinded on the other side of the pulley, a different length of cable is avaliable, because the stress on the cable changes.

2 Modeling approach

In order to achieve a fast simulation speed, several methods are used to simplify the mechanics of the cable system components. The concepts and ideas behind the modeling approach are detailed in this section.

2.1 Basic concepts of the cable model

Every cable consists of n cable segments. Each segment has a variable length and is comprised of two masses, interconnected with a spring damper system to handle tensile strength and lengthening, see Figure 3. The masses are positioned at the ends of the cable segment, avoiding unnecessary transformations from the pose of the connectors to the location of a single central mass. By separating the segments' mass into two separate point masses with half the weight $m/2$, the calculation of rotational inertia of a single body can be omitted, further reducing complexity (but of course by introducing errors such as ignorance of rotations in longitudinal directions of the cable).
In general, the cable segment dynamics is only defined by forces on the two separate masses. Torques (e.g. caused by bending stiffness) are generated by a pair of opposite forces f_{bend} on the two masses of the segment. In order to calculate effects caused by bending of the cable, the angle between the two segments has to be known, therefore the direction of the cable segment is communicated to the adjunct segments. All forces are calculate in the world frame.

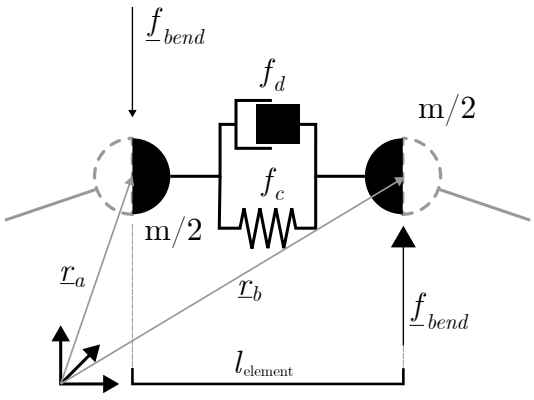


Figure 3. A single cable segment with point masses at the ends and acting forces.

2.2 Basic concept of winches and pulleys

By default, a cable has a fixed length, only subject to change under tensile forces. However, technical systems

DOI
10.3384/ecp21727

oftentimes use winches to unroll or roll up the cable. While it is possible to use forces to bend a FEM based cable around a winch drum, this approach is very costly in terms of calculation time. In order to achieve fast simulation results, suitable for real-time and optimization scenarios, the winch concept in this library differs from this approach: Each cable has a variable length and mass, and it is possible to control length and mass changes from the cable connector. This allows for an external component, in this case winches and pulleys, to change the length of a cable, even if there is no actual unwinding process. Of course, this simplification introduces several modeling errors, such as an unrealistic mass transport behavior and a constant mass and inertia of the winch. The latter is mitigated by also varying the mass of the winch to account for the mass transfer between the winch and the unrolling cable.

2.3 Elongation of a cable on a winch

In many applications where a load is transported with a winch, the load acting on the cable might change during operations. One example for this is an elevator, where the load changes with the exchange of passengers. A noticeable effect is the persistence of the lengthening of a cable under load when rolled up on the winch due to friction forces on the winch. Therefore, when modeling a simplified winch, the elongated state of the cable has to be preserved during the roll up of the cable, and restored during the unroll process. This can be implemented e.g. with a look-up table or other kinds of storage systems.

3 Implementation

The Modelica language allows for a distributed, object-oriented modeling approach, therefore the library is structured in models and sub-models explained in this chapter.

3.1 Interfaces

As explained in the previous section, the concept of the DLR Cables Library reduces the complexity of a full multi-body modeling approach to an point mass based series of cable segments. Therefore, the standard Multibody Frame interfaces cannot be used, and a special `CableFrame` connector without orientations and torques is introduced:

Listing 1. CableFrame connector

```
connector CableFrame
  SI.Position r_0[3];
  Real e_pre[3] "The normalized direction
      of the last cable element in world
      coordinates";
  Real e_next[3] "The normalized direction
      of the next cable element in world
      coordinates";
  flow SI.Force f[3]
    "Cut-force resolved in WORLD frame.";
end CableFrame;
```

It is notable, that the direction of the previous and next cable segment is also communicated with this connector. This enables the calculation of the bending stiffness and damping.

3.2 Cable

The cable model comes in three variations, only differing in the available connectors (see figure 4). The models can either be connected with other cables or Modelica MultiBody frames on either side. Internally, the cable models hold an array of n `CableSegment` models, representing the cable segments. Only the first and last `CableSegment` models are connected to the outside world, the other $n-2$ segments are connected to each other in a for-loop connect statement. Other tasks of the

Figure 4. The three available cable models (from left to right): CableCC, Cable MC, CableCM.

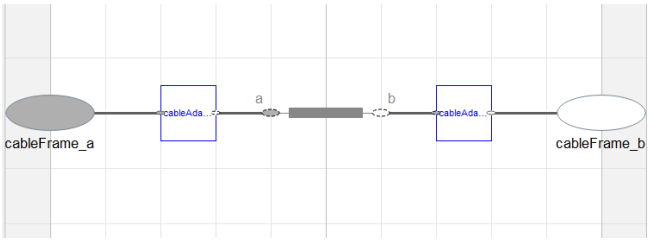


Figure 5. Cable model with two cable connectors (can be connected to other cables, pulleys or winches). In the middle of the diagram, the model array for the n cable segments is visible.

Cable models are the initialization of the cable segment positions, the calculation of the elongation under load and the calculation of the overall cable length. Initially the cable has an overall lenght of l_{cable}. Additionally, the cable has an input Δl, defining the length change of the cable caused by pulleys and winches. The cable is parameterized using the following parameter record:

```
record CableData "base class to hold cable
    information for 1m cable"
  parameter String Name "Name of cable";
  parameter Real forces[:,2]
    "Linear approximation of force due to
        stiffness of the cable of 1m.
        Current force will be calculated
        with [del_l1[m], Force_1 [N];
        del_l2[m], Force_2 [N] ].";
  parameter Real d_axial(unit="N.s.m/m") "
      Damping constant along direction of
      cable element";
```

```
  parameter SI.ModulusOfElasticity E_bend "
      Modulus of elasticity of bending";
  parameter SI.RotationalDampingConstant
      d_bend "Constant damping constant
      against bending.";
  parameter SI.LinearDensity linearDensity
      "Mass of 1m cable";
  parameter SI.Diameter diameter "Diameter
      of the cable";
  parameter SI.Force f_break "Ultimate
      tensile strength of cable";
end CableData;
```

3.3 Cable Segment

The `CableSegment` model contains the physics of the discrete cable segment. As previously explained, the cable consists of point masses, assembled as cable line via tensile and bending forces. Figure 6 shows the modular approach of the model. The two masses are positioned at the `CableFrame` positions $\underline{r}_a$ and $\underline{r}_b$. Between the two masses, the tensile forces and bending forces are applied. The tensile force is a simple spring/damper force along the normal between the two masses $\underline{e}$, with a nonlinear spring force $f_c(l-l_0)$ and a constant damping parameter d:

$$\underline{f}_{tens,c} = f_c(l-l_0)\cdot\underline{e} \tag{1}$$

$$\underline{f}_{tens,d} = d\frac{\partial}{\partial t}(l-l_0)\cdot\underline{e} \tag{2}$$

The nonlinear spring force $f_c(l-l_0)$ is defined via an interpolated table as forces depending on the lengthening of the cable.

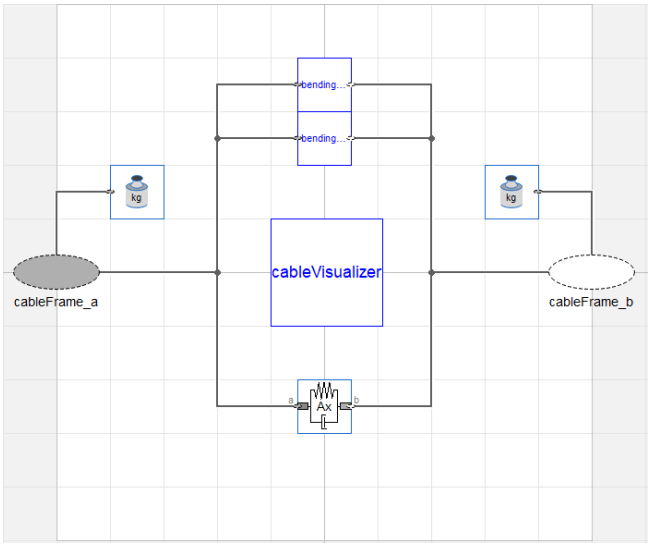


Figure 6. Modelica Implementation of the `CableElement` model

The bending forces can be calculated in two different ways, either with a fast non-physical approximation formula, or via the exact bending radius and modulus of elasticity of the cables material.

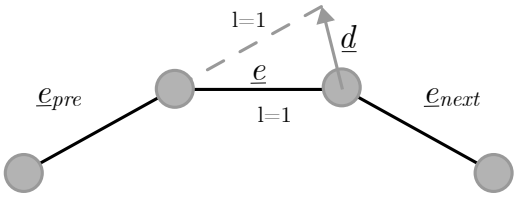


Figure 7. Calculation of the deflection vector $\underline{d}$ between two cable segments. The length of the cable segments is normalized for this calculation

The simplified calculation method to calculate the bending forces uses the deflection vector $\underline{d}$ from the normal axis of the last segment to obtain a measure for the bending of the cable (see Figure 7). The resulting bending forces on the masses at the position $\underline{r}_a$ and $\underline{r}_b$ are then calculate as follows,

$$\underline{f}_{bend,c} = \underline{d}\cdot c_{bend} \tag{3}$$

$$\underline{f}_{bend,d} = \frac{\partial \underline{d}}{\partial t}\cdot d_{bend} \tag{4}$$

$$\underline{f}_{bend} = \underline{f}_{bend,c} + \underline{f}_{bend,d} \tag{5}$$

where c_{bend} and d_{bend} are a non-physical parameter defining the bending stiffness and damping.

The physically correct but slower calculation of the bending forces uses the calculation of the modulus of elasticity of the material: The axes $\underline{e}_{tau,pre}$ and $\underline{e}_{tau,next}$ define the directions of the bending rotation axes between the current segment and the previous, respectively the next section:

$$\underline{e}_{tau,pre} = ||\underline{e}_{pre}\times\underline{e}|| \tag{6}$$

$$\underline{e}_{tau,next} = ||\underline{e}\times\underline{e}_{next}|| \tag{7}$$

Utilizing the angles α_{pre} and α_{next} defining the angle between the current segment and the previous or next segment, the bending radii κ_{pre} and κ_{next} can be calculated as

$$\kappa_{pre} = \frac{\tan(\frac{\alpha_{pre}}{2})}{l_{element}} \tag{8}$$

$$\kappa_{next} = \frac{\tan(\frac{\alpha_{next}}{2})}{l_{element}} \tag{9}$$

The bending torque is replaced with a pair of forces, where the directions are

$$\underline{n}_{pre} = ||(\underline{e}_{tau,pre}\times\underline{e})|| \tag{10}$$

$$\underline{n}_{next} = ||(\underline{e}_{tau,next}\times\underline{e})|| \tag{11}$$

The bending stiffness force on `CableFrame_a` and `CableFrame_b` then results in

$$\underline{f}_{bend,c,a} = -\underline{n}_{pre}\cdot\frac{\kappa_{pre}\cdot E_{bend}\cdot I_y}{l_{element}} \tag{12}$$

$$\underline{f}_{bend,c,b} = \underline{n}_{next}\cdot\frac{\kappa_{next}\cdot E_{bend}\cdot I_y}{l_{element}} \tag{13}$$

DOI
10.3384/ecp21727

with E_{bend} being the modulus of elasticity of the material and $I_y(=I_x)$ being the inertia of the segment. The bending damping is implemented as

$$\underline{f}_{bend,d,a} = -\underline{n}_{pre} \cdot d_{bend} \cdot \frac{\partial(\kappa_{pre})}{\partial t} / l_{element} \tag{14}$$

$$\underline{f}_{bend,d,b} = \underline{n}_{next} \cdot d_{bend} \cdot \frac{\partial(\kappa_{next})}{\partial t} / l_{element} \tag{15}$$

The summarized bending forces are then obtained with

$$f_{bend,a} = \underline{f}_{bend,c,a} + \underline{f}_{bend,d,a} \tag{16}$$

$$f_{bend,b} = \underline{f}_{bend,c,b} + \underline{f}_{bend,d,b} \tag{17}$$

3.4 Winch

As mentioned in the last section, the winch comprises of a rotating, cylindrical mass with a cable attached to it at a defined point in space. Depending on the type of Winch,

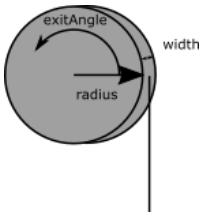


Figure 8. Winch parameters

the position of the cable exit point is either fixed relatively to the non-rotating winch coordinate system (`Winch`) or is moved left and right and with varying radius while the winch rotates (`OscillatingWinch`). In the case of a fixed cable exit point the position defined via the parameter `exitAngle` (α_{exit}, see figure 8):

$$\underline{r}_{lever} = \{r \cdot \cos(\alpha_{exit}), 0, r \cdot \sin(\alpha_{exit})\} \tag{18}$$

For the `OscillatingWinch`, the exit point moves left and right and the radius changes with the rotation angle ϕ of the winch so it is defined as

$$\underline{r}_{lever} = \{r(\phi) \cdot \cos(\alpha_{exit}), y(\phi), r(\phi) \cdot sin(\alpha_{exit})\} \tag{19}$$

where r_{phi} describes a discrete change of the radius ($\pm$ diameter of the cable) when the cable drum is covered with one layer of the cable or one layer has been fully unrolled. The function $y(\phi)$ is described as a saw tooth function to create the meandering movement of the cable with the winch rotation.

The length (and mass) change of the attached cable is defined as function of the rotational velocity $\frac{\partial}{\partial t}\phi$ of the winch:

$$\frac{\partial}{\partial t}\Delta l_{cable} = \frac{r \cdot \frac{\partial}{\partial t}\phi}{\eta_{sf}} \tag{20}$$

Hereby η_{sf} is the so called stretch factor, defining the relative elongation of the cable, potentially resulting from rolling up a cable under tension:

$$\eta_{sf} = l_{cable,current} / l_{cable,nominal} \tag{21}$$

As the stretch factor can change with varying load on the cable, it has to be stored and restored while rolling up or unrolling the cable on the winch. For this purpose, the `spatialDistribution` operator of the Modelica language is being used.
The forces and torques resulting from the cable pulling on the winch are applied also to the MultiBody frame of the winch. For this purpuse, the resulting torques are divided into torques acting on the winch' rotational axis and torques acting in other directions.

3.5 Pulley

A pulley is implemented as two winches, connected on their rotational axes, see Figure 9: The pulley has two po-

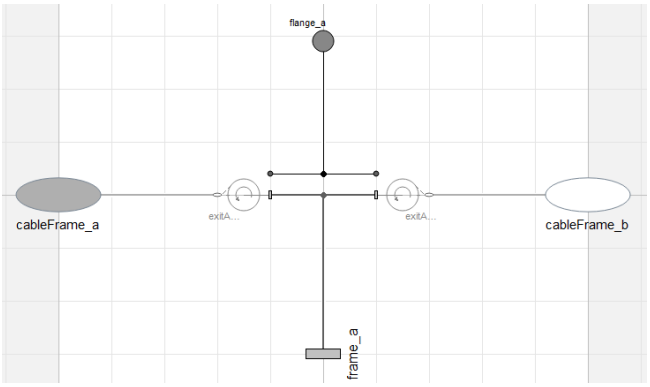


Figure 9. Pulley model

sitions for the cable exit point, defined by the parameters `exitAngleA` and `exitAngleB`.

3.6 Initialization

At the beginning of every simulation run, the spatial position of the cable segments has to be initialized. As there is a multitude of possible initialization scenarios, the initialization routine is implemented as a replaceable function. The base class for this initialization function has the following interfaces:

```
partial function baseCableInitialization
  input Real r_a[3]={0,0,0} "Position of
      cable start";
  input Real r_b[3]={1,0,0} "Position of
      cable end";
  input Real l=10 "Length of cable";
  input Integer n=10 "Number of cable
      elements";
  input Real g[3]={0,0,-9.8} "Gravity";
  output Real r_start[n,3] "start positions
       of cable elements";
  output Real delta_l_init "Length
      difference from initialization";
end baseCableInitialization;
```

The inputs are provided by the cable model as these are known parameters and states. The user can now use this base class to create own initialization routines and calculate the resulting start points of the cable segments `r_start` and a length difference to the nominal cable length `delta_l_init` (e.g. to account for a cable initialized under tension).

There are three initialization routines provided with the library:

1. Catenary initialization:
   Depending on the positions of the ends of the cable and the user-defined length l_{cable}, the catenary line is calculated and all points of the cable segments are initialized on that line:

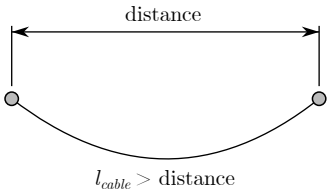


Figure 10. Initialization of a cable on a catenery curve.

2. Linear initialization:
   Depending on the positions of the ends of the cable, the cable segments are distributed equally on the direct connection between those two points, not changing the user-defined cable length l_{cable}:

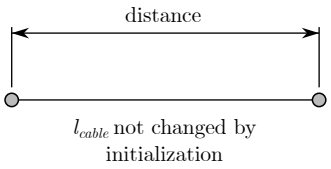


Figure 11. Initialization of a cable on a direct line. As the parameter l_{cable} is not changed by this initialization routine, the cable can be either stretched or compressed.

3. Linear initialization with adaptive length:
   Depending on the positions of the ends of the cable, the cable segments are distributed equally on the direct connection between those two points. The user defined cable length l_{cable} is modified, so that the cable exactly fits between the two mount points. It is possible to set a stretch factor to shorten the cable and to initialize it pre-stretched.

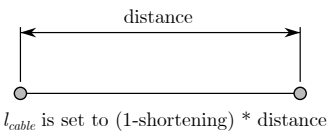


Figure 12. Initialization of a cable on a direct line. The length of the cable is set to distance, but with a defined shortening factor to account for initial tension on the cable.

3.7 Visualization

In order to support a wide range of visualization tools, the library uses replaceable models for the visualizer blocks. Right now, the following visualization tools are supported:

- Modelica tool visualization (Otter, Elmqvist, and Mattsson 2003)
- DLR Visualization library (Hellerer, Bellmann, and Schlegel 2014)
- DLR Visualization2 library (Kümper, Hellerer, and Bellmann 2021)

As cables and winches are easily visualized with cylinders, only few visualizer blocks are necessary.

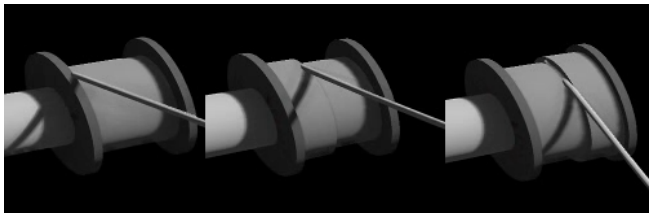


Figure 13. Visualization of the `OscillatingWinch` showing the process of rolling up a cable using the DLR Visualization2 Library.

4 Application examples

In this section, some application examples, combining the DLR Cables Library with Modelica MultiBody commponents are presented. The first example is a construction crane, where the Cables library is used to simulate the mechanics of the load bearing cable and the cable moving the cable trolley. In the second example, the Cables library is used to simulate a gravity compensation mechanism to enable testing of a space robot system on earth.

4.1 Construction Crane

As part of the authors technology transfer activities, a detailed multi-domain model of a construction crane has been developed (see Figure 14).

It is used mainly for teaching and demonstration purposes, and features the mechanical structure, drive trains and cables. The three drive train each consist of the electric drive (asynchronous motor), a vector motor controller, thermal effects, gearbox and brakes (See Figure 15).
For educational purposes, the model also features a calculation of its center of mass, demonstrating the stability of the crane with different loads, varying jib working radius and counterweights.
The model also contains a simplified air drag model for the larger structures and load, further showing the effects of wind load on the crane stability.

The construction crane uses two cable systems and a rotational joint to move its load in three dimensions. The structure of the crane is described in Figure 16. The main

DOI
10.3384/ecp21727

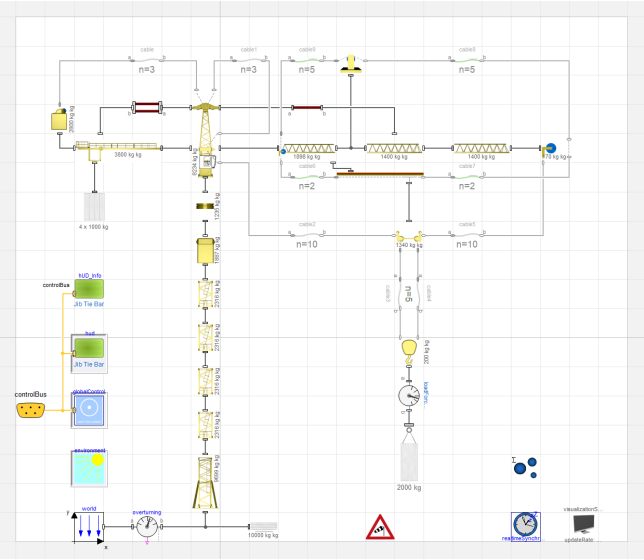


Figure 14. Modelica model of a construction crane. The model can be controlled interactively.

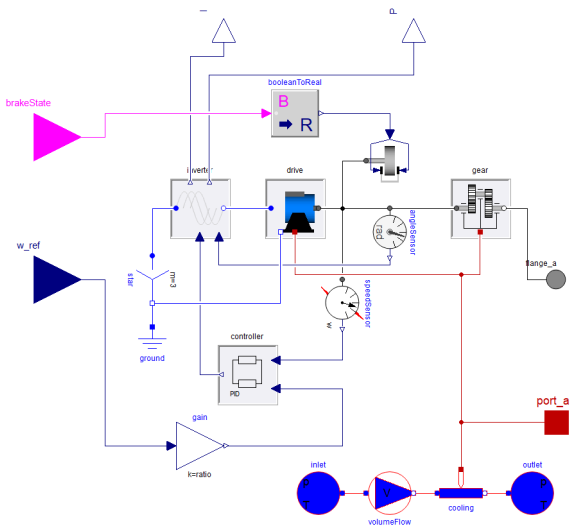


Figure 15. Modelica model of the main winch drive train with asynchronous machine (`IM_SquirrelCage` from the MSL), gearbox, PID controller, inverter, brake and a cooling system.

winch (red) is used to lift the load up and down. The cable attached to it (orange) runs from the main winch in the back of the crane through two pulleys in the tower to the trolley. There, it goes down via another pulley to the load pulley and back up again to another pulley attached to the trolley. It is fixated at the end of the cranes jib. The trolley itself is pulled back and forth with the second cable system (green), featuring a closed loop of three pulleys, each one at the front and end of the crane jib and an actuated one in between. The jib and crane top is rotated with a rotational joint between the crane base and the crane top. In total, the model uses nine cable blocks, one winch and six pulley blocks. With 70 cable segments in total, the model is running in real-time on a Intel Xeon W-2133 with 3.6GHz (Integrator: Sdirk34h). It is visualized in 3D and can be controlled by the user with on-screen controls from the DLR Visualization 2 library (see Figure 17). Figure 18 shows some simulation data of a simple crane maneuver. After initialization, the crane trolley is driven $5m$ forward at $t = 10s$. The top graph shows the position of the trolley, the middle graph the length of the load cables between trolley and load, and the bottom graph the tensile force on the trolley. Several cable related effects are visible, e.g. the swaying of the load, leading to length changes of the two load cables (middle plot), or the forces of the trolley winch system, resulting in the movement of the trolley sled (bottom plot). The crane can be controlled either via an interface to an external PLC, utilizing a ProfiNet interface, or by using on-screen overlays as a graphical joystick and sliders. Additionally, predefined reference trajectories can be used for repeatable simulations.

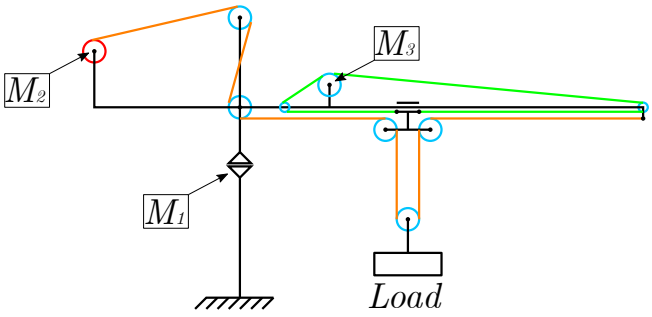


Figure 16. The structure of the construction crane, with the actuated rotational joint (M1), the main winch (M2, red) for the load cable (orange) and the trolley actuator (M3) driving the trolley cable (green). Pulleys are drawn in blue.

Figure 17. Interactive Simulation of the construction crane with on-screen controls (bottom left), real time forces of the jib tie load (middle left), and status information on the drives (upper left).

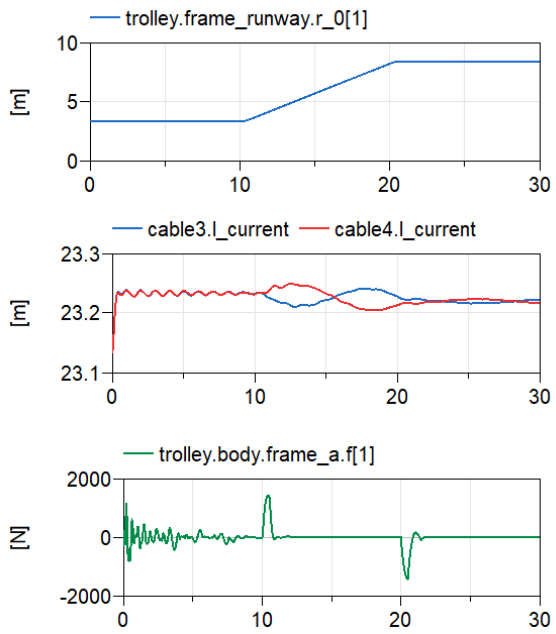


Figure 18. Trolley position (top), length of load cables between trolley to load (middle), combined forces on the trolley (bottom).

4.2 Gravity Compensation for Space Robot Arm

Space robots play an important role in on-orbit servicing operations, including the repair, maintenance and refuelling of satellites. These robotic systems are designed to operate in a zero-gravity environment, thus a gravity compensation mechanism is required to test them on Earth. Therefore, the Motion Suspension System (MSS) (Elhardt et al. 2023), which is visualized in figure 19, has been developed. It allows a force to be applied to the robot in a desired direction. The MSS consists of four cable winches which are arranged in a rectangle on the floor around the robot arm. Pulleys on the ceiling guide the cables to the robot, where they are connected to each other. The suspension cable connects the four cables from the winches to the third segment of the robot arm. The simulation model consists of four pulley blocks, four winch blocks and nine cable blocks with a total of 57 cable segments. The cables are connected to a seven degree of freedom robot using a passive two degree of freedom coupling mechanism. The inputs to the model are the winch torques and the robot torques.

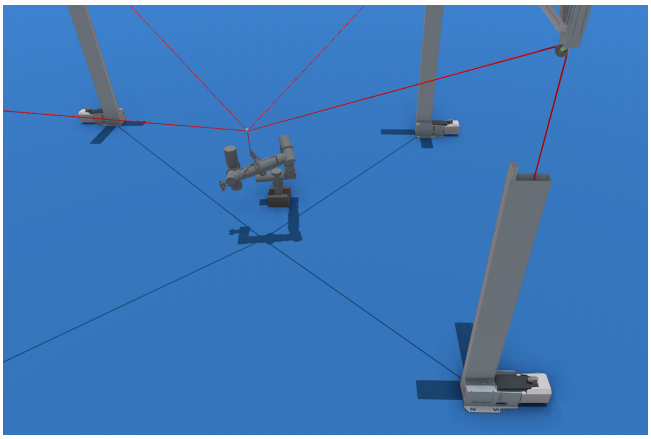


Figure 19. Visualization of the simulated Motion Suspension System.

Figure 20 shows simulation data where the robot is held in a fixed position and the winch torques are increased from an initial 4Nm to 8Nm at t=3s, in order to compensate the gravitational forces acting on the robot. The upper plot shows the forces on the four winch cables and the force on the suspension cable, which connects the four cables to the robot. The lower plot shows the robot joint torques two and three, as these are the joints most supported by the MSS. At the beginning of the simulation, the forces and torques oscillate strongly because the node position of the four winch cables is not perfectly initialised. After about 1.5s the system stabilises and a suspension force of about 130N is applied to the robot. In this case, the drive torques in joints two and three must be about -120Nm and -37Nm respectively for the robot to hold its position. By increasing the winch torque at t=3s, the suspension force is increased to about 260N. In this case the joint torques are reduced to almost zero.

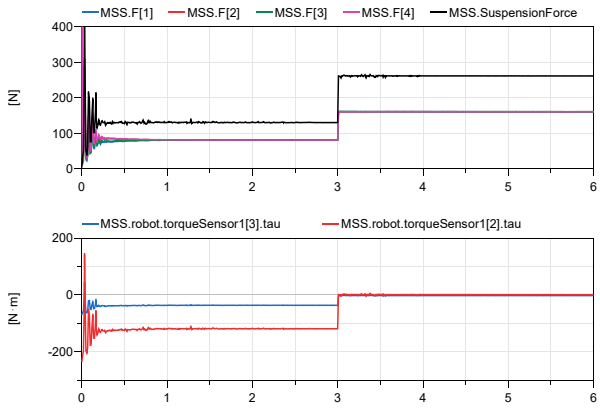


Figure 20. Plots of winch cable forces and suspension cable force (top) and robot joint torques two and three (bottom).

5 Conclusion and Outlook

In this paper, a new pure Modelica based library for fast simulation of cable systems has been presented and its methodical background and implementation has been discussed. An important factor in the reliability of simulation is the thorough validation of the models, in (Seefried and Bellmann n.d.), the validation methods for the cable models, and more real-world examples are discussed. In the future, the library might be extended with more components as more detailed and realistic non-linear bending force models, arbitrary forces acting anywhere on the cable (e.g. to simulate a pulley running along a fixed cable) and collision detection of cables with the environment based on the DLR ContactDynamics Library (Buse, Pignede, and Barthelmes 2023).

DOI
10.3384/ecp21727

References

Berger, Maik and Stefan Heinrich (2011). "Analysis of Belt Drives with circular and variable Pulleys in SimulationX". In: DOI: 10.3403/30219039u.

Börner, Denise (2016-11). URL: https://www.esi-group.com/sites/default/files/news-release/527/pr_esi_group_simulationx_3.8_en.pdf.

Buse, Fabian, Antoine Francois Xavier Pignede, and Stefan Barthelmes (2023-Oktober). "A Modelica Library to Add Contact Dynamics and Terramechanics to Multi-Body Mechanics". In: *Proceedings of the 15th International Modelica Conference*. Vol. 204. Linköping Electronic Conference Proceedings. Modelica Association and Linköping Electronic Conference Proceedings. URL: https://elib.dlr.de/200107/.

Elhardt, Ferdinand et al. (2023-11). "The Motion Suspension System – MSS: A Cable-Driven System for On-Ground Tests of Space Robots". In: *16th International Federation of Theory of Machines and Mechanisms World Congress, IFToMM WC 2023*. Vol. 148. Mechanisms and Machine Science 2. Springer, Cham, pp. 379–388. URL: https://elib.dlr.de/199117/.

Hellerer, Matthias, Tobias Bellmann, and Florian Schlegel (2014-03). "The DLR Visualization Library - Recent development and applications". In: Modellica. ISSN: 1650-3686. DOI: 10.3384/ecp14096899.

Kümper, Sebastian, Matthias Hellerer, and Tobias Bellmann (2021-09). "DLR Visualization 2 Library - Real-Time Graphical Environments for Virtual Commissioning". In: Modelica 2021. ISSN: 1650-3686. DOI: 10.3384/ecp21181197.

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003-11). "The New Modelica MultiBody Library". In: *3rd International Modelica Conference*. Ed. by Peter Fritzson. LIDO-Berichtsjahr=2003, pp. 311–330. URL: https://elib.dlr.de/11987/.

Seefried, Andreas and Tobias Bellmann (n.d.). "Validating the DLR Cables Library with Experiments and Parameter Optimization". In: *Asian Modelica Conference 2024*.

Wu, Jianchen et al. (2023-12). "Object-Oriented Modelling of Flexible Cables based on Absolute Nodal Coordinate Formulation". In: Modelica 2023. ISSN: 1650-3686. DOI: 10.3384/ecp20453.

DOI
10.3384/ecp21727

Validating the DLR Cables Library with Experiments and Parameter Optimization

Andreas Seefried[1] Tobias Bellmann[1]

[1]Institute of System Dynamics and Control, German Aerospace Center (DLR), Germany,
{Andreas.Seefried,Tobias.Bellmann}@dlr.de

Abstract

The advantages of modelling and simulation are widely known: Optimizing systems before production, generating alternatives in a few clicks, reducing costs, monitoring, digital twin, etc. The quality of the simulation depends heavily on the quality of the modeling, making it an essential task. The DLR Cables library, which we presented in another work, allows the simulation of steel cables, focusing on use cases where their dynamic behavior is of interest, such as cranes and elevators, but also special motion systems using cables and amusement rides. There, the numerical approach based on finite elements is explained in detail and it is also shown that some simplifications are accepted in order to improve the computational effort. This paper presents the crucial tasks of validation and parameterization of the model, specifically focusing on the material properties of bending stiffness and bending damping. To achieve this, a series of experiments were carried out on four different cables. Optical systems are used to record the cables and to compare them with the simulation. For some of the experiments, we were able to show a good match between reality and simulation, but it also became clear that a linear approach may not be sufficient depending on the application.

Keywords: Modelica, Steel Cables, Validation

1 Introduction

In this paper, we present our results of a validation campaign for a Modelica library that can be used to simulate steel cables, the DLR Cables library (Bellmann, Seefried, and Bernhofer n.d.). The focus of that library was primarily on computational speed, as our approach to describing a cable through multiple discrete bodies coupled together becomes computationally intensive with the standard Modelica Multibody Library (Otter, Elmqvist, and Mattsson 2003). Therefore, in our approach, cable elongation, cable torsion, and cable bending are considered separately with simplified equations of motion. This leads to an error that needs to be estimated.

For the validation campaign presented here, existing measurement systems are utilized. The goal is to validate the modelling of a cable in order to be able to simulate more complex systems, where the dynamic behavior of a cable is crucial, like Jomartov et al. (2023), Katliar et al. (2017), Elhardt et al. (2023), and Yan et al. (n.d.).

In the second section, the cable model is outlined briefly, and the potential parameters are listed, including those already available and those to be determined through experiments. Section three presents two different measurement setups and the methodology for evaluation, as well as the results from the measurements. The paper concludes with a discussion of the results and future endeavors aimed at further improving cable modelling.

2 Cable Model and Available Data for Parameterization

Modelling a steel cable is a complex task. In the paper "The DLR Cables Library" (Bellmann, Seefried, and Bernhofer n.d.), we present an approach where a cable is divided in a defined number of elements that are connected in a row. To model the behavior of the cable, the calculation of elongation, bending and torsion are separated. This reduces the computational time compared to an approach using the standard Modelica Multibody components but physical effects are also lost, the influence of which will also be investigated in this validation study. Additionally, depending on the task, different levels of complexity can be used for the three main components.

In the literature, there are various approaches for identifying cable behaviour and to obtain information for parameterization. For cable elongation, data from the manufacturer is usually available. Here, a defined length of cable is clamped in a test rig and subjected to various tensile loads. The resulting elongation is recorded and stored in a table. Such a potential nonlinear characteristic curve can be represented in the presented model. The axial damping along the cable is currently implemented as a linear case. For cable bending, various setups can be found in the literature, which either measure the resistance force of the cable depending on lateral deflection (Z. Chen et al. 2015) or analytical or numerical considerations for bending stiffness are carried out (Zhu, Ren, and Xiao 2011; Papailiou 1995). In the library, a more complex model is available that takes the curvature into account and thus becomes nonlinear for larger bending. The parameters for bending stiffness and bending damping are single values (no lookup table). For torsion we use a simple linear stiffness and damping model. For bending and torsion, data

acquisition is not easy. Due to the missing parameters and the uncertainties resulting from the modeling, validation is unavoidable. The study of cable research shows more effects that are not implemented until now, for example damping due to interwire friction (Spak, Agnes, and Inman 2013; Y. Chen, Meng, and Gong 2017) or variable bending stiffness due to axial load (Papailiou 1995). These are not implemented in the current model and can be considered for further refinement if needed.

3 Experimental Setup and Parameter Identification

At DLR, highly versatile and precise laser measurement systems are available, therefore a test rig setup has been devised to utilize the existing equipment to record the behaviour of different steel cables as accurately as possible and compare it with our model (Häusler 2019). The following setup was used for four different steel cables from Pfeifer, see Table 1. The manufacturer provided detailed data for the stiffness of the cable. The focus of the following experiments was on the bending stiffness and bending damping.

Name	D [mm]	Weight q_0 [kg m^{-1}]
10 P 524	10	43
16 P 524	16	110
20 P 524	20	172
16 PN 152/9	16	104

Table 1. Cables used for the experiment. The P 524 cable has a plastic coated steel core while the PN 152/9 has only a steel core. Both have an ordinary lay.

3.1 Experiment 1: Horizontal Clamped Cable

To identify static parameters, a short piece of cable is clamped horizontally in a holding device. The aim is to check the extent to which the bending beam theory applies to such a short piece and to determine the Modulus of Elasticity on the basis of the measured values, see Fig. 1.

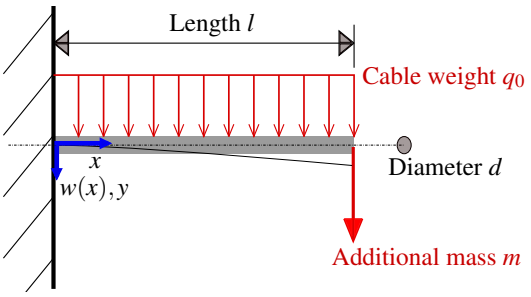


Figure 1. Using beam theory for a short piece of cable to get the Modulus of Elasticity.

Different masses m are applied to the free end of the piece of cable. A line laser scanner (Keyence LJ-V7300) measures the deformation of the cable section. Hysteresis behavior was observed during the tests. Depending on whether the weight tends to be deflected upwards or downwards, different positions result after releasing the weight, see Fig. 2. This indicates a kind of static friction in the cable. Since this is not currently represented by the cable model, the mean value is used for further consideration.

In Fig. 3, the mean deflections over the distance from fixed clamping are shown with different additional masses m for the four cables.

From these measurements, the Modulus of Elasticity can be calculated (Byskov 2013). At this moment, we've used a simple approach with small deformations.

For the constant load from the cable, the deflection of the beam can be described by

$$w_{q_0}(x) = \frac{q_0 l^4}{24EI}\left(6\left(\frac{x}{l}\right)^2 - 4\left(\frac{x}{l}\right)^3 + \left(\frac{x}{l}\right)^4\right), \quad (1)$$

for the additional mass m it is

$$w_m(x) = \frac{mgl^3}{6EI}\left(3\left(\frac{x}{l}\right)^2 - 4\left(\frac{x}{l}\right)^3\right). \quad (2)$$

with g the gravitational constant and

$$I = \frac{r^4\pi}{4} \quad (3)$$

the second moment of area for a cylindrical cross section with a radius r of the cable. Combining Eq. (1) and Eq. (2) with $w(x) = w_{q_0}(x) + q_m(x)$ and solving for E leads to

$$E(x) = \frac{q_0 l^4}{24w(x)I}\left(6\left(\frac{x}{l}\right)^2 - 4\left(\frac{x}{l}\right)^3 + \left(\frac{x}{l}\right)^4\right) + \frac{mgl^3}{6w(x)I}\left(3\left(\frac{x}{l}\right)^2 - 4\left(\frac{x}{l}\right)^3\right) \quad (4)$$

Fig. 4 shows the resulting Moduli of Elasticity for the cables. The first 5 cm are not meaningful but then the curves align and show an approximately constant behavior, which was used for parameterization in the library, see Table 2. It is planned to carry out further measurements with more complex theories and other bending tests like presented in Cao and Wu (2018) or Z. Chen et al. (2015).

Name	D [mm]	E-module [N m^{-2}]
10 P 524	10	0.65×10^8
16 P 524	16	1.9×10^8
20 P 524	20	2.1×10^8
16 PN 152/9	16	1.75×10^8

Table 2. Experimentally obtained Moduli of Elasticity

3.2 Experiment 2: Free Swinging Cable

To measure the dynamic behavior of the cables, we use a very precise laser tracking system (Leica AT 960). A 8m cable is mounted on the ceiling in our lab. A mass of 10 kg is added at the free end of the cable. For the experiment,

DOI
10.3384/ecp21737

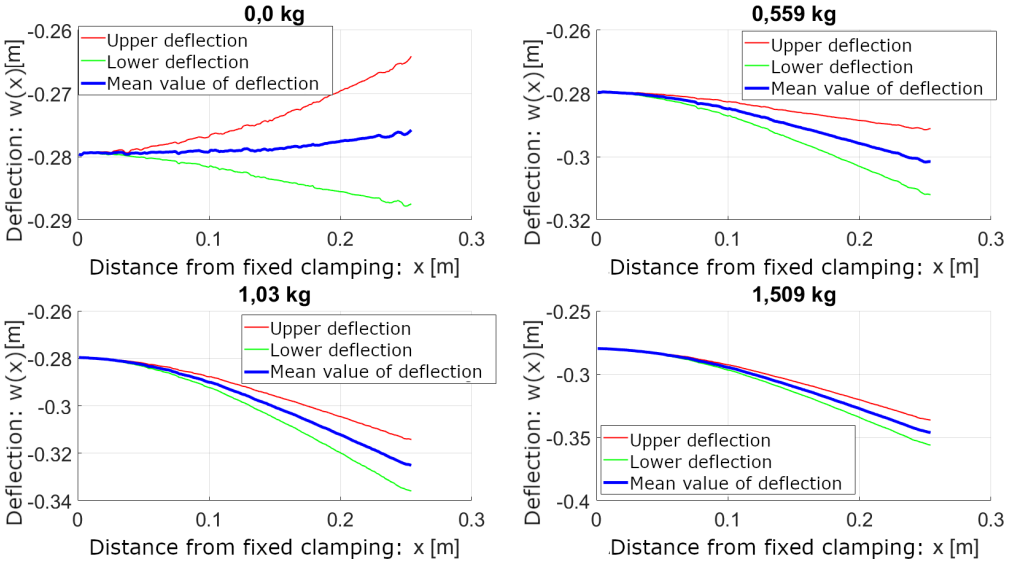


Figure 2. Experiment 1: Upper and lower deflection of cable with different weights at the free end of the cable. Because of static friction, the cable rests at different positions depending on the direction. Here, the deflections of PN 152 are shown.

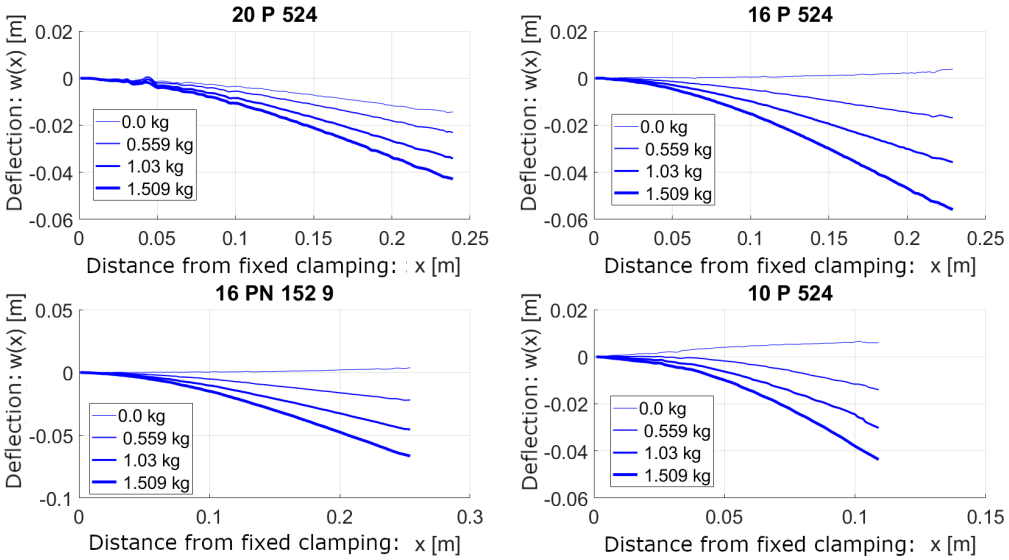


Figure 3. Experiment 1: Mean deflections of the cable section with different weights at the free end.

we put the free end of the cable to five different positions and release the end. This way, the cable swings in different but repeatable ways. A marker for the laser tracker is attached at the cable on nine different positions. The laser tracking system can only track one target at the same time, so there are 45 runs to do for each cable. Figure 5a shows the cable with the different starting positions and the positions of the markers for the laser tracking system. Figures 5b and 5c show two different starting positions for the cable and the evolution of the swinging behavior. With such different starting positions, excitation at various frequencies within the cable can be achieved. These are harmonics and the decay of these movements helps to identify the cable bending and damping.

On the modelling side, we also use a 8m long cable with the same mass at the end. The number of elements is set to 50, so that the transitions between the discrete cable elements match the positions of the markers. The error

$$e_1(t) = (r_{E,1}(t) - r_{S,1}(t))^2 \tag{5}$$

describes the time dependent distance of the position of the cable in the experiment, here at marker '1', $r_{E,1}(t)$, and the corresponding position of the cable in the simulation, $r_{S,1}(t)$. The integral of $e_1(t)$ describes the cost function

$$J_1 = \int_0^{t_e} e_1(t)\,\mathrm{d}t \tag{6}$$

and is taken to find the optimal parameter for the damping bending by minimizing J_1. For a more complex optimization, where all positions of the markers are optimized at

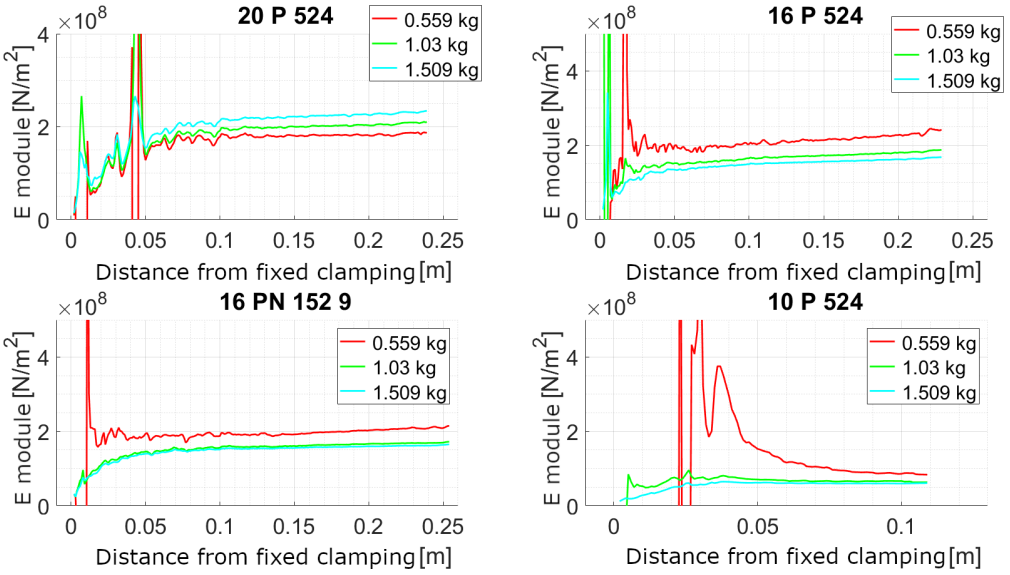
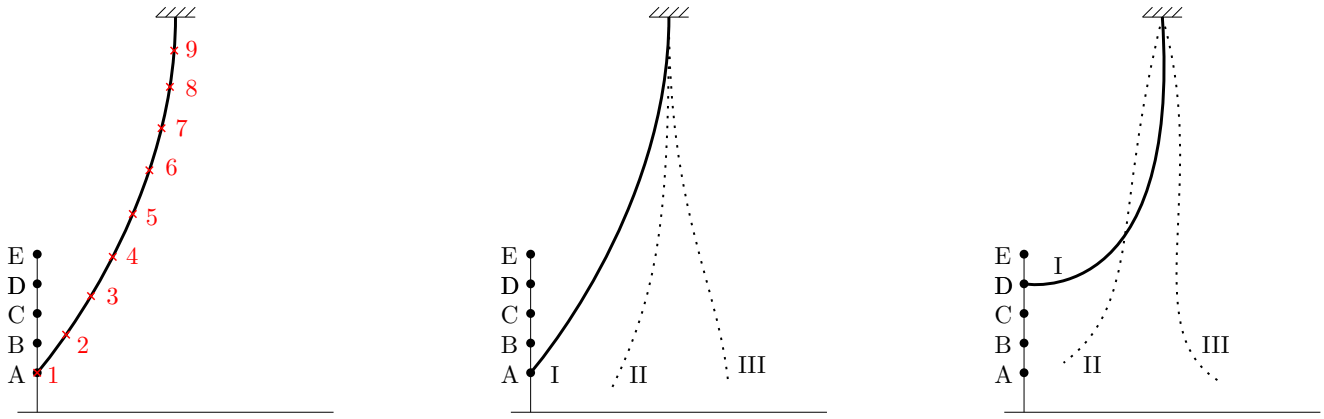


Figure 4. Moduli of Elasticity of the different cables depending on their distance to the fixed clamping



(a) Numbering of starting positions (A to E) and positions of the marker of the laser tracker (1 to 9).

(b) Example of swinging cable with starting position A (I). After a few milliseconds, the cable has the configuration II and then III.

