MathWorks
AUTOMOTIVE CONFERENCE 2024
North America

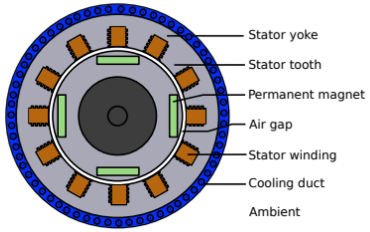
Deep Learning–Based Reduced Order Models for Electric Motors

Shyam P. Keshavmurthy, PhD, MathWorks

MathWorks

Motivation

- EV’s rely on PMSM motors as their Main Traction Device
- Temperature Excursions in these Motors leads to loss of Torque efficiency and eventual failures
- Need test these devices over possible Thermal Regimes
- Dyno testing is costly and can lead to degraded devices
- Simulation is a must, but faster simulations are essential and Virtual Sensors are bonus



Reduced Order Modeling

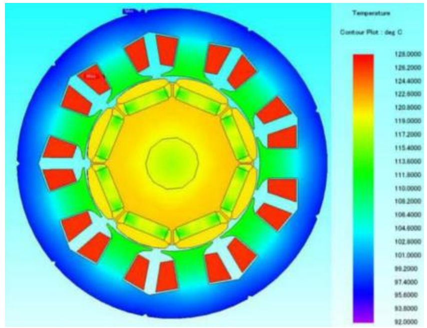
What

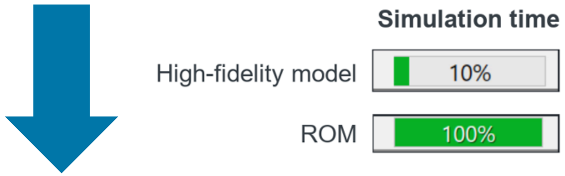
- Techniques to **reduce the computational complexity** of a computer model
- **Provide reduced, but acceptable fidelity**

Why

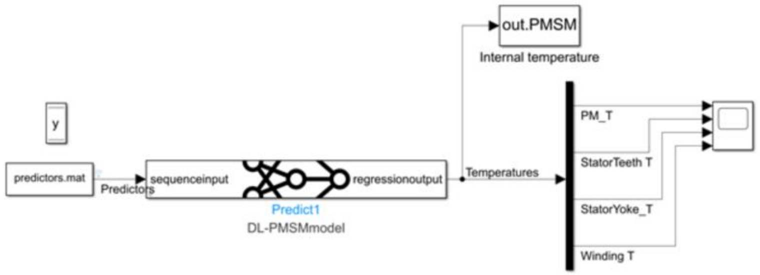
- Enable simulation of FEA models in Simulink
- Perform hardware-in-the-loop testing
- Develop virtual sensors, Digital twins
- Perform control design
- Enable desktop simulations for orders-of-magnitude longer timescales

High-fidelity model

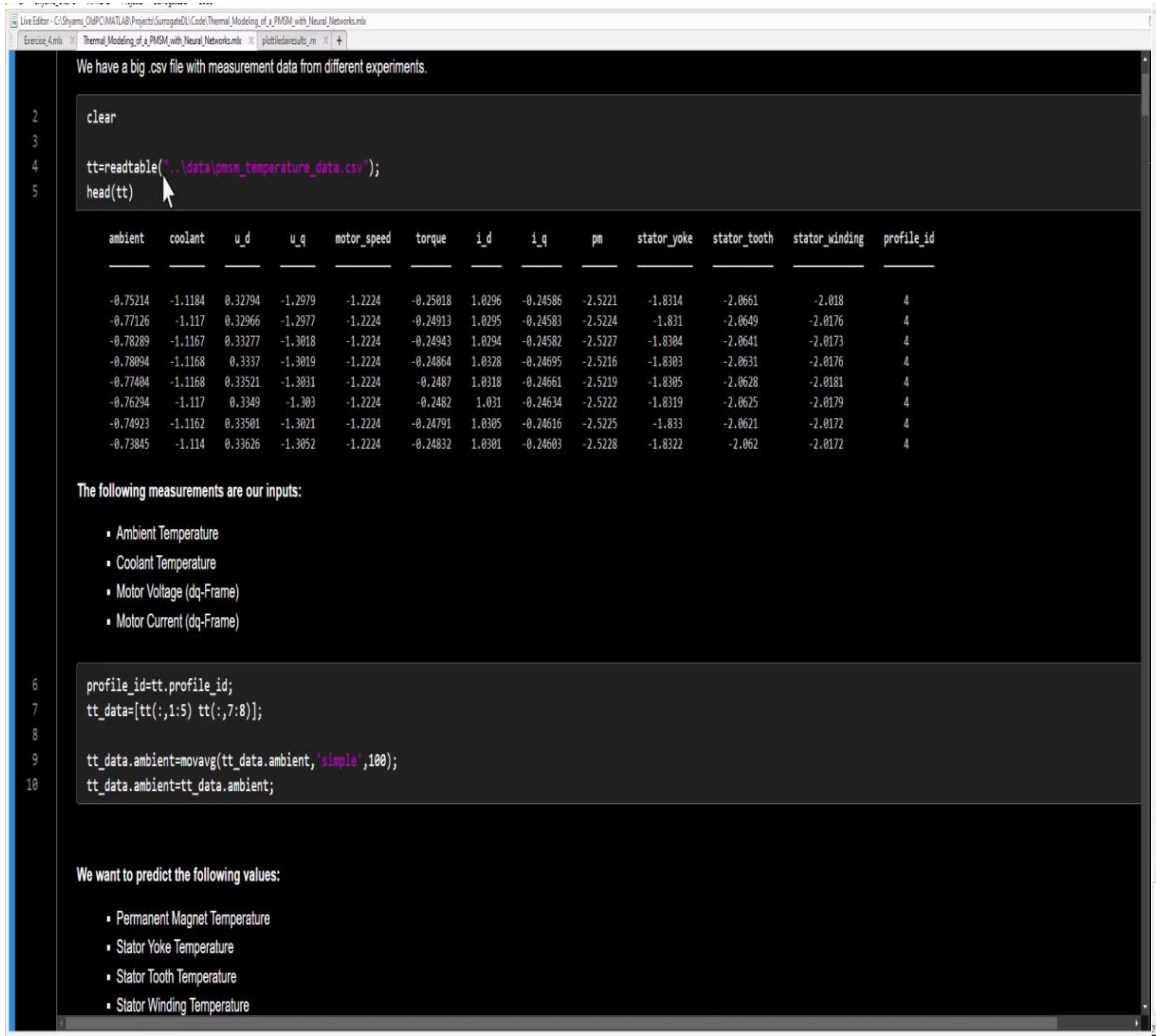




Reduced-Order Model (ROM)



Increase Simulation Speed With Reduced Order Modeling



We have a big .csv file with measurement data from different experiments.

```
clear

tt=readtable("..\data\pmsm_temperature_data.csv");
head(tt)
```

ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
-0.75214	-1.1184	0.32794	-1.2979	-1.2224	-0.25018	1.0296	-0.24586	-2.5221	-1.8314	-2.0661	-2.018	4
-0.77126	-1.117	0.32966	-1.2977	-1.2224	-0.24913	1.0295	-0.24583	-2.5224	-1.831	-2.0649	-2.0176	4
-0.78289	-1.1167	0.33277	-1.3018	-1.2224	-0.24943	1.0294	-0.24582	-2.5227	-1.8304	-2.0641	-2.0173	4
-0.78094	-1.1168	0.3337	-1.3019	-1.2224	-0.24864	1.0328	-0.24695	-2.5216	-1.8303	-2.0631	-2.0176	4
-0.77404	-1.1168	0.33521	-1.3031	-1.2224	-0.2487	1.0318	-0.24661	-2.5219	-1.8305	-2.0628	-2.0181	4
-0.76294	-1.117	0.3349	-1.303	-1.2224	-0.2482	1.031	-0.24634	-2.5222	-1.8319	-2.0625	-2.0179	4
-0.74923	-1.1162	0.33501	-1.3021	-1.2224	-0.24791	1.0305	-0.24616	-2.5225	-1.833	-2.0621	-2.0172	4
-0.73845	-1.114	0.33626	-1.3052	-1.2224	-0.24832	1.0301	-0.24603	-2.5228	-1.8322	-2.062	-2.0172	4

The following measurements are our inputs:

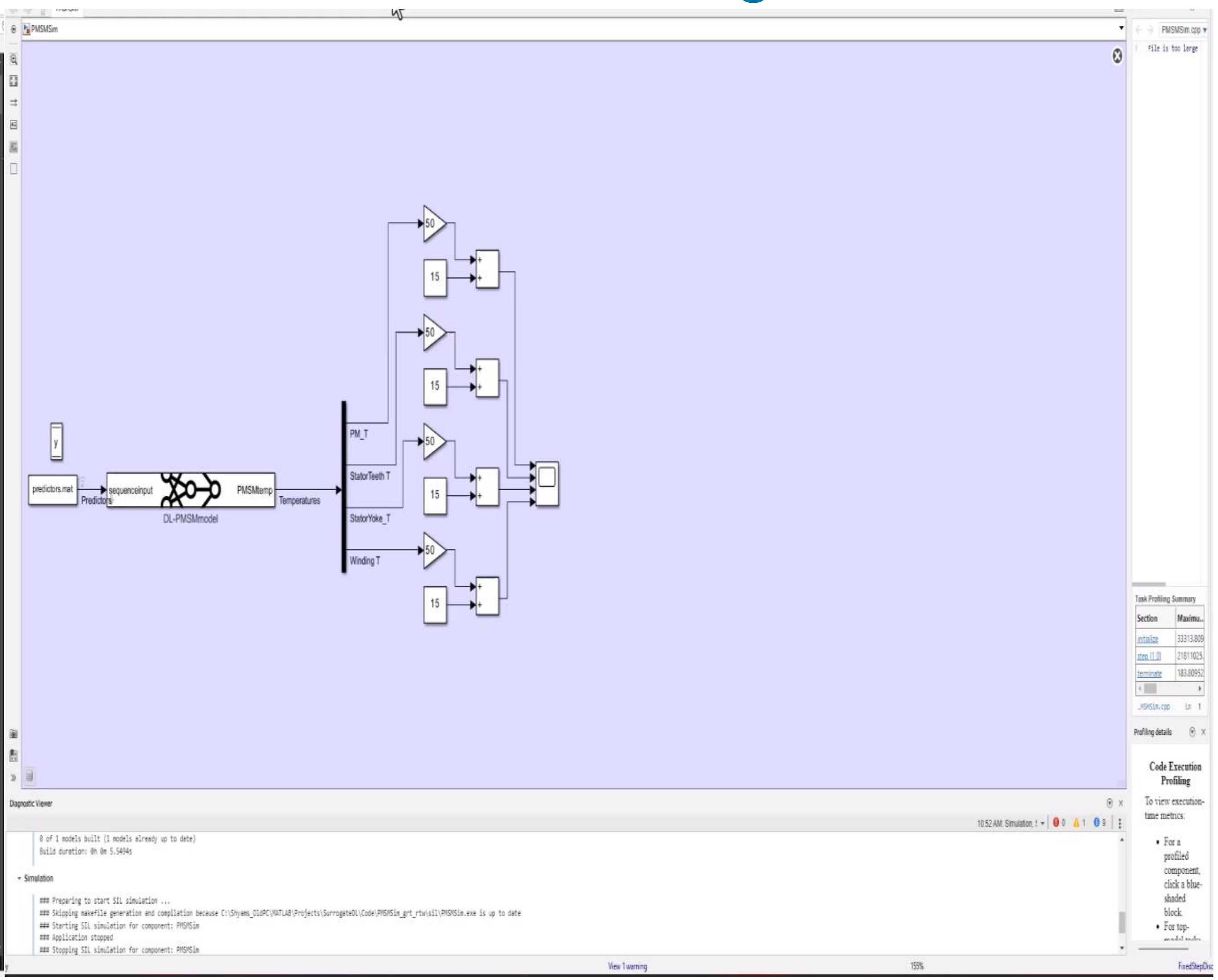
- Ambient Temperature
- Coolant Temperature
- Motor Voltage (dq-Frame)
- Motor Current (dq-Frame)

```
profile_id=tt.profile_id;
tt_data=[tt(:,1:5) tt(:,7:8)];

tt_data.ambient=movavg(tt_data.ambient,'simple',100);
tt_data.ambient=tt_data.ambient;
```

We want to predict the following values:

- Permanent Magnet Temperature
- Stator Yoke Temperature
- Stator Tooth Temperature
- Stator Winding Temperature

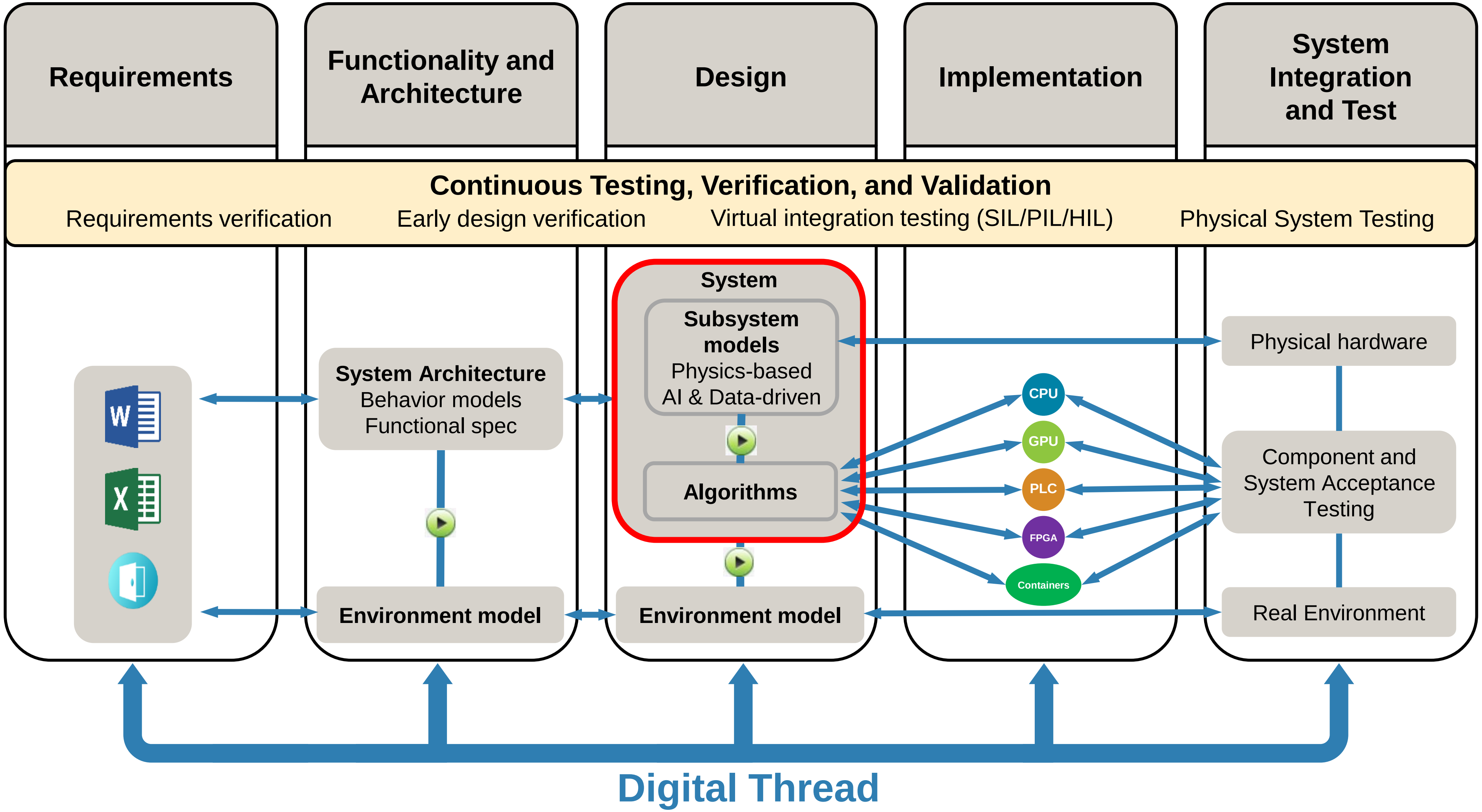


Common Challenges in Operationalizing Models

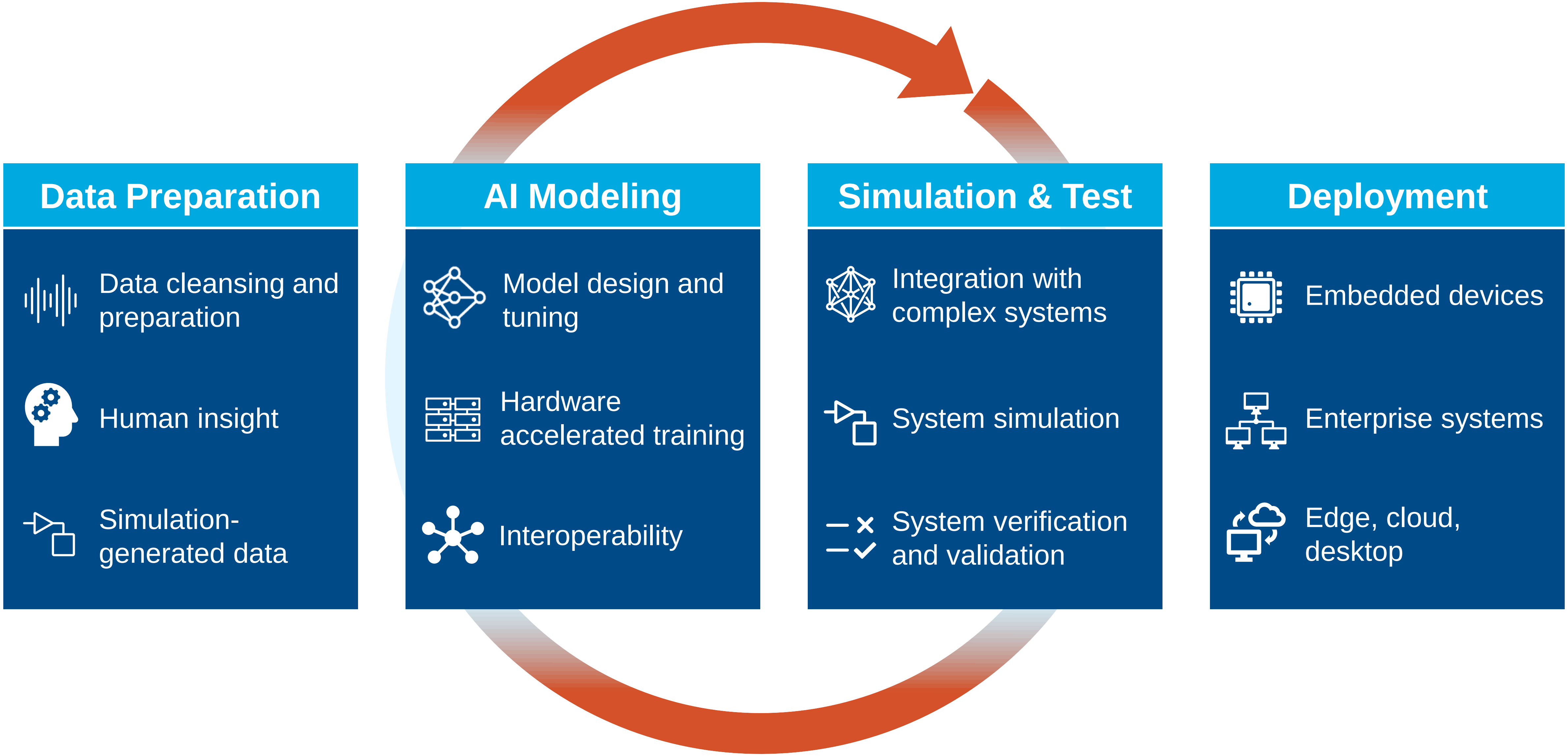
High fidelity models, such as ones from 3rd party FEA tools, are too slow for system level simulation and HIL testing.

