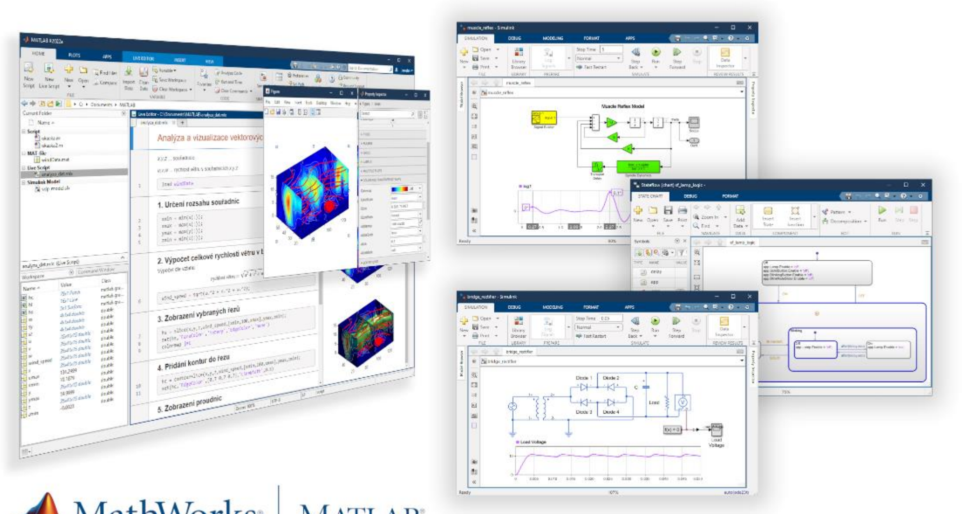
AI for Electrification



MathWorks Authorized Reseller | MATLAB & SIMULINK

Michal Blaho

blaho@humusoft.sk

www.humusoft.cz
info@humusoft.cz

www.mathworks.com

HUMUSOFT

AI models for engineered systems

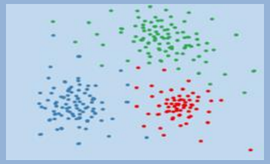
ARTIFICIAL INTELLIGENCE

Any technique that enables machines to mimic human intelligence

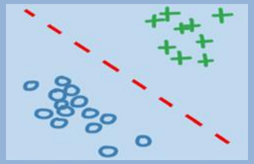
MACHINE LEARNING

Statistical methods that enable machines to “learn” tasks from data without explicitly programming

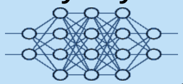
UNSUPERVISED LEARNING
(No Labeled Data)



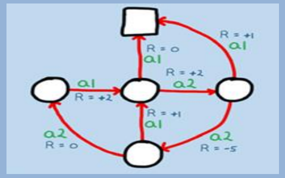
SUPERVISED LEARNING
(Labeled Data)



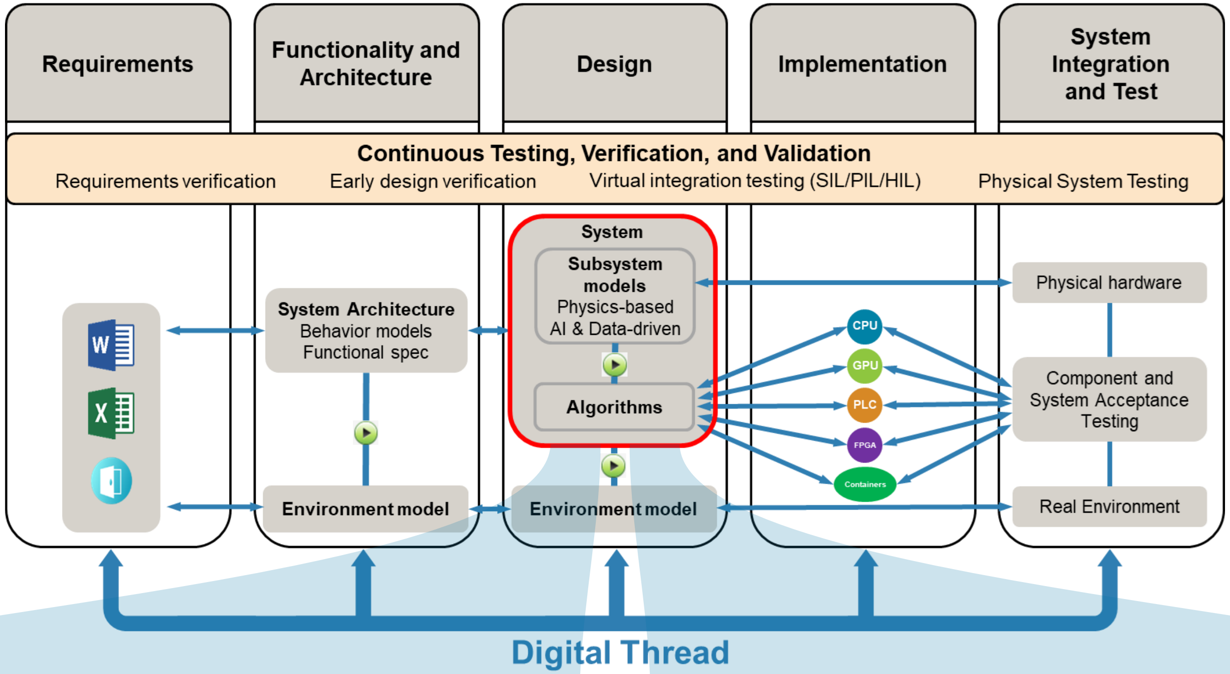
DEEP LEARNING
(Neural networks with many layers)



REINFORCEMENT LEARNING
(Interaction Data)



Integrate AI models into Model-Based Design



AI for component modeling

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

AI for algorithm development

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

AI-driven system design

Data Preparation

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

AI Modeling

- Model design and tuning
- Hardware accelerated training
- Interoperability

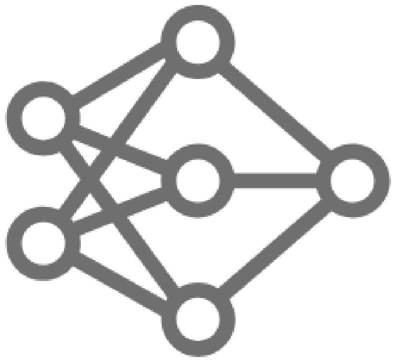
Simulation & Test

- Integration with complex systems
- System simulation
- System verification and validation

Deployment

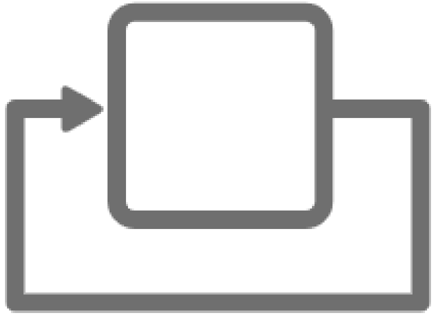
- Embedded devices
- Enterprise systems
- Edge, cloud, desktop

AI for Electrification



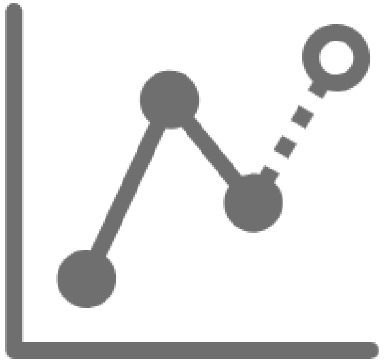
[Reduced order modeling]

[Virtual sensors]



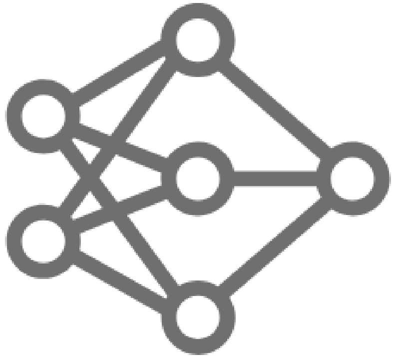
[Control strategy]

[Predictive maintenance]



[Energy forecasting]

AI for Electrification



Reduced order modeling

Virtual sensors

Control strategy

Predictive maintenance

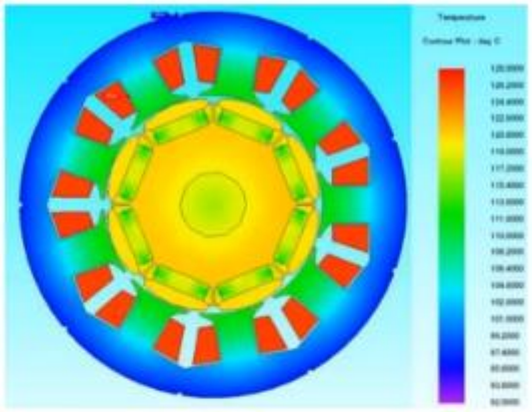
Energy forecasting

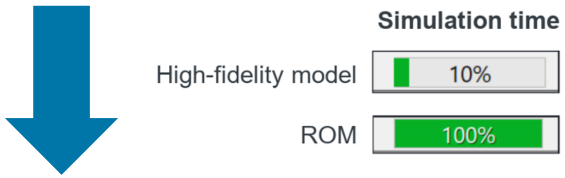
Reduced Order Modeling

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

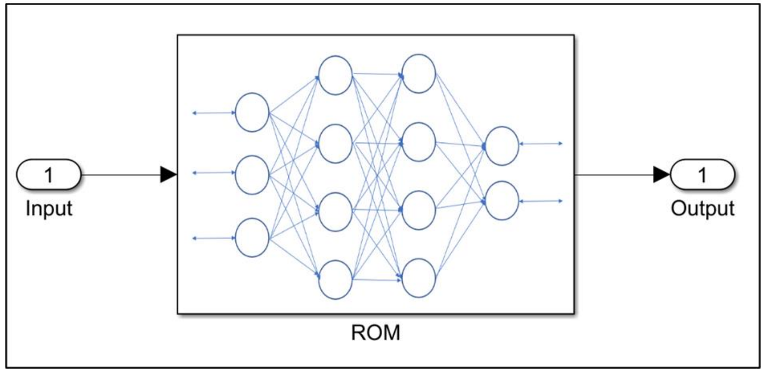
- Why
  - Enable simulation of FEA models in Simulink
  - Perform hardware-in-the-loop testing
  - Develop virtual sensor, Digital twins
  - Perform control design
  - Enable desktop simulations for order-of-magnitude longer timescales

High-fidelity model

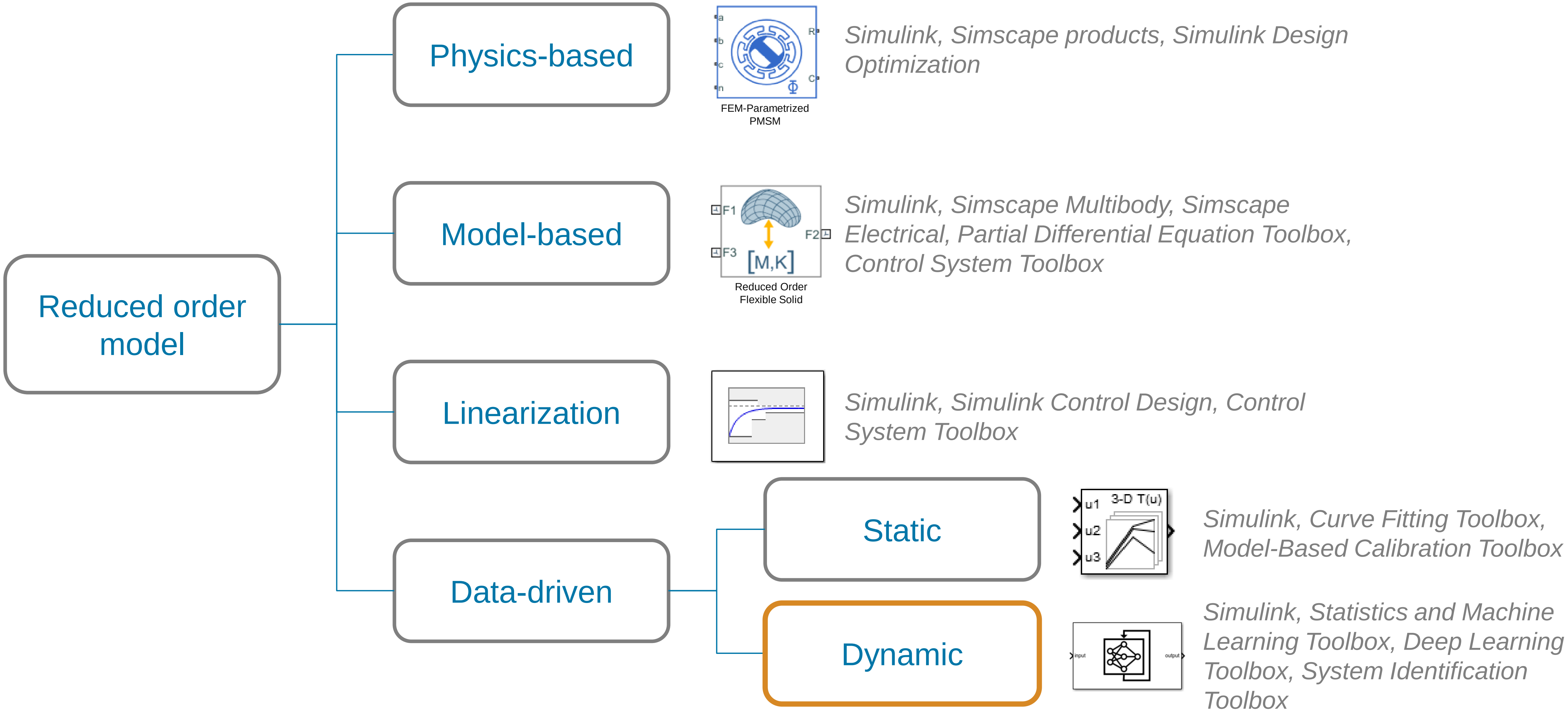




Reduced-Order Model (ROM)

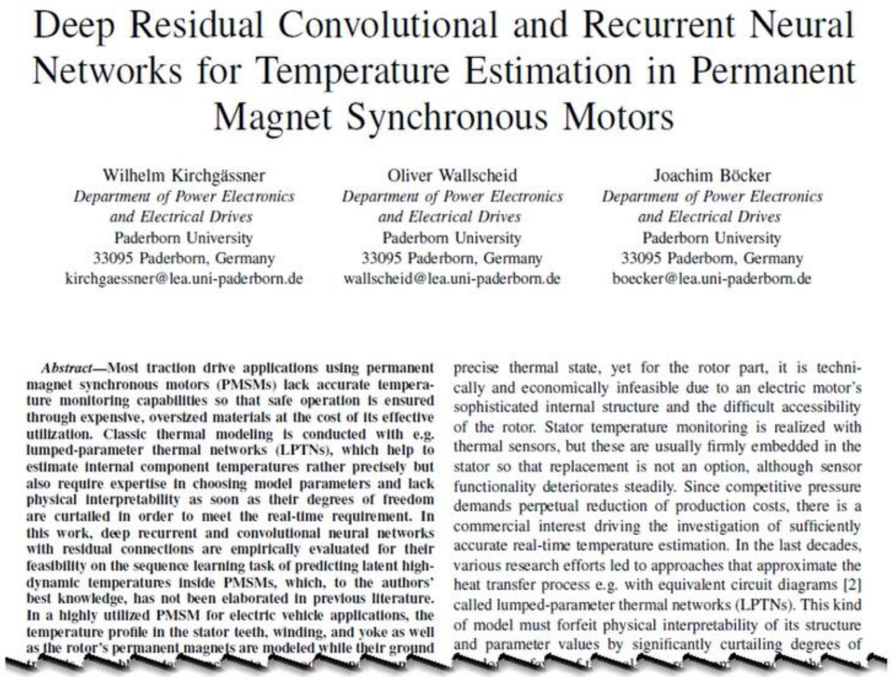


Reduced Order Modeling



Example: Temperature Excursions in motors

- Motivation
  - temperature excursions leads to loss of torque efficiency and failures
  - need to test over possible thermal regimes
  - dyno-testing is expensive, may lead to degradation
  - faster simulations are essential
- Solution
  - Reduced Order Modeling
  - Preprocess collected data
  - Create AI model
  - Use model in simulation


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

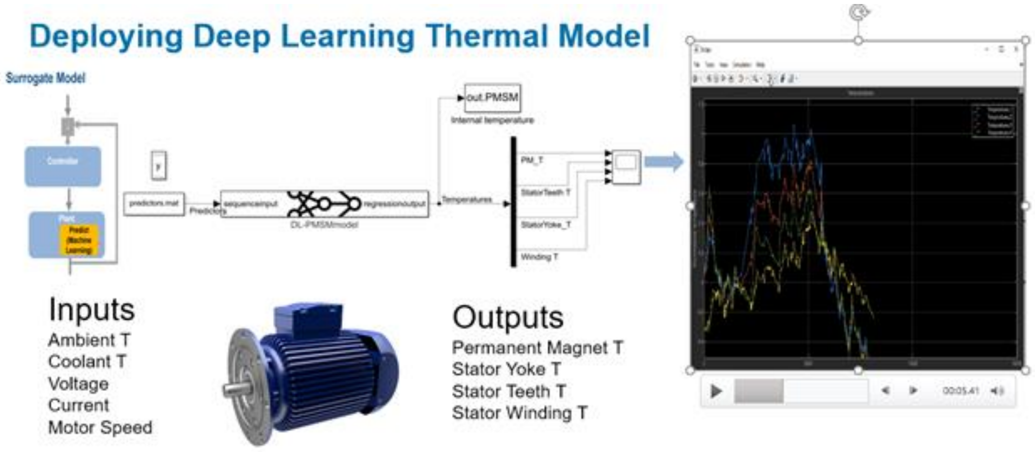
Wilhelm Kirchgässner
Department of Power Electronics and Electrical Drives
Paderborn University
33095 Paderborn, Germany
kirchgaessner@lea.uni-paderborn.de

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

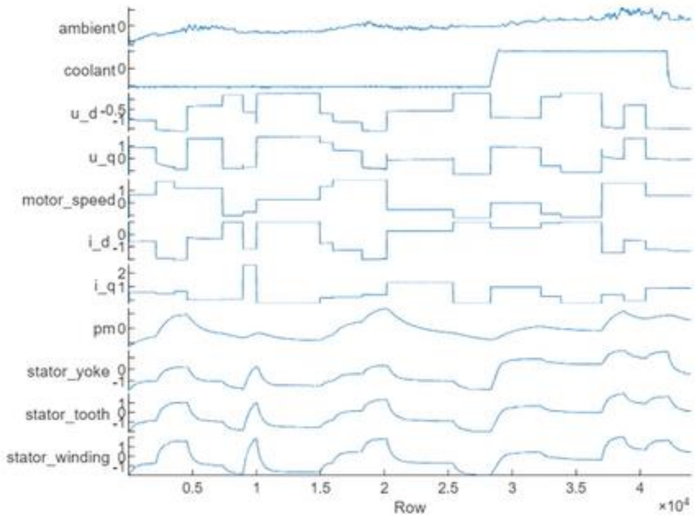
Abstract—Most traction drive applications using permanent magnet synchronous motors (PMSMs) lack accurate temperature monitoring capabilities so that safe operation is ensured through expensive, oversized materials at the cost of its effective utilization. Classic thermal modeling is conducted with e.g. 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

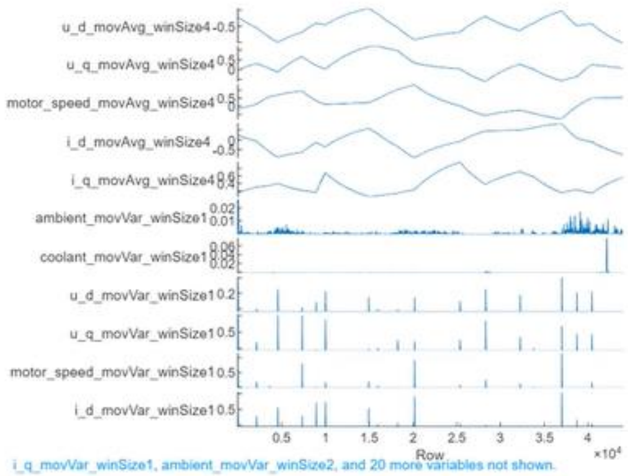


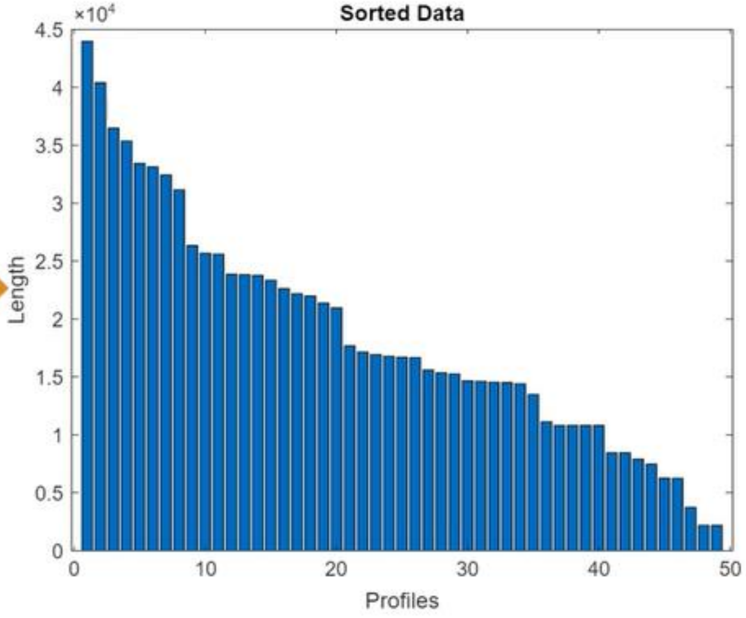
Data Preparation

Raw CSV Data



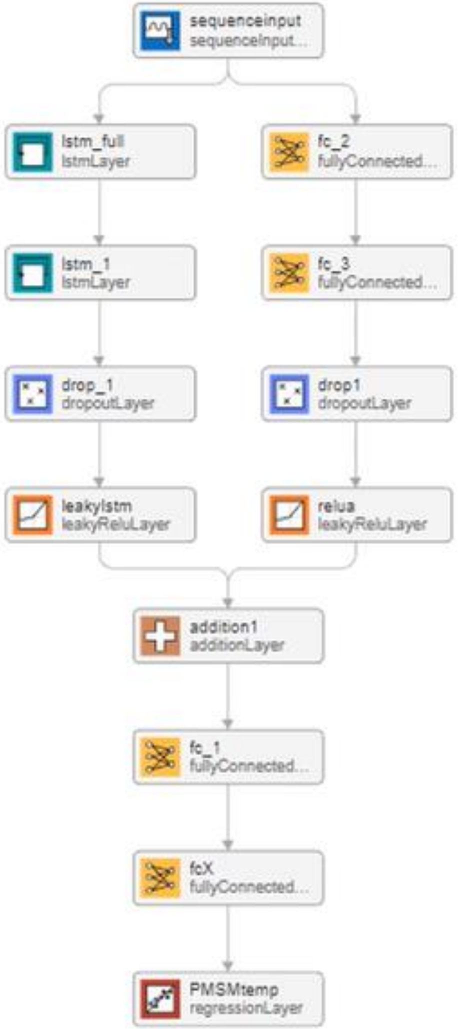
Additional Features





- 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

Model Development

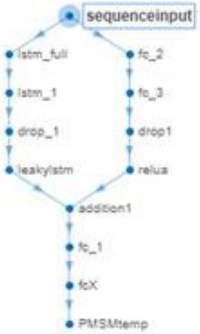


Analysis for training in Deep Network Designer

Name: Network from Deep Network Designer

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

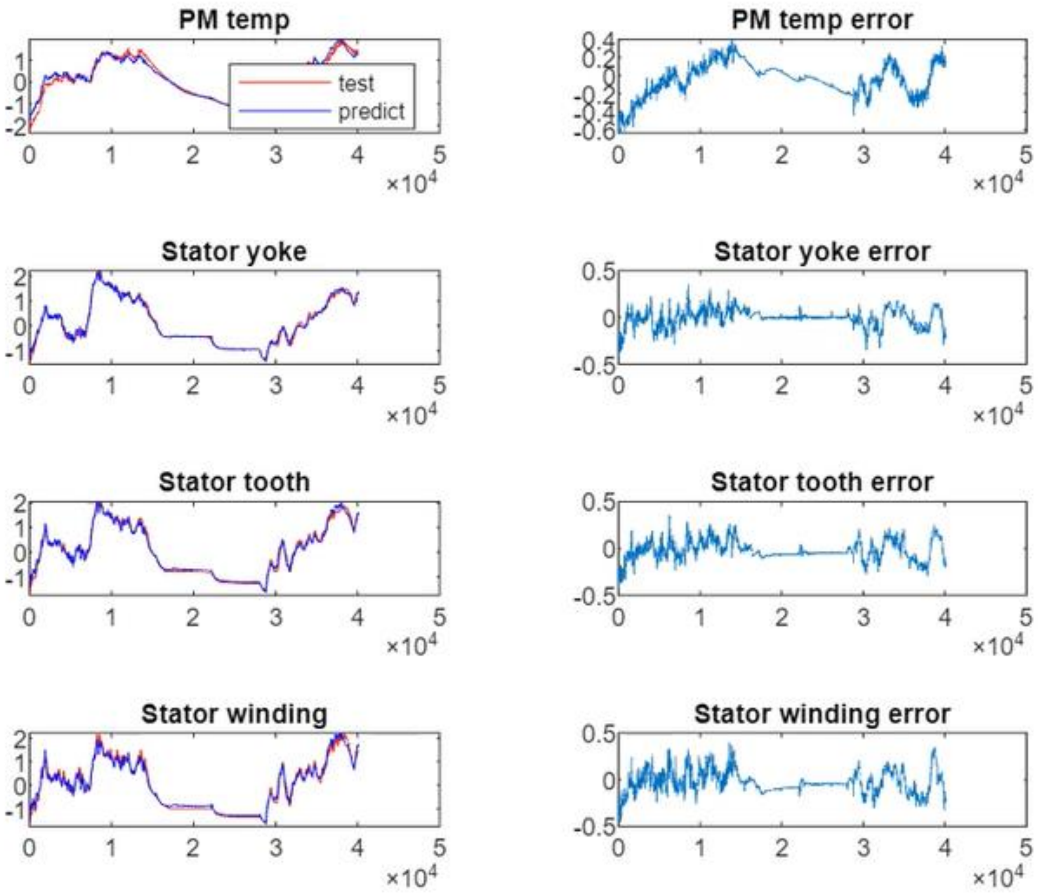
2M total learnables | 13 layers | 0 warnings | 0 errors

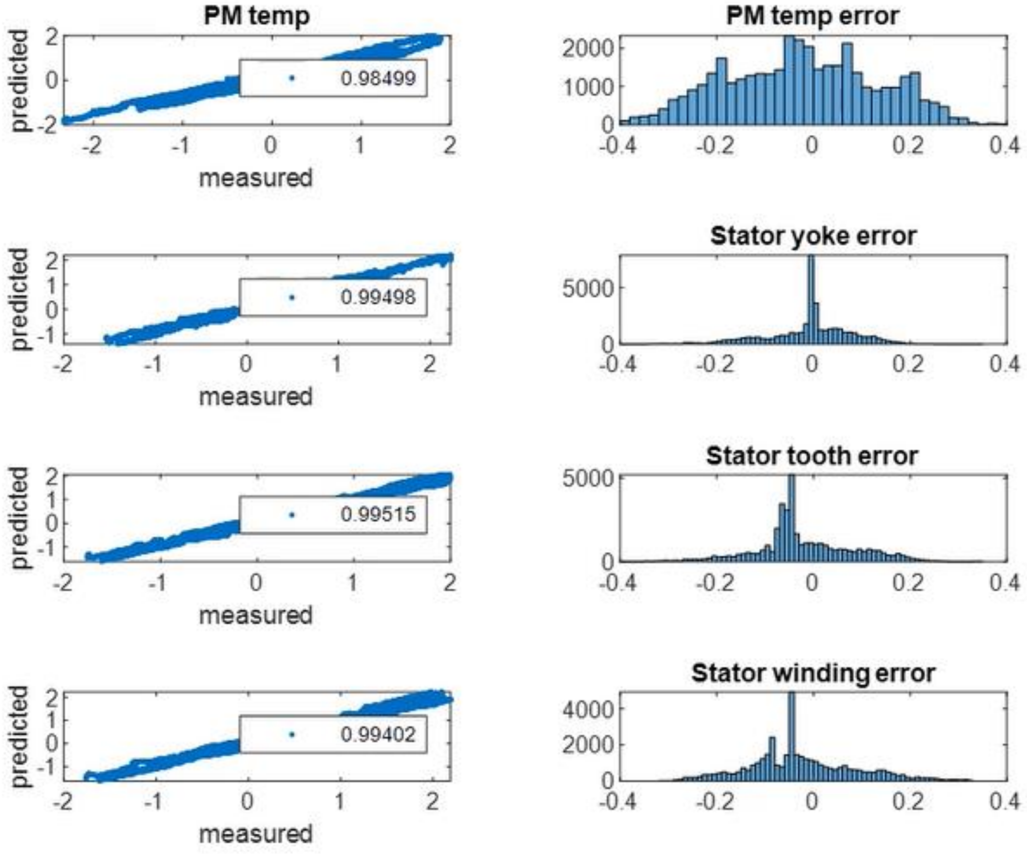


ANALYSIS RESULT

	Name	Type	Activations	Learnable Proper...	Stat
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)	-	-