(c) Example of swinging cable with starting position D (I). After a few milliseconds, the cable has the configuration II and then III with strong harmonics.

Figure 5. Experimental setup of the 8m long cable mounted at the ceiling with different starting positions.

the same time, the cost function expands to

$$J = \int_0^{t_e} e_1(t) + e_2(t) + \ldots + e_9(t)\,\mathrm{d}t. \tag{7}$$

In Fig. 6, the model is shown of that optimization to measure the difference between the recorded first marker position and the corresponding point at the cable, here at the additional mass. The optimization is implemented with the *DLR Optimization library* (Pfeiffer 2012). The optimization variable is the parameter for the bending damping, `d_bend`.

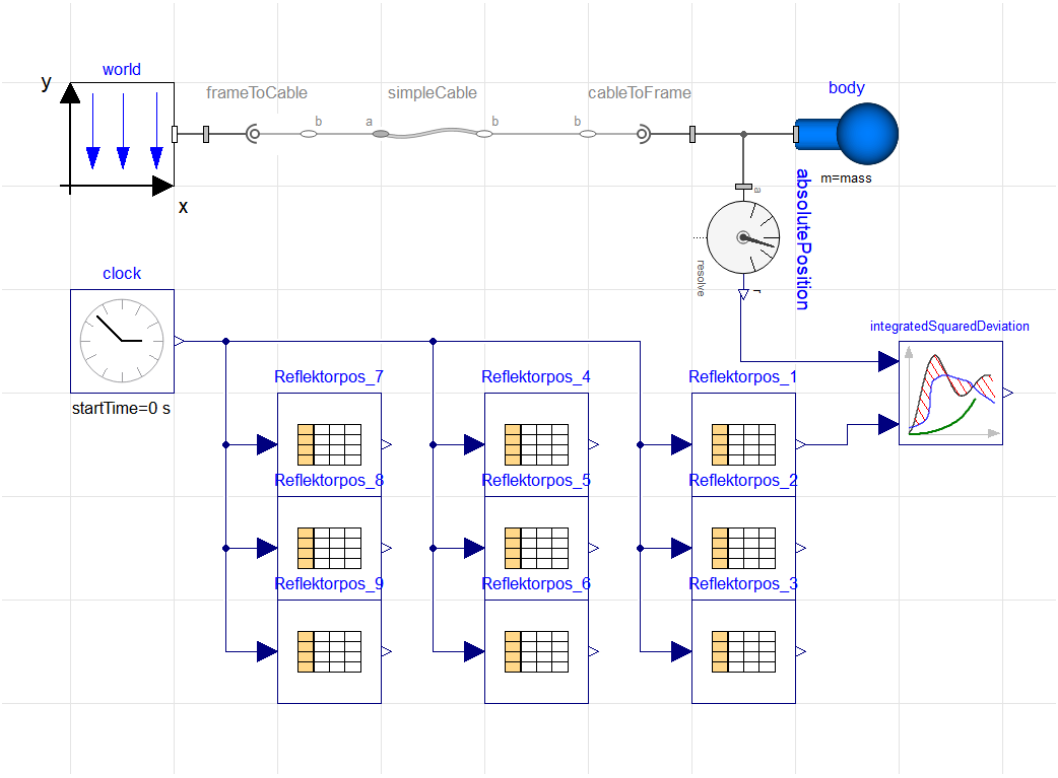
In Fig. 6b the output of an optimization is shown where a small change of the bending damping leads to a very precise match of the simulated and the real movement. Here, the cable was released at the starting position A.

The cable is then deflected both in the simulation and in the test and released from the start position 'E'. After a short time, the oscillation behavior of the lowest point '1' is very similar to the previous case. On the other hand, clear changes can be seen at position '7', see Fig. 7a. It is also easy to see that the simulation and the measurement do not match well in terms of the amplitude of the harmonics. Even after renewed optimization, see Fig. 7b, the result is not as good as expected.

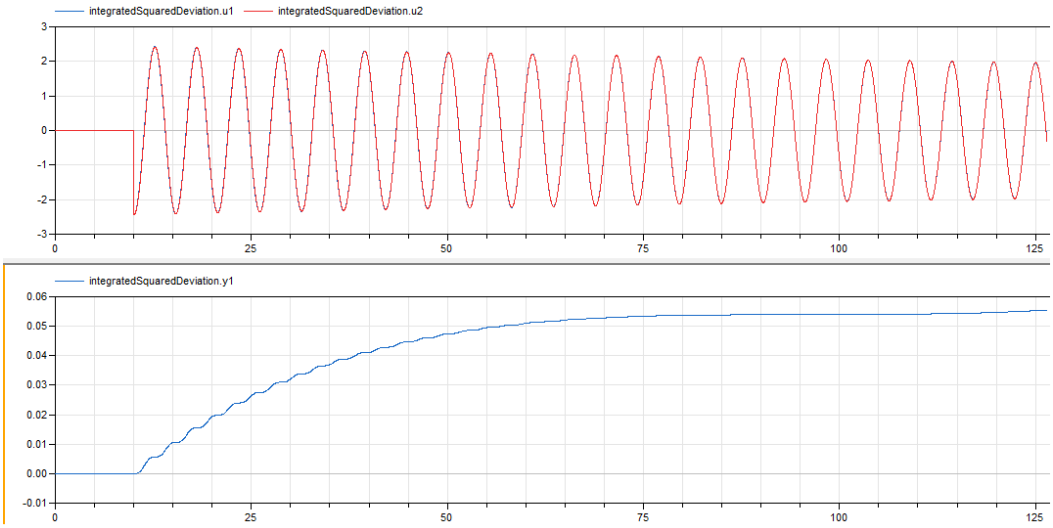
4 Discussion and Conclusion

The validation of simulation models is crucial in order to be able to rely on the results. In a first step, the parameters for bending stiffness and bending damping were deter-

DOI
10.3384/ecp21737



(a) Modelica model to run an optimization with the *DLR Optimization library* (Pfeiffer 2012). Here, only the position of the additional mass is compared in simulation and from real measurements, that have been recorded and are used here as a lookup table.



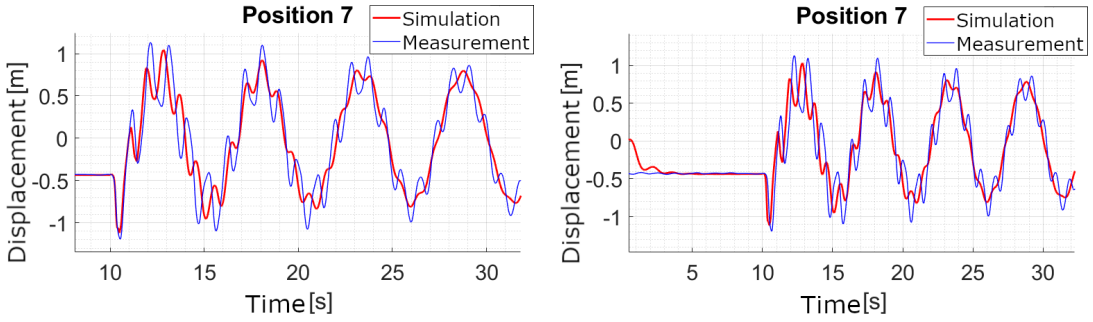
Evaluation 100:

Tuner parameters	name	value	difference to start	min	max
	d_bend	0.063	2.778e-3	0.04	0.1

Criteria	name	scaled criteria	diff. to start	unscaled criteria	usage	demand value
	integratedSquaredDeviation.y1	0.061	-95.7%	0.061	minimize	1
	Sum of squared criteria	3.736e-3	-99.8%			

(b) Result of the optimization shown in Fig. 6. A small change of the bending damping parameter `d_bend` leads to a very good match between the measured, real swinging of the cable and the simulation.

Figure 6. Model made in Dymola of a cable simulated with the Cables library and compared to real measurement. The position of marker '1' and the starting position is 'A' is shown, see Fig. 5a. The comparison is used to optimize the bending damping parameter.



(a) Comparison of simulation and measurement for a cable, started at position 'E'.

(b) Comparison of simulation and measurement for a cable, started at position 'E'. Even after optimization, the curves of the higher frequency do not match well.

Figure 7. Comparison of a swinging cable P 524 with 10 mm diameter simulated with the Cables library and real measurement. The position of marker '7' and the starting position is 'E' is shown, see Fig. 5a.

mined experimentally in this work and transferred to the simulation model using an optimization-based method. For simple pendulum movements, the agreement between the simulation model and reality is very good. In the case of excitation, where higher frequencies also occur in the cable itself, the simulation shows a sufficient fit with the reality but can still be improved, for example by choosing a more complex model. Changing the number of elements did not improve the results thus the authors assume that the modelling itself need to be updated if such complex motions are of interest.

An extension of the bending model with a lookup table for stiffness and damping to enable non-linearity could help to improve the accuracy of the simulation compared to the reality. Furthermore, according to literature, the effect of axial load is high so that those experiments should be made, too. At the moment, another validation is carried out using the Motion Suspension System - MSS (Elhardt et al. 2023) and the results will taken into account to refine the modelling of the cable.

References

Bellmann, Tobias, Andreas Seefried, and Thomas Bernhofer (n.d.). "The DLR Cables Library". In: *Asian Modelica Conference 2024*.

Byskov, Esben (2013). *Elementary Continuum Mechanics for everyone: with applications to Structural Mechanics*. Vol. 194. Springer Science & Business Media.

Cao, Xin and Weiguo Wu (2018). "The establishment of a mechanics model of multi-strand wire rope subjected to bending load with finite element simulation and experimental verification". In: *International Journal of Mechanical Sciences* 142, pp. 289–303.

Chen, Yuanpei, Fanming Meng, and Xiansheng Gong (2017). "Study on performance of bended spiral strand with interwire frictional". In: *International Journal of Mechanical Sciences* 128-129, pp. 499–511. ISSN: 0020-7403. DOI: 10.1016/j.ijmecsci.2017.05.009.

Chen, Zhihua et al. (2015-10). "Experimental research on bending performance of structural cable". English. In: *Construction and Building Materials* 96, pp. 279–288. ISSN: 0950-0618. DOI: 10.1016/j.conbuildmat.2015.08.026.

Elhardt, Ferdinand et al. (2023). "The Motion Suspension System–MSS: A Cable-Driven System for On-Ground Tests of Space Robots". In: *IFToMM World Congress on Mechanism and Machine Science*. Springer, pp. 379–388.

Häusler, Leonhard (2019). "Identifikation und Modellierung von Stahlseilen". Bachelorarbeit. Hochschule München. URL: https://elib.dlr.de/139717/.

Jomartov, Assylbek et al. (2023-03). "Simulation of suspended cable-driven parallel robot on SimulationX". In: *International Journal of Advanced Robotic Systems* 20.2. ISSN: 1729-8814.

Katliar, Mikhail et al. (2017). "Nonlinear model predictive control of a cable-robot-based motion simulator". In: *Ifac-papersonline* 50.1, pp. 9833–9839.

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003-11). "The New Modelica MultiBody Library". In: *3rd International Modelica Conference*. Ed. by Peter Fritzson. LIDO-Berichtsjahr=2003, pp. 311–330.

Papailiou, Konstantin O (1995). "Bending of helically twisted cables under variable bending stiffness due to internal friction, tensile force and cable curvature". In: *Doctor of Technical Sciences thesis, ETH, Athens, Greece* 168.

Pfeiffer, Andreas (2012). "Optimization library for interactive multi-criteria optimization tasks". In.

Spak, Kaitlin, Gregory Agnes, and Daniel Inman (2013-01). "Cable Modeling and Internal Damping Developments". In: *Applied Mechanics Reviews* 65.1. DOI: https://doi.org/10.1115/1.4023489.

Yan, Fei et al. (n.d.). "Dynamic modelling and parameter identification for cable-driven manipulator". In: *Current Science* 116.8 (), p. 1331. ISSN: 0011-3891. DOI: 10.18520/cs/v116/i8/1331-1345.

Zhu, W. D., H. Ren, and C. Xiao (2011-04). "A Nonlinear Model of a Slack Cable With Bending Stiffness and Moving Ends With Application to Elevator Traveling and Compensation Cables". In: *Journal of Applied Mechanics* 78.4. DOI: https://doi.org/10.1115/1.4003348.

A Study on the Methodology to Develop Virtual Drive Environment for Autonomous Driving Evaluation

Wonyul Kang[1] Jongho Park[1] Daeoh Kang[1]

[1]iVH, South Korea

Abstract

It takes many hours and restrictions once AD(Autonomous Driving) evaluation based on real tests. This paper presents a methodology for development of virtual driving environment that can replace the real vehicle test. When developing a virtual driving environment, it is important to develop the same virtual element model (Road, Vehicle model, etc.) as the real world. So the high-occupancy BRT (Bus Rapid Transit) bus route in Cheongna zone was modelled using the MMS(Mobile Mapping System) as the openDRIVE format which is the ASAM(Association for Standardization of Automation and Measuring Systems) road standard. In addition, we develop a vehicle model that simulates the dynamic performance of BRT based on Modelica language. Finally, we develop an interface module that integrates the virtual environment, the vehicle model, and the driver model. In conclusion, this paper present virtual test drive platform for AD Evaluation.

Keywords: Autonomous, BRT, Vehicle Model, AWS

1 Introduction

The recent leader companies in autonomous driving are not automobile manufacturers, but IT companies or venture companies. In the case of Google Waymo, it uses Carcraft software to verify edge cases that cannot be tested in real cars and performs tests by driving more than 8 million miles a day virtually. This is the distance that in virtual reality, the real vehicle travels about three times the distance that can be driven in one year. Therefore, it is essential to accelerate the autonomous driving logic based on the virtual driving environment.[1)]

The virtual driving environment consists of basic elements such as road network/logic, road surface, traffic flow, vehicle dynamics model, driver model, weather model, and sensor model. Existing studies have evaluated representative scenarios after creating artificial road models, and in the case of vehicle dynamics models, there is a problem that the low-DOF vehicle model cannot be applied to reflect the vehicle's behavior in response. [2)] The vehicle's responsiveness here means the time and behavior until the vehicle responds to the vehicle's steering and acceleration/deceleration pedal inputs. The factors affecting the vehicle's response are determined by the structure and characteristics of the tire, powertrain, and suspension and in the case of mathematical vehicle models and low-DOFvehicle models, it is difficult to reproduce the characteristics of the vehicle's response to the input.

Therefore, as an alternative to the existing representative scenario-based autonomous driving evaluation method, this study proposes a virtual driving environment that can evaluate control logic in driving distance-based random traffic situations and a vehicle model construction and linkage technique that can reflect the vehicle response and dynamic characteristics of the real vehicle.

In addition, the virtual driving environment developed in this study was developed as a simulation standard defined by the Association for Standardization of Automation and Measuring Systems (ASAM) for usability. The simulation standards defined by ASAM are openDRIVE for roads, openCRG for road surfaces, and openSCENARIO for scenarios. The detailed research sequence is as follows.

This study was conducted to virtually evaluate the autonomous driving logic of the large-capacity refractive type BRT (Bus Rapid Transit) operating in the Cheongna district of Incheon.

As for the order of the study, first, a high-definition virtual driving environment was constructed. Based on MMS, the BRT route of Cheongna district was scanned. By extracting road information from the scanning data, the openDRIVE file was modeled and the actual road was virtualized. Next, a scenario model for driving on a BRT driving route was constructed. In this study, a traffic model was created around the control vehicle (Ego car) by using Pulk Traffic (Random Traffic) supported by the VTD (Virtual Test Drive). The driver model is an autonomous driving logic to be developed and evaluated, and in this study, the autonomous driving logic provided by VTD was used. The sensor model used for autonomous driving performed autonomous driving according to object list information using the Perfect sensor provided by VTD

The second is the development of a modelica-based high fidelity vehicle dynamics model. In order to reflect the dynamic performance of the actual BRT vehicle, the

vehicle dynamics model was modeled using Dymola, a 1D analysis simulation software. In addition, a BRT handling test was performed to validate model responsiveness.
Third, an interface module was developed to interface a virtual driving environment with a 3rd part vehicle dynamic model. The interface module interworks data between the VTD and the dymola vehicle model based on UDP (User Datagram Protocol) communication.
Finally, the driver's seat Mock-up of an actual BRT vehicle was manufactured. And a standardized high-definition virtual environment was developed.

2 Building virtual driving environments based on real roads

In Chapter 2, a virtual driving environment was developed to evaluate large-capacity BRT. For the static element, a high-definition road model was developed using MMS equipment.

2.1 Road Scanning Based on MMS

In Section 2.1, an MMS-based road network model was developed. In this study, the Pegasus 2 Ultimate equipment of Leica Geosystems was used and scanned at a speed of 50 kph. The driving route is from Cheongna International City Station to Gajeong Station. The BRT in Cheongna District has two routes, and the route bus is bus 701 and 702. The scanning data consists of image data generated from a 360-degree camera and four cameras, and is used to generate LAS (LAS) data. Table 1 shows the MMS equipment specifications, and Figure 1 is the vehicle mounting and calibration setup diagram of the MMS equipment.

Table 1 Mobile mapping system specification (Pegasus 2 Ultimate)

	Specification	Remark
Camera	4 ea	12M Pixels
360° camera	1 ea	24MP panorama
Scanner	Z+F 9012	
accuracy	0.015m(Vertical), 0.02m(Horizontal)	

Fig. 1 Mobile mapping system calibration setup

2.2 openDRIVE Road modeling

The LAS data is generated by mapping the image information of the camera corresponding to the cloud points x, y, and z points generated from the LiDAR sensor. The LAS data generation process is as follows. First, the x, y, and z point information, which are cloud point location information, is acquired. A colored cloud point data base is generated by inputting red green blue (RGB) values of pixel information corresponding to x, y, and z of the point by using the camera image information captured when generating the cloud point. The LAS data generated for the BRT driving route in this study is shown in Figure 2.

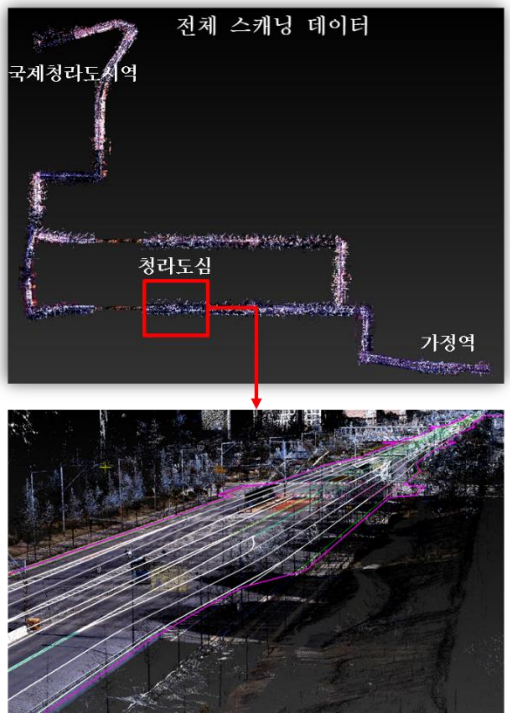


Fig. 2 Scanned las data of BRT route

Next is the classification of cloud points. Also called point grouping of points, LAS data was performed using the AI logic of the Road Factory. Grouping proceeds by object, and point grouping is performed by distinguishing objects such as trees, roads, buildings, and wires. Through point grouping, the user can extract only the LAS data of the desired object. In this study, a road point group was used to utilize LAS data for the road model, and later elements such as buildings and trees were separately modeled using the VTD Road Designer.
Next, openDRIVE-based BRT driving route was created. openDRIVE is an ASAM standard that defines road network information. In openDRIVE, road lanes, widths, altitudes, and bank angles were fitted with a third-order polynomial. In order to virtualize random lanes, vehicle widths, altitude information, etc. extracted based on LAS data generated from MMS equipment, it is impossible to virtualize real roads with a combination of general straight or curved shapes. Therefore, in this study, fitting based on a third-order polynomial was performed, and all road information is defined by the values of a, b, c, and d in Equation (1).

$$y= a+bx+cx^2+dx^3 \quad (1)$$

DOI
10.3384/ecp21743

Here, x is information on the length direction of the road, and y is the width, altitude, and bank angle of the road. In openDRIVE, road attribute information on the road length direction is expressed by the definition of a, b, c, and d values.

In openDRIVE, you can define surrounding environmental objects in style items. openDRIVE includes only location information for user data, and actual graphic data is managed in other databases. In this study, graphic data such as basic buildings, trees, guardrails, streetlights, and traffic lights provided by VTD were used, and openDRIVE includes location information of graphic data. In this way, VTD's virtual environment consists of two types of road network files called openDRIVE and osgb graphic files in the form of openscene graph. In this study, graphic elements such as road marks, stop lines, traffic lights, guardrails, trees, buildings, and underground roads were modeled. Figure 3 shows a landscape graphic model.

Fig. 3 Landscape & tunnel graphic model

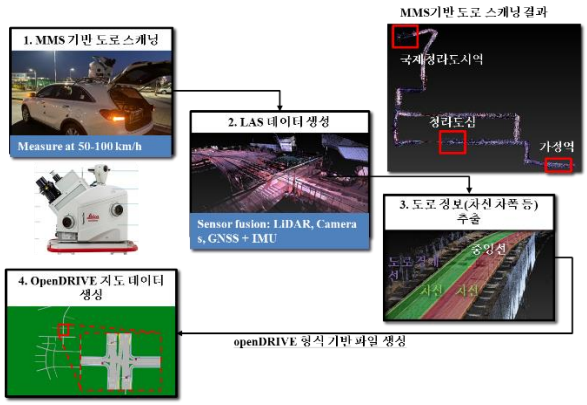


Fig. 4 openDRIVE modeling process based on MMS

Figure 4 shows the process of creating an MMS-based open DRIVE model

2.3 BRT Autonomous-Driving Testing Scenario Modeling

In this study, the scenario was modeled using the openSCENARIO standard. openSCENARIO is the scenario standard of ASAM and expresses the dynamic behavior of an object during simulation. This study evaluated the performance of the controller by modeling the general driving situation for BRT driving routes as a scenario using Random traffic. In the next year, the controller performance evaluation will be improved by developing a test automation function that considers the edge case (cut-in, cut-out, fedarian scenario etc.) for control logic performance evaluation.

As for the scenario for BRT evaluation, a scenario model was constructed in which the location of the bus stop was modeled on the bus line BRT 701 and then stopped for 30 seconds at the bus stop and then departed. The traffic model utilized VTD's Pulk traffic. Pulk Traffic creates a traffic model around a control vehicle (ego) as random traffic provided by VTD. The traffic model generation conditions are as follows. 60 traffic volumes were generated centering on ego vehicles and the traffic flow in the urban area was reflected. In addition, the vehicle distribution was defined as front:40%, rear:30%, left:15%, and right:15% centering on the ego vehicle. The distribution of vehicle types was defined as passenger cars: 75%, van:10%, bus:5%, truck:5%, and two-wheeled vehicles:5%. Details are shown in Figure 5.

Fig. 5 Scenario model based on openSCENARIO

3 BRT Vehicle Model Modelling and Validation

In Section 3, a vehicle dynamics model based on the Modelica language was developed to implement a real vehicle response to steering input, acceleration, and braking pedal. The chassis system modeled a three-axis vehicle model that can simulate bimodal behavior and modeled the steering system on each axis. The powertrain model was deveoloped by modeling batteries, motors, diesel engines, and controllers based on Dimola's Battery Library and Vehicle Dynamics Library. The tire model expressed nonlinear tire behavior by applying the Pacejka Tire model.

3.1 Chassis modeling

In order to implement AWS in the chassis system, the steering function independent of each axis was modeled by applying the steering system to each axis of the vehicle.

The second and third axes were modeled as a Hook (Ball joint) joint to simulate the bimodal behavior of the vehicle. The suspension types on the 1, 2, and 3 axes are all double wishbone types, and the anti-roll bar is modeled on each axis. Figure 6 shows the block diagram of the suspension type.

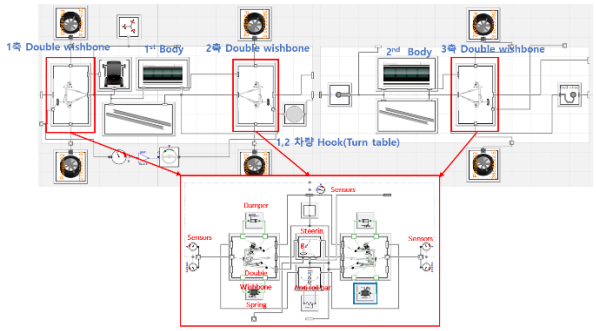


Fig. 6 Modelica based chassis model

The spring of each shaft was modeled as an air spring model, and the air spring formula and performance graph are shown in Equations (2) and Fig.7 below.

$$V_0 = A_v * 0.2$$

$$F_p = dP * A_f$$

$$F_s = c_k * s_{rel}$$

$$V = V_0 + A_v * s_{rel}$$

$$dP = \frac{(P_0 + P_{atm}) * V_0^n}{V^n} - P_{atm} - P_0 \quad (2)$$

$$-F + F_0 + F_p = F_s$$

$$s_{rel} = displacement\ of\ Flange.b - displacement\ of\ Flange.b$$

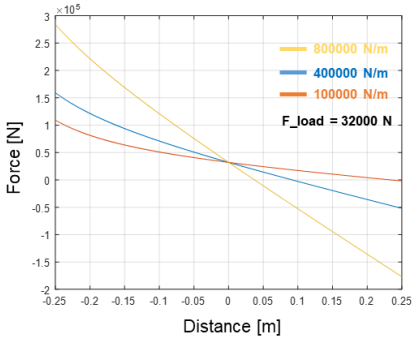


Fig. 7 Air spring stiffness

3.2 Powertrain modeling

The large-capacity BRT is a diesel engine-based serial hybrid vehicle, and Euro 6 209 kW diesel engine and 140 kW generator are applied. The battery was modeled based on Modelon's Battery Library, and the motor was constructed as a model that controls the torque as an angular acceleration target using a controller. The controller converts the displacement of the accelerator pedal into voltage to power the vehicle.

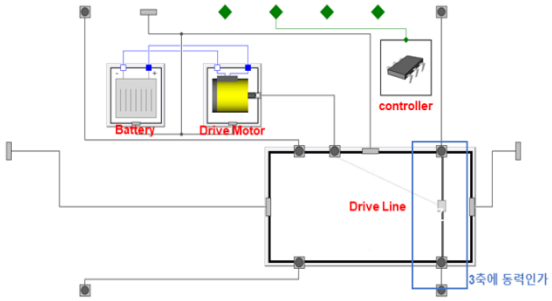


Fig. 8 Modelica based powertrain model

The figure below shows the powertrain integration template. The brake module and powertrain module are interlocked with the chassis module for each shaft.

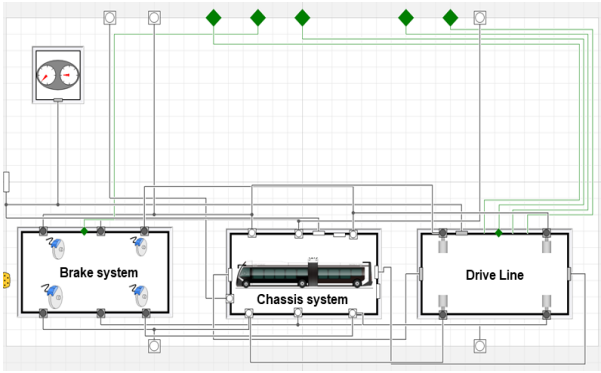


Fig. 9 Modelica based BRT template model

The vehicle dynamics model becomes a sub-module of the test mode, enabling virtual simulation of vehicle dynamics tests.

3.3 Vehicle model Validation

In order to validate the vehicle dynamics model developed in Section 3.2, a real vehicle-based dynamic characteristic test was performed. In this study, tests were conducted in the airways under construction due to the test vehicles and vehicle movement conditions, and acceleration, braking, and double lane change test modes were selected to vaildate the vehicle dynamics model. The maximum speed of a large-capacity BRT is 80 kph. In this study, the acceleration test measured data from a stopped state to reaching 70 kph, and in the case of a braking test, the test data from 70 kph to a stop was measured. Due to the safety of the BRT vehicle of the double lane change, steering

DOI
10.3384/ecp21743

was limited so that a lateral acceleration of 0.2 g occurred at 60 kph for safety reasons.

In this study, the data required for validation were measured through the IMU (Inertial Measurement Unit) sensor installed in the target vehicle. In the IMU sensor, vehicle state information and three-axis data were measured to measure the result of each test mode. In the test, commercial electric buses are refractive buses, and IMU sensors were attached to Front and Rear to vaildate bimodal behavior to obtain measurement results, respectively.
The steering signal and pedal information were measured through the vehicle's controller area network (CAN) communication. The time vs steering angle information obtained in the actual vehicle test was applied to the vehicle dynamics model validation environment, and the dynamic simulation result value and the measured value in the actual vehicle test were compared. The figure below shows that a sensor for measurement is installed on a refractive bus.

Fig. 10 BRT test equipment setup

The vehicle model validation built in this study was performed based on the results of the actual vehicle conducted by the experimental method defined. the simulation validation environment was modeled as shown in the figure below to vaildate the results and model for the acceleration test, braking test, and double lane change test.

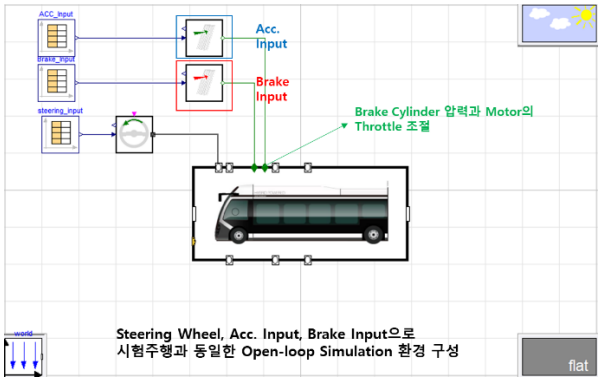


Fig. 11 Vehicle model validation environment

For the validation environment, the steering angle and acceleration and braking pedal values measured in the vehicle dynamics test were applied to the vehicle model as input values.
The actual vehicle test was conducted three times each of acceleration, braking, and double lane change tests, and the analysis of the performance index was performed to vaildate the vehicle dynamics model as the average of the three performance index values in each test.7)

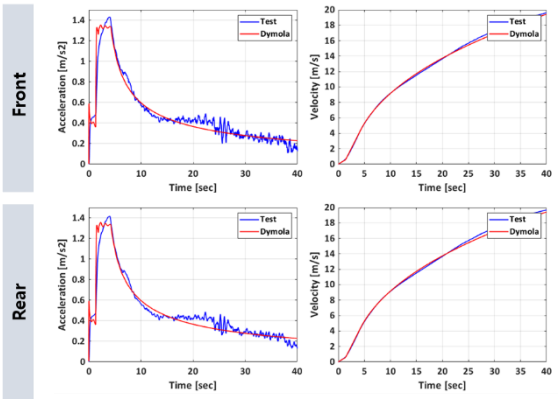


Fig. 12 Acceleration test result

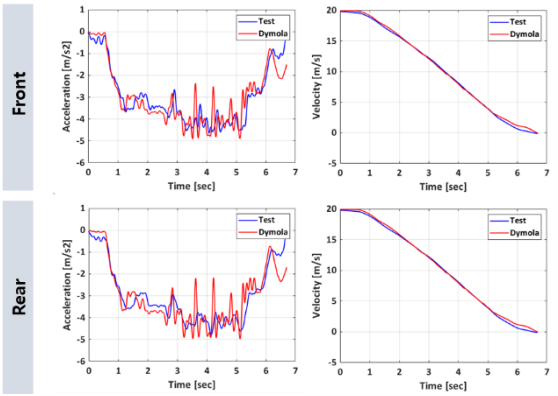


Fig. 13 Braking test result

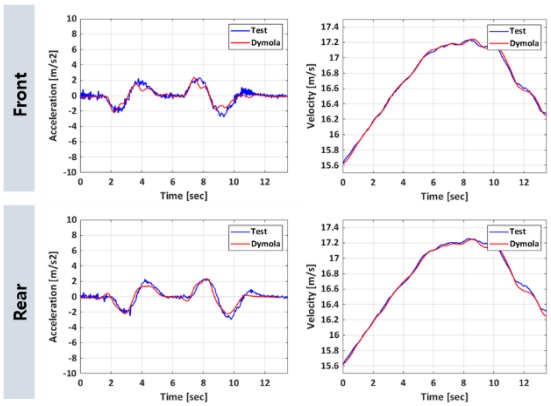


Fig. 14 Double lane change result

The validation was performed based on the speed and acceleration of the front and rear body of the large-capacity BRT. Acceleration and braking tests were verified based on the longitudinal speed and acceleration of the vehicle body, and in the case of a double lane change, they were verified based on the lateral speed and acceleration. The index is expressed as RMSPe as shown in Equation (10), and it shows that the model accuracy of the vehicle dynamic characteristics test results in Table 4 is over the target 80%. In addition, it is thought that the model developed by satisfying the model accuracy of 80% or more and the real-time performance of the model at the same time can be sufficiently used as a vehicle model in a virtual driving environment. Details of the vehicle model validation results are shown in the table below.

$$\boldsymbol{Error}(\%) = \sqrt{\frac{1}{n}\sum\left(\frac{y_s - y_t}{y_t}\right)^2} \times \mathbf{100} \qquad (10)$$

where $y_{s=}$ simulation result
$y_{t=}$ test result

Table 1 Vehicle Test results(RMSPe)

		Acceleration test	Braking test	Double lane change test
Front	Acceleration RMSPe(%)	11.9	16.81	16.91
	Velocity RMSPe(%)	1.92	2.24	1.2
Rear	Acceleration RMSPe(%)	7.34	18.56	19.38
	Velocity RMSPe(%)	2.14	2.31	1.6

4 Development of Virtual Driving Environment and Vehicle Model Interface Module

In Section 4, an interworking environment between a virtual driving environment and a vehicle model was constructed. First, after defining the interworking data items, the virtual environment can be controlled based on the C++ source code defined by the user in the virtual driving environment Plug-in environment. The C++ source code includes vehicle control and communication modules, which can be linked with vehicle models. In this study, a communication module in a virtual driving environment and a communication module of a vehicle dynamics model were constructed, respectively. Figure 15 shows the virtual driving environment and the interface environment between vehicle models. 8)

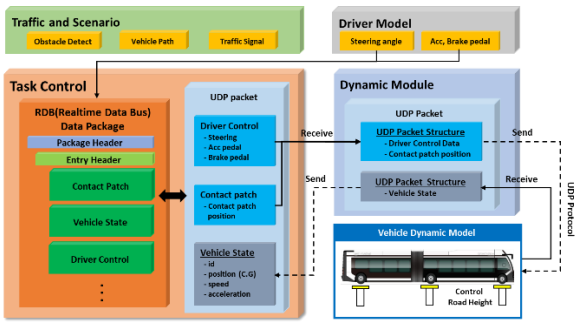


Fig. 15 Data flow based on interface modules

4.1 Modelling UDP Communication Module for Virtual Driving Environments

All data in a virtual driving environment is defined in a data structure called a Runtime Data Bus (RDB). RDB is a standard that defines all data calculated in a virtual driving environment as a structure. Data refers to all components for a virtual environment, such as speed, acceleration, driver signal, location information, pedestrian location, traffic light location, current signal, etc., for all vehicles. The RDB can be output and controlled by a user in a C++ development environment and functions the same as CAN data of a vehicle. In a C++ development environment, the RDB has HEADER and DATA items, similar to a general Packet, and is defined in a structure format.

In order to communicate the RDB data of the virtual driving environment, the C++ source code was generated and the code was executed in conjunction with the virtual driving environment in a Plug-in format. For example, in the RDB structure mentioned in the above section, a code for outputting and transmitting vehicle state information was modeled and a file generated after compile was plug-in to the virtual environment to build an interface environment.

The Driver Model uses the actual steering device and pedal device to provide steering signals by the actual

DOI
10.3384/ecp21743

driver and controls the vehicle with autonomous driving logic when switching control. The user-defined Driver signal is controlled by the Task Control module, a central control module of the virtual driving environment, and reflects the vehicle response by transmitting the driver control signal to the vehicle model.

4.2 UDP Communication Module Modelling for Vehicle Model

Vehicle data was defined with strucrture data to control the BRT model. The structure is composed of vehicle state information and altitude information at each wheel to reflect the altitude of the road surface in a virtual driving environment.

One contact point between the tire and the road in the vehicle dynamics model was designated per wheel. Figure 16 shows the contact mechanism between road and vehicle models on openDRIVE.

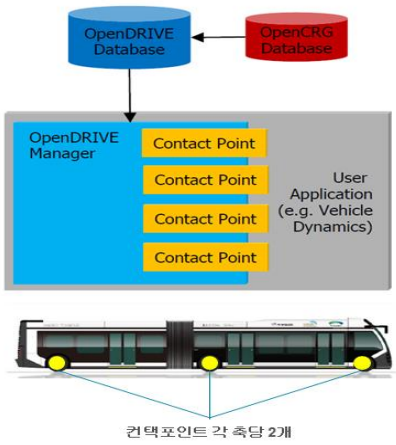


Fig. 16 Road contact mechanism

An interface module was constructed to receive RDB data transmitted in the virtual environment from the vehicle model and transmit vehicle data to the virtual environment. A module capable of receiving actual driver input signals or vehicle controller signals on the steering, acceleration, and deceleration pedals of BRT vehicles, and a module that transmits vehicle state information to the virtual environment were constructed as shown in the figure below.

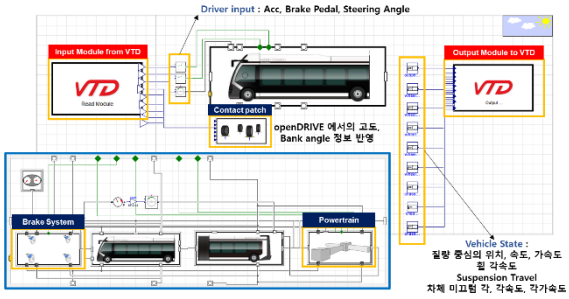


Fig. 17 Interface module for vehicle model

5 Development of Virtual Driving Environment Platform for Autonomous Driving Evaluation

In Section 5, a virtual driving environment platform was constructed using Vires' Virtual Test Drive. In this study, in order to virtually develop the autonomous driving logic of a large-capacity BRT running on the real road in Cheongna District, a virtual driving environment platform was constructed that integrated the previously built road, scenario, and vehicle model based on interface modules. Figure 18 is a summary of the virtual driving environment platform presented in this study.

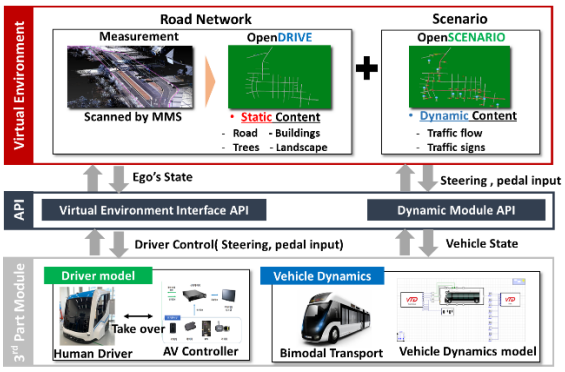


Fig. 18 Virtual drive environment platform for autonomous driving evaluation

In Section 5, the virtual environment of bus routes 701 and 702 of the Cheongna district constructed in this study. The driving simulator manufactured the driver seat of the actual BRT vehicle as a mock-up and the steering angle and pedal input of the controller are implemented through the actuator. In the driving simulator, when the control authority is transitioned, the driver can control the vehicle, and the driver's steering and pedal inputs are applied to the vehicle model, constructing an environment in which the driver can directly drive. The figure below is a diagram of driving the BRT bus route based on an actual driving simulator. The driving rate of the simulation was checked in a random traffic situation, and the driving rate was defined as the number of deviation cases and collisions per number of round trips on the driving route. Based on the VTD autonomous driving logic, 0 cases of route deviation and 0 cases of collision were shown during 236 round trips.

Fig. 19 Driving simulator based on virtual drive environment platform

6 Conclusion

In this study, a high-definition virtual driving environment was developed to evaluate the autonomous driving logic in the preceding stage before converting a large-capacity BRT operating in Cheongna into an autonomous driving system. The VTD (Virtual Test Drive) control logic and the virtual driving environment were linked to the same communication environment as the autonomous driving control logic to be applied to the real vehicles, and the autonomous driving performance for bus routes was evaluated. The VTD autonomous driving control logic will be evaluated by being replaced by the autonomous driving control logic that will be developed for real vehicles in the future. In addition, the vehicle model developed in this study implemented pedal input and powertrain characteristics that could not be implemented when using a low-DOF vehicle model, and a high-fidelity vehicle model was constructed that reflected the response and behavioral characteristics of the vehicle. Through this, a methodology was proposed to develop a controller in consideration of the vehicle response delay phenomenon when developing an autonomous driving controller in virtual. Finally, large-capacity BRT is a public transportation that is actually in operation, and there are many restrictions when evaluating autonomous driving logic based on actual vehicles, and there is a high risk of accidents. Therefore, using the virtual driving environment developed in this study, the control logic was evaluated virtually for a driving distance of 7080 km one-way in the preceding stage. The results derived from this paper are as follows.

(1) A real road virtualization technique based on MMS was established, and a road model that can be evaluated in a simulation environment was constructed by creating a road model of openDRIVE, the ASAM standard.

(2) Through the development of a modelcia-based vehicle dynamics model, a high-precision vehicle model that simulates the dynamic characteristics of the actual vehicle was constructed.

(3) Through the development of interface modules, an environment that can interface 3rd part modules (vehicle models, driver models, and logic) to a virtual driving environment in a plug-in format was established.

(4) Based on the ASAM standard, a highly useful platform was established by constructing a virtual driving environment for autonomous driving evaluation. In the future, when changing the road model, scenario, and vehicle model of the ASAM standard, the platform developed in this study can be used as it is.

In this study, control logic evaluation was conducted in random traffic situations around control vehicles, and actual traffic volume will be applied using traffic simulation tools in the future. In addition, we plan to evaluate the control logic based on the mileage by applying edge cases, accidents, and unexpected situations to the traffic model. Control logic is planning to build and upgrade an evaluation automation environment based on this platform by applying the BRT autonomous driving logic for real vehicles.

7 References

1) State of California Department of Motor Vehicles, Disengagement Report 2019, https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/, 2020.
2) Schiller M, Dupius M, Krajzewicz D, Kern A, & Knoll A, “Multi-resolution traffic simulation for large-scale high-fidelity evaluation of VANET applications.”, Springer Cham, In Simulating Urban Traffic Scenarios, pp.17-36, 2019.
3) J Jo, W Kang, D Kang, “Vehicle Dynamics Model and Tire Filer for Ride Comfort Analysis”, Transactions of KSAE, Vol. 28, No. 12, pp.859-864, 2020.
4) Vehicle Dynamics Library, Modelon, 2019.
5) Vehicle Dynamics – theory and application, Reza N. Jazar, Springer, 2015.
6) Hans B. Pacejka, "Tyre and Vehicle Dynamics", pp.483-512, 2006.
7) M Hyun, J Yoon, G Lee, “Study on Suspension Bush Model for Predicting Frequency and Amplitude Dependent Nonlinear Dynamic Characteristics”, Transactions of KSAE, Vol 28, No.9, pp.621-628. 2020.
8) M Lee, “A study on the construction of commercial electric vehicle fuel efficiency analysis platform based on real road virtual test driving”, Master’s thesis, Kookmin university, Seoul, 2020

DOI
10.3384/ecp21743

Community Updates to the DLR ThermoFluid Stream Library

Raphael Gebhart[1] Philipp Jordan[2] Peter Stein[2] Peter Junglas[3] Niels Weber[1] Peter Eschenbacher[1] Dirk Zimmer[1]

[1]Institute of System Dynamics and Control, German Aerospace Center (DLR) Germany, {raphael.gebhart, niels.weber, dirk.zimmer}@dlr.de
[2]HTWG Konstanz (University of Applied Sciences), Germany, {peter.stein,philipp.jordan}@htwg-konstanz.de
[3]PHWT-Institut, PHWT Vechta/Diepholz, Germany, peter@peter-junglas.de

Abstract

Since its inception in 2021, the user base of the DLR ThermoFluid Stream Library has steadily grown. This growth was accompanied by improved or refined models, new additions such as models for static head pressure and also new examples, especially for teaching. This paper summarizes these updates for the reader and reports on the recent developments.

Keywords: Thermal Modeling, Pipes, Pumps, Heat Exchangers, Static Head, Media modeling

1 Introduction

Since winning the Modelica Library Award in 2021, the DLR ThermoFluid Stream (Zimmer, Meißner, and Weber 2022) Library steadily grew in popularity with users all around the world.

To further encourage both usage but also contributions, we staged the first ThermoFluid Stream (TFS) community event, which took place as a pure online event on June 19, 2024.

The preparations for this event led to many notable contributions for this library which have widened its application field but also have increased the fidelity of its components. This paper reports on these contributions.

1.1 What is special about the DLR ThermoFluid Library?

The ThermoFluid Stream Library distinguishes itself from other approaches by its robust modeling concept (Zimmer 2020) that avoids the formulation of large non-linear equation systems that span across components. Also it supports initialization at zero mass-flow which makes the library relatively easy to use even for complex architectures.

1.2 Overview

Figure 1 providers a quick overview of the library for readers that are unfamiliar with its content. We will highlight certain elements and sections throughout this paper to provide guidance where to find the additions.

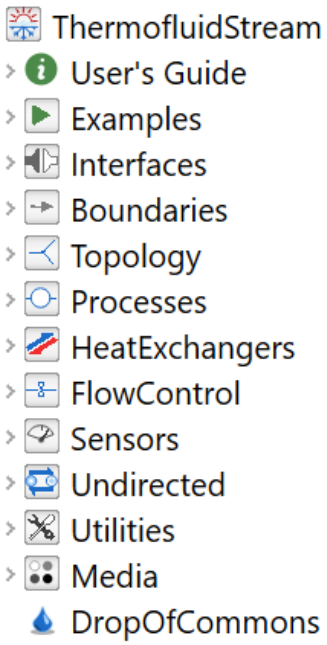


Figure 1. Overview of the TFS library

2 New Concepts and Interfaces

2.1 Static Head Pressure

Pressure increase due to gravitational acceleration had been largely ignored by the prior versions of the Thermofluid Stream Library. A simple tank model with its height vector defined as being parallel to the gravitational vector was the only exception.

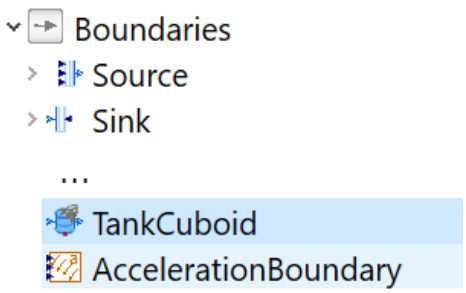


Figure 2. cuboid tanks and global acceleration model

The new update now includes models that aim to provide a more general solution than just constant gravitational acceleration. Instead the thermofluid system can even be regarded as part of a moving object with a globally defined vector for acceleration that can change over time. This is for instance relevant for applications in aircraft or spacecraft.

To this end, a second inner/outer model is provided to specify the acceleration vector. There exists a static head model for pipes that computes the static head pressure based on the coordinates of its starting and end point. Furthermore, there is an elaborate model of cuboid tanks that approximates the liquid surface based on the direction of the acceleration vector and the geometry. Knowing this, enables the computation of the static head pressure for its various inlets and outlets.

This solution is also provided for bi-directional components.

2.2 Interface to TIL Media

The usage of sophisticated media models plays an important role in the modeling of thermal architectures. Robust media models with high computational performance are inevitably needed, especially for phase changing refrigerants that become more important in future thermal management systems. As such media models are provided in the TILMedia Suite (*TIL Media Suite* 2024) developed by TLK Thermo GmbH, an interface to enable their usage within the TFS was developed (TILMediaWrapper).

The current state of the wrapper is a Dymola specific implementation and only supports vapor-liquid equilibrium (VLE) fluids (phase changing media). In general, the interface follows the structure of the TILMedia Suite itself. The basic idea is to adjust the VLE-Fluid model of the TILMedia Suite with pressure, enthalpy and mass fraction as independent variables to match the TFS media interface. In the corresponding package, the functions of partialTwoPhaseMedium are redeclared and the thermodynamic properties are calculated with the functions from the TILMedia Suite.

For each media model that needs to be interfaced, an according substance record has to be defined and then custom media models can easily be added. Also media models inherited from other libraries as Refprop, Coolprop or the VDI Heat Atlas are available and can be added.

Keep in mind that the package only provides the functionality of an interface to the TILMedia Suite and does not include the media models itself! For usage of the media models, a license of the TILMedia Suite is necessary.

3 Improved Components

3.1 New Pipe and Fitting Models

For this update, piping models for pipes and fittings respectively like curved bend, edged bend, sudden contraction and expansion, as well as edged orifice have been developed utilizing the provided pressure loss functions in `Modelica.Fluid.Dissipation.PressureLoss`. The components calculate either the pressure change from inlet to outlet Δp depending on the mass flow rate $\dot{m}$ or the mass flow rate $\dot{m}$ depending on the pressure change Δp along the component using medium properties and geometry. The advantage of re-using standard library components for TFS is in their well known and well validated background.

Additionally, several new models for diffuser, Y-splitter and Y-junction flow have been implemented, based on Idel'chik and using the TFS framework. All novel models are verified against literature or partially validated against CFD flow simulations.

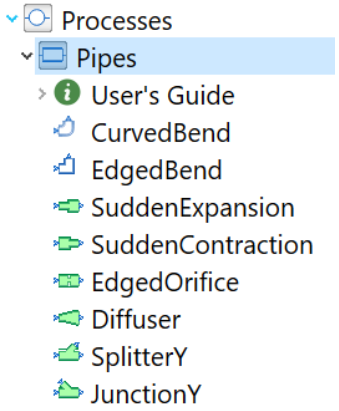


Figure 3. new pipe models

The curved and edged bend models are adaptions of the respective models from `Modelica.Fluid.Fittings.Bends` to enable their usage within the TFS environment. Both bends consider wall friction as well as geometry induced pressure losses. The flow path of the edged bend is assumed to be five perimeters up- and downstream of the edge. For incompressible application it is recommended using default setting `assumeConstantMaterialProperties = true`, while neglecting this choice is beneficial for compressible calculations e.g. $Ma > 0.3$.

