Hybrid physical-AI based system modeling and simulation approach demonstrated on an automotive fuel cell

Moritz Hübel[1] Nirmala Nirmala[1] Michael Deligant[2] Lixiang Li[3]

[1]Modelon Deutschland GmbH, Hamburg, Germany, Moritz.hubel@modelon.com
[2]Arts et Metiers Institute of Technology CNAM, LIFSE, HESAM University, Paris, France,
[3]Modelon Inc., Ann Arbor, United States

Abstract

This paper presents an approach on how to train a Neural Network model based on a detailed physical Modelica model. The necessary steps to generate training data from simulation will be explained as well as the generation process of a surrogate model. It will be shown, how the surrogate will be re-integrated into the Modelica system model. A benchmark based on accuracy and simulation performance will be performed. The tools used are Modelon Impact, an online modeling and simulation platform, the TensorFlow/Keras toolbox in a Jupyter Notebook which provides a Python-based interface for generating Neural Networks, and the Modelica Neural Network Library that provides functions for constructing Neural Networks within Modelica. The approach is demonstrated on an automotive fuel cell model which is part of an overall vehicle system model. One possible application is to train the neural network via repeated simulations and then to reuse it as an embedded software component for efficiently estimating fuel use and range for various driving cycles and ambient conditions.

Keywords: Machine Learning, Neural Networks, Hybrid Models, Hydrogen, Fuel Cell.

1 Introduction

Model-based system design and engineering plays a major role, not only in the development of new technical systems but also in supporting efficient usage or operation. On the one hand, Modelica as an open-standard multi-domain programming language can be used to describe complex technical systems on a fundamental basis. Text-book equations are often implemented on a component level, which can then be used to allow a graphical composition of system models using connection ports at the model interfaces to provide boundary conditions locally and close the equation system. Applying good modeling practices, replaceable models for different components can be implemented, allowing fidelity adaptation of the system by choosing different component models for different applications. Some applications require complex mathematical formulations that are necessary to describe a physical problem accurately and thereby sacrificing on computational performance during simulation of the model. Machine learning on the other hand allows creating models based on data without necessarily understanding the correlations between the inputs and the outputs on a fundamental basis. Neural Networks are a common approach to create models that can accurately predict the outputs based on different input combinations after the model has been trained sufficiently well. Neural Networks consists of node layers that are structurally inspired by the biological brains that can transmit signals to other neurons based. Due to their similarity with the biological counterpart, Neural Networks are categorized as a method of artificial intelligence (AI).

A hybrid physical-AI based model can consist of both components: models derived from first principal physics as well as data-based models such as Neural Networks. Especially the availability of physical component models providing an extensive data base for training, allows creation of hybrid models which can achieve better simulation performance while not sacrificing accuracy for a given question. Known physical relations in specific components can be used to train surrogate models in physics-guided machine learning processes [1]. That way, computationally expensive components can be replaced, and simulation performance can be increased if the specific component is not of interest for a specific set of internal calculations but needed to provide boundary conditions for other components in a system. In comparison to other regression techniques, like map or polynomial fitting, the Neural Network based approach allows for representation of strong non-linearities superior to polynomial fitting while allowing a higher degree of freedom and less data need compared to a 1:1 data mapping. Especially when more than two independent inputs need to be mapped, standard map-based approaches quickly run into limitations of limited matrix dimensions in various tools which is less critical for Neural Networks. In this paper, an approach will be presented that uses a detailed physical fuel cell model of a fuel cell vehicle to generate a reduced order model of the fuel cell itself to study fuel consumption for different driving cycles. Section 2 will introduce the problem and the underlying physical sub-models in detail. In section 3, the workflow to create the hybrid model will be

DOI 10.3384/ecp193157 Proceedings of Asian Modelica Conference 2022 November 24-25, 2022, Tokyo, Japan

described. Section 4 will show a benchmark of accuracy and performance of the hybrid model against the detailed physical model and section 5 will show the result for a long-term driving cycle simulation.

2 Problem Statement

Following the trends to reduce greenhouse gas emissions and save resources, one proposed approach for the mobility sector is hydrogen-powered fuel cells to generate electricity on demand and allowing a sufficient range while reducing the battery size drastically. A comprehensive model of such a fuel-cell vehicle has been developed in Modelica, a comprehensive summary on the underlying sub models has been published [2], [3].