Creating a ROM that produces desired results in terms of speed, accuracy, interpretability, etc.

Integrating AI into Model-Based Design (Focus on Subsystem Models)



AI-driven system design



AI-driven system design

Data Preparation

- Data cleansing and preparation
- Human insight
- Simulation-generated data

Data Source

Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors

Wilhelm Kirchgässner
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
kirchgaessner@lea.uni-paderborn.de

Oliver Wallscheid
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
wallscheid@lea.uni-paderborn.de

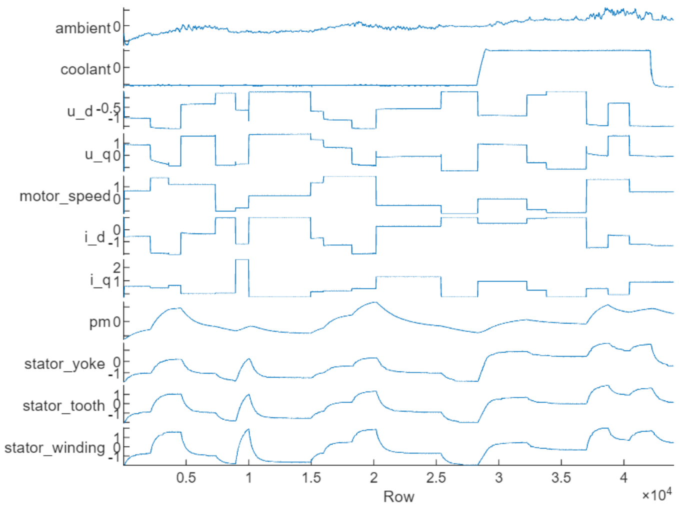
Joachim Böcker
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
boecker@lea.uni-paderborn.de

***Abstract*—Most traction drive applications using permanent magnet synchronous motors (PMSMs) lack accurate temperature monitoring capabilities so that safe operation is ensured through expensive, oversized materials at the cost of its effective utilization. Classic thermal modeling is conducted with e.g. lumped-parameter thermal networks (LPTNs), which help to estimate internal component temperatures rather precisely but also require expertise in choosing model parameters and lack physical interpretability as soon as their degrees of freedom are curtailed in order to meet the real-time requirement. In this work, deep recurrent and convolutional neural networks with residual connections are empirically evaluated for their feasibility on the sequence learning task of predicting latent high-dynamic temperatures inside PMSMs, which, to the authors' best knowledge, has not been elaborated in previous literature. In a highly utilized PMSM for electric vehicle applications, the temperature profile in the stator teeth, winding, and yoke as well as the rotor's permanent magnets are modeled while their ground**

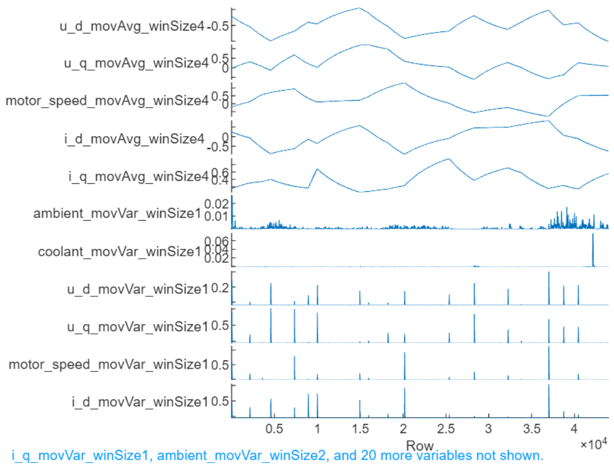
precise thermal state, yet for the rotor part, it is technically and economically infeasible due to an electric motor's sophisticated internal structure and the difficult accessibility of the rotor. Stator temperature monitoring is realized with thermal sensors, but these are usually firmly embedded in the stator so that replacement is not an option, although sensor functionality deteriorates steadily. Since competitive pressure demands perpetual reduction of production costs, there is a commercial interest driving the investigation of sufficiently accurate real-time temperature estimation. In the last decades, various research efforts led to approaches that approximate the heat transfer process e.g. with equivalent circuit diagrams [2] called lumped-parameter thermal networks (LPTNs). This kind of model must forfeit physical interpretability of its structure and parameter values by significantly curtailing degrees of