The models for sudden contraction and expansion of a circular pipe are utilizing the sudden-change pressure loss function from `Modelica.Fluid.Dissipation.Orifice` and extend the novel partial model `SISOFlow_nonConstArea`, which is based on the Bernoulli equation. The models for sudden contraction and expansion respectively are basically identical and differ only in their icons as well as flow direction.

The edged orifice model is equivalent to `Modelica.Fluid.Dissipation.Orifices.ThickEdgedOrifice` and therefore it shares the same pressure loss function. The difference to Modelica Standard Library (MSL) is in the definition of the Darcy friction factor λ_{fric} as parameter, which enables a calibration of the model.

The diffuser model is a model of a conical diffuser

DOI
10.3384/ecp21751

based on equations from Idel'chik (1966). Here, the local coefficient of flow resistance consists of two components, one for expansion and the other for friction. k_1 characterizes the state of the boundary layer, here a uniform velocity profile is assumed ($k_1 = 1$).

$$\zeta_{dif} = k_1 \cdot \zeta_{exp} + \zeta_{fr} \tag{1}$$

For diffuser opening angles $\alpha \leq 40°$ the flow resistance is calculated as

$$\zeta_{exp,1} = k_2 \cdot \tan\frac{\alpha}{2} \cdot \sqrt[4]{\tan\frac{\alpha}{2}} \cdot (1 - \frac{A_0}{A_1})^2 \tag{2}$$

with k_2 for the cross sectional shape of the diffuser (circular: $k_2 = 3.2$) and an inlet to outlet area-ratio A_0/A_1. For $\alpha > 60°$ a polynomial function is used

$$\zeta_{exp,2} = p_1 \cdot \alpha^2 + p_2 \cdot \alpha + p_3 \tag{3}$$

with the coefficients $p_{1,2,3} = f(A_0/A_1)$ which are manually fit to the diagrams in Idel'chik (1966). The transition region $40° \leq \alpha \leq 60°$ is approximated by a cubic-hermite spline (C-Spline). The wall friction is determined by

$$\zeta_{fric} = \frac{1.5 \cdot \lambda}{8 \cdot \sin\frac{\alpha}{2}} \cdot \left(1 - \frac{A_0}{A_1}\right)^2 \tag{4}$$

Equation 4 is an adaption of Idel'chik (1966), since the original equation leads to non-physical pressure losses for small angles and it is based on the work of Mfon et al. (2019).

The model for flow splitter assumes a diverging y-pipe of branching angles $15° \leq \alpha \leq 90°$ with circular cross section. For implementation of the basic flow equations a partial model similar to `ThermofluidStream.Topology.SplitterN` is used, but limited to 2 outlets and extended by dynamic pressure. Two geometry types are distinguished
I) straight pipe with attached pipe branch ($A_b + A_s > Ac$ and $A_c = A_s$)
II) straight pipe splitting in two smaller branches, whereby the inlet area equals the overall outlet are ($A_b + A_s = A_c$). Depending on the users choice the corresponding pressure loss function based on Idel'chik (1966) is used. Each function calculates both flow resistances, one in the straight channel ζ_s and the other in the pipe branch ζ_b with respect to the velocity in the common channel v_c and outputs the respective pressure loss Δp. The model is limited with respect to the flow design direction.

The junction model is the flow-reversed equivalent to the splitter model. It describes the same geometry specifications and differentiation as the y-shaped splitter. Since mixing fluid streams is far more complex than separating, a specialized partial model is required. This model uses an adaption of `ThermofluidStream.Topology.JunctionN`, calculating the mixing properties for two fluid streams of the same medium, while taking the conservation of kinetic energy into account. The extending model calls the pressure loss functions based on (Idel'chik 1966), depending on the geometry type.

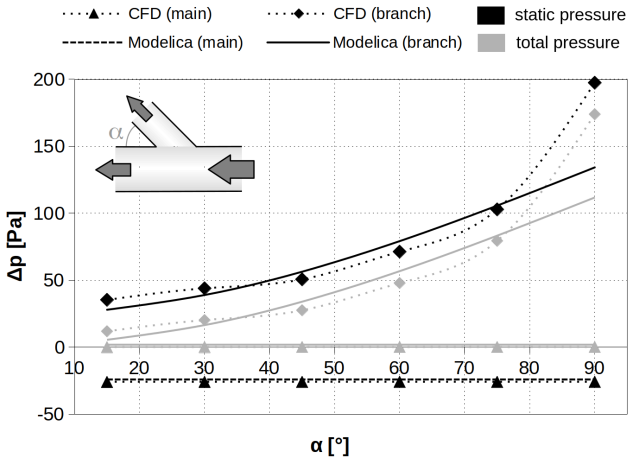


Figure 4. Validation of the Modelica splitter flow model against CFD

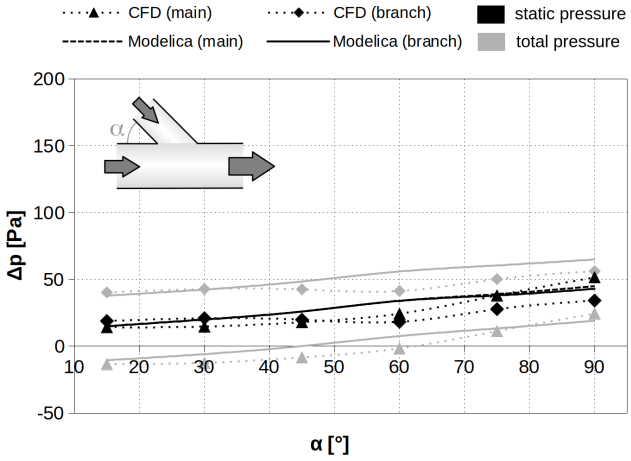


Figure 5. Validation of the Modelica junction flow model against CFD

As mentioned above, some models have been verified against CFD calculations. Those models representing a re-use of the Modelica standard library did not require a validation, only a verification. The models for splitter and junction flow are new, and therefore the were partially validated. The validation has been performed against ANSYS CFX, whereby water as incompressible fluid was taken. In CFD the walls have been treated as free slip walls, because the 1-D models also don't consider wall friction losses. The calculations were performed against a range of splitter/junction angles. Figure 4 and Figure 5 show the result of the validation for both, the static (black) as well as the total (grey) pressure change. Considering the complexity of a 3D flow and the generic usage of such a 1D model, both cases show a good agreement and are therefore useful for pipe flow simulations.

3.2 New Pump Models

Models for simple pumps and centrifugal pumps have been developed. Following the TFS approach all models avoid non-linear equation systems and work for zero mass flow rate $\dot{m}$ and/or zero angular velocity ω.

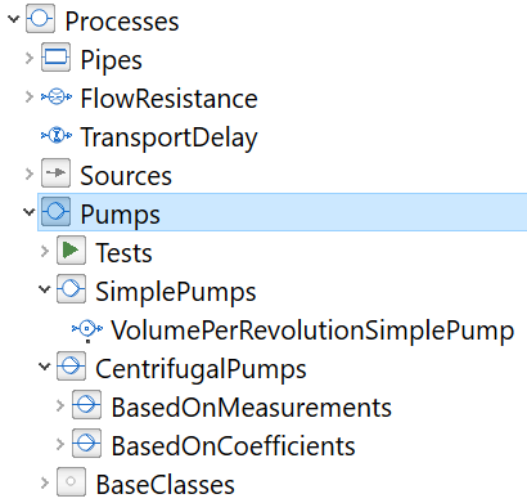


Figure 6. new pump models

A simple pump uses a constant efficiency η and ensures either

1) a given pressure difference Δp or
2) a given outlet pressure p_{out}, comparable to a an electrical voltage source `Modelica.Electrical.Analog.Sources.SignalVoltage`, or
3) a given mass flow rate $\dot{m}$ or
4) a given volume flow rate $\dot{V}$, comparable to an electrical current source `Modelica.Electrical.Analog.Sources.SignalCurrent`.

The set-point may be either a constant parameter or a time varying signal using real input connector.

In addition, there is a simple piston pump model with a fixed volume flow per revolution using a mechanical flange connector comparable to `Modelica.Electrical.Analog.Basic.RotationalEMF`.

Centrifugal pumps use similarity laws for head $h \sim \omega^2$, volume flow rate $\dot{V} \sim \omega$ and power $P \sim \rho\omega^3$ of angular velocity ω and density ρ and quadratic polynomials for head h and power P as a function of volume flow rate $\dot{V}$. The 6 coefficients can be determined by approximation of measurements at reference angular velocity ω_{ref} of head h_i and power P_i at volume flow rates $\dot{V}_i$. In addition, a method was developed to scale existing pumps or to estimate a reasonable centrifugal pump requiring only minimal user knowledge. The TFS provides pump data (measurements and coefficients suitable for scaling) of 19 centrifugal pumps using records. Thereby the user may add a new pump simply by defining a new record.

3.3 New 2-Phase Heat Exchanger Models

When condensation or evaporation happens in heat exchangers, the heat exchange rates change dramatically. Therefore, models become very inaccurate if they describe heat exchangers by constant coefficients.

However in cooling processes, phase changes are occur regurlarly. We therefore developed models for heat exchangers which calculate the heat exchange rates based on the physical processes.

Three types of processes have been considered:

- film condensation of a 2-phase fluid
- evaporation of a 2-phase fluid
- condensation of water vapor in liquid air

With these processes the cooling of humid air as well as cooling by a vapor cycle can be remarkably precisely described. Besides the physical constants of the media only geometric data of the heat exchanger are required.

The description of the physical processes is taken from the text book of (Baehr and Stephan 2010).

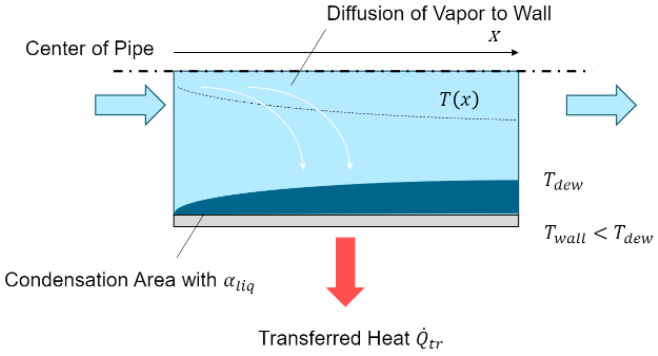


Figure 7. Illustration of the condensation of vapour at the walls of a pipe

4 Tool Support

4.1 Compatibility

We check the compatiblity to differemt Modelica compilers, most notably OpenModelica (Fritzson 2018). These cross checks ensure that the library is available for a wide range of users for free.

4.2 Improved Performacce

A notable improvement regarding the simulation performance has been made in Dymola that benefits the simulation of 2-phase systems. Thanks to this implementation, the examples of vapour cycle and heat pump now run three times faster. The improvements become available with version 2025x.

DOI
10.3384/ecp21751

5 Application

5.1 Application in Teaching

At HTWG Konstanz in Germany, the TFS is used for teaching of system design and simulation focusing on renewable energy systems. The TFS library provides a wide-ranging set of important components, media as well as examples suitable for teaching purpose. Its simple application and robustness enables students to achieve rapid progress in modeling and simulation. Besides the demonstration in lecture, HTWG Konstanz places great value on practical training. In practice lessons and projects the students are empowered to model and simulate complex thermo-fluid systems using TFS as well as developing on models using the TFS-framework.

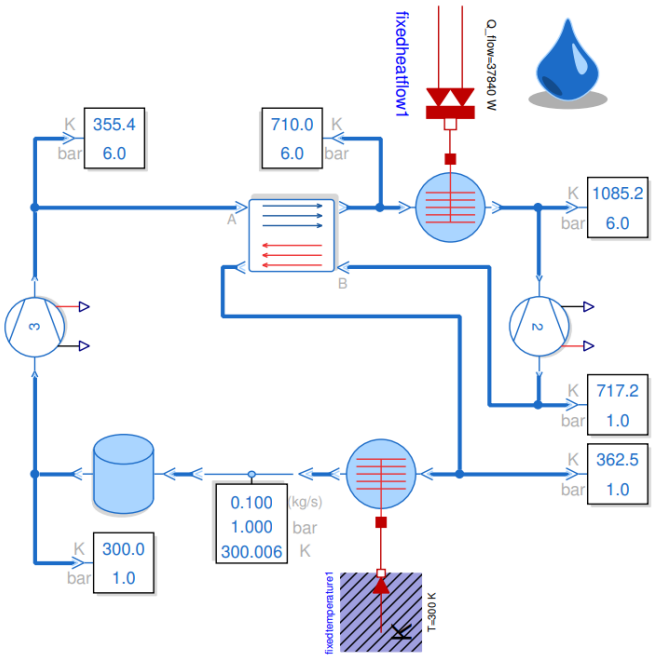


Figure 8. Example of an Ericsson cycle taken from (Junglas 2023)

At PHWT in Vechta/Diepholz in Germany the study of classic thermodynamic cycles is part of the curriculum. Using only literature, this is often very dry and provides little excitement for the students. Using the TFS enabled us to create models of classic cycles such as the Ericsson cycle and the Clausius-Rankine cycle (Junglas 2023). While setting up these simulation models is still substantial work, it can be made available to the students for free using OpenModelica.

5.2 Usage for Aircraft Systems

In the European project Thema4HERA which is part of the Clean Aviation initiative, the aircraft industry, most notably the system suppliers research on new solutions for aircraft thermal management. Here the DLR ThermoFluid Stream Library is used as basis for a Digital Twin. The corresponding model focuses on the main environmental control system including the different cabin sections and is able to perform full mission evaluation.

6 Outlook

We were very glad seeing the DLR ThermoFluid Stream library grow so much in popularity. First, this is a validation of our effort and clearly reveals how much users appreciate robust components and a robust modeling methodology. Second, it encourages us to go on with the development of the library and improve its quality. Finally, we want invite the Modelica community to contribute to this library. A healthy community of users and developers is ultimately best driver for success.

Acknowledgements

We would like to thank Ingela Lind from Saab for the contribution regarding the static head pressure. Furthermore we would like to thank Francesco Casella for always keeping an eye on the OpenModelica compatibility. Also we like to thank the team of Dag Brück from Dassault Systèmes for their effort in improving the performance regarding the vapour cycle models.

References

Baehr, H.D. and K. Stephan (2010). *Wärme- und Stoffübertragung*. Springer Berlin Heidelberg. ISBN: 9783642101946. URL: https://books.google.de/books?id=U60oBAAAQBAJ.

Fritzson, Peter et al. (2018). "The OpenModelica Integrated Modeling, Simulation, and Optimization Environment". In: *Proceedings of The American Modelica Conference 2018*.

Idel'chik, I.E. (1966). *HANDBOOK OF HYDRAULIC RESISTANCE - Coefficients of Local Resistance and Friction*. Isreal Program for Scientific Translations.

Junglas, Peter (2023). "Implementing Thermodynamic Cyclic Processes Using the DLR Thermofluid Stream Library". In: *Simulation Notes Europe* 33, pp. 175–182. DOI: 10.11128/sne.33.sw.10665.

Mfon, Samuel et al. (2019). "A Semi-empirical Model for Estimation of Pressure Drop Coefficient of a Conical Diffuser". In: *Chemical Engineering* 74, pp. 1003–1008. DOI: 10.3303/CET1974168.

TIL Media Suite (2024). URL: https://www.tlk-thermo.com/en/software/tilmedia-suite (visited on 2024-07-26).

Zimmer, Dirk (2020). "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems* 26.3, pp. 204–233. DOI: 10.1080/13873954.2020.1757726. URL: https://doi.org/10.1080/13873954.2020.1757726.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790. URL: https://www.mdpi.com/2079-9292/11/22/3790.

DOI
10.3384/ecp21751

An Integrated Optimization and Orchestration Toolchain for Adaptive Optimal Control in Modelica Simulations

Zizhe Wang[1,2]

[1]Boysen-TU Dresden-Research Training Group, Dresden, Germany
[2]Software Technology Group, Technische Universität Dresden, Germany
zizhe.wang@tu-dresden.de

Abstract

This paper introduces a novel Python-based toolchain, "OptiOrch", designed to enhance optimal control in Modelica-based simulations by integrating an optimization framework and an orchestration workflow. OptiOrch leverages the "MOO4Modelica" optimization framework, which supports both single- and multi-objective parameter optimization, and incorporates the "ModelicaOrch" orchestration workflow to dynamically adapt models based on real-time input data and goals. The toolchain features a user-friendly interface, feature model transformation, parallel computing, and automated workflow coordination, making it a powerful and generalized solution for various applications. Practical examples and a case study demonstrate how this toolchain can be effectively applied to Modelica systems for optimal control.

Keywords: Modelica, simulation, optimization, multi-objective optimization, parallel computing, self-adaptive systems, optimal control, feature model

1 Introduction

Digital twins are becoming increasingly important in research and development. These virtual replicas of physical systems allow for real-time monitoring and optimization. The Modelica language, proposed by Fritzson and Engelson 1998, has emerged as the leading equation-based modeling language for multi-domain, multi-physical systems. It has been widely used in industry and academia to build digital twins of complex systems, such as the modeling and simulation of integrated energy systems (Senkel et al. 2021). Modelica-based software, both open-source and commercial, offers user-friendly graphical interfaces and advanced debugging capabilities. Notable examples include OpenModelica (Fritzson, Aronsson, et al. 2005) (Fritzson, Pop, et al. 2022) (Modelica Association 2023) as an open-source option, and commercial environments like Dymola (Elmqvist 1979) (Brück et al. 2002), Modelon Impact, SimulationX, MWorks (Chen and Wei 2008). Additionally, various open-source and commercial Modelica libraries are available in the community, with the Modelica Standard Library (current release v4.0.0 as of July 2024) serving as the foundational library for all Modelica environments.

Modern multi-domain, multi-physical systems are complex entities comprising diverse components, highlighting the importance of advanced modeling languages like Modelica in their development and optimization. Single-objective optimization is often inadequate for these complex products, necessitating multi-objective optimization (MOO) to address various competing objectives. For instance, in the field of renewable energy, optimizing a wind farm might involve balancing energy output and the minimization of environmental impact (Thirunavukkarasu, Sawle, and Lala 2023). However, the current Modelica ecosystem lacks robust support for MOO, particularly in terms of a generalized open-source framework. This work addresses this gap by proposing a comprehensive open-source MOO framework that leverages Modelica and the Python ecosystem using the OMPython API (Ganeson et al. 2012). As systems become more complex, modeling them also becomes more challenging, especially since many systems need to self-adapt based on different contexts/conditions and performance targets. For example, in cloud computing systems, hardware components like CPU cores and frequencies need to be adjusted based on user demands and specific tasks to achieve the optimal energy-performance balance. Modelica, as a powerful modeling language, is particularly useful for optimizing self-adaptive systems, especially in achieving optimal control.

2 Background

2.1 Optimization and MOO in Modelica

According to Sharma and Kumar 2022, optimization techniques can be categorized into three types: exact (classical) methods, heuristic and meta-heuristic methods, and hybrid methods combining elements of both. Exact methods aim to find optimal solutions within a small, manageable solution space but are often impractical for complex real-world applications. Heuristic and meta-heuristic methods, while not guaranteeing optimal solutions, are more feasible and effective for such scenarios. Hybrid methods leverage the strengths of both exact and heuristic approaches to mitigate their weaknesses. MOO techniques, often classified as stochastic meta-heuristic methods, are divided into three classes: evolutionary, swarm-based, and hybrid algorithms.

Different Modelica environments provide support for single-objective optimization tasks. The current state of MOO in Modelica involves various tools. In the commercial sphere, software like Dymola and Modelon Impact offers MOO support. Dymola includes a comprehensive optimization library (Pfeiffer 2012) (current version v2.2.6 as of July 2024) focusing on general optimization algorithms, including the weighted sum method, which converts a MOO problem into a single-objective problem by assigning weights to each objective. Although Dymola supports MOO, it may lack the specialized algorithms of dedicated optimization tools. Therefore, frameworks like the one by Leimeister 2019 have been designed for Dymola. Modelon Impact provides a cloud-based platform with robust optimization features, enabling users to effectively handle complex multi-objective problems in various engineering and industrial applications. OpenModelica, a prominent open-source Modelica environment, integrates with external optimization libraries and tools to facilitate MOO. However, its specific tool, OMOptim (Thieriot et al. 2011), primarily designed for single-objective optimization, has been excluded from the OpenModelica software and is not currently maintained or further developed. Consequently, developers often create custom methods to integrate Python-based libraries tailored to their specific optimization needs.

There is a critical need for a universal, open-source optimization framework capable of addressing both single-objective and multi-objective optimization tasks. Therefore, a primary objective of this work is to tackle this challenge by developing a comprehensive, open-source optimization framework that robustly supports both single-objective and multi-objective optimization scenarios.

2.2 Optimal Control of Self-adaptive Systems

In real-world, many systems are self-adaptive. For example, a cellphone reduces hardware functionality to conserve power when its battery is low. Similarly, energy systems adjust operations based on user demand, and traffic light systems adapt themselves according to traffic flow. These scenarios require dynamic simulations that can update configurations and parameters based on the real-time conditions and goals. Currently, Modelica environments require developers to manually write scripts for continuous optimization and adaptation. Implementing an automated workflow that optimizes and updates simulations as needed would be significantly more practical and efficient. Such an automated workflow is crucial as it would enable systems to continuously monitor their state and environment, triggering updates to the simulation model with optimized parameters and configurations. This approach ensures that the simulation remains accurate and effective, thereby significantly enhancing the system's performance and reliability. Therefore, another primary objective is to design a robust, automated workflow for generalized orchestration, enabling optimal control for self-adaptive systems in the Modelica ecosystem.

3 The Optimization Framework

Figure 1 illustrates the concept and the structure of the MOO4Modelica optimization framework. Key components of this framework are feature model transformation and optimization operation.

3.1 Feature Model Transformation

This component can be used to transform the Modelica models into feature models. This allows the users to analyze and select parameters and variables that need to be varied and optimized, especially for large-scale models, this would be beneficial. It also allows developers to locate corresponding parameters and variables as well as to identify their relationships in the models quickly.

The method for transforming a Modelica model into a feature model is inspired by the approach described by Zhang et al. 2022. By parsing the selected Modelica model with ANTLR (Parr and Quong 1995), the parameters and variables, along with their types and values, as well as equation sets, can be progressively and recursively obtained. These will then form a feature model, which is saved as a `JSON` file. Still, the users can choose the parameters and variables they want without transforming a Modelica model into a feature model. That is the reason why the two key components are decoupled.

`modelica.g4` This is a grammar file of ANTLR for the Modelica language[1]. It defines the syntax rules that ANTLR uses to generate a lexer and parser for Modelica. Lexer rules specify how to recognize the smallest units (tokens) e.g. keywords, identifiers, and operators. Parser rules define how these tokens are combined to form valid Modelica constructs like expressions, statements, and declarations, specifically in Modelica e.g. classes, components (parameters and variables), and equations. With the help of these rules, the parser generated by ANTLR can understand and process Modelica code.

`parse_model.py` This process parses a Modelica model to extract its components, including parameters and variables, along with their values. Utilizing the ANTLR-generated lexer and parser to construct a parse tree and traverse it to gather the relevant information. Pseudocode 1 illustrates the workflow. By systematically extracting these components, the framework enables users to efficiently identify and manipulate key model elements.

`feature_model.py` This process invokes the `parse_model` function to parse a Modelica model, extracting its components and equations and organizing them into a hierarchical feature model. It includes functionalities to display the feature model and save it to a `JSON` file, which facilitates interoperability with other tools and platforms, enhancing the flexibility and utility of the framework. Pseudocode 2 illustrates the workflow.

[1] https://github.com/antlr/grammars-v4/tree/master/modelica

DOI
10.3384/ecp21757

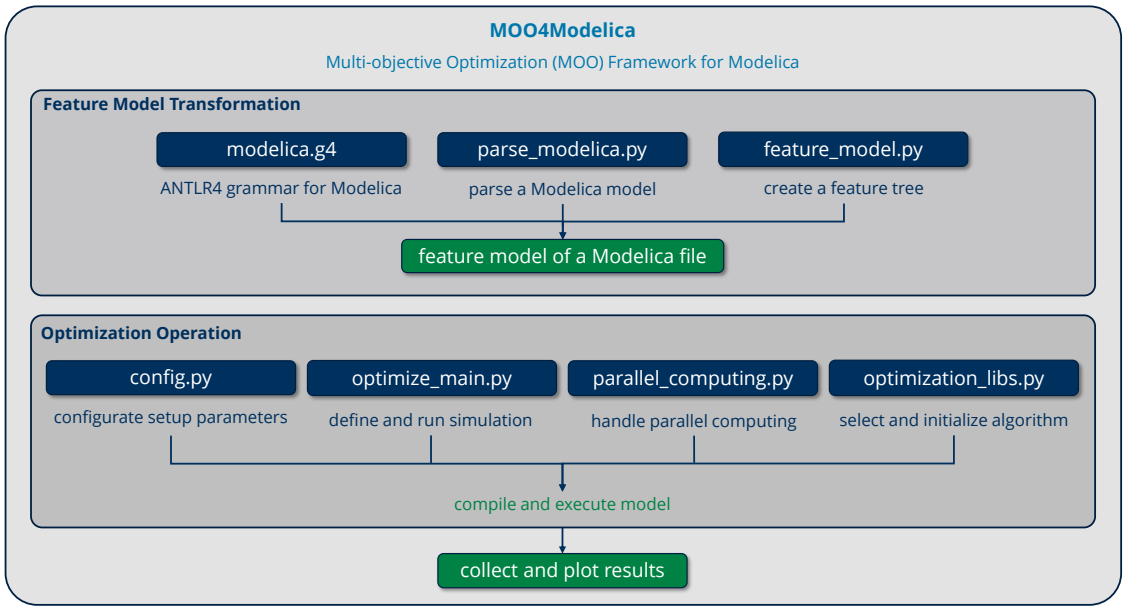


Figure 1. Structure of the framework *MOO4Modelica*.

Pseudocode 1 Modelica Parsing

```
1. Import Libraries and Setup
    Import antlr4 library and lexer/parser;
2. Define FeatureExtracton Class
    Create lists for components and equations;
    Define method to handle component clauses
        Extract component and add to list;
    Define helper method to extract the value
        Extract and add values to the list;
    Define method to handle equation sections
        Extract and add equations to the list;
3. Define parse_model function
    Read and parse Modelica file into parse tree;
    Initialize FeatureExtractor, traverse tree;
    Return components and equations;
4. Main execution block
    Call parse_model and print results;
```

Pseudocode 2 Feature Model Extraction

```
1. Import Libraries and Setup
2. Define FeatureModel Class
    Define method to add components/equations
        Create node for component type and name;
        Add parameters as children nodes;
        Add equation as a node;
    Define method to convert to dictionary
        Convert tree to dictionary format;
3. Main Execution Block
    Call parse_model to get components;
    Initialize FeatureModel with model name;
    Add components and equations to feature model;
    Display and save feature model to JSON;
```

3.2 Optimization Operation

The second component is used for operating optimization tasks. This can be used for both single-objective and multi-objective optimization. This component uses OMPython (Ganeson et al. 2012) as the bridge to connect the simulation of Modelica models to Python. All the global settings have been abstracted into the `config.py`. The main workflow of the optimization operation is outlined in the following steps:

- Step 1: Basic settings
  - Set model name, path, and simulation time.
  - Import external library (if needed).
  - Configure plot diagram settings.
- Step 2: Selection of parameter(s) to be varied
  - Set the range and data type.
- Step 3: Selection of objectives to be optimized
  - Set precision (decimal places) of the results.
- Step 4: Optimization options
  - Select optimization type and algorithm.
  - Set population size and number of generations.
- Step 5: Parallel computing options
  - Enable or disable parallel computing.
  - Set the number of CPU cores to be used.

`optimize_main.py` This process sets up and executes the optimization using configured algorithms and parameters. It involves defining the optimization problem, initializing the algorithm, running the optimization, and subsequently printing and plotting the results. As the main driver for conducting and analyzing the optimization, it ensures a streamlined and efficient workflow. Pseudocode 3 illustrates the workflow.

Pseudocode 3 Optimization

1. Import Libraries and Configuration
2. Define `OptimizationProblem` Class
  Define superclass initializer
    Initialize with `PARAMETERS` need to be varied;
    Initialize with `RESULTS` need to be minimized;
    Initialize with `RESULTS` need to be maximized;
    Set bounds for the parameters;
    Call superclass initializer
  Implement `evaluate` method;
    Convert parameter values to list of dictionaries;
    Parallel processing for evaluation with `n_jobs`;
    Negate objectives that need to be maximized;
    Store results;
3. Initialize Algorithm
  Initialize algorithm based on configuration;
4. Run Optimization and Handle Clean Up
  Define problem instance;
  Cleanup temporary directories;
5. Collect, Print, and Plot Results
  Iterate through results;
  Negate back maximized objectives;
  Print each solution with formatted results;
  Create scatter plot with results;

`parallel_computing.py` This process enhances computational efficiency by facilitating parallel execution of simulations. It defines functions for running simulations with different parameter sets concurrently using the `joblib`[2] library, significantly speeding up data processing and model evaluations. This is essential for handling computationally intensive tasks by leveraging parallel processing capabilities. Pseudocode 4 shows the workflow.

`optimization_libraries.py` This module provides a unified interface for initializing and configuring various optimization algorithms for the optimization operation. By abstracting the complexity of setting up different optimization libraries and algorithms, it simplifies the process of switching between them and configuring their parameters. This module includes the powerful open-source framework *pymoo*, introduced by Blank and K. Deb 2020 which offers state-of-the-art algorithms and features for visualization and decision-making. In this script, users can easily extend its capabilities to meet their specific needs.

[2]https://joblib.readthedocs.io

Pseudocode 4 Parallel Computing

1. Import Libraries and Configuration
2. Initialize Variables
  Initialize `temp_dirs` for temporary directories;
3. Define `optimization_function`
  Create `temp_dir` for each worker;
  Attempt for each worker
    Create OpenModelica session `omc`;
    Copy, load and build model in `omc`;
    Set parameters and simulate model in `omc`;
    Retrieve and return simulation results;
    Shutdown `omc`;
4. Define `shutdown_omc`
  Quit and close `omc`;
  Print success or error message;
5. Define `cleanup_temp_dirs`
  Attempt to remove `temp_dir`;
  Print success message and break loop if successful;
  If `PermissionError` occurs, sleep for backoff;

3.3 Examples

The first example features a simple heating system modeled using Modelica. In this model, increasing the **heating power** will raise the room temperature more quickly, thereby enhancing **human comfort** compared to slower heating. However, this approach results in higher **energy consumption**. Additionally, setting the **target temperature** too high can also decrease **human comfort**. In this context, the key parameters to be adjusted are **heating power** and **target temperature**. The objective is to find the optimal settings that maximize **human comfort** while minimizing **energy consumption**. Table 1 summarizes this scenario, including the parameters to be adjusted, objectives, and the goal.

Parameters	**Objectives**
Heating Power	Human Comfort
Target Temperature	Energy Consumption
Goal	
Maximize Human Comfort	
Minimize Energy Consumption	

Table 1. Parameters, objectives, and goal-setting of example 1.

A feature tree is not required in this simple heating system. The configurations and parameters are directly set in the `config.py`. The default configuration has been used for this example, and the simulation time has been set to 3000 seconds. The ranges for **heating power** and **target temperature** are 1000 - 5000 Watts and 280 - 310 Kelvin, respectively. The NSGA2 algorithm (Kalyanmoy Deb et al. 2002) has been chosen for the optimization. The result shown in Figure 2 displays the corresponding Pareto front of **human comfort** versus **energy consumption**.

DOI
10.3384/ecp21757

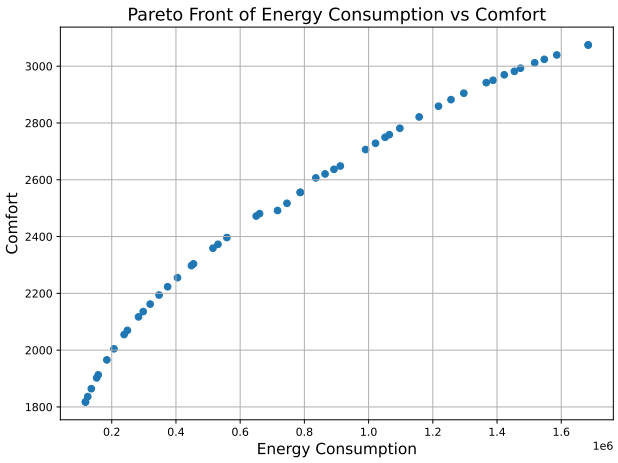


Figure 2. Pareto front of the simple heating system.

Another example involves an electric driving robot modeled using Modelica. This model focuses on a robot where the challenge is to find an optimal balance between **travel distance** and **energy consumption** at various **driving speeds**. In this scenario, **driving speed** is the key parameter to be adjusted. The objective is to fine-tune the speed to achieve the longest possible **travel distance** while minimizing **energy consumption**. Table 2 summarizes this scenario, including the parameter to be adjusted, objectives, and the goal.

Parameter	**Objectives**
Driving Speed	Travel Distance Energy Consumption
Goal	
Maximize Travel Distance Minimize Energy Consumption	

Table 2. Parameters, objectives, and goal-setting of use case 2.

The default configuration was used for this example, and the simulation time was set to 3000 seconds. The range for **driving speed** is 3 - 15 m/s (10.8 - 54 km/h). The NSGA2 algorithm has been chosen for the optimization. Since the diagrams of the Pareto front are similar, the result for the second use case is not shown here.

The results can inform decision-making processes for various applications. For instance, the first example helps decision-makers efficiently develop a strategy for heating a room. The second example involves multiple scenarios: (1) Optimizing an electric robot or electric vehicle for the target range and performance etc. (Dharumaseelan et al. 2021); (2) Optimizing electric car-sharing systems by analyzing vehicle states to select the most suitable car for a client, balancing environmental and economic considerations (Hamroun, Labadi, and Lazri 2020).

4 The Orchestration Workflow

Figure 3 shows the four components of the "ModelicaOrch" orchestration workflow.

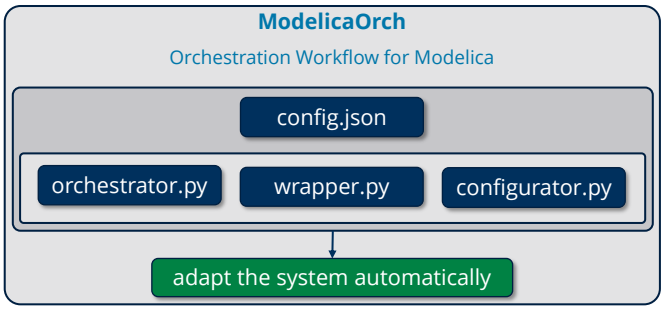


Figure 3. Structure of the workflow *ModelicaOrch*.

`config.json` The configuration file. It also dynamically adapts the MOO4Modelica configuration file.

`orchestrator.py` It initializes components, reads data, runs the optimization, and manages the entire simulation and evaluation loop. It acts as the central orchestration unit, ensuring that each component functions correctly and in sync with the others.

`wrapper.py` It manages the optimization process using MOO4Modelica, handles optimization results, and provides parameter sets for simulation. It effectively bridges the optimization and optimal control processes, ensuring that the best possible configurations are tested.

`configurator.py` The configurator updates the configuration based on the current status. It also prepares and sets parameters for the simulation.

Figure 4 shows how the orchestration workflow operates. After reading input data it enters the adaptive control loop that iterates over defined time units. In each iteration, the `orchestrator` calls the `configurator` to update optimization configurations and the `wrapper` to assign and retrieve optimized parameter sets found by MOO4Modelica. The `configurator` simulates and evaluates these sets to check if the goal is satisfied. If satisfied, the result is added to the final report; if not, the system tries the next parameter set until all options are exhausted. This process repeats until all time units are iterated, culminating in a comprehensive final report of the optimization results. This report can be used to refine the system and guide future adjustments. The adaptive nature of this workflow allows for continuous improvement and ensures that the system can respond effectively to changing conditions and requirements. By automating the optimization and adaptation process, the workflow significantly reduces the need for manual intervention, allowing for more efficient and reliable system management. This capability is particularly beneficial in complex systems where numerous parameters and configurations need to be considered.

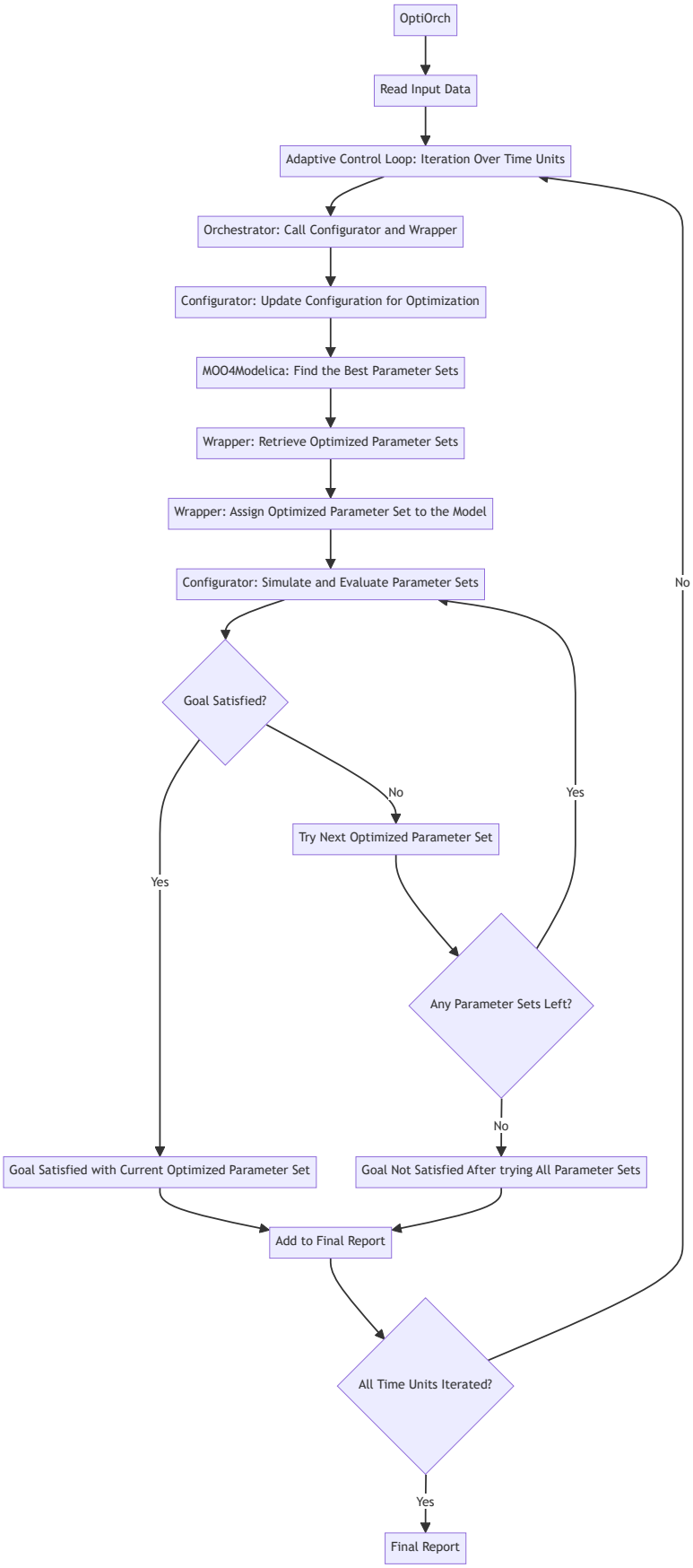


Figure 4. Flowchart of the orchestration workflow: The adaptive control loop iteratively optimizes parameters, updates configurations, evaluates models, and checks goal satisfaction over defined time units, systematically documenting results into a final report.

DOI
10.3384/ecp21757

5 Case Study

In the future, autonomous driving vehicles need distributed edge computing systems for low latency as well as high security and privacy, such as computing and data sharing locally. Based on this background, the case study built a simple edge computing system powered by Photovoltaic. Key parameters in this model are active CPU cores and frequency; different combinations of these parameters will result in various combinations of performance (computing power) and remaining energy, which are the key variables. In real applications, traffic flow varies at different times. For example, during the morning rush hour, available energy is still low, but user demand (the required computing power) is high. From 10 AM to 4 PM, user demand is low, and available energy is medium. During the afternoon rush hour, user demand is high, and the available energy is also high. Therefore, the system needs to adapt to different available energy levels and user demands to meet the user demand while keeping the remaining energy ≥ 0 for each defined time unit.

Input

- Energy Available (hourly)
- User Demand (hourly)

Goals

- **Meet User Demand:** The system aims to provide the necessary performance to meet user demand. User demand is considered satisfied when the system's performance (computing power the system provides) $\geq$ user demand.
- **Maximize Energy Efficiency:** The system seeks to optimize energy consumption to prolong operation and maintain efficiency, ensuring that the remaining energy ≥ 0 at the end of the simulation.

Listing 1 shows the configuration for this case study. The defined time unit is the hour, and the adaptive control loop runs for each hour in the simulation. For convenience, the time range has been selected between 8 AM and 12 AM, despite the input data having a time range of 24 hours. The model will be simulated for one hour (3600 seconds). Both objectives are set to be maximized, and the bounds and data types for parameters to be tuned are set to 1 - 4 (integer) and 1.0 - 3.0 (float), respectively. The goal expressions are set such that performance needs to be greater than or equal to user demand, and the energy should not run out (remaining energy should not be negative). `N_JOBS` has been assigned as "-1", which means that parallel computing is enables for the optimization, using all CPU cores. The `CONFIG_PATH` is the configuration file of the MOO4Modelica optimization framework. For each time unit, the orchestration configuration file will also update MOO4Modelica's configuration file to run optimization.

Listing 1. The configuration file for the case study.

```
"DATA_FILE_PATH": "data.txt",
"CONFIG_PATH": "config.json",
"MODEL_FILE": "ITSystem.mo",
"SIMULATION_TIME": 3600,
"TIME_CONFIG": {
    "START_TIME": 8,
    "END_TIME": 12,
    "TIME_UNIT": "hour"
},
"OBJECTIVES": [
    {"name": "remainingEnergy",
        "maximize": true},
    {"name": "performance",
        "maximize": true}
],
"TUNABLE_PARAMETERS": {
    "PARAMETERS": [
        "activeCores",
        "cpuFrequency"],
    "PARAM_BOUNDS": {
        "activeCores": {
            "bounds": [1, 4],
            "type": "int"},
        "cpuFrequency": {
            "bounds": [1.0, 3.0],
            "type": "float"}
    }
},
"INPUT_PARAMETERS": {
    "available_energy": "
       availableEnergy",
    "user_demand": "userDemand"
},
"CRITERIA": {
    "GOAL_EXPRESSION": [
        "evaluation_results['
           performance'] >=
           simulation_inputs['
           user_demand']",
        "evaluation_results['
           remainingEnergy'] >= 0"
    ]
},
"OPTIMIZATION_CONFIG": {
    "USE_SINGLE_OBJECTIVE": false,
    "ALGORITHM_NAME": "nsga2",
    "POP_SIZE": 10,
    "N_GEN": 10
},
"LIBRARY_CONFIG": {
    "LOAD_LIBRARIES": false,
    "LIBRARIES": [
        {"name": "", "path": ""}
    ]
},
"N_JOBS": -1
```

Figure 5 demonstrated the visualized result of the case study. At 8 AM, neither goal is satisfied. At 9 AM, the first goal is not satisfied. From 10 AM to 12 PM, both goals are satisfied. Based on these results, it is clear that for 8 AM and 9 AM, we need more power and updated hardware to meet the system requirements. For the remaining periods, we have already found the best configurations, which we can now implement into the real hardware.

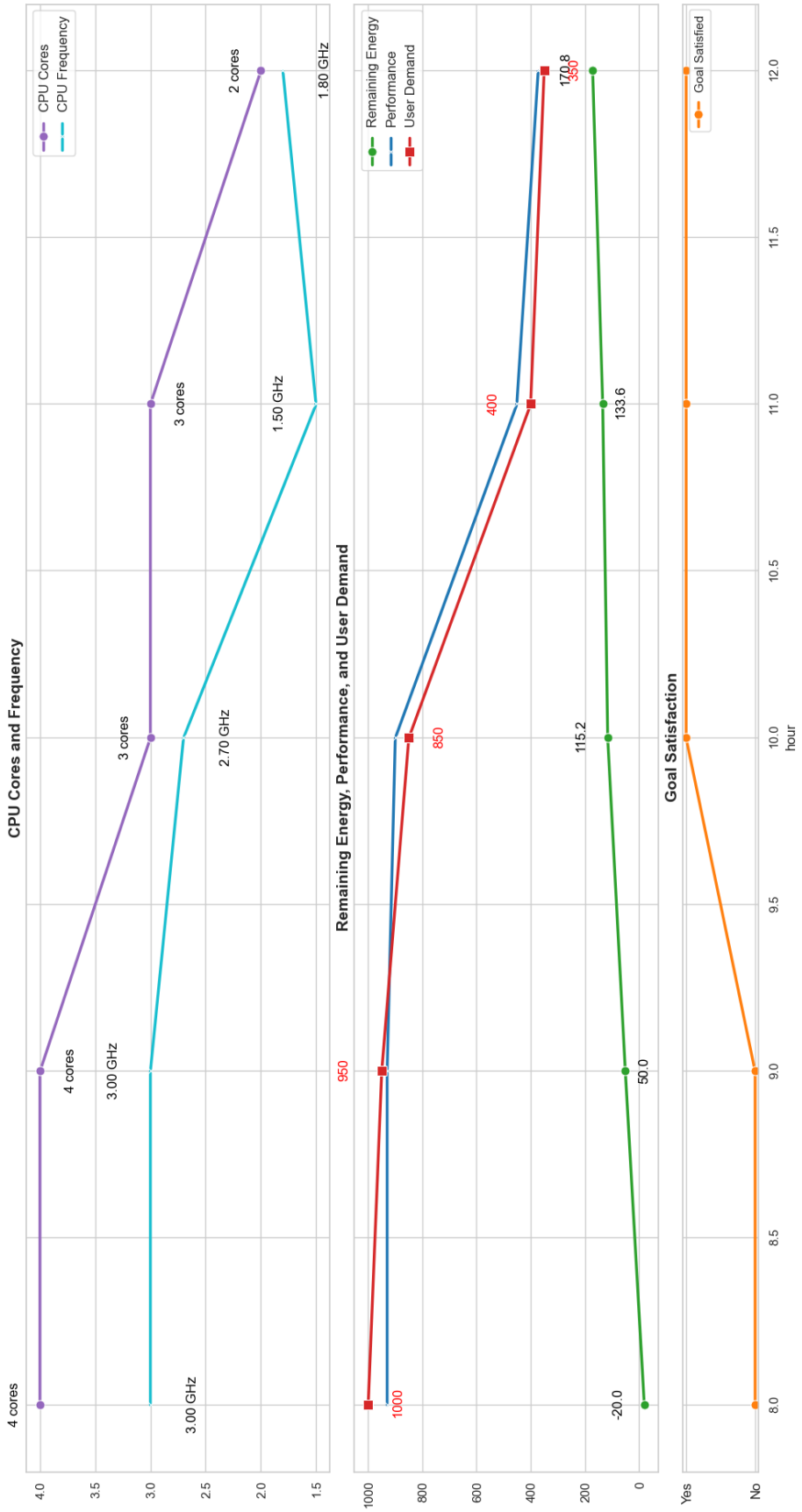


Figure 5. Visualized result of the case study

DOI
10.3384/ecp21757

6 Conclusion and Future Work

OptiOrch[3][4] is a toolchain integrating the MOO4Modelica optimization framework and the ModelicaOrch orchestration workflow. MOO4Modelica facilitates both single- and multi-objective optimization in Modelica-based simulations, featuring user-friendly setup configurations and practical feature model transformations. It leverages parallel computing to enhance performance. ModelicaOrch orchestrates the entire workflow, coordinating the optimization process and updating configurations dynamically. Together, they enable efficient and optimal control in complex Modelica simulations. Additionally, this toolchain is designed to be flexible and extensible, allowing users to adapt it to a wide range of optimization and orchestration scenarios.

Despite its robust capabilities, optimizing and orchestrating large-scale models can be both resource-intensive and time-consuming. To address this challenge, it would be interesting to investigate strategies such as using surrogate models for Modelica-based simulation and optimization (Costa Paulo et al. 2023) or implementing adaptive instance reduction (automatic search space reduction) to reduce the computation complexity. How these two concepts work in the Modelica ecosystem presents interesting research topics. In real-life applications such as edge computing systems, tasks can vary significantly, requiring the system to dynamically allocate resources (CPU, memory, etc.) based on their current demands and priorities. A future goal of this work is to integrate the toolchain with real-time hardware configurators. To achieve this, an architecture will be developed that combines simulation-based optimal control with real-time hardware configurators. By incorporating software like MQuAT (Multi-Quality Auto-Tuning by Contract Negotiation) by Götz 2013 and BRISE (Benchmark Reduction via Adaptive Instance Selection) by Pukhkaiev 2023 into the architecture shown in Figure 6, we can effectively fine-tune and configure real hardware systems to maximize performance and energy efficiency.

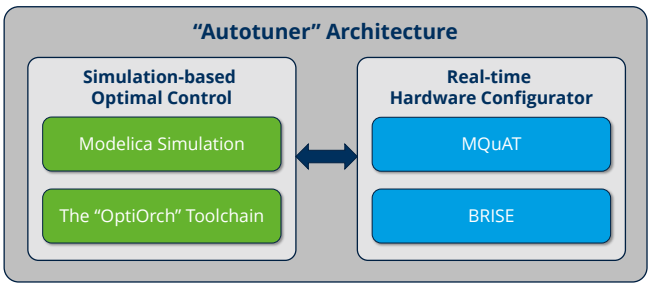


Figure 6. The "autotuner" architecture will define the interface between simulation-based optimal control and real-time hardware configurator.

