

Reduced Order Models for Semiconductor Production Equipment



Mahaveer Satra

*Industry Application Engineer,
MathWorks
msatra@mathworks.com*

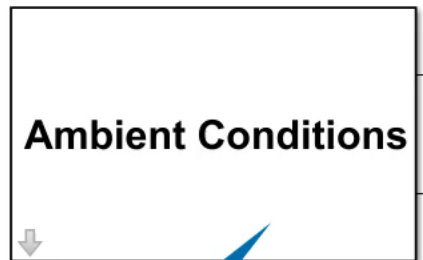


SIMULATION DEBUG MODELING FORMAT APPS **VARIANT SUBSYSTEM**

FILE LIBRARY PREPARE SIMULATE REVIEW RESULTS

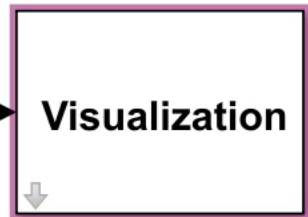
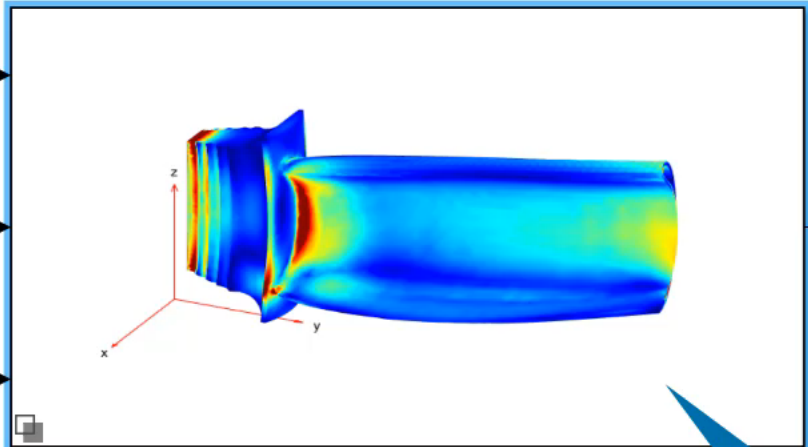
New Open Save Print Library Browser Log Signals Add Viewer Signal Table Stop Time: 7000 Normal Step Back Run Step Forward Stop Data Inspector Logic Analyzer Simulation Manager

SystemLevelSim_JetEngineBlade Reduced order model



Temperature and pressure conditions

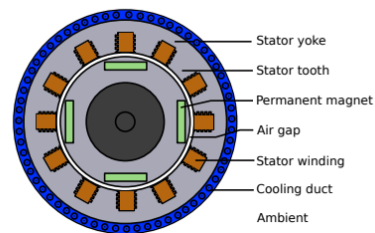
Cooling temperature controller



FEA model to compute maximum tip displacement

How about motors?

- Permanent Magnet Synchronous motors are backbone for electrification
- 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



Key takeaways

Enable

Reuse of full-order high-fidelity models for system-level simulations, Hardware-in-the-Loop (HIL) testing, nonlinear control design, and virtual sensor modeling.

Explore

Various ROM techniques in MATLAB to find the best method.

Common challenges

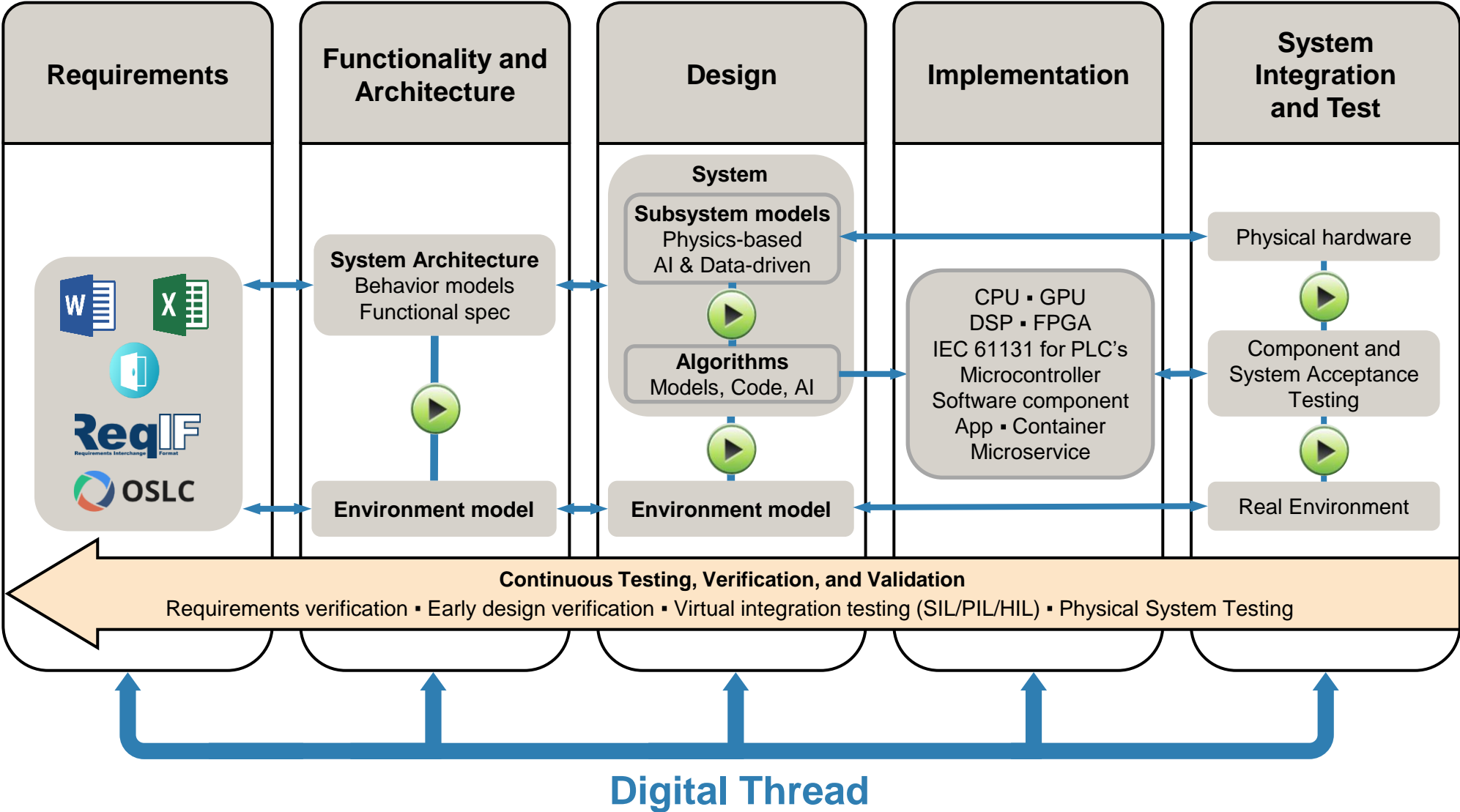


High fidelity models, such as ones from 3rd party FEA/CFD tools, are too slow for system level simulation, control design, and HIL testing.

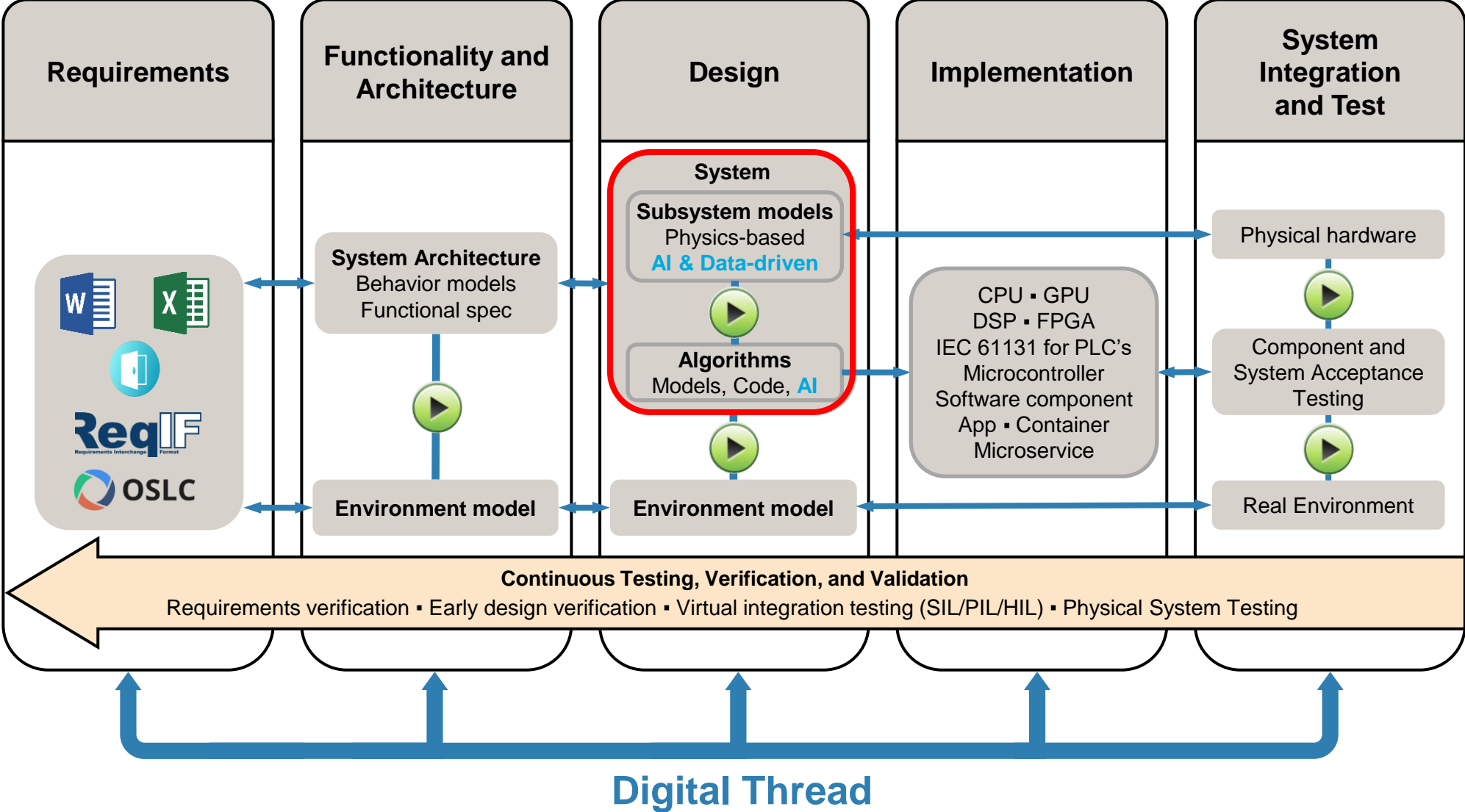


Creating a ROM that produces desired results in terms of speed, accuracy, interpretability, etc.

