

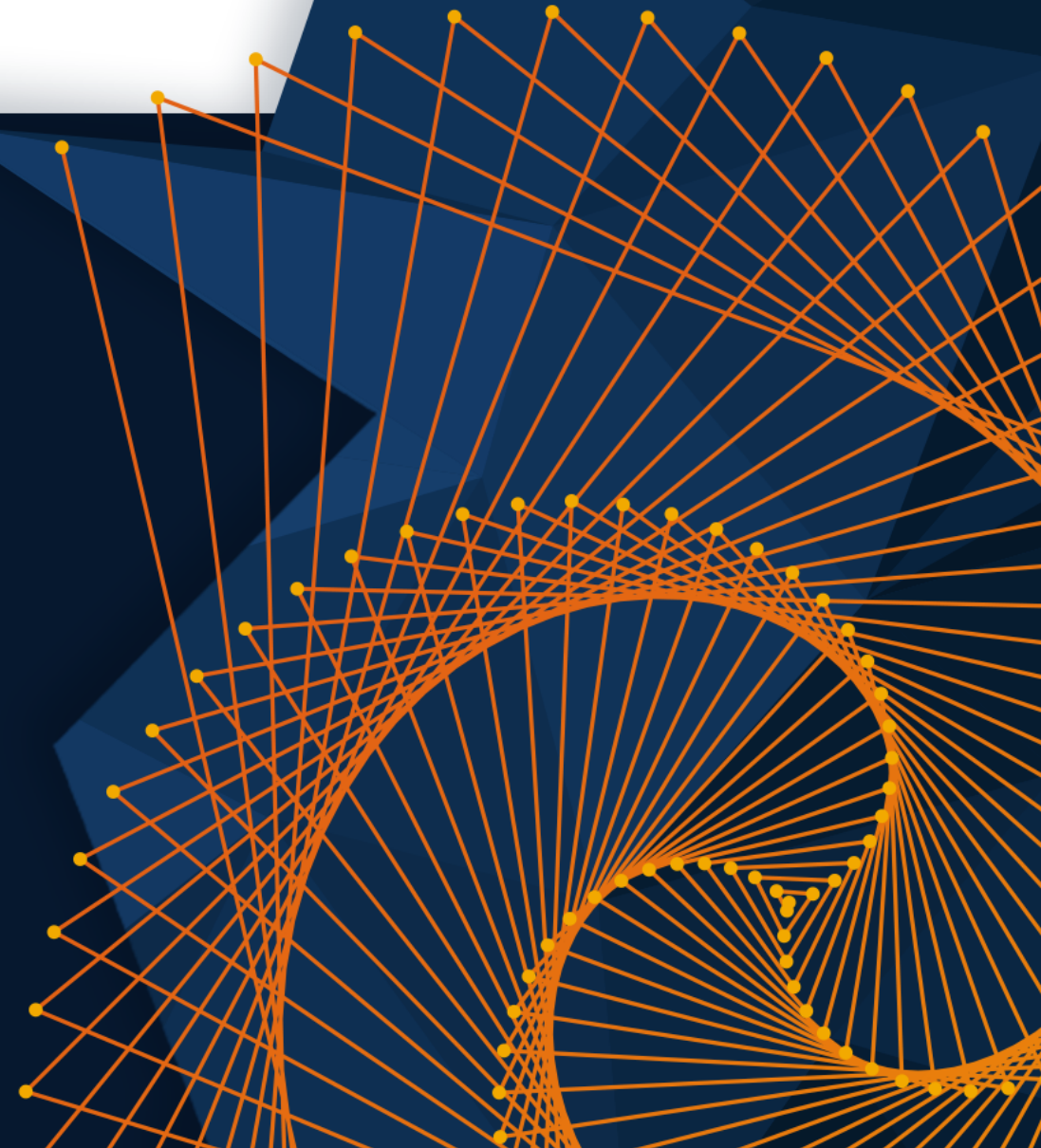
# SCHAEFFLER

## Thermal Neural Network for Temperature Modeling in E-Motors

*Dr. Tobias MORODER, Schaeffler Technologies AG & Co. KG*



MATLAB EXPO



# Outline

1. Motivation
2. Test Setup
3. Thermal Neural Network
4. Workflow In MATLAB / Simulink
5. Results
6. Simulink Implementation
7. MATLAB / Simulink Code
8. Discussion
9. Summary