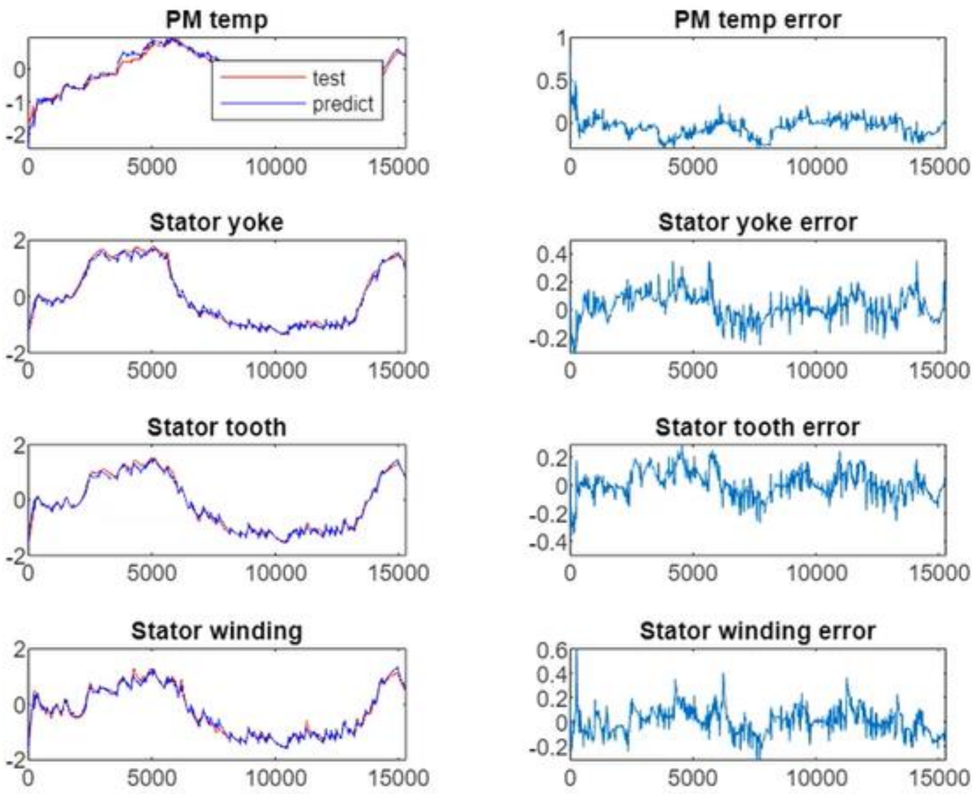
Testing on a long profile

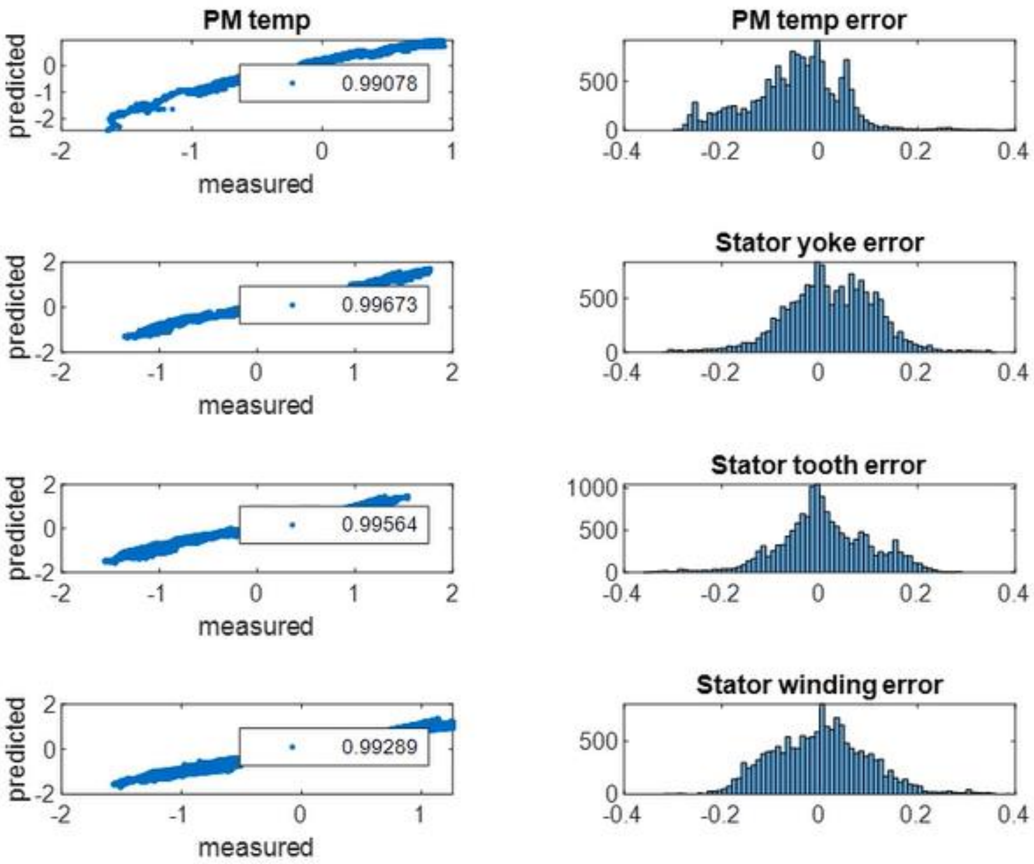




- All correlation values are about 0.99 and error distribution is unbiased hence model captures trend and Magnitude

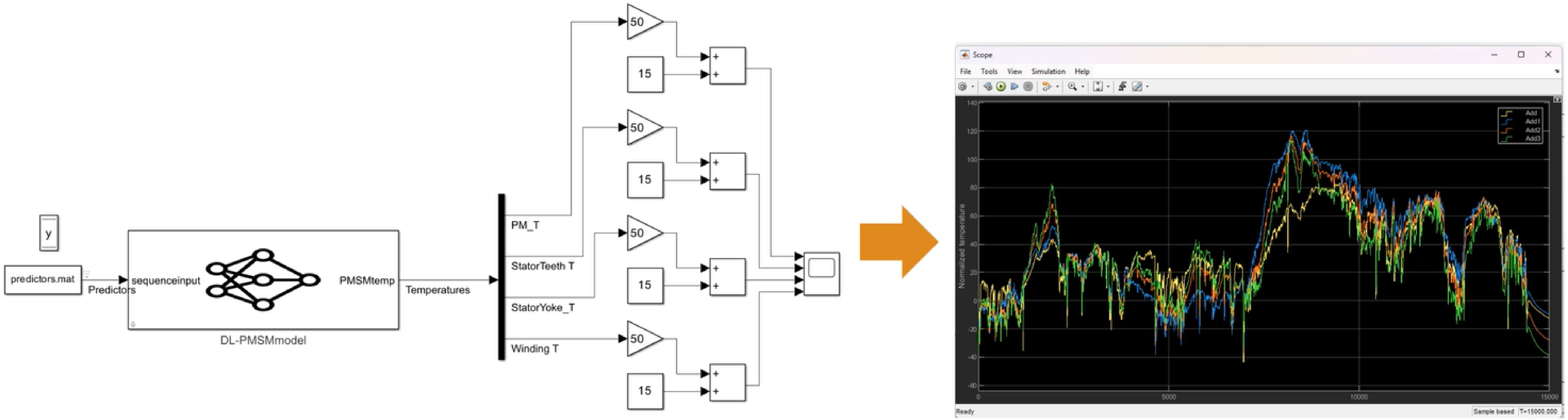
Testing on a short profile





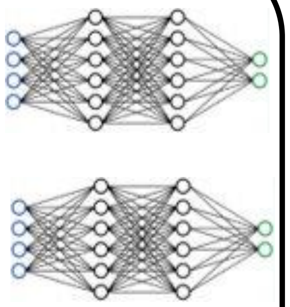
- All correlation values are about 0.99 and error distribution is unbiased hence model captures trend and Magnitude

Deployment to Simulink

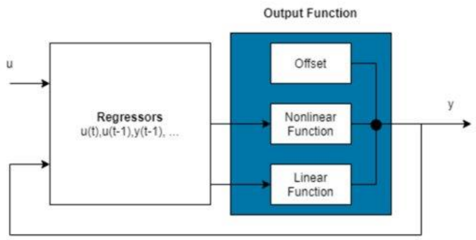


Nonlinear Modeling Capabilities

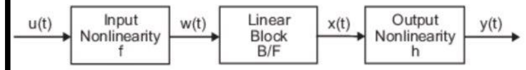
$$\begin{cases} \dot{x} = f(x, u) \\ y = g(x, u) \end{cases}$$



Neural State-Space



Nonlinear ARX



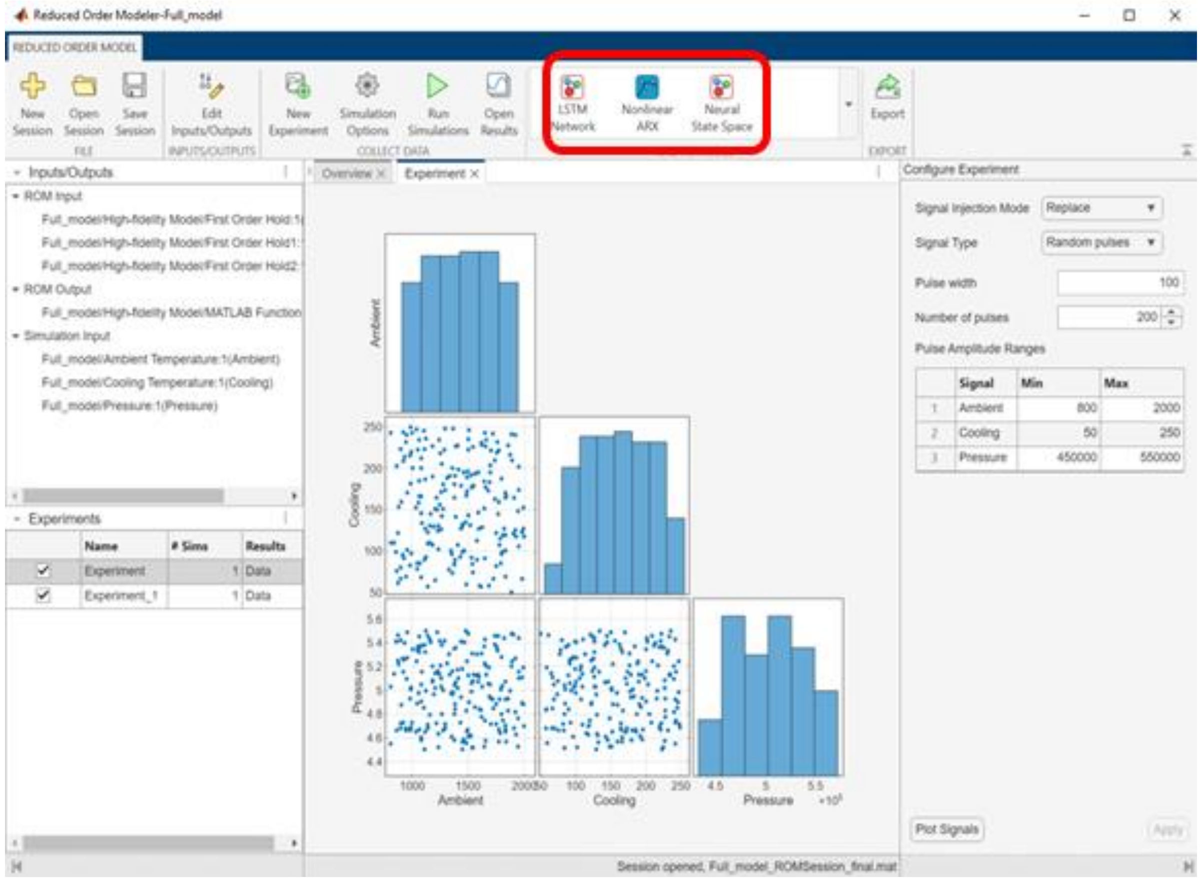
Hammerstein Wiener

Leverage AI techniques without being experts in field of AI

Combine insights and knowledge of physics of your system with AI techniques

Reduced Order Modeling Support Package

- Set up the design of experiments and generate input-output training
- Train and compare AI-based reduced order models using pre-configured templates
- Export AI-based surrogate models to Simulink for system-level simulation, control design, and HIL testing
- Export reduced order models as Functional Mockup Units (FMUs) for use outside of MATLAB and Simulink (with Simulink Compiler)



AI for Electrification

Reduced order modeling

Virtual sensors

Control strategy

Predictive maintenance

Energy forecasting

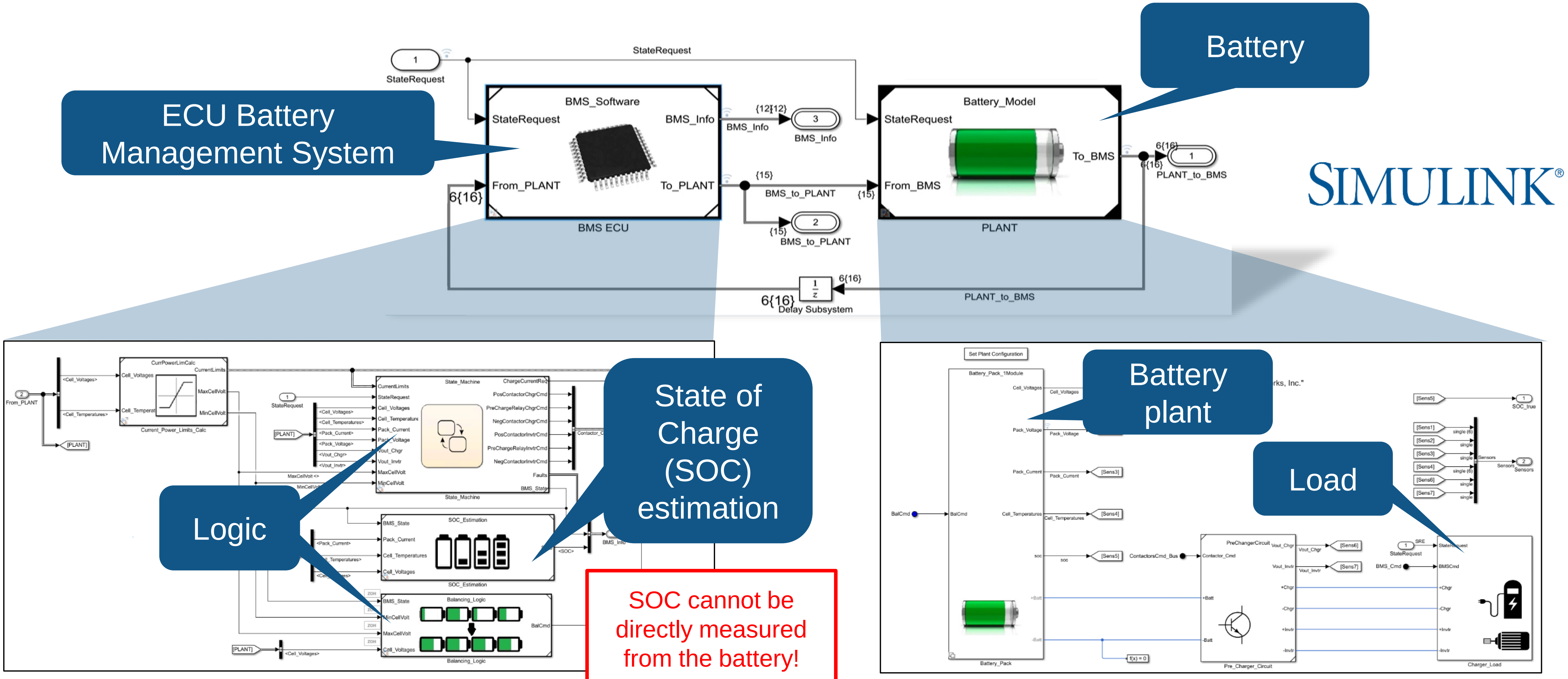
Virtual Sensor

- Mimics a physical sensor using data from other measurements
  - sometimes also called a Soft sensor

Why are Virtual Sensors relevant?

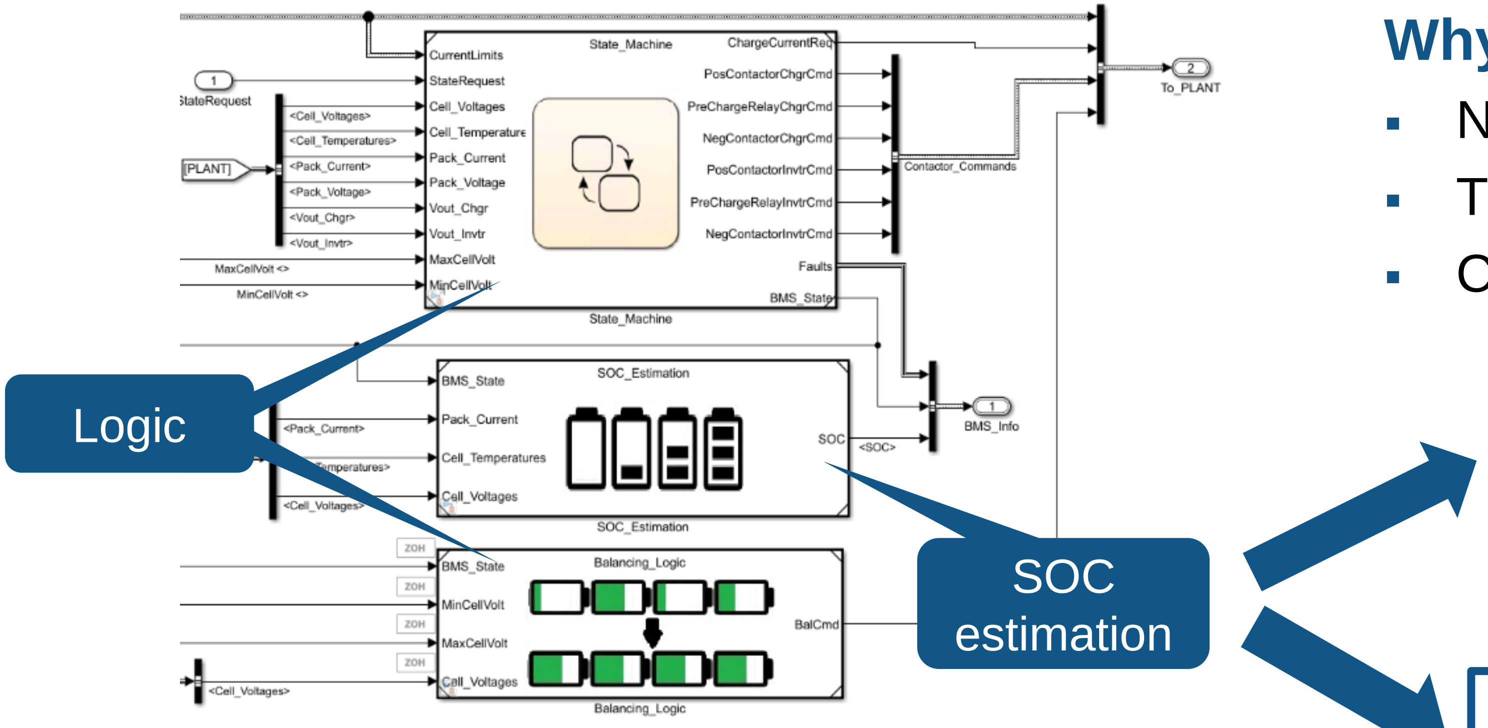
- A physical sensor may be:
  - Expensive
  - Slow
  - Noisy
  - Unreliable
  - Not feasible
  - Degrading over time
  - Requiring redundancy
  - etc.

Physical sensor might not be feasible



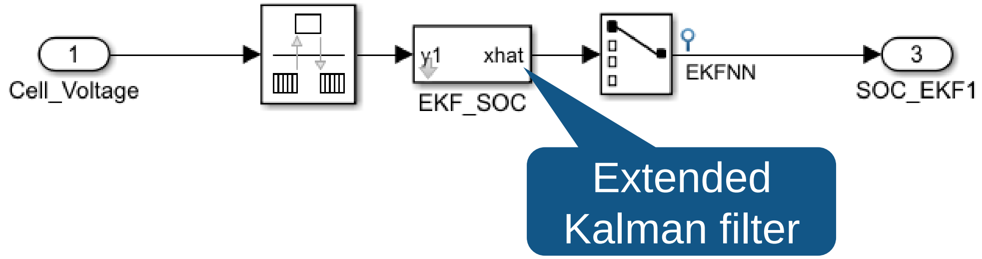
SOC cannot be directly measured from the battery!

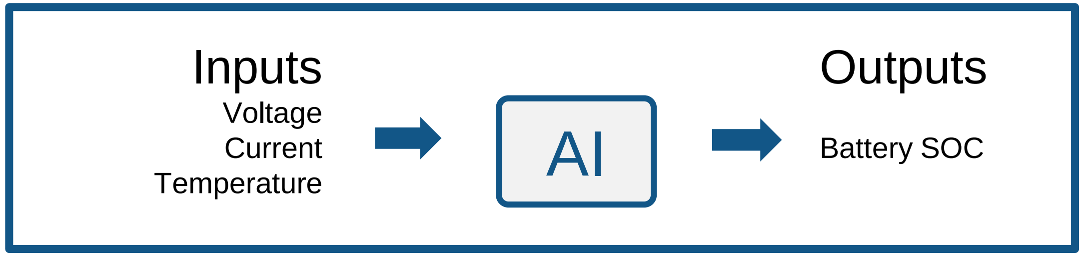
Virtual sensor for Battery State of Charge (SOC) estimation



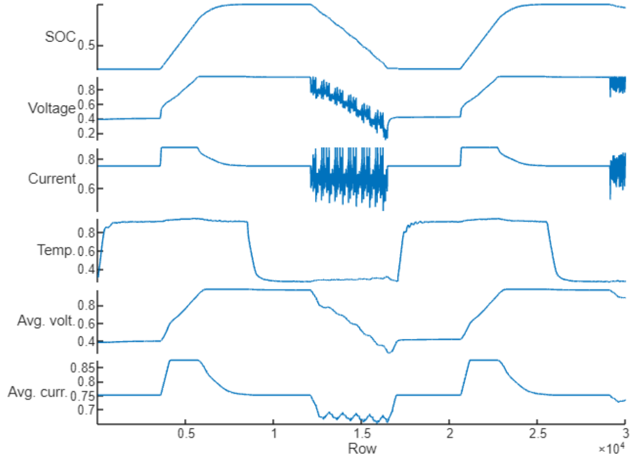
Why AI over Kalman filtering?

- No need of an internal battery model
- Training directly on measured data
- Capture very complex data relationships

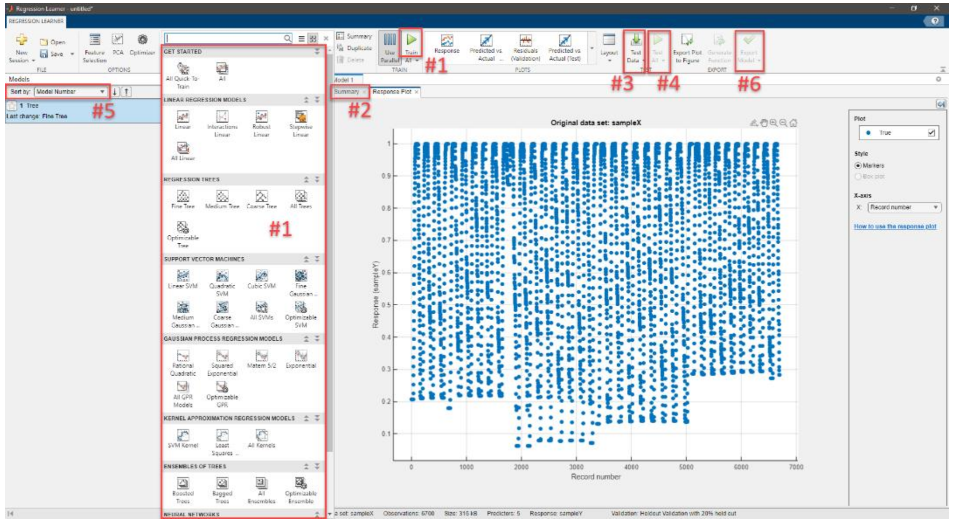




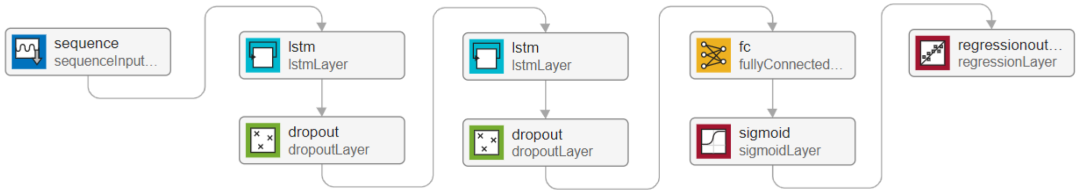
Example: State of Charge estimation



- Task
  - create a virtual sensor model
  - test multiple machine learning models
  - test LSTM network

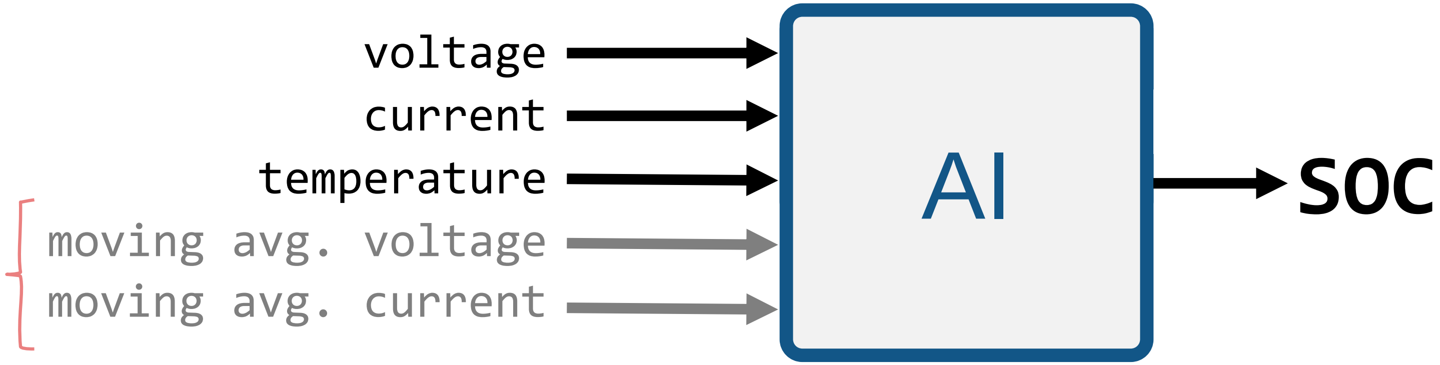


- Solution
  - loading and partitioning of data
  - creation of an AI model
  - model training
  - evaluate the model's accuracy