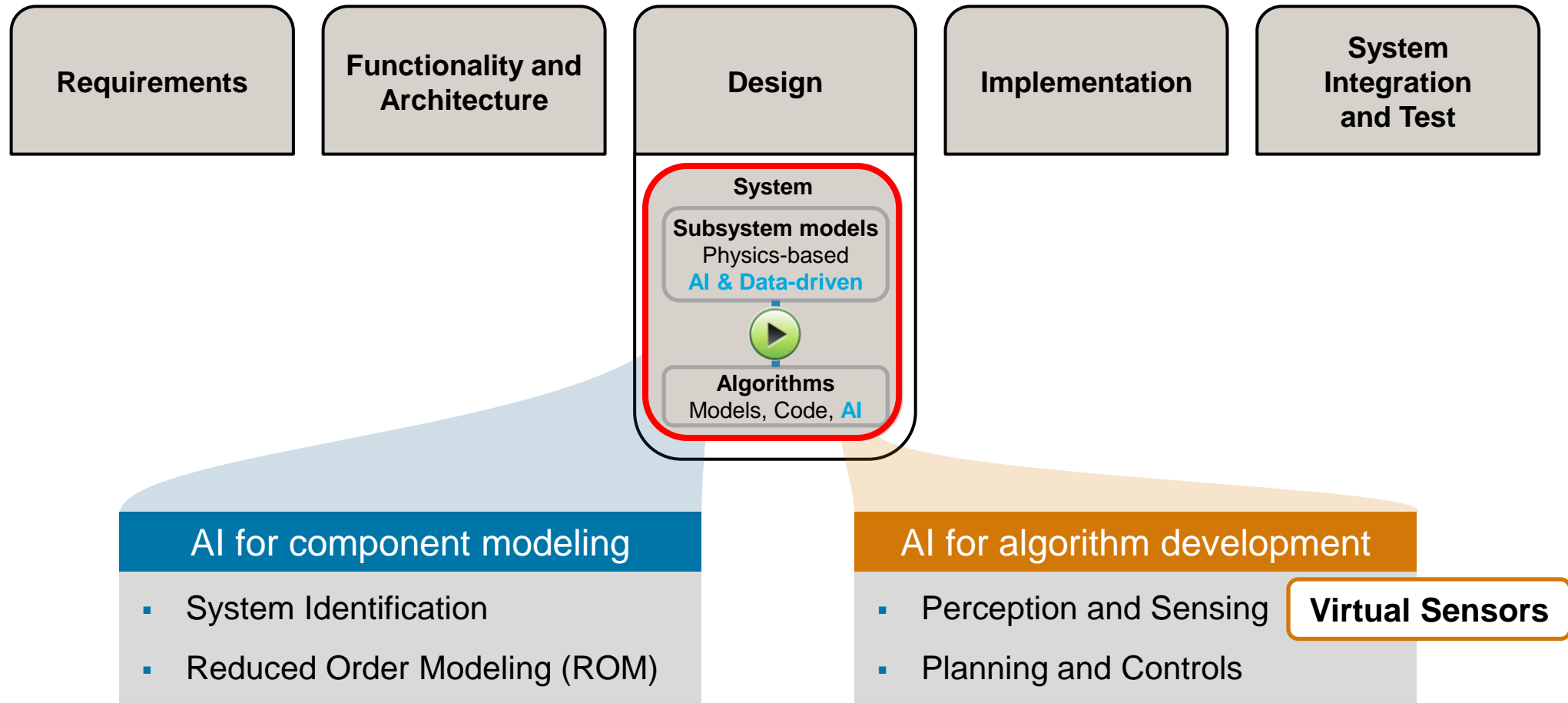
Model-Based Design



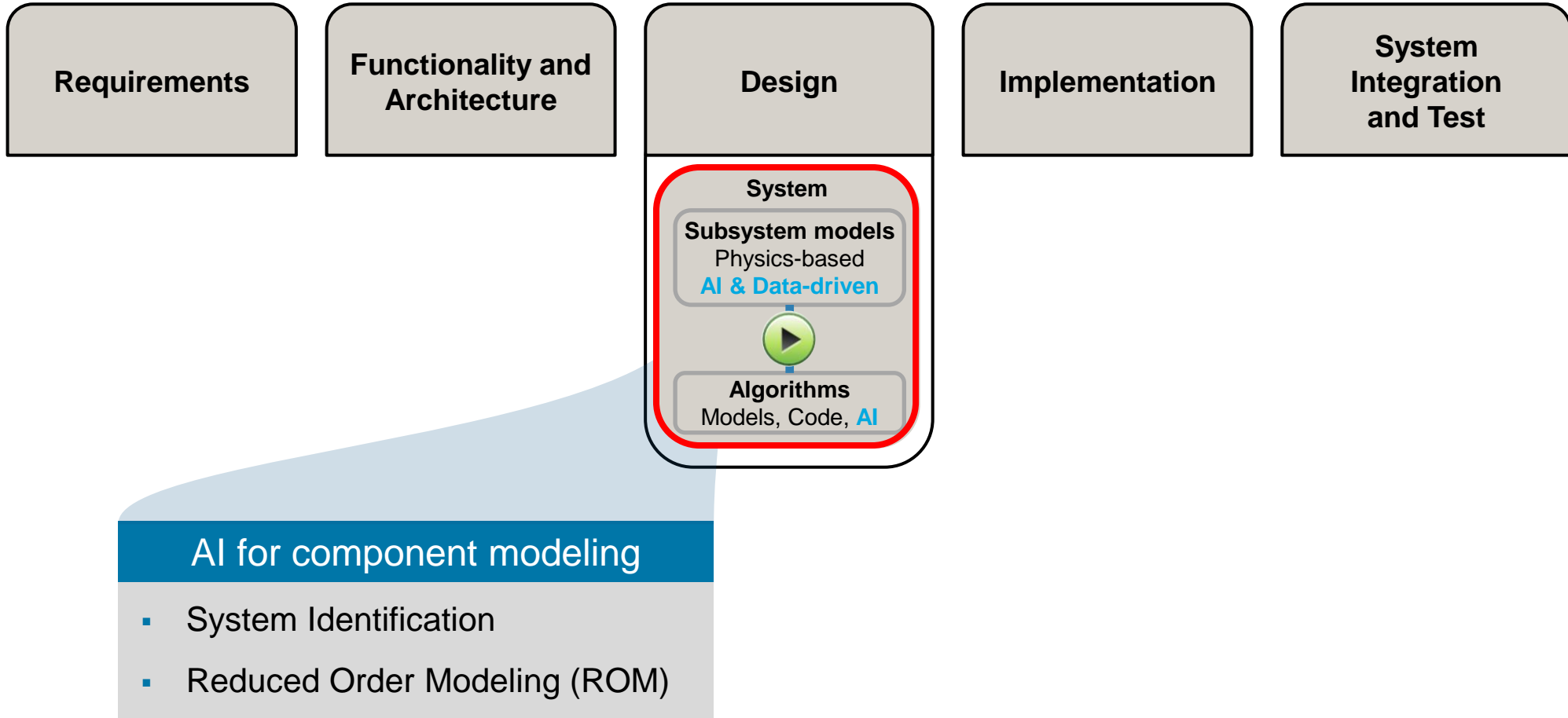
Integrating AI into Model-Based Design



AI for component modeling and algorithm development



Focus today



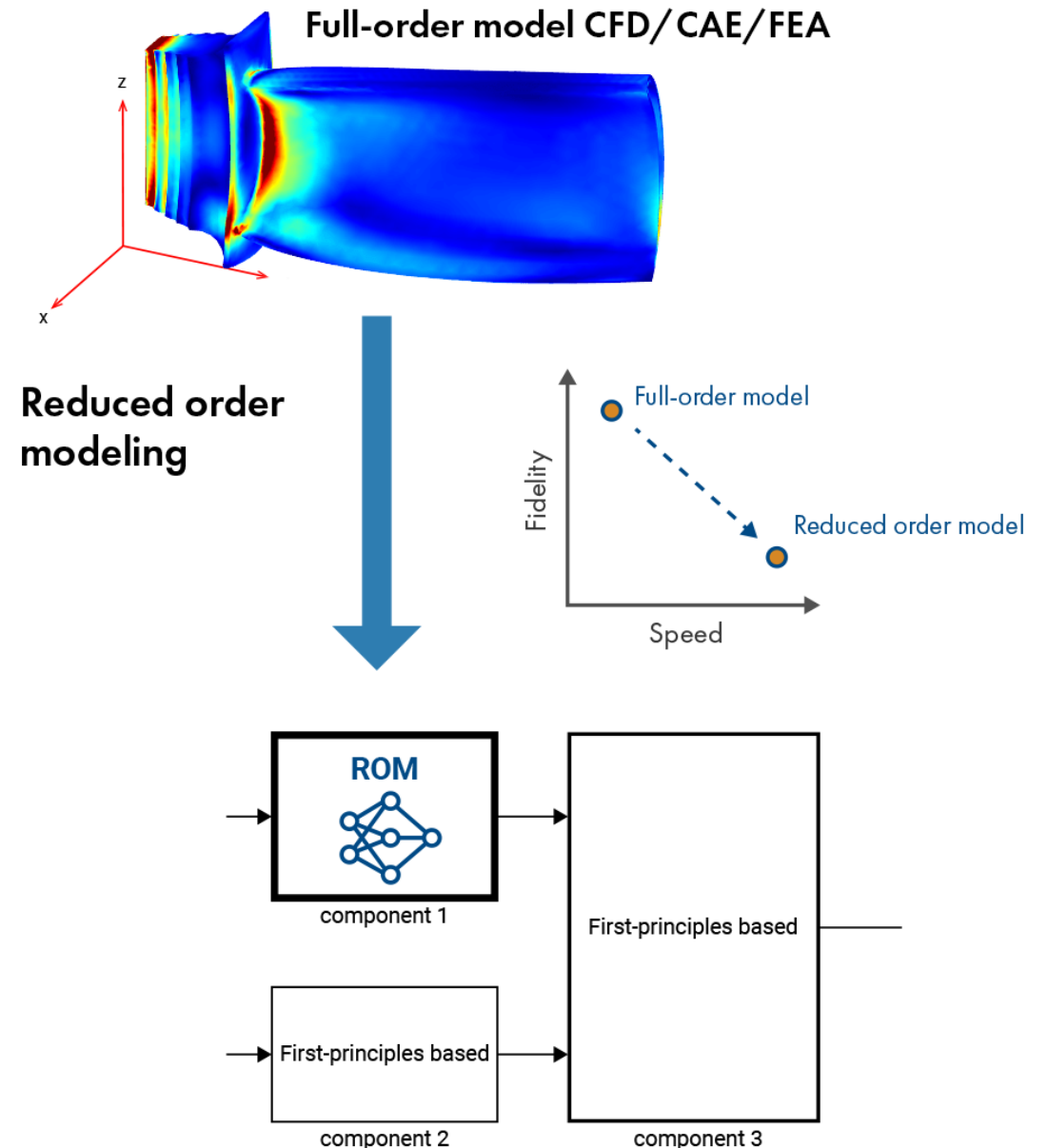
Reduced order modeling

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
- Perform control design
- Develop virtual sensors, Digital twins
- Perform design exploration

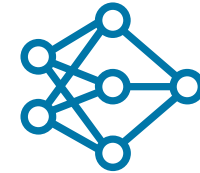


Reduced order modeling

How

AI-Based
Data-driven

Inputs
Charging Rate
SoC



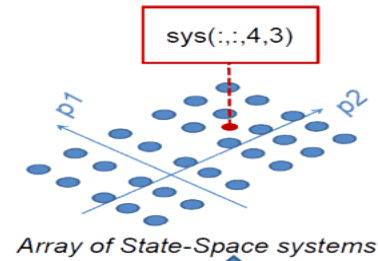
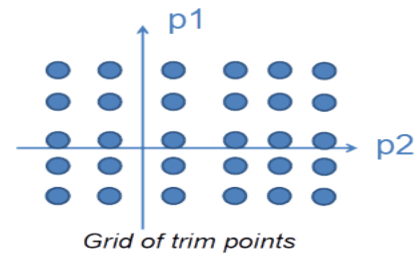
Outputs
Potential
Difference ($V_s - V_e(V)$)

AI model

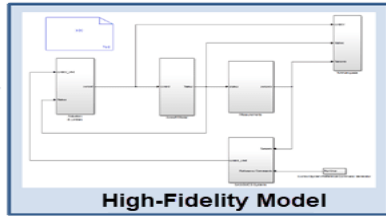
focus today

Reduced order
model

Linearization



Loop through the grid
of trim points



Identify local model at
each trim point

Model-based

FEA
Software

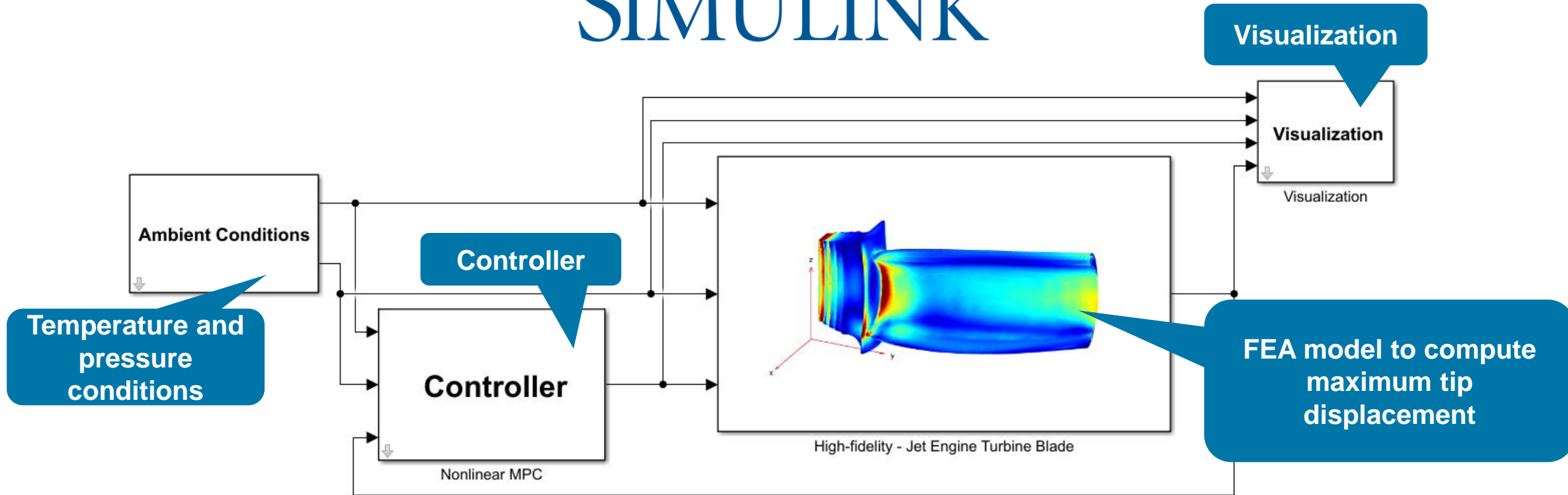


Simulink
Simscape Multibody
Control System Toolbox

Example overview

Replacing a high-fidelity jet engine turbine blade model with an AI-based reduced order model

SIMULINK®

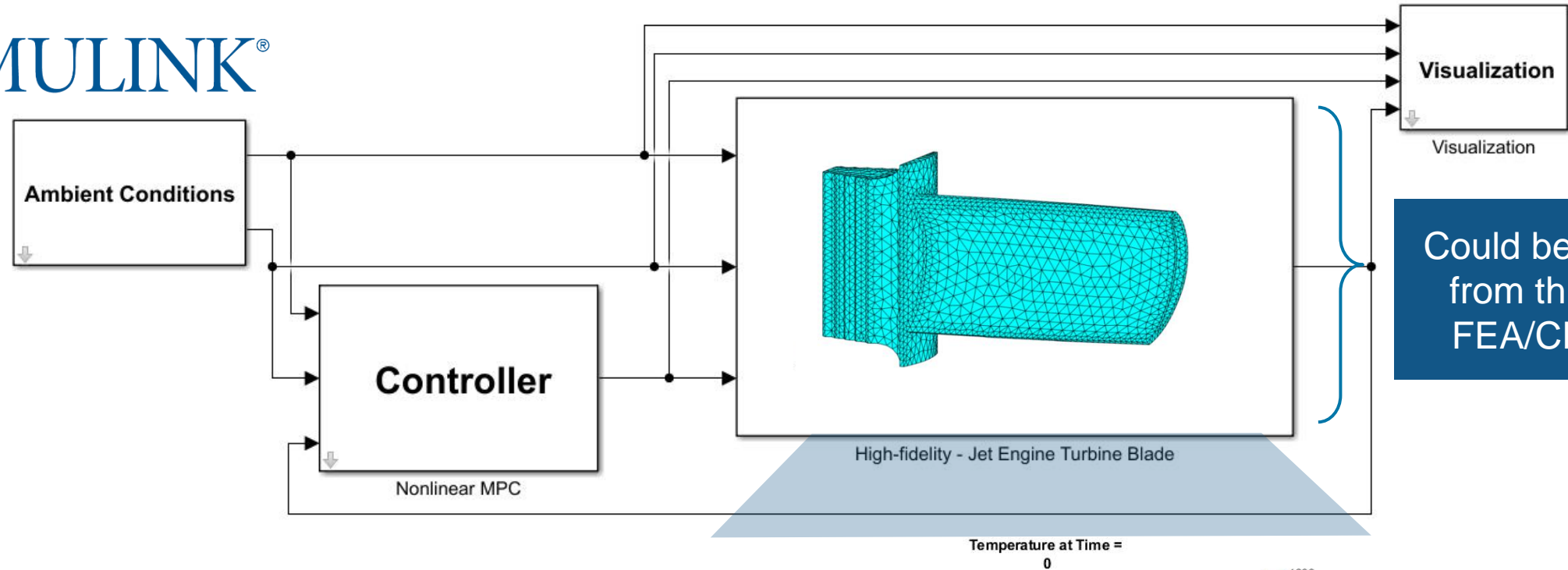


Closed-loop temperature control

Example overview

Replacing a high-fidelity jet engine turbine blade model with an AI-based reduced order model

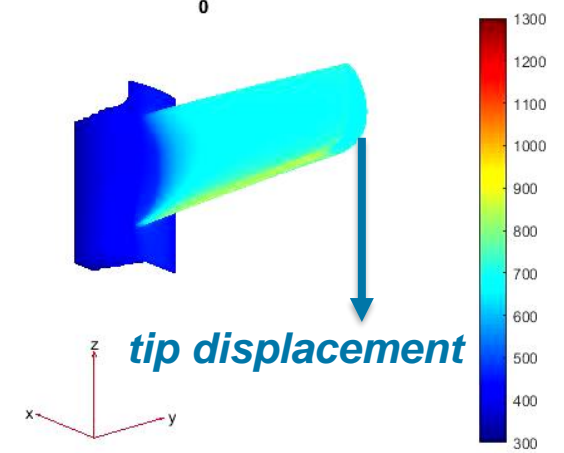
SIMULINK®



~30 seconds per time step for solving FEA models



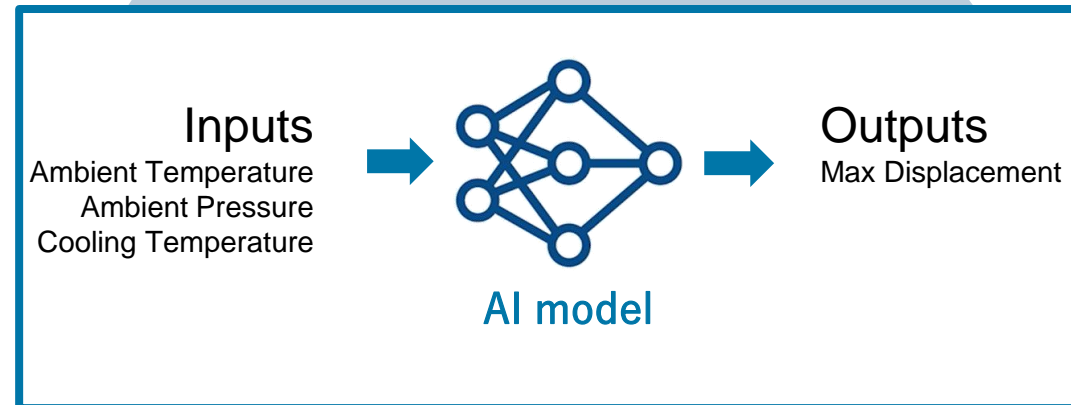
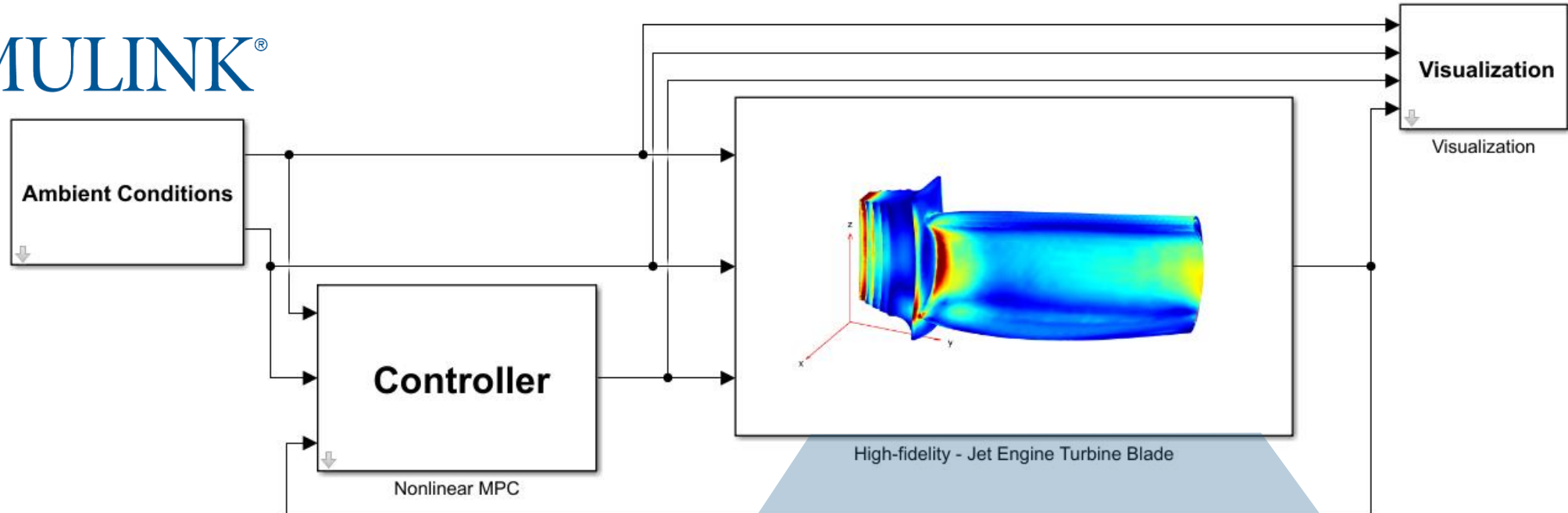
Not suitable for control design and HIL testing



Example overview

Replacing a high-fidelity jet engine turbine blade model with an AI-based reduced order model

SIMULINK®



Create AI-based ROMs using the reduced order modeling support package

Set up Design of Experiments (DoE)

Generate input-output data from full-order, high-fidelity subsystems

Train and compare AI-based reduced order models using preconfigured templates

Export trained reduced order models into Simulink or outside of Simulink through FMUs

Products and Services Search MathWorks.com

Reduced Order Modeling with MATLAB and Simulink

Create AI-based reduced order models

[Download add-on \(beta\)](#)

Simulink Add-On for Reduced Order Modeling is designed to be used with models modeled in Simulink, including full-order models for system-level desktop simulation, high-fidelity models for system-level simulation, and reduced order models for system-level simulation.