[3]Repository: https://git-st.inf.tu-dresden.de/wang/OptiOrch
[4]Documentation: https://wangzizhe.github.io/OptiOrch

Acknowledgements

The author would like to thank the Boysen–TU Dresden–Research Training Group for the financial and general support that has made this contribution possible. The Research Training Group is co-financed by the Friedrich and Elisabeth Boysen Foundation and the TU Dresden.

References

Blank, J. and K. Deb (2020). "pymoo: Multi-Objective Optimization in Python". In: *IEEE Access* 8, pp. 89497–89509.

Brück, Dag et al. (2002). "Dymola for multi-engineering modeling and simulation". In: *Proceedings of modelica*. Vol. 2002. Citeseer.

Chen, Xia and Zhongchao Wei (2008). "A new modeling and simulation platform-MWorks for electrical machine based on Modelica". In: *2008 International Conference on Electrical Machines and Systems*. IEEE, pp. 4065–4067.

Costa Paulo, Breno da et al. (2023). "Surrogate model of a HVAC system for PV self-consumption maximisation". In: *Energy Conversion and Management: X* 19, p. 100396.

Deb, Kalyanmoy et al. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2, pp. 182–197.

Dharumaseelan, Elavarasan et al. (2021). "Model Based Analysis and Multi-objective Optimization of an Electric Pickup truck for Range, Acceleration, Drivability, Handling and Ride Comfort Performances". In: *2021 IEEE Transportation Electrification Conference (ITEC-India)*. IEEE, pp. 1–6.

Elmqvist, Hilding (1979). "DYMOLA-a structured model language for large continuous systems". In.

Fritzson, Peter, Peter Aronsson, et al. (2005). "The OpenModelica modeling, simulation, and development environment". In: *46th Conference on Simulation and Modelling of the Scandinavian Simulation Society (SIMS2005), Trondheim, Norway, October 13-14, 2005*.

Fritzson, Peter and Vadim Engelson (1998). "Modelica—A unified object-oriented language for system modeling and simulation". In: *ECOOP'98—Object-Oriented Programming: 12th European Conference Brussels, Belgium, July 20–24, 1998 Proceedings 12*. Springer, pp. 67–90.

Fritzson, Peter, Adrian Pop, et al. (2022). "The OpenModelica integrated environment for modeling, simulation, and model-based development". In: Mic.

Ganeson, Anand Kalaiarasi et al. (2012). "An OpenModelica python interface and its use in PySimulator". In.

Götz, Sebastian (2013). "Multi-Quality Auto-Tuning by Contract Negotiation". In.

Hamroun, A, K Labadi, and M Lazri (2020). "Modelling and performance analysis of electric car-sharing systems using Petri nets". In: *E3S Web of Conferences*. Vol. 170. EDP Sciences, p. 03001.

Leimeister, Mareike (2019). "Python-Modelica framework for automated simulation and optimization". In.

Modelica Association (2023-03). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.pdf.

Parr, Terence J. and Russell W. Quong (1995). "ANTLR: A predicated-LL (k) parser generator". In: *Software: Practice and Experience* 25.7, pp. 789–810.

Pfeiffer, Andreas (2012). “Optimization library for interactive multi-criteria optimization tasks”. In.

Pukhkaiev, Dmytro (2023). “A Software Product Line for Parameter Tuning”. In.

Senkel, Anne et al. (2021). “Status of the transient library: Transient simulation of complex integrated energy systems”. In: *Modelica Conferences*, pp. 187–196.

Sharma, Shubhkirti and Vijay Kumar (2022). “A comprehensive review on multi-objective optimization techniques: Past, present and future”. In: *Archives of Computational Methods in Engineering* 29.7, pp. 5605–5633.

Thieriot, Hubert et al. (2011). “Towards design optimization with OpenModelica emphasizing parameter optimization with genetic algorithms”. In: *Proceedings of the 8th International Modelica Conference*. Vol. 63, pp. 756–762.

Thirunavukkarasu, M, Yashwant Sawle, and Himadri Lala (2023). “A comprehensive review on optimization of hybrid renewable energy systems using various optimization techniques”. In: *Renewable and Sustainable Energy Reviews* 176, p. 113192.

Zhang, Congcong et al. (2022). “A Multi-objective Optimization Algorithm and Process for Modelica Model”. In: *2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*. IEEE, pp. 9–13.

DOI
10.3384/ecp21757

Modeling Fuel Cell Electric Vehicle for Performance Prediction and Optimal Component Selection

Juhyeong Park[1] Kwonhee Suh[2]
Junbeom Lee[1] Donkyu Joo[1] Junghun Yun[1] Daeoh Kang[1*]
[1]iVH, Republic of Korea, {jhp, jblee, dkjoo, jhy, bigfive*}@ivh.co.kr
[2]KIA Corporation, Republic of Korea, lastadam@kia.com

Abstract

This study involves modeling and simulating a Fuel Cell Electric Vehicle (FCEV) to predict whether it meets the target performance requirements. The FCEV model includes an electrified powertrain, composed of a hydrogen fuel cell, motor, battery, and controller, along with a chassis model. A test environment was also modeled to evaluate these components. Different combinations of chassis and motor candidates were examined to predict vehicle performance for each configuration and determine if the target requirements were met. The results of this study served as a reference for selecting optimal components during the development process.

Keywords: Fuel cell electric vehicle, Electrified powertrain, Model based system engineering

1 Introduction

With the increasing demand for eco-friendly vehicles, such technologies are gaining attention in various fields, including the military. Military electric vehicles, in particular, are highly regarded for their low heat emission, reduced noise, which enhances concealment, and the inherent mobility unique to electric vehicles.

According to the IP Defense Forum (2024), South Korea views the utilization of hydrogen fuel cell vehicles in military operations positively. Fuel Cell Electric Vehicles (FCEVs) are especially preferred for their rapid refueling capabilities and long driving range, further highlighting their suitability as military electric vehicles.

Additionally, the International Energy Agency (2019) predicts that hydrogen will account for 24% of the global energy mix by 2050. In response to this global trend, South Korea is actively refining its policies and regulations. Against this backdrop, FCEVs are emerging as a vital solution that meets the dual objectives of sustainable energy transition and advancements in military technology.

Model-Based Systems Engineering (MBSE) utilizing electric vehicle models can significantly streamline the design and development process, enabling more efficient achievement of target performance goals. According to Shevchenko, N. (2020), MBSE enhances traceability across requirements, design, analysis, and validation, ensuring consistency and efficiency throughout the system's lifecycle. Additionally, performance prediction through modeling supports optimal component selection and facilitates effective risk management as specifications evolve during the development process.

In this study, an electrified powertrain model comprising key components of an FCEV was developed, and simulations were conducted on various component specifications to predict performance. Through this process, optimal components were selected, and specifications were evaluated.

2 Vehicle Modeling

Vehicle models consist of a chassis, an electrified powertrain, and a brake model. In this study, two chassis models and three electrified powertrain models were created based on their specifications. By combining these, a total of four vehicle models were generated.

Table 1. Architecture combinations of each vehicles

Number	*Chassis*	*Powertrain*
1	1	1
2	2	1
3	2	2
4	2	3

2.1 Chassis

The chassis model calculates the vehicle's behavior based on vehicle dynamics, taking into account driving force, braking force, steering input, and driving resistance. Driving and braking forces are input from the powertrain and brake models, while steering input is provided by the driver model. Driving resistance is calculated through each component of the chassis model.

The chassis model comprises body, suspension, and tire models. The body model includes a mass model and an

aerodynamics model, with parameters set for sprung mass, center of gravity, inertia, drag coefficient, and frontal area to calculate air resistance. For efficiency, the suspension model also adopts a lumped mass approach, focusing on wheel center position and spring and damping characteristics. The tire model includes wheel weight information and calculates rolling resistance based on a rolling resistance coefficient. In this study, two types of chassis models were created for each specification based on Vehicle Dynamics Library from *Modelon AB*(2021).

Table 2. Comparison between chassis models.

Specifications	*Chassis 1*	*Chassis 2*
GVW	+	++
Tire dynamic radius	+	++
Frontal Area	++	+

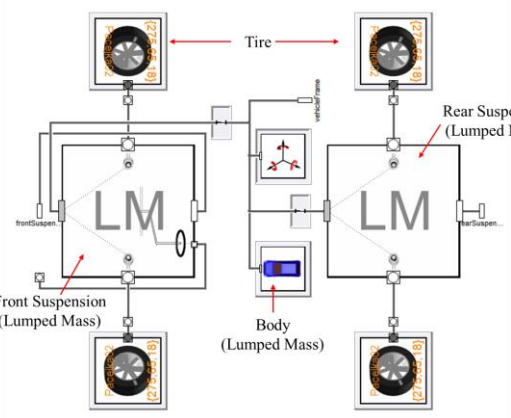


Figure 1. Chassis model.

2.2 Electrified Powertrain

The electrified powertrain model for the FCEV consists of the following subsystems. The subsystems are based on Electrification Library from *Modelon AB*(2021).

- Hydrogen Fuel Cell
- Motor
- Battery
- Controller

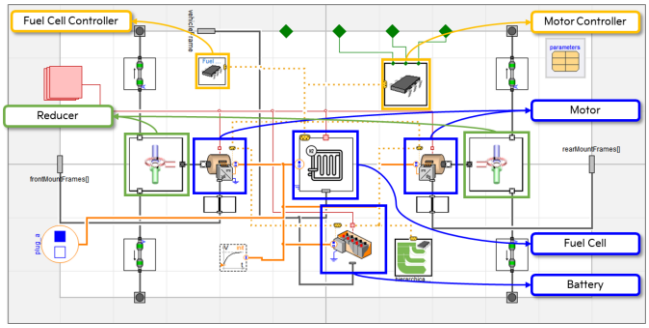


Figure 2. Electrified powertrain model for FCEV.

The hydrogen fuel cell model is designed to calculate hydrogen consumption based on the power demand, using a battery model as its foundation. The model incorporates the current-voltage characteristic curve, with resistance values tuned to reflect this curve. A tabular model with current-hydrogen consumption curve data is used to calculate hydrogen consumption according to the current level. The load model connected to the hydrogen fuel cell model simulates the power consumption of the Balance of Plant (BOP), enabling the calculation of the gross and net output of the hydrogen fuel cell.

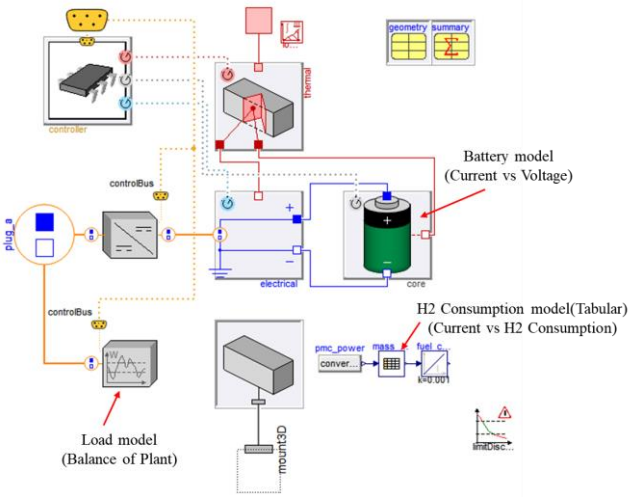


Figure 3. Simple Fuel cell model based on battery model.

DOI
10.3384/ecp21767

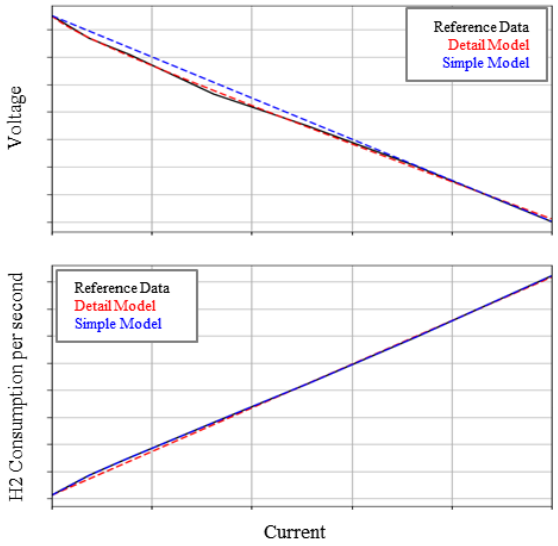


Figure 4. Fuel cell model(simple) validation.

The motor model simulates either a DC motor or an AC motor with an inverter. The motor's torque map model calculates the maximum torque based on motor speed according to the set maximum power and torque limits, restricting torque if the demanded torque exceeds the calculated maximum. The efficiency model uses efficiency values or an efficiency map to determine the power consumption based on motor output. The calculated torque is transmitted to the chassis model via a mechanical connector, while the consumed power is sent to a power source model, such as a battery or hydrogen fuel cell, via an electrical connector to calculate SOC or hydrogen consumption. Thermal losses, calculated from the difference between consumed power and mechanical output, allow the motor's temperature to be tracked within the thermal model.

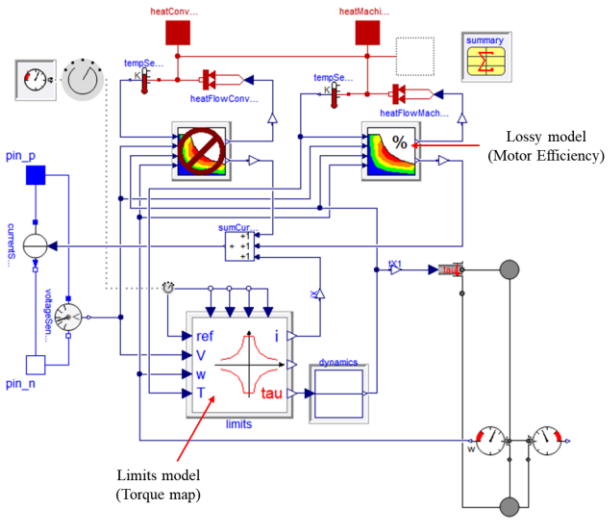


Figure 5. Electric machine(motor & generator) model.

The battery model is configured to supplement the output of the hydrogen fuel cell. The battery characteristics are modeled by setting the cell capacity, OCV curve, internal resistance properties, and cell configuration details. The capacity model calculates SOC based on the set capacity and consumed charge. The OCV model simulates the discharge characteristics of the battery according to SOC. The resistance model represents the internal resistance of the battery cells, calculating voltage drop and, through losses, determining the battery's temperature.

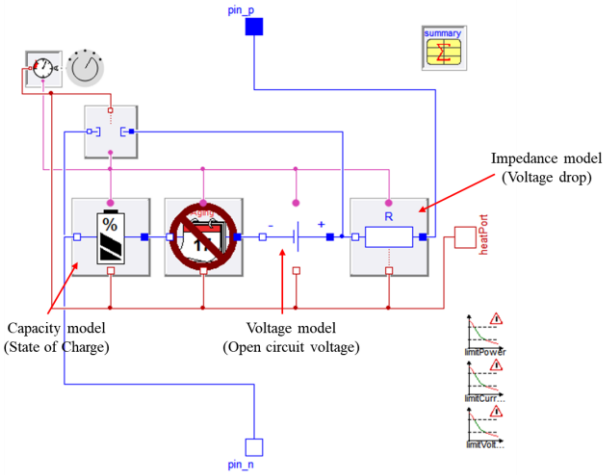


Figure 6. Battery model.

The controllers calculate the output of the motor and hydrogen fuel cell, respectively. The motor controller determines the required torque of the motor based on the accelerator pedal input, controlling the motor model accordingly. In the motor model, the output torque is determined based on the demanded torque and the torque map.

The hydrogen fuel cell system controller calculates the gross power based on the motor's required power. It then computes the BOP's power consumption according to the calculated gross power and transfers this to the load model within the hydrogen fuel cell, ensuring that the net output is supplied to the motor.

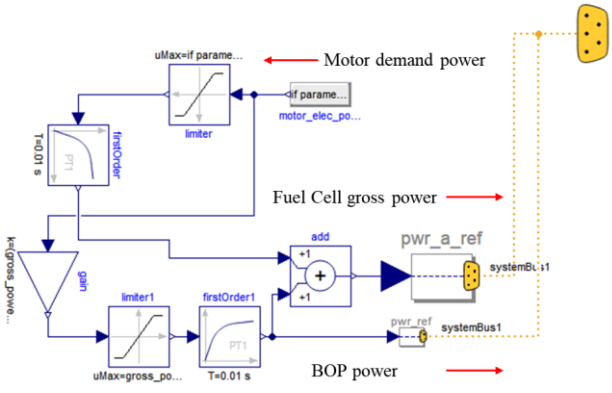


Figure 7. Fuel cell system controller

The hydrogen fuel cell system controller calculates the gross power based on the motor's required power. It then computes the BOP's power consumption according to the

calculated gross power and transfers this to the load model within the hydrogen fuel cell, ensuring that the net output is supplied to the motor.

Three types of electrified powertrain models were created based on the specifications and combinations of each component.

Table 3. Comparison of electrified powertrain configurations.

Specifications	*1*	*2*	*3*
System voltage	++	+++	+
Peak power	++	++	+
Peak torque	+	+	+
Continuous power	+	+	+
Continuous torque	+	++	++
Max speed of motor	++	+	+
Reduction Ratio	+	++	++
Efficiency of powertrain	+ (Const.)	++ (Map)	++ (Map)

3 Performance Anaysis

To evaluate whether each vehicle meets the required performance, a vehicle performance evaluation environment is modeled and configured according to the test conditions for assessing each requirement. Subsequently, the performance of each vehicle, composed of various subsystem combinations, is evaluated.

3.1 Test Environment Modeling

The vehicle performance evaluation environment consists of vehicle, driver, road surface, and atmosphere models. The driver model controls the vehicle model by providing acceleration/brake pedal inputs and steering inputs. The driver model is broadly classified into open-loop and closed-loop models.

The open-loop model delivers predefined acceleration and brake pedal inputs directly into the vehicle model without any feedback control. This approach is suitable for evaluating acceleration and top speed through full-throttle scenarios.

On the other hand, the closed-loop model controls the vehicle's speed by adjusting the acceleration and brake pedal inputs to follow a predefined speed profile. By comparing the vehicle's current speed with the speed defined in the profile, the inputs are dynamically adjusted. This method is suitable for evaluating fuel efficiency or energy consumption during specific speed profile driving.

The road surface model defines the road characteristics by setting the friction coefficient and the lateral/longitudinal slope of the surface. This makes it suitable for evaluating vehicle performance in scenarios such as driving on inclined roads.

3.2 Requirements

The vehicle performance evaluation criteria include five items: acceleration performance, maximum speed, gradeability, maximum grade speed, and driving range.

Acceleration performance is evaluated by measuring the time it takes for the vehicle to reach the target performance on flat terrain using the peak performance of the powertrain.

Maximum speed is determined by assessing the highest speed the vehicle can achieve on flat terrain based on the continuous performance of the powertrain.

Gradeability tests the vehicle's ability to start from a standstill and maintain a certain speed on steep slopes, utilizing the powertrain's peak performance.

Maximum grade speed evaluates the maximum speed the vehicle can achieve on a general incline using the continuous performance of the powertrain.

Lastly, driving range is estimated by analyzing hydrogen consumption during constant-speed driving on flat terrain, using the continuous performance of the powertrain to predict the total distance the vehicle can travel.

Table 4. Performance requirements and test conditions.

Performance	*Road*	*Motor*	*Velocity Control*
Acceleration	Flat	Peak	Full throttle
Max speed	Flat	Cont.	Full throttle
Gradeability	Very steep slope	Peak	Full throttle
Max gradient speed	Moderate slope	Cont.	Full throttle
Driving Range	Flat	Cont.	Controlled for contant speed

3.3 Results

The acceleration performance evaluation results showed that all four vehicles met the requirements. Vehicles 3 and 4 demonstrated the best acceleration performance, while Vehicle 2 had the lowest acceleration performance. This is attributed to the increased reduction ratio in Vehicles 3 and 4, which provided greater torque amplification for the same motor torque output, despite their heavier weight.

DOI
10.3384/ecp21767

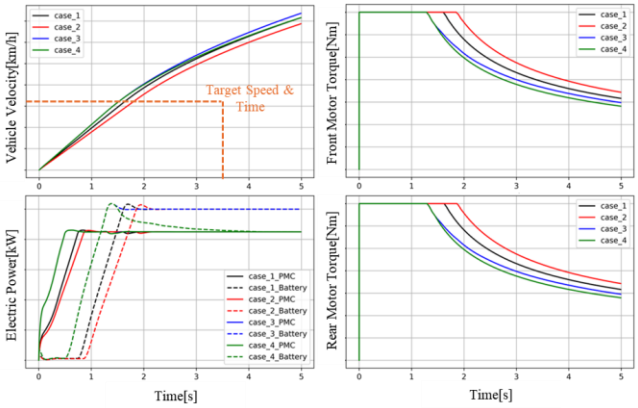


Figure 8. Result of acceleration test.

The maximum speed evaluation results showed that Vehicles 3 and 4 achieved the highest top speeds, while Vehicle 1 recorded the lowest. Although the continuous output of the powertrain was identical across all four vehicles, Vehicle 1's larger frontal area resulted in greater aerodynamic drag, leading to a lower maximum speed. In contrast, the improved aerodynamics and enhanced powertrain efficiency of Vehicles 3 and 4 contributed to their superior top speed performance.

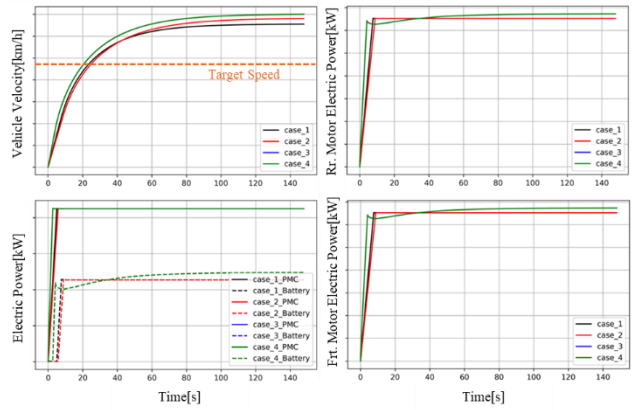


Figure 9. Result of max speed test.

The gradeability evaluation results showed that Vehicles 3 and 4 performed the best, while Vehicle 2 failed to meet the requirements. Compared to Vehicle 1, Vehicle 2 maintained the same powertrain performance but experienced increased gradient resistance due to its heavier weight. Vehicles 3 and 4 demonstrated significant improvements in gradeability, attributed to the increased reduction ratio, which amplified torque and enhanced their climbing performance.

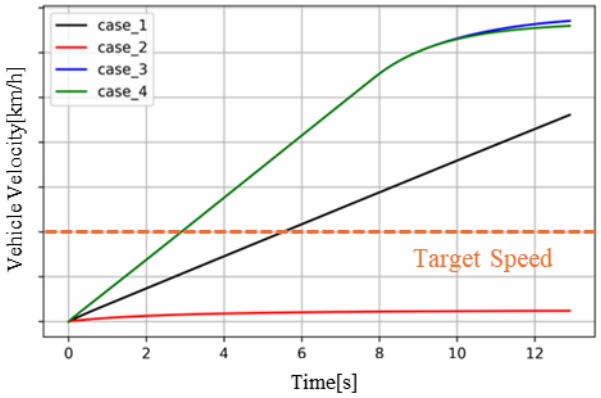


Figure 10. Result of gradeability test.

The maximum grade speed evaluation results indicated that Vehicle 2 had the lowest performance due to increased gradient resistance caused by its heavier weight. In contrast, Vehicles 3 and 4 demonstrated the best performance, attributed to the improved powertrain efficiency.

The driving range evaluation results revealed that Vehicles 3 and 4 achieved the best performance. This outcome is attributed to their superior aerodynamics and enhanced powertrain efficiency compared to the other two vehicles.

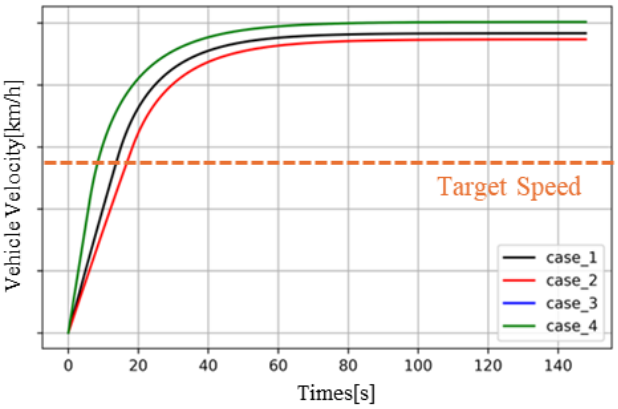


Figure 11. Result of max gradient speed test.

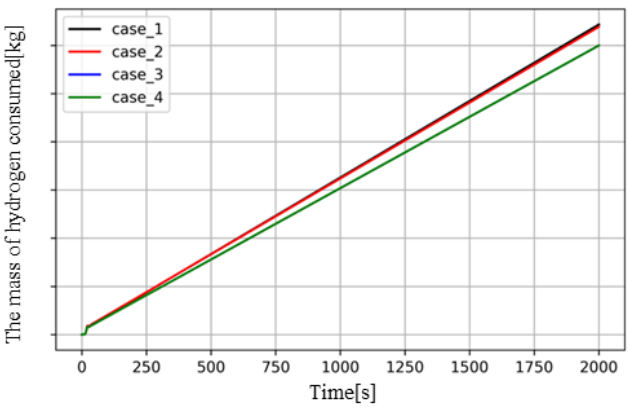


Figure 12. Result of driving range test.

Table 5. Summary of evaluation results.

Performance	*Vehicle (+: Passed / -: Failed)*			
	1	2	3	4
Acceleration	++	+	+++	+++
Max speed	+	++	+++	+++
Gradeability	+	-	++	++
Max gradient speed	++	+	+++	+++
Driving Range	+	+	++	++

4 Conclusion

In this study, modeling and simulation were conducted to predict the performance of FCEVs. Instead of using detailed specifications of hydrogen fuel cells, which are critical to information security, the study utilized results from component-level tests to develop a hydrogen fuel cell model. This approach provides a foundation for predicting FCEV performance and examining optimal component combinations during the early stages of design.

However, as the model does not fully reflect the physical and chemical characteristics of actual hydrogen fuel cells, further validation is required. Additionally, studies on the control of electrical flows between the hydrogen fuel cell and the battery in real FCEV systems are necessary.

Future research will focus on validating the model using test results from actual vehicles and conducting comparative validation with models incorporating detailed hydrogen fuel cell specifications and characteristics. Moreover, research will explore methods for integrating the control of electrical flows between the hydrogen fuel cell and the battery to enable more realistic performance predictions.

Acknowledgements

This work has been supported by KIA.

References

IP Defense Forum (2024). “South Korea aims to fuel military with hydrogen technology”. URL: https://ipdefenseforum.com/2024/07/south-korea-aims-to-fuel-military-with-hydrogen-technology/

International Energy Agency (2019). “The Future of Hydrogen”. URL: https://www.iea.org/reports/the-future-of-hydrogen

Shevchenko, N. (2020, December 21). An Introduction to Model-Based Systems Engineering (MBSE). Retrieved November 14, 2024, from https://doi.org/10.58012/d464-qf49.

Modelon AB (2021). *Vehicle Dynamics Library Version 4.0.*

Modelon AB (2021). *Electrification Library Version 1.7.*

Vehicle Health Monitoring for Driving Safety using Co-simulation between Dymola and Simulink

Yeongmin Yoo[1] Yong Ha Han[1] Dae-Un Sung[1] Kyung-Woo Lee[1]

[1]Hyundai Motor Company, Republic of Korea, {yym9514,yongha,dusung,caselee}@hyundai.com

Abstract

A vehicle dynamics model-based health monitoring process is presented to enhance driving safety. The vehicle model can simulate driving by reflecting degradation performance of suspension and tires. The model was developed using Dymola, and driving simulation was performed by integrating the lane keeping assistant system with the vehicle model using Simulink. The degradation behavior was monitored with *k*-nearest neighbor and Gaussian mixture model. The remaining useful life for vehicle components was predicted using Gaussian process regression. The proposed method predicts remaining useful life with a 95% confidence level for vehicle components to improve safety for driving.

Keywords: Vehicle Health Monitoring, Lane Keeping Assistant System, Prognostics and Health Management, Anomaly Detection, Remaining Useful Life

1 Introduction

As driving technology is gradually becoming automated, the functional safety of vehicle is becoming more important. The advanced driver assistance system (ADAS) assists the driver for convenient of driving. However, it is need for a technology to prevent risks caused by vehicle defects that may occur while driving.

Recently, prognostics and health management (PHM) technology for detecting defects in vehicle parts and predicting lifetime has been applied. PHM is an engineering approach that enables real-time health assessment of a system under its actual operating conditions, as well as the prediction of its future state based on up-to-date information, by incorporating various disciplines including sensing technologies, physics of failure, machine learning, modern statics, and reliability engineering. It enables engineers to turn data and health states into information that will improve our knowledge on the system and provide a strategy to maintain the system in its originally intended function. While PHM has roots from the aerospace industry, it is now explored in many applications including manufacturing, automotive, railway, energy and heavy industry (Kim et al., 2017; Sankararaman and Goebel, 2015).

In the automotive area, it is possible to provide vehicle state information to the driver and the maintenance company. This ensures vehicle maintenance efficiency and driving safety. It is necessary to obtain degradation data of components by monitoring the vehicle state. However, it takes a significant amount of time to obtain the degradation data.

This study proposes a vehicle dynamics model-based PHM process. The vehicle modeling, including degradation of suspension and tires, was performed using Dymola. The driving simulation was performed by integrating the lane keeping assistant system (LKAS) with the vehicle model using Simulink. The vehicle model was imported in the Simulink environment using functional mock-up interface (FMI). The LKAS is a control system that aids a driver in maintaining safe travel within a marked lane of a highway. Simulation data-based machine learning was used to determine normal/abnormal vehicle states and to assess the lifetime for vehicle components as shown in Figure 1.

The remainder of this paper is organized as follows. Section 2 explains the degradation modeling for vehicle components and the co-simulation process between Dymola and Simulink. Section 3 explains machine learning algorithms used for vehicle states and lifetime assessment. Finally, Section 4 concludes the study and discusses future study plans.

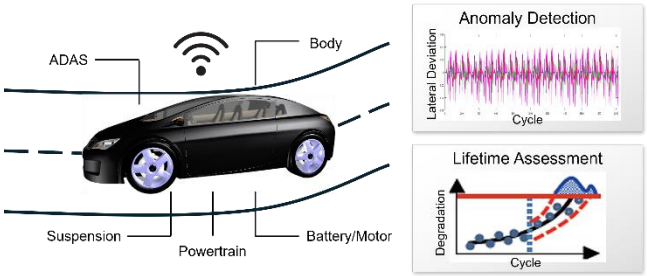


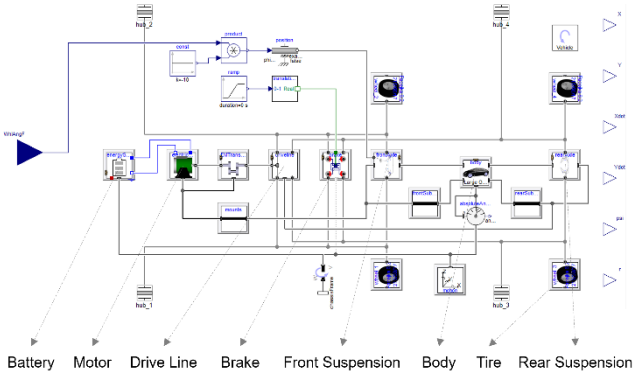
Figure 1. Virtual driving simulation-based vehicle health monitoring.

2 Virtual Driving Simulation

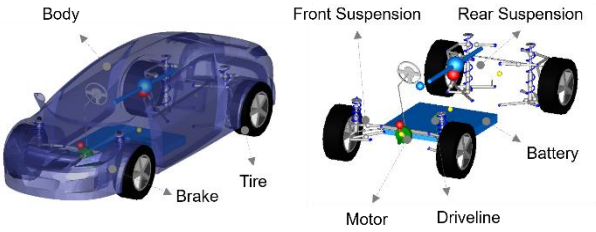
2.1 Vehicle Modeling

The vehicle model was developed using components from Claytex and vehicle systems modelling and analysis (VeSyMA) libraries in Dymola (Deuring et al., 2011; Yoo et al., 2018). The model was comprised of subsystems for vehicle body, suspension, driveline, electric motor, battery, brake, and tires as shown in Figure 2.

The front suspension was designed with a MacPherson strut suspension. The rear suspension was designed with an integral link suspension. The suspension blocks included shock absorber, rubber bush and stabilizer bar components. The tires were designed with 215/50R17 Pacejka model. The electric motor was designed with an AC induction motor on the front wheel, and the maximum torque of the motor was 350 Nm. The battery was designed with a 240 V voltage and 85.5 kWh capacity.



(a)

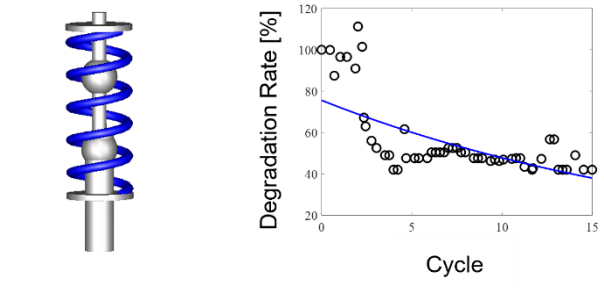


(b)

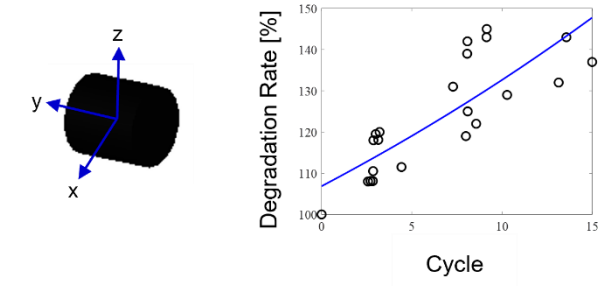
Figure 2. Vehicle model developed using components from Claytex and VeSyMA libraries: (a) graphics and (b) animation views.

2.2 Degradation Modeling

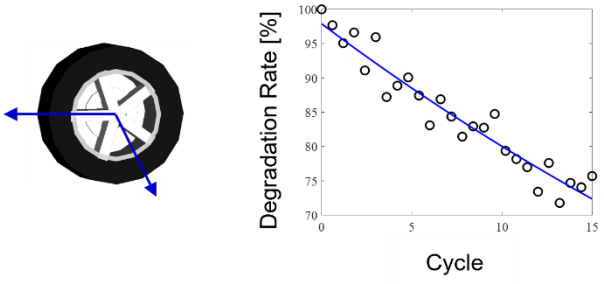
The shock absorber, rubber bush, and tire models were developed for degradation simulation of vehicle driving. The degradation rates for shock absorber damping, rubber bush stiffness, and tire friction coefficients were reflected in the models as shown in Figure 3. As the cycle increases, the degradation rates of the damping and friction coefficients decrease, but the stiffness increases.



(a)



(b)



(c)

Figure 3. Degradation models: (a) shock absorber damping, (b) rubber bush stiffness and (c) tire friction coefficients.

The degradation rate data were converted into exponential functions and reflected in the components as shown in Listing 1. D_damping is shock absorber damping. D_stiffness is rubber bush stiffness. D_friction is tire friction coefficient. Param1 and Param2 are coefficients of exponential function.

Listing 1. Degradation equations

```
equation
D_damping=Param2_damping*exp(Param1_damping*cycle) ;

equation
D_stiffness=Param2_stiffness*exp(Param1_stiffness*cycle) ;

equation
D_friction=Param2_friction*exp(Param1_friction*cycle) ;
```


DOI
10.3384/ecp21773

2.3 FMI-based Co-simulation

The vehicle model was converted to functional mock-up unit (FMU) to co-simulate with LKAS in Simulink as shown in Figure 4. The input ports of the FMU are steering angle and longitudinal velocity. The output ports are longitudinal position, lateral position, longitudinal velocity, lateral velocity, yaw angle, and yaw rate.

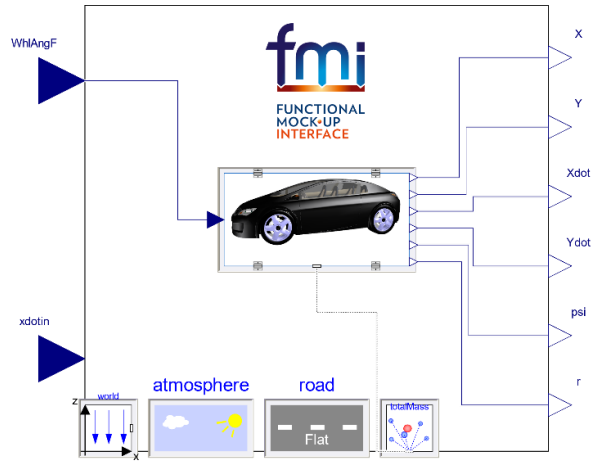
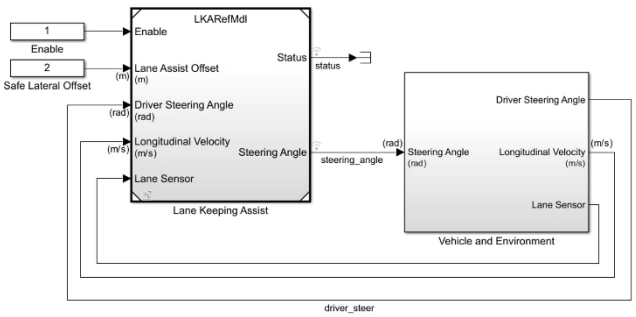
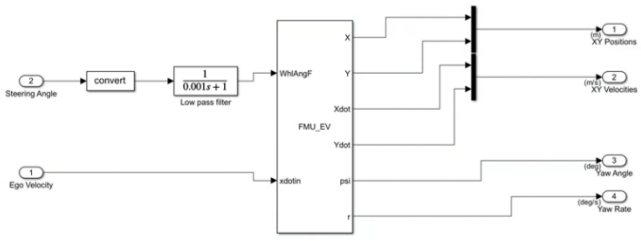


Figure 4. Vehicle model converted to FMU.

The LKAS (Lee et al., 2014) detects when the vehicle deviates from a lane and automatically adjusts the steering to restore proper travel inside the lane without additional input from the driver. The LKAS was constructed as shown in Figure 5.



(a)



(b)

Figure 5. LKAS in Simulink: (a) LKAS blocks and (b) vehicle dynamics block imported with FMU.

The lane keeping assist block contains lane center estimation, lane keeping controller, lane departure detection, and assist blocks. The lane center estimation block outputs the data from lane sensors to the lane keeping controller. The goal for the lane keeping controller block is to keep the vehicle in its lane and follow the curved road by controlling the front steering angle. The lane departure detection block outputs a signal that is true when the vehicle is too close to a detected lane. The assist block decides if the lane keeping controller or the driver takes control of the vehicle. The switch to assisted steering is initiated when a lane departure is detected.

The vehicle and environment block contains vehicle dynamics, scenario reader, vision detection generator, and driver blocks. The vehicle dynamics block includes the vehicle model converted to FMU. The input and output ports of the block are the same as the FMU. The scenario reader block generates the ideal left and right lane boundaries based on the position of the vehicle with respect to the scenario. The vision detection generator block takes the ideal lane boundaries from the scenario reader block. The driver block generates the driver steering angle based on the driver path.

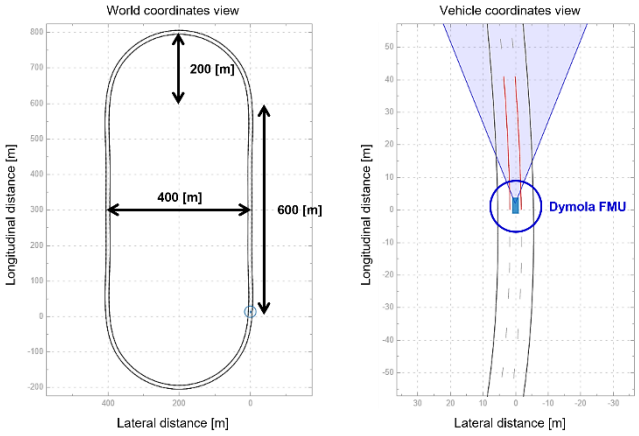


Figure 6. Driving road view.

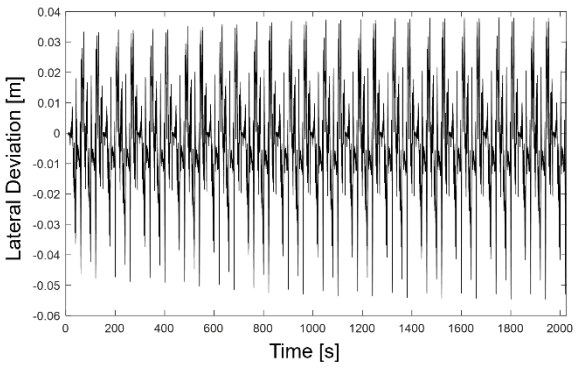


Figure 7. Lateral deviation between road centerline and vehicle for 15 laps of driving.

The driving road was designed with elliptical type as shown in Figure 6. The initial vehicle speed was 15 km/h. It was increased to 60 km/h within 15 seconds, after which it ran 15 laps while remaining constant. The lateral safety distance of the LKAS was set to 1 meter. Figure 7 shows the lateral deviation between road centerline and vehicle for 15 laps of driving. The smaller lateral deviation means better driving safety. However, the lateral deviation increased due to degradation of vehicle components.

3 Vehicle Health Monitoring

3.1 Feature Extraction

The feature extraction was performed to analyze the lateral deviation data. It refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. The eight features were extracted from the lateral deviation data as shown in Figure 8. The features were mean, max, root mean square, skewness, kurtosis, crest factor, impulse factor, and shape factor. It is important to find an effective factor that has consistent tendency to increase or decrease in proportion to vehicle mileage.

Principal component analysis (PCA) was performed for eight features as shown in Figure 9. It is a dimensionality reduction method used to simplify a large data set into a smaller set while still maintaining significant patterns and trends. The data is linearly transformed onto a new coordinate system such that the directions capturing the largest variation in the data can be easily identified. The first principal component is the direction in space along which the data points have the highest or most variance. The larger the variability captured in the first component, the larger the information retained from the original dataset. The second principal component accounts for the next highest variance in the dataset and must be uncorrelated with first principal component. The third principal component is the same way.

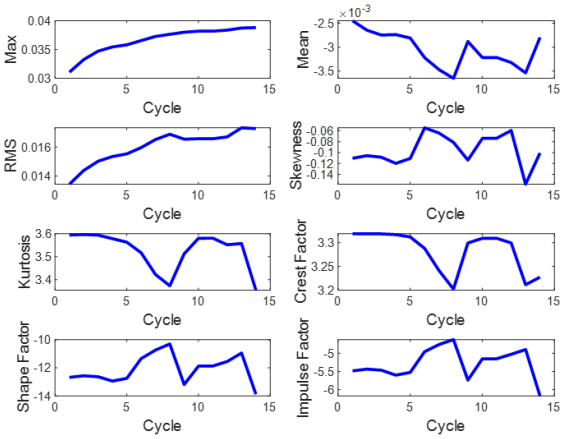


Figure 8. The eight features of lateral deviation data.

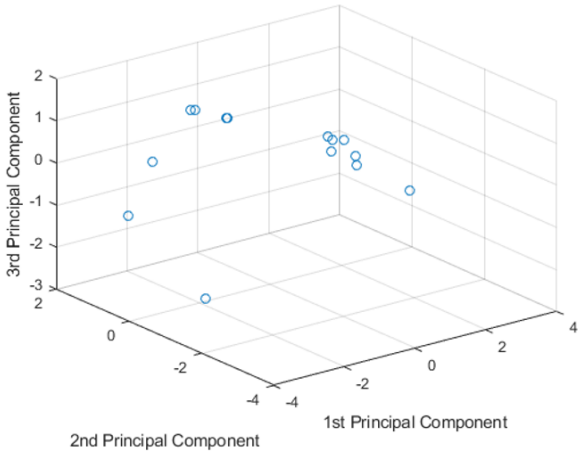
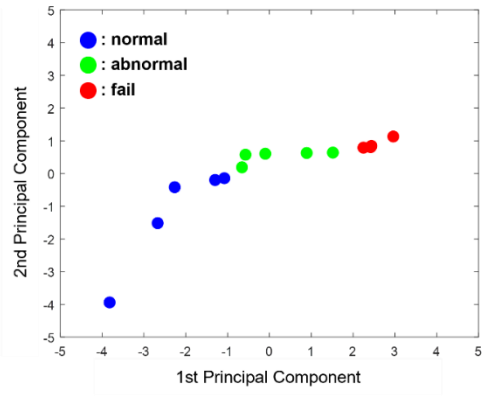


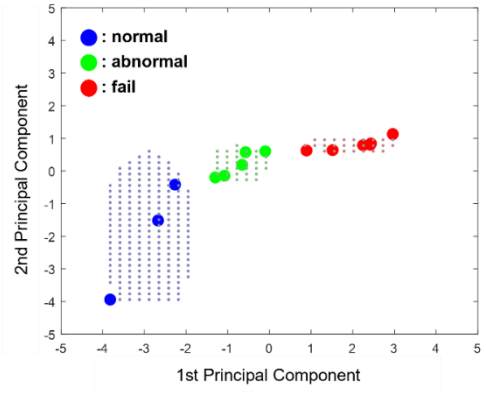
Figure 9. PCA for eight features.

3.2 Anomaly Detection

The lateral deviation data were classified into normal, abnormal, and failure states using machine learning algorithms as shown in Figure 10.



(a)



(b)

Figure 10. Classifiers of normal and abnormal: (a) k-NN and (b) GMM.

DOI
10.3384/ecp21773

The *k*-nearest neighbor (*k*-NN) is a non-parametric classification method (Cunningham and Delany, 2021). It tries to classify an unknown sample based on the known classification of its neighbors. If the classification of a sample is unknown, then it could be predicted by considering the classification of its nearest neighbor samples. Given an unknown sample and a training set, all the distances between the unknown sample and all the samples in the training set can be computed. The distance with the smallest value corresponds to the sample in the training set closest to the unknown sample. Therefore, the unknown sample may be classified based on the classification of this nearest neighbor. The *k* was set to three for the classification of normal, abnormal, and failure for driving performance.

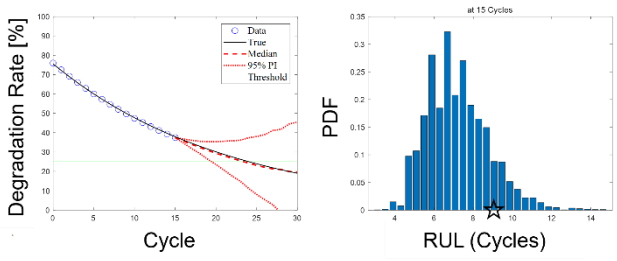
The Gaussian mixture model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities (Reynolds, 2009). The GMM parameters are estimated from training data using the maximum a posteriori estimation from a well-trained prior model.

3.3 Lifetime Assessment for Components

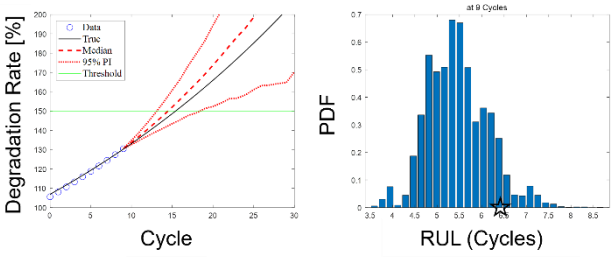
The lifetime of shock absorber, rubber bush, and tire was assessed using Gaussian process (GP) regression. The GP regression (Schulz et al., 2018) is one of regression-based methods used for data-driven prognostics, which is a linear regression like the least squares method. The difference between GP and ordinary linear regression is whether the correlation in errors between a regression function and data are considered or not. It is assumed that errors are independent and identically distributed in the ordinary linear regression, while they are assumed to be correlated in GP.

As the results, the degradation and remaining useful life (RUL) are predicted as shown in Figure 11. The data point means the average value of degradation datasets, and the true line (i.e., solid line) is a fitting curve for actual degradation data. The median line and the 95% prediction interval (PI) line (i.e., dotted line) are the probabilistic results predicted using the GP regression.

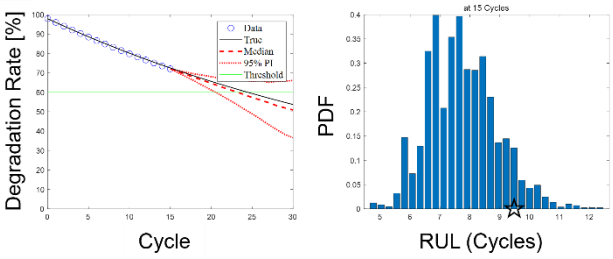
Table 1 shows the percentiles of RUL distribution. It can be confirmed that the 95% PI of the predicted RUL distribution of each degradation component was satisfied with respect to the actual RUL value.



(a)



(b)



(c)

Figure 11. GP prediction results: (a) shock absorber, (b) rubber bush, and (c) tire.

Table 1. Percentiles of RUL distribution.

Component	True	Estimation	95% PI
Damper	9.04	6.97	4.76~10.69
Bush	6.42	5.41	4.33~7.03
Tire	9.48	7.67	5.77~10.26

4 Conclusions

The health monitoring process using virtual driving simulation based on a vehicle dynamics model was proposed. It takes a significant amount of time to obtain the actual degradation data. To solve this problem, degradation data was obtained using co-simulation between Dymola and Simulink, and anomaly detection and lifetime assessment based on machine learning algorithms were performed. The main results are as follows:

- The virtual driving simulation involving degradation of vehicle components was constructed using Dymola and Simulink.
- The degradation behavior was monitored with *k*-NN and GMM.
- The GP regression predicted RUL with a 95% confidence level for vehicle components to improve safety for driving.

