MATLAB EXPO

Reduced Order Modeling (ROM) with AI: Accelerating Simulink Analysis and Design

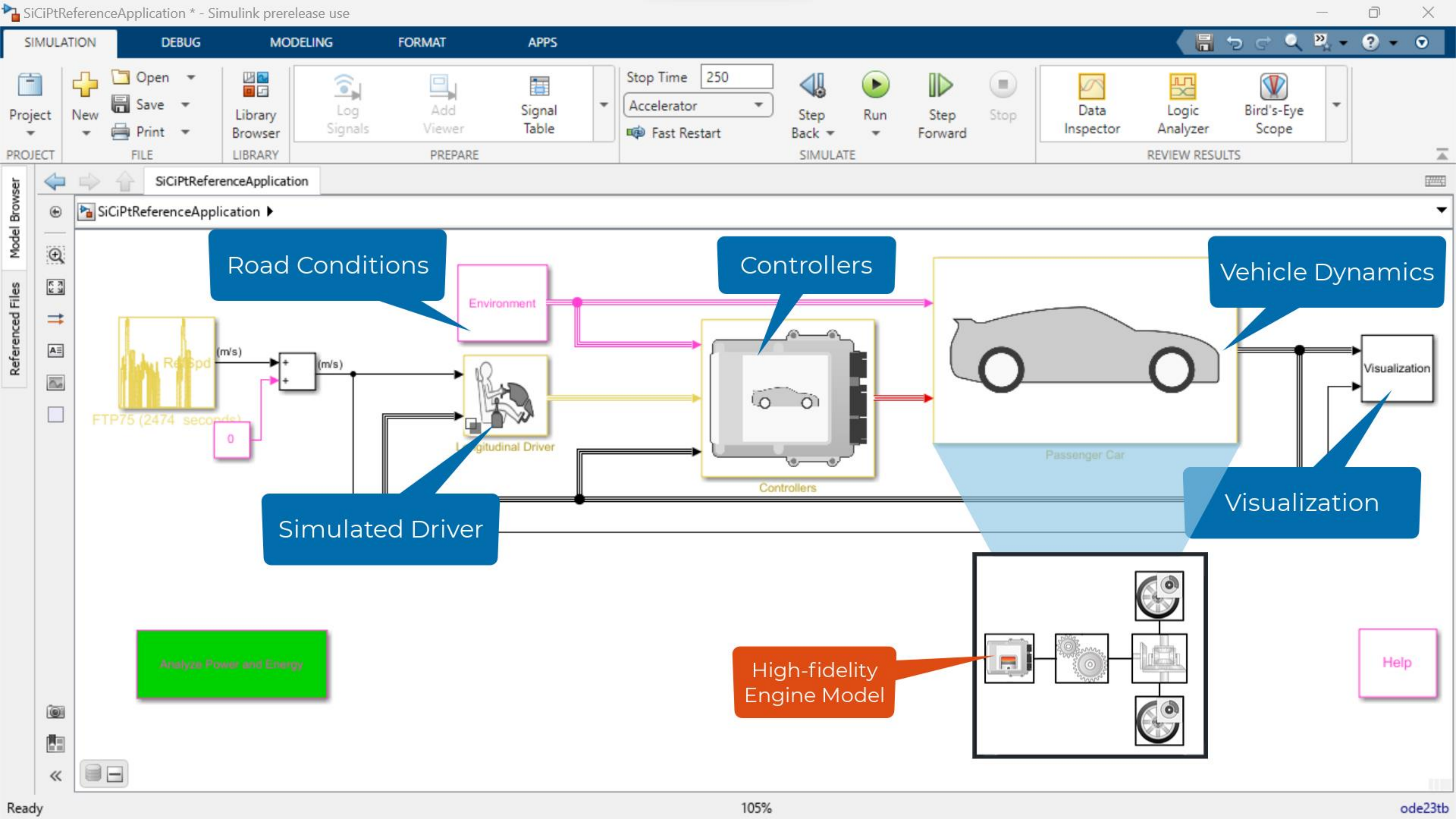
Kishen Mahadevan, MathWorks

(He/His)

Terri Xiao, MathWorks

(She/Her)

MathWorks


SiCiPtReferenceApplication * - Simulink prerelease use
SIMULATION
DEBUG
MODELING
FORMAT
APPS
Project
New
Open
Save
Print
Library Browser
Log Signals
Add Viewer
Signal Table
Stop Time
250
Accelerator
Fast Restart
Step Back
Run
Step Forward
Stop
Data Inspector
Logic Analyzer
Bird's-Eye Scope
PROJECT
FILE
LIBRARY
PREPARE
SIMULATE
REVIEW RESULTS
Model Browser
Referenced Files
SiCiPtReferenceApplication
Road Conditions
Controllers
Vehicle Dynamics
Environment
(m/s)
FTP75 (2474 seconds)
Longitudinal Driver
Controllers
Passenger Car
Visualization
Simulated Driver
Analyze Power and Energy
High-fidelity Engine Model
Help
Ready
105%
ode23tb

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.

Common Challenges

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.

