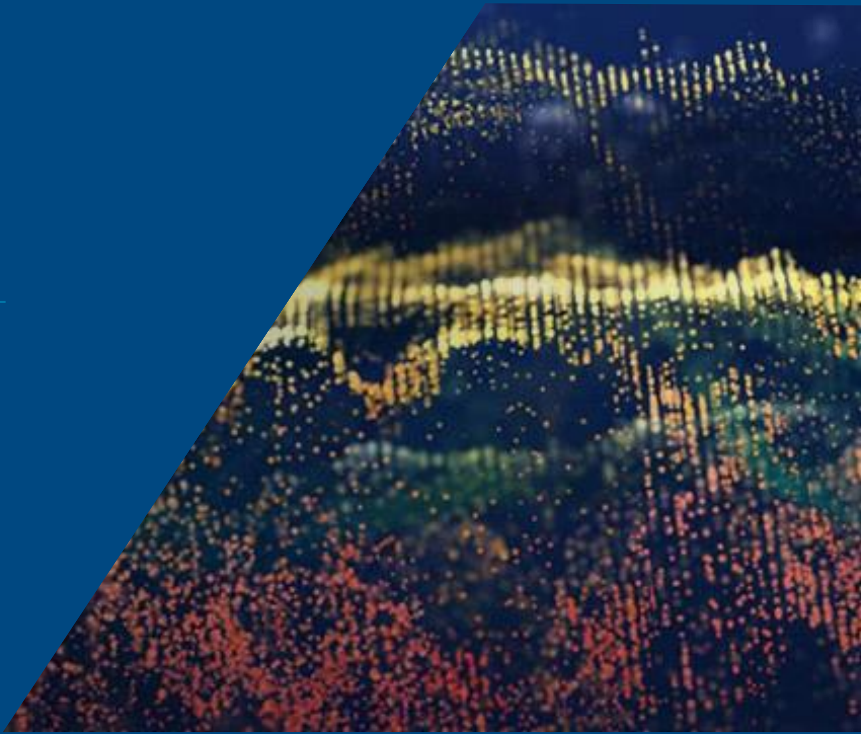
AI and the Power of Simulation

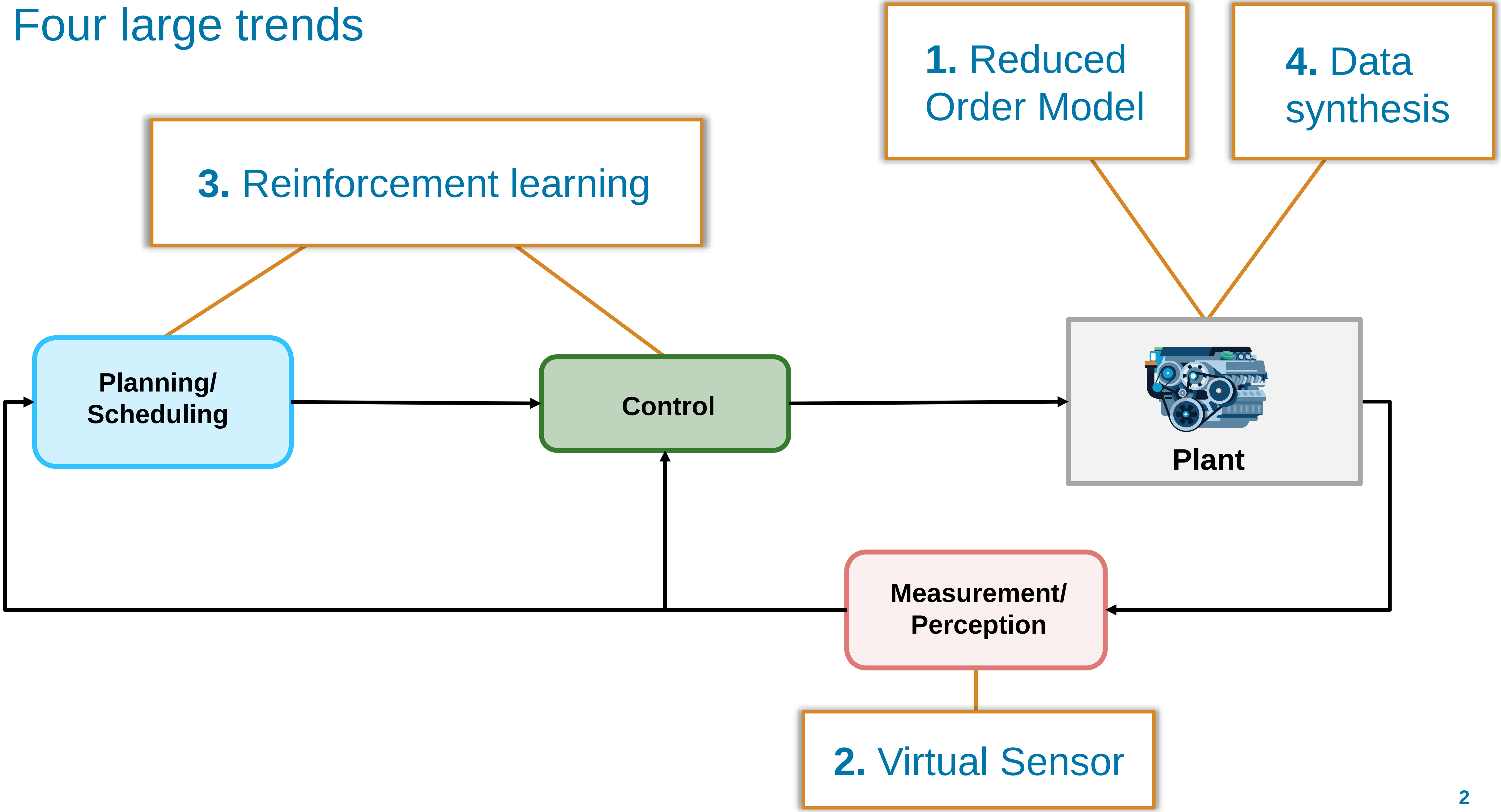
Antti Löytynoja
Senior Application Engineer, MATLAB
MathWorks
aloytyno@mathworks.com

Valtteri Forsman
Application Engineer, Simulink
MathWorks
vforsman@mathworks.com



© 2022 The MathWorks, Inc.

Four large trends



1. Reduced Order Model

4. Data synthesis

3. Reinforcement learning

Planning/ Scheduling

Control

Plant

Measurement/ Perception

2. Virtual Sensor

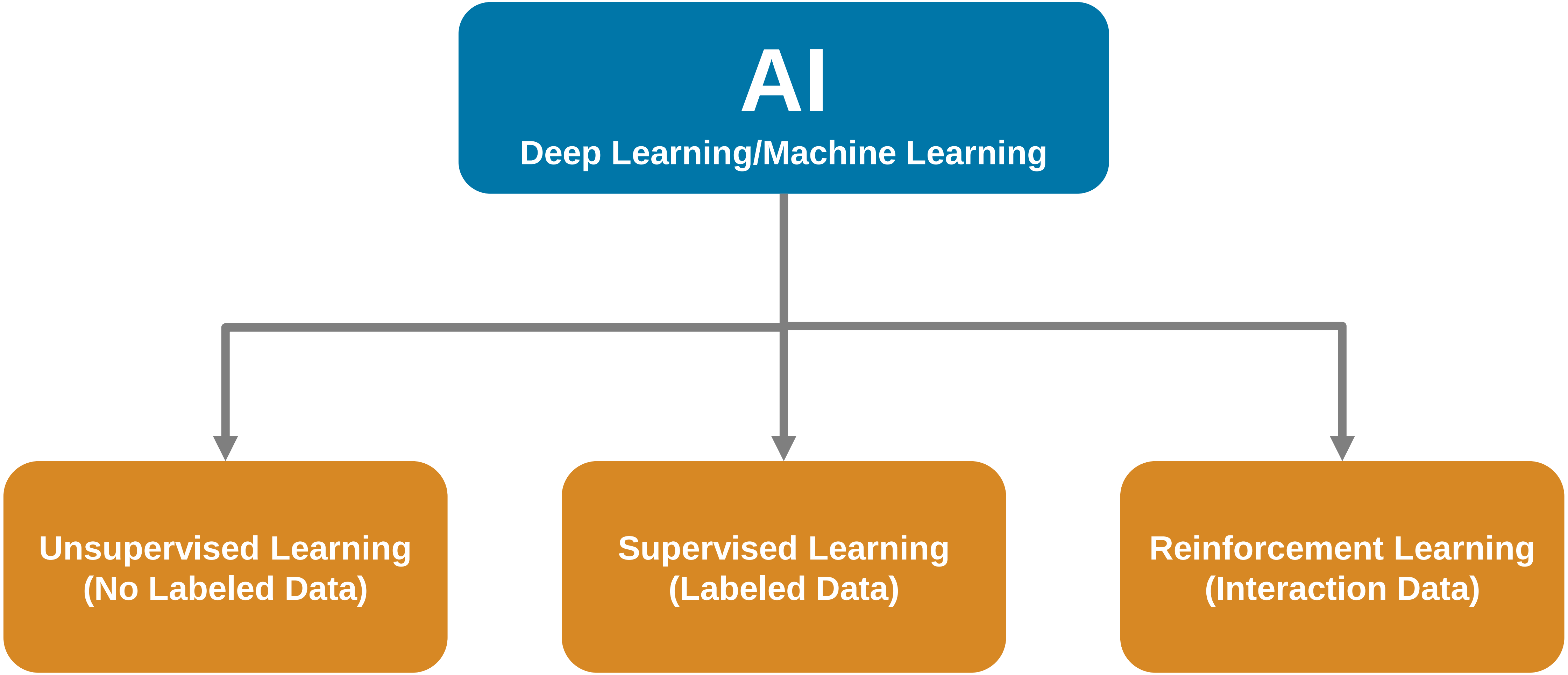
Why these trends?

They help you **overcome** common **challenges**

- Models/Algorithms are too slow to run in real time
- Models are unnecessarily complex
- Models are inaccurate
- Measurements are difficult to obtain

AI Landscape

Machine Learning vs Deep Learning vs Reinforcement Learning



AI

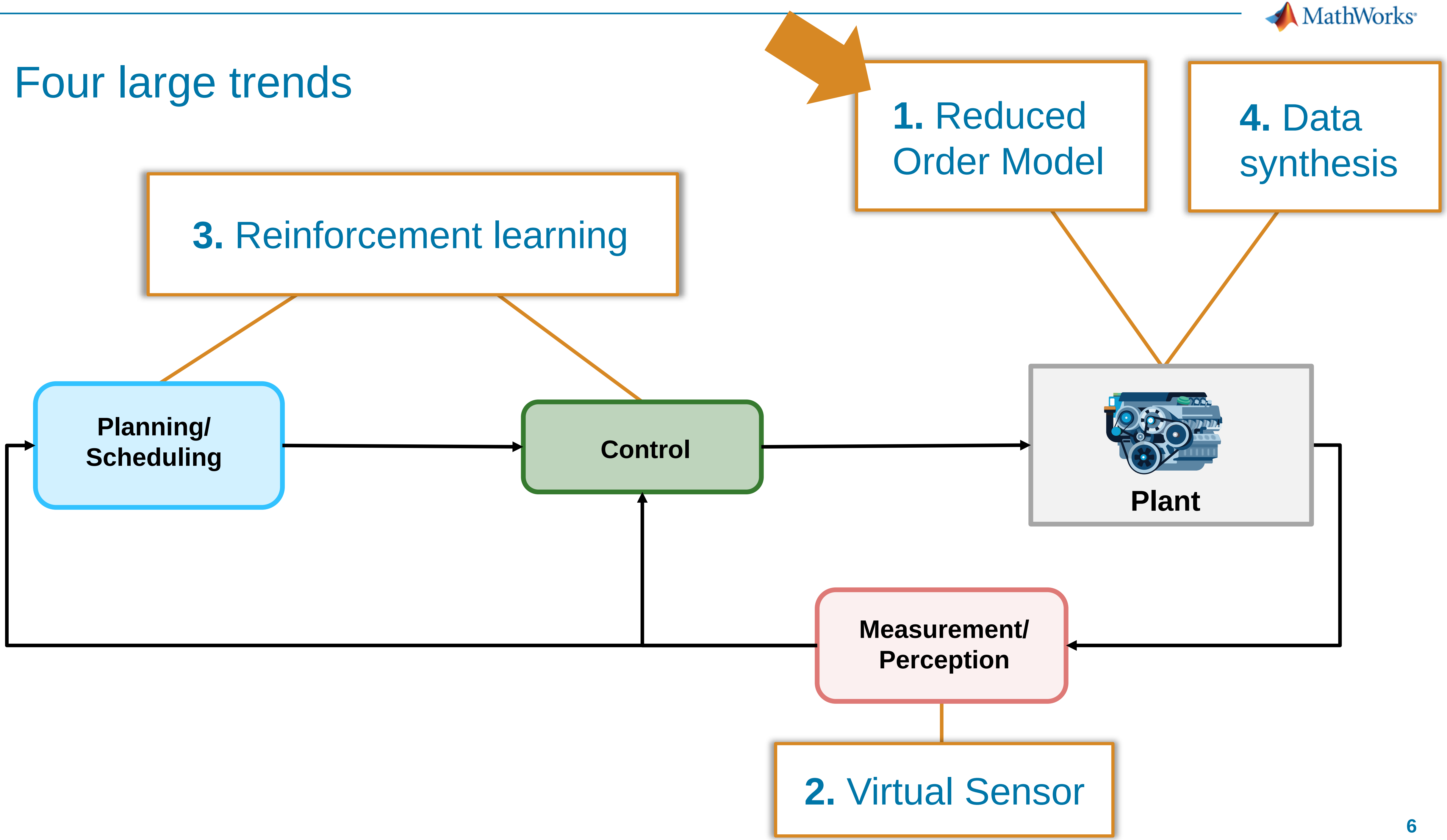
Deep Learning/Machine Learning

Unsupervised Learning
(No Labeled Data)

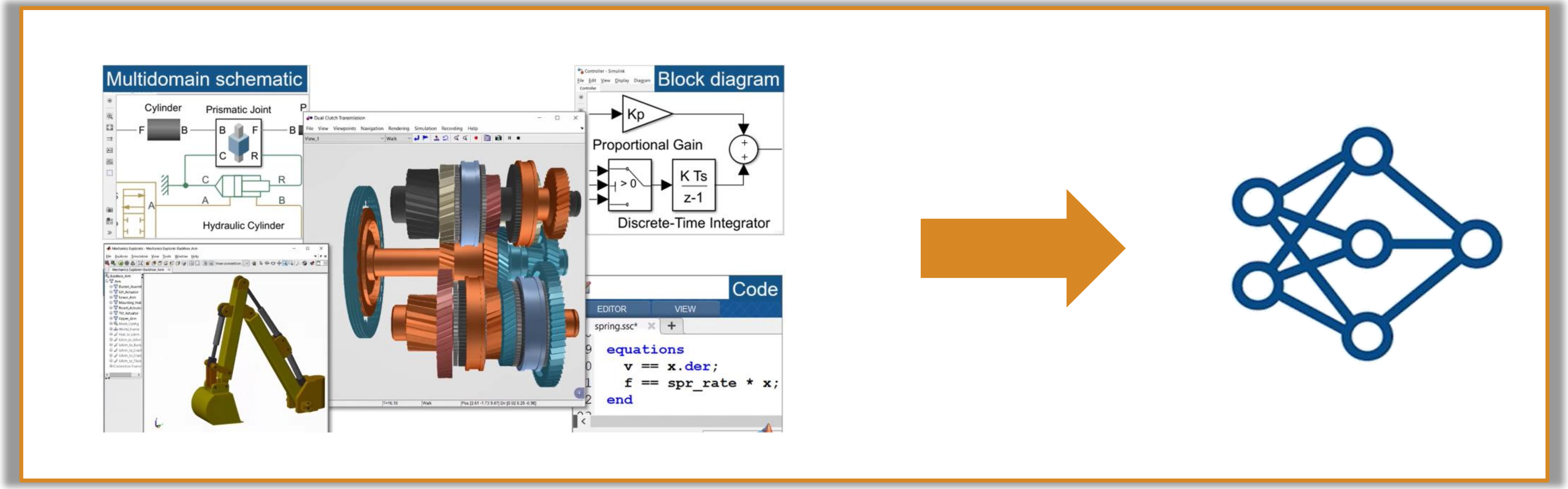
Supervised Learning
(Labeled Data)

Reinforcement Learning
(Interaction Data)

Four large trends



Trend 1 **Reduced Order Modelling (ROM)**



```
equations
  v == x.der;
  f == spr_rate * x;
end
```

Why?!

