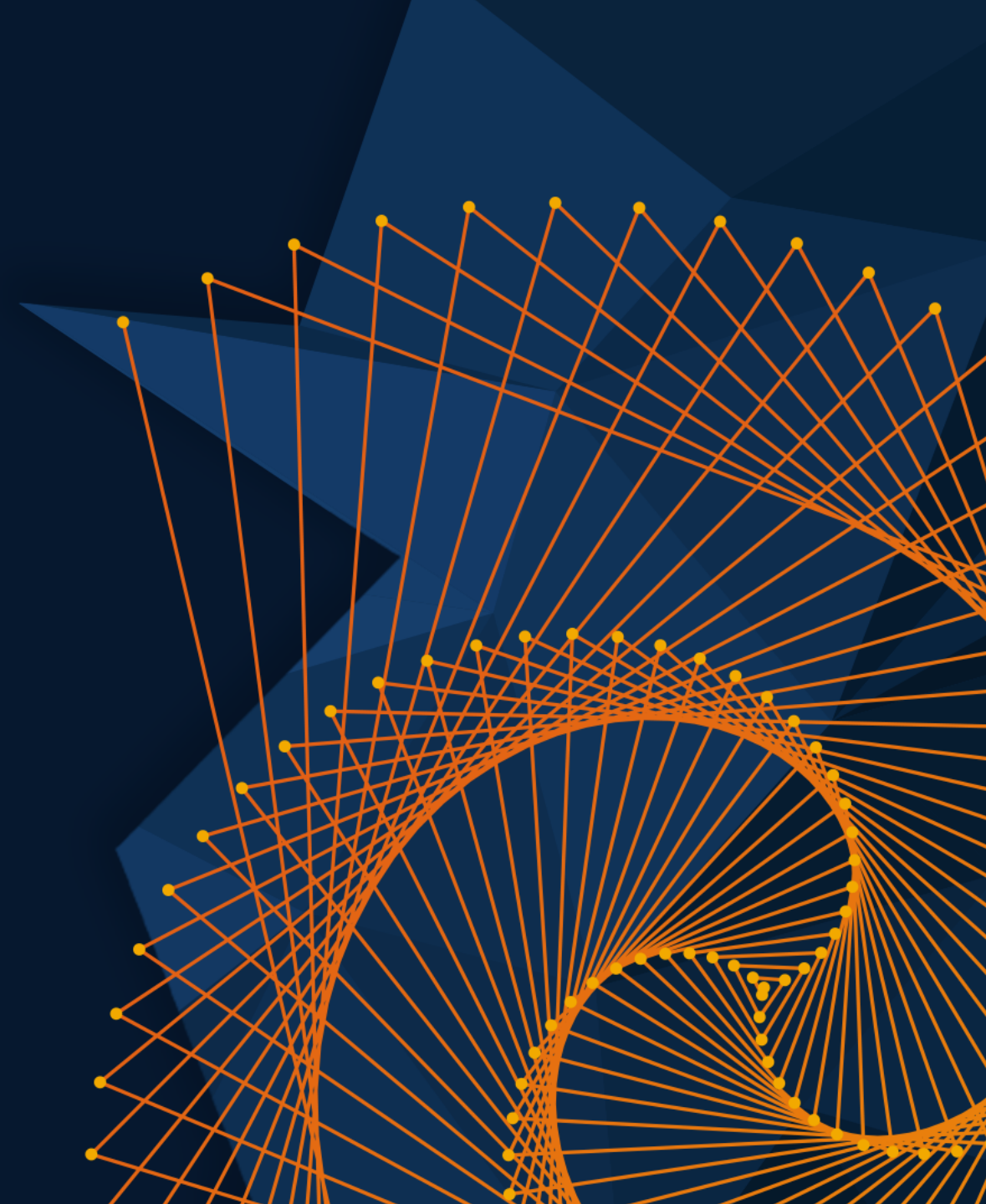


MATLAB EXPO

June 4, 2024 | Warsaw

AI with Model-Based Design: Reduced Order Modeling

Sebastian Bomberg, MathWorks



SIMULATION DEBUG MODELING FORMAT APPS **VARIANT SUBSYSTEM**

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

FILE LIBRARY PREPARE SIMULATE REVIEW RESULTS

SystemLevelSim_JetEngineBlade x Reduced order model x

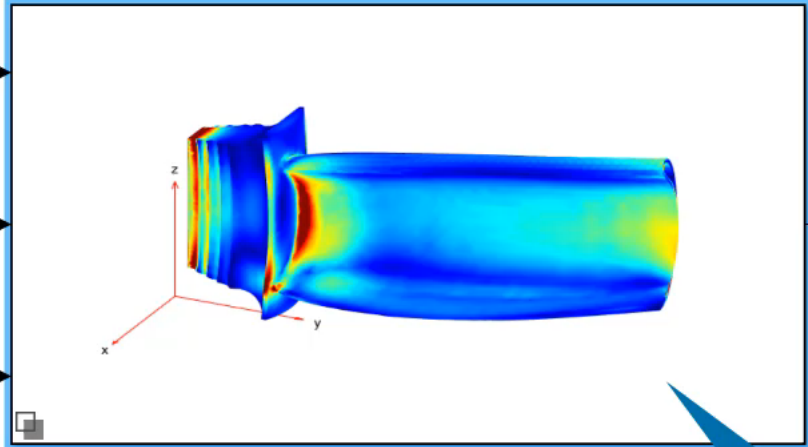


Temperature and pressure conditions

Cooling temperature controller



Nonlinear MPC



Visualization

FEA model to compute maximum tip displacement

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.

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

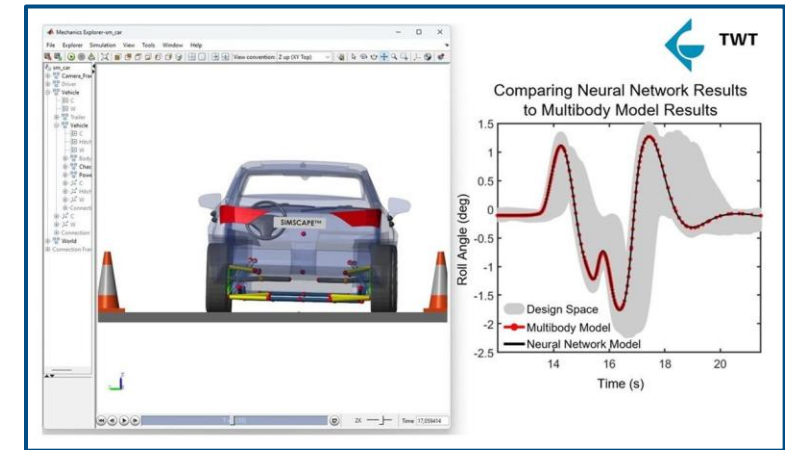
Various ROM techniques in MATLAB to find the best method.

TWT GmbH Develops New Workflow for Tuning Automotive Suspension Designs Using Deep Learning and Multibody Simulation

Using MATLAB, Simulink, and Deep Learning Toolbox, TWT GmbH has accelerated the optimization of automotive suspension designs via a deep learning network trained using data from simulations of a Simscape multibody model.

Key Outcomes/Advantages:

- New workflow with MATLAB and Simulink improved suspension design, helping reduce roll angle by up to 50%
- Using Global Optimization Toolbox reduced optimization time for suspension simulations from 16 days to 5 minutes
- Single-environment workflow established for multibody simulation and deep learning

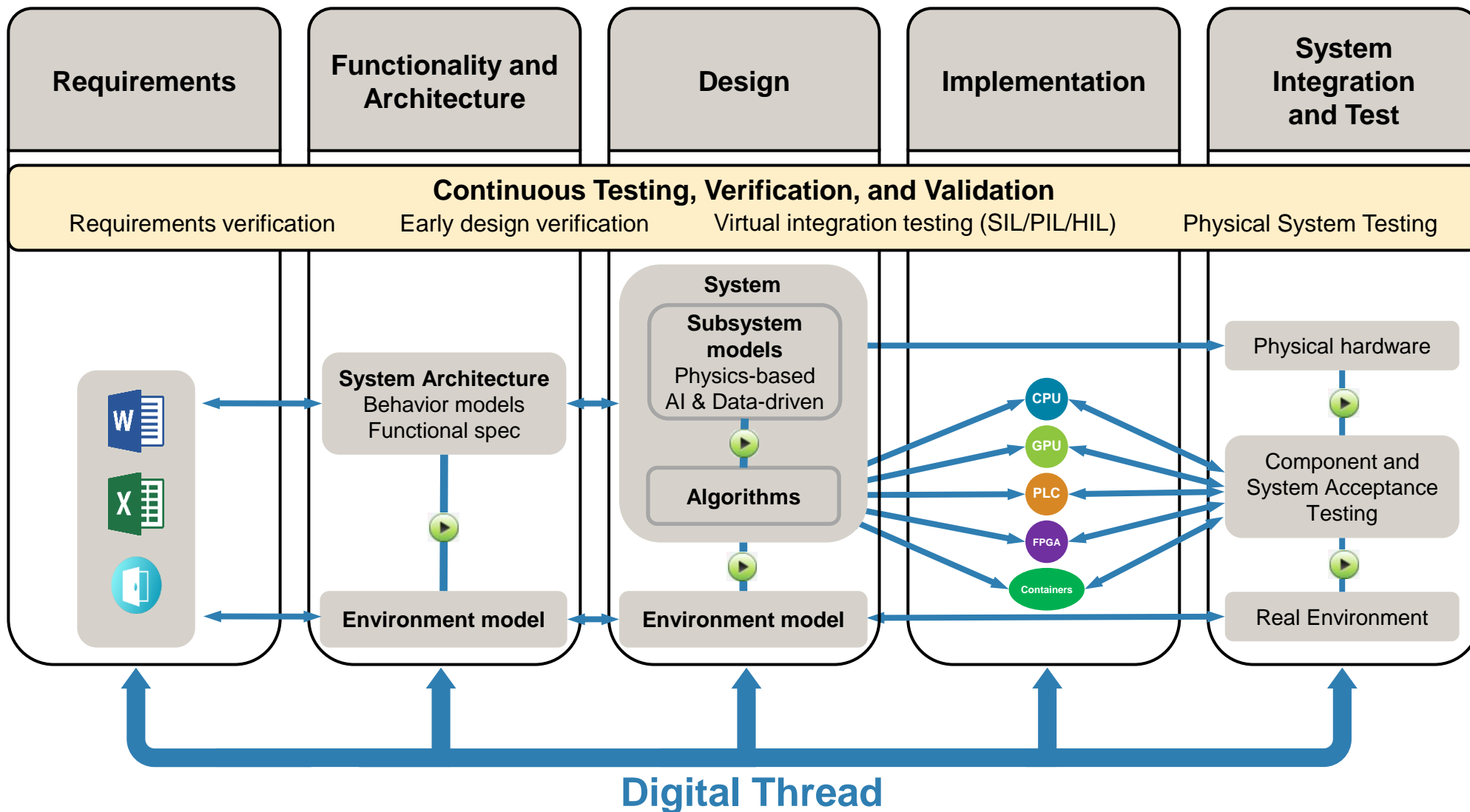


Comparison of simulation results between a Simscape vehicle model with a multibody suspension and a neural network trained using synthetic data from the Simscape model.