Reduced Order Modeling

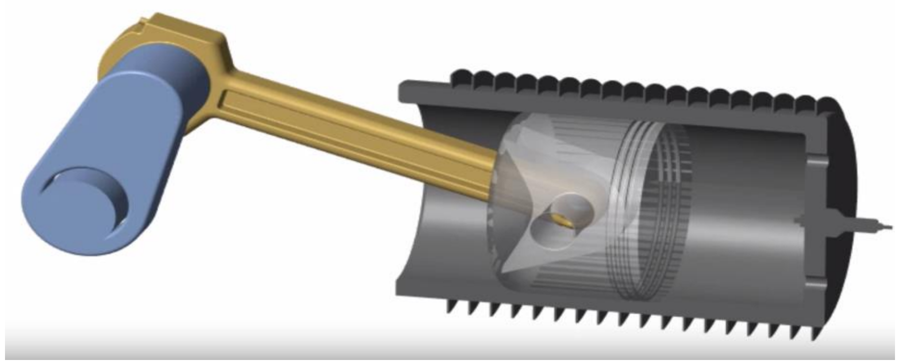
What

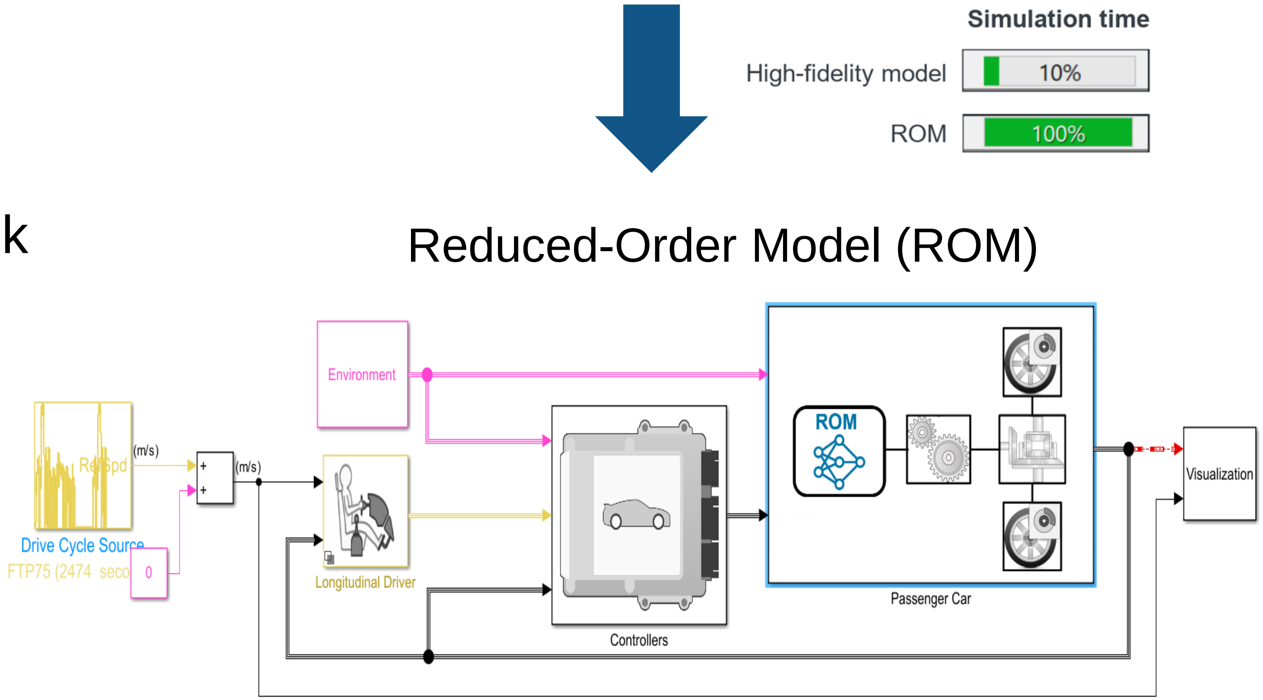
- Techniques to **reduce the computational complexity** of a computer model
- **Provide reduced, but acceptable fidelity**

