

Reduced Order Modeling App

Modelon Innovate 2024 – Day 2



Presenter's Info

From Jakarta, Indonesia

Located in Hamburg, Germany

Background in Mechatronics and Computational Mechanics

8 years experience in modeling and simulation

Simulation engineer at Modelon

- Development of Modelica libraries
- Customer projects
- Technical support



Nirmala

nirmala.nirmala@modelon.com

But first..

Modelon Impact Web-App basic

Introductory session by Erik Durling available as recording

- ✓ Jupyter - Voila - Modelon Impact UX
- ✓ Custom user interface for various use-cases
- ✓ Framework and workflow for sharing web-app

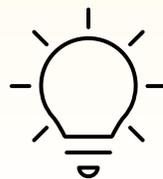
Why reduced order models (ROMs)?

physical modeling:

- Known correlations → implement equations
- High fidelity i.e., accuracy, multitude of parameters

however..

- Physical models → computationally expensive
- Use-case might not need detailed model
- Insufficient knowledge of the governing equations



Generate reduced order models to be used in Modelica environment!

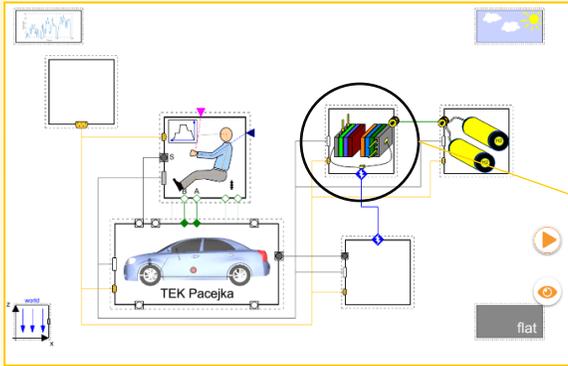
Reduced order modeling workflow

Modelon Impact Web-app to generate neural-network based ROMs

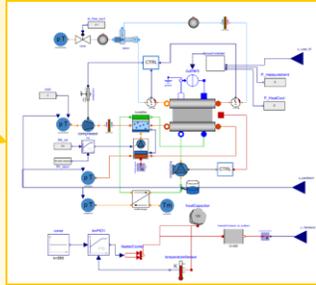
Overview

1

Modelon Impact

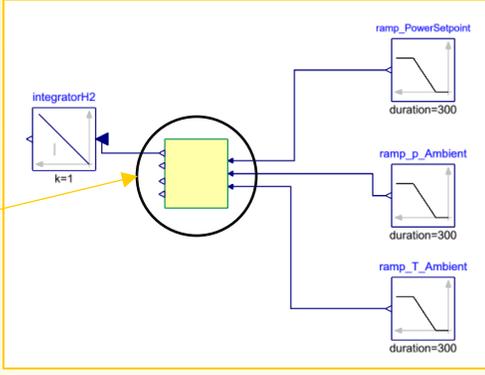


2



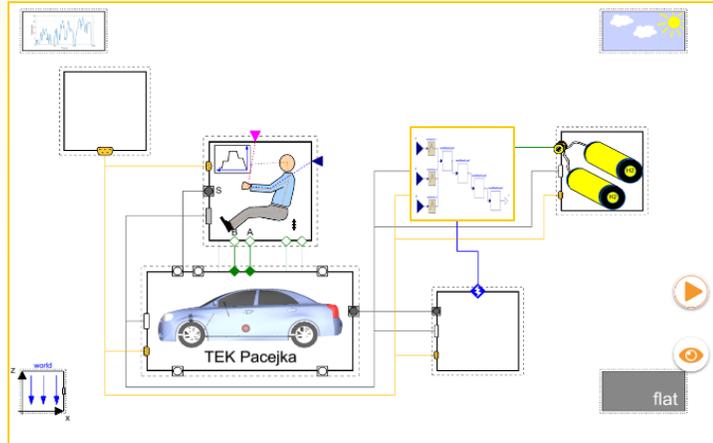
3

Modelon Impact



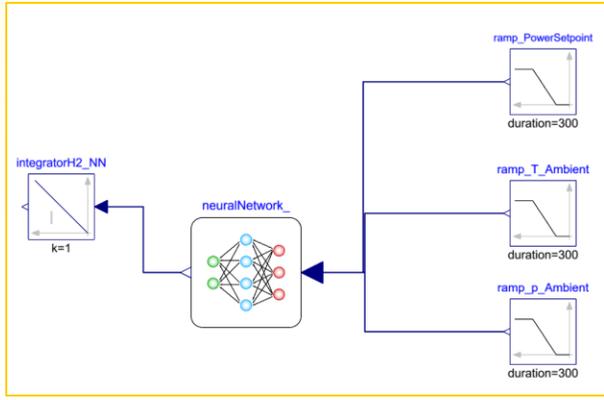
6

Modelon Impact

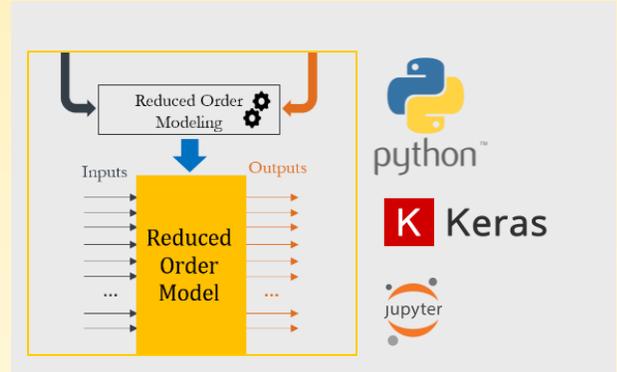


5

Modelon Impact



0



4

voilà

1. Initial Simulation

1. Initial Simulation → 2. Batch Simulation → 3. Network Generation → 4. Network Training → 5. Network Export

- Specify server name, workspace name, and path to executable (physical) model
- Run single simulation to initialize Step 2

Server name:

Workspace name:

Path to executable model:

Model loaded successfully

Starting initial simulation

Finished initial simulation in 24.54 seconds