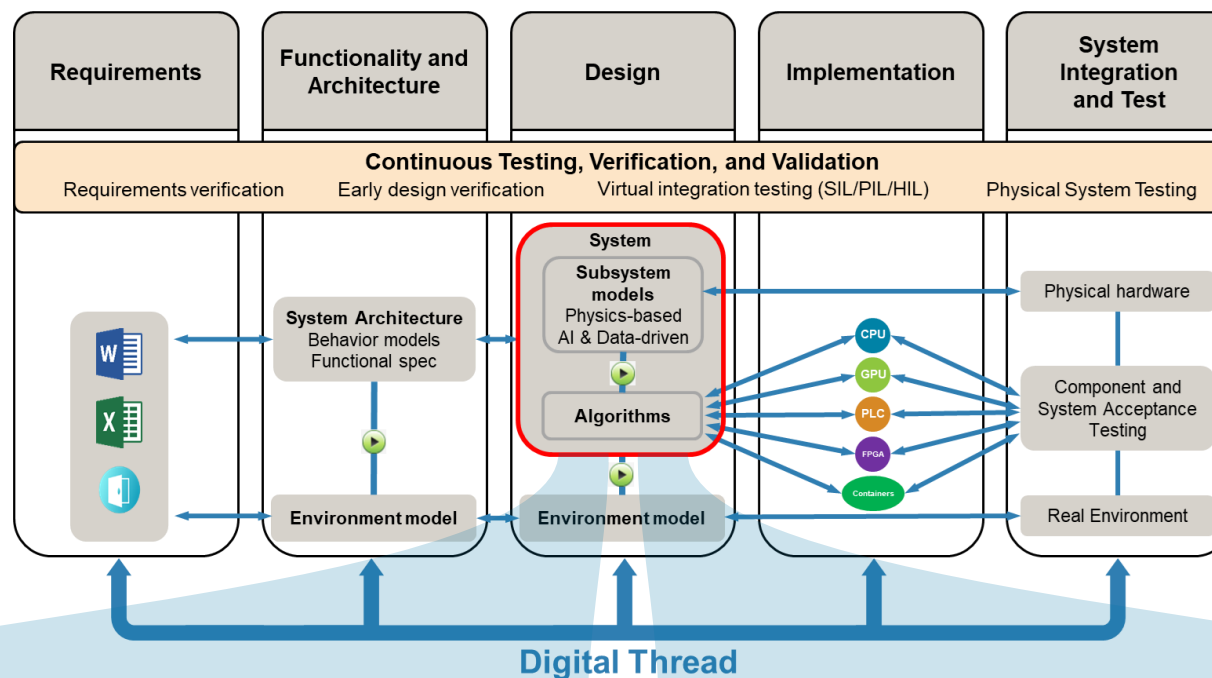
“As expected, performance was satisfactory, proving that all three networks could generalize well on new data. While the Keras neural network managed to fit the first peak almost perfectly, it was not as accurate with the remaining plot. The hybrid neural network was an improvement and generally fit the real curve. Finally, as expected, the MATLAB neural network had the best performance, and its prediction was extremely accurate.”

- Georgios Papageorgiou, TWT GmbH Science & Innovation

Model-Based Design



Integrate AI models into MBD for system-level simulation and code generation



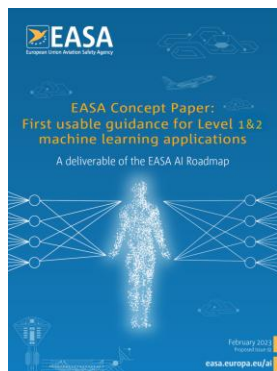
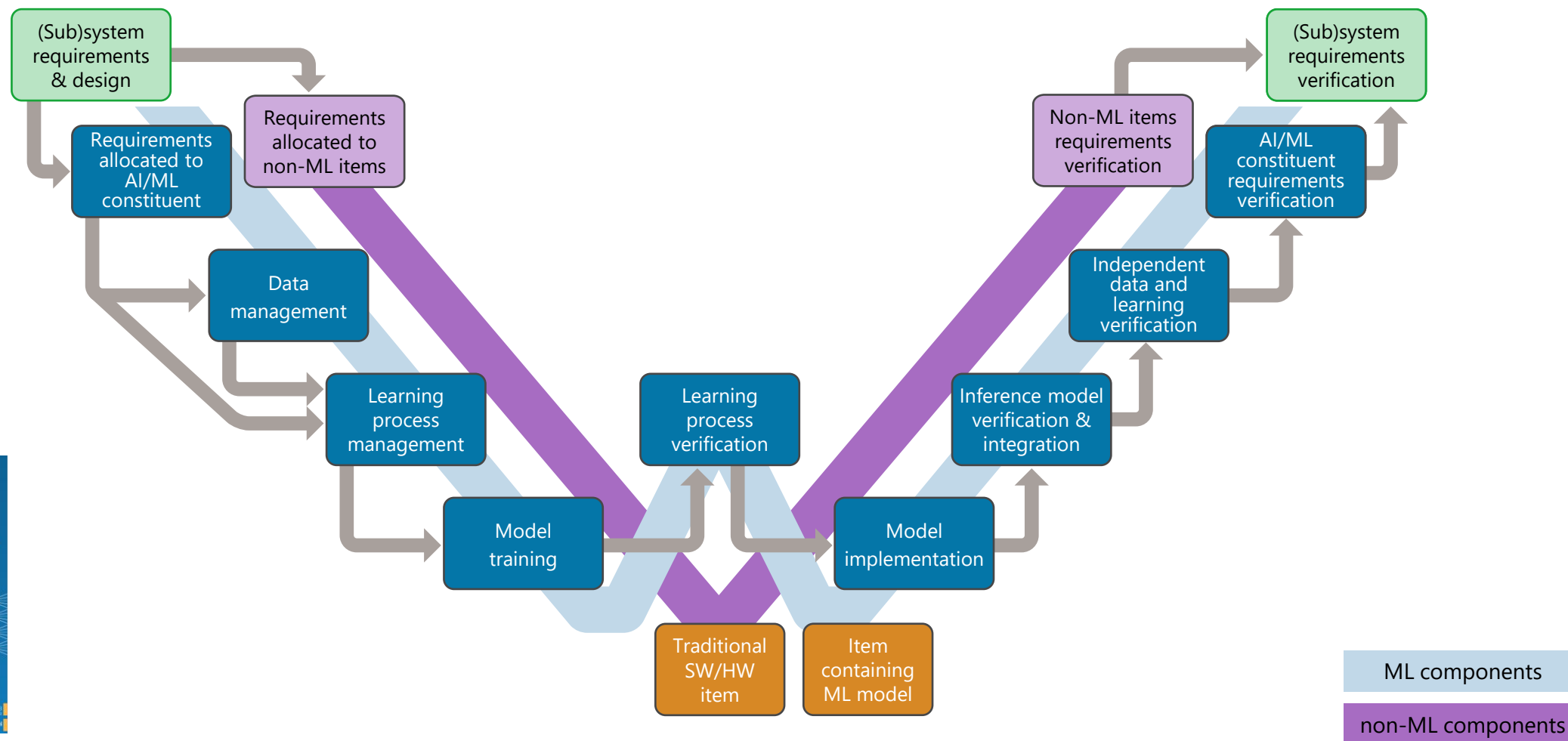
AI for component modeling

- HIL testing and system-level simulation for high-fidelity models
- Modeling component dynamics from data when first-principles models cannot be obtained

AI for algorithm development

- Virtual sensor modeling
- Sensor fusion
- Object detection

W-shaped development process adapting the classical V-shaped cycle to ML applications



Source: [EASA Concept Paper Proposed ISSUE 02 'First usable guidance for Level 1&2 machine learning applications'](#)

Credit: EASA, Daedalean

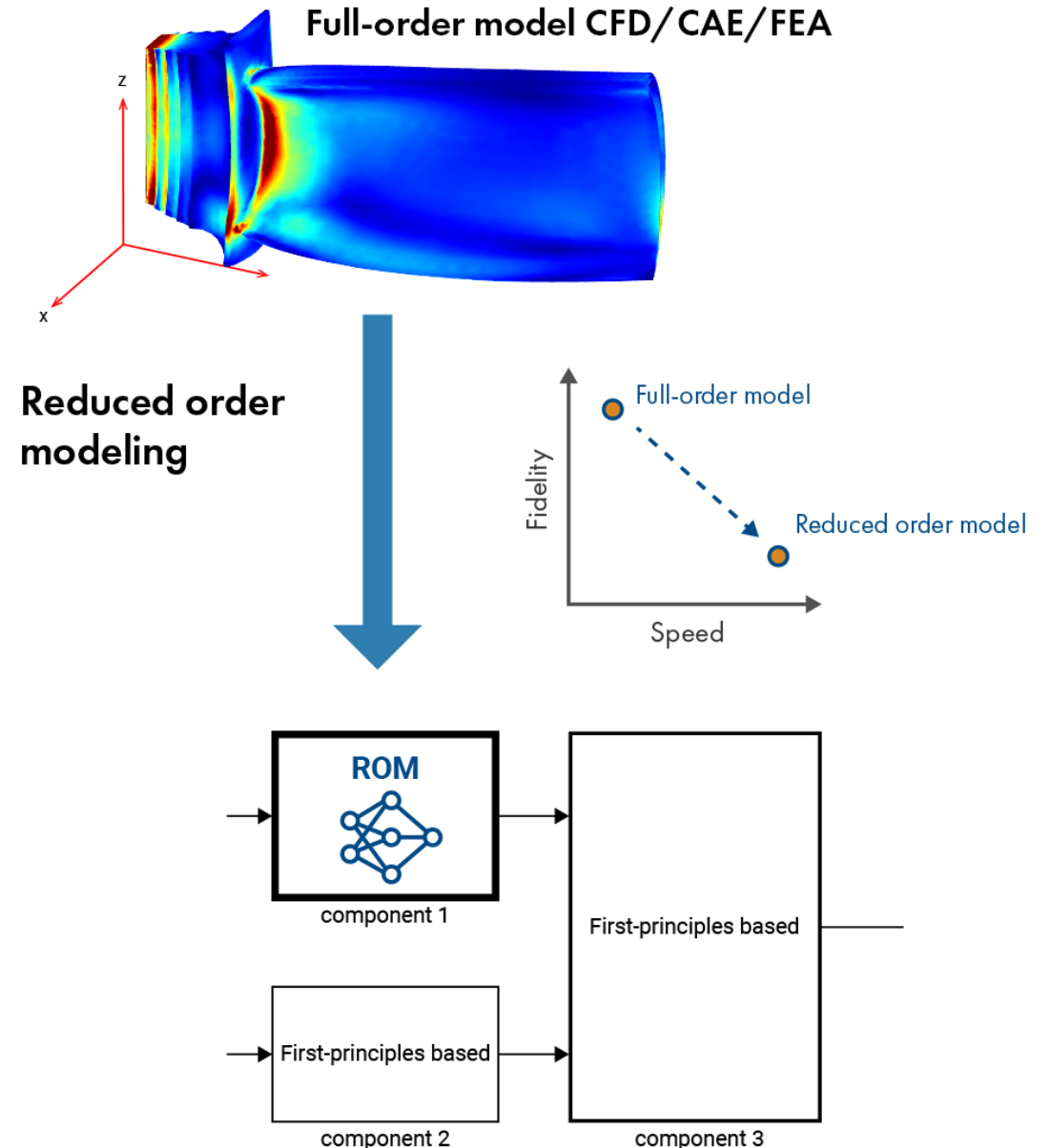
Reduced Order Modeling

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
- Enable desktop simulations for orders-of-magnitude longer timescales

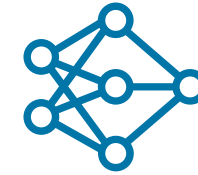


Reduced Order Modeling

How

AI-Based
Data-driven

Inputs
Ambient Temperature
Ambient Pressure
Cooling Temperature



AI model



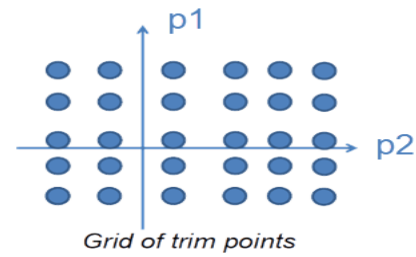
Outputs
Max Displacement

focus today

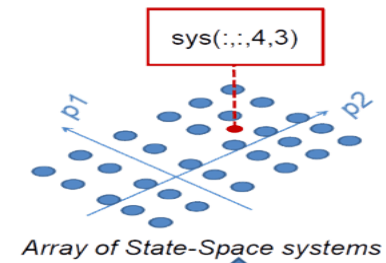
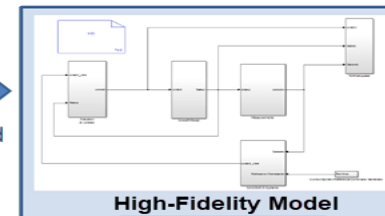
Reduced order
model