Create faster more light weight models of High-Fidelity Physics based Models for tasks when **speed** is more important than accuracy

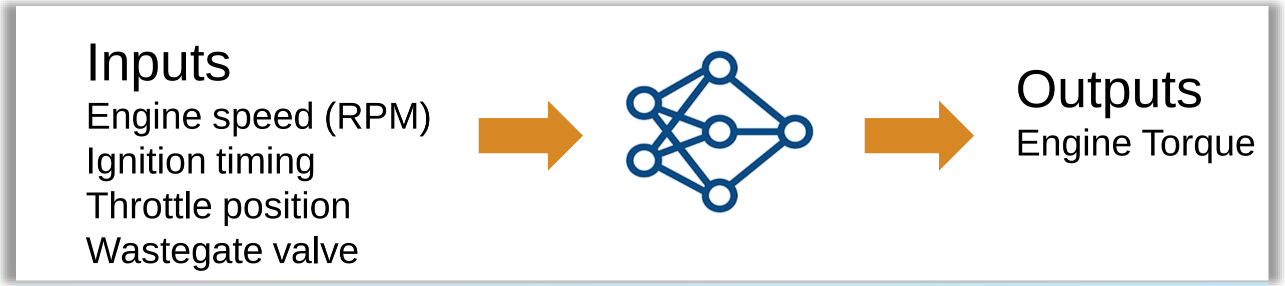
How?

Train on simulated data from the high-fidelity model and/or real data

Demo Description

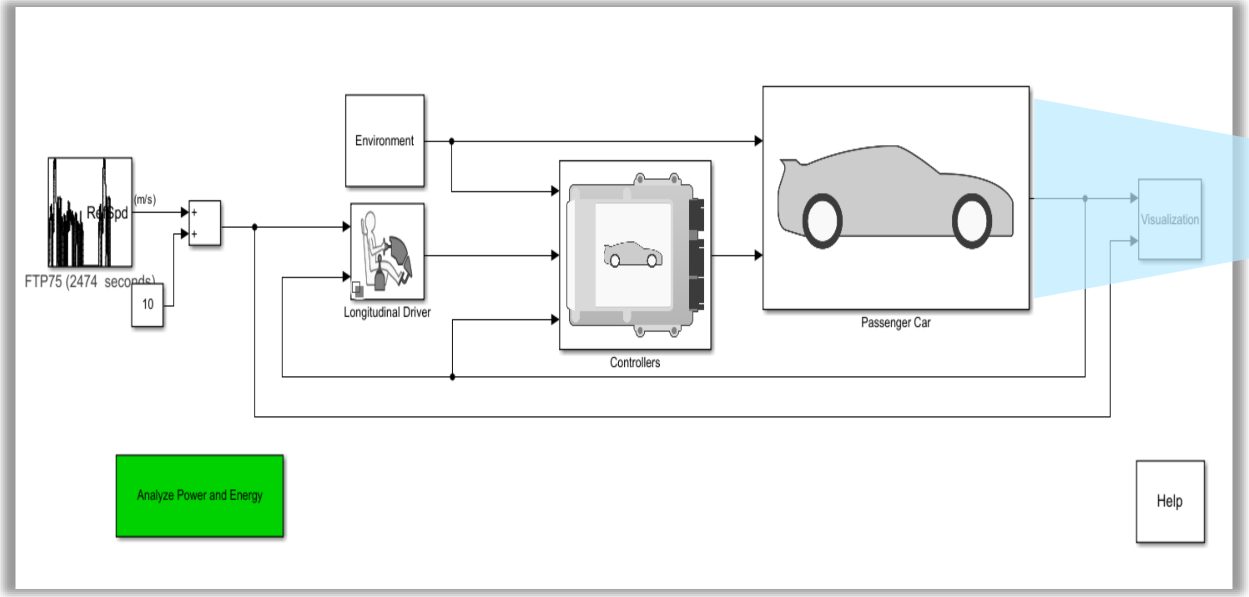
Engine torque estimation

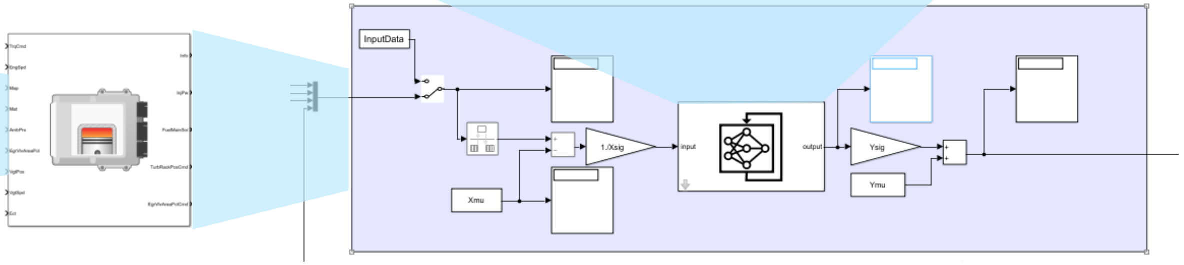
Engine model 10-50x faster
Overall simulation 2.5x faster



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

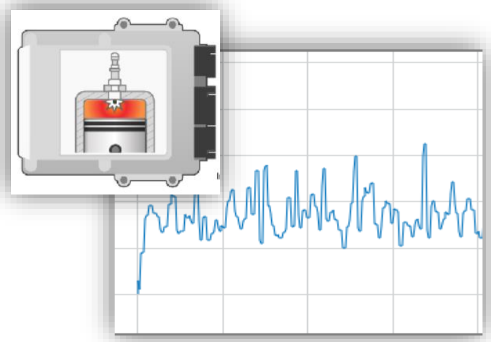
Outputs
Engine Torque

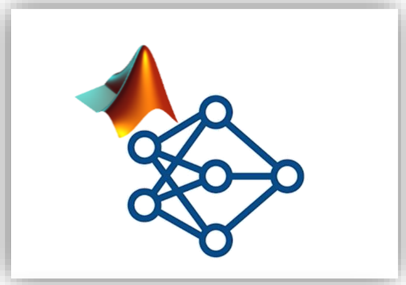


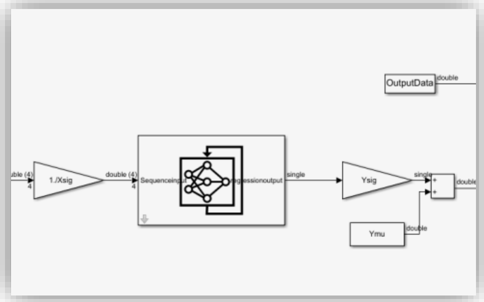


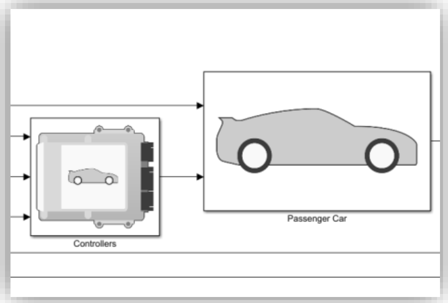
- Replacing the high-fidelity SI engine model with AI
- Speed up model to get real-time simulation

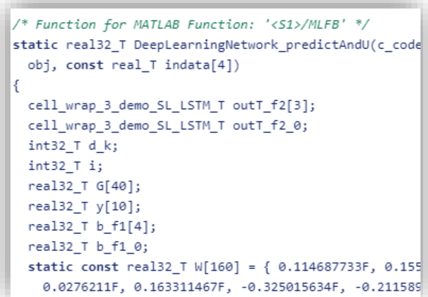
Workflow











```
/* Function for MATLAB Function: '<S1>/MLFB' */
static real32_T DeepLearningNetwork_predictAndU(c_code
  obj, const real_T indata[4])
{
  cell_wrap_3_demo_SL_LSTM_T outT_f2[3];
  cell_wrap_3_demo_SL_LSTM_T outT_f2_0;
  int32_T d_k;
  int32_T i;
  real32_T G[40];
  real32_T y[10];
  real32_T b_f1[4];
  real32_T b_f1_0;
  static const real32_T W[160] = { 0.114687733F, 0.155
    0.0276211F, 0.163311467F, -0.325015634F, -0.211589
```

Data Preparation

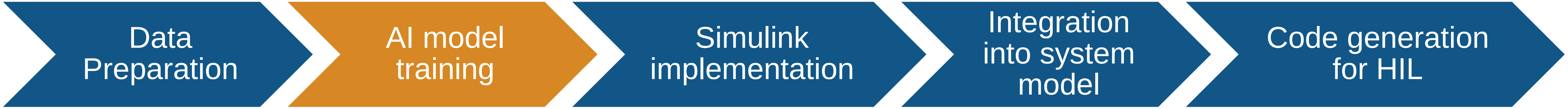
AI model training

Simulink implementation

Integration into system model

Code generation for HIL

AI model training



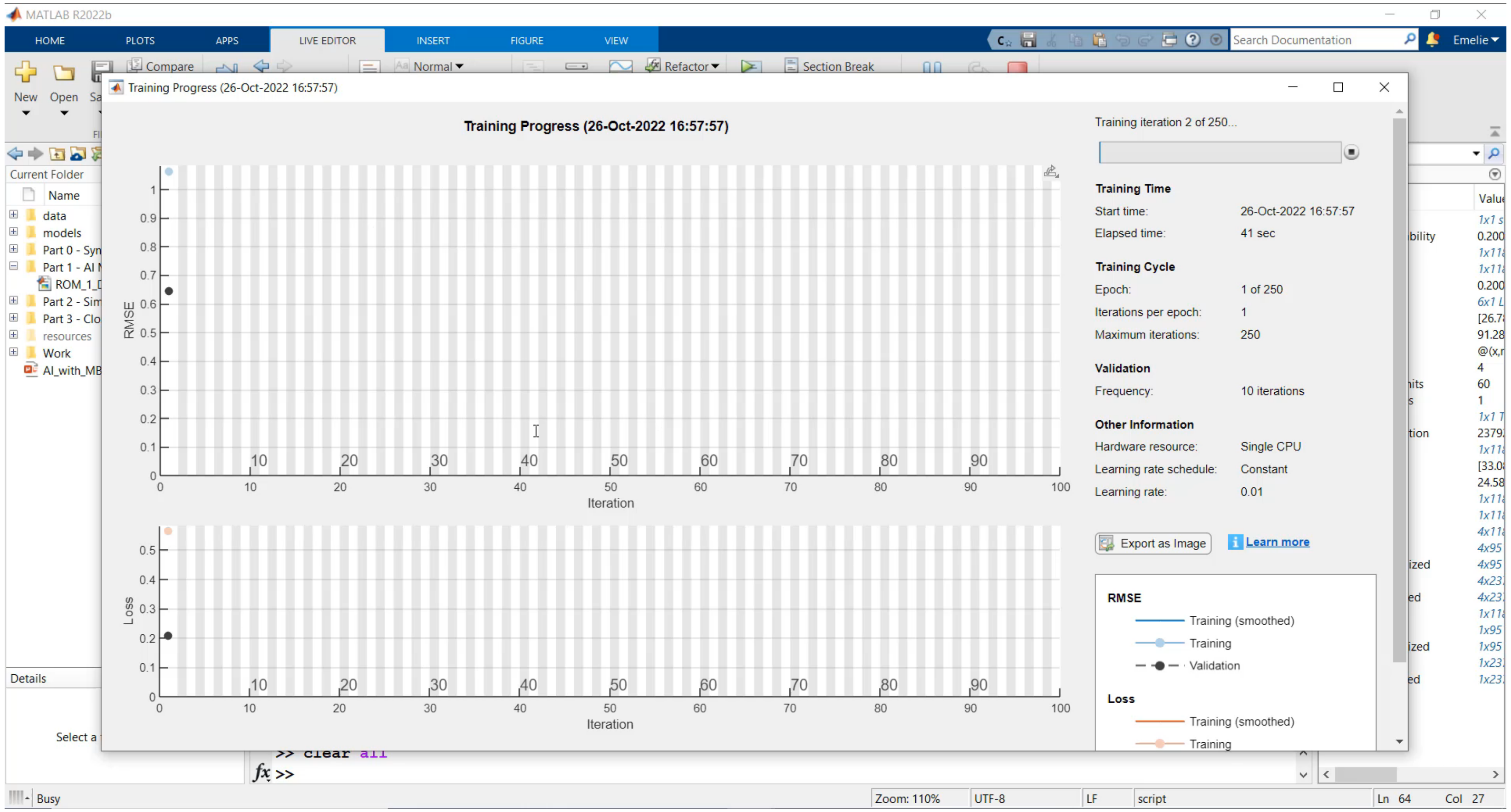
Data Preparation

AI model training

Simulink implementation

Integration into system model

Code generation for HIL



Training Progress (26-Oct-2022 16:57:57)

Training iteration 2 of 250...