The use-case of this paper is to calculate the range of the vehicle for a long route as quickly as possible and thereby demonstrate the performance improvement of a hybrid-model consisting of physical components and trained Neural Network models. Other potential improvements such as model solvability and robustness will not be discussed here.

A high-level schematic of the model in the Modelica modeling and simulation platform Modelon Impact is shown in Figure 1.

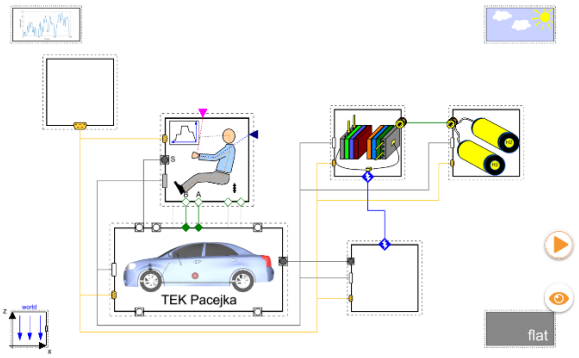

Figure 1: Top level Schematic of the vehicle system model, showing the replaceable drive cycle component (top left), the ambient condition component (top right) as well as the coupled vehicle model with driver, controls, drive train, fuel cell and hydrogen tank.

The model includes a driving cycle input defining the desired velocity trajectory for the vehicle. The WLTC2 Class 2 cycle [4], shown in Figure 2 is used here as a reference.

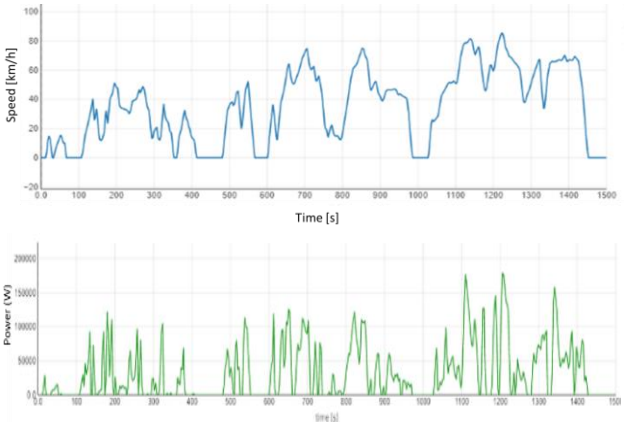

Figure 2: WLT2 driving cycle, velocity vs. time used as input for the reference scenario (top) and mechanical power at the drive-train shaft (bottom)

The vehicle model includes chassis, tires, breaks, and interacts with the driver and controls model. Thereby, it will define the propulsion power (torque and angular velocity) of the motor, considering the aerodynamic losses, rolling friction, and braking losses. The hybrid drivetrain includes a small battery, battery converter, fuel cell converter, and a DC motor. The electric power is provided by a proton exchange membrane (PEM) fuel cell stack fueled from a hydrogen storage tank.

3 Hybrid Model Generation Workflow

Generation of Proper Hybrid Models for Smarter Vehicles is the core topic of a research project funded by the German Federal Ministry for Economic Affairs and Climate Action. Different options to generate and integrate data-based models with physical Modelica models and tools are investigated. [5] have presented an approach to replace numerically inefficient and fragile non-linear equation blocks with surrogates during compilation. Another related workflow will be presented here that instead of interacting on the compiler level, utilizes the existing Modelica structure of replaceable sub-models for each component. This concept will be used to not only allow selection of different fidelity, first-principle physical models but also to integrate Neural Network surrogate models. Bringing the model back as part of the original Modelica system model will result in a hybrid model that can have superior performance and acceptable accuracy, so that it can be used for more applications such as those requiring real-time capabilities or even deployed as conventional FMU or eFMU on embedded hardware eventually.