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





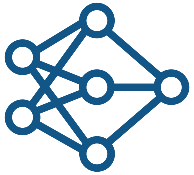
Reduced Order Modeling techniques

How

Reduced order model

AI-Based Data-driven

Inputs
Engine speed (RPM)
Ignition timing
Throttle position
Wastegate valve

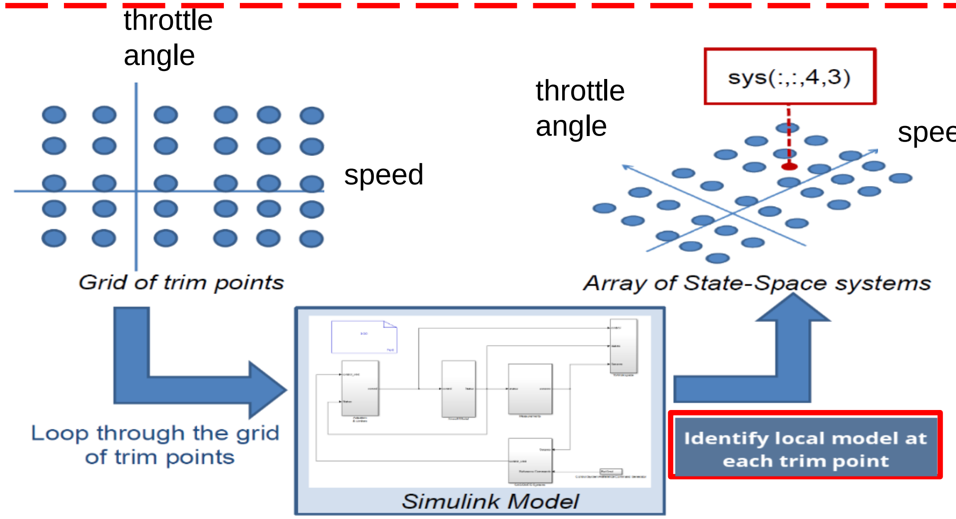


AI model

Outputs
Engine Torque

focus today

Linearization

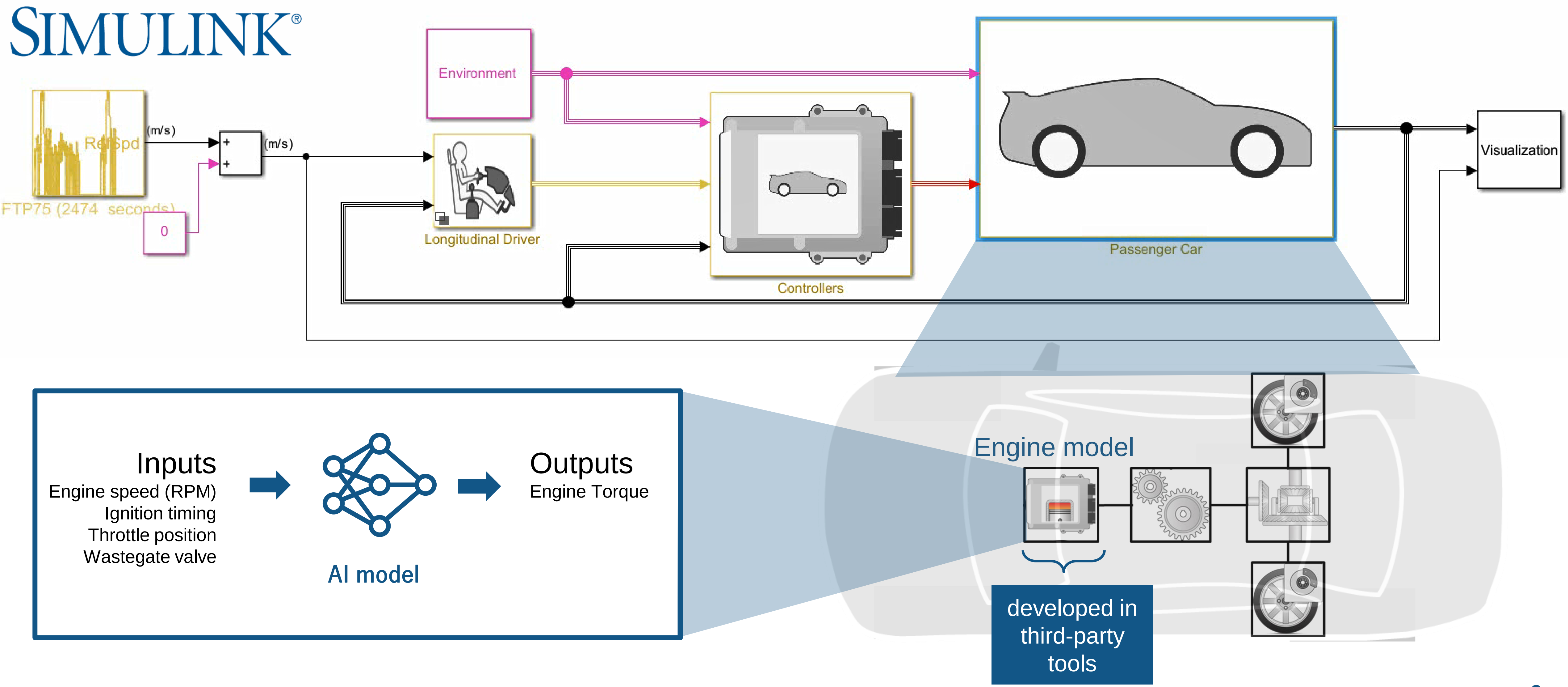


Model-based

FEA Software → **Simulink Simscape Multibody Control System Toolbox**

Example overview

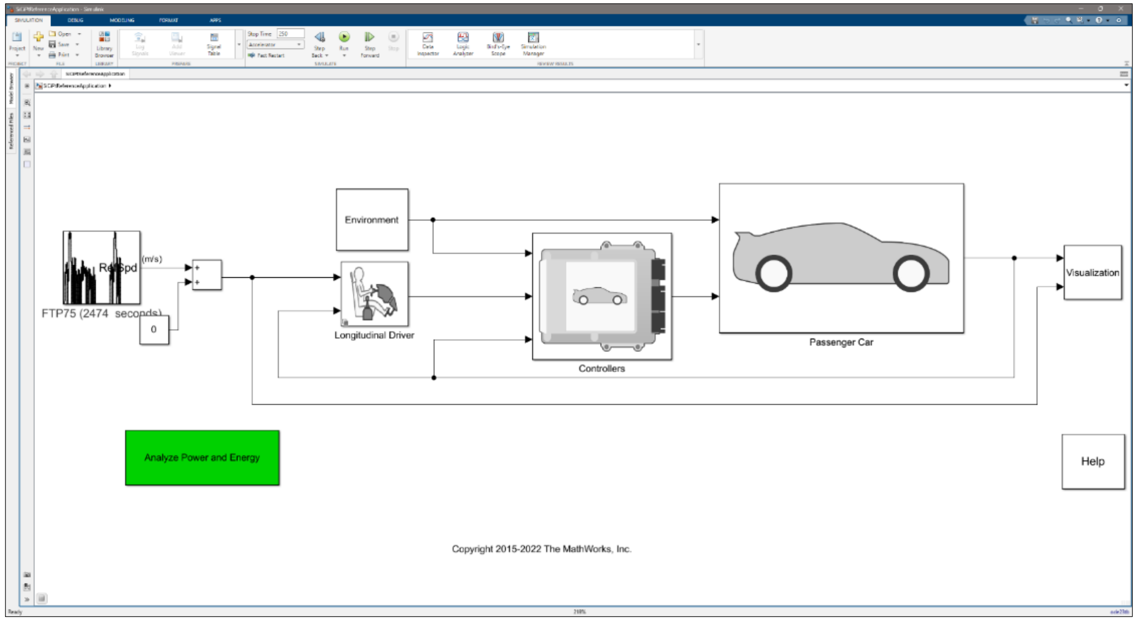
Replacing a first-principles engine model with an AI-based Reduced Order Model



Generate synthetic data for training

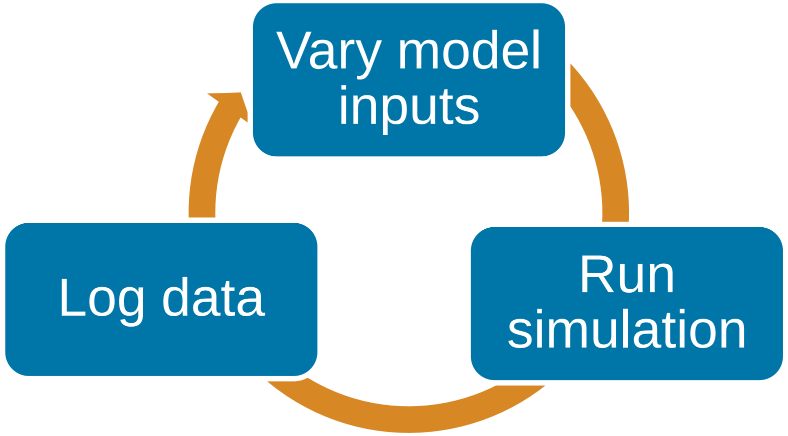
Data Preparation | AI Modeling | Simulation & Test | Deployment

Perform Design of Experiments (DoE) and generate synthetic data from Simulink model



DoE = 512×3 table

	EngTrqReq	EngSpdR...	SpkAdvOfst
1	60	2000	-30
2	128	2500	15
3	94	2750	8
4	111	2875	-19
5	77	2625	-11
6	144	2125	4
7	85	2563	-21
8	119	3313	-28
9	68	2938	21



Inputs
Engine speed (RPM)
Ignition timing
Throttle position
Wastegate valve

Output
Engine Torque

AI techniques that are suited for modeling dynamic systems

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

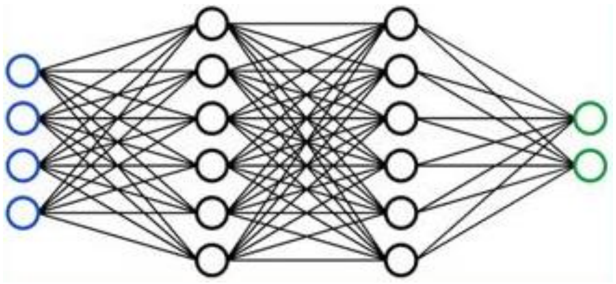
AI-Based (Data-Driven) ROM

- Neural State Space / Neural ODE
- LSTM
- Nonlinear ARX models
- etc.

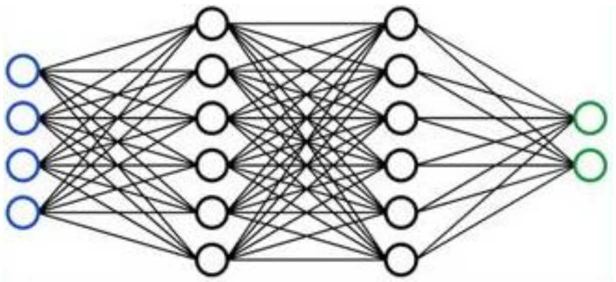
Create deep-learning based nonlinear state-space models without having to be a deep learning expert

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*)

Neural state-space model

Training Neural State Space Models

This example shows how to train and evaluate Neural State Space to model the behaviour of a vehicle engine.