Linearization

Model-based



Loop through the grid
of trim points



Identify local model at
each trim point

FEA
Software

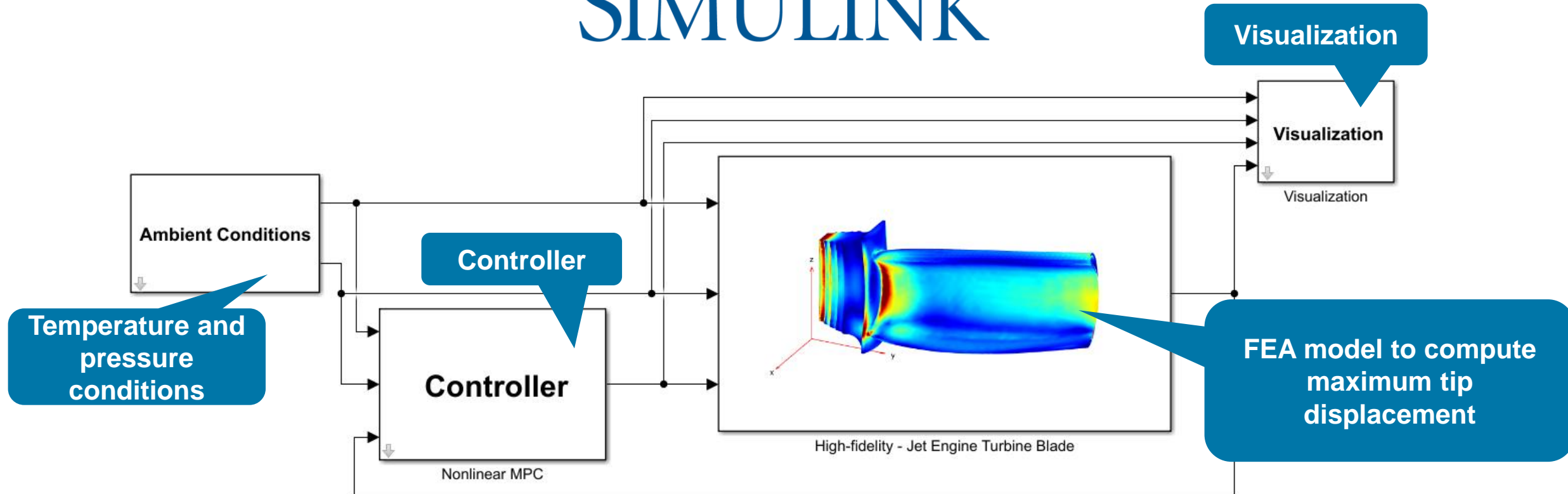


Simulink
Simscape Multibody
Control System Toolbox

Example overview

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

SIMULINK®

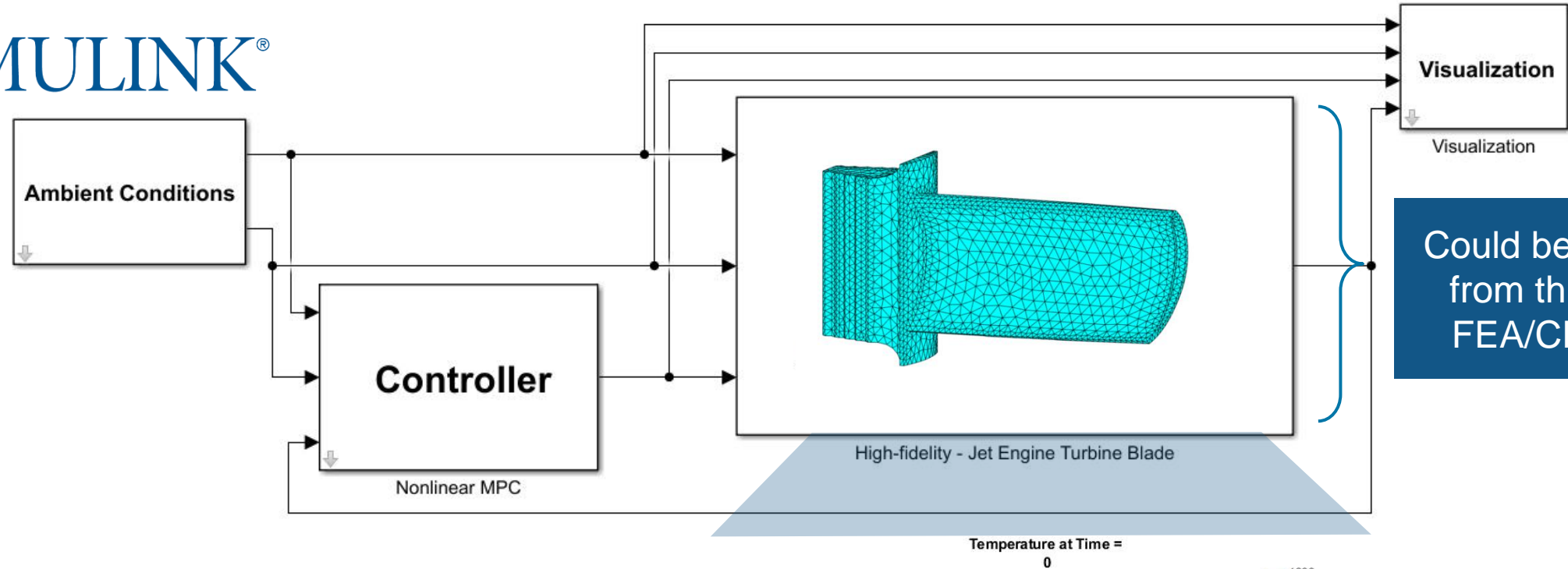


Closed-loop temperature control

Example overview

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

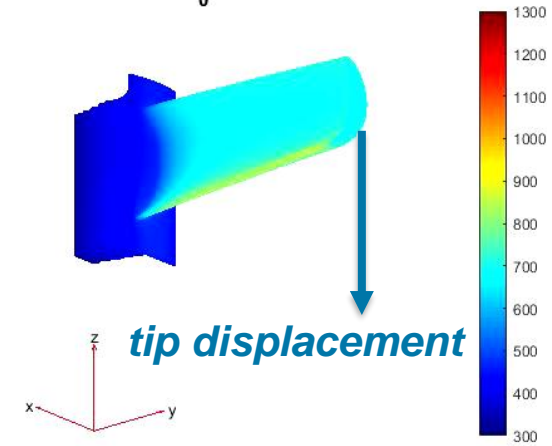
SIMULINK®



~30 seconds per time step for solving FEA models



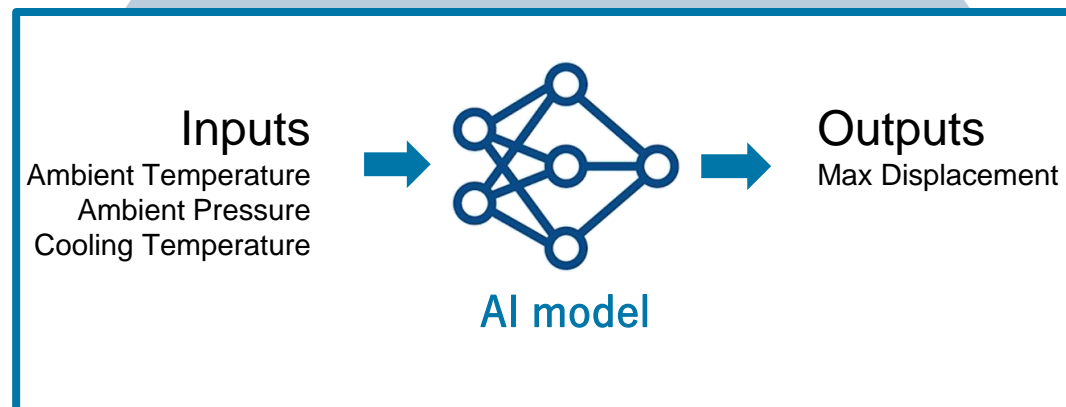
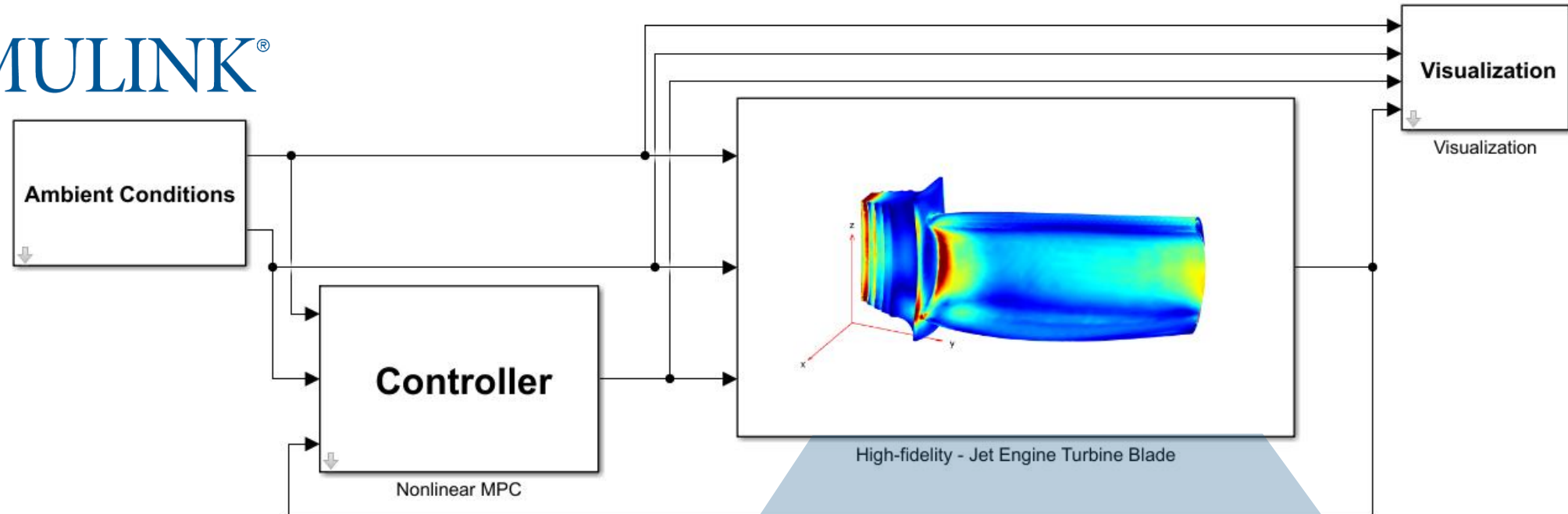
Not suitable for control design and HIL testing



Example overview

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

SIMULINK®



Introducing Simulink Add-On for Reduced Order Modeling

Create AI-based reduced order models (ROM)

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

Reduced Order Modeler App

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

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

- Set up the design of experiments and generate input-output data from full-order, high-fidelity subsystems
- Train and compare AI-based reduced order models using preconfigured templates
- Export AI-based surrogate models to Simulink or outside of Simulink through FMUs
- Export reduced order models as Full-Order Models (FOMs) or Reduced Order Models (ROMs) to the Simulink Compiler

Reduced Order Modeling

Full-order model CFD/CAE/FEA

ROM component 1

First-principles based component 2

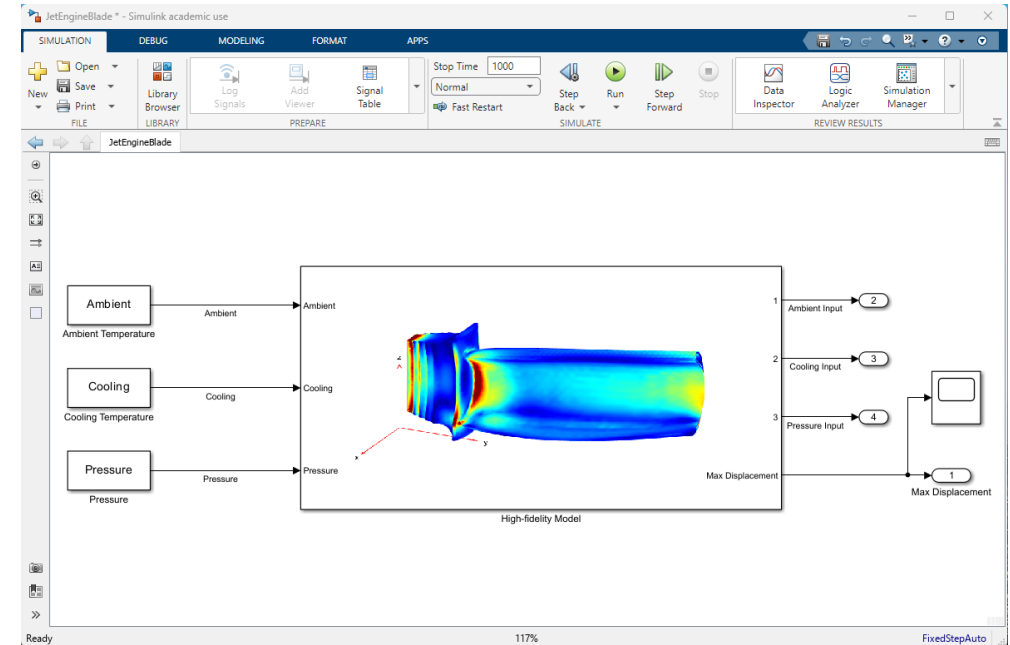
First-principles based component 3

Fidelity vs Speed graph showing Full-order model and Reduced order model.