Data Preparation

Data source: McMaster University*



Computed over a 500 sec window

Efficient labeling in the lab (Coulomb Counting)

$$SOC(t) = \frac{1}{C}\int_0^t I(p)dp$$

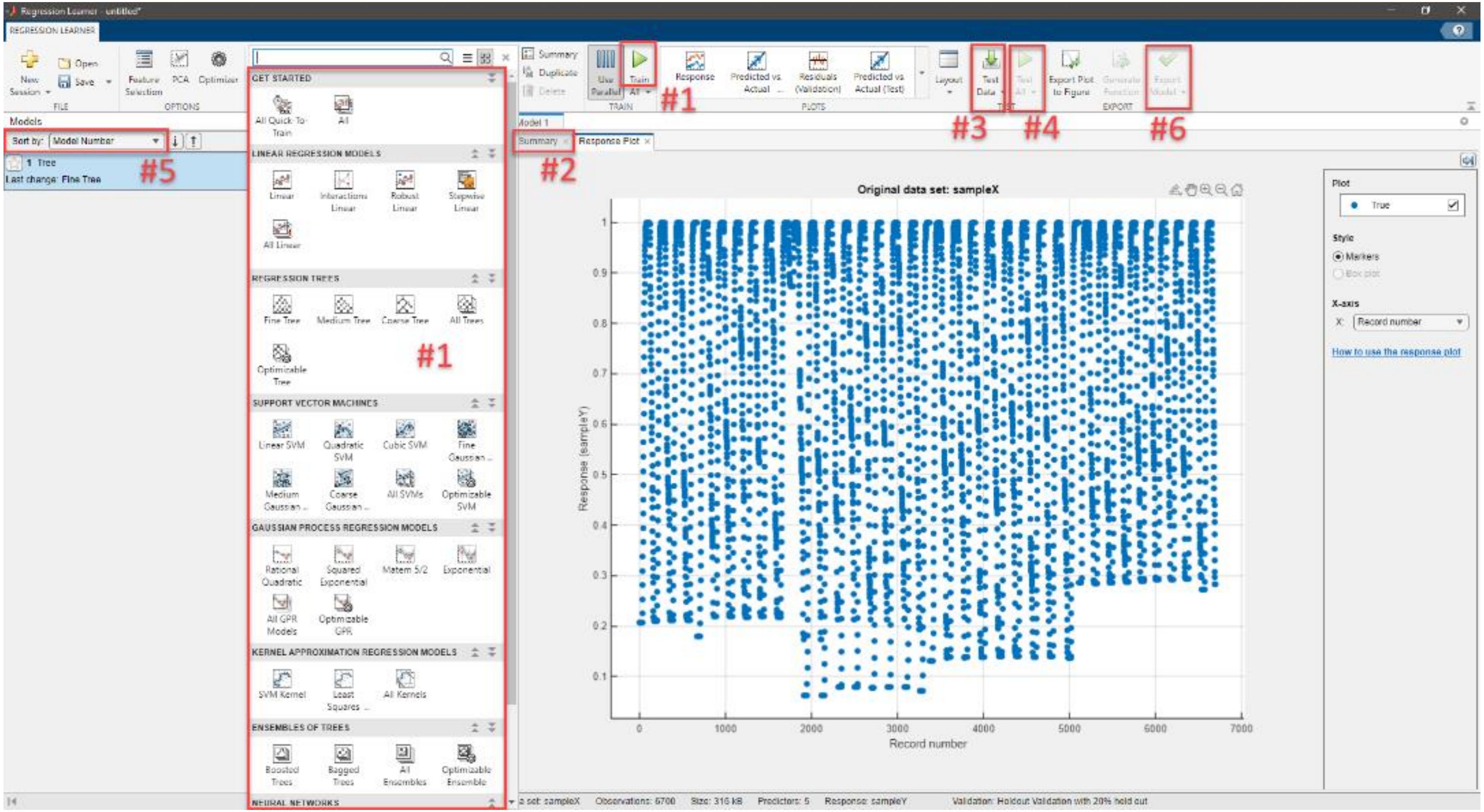
Predictors

Response

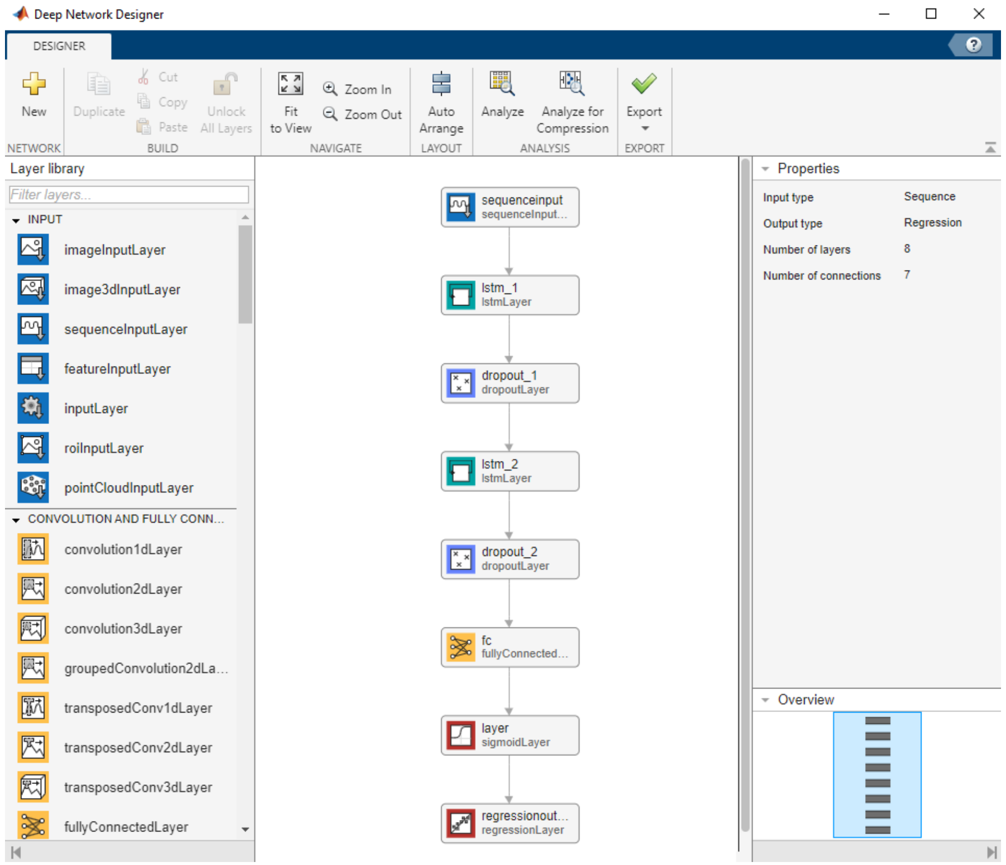
	1 Voltage	2 Current	3 Temperature	4 Moving Average Voltage	5 Moving Average Current	6 SOC
1	0.7510	0.3851	0.3031	0.7510	0.3851	0.2064
2	0.7510	0.3852	0.3046	0.7510	0.3851	0.2064
3	0.7510	0.3852	0.3061	0.7510	0.3852	0.2064
4	0.7510	0.3852	0.3076	0.7510	0.3852	0.2064
5	0.7510	0.3852	0.3091	0.7510	0.3852	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

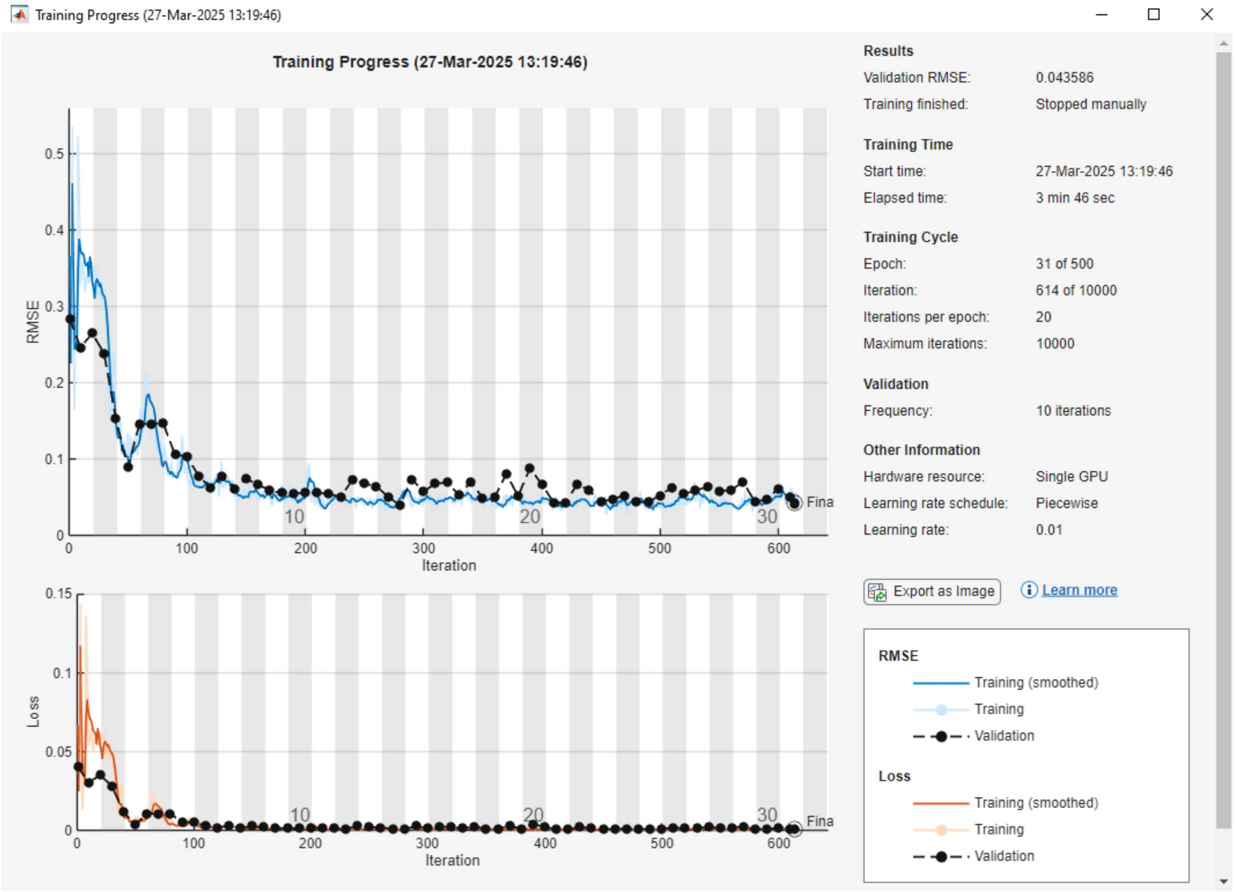
* https://data.mendeley.com/datasets/cp3473x7xv/3

Train model with Regression Learner App

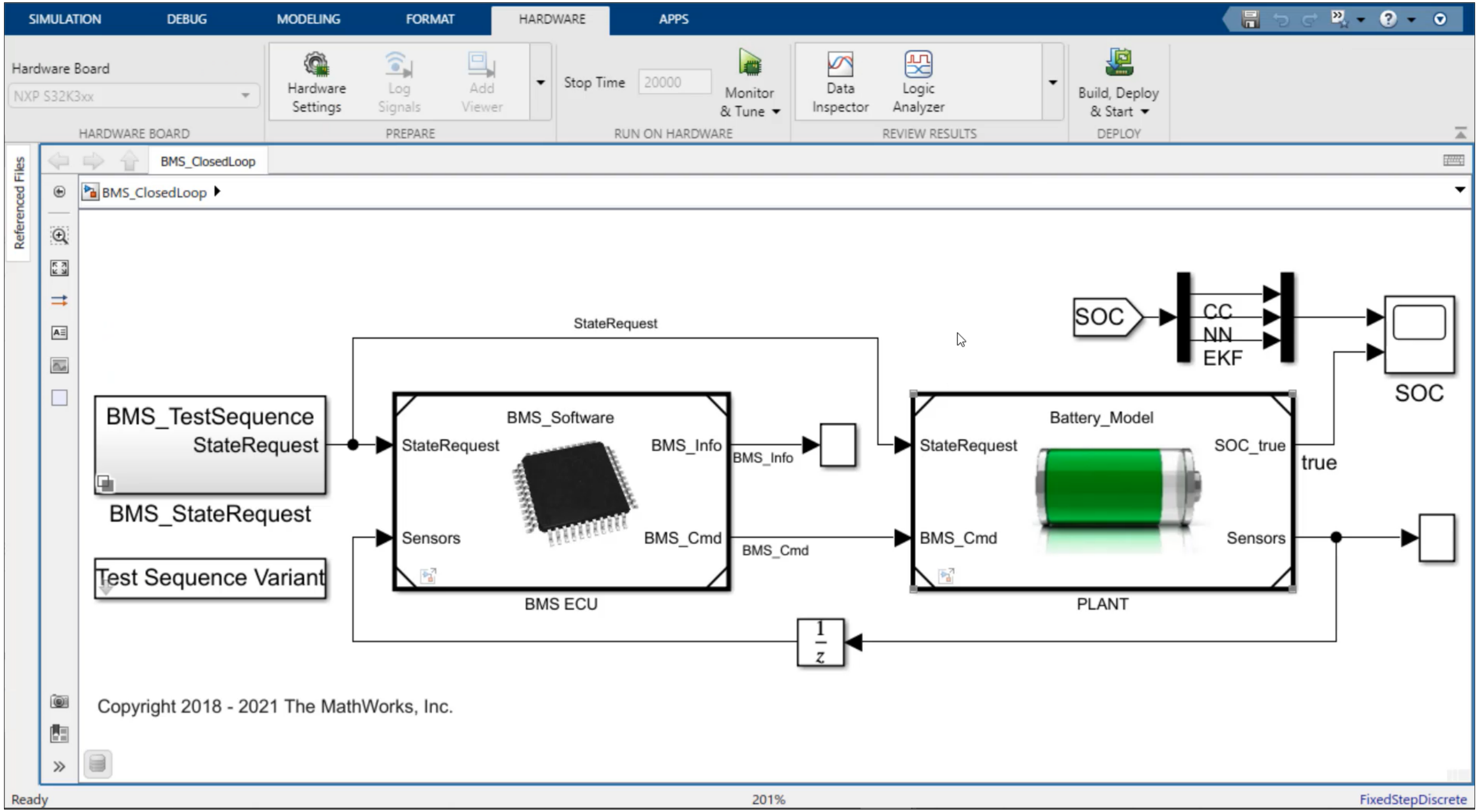


Train model with Deep Network Designer





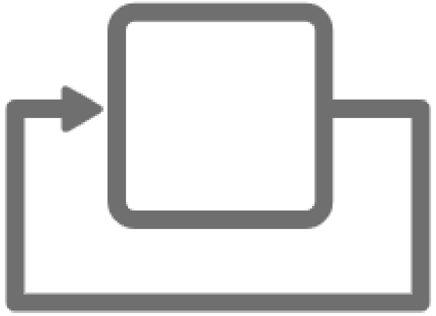
Closed-Loop System-Level Simulation



AI for Electrification

Reduced order modeling

Virtual sensors



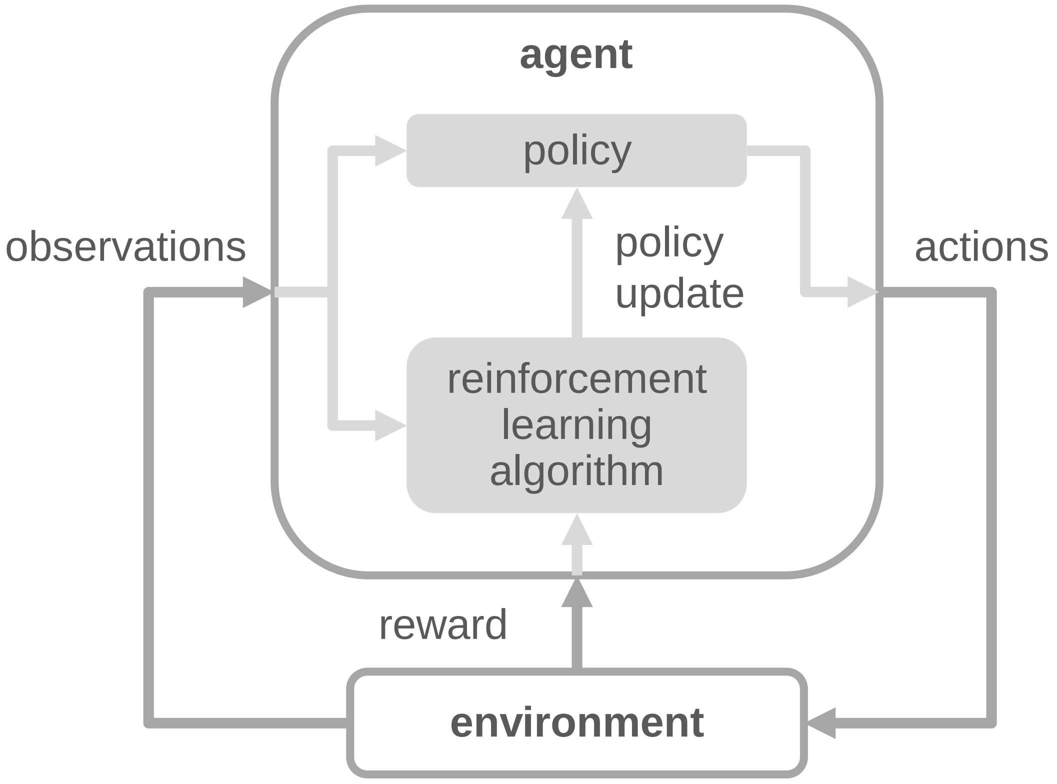
Control strategy

Predictive maintenance

Energy forecasting

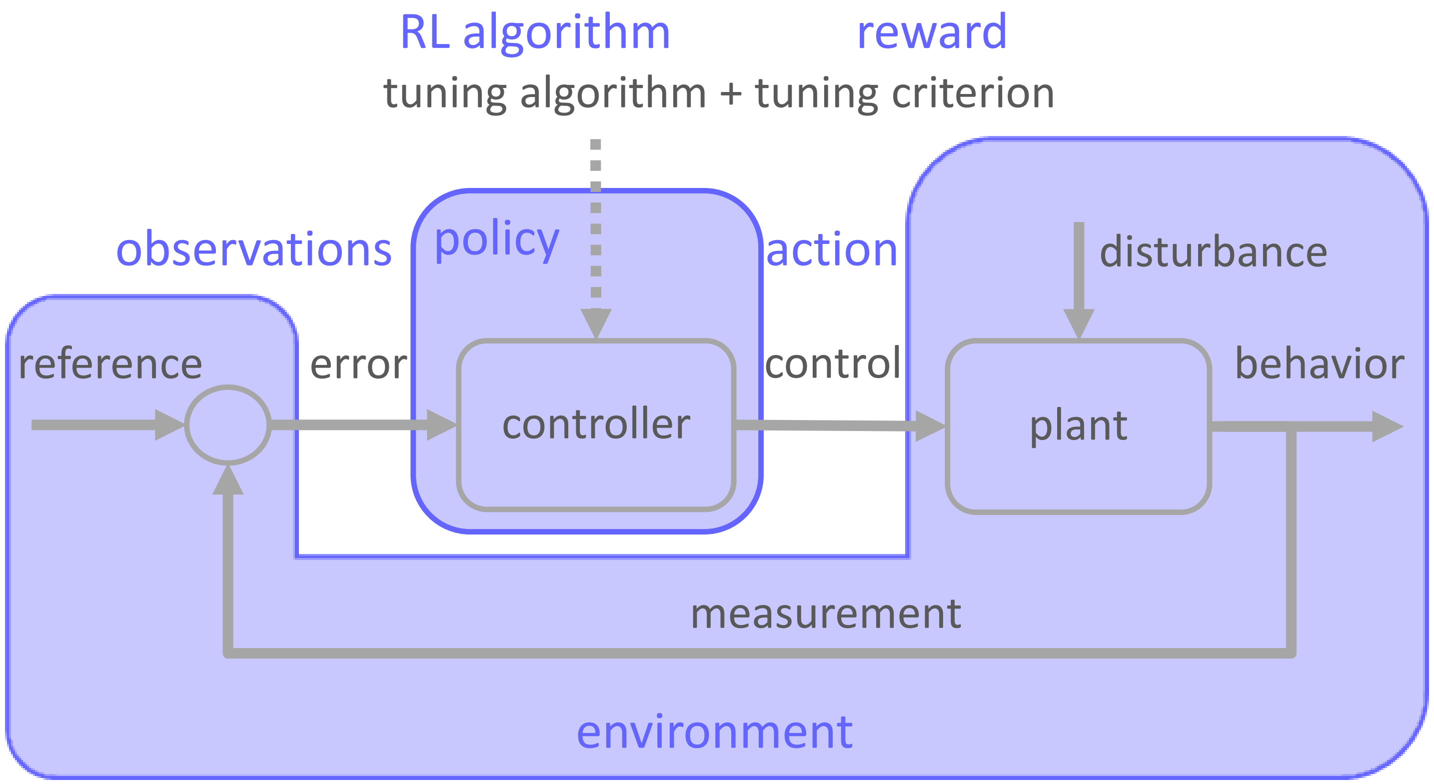
Reinforcement learning

- How it works
  - computer agent learns optimal behavior through repeated interactions with a dynamic environment
- Goal
  - maximize reward in the long term
- Policy
  - deep neural network (most common)
  - control system, decision-making algorithm
- Use
  - where traditional methods are difficult to formulate
  - for difficult to interpret signals (e.g. image)

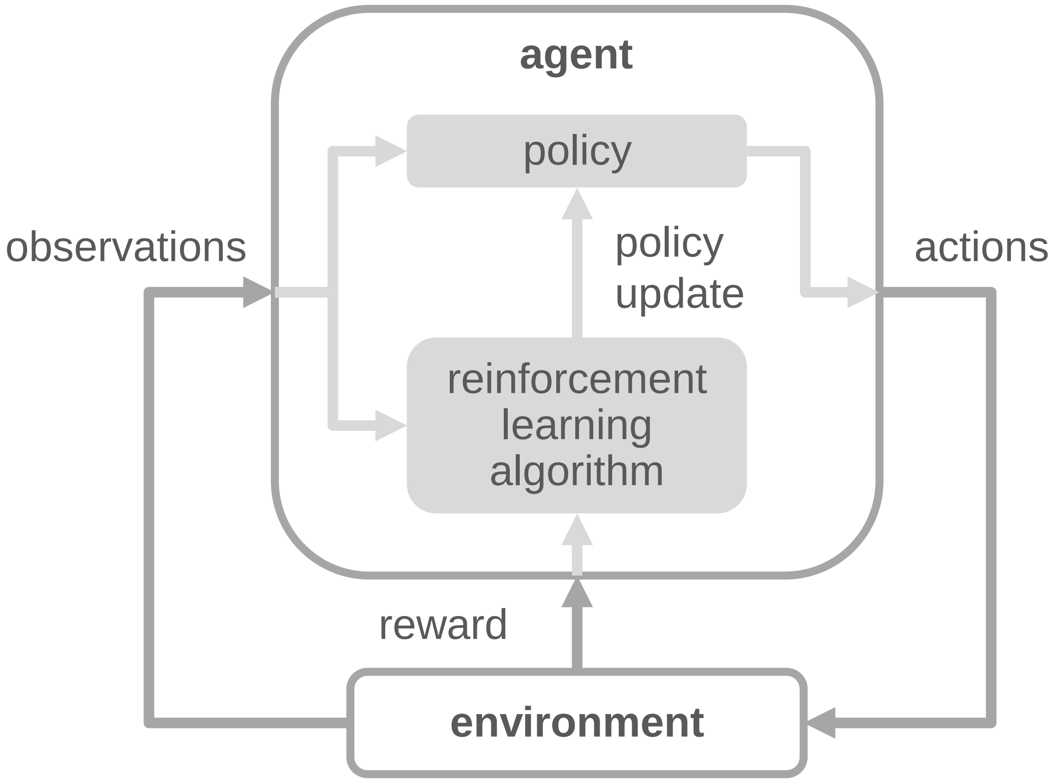


Reinforcement Learning vs Controls

Control system



Reinforcement learning system



Reinforcement Learning Workflow

- Simulation models or real hardware
- Virtual models are safer and cheaper

- Deep network? Table? Polynomial?