With Simulink Add-On for Reduced Order Modeling, you can:

- Set up the design of experiments and generate input-output data
- Train and compare AI-based reduced order models
- Export AI-based surrogate models to MATLAB and Simulink
- Export reduced order models as Functional Mock-up Units (FMUs) to Simulink Compiler)

Reduced Order Modeler App

The screenshot shows the Reduced Order Modeler App interface. The top toolbar includes icons for 'LSTM network', 'Nonlinear ARX', and 'Neural State Space', which are highlighted with a red box. The main workspace displays a Simulink model with various input and output blocks. A 'Configure Experiment' panel on the right shows a table of input signals and their ranges:

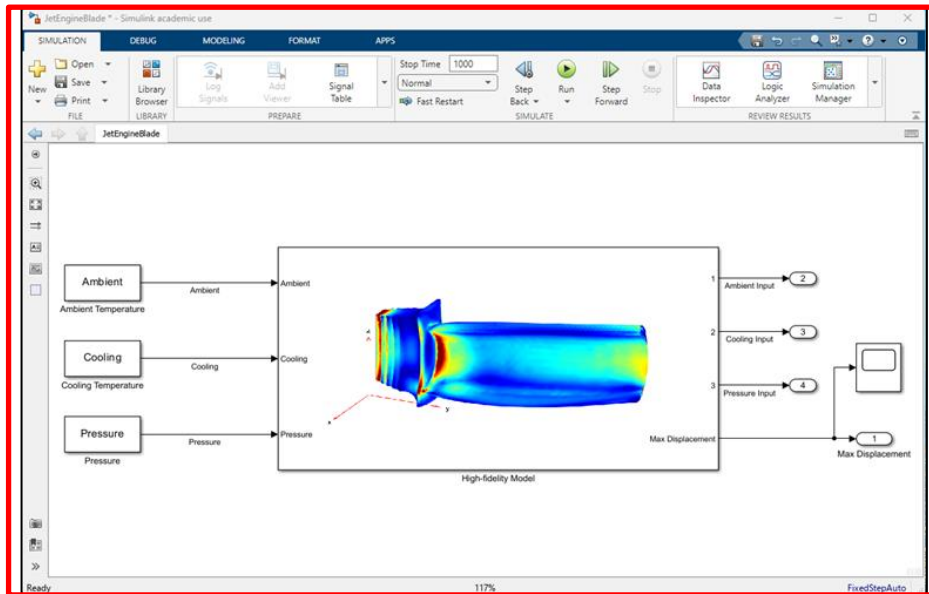
Signal	Min	Max
1 Ambient	800	2000
2 Cooling	50	250
3 Pressure	450000	550000

Below the app screenshot, a diagram illustrates the workflow of Reduced Order Modeling:

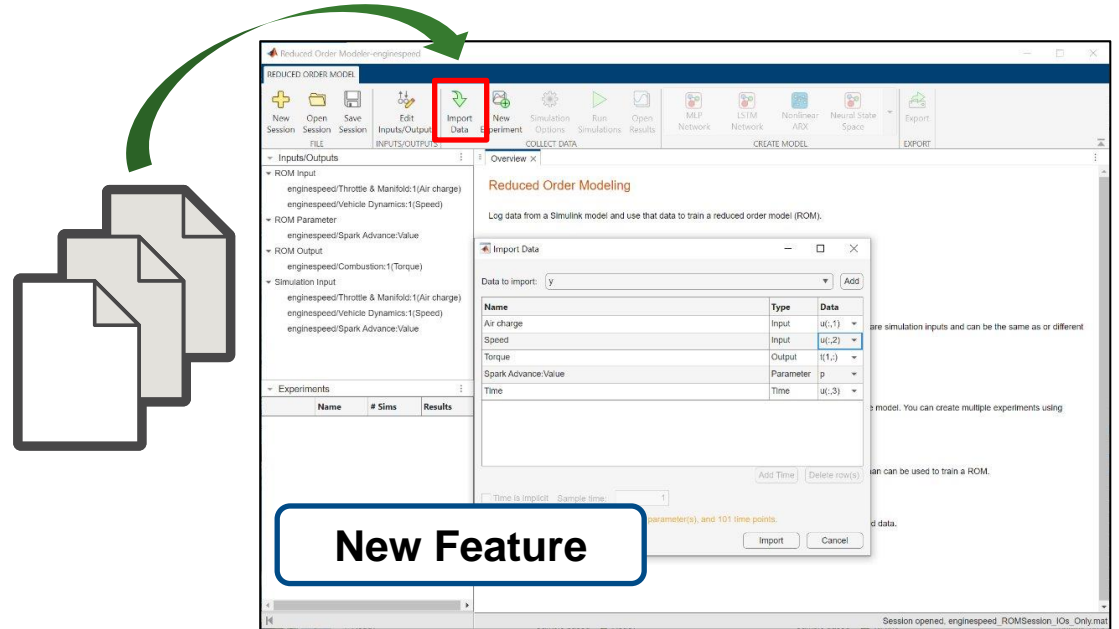
Full-order model CFD/CAE/FEA → **Reduced Order Modeling** → **ROM component 1** and **First-principles based component 2** → **First-principles based component 3**

A graph shows the relationship between Fidelity and Speed, with a dashed arrow indicating the transition from a high-fidelity full-order model to a lower-fidelity reduced order model.

Prepare data for training AI models



Generate synthetic data from Simulink/Simscape models

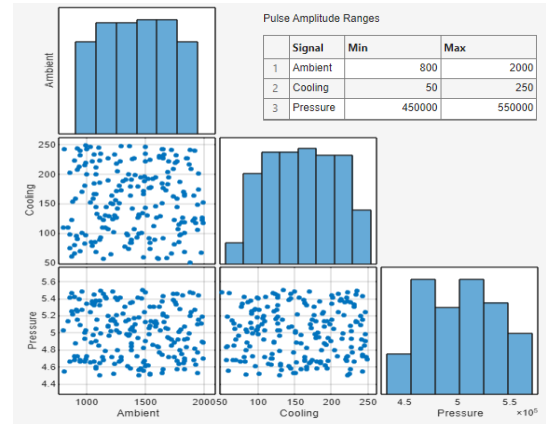


New Feature

Import pre-collected data from high-fidelity model into the app

Synthetic Data Generation

Design of Experiments

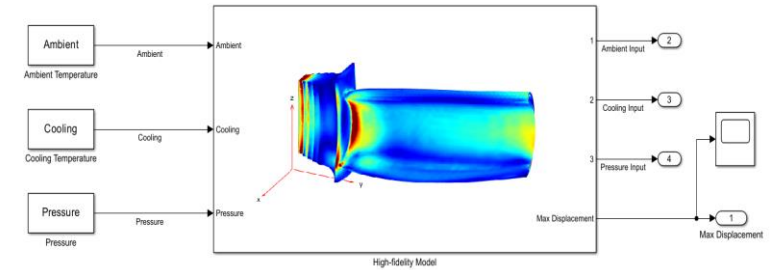
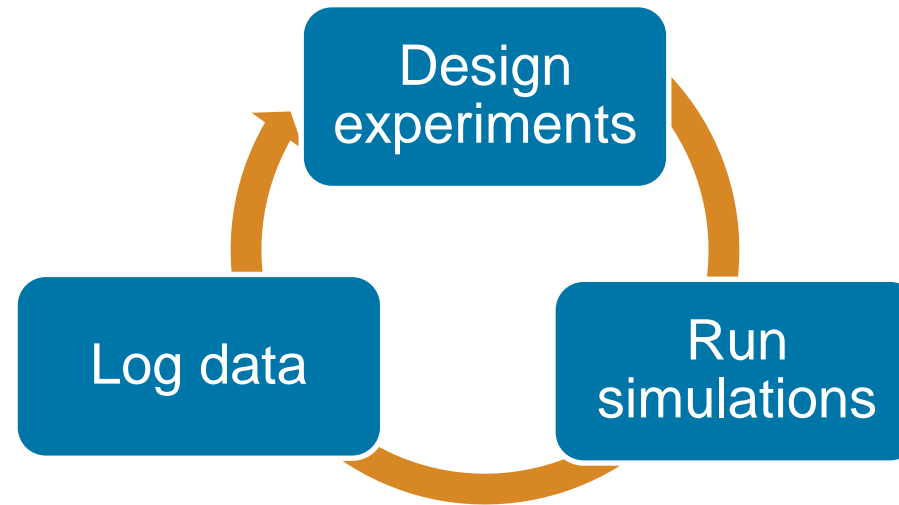


Input features

Ambient Temperature
Ambient Pressure
Cooling Temperature

Response

Max Displacement



Data Preparation

AI Modeling

Simulation & Test

Deployment

Synthetic Data Generation

Design of Experiments

Getting Started with Reduced Order Modeling Support Package

What Is Reduced Order Modeling?
 Reduced order modeling is a technique for simplifying full order high-fidelity models by reducing their computational complexity, while preserving their dominant behavior. Working with a reduced order model (ROM) can simplify analysis and control design.

Why Use Reduced Order Modeling?
 Using reduced order modeling techniques, you can:

- **Enable use of 3rd party FEA/FEM/CFD models for system-level simulation in Simulink® including hardware-in-the-loop testing** — You can combine multiple complex component-level models, including third-party finite element method (FEM) or finite element analysis (FEA) models, into system-level simulation models in Simulink by replacing the complex models with the corresponding ROMs. ROMs are also useful for hardware in-the-loop testing as they allow real-time simulations. Engineers can create ROMs representing the physical components of the system, which can run on a real-time machine for testing of the control algorithm on embedded hardware. The reduced computational complexity of ROMs make such testing more feasible.
- **Create virtual sensors** — You can use ROMs as virtual sensors for estimating or predicting signals of interest when measuring those signals by using a physical sensor is impractical or impossible.
- **Perform control design** — The reduced complexity of ROMs can make control design tasks more tractable. You can design your controller for the reduced order model of a plant and then validate the controller on the original high-fidelity system. You can also use ROMs for control algorithms that require internal prediction models, such as nonlinear model predictive control.
- **Create digital twins** — You can create or simplify digital twin models using ROMs. Doing so makes the digital twins more computationally efficient and more suitable for periodic updates to represent the current state of the operational asset.

Reduced Order Modeler App Workflow
 The general workflow of the Reduced Order Modeler app involves logging data from a Simulink model and using that data to train a ROM. It includes the following steps.

```

  graph LR
    A[Open Model and App] --> B[Select Inputs and Outputs]
    B --> C[Specify Experiments]
    C --> D[Run Model]
    D --> E[Create Reduced Order Model]
    E --> F[Export Model]
    F --> G[Replace Blocks in Simulink Model]
  
```

Command Window
 >>

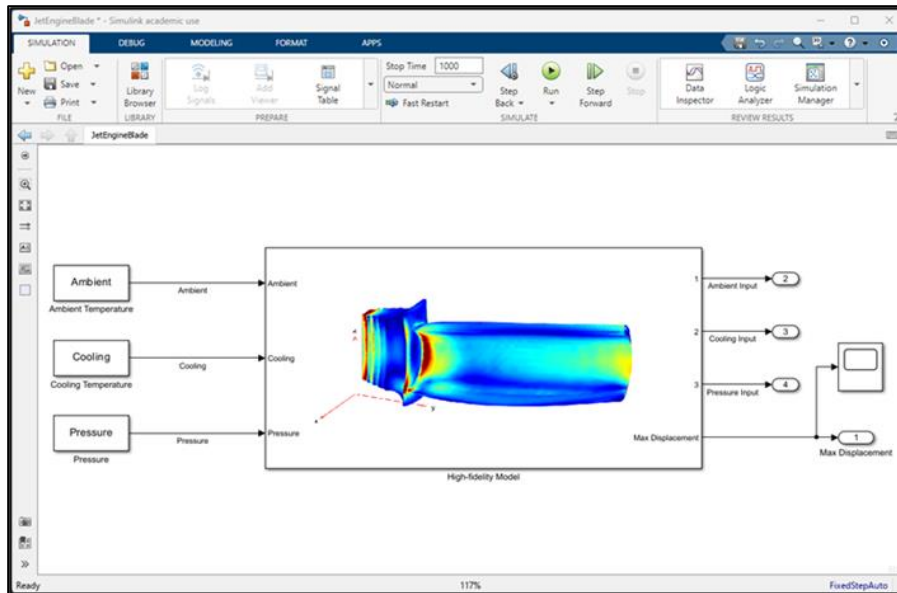
Data Preparation

AI Modeling

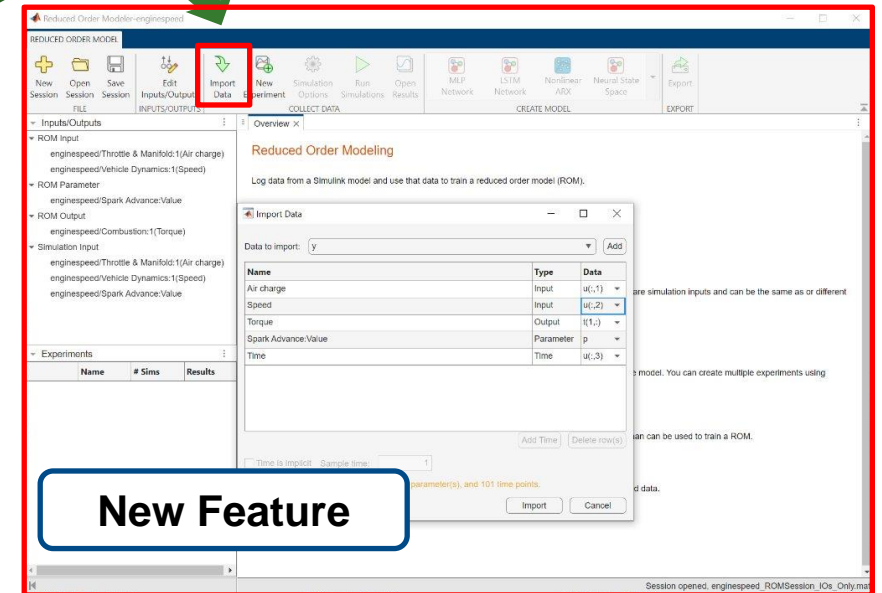
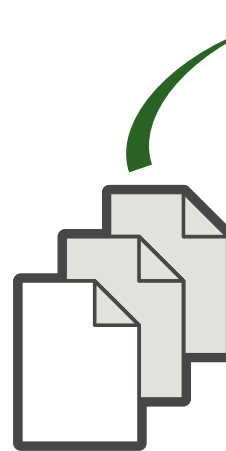
Simulation & Test

Deployment

Prepare data for training AI models



Generate synthetic data from Simulink/Simscape models



Import pre-collected data from high-fidelity model into the app

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

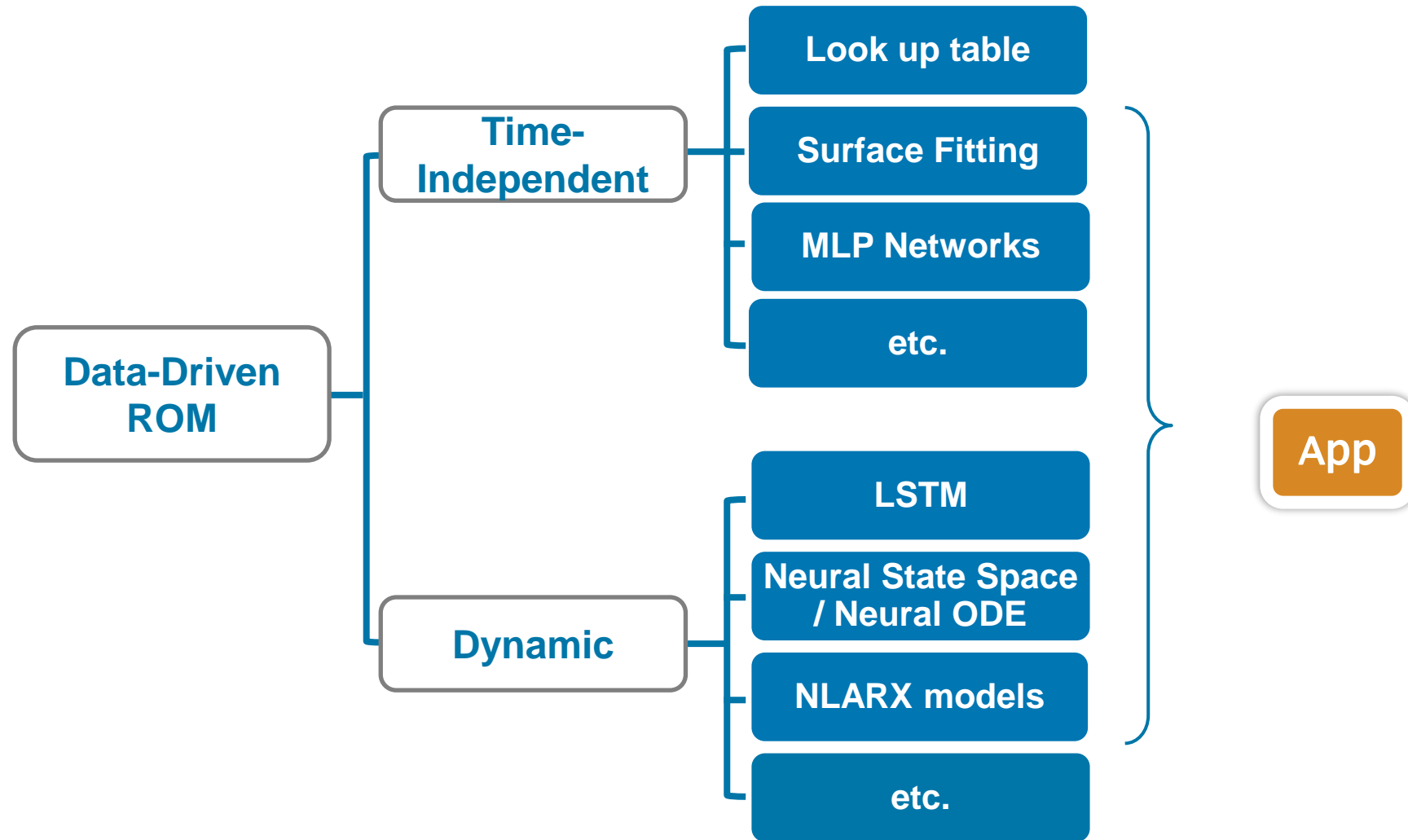
Oliver Wallscheid
*Department of Power Electronics
and Electrical Drives*
Paderborn University
33095 Paderborn, Germany
wallscheid@lea.uni-paderborn.de