Generate data for training



Physical system



Simulink/Simscape

Data Preparation

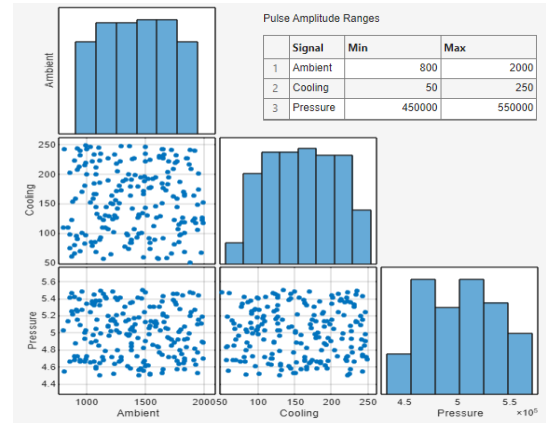
AI Modeling

Simulation & Test

Deployment

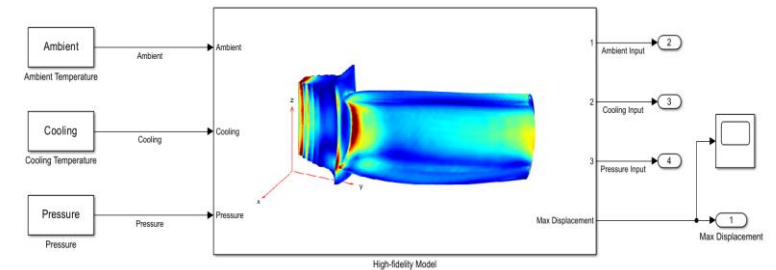
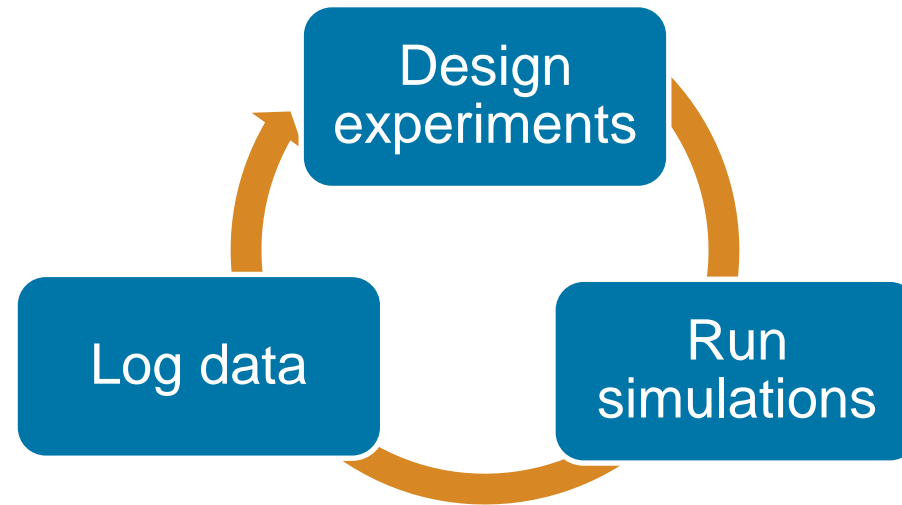
Synthetic Data Generation