Training Time

Start time: 26-Oct-2022 16:57:57
Elapsed time: 41 sec

Training Cycle

Epoch: 1 of 250
Iterations per epoch: 1
Maximum iterations: 250

Validation

Frequency: 10 iterations

Other Information

Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01

Export as Image | Learn more

RMSE

- Training (smoothed)
- Training
- Validation

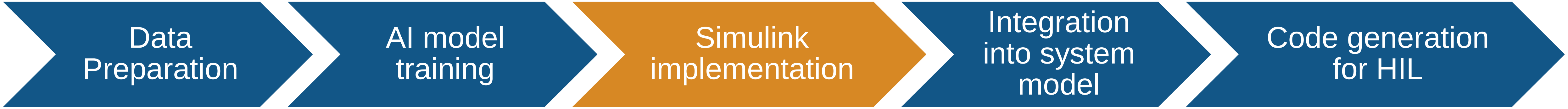
Loss

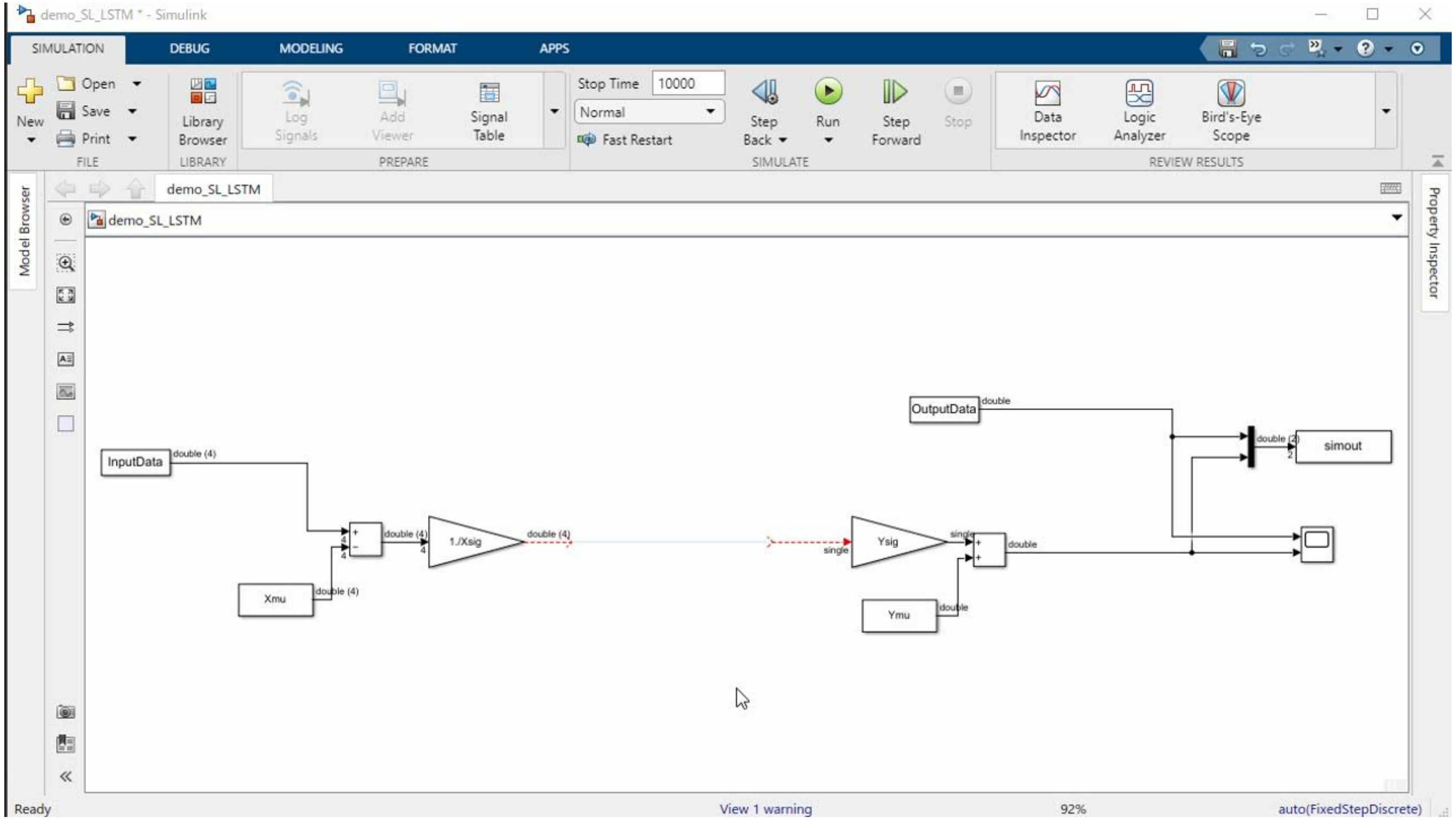
- Training (smoothed)
- Training

```
>> clear all
fx >>
```

```
save('net.mat','net')
```

Simulink implementation of learning models





Stop Time 10000

Normal

Ready

View 1 warning

92%

auto(FixedStepDiscrete)

Integrate into a system-level model for overall simulation

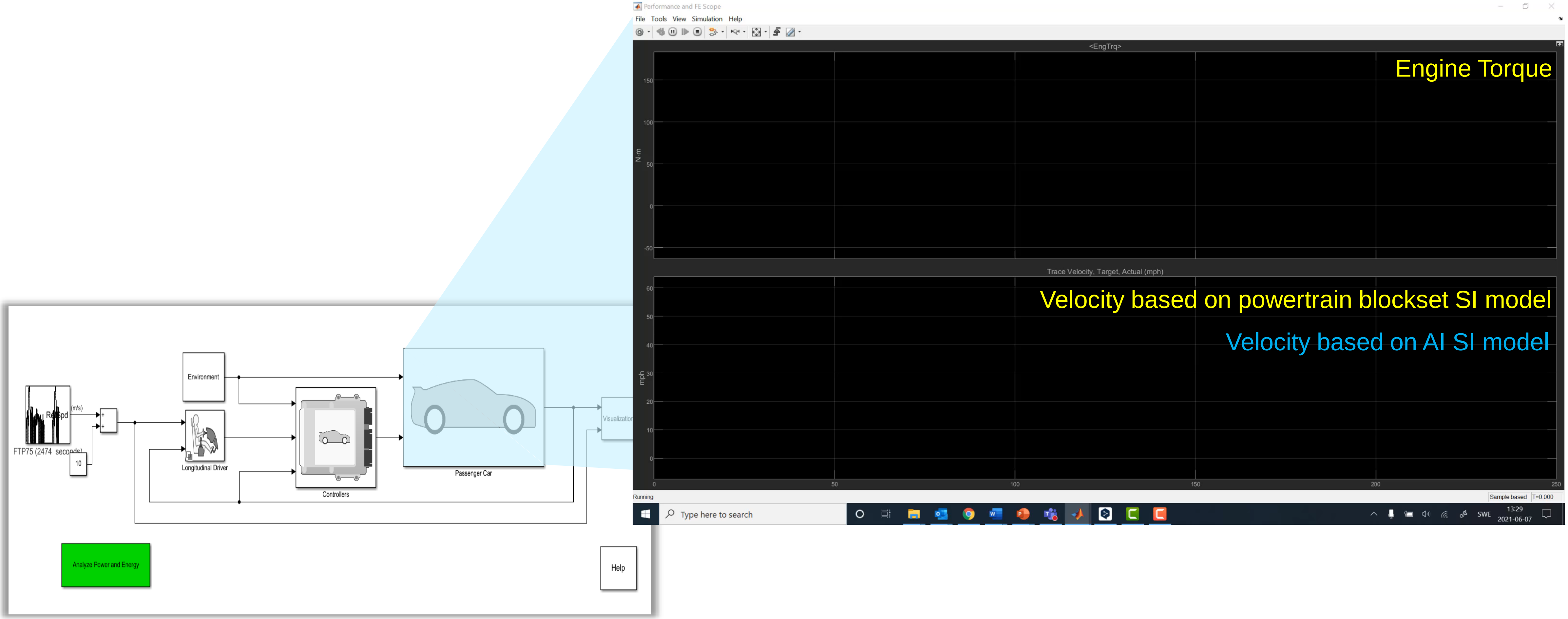
Data Preparation

AI model training

Simulink implementation

Integration into system model

Code generation for HIL

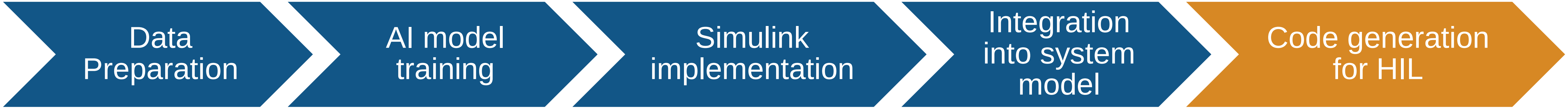


Engine Torque

Velocity based on powertrain blockset SI model

Velocity based on AI SI model

Generate code for LSTM model



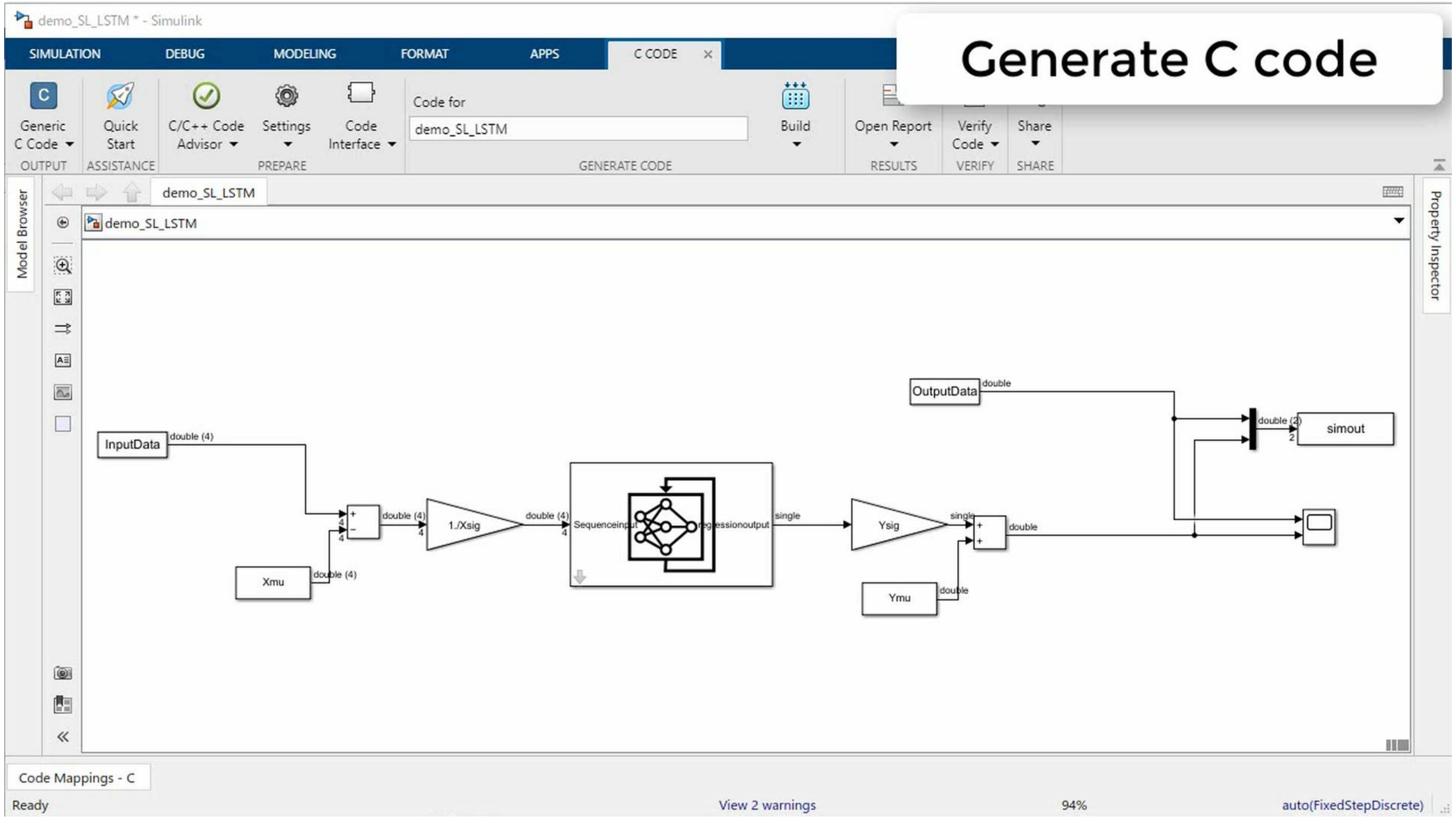
Data Preparation

AI model training

Simulink implementation

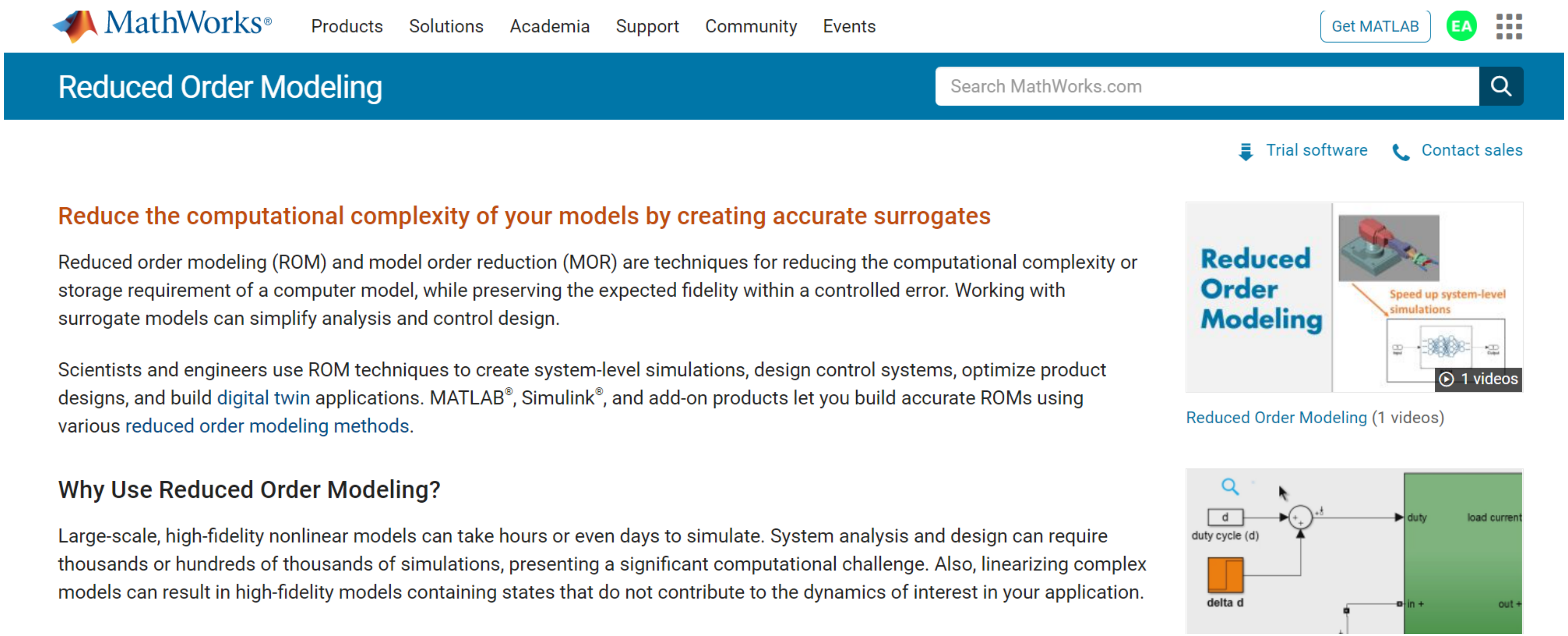
Integration into system model

Code generation for HIL



Generate C code

Want to learn more??