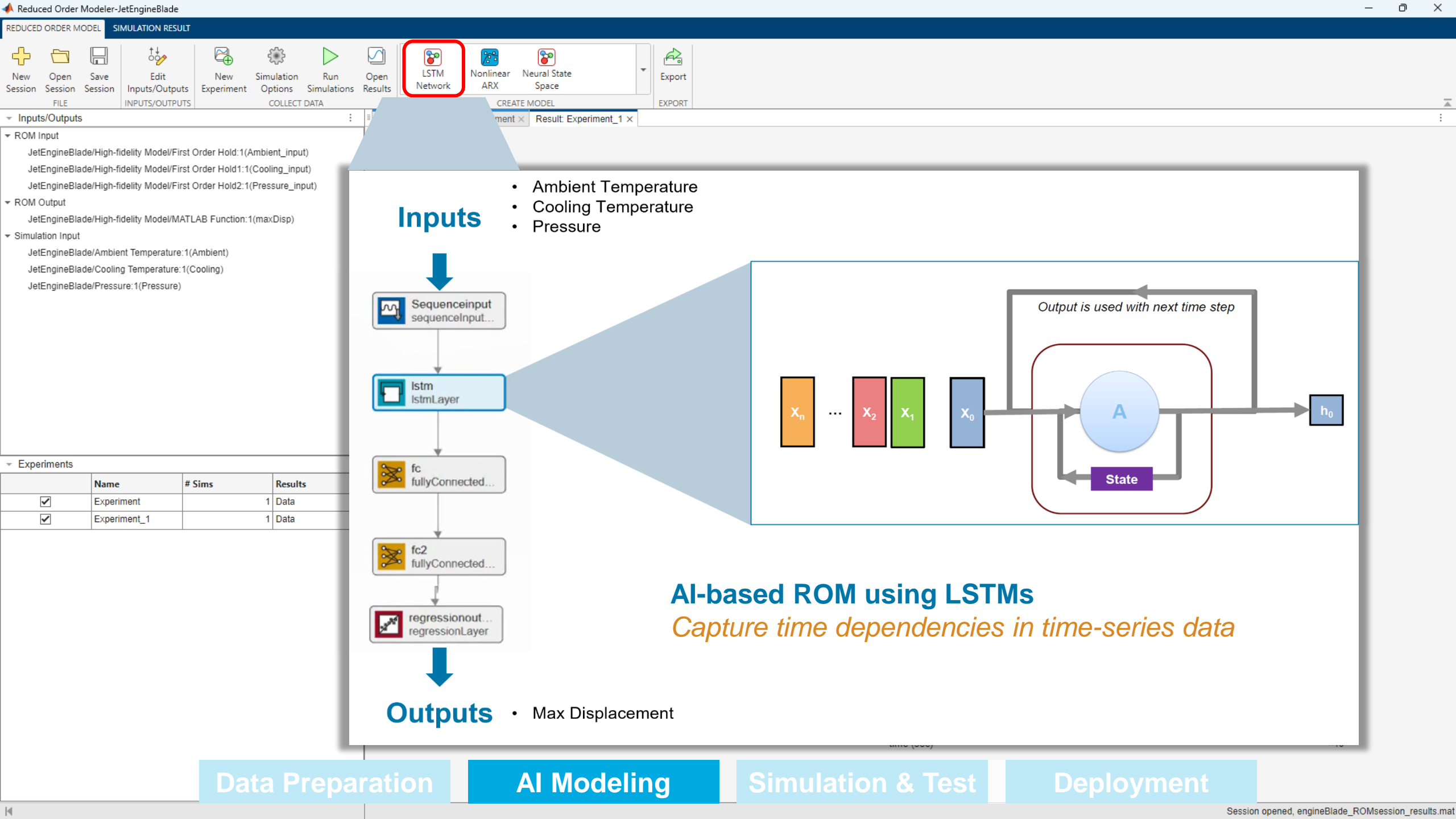
Joachim Böcker
*Department of Power Electronics
and Electrical Drives*
Paderborn University
33095 Paderborn, Germany
boecker@lea.uni-paderborn.de

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

Data-driven ROM





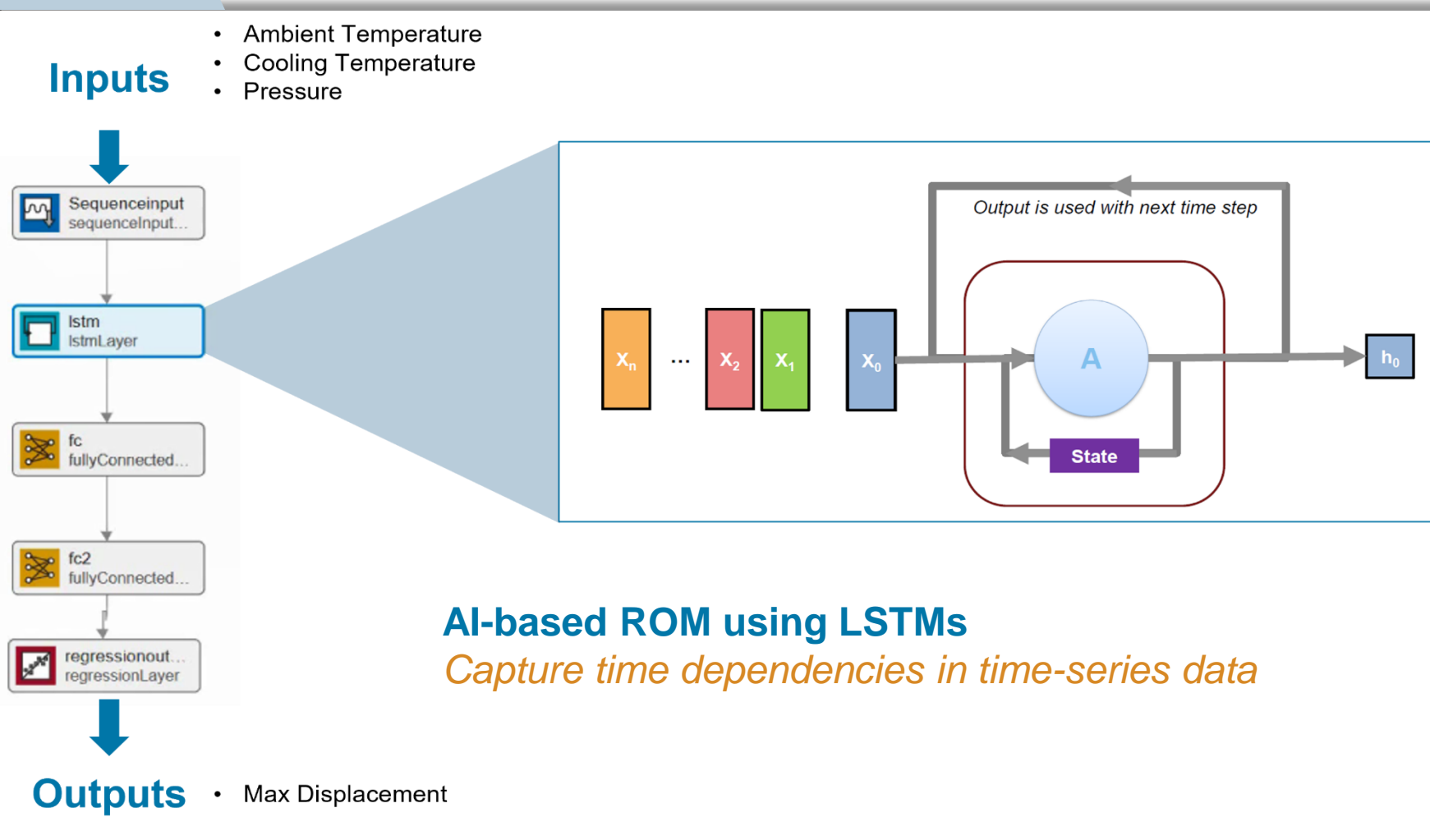
Toolbar icons: New Session, Open Session, Save Session, Edit Inputs/Outputs, New Experiment, Simulation Options, Run Simulations, Open Results, LSTM Network (highlighted), Nonlinear ARX, Neural State Space, Export.

Inputs/Outputs

- ROM Input
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Ambient_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Cooling_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold2:1(Pressure_input)
- ROM Output
 - JetEngineBlade/High-fidelity Model/MATLAB Function:1(maxDisp)
- Simulation Input
 - JetEngineBlade/Ambient Temperature:1(Ambient)
 - JetEngineBlade/Cooling Temperature:1(Cooling)
 - JetEngineBlade/Pressure:1(Pressure)

Experiments

	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment	1	Data
<input checked="" type="checkbox"/>	Experiment_1	1	Data



Simulation Result of 1

Show output only

Show as scatter plot

SIMULATION RESULTS OPTIONS

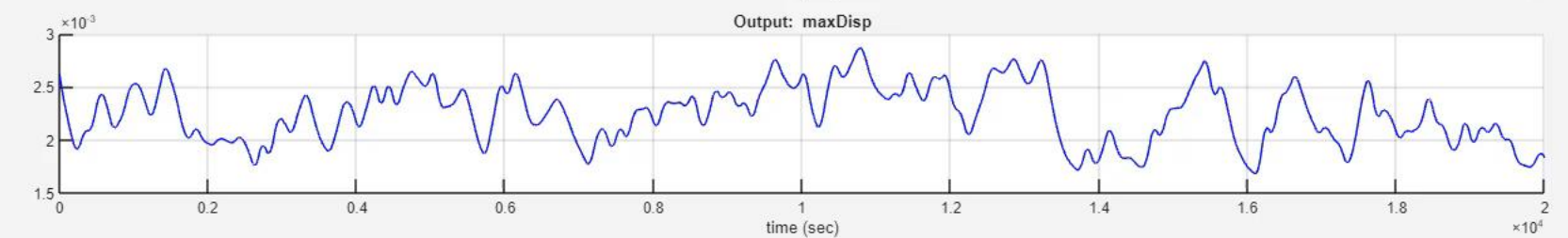
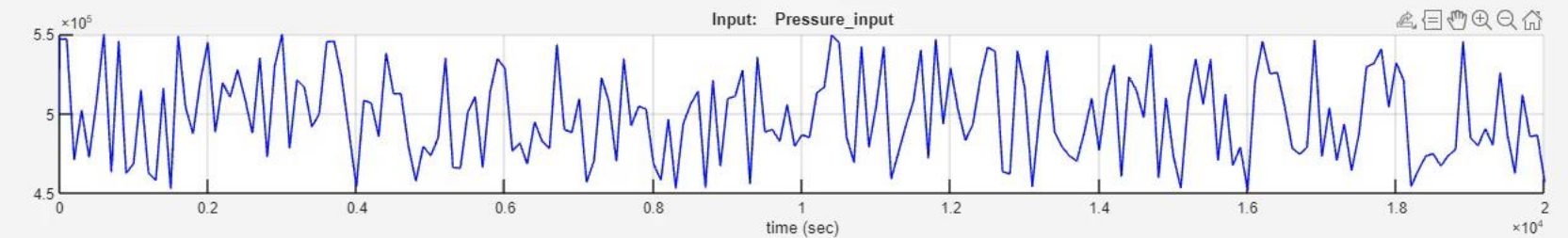
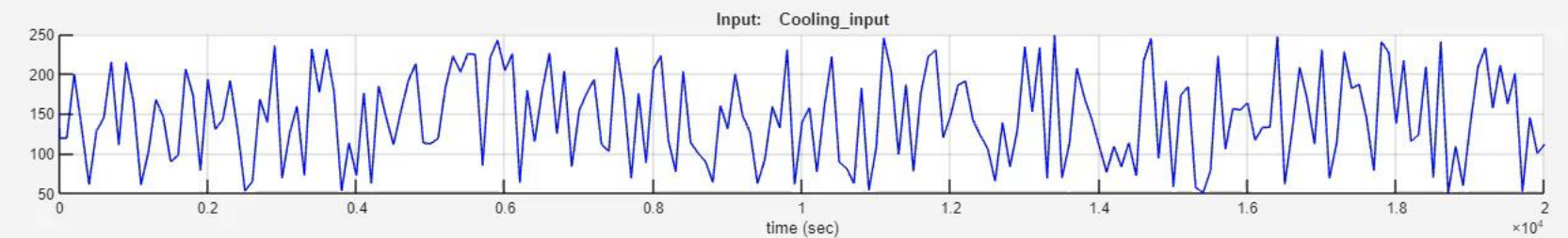
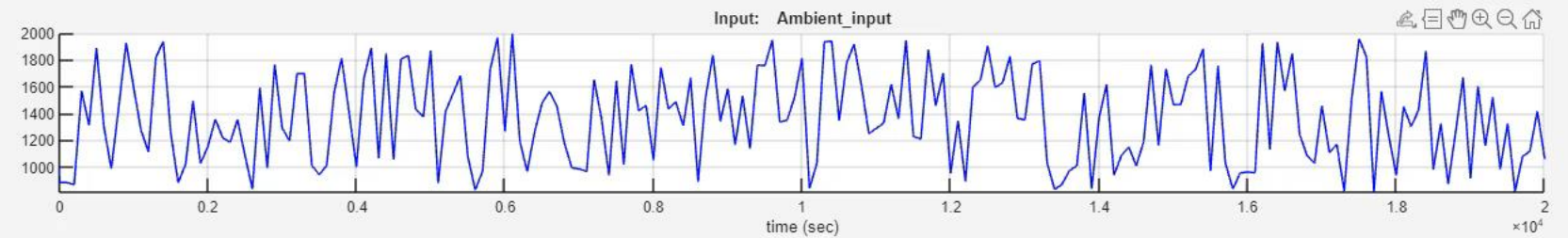
Inputs/Outputs

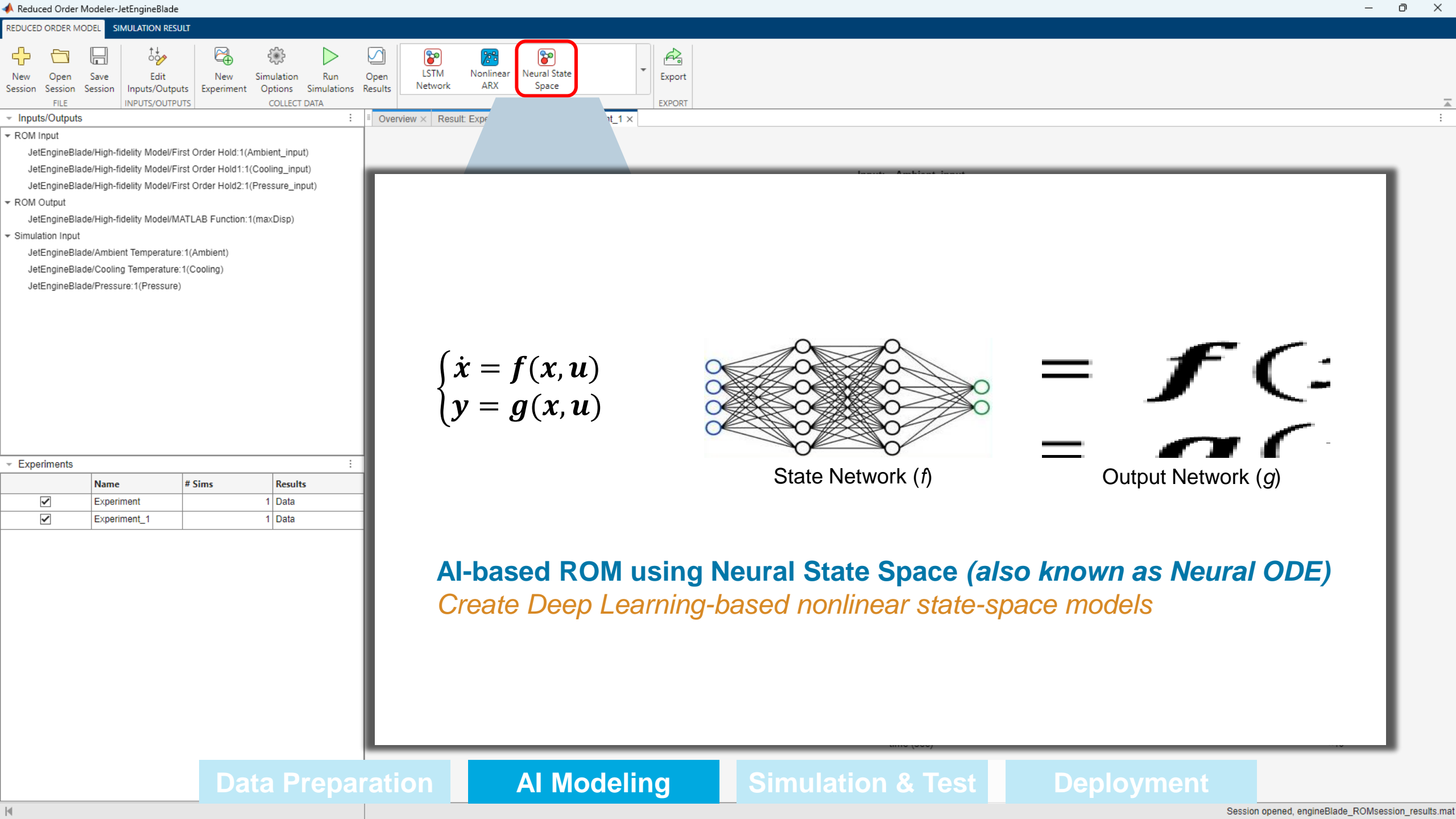
- ROM Input
 - JetEngineBlade/High-fidelity Model/First Order Hold:1(Ambient_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Cooling_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold2:1(Pressure_input)
- ROM Output
 - JetEngineBlade/High-fidelity Model/MATLAB Function:1(maxDisp)
- Simulation Input
 - JetEngineBlade/Ambient Temperature:1(Ambient)
 - JetEngineBlade/Cooling Temperature:1(Cooling)
 - JetEngineBlade/Pressure:1(Pressure)