(in words)

1. Which part of the system I would like to replace?
 - Select subcomponent or subsystem
2. Create wrapper: select input and output, add interfaces
3. Create test bench to generate training data

[Setup Modelica dependencies incl. *NeuralNetwork* open-source library]

4. Deploy web-app*
5. Validate ROM against physical model
6. Use ROM in system

***0 - Web-app setup**

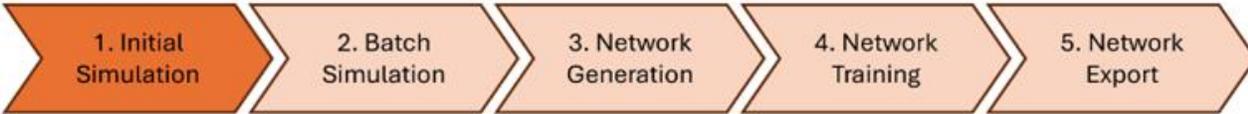
Install with bash script

- Open JupyterLab
- Open terminal from within the project
- Type *bash install.sh*

How to use the web-app

Step 1 – setup and initial simulation

▼ 1. Initial Simulation



- Specify server name, workspace name, and path to executable (physical) model
- Run single simulation to initialize Step 2

Server name: ▼

Workspace name: ▼

Path to executable model:

Model loaded successfully

Starting initial simulation

Finished initial simulation in 16.16 seconds

NOTE:

requires that test bench can be compiled and simulated as it is

Web-app is not meant to debug failing model

Step 2 – data generation

2. Batch Simulation

- Specify network input(s) & output(s) including min/max limits as well as batch simulation setup
- Run batch simulation to generate training data for Step 4

Input(s) and Output(s)

No. of inputs: 4 No. of outputs: 1 Refresh inputs/outputs

Input 1	priminlet_m_flow	Min	0,1	Max	2,1
Input 2	secinlet_m_flow	Min	0,1	Max	2,1
Input 3	priminlet_T	Min	293,15	Max	353,15
Input 4	secinlet_T	Min	293,15	Max	353,15
Output 1	heatExchanger.summary_Q_flow				

Batch Simulation Setup

The results from the batch simulation will be used as training data for the neural network. More data generally leads to a more accurate surrogate, but also requires more time to generate. Maximum batch size is 5000. Different networks can be trained with the same data in the following steps. Sobol distribution is highly recommended.

