



# AI-Powered Virtual Sensors in Embedded Applications



Shang-Chuan Lee, Ph.D

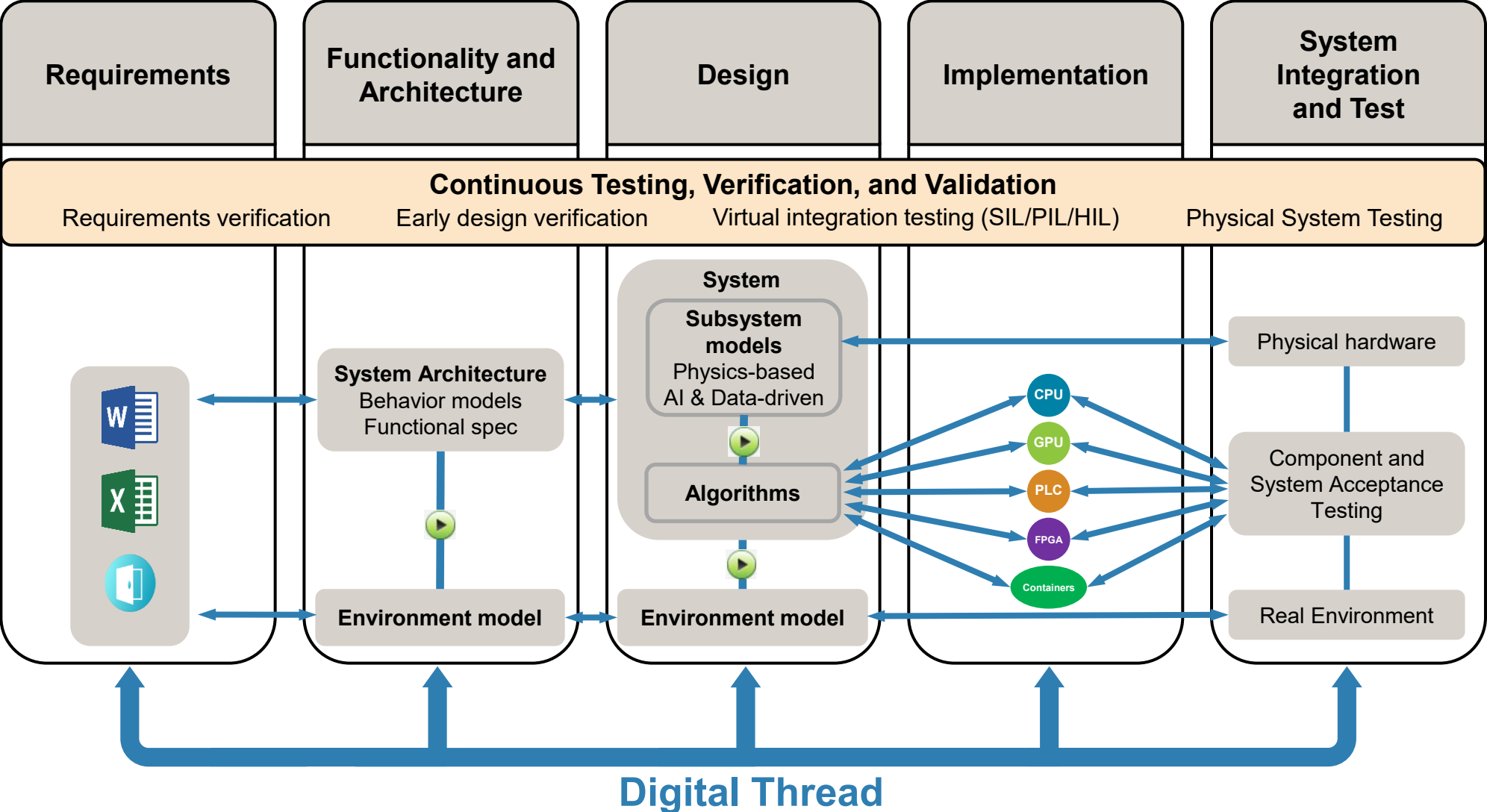
Principal Application Engineer  
MathWorks



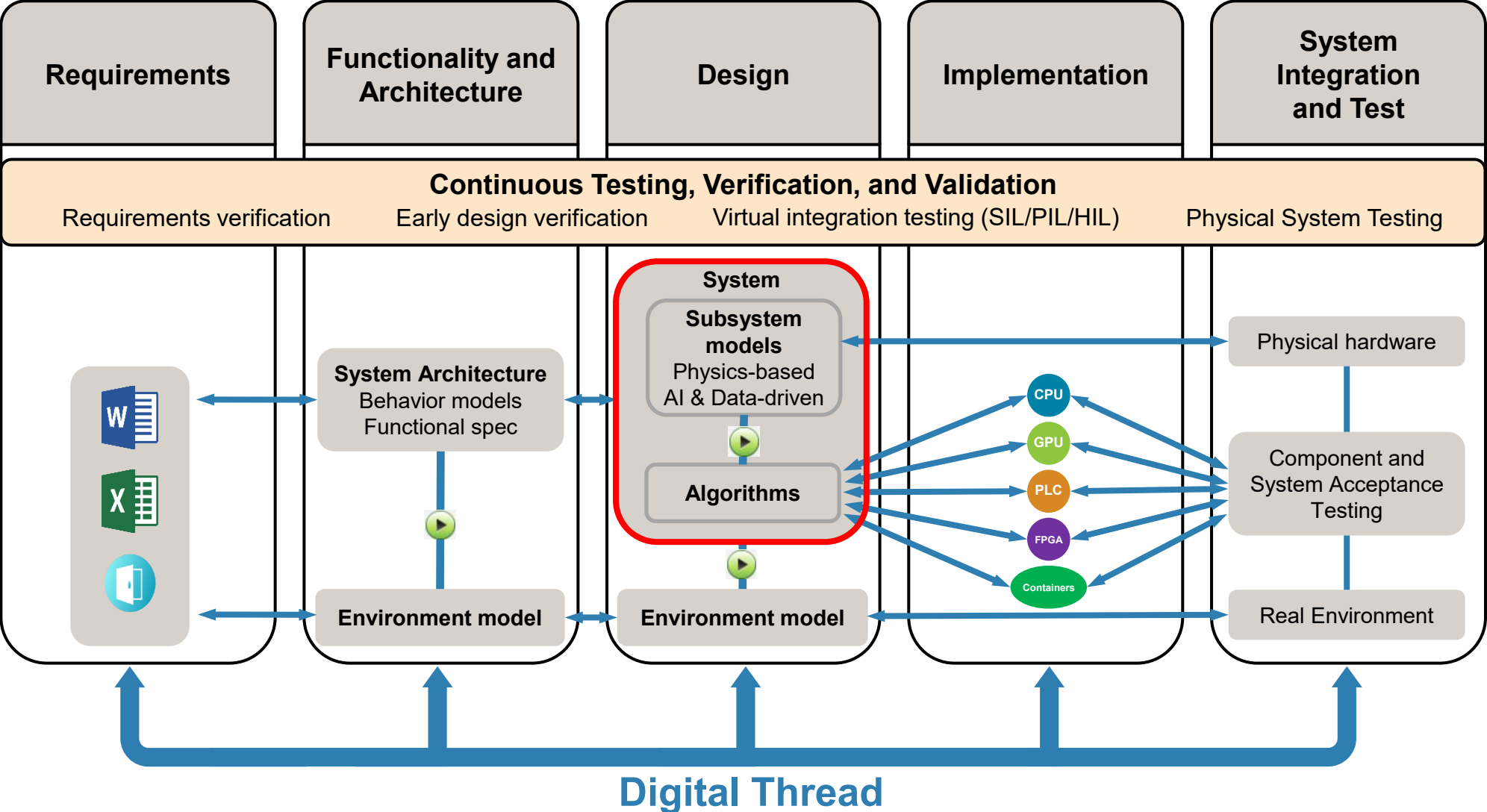
# Agenda

- AI in Model-Based Design Workflow
  - From modeling to deployment
- Virtual Sensors in System-Level Simulation
  - Electric motor rotor position tracking for controls
  - Battery SOC estimation
- PIL testing and production deployment
  - Profile code performance and evaluate trade-offs