Experiments

	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment	1	Data
<input checked="" type="checkbox"/>	Experiment_1	1	Data

Overview × Result: Experiment × Result: Experiment_1 × Experiment_1 ×





Inputs/Outputs

ROM Input

- JetEngineBlade/High-fidelity Model/First Order Hold:1(Ambient_input)
- JetEngineBlade/High-fidelity Model/First Order Hold1:1(Cooling_input)
- JetEngineBlade/High-fidelity Model/First Order Hold2:1(Pressure_input)

ROM Output

- JetEngineBlade/High-fidelity Model/MATLAB Function:1(maxDisp)

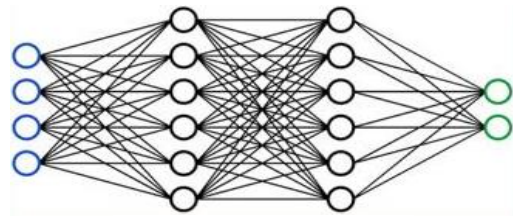
Simulation Input

- JetEngineBlade/Ambient Temperature:1(Ambient)
- JetEngineBlade/Cooling Temperature:1(Cooling)
- JetEngineBlade/Pressure:1(Pressure)

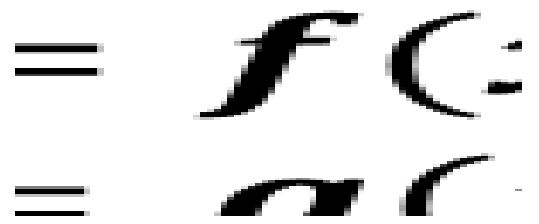
Experiments

	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment		1 Data
<input checked="" type="checkbox"/>	Experiment_1		1 Data

$$\begin{cases} \dot{x} = f(x, u) \\ y = g(x, u) \end{cases}$$



State Network (f)



Output Network (g)

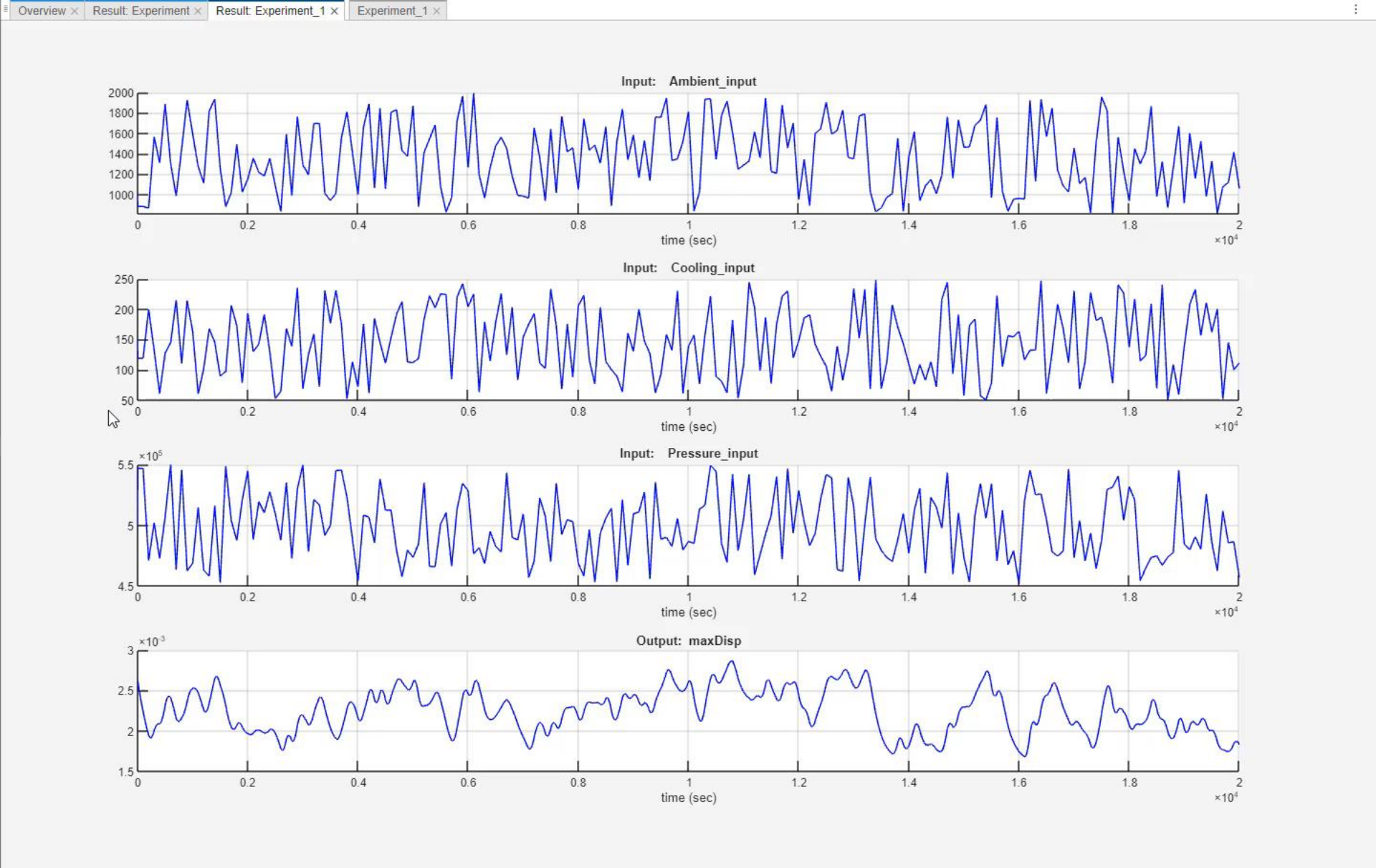
AI-based ROM using Neural State Space (also known as Neural ODE)
Create Deep Learning-based nonlinear state-space models

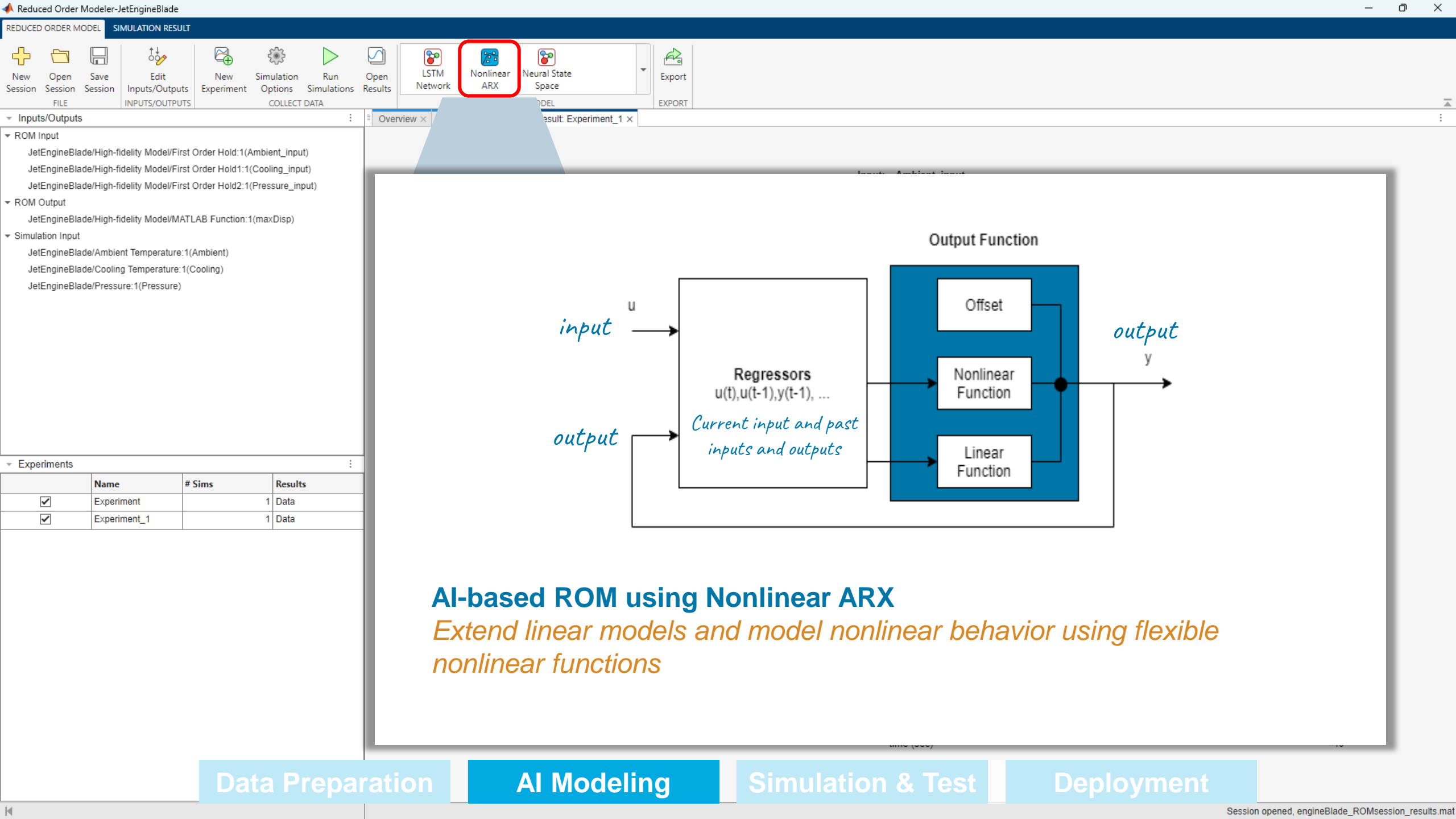
Inputs/Outputs

- ROM Input
 - JetEngineBlade/High-fidelity Model/First Order Hold:1(Ambient_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Cooling_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold2:1(Pressure_input)
- ROM Output
 - JetEngineBlade/High-fidelity Model/MATLAB Function:1(maxDisp)
- Simulation Input
 - JetEngineBlade/Ambient Temperature:1(Ambient)
 - JetEngineBlade/Cooling Temperature:1(Cooling)
 - JetEngineBlade/Pressure:1(Pressure)

Experiments

	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment		1 Data
<input checked="" type="checkbox"/>	Experiment_1		1 Data





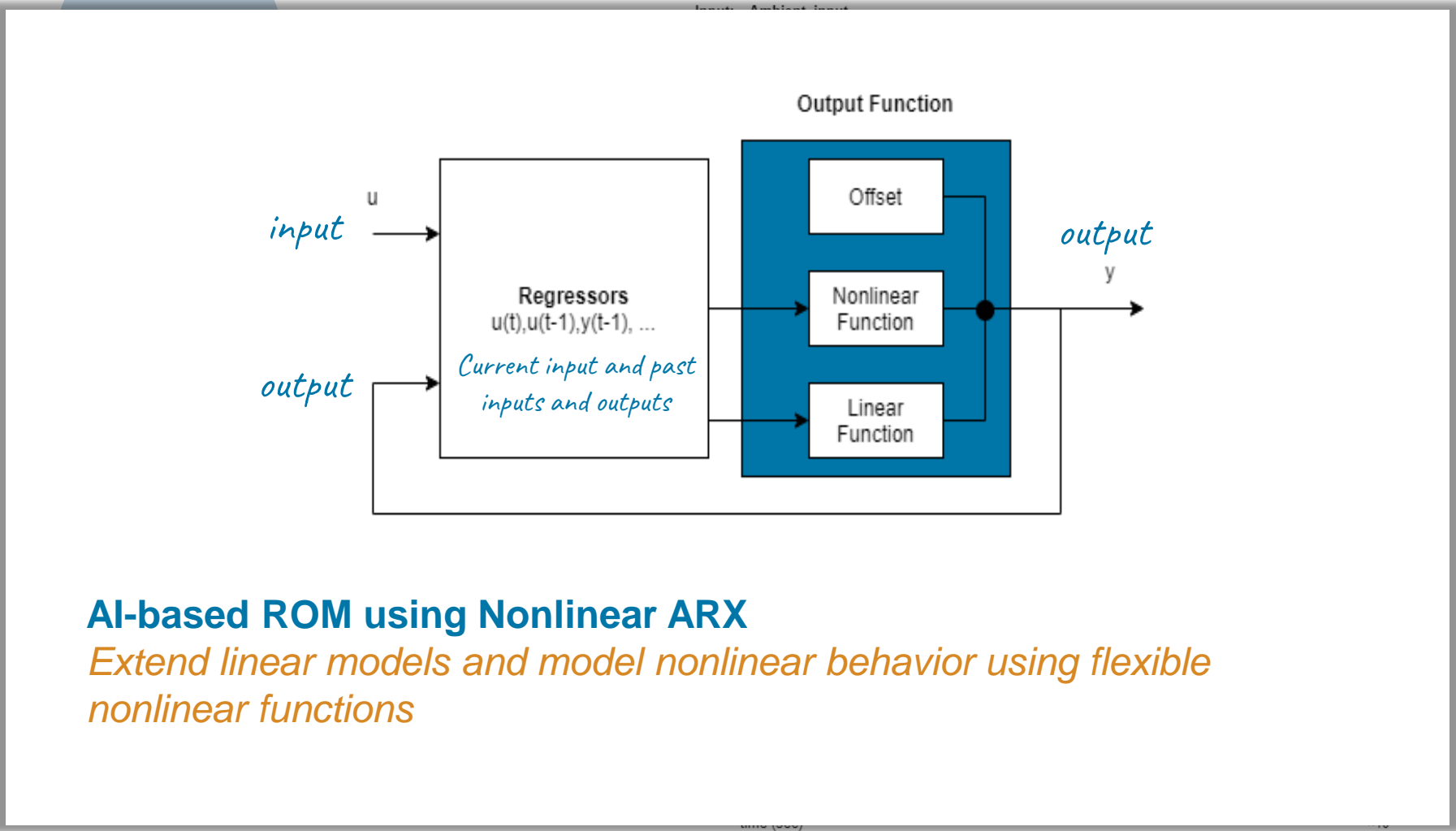
Toolbar icons: New Session, Open Session, Save Session, Edit Inputs/Outputs, New Experiment, Simulation Options, Run Simulations, Open Results, LSTM Network, **Nonlinear ARX**, Neural State Space, Export.

Inputs/Outputs

- ROM Input
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Ambient_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold1:1(Cooling_input)
 - JetEngineBlade/High-fidelity Model/First Order Hold2:1(Pressure_input)
- ROM Output
 - JetEngineBlade/High-fidelity Model/MATLAB Function:1(maxDisp)
- Simulation Input
 - JetEngineBlade/Ambient Temperature:1(Ambient)
 - JetEngineBlade/Cooling Temperature:1(Cooling)
 - JetEngineBlade/Pressure:1(Pressure)

Experiments

	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment		1 Data
<input checked="" type="checkbox"/>	Experiment_1		1 Data

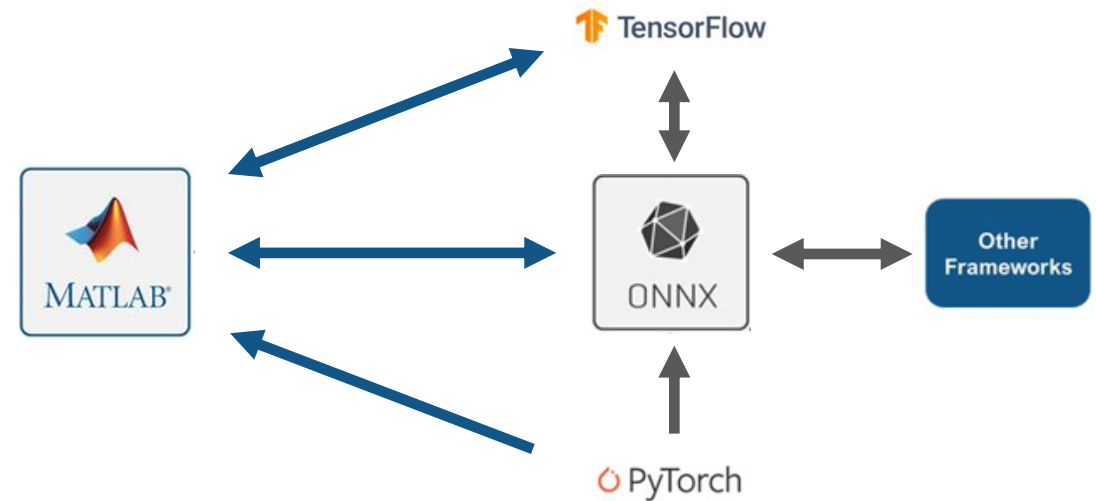


AI-based ROM using Nonlinear ARX
Extend linear models and model nonlinear behavior using flexible nonlinear functions