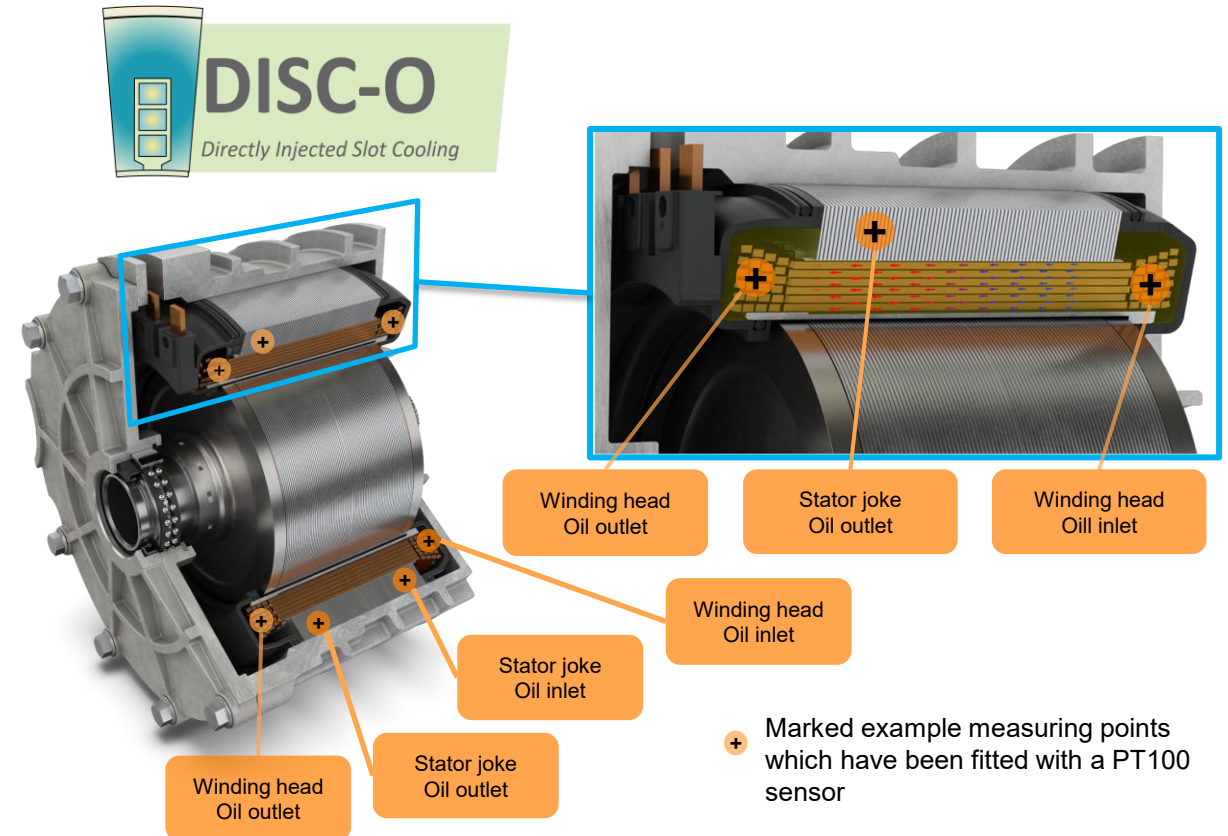
# Motivation – Thermal Management

- Initial Situation

- DISC-O: Novel Schaeffler e-motor prototype cooling at windings directly
- Temperature knowledge is critical for safe and efficient operation
- Virtual sensing by **thermal model**  
→ **Use Physics Informed AI**

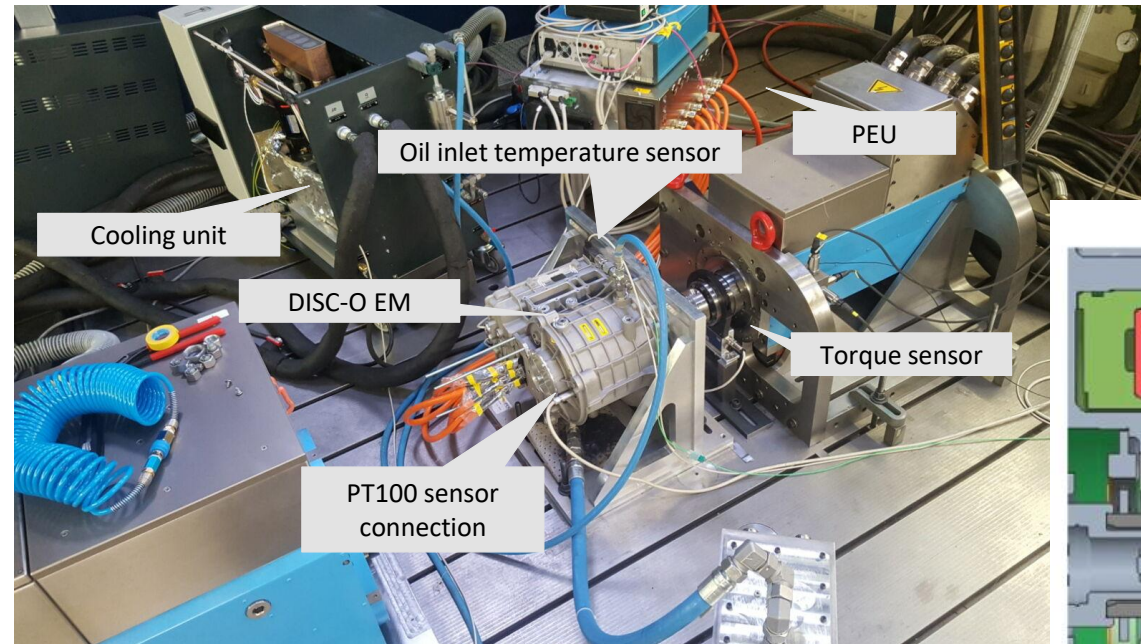
- Requirements on thermal model

- Fast & accurate temperature estimates
- Trust in model
- Initialization strategy
- Variable sample rate
- ...