Design of Experiments



Input features
 Ambient Temperature
 Ambient Pressure
 Cooling Temperature

Response
 Max Displacement



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
fx >>

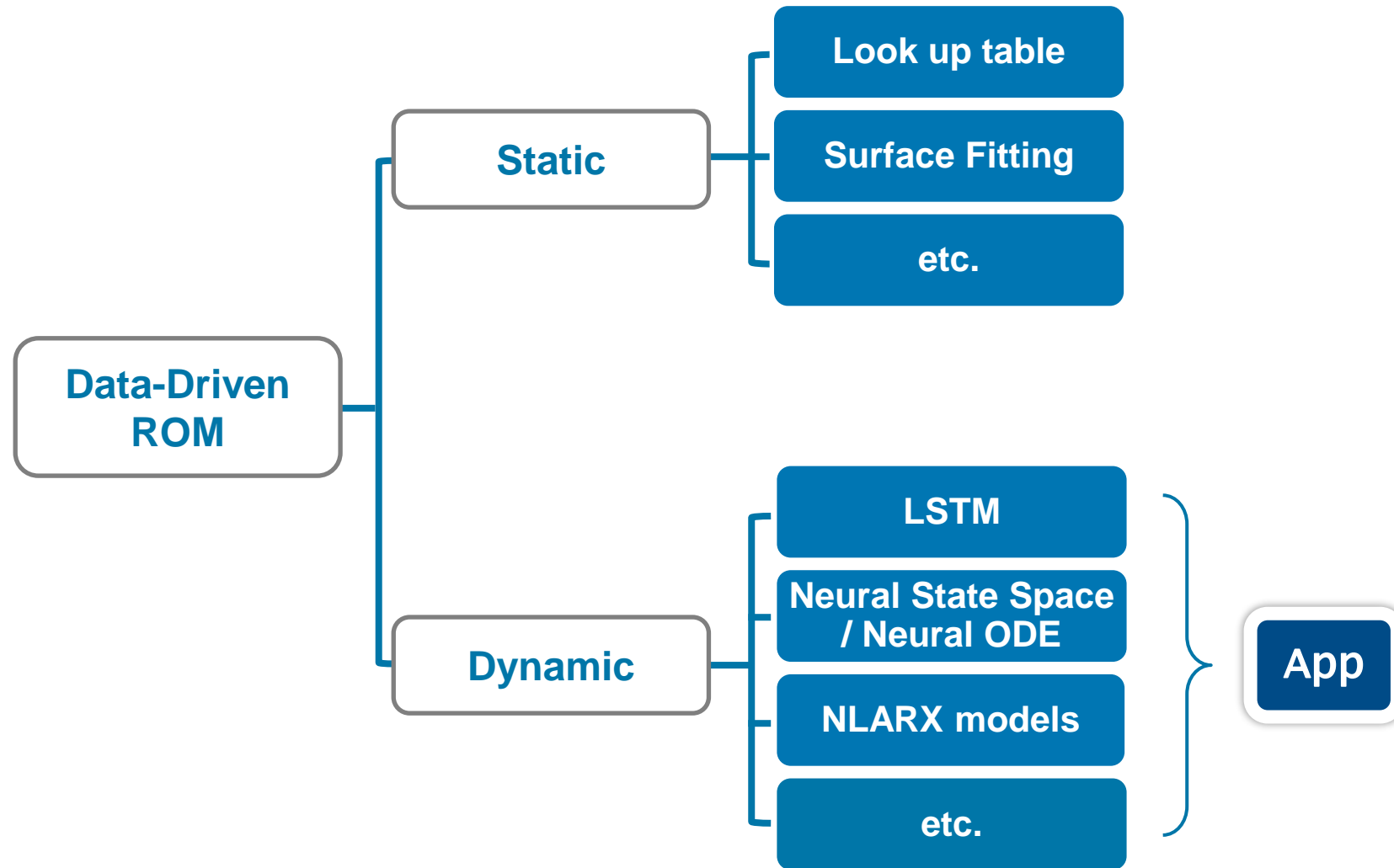
Data Preparation

AI Modeling

Simulation & Test

Deployment

Data-driven ROM

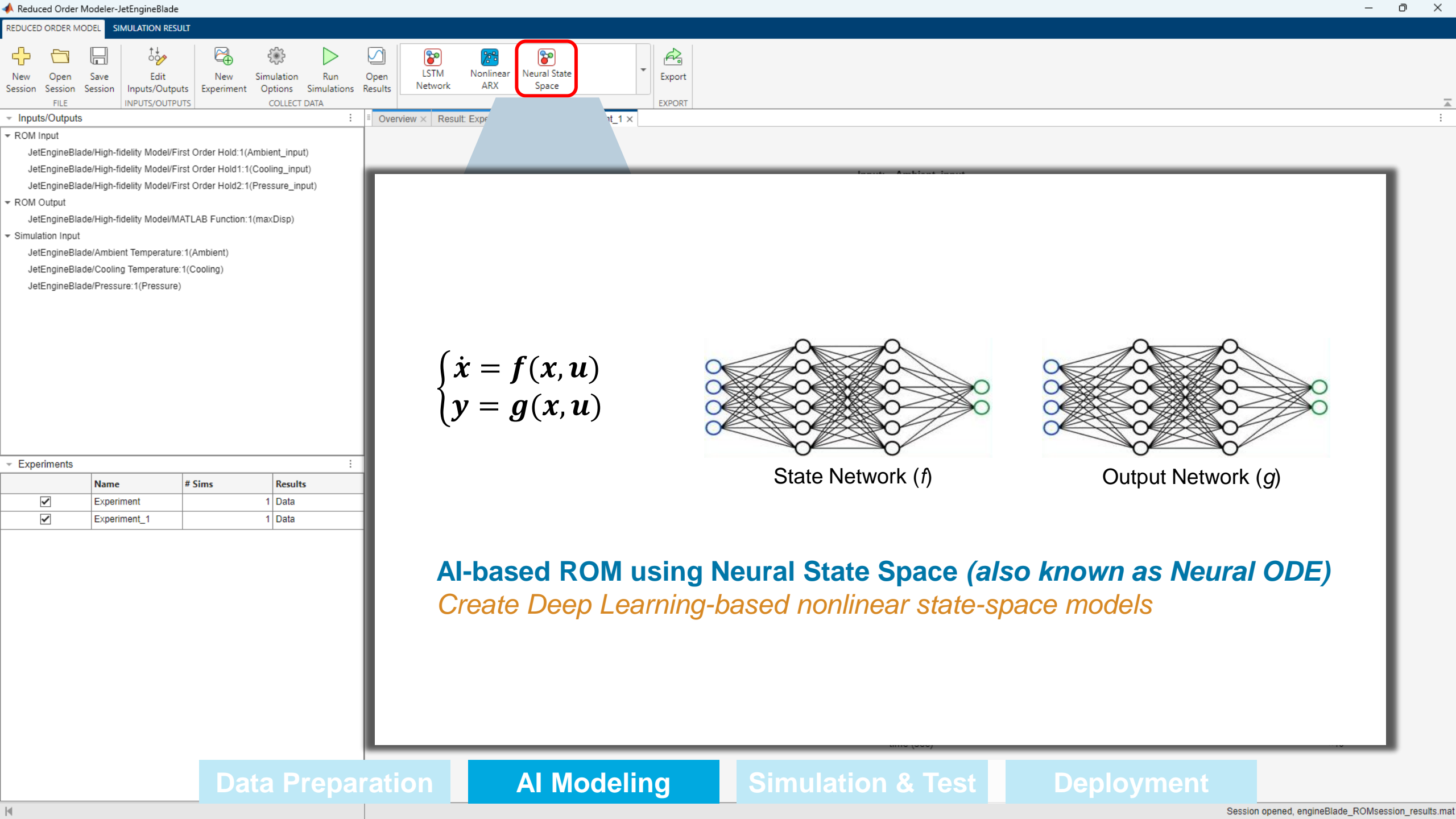


Data Preparation

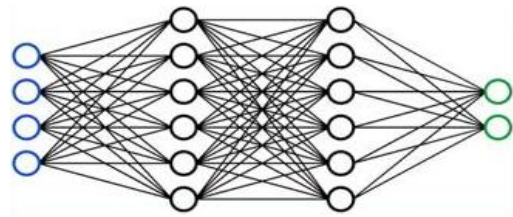
AI Modeling

Simulation & Test

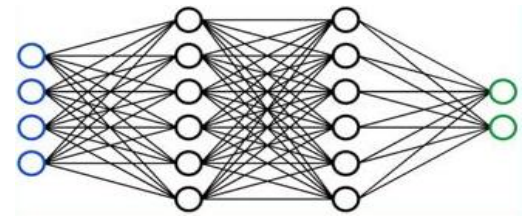
Deployment



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



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

Data Preparation

AI Modeling

Simulation & Test

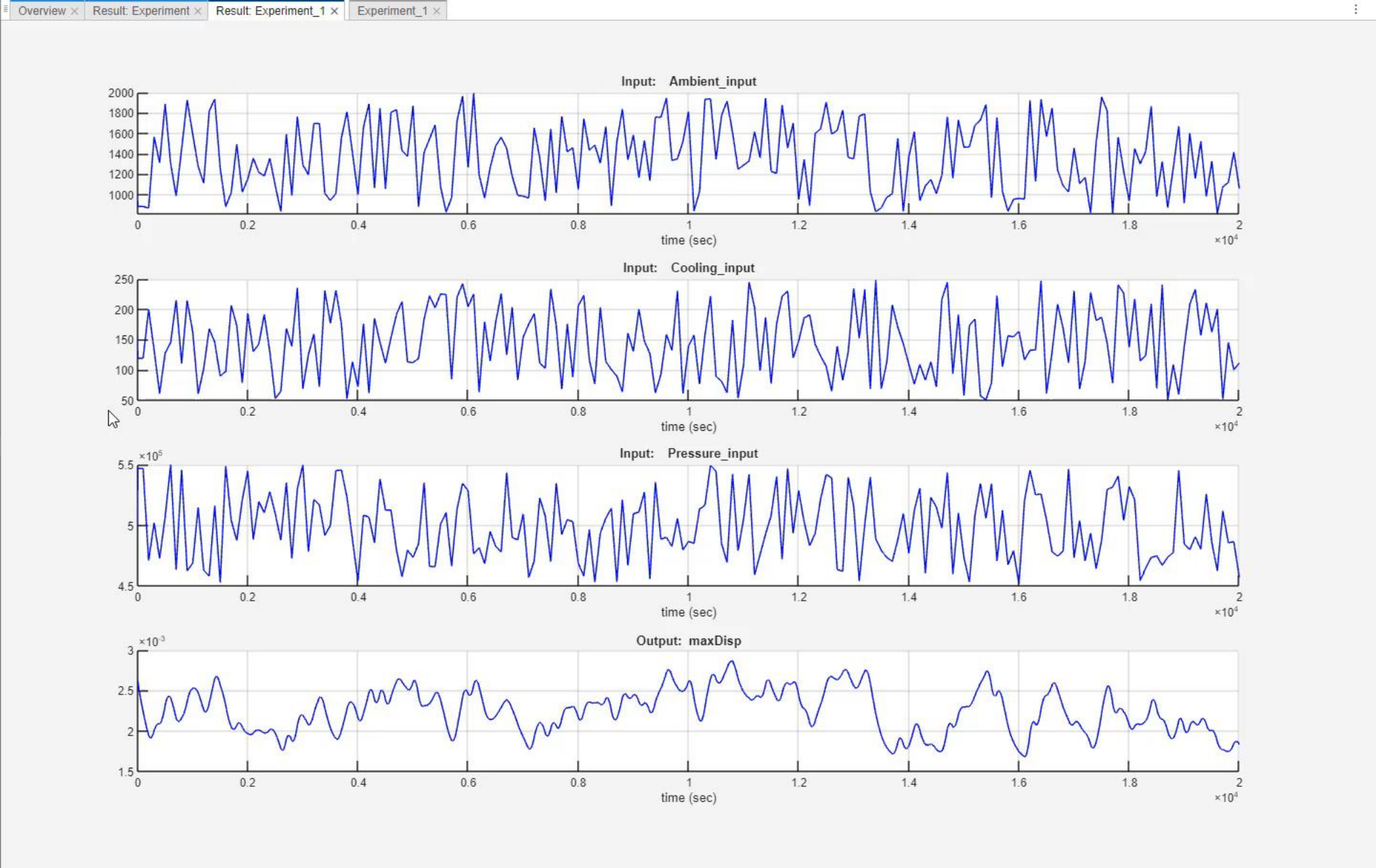
Deployment

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

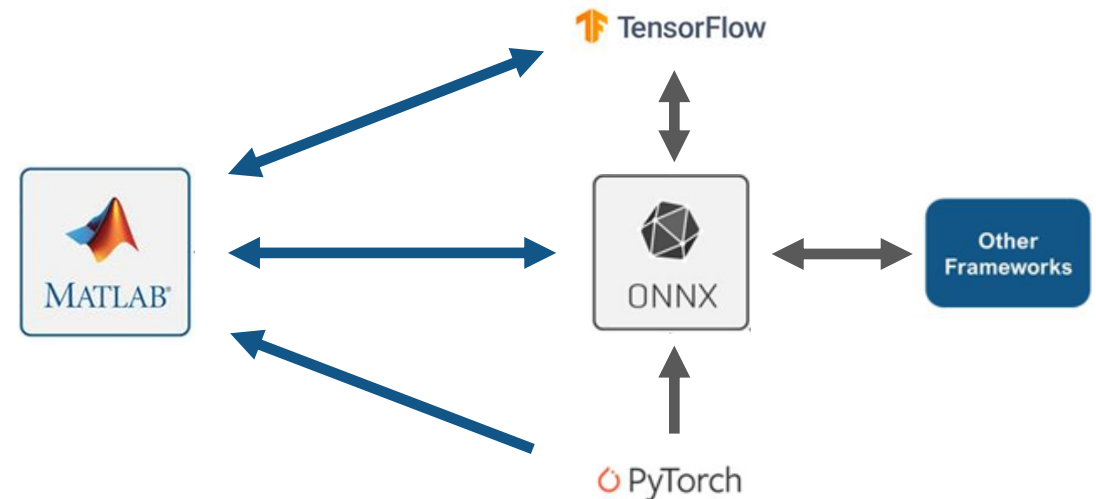
	Name	# Sims	Results
<input checked="" type="checkbox"/>	Experiment		1 Data
<input checked="" type="checkbox"/>	Experiment_1		1 Data



MATLAB interoperates with other frameworks

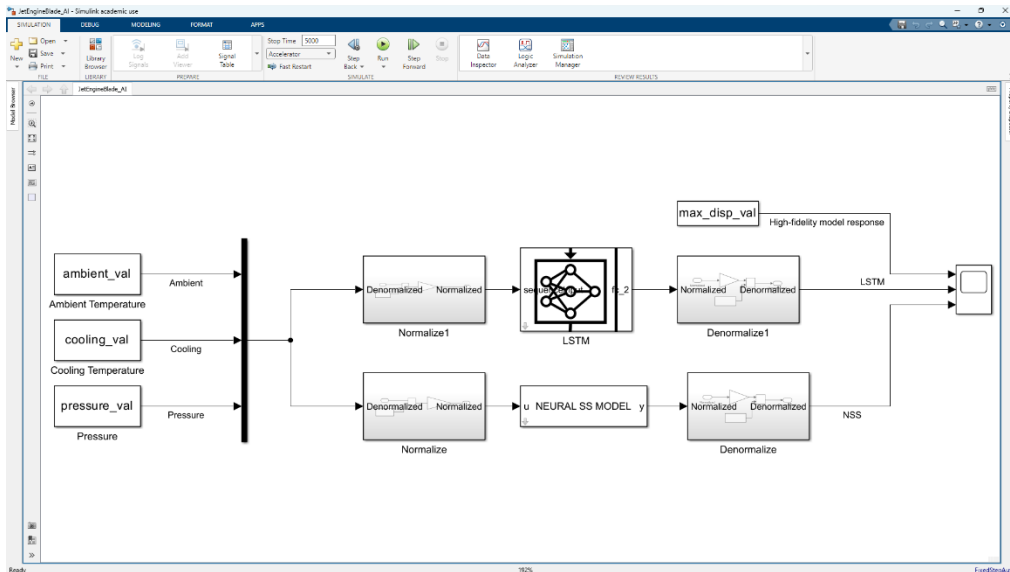
Framework interoperability bridges the gap between data science, engineering and production