DOI
10.3384/ecp193157

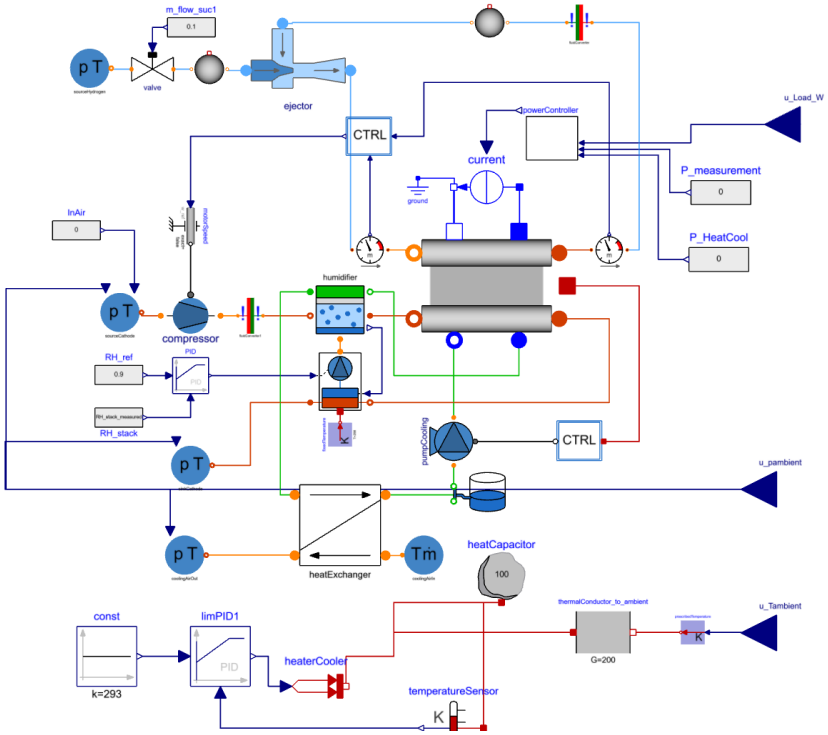

Figure 3: Overview of the physical system model including a detailed fuel cell stack and a simplified heating and cooling system of the car as well as control blocks. The independent inputs used for this study are the fuel cell load, the ambient pressure and the ambient temperature indicated by the connection ports on the right

3.1 Physical Model (Modelica)

The detailed Modelica model of the system is illustrated in Figure 3. The model includes the following components: a PEM fuel stack cell with empirical model for polarization, and dynamic mass and energy balance for anode, cathode and cooling channels; an ejector model to recirculate excessive hydrogen; a humidifier model to control the humidity of incoming air and recover some of the waste heat in the cooling loop; a cooling loop with heat exchange, pump and tank. The medium is represented as an ideal gas mixture with moisture using the NASA 7-coefficient model including the following components: H2, CO, CO2, H20, N2, O2. In addition, a simple heating and cooling system has been added, considering heat transfer to ambient and maintaining a convenient cabin temperature at 293K.

The following figures are giving an indication on the complexity of the model:

- Continuous states: 82
- Variables: 2572
- Linear equation blocks: 13
- Non-linear equation blocks: 4

To calculate fuel consumption for a given route, three independent inputs have been identified:

- Fuel cell power as a resulting output of the vehicle model for a given drive cycle (speed vs. time).
- Ambient temperature, primarily affecting the vehicles heating/cooling system but also the temperature and losses of the fuel cell.
- Ambient pressure affecting the air compressor

3.2 Modelica Simulation Tool

Modelon Impact is a cloud native Modelica modeling and simulation platform that has been used here. It can interact with Python through Rest API, e.g. using Jupyter Notebooks [6] or integrating Python scripts directly into the user interface using so called Custom Functions. Thus, allowing an easy integration of physical Modelica models with many AI-based models from Python environment.