MATLAB interoperates with other frameworks

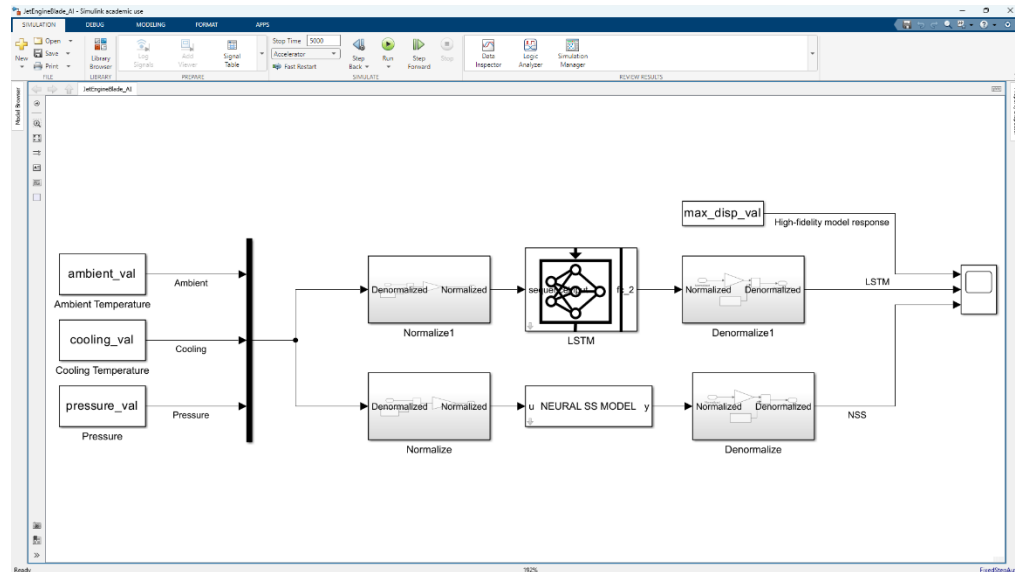
Framework interoperability bridges the gap between data science, engineering and production

TensorFlow-Keras Import	R2017b
ONNX Converter (Import & Export)	R2018a
TensorFlow Converter (Import)	R2021a
TensorFlow Converter (Export)	R2022b
PyTorch Converter (Import)	R2022b

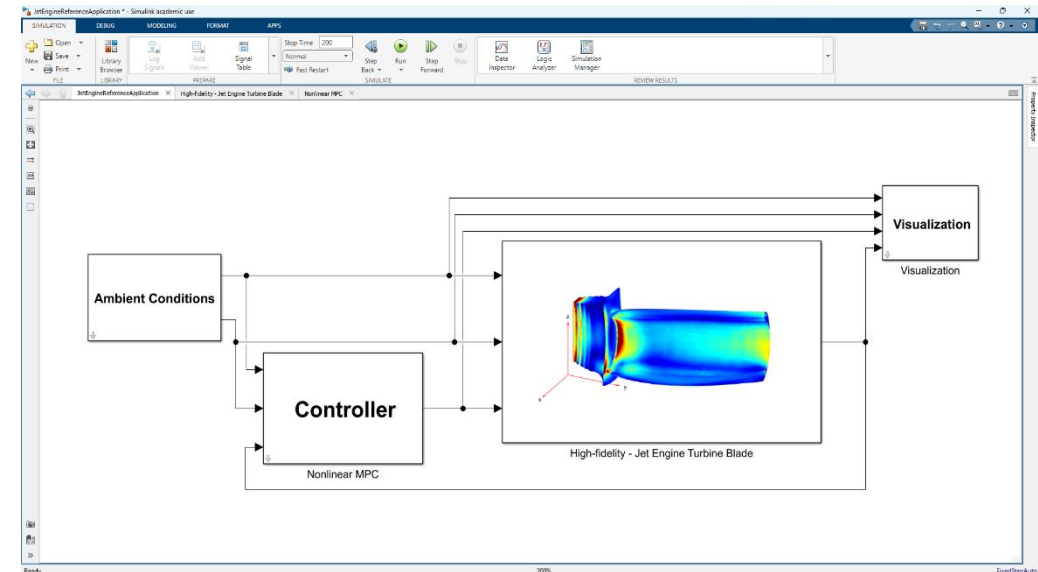


Integrate your AI model for system-level simulation and test

Integration of trained AI model into Simulink



System-level simulation



AI libraries in Simulink are expanding to include more AI blocks for more applications

Specialized

Audio Toolbox

Computer Vision Toolbox

AI Core

Statistics and Machine Learning Toolbox

Deep Learning Toolbox

System Identification Toolbox

Integration of trained AI models into Simulink

The screenshot displays the MATLAB R2023b Live Editor interface. The top menu bar includes HOME, PLOTS, APPS, LIVE EDITOR, INSERT, and VIEW. The current folder is 'C:\Documents > ROMSeminar > SimulationAndCodeGeneration'. The workspace shows variables: trainingOutput_lstm (1x1 struct) and trainingOutput_nss (1x1 struct). The main editor window displays a document titled 'Experiment to train a NSS model' with the following content:

Experiment to train a NSS model

Train a NSS model. Hyper-parameters for training are:

- NumberInputLags - The number of lagged inputs to use, an integer ≥ 0
- NumberOutputLags - The number of lagged outputs to use, an integer ≥ 0
- NumberLayers - The number of layers in the MLP, an integer > 0
- NumberUnits - The number of hidden units in each layer, an integer > 0
- SampleRate - Sample rate of the model, a real > 0

The tuning follows the following automated steps:

1. Extract and resample the training data
2. Train the NSS model
3. Evaluate model on test data (if available)

```

1 function output = Experiment2_training1(params,monitor)
2
3
4
5
6 % TestSplit - For multiple data sets the percentage of data sets to use
7 % for testing, a double in range [0 100]. The test data sets are selected
8 % randomly from the available data sets.
9 testSplit = 20;
10
11 % BatchSize - The number of data points to use when converting signals into
12 % min-batches, i.e., collections of smaller signal segments.
13 batchSize = 20;

```

The Command Window shows the prompt `>>`.

Data Preparation

AI Modeling

Simulation & Test

Deployment

Integration of trained AI models into Simulink

Simulink Profiler

Path	Time Plot (Dark Band = Self Time)	Total Time (s)	Self Time (s)	Number of Calls
▼ JetEngineBlade_AI		17.207	1.807	2014
> LSTM		11.465	0.000	0
Scope1		3.895	3.895	1004
> Neural State Space Model		0.028	0.000	0
From Workspace1		0.008	0.008	1003
Ambient Temperature		0.002	0.002	1003
Cooling Temperature		0.001	0.001	1003
Pressure		0.001	0.001	1003
> Normalize1		0.000	0.000	0
> Denormalize1		0.000	0.000	0
> Denormalize		0.000	0.000	0
> Normalize		0.000	0.000	0

Neural state-space model is approximately 1e6x faster than the FEA model

Deep Learning Toolbox verification library

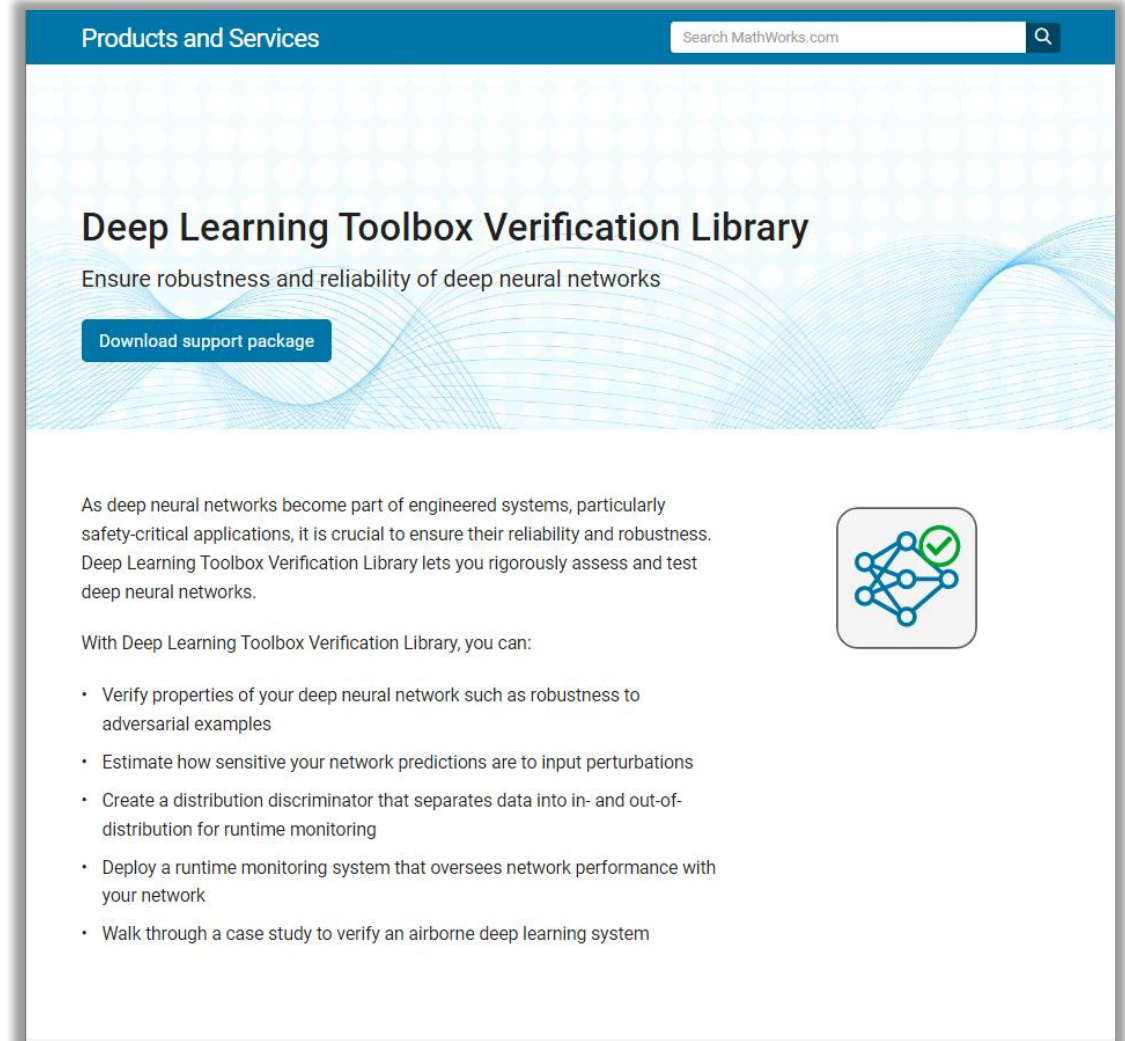
Ensure robustness and reliability of deep neural networks

Verify Deep Neural Network
Robustness for Classification

Estimate Deep Neural Network
Output Bounds for Regression

Build Safe Deep Learning Systems
with Runtime Monitoring

Case Study: Verifying an
Airborne Deep Learning System



The screenshot shows the product page for the Deep Learning Toolbox Verification Library. At the top, there is a navigation bar with "Products and Services" and a search bar. The main heading is "Deep Learning Toolbox Verification Library" with the subtext "Ensure robustness and reliability of deep neural networks". A prominent blue button labeled "Download support package" is visible. Below this, a paragraph explains the importance of ensuring reliability and robustness for deep neural networks in safety-critical applications. To the right of this text is an icon of a neural network with a green checkmark. Further down, a section titled "With Deep Learning Toolbox Verification Library, you can:" lists five key capabilities: verifying properties like robustness to adversarial examples, estimating sensitivity to input perturbations, creating distribution discriminators for runtime monitoring, deploying runtime monitoring systems, and walking through a case study for an airborne system.


Products and Services

Deep Learning Toolbox Verification Library

Ensure robustness and reliability of deep neural networks

[Download support package](#)

As deep neural networks become part of engineered systems, particularly safety-critical applications, it is crucial to ensure their reliability and robustness. Deep Learning Toolbox Verification Library lets you rigorously assess and test deep neural networks.

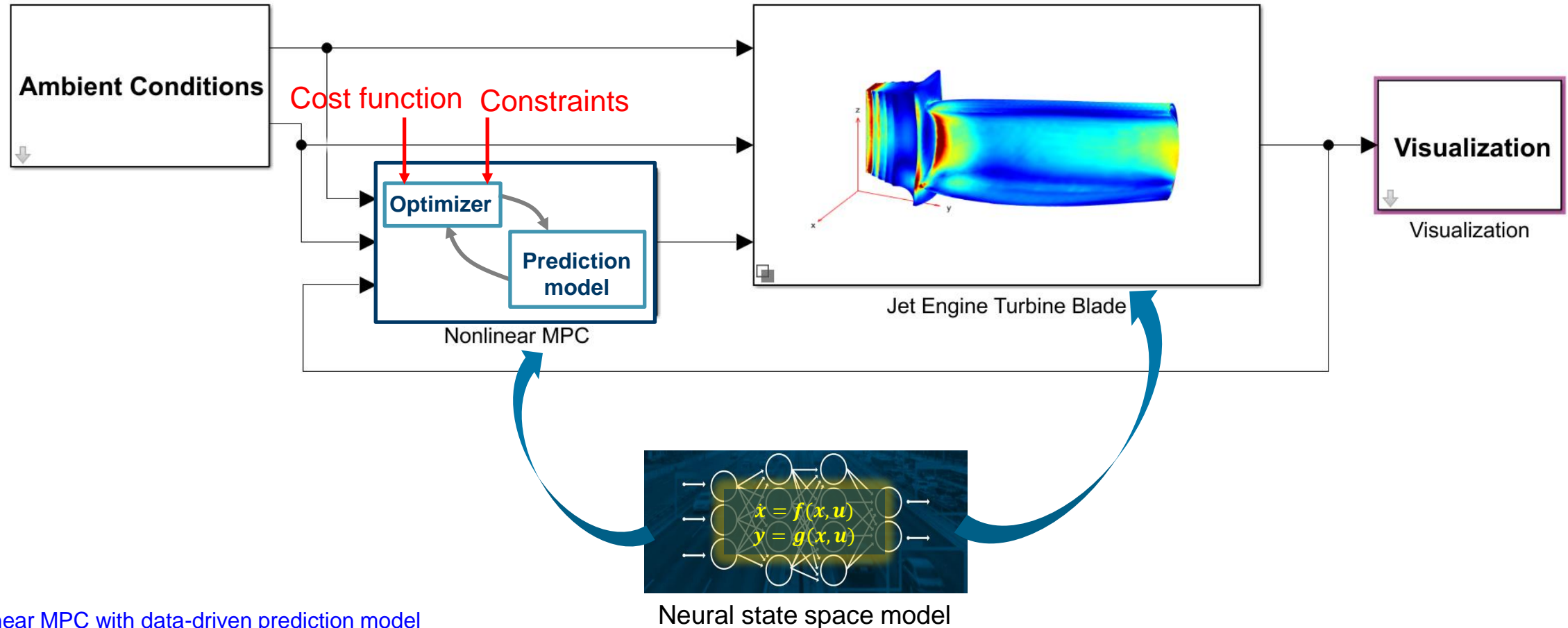


With Deep Learning Toolbox Verification Library, you can:

- Verify properties of your deep neural network such as robustness to adversarial examples
- Estimate how sensitive your network predictions are to input perturbations
- Create a distribution discriminator that separates data into in- and out-of-distribution for runtime monitoring
- Deploy a runtime monitoring system that oversees network performance with your network
- Walk through a case study to verify an airborne deep learning system

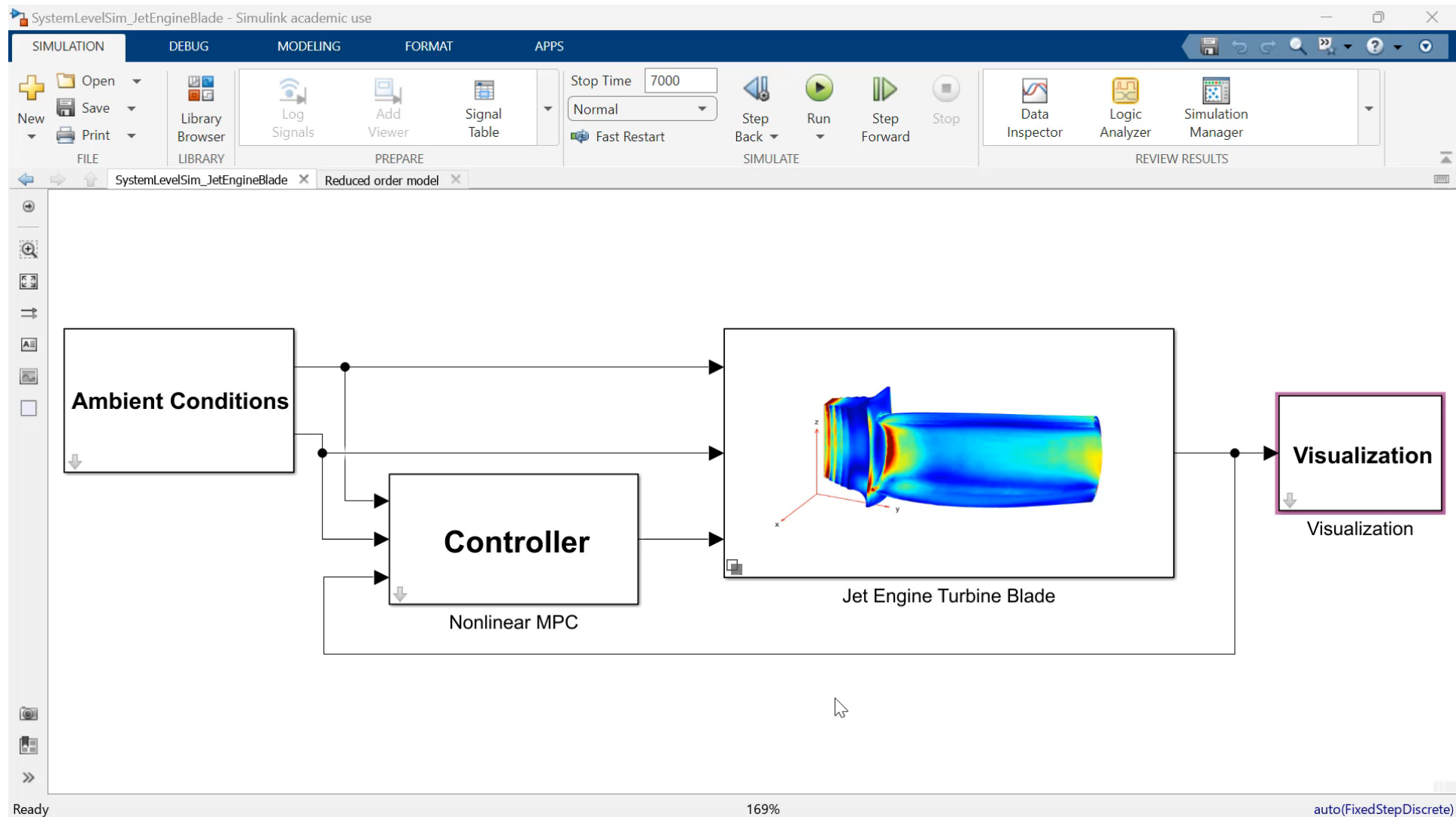
Control Design with Model Predictive Controller