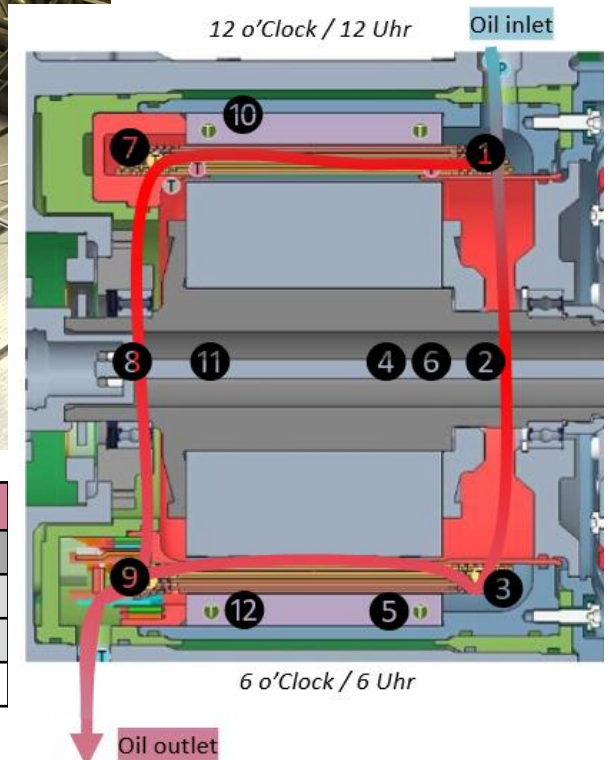


# Test Setup

- Dataset
  - 12 PT100 temperature sensors built into prototype
  - Various drive cycles are recorded and their selection and control are targeted for AI model training
  - 300h total recording
  - [K. Wolter et al., VDI-Berichte 2445, 367 \(2024\)](#)



	Oil inlet			Oil outlet		
	12 o'clock	3 o'clock	6 o'clock	12 o'clock	3 o'clock	6 o'clock
Winding Head	XG1 ①	XG2 ②	XG3 ③	XG7 ⑦	XG8 ⑧	XG9 ⑨
Stator Yoke		XG4 ④	XG5 ⑤	XG10 ⑩	XG11 ⑪	XG12 ⑫
Stator Tooth		XG6 ⑥				

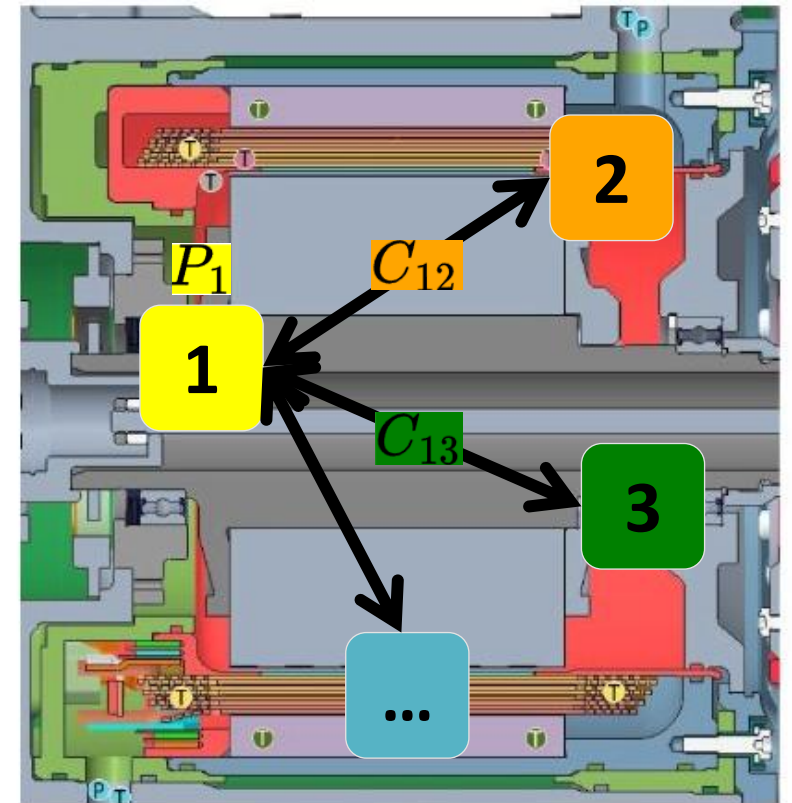


# Thermal Neural Network (TNN)

- [W. Kirchgässner et al., Eng. Appl. Artif. Intell., vol. 117 \(2023\)](#)
- Combines engineering know-how and machine learning  
→ Lumped Parameter Thermal Networks
- Temperature can change due to:
  1. Heat produced in region = Power loss
  2. Heat flow from/to neighbours → Proportional to temperature difference
- Example

$$T_1 \leftarrow T_1 + K_1 \Delta t [P_1 + C_{12}(T_2 - T_1) + C_{13}(T_3 - T_1) + \dots]$$