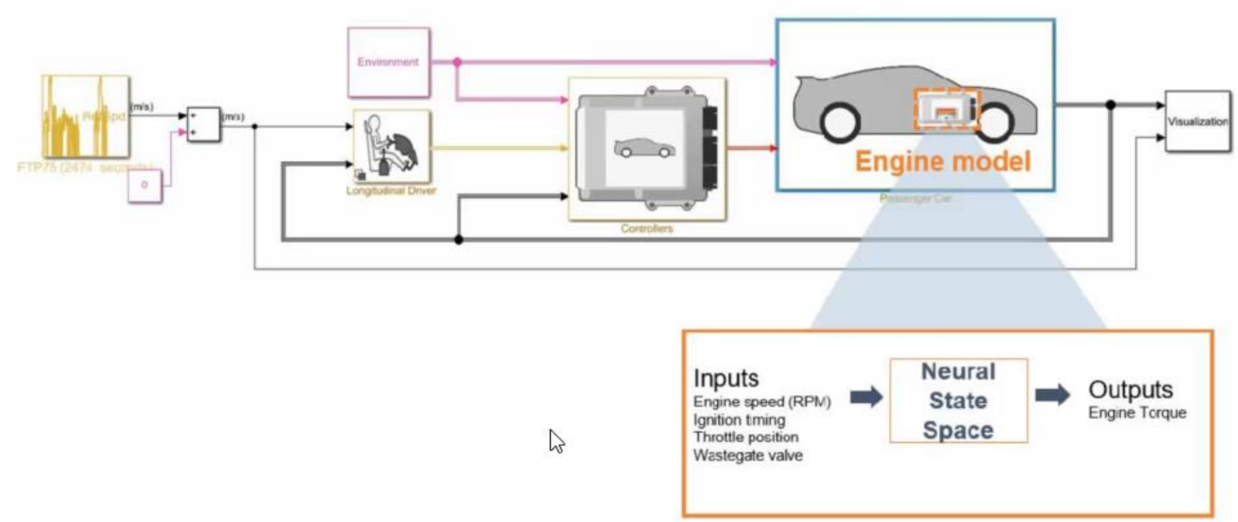


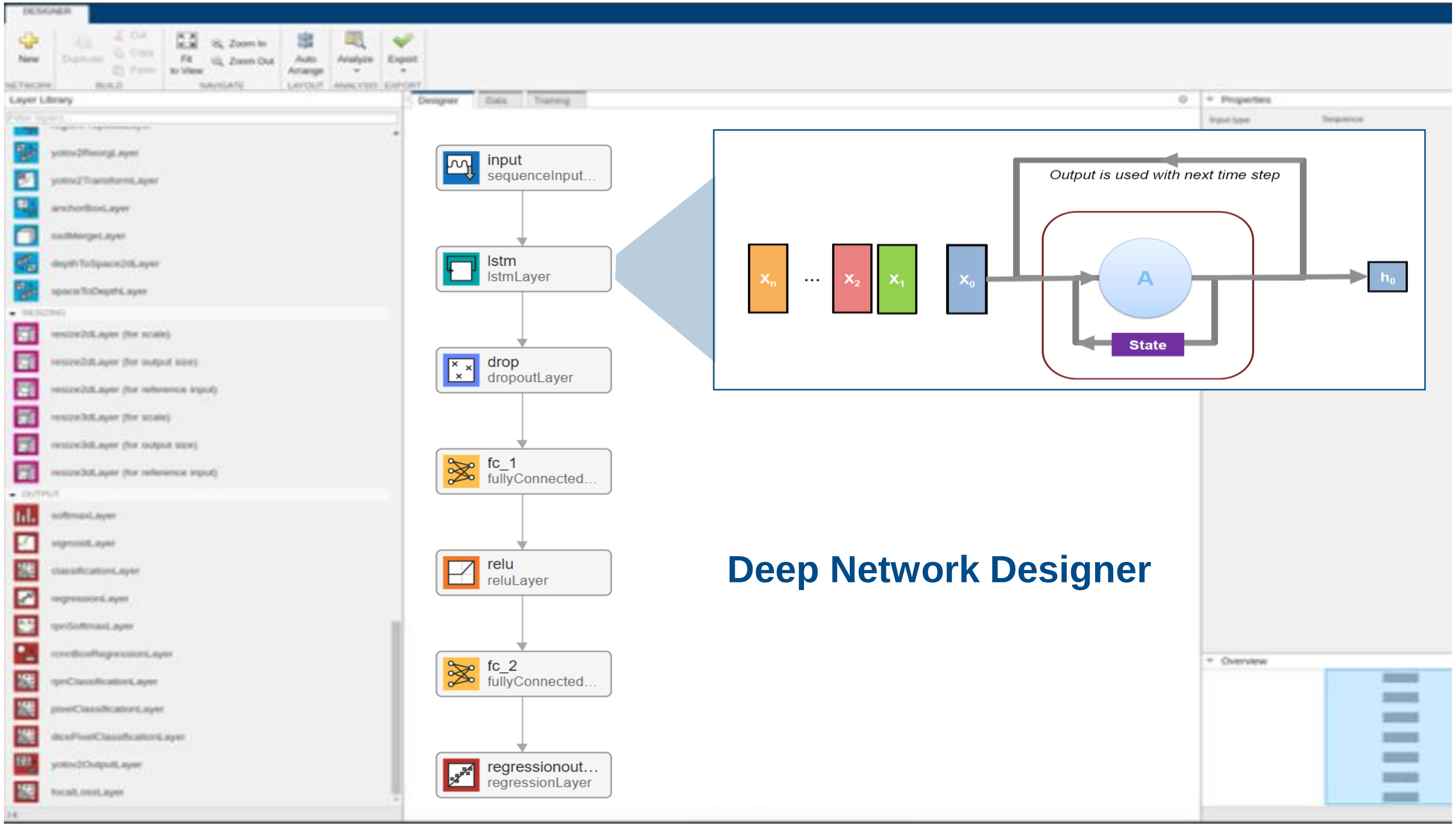
Table of Contents

1. Data preparation
   1.1. Prepare training and validation data
   1.2. Visually explore the data
2. Design and Train Neural State Space Model
3. Validate the Model

Project path

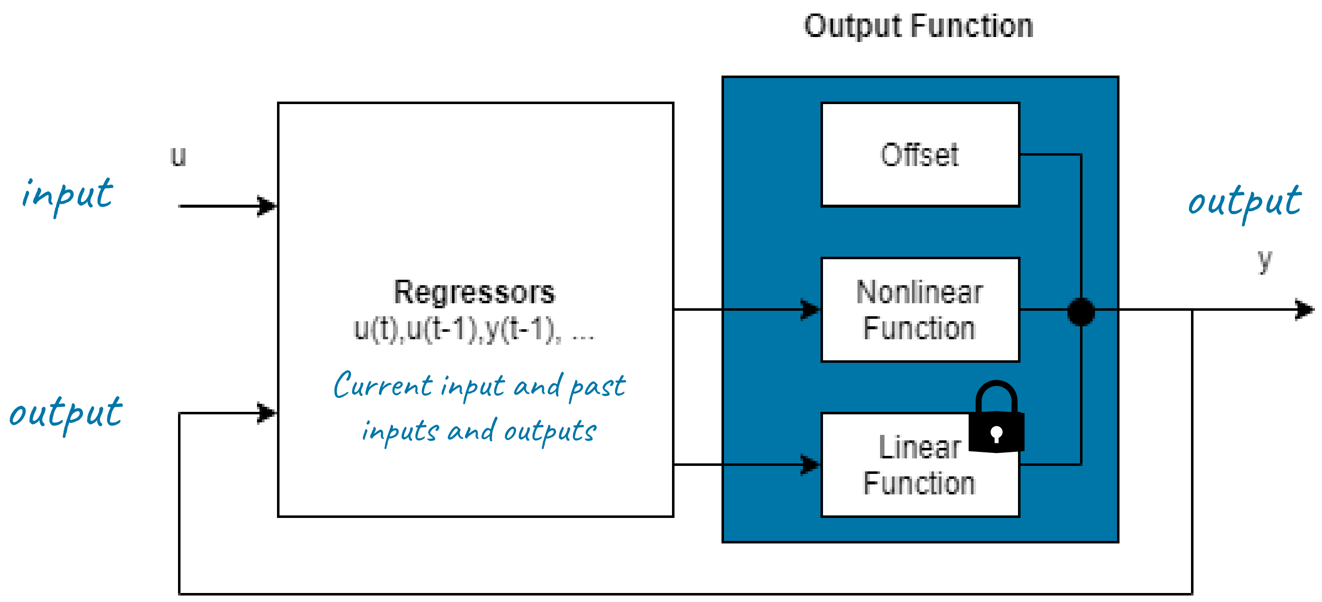
Capture time dependencies in time-series data using LSTM