Reduced Order Modeling

Reduce the computational complexity of your models by creating accurate surrogates

Reduced order modeling (ROM) and model order reduction (MOR) are techniques for reducing the computational complexity or storage requirement of a computer model, while preserving the expected fidelity within a controlled error. Working with surrogate models can simplify analysis and control design.

Scientists and engineers use ROM techniques to create system-level simulations, design control systems, optimize product designs, and build digital twin applications. MATLAB®, Simulink®, and add-on products let you build accurate ROMs using various reduced order modeling methods.

Why Use Reduced Order Modeling?

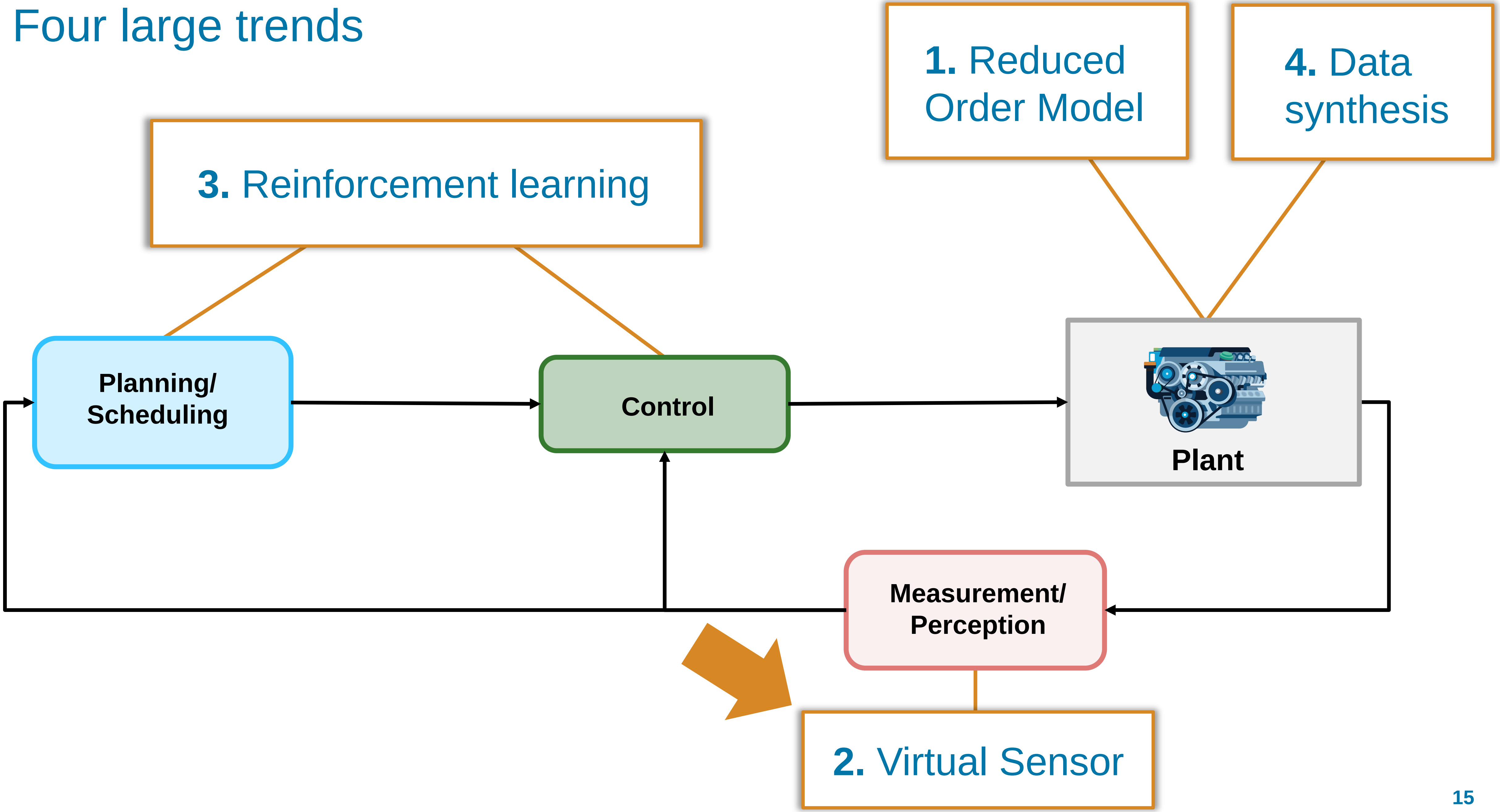
Large-scale, high-fidelity nonlinear models can take hours or even days to simulate. System analysis and design can require thousands or hundreds of thousands of simulations, presenting a significant computational challenge. Also, linearizing complex models can result in high-fidelity models containing states that do not contribute to the dynamics of interest in your application.

Reduced Order Modeling (1 videos)

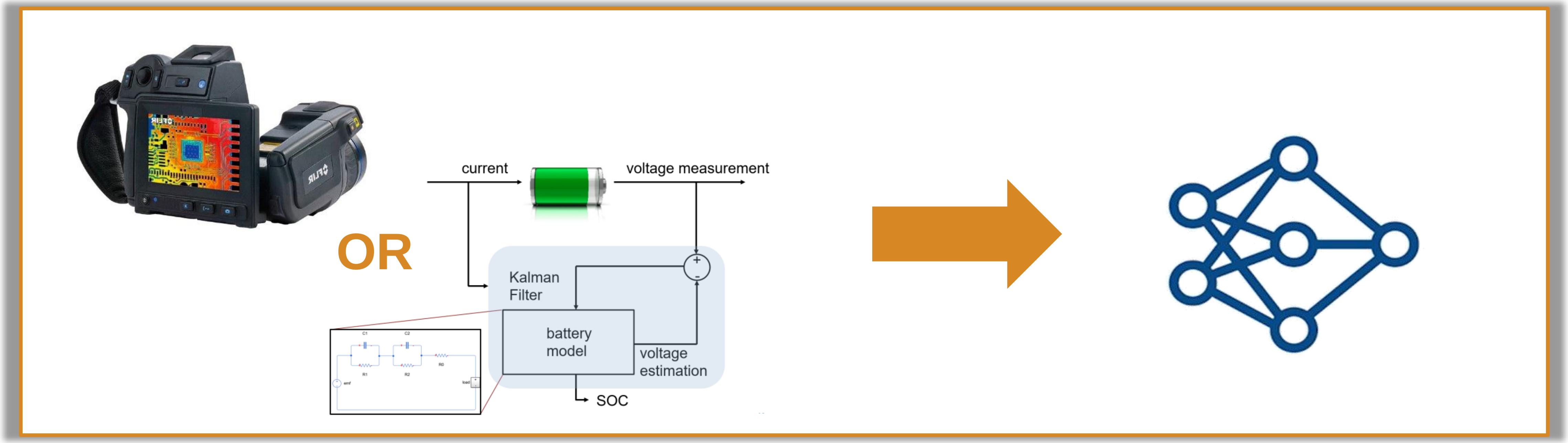
Link to webpage

Link to video

Four large trends



Trend 2 **AI based Virtual Sensors**



Why?! A physical sensor may be:

- Expensive
- Noisy
- Degrading over time
- Impossible to place
- …

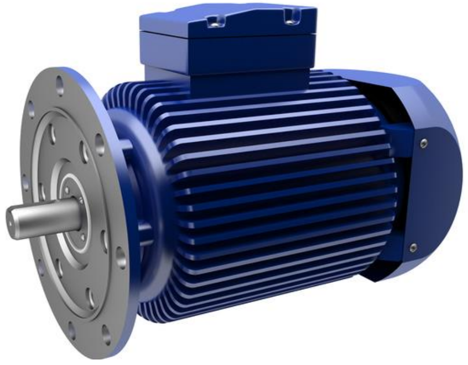
How?

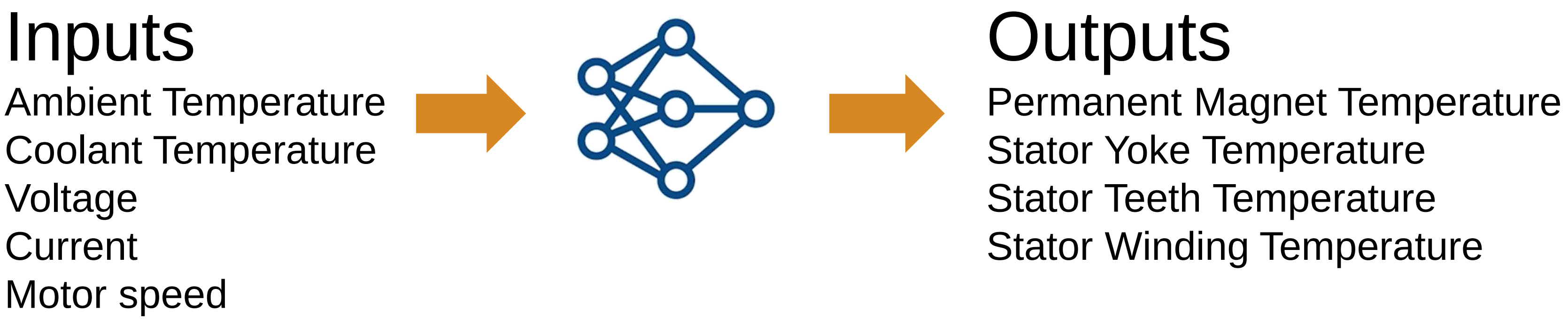
Train a model that can predict the wanted measurement using data from existing sensors

AI based Virtual Sensors for Temperature Estimation

Use lab data to replace **expensive** sensors with AI based sensor

- Permanent Magnet Synchronous Motor (electric motor)
- Temperature Estimation
- Data collected via contactless infrared sensor





Inputs

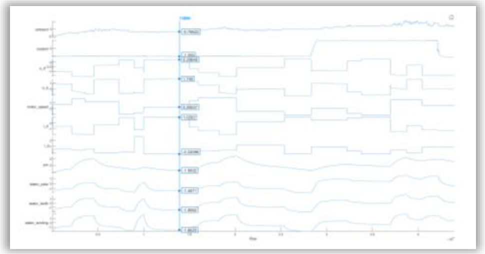
Ambient Temperature
Coolant Temperature
Voltage
Current
Motor speed

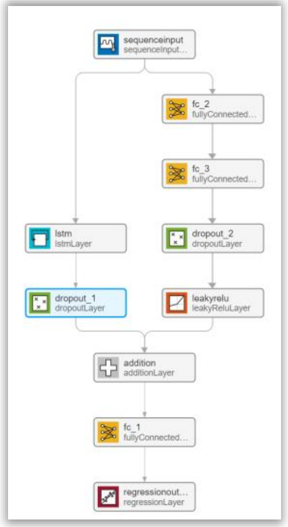
Outputs

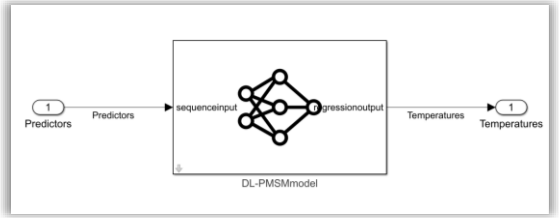
Permanent Magnet Temperature
Stator Yoke Temperature
Stator Teeth Temperature
Stator Winding Temperature

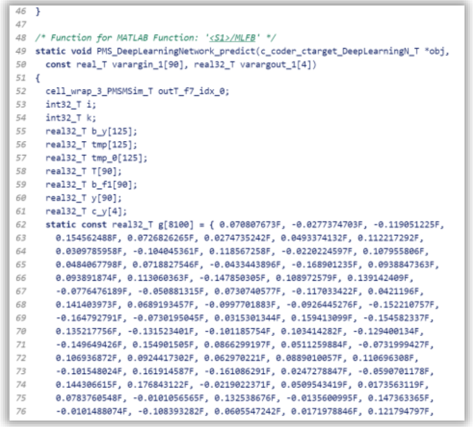
Link to blog post

Workflow



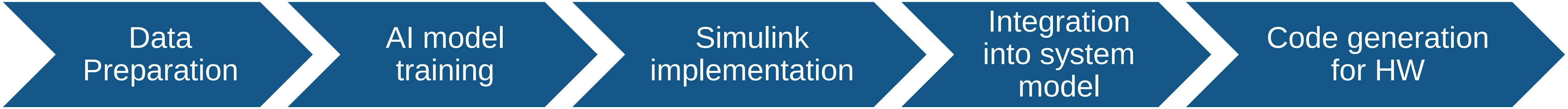






```
}

/* Function for MATLAB Function: '<S1>/MLFB' */
static void PMS_DeepLearningNetwork_predict(c_coder_ctarget_DeepLearningN_T *obj,
  const real_T varargin_1[90], real32_T varargout_1[4])
{
  cell_wrap_3_PMSMSim_T outT_f7_idx_0;
  int32_T i;
  int32_T k;
  real32_T b_y[125];
  real32_T tmp[125];
  real32_T tmp_0[125];
  real32_T T[90];
  real32_T b_f1[90];
  real32_T y[90];
  real32_T c_y[4];
  static const real32_T g[8100] = { 0.070807673F, -0.0277374703F, -0.119051225F,
    0.154562488F, 0.0726826265F, 0.0274735242F, 0.0493374132F, 0.112217292F,
    0.0309785958F, -0.104045361F, 0.118567258F, -0.0220224597F, 0.107955806F,
    0.0484067798F, 0.0718827546F, -0.0433443896F, -0.168901235F, 0.0938847363F,
    0.093891874F, 0.113060363F, -0.147850305F, 0.108972579F, 0.139142409F,
    -0.0776476189F, -0.050881315F, 0.0730740577F, -0.117033422F, 0.0421196F,
    0.141403973F, 0.0689193457F, -0.0997701883F, -0.0926445276F, -0.152210757F,
    -0.164792791F, -0.0730195045F, 0.0315301344F, 0.159413099F, -0.154582337F,
    0.135217756F, -0.131523401F, -0.101185754F, 0.103414282F, -0.129400134F,
    -0.149649426F, 0.154901505F, 0.0866299197F, 0.0511259884F, -0.0731999427F,
    0.106936872F, 0.0924417302F, 0.062970221F, 0.0889010057F, 0.110696308F,
    -0.101548024F, 0.161914587F, -0.161086291F, 0.0247278847F, -0.0590701178F,
    0.144306615F, 0.176843122F, -0.0219022371F, 0.0509543419F, 0.0173563119F,
    0.0783760548F, -0.0101056565F, 0.132538676F, -0.0135600995F, 0.147363365F,
    -0.0101488074F, -0.108393282F, 0.0605547242F, 0.0171978846F, 0.121794797F,
```