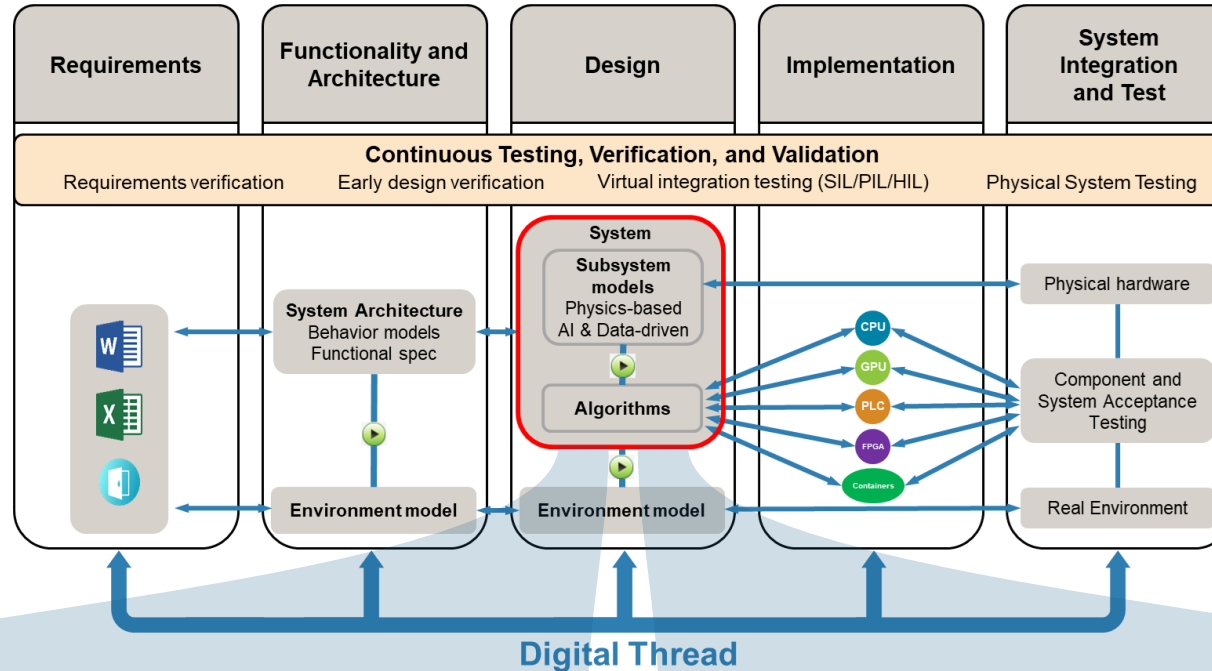
# Model-Based Design



# Integrating AI into Model-Based Design



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



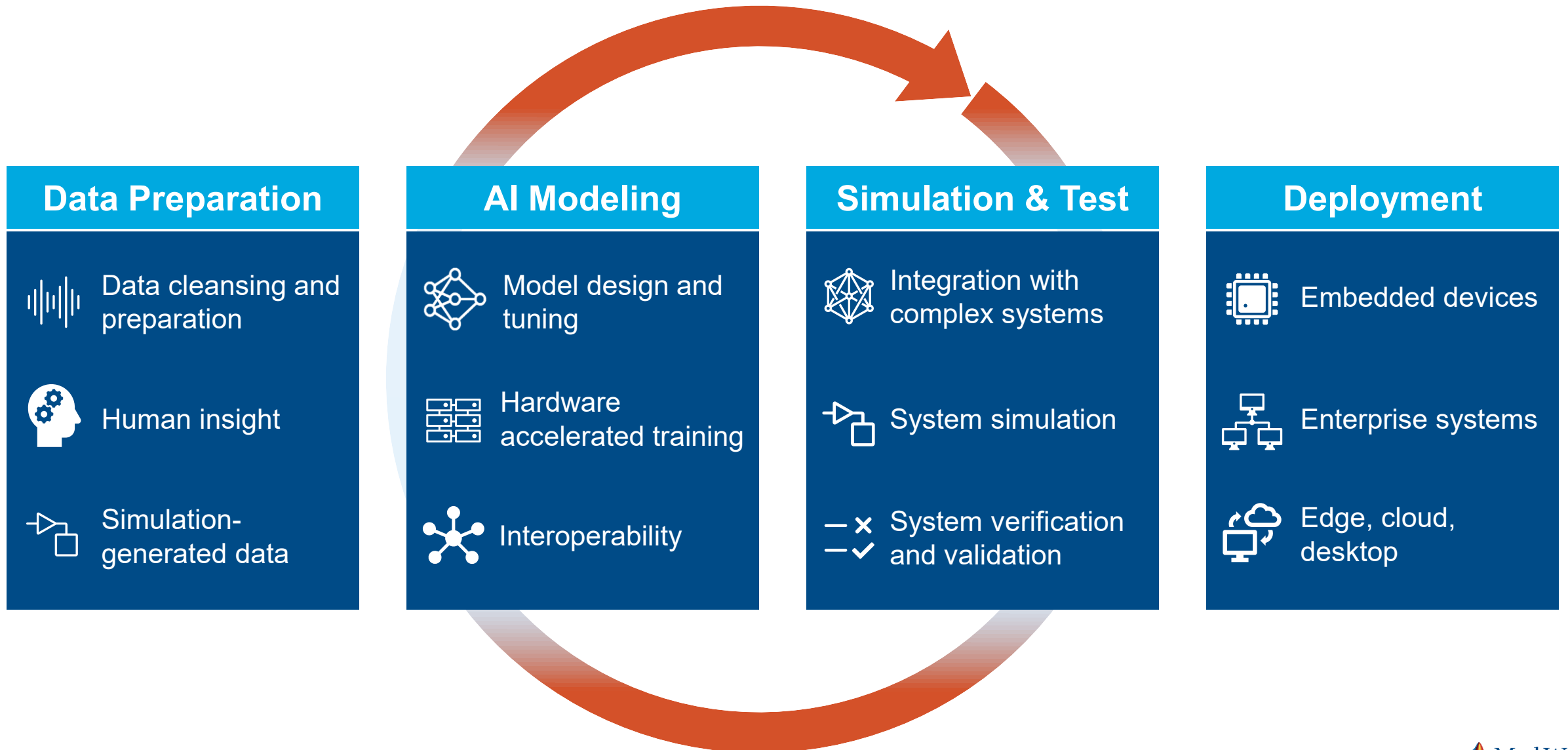
## AI for component modeling

- Speeding up desktop and HIL simulations
- Modeling component dynamics from data when first-principles models cannot be obtained

## AI for algorithm development

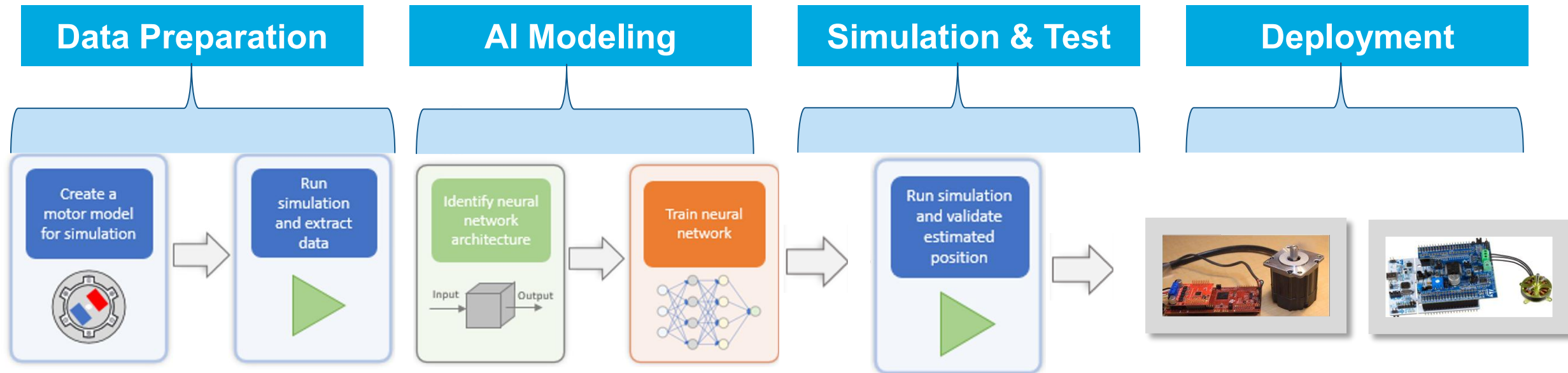
- **Virtual sensor modeling**
- Sensor fusion
- Object detection