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



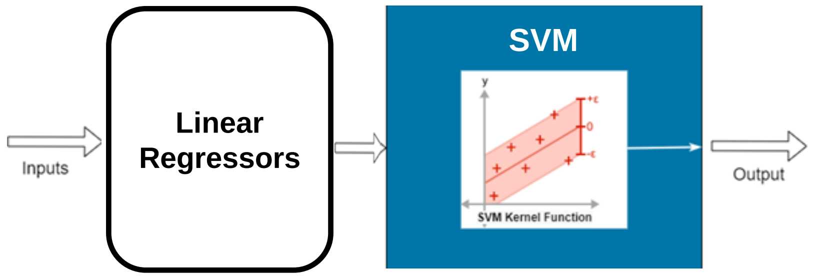
Include insights and knowledge of physics of your system using Nonlinear ARX Models

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



Extend linear models and model nonlinear behavior using flexible nonlinear functions

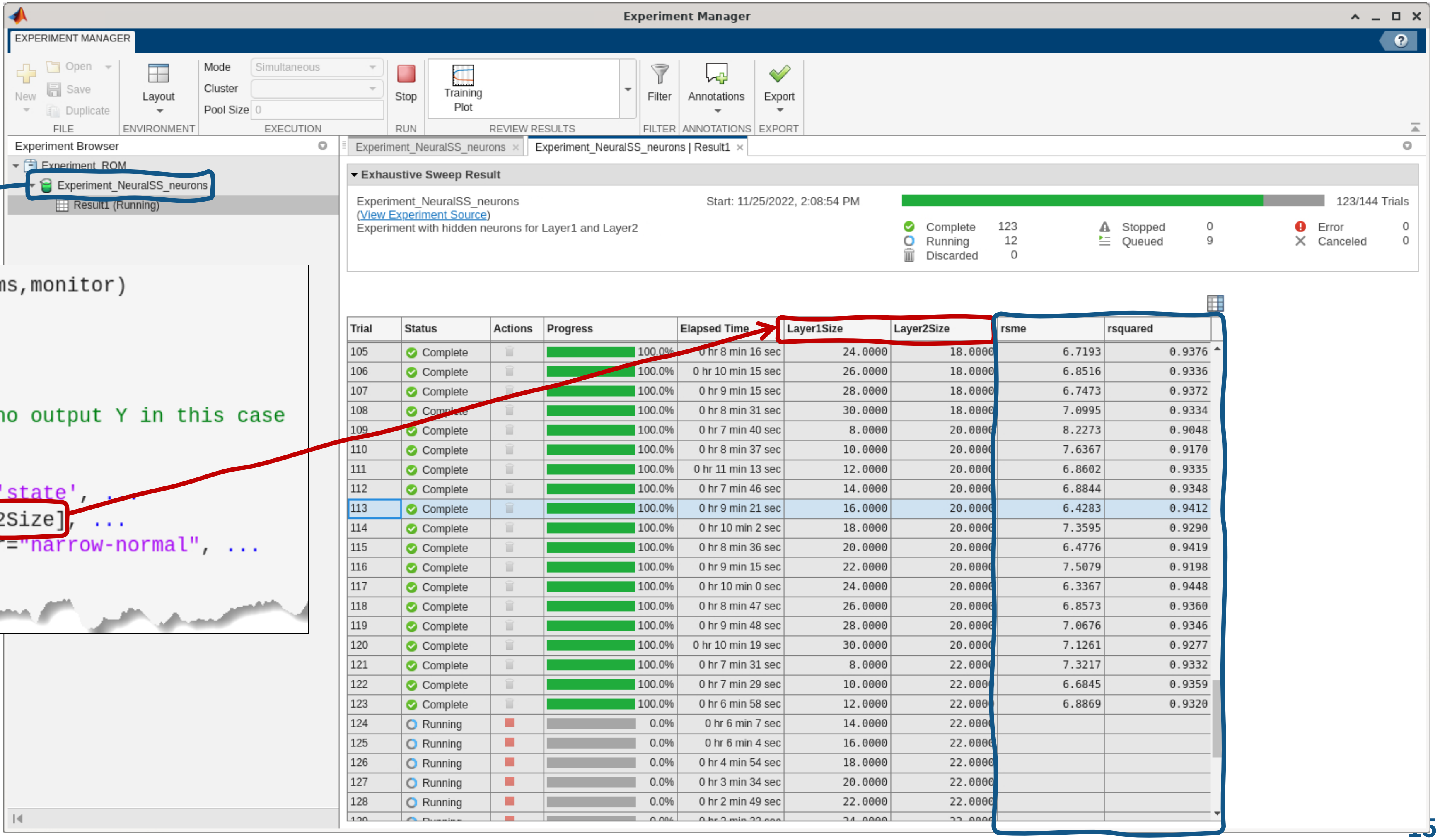
Selected configuration



Design and run experiments to train and compare your AI models with Experiment Manager

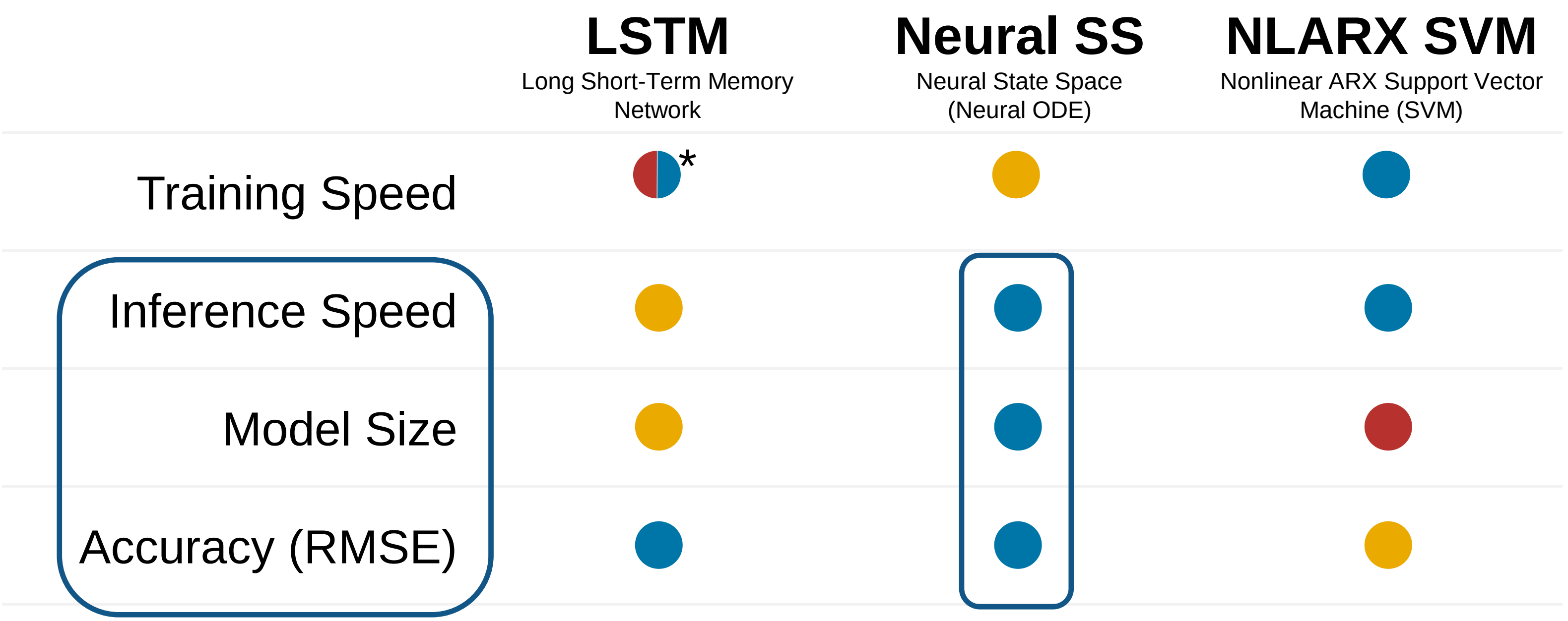
Data Preparation | **AI Modeling** | Simulation & Test | Deployment

```
function output = Experiment_NeuralSS_neurons(params,monitor)

    load engineData_neuralSS.mat; %#ok<*LOAD>
    load parameters;

    nssobj = idNeuralStateSpace(1,NumInputs=4); % no output Y in this case

    % Configure state network
    nssobj.StateNetwork = createMLPNetwork(nssobj,'state', ...
        LayerSizes=[params.Layer1Size params.Layer2Size], ...
        WeightsInitializer="glorot",BiasInitializer="narrow-normal", ...
        Activations='tanh');
```