Data Preparation

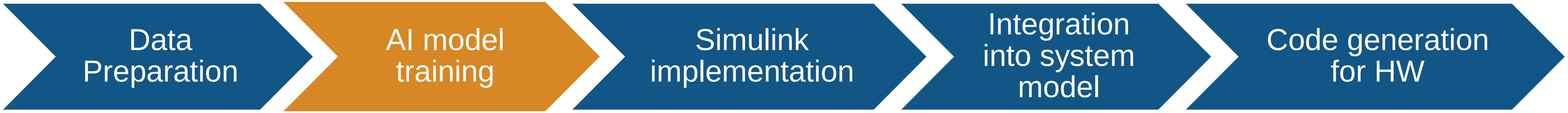
AI model training

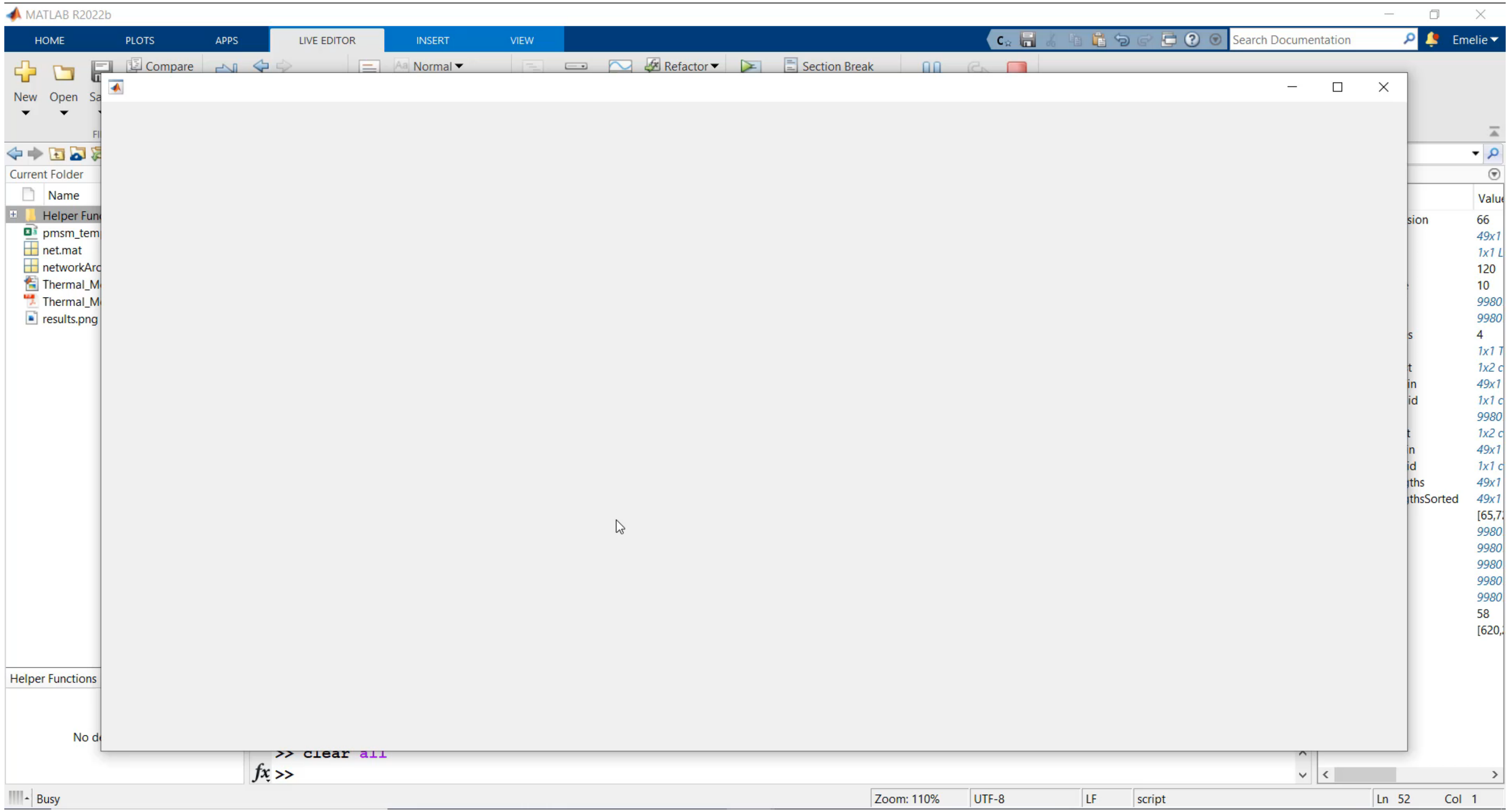
Simulink implementation

Integration into system model

Code generation for HW

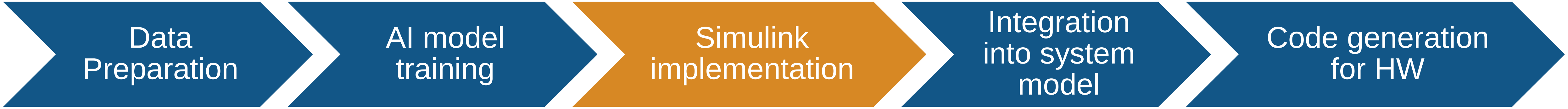
AI model training

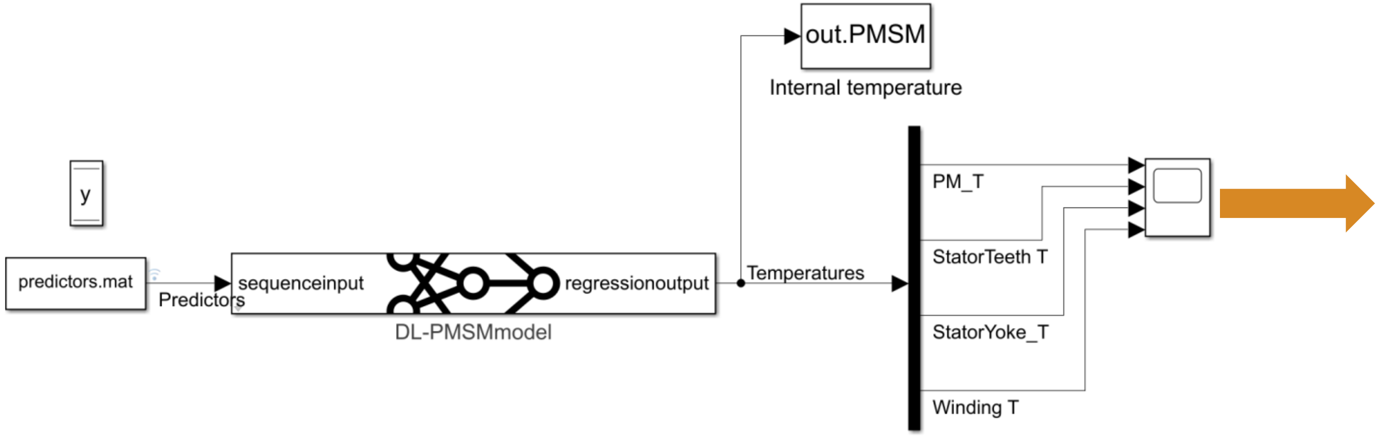


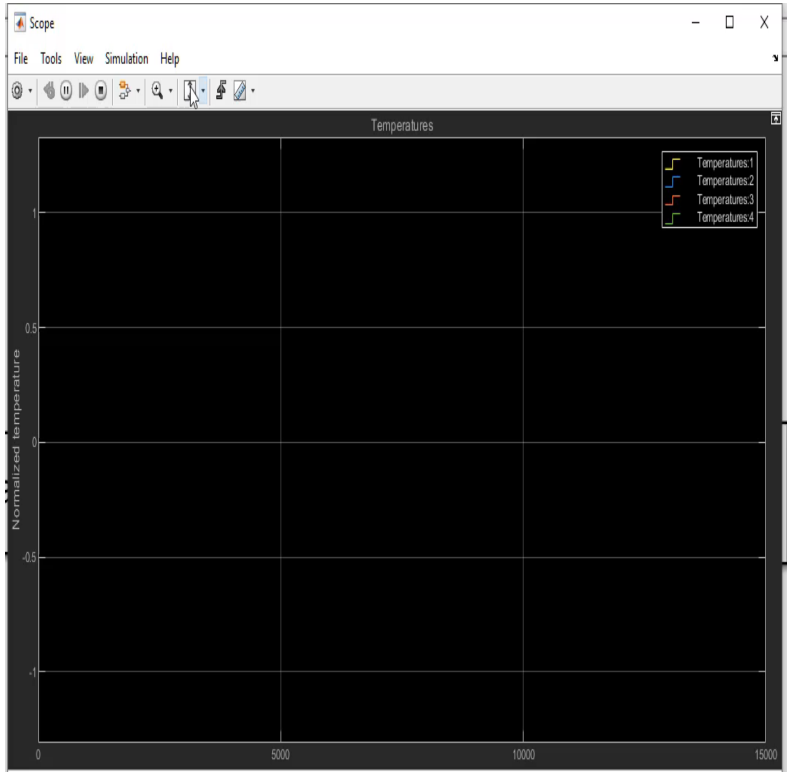


```
>> clear all
fx >>
```

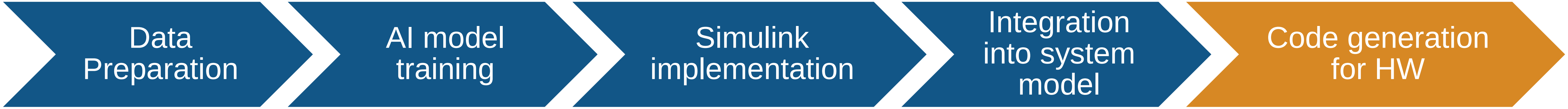
Simulink Implementation

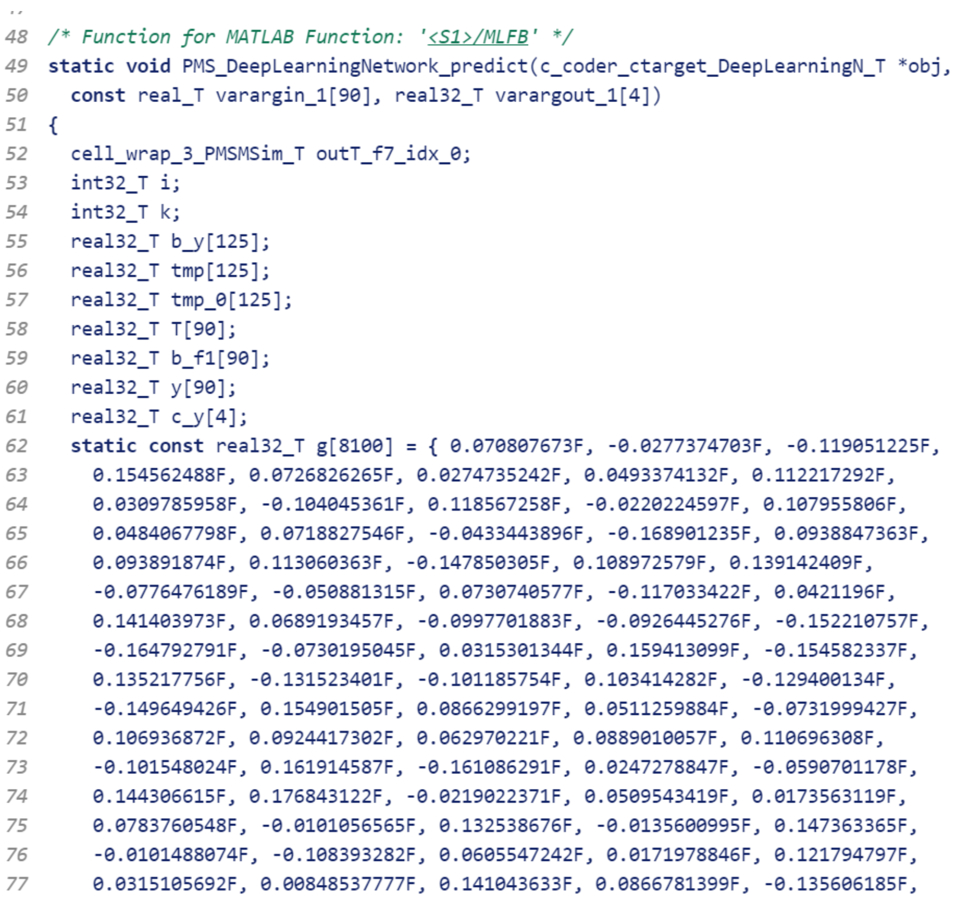






Code generation

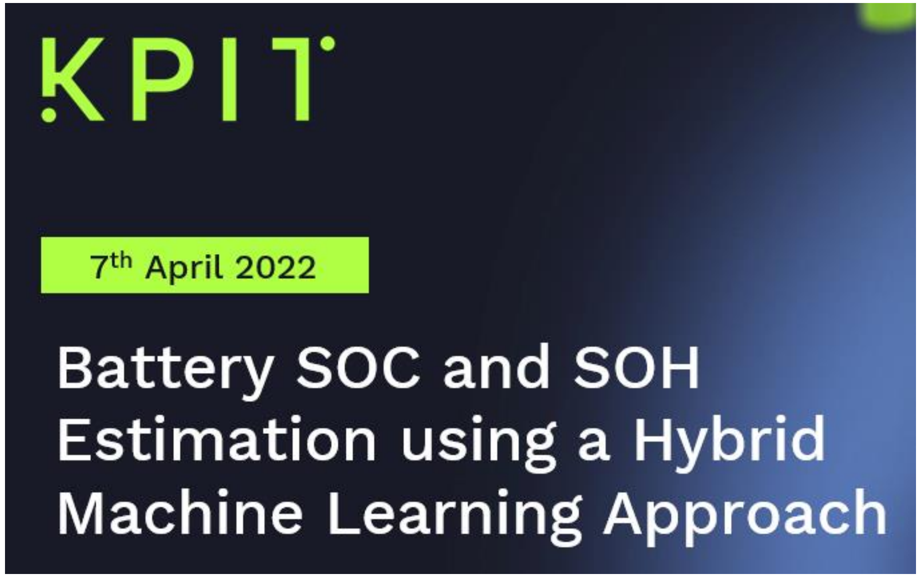




```
/* Function for MATLAB Function: '<S1>/MLFB' */
static void PMS_DeepLearningNetwork_predict(c_coder_ctarget_DeepLearningN_T *obj,
  const real_T varargin_1[90], real32_T varargout_1[4])
{
  cell_wrap_3_PMSMSim_T outT_f7_idx_0;
  int32_T i;
  int32_T k;
  real32_T b_y[125];
  real32_T tmp[125];
  real32_T tmp_0[125];
  real32_T T[90];
  real32_T b_f1[90];
  real32_T y[90];
  real32_T c_y[4];
  static const real32_T g[8100] = { 0.070807673F, -0.0277374703F, -0.119051225F,
    0.154562488F, 0.0726826265F, 0.0274735242F, 0.0493374132F, 0.112217292F,
    0.0309785958F, -0.104045361F, 0.118567258F, -0.0220224597F, 0.107955806F,
    0.0484067798F, 0.0718827546F, -0.0433443896F, -0.168901235F, 0.0938847363F,
    0.093891874F, 0.113060363F, -0.147850305F, 0.108972579F, 0.139142409F,
    -0.0776476189F, -0.050881315F, 0.0730740577F, -0.117033422F, 0.0421196F,
    0.141403973F, 0.0689193457F, -0.0997701883F, -0.0926445276F, -0.152210757F,
    -0.164792791F, -0.0730195045F, 0.0315301344F, 0.159413099F, -0.154582337F,
    0.135217756F, -0.131523401F, -0.101185754F, 0.103414282F, -0.129400134F,
    -0.149649426F, 0.154901505F, 0.0866299197F, 0.0511259884F, -0.0731999427F,
    0.106936872F, 0.0924417302F, 0.062970221F, 0.0889010057F, 0.110696308F,
    -0.101548024F, 0.161914587F, -0.161086291F, 0.0247278847F, -0.0590701178F,
    0.144306615F, 0.176843122F, -0.0219022371F, 0.0509543419F, 0.0173563119F,
    0.0783760548F, -0.0101056565F, 0.132538676F, -0.0135600995F, 0.147363365F,
    -0.0101488074F, -0.108393282F, 0.0605547242F, 0.0171978846F, 0.121794797F,
    0.0315105692F, 0.00848537777F, 0.141043633F, 0.0866781399F, -0.135606185F,
```