Distribution method:

Sobol
 Full factorial

No. of cases for Sobol: 250

Input interval for Full Factorial:

4 4 4 4

Simulation end time (s): 10

Start batch simulation

Starting 250 simulations (expected total serial runtime: 3234.6 seconds)

Finished 250 simulations in 40.3 seconds (speedup: 80.3). Click to plot results.

Starting 250 simulations (expected total serial runtime: 4038.8 seconds)

Finished 250 simulations in 41.8 seconds (speedup: 96.6). Click to plot results.

Batch Simulation Setup

The results from the batch simulation will be used as training data for the neural network. More data generally leads to a more accurate surrogate, but also requires more time to generate. Maximum batch size is 5000. Different networks can be trained with the same data in the following steps. Sobol distribution is highly recommended.

Distribution method:

Sobol
 Full factorial

No. of cases for Sobol: 250

Input interval for Full Factorial:

4 4 4 4

Simulation end time (s): 10

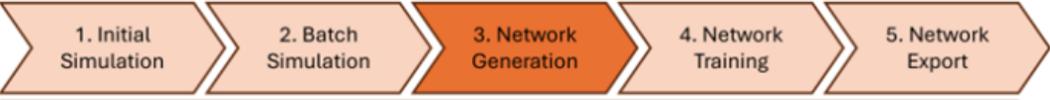
Start batch simulation

Starting 250 simulations (expected total serial runtime: 3234.6 seconds)

Finished 250 simulations in 40.3 seconds (speedup: 80.3). Click to plot results.

Step 3 – neural network structure

▼ 3. Network Generation



- Specify the network structure: layers, neurons per layer, transfer functions, dropout
- Generate (untrained) network structure (TensorFlow)

No. of layers: Refresh layers

No. of neurons for each layer:

Activation function for each layer (last = output layer):

Use dropout (may reduce overfitting):
 False
 True

Dropout rate: 0.30

Neural network generated successfully. Click to display model structure.

NOTE:

Feedforward neural network (FNN) is employed

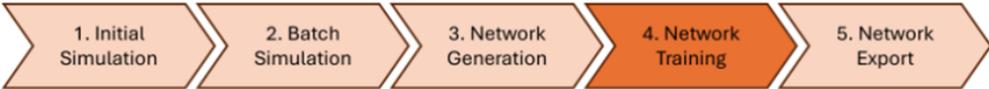
Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	20
dense_1 (Dense)	(None, 4)	20
dense_2 (Dense)	(None, 4)	20
dense_3 (Dense)	(None, 4)	20
dense_4 (Dense)	(None, 1)	5

=====
Total params: 85 (340.00 Byte)
Trainable params: 85 (340.00 Byte)
Non-trainable params: 0 (0.00 Byte)

Step 4 – training

4. Network Training



- Specify training setup
- Train network structure from Step 3 with training data from Step 2 (TensorFlow)

No. of epochs:

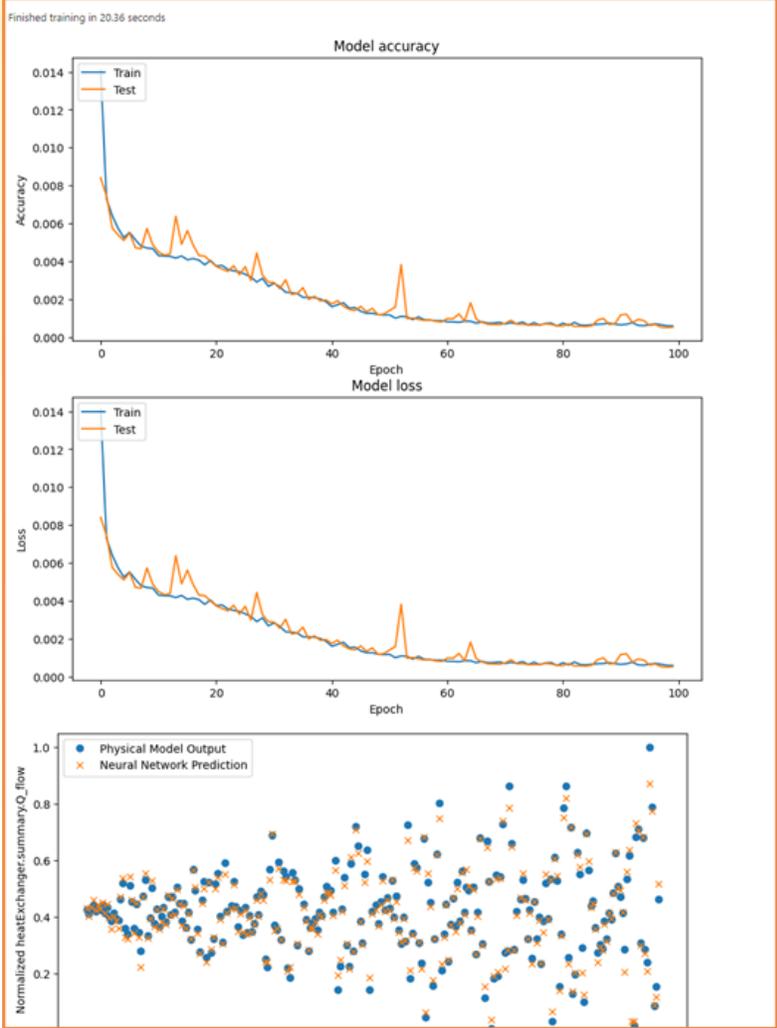
Validation split:

Optimizer used for training:

- Adam
- SGD

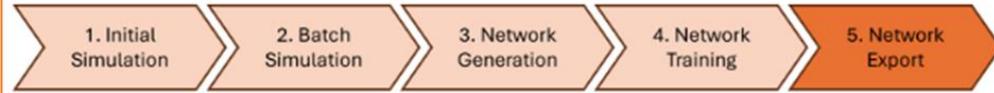
Starting training (this might take a while)

Finished training in 20.36 seconds



Step 5 – export to workspace

5. Network Export



- Specify export path (trained TensorFlow network from Step 4 is translated to Modelica automatically)
- Export network model to workspace to utilize it (be aware of overwriting!)

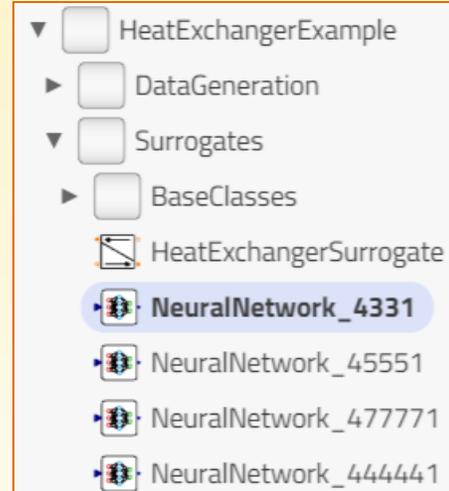
Export path for Modelica network model:

`/home/jovyan/impact/local_projects/Reduced-order-modeling/Library/HeatExchangerExample/Surrogates`

The model's name ends with the network's structural ID, which consists of integers representing the no. of inputs, no. of neurons for each layer, and no. of outputs (e.g. 48882).

NOTE: Exporting will **OVERWRITE** an existing model with the same name!