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

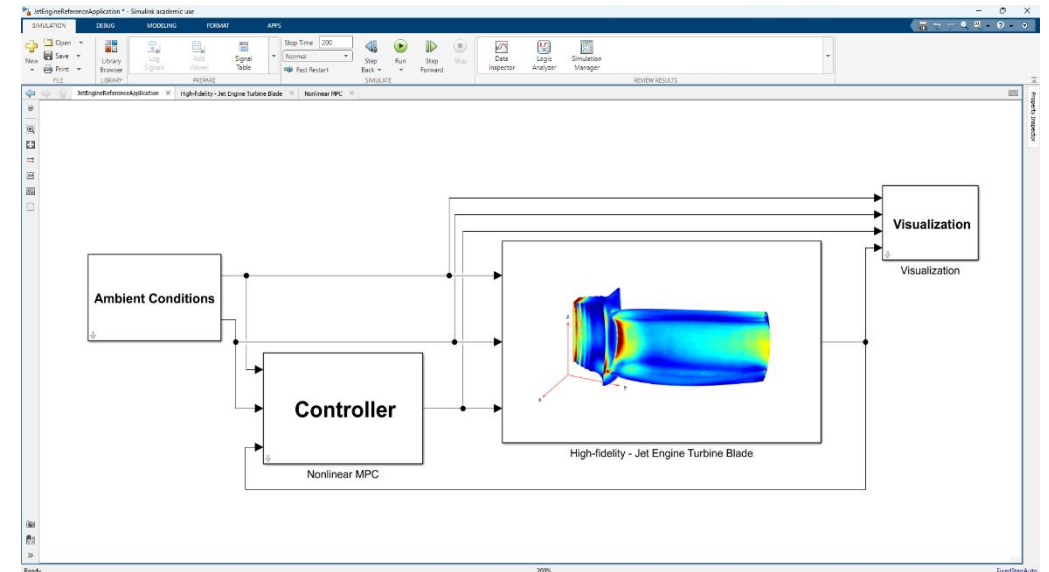


Integrate your AI model for system-level simulation and test

Integration of trained AI model into Simulink



System-level simulation



Deep Learning Toolbox Verification Library

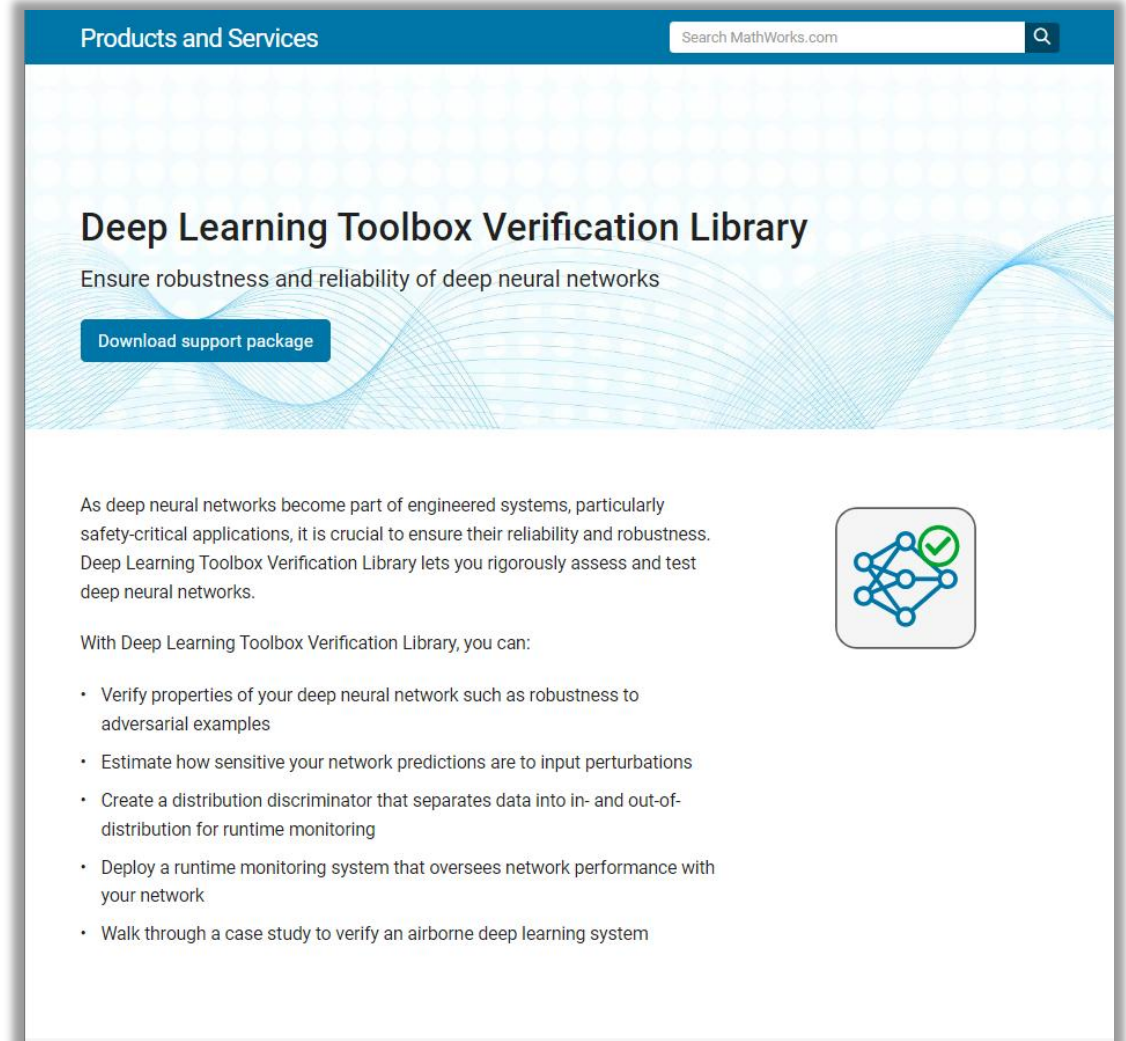
Ensure robustness and reliability of deep neural networks

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. The page has a blue header with the text "Products and Services" and a search bar containing "Search MathWorks.com". The main content area features the title "Deep Learning Toolbox Verification Library" and the subtitle "Ensure robustness and reliability of deep neural networks". A blue button labeled "Download support package" is positioned below the subtitle. The page includes a paragraph of introductory text, a list of capabilities, and a small icon of a neural network with a green checkmark.

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



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 `fu >>`.