- Select training algorithm
- Tune hyperparameters

- Trained policy is a standalone function

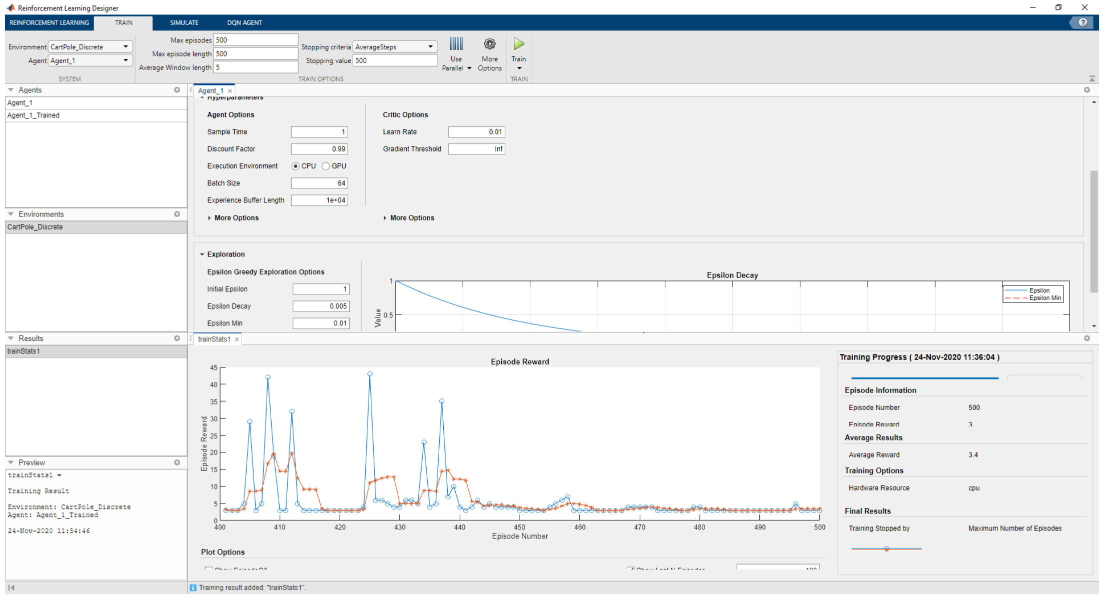
Environment **Reward** **Policy representation** **Agent** **Training** **Deployment**

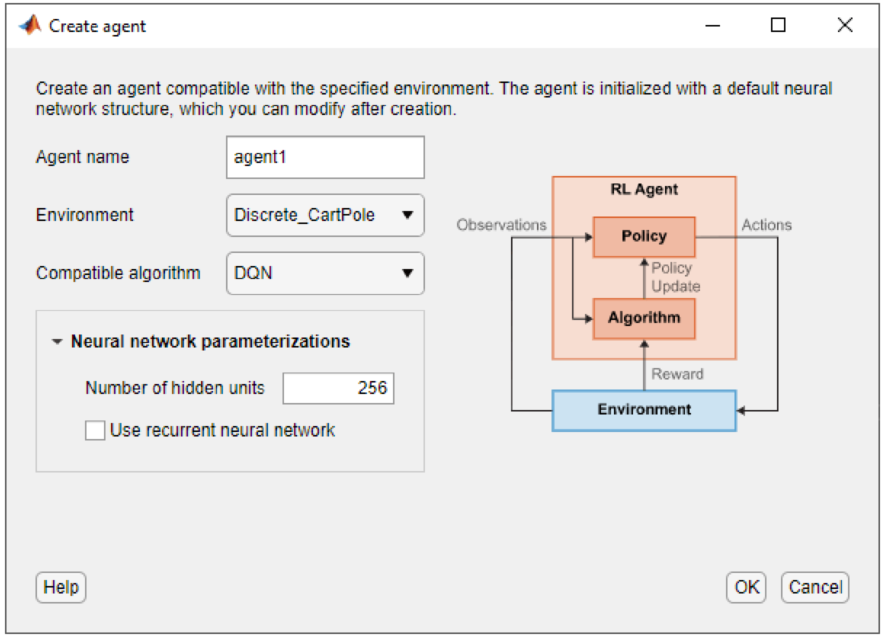
- Numerical value that evaluates goodness policy
- Reward shaping can be challenging

- Large number of simulations needed
- Parallel & GPU computing can speed up training
- Training could still take hours or days

Reinforcement Learning Designer

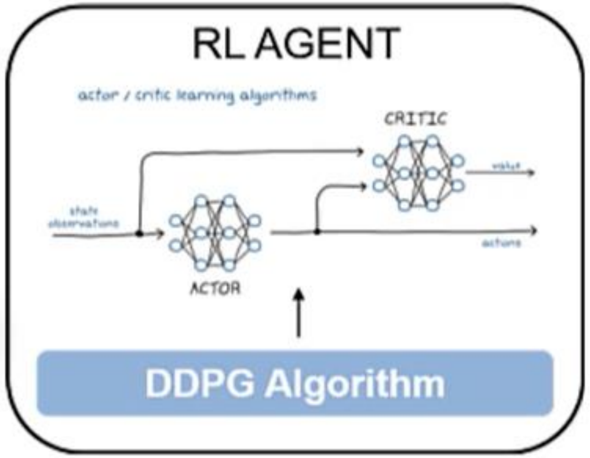
- Design, train, and simulate agents using a visual interactive workflow
- Import existing MATLAB/Simulink environments or create a predefined one
- Create or import agents
- Train and simulate the agent in the app
- Analyze simulation results and refine agent parameters
- Export the final agent to the MATLAB workspace for further use and deployment



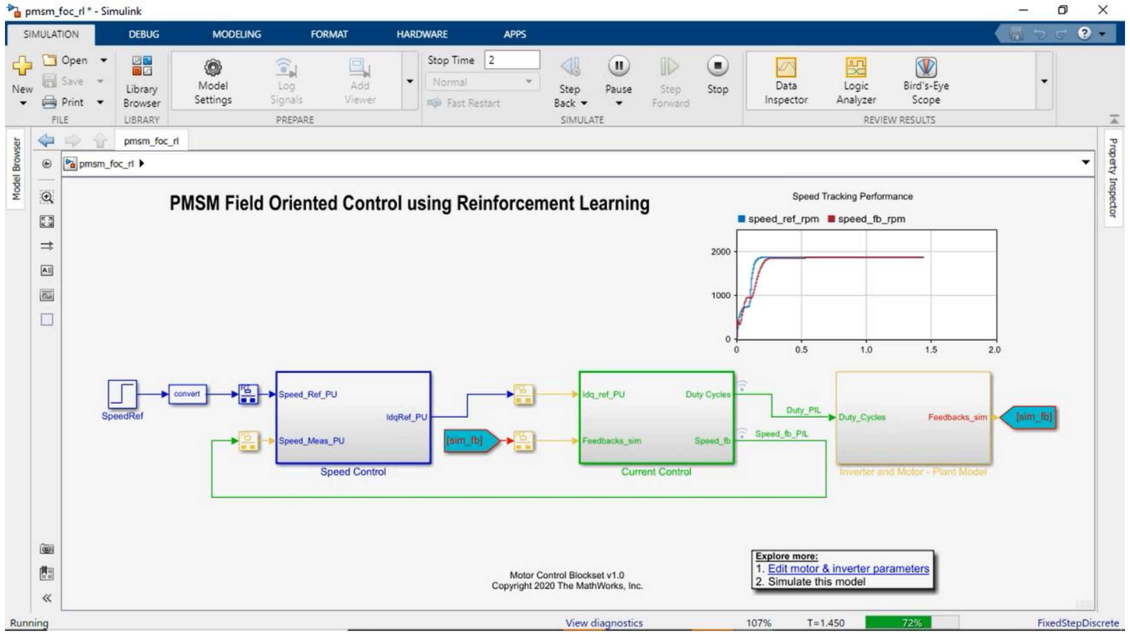


Example: FOC with Reinforcement Learning

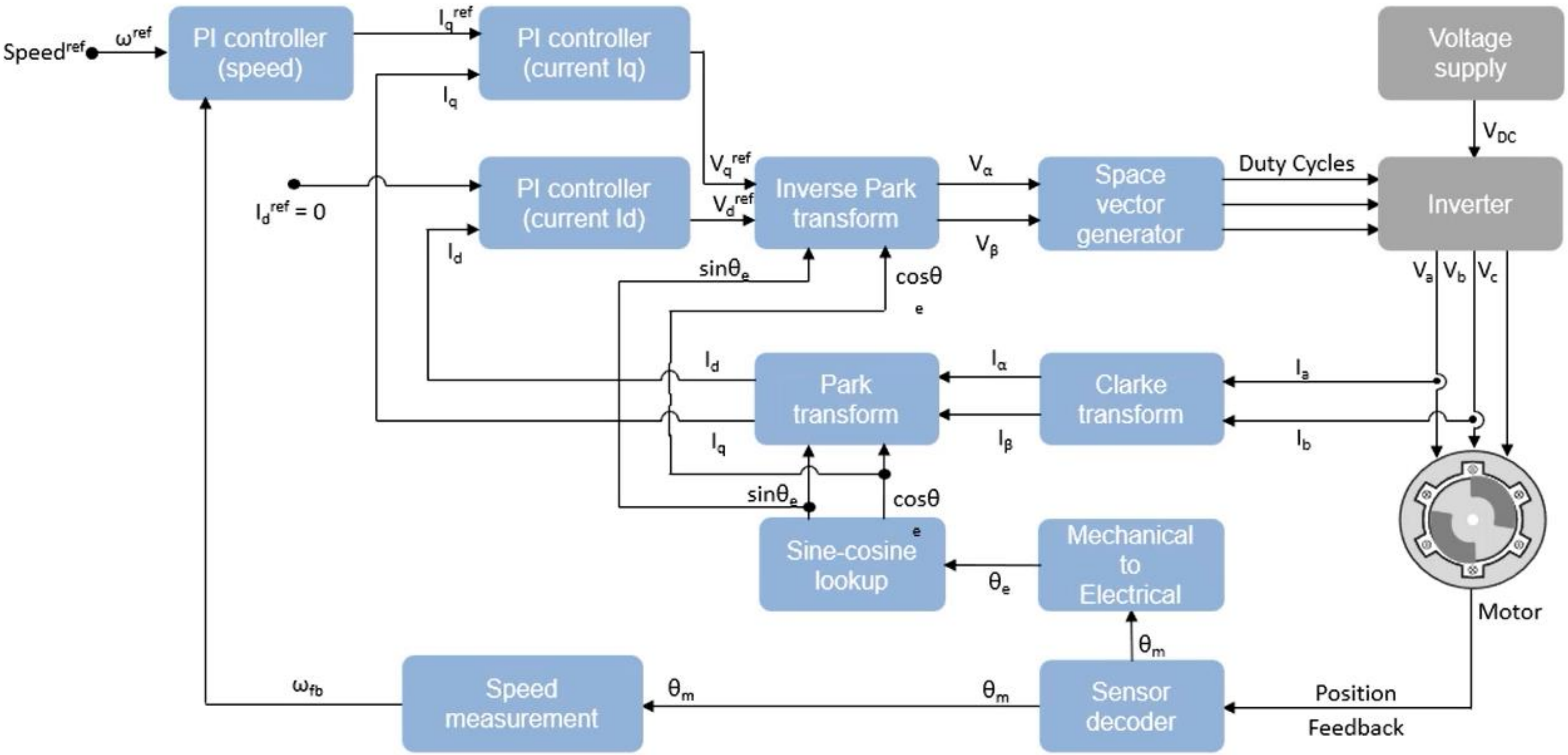
- Motivation
  - nonlinear systems, single controller for multiple operation conditions
  - multiple inputs multiple outputs



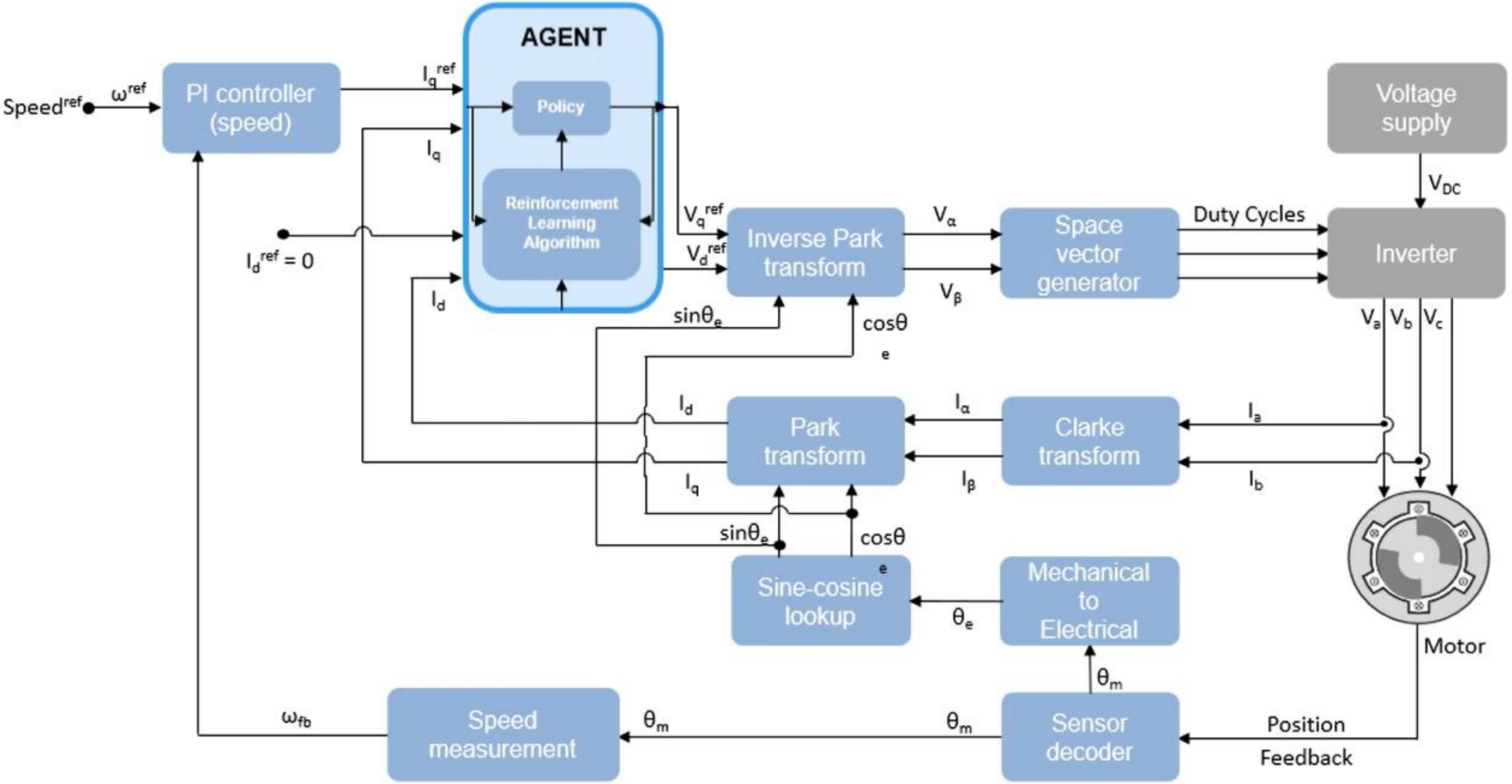
- Solution
  - create Simulink model (environment)
  - create reward function
  - define actor and critic networks
  - train RL agent
  - simulate policy and compare with PI speed controllers

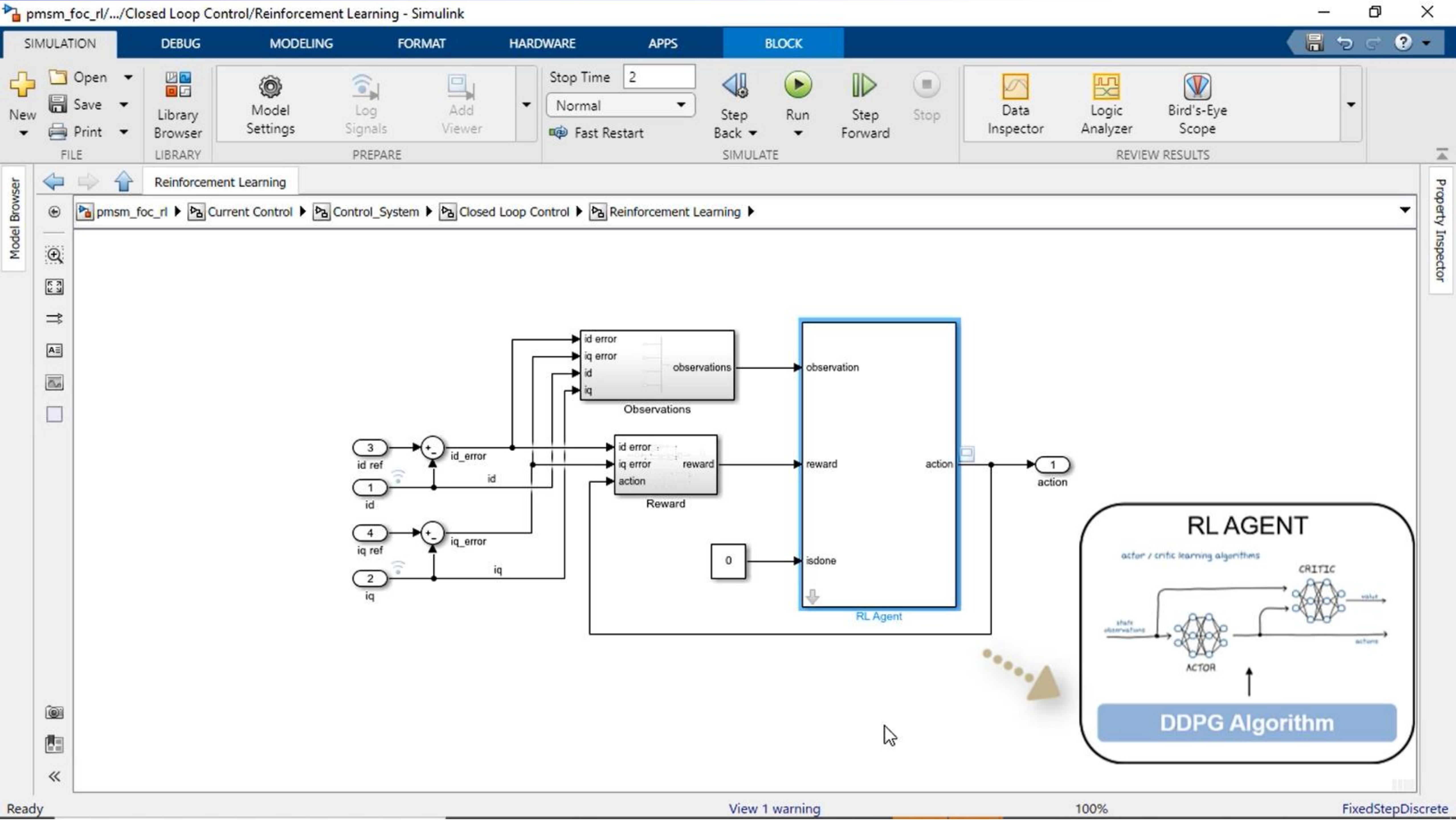


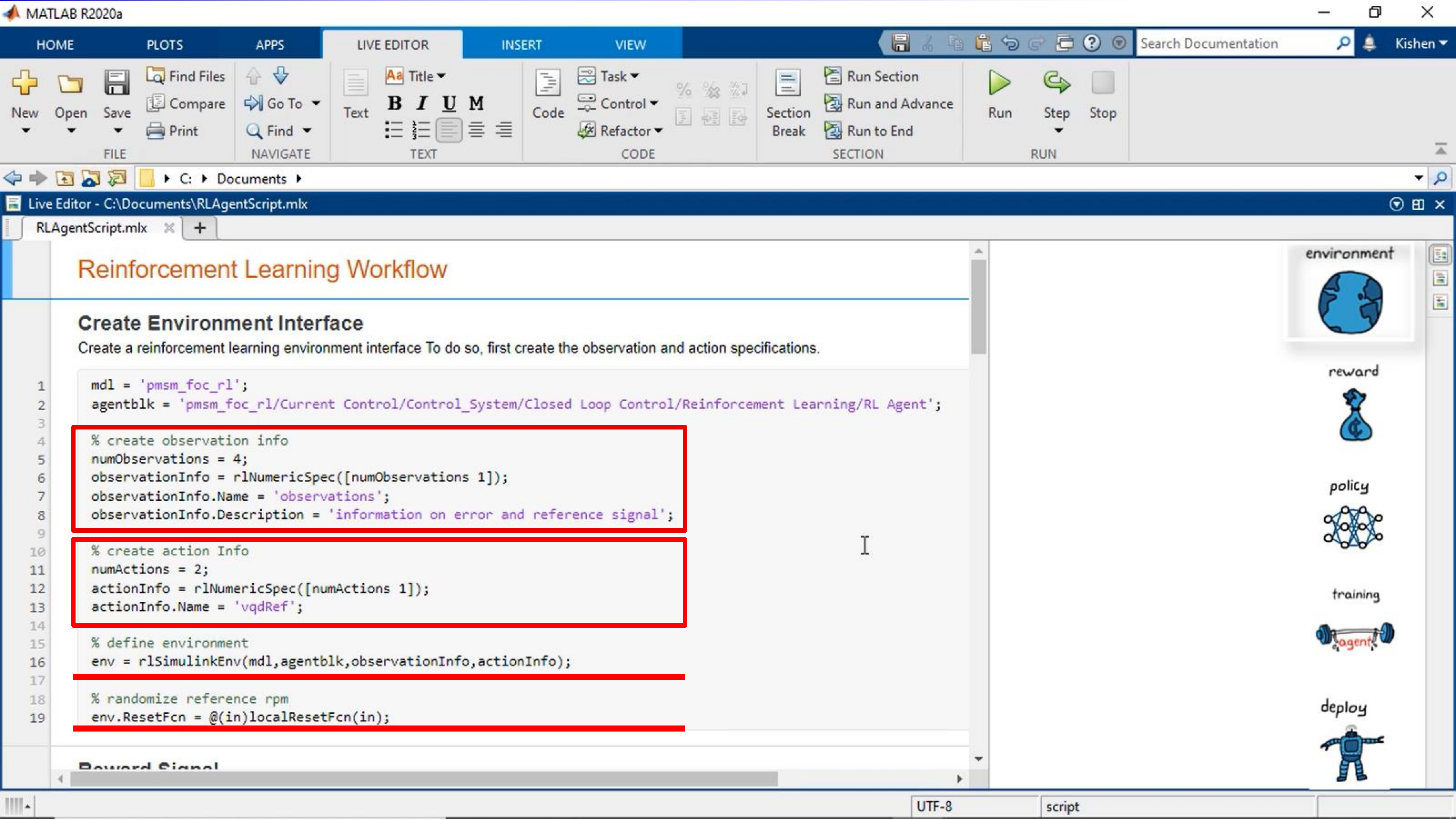
Field Oriented Control Architecture

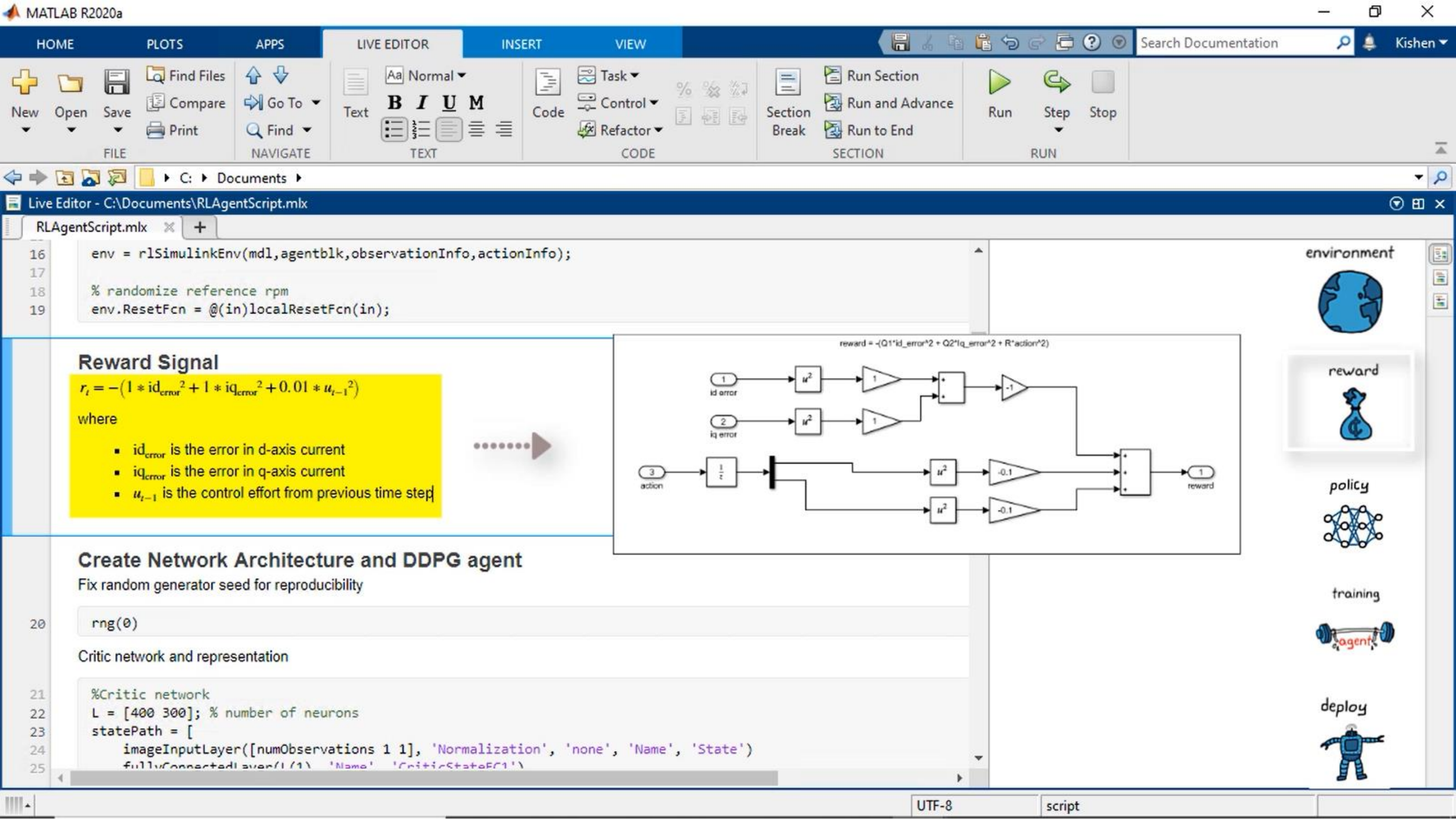


Field Oriented Control Architecture using RL




pmsm_foc_rl/.../Closed Loop Control/Reinforcement Learning - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
HARDWARE
APPS
BLOCK
New
Open
Save
Print
FILE
Library Browser
LIBRARY
Model Settings
Log Signals
Add Viewer
PREPARE
Stop Time
2
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
SIMULATE
Data Inspector
Logic Analyzer
Bird's-Eye Scope
REVIEW RESULTS
Model Browser
Property Inspector
Reinforcement Learning
pmsm_foc_rl
Current Control
Control_System
Closed Loop Control
Reinforcement Learning
id error
iq error
id
iq
observations
Observations
observation
reward
action
isdone
RL Agent
id error
iq error
action
reward
Reward
0
3
id ref
1
id
4
iq ref
2
iq
id_error
iq_error
id
iq
1
action
RL AGENT
actor / critic learning algorithms
CRITIC
ACTOR
DDPG Algorithm
Ready
View 1 warning
100%
FixedStepDiscrete


MATLAB R2020a
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Search Documentation
Kishen
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Text
Title
TEXT
Code
Task
Control
Refactor
CODE
Section Break
Run Section
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
C: ▸ Documents ▸
Live Editor - C:\Documents\RLAgentScript.mlx
RLAgentScript.mlx
Reinforcement Learning Workflow
Create Environment Interface
Create a reinforcement learning environment interface To do so, first create the observation and action specifications.
mdl = 'pmsm_foc_rl';
agentblk = 'pmsm_foc_rl/Current Control/Control_System/Closed Loop Control/Reinforcement Learning/RL Agent';
% create observation info
numObservations = 4;
observationInfo = rlNumericSpec([numObservations 1]);
observationInfo.Name = 'observations';
observationInfo.Description = 'information on error and reference signal';
% create action Info
numActions = 2;
actionInfo = rlNumericSpec([numActions 1]);
actionInfo.Name = 'vqdRef';
% define environment
env = rlSimulinkEnv(mdl,agentblk,observationInfo,actionInfo);
% randomize reference rpm
env.ResetFcn = @(in)localResetFcn(in);
Reward Signal
environment
reward
policy
training
agent
deploy
UTF-8
script


MATLAB R2020a
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Search Documentation
Kishen
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Normal
Text
TEXT
Code
Task
Control
Refactor
CODE
Section Break
Run Section
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
C: ▸ Documents ▸
Live Editor - C:\Documents\RLAgentScript.mlx
RLAgentScript.mlx
16 env = rlSimulinkEnv(mdl,agentblk,observationInfo,actionInfo);
17
18 % randomize reference rpm
19 env.ResetFcn = @(in)localResetFcn(in);
Reward Signal
$r_t = -(1 * \mathrm{id}_{error}{}^2 + 1 * \mathrm{iq}_{error}{}^2 + 0.01 * u_{t-1}{}^2)$
where
id_{error} is the error in d-axis current
iq_{error} is the error in q-axis current
u_{t-1} is the control effort from previous time step
reward = -(Q1*id_error^2 + Q2*iq_error^2 + R*action^2)
id error
iq error
action
reward
-1
-0.1
Create Network Architecture and DDPG agent
Fix random generator seed for reproducibility
20 rng(0)
Critic network and representation
21 %Critic network
22 L = [400 300]; % number of neurons
23 statePath = [
24 imageInputLayer([numObservations 1 1], 'Normalization', 'none', 'Name', 'State')
environment
reward
policy
training
agent
deploy
UTF-8
script

MATLAB R2020a

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

Live Editor - C:\Documents\RLAgentScript.mlx

RLAgentScript.mlx

Create Network Architecture and DDPG agent

Fix random generator seed for reproducibility

```
rng(0)
```

Critic network and representation

```
%Critic network
L = [400 300]; % number of neurons
statePath = [
    imageInputLayer([numObservations 1 1], 'Normalization', 'none', 'Name', 'State')
    fullyConnectedLayer(L(1), 'Name', 'CriticStateFC1')
    clippedReluLayer(10,'Name','CriticClip1')
    fullyConnectedLayer(L(2), 'Name', 'CriticStateFC2')];
actionPath = [
    imageInputLayer([numActions 1 1], 'Normalization', 'none', 'Name', 'Action')
    fullyConnectedLayer(L(2), 'Name', 'CriticActionFC1')];
commonPath = [
    additionLayer(2,'Name', 'add')
    clippedReluLayer(10,'Name','CriticCCommonClip')
    fullyConnectedLayer(1, 'Name', 'CriticOutput')];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
```

UTF-8 script Ln 27 Col 58

MATLAB R2020a

Live Editor - C:\Documents\RLAgentScript.mlx

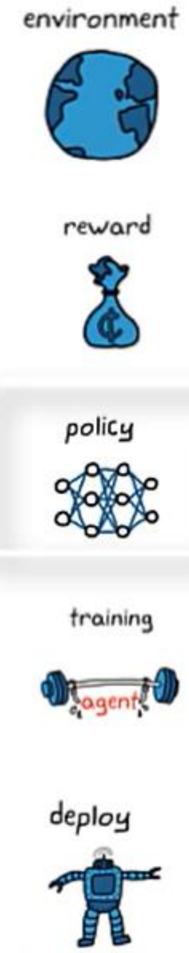
RLAgentScript.mlx

```
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');

% View the critic network configuration.
% figure plot(criticNetwork)

% create the critic representation
criticOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1,'L2RegularizationFactor',1e-4,'U:
critic = rlQValueRepresentation(criticNetwork,observationInfo,actionInfo,'Observation',{'State'},'Action',{'Act:
```