3.3 Neural Network model generation

Classical machine learning can be categorized into supervised and unsupervised methods. The goal of the generated surrogate for fuel cell component that can be used to predict fuel consumption from requested electrical power here is to predict data from defined inputs, so to perform a regression task and falls into supervised methods. Generating Neural Networks became a very popular method for Machine Learning, yielding a range of tools. Commonly used tools in the Python environment includes TensorFlow/Keras (developed by Google) and PyTorch (developed by Meta). Also, the Julia language provides an efficient

DOI
10.3384/ecp193157
Proceedings of Asian Modelica Conference 2022
November 24-25, 2022, Tokyo, Japan

environment for generating neural networks and combining with FMUs, Modelica [7] or Modia [8].
The approach presented here relies on using the TensorFlow/Keras toolbox in a Jupyter Notebook environment to generate a neural network from the detailed Modelica model.
For the main question addressed in the presented use-case, identifying the fuel consumption for different operating inputs, it is assumed that the fuel cell dynamics play a minor role and therefore, a quasi-static surrogate based on classical Neural Networks can be used. This assumption will be verified for a specific driving cycle in Section 4.
To train the model, samples of the three independent inputs are prepared, in this case, Saltelli [9] samples are used. Saltelli sampling is an efficient way to reduce the number of necessary data sets while keeping representative behavior over the considered data ranges. The number of Saltelli samples will be

$$\#Sa = N\ (2 * D + 2)$$

Where `D` is the number of free parameters, three in this study and `N` is the requested number of samples each, 10 here. Saltelli's extension of the popular quasi-random low-discrepancy Sobol sequence is used to generate coniform samples of the parameters space. To derive the training data set, a range for the inputs was specified as follows:

Power:	40kW to 140kW
Ambient Temperature:	253K to 333K
Ambient Pressure:	90kPa to 105kPa

As 10 samples in each range where created, the overall number of data sets or required simulation points of the detailed model is 80. Important outputs such as fuel flow, heating power and fuel cell current for the generated datasets are presented in Figure 4.

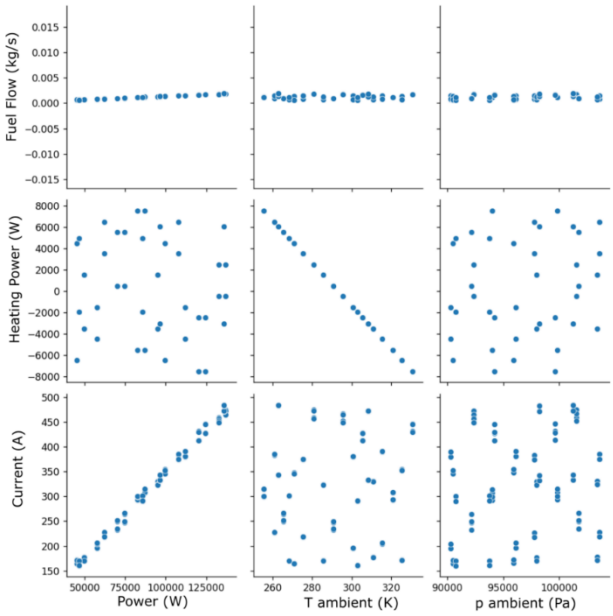

Figure 4: Physical model outputs used as training data. Showing current (bottom), heating power (middle) and fuel consumption (top) for different power (left), ambient temperature (middle) and ambient pressure (right).

Using the TensorFlow/Keras package, a structure for the Feed Forward Neural Network with an input layer, three hidden layers with 5 neurons each and an output layer is defined. Hyperbolic tangent Thanh is used as the activation function for all the neurons. The structure has been defined iteratively, increasing the number of layers and neurons until quantitative agreement of the output could be achieved without setting a specific criterion.

After defining the Neural Network model structure, the training of the weights and biases for all layers is performed using 40 sets of the physical models normalized simulation results while the other 40 normalized output sets are used as test data. During the training epochs, the accuracy (mean of squares of errors against the reference data) of the prediction improves as shown in Figure 5. Normalization of the values is required since working with physical SI units, values will differ several orders of magnitude.

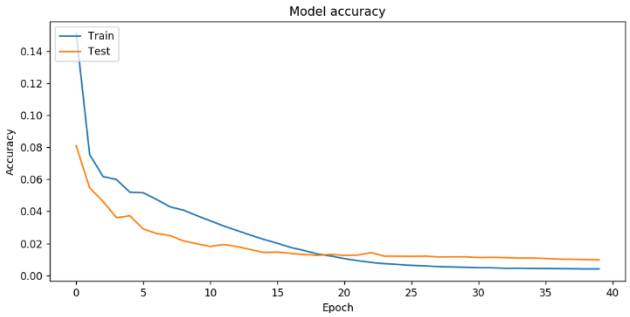

Figure 5: Accuracy of the Neural Network against the reference data from the physical model during training and test.