Trial	Status	Actions	Progress	Elapsed Time	Layer1Size	Layer2Size	rsme	rsquared
105	Complete		100.0%	0 hr 8 min 16 sec	24.0000	18.0000	6.7193	0.9376
106	Complete		100.0%	0 hr 10 min 15 sec	26.0000	18.0000	6.8516	0.9336
107	Complete		100.0%	0 hr 9 min 15 sec	28.0000	18.0000	6.7473	0.9372
108	Complete		100.0%	0 hr 8 min 31 sec	30.0000	18.0000	7.0995	0.9334
109	Complete		100.0%	0 hr 7 min 40 sec	8.0000	20.0000	8.2273	0.9048
110	Complete		100.0%	0 hr 8 min 37 sec	10.0000	20.0000	7.6367	0.9170
111	Complete		100.0%	0 hr 11 min 13 sec	12.0000	20.0000	6.8602	0.9335
112	Complete		100.0%	0 hr 7 min 46 sec	14.0000	20.0000	6.8844	0.9348
113	Complete		100.0%	0 hr 9 min 21 sec	16.0000	20.0000	6.4283	0.9412
114	Complete		100.0%	0 hr 10 min 2 sec	18.0000	20.0000	7.3595	0.9290
115	Complete		100.0%	0 hr 8 min 36 sec	20.0000	20.0000	6.4776	0.9419
116	Complete		100.0%	0 hr 9 min 15 sec	22.0000	20.0000	7.5079	0.9198
117	Complete		100.0%	0 hr 10 min 0 sec	24.0000	20.0000	6.3367	0.9448
118	Complete		100.0%	0 hr 8 min 47 sec	26.0000	20.0000	6.8573	0.9360
119	Complete		100.0%	0 hr 9 min 48 sec	28.0000	20.0000	7.0676	0.9346
120	Complete		100.0%	0 hr 10 min 19 sec	30.0000	20.0000	7.1261	0.9277
121	Complete		100.0%	0 hr 7 min 31 sec	8.0000	22.0000	7.3217	0.9332
122	Complete		100.0%	0 hr 7 min 29 sec	10.0000	22.0000	6.6845	0.9359
123	Complete		100.0%	0 hr 6 min 58 sec	12.0000	22.0000	6.8869	0.9320
124	Running		0.0%	0 hr 6 min 7 sec	14.0000	22.0000		
125	Running		0.0%	0 hr 6 min 4 sec	16.0000	22.0000		
126	Running		0.0%	0 hr 4 min 54 sec	18.0000	22.0000		
127	Running		0.0%	0 hr 3 min 34 sec	20.0000	22.0000		
128	Running		0.0%	0 hr 2 min 49 sec	22.0000	22.0000		

Manage AI tradeoffs for your system



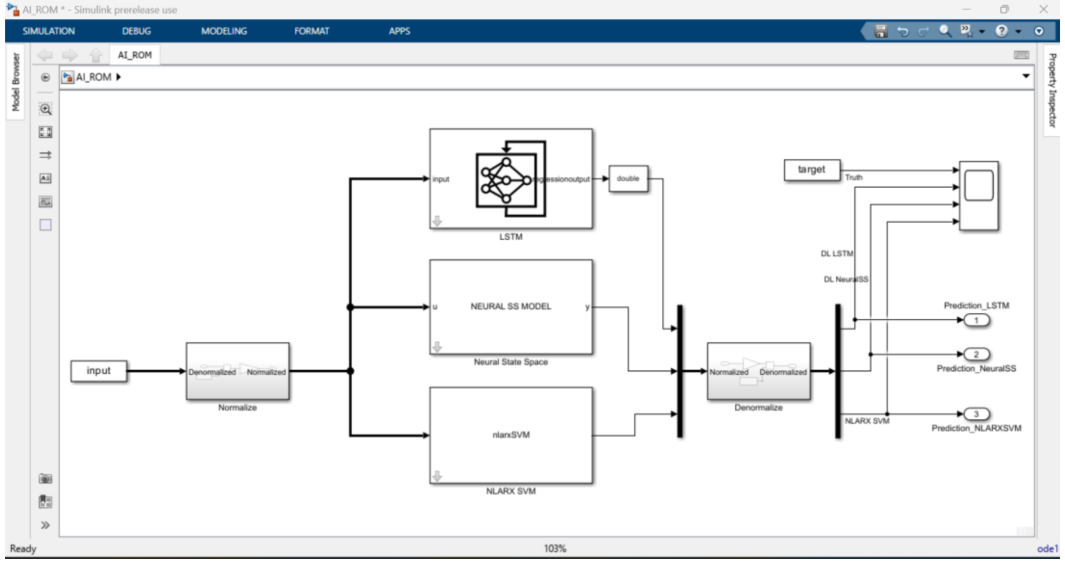
Results are specific to Vehicle Engine ROM example