The future study plan is to construct vehicle health monitoring system using virtual driving simulation for battery degradation.

References

Kim N. H., An D., and Choi J. H. (2017). "Prognostics and health management of engineering systems". Switzerland: Springer International Publishing.

Sankararaman S., and Goebel K. (2015). "Uncertainty in prognostics and systems health management". *International Journal of Prognostics and Health Management* 6, pp. 1-14.

Deuring A., Gerl J., and Wilhelm H. (2011). "Multi-domain vehicle dynamics simulation in Dymola". *Proceedings of the 8th International Modelica Conference* 63, pp. 13-17.

Yoo Y., Lee S., Yoon J., and Lee J. (2018). "Modelica-based dynamic analysis and design of lift-generating disk-type wind blade using computational fluid dynamics and wind tunnel test data". *Mechatronics* 55, pp. 1-12.

Lee J., Choi J., Yi K., Shin M., and Ko B. (2014). "Lane-keeping assistance control algorithm using differential braking to prevent unintended lane departures". *Control Engineering Practice* 23, pp. 1-13.

Cunningham P., and Delany S. J. (2021). "K-nearest neighbour classifiers-a tutorial". *ACM computing surveys* 54, pp. 1-25.

Reynolds D. A. (2009). "Gaussian mixture models". *Encyclopedia of Biometrics* 741, pp. 659-663.

Schulz E., Speekenbrink M., and Krause A. (2018). "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions". *Journal of mathematical psychology* 85, pp. 1-16.

DOI
10.3384/ecp21773

Study on nuclear and renewable hybrid energy system performance prediction by using Modelica

Hae-Ryong Hwang[1*], Daeoh Kang [2], Kagsu Jang[3], Jiyeon Kang[3]
[1]*ISMR, 210, Dongho-ro, Jung-gu, Seoul, Republic of Korea, 04601*
[2]*iVH, 19, Yangjaecheon-ro 17-gil, Seocho-gu, Seoul, Republic of Korea, 06754*
[3]*KEPCO E&C, 269 Hyeoksin-ro, Gimcheon-si, Gyeongsangbuk-do, Republic of Korea, 39660*
[*]*Corresponding author: harold.hwang@ismr.co.kr*

1. Introduction

The production and use of electric energy is constantly evolving in the building, industrial and transportation areas [1,2].

Looking at the electric energy production sector, the market share of variable renewable energy such as wind and solar power (PV) is continuously increasing.

Among the use sectors, the transportation area is encouraged to use eco-friendly fuels, and accordingly, the use rate of battery and hydrogen electric fuel vehicles is rapidly increasing. In the building area, energy efficiency is increasing due to building energy management system (BEMS) and high-insulation materials.

Since 2010, researches in nuclear-renewable hybrid energy system (NRHES) have been actively conducted to maximize efficiency in electric energy production and use.

The NR HES, which combines production and use, is a system that actively controls electricity production and use and surplus energy utilization [3].

In this study performance analysis is performed for the architecture depicted in the former study[3] by digital twin for securing operation efficiency of NRHES.

Digital twin is built based on multi-physics system using Modelica. All components were modeled to consider electricity, heat, mechanics and environment at the same time[4,5].

2. Methods and Results

This study was proceeded in two stages for the evaluation of the efficiencies of three proposed architectures of NRHES. The first step is to build an NRHES digital twin using Modelica. Renewable energy consists of solar and wind power. Nuclear power consists of a steam generator and a turbine generator. Energy is stored in battery energy storage system(ESS) and Thermal storage. The industrial process produces hydrogen. Only low temperature water electrolysis (LTE) was considered in this study.

The second step is to define scenarios for each energy storage method and predict the optimal architecture through efficiency analysis. There are three scenarios used. The first is to use only battery ESS, and the second is to use the battery ESS and the low-temperature water electrolysis (LTE) at the same time. Finally, battery ESS, LTE, and thermal storage are added all together.

2.1. Digital twin construction of NRHES system

As shown in Figure 2.1 and Table 2.1, NRHES is composed of renewable energy block, nuclear energy block, energy storage block, industrial process block, demand load block and controller. The detailed modeling method for each block is as in the subsection. All module m

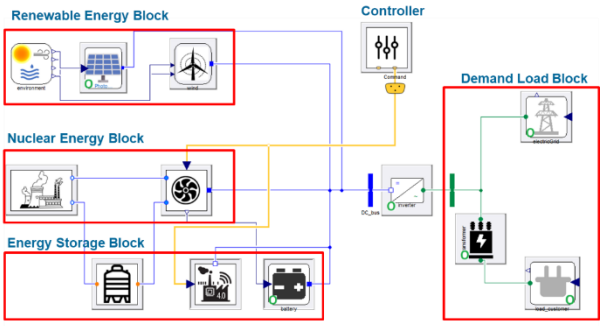


Fig 2.1. NRHES architecture

Table 2.1 NRHES sub-module list

Part Name	Content
Nuclear steam generator	Steam production model with heat from reactor
PV panel	Solar Photon-powered models
Wind turbine	Power generation model with wind velocity as input
ESS	ESS Model using battery
IP (electrolyzer)	Hydrogen model in LTE method
Thermal storage	Heat storage model with salt water as medium

2.1.1. Nuclear Steam Generator

The steam generator is a key component for the operation and the safety of the plant because it is responsible for the generating steam and cooling of the reactor.

Water and steam flow is as follows. The primary water flows into U-tubes and yields its heat to the secondary water. The secondary water, circulating outside the U-tubes, is liquid at the inlet of the steam generator, then flows down the outer part of the steam generator and starts to boil when reaching the bottom center part of the steam generator, until the top of the boiling section.

There, the ratio between the total flow rate and steam flow rate (circulation rate) reaches a value of 4 to 5 at nominal power.

This part of the steam generator is called the riser, where the flow is mainly two-phase (a mixture of water and steam). Moreover, due to the non-homogeneity of heat exchange inside the riser, two regions must be considered. When secondary water is flowing outside the first half part of U-tubes with hot primary water flowing in, the region is called “hot leg”. When flowing outside the other half part of U-tubes with cooler primary water flowing in, the regions is called “cold leg”.

The water and steam mixture passes then through separators where the two phases are separated in the upper part of the steam generator. The liquid part goes back to the steam generator feedwater, and the vapor part goes to the turbine. This part of the steam generator is called the dome.

The thermal load on the riser tube increases at t=500. This increases the pressure initially before the pressure controller increases the opening of the high-pressure steam valve to control the pressure back to the set point, increasing the steam flow and increasing the production power.

The current model produces 275 ℃ of steam from nuclear energy block. The main steam is produced by the reactor and controlled by the auxiliary heat source, using default parameters for pumps and capacitors.

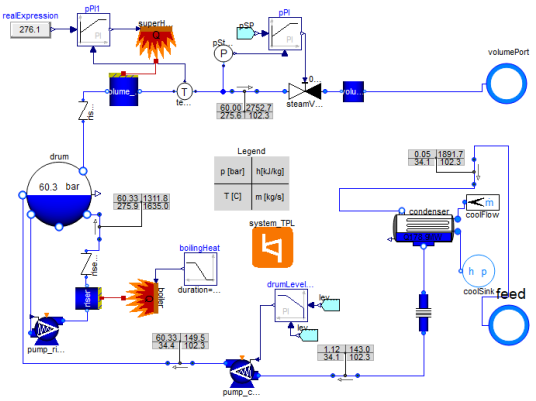


Fig 2.2 Close loop steam cycle

As a result of model verification, the nuclear steam generator produces 60.8 MW of electric power, which predicts approximate results with the specifications of 60.0 MW. A comparison of the physical quantities at key points is shown in Table 2.2.

Table 2.2 Comparison of steam cycle state with reference data

Location	Variable	Model	Reference	Error
Turbine inlet	Temperature	275.6	276.0	0%
	Pressure	60.0	60.0	0%
	Enthalpy	2752.0	2787.0	1%
	Mass flow rate	102.3	102.7	0%
Turbine outlet	Temperature	34.1	43.0	21%
	Pressure	0.057	0.086	34%
	Enthalpy	1892.0	1838.0	-3%
	Mass flow rate	102.3	74.5	-37%

2.1.2. Wind turbine

The wind turbine components take wind speed data as input and carry the power generated based on the given parameters.

The main model parameters are summarized as follows:

1) Environmental factors: Typical values of air density, reference height and friction coefficients for which wind speed is measured, and friction coefficients for different areas are shown in Figure 2.3.

2) Shape of wind turbine: hub height and blade length L (Own area A is calculated based on blade length). Rated power capacity, the rated wind speed is calculated from the given correlation.

3) Cut-in and cut-out wind speeds: represent the operating conditions of the turbine.

4) The total power is calculated to reflect efficiency.

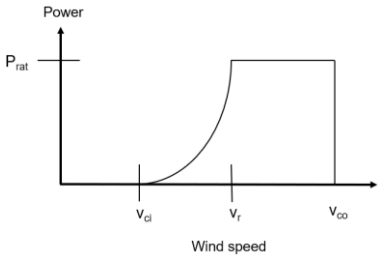


Fig 2.3. Turbine model

Between the cut-in and rated speeds, the power is calculated as follows.

$$P_{calc} = P_{rat} \cdot \frac{(v^3 - v_{ci}^3)}{v_{rat}^3 - v_{ci}^3} \quad (1)$$

$$P_{wind} = P_{calc} \cdot \eta \quad (2)$$

As a result of the model verification, the power output per hour of the prediction and reference was found very similar.

DOI
10.3384/ecp21779

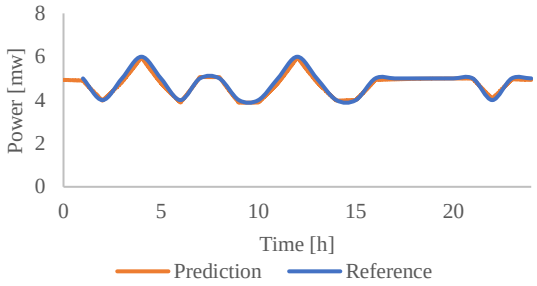


Fig 2.4. wind power correlation results

2.1.3. Photovoltaics Power

The solar module and array generate DC power and are next connected to DC-AC inverter. Inverters typically contain control logic to regulate the DC voltage of the photovoltaic module. This is typically a Maximum Power Point Tracker (MPPT).

Solar modules do not generate the same amount of power for a given environmental boundary condition (mostly illuminance, temperature, and air mass). In fact, the power depends on the DC voltage of the module pin. This is easily understood by considering the 0V voltage drop between the pins. Whatever the current is, the power becomes zero. However, in grid applications, power sources such as solar modules must provide optimal power. Therefore, the DC voltage should be set to a value that maximizes the power of the module.

This model allows you to extract the maximum power point analytically from the equation. This prevents the inverter DC voltage from being set using an external controller (tracer), which is typically computationally expensive.

When MPPT is enabled, the current on the pin corresponding to the module operating at the maximum power point is set.

When MPPT (default) is disabled, the actual current corresponding to the voltage set on the pin is provided. This still allows you to analytically extract the maximum power point voltage and output it via real output, but a feedback loop (mostly via an inverter) is required to set the voltage at this value.

Power output depends on solar radiation and the efficiency of the solar components. The mitigation factor eta is defined as a parameter that depends on PV component aging, wiring loss, and dust cover.

$$P(t) = P_{rat} \cdot \eta_{PV} \cdot G(t)[1 + \alpha_{PV} \cdot (T_{PV} - 298.15)] \quad (3)$$

$$, T_{PV} = \frac{T_{ambient} + A \cdot \Gamma}{1 + B \cdot \Gamma}$$

$$, A = 1 - \frac{\eta_{STC} \cdot (1 - 25\alpha_{TP})}{0.9}$$

$$, B = \frac{\alpha_{TP} \cdot \eta_{STC}}{0.9}$$

$$, \Gamma = \frac{(T_{NOCT} - 293.15)G(t)}{0.8}$$

Based on the results of the reference, the model verification confirmed that the power output per hour was more than 98% consistent.

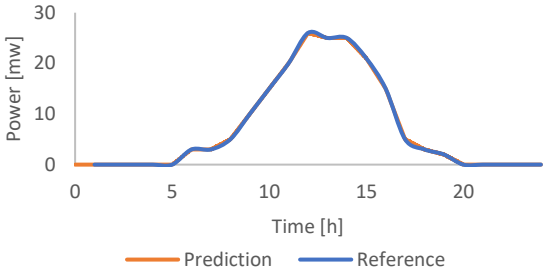


Fig 2.5. PV panel correlation results

2.1.4 Industrial Process(IP)

This is an alkaline electrolyte model for use in the microgrid optimization framework. This model has the following connectors:

- Fluid ports for hydrogen flow generated in electrolytic cells.
- Current control input. This input determines the current that the stack uses to produce hydrogen.
- Electrolytic cell pin - Connect the electrolytic cell stack to the electrical grid. This model assumes an ideal DC/DC converter that converts electrolytic cell voltage to system voltage.
- Electrolytic pin - similar to the electrolytic pin, but instead uses the same principle to connect the compressor. The ideal DC/DC converter between the compressor and the system voltage level.

The model used in this task used the following assumptions.

- The electrolysis process assumes a constant operating point (constant pressure and temperature). The cooling requirements required to maintain the electrolytic cell at a constant temperature are calculated and monitored by the model variable Qdot_cool.
- The compressor operation required for the calculation assumes ideal isothermal compression and constant pressure on the electrolytic cell side (inlet).

LTE model is Semi-Empirical model. The main equation is shown in equations below and is calculated between the current density and the production of electrolysis hydrogen.

$Cell\ Voltage$:

$$V_{cell} = V_{rev} + r \cdot J + s \cdot \log_{10}(t \cdot J + 1) \quad (4)$$

$Hydrogen\ flow\ rate$:

$$\dot{n}_{H2} = \eta_f \cdot n_{cell} \cdot \frac{I}{z \cdot F} \quad (5)$$

The Faraday efficiency:

$$\eta_f = f_2 \cdot \left(\frac{J^2}{f_1 + J^2}\right) \quad (6)$$

Since there is no verification data of LTE model, only the results produced by the model were analyzed.

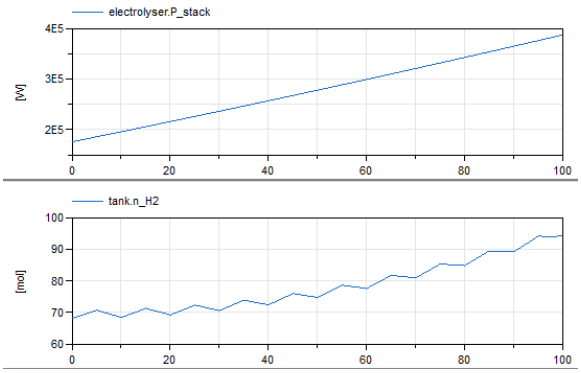


Fig 2.6. Electrolyzer Performance for hydrogen production

2.1.5 Thermal Storage

The thermal storage comprises an indirect thermal energy storage system consisting of a valve, a pump and two tanks with a heat exchanger and the molten salt. Figure 2.7 shows both energy charging and discharging where the heat transfer fluid exchanges energy with the storage system.

The initial temperature of the thermal storage is 240 ℃. Heat can be stored up to 290 ℃ through heat exchange.

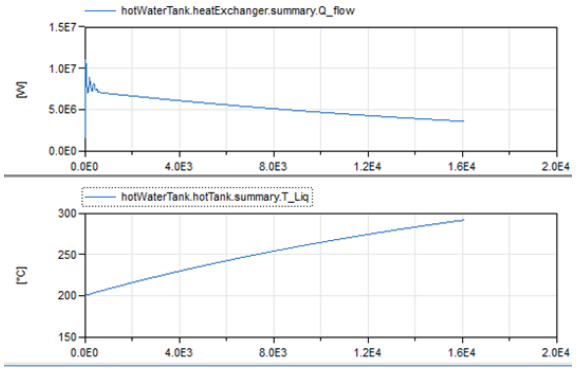


Fig 2.7. Thermal storage sample results

2.1.6. NRHES model Validation

Figure 2.8 is verification system based on Kim [3]. It consists of steam generator, load, battery ESS, etc. Digital twin with identical structure was configured in this task

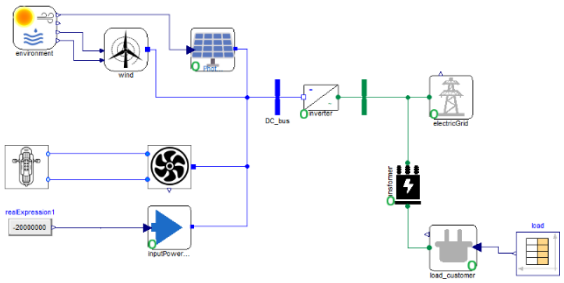


Fig 2.8. NRHES architecture digital twin model

As a result of the model verification, it was confirmed that the reference result and the prediction result provided were within 1% of the error rate in both operation periods. This demonstrates the significance of the digital twin model proposed in this study.

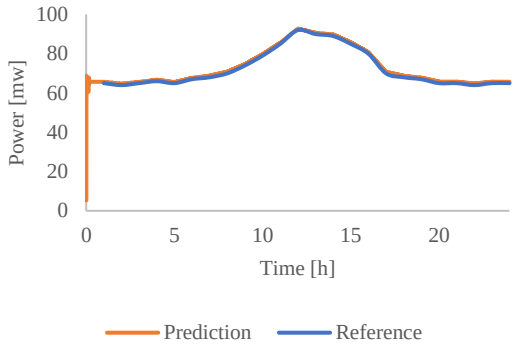


Fig 2.9. Total generation power

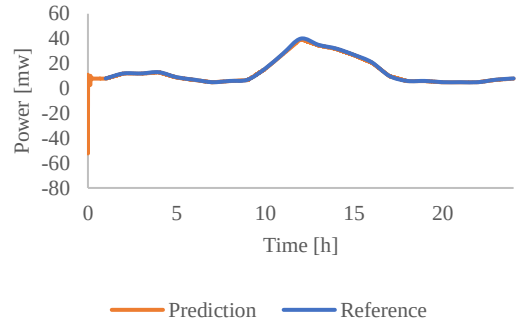


Fig 2.10. Surplus power

2.2. Performance Analysis by NRHES Scenario

Based on battery ESS, thermal storage, and LTE, we analyzed the power production patterns of three NRHES scenarios. Each scenario is expected to help develop an understanding and utilization strategy for NRHES.

2.2.1. Scenario 01: Add ESS Only

Model

Only battery ESS has been added to the reference

DOI
10.3384/ecp21779

model in Chapter 2 with battery ESS Controller. The controller determines the battery charge by monitoring the renewable energy output, surplus power, etc. in real time.

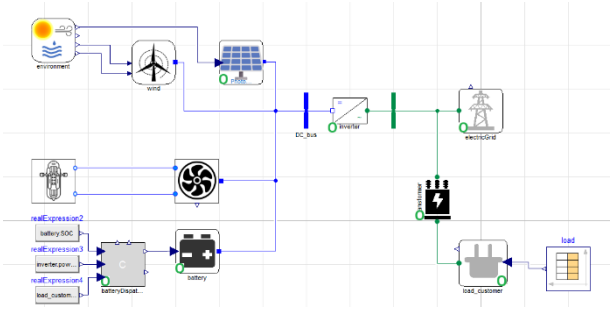


Fig 2.11. Scenario 01 Model

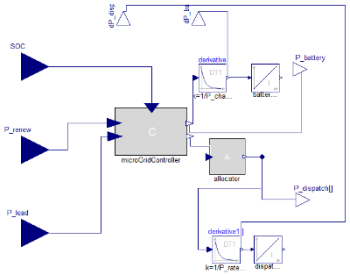


Fig 2.12. ESS Controller Model

Results

Simulation results show a 22% reduction in excess power when using battery ESS.
In the absence of energy storage, the excess power is 34.8 MW root-mean-square(rms). When the battery is added, the excess power is 27.3 MWrms.

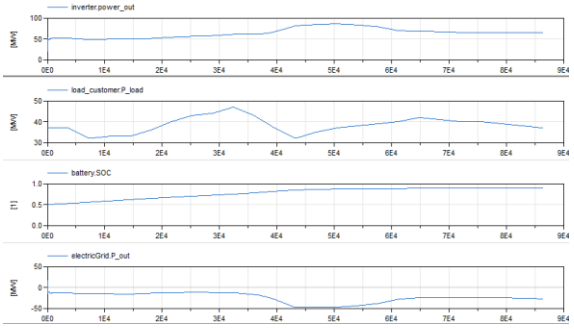


Fig 2.13. Scenario-01 Results

2.2.2 Scenario 02: Scenario 01 + IP added

Model

IP was added to Scenario 01 model. Electrolyzer and controller are also added. The controller determines hydrogen production by monitoring renewable energy production, surplus power, etc. in real time.

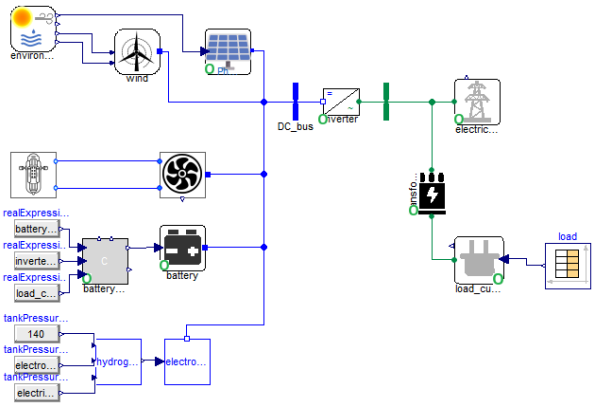


Fig 2.14. Scenario 02 model

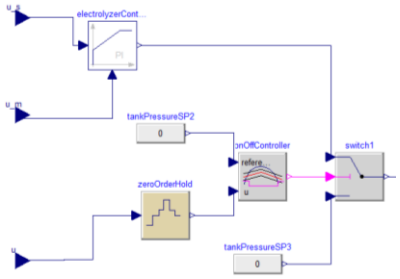


Fig 2.15. Electrolyzer controller

results

Simulation results show a 70% reduction in excess power when using IP.
When IP is added, the excess power is 10.64 MWrms.

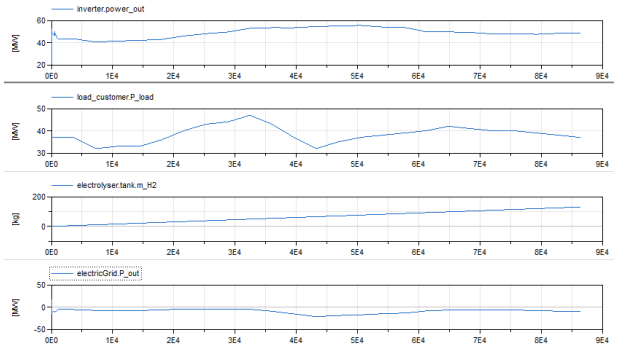


Fig 2.16. Scenario 02 Results

2.2.3 Scenario 03: Scenario 02 + Thermal Storage added

Model

Thermal storage has been added to the Scenario 02 model. In addition, the controller has been modeled. The controller determines the amount of heat exchange by monitoring the renewable energy output, surplus power, etc. in real time.

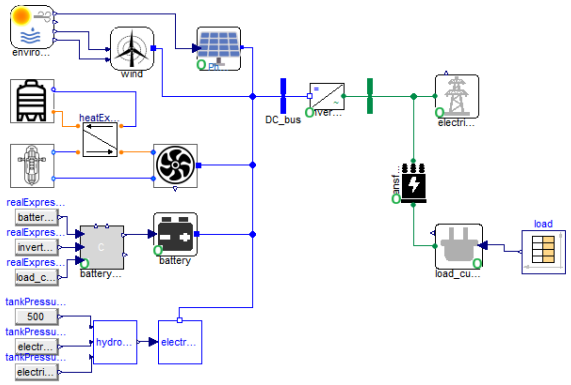


Fig 2.17. Scenario 03 model

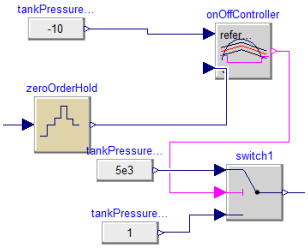


Fig 2.18. Scenario 03 Controller

Results

Simulation results show a 75% reduction in excess power when using IP.
When IP is added, the excess power is 9.6 MWrms.

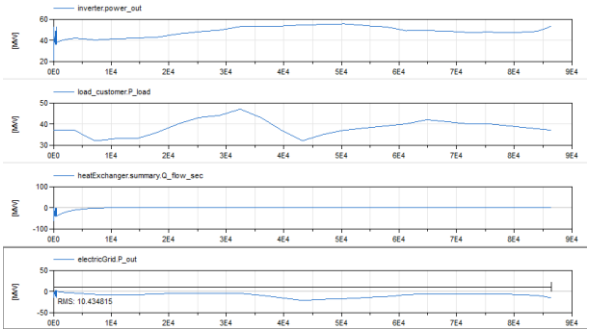


Fig 2.19 Scenario 03 results

3. Conclusions

This paper conducted a study to analyze the performance of NRHES architectures by different scenarios. Digital twin is built based on a multi-physical integrated model using Modelica. The architecture identified three methods according to the strategy for using surplus energy, and the analysis results are as follows.

NRHES was built using the digital twin technology, and research was conducted on the efficiency of each operation strategies.

- Scenario 1: A model that adds battery ESS to the reference model. The battery capacity was set at 20 MWh. As a result of analyzing the efficiency of using surplus power, it was confirmed that 30% was improved compared to the reference model.

- Scenario 2: model that adds low-temperature water electrolysis to the scenario 1 model. The daily hydrogen production was determined to be 130 kg. As a result of analyzing the efficiency of using surplus power, it was confirmed that it was improved by 70% compared to the reference model.

Scenario 3: Model that adds a heat storage device to the Scenario 2. As a result of analyzing the efficiency of using surplus power, it was confirmed that 75% was improved compared to the reference model.

If the cost model is added in the future, it will be possible to study the economics based NRHES architecture and optimal operation scenario.

REFERENCES

[1] Mark Ruth, Dylan Cutler, Francisco Flores-Espino, and Greg Stark, The Economic Potential of Nuclear-Renewable Hybrid Energy Systems Producing Hydrogen, NREL/TP-6A50-66764, April 2017

[2] Kim, Jong Suk, Bragg-Sitton, Shannon, Boardman, Richard, Status on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis Plant & Gas Turbine Power Plant, DO: 10.2172/1333156

[3] Sung Ho Kim*, Chang Kyu Chung, Hee Hwan Han, Byung Jin Lee, Development of Nuclear-Renewable Hybrid Energy System using Thermal Energy Storage for Industrial Processes, Transactions of the Korean Nuclear Society Autumn Meeting Goyang, Korea, October 24-25, 2019

[4] Roberto Ponciroli, Yu Tang, and Richard B. Vilim , Characterization of Flexible Operation Performance of N-R HES Components in Support of a Model-Based Pre-conditioner, Philip Eberhart et al., ANL/NE-18/6

[5] Ruediger Franke and Hansjuerg Wiesmann, Flexible Modeling of Electrical Power Systems – the Modelica PowerSystems Library, Proceedings of the 10th International Modelica Conference, Lund, Sweden, March 10-12, 2014

DOI
10.3384/ecp21779

Testing Large Scale System Simulation using Linear Implicit Equilibrium Dynamics

Dirk Zimmer

Institute of System Dynamics and Control,
German Aerospace Center (DLR), dirk.zimmer@dlr.de

Abstract

The concept of flattening where all model equations are collected in a single set is deeply hardwired into the Modelica language. While flattening enables effective symbolic manipulation such as the reduction of systems with a higher-index, it also imposes limitations. Two of these limitations are that the code generation for very large systems may not scale very well and that a statement on the regularity of the system often cannot be made before the flattening took place. Whereas it is difficult to overcome these limits in the general case, there is a sub-class within Modelica models that is comparatively easy to precompile while still enabling the modeling of complex systems. This paper explores this path and presents first experiments on the scalability for larger systems.

Keywords: Compilation, Modelling Methodology, Large Scale Systems

1 Introduction

1.1 Motivation

This work has its origins in the quest for robust modeling methodologies. The goal of such a methodology is to enable a sufficient statement for object-oriented modeling:

Any valid combination of components (under rules of limited complexity) shall have a solution representing a physical system.

In practice that means, that robust components will lead to robust overall system simulation. One particular solution for this task is the class of Linear Implicit Equilibrium Dynamics or short: LIED.

Because solvability can be guaranteed upfront, also the compilation of this class of models can be simplified, even to such a degree that the pre-compilation of components can be enabled, making it attractive for large system simulation. However, before explaining LIED we shall revisit the current approaches for precompiling or simulating larger systems in Modelica.

1.2 State of the Art

The problems that occur with the compilation of very large Modelica models has been noted more than 15 years ago (Zimmer2009) with suggestions how to mitigate this issue with a post flattening analysis. While this never came to any fruitful results, multiple other approaches have been conceived for this problem. Most notably a full test suite has been provided (Casella2015) to help the simulation of large system in OpenModelica (Braun2017). This work is now supported by a new research compiler MARCO (Agosta2023).

One (albeit limited) approach is to find a particular solution for large arrays or vectors. Per default, these elements will all be scalarized, leading to a large number of variables. It would be very helpful keeping them as one entity.

A specific solution for this approach can be found for instance in (Neumayr2023). For a general Modelica compiler, it is however not always clear which vector or matrices can avoid scalarization so that casualization still can take place. This has been investigated by (Abdelhak2023) and is also experimented on in MARCO.

Other attempts originate from a different perspective. The pre-analysis or even pre-compilation of components is also useful for variable-structure or multi-modal systems (Benveniste2019). One elaborate proposal in this direction for the general case stems from (Benveniste2023).

Pre-compilation is of course not only useful for multi-modal systems but (as shown in this paper as well) is a promising technique for large scale systems as well. Indeed, the paper (Benveniste2023) provides a scalarization example of the casualization algorithm.

One common challenge of all of these approaches is that they add complexity to existing Modelica compilers and a full Modelica compiler is already very complex. Some of the approaches hence switch to other experimental modeling languages.

Also, it shall be noted that the support for large models has meanwhile been improved in existing tools. An order of magnitude could be gained in the last decade for certain commercial tools, seemingly by improving the quality of implementation and the availability of more memory.

This paper now focusses not on a general solution but on a special class of models. Let us hence study how this class is being defined.

2 Linear Implicit Equilibrium Dynamics

Linear Implicit Equilibrium Dynamics (LIED) is technically defined as a special class of Differential Algebraic Equation (DAE) Systems.

2.1 Formal Definition

A DAE system with potential state derivatives $\dot{\boldsymbol{x}}$, time t and algebraic variables $\mathbf{w}$

$$\mathbf{0} = \boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \mathbf{w}, t)$$

is defined as LIED system when it can be transformed into the following form:

$$\begin{bmatrix} \mathbf{w}_E \\ \dot{\boldsymbol{x}}_E \end{bmatrix} = \mathrm{g}(\boldsymbol{x}_I, \boldsymbol{x}_E, t)$$

$$\mathbf{A}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E) \begin{bmatrix} \mathbf{w}_I \\ \dot{\boldsymbol{x}}_I \end{bmatrix} = \mathrm{f}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E, t)$$

We see that both the algebraic variables as well as the state derivatives can be split into a fully explicit part ($\dot{\boldsymbol{x}}_E$; $\mathbf{w}_E$) and a part ($\dot{\boldsymbol{x}}_I$; $\mathbf{w}_I$) with a linear system in implicit form expressed by the regular matrix $\mathbf{A}$. Furthermore, the following conditions shall hold true:

$$\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \subseteq \dot{\boldsymbol{x}}$$
$$\mathbf{w}_E \cap \mathbf{w}_I \supseteq \mathbf{w}$$
$$\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \cap \mathbf{w}_I \supseteq \dot{\boldsymbol{x}}$$
$$\dot{\boldsymbol{x}}_E, \dot{\boldsymbol{x}}_I, \mathbf{w}_E, \mathbf{w}_I \text{ are all disjoint}$$

These conditions essentially mean that it is allowed to perform certain symbolic mechanism of index reduction such as the dummy derivative method (Mattsson1993) originating from (Pantelides1988). Using this method, states variables of $\boldsymbol{x}$ can be transformed to algebraic variables in $\mathbf{w}_I$ and further derivatives may be added to $\mathbf{w}_I$ or $\mathbf{w}_E$. In practice, this is important because it means that the linear implicit dynamics can be expressed by far fewer states than suggested by the vector $\boldsymbol{x}$ of the original DAE formulation.

2.2 Informal Explanation

The formal definition above may be primarily perceived as a relatively strong restriction on the model equations and not many systems may be intuitively expected to fall into this category. Surprisingly, LIED can be applied successfully for the object-oriented modelling of complex thermo-fluid architectures (Zimmer2020,2022) or to mechanical systems with stiff contacts (Zimmer2023).

The idea is that the non-linear behavior of the slow mode is explicitly expressed whereas the fast dynamics that typically is needed to uphold non-linear constraints is expressed by a linear implicit system that fulfils the constraint in its equilibrium. Hence the name: linear implicit equilibrium dynamics. The equilibrium dynamics is thereby often a replacement dynamic and only an approximation of reality (as all modelling is).

2.3 Use Cases for LIED

As the above references demonstrate, LIED has been applied using Modelica for the object-oriented modeling of thermo-fluid or mechanical systems. To this end, it is necessary to use triplets as interface of the model components that consist in a signal for the explicit non-linear part and a pair of potential and a pair of potential and flow for the implicit part as presented in Table 1.

Table 1. Connection triplets for the object-oriented modelling of LIED Systems.

Domain	Signal	Potential	Flow
trans. mechanics	position: r [m]	velocity: v [m/s]	force: f [N]
rotational mechanics	angle: φ [rad]	angular velocity: ω [rad/s]	torque: τ [Nm]
Thermo-fluid streams	Thermo-dynamic state: $\widehat{\Theta}$	inertial pressure r [Pa]	mass-flow rate: $\dot{m}$ [kg/s]

More background on the derivation of these triplets can be found in (Zimmer2024). It goes beyond the scope of this paper how the equations are formulated in detail but the two Modelica model diagrams shown in Figure 1 and Figure 2 may illustrate the practical usefulness.

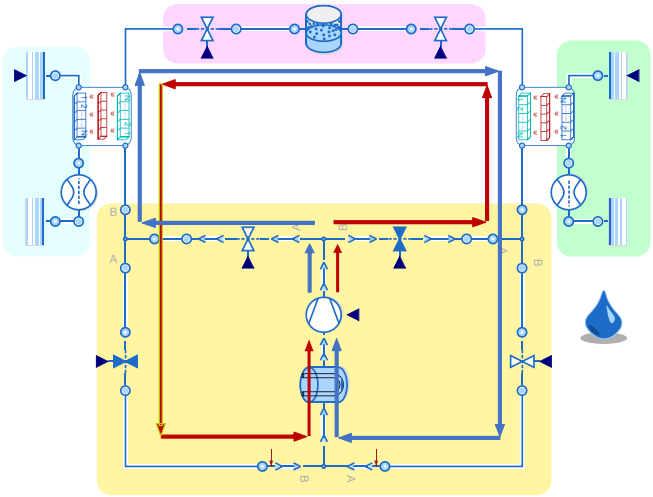


Figure 1. Model diagram of a reversible heat pump systems using the ThermoFluid Stream Library.

DOI
10.3384/ecp21785

Especially the ThermoFluid Stream Library has meanwhile become a popular OpenSource library used by academia (Junglas2023) and by industry.

LIED systems have very benevolent characteristics for object-oriented modelling. Following simple connection rules, the resulting matrix $\mathbf{A}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E)$ will be regular and an a-priori statement on solvability can be given (Zimmer2020). This makes this class of modeling very robust and prevents many computational simulation errors.

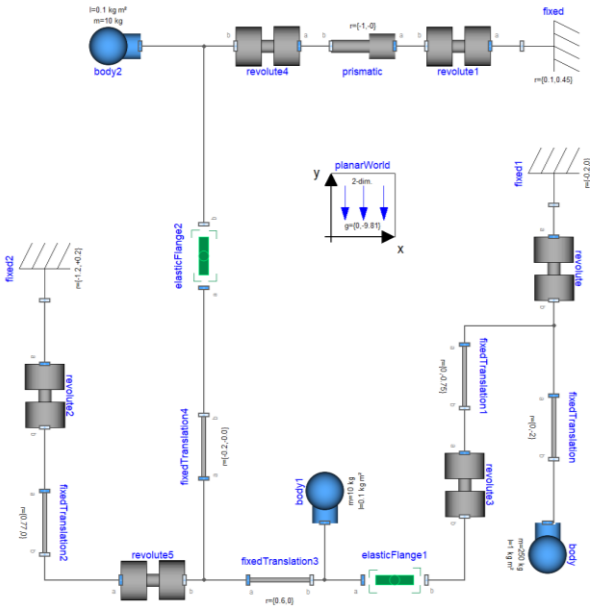


Figure 2. Model diagram of a reversible heat pump systems using the ThermoFluid Stream Library.

New libraries are currently under development that follow this methodology.

3 Potential Compilation of LIED Systems

It turns out that components modeled according to the LIED Methodology have a predictable computational structure. As outlined in (Zimmer 2024), suitable variables for states, their derivatives, for tearing variables and residuals are known a-priori on the level of individual components. This means that flattening is not necessary for LIED systems anymore and that instead the compilation could be performed on the component level. Such a compiler has not yet been developed. So far only a sketch exists on how Modelica models would need to be prepared to enable a component wise compilation.

From a dedicated compiler for LIED systems, we would expect the following advantages:

- Better scalability in terms of code generation speed and memory consumption
- Better error messages based on the connection rules on component level instead on equation level

There may be also disadvantages to be expected:

- Slower simulation code (at least for smaller systems) due to lack of symbolical optimization
- Of course: such a compiler is only applicable of the subset of LIED system and not a full Modelica compiler.

Hence, before making the effort of developing a dedicated compiler for LIED systems, it makes sense to study these benefits and drawbacks of the corresponding compile target. To do so, a dedicated LIED simulator was implemented in C++.

4 Building a Simulator for the LIED Compilation Target.

When compiling a LIED component from Modelica into C++, how would the corresponding C++ code look like? Essentially each LIED component can be represented by a C++ class. The class may thereby define the following elements:

- Interfaces for the signals
- Member functions for the blocks that process these signals
- Local parameters and variables
- Special member objects of type `ContinuousState` that define state variables and their derivatives
- Special member objects of type `Tearing` that define tearing variables and their linear dependent residuals.
- A mandatory virtual member function `metainfo` that provides meta information for the simulation engine.

To better understand these elements, let us look at a concrete example. Figure 3 presents a simple planar mechanical system of a crane crab (a pendulum attached to a slider). To each component of the system, 3 computational blocks are assigned of different color: blue, green, and orange. The blue and green blocks form thereby a computational sequence directed from the root whereas the orange signal leads to the root. For this particular domain of LIED systems, the blue signal contains the positional state and undergoes non-linear transformations. The green signal represents the velocity. Its derivative may be used as tearing variable for which the orange signal that represents the force provides a linear response.

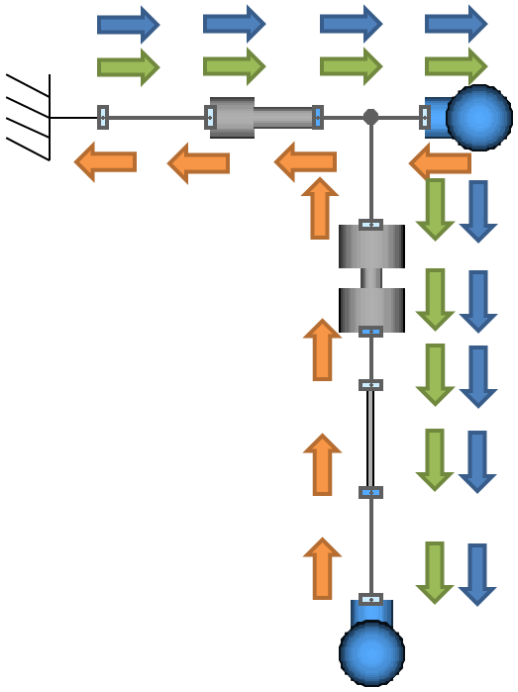


Figure 3. Crane crab modelling diagram

Let us look at one example component that contains all of the elements described above: the revolute joint. In Figure 4, the 3 member functions are marked with the colors to match Figure 3.

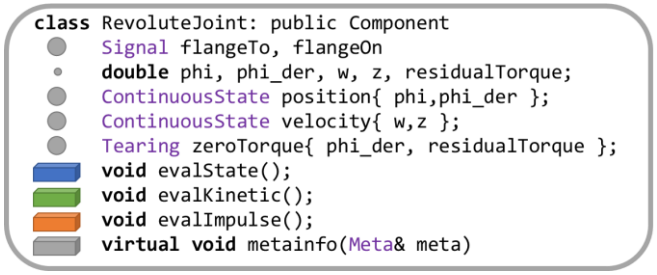


Figure 4. C++ class diagram of a model component.

The virtual member function `metainfo` is crucial to understand how the simulator works. Here is the corresponding code for the `RevoluteJoint`.

Listing 1. Function for metainformation on the revolute joint.

```
virtual void metainfo(Meta& meta) override {
  meta.regComp(&flangeTo,"flangeA");
  meta.regComp(&flangeOn,"flangeB");
  meta.regComp(&position, "position");
  meta.regComp(&velocity, "velocity");
  meta.regComp(&zeroTorque, "zeroTorque");

  meta.regVariable(&phi,"phi: [rad]");
  meta.regVariable(&phi_der,"phi_der:[1/s]");
  meta.regVariable(&w, "w_kin: [1/s]");
  meta.regVariable(&z, "z: [1/s2] ");

  meta.regVariable(&residualTorque,"…");
  meta.addBlock(this,
    [](Component* c) {
      return ((RevoluteJoint*)c)->evalState();
      },
      Signals{&flangeTo.state,
              &position.state },
      Signals{&flangeOn.state}
  );
  meta.addBlock(this,
   [](Component* c) {
     return ((RevoluteJ…*)c)->evalKinetic();
     },
     Signals{&flangeTo.kinetic,
             &velocity.state,
             &zeroTorque.tearing },
     Signals{&flangeOn.kinetic}
   );
  meta.addBlock(this,
   [](Component* c) {
     return ((RevoluteJoint*)c)->evalForce();
     },
    Signals{&flangeOn.j},
    Signals{&flangeTo.j,
            &position.derivative,
            &velocity.derivative,
            &zeroTorque.residual}
  );
};
```

The `Meta` object, whose reference is passed to the `metainfo` function, crawls through the entire model in a recursive way and collects all information of interest. There are different child classes of the abstract `Meta` class: for instance, the `DiagnosisMeta` class collects meta information about the variable names. In order to construct the simulation code, the `StructuralMeta` class is most relevant to us. It collects all the information which block depends on which signals and what variables are states, derivatives, tearing variables or residuals. Given this information callback objects can be created. The final simulation program is then simply a list of these call backs. Sections for dynamic evaluation and tearing of linear equation systems are thereby separately marked.

The following listing shows the top-level representation of the crane crap model in C++ as potential compile target. As you can see, it is possible to maintain the original structure of the model. The reader may use these examples to trace the recursive calls of the `metainfo` function.

Listing 2. CraneCrab Example in the compile target.

```
class CraneCrab : public Component  {
public:
  Fixed fixed{};
  PrismaticJoint  prismatic1{Vector2d{1.0,0}};
  Body body1{1.0,0.1};
  RevoluteJoint revolute2{};
  FixedTranslation rod2{Vector2d{0.5,1.5}};
  Body body2{0.5,0.05};
```


DOI
10.3384/ecp21785

```
Connections con {
   Connection{&fixed.flangeOn,
              &prismatic1.flangeTo},
   Connection{&prismatic1.flangeOn,
       &body1.flangeTo},
   Connection{&prismatic1.flangeOn,
              &revolute2.flangeTo},
   Connection{&revolute2.flangeOn,
              &rod2.flangeTo},
   Connection{&rod2.flangeOn,
              &body2.flangeTo}
 };

 virtual void metainfo(Meta& meta) override {
   meta.regComp(&fixed, "fixed");
   meta.regComp(&prismatic1, "prismatic1");
   meta.regComp(&body1, "body1");
   meta.regComp(&revolute2, "revolute2");
   meta.regComp(&rod2, "rod2");
   meta.regComp(&body2, "body2");
 };
};
```

Figure 5 presents the overall architecture of the simulation program. We can go through its main elements from the left to the right.

Models consist of Components, Signals and Connections, whereby both signals as well as components may have member functions that represent blocks.

When a model gets instantiated, it gets prepared for its evaluation. This is the function of the `ModelEvaluation` class. It will use the `StructuralMeta` crawler to collect all necessary meta information. Then all blocks will be sorted and the sections representing torn linear equation systems will be compressed and marked. Compression hereby means that all blocks not belonging to the linear system are placed outside this section.

The Simulator class will now apply a numerical ODE solver. Currently `zimsim` features a generalized solver for explicit Runge-Kutta Fix Step Methods with Butcher Tableaus for order 1 to 4, Backward Euler, and ESDIRK23 (Jørgensen2018) as solver algorithms.

When the simulator generates results, it calls the Recorder class. The recorder then decides on its own what to store at which frequency. The recorder may use a separate crawler on the simulated model to retrieve information on variable names, model hierarchy etc. Also, it may trigger the model evaluation if separate evaluations are needed for the generation of output variables. The recorder class is defined in a very generic manner and can be used to multiplex the output to other recorders.

We see that the design of a simulator for pre-compiled components may differ significantly from the monolithic generation of simulation code.

The structure of the model is maintained for the simulation code and instead of having to decide which meta-information to provide for the output before translation, this decision can be done at simulation time or even after simulation took place.

What are the benefits and drawbacks of such an architecture? We would expect a much lower memory consumption because meta-information can be generated on demand. On the other hand, we would expect a higher computational overhead: alias variables are not eliminated and linear equation systems need to be numerically solved instead of symbolically.

To better quantify these benefits and drawbacks, a scaling experiment is being performed where memory consumption and computational speed is being measured for models of different size.

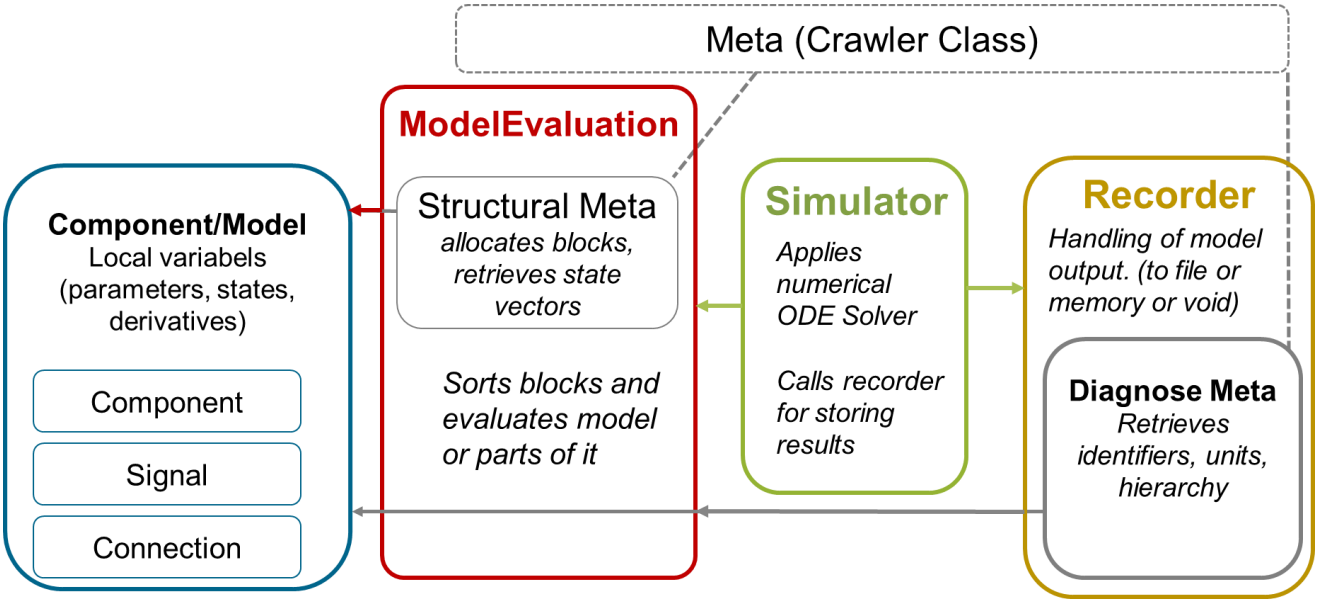


Figure 5. Architecture of the object-oriented simulator program

5 Evaluation of the Scaling Properties.

The crane crab model of Figure 3 has been used to perform a simple scaling experiment. Using a logarithmic grid of factor 4, 1 to 4,194,304 crane crabs have been instantiated and simulated using `zimsim`. The same exercise has been performed within an equivalent Modelica model using the same equations. OpenModelica (Fritzson2019) (abbreviated by OM) and a commercial Modelica tool have been used for the simulation tests. Here, this generates models ranging from 206 equations up to 792 million equations. In Figure 6 to Figure 8, we use the equation number since this quantity is more familiar to Modelica users.

Comparing simulation speed and memory consumption across tools is not trivial. For instance, the simulation speed may be influenced by the generation of output. For this reason, output generation was largely suppressed. Memory consumption refers to the estimated peak consumption of the whole program.

In general, one should be very careful when comparing absolute numbers for a specific case. What matters to a scaling test though is the slope in the log-log diagram.

All tests have been performed on a machine with Intel(R) Core(TM) i7 at 1.80 GHz and 32GB of RAM using Windows 10 as operating system.