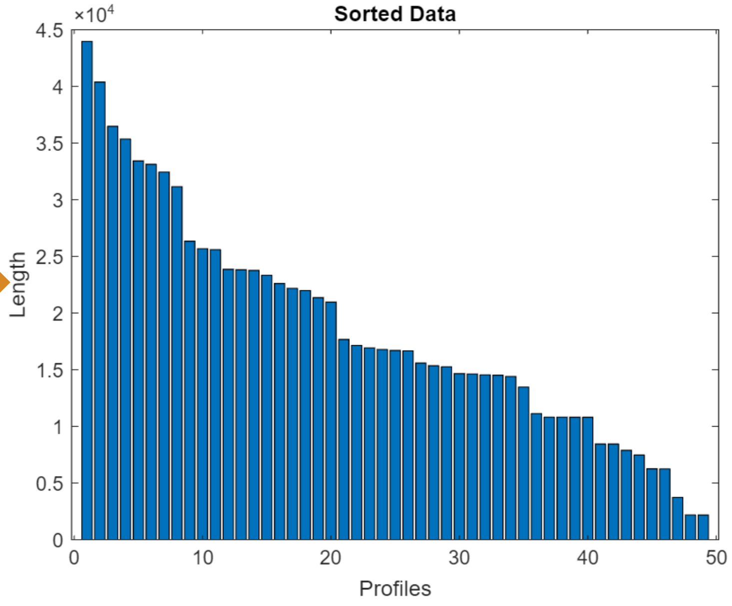
Workflow –Data Preparation

Raw CSV Data



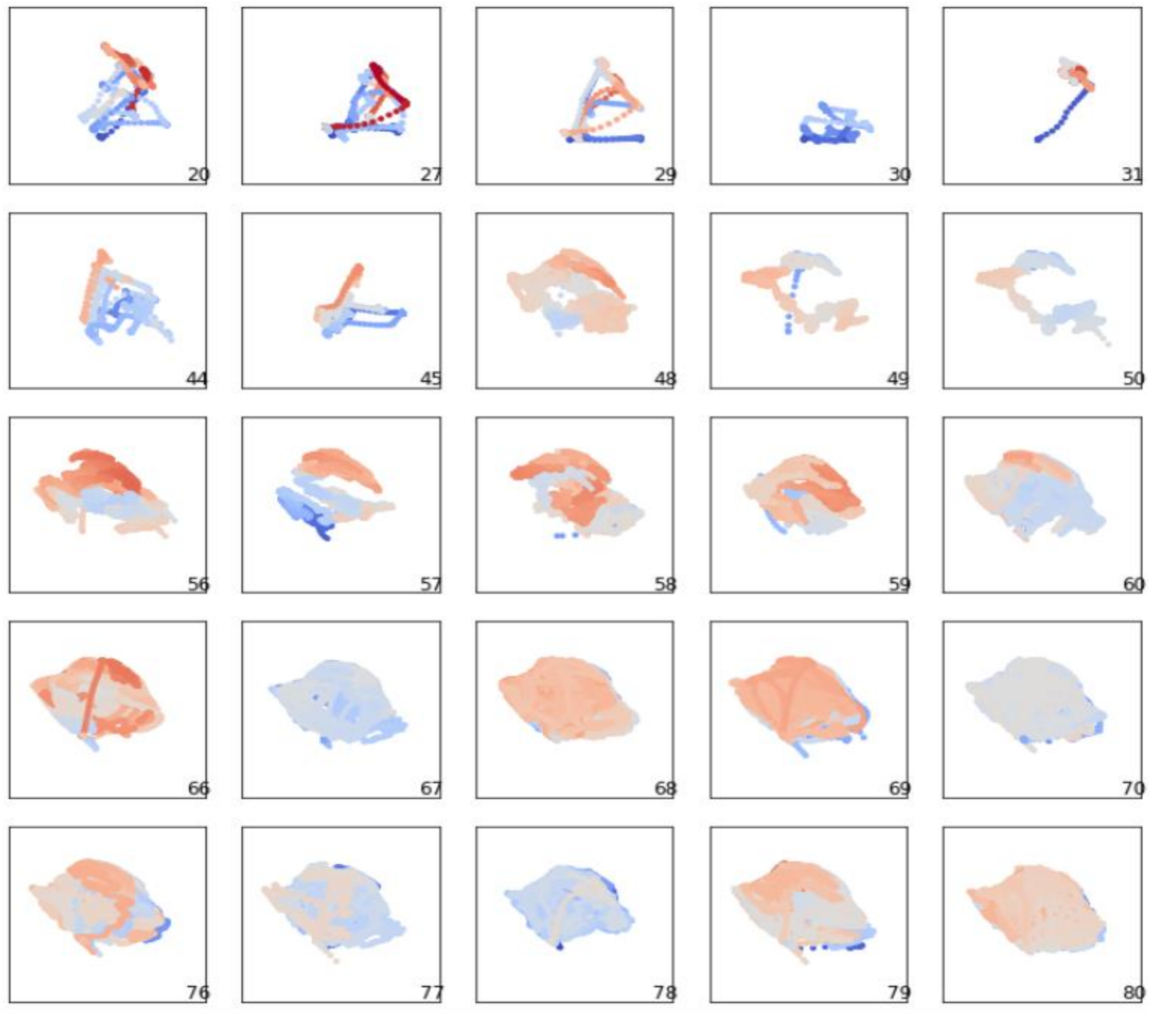
Additional Features





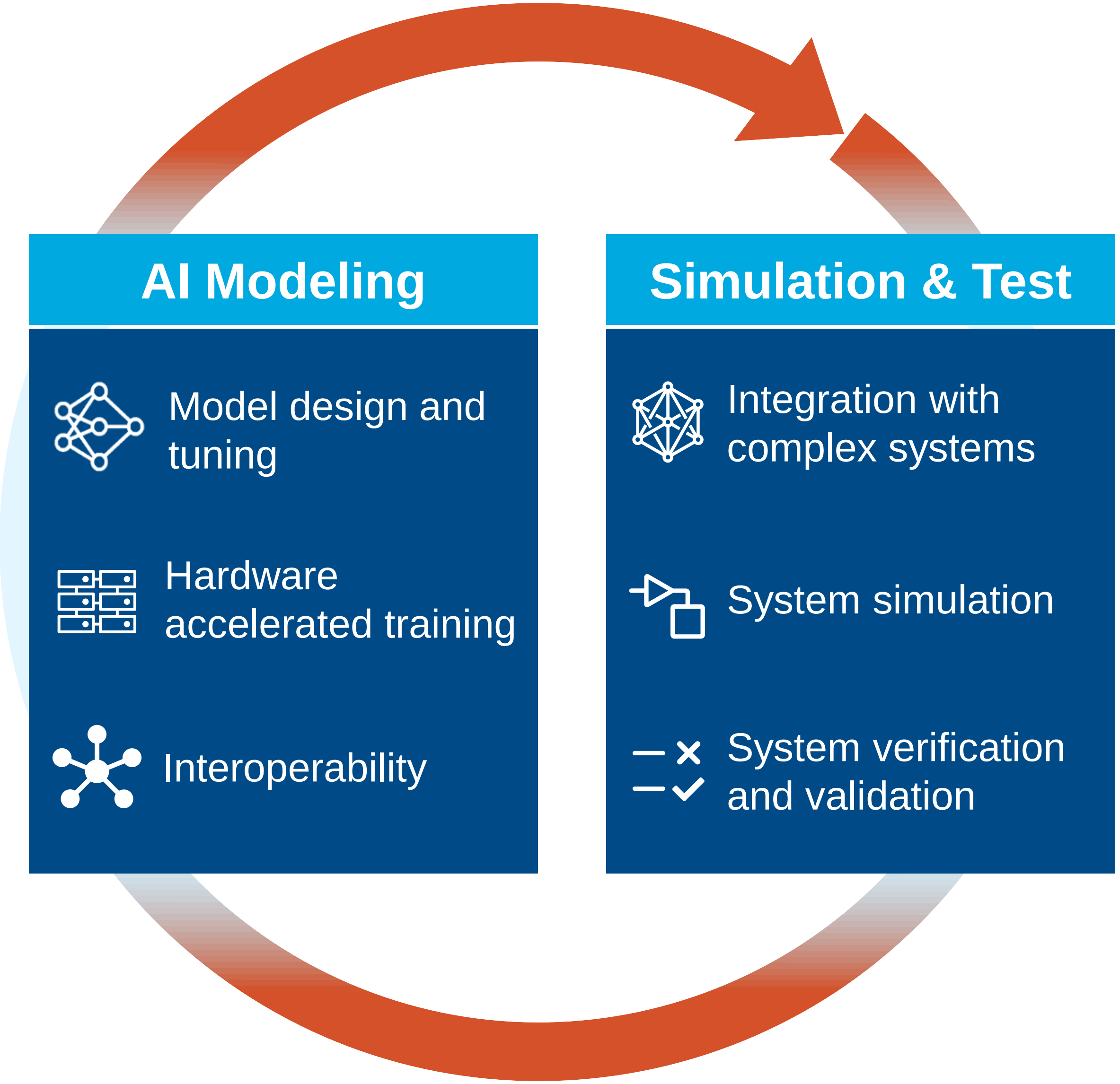
Sorted Data includes drive cycles of different lengths and Ambient Conditions,
DOE of design space to cover edge cases
Sorting helps to keep the mini-batch computation efficient with minimal padding