Other use cases of Virtual Sensors?

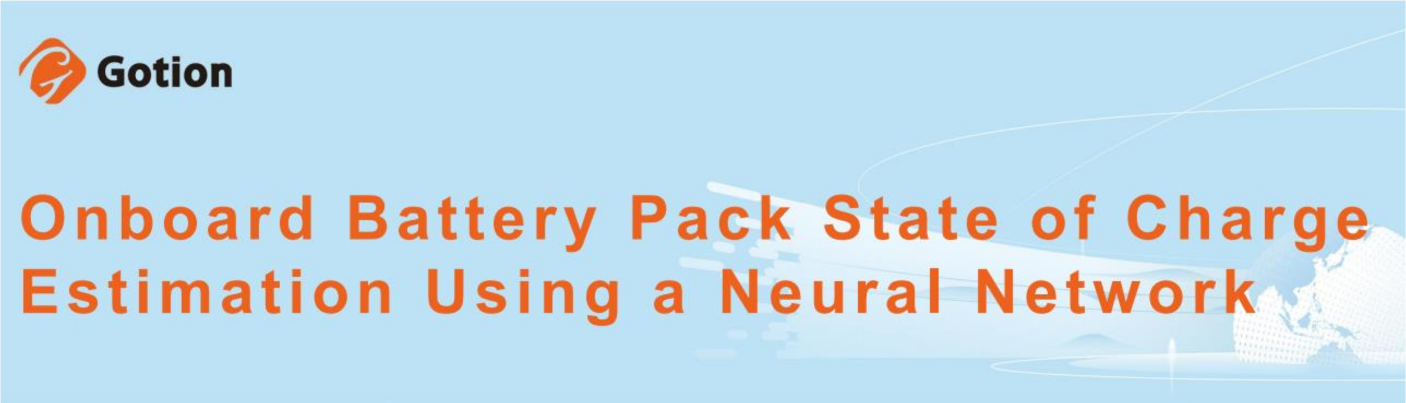
AI based Virtual Sensors for State Of Charge Estimation



KPIT

7th April 2022

Battery SOC and SOH Estimation using a Hybrid Machine Learning Approach



Gotion

Onboard Battery Pack State of Charge Estimation Using a Neural Network

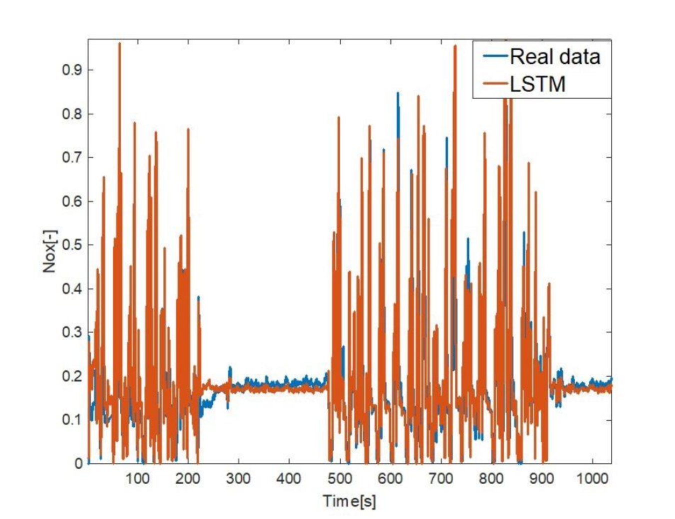
Link to video from
MathWorks Automotive Conference

Link to video from
MathWorks Automotive Conference

Other use cases of Virtual Sensors?

AI based Virtual Sensors for NOx Estimation

RENAULT



Link to article

STELLANTIS

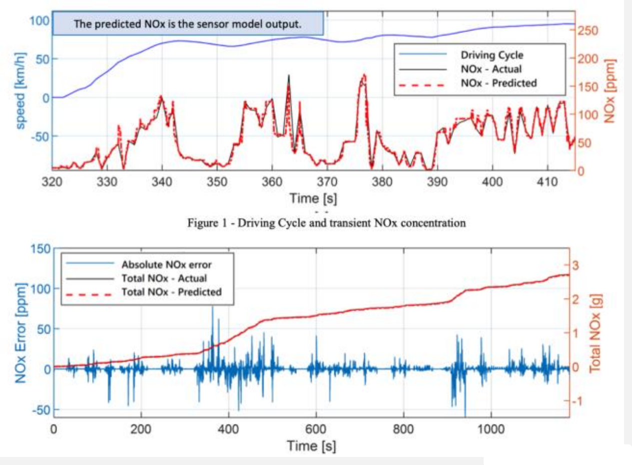
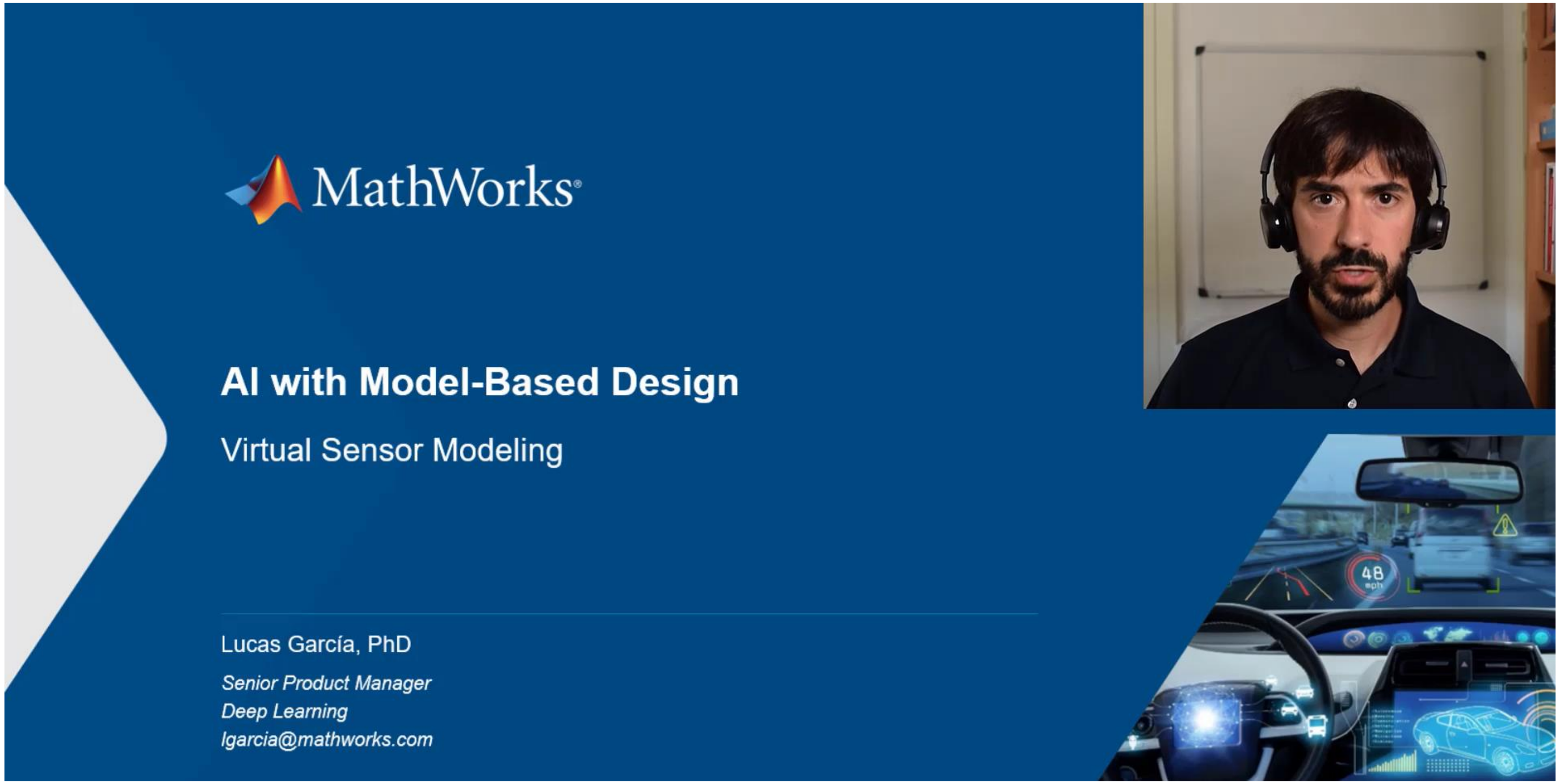


Figure 1 - Driving Cycle and transient NOx concentration

Link to presentation

Learn more??