Data Preparation

AI Modeling

Simulation & Test

Deployment

Integration of trained AI models into Simulink

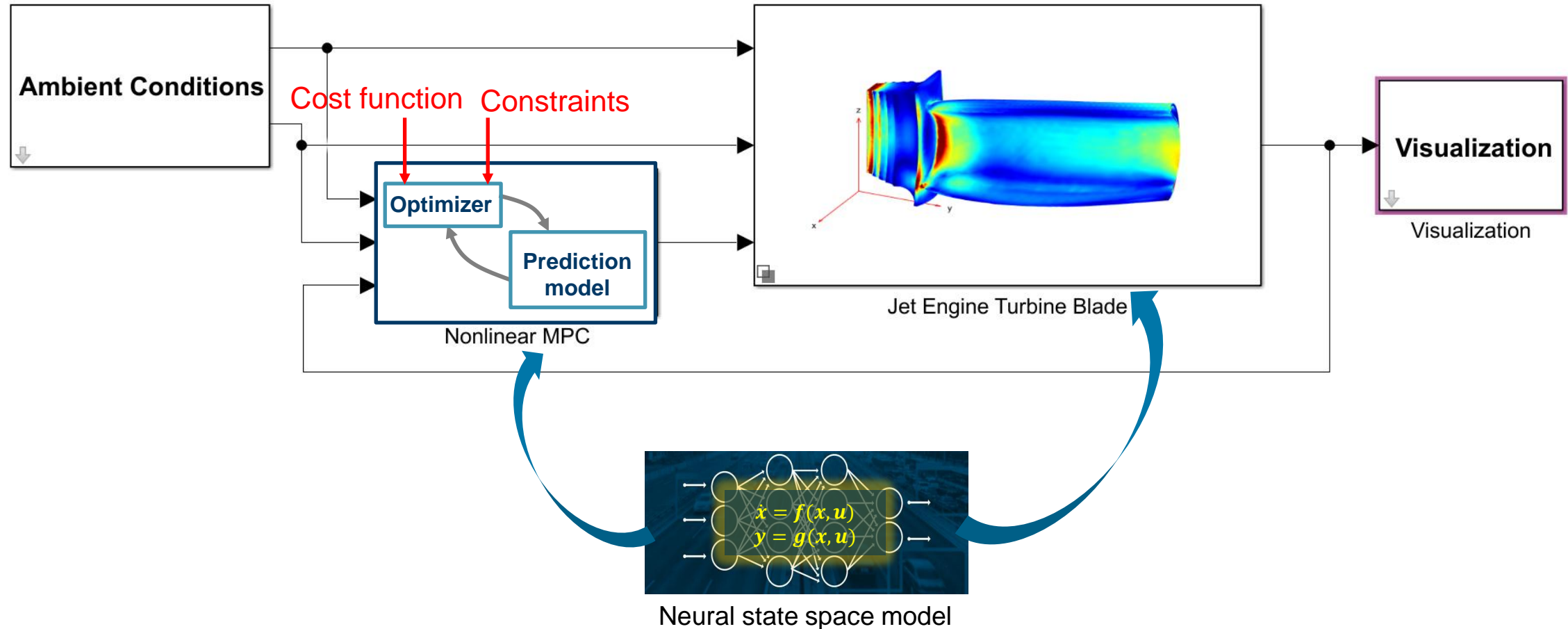
Simulink Profiler

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

Control Design with Model Predictive Controller

SIMULINK®



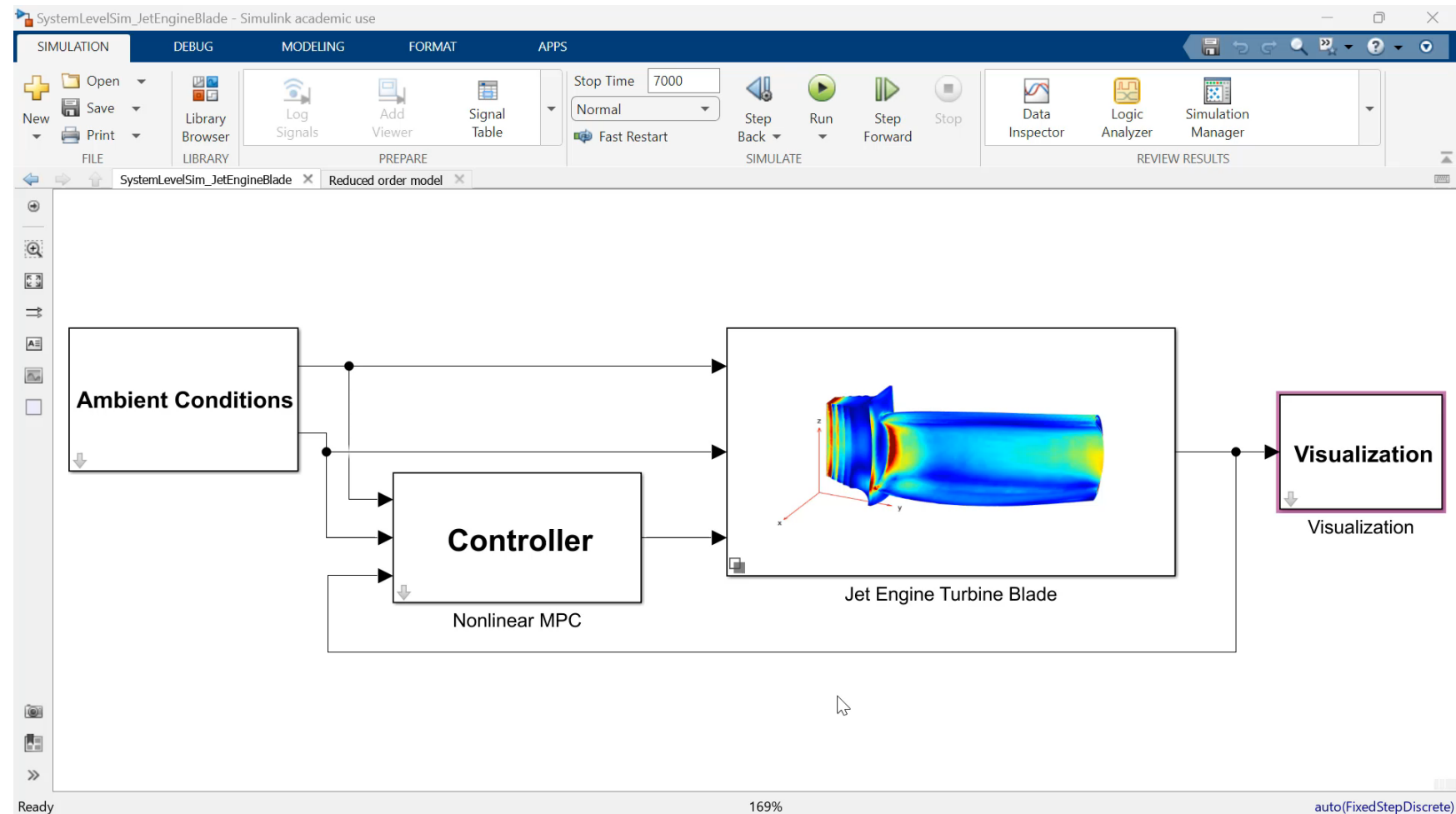
Data Preparation

AI Modeling

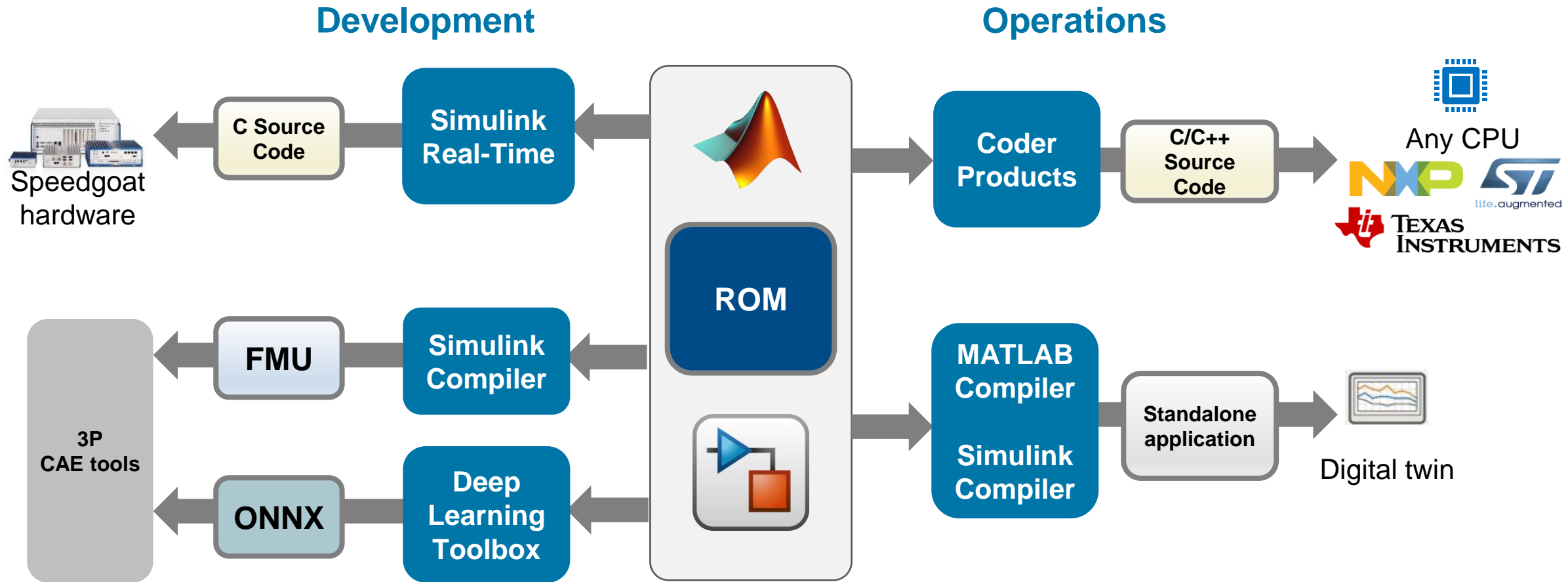
Simulation & Test

Deployment

System-level simulation

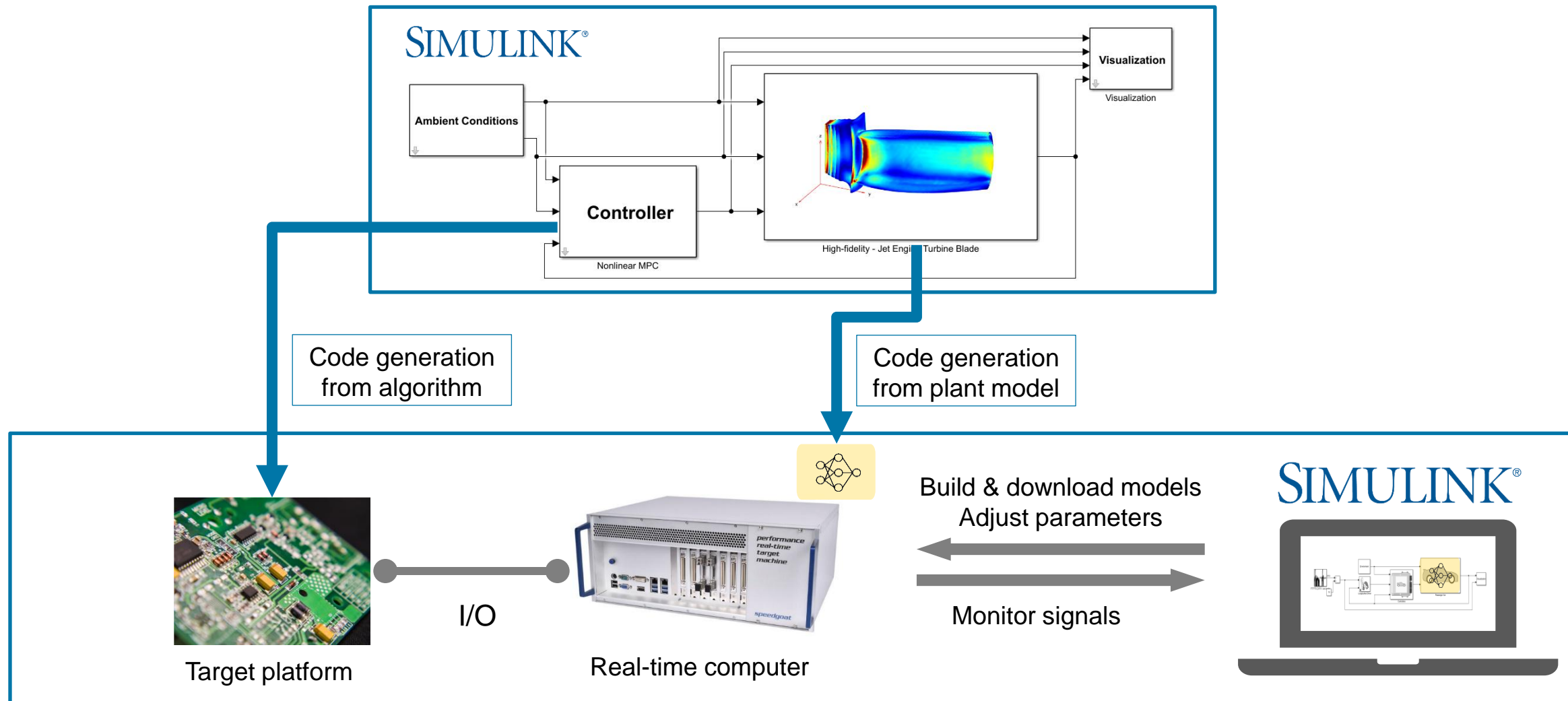


Use ROMs outside of Simulink, for development and operation stages



Hardware-in-the-loop simulation

System-level integration and test



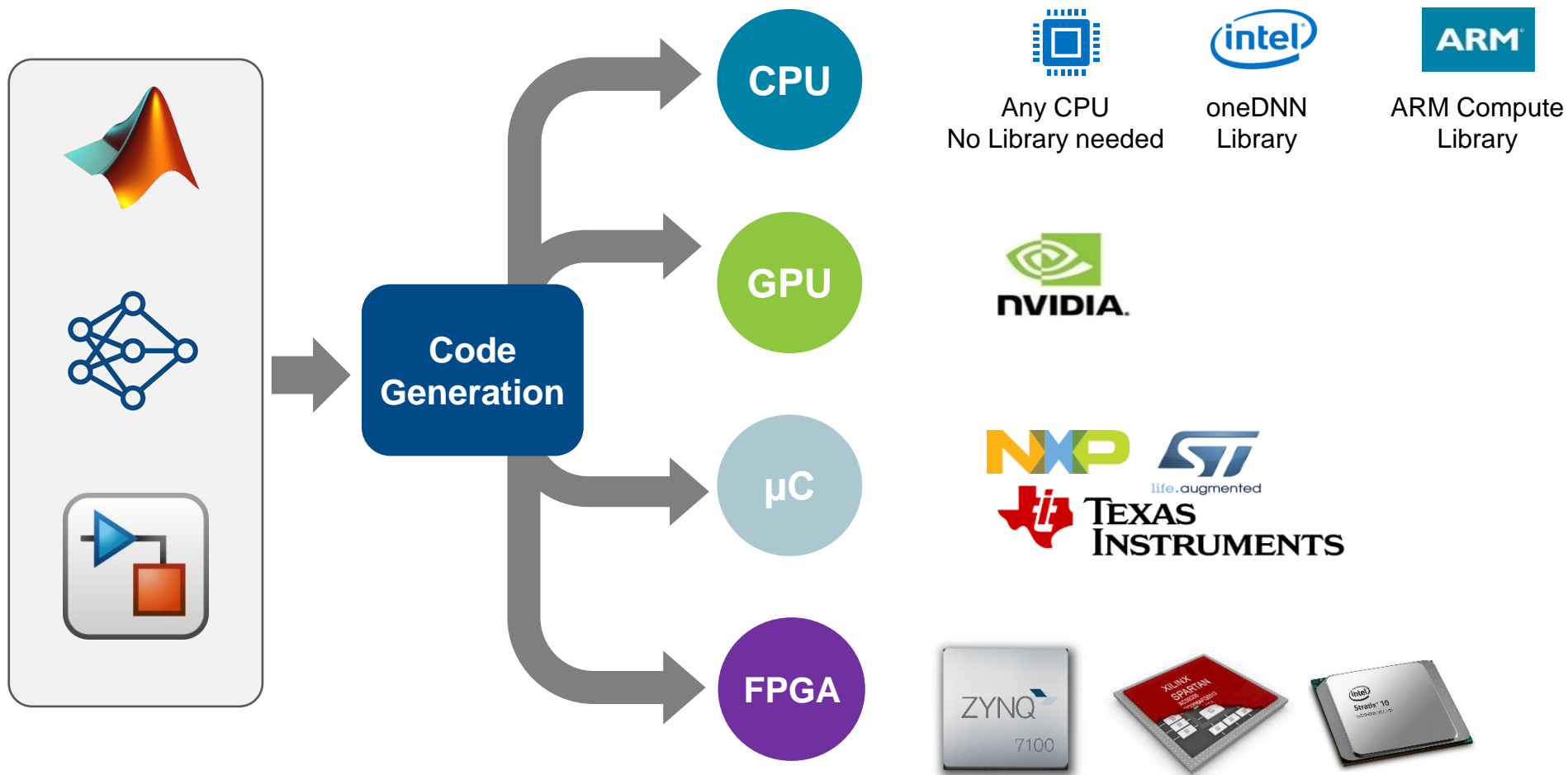
Data Preparation

AI Modeling

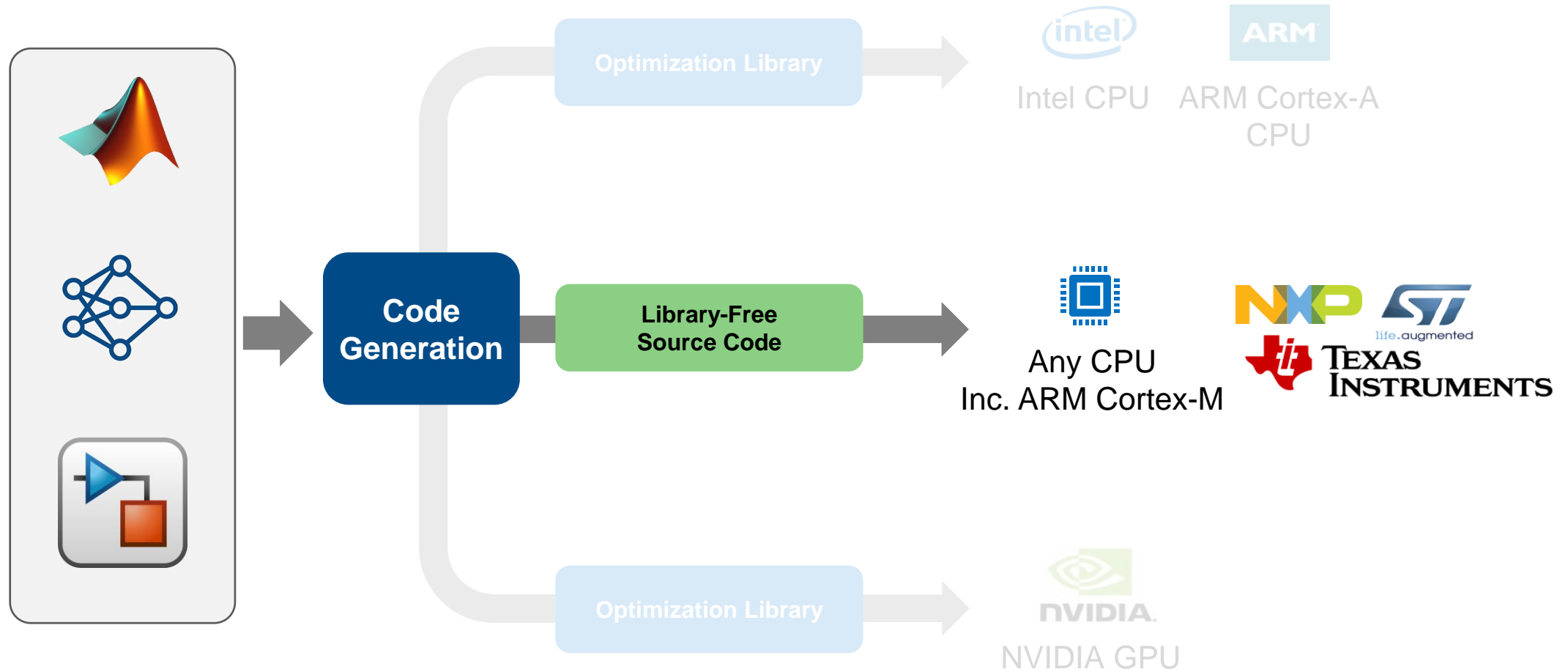
Simulation & Test

Deployment

Deploy to target with zero coding errors



Generate Library-Free C/C++ Code for Deep Learning Networks



Data Preparation

AI Modeling

Simulation & Test


Deployment

AI workflow for engineered systems

Data Preparation

 Data pre-processing


 Feature engineering

 Simulation-generated data

AI Modeling


 Model design and tuning



 Python interoperability

 Explain models and predictions

Simulation & Test

 Integration with complex systems

 System simulation with AI models