DOI
10.3384/ecp193157

3.4 Hybrid Model

In the next step, the generated neural network model needs to be transferred back into the Modelon Impact platform. This permits users to benefit from a graphical model representation and convenient parameterization, structural adaptations, and post-processing. The transfer could be implemented in different ways, e.g.:

- FMU: As most system simulation tools, Modelon Impact allows direct import of Functional-Mockup units. This basically provides a wrapped-C-code with a standardized interface. However, FMU export from Python environments is currently under development and turned out to be not reliably working here.
- Using external C-code directly: a similar approach consists in converting the surrogate model into C-code. This approach is similar to the previous FMU approach with a less strict requirement on compliant FMU wrappers, however, since the graphical representation would first need to be created, this approach was not followed here.
- Implementing in Modelica: Introduced and published as “Neural Network Library” by [10], the structure of the Neural Network can be stored as Modelica code directly. Individual Layers can be represented by models, a set of pre-defined activation functions are available, the coefficients for weights and biases for each layer can be stored in the Modelica code or as external data file.

The approach presented in this paper will rely on the Neural Network Modelica library. The main advantage is the absence of compatibility issues and the availability of a graphical network representation. A potential disadvantage might be the adaption towards larger and more complex networks and regular structural updates during iterative surrogate generation processes. Figure 6 Shows the Neural Network structure as a Modelica model. It contains three generic inputs (u1, u2, u3), the input layer as well as the three hidden layers as introduced in the previous section. Connections between the layers and to the output (y) are vectorized.

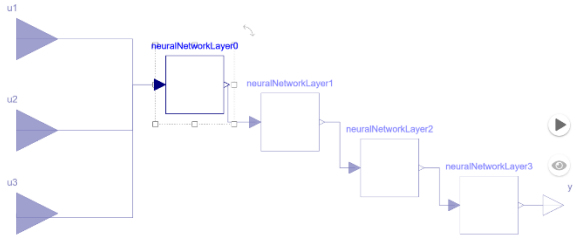

Figure 6: Neural Network Modelica model structure

4 Benchmark

To benchmark and validate the Neural Network surrogate, a comparison on accuracy, model complexity metrics and performance data against the original Modelica model is done. Figure 7 shows normalized fuel consumptions for the original model vs. the TensorFlow prediction and the Modelica surrogate. While both predictions usually match well as expected, some deviations from the original model can be observed due to the simple Neural Network structure used here. Also, minor deviations between the surrogate from TensorFlow against the surrogate based on Modelica can be observed.

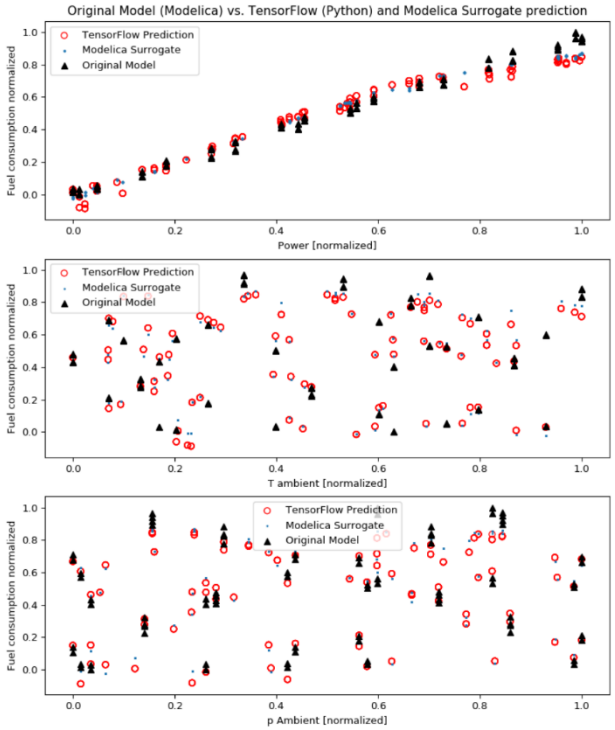

Figure 7: comparing steady-state result points for varying power (top), varying ambient temperature (center) and varying ambient pressure (bottom)

In addition to the steady-state analysis, a transient scenario has been considered for comparison, involving a scheduled load change of the fuel cell power setpoint as shown in Figure 8 while keeping the ambient temperature and pressure constant.