Actor network and representation

```
actorNetwork = [
    imageInputLayer([numObservations 1 1], 'Normalization', 'none', 'Name', 'State')
    fullyConnectedLayer(L(1), 'Name', 'actorFC1')
    tanhLayer('Name','tanh1')
    fullyConnectedLayer(L(2), 'Name', 'actorFC2')
    tanhLayer('Name','tanh2')
    fullyConnectedLayer(numActions, 'Name', 'Action')
    tanhLayer('Name','tanh3')
    ];

% create the actor representation
actorOptions = rlRepresentationOptions('LearnRate',1e-03,'GradientThreshold',1,'UseDevice','cpu');
actor = rlRepresentation(actorNetwork,observationInfo,actionInfo,'Observation',{'State'},'Action',{'tanh3'},act
```

Create DDPG Agent



UTF-8 script Ln 52 Col 30

MATLAB R2020a

Live Editor - C:\Documents\RLAgentScript.mlx

RLAgentScript.mlx

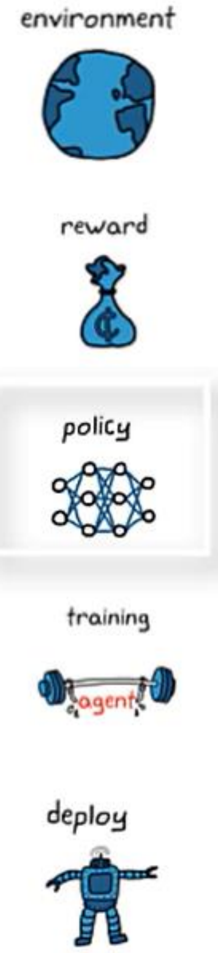
```
% create the actor representation
actorOptions = rlRepresentationOptions('LearnRate',1e-03,'GradientThreshold',1,'UseDevice','cpu');
actor = rlRepresentation(actorNetwork,observationInfo,actionInfo,'Observation',{'State'},'Action',{'tanh3'},act
```

Create DDPG Agent

```
Ts_agent = 2e-04;
agentOptions = rlDDPGAgentOptions;
agentOptions.SampleTime = Ts_agent; % Sample time for the controller
agentOptions.ExperienceBufferLength = 1e6;
agentOptions.DiscountFactor = 0.99;
agentOptions.NoiseOptions.Variance = 0.1;
agentOptions.NoiseOptions.VarianceDecayRate = 1e-6;
agentOptions.NoiseOptions.VarianceMin = 0.025;
agentOptions.NoiseOptions.MeanAttractionConstant = 1;

agent = rlDDPGAgent(actor,critic,agentOptions);
```

Train Agent

```
maxepisodes = 2000;
maxsteps = ceil(T/Ts_agent);

trainingOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes, ...
    'MaxStepsPerEpisode',maxsteps, ...
    'Verbose',false, ...
```



UTF-8 script Ln 69 Col 47

MATLAB R2020a

Live Editor - C:\Documents\RLAgentScript.mlx

RLAgentScript.mlx

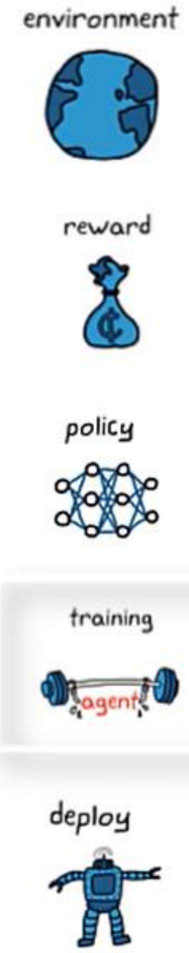
```
agent = rlDDPGAgent(actor,critic,agentOptions);
```

Train Agent

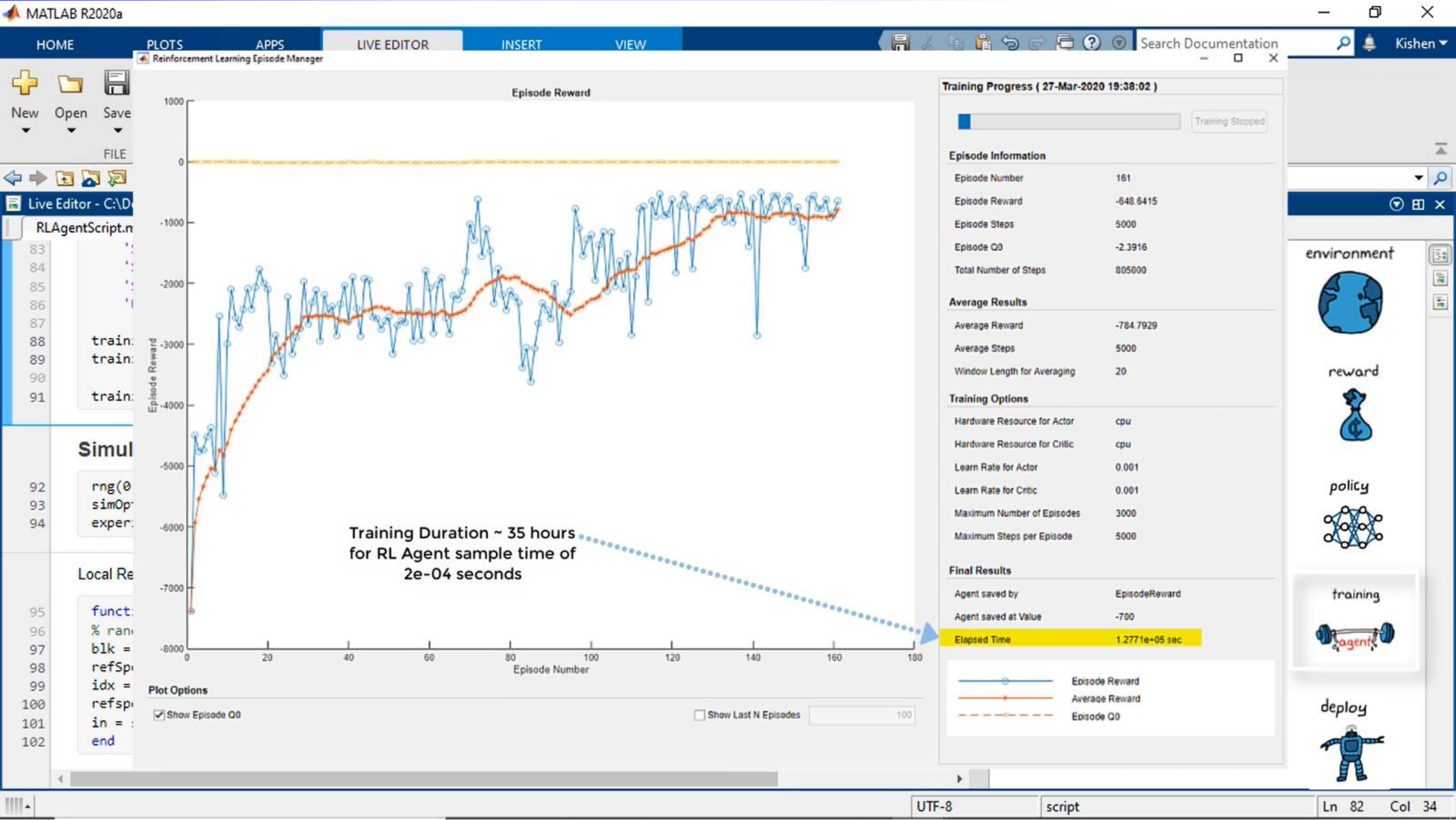
```
maxepisodes = 2000;
maxsteps = ceil(T/Ts_agent);

trainingOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes, ...
    'MaxStepsPerEpisode',maxsteps, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward',...
    'StopTrainingValue', -200,...
    'ScoreAveragingWindowLength',20,...
    'SaveAgentCriteria','AverageReward',...
    'SaveAgentValue', -700,...
    'UseParallel', true);

trainingOpts.ParallelizationOptions.Mode = 'async';
trainingOptions.ParallelizationOptions.DataToSendFromWorkers = 'experiences';

trainingStats = train(agent,env,trainingOpts);
```

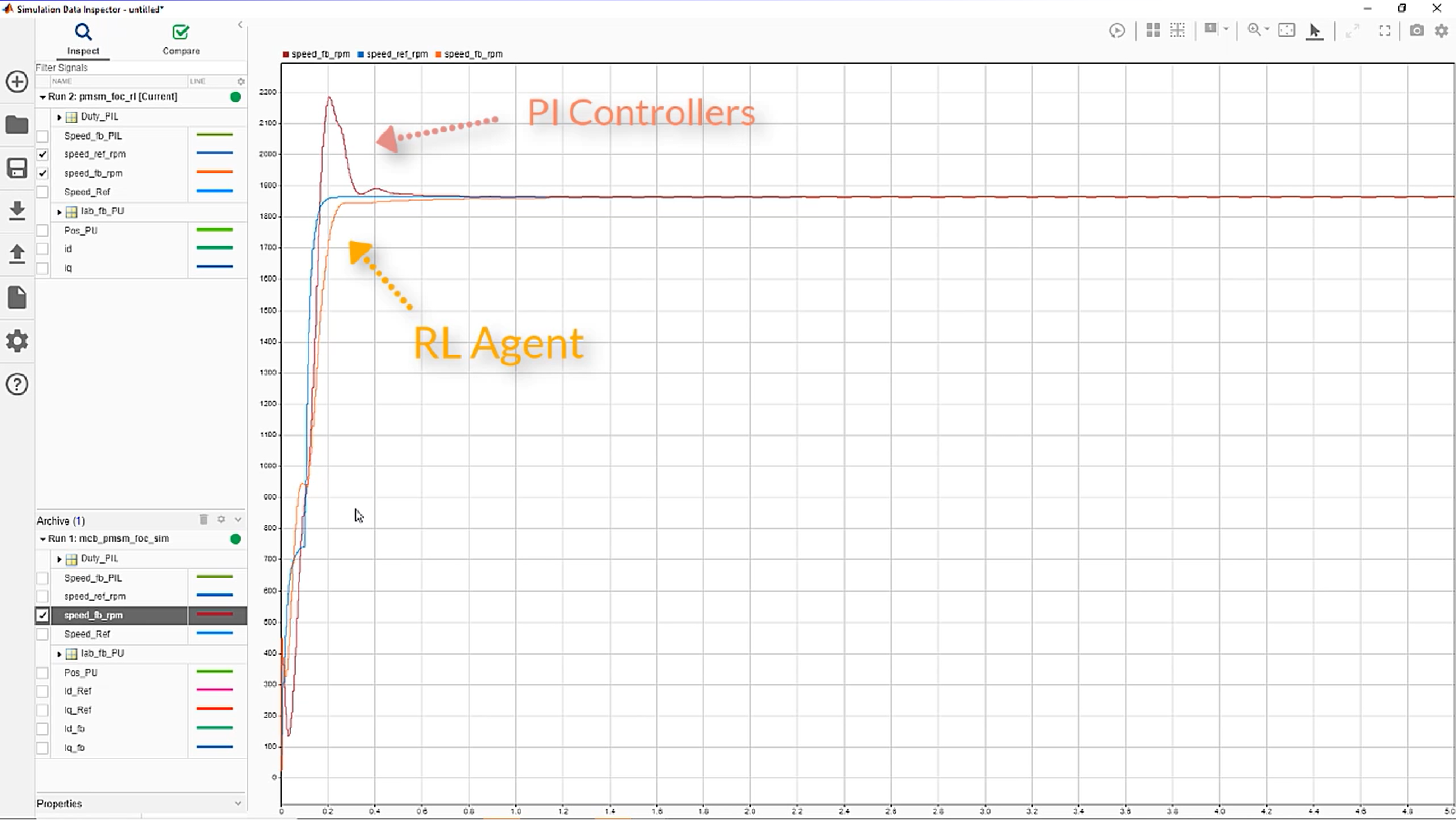
Simulate DDPG policy (Deploy)

```
rng(0)
```



UTF-8 script Ln 82 Col 34


MATLAB R2020a
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Search Documentation
Kishen
New
Open
Save
FILE
Live Editor - C:\D
RLAgentScript.m
Reinforcement Learning Episode Manager
Episode Reward
Episode Reward
Episode Number
Training Duration ~ 35 hours
for RL Agent sample time of
2e-04 seconds
Plot Options
Show Episode Q0
Show Last N Episodes
Training Progress (27-Mar-2020 19:38:02)
Training Stopped
Episode Information
Episode Number 161
Episode Reward -648.6415
Episode Steps 5000
Episode Q0 -2.3916
Total Number of Steps 805000
Average Results
Average Reward -784.7929
Average Steps 5000
Window Length for Averaging 20
Training Options
Hardware Resource for Actor cpu
Hardware Resource for Critic cpu
Learn Rate for Actor 0.001
Learn Rate for Critic 0.001
Maximum Number of Episodes 3000
Maximum Steps per Episode 5000
Final Results
Agent saved by EpisodeReward
Agent saved at Value -700
Elapsed Time 1.2771e+05 sec
Episode Reward
Average Reward
Episode Q0
environment
reward
policy
training
agent
deploy
UTF-8
script
Ln 82
Col 34


Simulation Data Inspector - untitled*
Inspect
Compare
Filter Signals
NAME
LINE
Run 2: pmsm_foc_rl [Current]
Duty_PIL
Speed_fb_PIL
speed_ref_rpm
speed_fb_rpm
Speed_Ref
Iab_fb_PU
Pos_PU
id
iq
Archive (1)
Run 1: mcb_pmsm_foc_sim
Duty_PIL
Speed_fb_PIL
speed_ref_rpm
speed_fb_rpm
Speed_Ref
Iab_fb_PU
Pos_PU
Id_Ref
Iq_Ref
Id_fb
Iq_fb
Properties
speed_fb_rpm
speed_ref_rpm
speed_fb_rpm
PI Controllers
RL Agent
2200
2100
2000
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
4.2
4.4
4.6
4.8
5.0

AI for Electrification

Reduced order modeling

Virtual sensors

Control strategy

Predictive maintenance

Energy forecasting

Why Perform Predictive Maintenance?

- Example: faulty braking system leads to wind turbine disaster

  https://youtu.be/7nSB1SdVHqQ

- Wind turbines cost millions of dollars

- Failures can be dangerous

- Maintenance also very expensive and dangerous

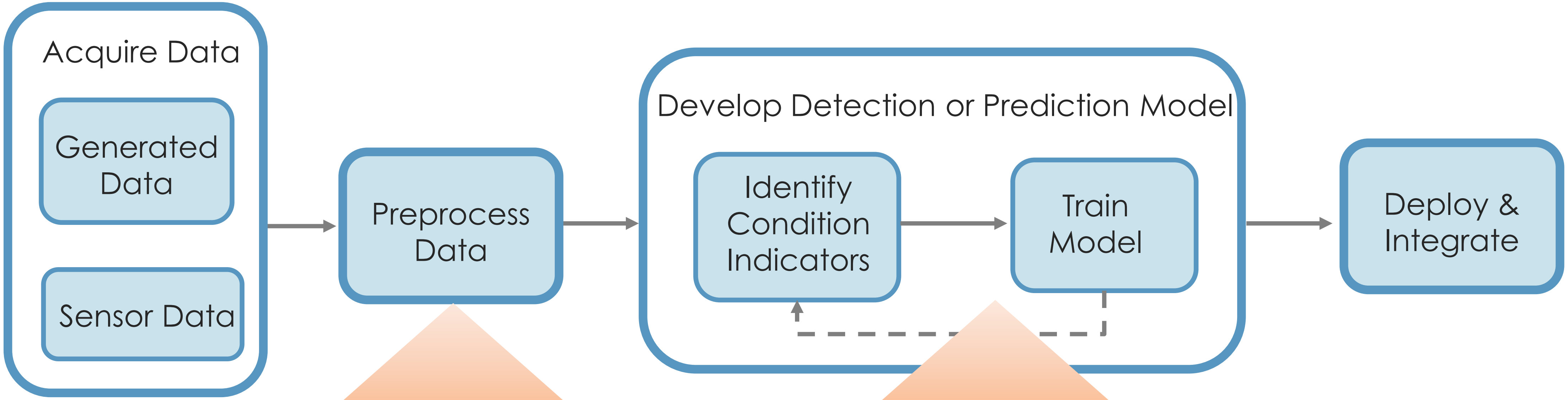
What Does a Predictive Maintenance Algorithm Do?

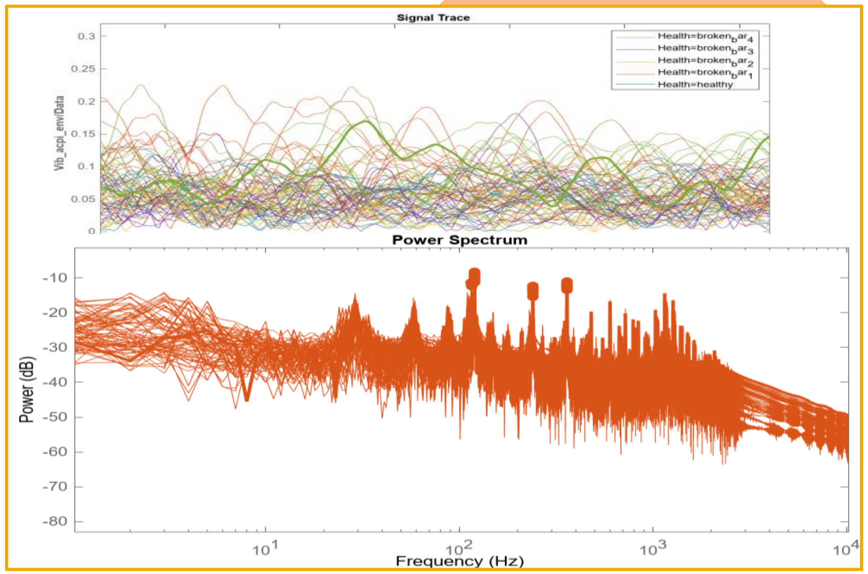
Data →

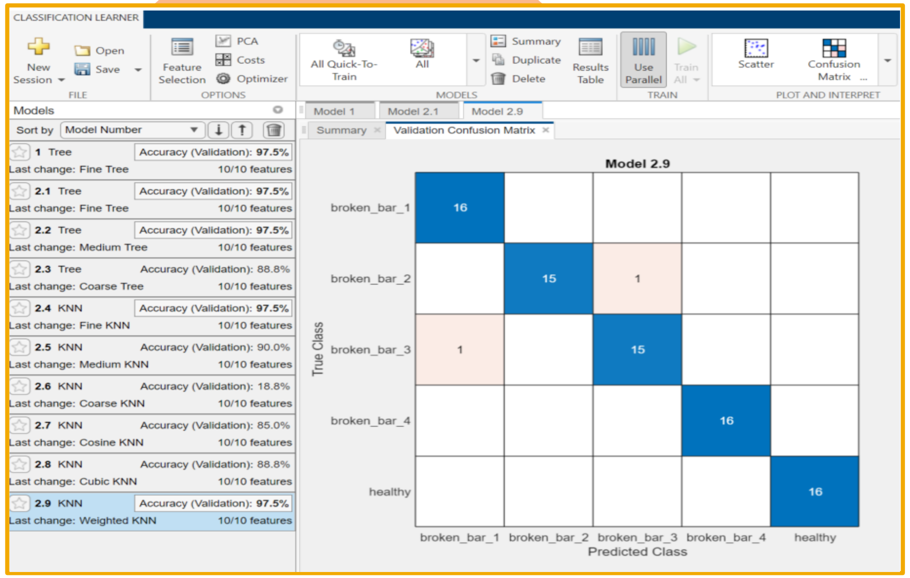
Is my machine operating normally?	**Anomaly Detection**
Why is my machine behaving abnormally?	**Fault Detection**
How much longer can I operate my machine?	**Remaining Useful Life Estimation**

→ **Decision**

Predictive Maintenance Algorithm Development Workflow



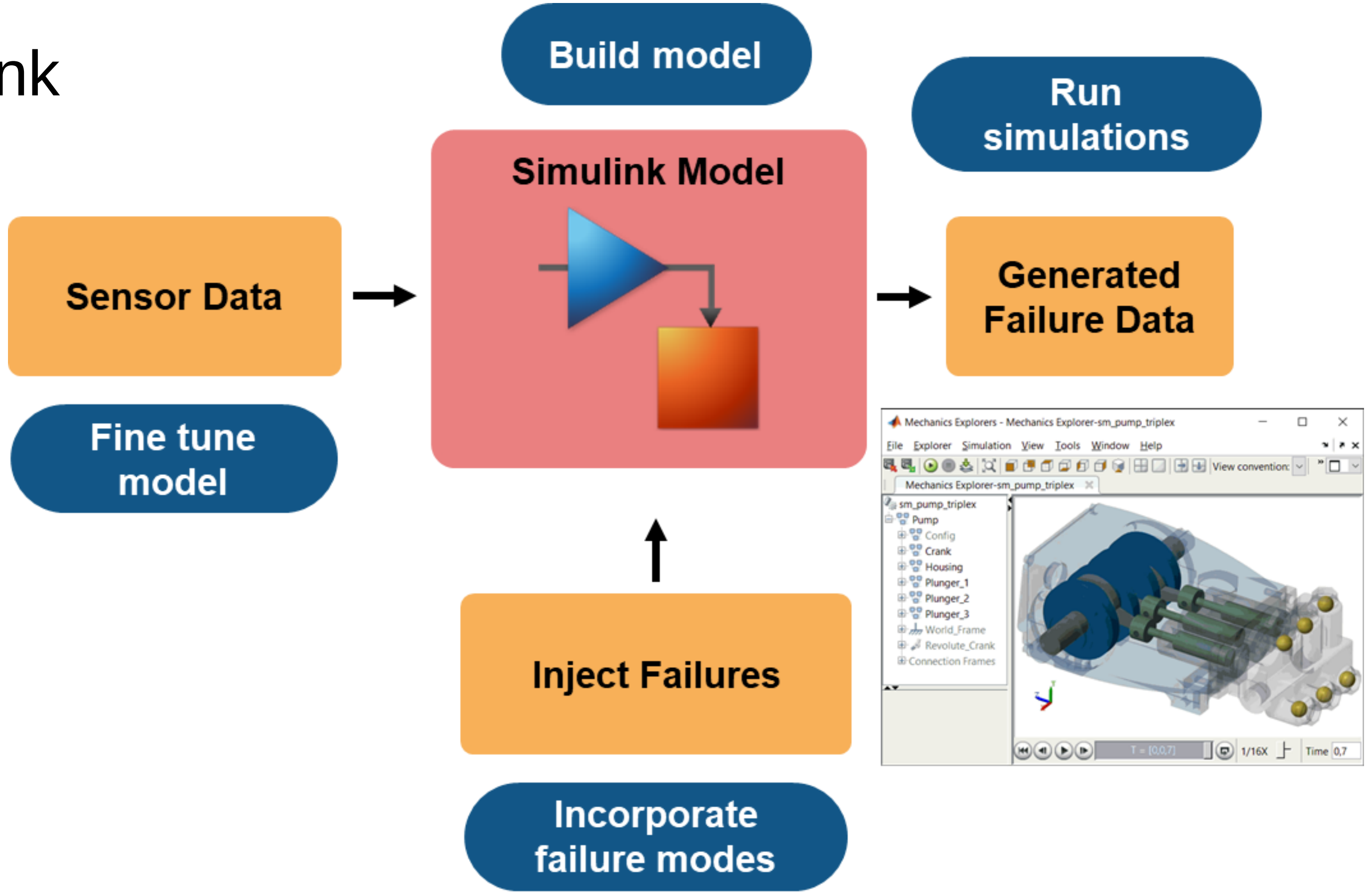




Failure data generation from a Digital Twin

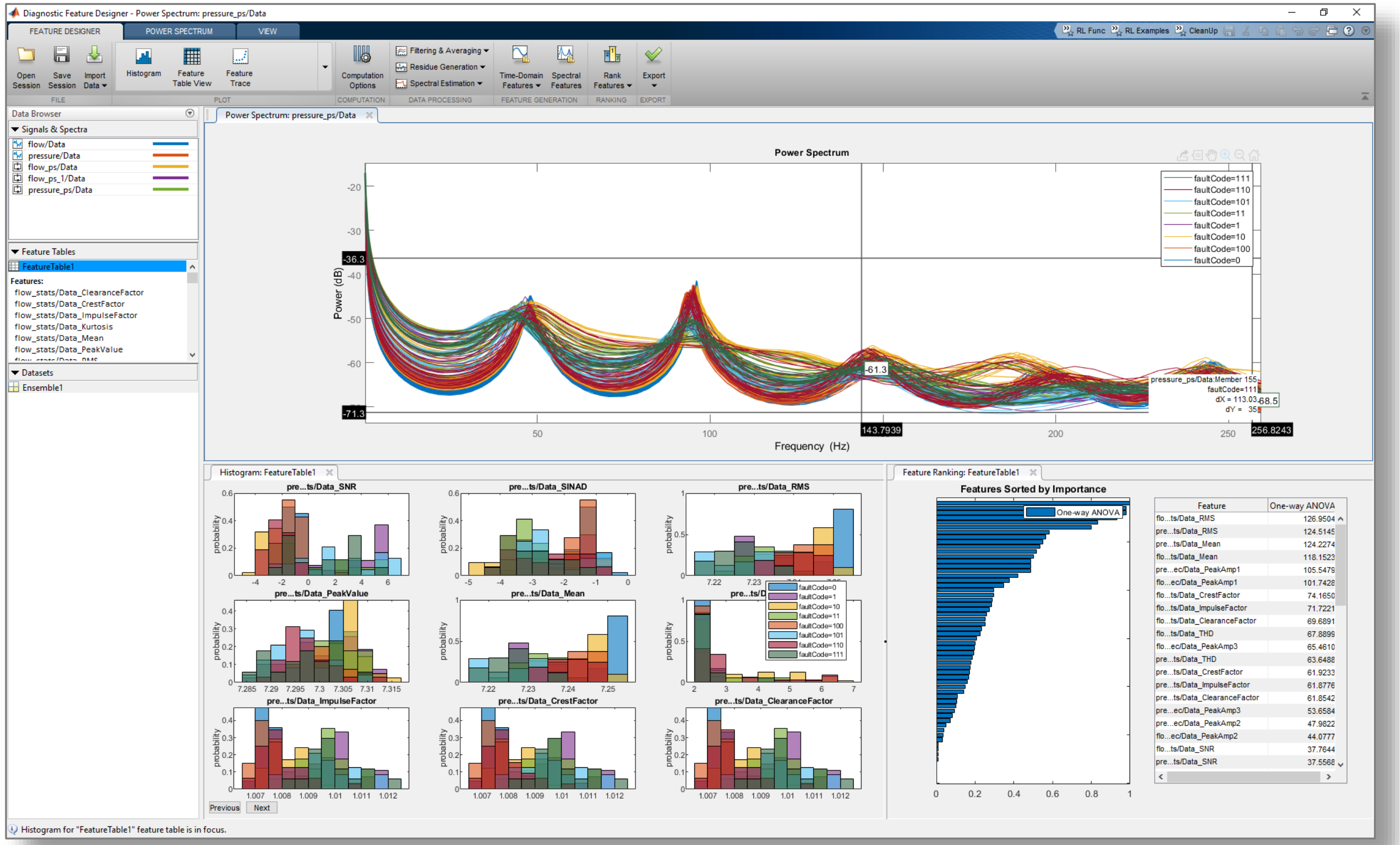
Leverage the engineering knowledge to enhance predictive maintenance

- Generate failure data from Simulink models
- Identify root cause of fault via parameter estimation
- Verify predictive maintenance algorithm in new scenarios with a digital twin



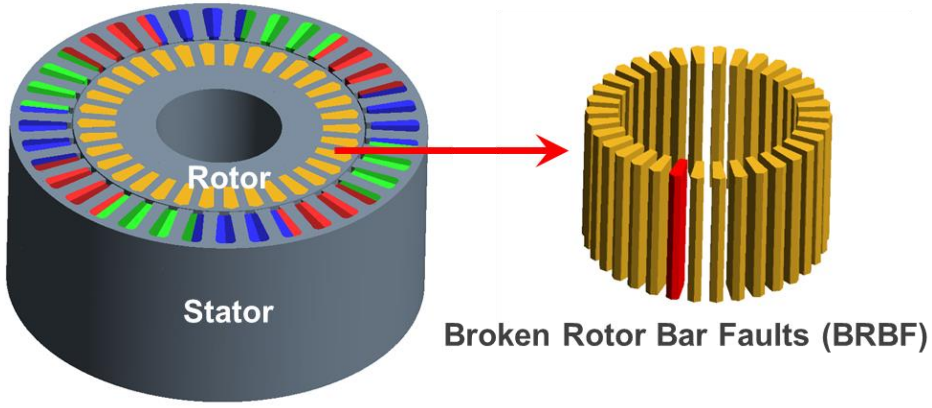
Diagnostic Feature Designer App

- Extract, visualize, and rank features from sensor data
- Use both statistical and dynamic modeling methods
- Work with out-of-memory data
- Explore and discover techniques without writing MATLAB code
- Generate MATLAB code from the App to automate feature extraction and ranking tasks