SIMULINK®



[Nonlinear MPC with data-driven prediction model](#)

System-level simulation



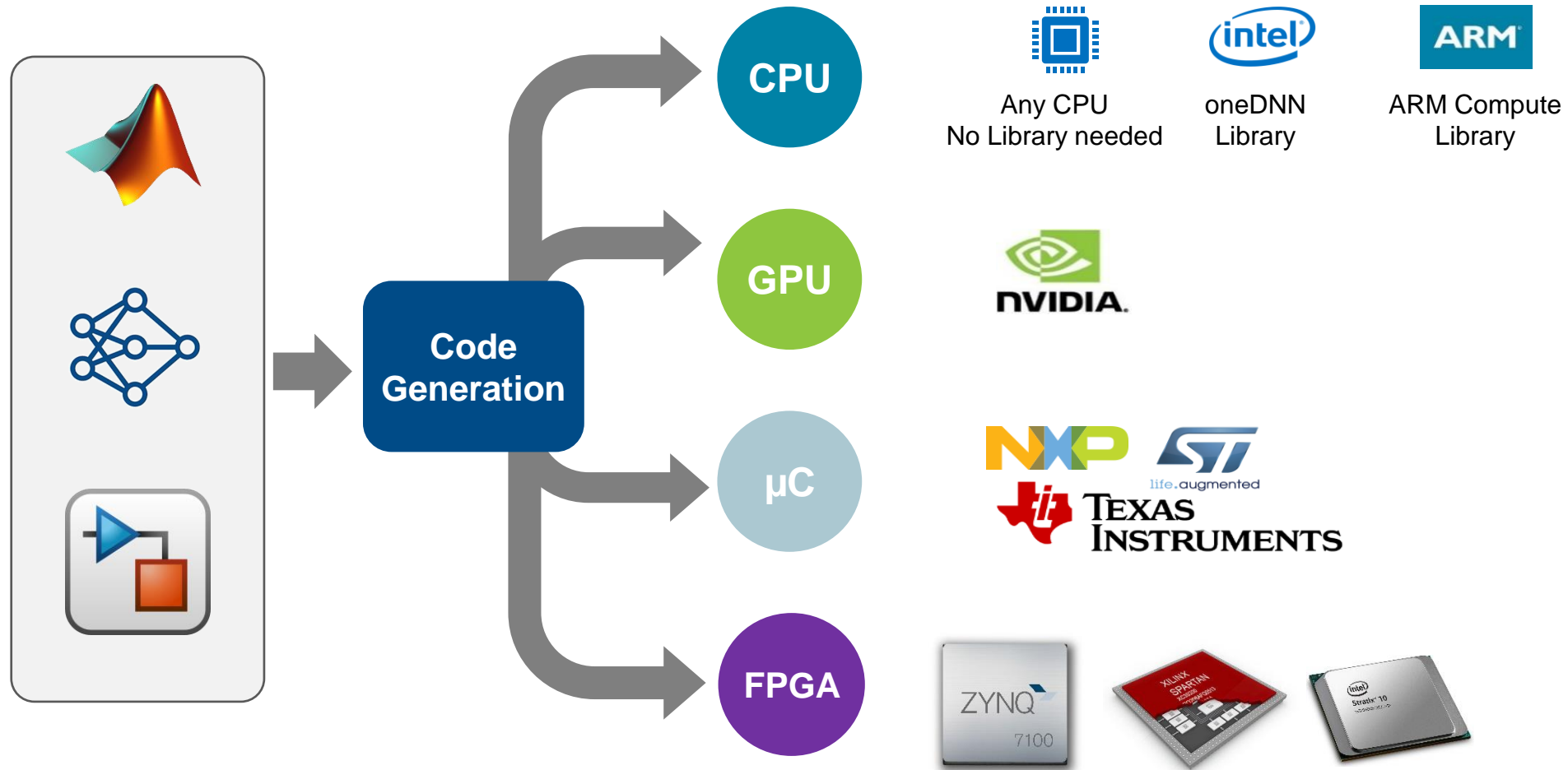
Data Preparation

AI Modeling

Simulation & Test

Deployment

Deploy to target with zero coding errors



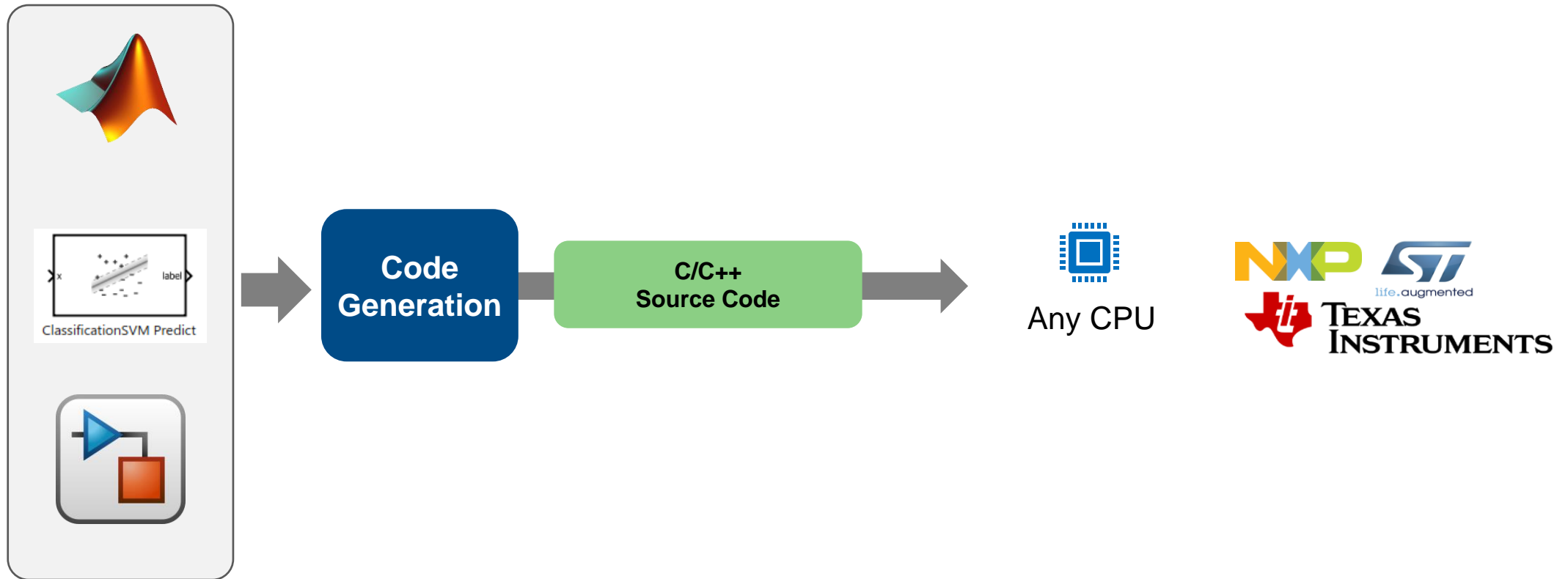
Data Preparation

AI Modeling

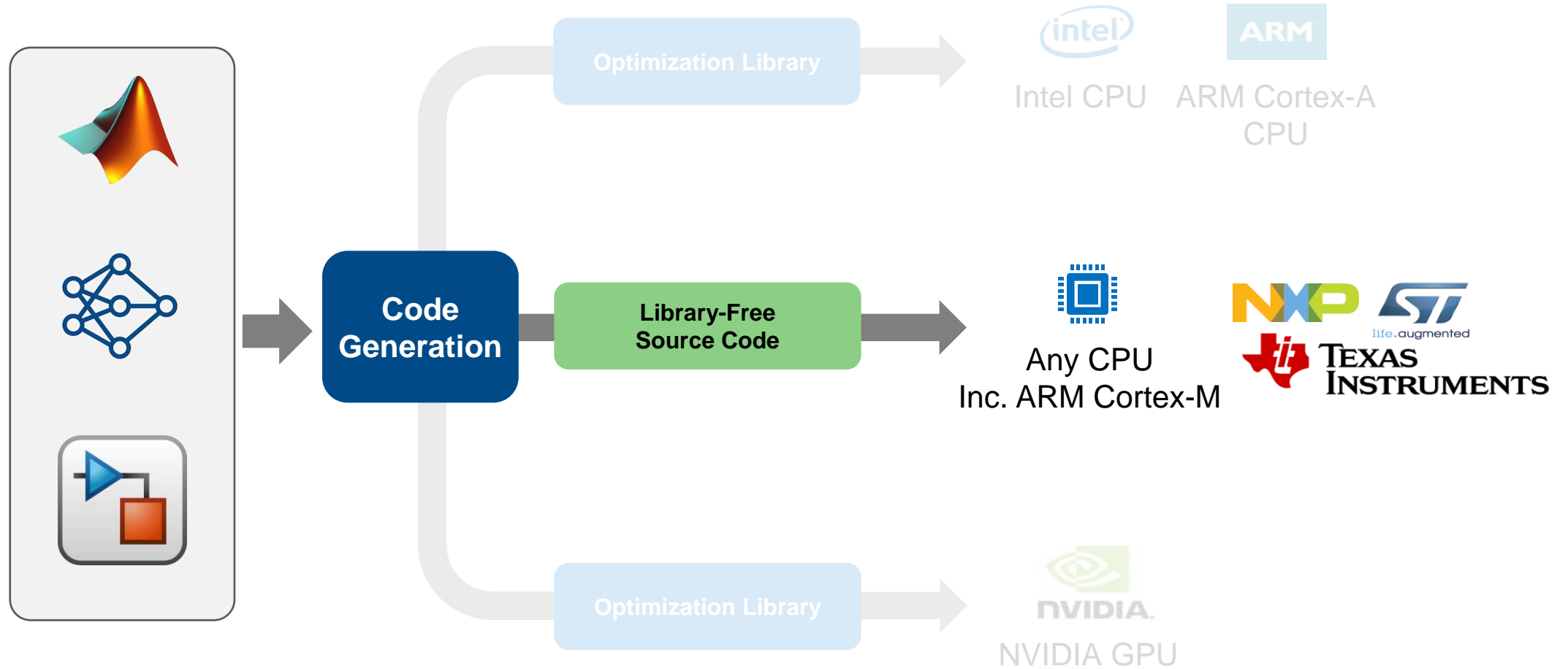
Simulation & Test

Deployment

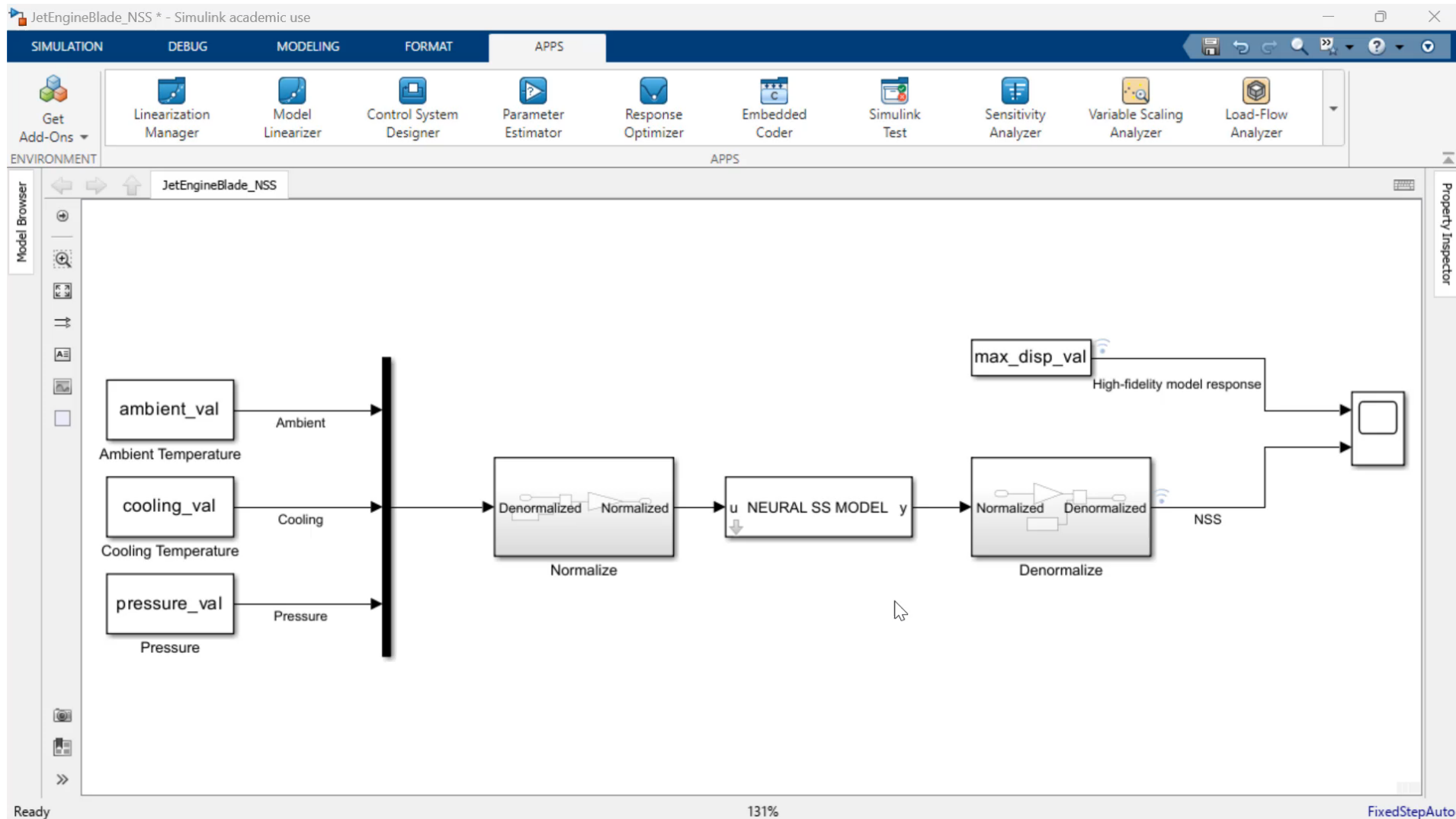
Use Embedded Coder to generate code for machine learning