All tools have been tested to their maximum extent. There are however orders of magnitude in difference between the tools. In the end, `zimsim` was limited by available memory. Also the memory consumption of the commercial tool caused disk swapping slowing down the whole process to such an extent that it did not finish within several hours. In OpenModelica, a clang linker error occurred for 1024 crane crabs. Simulation performance could thus not be estimated. Memory consumption and compile time for this data point were replaced by values that exclude the linking.

5.1 Memory Consumption

`zimsim` has been written with efficient use of memory in mind: Meta information is generated only on-demand and also the object-oriented formulation enables to do more computation on the steak than on the heap.

Peak memory usage for model translation, compilation and simulation is orders of magnitudes lower using `zimsim`. This is shown in Figure 7. The likely reasons are:

- Avoidance of flattening leading to much less instruction code.
- On demand generation of meta-information
- More variables on stack than on heap

For small models, comparing the memory usage hardly makes any sense and is also not fair because we compare the memory usage of a dedicated simulator (LIED) with the one of a whole modeling and simulation environment. The two orders of magnitude for small models is thus not surprising. What is surprising that this gap does not significantly close for larger models. Peak memory consumption seems to appear in the tools at model translation for the Modelica tools but this is also true for `zimsim.` The last data point of Figure 7 is misleading. The model was assigned 29GB of RAM by the operating system while the estimated remaining 15 GB of memory demand where provided by disk swapping. This also shows in the performance of model instantiation but not in simulation because this can be performed within the provided 29GB.

5.2 Computation Time

Pure simulation time is compared here. Time for translation, compilation and instantiation is ignored. Generation of output has been reduced to a negligible amount.

Figure 8 reveals that simulation speed of `zimsim` is competitive to OpenModelica but slower for small models than a commercial tool. The likely reason is the numerical solution of linear equation systems in `zimsim` instead of symbolical transformation. This primarily reveals one thing: flattening for the sake of flattening is not worth it. Structured code is also fast. Flattening is then a good idea when you can replace an iterative solver with a symbolic solution. (however, there might be ways to generate structured code that reduces iterations as well)

We can observe that for larger systems, the performance penalty is smaller, the reason is unknown.

A side remark: The performance study has only been performed using explicit solvers, ignoring the computation of the Jacobian which is very relevant for implicit solvers. However, implicit solvers scale badly for very large systems.

5.3 Time for Translation + Compilation or Instantiation

One expected advantage of pre-compilation would be that a model could be instantiated almost immediately. Hence, the time for model instantiation in `zimsim` is displayed in Figure 6.

The instantiation essentially uses the famous Tarjan algorithm for strong component analysis that provides a partial order of the dependence graph as side effect. It can be performed very efficiently. For it to take longer than a single second, models of more than 1 million equations are needed.

Time for Translation and Compilation in the Modelica tool is added to the chart. This measurement is not very precise (at least 20% error bar) and for all smaller models completely dominated by the C compiler. The comparison is hence not fair for smaller models.

DOI
10.3384/ecp21785

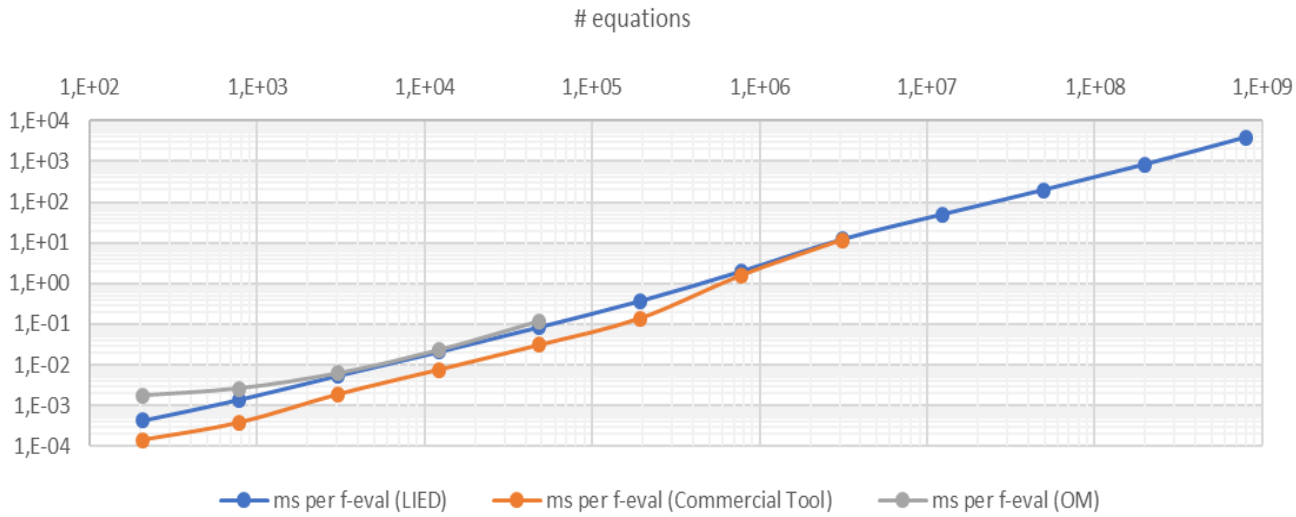


Figure 8: Computation speed: milliseconds per model evaluation over number of equations

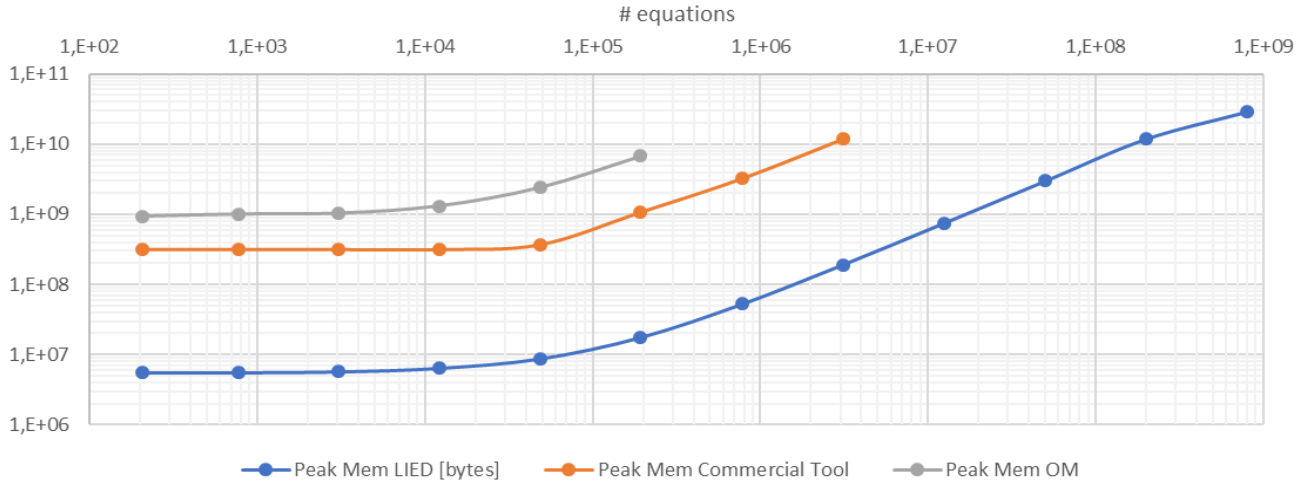


Figure 7. Peak memory usage in bytes over number of equations

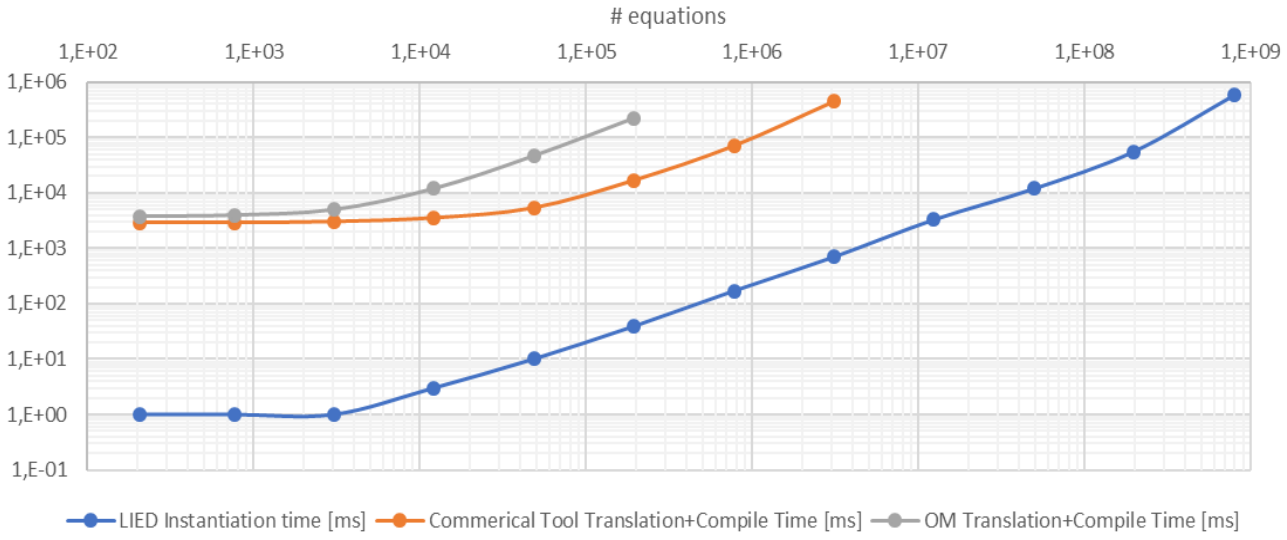


Figure 6. Instantiation Time or Translation+Compile time in ms over number of equations

6 Conclusion

The goal of this paper is to explore and compare alternative pathways for the compilation of simulation code. Naturally flattening and the generation of monolithic rather unstructured simulation code are procedures inherent to compiling Modelica. These procedures enable the effective symbolical manipulations and the removal of alias variables.

How strong these advantages are compared to a more structured generation of simulation code that compiles on the component level was however examined sparsely.

The preliminary examination presented in this paper is not sufficient to provide conclusive answers yet. Improvements in the algorithm and simulation code are needed because many potentials for code optimization have not yet been exploited. Furthermore, a much wider set of examples is needed. At the current stage, it merely indicates that the performance gain by flattening per se is not very significant, symbolic solution are preferable over iterative ones and it also shows that pre-compilation has a great potential to be way more memory efficient enabling the simulation of much larger models.

We also have to recall that the component-wise compilation could not be performed on arbitrary models but only on models following the LIED methodology. Albeit this is a remarkably powerful subclass of Modelica, it is still only a subclass.

Acknowledgement

This work was supported by the ITEA4 research project OpenScaling with financial support of the German federal ministry of education and research referring to the grant number: 01IS23062C

SPONSORED BY THE

Federal Ministry of Education and Research

References

Abdelhak, K., F. Casella, B. Bachmann (2023) Pseudo Array Causalization. Proceedings of the 15th International Modelica Conference, Aachen

Agosta F., et al. (2023) MARCO: An Experimental High-Performance Compiler forLarge-Scale Modelica Models Proceedings of the 15th International Modelica Conference, Aachen.

Benveniste, Albert, Benoît Caillaud, Hilding Elmqvist, et al. (2019). “Multi-Mode DAE Models - Challenges, Theory and Implementation”. In: Computing and Software Science – State of the Art and Perspectives. Vol. 10000. Lecture Notes in Computer Science. Springer, pp. 283–310.

Benveniste, A., B. Caillaud, M. Malandain, J. Thibault (2023) Towards the separate compilation of Modelica: modularity and interfaces for the index reduction of incomplete DAE systems Proceedings of the 15th International Modelica Conference, Aachen.

Braun, Willi, Francesco Casella, and Bernhard Bachmann (2017). “Solving large-scale Modelica models: new approaches and experimental results using OpenModelica”. In: Proceedings of the 12th International Modelica Conference.

Casella, Francesco (2015). "Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives", Proceedings of the 11th International Modelica Conference, doi:10.3384/ecp15118459

Fritzson, Peter A. et al. (2019) “The OpenModelica Integrated Modeling, Simulation, and Optimization Environment.” Proceedings of The American Modelica Conference 2018, October 9-10, USA

Jørgensen, J.B., Kristensen M. R. and Grove, P. (2018) A Family of ESDIRK Integration Methods. arXiv Numerical Analysis eprint :1803.01613

Junglas, P. (2023) Implementing Thermodynamic Cyclic Processes Using the DLR Thermofluid Stream Library. Simulation News Europe E 33(4)

Mattsson, S.E., Gustaf Söderlind (1993). “Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives” In: SIAM Journal on Scientific Computing 1993 14:3, 677-692

Neumayr, A.; Otter, (2023) M. Modelling and Simulation of Physical Systems with Dynamically Changing Degrees of Freedom. *Electronics*, doi.org/10.3390/electronics12030500

Pantelides, C. (1988), The consistent initialization of differential-algebraic systems, SIAM J. Sci. Statist. Comput., 9, 213–231

Zimmer, Dirk (2009) Module-Preserving Compilation of Modelica Models. In: *Proceedings 7th Modelica Conference*, Como, Italy, Sep. 20-22

Zimmer, D. (2020), Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks. Mathematical and Computer Modelling of Dynamic Systems, Vol 26, Issue 3.

Zimmer, D., N. Weber, M. Meißner (2022) The DLR ThermoFluid Stream Library. MDPI Electronics - Special Issue.

Zimmer, D., C. Oldemeyer (2023). “Introducing Dialectic Mechanics”. Proceedings of the 15th International Modelica Conference, Aachen.

Zimmer D. (2024) Object-Oriented Modeling of Classic Physical Systems using Linear Implicit Equilibrium Dynamics Preprints 2024, 2024031139

Requirements-based, early stage Architecture Performance Validation on a Brake System Use Case

Sinyoung Kang[1] Marcel Gottschall[2] Sunghyun Cho[1] Torsten Blochwitz[2]

[1]Hyundai Motor Company (HMC), Republic of Korea {Newyoung,hyuni}@hyundai.com
[2]ESI Group, Germany, firstname.lastname@esi-group.com

Abstract

Automotive industry OEMs and suppliers are progressing their engineering processes and performance to the next maturity level gearing to digital thread solutions. Current challenges like continuous engineering, virtual certification, distributed development, consolidated virtual proving grounds, homologation, digital twin and operational applications, require well informed decision making in a comprehensive, reliable, traceable and customizable environment. In particular, in automotive domain, with widespread tight collaborative ecosystems between integrators and suppliers, the capability of tracing each decision and its underlying artifacts becomes a key value of an engineering platform.

This paper will outline a middleware approach to reuse generated artifacts and their relationships in a federated engineering environment and applying Modelica simulation models as the key integrator between architectural decision making and subsequent development up to detailed 3D design. Based on an exemplary, automotive brake system setup, the benefits of integrated data and workflows from specification to virtual architecture exploration and design verification are highlighted to motivate their value towards a realization of continuous model-based systems engineering methodologies.

Keywords: MBSE, System Simulation, Virtual Validation, Automotive Brake System

1 Introduction

The increasing complexity of vehicle development has led to a continuous effort to adopt system engineering-based approaches. Organic integration and management through system engineering are essential to achieve optimal performance and reliability. Recently, model-based systems engineering (MBSE) methodologies have emerged as an alternative to document-centric systems engineering, enabling a holistic view and systematic management of the growing complexity of requirements.

The vehicle architecture validation process for braking performance involves various aspects and engineering disciplines, such as braking distance, deceleration, and thermal capacity. However, past performance validation efforts have lacked systematic systems engineering review, and requirements and test case management have not been well-structured, leading to significant challenges, both in traceability and integrated performance assessment, particularly in the early stages of vehicle development. Specifically, a structured approach to linking requirements and 1D simulation models has been lacking, and MBSE methods have been proposed as a solution to systematically manage these relationships in theory.

In the daily engineering reality, the disconnection between requirements, MBSE-based architecture models, and 1D models, each managed in separate specialized software, has hindered effective traceability management.

Therefore, the primary objective of this paper is to implement a *digital thread* that organically connects and manages requirements, architecture models, and 1D performance models, covering the early stages of development of cyberphysical systems[1]. By establishing a database-centric digital thread, the aim is to build an integrated and efficient management approach that ensures traceability and consistency. Specifically, this study will leverage MBSE methodologies to effectively integrate SysML-based architecture with 1D models and design requirements. This approach not only addresses the current inefficiencies, but also lays the foundation for a more robust and scalable systems engineering process.

2 Motivation and Implementation Approach

Achieving digital continuity along the engineering cycle as shown in Figure 1, leveraging the available information and digital assets, is a major contributor to more efficient development processes forced by cost and cycle time reduction needs. However, this is not yet state of the art, in particular in real-world, multipartner, industrial and productive applications and ecosystems.

2.1 The Need for MBSE at HMC

Traditionally, product development has been conducted independently in various isolated disciplines such as mechanical, electrical/electronic, and control engineering. This approach has led to challenges in addressing issues that arise during the development process, which can result in difficulties in design, product integration, and ver-

[1]technical systems that consist of a physical part and a software/controller component

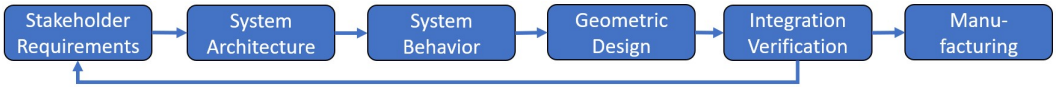


Figure 1. Flat representation of the well known V-cycle model (only major steps, excluding operation of system of interest)

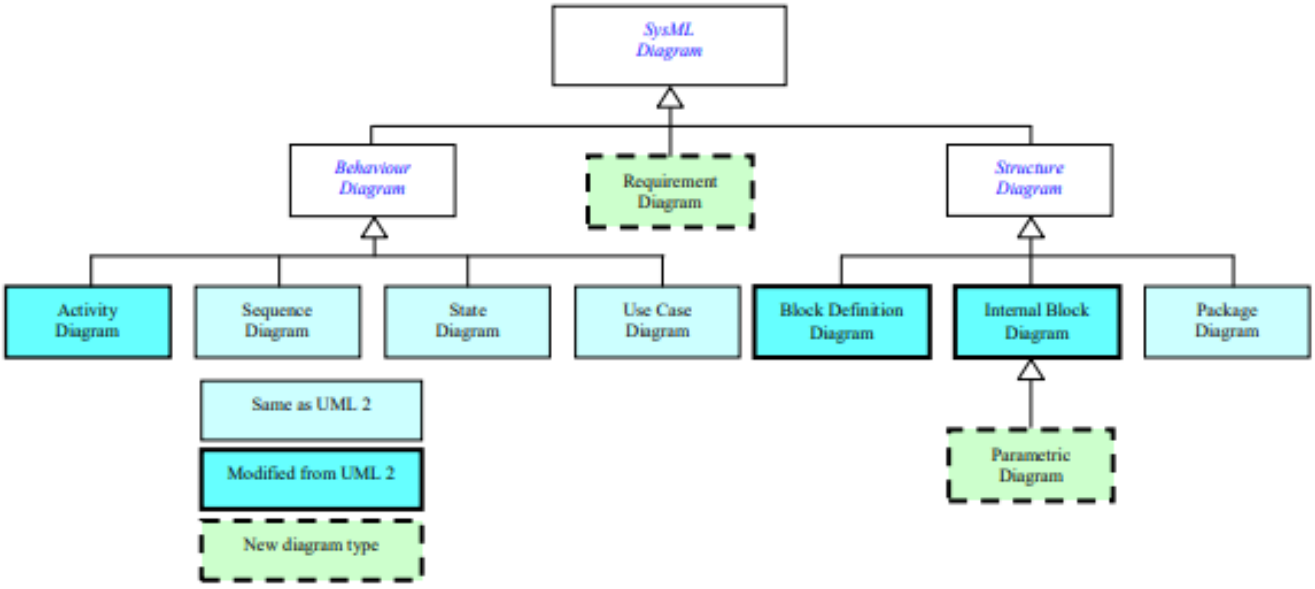


Figure 2. OMG SysML Diagram Taxonomy (Hause et al. 2006)

ification, ultimately leading to delays in product launch. With the increasing complexity and intelligence of modern products, there is a growing need to address these limitations. Model-based technologies, like MBSE have emerged as an alternative to overcome the limitations of traditional systems engineering and development frameworks. Those approaches are lacking of effective communication and collaboration across different engineering domains. These siloed methods often result in misaligned objectives, conflicting requirements, and a fragmented understanding of how changes in one area can impact others. Consequently, critical design decisions may be made without fully considering their implications on the overall system performance, leading to inefficiencies and increased risk of failure in the final product.

Hence, in this research and proof of concept application, the focus is on utilizing the MBSE methodology in the domain of brake system performance, with the aim of not only validating the approach itself within a single performance area and system, but also exploring the potential for understanding the relationships and traceability between multiple performance aspects and across multiple systems within complex, multidimensional design processes.

2.2 Architecture System Modeling

The Systems Modeling Language (SysML) is a standardized language developed to effectively implement MBSE, (Object Management Group 2022). SysML, defined by the Object Management Group, is an extensible, graphical language that can comprehensively represent the *structure*, *behavior*, *requirements*, and *parameters* of complex systems in a single model representation.

While the widely used Unified Modeling Language (UML) has been primarily focused on software-centric modeling, SysML extends it to support modeling across various domains, including hardware, software, information, and processes, encompassing the entire system. Thus it ideally covers the scope of cyberphysical systems, like the mentioned *brake system*. The key SysML diagrams shown in Figure 2 include Requirement Diagrams, Structure Diagrams, Behavior Diagrams, and Parametric Diagrams, which enable clear visualization and analysis of different aspects of the system.

Specifically, when modeling dedicated performance domains, such as braking performance, SysML offers the following advantages (Hause et al. 2006):

- **Requirement Traceability**: The Requirement Diagram (REQ) allows for the clear definition of system requirements and enables tracing them throughout the design and verification process to ensure requirement fulfillment.
- **Structural Analysis**: The Block Definition Diagram (BDD) and Internal Block Diagram (IBD) can be used to explicitly express the functional and logical structure of the system, helping to clarify the relationships between system components and maintain consistency from the design stage to the integration and verification stages.

DOI
10.3384/ecp21793

- **Behavioral Modeling**: Diagrams such as Activity Diagrams (AD) and State Machine (SM) Diagrams can be used to model the dynamic, but abstract, non-physical behavior of the system, enabling the simulation of operational scenarios and the identification of potential issues in advance.

- **Parametric Analysis**: Parametric Diagrams can be utilized to define the performance parameters of the system and analyze the system's performance under various conditions.

Based on these SysML features and capabilities, the present work will explore the benefits of such hierarchically concatenated artifacts, when they are reused in physical simulation domain. It is expected, that a *seamlessly* integrated workflow consisting of a) summarizing the verification requirements for braking performance in vehicle development, b) representing traceability of these requirements through the relationships between Requirement Diagrams and Block Definition Diagrams, and c) extracting structural analysis of the brake system architecture through Block Definition Diagrams and Internal Block Diagrams will demonstrate the effectiveness of MBSE methodology for real, cross-domain early stage engineering processes for hierarchical multiphysical systems.

2.3 Limitations of existing Approach at HMC

In current systems engineering practices, requirements are typically managed using documents such as Excel or Word, or requirement management software like IBM *DOORS*. However, it is not straightforward to link the requirements managed or handled in these documents or software to the Requirement Diagram in a SysML-based architecture model, particularly in typical multi-vendor engineering tool landscapes that are in place at enterprise level. While some SysML authoring tools like Dassault Systems *CAMEO Systems Modeler* provide plugins, these often expect the requirements to be exchanged as a ReqIF standard XML file, necessitating additional, manual steps to directly integrate the requirements in the model and continuous process.

Furthermore, there are even greater challenges in connecting the SysML architecture model to the analysis models used for actual verification. To validate the content of the SysML model, 1D and 3D analyses, or physical testing must be performed. Even if the SysML architecture model is used as a basis, significant time and effort are still required to set up the analysis models, especially for 1D, physical analysis approaches like Modelica-based solutions (Modelica Association 2021), where a separate 1D model needs to be created, despite the similarities in the structure. Recent standards, like Modelica SSP[2] (Modelica Association 2022) are tackling some of these integration aspects between architecture and simulation domains, but are still far away from productive application and dissemination.

As described above, while the benefits of MBSE-based architecture models are recognized, it remains difficult to maintain traceability between requirements, test cases, and 1D simulation models, as these are often managed in separate tools. Additionally, it is challenging to verify 1D model results against the requirements-based validation.

2.4 Digital Thread Implementation

To address the inconvenience caused by the incompatibilities in the working environments for requirements, architecture models, and 1D models, we conceived the idea of integrating the *artifacts*, created by the engineering clients, instead of connecting relevant software tools one-to-one by import/export actions.

In order to achieve such tool agnostic, vendor neutral continuous engineering platform, a microservice-based middleware approach is implemented as shown in Figure 3. Consisting of OSLC[3] compliant, domain-specific webservices, e.g. requirements- or testmanagement, this layer provides full flexibility and scalability with respect to the considered process. It stays intact even when frontend (engineering clients) or backend (data management) tools are replaced over time, thus keeping consistency throughout the linked engineering data, (Open Services Project 2021). Moreover, depending on the features of the applied backend systems, the middleware layer can also provide general support functions like user management or versioning. However, usually such capabilities are integrated in modern datamanagement backbones, e.g. PLM[4] systems. On the other side, the engineering clients are accessing the engineering data by dedicated plugins (as the only component to be adapted in case of changes in tooling), which replicate the corresponding, desired engineering workflows using the linked data that is exposed by the middleware. Hence, from user perspective, this reflects the *single source of truth* paradigm, as each participant that is part of such integrated process consumes or pushes his information from or to the same entry point without data replication.

With this technology, we connected a representative multi-vendor engineering environment, covering early stage design and verification phases:

- the requirements management software **IBM** DOORS (classic),

- the SysML architecture modeling tool **Sparx Systems** Enterprise Architect,

- and the Modelica-based 1D analysis tool **ESI** SimulationX,

to ensure traceability and enable efficient architecture-level braking performance validation. Furthermore, we

[2]System, Structure and Parametrization

[3]Open Services for Lifecycle Collaboration

[4]Product Lifecycle Management

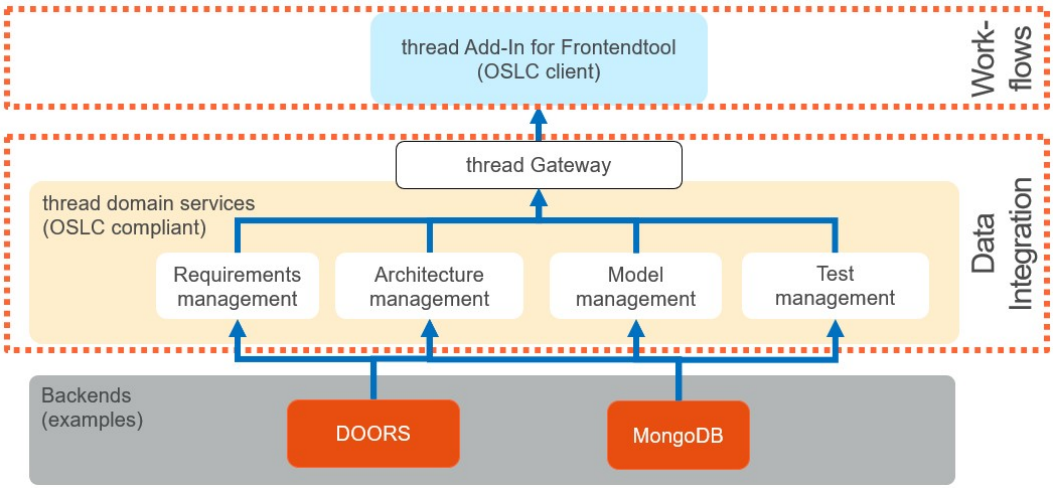


Figure 3. Tool agnostic, digital thread middleware approach based on OSLC compliant webservices

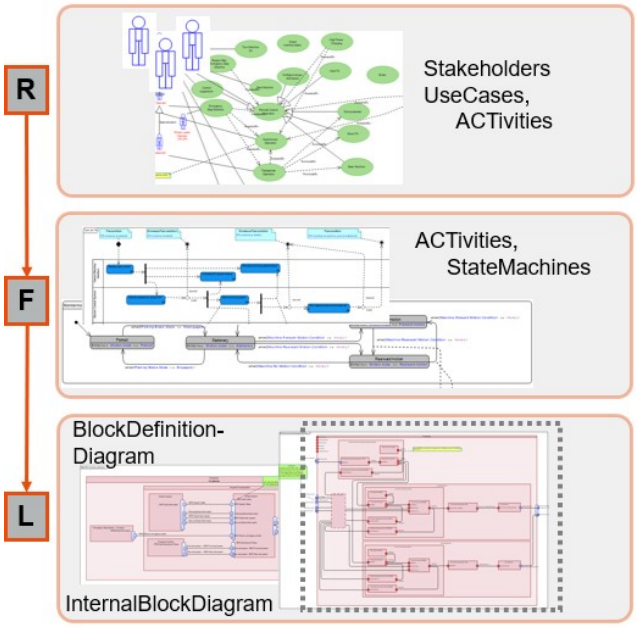


Figure 4. MBSE approach linking design artifacts, logical structure (L) *implements* functions (F) that *satisfy* requirements (R)

implemented a bidirectional traceability structure, not just a unidirectional link from requirements to the 1D simulation model. By leveraging the OSLC technology and a database-driven approach, we were able to establish a connection that allows the verification results from the 1D performance model to be reflected back in the requirements definition and management system. The application of the implementation will be demonstrated and discussed in the next sections.

3 Design Methodology

Following the principles of MBSE, a formal and clearly structured process, shown in Figure 4, enables the derivation of structural descriptions (components and their interrelation) of the target architecture, as the main input for the simulation domain. Starting with high level requirements or stakeholder needs (R) on the system of interest, a functional architecture (F) is designed which satisfies the requirements. Applying various abstract modeling and analysis descriptions like Use Case- and Activity Diagrams or State Machine simulations, enable the (non physical) justification of the design and a further decomposition of the functional behavior into logical components (L). Apart from logical and discrete simulations, the systems architecture has to be verified against performance requirements by 1D physical simulation, before propagating system sizing and design parameters to the 3D design stage.

However, historically, systems engineering (holistic) and simulation (domain experts) have been disconnected, which results in a gap in the level of detail for the logical structure description (L), as the architecture definition in MBSE output is still abstract and does not fulfil physical simulation needs. This issue, recently discussed in (Cederbladh et al. 2024), has to be handled and adressed, when aiming for a cross-domain digital thread implementation with seamless integration from architecture to performance simulation. Applying stereotyping to connections between different logical components (L), maps the abstract layer of description in SysML language (e.g. "two components are exchanging energy") to the information in physical domains (e.g. "electrical connection is required between motor and battery"), that is necessary to build corresponding physical simulation models to be used for the architecture verification. Hence, these physical models are not only serving the actual performance validation activities, but also enable the detailing and specification of interface definitions on the logical layer based on simulation justification, which is absolutely required for complex systems, early stage architecture exploration activities. With this extension to SysML model descriptions, it becomes possible to automate the reuse of architecture information for generating physical simulation model tem-

DOI
10.3384/ecp21793

1.5 EnergySavingRate Energy Saving Rate > 0.9 (90%) according to Equation provided Energy saving rate = regenerative braking energy/total braking energy (Total braking energy = regenerative braking energy + mechanical braking energy)	1	Passed	http://10.5.218.142/sandbox/reqmgmt/656744f742273550bcbcb828

Figure 5. Specific case for brake thermal management, representation of the performance requirement in DOORS with content (first column), version (second column), virtual verification result (third column) and ID of OSLC artifact (fourth column)

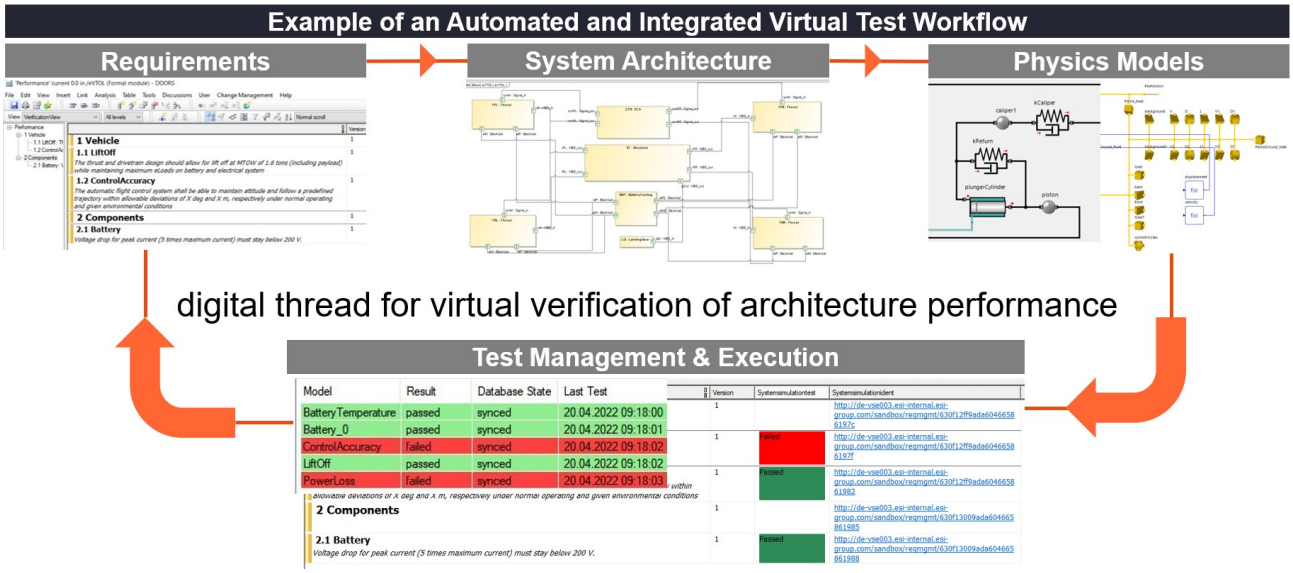


Figure 6. Closed loop, MBSE-based early stage, architecture and design verification using 1D Modelica models replicating virtual test scenarios defined by system requirements (REMARK: due to IP restrictions, the screenshots do not show the actual brake system models, but demonstrate the concept and workflow)

plates, that are linked to certain requirements and used in subsequent performance analysis tasks. Another benefit of this cross-domain integration from design perspective, is the inheritance of global architecture design parameters that automatically become (Modelica) parameters of the resulting behavioral models, as demonstrated below.

3.1 Brake System Performance Use Case

To validate the braking performance, the vehicle development requirements managed internally by the company were consolidated, serving as a foundation for establishing the objectives to be verified through a 1D simulation model. It should be remarked, that there are of course plenty of structural and performance requirements on the brake system, referring to different hierarchy levels (subsystems, components), which are qualified for virtual verification by simulation and have been used in this study, e.g stopping distance with ABS or several system- and unit fail scenarios. However, to demonstrate an efficient, integrated validation process, the *Energy Saving Rate* verification shown in Figure 5 is selected as a representative case, as the Modelica-based multiphysics model includes brake hydraulic, mechanical, thermal and electrical system behavior for simulating braking performance .

The Energy Saving Rate (ESR) is obtained through energy analysis. First, energy is calculated by integrating power during braking periods in driving cycles, such as the WLTP[5] for Hybrid Electric Vehicles (HEV) and Electric Vehicles (EV) that are equipped with regenerative braking. Then, the calculation ESR formula is as follows:

- ESR = Regenerative Braking Energy / Total Braking Energy, with
- Total Braking Energy = Regenerative Braking Energy + Mechanical Braking Energy

Each company has established target values for the ESR, which is considered confidential information within each organization. Therefore, a sample value of approximately 0.9 has been set for the target in this study.

3.2 Process Demonstration

As described above and summarized in Figure 6, the virtual validation process consists of 4 major steps, seamlessly linking and integrating the disciplines of requirements, functional and logical architecture and physical simulation models for design and verification tasks. With the implemented digital thread, a closed-loop workflow with digital continuity is established, enabling collaboration between the R-F-L-P and test domains. Such setting will allow for certification credits based on simulation results, as end-to-end traceability is one of the key require-

[5]Worldwide Harmonized Light Vehicles Test Procedure

ments in regulations like ISO 26262, (Peraldi-Frati and Albinet 2010; Maro, Staron, and Steghöfer 2017).

The braking system and performance requirements established earlier, are stored in the digital thread database backend by a plugin in the DOORS client. Please note, that this digital thread backend is running in parallel to databases that are potentially in place when common engineering tools client/server architectures are deployed. Depending on their flexibility with respect to the datamodel modifications and extensions, another multi-purpose database might be required to handle the linkages between the different engineering artifacts. In a second step, the requirements are retrieved into the architectural design tool Enterprise Architect, again using a dedicated plugin. This way, the requirements and their hierarchy are automatically added to the SysML model, as a starting point for the functional design step.

Using the requirements retrieved via the digital thread, the braking functional elements were created in the architecture model, represented through requirements diagrams and block definition diagrams. Additionally, the physical structure necessary to implement the defined functions for brake validation is depicted in an internal block diagram, facilitating architecture modeling based on physical structure analysis. This process allows for the definition of traceability among requirements, functional definitions, and physical structures, with their interrelationships expressed through connections between blocks. A more detailed description of the different, formalized steps can be found in Aleksandraviciene and Morkevicius (2018). It is one of the key strengths of a holistic systems model representation, to easily express the allocation of architectural components to certain requirements for verification purposes in a user convenient, graphical way.

After the logical structure breakdown is derived from the functional design, the various connections between the different components are allocated to dedicated physical domains, like hydraulics, electrics or mechanics, using stereotyping[6]. This step drives an active collaboration and iteration between the systems- and simulation engineer, as the latter is the domain expert on specific systems and has knowledge on potential limitations or boundary conditions when mapping physical realizations to a certain logical structure. In addition, the already mentioned architecture design parameters are defined and added to the corresponding logical components. Such parameters refer to system level, *global sizing* or *structural variants inside a certain architecture* descriptions and are reused to define simulation runs downstream in the verification process.

The physical structure - or more precisely, the logical structure augmented with physical information - represented in SysML IBDs, is organized specifically for the brake system to evaluate braking performance. In particular, for complex systems with many hierarchical levels, the corresponding physical models become big with high computational efforts, leading to weak performance in use cases like architecture exploration with large amount of simulation runs. Hence, the concept of *partial IBDs* is introduced for improved virtual testing performance. As shown in Figure 7, dedicated IBDs, focussing on specific aspects of a complex logical component, like thermal or mechanical domain, are extracted from the inner structure described by the full (multiphysics) architecture IBD. These partial IBDs only show components and subsystems, that are contributing to the specific aspect of the component or system, like the thermal behavior in the current demonstration use case. As mentioned above, these architectural components are then linked or allocated to the corresponding (sub-system) requirements. Hence, there is a clever separation between the reduced architectural models used for physical verification runs (here: the thermal management model), and the full logical structure representation (here: the brake system model), used for design descriptions like *bill of materials* of a system. Another benefit of this approach is the easy support of function-based design and testing methods in the 1D simulation domain, because of the allocation of the reduced architectural models to functions and requirements, following the R-F-L process. In summary, structural information in (partial) IBDs and linking of architecture to requirements are key information and input for the subsequent step of system simulation in the design process. The available, integrated information of a system adheres to the Model-Based Systems Engineering methodology, enabling the direct reuse of the internal block diagram for cross-domain linking of architecture-with the corresponding 1D simulation model.

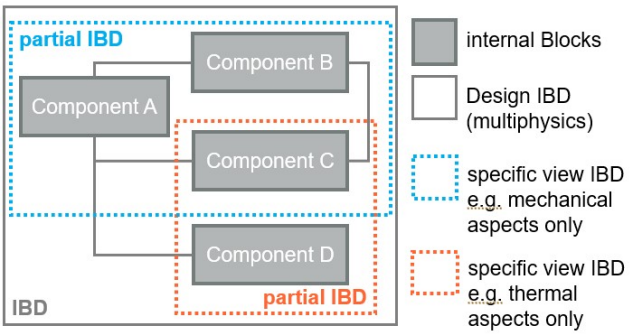


Figure 7. Concept of partial IBDs as representation of specific aspects, e.g. a single physical domain, of a logical component for integrated virtual verification purposes

Similar to the requirements, relevant information of the SysML model from process perspective, like the IBD representing the physical structure of the brake system, or tracelinks between requirements and the internal block diagram, are uploaded to the digital thread environment's database using the plugin developed for Enterprise

[6]adding additional information to modeling components

DOI
10.3384/ecp21793

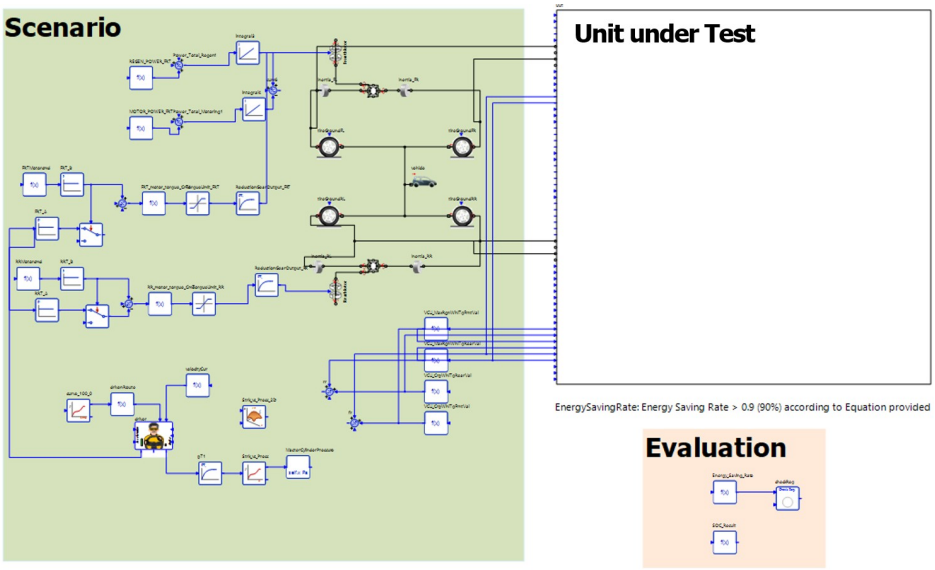


Figure 8. Testmodel (simplified) for the thermal management requirement example, consisting of 3 main components: a) design model instance (*Unit under Test*), b) stimulation, environment and parameters (*Scenario*), and c) test verdict extraction (*Evaluation*)

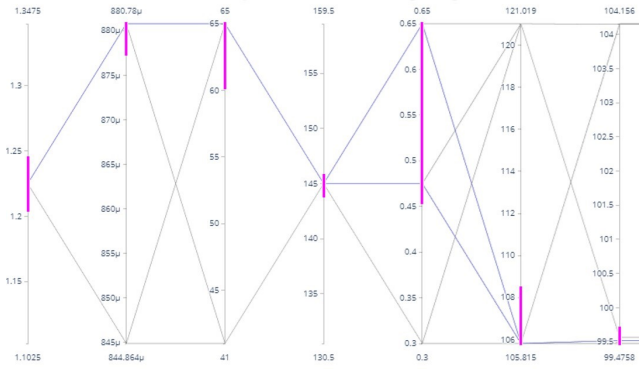


Figure 9. Design parameter study (sizing) to identify simulation runs within a desired range of model parameters (pink bars) using python scripting capabilities integrated in the model

Architect. This is required, as the version of the tool used in this study is not connected to any data backend but relies on local file storage.

In the following, by utilizing the SimulationX plugin, a connection is established with the database through the digital thread, enabling the automatic generation of Modelica libraries (packages) and types (models), that correspond to and reflect the structure of the IBD of the SysML architecture model. These automatically generated Modelica templates implement the connections and ports as well as the system hierarchy represented in the IBD. However, it is important to note that the Modelica blocks derived from the SysML architecture model are not immediately executable models, they merely represent the architectural structure of the brake system. Therefore, to conduct a 1D performance analysis, it is necessary to add components capable of performing such operations.

Thus, the simulation engineering process is taking place in a next step, creating physical models inside the templates, describing the actual behavior of the components. Please be aware, that the model generation can only be automated based on available information, and as mentioned above, SysML language is limited to an abstract layer of system modeling not intended for physical simulation. This physical modeling can happen, either by using native Modelica models from the library, e.g. a valve or heat exchanger, or reusing models from other, external sources by means of FMI, (Modelica Association 2014), e.g. a controller modelled in The Mathworks *Simulink*. The internal components of the brake system Modelica blocks are detailed to enable 1D analysis of braking performance. Again, the engineering artifacts (system simulation models) are shared with the stakeholders by uploading the information to the digital thread database using the SimulationX plugin.

To enable the actual virtual verification and to measure brake performance, the physical models of the brake system architecture have to be augmented as shown in Figure 8. Now, the previously created tracelink between architectural components and requirements are reused by the SimulationX plugin to support the simulation engineer and show a list of *test cases*, consisting of requirements that need to be tested by simulation and are already allocated to structural components. By selecting a certain test case, a *testmodel* is automatically generated, instantiating the appropriate Modelica component from the generated system library as *Unit under Test* in the figure above. The architecture design parameters, discussed before, are present on the corresponding Modelica component and inside the Unit under Test block, and can be used to execute variants studies (design space or architecture exploration) or component optimization tasks as shown in Figure 9 within a given system structure. Such analysis

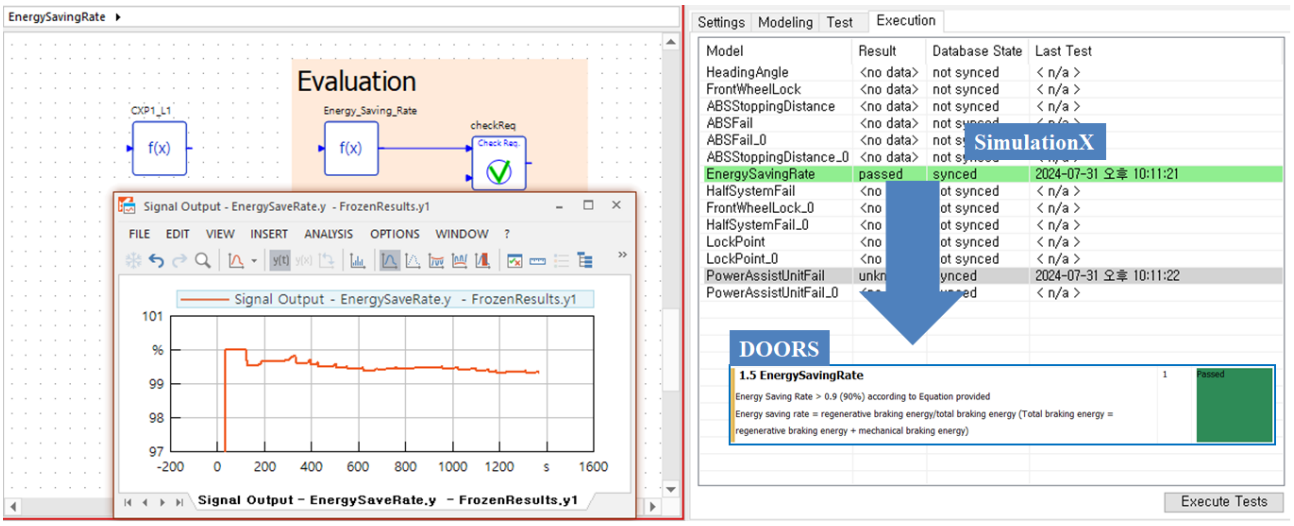


Figure 10. Evaluation result of braking scenario for thermal management requirement use case, transient behavior in the Modelica model (left), test verdict representation in tool plugins (right)

workflows can be automated easily by considering only valid simulation runs, since the model is connected by the digital thread to requirements to be tested and fulfilled.

In order to replicate a test scenario described by the linked requirement, additional components need to be modeled by the simulation engineer. As shown in the thermal management model above, environment and stimulation incorporating virtual vehicle models, including the driver, signals, vehicle weight, and wheelbase, etc. are added to the model collected in the green *Scenario* part. Moreover, the testmodel is augmented by the test verdict analysis, using dedicated feature extraction and requirements fulfilment elements[7] from the associated SimulationX Modelica library, collected in the orange *Evaluation* part. Finally, these ready to use simulation models are synchronized with the digital thread database using the SimulationX plugin. With the defined and established links along the process and end to end traceability from requirement to virtual verification (simulation) result is deployed.

In a last step closing the loop shown in Figure 6, the test results are synchronized with the requirements management system DOORS using the corresponding plugin as shown in Figure 5, thus realizing a (simple) test management and tracking system. Figure 10 illustrates the verification of the ESR using the discussed thermal management 1D simulation model and demonstrates, how traceability between the verification results of the braking performance 1D model and the requirements is established in this proof of concept study.

4 Discussion of Decision Making

The demonstrated requirements-based early stage architecture performance validation process is expected to bring significant benefits to daily engineering and design efforts. Developing and certifying commercial vehicle components is challenging, in particular in the demonstrated example of safety critical elements like braking systems. Because of the complicated interrelations between the various systems, sub-systems and components, new techniques for digital continuity, reusability and traceability of development artifacts need to be applied, to handle the complexity throughout the development cycle. With the MBSE methodology, a holistic view and tracking along the increasing maturity level of the system of interest becomes possible in real life applications. Based on the natural, structured R-F-L-P steps, a complex cyberphysical system with multiple hierarchy levels and tight interactions with other systems, gets feasible to manage. Applying simulation results from multilevel models (from simple 0D to 1D/3D) for integrated decision making gives confidence on every decision gate from one domain to the other, throughout the early stage architecture design and verification process. This incremental approach allows to cascade and derive lower level, more detailed requirements on sub-systems and components based on the initial, top level stakeholder needs on product level.

However, it should be noted the introduction and deployment of this process will have a considerable impact on current workflows and will necessitate changes in both processes and personnel (mindset). Apart from these general side effects, the key attributes and considerations from industrial users perspective are summarized below as they share positive impact on engineering and design processes, contributing to a more efficient and systematic working environment.