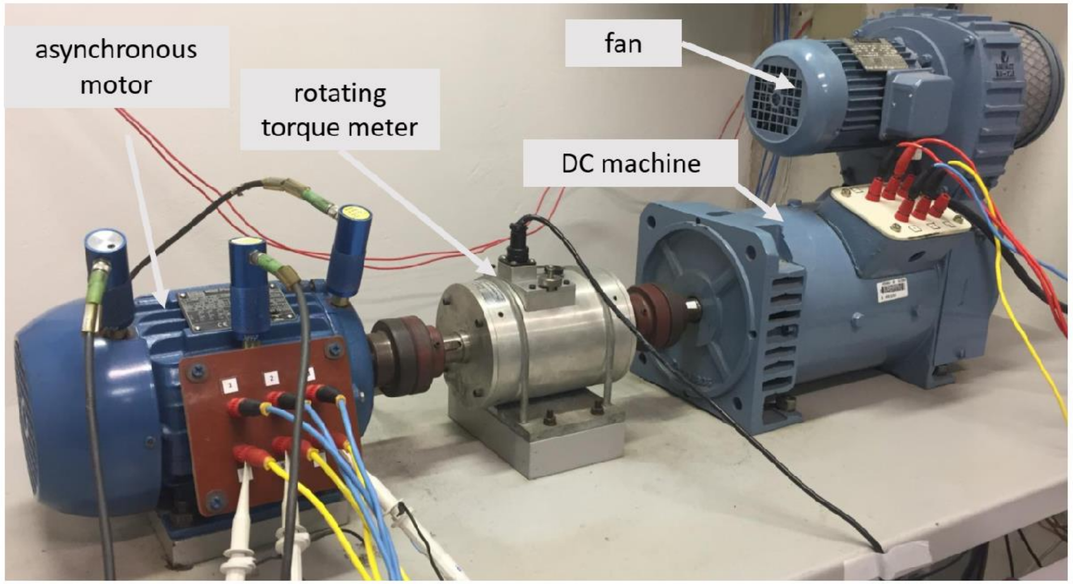


Example: Identify Motor Faults

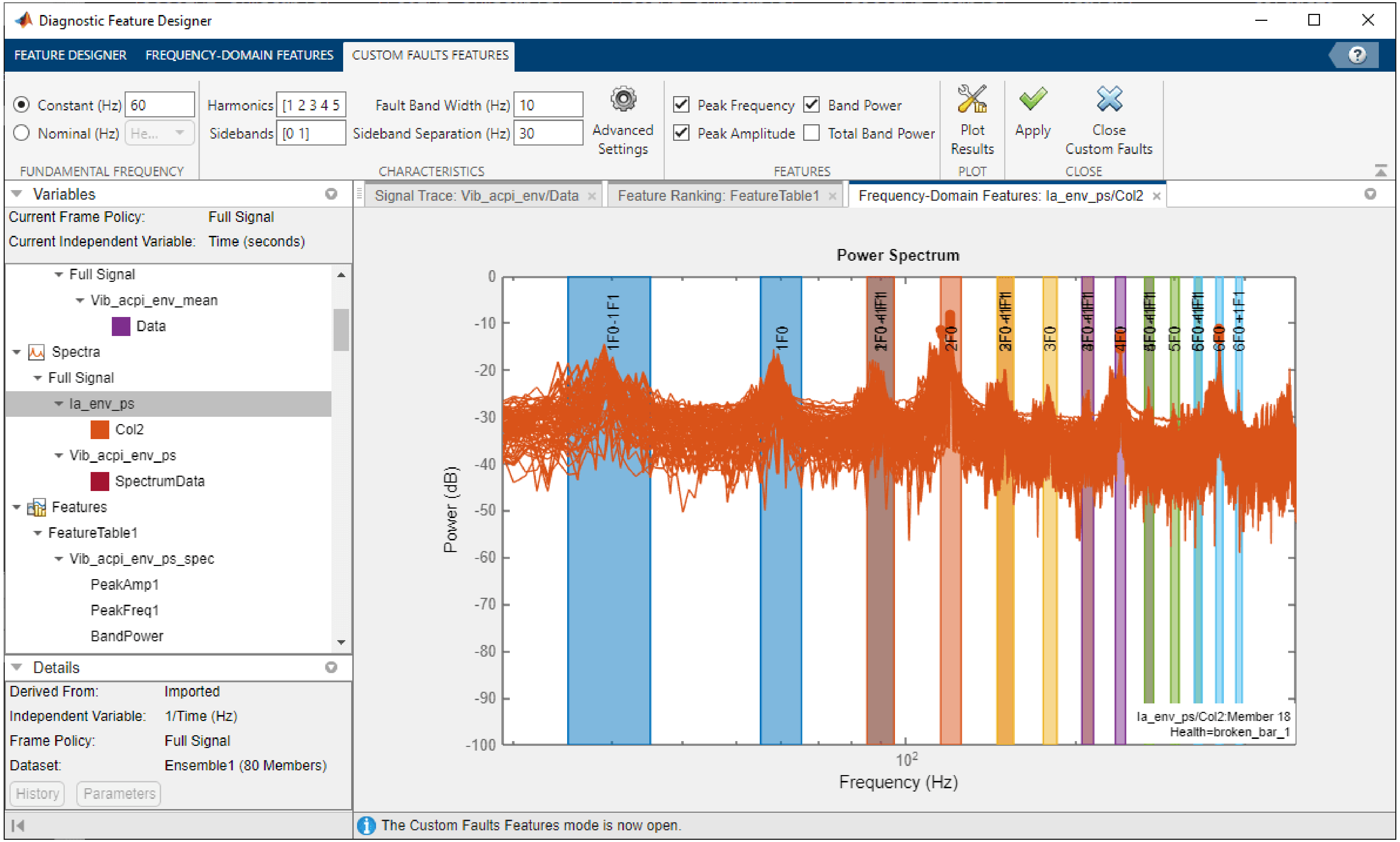
- Motivation
  - diagnosing rotor broken bar
  - usage of mechanical and electrical signals



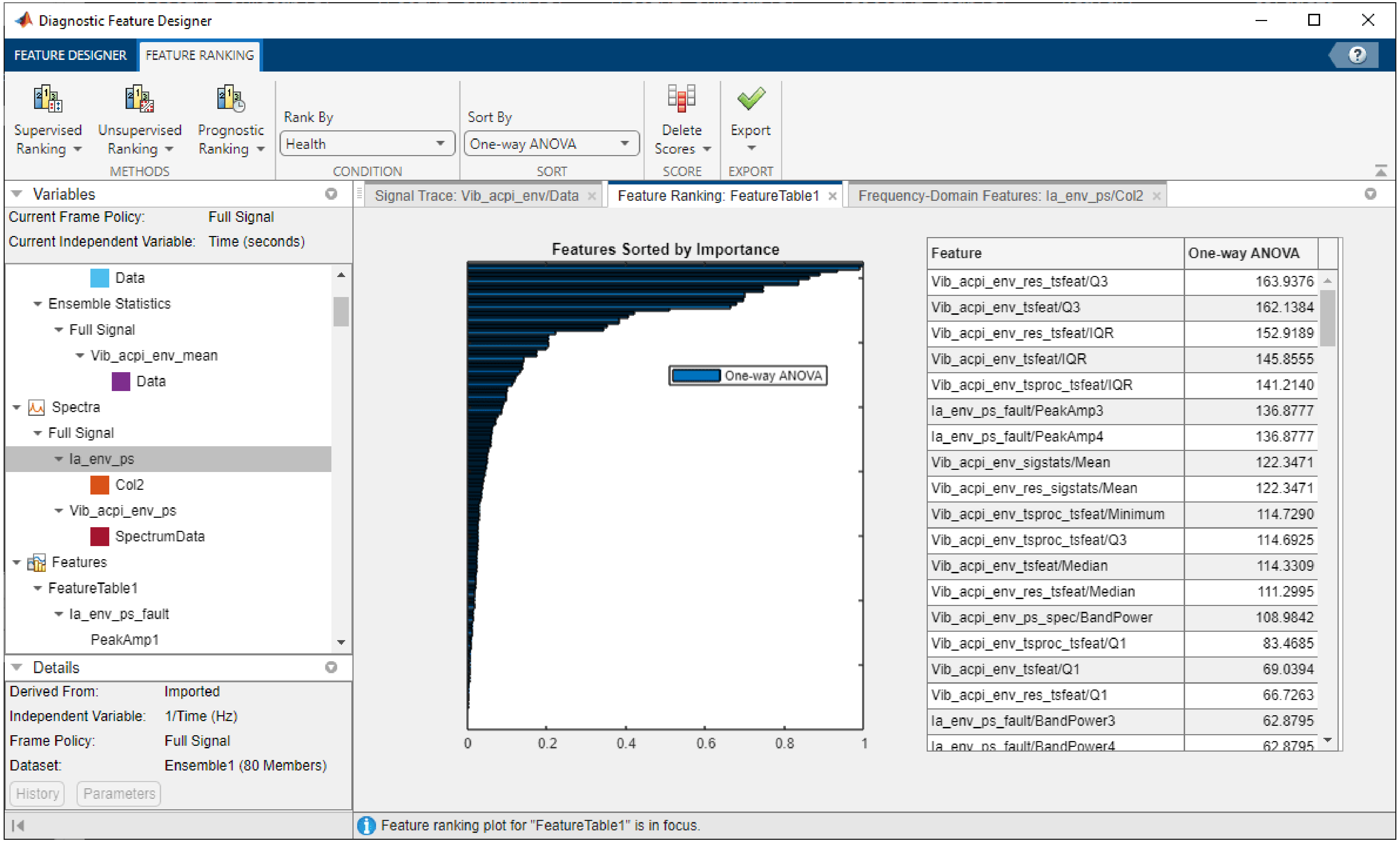
- Solution
  - load dataset
  - extract features
  - rank features
  - export features to Classification Learner App and build a model
  - evaluate model accuracy



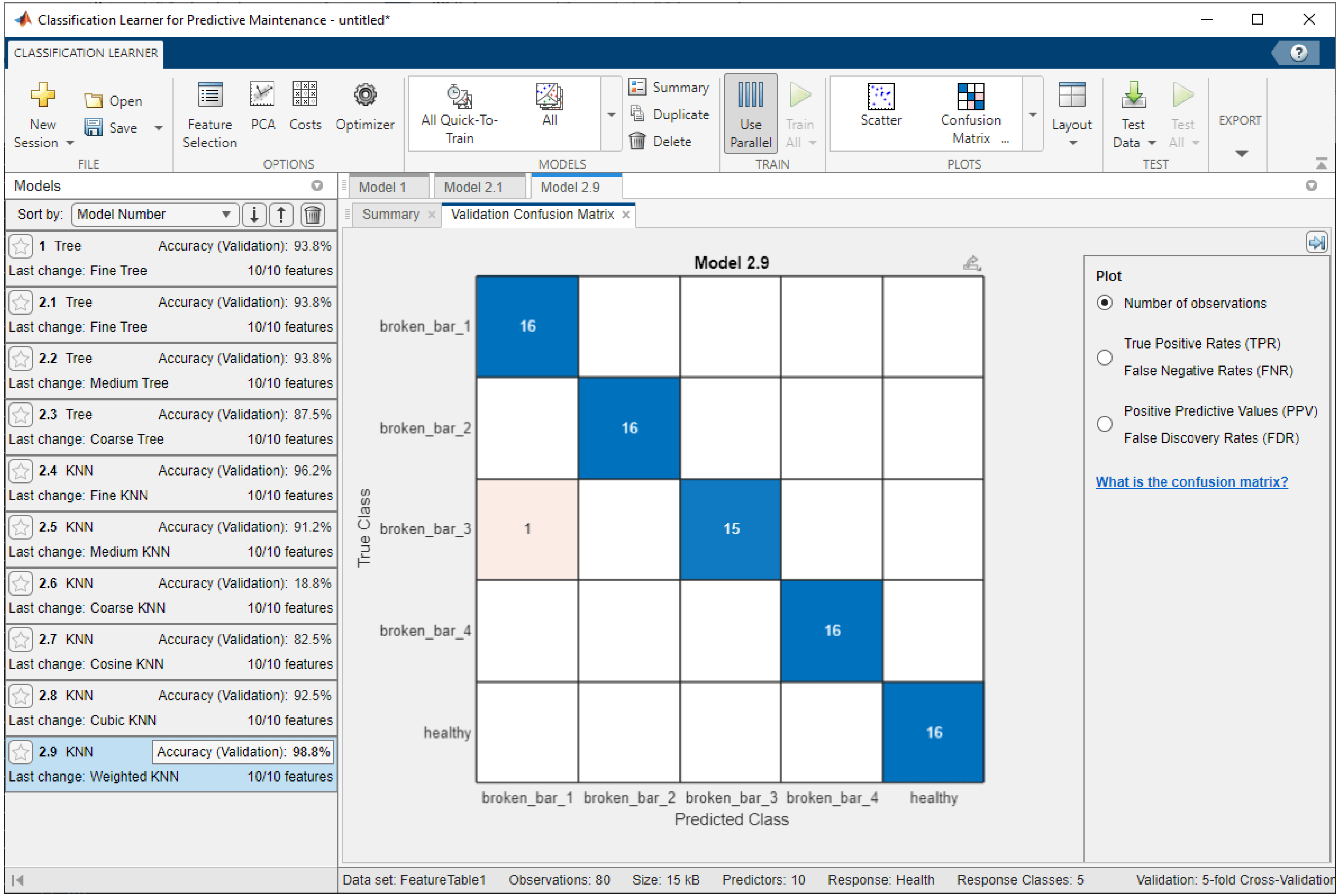
Frequency-Domain Features



Rank Features

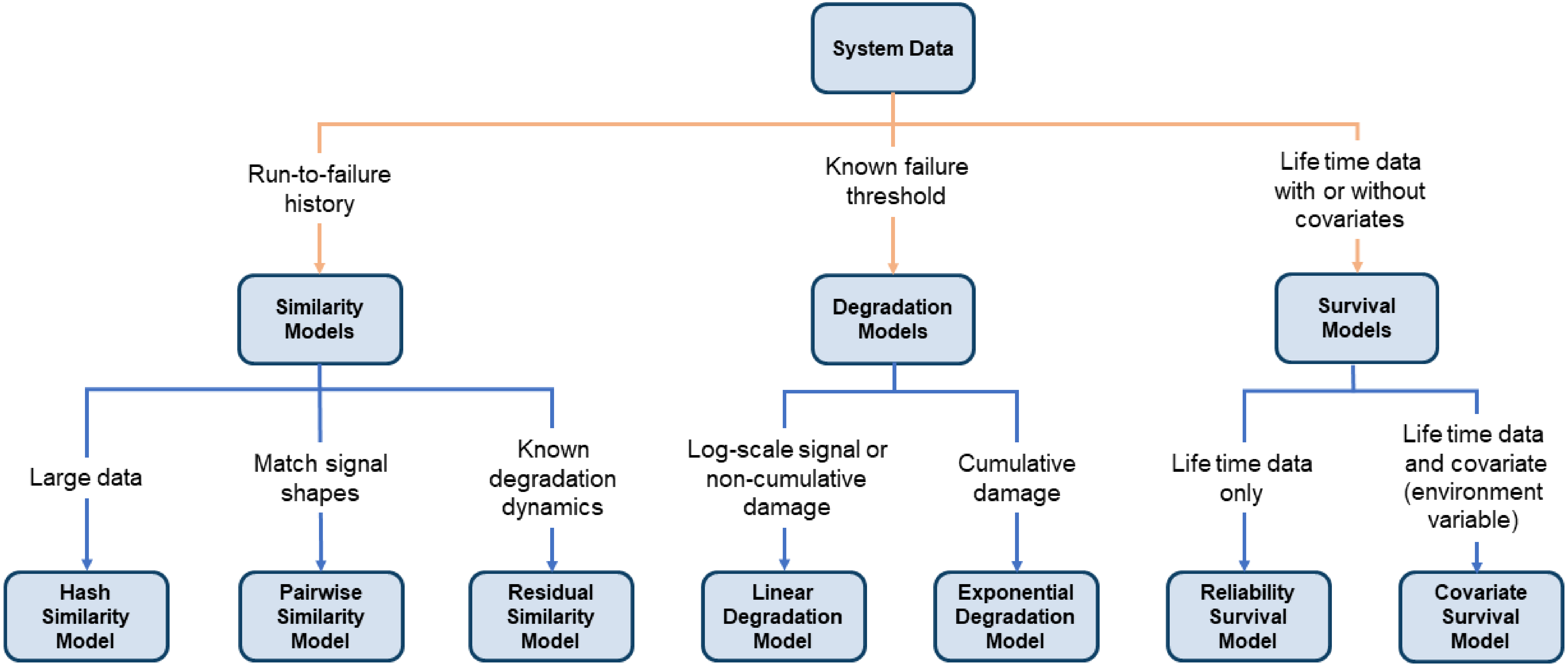


Faults classification



RUL Methods and when to use them

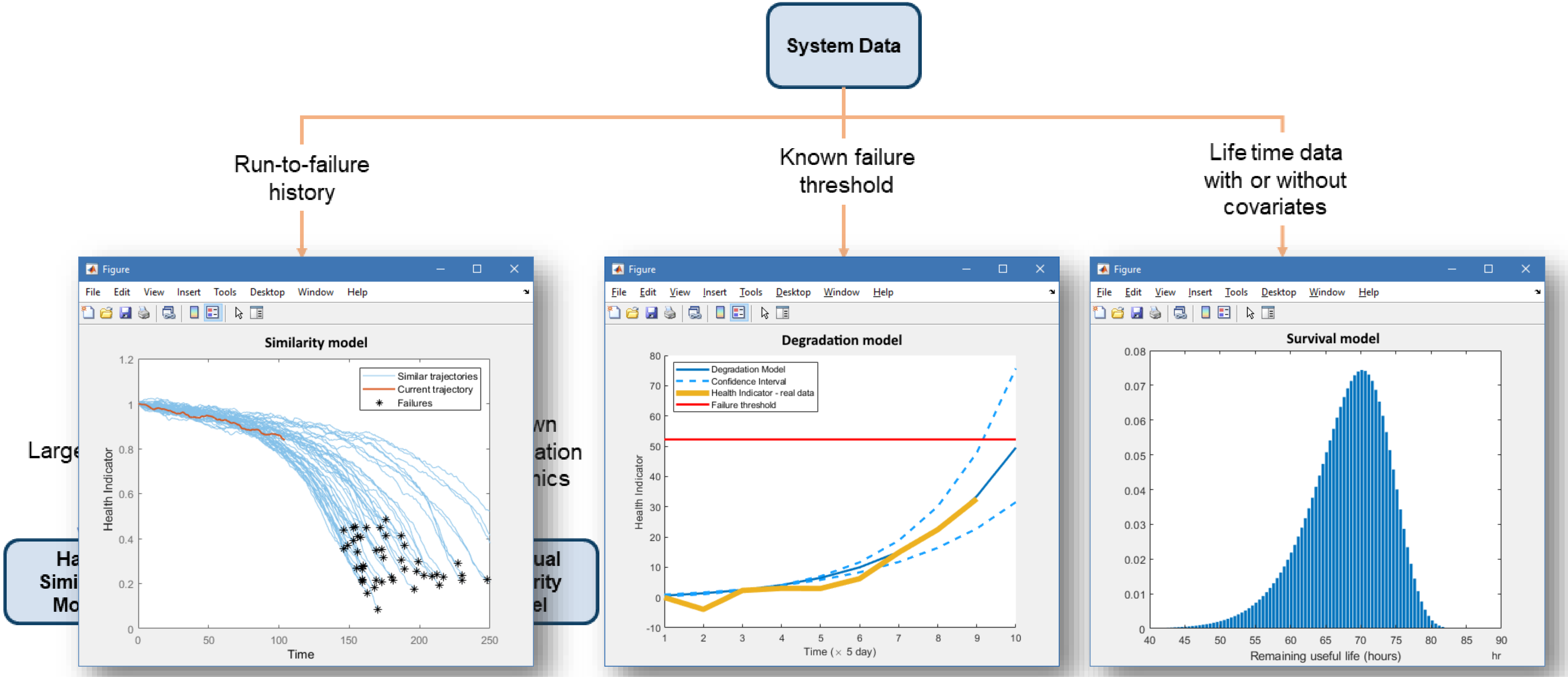
Requirement: Need to know what constitutes failure data



[Details on model selection in the documentation]

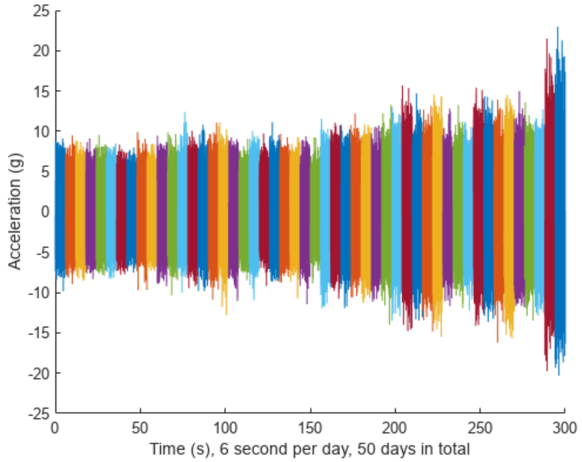
RUL Methods and when to use them

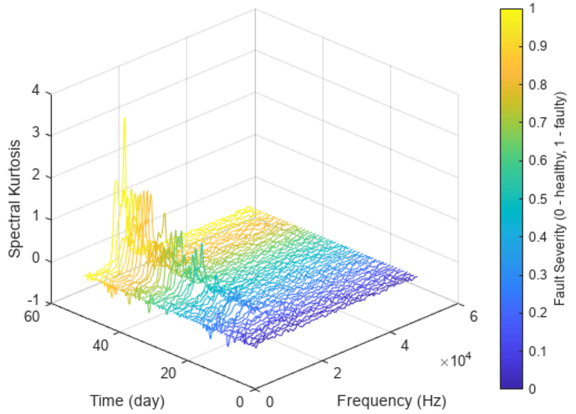
Requirement: Need to know what constitutes failure data



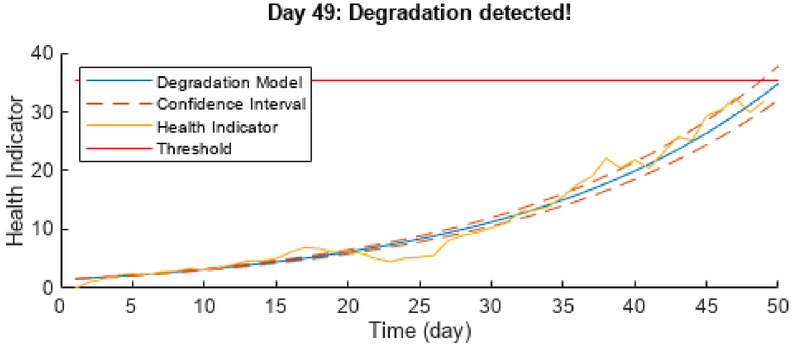
Example: Wind Turbine High-Speed Bearing Prognosis

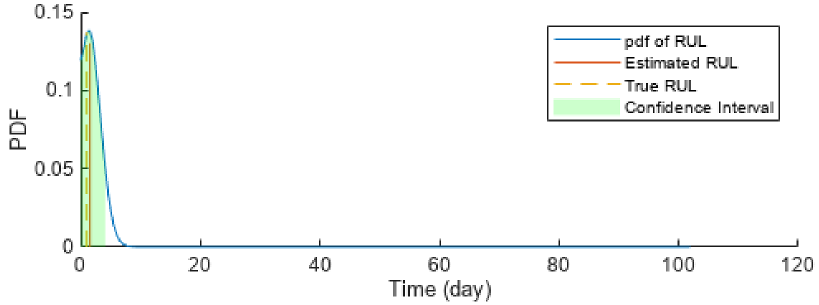
- Motivation
  – predict Remaining Useful Life of a wind turbine bearing
  – detect the significant degradation trend





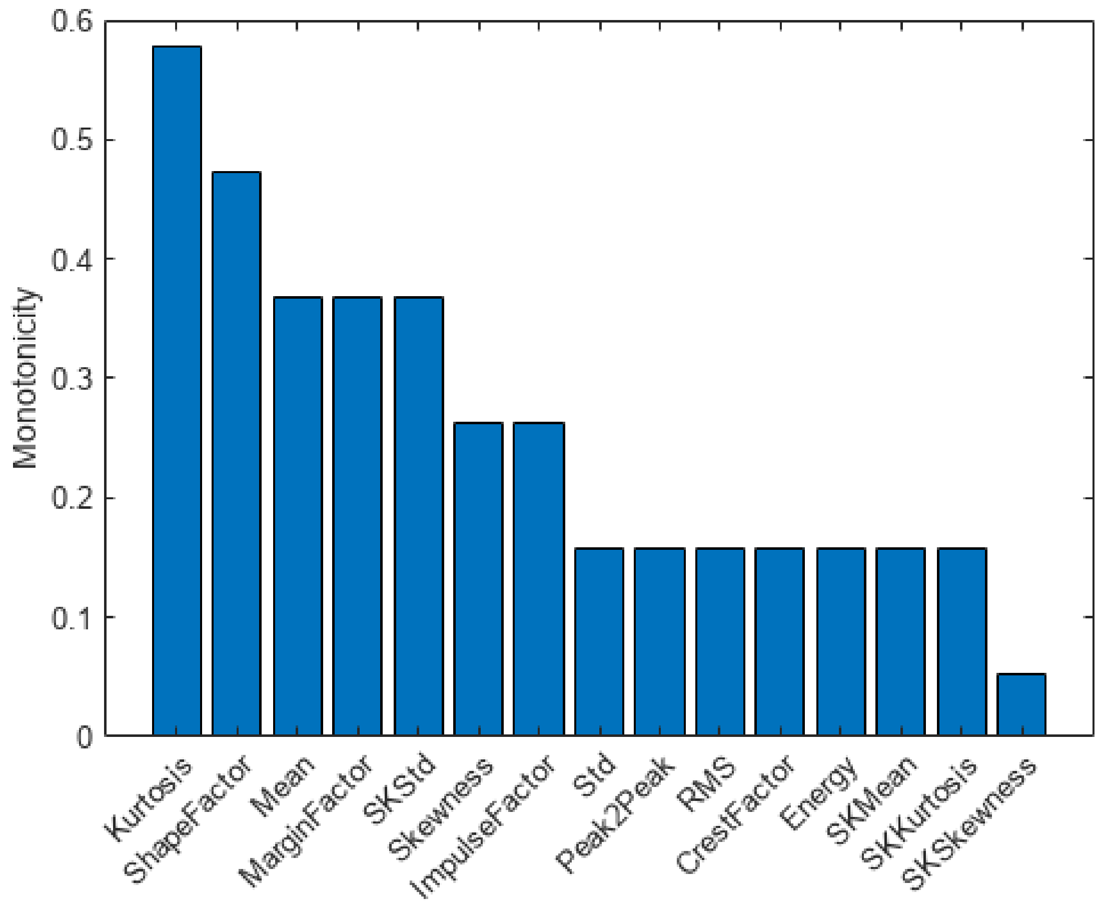
- Solution
  – import data
  – extract features
  – rank features
  – fit exponential degradation model
  – predict the RUL and update the parameter distribution