Generate library-free C/C++ code for deep learning networks



Generate Library-Free C Code for Deep Learning Networks



Data Preparation

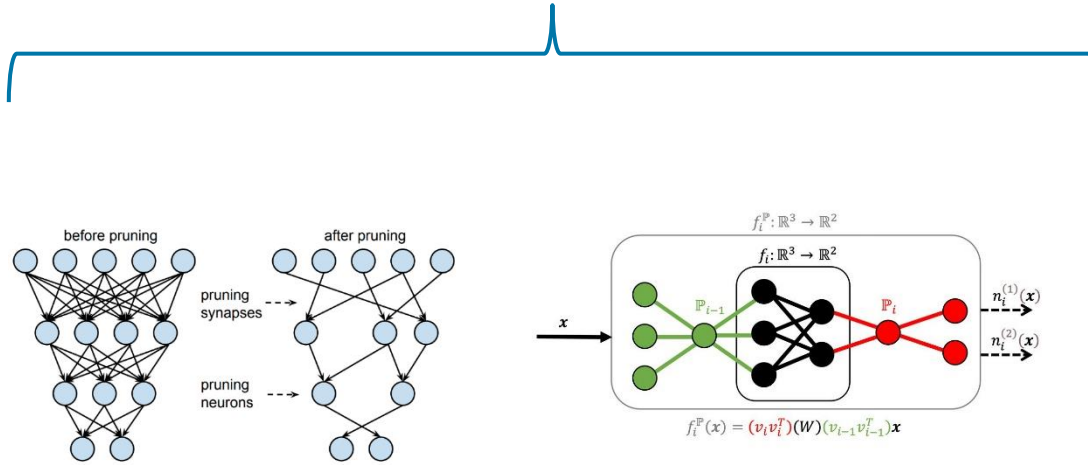
AI Modeling

Simulation & Test

Deployment

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

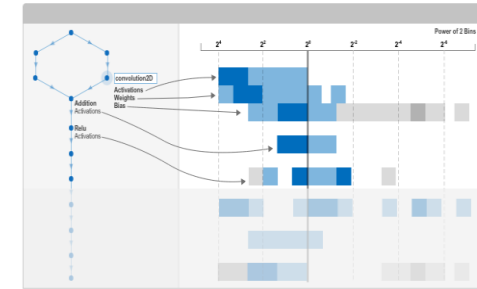
Structural Compression



Pruning
convolutional neural
networks

Projection of deep
neural networks

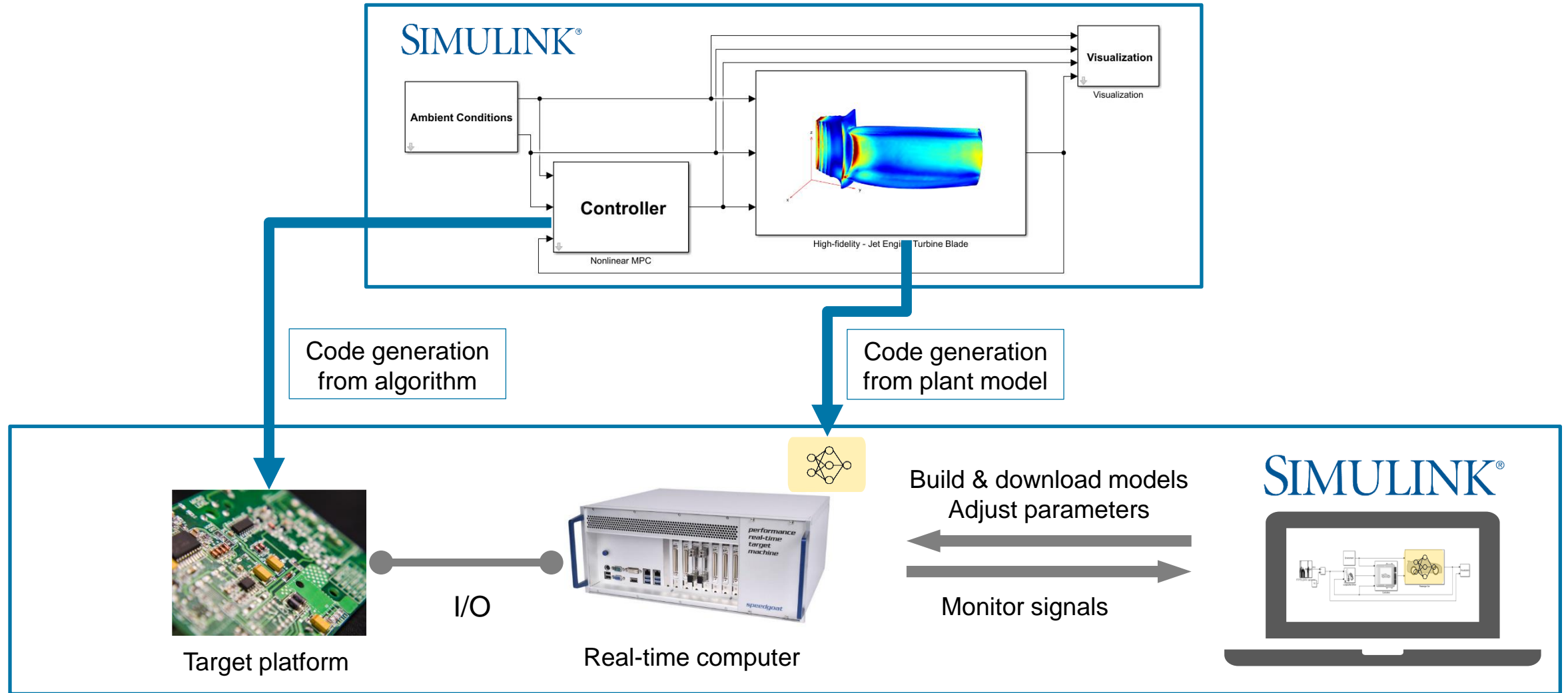
Datatype Compression



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

Hardware-in-the-loop simulation

System-level integration and test



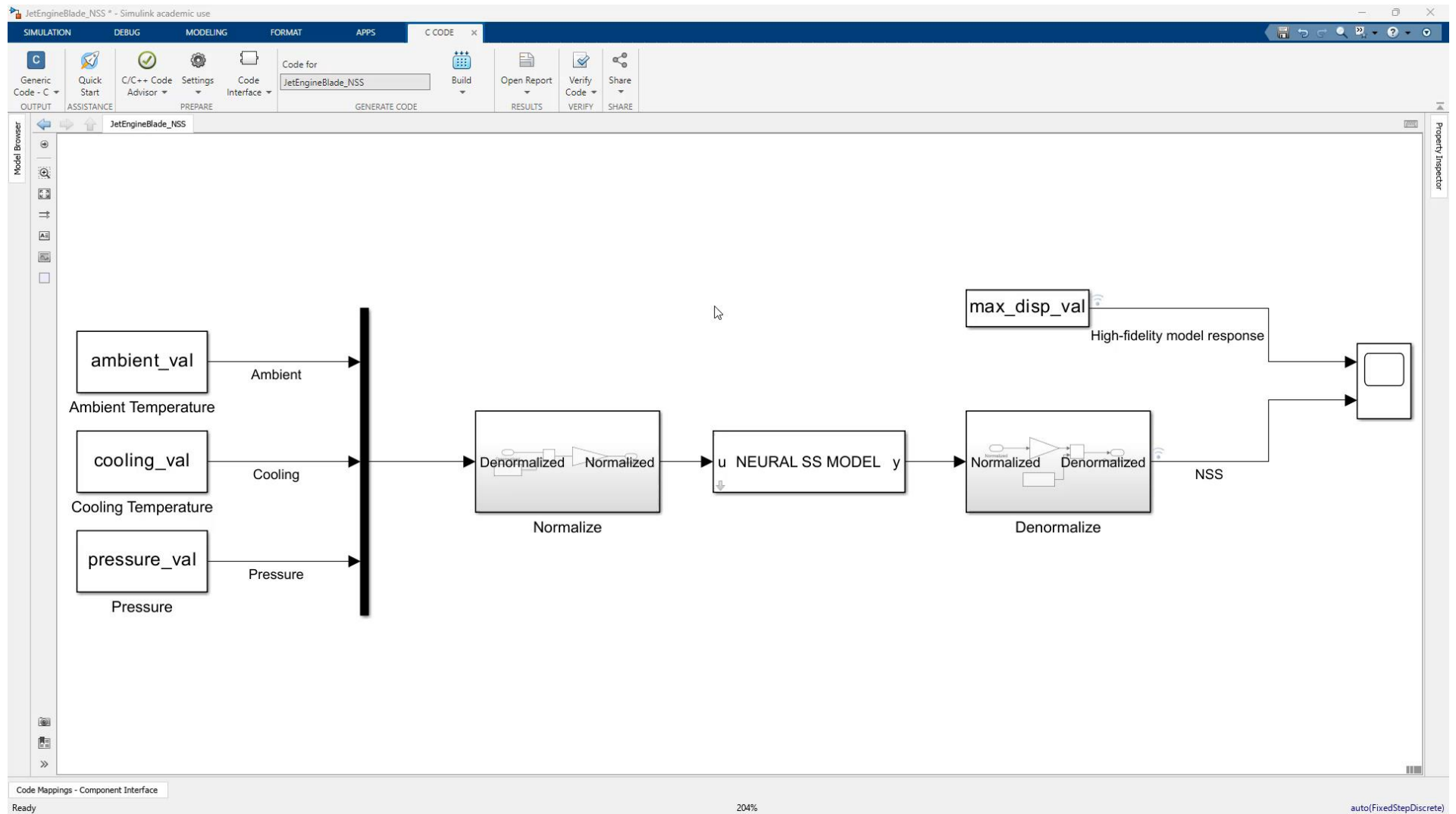
Data Preparation

AI Modeling

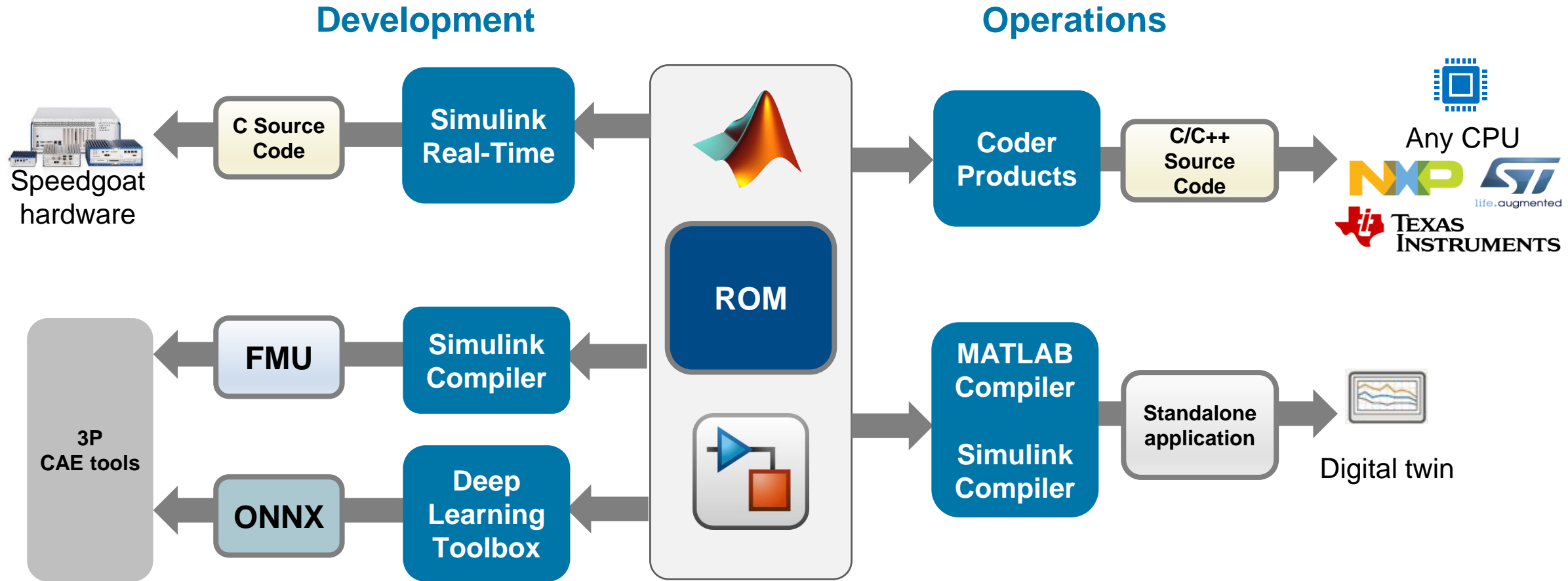
Simulation & Test

Deployment

Hardware-in-the-loop simulation



Use ROMs outside of Simulink, for development and operation stages



Manage AI tradeoffs for your system

	LSTM Long Short-Term Memory Network	Neural State Space (Neural ODE)
Training Speed	●	●
Interpretability	●	●
Inference Speed	●	●
Model Size	●	●
Accuracy (RSME)	●	●

Results are specific to Jet Engine Blade Example

Better ●	Okay ●	Worse ●
----------	--------	---------

Reduced order modeling user stories



Subaru developed a **surrogate AI model to optimize transmission hydraulic systems**, achieving a **99% reduction in calculation times** compared to the original third-party 1D model

[Link to user story](#)



Cummins implemented a **deep learning neural network** to improve the **speed of engine cycle simulations for performance predictions**

[Link to user story](#)

MathWorks service and support mechanisms

MathWorks has a team of over 700 **customer-facing engineers** – we welcome the opportunity to discuss how you can get the most out of your software investments and achieve your goals.



Technical Support

- Product questions
- General support
- 508-647-7000



AE (Application Engineering) Support

- Product/Capability demonstrations
- Workshops, Webinars, etc.
- Evaluation support



Extended AE Support

- Guided support for adoption of new tools/processes
- Deep Engagements
- Proof of Concept



Professional Courses

- Paid training on specific tools and/or processes
- On-site, web-based instructor lead, & self-paced online



Consulting Engineering

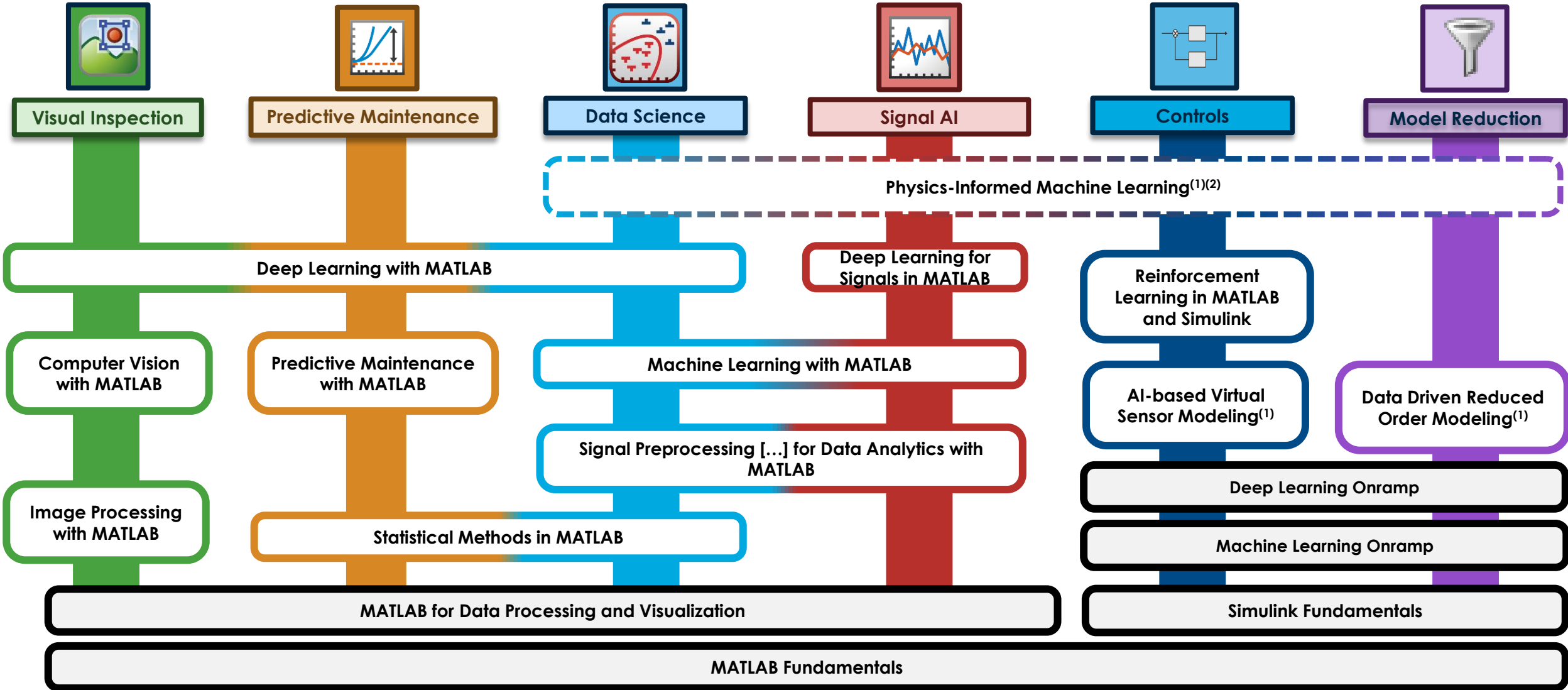
- Paid engagements (custom targets, tool customization, advisory services)

Complimentary

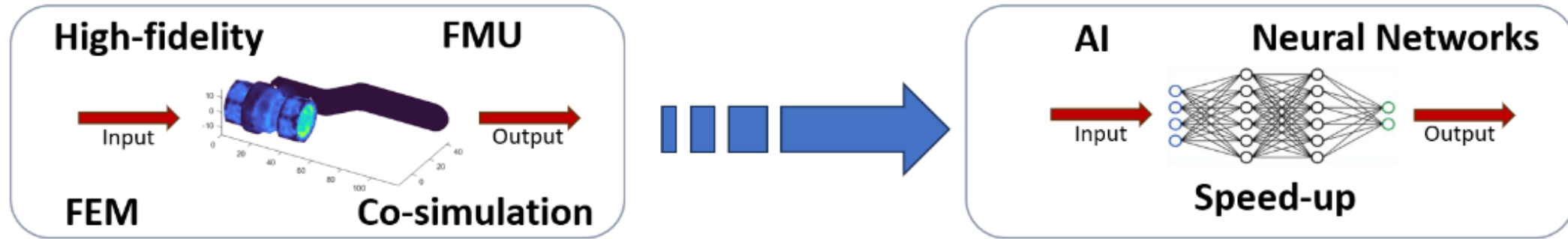
Funded

(1) available as private-only
 (2) module, not a full training

AI training pathways



2-day custom training available for data-driven reduced order modeling



- Generating data for reduced order modeling
- Data-driven AI-ROM models as surrogates for high-fidelity components
- Validation and simulation with reduced order models
- Recurrent neural networks, neural state space models, and system identification techniques

[Full Course Outline](#)

*Available upon request as private training only

Key takeaways

Enable

Reuse of full-order high-fidelity models for system-level simulations, Hardware-in-the-Loop (HIL) testing, nonlinear control design, and virtual sensor modeling

Explore

Various ROM techniques in MATLAB to find the best method.

- **Generate synthetic data** from Simulink
- **Train AI Models** to replace high-fidelity battery electrochemical and PMSM model
- **Integrate trained AI model into Simulink** for control design and system-level simulation
- **Generate C code and perform HIL tests**