# AI-driven system design

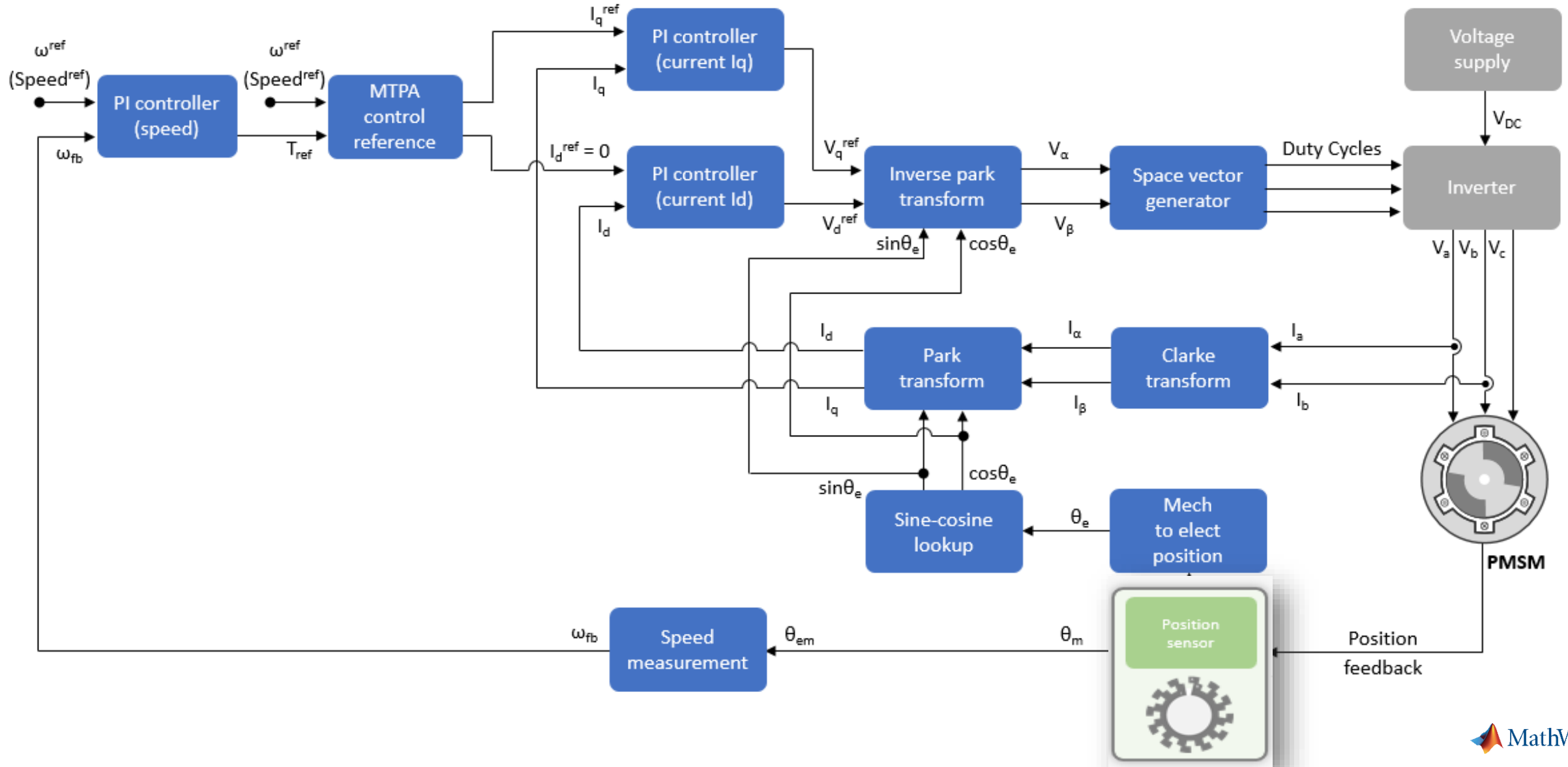


# AI-driven system design

## Workflow for using AI for PMSM position estimation

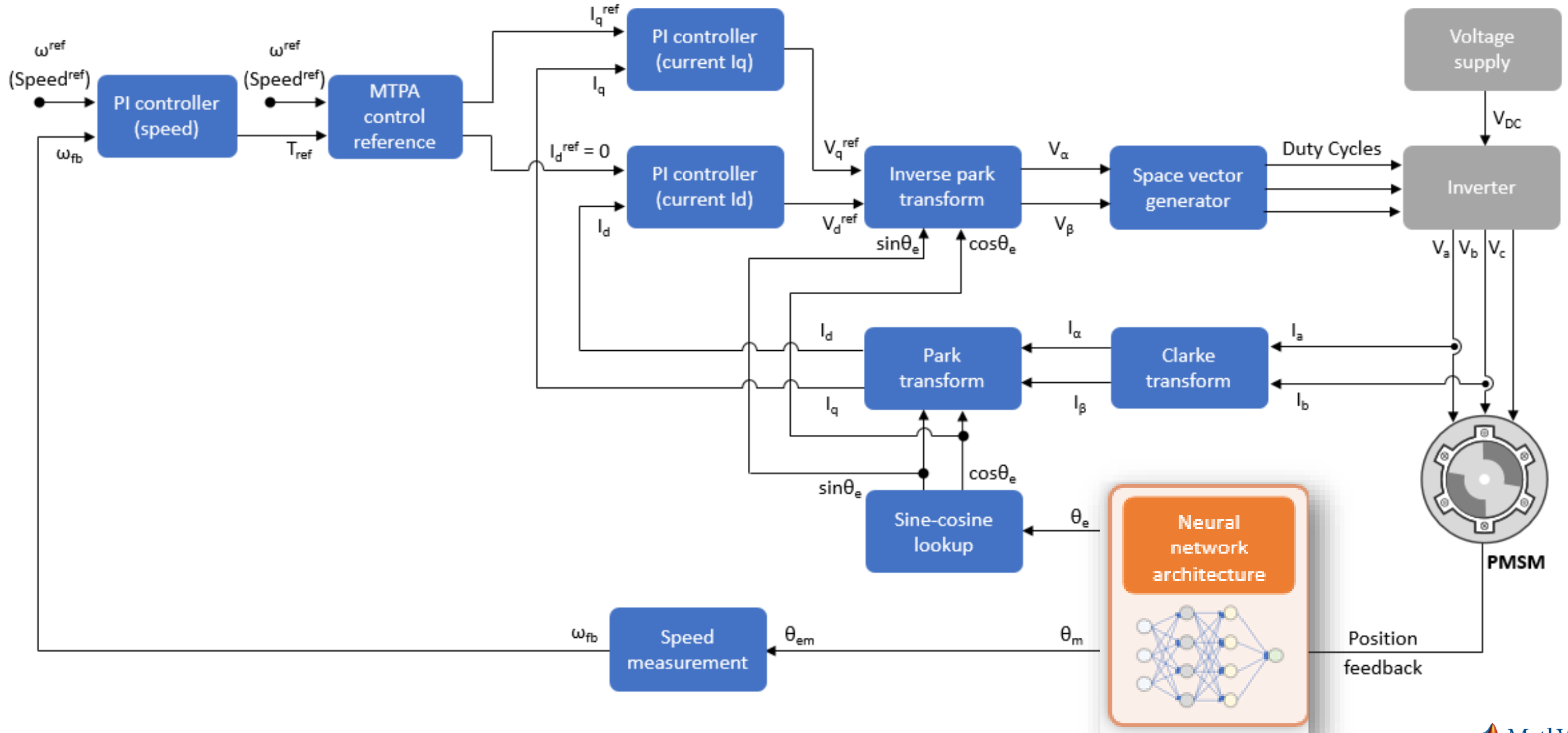


# Field-oriented control using physical position sensors

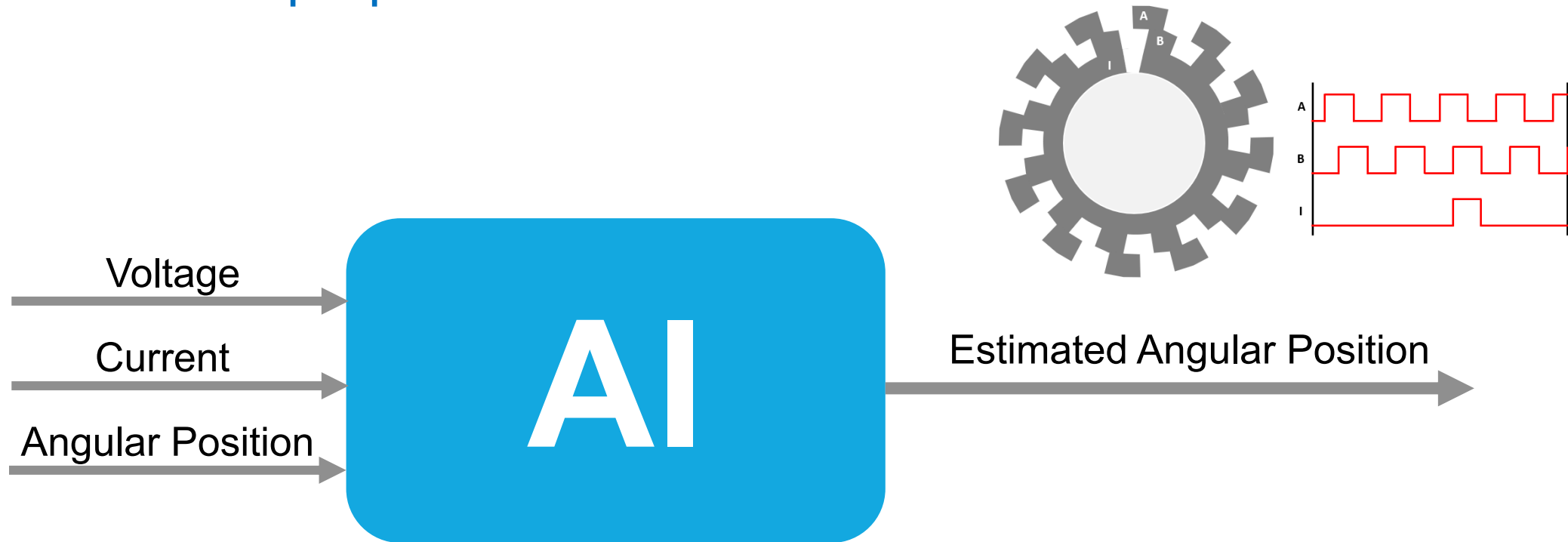




# Field-oriented control using AI for position sensors



# Data collection preparation



Collect the simulation data

Speed in PU ( $\omega$ )	Load Torque in PU (TL)	Number of data collected (Electrical cycles)
$\omega < 0.1$	0.05	100
$0.1 < \omega < 1$	$0 < TL < 1$	1000

Data collection range

## 3 options for AI Modeling:

1

Train in MATLAB's **Machine Learning** Framework

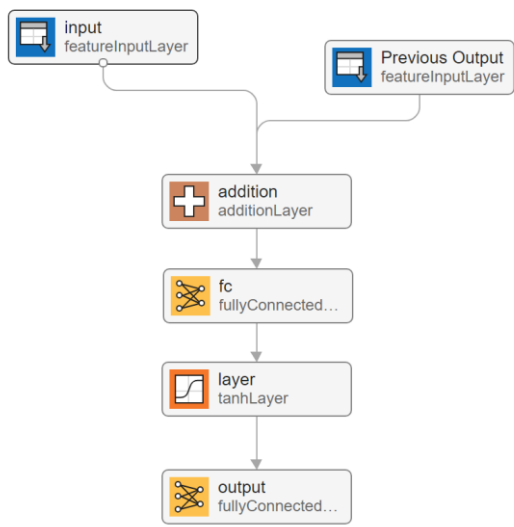
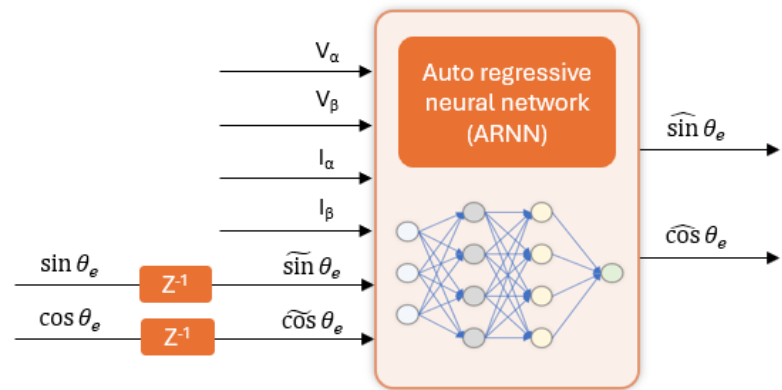
2