MathWorks

AI with Model-Based Design

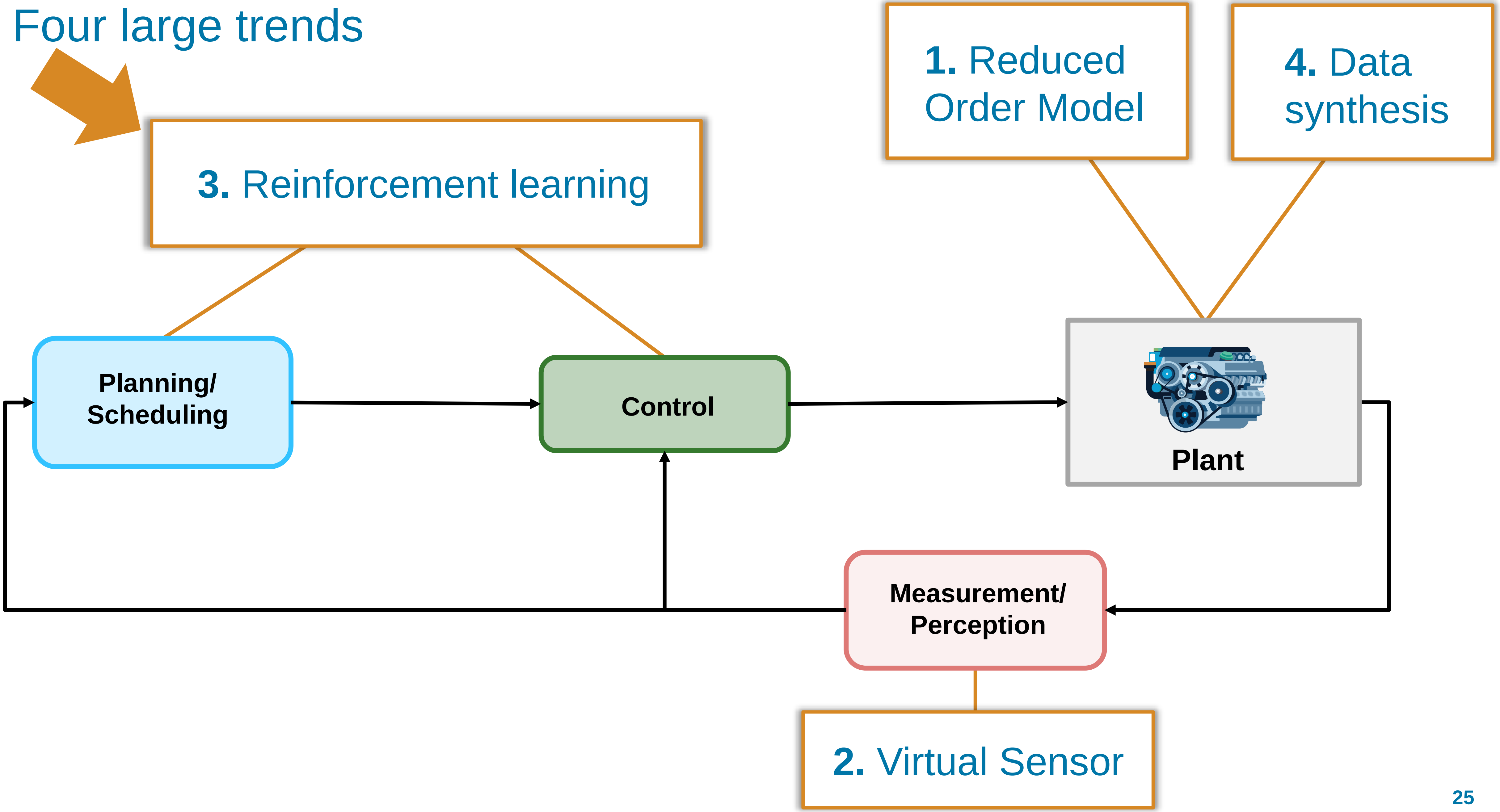
Virtual Sensor Modeling

Lucas García, PhD

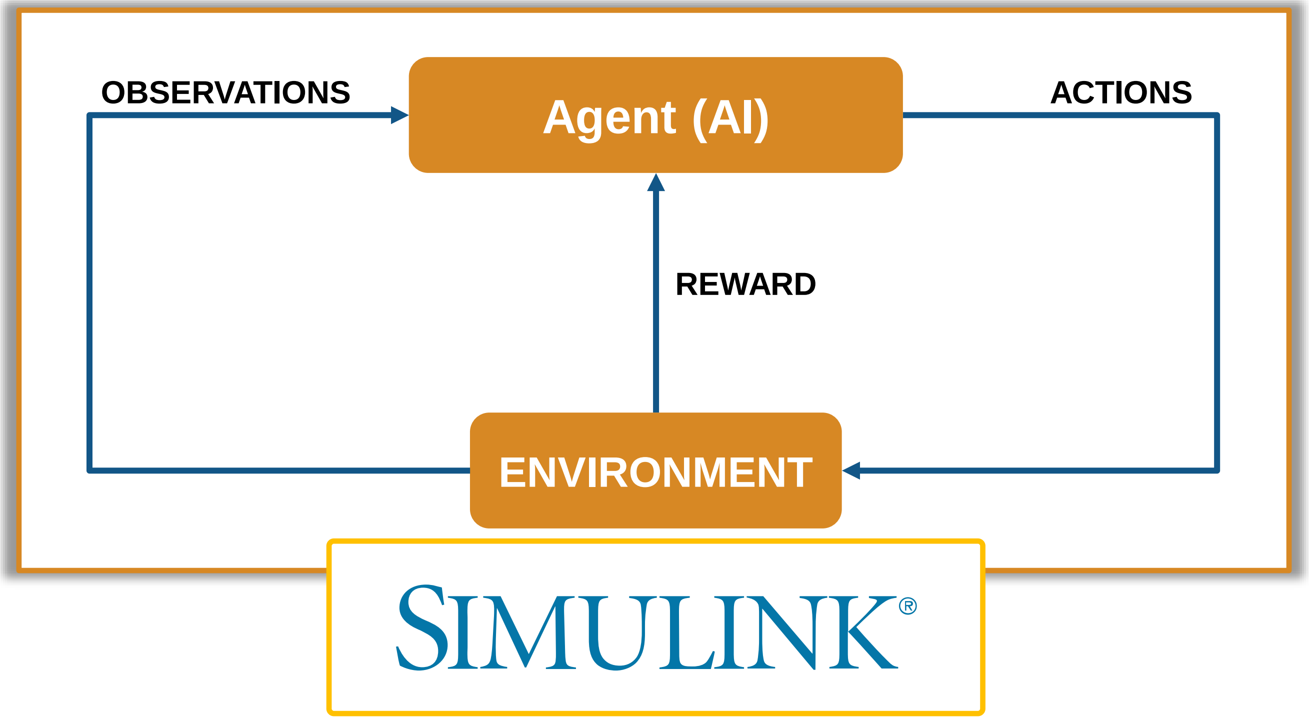
Senior Product Manager
Deep Learning
lgarcia@mathworks.com

[Video - AI with Model Based Design: Virtual Sensor Modelling]

Four large trends



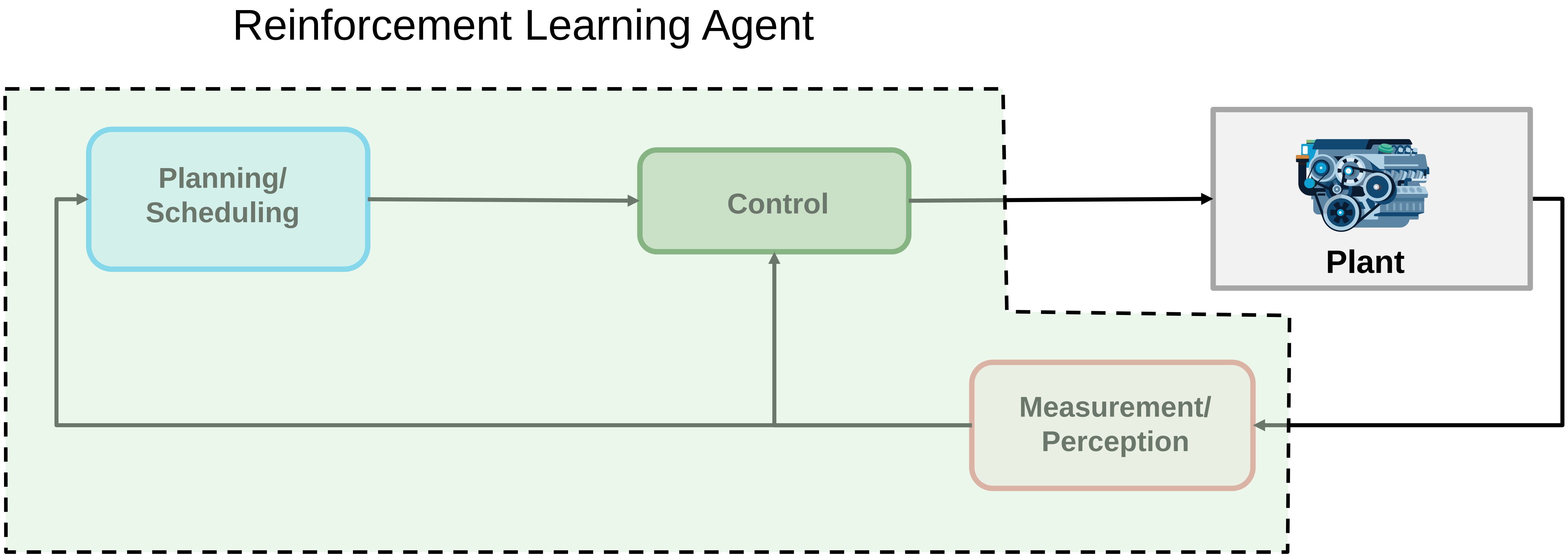
Trend 3 **Reinforcement Learning based Controls**



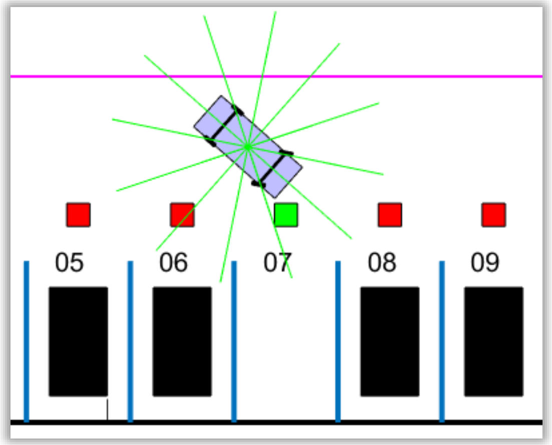
Why?!

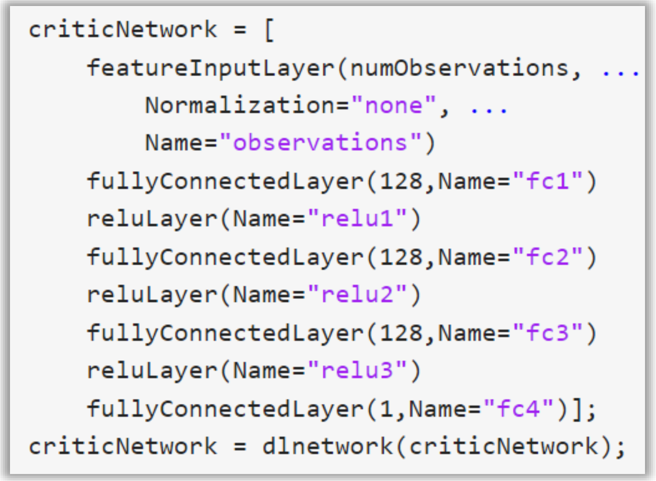
- The system you want to control or make decisions for is highly non-linear or uncertain
- Get an end-to-end solution

End-to-end Reinforcement Learning

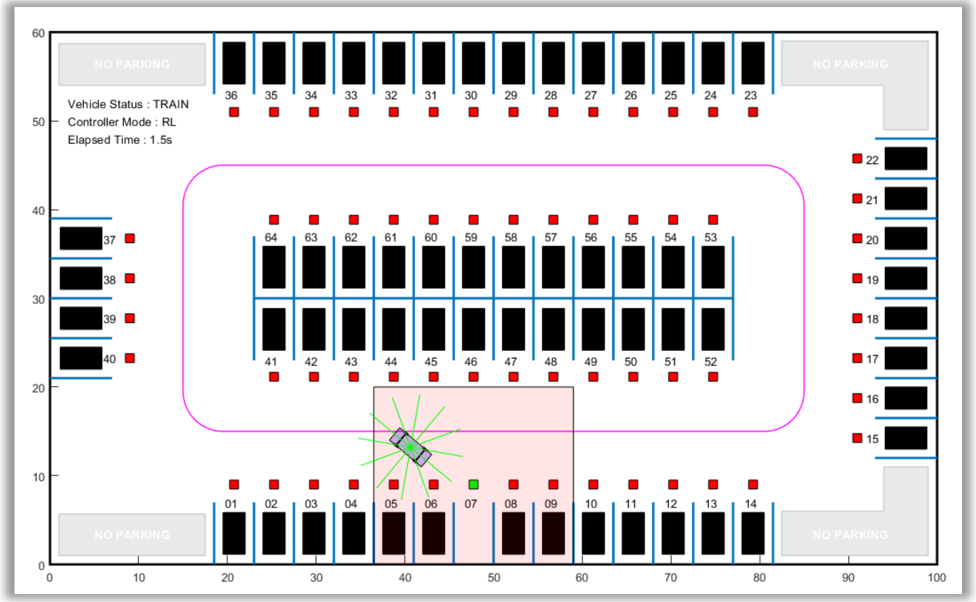


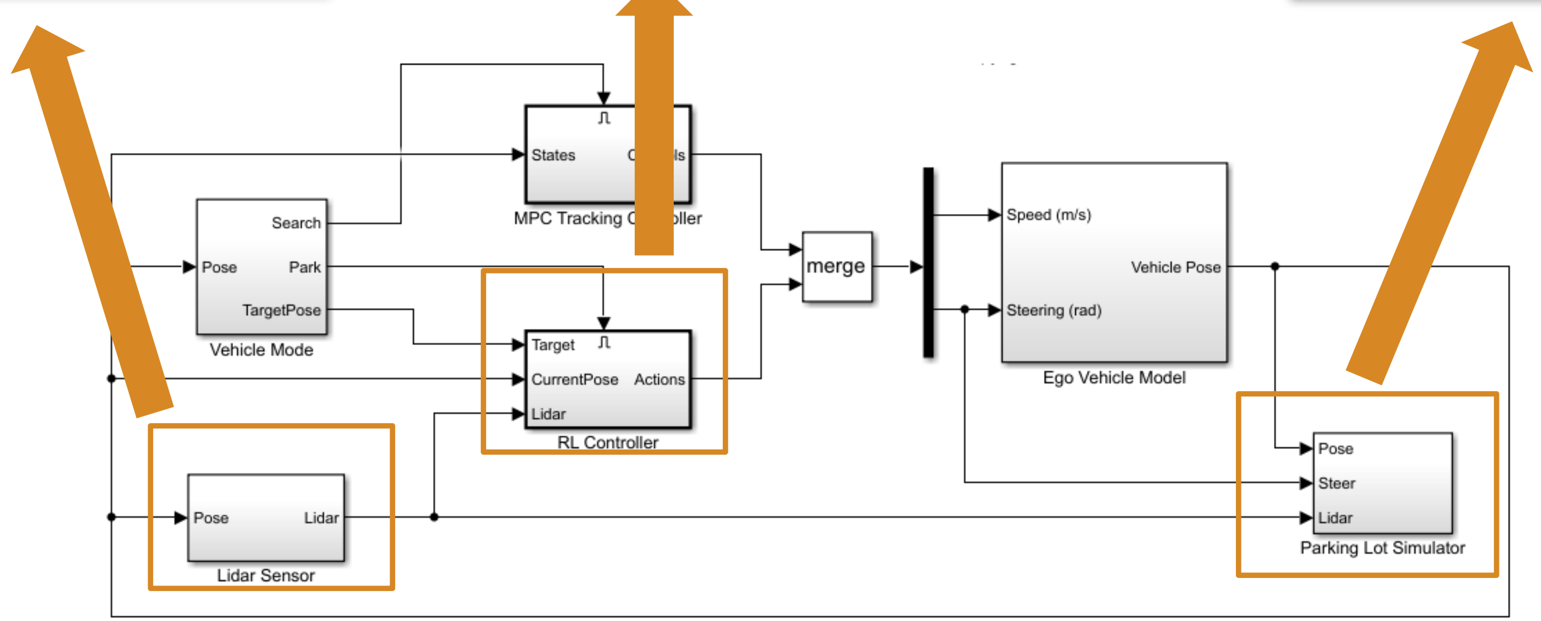
Automatic Parking Example



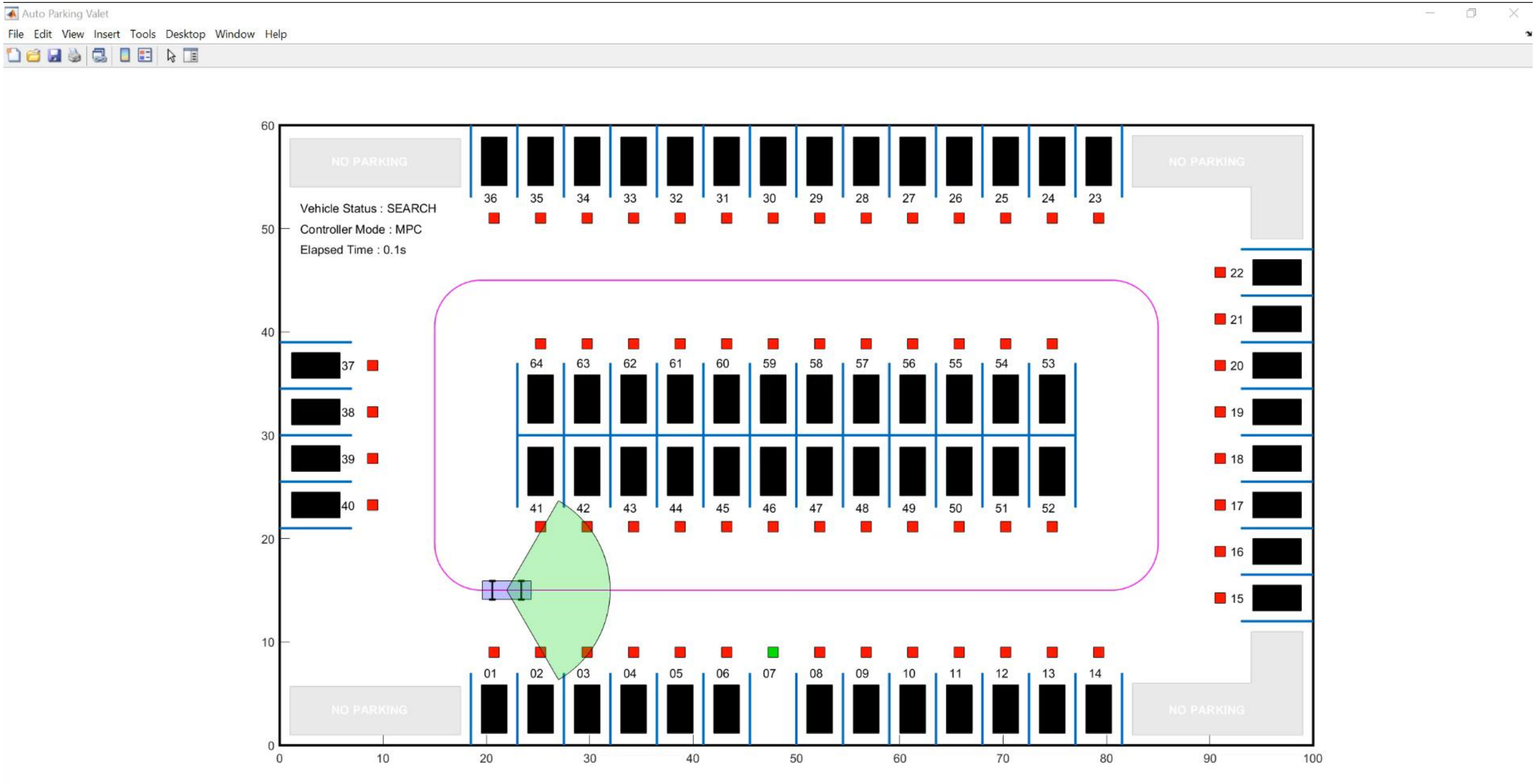


```
criticNetwork = [
    featureInputLayer(numObservations, ...
        Normalization="none", ...
        Name="observations")
    fullyConnectedLayer(128,Name="fc1")
    reluLayer(Name="relu1")
    fullyConnectedLayer(128,Name="fc2")
    reluLayer(Name="relu2")
    fullyConnectedLayer(128,Name="fc3")
    reluLayer(Name="relu3")
    fullyConnectedLayer(1,Name="fc4")];
criticNetwork = dlnetwork(criticNetwork);
```