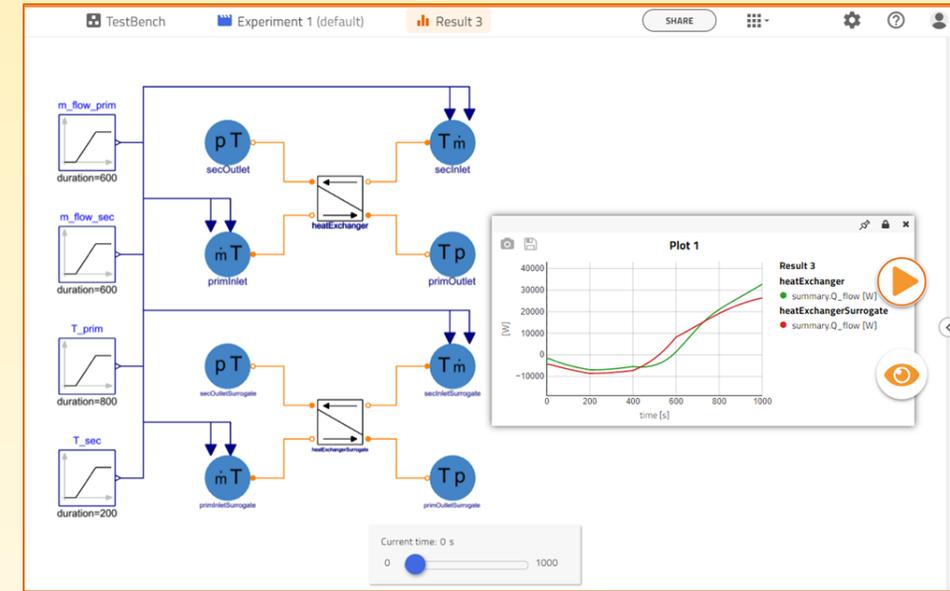
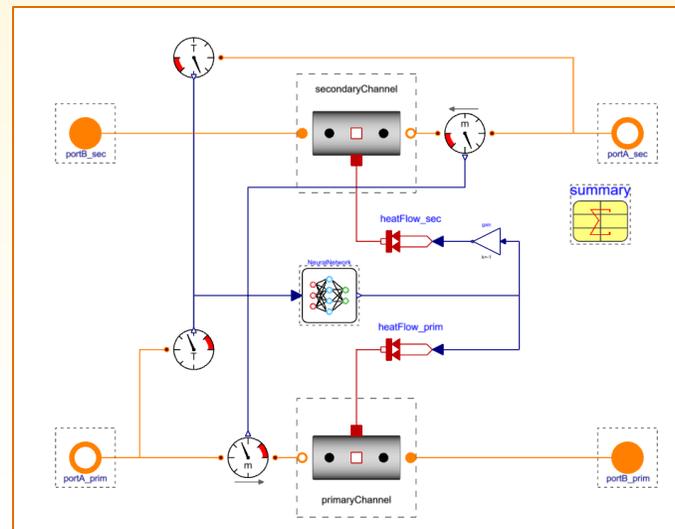
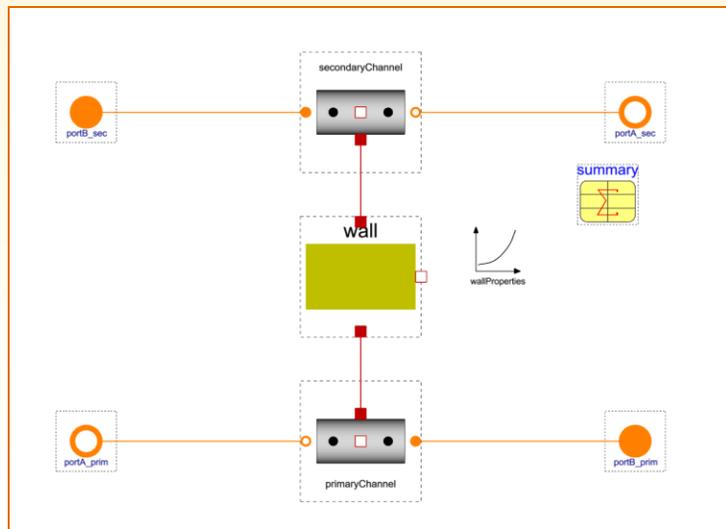
Start network export



Reduced order model examples

Heat Exchanger

- replacing heat exchanger
- input: mass flow rate and temperature for primary and secondary
- output: heat flow rate
- wrapper for reduced order HX:
 - conversion to fluid ports



Input(s) and Output(s)

No. of inputs: 4 No. of outputs: 1 Refresh inputs/outputs

Input	Min	Max
Input 1: primInlet_m_flow	0,1	2,1
Input 2: secInlet_m_flow	0,1	2,1
Input 3: primInlet_T	293,15	353,15
Input 4: secInlet_T	293,15	353,15
Output 1: heatExchanger.summary_Q_flow		

Batch Simulation Setup

The results from the batch simulation will be used as training data for the neural network. More data generally leads to a more accurate surrogate, but also requires more time to generate. Maximum batch size is 5000. Different networks can be trained with the same data in the following steps. Sobol distribution is highly recommended.