* *if trained using a GPU. Testing made with GPU NVIDIA A100*

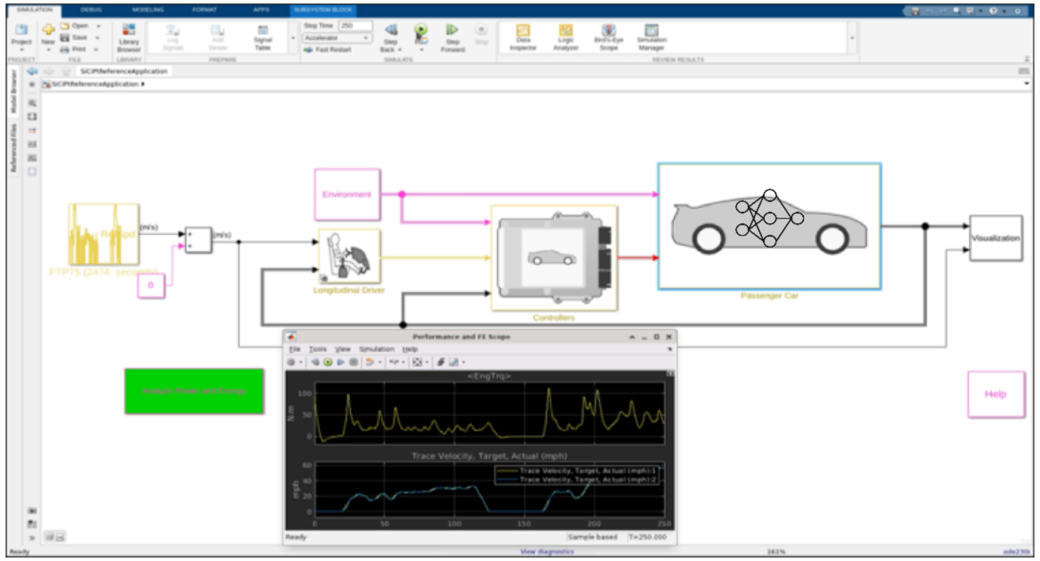
System-level simulation

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

Integration of trained AI model into Simulink

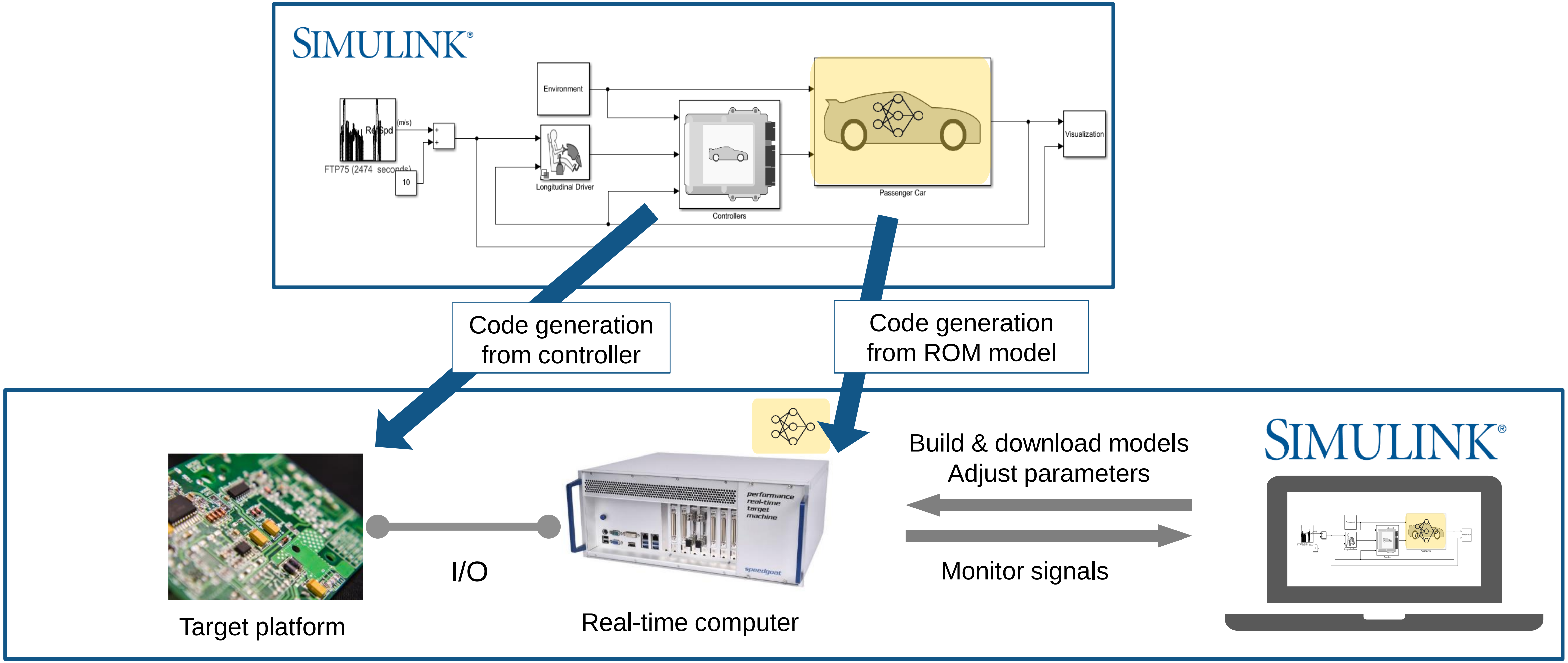


System-level simulation



Hardware-in-the-loop simulation

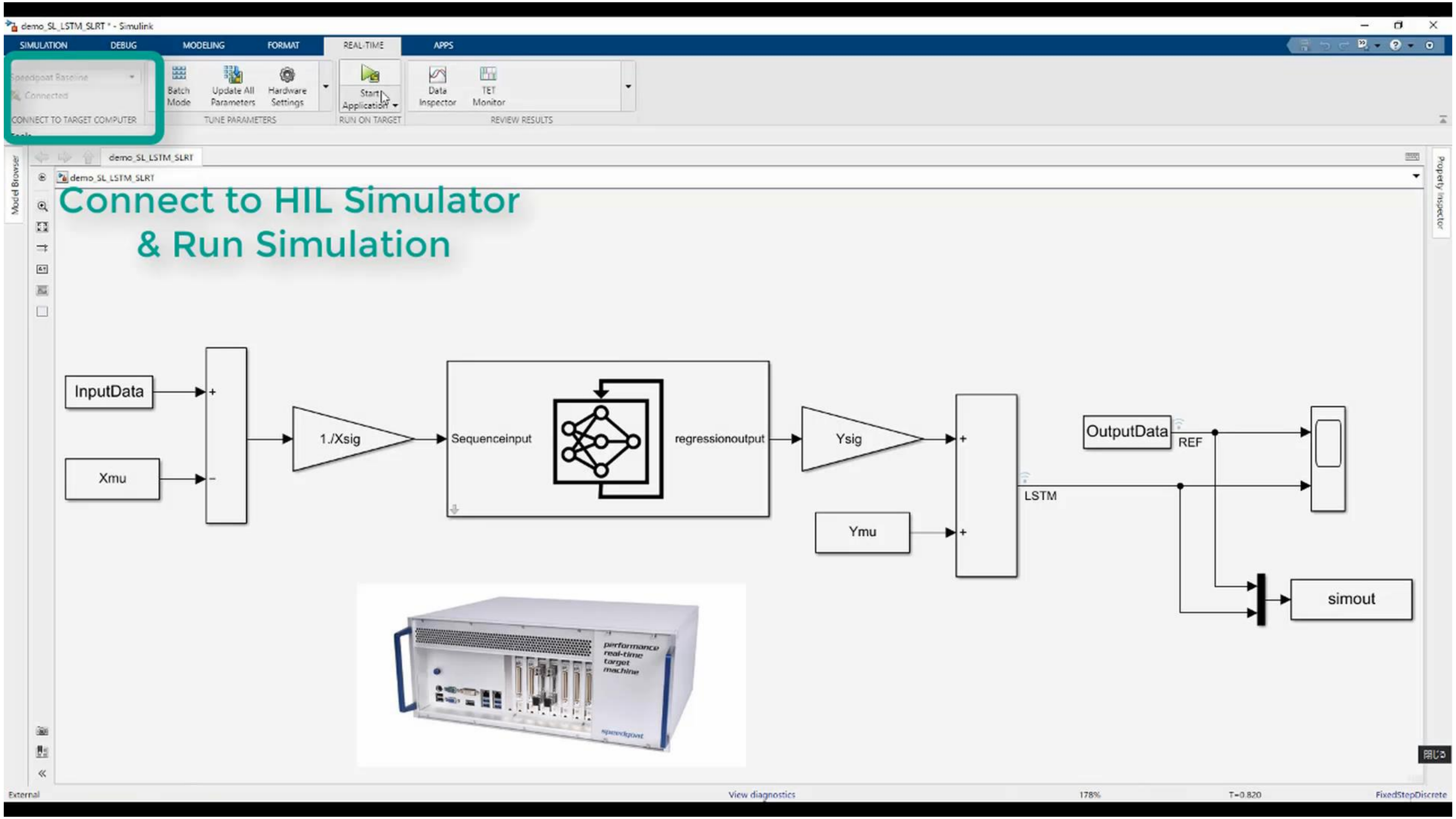
Data Preparation | AI Modeling | Simulation & Test | **Deployment**