Import model from  
**TensorFlow or PyTorch**

3

Train in MATLAB's **Deep Learning** Framework

# Train the neural network



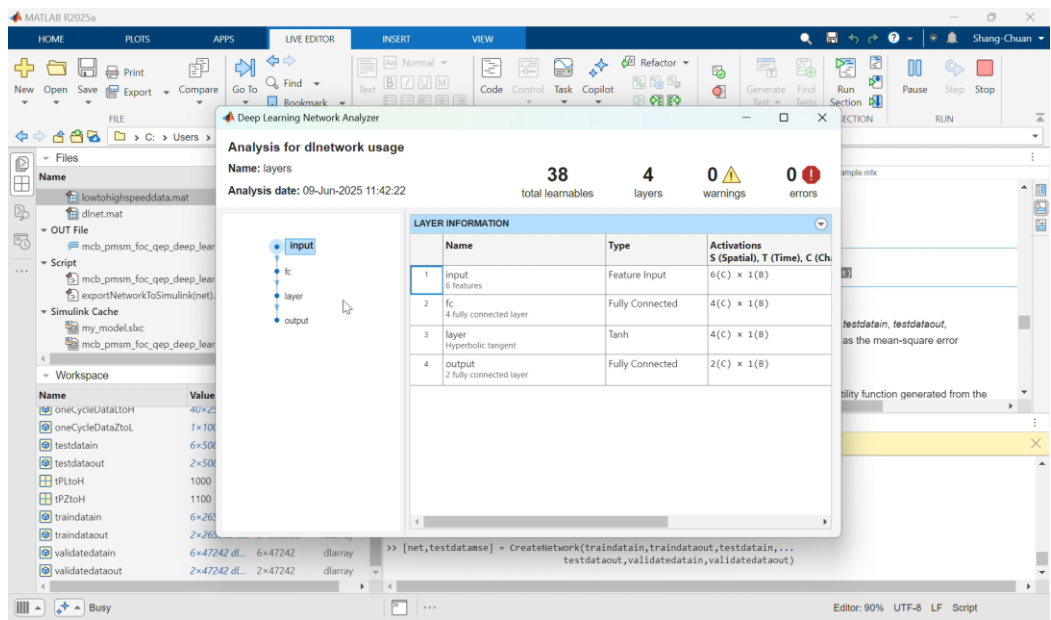
```
function [net,testdatamse] = CreateNetwork(traindatain,traindataout,testdatain,...
                                         testdataout,validatedatain,validatedataout)

% trains a network using dlnetwork

% Copyright 2024 The MathWorks, Inc.

% rng(1); % Fix the seed number
rng('default')

layers = [
    featureInputLayer(6)
    fullyConnectedLayer(4)
    tanhLayer
    fullyConnectedLayer(2,'Name','output')
];
```



Data Preparation

AI Modeling

Simulation & Test

Deployment

# Exported trained neural network to Simulink model

The screenshot displays the MATLAB R2025a environment. The 'LIVE EDITOR' tab is active, showing a script named 'FOCPMSMUsingPosEstByNeuralNtwExample.mlx'. The script contains the following code:

```
rng('default')
layers = [
    featureInputLayer(6)
    fullyConnectedLayer(4)
    tanhLayer
    fullyConnectedLayer(2, 'Name', 'output')
];
% Run this to visualise the dlnetwork
% analyzeNetwork(layers)
```

A red box highlights the `layers` array definition. Below the code, a note states: "Note: The example uses the `exportNetworkToSimulink` function from Deep Learning Toolbox to export the trained ARNN network (`net`) to the Simulink layers blocks so that you can use the network for simulation and code generation."

Below the note, the text "Run the following command to execute the function." is followed by a text box containing the command:

```
exportNetworkToSimulink(net)
```

The text below the text box reads: "The function accepts the `dlnetwork` object `net` to create the Simulink layers blocks in a new Simulink model as shown in the following figure:"

The 'Workspace' window on the left shows the following variables:

Name	Value	Size	Class
dPTorLtoH	25	1×1	double
lowtohighspeeddata	40×25 Simu...	40×25	Simulink.Si...
net	1×1 dlnetwo...	1×1	dlnetwork
oneCycleDataLtoH	40×25 Simu...	40×25	Simulink.Si...
oneCycleDataZtoL	1×100 Simu...	1×100	Simulink.Si...
testdatain	6×50859 dl...	6×50859	dlarray
testdatamse	7.2062e-06	1×1	double
testdataout	2×50859 dl...	2×50859	dlarray
tPLtoH	1000	1×1	double
tPZtoH	1100	1×1	double
traindatain	6×265332 d...	6×265332	dlarray
traindataout	2×265332 d...	2×265332	dlarray
validatedatain	6×47242 dl...	6×47242	dlarray

The 'Command Window' at the bottom shows the following output:

```
New to MATLAB? See resources for Getting Started.
View summary with summary.

testdatamse =

7.2062e-06

>>
```

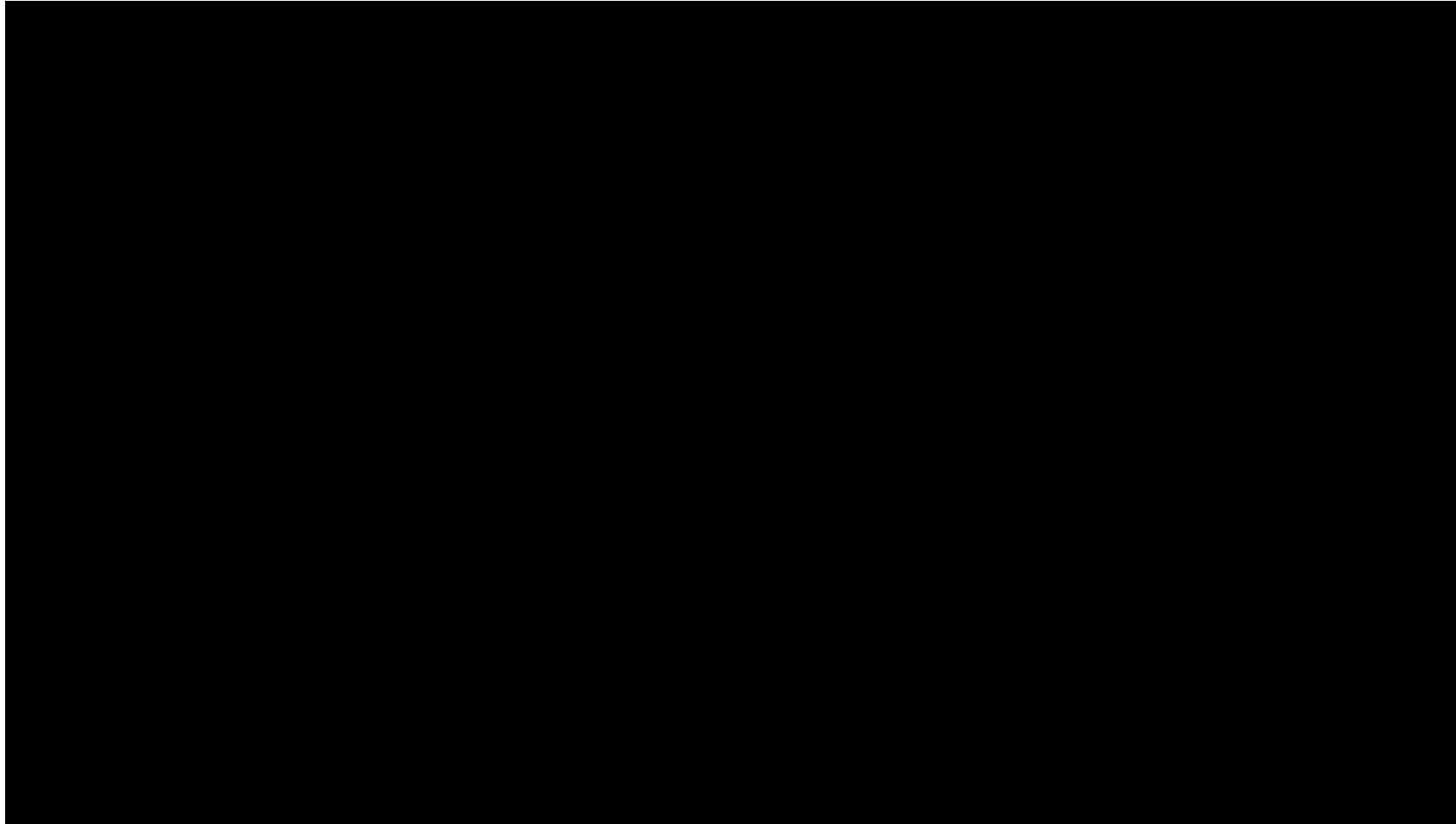
Data Preparation

AI Modeling

Simulation & Test

Deployment

# Deploy speed control using neural network in hardware



Data Preparation

AI Modeling

Simulation & Test

Deployment

# Agenda

AI in Model-Based Design Workflow

From modeling to deployment

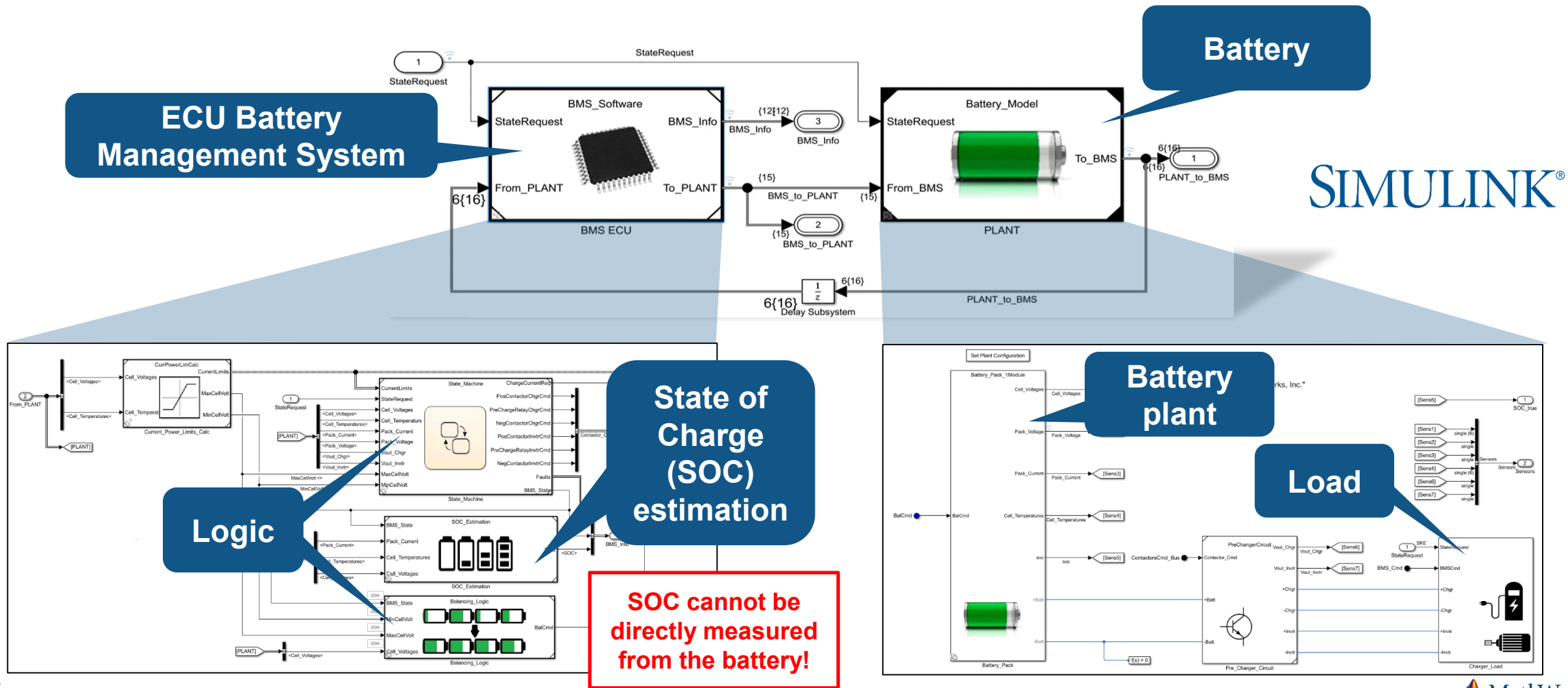
- Virtual Sensors in system-level simulation
  - Battery SOC estimation