Benefits in Daily Engineering and Design Tasks: By preventing redundancy and omissions in requirements and functions while ensuring traceability, engineers will have clearer criteria for their work. This will reduce ambiguity in communication and facilitate collaboration among team members.

[7] providing a "passed", "failed" or "undefined" result depending on specified conditions

DOI
10.3384/ecp21793

Modeling Guidance: Guidelines have been developed and need to be applied to map SysML models to simulation requirements. As the SysML language is "method agnostic", the degree of freedom for the systems engineer creating the logical structure needs to be limited to enabled the seamless reuse of information in the subsequent, physical model-based design. These guidelines enhance the consistency of modeling efforts for architecture design and performance validation, supporting effective decision making.

Complexity Resolution and Consistency in Decision Making: By structuring complex and diverse requirements to build the system architecture, it becomes possible to analyze the system before detailed design. This approach allows for easy tracking of components affected by changes and provides linkage information for impact analysis. Furthermore, decision making based on structured information and relationships during the MBSE process supports objective and consistent decisions, independent of the decision maker's intent or the engineer's expertise.

Role of Digital Thread: The digital thread plays a crucial role in systematically connecting information through model-based approaches, enhancing the overall understanding of automotive systems through integrated analysis. Furthermore, when multiple personnel can collaborate via a database-driven digital thread, it facilitates more efficient teamwork and information sharing.

5 Conclusions

This study proposes a Model-Based Systems Engineering (MBSE) methodology, extended to and integrated with model-based design activities, as a means to address the complexities involved in vehicle design and validation, specifically within the context of performance validation for braking systems at early stages of vehicle development. The implementation of digital threads has facilitated the integration of requirements, abstract architecture models, and 1D physical simulation models. The significance of this research and proof of concept application, from an industrial early adopter perspective, can be defined as follows:

- The methodology provides an integrated and visual modeling environment for vehicle development researchers, demonstrating how MBSE-based, design and validation processes enhance quality and reliability.

- By connecting requirements, architecture-, and 1D simulation models through cross-domain digital threads, the methodology enables more efficient workflows. Notably, the internal block diagrams of SysML architecture models can be directly transformed into Modelica blocks as components of 1D models, thereby reducing the inconvenience of reworking defined elements from SysML in the performance model, thus improving overall process efficiency.

- The verification results of requirements achieved with the 1D simulation models can be directly validated and documented in the requirements management software, establishing a closed-loop requirements-based verification framework.

However, the applied MBSE methodology is currently implemented and demonstrated on the validation of braking performance at the early stage of vehicle development, and the integration with other performance domains and software remains a significant challenge. Particularly, despite being at the early stage, verification through 3D analysis presents constraints that hinder the direct reusage of SysML architecture modeling results, similar to the 1D simulation domain. Nonetheless, in other performance areas where verification is conducted using software, based on the Modelica language, it is anticipated that implementing connections through digital threads, as demonstrated in this study, will lead to more efficient operational directions. A concrete idea in this direction is the seamless reuse and continuous integration of architecture, simulation models and scenarios in the virtual verification of Electronic Control Units by SiL and HiL[8] testing methods. Further automation of test model generation, based on available SysML descriptions like sequence- or parametric diagrams, will add on top of the aforementioned improved usability and efficiency.

Additionally, the utilization of automated parameter value selections in heterogenous verification environments (like co-simulation of Modelica and CarMaker) becomes possible and can significantly enhance development efficiency, (Lu et al. 2021). This would leverage the potential of our digital thread integration approach to streamline processes and improve outcomes in vehicle system design and validation at HMC and similar industrial applications.

References

Aleksandraviciene, Aiste and Aurelijus Morkevicius (2018). *NoMagic Magic Grid - Book of Knowledge*. 1st ed. Vitae Litera.

Cederbladh, Johan et al. (2024). "Correlating Logical and Physical Models for Early Performance Validation - An Experience Report". In: *IEEE Systems Conference SYSCON2024*. IEEE.

Hause, Matthew et al. (2006). "The SysML modelling language". In: *Fifteenth European systems engineering conference*. Vol. 9, pp. 1–12.

Lu, Jinzhi et al. (2021). "Model-based systems engineering toolchain for automated parameter value selection". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.4, pp. 2333–2347.

Maro, Salome, Miroslaw Staron, and Jan-Philipp Steghöfer (2017). "Challenges of establishing traceability in the automotive domain". In: *Software Quality. Complexity and Challenges of Software Engineering in Emerging Technologies: 9th International Conference, SWQD 2017, Vienna, Austria, January 17-20, 2017, Proceedings 9*. Springer, pp. 153–172.

[8] Software- and Hardware-in-the-Loop

Modelica Association (2014-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2021). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.5*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.5/MLS.html.

Modelica Association (2022). *System Structure and Parameterization Version 1*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Object Management Group (2022). *OMG Systems Modeling Language Version 1*. Tech. rep. Massachusetts: Object Management Group. URL: https://www.omg.org/spec/category/systems-engineering/.

Open Services Project (2021). *Open Services for Lifecycle Collaboration Version 3*. Tech. rep. Open Services Project. URL: https://open-services.net/.

Peraldi-Frati, Marie-Agnès and Arnaud Albinet (2010). "Requirement traceability in safety critical systems". In: *Proceedings of the 1st Workshop on Critical Automotive applications: Robustness & Safety*, pp. 11–14.

DOI
10.3384/ecp21793

A Study on Model-Based Thermal Management Systems Architecture Modeling and Energy Efficiency Prediction of Fuel Cell Electric Vehicles

Junbeom Lee[1] Kwonhee Suh[2] Juhyeong Park[1]
Donkyu Joo[1] Junghun Yun[1] Daeoh Kang[1*]
[1]iVH, Republic of Korea, {jblee, jhp, dkjoo, jhy, bigfive*}@ivh.co.kr
[2]KIA Corporation, Republic of Korea, lastadam@kia.com

Abstract

The purpose of this study is to predict the energy efficiency of Fuel Cell Electric Vehicles (FCEVs) based on the configuration of the Thermal Management System (TMS). The energy efficiency of an FCEV is closely tied to the effective thermal management of the electric powertrain. Therefore, this paper investigates TMS modeling for FCEVs and multi-physics modeling of FCEVs. The main modules of the target multi-physics FCEV model include the vehicle dynamics model, thermal management system, electric powertrain and controller. The main research focus is the water-cooled TMS architecture. The main thermal management components (heat-generating components) include the fuel cell, battery, motor and BOP (Balance of Plant). The model was developed using Modelica and used to predict the energy efficiency under various driving conditions (extremely cold, moderate and extreme heat) and driving conditions.

Keywords: Vehicle System Engineering, Thermal Management System, Fuel Cell Electric Vehicle

1 Introduction

The transition towards eco-friendly and sustainable transportation has accelerated the development of alternative powertrain systems, such as Fuel Cell Electric Vehicles (FCEVs). FCEVs, which use hydrogen fuel cells as their primary energy source, offer the advantages of high efficiency and zero-emission operation. However, they present critical challenges in terms of thermal management. Efficient thermal management is essential to maintain the optimal operating of fuel cells and other key components. This ensures passenger comfort, extends vehicle life, and optimizes overall energy consumption.

Thermal management in FCEVs is highly complex, as it must account for varying power demands, dynamic environmental conditions, and the unique operational characteristics of fuel cells, which generate both heat and water as byproducts. A robust TMS is necessary to effectively distribute and dissipate heat, minimizing energy losses and optimizing performance under diverse driving conditions. To optimize the TMS architecture of FCEVs, it is crucial to develop multi-physics models integrating thermal, electrical, and dynamic systems.

This paper proposes a Modelica based modeling approach of TMS architecture and the prediction of FCEV performance under varying conditions. Model-based approaches provide a structured and predictive method for designing and analyzing the TMS architecture of FCEVs, especially in the early stages of design. Modelica, a non-proprietary, object-oriented, equation-based modeling language, is particularly suitable for this purpose. Modelica supports the integration of thermal, electrical, and fluid systems into unified multi-domain models, enabling the analysis of complex interactions within FCEV TMS and the prediction of energy efficiency using a single simulation framework.

In this study, we aim to develop a model-based approach using Modelica to model the TMS architecture of FCEVs and accurately predict energy efficiency under various operating conditions. The objective of this research is to provide a systematic method for TMS design and control that enhances the energy efficiency and reliability of FCEVs.

2 Methodology

In this paper, vehicle dynamics, electric powertrain, and TMS models are developed to predict the performance changes of FCEVs by TMS architecture. The vehicle dynamics model was built using the Vehicle Dynamics Library from *Modelon AB*, while the electric powertrain model was based on *Modelon AB*'s Electrification Library. The TMS was modeled using the Thermal Systems Library from *TLK-Thermo GmbH*. The integrated model was constructed using Dymola from *Dassault Systèmes*.

2.1 Vehicle Model

The vehicle model is organized in a hierarchical structure, and includes detailed sub-systems such as the Chassis System, Thermal Management System, Powertrain System, and Brake System.

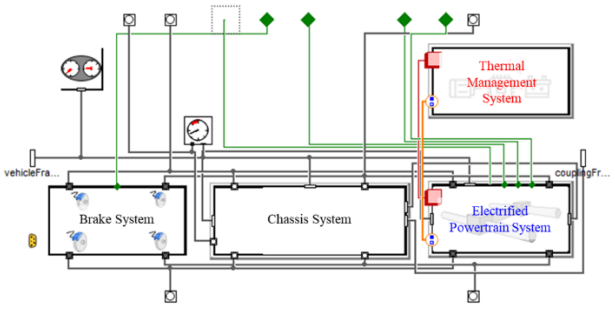


Figure 1. Vehicle Model

Each sub-system exchanges Dynamics, Electrical, and Thermal signals. The Powertrain System transfers the heat generated during operation to the TMS, and the powertrain model transmits both the power and movement produced by the Fuel Cell, battery, and motor to the Chassis System.

2.2 Chassis Model

The chassis model consists of front and rear suspension, frame, wheel, and body models. The suspension model transmits vibrations from the wheel model to the body. It is constructed using a multi-body dynamics approach. However, since this model is designed for energy efficiency prediction, the bushing models are simplified and replaced with joint models.

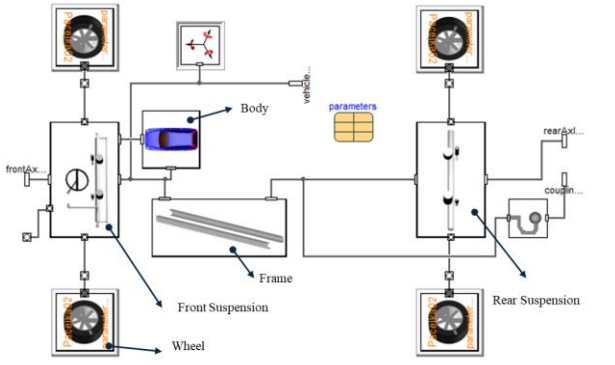


Figure 2. Chassis Model

2.3 Electrified Powertrain

The main components of the electrified powertrain model are the battery model, hydrogen fuel cell, and motor.

The battery model is parameterized with cell capacity, open-circuit voltage curves, internal resistance values, and cell configurations. It generates power while producing heat during operation.

The hydrogen fuel cell model is built based on the battery model. It calculates hydrogen consumption based on power demand and shares a similar structure with the battery model. By connecting a load model that simulates the power consumption of the Balance of Plant (BOP), the total output (gross output) and net output can be calculated. Like the battery model, the fuel cell model generates heat during power production.

The motor model incorporates torque maps and efficiency maps as parameters and includes a maximum torque limit. It generates driving torque according to the vehicle's speed requirements by receiving power from the battery model and hydrogen fuel cell model. The generated driving torque is transmitted through the driveline to control the vehicle's speed. Heat is produced due to losses occurring during torque generation.

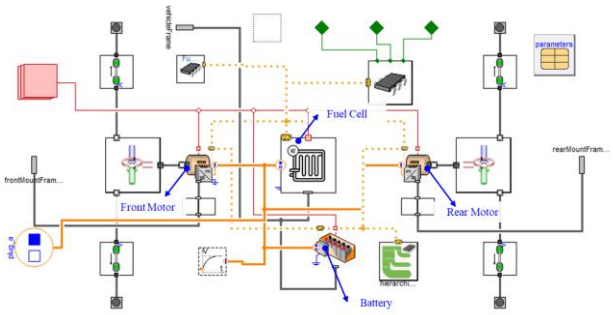


Figure 3. Electrified Powertrain Model

2.4 Thermal Management System

In this vehicle model, the primary heat-generating components are the hydrogen fuel cell, battery, and motor. Due to the different optimal operating temperatures of each component, the cooling cycles are configured independently. In addition to the 4 major heat-generating components mentioned above, other components requiring thermal management include the Low Voltage DC-DC Converter (LDC) for low-voltage equipment in the vehicle, the Balance of Plant (BOP) responsible for managing fuel, moisture, air, and power distribution, and the Fuel Cell DC-DC Converter (FDC), which ensures stable delivery of electricity generated by the hydrogen fuel cell stack to the high-voltage battery for charging or to the inverter.

Table 1. List of heat generating components by cycle

Cycle Name	Heat Components Name
Battery Cycle	Battery
PE Cycle	Front PE

DOI
10.3384/ecp217103

	Rear PE
	BOP
	LDC
	FDC
FC Stack Cycle	Fuel Cell

Each cooling cycle operates independently, forming a total of three separate cycles. There is no cross-flow or mixing between the coolant of these cycles. Different cooling liquids should be used depending on operating conditions (voltage and current), but this model is simplified by using a single cooling liquid for all cycles.

Table 2. Coolant by cooling cycle

Cycle Name	Liquids
Battery Cycle	Ethylene Glycol 50%
PE Cycle	
FC Stack Cycle	

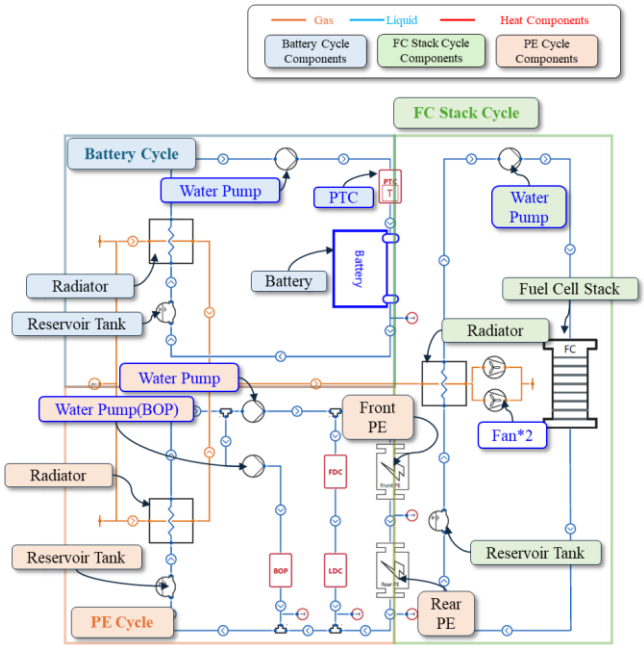


Figure 4. TMS Architecture

The actuators for each cycle are as follows, and they consume power during operation. Consequently, the consumed power is accounted as a load, which reduces the battery's available power.

Table 3. List of Actuators by Cooling Cycle

Cycle Name	Actuator
Battery Cycle	Water Pump
	PTC
PE Cycle	Water Pump
	Water Pump(BOP)
FC Stack Cycle	Water Pump
Common	Fan

2.5 Controller

To meet the target temperature of each cycle, the actuators within each cycle must be operated based on the state variables of the cycle and its components. Accordingly, a controller model is required, where the controller adjusts the actuators according to the temperatures of the components within the cycle. The actuators in each cycle control the water pump based on the outlet temperatures of individual components and ambient temperature. In the battery cycle, the PTC (Positive Temperature Coefficient heater) is additionally controlled to heat the battery as needed. The fan adjusts its volumetric flow rate using the outlet temperatures of the battery, fuel cell stack, PE, LDC, BOP, and the ambient temperature.

2.6 Scenario

The Chassis, Electrified Powertrain, Thermal Management System, and Controller configured above are integrated into the Vehicle Model. In order to predict the power consumption of a vehicle model, the vehicle model is driven according to the Drive Cycle, and the Drive Cycle utilized is the Worldwide harmonized Light vehicles Test Cycle (WLTC). The WLTC (Worldwide harmonized Light vehicles Test Cycle) is a globally standardized driving cycle used to measure the fuel consumption, CO_2 emissions, and pollutant emissions of passenger cars and light-duty vehicles. It was developed by the UNECE (United Nations Economic Commission for Europe) as part of the WLTP (Worldwide harmonized Light vehicles Test Procedure) framework. In this study, to reflect the limitations of the electrified powertrain in the vehicle model, the drive cycle was modified with an upper speed limit of 120 kph, and performance predictions were conducted based on the adjusted cycle. In this scenario, the WLTC was repeated for 2 cycles.

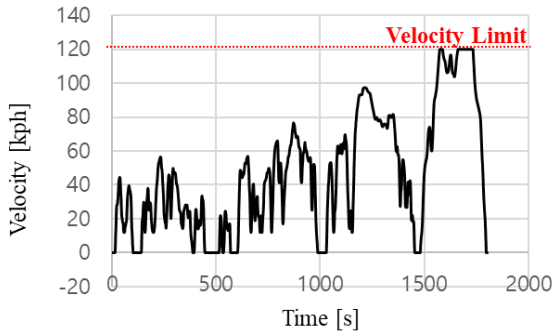


Figure 5. WLTC speed profile reflecting vehicle driving limit performance

3 Result

- Components Temperature(Extreme Hot)

In the extreme hot scenario, it was observed that the temperature of the Front and Rear PEs requires control for only a short duration, yet it remains within the target temperature range. The FC stack also showed a short control duration, successfully converging within the target temperature range. For the battery, due to the minimal operational time governed by the control conditions (drive control), heat generation was negligible, resulting in insignificant temperature changes.

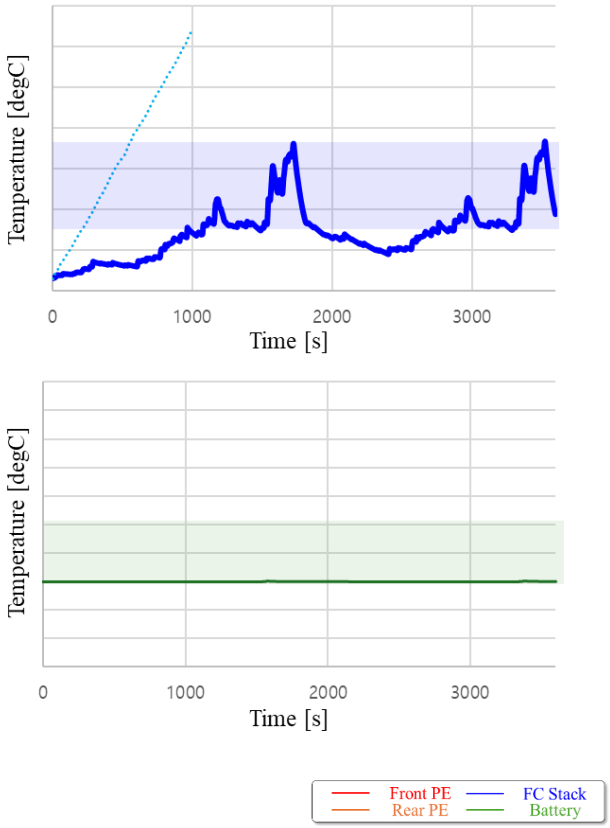
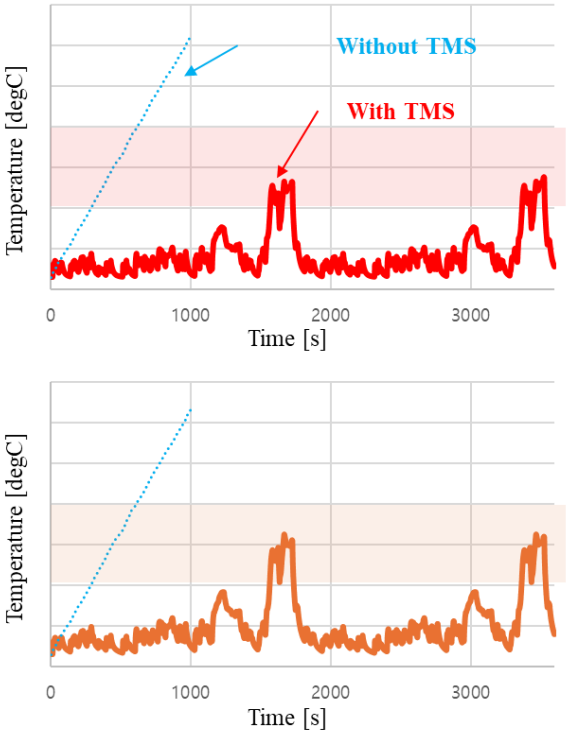
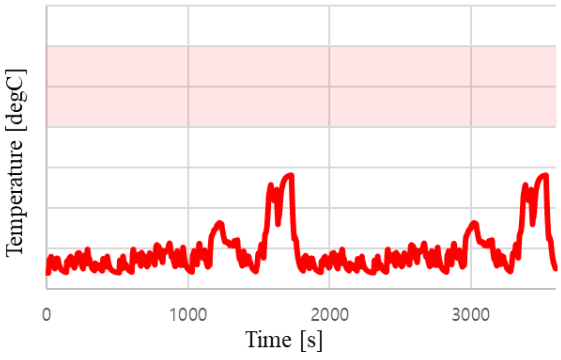


Figure 6. Components Temperature in Extreme Hot



- Components Temperature(Moderate)

Under moderate conditions, it was confirmed that the temperature levels of the components are lower compared to extreme heat conditions. Rear PE is located further back than the Front PE, resulting in slightly higher temperatures, but it still did not reach the optimal temperature. FC Stack reached the optimal temperature range during certain high-speed sections, indicating that its temperature is being effectively controlled by the controller. Meanwhile, the battery exhibited very low heat generation due to driving conditions, similar to extreme heat conditions.



DOI
10.3384/ecp217103

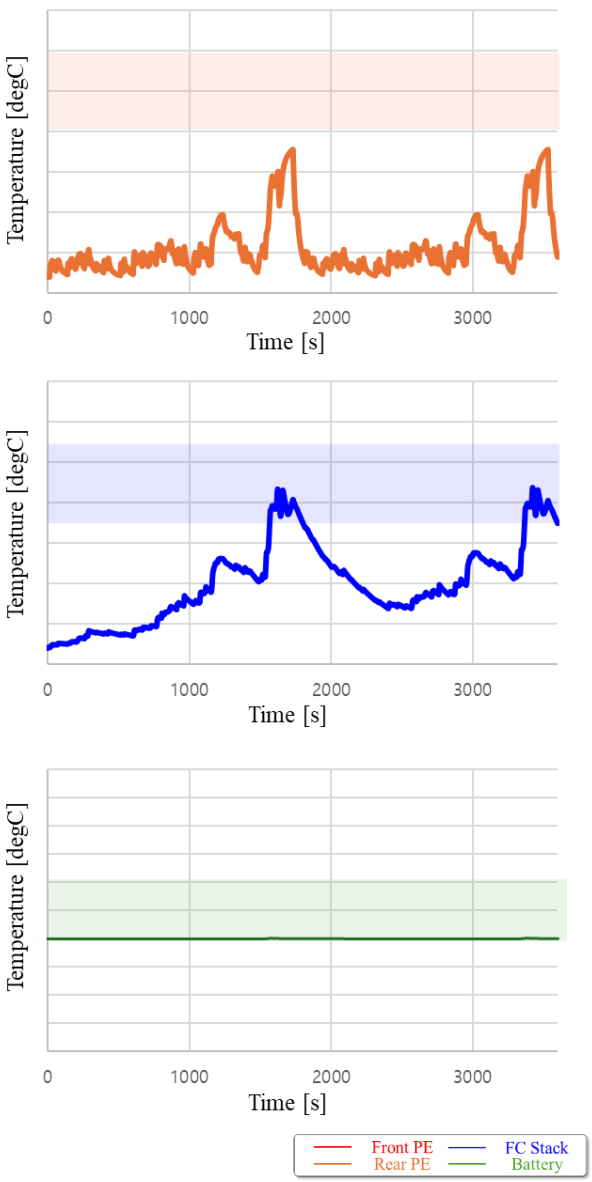


Figure 7. Components Temperature in Moderate

- Components Temperature(Extream Cold)

Under extreme cold conditions, it was observed that the Front PE, Rear PE, and FC Stack could not reach their target temperatures solely through self-heating. Particularly for the FC Stack, failing to achieve its target temperature can lead to overall efficiency degradation, reduced lifespan, and decreased power output. Therefore, it is necessary to consider the integration of forced heating components, such as a COD heater and air heater, within the system architecture.

The battery, equipped with a PTC heater, raises the cooling water temperature within the battery cycle when the ambient temperature is low. After reaching the target temperature, temperature control is carried out to maintain the optimal temperature range. This demonstrates the effective temperature management within the battery cycle under extreme cold conditions.

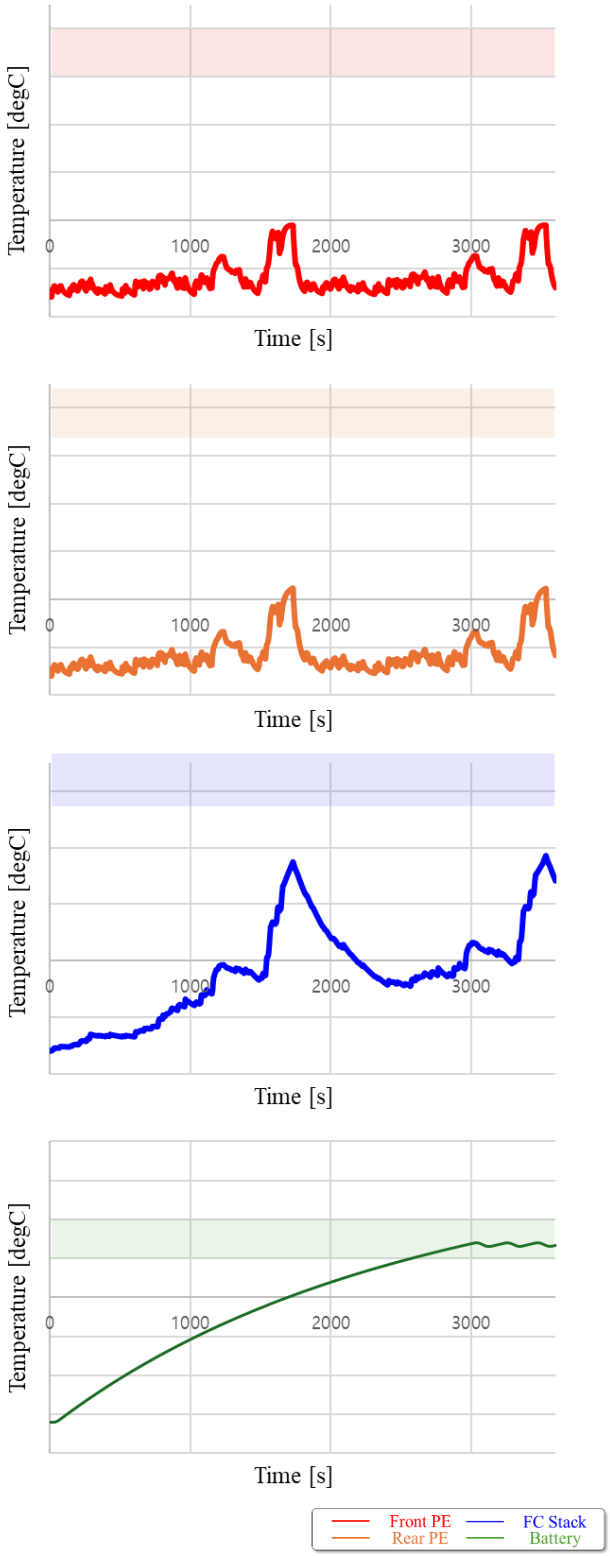


Figure 8. Components Temperature in Extreme Cold

- Hydrogen consumption

The model developed in this study does not include parameters that can reflect individual component performance changes based on their temperatures. Therefore, the changes in vehicle system performance

due to component temperatures are predicted through variations in hydrogen consumption.

Under extreme hot conditions, the presence of a TMS leads to increased hydrogen consumption due to the operation of fans and water pumps to maintain component temperatures within the optimal range. In this study, as efficiency data based on component temperatures was not considered, it was observed that hydrogen consumption is higher with the application of TMS compared to when TMS is not applied.

Even under moderate conditions, the presence of a TMS results in increased hydrogen consumption, as fans and water pumps are operated to control component temperatures within the optimal range. However, compared to extreme heat conditions, the duration of temperature control to maintain the target component temperatures is shorter, leading to lower hydrogen consumption.

Under extreme cold conditions, hydrogen consumption increased significantly compared to extreme heat and moderate conditions. This is attributed to the substantial rise in high-voltage power usage caused by the operation of the PTC heater.

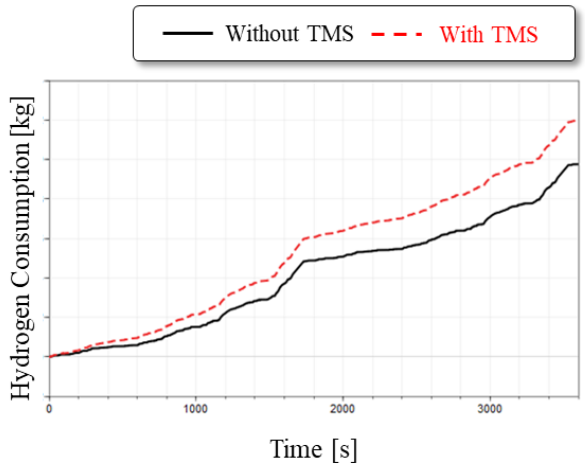


Figure 9. Hydrogen consumption by With/Without TMS in Extreme cold conditions

Table 4. Hydrogen consumption according to ambient conditions and TMS

Ambient Condition	TMS	Hydrogen Consumption
Extreme Hot	X	+
	O	+++
Moderate	X	+
	O	++
	X	+
Extreme Cold	O	++++

4 Conclusion

This study developed a model-based approach for modeling the TMS architecture and predicting the energy efficiency of FCEVs using Modelica. By integrating vehicle dynamics, thermal management, and electric powertrain systems, this model highlights the importance of thermal management in optimizing system performance under various environmental conditions. The study results indicate that TMS operations, influenced by external conditions, activate components such as fans, water pumps, and PTC heaters, leading to variations in hydrogen consumption. Under extreme heat conditions, TMS increases hydrogen consumption as it controls component temperatures, while in moderate conditions, the impact is relatively smaller due to shorter control durations. In contrast, under extreme cold conditions, hydrogen consumption increases significantly as the PTC heater operates to achieve target temperatures, emphasizing the necessity of effective thermal management in extreme conditions.

However, the current model does not include performance change data for individual components based on temperature variations, preventing direct assessment of TMS-related energy efficiency changes. This study underscores the importance of advanced TMS architectures for maintaining optimal component temperatures and improving energy efficiency. Future research should focus on parameterizing performance changes in individual components caused by temperature variations and exploring alternative thermal management strategies that can reduce energy losses while maintaining system stability.

Acknowledgements

This work has been supported by KIA Corporation.

References

Modelica Association (2023-03). *Modelica – A Unified Object Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association.
URL: https://specification.modelica.org/maint/3.6/MLS.pdf

Stellato, M., Bergianti, L., Batteh, J. (2017). *Powertrain and Thermal System Simulation Models of a High Performance Electric Road Vehicle* 12th International Modelica Conference, Prague, Czech Republic, pp. 171-180, May 15-17,

Batteh, J., Gohl, J., Sureshkumar, C.(2014). *Integrated Vehicle Thermal Management in Modelica: Overview and Applications* 10th International Modelica Conference, Lund, Sweden

Modelon AB (2021). *Vehicle Dynamics Library Version 4.0.*

Modelon AB (2021). *Electrification Library Version 1.7.*

TLK-Thermo GmbH(2022). *Thermal Systems Library Version 1.9.*

DOI
10.3384/ecp217103

Digital Human Body Model for Occupant Monitoring System

Man Yong Han[1] Yong Ha Han[2] Hyung Yun Choi[3]

[1, 2] Hyundai Motor Company, Republic of Korea,
{myhan1037,yongha}@hyundai.com

[3]Dept. of Mechanical and System Design Engineering, Hongik Univ.
Korea, hychoi@hongik.ac.kr

Abstract

Occupant monitoring systems have been developed and used for Autonomous Driving (AD) level 3+. These occupant monitoring systems have limitations in accuracy and measurement items. To compensate for this, a Digital Human Body Model (DHBM) based on the Modelica language is developed, and its features are introduced. Inverse Kinematics (IK) and Inverse Dynamics (ID) DHBMs are interlocked with the occupant monitoring system to increase measurement accuracy and calculate various information such as motion sickness and fatigue. However, simulation of occupant behavior prediction is impossible. Forward Dynamics (FD) DHBM is a model that implements the characteristics of the live human studied through experiments and can predict occupant behavior. However, parameter verification is necessary to trust the results of FD DHBM. It is developing real-time validation and parameter update algorithms for FD DHBM using occupant monitoring data, which are expected to be available in various fields such as comfort and safety.

Keywords: Digital Human Body Model (DHBM), Occupant Monitoring, Autonomous Driving (AD), Active Safety System (ASS)

1 Introduction

For autonomous driving level 3+, technology development and practical application of sensors, Advanced Driver-Assistance Systems (ADAS), and Active Safety Systems (ASS) are underway. ASS increases the pre-crash phase before impact and minimizes crash severity. Due to ASS, occupants have time to react, and physical characteristics such as sitting posture and muscle strength can affect their behavior. As a result, research on occupant behavior in driving and pre-crash phases has become necessary. Virtual simulation is needed to research various vehicle and occupant conditions effectively. However, Human Body Models such as the Global Human Body Model Consortium (GHBMC) and Total Human Model for Safety (THUMS) for virtual simulation are specialized in crash simulation and injury prediction and validated for PostMortem Human Subjects (PMHS) -based in-crash (John J. Combest 2018; Toyota Motor Corporation 2021). In addition, because it is an FE-based model, the CPU cost is high and difficult to simulate for the long term, making it unsuitable for long-term passenger behavior research.

The Occupant Model For Integrated Safety (OM4IS) consortium and several studies have conducted occupants' behavior studies based on volunteer experiments (Kirschibichler 2014; Huber 2015). OM4IS measured and analyzed occupant kinematics assuming the operation of the ASS. With the recent advances in posture detection technology, in-cabin occupant monitoring systems are installed in vehicles (e.g., Mobis DSM). In PC/mobile environments, machine learning-based human body monitoring systems (e.g., Google MediaPipe, Yolo v4) are being developed and are expected to be applied to actual vehicles in the future under Software Defined Vehicle (SDV) environments. Occupants' actual status and movement can be monitored and used for safety, comfort, etc. However, since the measured occupant movement is displacement-based data, the precision is relatively low, and it is impossible to predict the occupant behavior; a method to supplement this is necessary. Methods using mathematical filters (e.g., Kalman filters) can eliminate noise, but reflecting human characteristics and predicting behavior is still challenging. A Digital Human Body Model (DHBM) based on the Modelica language with low CPU cost and acausal characteristics was developed to supplement the occupant monitoring system.

Modelica language can build a multi-body system model and has acausal characteristics, which will be described later in Inverse Kinematics (IK), Inverse Dynamics (ID), and Forward dynamics (FD) models, which can be built based on a single basic model. It has the advantage of quickly conducting long-term simulations with low CPU costs. Moreover, since the Modelica language supports the Functional Mock-up Unit (FMU), it can conduct various simulations with other S/Ws that support the Functional Mock-up Interface (FMI). The control unit of the DHBM used the blocks module of Modelica (e.g., Continuous, nonlinear, table), and the human model part used the mechanic's modules such as multibody, rotational, and translational modules. It was conducted using Open-Modelica and ESI's SimulationX.

2 DHBM for Occupant Monitoring

In an environment where occupants are monitored, a DHBM for improving accuracy and calculating additional information has been created for more diverse use.

2.1 Base of DHBM

The basic DHBM consists of the pelvis as the root and is deployed to the end of each body (Head, Upper-, Lower-extremity). The body segments are set to rigid-body, and each segment contains body properties such as center of mass, mass, and inertia information. The body segments consist of the head, neck, trunk, upper arm (UA), forearm (FA), upper leg (UL), lower leg (LL), and foot. Depending on the occupant monitoring information or needs, the trunk shall be two (UT, LT) or three (UT, CT, LT). An articulated joint connects each segment, and each joint except the elbow (1D) is composed by connecting three 1D revolute joints in series. In the case of the lower extremity, the joints are fixed as necessary to prevent movement. Properties of body segments and joint positions are based on values corresponding to the GHBMC AM 50th %tile or the subject's body dimensions. These values are fixed before the analysis begins.

2.2 Seat-Human Interaction & Seat Belt

In situations where lateral acceleration occurs, such as evasive lane change, seat cushions, and bags affect human body behavior. Conversely, in emergency braking situations, the effect of the seat back is eliminated, and the seat cushion and seat belt affect occupant movement. A seat-human interaction module and a seat belt module were developed to describe this.

For seat back-human interaction, contact force was calculated using an elastic-gap block. For seat cushion-human interaction, contact force was calculated using a foam characteristic table and kinematics between the cushion and lower body and applied to the lower body.

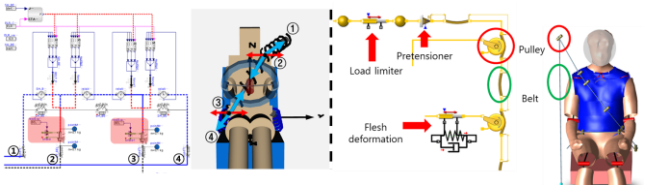


Figure 1. Seat belt model in Open-Modelica (left), and in SimulationX (right)

The seat belt was composed of three parts and four contact points, and the belt loosening amount was measured with a dummy spring, as shown in Figure 1. Based on this, the belt tension force was calculated, and the belt tension force was applied to the trunk. In the lateral direction of the belt, lateral friction force was applied to the DHBM using a sliding part. In SimulationX, seat belts were implemented using a belt, belt preset, and pulley block, as shown in Figure 1.

2.3 Inverse Kinematics (IK) of DHBM

The occupant monitoring information used in this study is the 3D position of the main landmarks and joints of the human body and includes a certain level of error. If IK analysis is performed using each position as an input, the length of each segment is continuously changed due to errors included in the input. This conflicts with the segment length of the dummy set in a rigid body, making normal IK simulation impossible. To solve this problem, a spring-damper connects each input point and the corresponding point of the human model to function as a buffer. The angle of each joint can be calculated through the IK DHBM, and acceleration that is difficult to calculate mathematically other than displacement and angle, it is possible to calculate angular velocity. However, the active torque of each joint cannot be calculated through the IK DHBM.

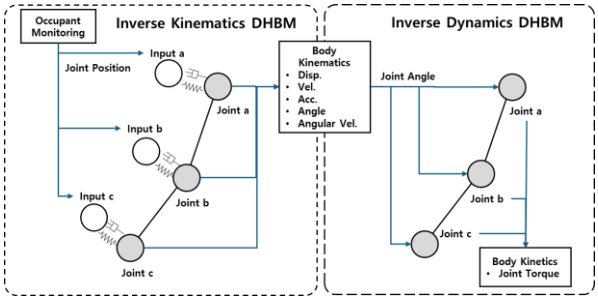


Figure 2. Conceptual Diagram of IK, ID DHBM

2.4 Inverse Dynamics (ID) of DHBM

To obtain the DHBM's kinetic information, an ID DHBM was constructed using the IK model's joint angle as the input. Each joint's kinetic information (joint torque) can be calculated using the acausal characteristics to satisfy the input kinematic (joint angle). This is the same as the occupant monitoring results. All kinematic and kinetic information can be obtained.

3 DHBM for Simulation

IK, ID DHBM can describe and analyze the behavior of passengers based on the measured (or under measurement) monitoring information, but it is impossible to predict the behavior of occupants. The FD DHBM, including the characteristics of a living human, can predict the occupant's behavior according to the vehicle's movement. Live humans exhibit different behaviors from the Anthropomorphic Test Device (ATD) or PMHS under the same external force conditions (Beeman 2012). This is due to the characteristics of live humans, and its typical characteristics include muscle contraction (co-contraction) and awareness (Kirschibichler 2014). Various

DOI
10.3384/ecp217109

characteristics of the human body needed for calculating occupant behavior and their application are as follows:

3.1 Active Joint Torque

The live human generates muscle force through the contraction and relaxation of various muscles, which is applied as a torque to rotate the body segment. Among the methods of modeling muscle contraction in the human body, calculating muscle contraction and muscle force according to the activation level by modeling all muscles is also used. This requires an excessive amount of computation. To speed up computation, the concept of active joint torque (equivalent torque applied to each joint) was introduced without calculating the contraction force of each muscle. The active joint torque actuator was configured based on the PID closed-loop control, and the change in the joint angle (or body segment angle) compared to the reference was set as an error. And the torque to be applied to each joint was set as an output. Choi (2016) verified an active elbow model reflecting active joint torque and co-contraction.

3.2 Co-contraction

Beeman (2012) conducted low-speed sled tests on ATD, PMHS, and human volunteers. The volunteer showed behavior between the PMHS and the ATD and differences in behavior according to the co-contraction level. It showed more similar behavior to ATD in the braced condition (high level of co-contraction) than in the relaxed condition (low level of co-contraction). As the co-contraction level increases, the human joint stiffs and behaves like a rigid body. It is applied to a human model by damping torque, and an example is shown in Figure 3.

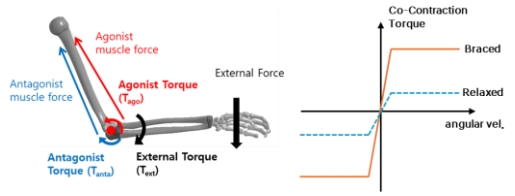


Figure 3. An overview of co-contraction (left), an example of co-contraction torque applied to the model (right)

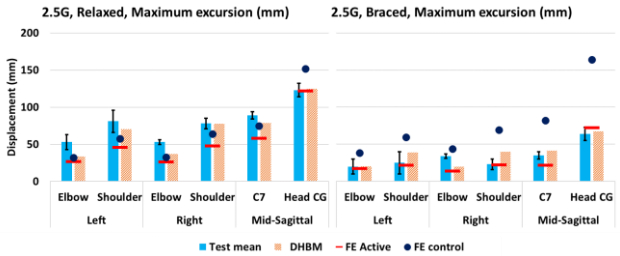


Figure 4. Validation result of low-speed sled test

A low-speed sled test confirmed DHBMs that reflect co-contraction (Beeman 2012). Compared to the results of the same validation of the GHBMC FE active model (Devane 2019), both relaxed and braced conditions showed high similarity except for the shoulder at the braced condition, as shown in Figure 4.

3.3 Awareness

OM4IS measures the behavior of passengers according to their awareness of evasive single-lane change and emergency braking situations assuming an autonomous driving situation. In the unaware condition (low level of awareness), in which passengers do not know the situation, they showed large behavior with a neural delay time longer than the informed condition (high level of awareness) in which all situations are known. The higher the awareness level, the shorter the neural delay time, so the torque is rapidly actuated, and the amount of excursion decreases. In the DHBM, awareness was reflected by not applying the active joint torque calculated during the neural delay time to the torque actuator (Han 2019).

3.4 Response Strategy

Vehicle occupants have two strategies: driver behavior and passenger behavior. In the case of the driver, to perform driving, the driver moves with a strategy of fixing the head position as much as possible and adjusting the gaze in the direction of progress. On the other hand, the passenger has a strategy of resisting the body more passively without considering the head position (Zikovitz 1999). For passengers, the same strategy as the driver can be taken, which can be viewed as a characteristic according to inter- and intra-subject variations. To reflect this, DHBM's PID control error used both the weighted joint and segment angles (Son 2023).

3.5 Forward Dynamics (FD) DHBM

The validation of critical parameters is necessary for the DHBM to describe occupants' behavior. The main parameters are PID gains, co-contraction, awareness level, and strategy ratio. The optimization technique can find parameters that show the same behavior as the subject test data. Using the OM4IS evasive lane change results, optimization targeting each subject's behavior and overall average behavior was conducted and reviewed (Han 2019). By applying floor input (or pelvis input) to the validated model and simulating it, the behavior of the human body according to a given vehicle movement can be predicted. Through this, it is possible to predict the occupants' behavior and use the result to optimize vehicle control. However, in the case of live humans, since inter- and intra-subject variations exist, the accuracy of the simulation may decrease as it deviates from the validation condition.

4 Use Case

IK DHBM can improve the accuracy of body kinematic information in occupant monitoring systems and calculate more information. For example, head translational acc., rotational vel. can be calculated, and motion sickness can

be calculated using models such as Motion Sickness Incidence (MSI).

ID DHBM is used to calculate kinetic information using occupant kinematic information. Active joint torques, the simulation results of ID DHBM, can be viewed as the energy needed to maintain posture. The regression equation obtained through the volunteer experiment can be converted into the Metabolic Equivalent of Task (MET) and kCal units, and the muscle fatigue of passengers caused by maintaining posture while driving can be checked.

FD DHBM uses vehicle movement to predict occupant kinematics and kinetics. Since occupant monitoring information is not needed, it is possible to simulate occupant behavior according to vehicle kinematics quickly. For example, it is possible to decide which ASS strategy is more dangerous to passengers under various conditions.

5 Conclusion

This paper introduces the types and characteristics of passenger monitoring systems and available Modelica language-based DHBMs. In addition, it presents the main characteristics of the human body found through ATD, PHMS, and live-human subject experiments and how to implement them in FD DHBM. The main characteristics of DHBM are as follows.

1. The main characteristics of the human body include active joint torque, co-contraction, awareness, and response strategy. These characteristics must be reflected in the human model to express the human body's behavior well in DHBM.

2. DHBMs using the Modelica language have a lower CPU time than FE models, can be simulated for a long time, and can be linked to the occupant monitoring system. It also supports FMU with FMI, enabling co-simulation with various programs.

3. The IK and ID HBMs enhance the accuracy of occupant monitoring information and calculate the kinematics and kinetics of occupants to enable quantitative evaluation, such as motion sickness and muscle fatigue. However, they have a disadvantage in using already measured (or being measured) information, and it is impossible to simulate behavior prediction.

4. FD HBM has the advantage of implementing co-contraction, awareness, and individual size, which are characteristics of the human body, so that occupant kinematics and kinetics can be calculated and predictive simulations can be performed. However, the validation of the model parameter is required, and the reliability of the FD HBM is guaranteed only for the conditions under which the model parameter validation is performed.

The currently developed DHBM based on the Modelica language has the advantage of quantitatively analyzing and utilizing occupant behavior and is being reviewed for use in more diverse fields. To improve the limitations of the FD model, an algorithm for periodic validation and parameter updates is being developed using the IK model or occupant monitoring information. It is expected that this will increase the accuracy of predicting passenger behavior.

References

Byeong Lak Son et al. (2023). “Simulation of Occupant Head Roll Motion Using Active Human Body Model”. In: *Trans. Korean Soc. Mech. Eng. C*, Vol. 11, No. 1, pp. 9-19. DOI: 10.3795/KSME-C.2023.11.1.009

Daniel C. Zikovitz and Laurence R Harris (1999). “Head tilt during driving”. In: *Ergonomics*. pp. 740-746. DOI: 10.1080/001401399185414.

Hyung Yun Choi et al. (2016). “Active Elbow Joint Model”. In: *Proceedings of the 1st Japanese Modelica Conference*. May 23-24. pp. 50-54. DOI: 10.3384/ecp1612450.

John J. Combest (2018). “Current Status and Future Plans of the GHBMC”. In: *7th International Symposium: Human Modeling and Simulation in Automotive Engineering.* Carhs GmbH. URL: https://www.ghbmc.com/wp-content/ uploads/ 2019/05/HMS18-18_Combest_Nissan-and-GHBMC.pdf.

Karan Devane et al. (2019). “Validation of a simplified human body model in relaxed and braced conditions in low-speed frontal sled tests”. In: *Traffic Injury Prevention*. pp. 832-837. DOI: 10.1080/15389588.2019.1655733.

Kirschibichler, S. et al. (2014). “Factors Influencing Occupant Kinematics during Braking and Lane Change Maneuvers in a Passenger Vehicle.” In: *IRCOBI Conference*. pp. 614-625. URL: https://www.ircobi.org/wordpress/downloads/irc14/ pdf_files/70.pdf.

Man Yong Han (2019). “Active Human Body Model for Virtual Vehicle Ride Simulation”. Ph.D. thesis. Hongik University, Mechanical Engineering. DOI: 10.23174/hongik. 000000023241.11064.0000288

Philipp Huber et al. (2015). “Passenger kinematics in braking, lane change and oblique driving maneuvers.” In: *IRCOBI Conference.* pp. 783-802. URL: https://www.ircobi.org/ wordpress/downloads/irc15/pdf_files/89.pdf.

Stephanie M. Beeman et al. (2012). “Occupant kinematics in low-speed frontal sled tests: Human volunteers, Hybrid III ATD, and PMHS”. In: *Accident Analysis and Prevention 47*, pp. 128-139. DOI: 10.1016/j.aap.2012.01.016.

Toyota Motor Corporation (2021). “Total Human Model for Safety (THUMS): Revolutionizing Crash Simulation to Support Safe Mobility for All. Development Story: Toyota Motor Corporation. URL: https://www.toyota.co.jp/thums/ contents/pdf/Toyota_THUMS_History_English.pdf.