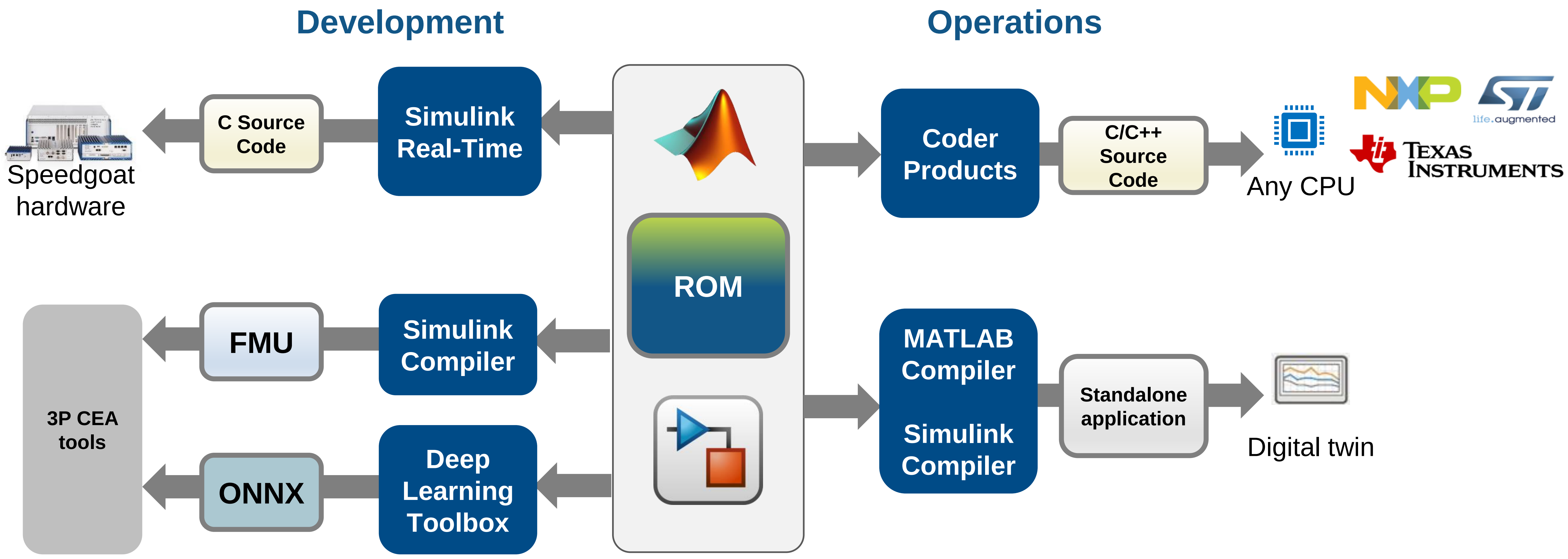
18

Hardware-in-the-loop simulation

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



Use ROMs outside of Simulink, for development and operation stages



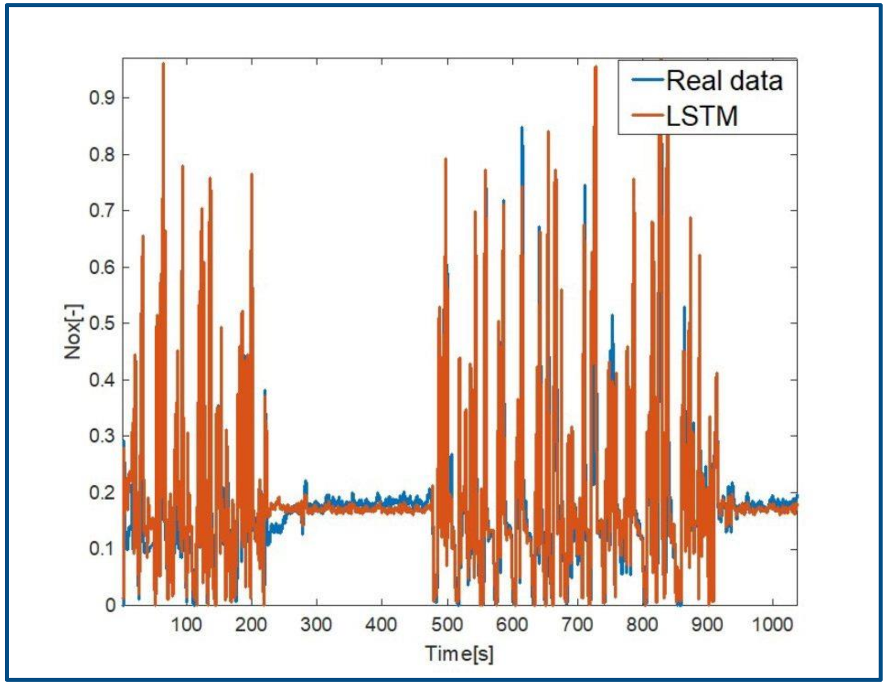
Renault Uses Deep Learning Networks to Estimate NO_X Emissions

Challenge
Design, simulate, and improve aftertreatment systems to reduce oxides of nitrogen (NO_X) emissions

Solution
Use MATLAB and Deep Learning Toolbox to model engine-out NO_X emissions using a long short-term memory (LSTM) network

Results
- NO_X emissions predicted with close to 90% accuracy
- LSTM network incorporated into after treatment simulation model
- Code generated directly from network for ECU deployment

Link to article



Measured NO_X emissions from an actual engine and modeled NO_X emissions from the LSTM network.

"Even though we are not specialists in deep learning, using MATLAB and Deep Learning Toolbox we were able to create and train a network that predicts NO_X emissions with almost 90% accuracy."

- Nicoleta-Alexandra Stroe, Renault

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.

MATLAB EXPO

Thank you

MathWorks®