Profile Characteristics

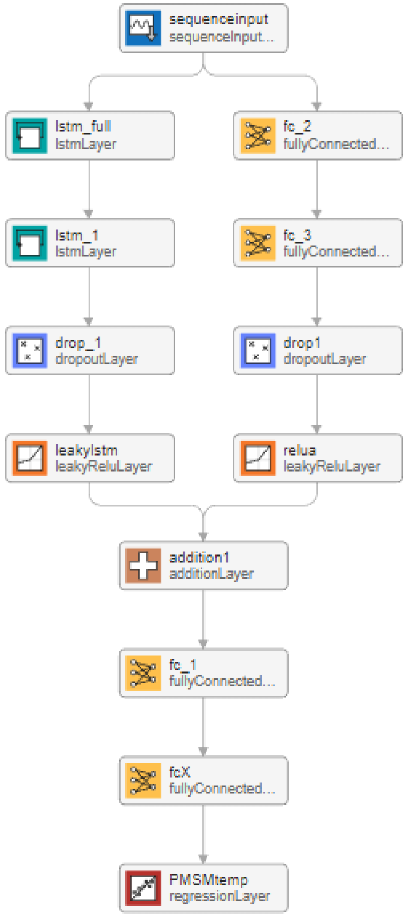


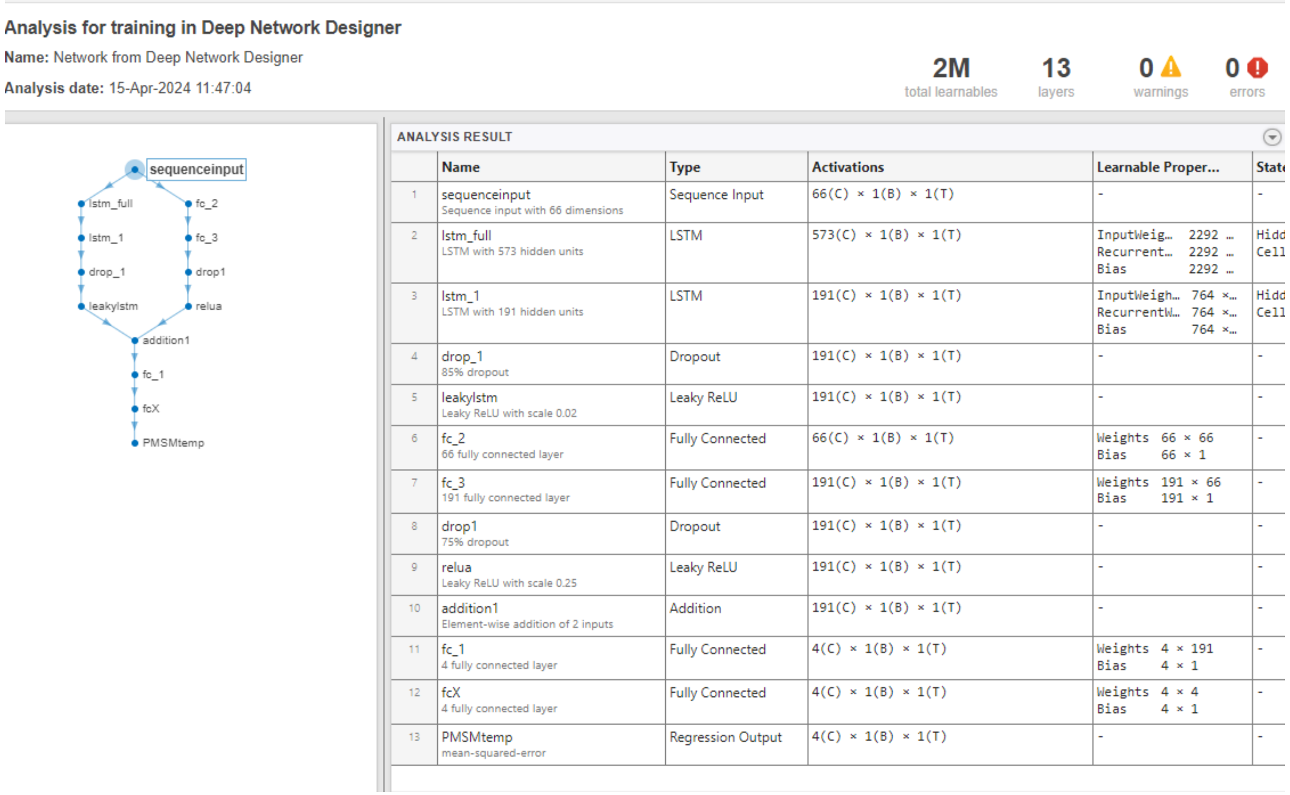
Drive cycles on **Torque-Speed** plane

AI-driven system design



AI-Model Development





Analysis for training in Deep Network Designer

Name: Network from Deep Network Designer

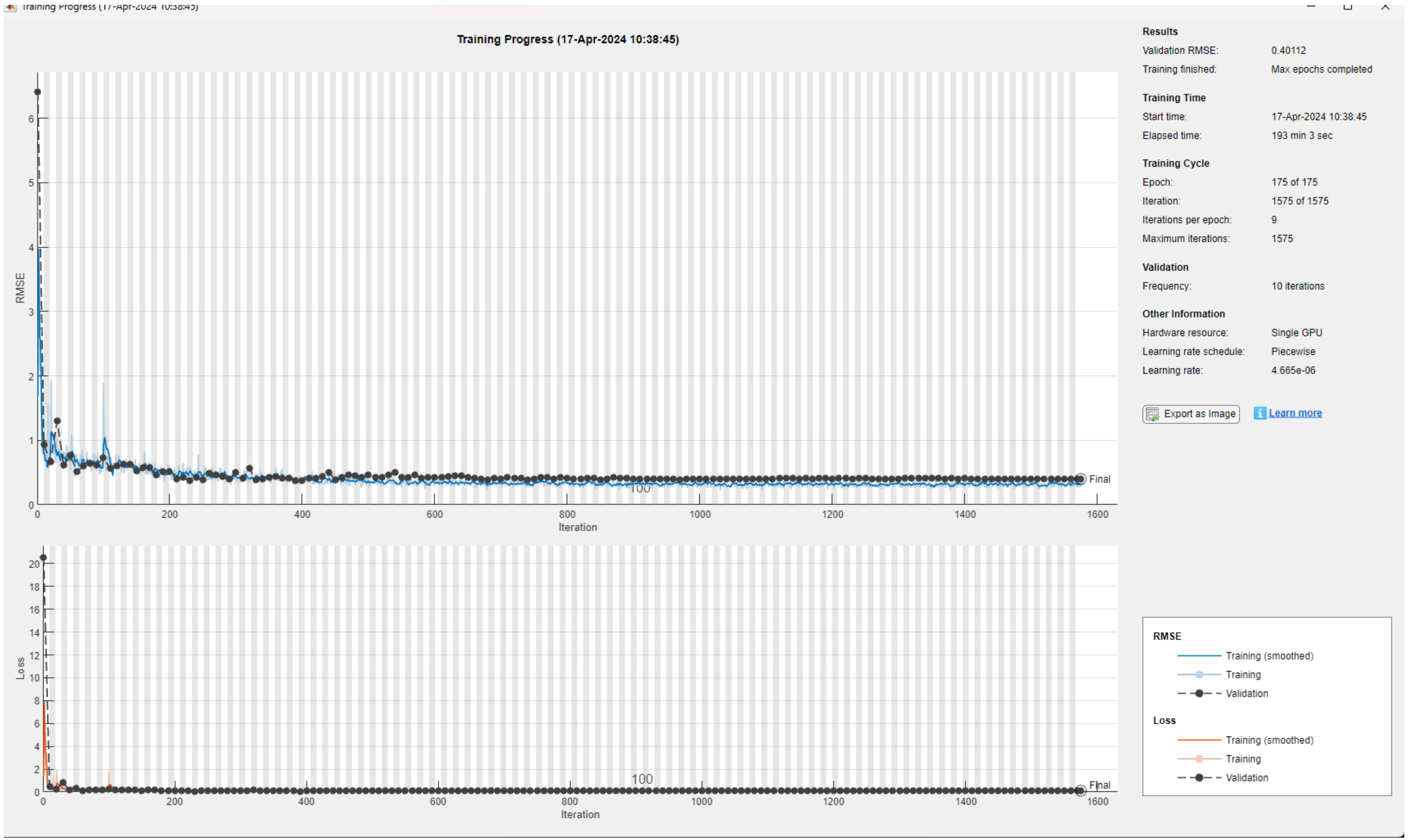
Analysis date: 15-Apr-2024 11:47:04

2M total learnables | 13 layers | 0 warnings | 0 errors

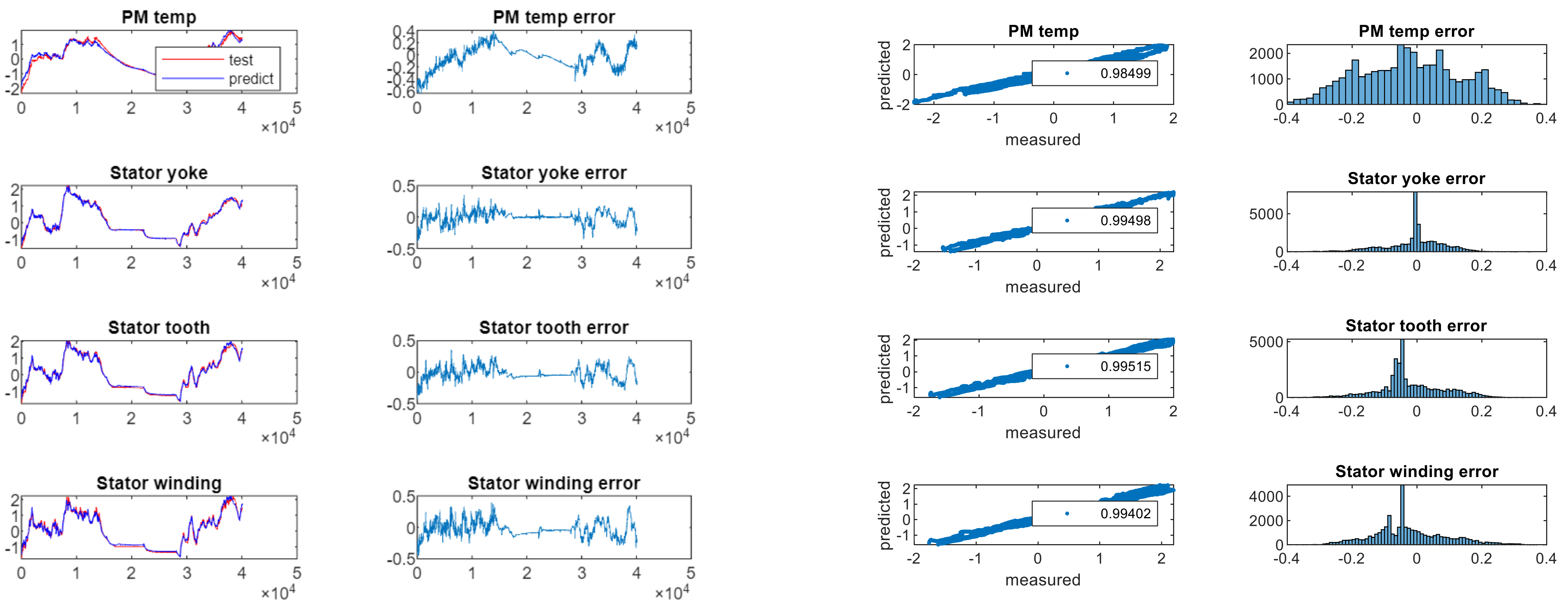
ANALYSIS RESULT

	Name	Type	Activations	Learnable Proper...	State
1	sequenceinput Sequence input with 66 dimensions	Sequence Input	66(C) × 1(B) × 1(T)	-	-
2	lstm_full LSTM with 573 hidden units	LSTM	573(C) × 1(B) × 1(T)	InputWeig... 2292 ... Recurrent... 2292 ... Bias 2292 ...	Hidd Cell
3	lstm_1 LSTM with 191 hidden units	LSTM	191(C) × 1(B) × 1(T)	InputWeigh... 764 ×... RecurrentW... 764 ×... Bias 764 ×...	Hidd Cell
4	drop_1 85% dropout	Dropout	191(C) × 1(B) × 1(T)	-	-
5	leakylstm Leaky ReLU with scale 0.02	Leaky ReLU	191(C) × 1(B) × 1(T)	-	-
6	fc_2 66 fully connected layer	Fully Connected	66(C) × 1(B) × 1(T)	Weights 66 × 66 Bias 66 × 1	-
7	fc_3 191 fully connected layer	Fully Connected	191(C) × 1(B) × 1(T)	Weights 191 × 66 Bias 191 × 1	-
8	drop1 75% dropout	Dropout	191(C) × 1(B) × 1(T)	-	-
9	relua Leaky ReLU with scale 0.25	Leaky ReLU	191(C) × 1(B) × 1(T)	-	-
10	addition1 Element-wise addition of 2 inputs	Addition	191(C) × 1(B) × 1(T)	-	-
11	fc_1 4 fully connected layer	Fully Connected	4(C) × 1(B) × 1(T)	Weights 4 × 191 Bias 4 × 1	-
12	fcX 4 fully connected layer	Fully Connected	4(C) × 1(B) × 1(T)	Weights 4 × 4 Bias 4 × 1	-
13	PMSMtemp mean-squared-error	Regression Output	4(C) × 1(B) × 1(T)	-	-

What Do we look in Training?