Electric motor rotor position tracking for controls

PIL testing and production deployment

Profile code performance and evaluate trade-offs

# System-level simulation of BMS



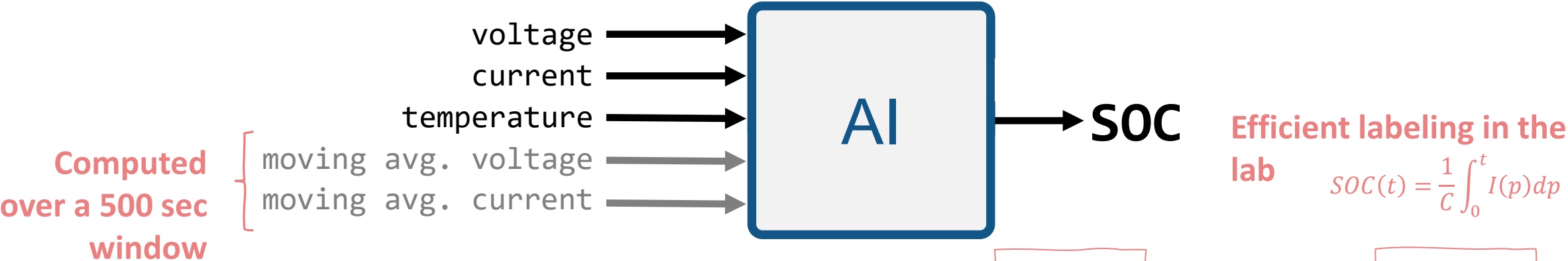


# Battery state-of-charge estimation using AI



# Data preparation

Data source: McMaster University\*



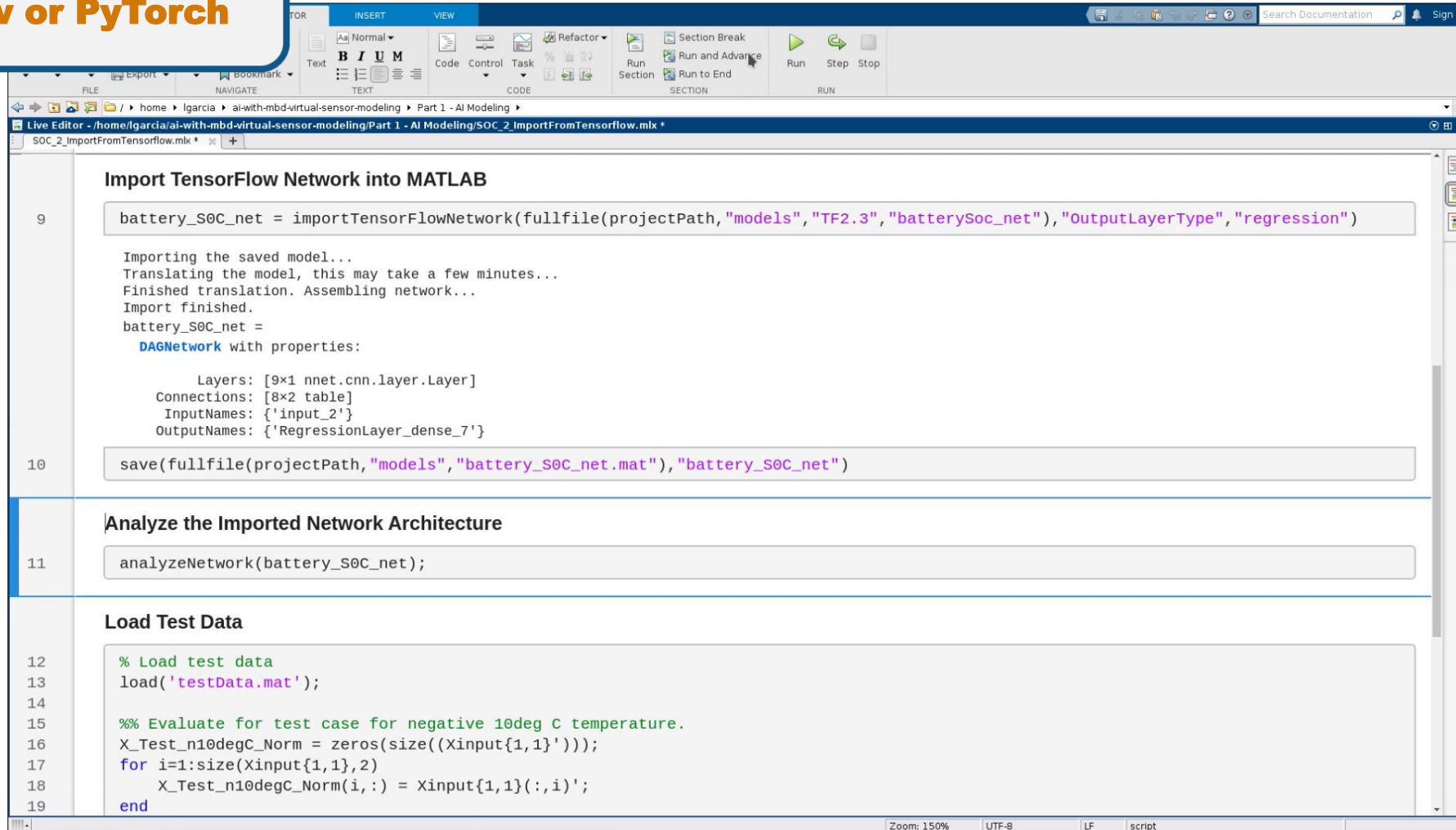
Predictors

Response

	1	2	3	4	5	6
	Voltage	Current	Temperature	Moving Average Voltage	Moving Average Current	SOC
1	0.7510	0.3851	0.3031	0.7510	0.3851	0.2064
2	0.7510	0.3852	0.3046	0.7510	0.3851	0.2064
3	0.7510	0.3852	0.3061	0.7510	0.3852	0.2064
4	0.7510	0.3852	0.3076	0.7510	0.3852	0.2064
5	0.7510	0.3852	0.3091	0.7510	0.3852	0.2064
6	0.7510	0.3852	0.3106	0.7510	0.3852	0.2064
7	0.7510	0.3852	0.3120	0.7510	0.3852	0.2064
8	0.7510	0.3852	0.3135	0.7510	0.3852	0.2064
9	0.7510	0.3852	0.3150	0.7510	0.3852	0.2064
10	0.7510	0.3852	0.3165	0.7510	0.3852	0.2064