Distribution method:

- Sobol
- Full factorial

No. of cases for Sobol: 250

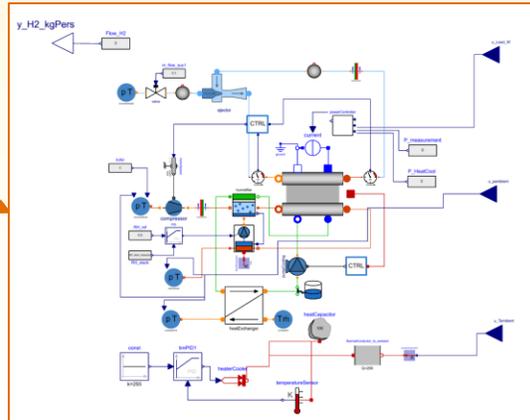
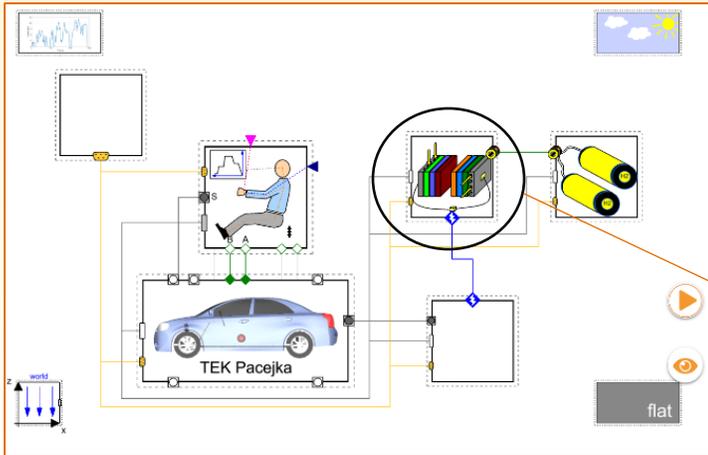
Input interval for Full factorial:

4 4 4 4

Simulation end time (s): 10

Fuel Cell Hybrid Vehicle

- replacing the fuel cell subsystem
- input: power setpoint, ambient pressure and temperature
- output: H2 consumption
- wrapper for fuel cell subsystem
 - scalar input and output connectors



Input(s) and Output(s)

No. of inputs:	3	No. of outputs:	1	Refresh inputs/outputs	
Input 1	targetPower	Min	100e3	Max	150e3
Input 2	T_ambient	Min	293	Max	303
Input 3	p_ambient	Min	1e5	Max	1.1e5
Output 1	fuelFlow				

Batch Simulation Setup

The results from the batch simulation will be used as training data for the neural network. More data generally leads to a more accurate surrogate, but also requires more time to generate. Maximum batch size is 5000. Different networks can be trained with the same data in the following steps. Sobol distribution is highly recommended.

Distribution method:

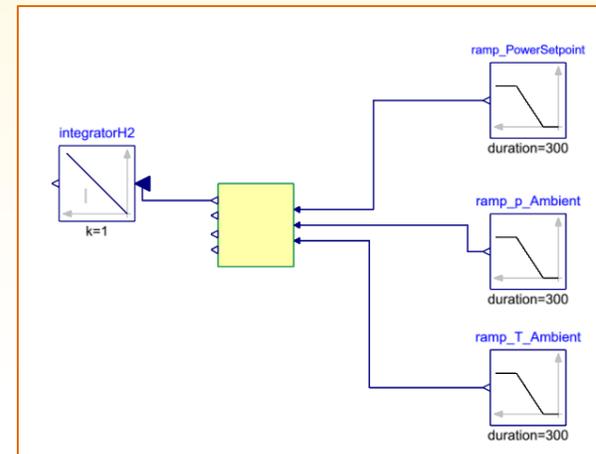
- Sobol
- Full factorial

No. of cases for Sobol: 12

Input interval for Full Factorial:

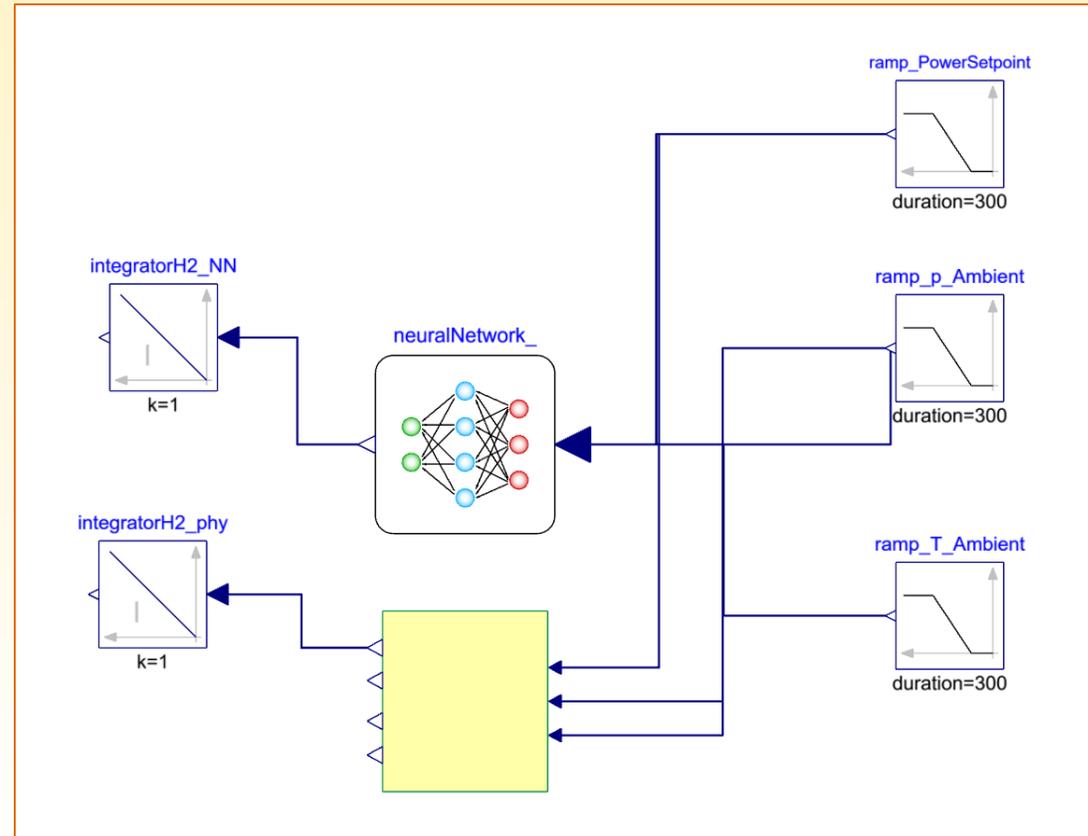
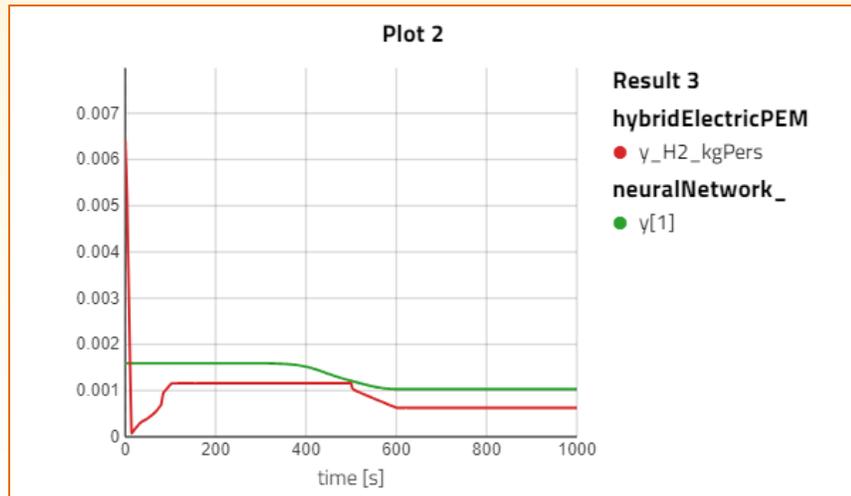
4 4 4

Simulation end time (s): 750



Performance

- Speed-up x5 – x500
- error 1-5%



Demo

Generating reduced order model for heat exchanger

Questions?

References:

PHyMoS project - <https://phymos.de>

NeuralNetwork Modelica library - <https://github.com/AMIT-HSBI/NeuralNetwork>



Modelon Impact client - <https://modelon-impact-client.readthedocs.io/en/latest/>

TensorFlow - <https://www.tensorflow.org/>



Modelon Impact

Meet Modelon Impact – a cloud platform for virtually designing, simulating, and analyzing industrial systems.