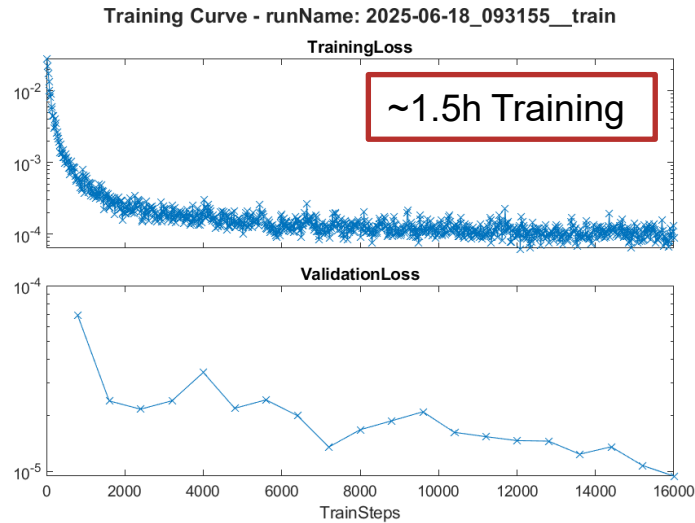
- TNN procedure
  1. Initialize temperatures
  2. Estimate power losses  $P$  and conductances  $C$  via neural nets
  3. Use update equation (with inverse capacitances  $K$ )



# Workflow In MATLAB / Simulink

- Steps

- Prepare data and define model
- Run model training
- Evaluate model
- Export trained model to Simulink
- Evaluate the model in Simulink
- Run code generation



tnn =

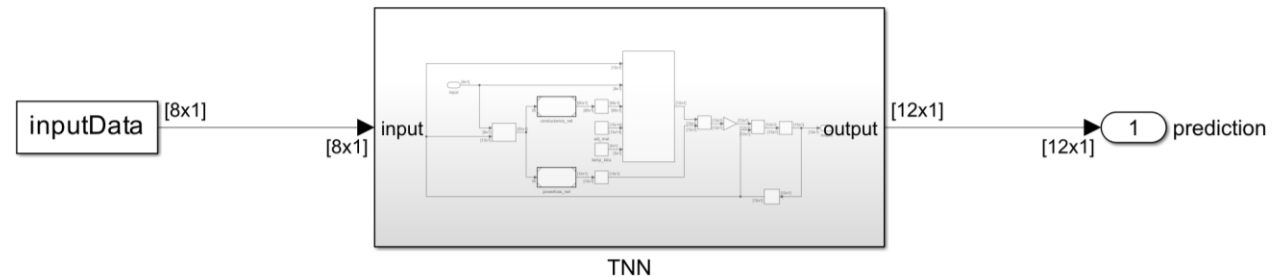
[dlnetwork](#) with properties:

```

Layers: [1x1 DiffEqLayer]
Connections: [0x2 table]
Learnables: [11x3 table]
State: [0x3 table]
InputNames: {'layer/in1' 'layer/in2'}
OutputNames: {'layer/out1' 'layer/out2' 'layer/out3' 'layer/out4'}
Initialized: 1
    
```

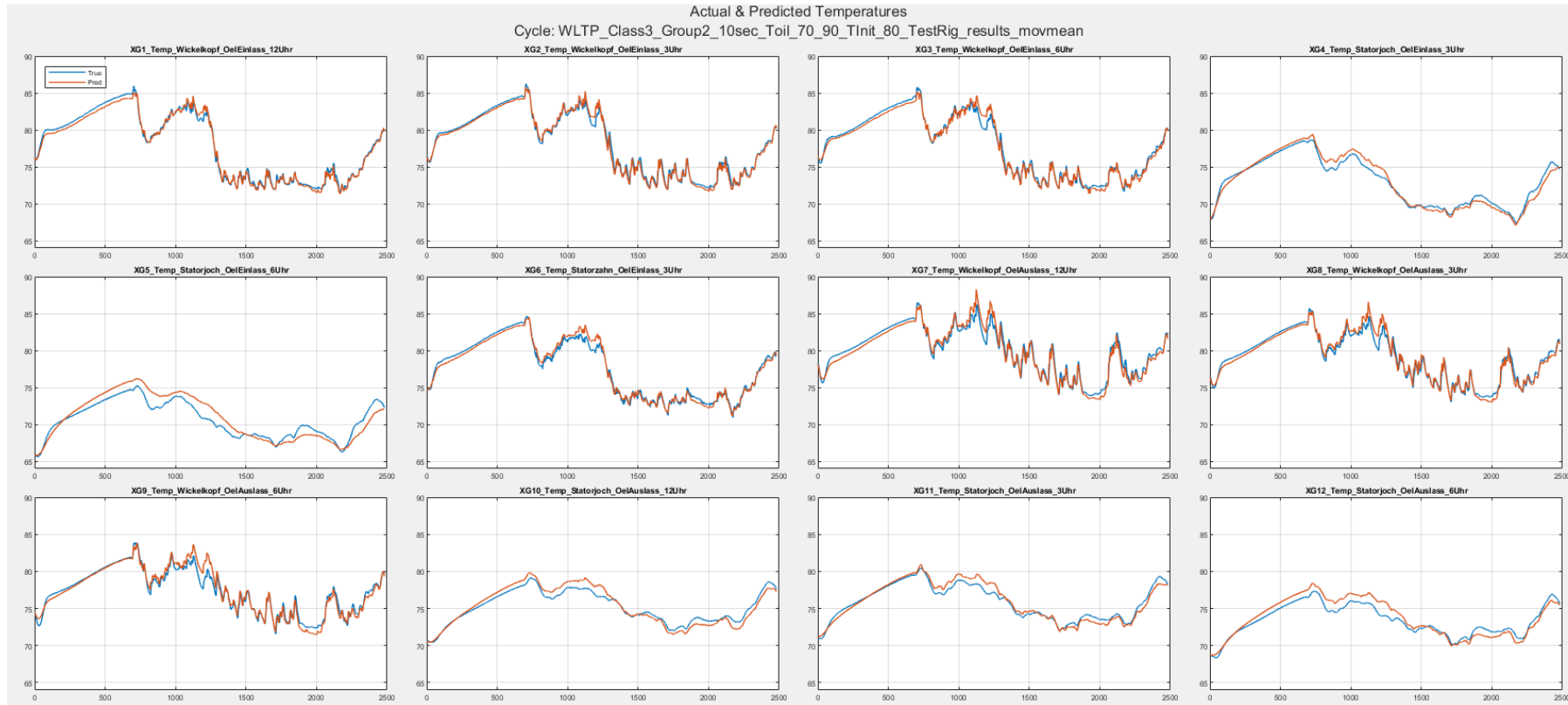
View summary with [summary](#).