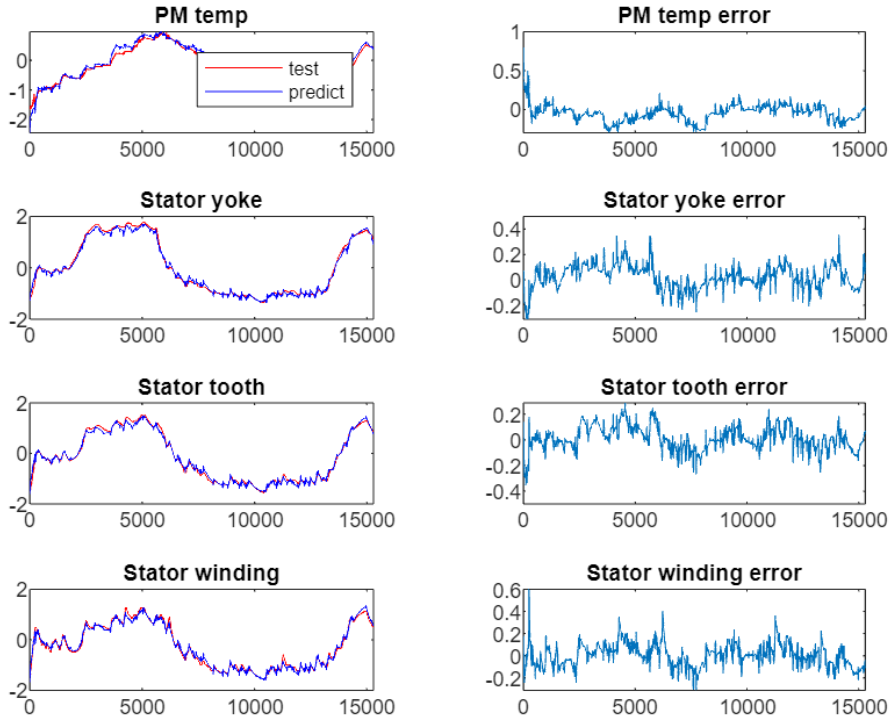


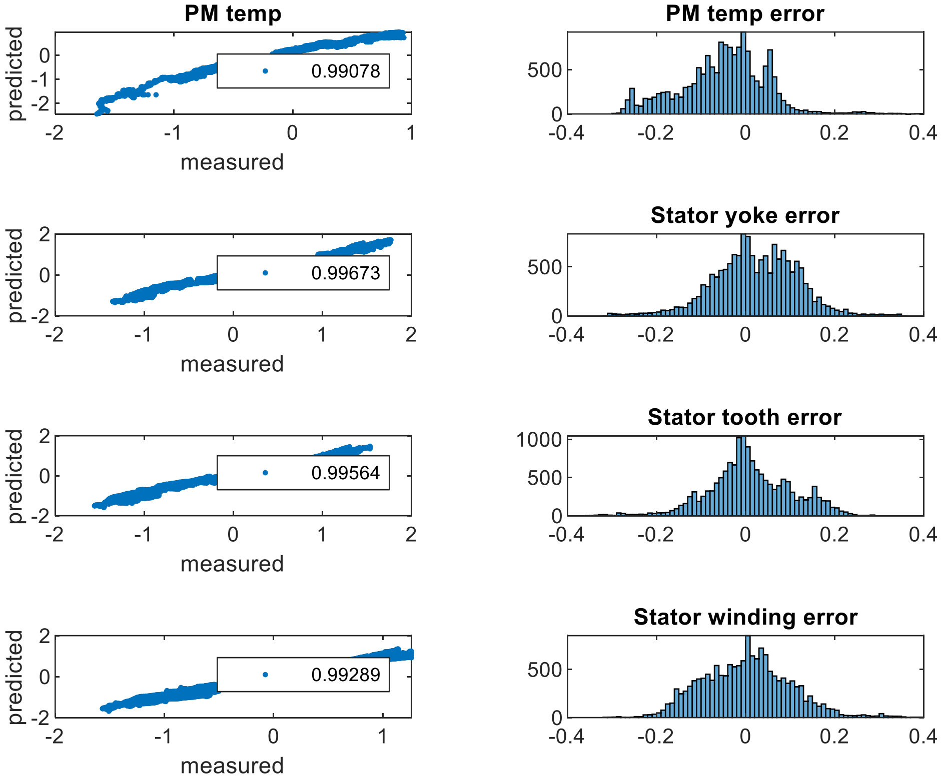
Testing Results on a long profile



All correlation values are about 0.99 and error distribution is unbiased hence model captures trend and Magnitude

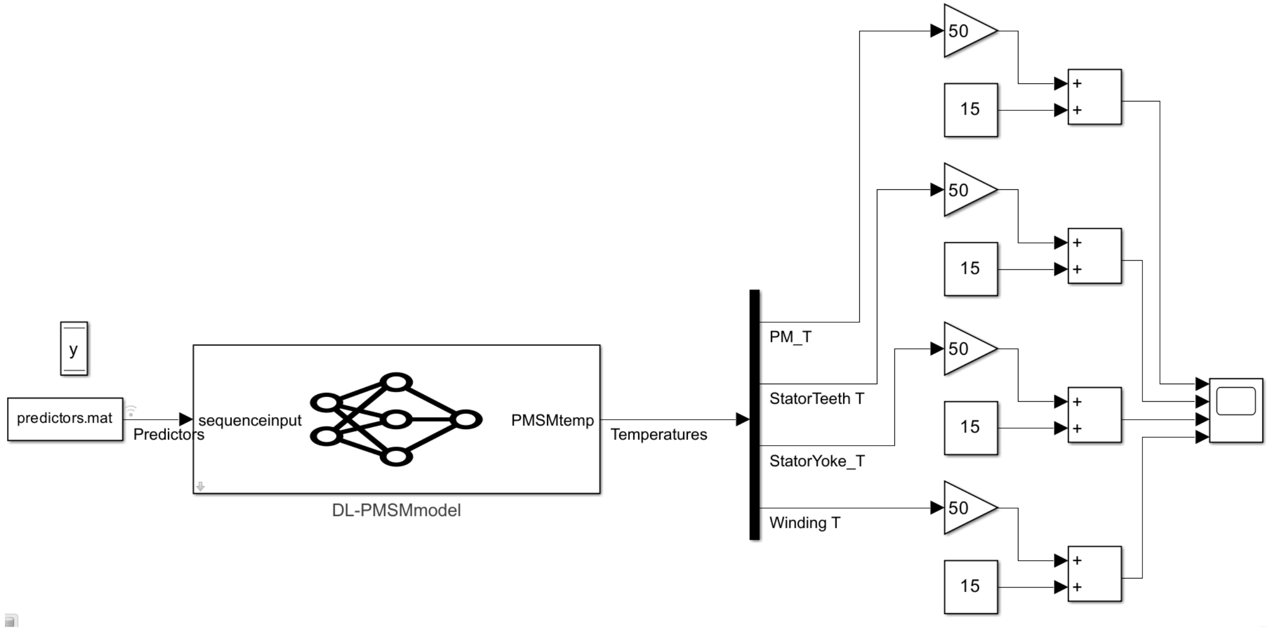
Test Results for a short profile





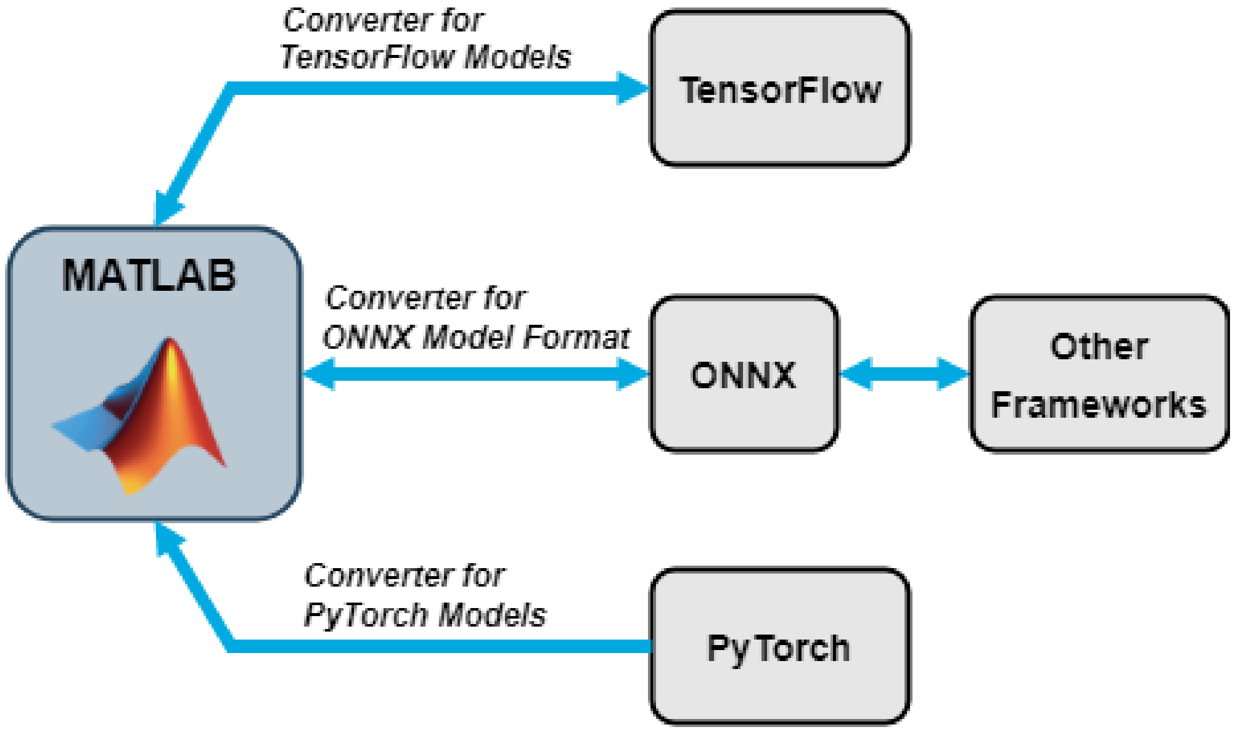
All correlation values are about 0.99 and error distribution is unbiased hence model captures trend and Magnitude