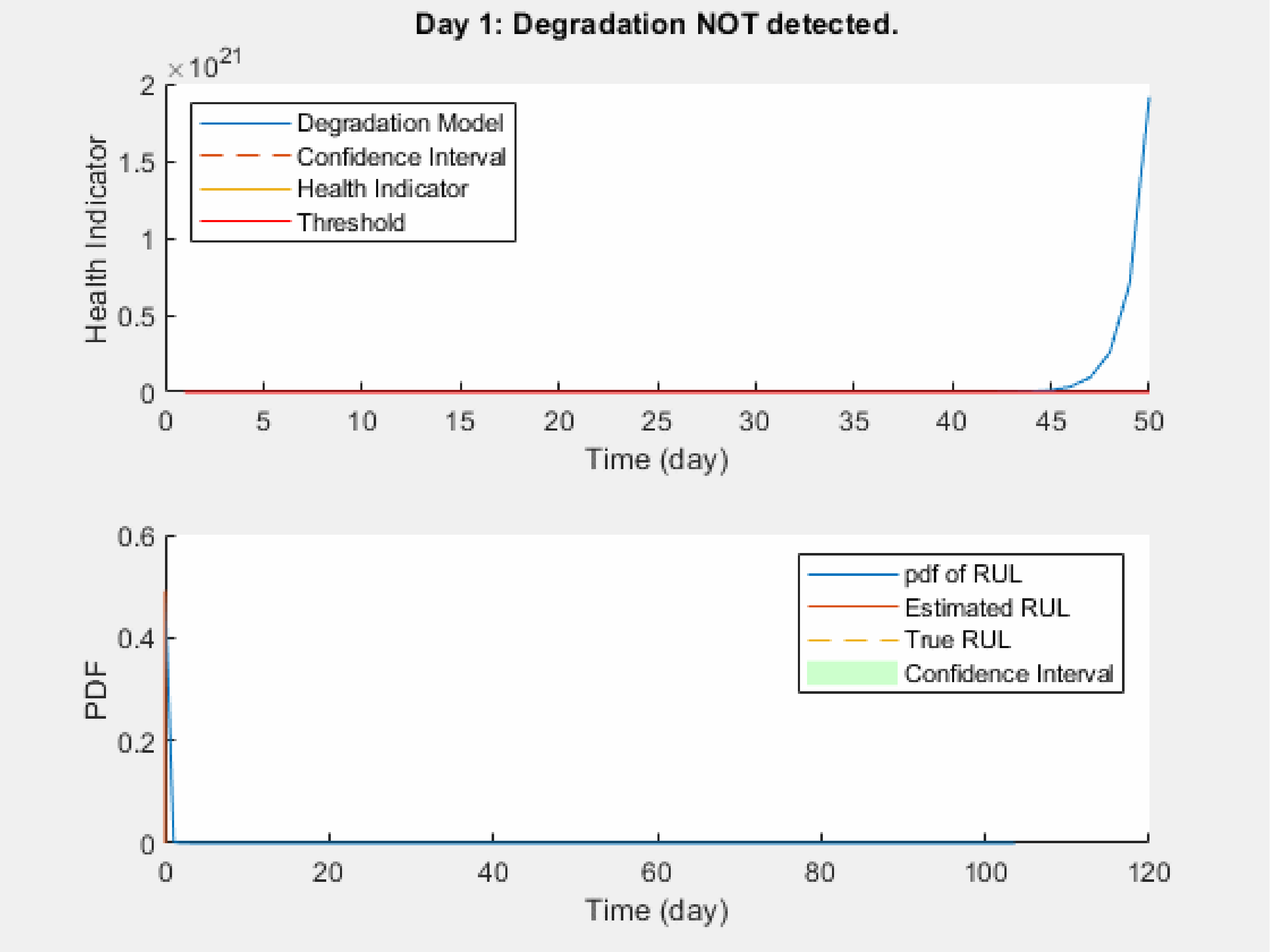


Feature Extraction and Feature Importance Ranking

- Time Domain Features
  - mean, std, skewness, kurtosis
  - peak2peak, rms
  - crestFactor, shapeFactor
  - ImpulseFactor, MarginFactor
- Spectral Kurtosis related features
  - mean, std, skewness, kurtosis
- Feature Selection
  - monotonicity
  - prognosability
  - trendability



Fit Exponential Degradation Model



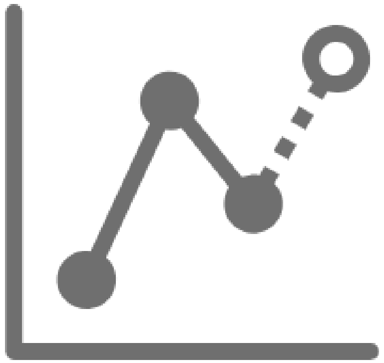
AI for Electrification

Reduced order modeling

Virtual sensors

Control strategy

Predictive maintenance



Energy forecasting

Energy Forecasting

- Load Forecasting
  - Improve system operation, planning and market participation
- Wind Forecasting
  - Understand uncertainty and risk for system operation, planning and market participation
- Solar Forecasting
  - Understand uncertainty and risk for system operation, planning and market participation
- Price Forecasting
  - Improve market participation and financial hedging

Who Needs Energy Forecasting?

Generation

Transmission

Distribution

Electricity Retail

Energy Traders

Large Electricity Consumers

(outside Energy industry)

Data Analytics Workflow

Access and Explore Data

- Files
- Databases
- Sensors

Preprocess Data

- Working with Messy Data
- Data Reduction/ Transformation
- Feature Extraction

Develop Predictive Model

- Model Creation e.g. Machine Learning
- Parameter Optimization
- Model Validation

Scale Up

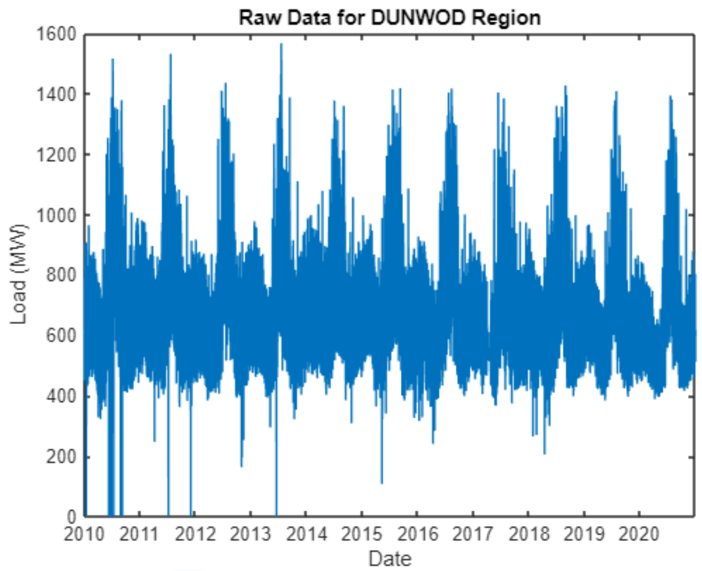
- Big Data Models and Architectures
- Clusters and Hardware
- Cloud

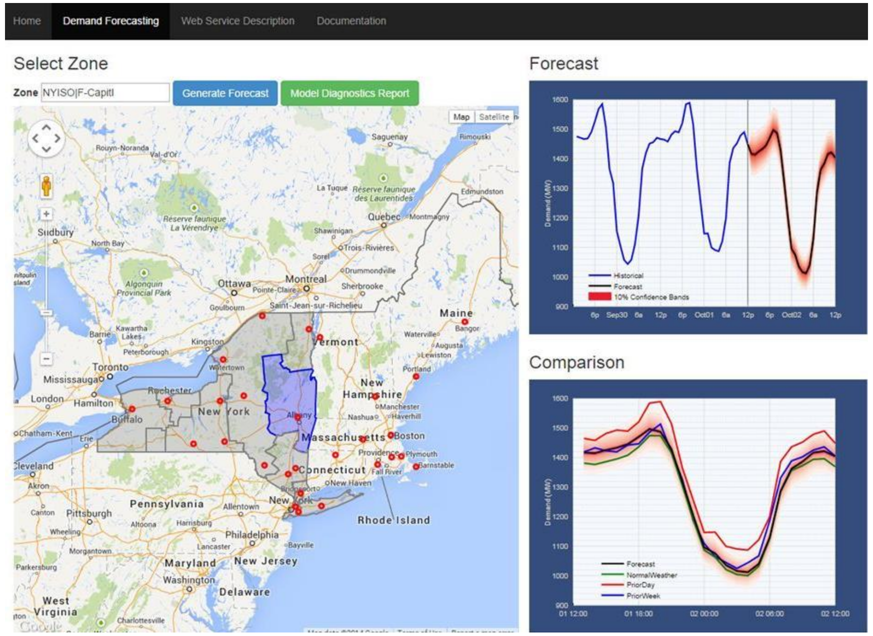
Integrate with Production Systems

- Desktop Apps
- Enterprise Scale Systems
- Embedded Devices and Hardware

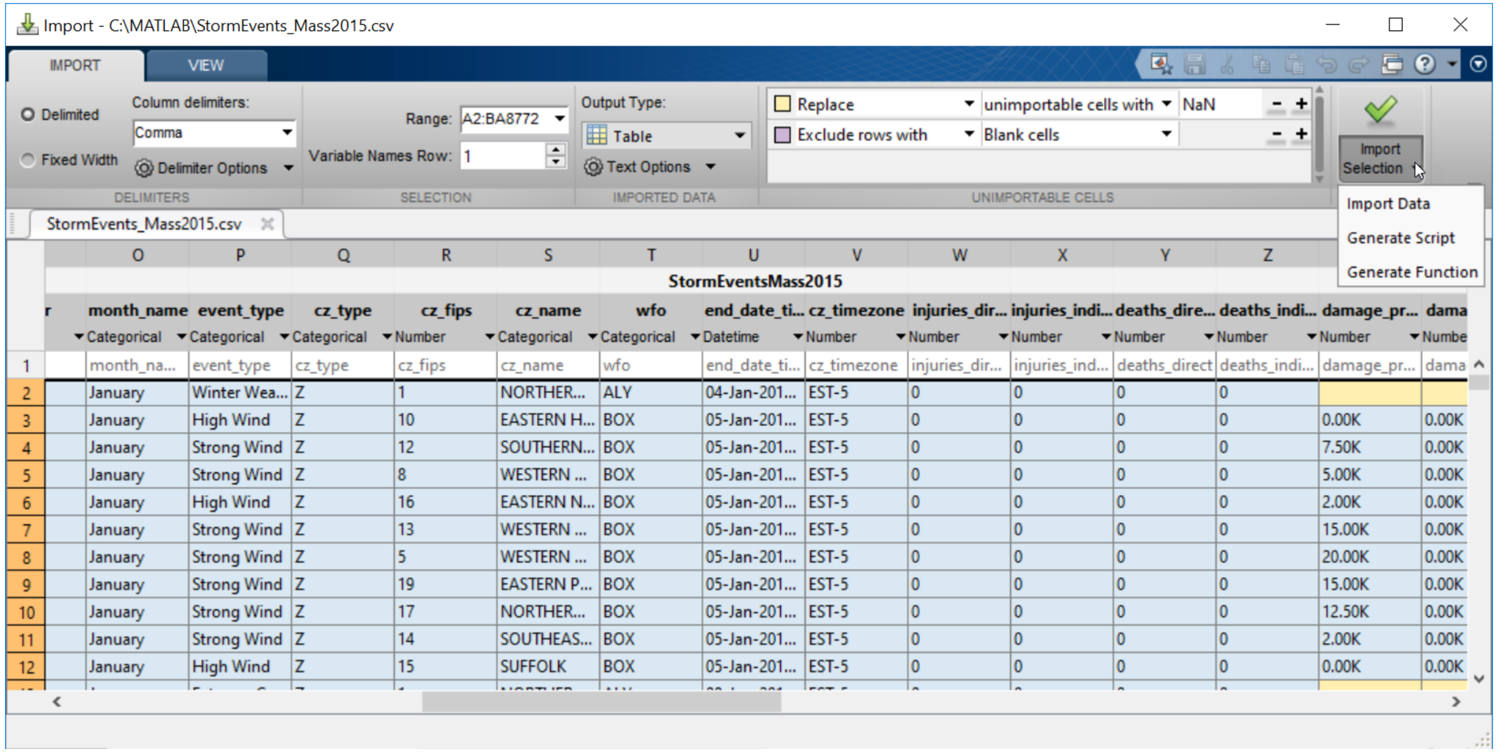
Example: Load Forecasting Study

- Motivation
  - plan how much electricity power plants will need to produce
  - insight into upcoming market dynamics
- Solution
  - Access historical load and weather data
  - Clean and preprocess data
  - Merge data from multiple sources
  - Perform time-series modeling to extract important predictors
  - Train a machine learning model to make predictions about future load





Access historical load and weather data



ThingSpeak

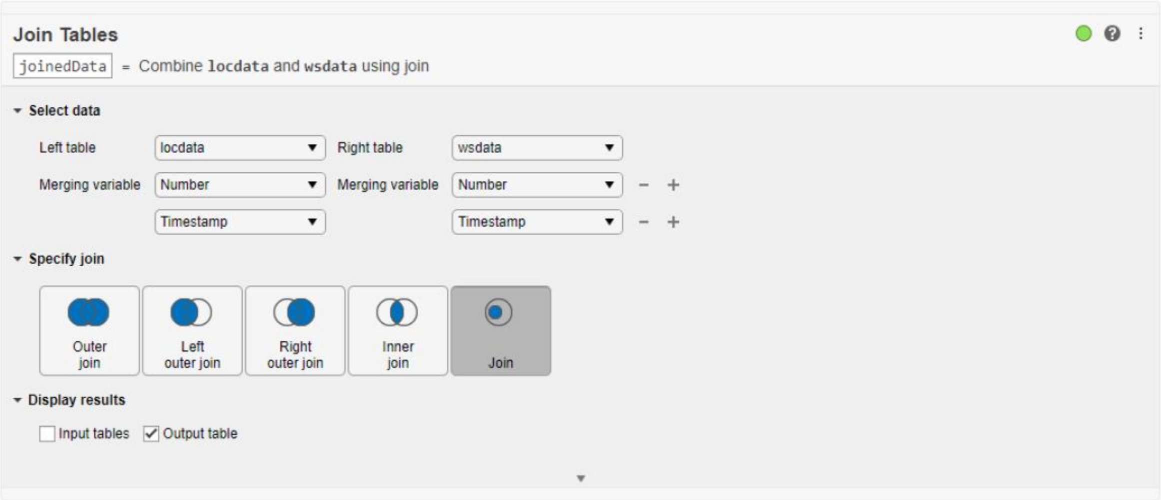
Use the best data type for the job

- table
  - mixed-type tabular data
  - flexible indexing, data organization
- timetable
  - time-stamped tabular data
  - indexing by time, time range
  - retiming, synchronizing
- datetime
  - representing a point in time
- categorical
  - discrete non-numeric data

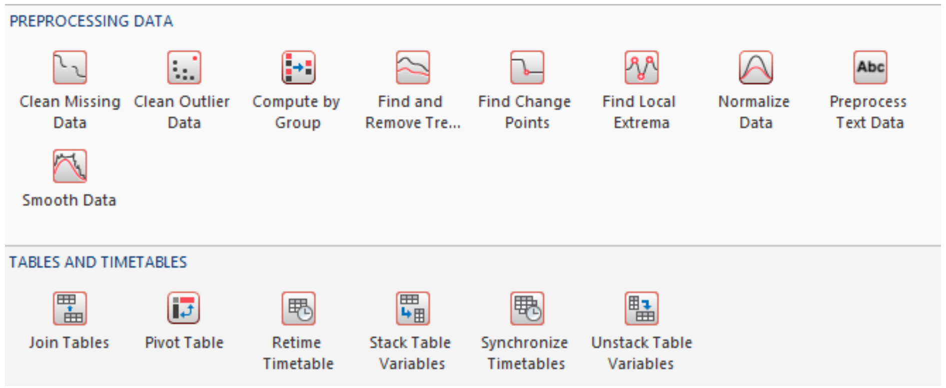
```
data(timerange("01-Jan-2017","17-Mar-2017"),:)
```

```
ans =  161×4 timetable
```

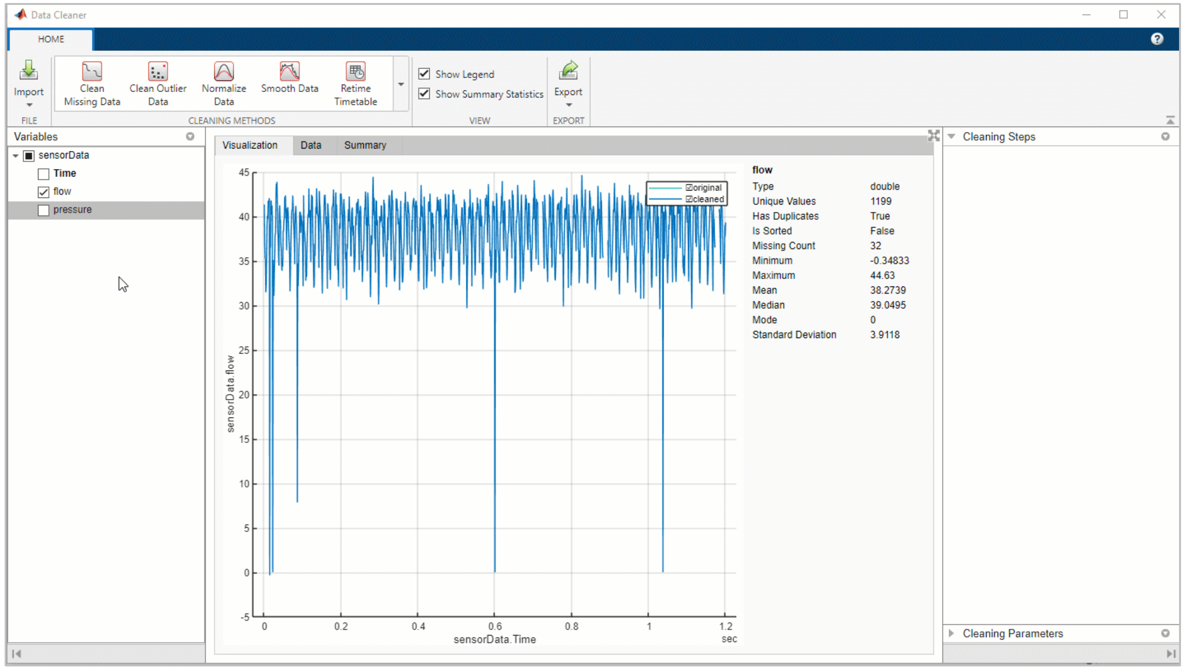
	begin_timestamp	state	event_type	event_narrative	damage_total
1	21-Jan-2017 13:02:00	GEORGIA	Thunderstorm…	"a tree was blown d…	0
2	21-Jan-2017 05:14:00	ALABAMA	Tornado	"the tornado first tou…	750
3	05-Jan-2017 04:00:00	OHIO	Winter Weather	"the county garage …	0
4	05-Mar-2017 18:00:00	OREGON	Snow	"there were reports …	0
5	04-Feb-2017 12:15:00	WYOMING	Wind	"the wydot sensor a…	0
6	08-Feb-2017 08:00:00	INDIANA	Winter Weather	"the observers locat…	0
7	18-Jan-2017 18:00:00	CALIFORNIA	Winter Weather	"a spotter in moonri…	0
8	07-Feb-2017 07:00:00	CALIFORNIA	Flood	"major flooding from…	0
9	13-Jan-2017 15:00:00	KANSAS	Ice Storm	"ice accretion was 3…	0