	MEAN	XG1_Temp_Wickelkopf_OelEinlass_12Uhr	XG2_Temp_Wickelkopf_OelEinlass_3Uhr
<b>MAX</b>	2.5051	1.78	2.2953
<b>RMSE</b>	0.577	0.33267	0.36611
<b>MAE</b>	0.4515	0.26063	0.28163
<b>R2</b>	0.99825	0.99961	0.99952
<b>CPK</b>	3.2484	5.1608	4.545



# TNN Results

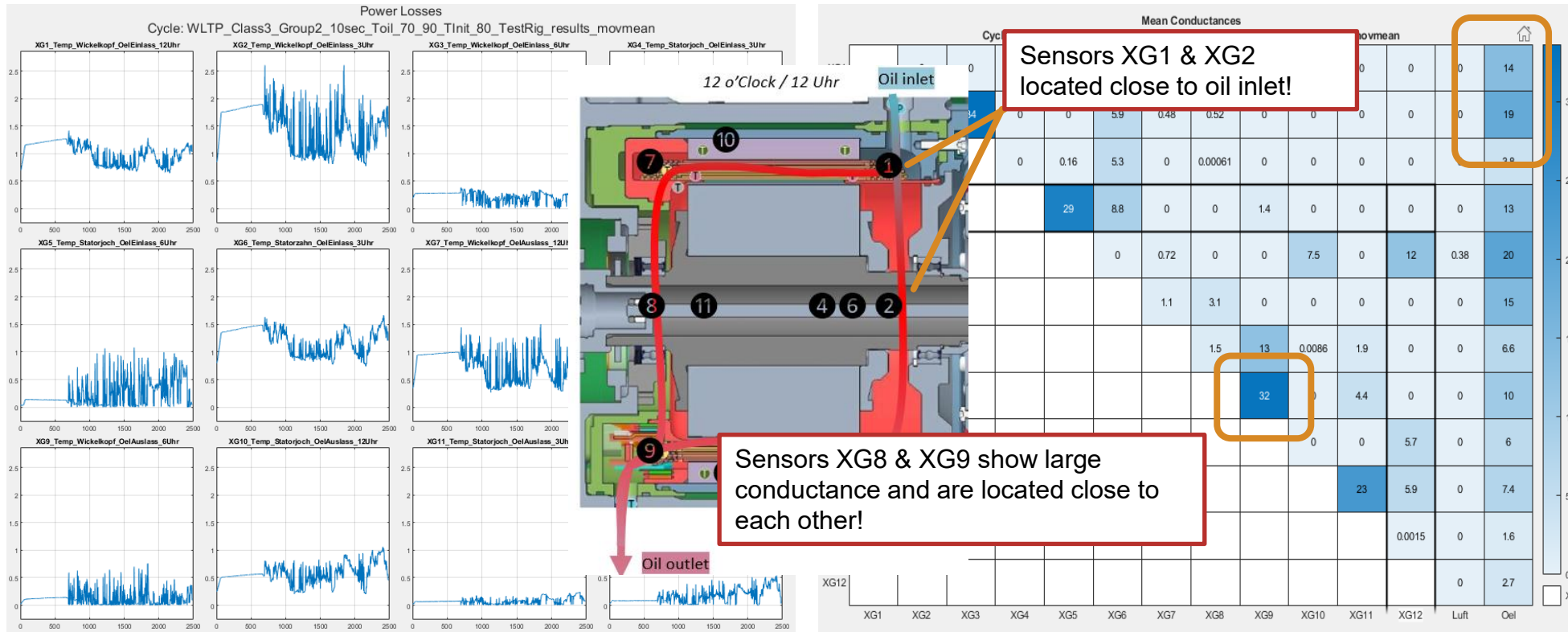
## Prediction Accuracy



- Excellent temperature prediction, meets requirements of  $\pm 5^{\circ}\text{C}$
- Better accuracy than pure ML approaches like MLP or LSTM