DOI
10.3384/ecp193157
Proceedings of Asian Modelica Conference 2022
November 24-25, 2022, Tokyo, Japan
161

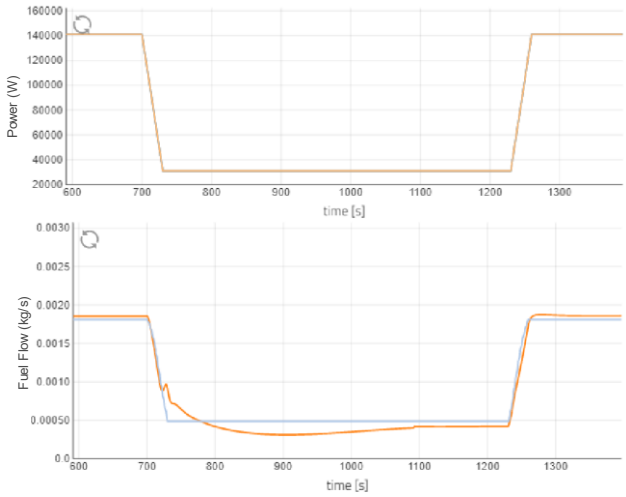

Figure 8: Fuel Cell load setpoint for both the physcial model (orange) and the surrogate (blue) model (top) and consumed hydrogen fuel flow for the load change scenario comparing the results of the original model with the surrogate (bottom)

The presented comparison of the fuel consumption showing well matching steady-state points with maximum differences of 2.7% in full load and 3.6% in low load. On the transients, deviations of physical model from the Neural Network surrogate model can clearly be seen. Effects, such as a control oscillation around 720s caused by the internal cooling flow supply of the fuel cell stack can not be reproduced by the steady-state surrogate.

Key metrics for complexity and performance are presented in Table 1, showing the superior performance of the Neural Network which is around 500 times faster compared to the physical model for this scenario.

Table 1: Complexity statistics and CPU time of surrogate model vs. physical model

	Surrogate	Physical Model
CPU Time	0.12s	49s
Continuous states	0	82
Variables	202	2572
Linear-Equations Blocks	0	13

Based on the performed analysis, it can be concluded that for prediction of the fuel consumption for a certain load variation within the trained data range, the Neural Network model can give reasonable results while being significantly better performing due to the removal of unused complexity. For the overall fuel consumption calculation in this artificial scenario, the accuracy is considered sufficiently well comparing with state-of-the art range predictions in modern fuel-cell vehicles that usually don't consider as many input parameters.

5 Simulation Scenario

A use-case scenario for the Neural Network model of the fuel-cell car could be a fuel consumption calculation of the vehicle for a given route the driver selects in the cars navigation system at different ambient conditions. While the physical model is validated and would be able to predict accurately from first principles, the model execution would take too long for this use-case. This fact becomes even more important considering the usage of lower-performance hardware used in automotive applications due to cost and weight advantages. Therefore, the usage of the surrogate model is proposed for predicting the fuel consumption of the car in a driving cycle, specifically the WLT2P-C2 introduced in section 2. The resulting fuel consumption for this scenario including the possible variation with changing boundary conditions is illustrated in Figure 9. Plausible outcome can be assumed based on the benchmark tests carried out in the previous sections.

The overall CPU time answering the specific question on "how much hydrogen will the vehicle consume for the given route under the different environmental conditions?" was about 2 seconds for a varying load including 5 sets of ambient temperature and 5 sets of ambient pressures, resulting in a total of 25 simulation scenarios.