\* <https://data.mendeley.com/datasets/cp3473x7xv/3>

### Import model from TensorFlow or PyTorch



The image shows a MATLAB Live Editor window with the following code:

```
9  battery_S0C_net = importTensorFlowNetwork(fullfile(projectPath,"models","TF2.3","batterySoc_net"),"OutputLayerType","regression")

    Importing the saved model...
    Translating the model, this may take a few minutes...
    Finished translation. Assembling network...
    Import finished.
    battery_S0C_net =
        DAGNetwork with properties:
            Layers: [9x1 nnet.cnn.layer.Layer]
            Connections: [8x2 table]
            InputNames: {'input_2'}
            OutputNames: {'RegressionLayer_dense_7'}

10  save(fullfile(projectPath,"models","battery_S0C_net.mat"),"battery_S0C_net")

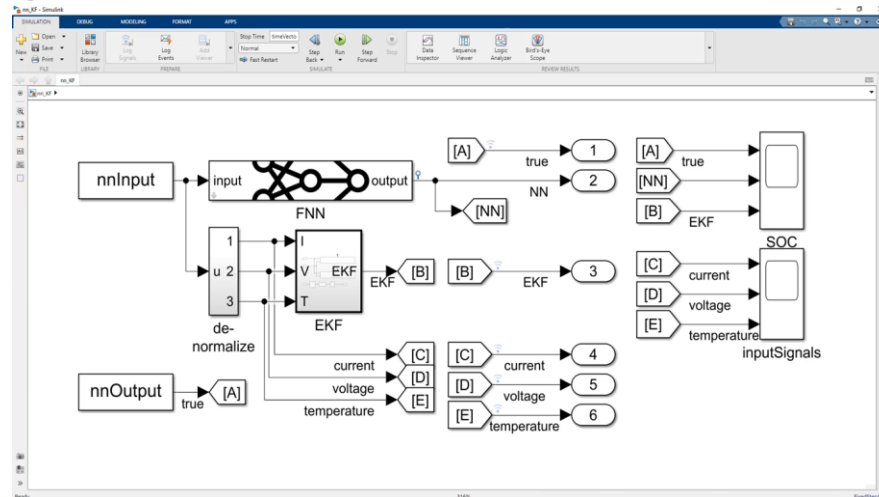
11  analyzeNetwork(battery_S0C_net);

Load Test Data

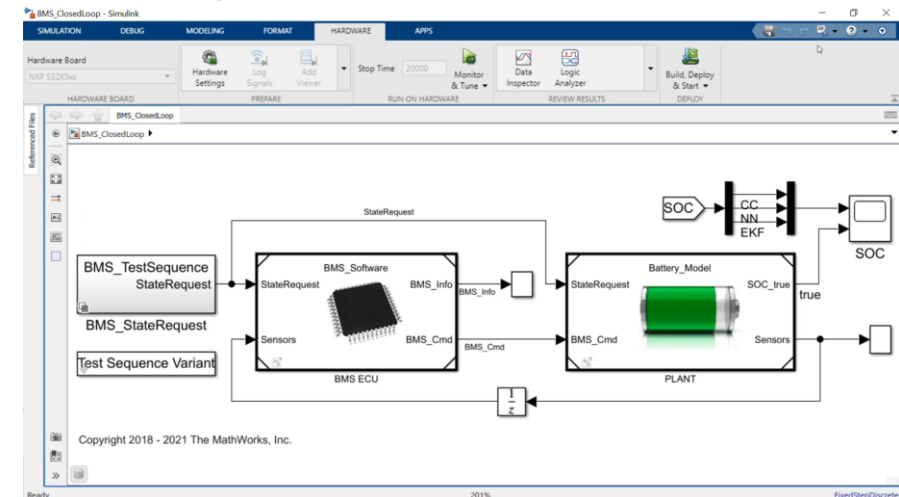
12  % Load test data
13  load('testData.mat');
14
15  %% Evaluate for test case for negative 10deg C temperature.
16  X_Test_n10degC_Norm = zeros(size((Xinput{1,1}')));
17  for i=1:size(Xinput{1,1},2)
18      X_Test_n10degC_Norm(i,:) = Xinput{1,1}(:,i)';
19  end
```

# Integrate your AI model for system-level simulation and test

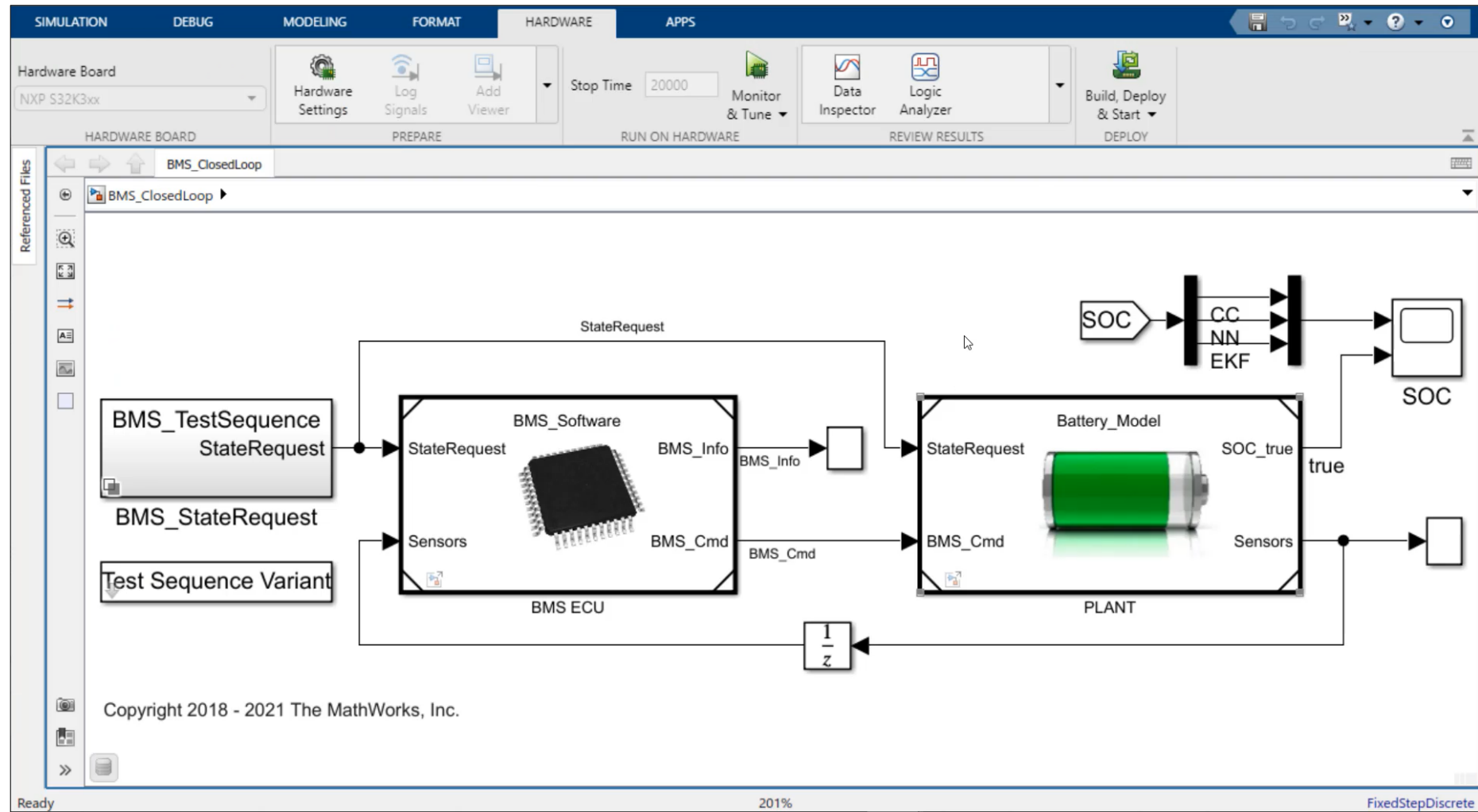
## Integration of trained AI model into Simulink