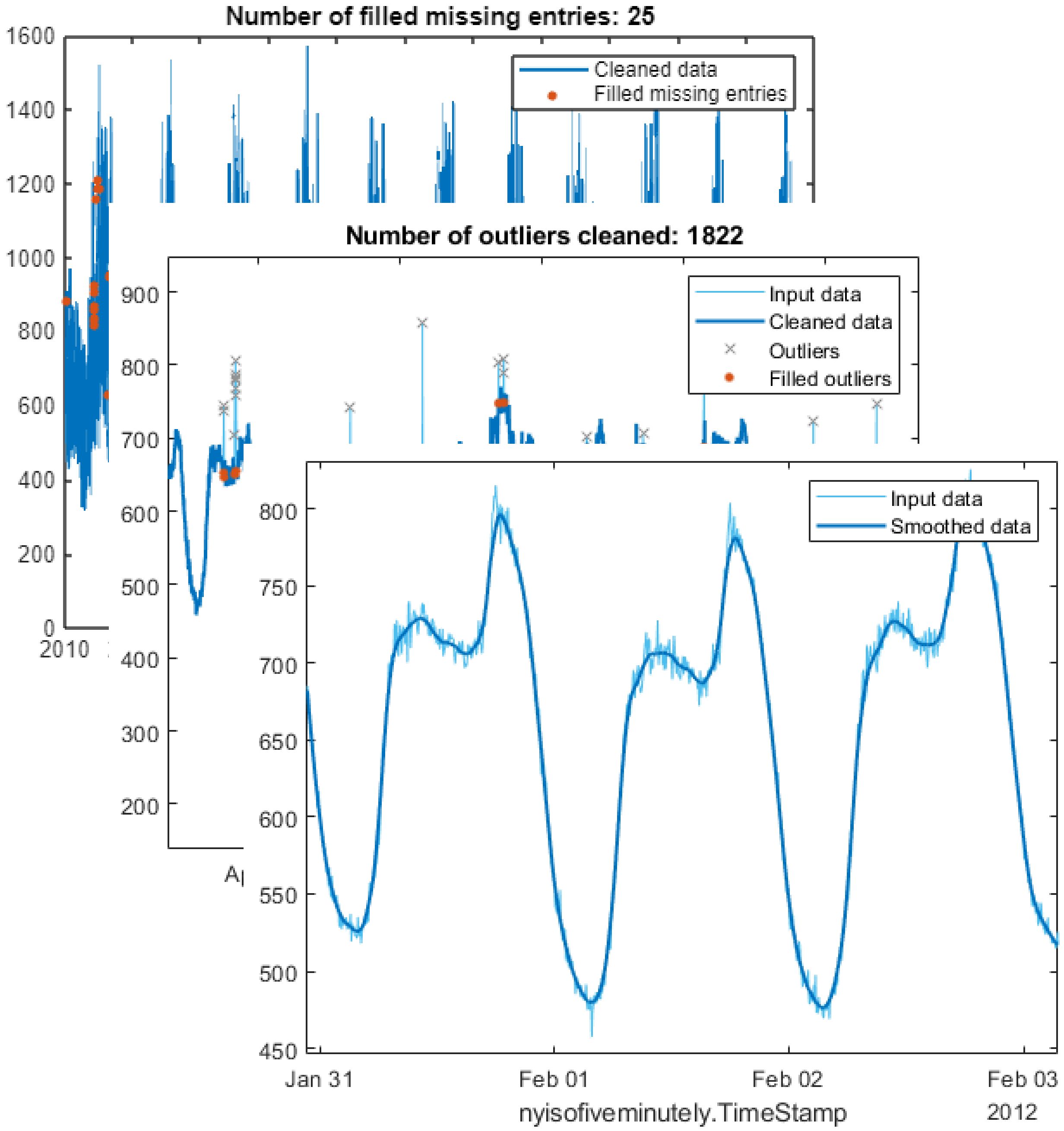
Preprocess Data



Live Editor Tasks



Data Cleaner App

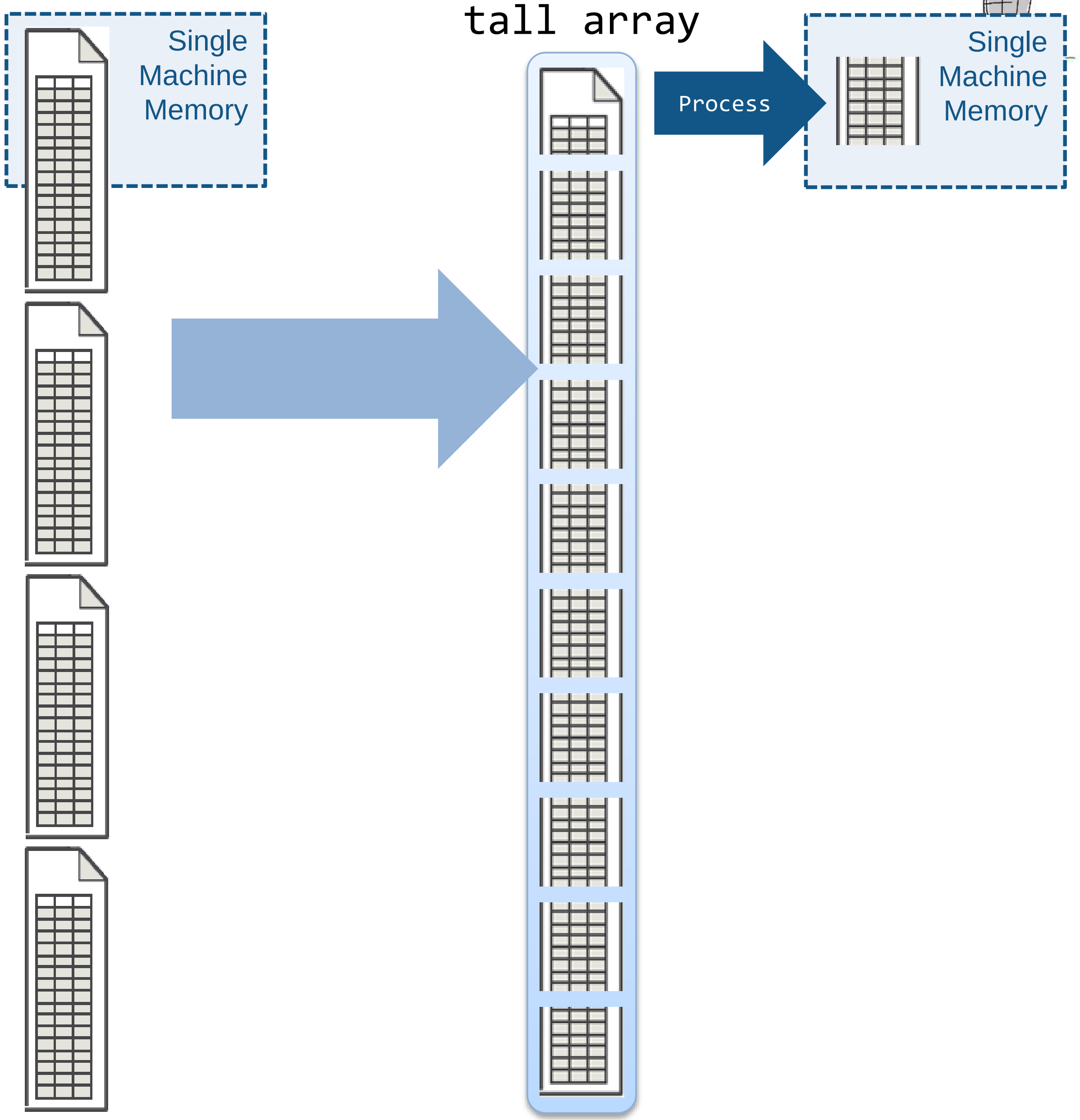


tall Arrays

- Automatically breaks data up into small “chunks” that fit in memory

- Tall arrays scan through the dataset one “chunk” at a time

- Processing code for tall arrays is the same as ordinary arrays

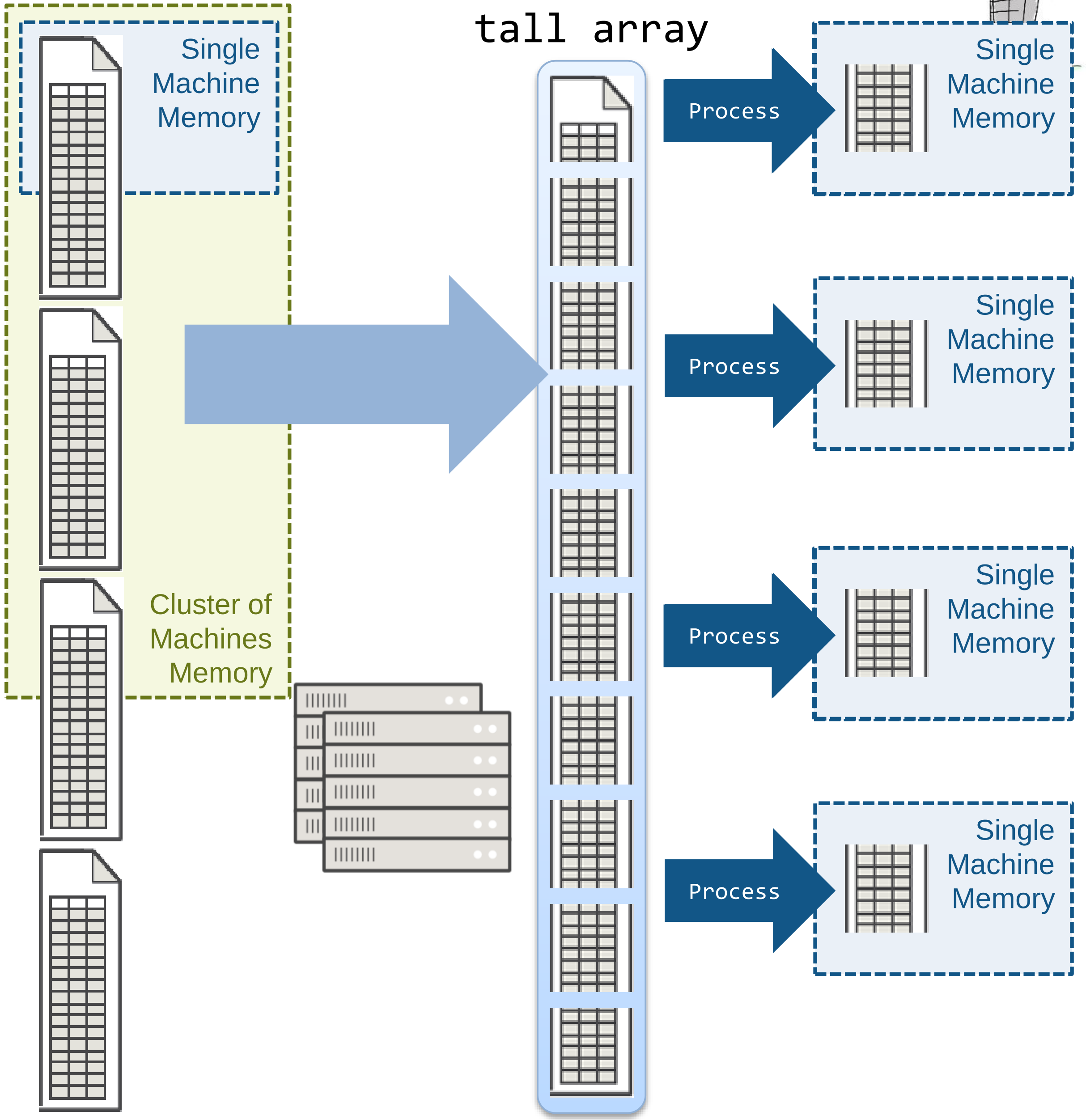


tall Arrays

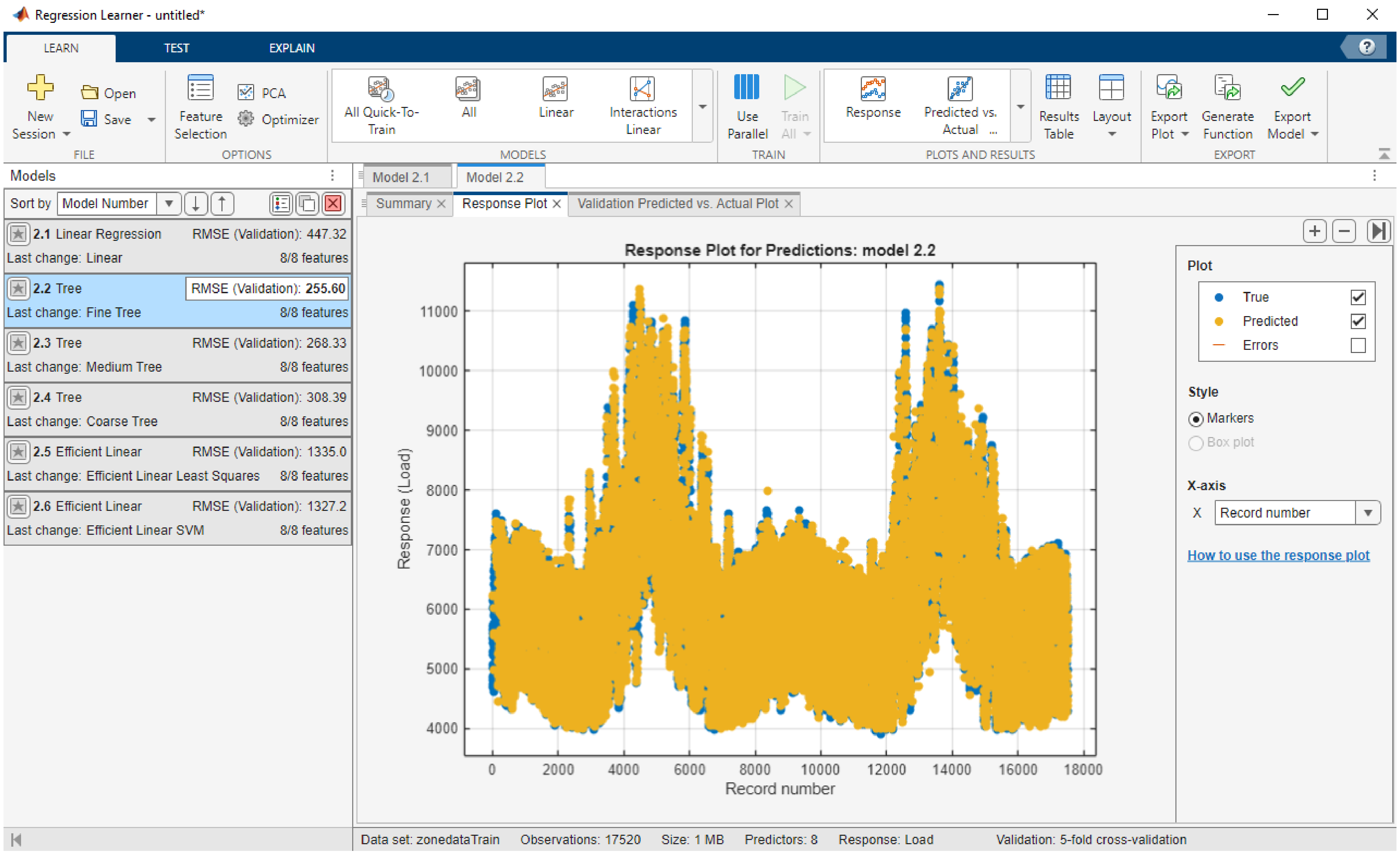
- With Parallel Computing Toolbox, process several “chunks” at once

- Can scale up to clusters with MATLAB Distributed Computing Server

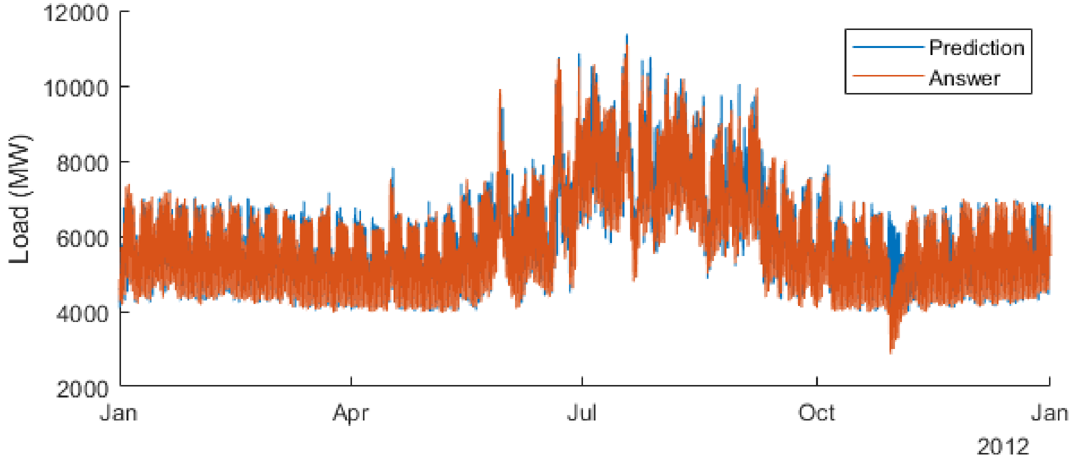
- Support for Spark and Hadoop

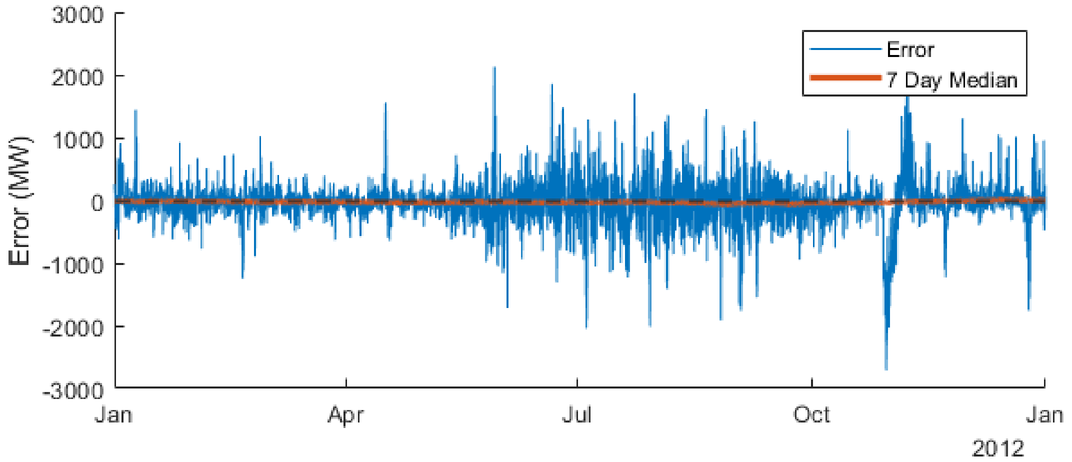


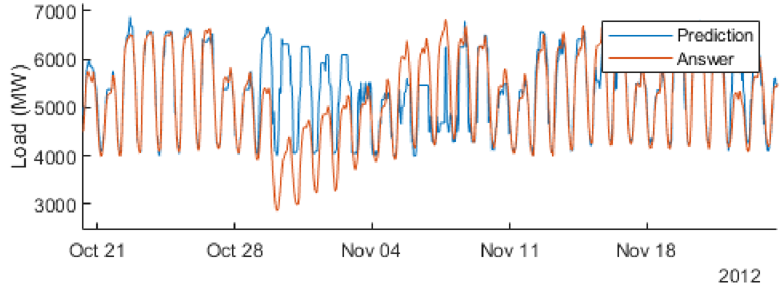
Train model with Regression Learner App

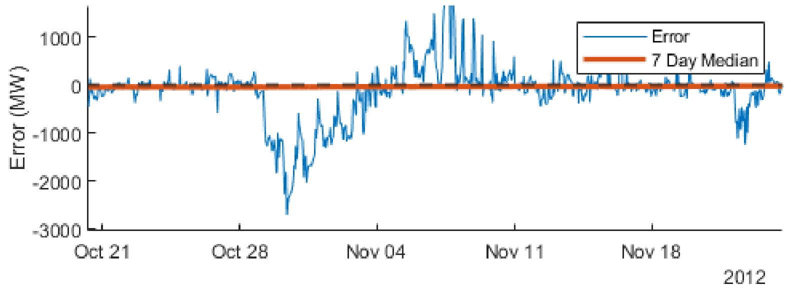


Load Prediction

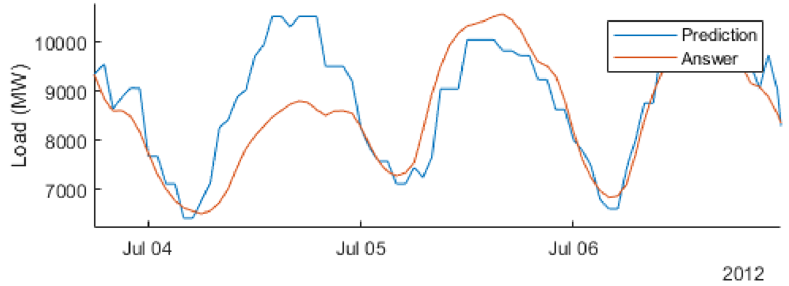


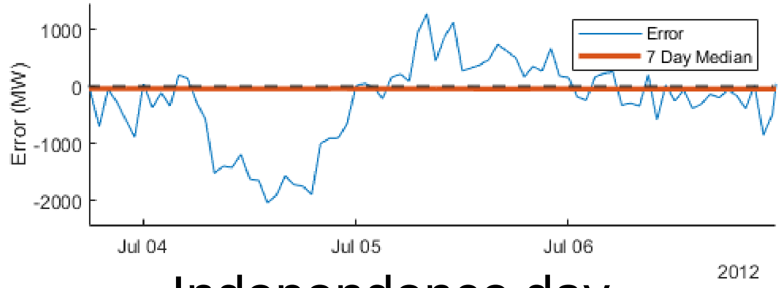






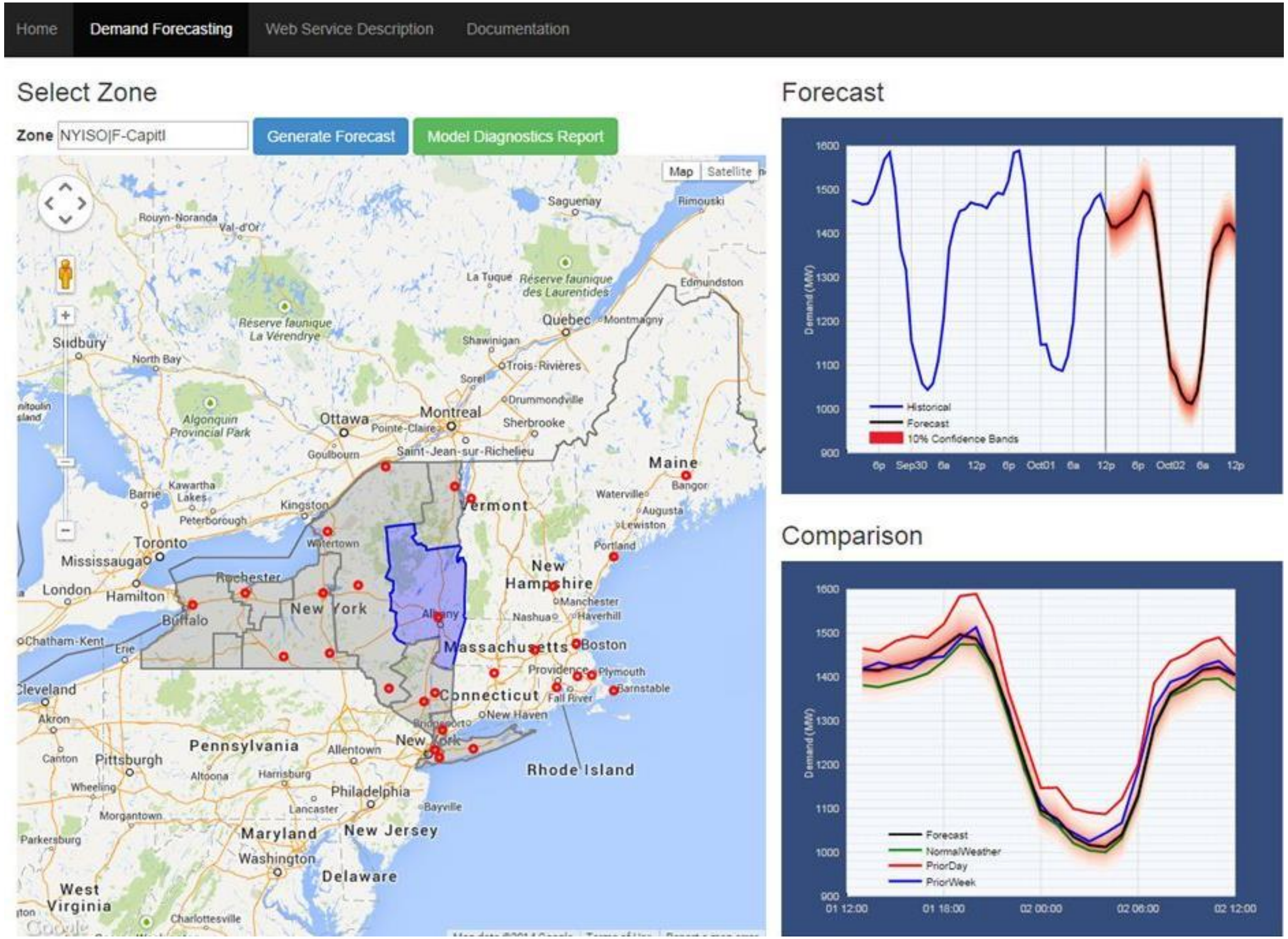
Hurricane



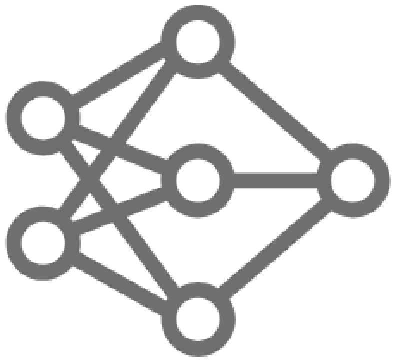


Independence day

Model Deployment

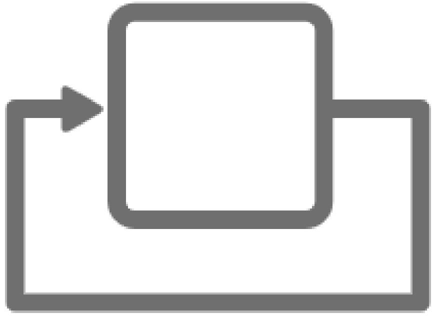


AI for Electrification



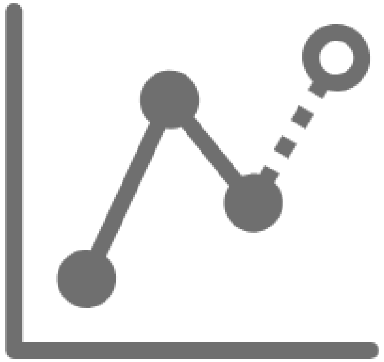
Reduced order modeling

Virtual sensors



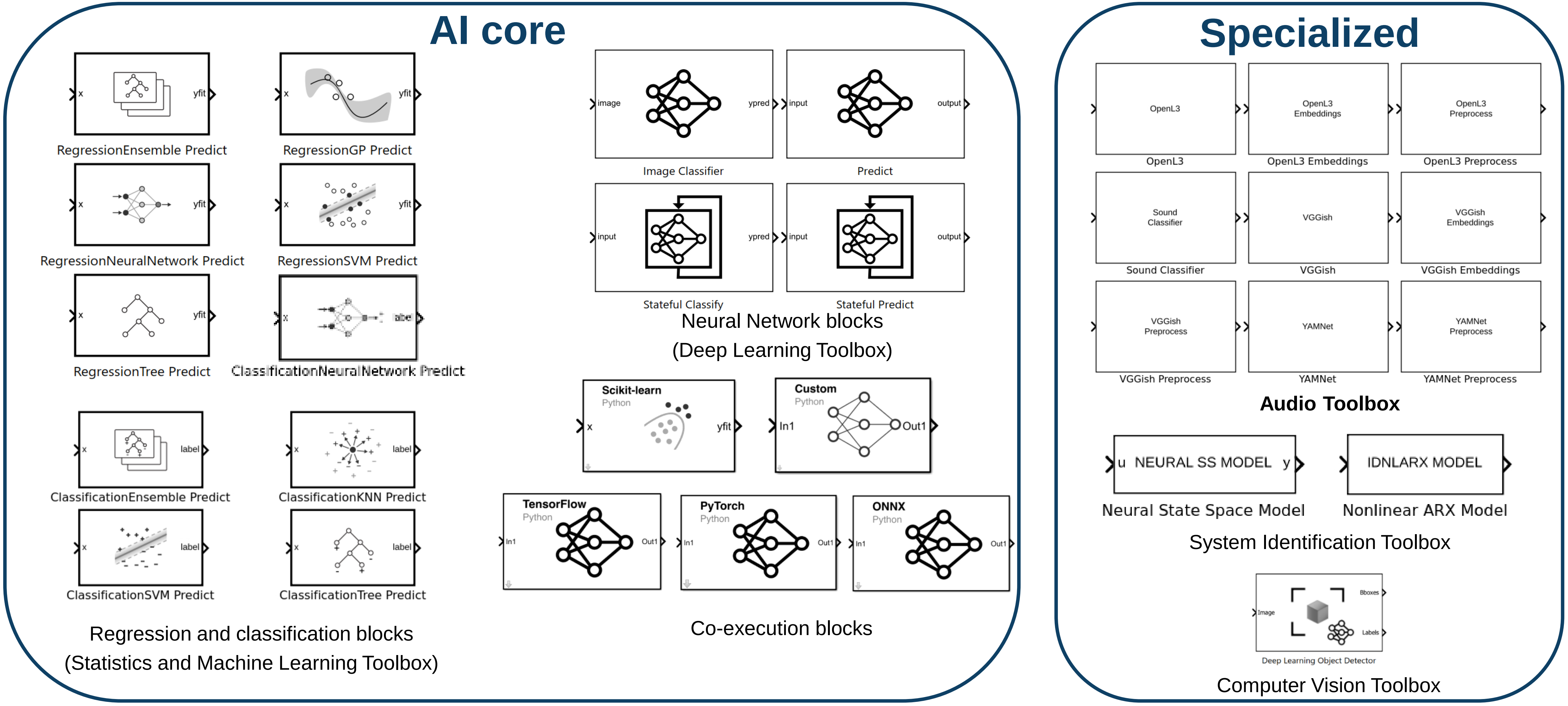
Control strategy

Predictive maintenance



Energy forecasting

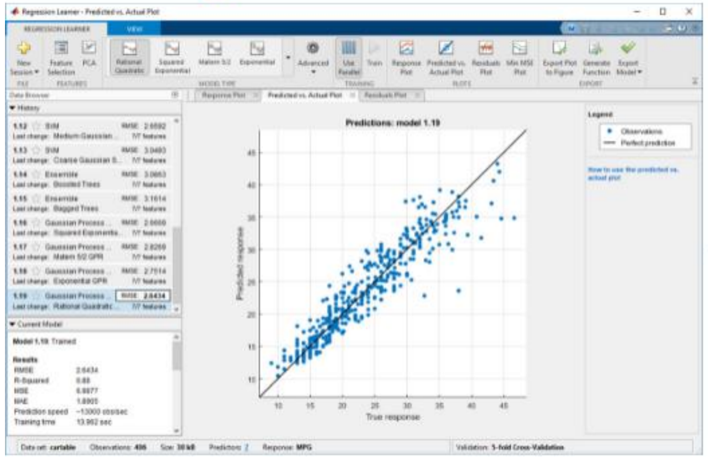
Simulink provides AI blocks



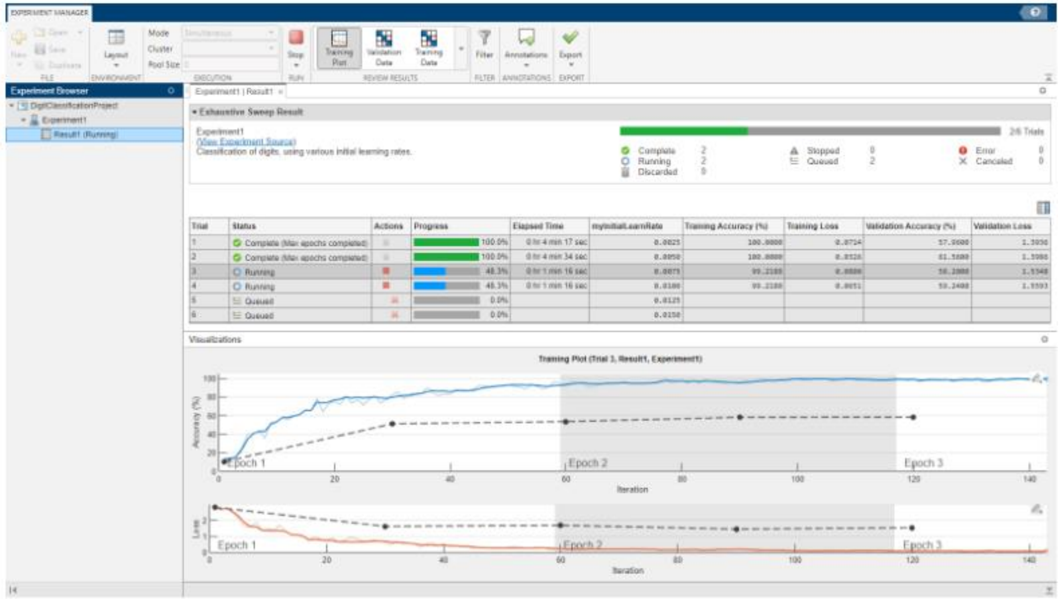
Optimize ML and DL models with Experiment Manager

Classification Learner

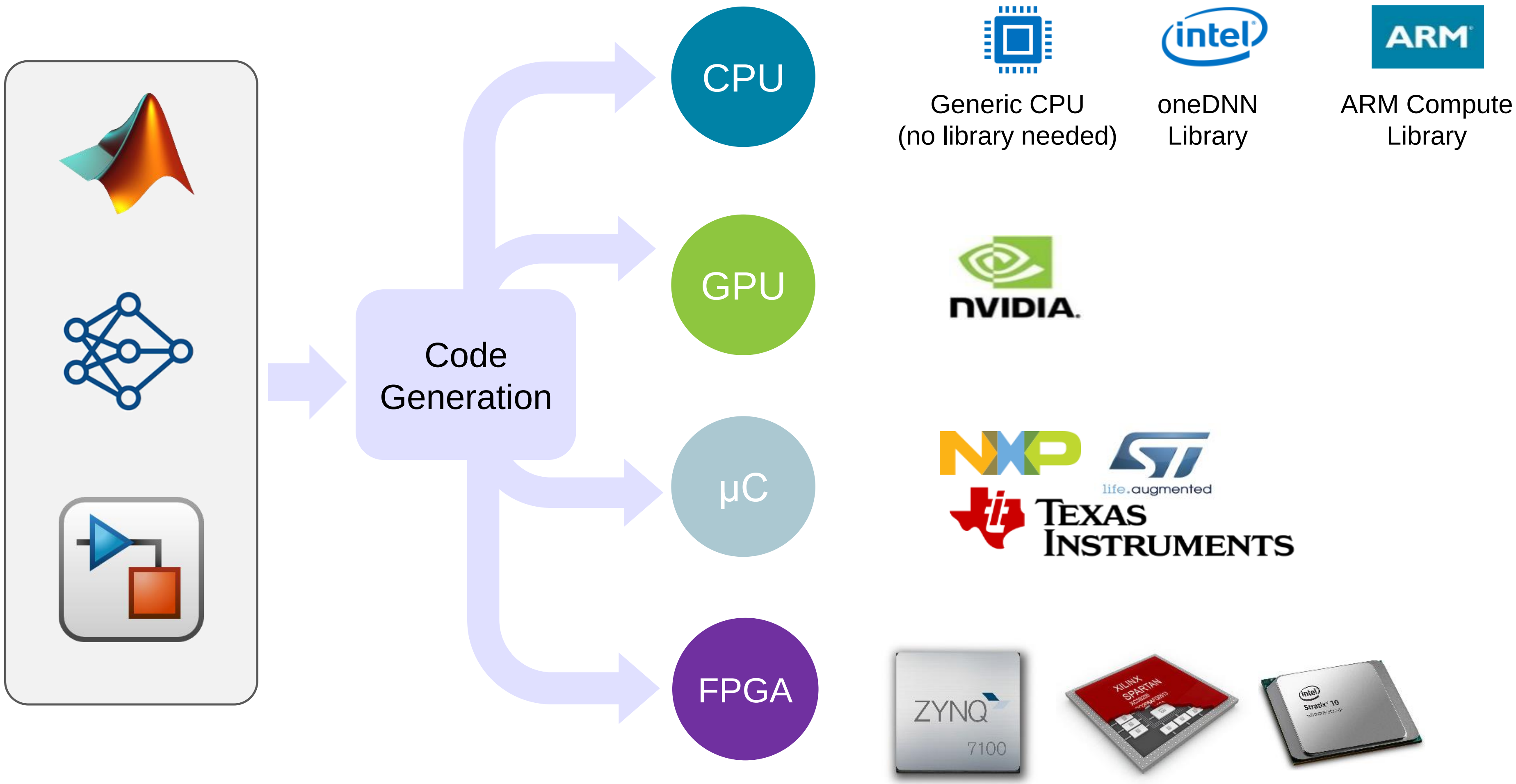
Regression Learner



Experiment Manager



Deploy models to target platforms



Thank you

Questions?