Modelling And Simulation of a Batch Reverse Osmosis Process Using Modelica

B Sai Mukesh Reddy[1], Mrugesh Joshi[2], Seongpil Jeong[3], Jaichander Swaminathan[1]*

[1]ICER, Indian Institute of Science Bengaluru, India[1] ,{saimukesh, jaichander}@iisc.ac.in

[2]Mechanical Engineering Discipline, IIT Gandhinagar, Gujarat, India[2], mrugesh.joshi@iitgn.ac.in

[3]Division of Energy & Environment Technology, KIST 02792, Seoul, Korea[3,] spjeong@kist.re.kr

Abstract

Batch operation of reverse osmosis (RO) has emerged as a promising strategy for enhancing energy efficiency and reducing fouling in seawater and brackish water desalination applications. This study implements a transient numerical model using Modelica to investigate the behavior of a batch seawater RO (BSWRO) system. The feed solution volume decreases and its salinity increases with time. It is pressurized by pumping product water into the other side of a piston. The model incorporates features such as the feed solution residence time in the inlet and outlet piping and captures the local variation in flux and concentration polarization over the membrane area. The instantaneous power consumption of the high-pressure and circulation pumps is calculated. The ease of adoption of Modelica underscores its utility in simulating complex transient and non-linear phenomena. The developed model can be expanded in the future to answer questions related to optimal control of batch RO systems.

Keywords: Batch Reverse Osmosis, Modelica, mathematical modeling, transient

Nomenclature

Symbol	Name	Units
Q	flowrate	l/h
s	salinity	g/l
t	time	s
P	pressure	Pa
v	velocity	m/s
ΔL	individual membrane discrete element length	m
ρ	density	kg/m^3
f	friction factor	-
A	area	m^2
α	permeability coefficient	m/s.Pa
D_h	hydraulic diameter	m
D_e	diffusivity	m^2/s
k	mass transfer coefficient	m/s
N_Re	Reynolds number	-
ν	kinematic viscosity	m^2/s
π	osmotic pressure	Pa
J	mass flux	kg/m^{2}s
D_i	internal diameter of pipe	m
Δx	individual pipe element length	m
w	specific energy consumption	kWh/m^3
n	Osmotic Pressure coefficient	-

Subscripts

Symbol	Definition
f	feed
r	reject
p	permeate, pipe
cs	cross section
avg	average
in	inlet
out	outlet
m	membrane-feed interface
t	tank
b	bulk
tot	total
atm	atmospheric

Abbreviations

Symbol	Definition
RO	Reverse Osmosis
BRO	Batch Reverse Osmosis
CP	Concentration Polarization
OM	Open- Modelica
PS	Pressure Sensor
ERD	Energy recovery devices
OOP	object-oriented programming
SEC	Specific Energy Consumption
HPP	High Pressure Pump

1 Introduction

The Reverse Osmosis (RO) process, which is widely utilized for desalination, water purification, and wastewater treatment, is commonly operated in continuous flow mode. Batch reverse osmosis (BRO) systems can achieve higher recovery rates with lower energy consumption compared to continuous RO systems. In batch RO, the applied pressure is increased gradually to make-up for the increasing osmotic pressure of the saline brine being treated. As a result, BRO process can approach the thermodynamic energy minimum more closely than continuous processes, enabling significant energy savings (Werber et al., 2017). Batch and hybrid batch/semi-batch configurations are especially suitable for high recovery

DOI 10.3384/ecp217113 Proceedings of the Asian Modelica Conference 2024 December 12-13, 2025, Jeju, Korea

desalination where the osmotic pressure of the final brine is significantly higher than that of the feed inlet (Patel et al., 2024). Additionally, batch configurations allow for enhanced process flexibility and can eliminate the need for energy recovery devices. To evaluate the economic feasibility of BRO process and design them optimally, mathematical modelling is employed.

Modeling and simulation of the BRO processes are critical for understanding the complex physical phenomena governing system behavior, such as pressure variation, flow rate and salinity changes, fouling dynamics, and power consumption. The variation of these phenomena locally across the membrane surface are often challenging to observe or measure experimentally, making modeling an essential tool for simulating various operating conditions and studying the behavior of concentration polarization and fouling. Through modeling, these parameters can be examined in greater detail, facilitating better optimization strategies and process designs that may not be achievable through experimental methods alone (Ghernaout, 2017).

While substantial research has focused on modeling continuous RO processes, with studies by Lee et al. (Lee et al., 2010) and Jeong et al. (Jeong et al., 2021) addressing membrane behavior, concentration polarization, and fouling in such systems, modeling of batch RO systems remains limited. Only a few studies (Barello et al., 2015; Kim & Park, 2024; Warsinger et al., 2016; Wei, n.d.), have explored batch processes in detail. However, comprehensive models specific to BRO systems that consider transient conditions, are limited. The need for more detailed models arises due to the unique dynamics of time-dependent concentration polarization and permeate removal, which are not fully captured by existing models.

Modelica, an object-oriented modeling language for complex systems (Fritzson & Engelson, n.d.; Introduction to Object-Oriented Modeling and Simulation with OpenModelica, n.d.; Loeffler et al., 2006), provides an ideal platform for simulating BRO processes. Unlike traditional simulation tools, Modelica enables the creation of reusable components that can be easily adapted to different system configurations, offering better flexibility in modeling BRO systems. Al-Zainati et al. (Al-Zainati et al., 2022) used Modelica for modeling a continuous RO system. Modelica's capabilities in developing dynamic control systems make it particularly useful for optimizing operational parameters such as pressure, flow rate, and recovery ratio, all of which are crucial for efficient BRO operation. Despite these advantages, no studies have yet applied Modelica to model BRO systems. This gives the opportunity to leverage this tool for simulating and controlling BRO processes.

Overall, this work presents the development of a BRO model using Modelica language and Open-Modelica as the tool to simulate key variables such as local pressure, flux, salinity variations and overall permeate production rate over time. By utilizing Modelica's strengths in reusability, flexibility, and control system modeling, this study proposes a promising approach for advancing the simulation of batch RO processes.

2 Theoretical Background

BRO process which is depicted in **Figure 1** operates by treating a fixed volume of recirculated feedwater until a desired overall recovery is achieved. This is different from a conventional continuous RO process where the desired recovery is achieved in a single pass through the membrane element. The process begins with feed water being filled into the tank, pipes, and membrane element, displacing the final reject brine from the previous cycle. The current model focuses on the dynamics of the system after this point, and hence the local salinity throughout the system is set equal to the feed salinity at $t = 0$.

This feed is circulated from the tank into the membrane element and back into the tank by a circulation pump that works to overcome the frictional losses in the channel and piping. This entire loop is pressurized to overcome the osmotic pressure and drive pure water flux through the membrane. Unlike in continuous RO, where the applied pressure at the inlet to the membrane is maintained at a nearly constant value with time, in BRO, pressure is increased with time to dynamically adjust for the increasing osmotic pressure as the solution gets more concentrated. This dynamic adjustment is the key reason for energy savings which minimizes the unnecessary excess pressure application in conventional continuous RO to achieve the same average flux. Due to these dynamic changes in pressures and concentrations over time, a transient model is needed to study the process in detail.

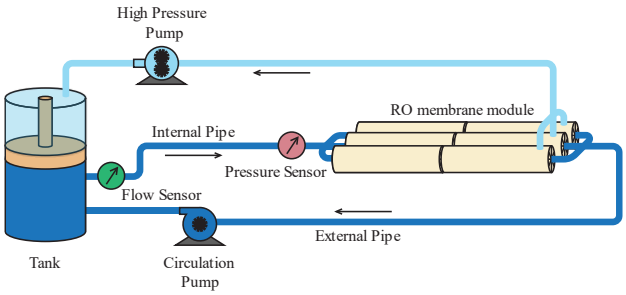


Figure 1 Batch Reverse Osmosis Process

2.1 Modelling of BRO process – Overall flowsheet

Modelling a BRO process involves capturing the dynamic nature of its operation, where feedwater undergoes cyclic pressurization and permeate recovery. One of the critical considerations is the variation in pressure over time, which must match the increasing osmotic pressure as the brine becomes more concentrated. Models must also account for concentration polarization (CP), which affects local flux and energy consumption by reducing effective driving pressure for water transport through the membrane. Energy consuming and energy recovery devices like

DOI
10.3384/ecp217113

pumps and turbines (if any) which are a part of the BRO systems, must be included in the model to estimate net energy consumption accurately.

Modelica's object-oriented programming (OOP) approach offers significant advantages in reducing redundant code and enabling efficient reuse of components. In the modelling of BRO systems, a reusable connector is developed to carry information on pressure, salinity, and flow rate. This connector can be flexibly utilized for the feed, reject, or permeate stream, depending on the specific application. Finite volume models for RO and pipe elements are created and structured to allow extension, facilitating the development of complete RO and pipe network models.

Additionally, separate models for tanks and pumps are designed for integration into the overall process. These individual components are then assembled into a complete flowsheet. This flowsheet serves as a basis for detailed mathematical simulations, for analysing system performance and transient dynamics in BRO processes. The overall architecture to model the BRO systems in Open-Modelica (OM) is given in the **Figure 2**.

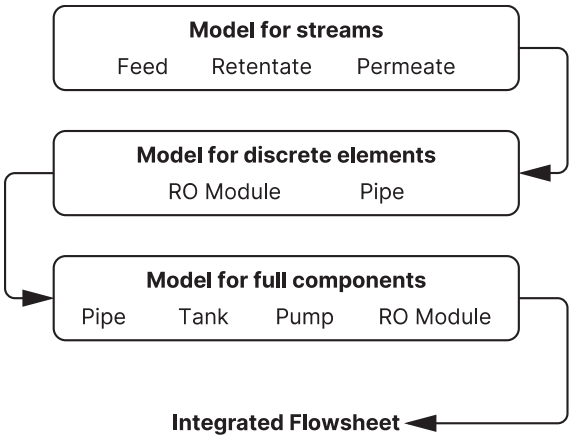


Figure 2 Architecture for BRO system modeling in OM

2.2 Modeling of individual components

2.2.1 Modeling of RO element

In the initial proof-of-concept model developed here, several simplifying assumptions are made such as treating membrane properties like water permeability coefficient (α) and feed density as constant. These assumptions can be relaxed in future work. While the RO process is often assumed to operate under steady-state conditions with negligible pressure losses due to pipe friction, this paper accounts for frictional pressure drop in the pipes and membrane channels. Concentration polarization near the membrane surface is modeled using the film theory to account for solute accumulation near the membrane interface due to water flux from the bulk solution.

The rate of change of local salinity at each feed node is calculated using Eq. (1). Since feed density variations are assumed to be small, Eq. (2) governs the overall solution mass balance at each node, accounting for water transport through the membrane as product. Local water flux is calculated using equation 1, which depends on pressure and osmotic differences across the membrane. Osmotic pressure of the feed is calculated using the equation 11, while the local concentration polarization is given by Eq. 10. The channel pressure drop is accounted for using the Haigen Poiseuille equation 3.

$$\frac{ds}{dt} = Q_{\mathrm{f}}.s_{\mathrm{f}} - Q_{\mathrm{r}}.s_{\mathrm{r}} - Q_{\mathrm{p}}.s_{\mathrm{p}} \quad (1)$$

$$Q_{\mathrm{f}} = Q_{\mathrm{r}} + Q_{\mathrm{p}} \quad (2)$$

$$\frac{P_{\mathrm{f}} - P_{\mathrm{r}}}{\Delta L} = \frac{2\rho v_{\mathrm{avg}}^2 f}{D_h} \quad (3)$$

$$v_{\mathrm{f}} = \frac{Q_{\mathrm{f}}}{A_{\mathrm{cs}}} \quad (4)$$

$$v_{\mathrm{avg}} = \frac{v_{\mathrm{f}} + v_{\mathrm{r}}}{2} \quad (5)$$

$$\mathrm{Re} = \frac{D_{\mathrm{h}} v_{\mathrm{avg}}}{\nu} \quad (6)$$

$$k = 2.53\frac{D_{\mathrm{e}}}{D_{\mathrm{h}}}.\left(16\mathrm{Re}^2 + \frac{0.4892}{\mathrm{Re}^{0.2964}}\right) \quad (7)$$

$$f = \frac{16}{\mathrm{Re}} + \frac{0.4892}{\mathrm{Re}^{0.036}} \quad (8)$$

$$J = \alpha\left(P_{\mathrm{avg}} - \pi_{\mathrm{m}}\right) \quad (9)$$

$$s_{\mathrm{m}} = s.e^{\frac{J}{k}} \quad (10)$$

$$\pi_{\mathrm{m}} = \pi_0\, s_{\mathrm{m}}^n \quad (11)$$

2.2.2 Modeling of Tank

The tank is assumed to be well-mixed, which ensures that the salinity within the tank remains uniform at any given time. This assumption eliminates the need to model spatial variations in the tank, which may be considered in later detailed analyses. The governing equations 12,13 represent volume and salinity balances in the tank. These equations incorporate time-dependent dynamics of volume and salinity based on inflow and outflow of the tank which are readily available using the connectors.

$$\frac{dV}{dt} = Q_{\mathrm{t,in}} - Q_{\mathrm{t,out}} \quad (12)$$

$$\frac{d(V \cdot s)}{dt} = s_{\mathrm{t,in}} \cdot Q_{\mathrm{t,in}} - s_{\mathrm{t,out}} \cdot Q_{\mathrm{t,out}} \quad (13)$$

2.2.3 Modeling of Pipe element

Modeling a pipe in Modelica to account for salinity and pressure variations involves representing mass and momentum balances along its length. The salinity changes are governed by equation 14, where flow rates (Q) and salinity (s) are measured at both inlet and outlet. Pressure drop is modeled using the Darcy-Weisbach equation 15, with the friction factor as f_p. The model assumes steady flow in a cylindrical pipe with constant fluid properties and negligible heat transfer. In Modelica, connectors handle pressure, flow rate, and salinity inputs and outputs, enabling integration with other system components. Mass and momentum balances are implemented with equations

DOI
10.3384/ecp217113
Proceedings of the Asian Modelica Conference 2024
December 12-13, 2025, Jeju, Korea

capturing salinity transport and pressure loss, while spatial discretization ensures accurate simulation of flow characteristics along the length of the pipe.

$$V\frac{ds}{dt} = s_{\mathrm{p,in}} \cdot Q_{\mathrm{p,in}} - s_{\mathrm{p,out}} \cdot Q_{\mathrm{p,out}} \quad (14)$$

$$\frac{P_{\mathrm{p,in}} - P_{\mathrm{p,out}}}{\Delta x} = \frac{\rho v_{\mathrm{avg}}^2 f_{\mathrm{p}}}{2 \cdot D_{\mathrm{i}}} \quad (15)$$

2.3 Flowsheet Simulation in Open Modelica

The individual models explained in section 2.2 for the tank, pipe, and RO modules were systematically integrated to create a complete flowsheet representation of the BRO process as shown in **Figure 3**. Each module is designed to operate independently, with connectors enabling data transfer. By combining these modular components, the flowsheet can capture the dynamic interactions between the tank, pipes, and the RO module. The flowsheet integrates these models to represent the physical experimental setup, providing a comprehensive framework for simulating the process.

In the overall flowsheet, initial conditions are set for salinity throughout the system. Initial values are also set for variables involved in non-linear equations to ensure numerical stability and solution convergence. These initial values allow the system to reach the desired operational state efficiently, replicating realistic startup and steady-state conditions. Using OM, the interconnected components facilitate the study of the BRO process under various scenarios, enabling detailed analysis of system performance, such as energy consumption, recovery efficiency, and pressure profiles. The modular structure of the flowsheet also allows for easy scalability and customization, supporting future extensions or modifications, such as adding energy recovery devices or advanced control schemes.

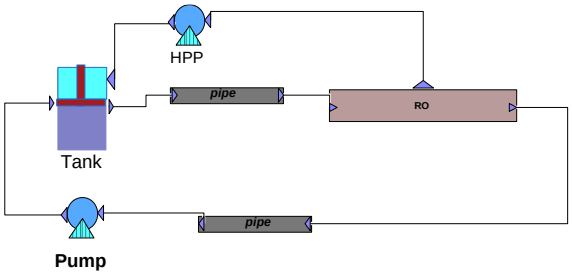


Figure 3 Flowsheet simulation of BRO process in Open Modelica

2.4 Specific Energy Consumption

The specific energy consumption (SEC) is a critical parameter in evaluating the efficiency of a BRO process. It is calculated based on the total mass of permeate produced and the energy consumption of the pumps over the operating time. For each pump the high-pressure pump and the recirculation pump, the energy consumption at each time step is determined as the function of the volumetric flow rate, the pressure differential, and the pump efficiency.

The equation 16 is used to calculate SEC in this study. Note that the actual SEC may be higher when factors such as permeate back-flux and pump energy usage during the cycle reset are considered.

$$w = \frac{W_{\mathrm{tot}}}{Q_{\mathrm{p}}} \quad (16)$$

$$W_{\mathrm{HPP}} = Q_{\mathrm{p}} \cdot (P_{\mathrm{t,out}} - P_{\mathrm{atm}})/\eta_{\mathrm{HPP}} \quad (17)$$

$$W_{\mathrm{CP}} = Q_{\mathrm{t,in}} (P_{\mathrm{t,out}} - P_{\mathrm{t,in}})/\eta_{\mathrm{CP}} \quad (18)$$

$$W_{\mathrm{tot}} = W_{\mathrm{HPP}} + W_{\mathrm{CP}} \quad (19)$$

3 Results and Discussions

The critical parameters and constants for the membrane are given in the **Table 1**. The simulations were conducted using Open-Modelica software version 1.23.0 on a computing system equipped with a 13th Gen Intel(R) Core (TM) i9-13900K processor (3.00 GHz), 64 GB RAM, and a 64-bit operating system. All transient simulations were carried out over a one-hour period, equivalent to 3600 seconds of runtime with a 60 second time step, to capture the system's dynamic behaviour under operational conditions.

Table 1 Parameters

Symbol	Value	Units	Description
RO membrane module			
W	20	m	Width of the channel
L	1	m	Length of the channel
H	7e-4	m	Height of the channel
N	50	-	Number of finite elements discretized
α	2.7e-7	m/(s·Pa)	Permeability coefficient
D_{e}	1.5e-9	m²/s	Diffusivity coefficient
D_{h}	8.2e-4	m	Hydraulic diameter
A_{mem}	40	m²	Available area for the membrane
A_{cs}	0.014	m²	Cross-sectional area of the channel
J_{avg}	15	kg/(m³·hr)	Average permeate flux
RR	0.5	-	Recovery Ratio
Tank			
V	1.2	m³	Volume of the tank
Area	0.4	m²	Area of the tank
Pipe			
D_{i}	0.0762	m	Internal Diameter of the pipe
L	0.5	m	Length of the pipe
Pumps			
η_{CP}	0.9	-	Efficiency of the circulation pump
η_{HPP}	0.9	-	Efficiency of the HPP

DOI
10.3384/ecp217113

3.1 Transient Analysis of Salinity and Flux Variation Along the Membrane Length

Figure 4 illustrates how local salinity changes across the length of a 1-meter membrane length over time. The membrane is discretized into 50 equal sections of 0.02-meter length each. The local salinity variation along the membrane length is plotted at different time-steps during the process. The first line corresponds to t = 1 min, the second at t = 11 min, and the final line at t = 60 min (1 hour).

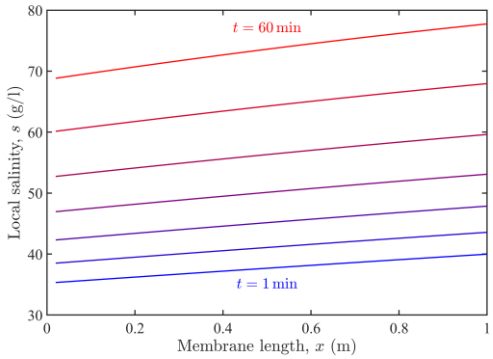


Figure 4 Local salinity profile along the length of the membrane for increasing time

Initially, all the water in the RO module is at 35 g/l salinity. But as the time progresses, salinity in RO increases due to permeate production. At the same time, feed entering the RO module is less saline compared to the local brine. Hence, at the start of the membrane, salinity is lower compared to end of the membrane where gradually more permeate is recovered along the flow. As time progresses, tank gets much more saline as less volume of low salinity water mixes with same amount of brine. Therefore, feed entering the module gets even more saline, resulting in steeper increase in in inlet salinity with time. This makes salinity gradient along the membrane more pronounced with time. This effect is captured in Figure 4.

3.2 Transient Analysis of Flux Variation Along the Membrane Length

Figure 5 illustrates how the flux changes along the membrane length at each single node of the membrane element. The spatial variance in flux increases with time. This can be explained by effect of salinity as shown in Figure 4 as well as pressure drop along the module length. At any given time, pressure at inlet is higher compared to pressure at the outlet. Therefore, first term in driving force $(P - \pi_m)$ is less at outlet compared to inlet. Also, as explained in Figure 4, salinity at inlet is lower compared to salinity at the outlet. Hence, the second term in the driving force is more at outlet compared to inlet. These both effects simultaneously make flux at outlet less compared to inlet. Specifically, as salinity gradient along the membrane length increases with increase in time, flux variance also increases with time. This effect is captured by the model and shown in **Figure 5**.

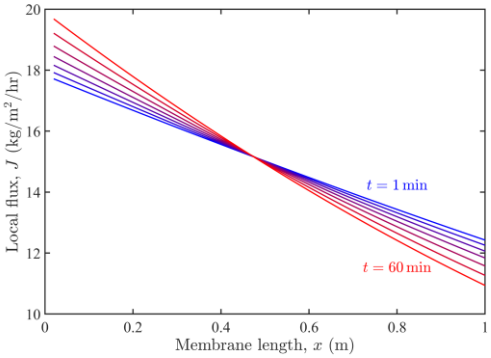


Figure 5 Local permeate flux profile along the length of the membrane for increasing time

3.3 Pressure Changes in the Tank

Figure 6 shows how pressure increases with time in batch RO operation when a constant flux is maintained. This is due to increase in average feed salinity as more permeate is recovered. This increased salinity results in increase in osmotic pressure of the tank and consequently of the feed. Therefore, pressure is further increased to maintain a constant permeate flux. As salinity increases slowly at first when the volume of the tank is larger, the same trend is observed for pressure in **Figure 4**, where the pressure increases more rapidly at later times.

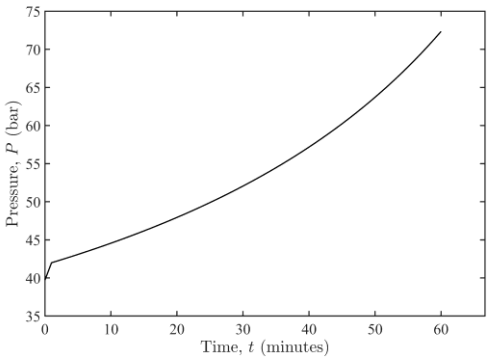


Figure 6 Pressure changes with time for a fixed average flux

3.4 Energy consumption

Figure 7 represents the relationship between the specific energy consumption (w) and the average permeate flux (J_{avg}). The x-axis shows the set or fixed average permeate flux, while the y-axis indicates the specific energy consumption of the two pumps used in the process, which operate under varying pressure conditions in the tank.

As the average permeate flux increases, the specific energy consumption rises significantly. This trend occurs because higher permeate flux requires greater pressure differences across the membrane, leading to increased energy demands for the pumps. At lower flux values, the energy consumption is relatively low due to smaller pressure requirements. However, as the flux increases, the pressure requirements rise non-linearly, causing a steep increase in specific energy consumption.

DOI
10.3384/ecp217113
Proceedings of the Asian Modelica Conference 2024
December 12-13, 2025, Jeju, Korea

This graph highlights the trade-off between achieving higher permeate flux and maintaining energy efficiency. The results emphasize the importance of optimizing operational parameters to balance water production and energy consumption in reverse osmosis systems.

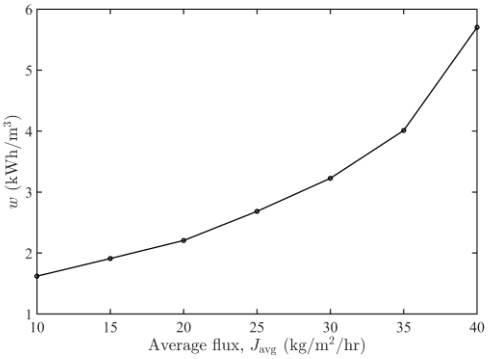


Figure 7 Average Permeate flux vs SEC

4 Conclusion

This study demonstrates the effectiveness of using an OOP approach in Open-Modelica to model and analyse the transient dynamics of BRO systems. By integrating modular components such as the membrane, tank, and pipe models into a comprehensive flowsheet, the system's behaviour under varying conditions was successfully simulated. The transient simulations revealed key insights into salinity and flux variation along the membrane length and variation in specific energy consumption rate of the system with time.

The relationship between average permeate flux and specific energy consumption underscores the need to optimize operating conditions to achieve a balance between performance and energy efficiency. The modelling framework helps explain the dynamic behaviour of BRO systems and paves the way for further optimization of the BRO process.

Acknowledgements

This collaborative work between Indian Institute of Science Bengaluru and Korean Institute of Science and Technology has been supported and funded by Department of Science and Technology, Government of India, (Project no. INT/Korea/P-690) and Indian Institute of Science Bengaluru.

References

Al-Zainati, N., Subbiah, S., Yadav, S., Altaee, A., Bartocci, P., Ibrar, I., Zhou, J., Samal, A. K., & Fantozzi, F. (2022). Experimental and theoretical work on reverse osmosis - Dual stage pressure retarded osmosis hybrid system. *Desalination*, *543*. https://doi.org/10.1016/j.desal.2022.116099

Barello, M., Manca, D., Patel, R., & Mujtaba, I. M. (2015). Operation and modeling of RO desalination process in batch mode. *Computers and Chemical Engineering*, *83*, 139–156. https://doi.org/10.1016/j.compchemeng.2015.05.022

Fritzson, P., & Engelson, V. (n.d.). *Modelica-A Unified Object-Oriented Language for System Modeling and Simulation*.

Ghernaout, D. (2017). Reverse Osmosis Process Membranes Modeling-A Historical Overview. *Journal of Civil, Construction and Environmental Engineering*, *2*(4), 112–122. https://doi.org/10.11648/j.jccee.20170204.12

Introduction to Object-Oriented Modeling and Simulation with OpenModelica. (n.d.).

Jeong, K., Son, M., Yoon, N., Park, S., Shim, J., Kim, J., Lim, J. L., & Cho, K. H. (2021). Modeling and evaluating performance of full-scale reverse osmosis system in industrial water treatment plant. *Desalination*, *518*. https://doi.org/10.1016/j.desal.2021.115289

Kim, G. Y., & Park, K. (2024). Application of batch reverse osmosis as an appropriate technology for inland desalination: Design, modeling, and operating strategies. *Desalination*, *592*. https://doi.org/10.1016/j.desal.2024.118185

Lee, S., Boo, C., Elimelech, M., & Hong, S. (2010). Comparison of fouling behavior in forward osmosis (FO) and reverse osmosis (RO). *Journal of Membrane Science*, *365*(1–2), 34–39. https://doi.org/10.1016/j.memsci.2010.08.036

Loeffler, M., Huhn, M., Richter, C., & Kossel, R. (2006). *The Modelica Association Modelica* (Vol. 4).

Patel, D., Ankoliya, D., Raninga, M., Mudgal, A., Patel, V., Patel, J., Mudgal, V., & Choksi, H. (2024). Batch Reverse Osmosis: Evolution from the Concept to the Technology. In A. Mudgal, P. Davies, M. Kennedy, G. Zaragoza, & K. Park (Eds.), *Advances in Water Treatment and Management* (pp. 175–200). Springer Nature Singapore.

Warsinger, D. M., Tow, E. W., Nayar, K. G., Maswadeh, L. A., & Lienhard V, J. H. (2016). Energy efficiency of batch and semi-batch (CCRO) reverse osmosis desalination. *Water Research*, *106*, 272–282. https://doi.org/10.1016/j.watres.2016.09.029

Wei, Q. J. (n.d.). *BATCH REVERSE OSMOSIS: EXPERIMENTAL RESULTS, MODEL VALIDATION, AND DESIGN IMPLICATIONS*.

Werber, J. R., Deshmukh, A., & Elimelech, M. (2017). Can batch or semi-batch processes save energy in reverse-osmosis desalination? *Desalination*, *402*, 109–122. https://doi.org/10.1016/j.desal.2016.09.028

Object Oriented Modeling of Single and Multi-Bed Pressure Swing Adsorption Processes using OpenModelica

Nikhil Sharma[1] Kannan Moudgalya[1] Sunil Shah[2]

[1]Department of Chemical Engineering, Indian Institute of Technology Bombay , India,
{nikhil_sharma,kannan}@iitb.ac.in
[2]Modelicon Infotech, Bengaluru, India sunil.shah@modelicon.in

Abstract

Pressure Swing Adsorption has been implemented to produce pure oxygen from air. Its model is solved using the methods of finite difference and orthogonal collocation on finite elements. Discrete events of this process are modelled using state graphs. Solution to the PSA process using a single bed is presented. With two beds, it is shown that it is possible to produce oxygen continuously. All of these have been done using OpenModelica, and the code is released as open source.
Keywords:Pressure swing adsorption, state graph, Open Source, High purity oxygen production

1 Introduction

Pressure Swing Adsorption (PSA) is a technology employed for separations and purification of gases. PSA operates on the basis of preferential adsorption of some gases on adsorbents, such as molecular sieves. Pressure is varied across the operations, and hence the name. (Skarstrom 1959) used the PSA process first time to enrich oxygen and nitrogen in a heatless drier. Skarstrom invented a two-bed cycle for the PSA to produce oxygen with pressure equalization step using zeolite 13X adsorbent. The PSA process has been widely utilised from then on ruthven1984principles. (Hassan, D. Ruthven, and Raghavan 1986) proposed a simple dynamic model to produce oxygen from the PSA process, It is based on linear rate of mass transfer and used Langmuir adsorption equilibrium equations. They further assumed that pressure remained constant during adsorption and desorption steps. (Farooq, D. Ruthven, and Boniface 1989) also introduced a kinetically controlled dynamic model for the Oxygen-PSA process. The advantage of kinetically controlled model is that mass transfer effects and axial dispersion are easy to calculate. (Farooq and D. Ruthven 1990) developed a linear driving force model and used carbon molcular sieves. The four steps used for the PSA cycle consists of:

1. Pressurisation.
2. Adsorption
3. Blow-down.
4. Purging with product.

PSA process is greatly influenced by design parameters such as bed length, time for each step viz. pressurization, feed, blowdown and purge. It is also influenced by feed and purge flow rates, production rate, temperature, pressure, etc. So it is imperative to obtain optimum amount of process variables.

In this work a general purpose OpenSource simulator for PSA is developed on the top of OpenModelica. Partial differential equations governing PSA model are solved numerically by developing generalized functions for finite difference and orthogonal collocation on finite elements techniques. Use of state-graphs enable visual and hence error-free implementation of the switching operation.

2 State-Graph Library in OpenModelica

StateGraph is an inbuilt library in OpenModelica which is used here for control applications. It is an upgraded finite state machine based on JGrafchart method that utilises Modelica feature of "action" language. The StateGraph has similar modeling capabilities as that of StateCharts with improved features. Main elements of a StateGraph are Steps and Transitions, as shown in Fig. 1.

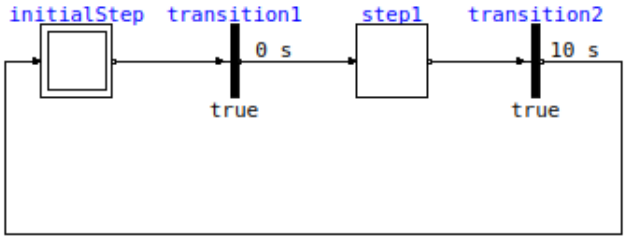


Figure 1. Steps and Transition blocks in StateGraphs

We explain the concept of StateGraph now, as it is used extensively in this wrok. Steps are represented by square boxes and transition by rectangles. Initial step is represented by double square boxes. Possible states of the StateGraphs are represented by Steps. A state can be changed by the use of Transitions. If a step is active then

associated Boolean variable is true and becomes false otherwise. A transition fires when condition associated with it becomes true leading to input step to change to inactive state and output step becomes active.

In Fig. 1 the wait time for transition is given as 0 sec and set true. Therefore at time 0 tranistion1 fires and step1 becomes active leading to initialStep become inactive. The wait time for transition2 is 10 sec. Therefore, transition2 fires after 10 sec of transition1 leading step1 to be inactive and initialStep becomes active and cycle continuous.

3 PSA Process

Pressure Swing Adsorption (PSA) is based on the preferential adsorption of some gases on adsorbents, such as molecular sieves. In general, PSA includes four-steps:

1. Pressurization: Pressure plays a crucial role in adsorption of gases on solids. A gas is particularly adsorbed on the solid bed at high pressure according to bed characteristics and desorbed at lower pressure. The first step is pressurization where bed is pressurized with high pressure inert gas. This helps in avoiding sudden shock during the high pressure feed inlet step.

2. Adsorption or Production: This step is the production process wherein the feed is introduced from the bottom of the bed. Some components of the gas mixture preferentially get adsorbed according to the adsorbate and adsorbent characteristics, thereby enabling the separation of gases. Pure product gas is taken out from the other end.

3. Counter-Current Blowdown: When the bed gets saturated with adsorbed gases, whatever feed comes in it goes as it is to the outlet. No more separation is now possible. So bed needs to be regenerated by exposing it to atmosphere/low pressure. Due to pressure difference components flow out of the bed.

4. Counter-Current Purge: Pure product gas at low pressure is passed counter current from the top of the bed. Desorption takes place and bed gets regenerated for using in next cycle. fig:O2Prod demonstrates this for oxygen production.

4 Numerical Solution of Adsorption in a Fixed Bed

In this section, we will outline the model for adsorption in a fixed bed, and the numerical methods to solve them.

We assume that the concentration gradients are mainly along axial direction because of high aspect ratio of the bed. Linear driving force for mass transfer is assumed. The concentration and the mole fraction of components in the gas stream are given by

$$C_j = \frac{Y_j P}{R T_g} \tag{1}$$

where, P and T_g are the total pressure and temperature of gas stream, respectively, C_j and Y_j are the concentration and mole fraction of component j.
Langmuir Model:

$$\frac{\partial \bar{q}_j}{\partial t} = -K_{L,j}(q_j^* - \bar{q}_j) \tag{2}$$

Component Mass Balance:

$$\varepsilon \frac{\partial C_j}{\partial t} + \frac{\partial u C_j}{\partial Z} = \varepsilon D_{ax}\Big(\frac{\partial^2 C_j}{\partial^2 Z}\Big|_{z=0^-}\Big) - (1-\varepsilon)\rho_P \frac{\partial \bar{q}_j}{\partial t} \tag{3}$$

Bulk Mass Balance:

$$\varepsilon \frac{\partial C_T}{\partial t} + \frac{\partial u C_T}{\partial Z} = \varepsilon D_{ax}\Big(\frac{\partial^2 C_T}{\partial^2 Z}\Big) - \sum(1-\varepsilon)\rho_P \frac{\partial \bar{q}_j}{\partial t} \tag{4}$$

Initial Conditions:
$t = 0$:

$$C_j(Z,0) = \bar{q}_j(Z,0) = 0 \tag{5}$$

Boundary Conditions:
Counter-current Pressurization Step:
$Z = L$:

$$\varepsilon D_{ax}\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^+} = -u(C_j|_{Z^-} - C_j|_{Z^+})$$

$Z = 0$

$$\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^-} = 0 \tag{6}$$

Production Step:
$Z = L$:

$$\varepsilon D_{ax}\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^+} = -u(C_j|_{z^-} - C_j|_{z^+}) \tag{7}$$

$Z = 0$:

$$\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^-} = 0 \tag{8}$$

Counter-current Blowdown Step:
$Z = 0$:

$$\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^-} = 0 \tag{9}$$

Counter-current Purge Step
$Z = L$:

$$\varepsilon D_{ax}\Big(\frac{\partial C_j}{\partial Z}\Big)\Big|_{Z^+} = -u(C_j|_{Z^-} - C_j|_{Z^+})$$

DOI
10.3384/ecp217119

$Z = 0$

$$\left(\frac{\partial C_j}{\partial Z}\right)\Big|_{Z^-} = 0 \tag{10}$$

Table 1. Variables used in Fixed Bed Model

C_j	Concentration of species (mol/m^3)
P	Total pressure (bar)
R	Gas Constant
Tg	Temperature of gas (K)
$\bar{q_j}$	Equilibrium Concentartion of species
q_j^*	Concentration of species in solid bed
ε	Porosity
u	Velocity(m/s)
D_{ax}	Axial dispersion coefficient
ρ_p	Density of particle (kg/m^3)
C_T	Bulk gas concentration

The easiest method to solve the above set of equations is the finite difference method, using which, the PDE is converted into ordinary differential equations (ODEs), which are solved using an integrator, such as DASSL. In this work, we divided the bed into 10 intervals of equal length.

Orthogonal collocation on finite elements is in general a faster method to solve the PDEs. We divide the entire length of the bed into three short subsections (0-0.3, 0.3-0.7, 0.7-1), which allows us to use a low order polynomial to approximate the solution. This in turn reduces the undesirable oscillatory phenomenon. Legendre third degree polynomials are used in this work to approximate the solution. It is possible to find the coefficients of the polynomials in the three subsections using appropriate continuity conditions. The resulting ODEs are solved using DASSL.

5 Modelling Pressure Swing Adsorption

In this section, we concentrate on operating the fixed bed in four different modes, namely, pressurization, production, blowdown, and purge. We begin with the example of oxygen production from air. The OpenModelica code developed in this work is available at (NikhilOM 2021).

5.1 Example: Oxygen Production from Air using PSA

In this section, we devote our attention to the process of producing high purity oxygen by separating it from nitrogen in air. Mathematical equations that describe the underlying adsorption process have already been presented in Sec. 4. The four step process for fixed bed adsorption is shown in Fig. 2.

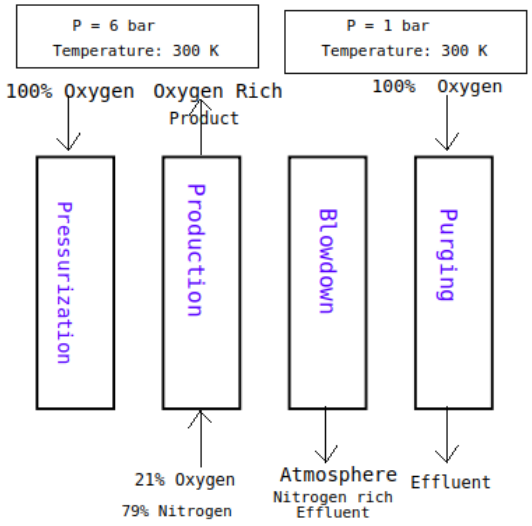


Figure 2. Four step adsorption process for fixed bed adsorption

The fixed bed is pressurized with pure O_2 at high pressure of 6 bar absolute pressure at 300 K. Feed is a mixture of N_2 and O_2 with 79% and 21% composition respectively is passed from the opposite end to that of pressurization. Here the adsorption of N_2 takes place and pure O_2 comes out till the bed is saturated. Next step after bed gets saturated is blowdown operation where bed is exposed to low pressure atmosphere and due to pressure difference adsorbed gases comes out after desorption and effluent is rich in N_2. During the purging operation pure O_2 is passed counter currently at low pressure of 1 bar and bed is regenerated with effluent coming out from the other end. The time intervals for pressurization, production, blowdown, and purging are taken respectively as 30%, 20%, 30% and 20% of the cycle time, as suggested by (Douglas M Ruthven 1984).

5.2 Implementing Discrete Events using State-Graphs

Depending on the four modes of operating the fixed bed, an appropriate model has to be solved. The model selection for each of the four modes is achieved by opening/-closing valves using using state-graphs, an inbuilt library in OpenModelica. Fig. 3 shows the diagram view of PSA process for the fixed bed adsorber for O_2 production in OpenModelica. It shows process steps are actuated by valves which in turn are modeled separately using State-Graphs shown in Fig. 4. The valve connected to pressurization unit is opened when boolean expression associated with it is true. As shown in Fig. 3 boolean expression connected to pressurised section becomes true when step is active which is evident from Fig. 4 that it happens at time 0. The same logic is applicable to opening and closing of valves associated with other steps.

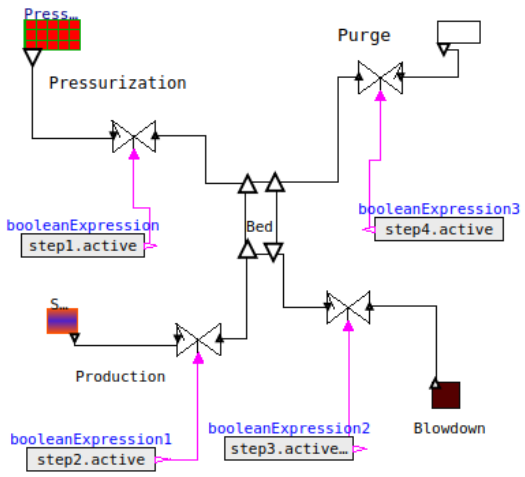


Figure 3. Diagram view of the four step PSA process in OpenModelica. Boolean expressions are activated as per the StageGraph in Fig. 4.

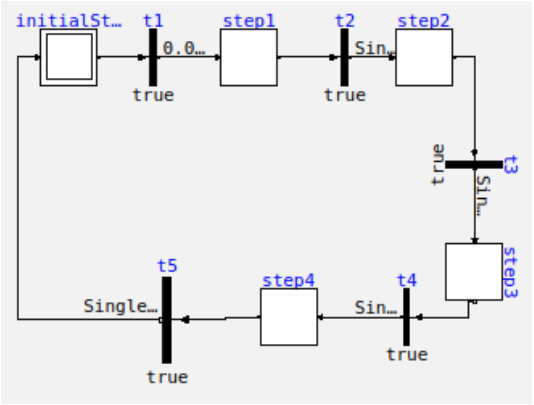


Figure 4. Steps Implementation of Fixed Bed Adsorber using StateGraphs. Variables step1, step2, step3 and step4 are activated respectively at (0.3,0.2,0.3,0.2) × (cycle time)

6 Simulation of Single Bed PSA Process

The parameters used for O_2 production in this work are given in Table 2 and Table 3.

For pressurization, pure O_2 at high pressure of 6 bar is passed for 30% of cycle time. During the production step feed air which is a mixture of N_2 and O_2 is supplied at the other end. The feed step takes place for 20% of the cycle time. For blowdown operation, the bed is exposed to atmosphere for 30% of the cycle time. During the purge operation, pure O_2 is again passed counter-currently at a low pressure of 1 bar for 20% of the cycle time. These values are tabulated in Table 4.

Fig. 5 shows the mole fraction of N_2 at outlet i.e at the point at each time. Cycle time is taken as 1100s. One can see from the figure that no N_2 comes out at time interval 600-780s, as the mole fraction of N_2 during this interval is zero. During this interval, pure oxygen is produced, as one

Table 2. Input Parameters

Particle Dia.	0.0038 m
Void Fraction, ε	0.5418
Particle Density, ρ	600 Kg/m^3

Table 3. Properties of Adsorbent

Species	**Max. Adsorbed Conc. q_m**	**b_0 Pa^{-1}**
N_2	14	4.96e-10
CO_2	7.90	1.55e-11

Table 4. Step Time for O_2 Production

Steps	**Time**
Pressurization	330 sec
Production	220 sec
Blowdown	330 sec
Purge	220 sec

can see from the O_2 mole fraction profile, given in Fig. 6. This process is repeated at other time intervals also, as can be seen from these figures.

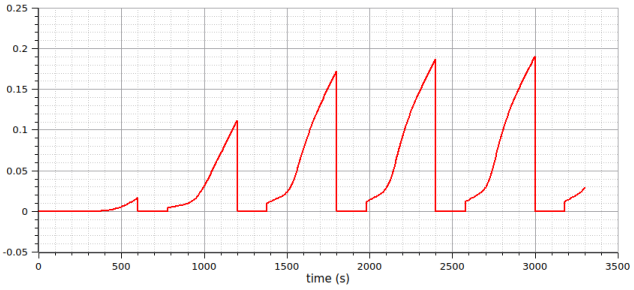


Figure 5. Mole Fraction of N_2 at outlet point

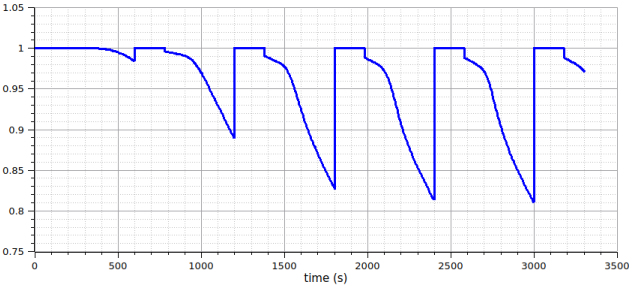


Figure 6. Mole Fraction of O_2 at outlet point

Using single bed adsorber pure oxygen can be obtained but the supply of pure Oxygen in the product is not continuous because the bed needs to be regenerated after saturation. One way to address this difficulty is to employ more

DOI
10.3384/ecp217119

beds.

7 Simulation of two bed PSA process

In this section, we show that it is possible to produce O_2 continuously with the help of two fixed beds. When product is taken out from a bed the other bed remains in the regenerated phase and vice-versa. Each of the two beds is operated exactly as described in the previous section.

The four step process for oxygen production using two beds of adsorbers is shown in Fig 7.

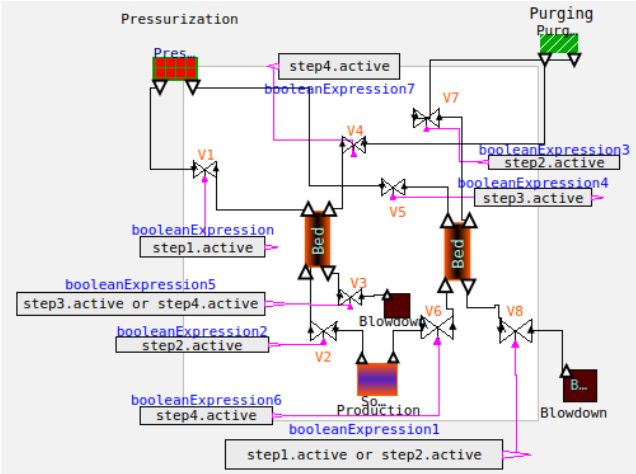


Figure 7. Multi-Cylinder adsorber for O_2 production

The operation of multi cylinder adsorber is described in Table 5.

Table 5. Four step process for multi cylinder bed

Step	Open valve	Bed-1	Bed-2
1	V1, V8	Pressurization	Blowdown
2	V2, V7, V8	Production	Purging
3	V3, V5	Blowdown	Pressurization
4	V3,V4 V6	Purging	Adsorption

Recall that we used the StateGraph in Fig. 4 for the single bed operation. The same StateGraph works for the two bed operation as well. The only difference is that now more than one valve is operated when a Boolean operation is active, as shown in Table 6.

Table 6. Steps Time for CO_2 Capture

Steps	Time
Pressurization	30 % of cycle time
Production	20 % of cycle time
Blowdown	30 % of cycle time
Purge	20 % of cycle time

The mole fraction of O_2 produced by the two bed PSA process is given in Fig 8. Blue line indicates the oxygen production in one bed, and the red line corresponds to that in the other bed. One can see that the oxygen mole fraction is almost 1 at all times.

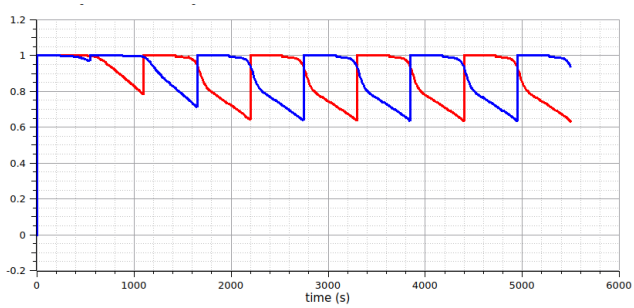


Figure 8. Mole fractions of O_2 at outlet of both beds for 1100 seconds

8 Conclusion

A model for PSA has been implemented and solved using the open source modeling and simulation environment OpenModelica. The object oriented modeling capability has enabled easy extension of a single bed PSA to multi-bed processes. The transitions occurring in PSA are implemented through the inbuilt library StageGraph in OpenModelica. As it is a visual method of programming, it is natural for engineers, and hence results in error-free coding. As the StateGraph capability is inbuilt, it results in error-free implementation as well.

Commercial software that is capable of modeling niche operations, such as PSA, can be prohibitively expensive to students, and to small and medium scale enterprises. As a result, this population may not have access to such important technologies. We are happy to partially address this issue by releasing our code as open source Code. We also believe that this is another initiative in the direction of collaborative content creation, and improving the employability of students, as articulated by (Nayak et al. 2019).

The authors would like to thank the Ministry of Education (previously MHRD), Government of India, for funding this work through the FOSSEE project at IIT Bombay.

References

Farooq, S and DM Ruthven (1990). “A comparison of linear driving force and pore diffusion models for a pressure swing adsorption bulk separation process”. In: *Chemical engineering science* 45.1, pp. 107–115.

Farooq, S, DM Ruthven, and HA Boniface (1989). “Numerical simulation of a pressure swing adsorption oxygen unit”. In: *Chemical Engineering Science* 44.12, pp. 2809–2816.

Hassan, MM, DM Ruthven, and NS Raghavan (1986). “Air separation by pressure swing adsorption on a carbon molecular sieve”. In: *Chemical Engineering Science* 41.5, pp. 1333–1343.

Nayak, P. et al. (2019). “Chemical Process Simulation Using OpenModelica”. In: *I&EC Research* 58, pp. 11164–11174.

NikhilOM (2021). *Pressure Swing Adsorption*. https://github.com/NikhilOM/Pressure-Swing-Adsorption/. [GitHub; accessed 20-Dec-2021].

Ruthven, Douglas M (1984). *Principles of adsorption and adsorption processes*. John Wiley & Sons.

Skarstrom, Charles W (1959). "Use of Adsorption Phenomena in Automatic Plant-Type Gas Analyzers". In: *Annals of the New York Academy of Sciences* 72.13, pp. 751–763.

DOI
10.3384/ecp217119