# TNN Results

## Interpretable / Explainable Components



- Power losses and thermal conductances (heat flow) yield additional insights and trust in ML model → Explainable AI

# TNN Results

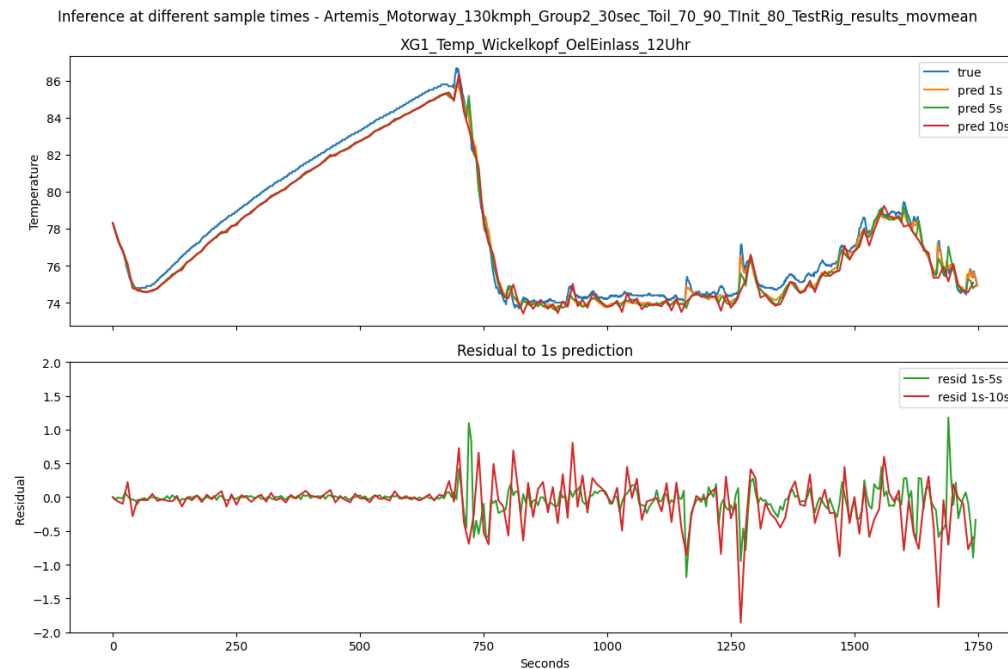
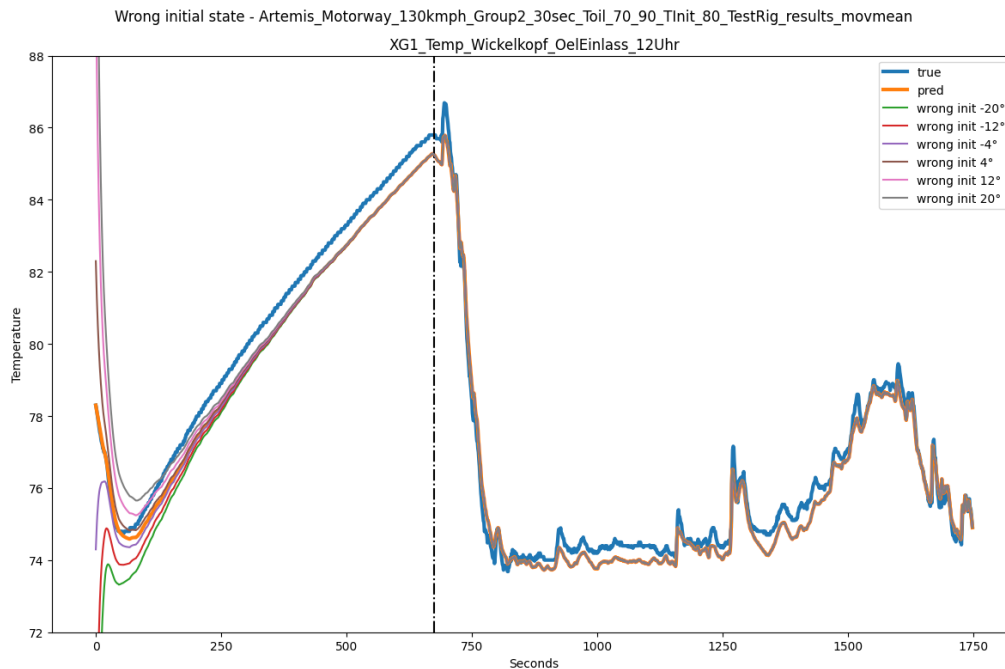
## Other Advantages