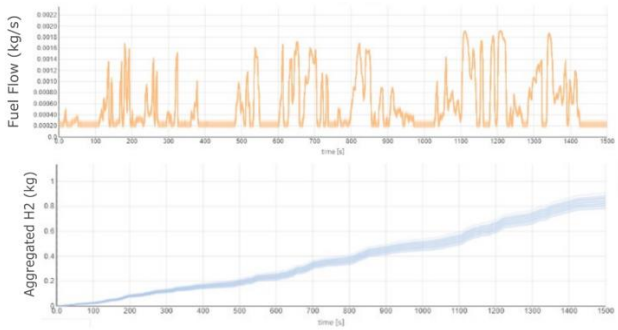

Figure 9: Resulting hydrogen consumption of the fuel cell for the WLTP2 driving cycle scenario with varying ambient temperatures and pressures (top) and aggregated hydrogen consumption of the fuel cell (bottom)

6 Summary and Outlook

The presented hybrid approach provides a powerful complementary feature to first principle based physical modeling which is typically used in Modelica models. The potential performance improvement has been demonstrated on an automotive fuel-cell use case, showing that simulation speed can easily be improved by a factor of 500 when only few outputs of a detailed model are relevant. The integrated Python interface in Modelon Impact allowed a convenient, scripting

DOI
10.3384/ecp193157

interface to TensorFlow/Keras and an easy data usage of physical model results for Neural Network training. The Modelica Neural Network library [10] enabled the usage of the generated Neural Network within the Modelica environment. The developed workflow can therefore be easily used for a variety of applications, including the speedup process of complex physical models or their sub-models for faster model-based design or improvement processes. In addition, application specific proper models can be generated and exported, e.g. as an FMU allowing the utilization for a subset of relevant questions while benefiting from tremendous performance improvements. One commonly known limitation of Neural Network models not addressed here is the usage outside the training data range. Unlike physical models, Neural Network models cannot be expected to predict behavior that has not been sufficiently covered by training data. This can result in very wrong predictions. In addition, further work is needed on capturing transient effects, as Feed-Forward Neural Network approaches only allow a steady state representation. However, the tight integration into the Modelica environment presented here allows the coupling with transient state representations at various points of a system model thus providing a promising solution for this challenge.

Acknowledgements

This research work has been partly funded by the Federal Ministry for Economic Affairs and Climate Action in the framework of PHyMoS – Proper Hybrid Models for Smarter Vehicles, Ref: 19I20022B, 2021-2024.

The models and workflows presented here have been developed by multiple contributors. Special thanks to Bryan Eisenhower for pioneering the work on surrogate modeling and developing workflows around the Modelica / Modelon eco-system.

7 References

[1] A. Daw, A. Karpatne and W. Watkins, Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling, http://arxiv.org/abs/1710.11431, 2017.

[2] S. Sigfridsson, "Fuel Cell Hybrid Vehicle Modeling," Masther Thesis, Lund University, Department of Automatic Control , 2018.

[3] S. Sigfridsson, L. Li, H. Runvik, J. Gohl, A. Joly and K. Soltesz, "Modeling of Fuel Cell Hybrid Vehicle in Modelica: Architecture and Drive Cycle Simulation," in *Proceedings of the 2nd Japanese Modelica Conference, Tokyo, Japan, May 17-18, 2018*, Tokyo, Japan, 2018.

[4] "Informal working group on Worldwide harmonized Light vehicles Test," 2012. [Online]. Available: https://unece.org/dhc-12th-session. [Accessed 16 08 2022].

[5] A. Heuermann, P. Hannebohm and B. Bachmann , "Replacing Strong Components with Artificial Neural Network Surrogates in an Open-Source Modelica Compiler," Linköping, 2022.

[6] P. Jupyter, "https://jupyter.org/," 2022. [Online].

[7] T. Thummerer, . J. Tintenherr and L. Mikelsons, "Hybrid modeling of the human cardiovascular system using NeuralFMUs," *Journal of Physics: Conference Series*, vol. 2090, 2021.

[8] F. Bruder and L. Mikelsons, "Modia and Julia for Grey Box Modeling," Linköping, 2021.

[9] J. Herman and W. Usher, "SALib: An open-source Python library for sensitivity analysis.," *Journal of Open Source Software*, vol. 2(9), 2017.

[10] F. Codeca and F. Casella, "Neural Network Library in Modelica," 2006.