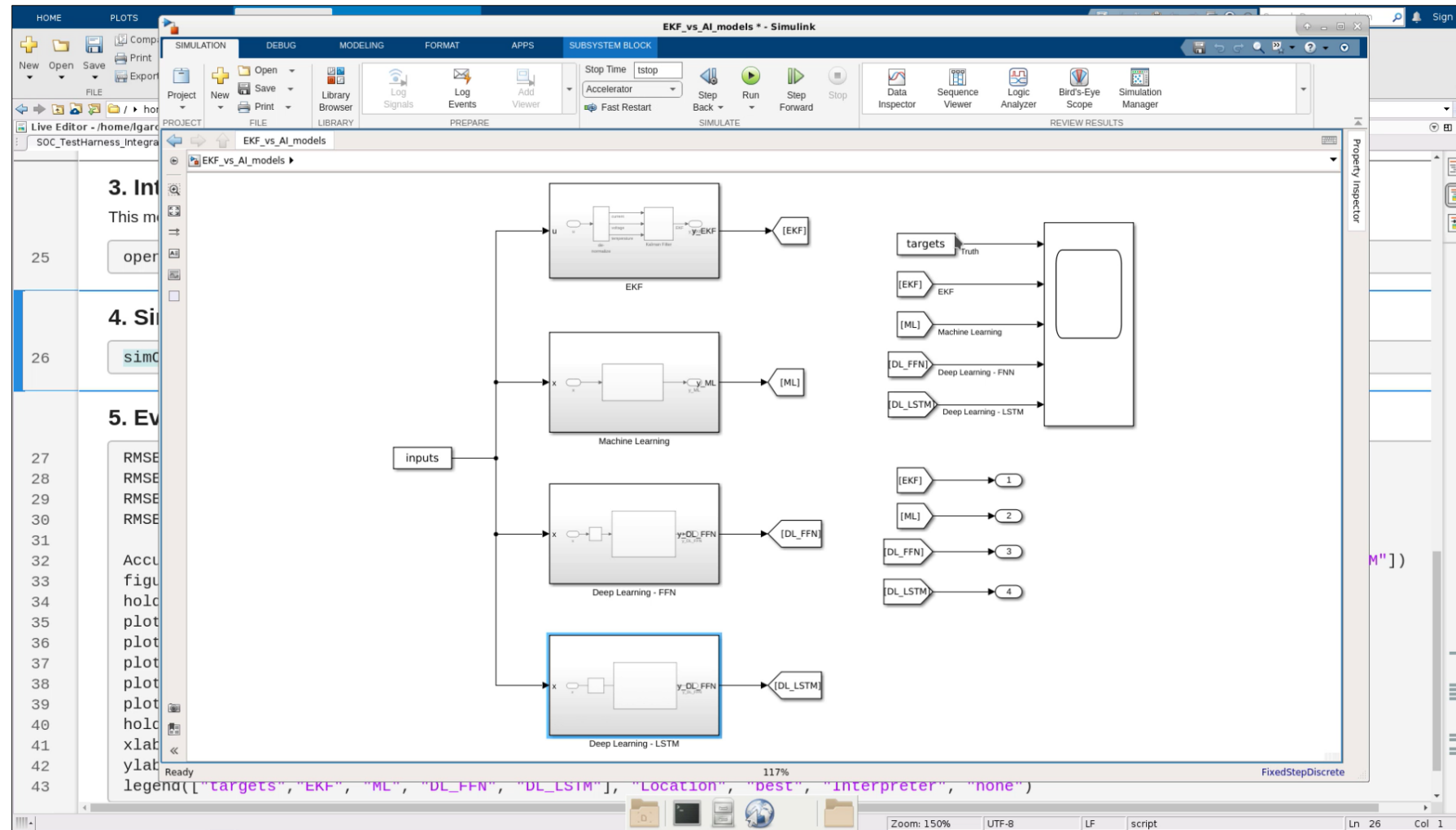
## System-level simulation



# Closed-loop system-level simulation



# Integration of trained AI models into Simulink



Data Preparation

AI Modeling

Simulation & Test

Deployment

# Agenda

Design and train AI-based virtual sensors

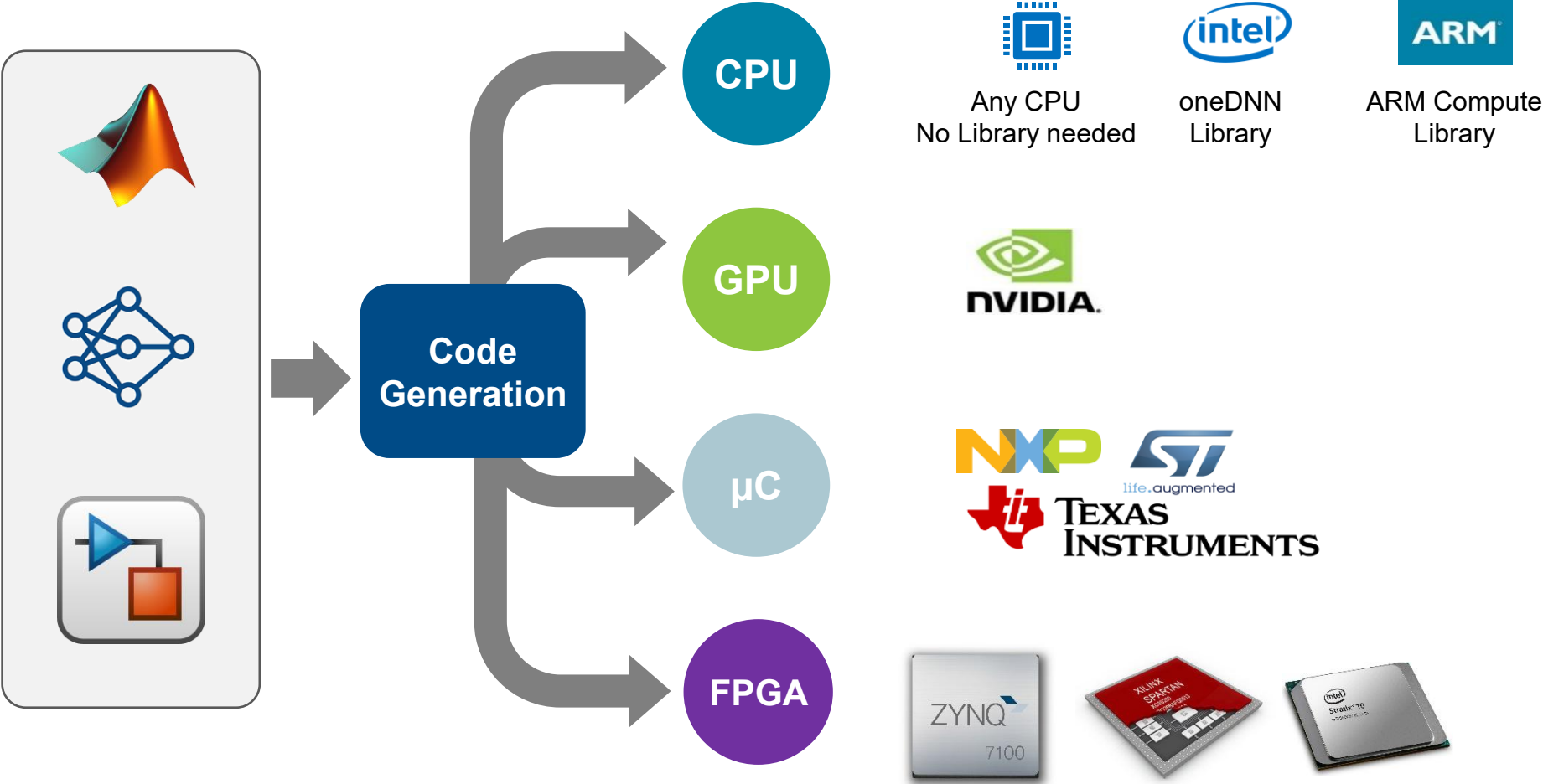
Battery SOC estimation

Electric motor rotor position tracking for controls

System-level simulation and verification

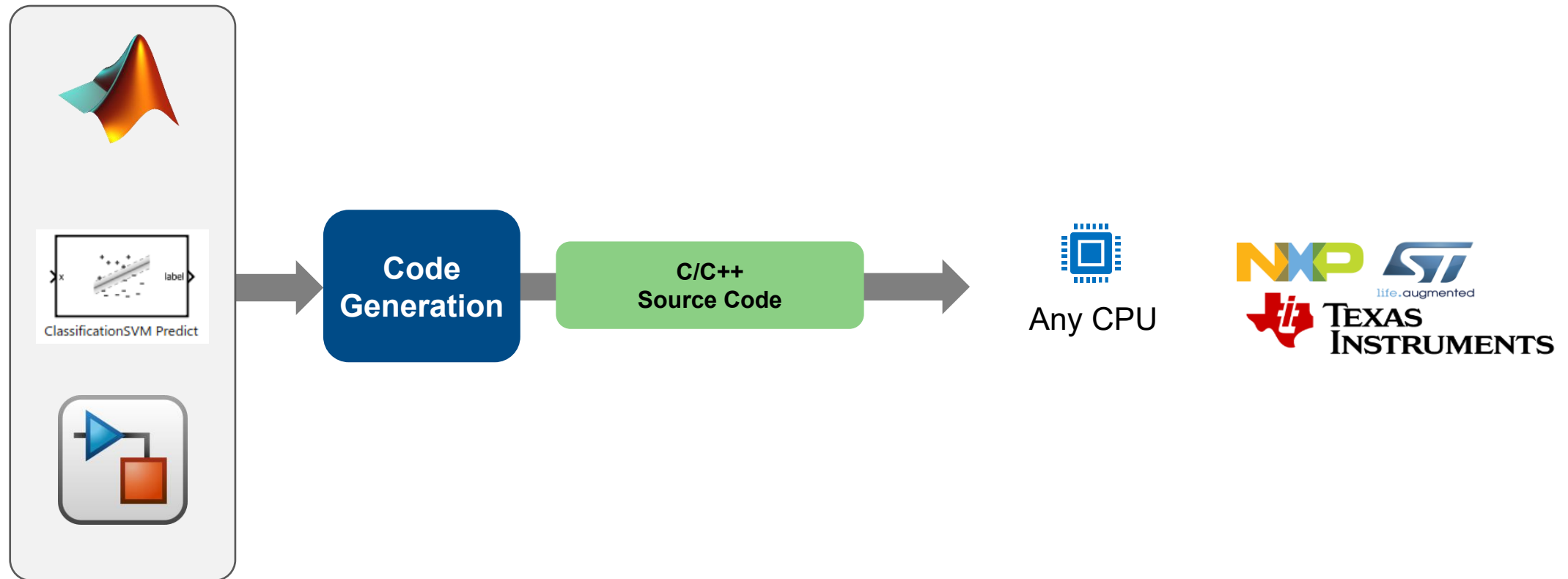
- **PIL testing and production deployment**
  - Profile code performance and evaluate trade-offs