- Initialization

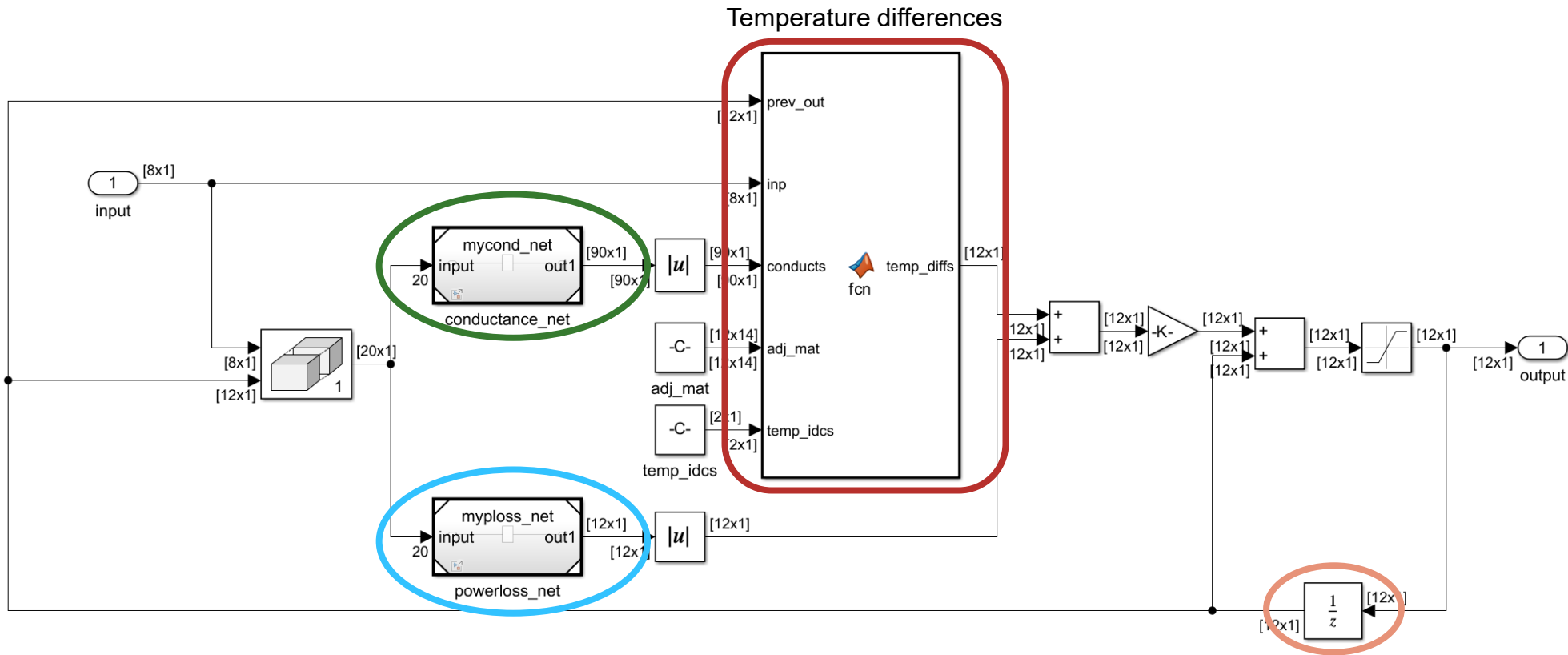
- Robust to initialization errors
- Hard for black box ML (LSTM, ...)

- Variable sample time

- *E.g.*, use 1 sec trained model on other inference sampling time  $\Delta t$



# Simulink Implementation



$$T_1 \leftarrow T_1 + K_1 \Delta t [P_1 + C_{12}(T_2 - T_1) + C_{13}(T_3 - T_1) + \dots]$$

# MATLAB / Simulink Code – How To Try It Yourself?

The screenshot shows the GitHub interface for the repository 'wkirgsn / thermal-nn'. The left sidebar displays the file structure, including folders like 'img', 'matlab', 'ref\_images', 'resources/project', 'src', and 'helper', along with various MATLAB and Python files. The main content area shows the 'matlab' directory with a commit history table and a README file.

Name	Last commit message	Last commit date
..		
ref_images	initial commit of MATLAB implementation	last month
resources/project	initial commit of MATLAB implementation	last month
src	initial commit of MATLAB implementation	last month
.gitignore	initial commit of MATLAB implementation	last month
README.md	initial commit of MATLAB implementation	last month
SECURITY.md	initial commit of MATLAB implementation	last month
TNN.prj	initial commit of MATLAB implementation	last month
license.txt	initial commit of MATLAB implementation	last month

**README.md**

### Thermal neural network for electric motor temperature estimation

This is a MATLAB® implementation of the thermal neural network as introduced by Kirchgässner et al. [1]. The code is the result of a collaboration with [Schaeffler Technologies AG & Co. KG](#). The underlying application is a virtual sensor for estimating the temperature within an electric motor based on other available sensor data. The code in this repo requires that the data file [2] (measures\_v2.csv) has previously been downloaded into a folder data/input relative to the project root. During the project startup, the location of the data file is checked, and if not present, you have the option to download the data automatically from the MathWorks Supportfiles.