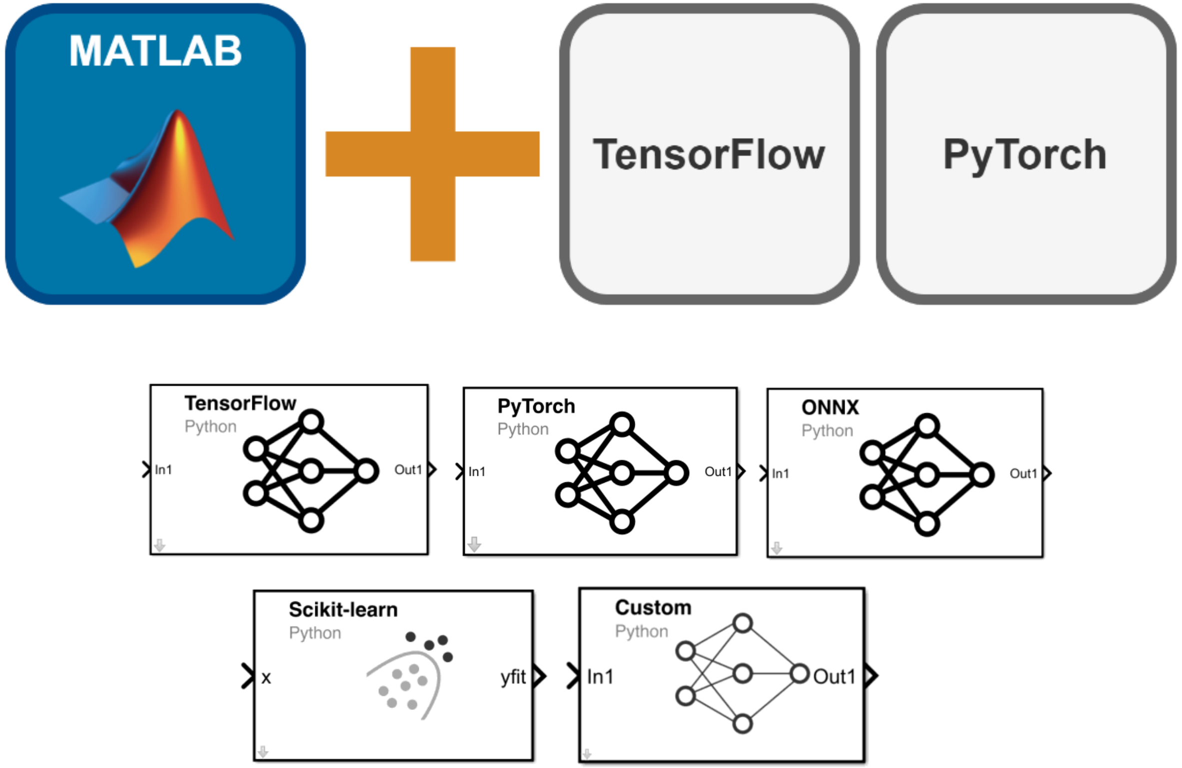
Deployment to Simulink



MATLAB works with Python-based frameworks

Framework Interoperability bridges the gap between data science, engineering and production





Data Preparation | **AI Modeling** | **Simulation & Test** | **Deployment**

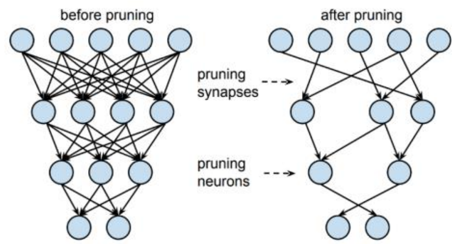
AI-driven system design

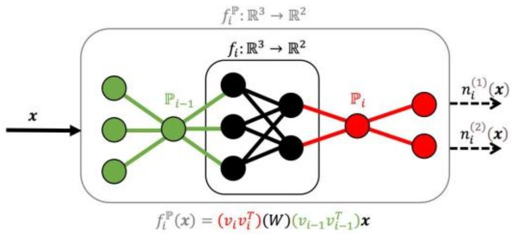
Deployment

- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

Model compression reduces Model for Deployment (reduces learnable from 2M to 850 K)

Structural Compression



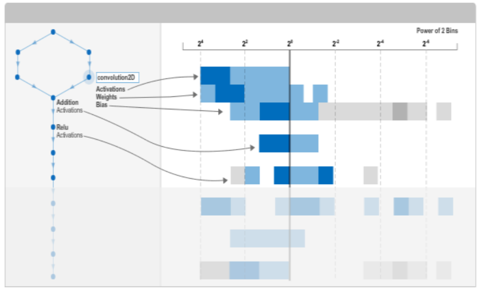


Pruning convolutional neural networks

Projection of deep neural networks

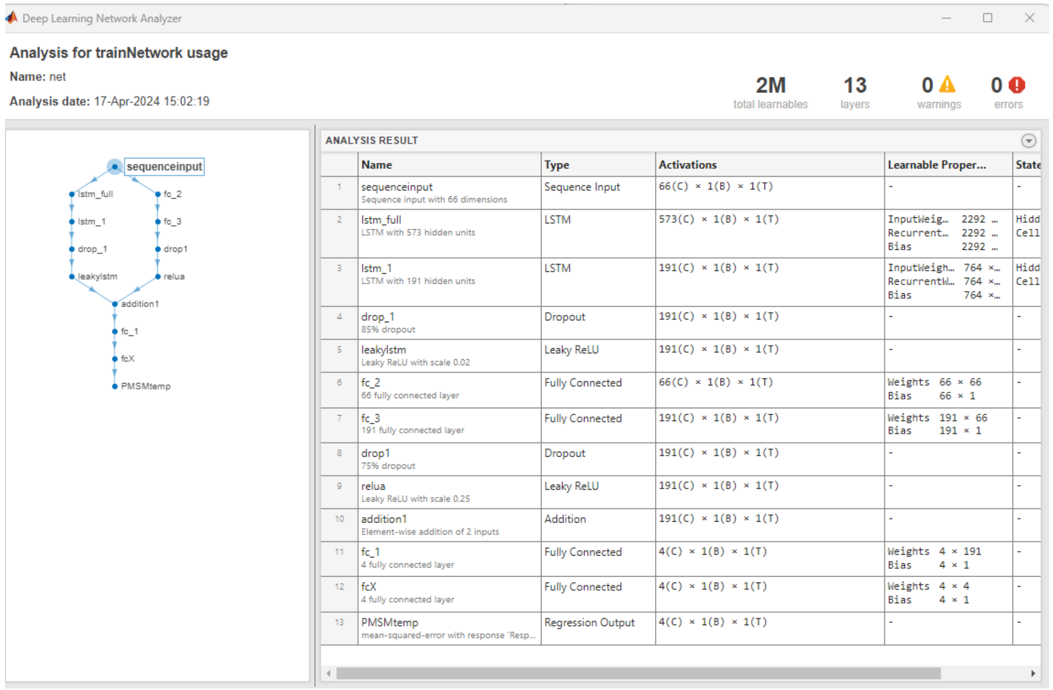
We'll take a closer look at the projection technique in today's workshop