—  System verification and validation
— 

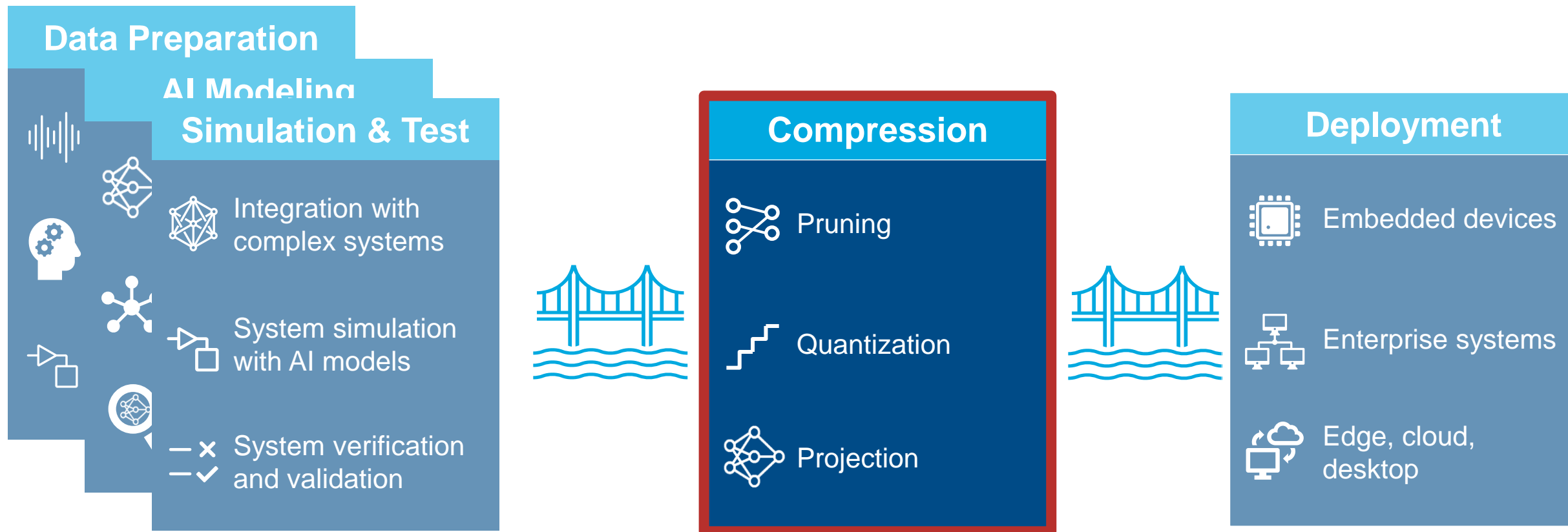
Deployment

 Embedded devices

 Enterprise systems

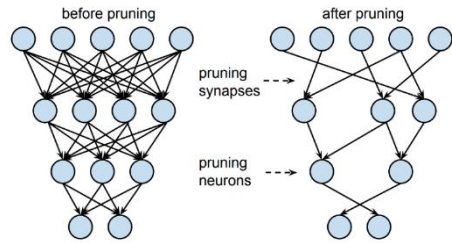
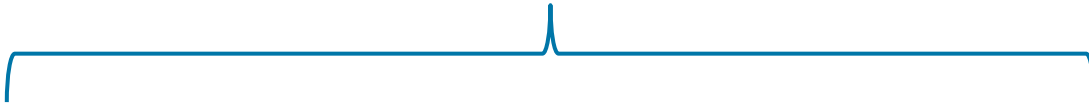
 Edge, cloud, desktop

Model compression bridges the gap between design & deployment

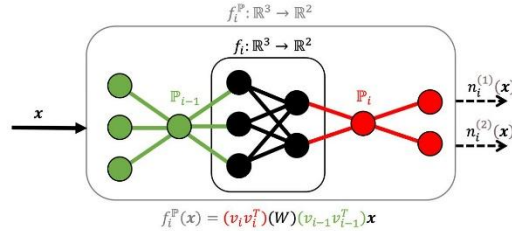


Model Compression

Structural Compression

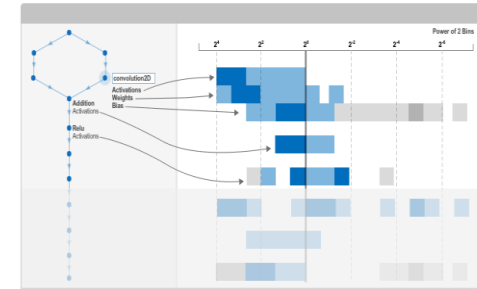


Pruning
convolutional neural networks



Projection of deep neural networks

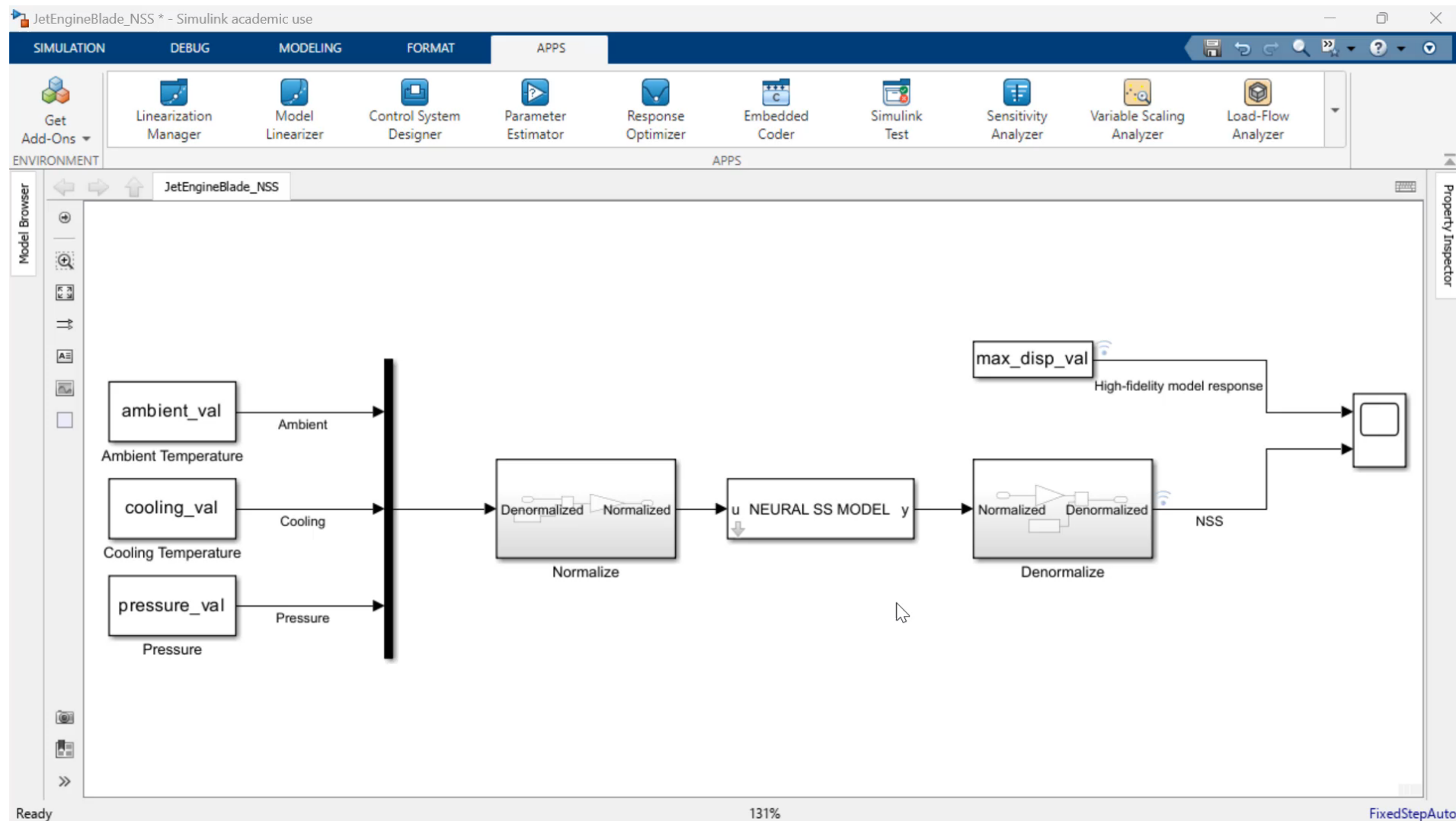
Datatype Compression



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

[Quantization, Projection, and Pruning](#)

Generate Library-Free C Code for Deep Learning Networks



Data Preparation

AI Modeling

Simulation & Test

Deployment

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 ●
----------	--------	---------

Key Takeaways

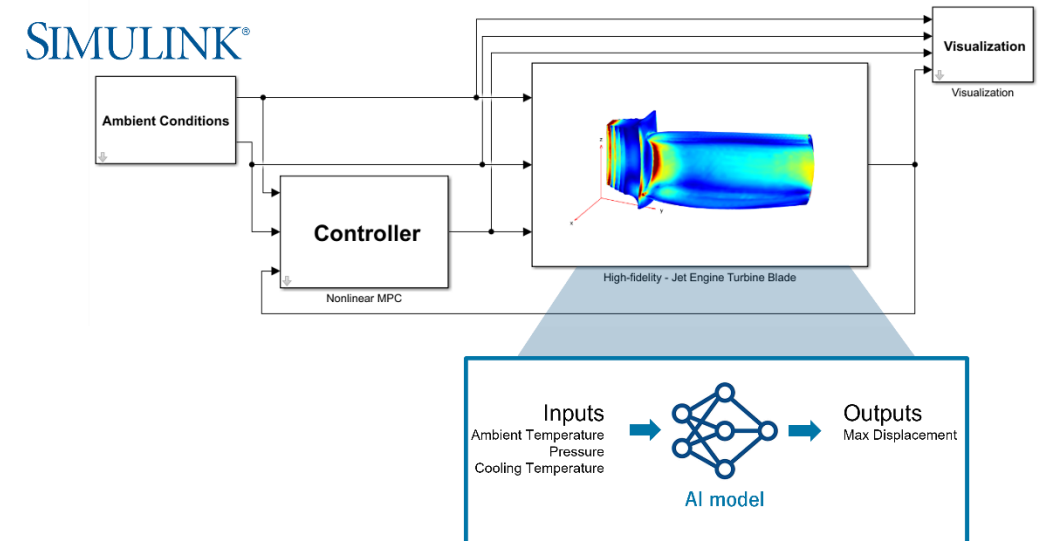
Enable

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 FEA model that computes tip displacement** of a jet engine blade
- **Integrate trained AI model into Simulink** for control design and system-level simulation
- **Generate C code and perform HIL tests**



MATLAB EXPO



© 2024 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.