# Deploy to target with zero coding errors

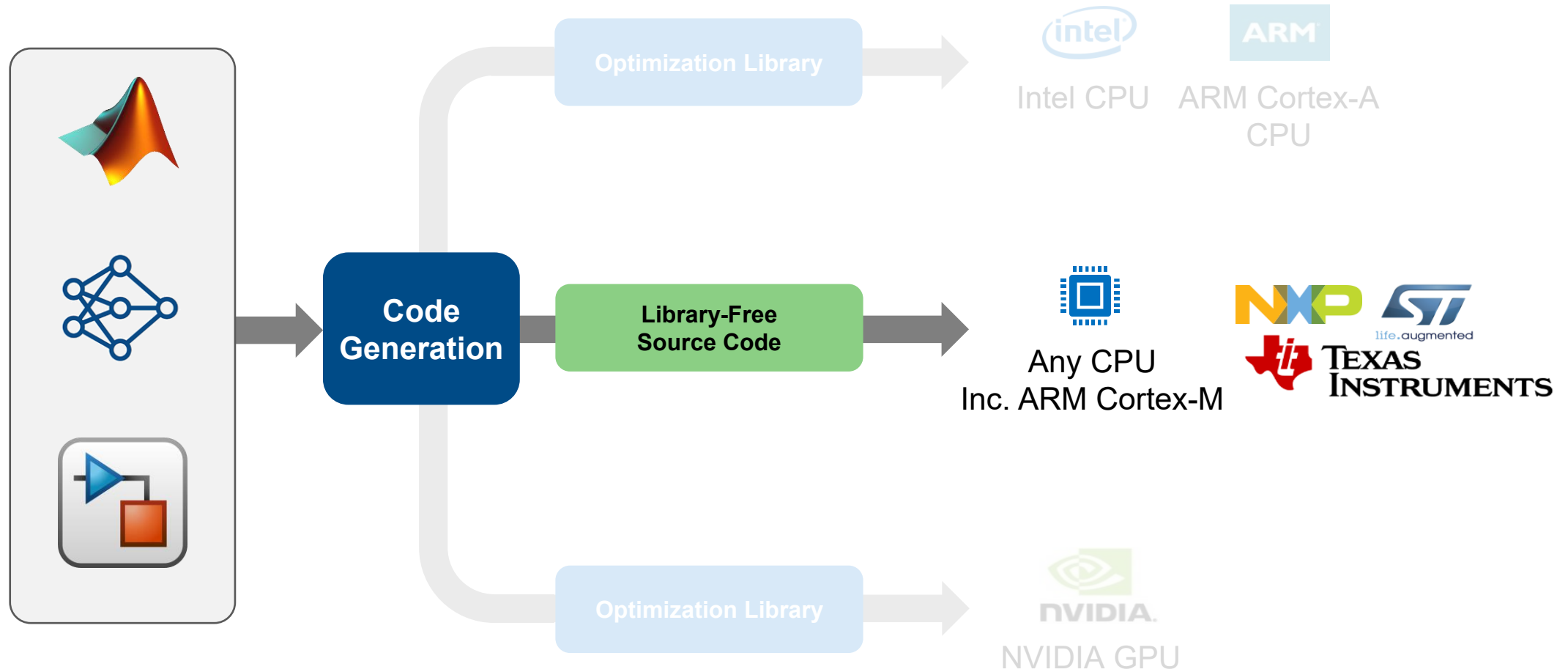




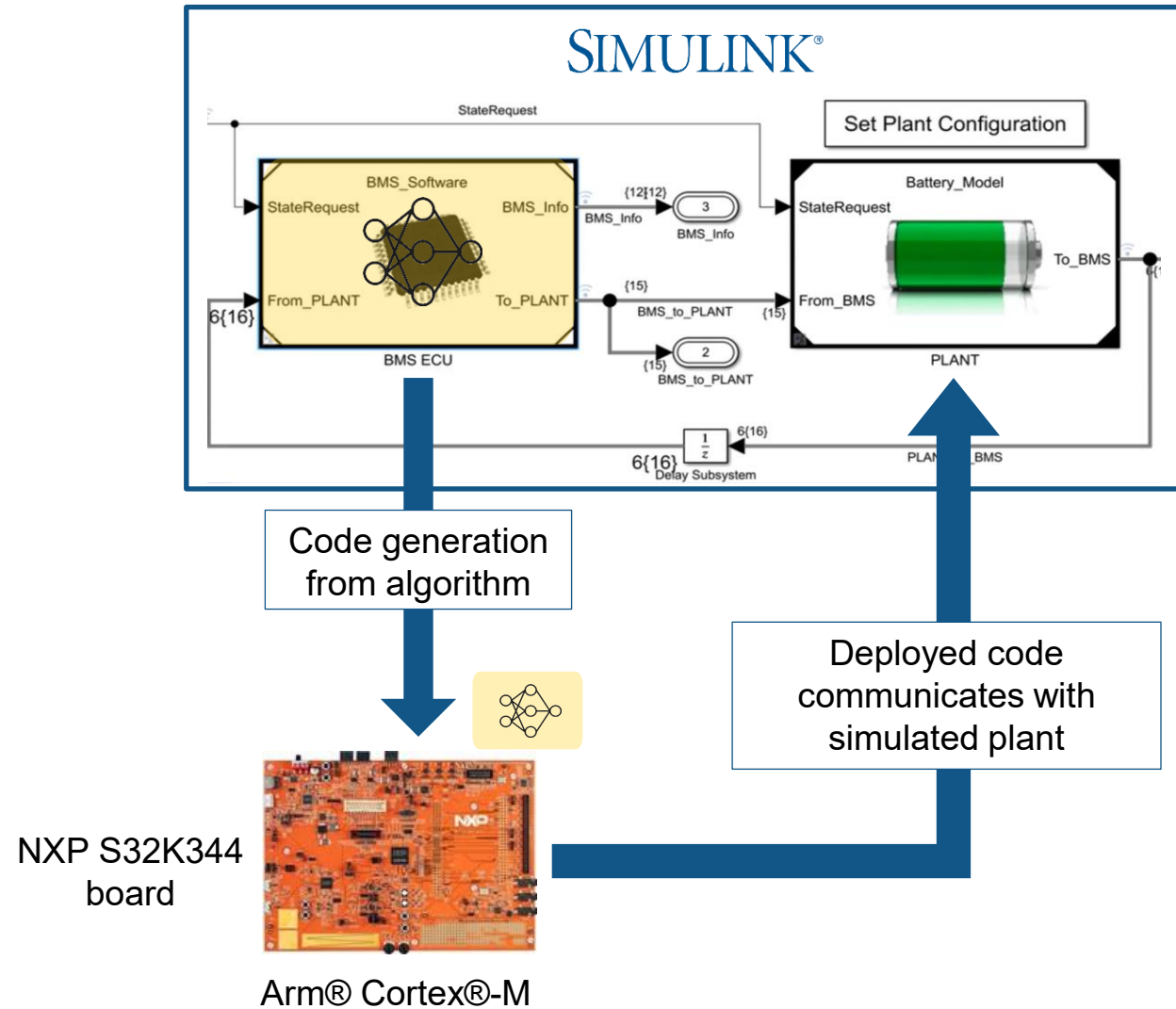
# Use Embedded Coder to generate code for machine learning



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



# Processor-in-the-loop testing on ARM Cortex-M7 processor



The image displays the hardware and software implementation of a State of Charge (SOC) estimation system. On the left, a photograph shows the S32K344 development board. The central part of the image is a screenshot of the MATLAB/Simulink 'SOC Estimation (PIL)' model. The model takes 'input' and 'measuredSOC' as inputs and outputs 'true' SOC, 'current', 'voltage', and 'temperature'. A 'Download finished' dialog box is visible, indicating the executable file has been downloaded to the S32K344 board. On the right, three plots show the 'true' and 'estim' SOC values over time, and the 'current', 'voltage', and 'temperature' signals. The bottom status bar shows the simulation is running at 99% completion.

# Manage AI tradeoffs for your system

	<b>EKF</b> Extended Kalman Filter	<b>Tree</b> Fine Regression Tree	<b>FFN</b> 1-hidden layer Feedforward Network	<b>LSTM</b> Stacked Long Short-Term Memory Network	<b>LSTM*</b> * Compressed Stacked Long Short-Term Memory Network
Preprocessing effort	●	●	●	●	●
Training Speed	N/A	●	●	●	●
Interpretability	●	●	●	●	●
Inference Speed	●	●	● ●	●	●
Model Size	●	●	●	●	●
Accuracy (RSME)	●	●	●	● ●	● ●

Results are specific to this Battery SOC Estimation example

Much Better

Better

Okay

Worse

# Key takeaways

## Data Preparation

Toolboxes for **domain-specific** pre/post-processing

## AI Modeling

**Low-code** workflow for AI Modeling through Apps

Import models from **TensorFlow**, **PyTorch** or other DL Frameworks

## Simulation & Test

**Simulink blocks for AI** models make integration easy

## Compression

**Model compression techniques** to reduce model size and speed up inference

## Deployment

**Code generation** for embedded targets (incl. library free source code for Deep Learning)

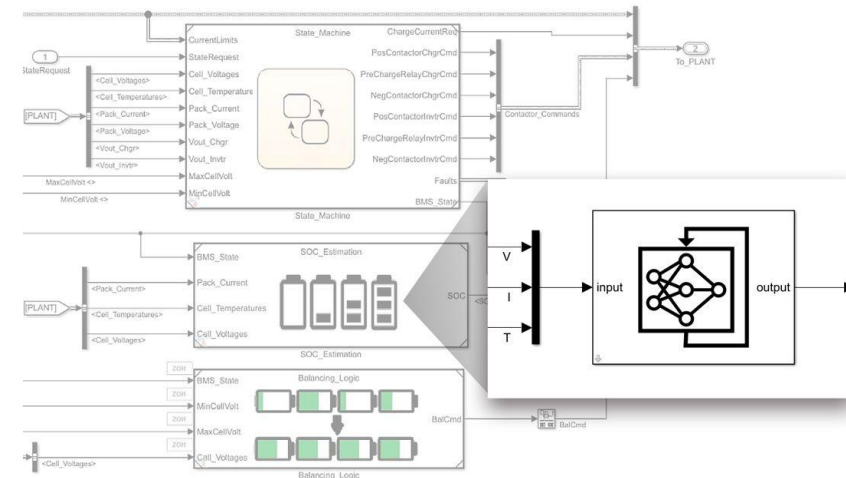
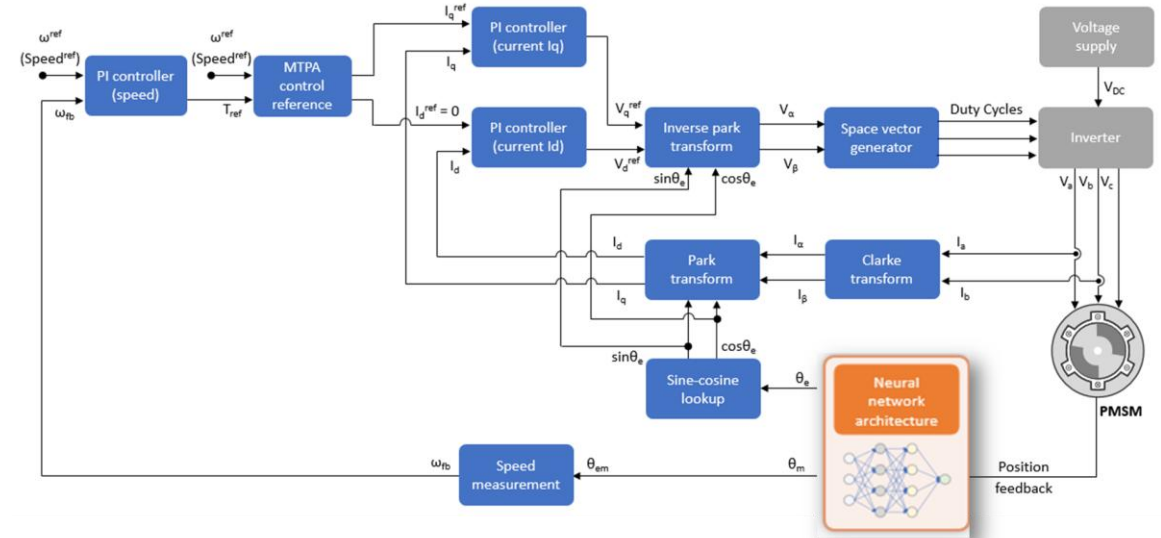
**Select and implement the optimal AI technique**

Model Accuracy



Deployment Efficiency

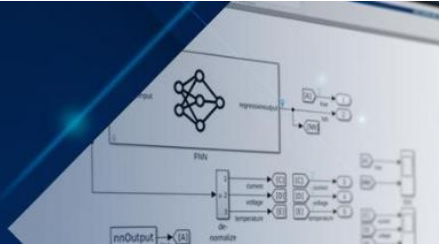
In summary, build a virtual sensor using AI and integrate into Simulink for system-level simulation and code generation



# Visit our website to see more AI for electrification resources

## AI for Electrification

Apply artificial intelligence (AI) techniques to the development and operations of electrical technologies



Scan this QR code to access the page