The figure consists of eight subplots arranged in a 2x4 grid. Each subplot shows a time series plot of temperature (°C) versus time (s). The top row shows the temperature response for different motor parameters: 'p1', 'p2', 'p3', and 'p4'. The bottom row shows the temperature response for different motor parameters: 'p5', 'p6', 'p7', and 'p8'. Each plot compares the 'measured' temperature (blue line) with the 'estimated' temperature (red line). The plots show that the estimated temperature closely follows the measured temperature, indicating high accuracy of the thermal neural network model.

- Contribution of MATLAB / Simulink implementation to original TNN paper repo: <https://github.com/wkirgsn/thermal-nn/tree/main/matlab>

- Note, this implements the original example and does not contain all features shown here

## Discussion

- TNN can be interpreted as Neural Ordinary Differential Equation
  - Fit first derivative to data

$$\frac{d}{dt}T(t) = f(T(t), t; \theta), \quad T(t_{i+1}) = T(t_i) + \Delta t f(T(t_i), t_i; \theta)$$

- [W. Kirchgässner et al., PEC-Himeji 2022- ECCE Asia, 2746](#)
- Virtual temperatures
  - Add unobserved / const. temperature nodes to structure → Acts as temporal delay
  - TNN is stupid if there is no heat sink, e.g., 1 temperature with power loss only
  - [N. Wiese et al., IEEE Trans. Transp Electrification, vol 11, 870, \(2024\)](#)
  - [A. I. Ramones et al., Energy Conversion and Management: X 27, 101140 \(2025\)](#)
- TNN approach applicable to other scenarios (power electronics, ...)

# Summary

- Thermal Neural Network

- [W. Kirchgässner et al., Eng. Appl. Artif. Intell., vol. 117 \(2023\)](#)
- Hybrid model mixing engineering know-how and machine learning
- Estimates power losses and conductances of Lumped Parameter Thermal Networks

- Advantage

- Excellent temperature estimates
- Explainable components
- Robust initialization by interpretable states
- Variable inference sample rate

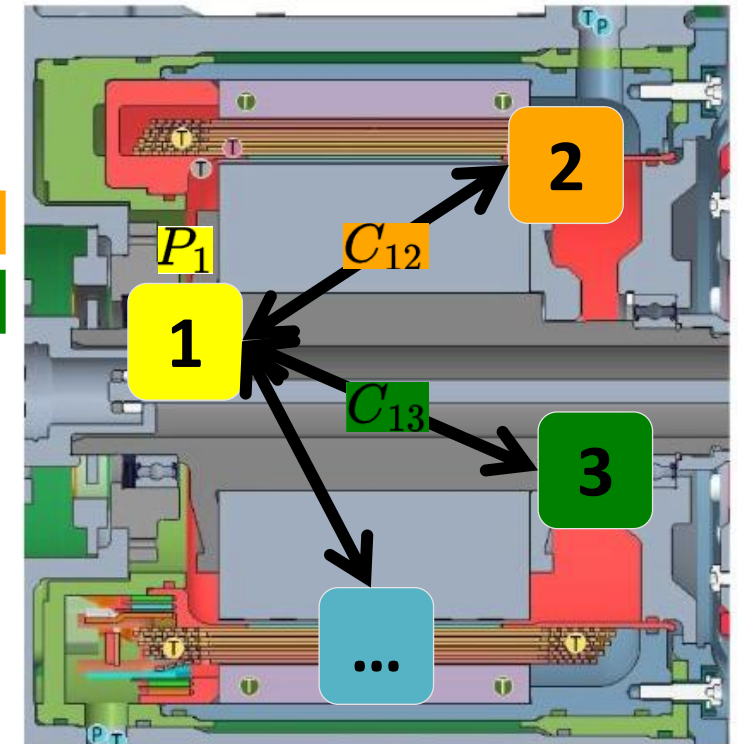
- Try the MATLAB implementation yourself!

- <https://github.com/wkirgsn/thermal-nn/tree/main/matlab>

- Contacts

- Dr. Tobias MORODER, Innovation Cluster Digital Solutions, [morodtbi@schaeffler.com](mailto:morodtbi@schaeffler.com)
- Dr. Georg GÖPPERT, Concepts R&D E-Mobility, [georg.goeppert@schaeffler.com](mailto:georg.goeppert@schaeffler.com)
- Dr. Jonas FUCHS, Innovation Cluster Digital Solutions [Jonas.Fuchs@schaeffler.com](mailto:Jonas.Fuchs@schaeffler.com)
- Marcel ADRIAN, Concepts R&D E-Mobility, [adriamrc@schaeffler.com](mailto:adriamrc@schaeffler.com)

$$T_1 \leftarrow T_1 + K_1 \Delta t [$$
$$P_1$$
$$+ C_{12}(T_2 - T_1)$$
$$+ C_{13}(T_3 - T_1)$$
$$+ \dots ]$$



# MATLAB EXPO

## Thank you



© 2025 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