Datatype Compression

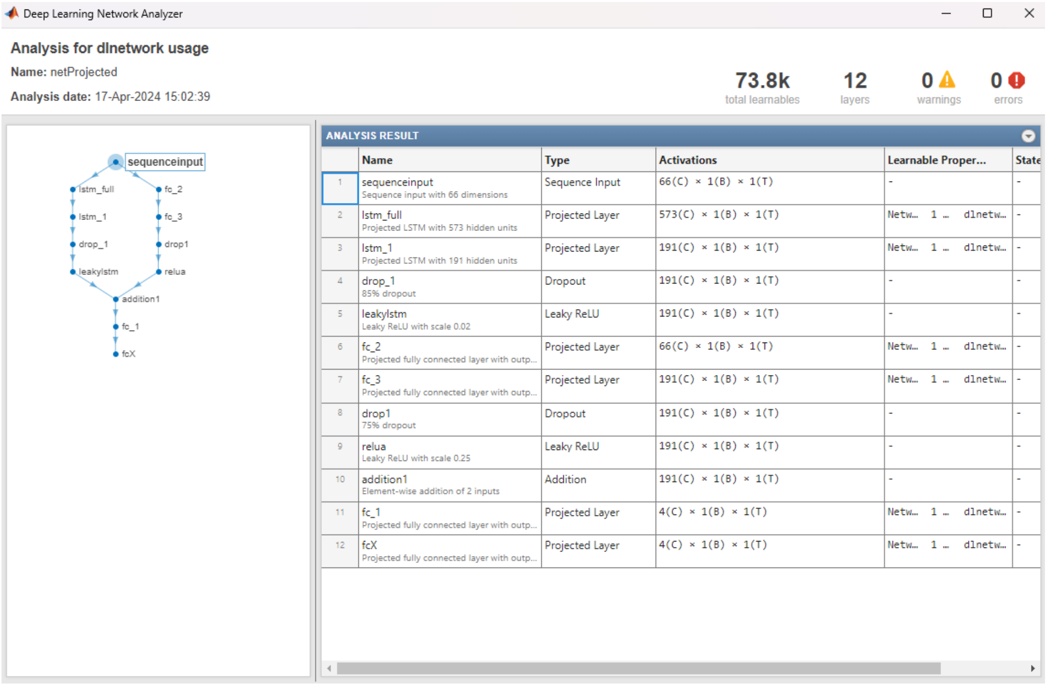


Quantization of network weights to lower precision datatypes (bfloat16, int8)

Compressed Network



Original



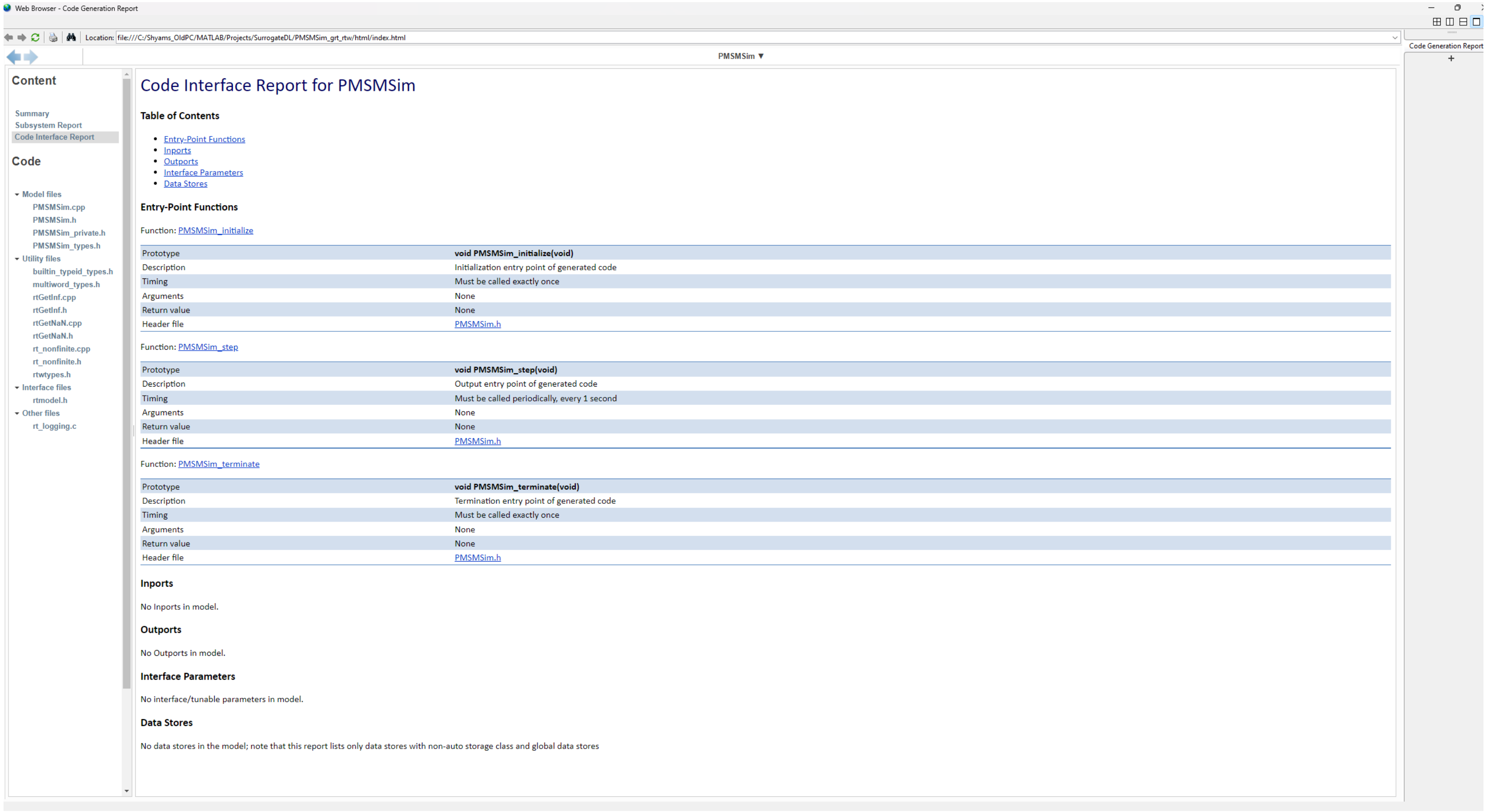
Projected

Approximately 25X reduction in Size

Data Preparation | AI Modeling | Simulation & Test | **Deployment**

20

Code Generation For HIL/SIL test


Web Browser - Code Generation Report

Location: file:///C:/Shyams_OldPC/MATLAB/Projects/SurrogateDL/PMSMSim_grt_rtw/html/index.html

PMSMSim ▼

Code Generation Report

Content

Summary
Subsystem Report
Code Interface Report

Code

Model files
PMSMSim.cpp
PMSMSim.h
PMSMSim_private.h
PMSMSim_types.h
Utility files
builtin_typeid_types.h
multiword_types.h
rtGetInf.cpp
rtGetInf.h
rtGetNaN.cpp
rtGetNaN.h
rt_nonfinite.cpp
rt_nonfinite.h
rtwtypes.h
Interface files
rtmodel.h
Other files
rt_logging.c

Code Interface Report for PMSMSim

Table of Contents

- Entry-Point Functions
- Inports
- Outports
- Interface Parameters
- Data Stores

Entry-Point Functions

Function: PMSMSim_initialize

Prototype	**void PMSMSim_initialize(void)**
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	PMSMSim.h

Function: PMSMSim_step

Prototype	**void PMSMSim_step(void)**
Description	Output entry point of generated code
Timing	Must be called periodically, every 1 second
Arguments	None
Return value	None
Header file	PMSMSim.h

Function: PMSMSim_terminate

Prototype	**void PMSMSim_terminate(void)**
Description	Termination entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	PMSMSim.h

Inports

No Inports in model.

Outports

No Outports in model.

Interface Parameters

No interface/tunable parameters in model.

Data Stores

No data stores in the model; note that this report lists only data stores with non-auto storage class and global data stores

Examine Generated Code

yams_OldPC\..._rtw\ert_main.cpp   C:\Shyams_OldPC\..._rtw\PMSMSim.cpp

```
// your application needs.  This example simply sets an error status in the
// real-time model and returns from rt_OneStep.
//
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag{ false };

  // Disable interrupts here

  // Check for overrun
  if (OverrunFlag) {
    rtmSetErrorStatus(PMSMSim_Obj.getRTM(), "Overrun");
    return;
  }

  OverrunFlag = true;

  // Save FPU context here (if necessary)
  // Re-enable timer or interrupt here
  // Set model inputs here

  // Step the model
  PMSMSim_Obj.step();

  // Get model outputs here

  // Indicate task complete
  OverrunFlag = false;

  // Disable interrupts here
  // Restore FPU context here (if necessary)
  // Enable interrupts here
}

//
// The example main function illustrates what is required by your
// application code to initialize, execute, and terminate the generated code.
// Attaching rt_OneStep to a real-time clock is target specific. This example
// illustrates how you do this relative to initializing the model.
//
int_T main(int_T argc, const char *argv[])
{
  // Unused arguments
  (void)(argc);
  (void)(argv);

  // Initialize model
  PMSMSim_Obj.initialize();

  // Attach rt_OneStep to a timer or interrupt service routine with
  //  period 1.0 seconds (base rate of the model) here.
  //  The call syntax for rt_OneStep is
  //
  //    rt_OneStep();

  printf("Warning: The simulation will run forever. "
         "Generated ERT main won't simulate model step behavior. "
         "To change this behavior select the 'MAT-file logging' option.\n");
  fflush((nullptr));
  while (rtmGetErrorStatus(PMSMSim_Obj.getRTM()) == (nullptr)) {
    //  Perform application tasks here
  }

  // Terminate model
  PMSMSim_Obj.terminate();
```

Key takeaways

Enable	Hardware-in-the-Loop (HIL) testing and system-level simulation for high-fidelity models.
Explore	Various ROM techniques in MATLAB to find the best method.

Conclusions

- MathWorks Tools Makes Data to Digital Twins(ROM) workflow easy
- An e-Motor ROM predicts e-Motor's all internal temperatures with similar trends and magnitudes as real data.
- ROM incorporation into Simulink with built-in infrastructure allows SIL/HIL testing faster and easier

MathWorks
AUTOMOTIVE CONFERENCE 2024
North America

Thank you