Link to example

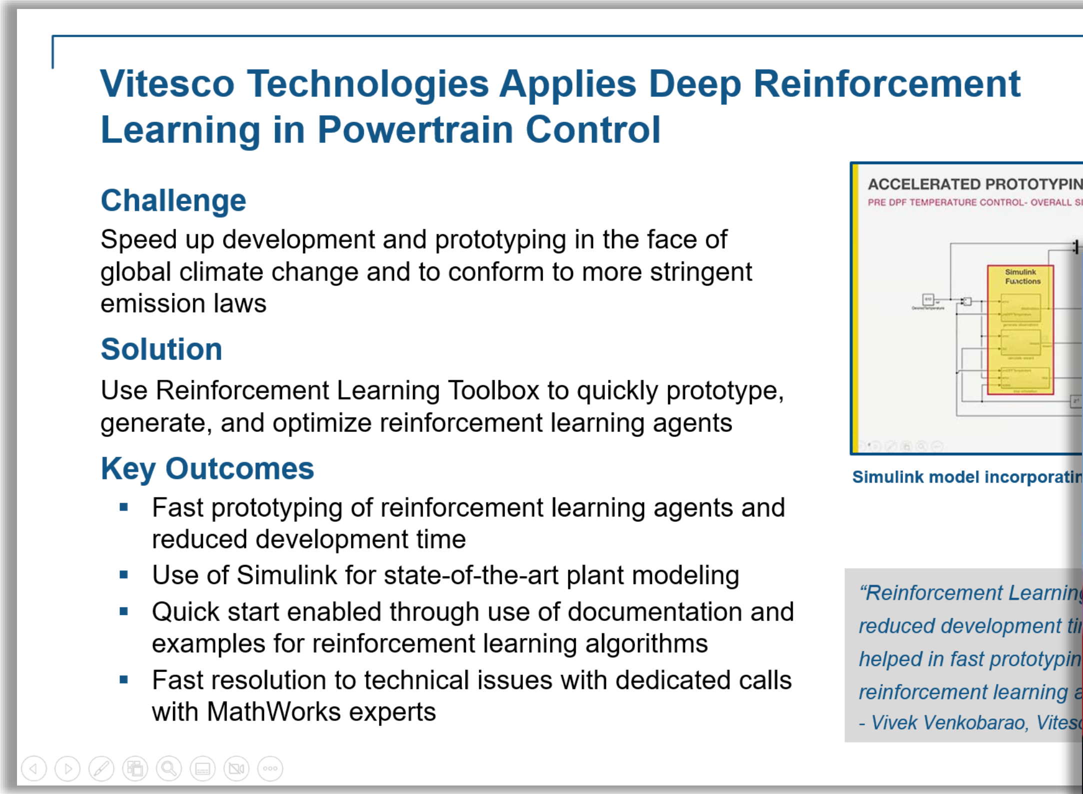


Vehicle Status : SEARCH

Controller Mode : MPC

Elapsed Time : 0.1s

RL based controls



Vitesco Technologies Applies Deep Reinforcement Learning in Powertrain Control

Challenge

Speed up development and prototyping in the face of global climate change and to conform to more stringent emission laws

Solution

Use Reinforcement Learning Toolbox to quickly prototype, generate, and optimize reinforcement learning agents

Key Outcomes

- Fast prototyping of reinforcement learning agents and reduced development time
- Use of Simulink for state-of-the-art plant modeling
- Quick start enabled through use of documentation and examples for reinforcement learning algorithms
- Fast resolution to technical issues with dedicated calls with MathWorks experts

ACCELERATED PROTOTYPING

PRE DPF TEMPERATURE CONTROL- OVERALL SIMULATION DIAGRAM

Simulink model incorporatin

> *"Reinforcement Learnin*
> *reduced development ti*
> *helped in fast prototypin*
> *reinforcement learning a*
> *- Vivek Venkobarao, Vitesc*

Link to customer presentation

A perspective on deploying Machine Learning to augment classic control design

Ali Borhan

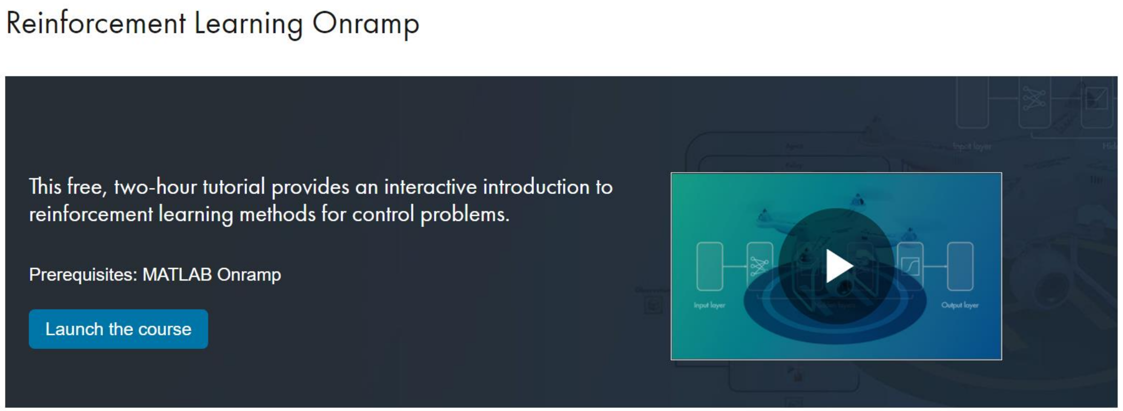
Manager – Cummins R&T

November 5, 2020

Link to customer presentation

Learn more?!

- Tech Talk video series on Reinforcement Learning concepts for engineers
- Reinforcement Learning Onramp



Reinforcement Learning Onramp

This free, two-hour tutorial provides an interactive introduction to reinforcement learning methods for control problems.

Prerequisites: MATLAB Onramp

Part 1: What Is Reinforcement Learning?
Get an overview of reinforcement learning from the perspective of an engineer. Reinforcement learning is a type of machine learning that has the potential to solve some really hard control problems.

Part 2: Understanding the Environment and Rewards
In this video, we build on our basic understanding of reinforcement learning by exploring the workflow. What is the environment? How do reward functions incentivize and agent? How are policies structured?

Part 3: Policies and Learning Algorithms
This video provides an introduction to the algorithms that reside within the agent. We'll cover why we use neural networks to represent functions and why you may have to set up two neural networks in a powerful family of methods called actor-critic.

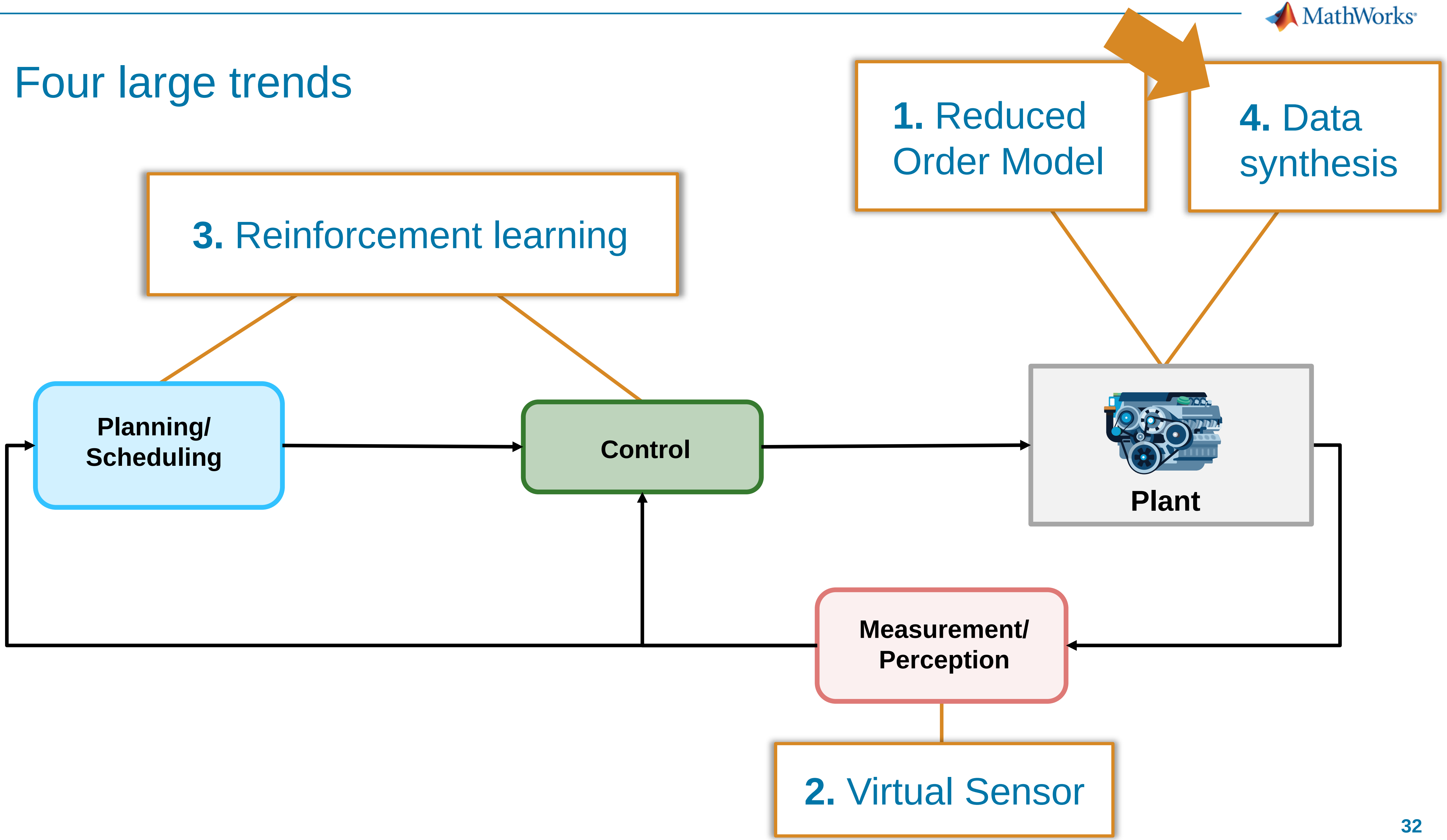
Part 4: The Walking Robot Problem
This video shows how to use the reinforcement learning workflow to get a bipedal robot to walk, and how we can set up the RL problem to look more like a traditional control problem by adding a reference signal to the design.

Part 5: Overcoming the Practical Challenges of Reinforcement Learning
There are a few challenges that occur when using reinforcement learning for production systems and there are some ways to mitigate them. This video covers the difficulties of verifying the learned solution and what you can do about it.

[Reinforcement Learning Onramp](#)

[Video series](#)

Four large trends



1. Reduced Order Model

4. Data synthesis

3. Reinforcement learning

Planning/ Scheduling

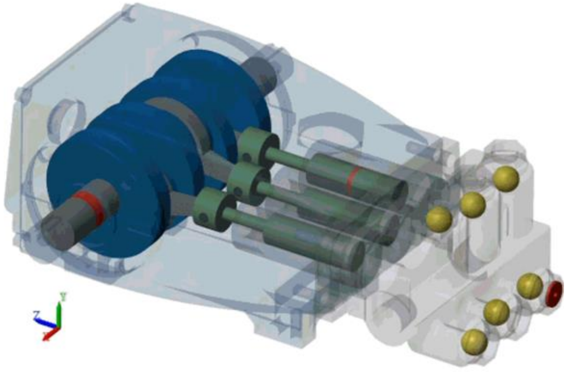
Control

Plant

Measurement/ Perception

2. Virtual Sensor

Trend 4 **Data Synthesis**



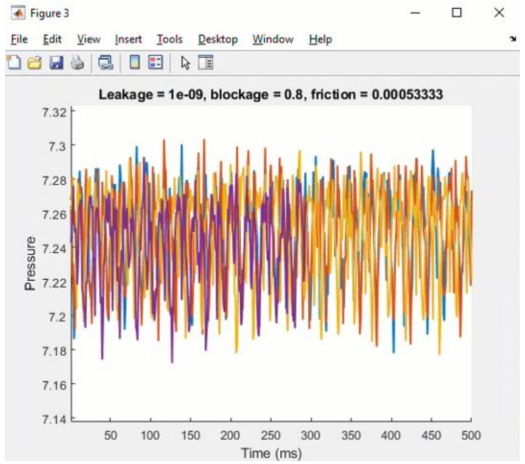
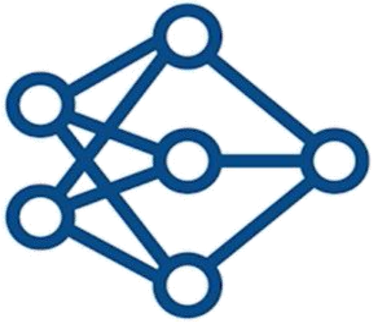


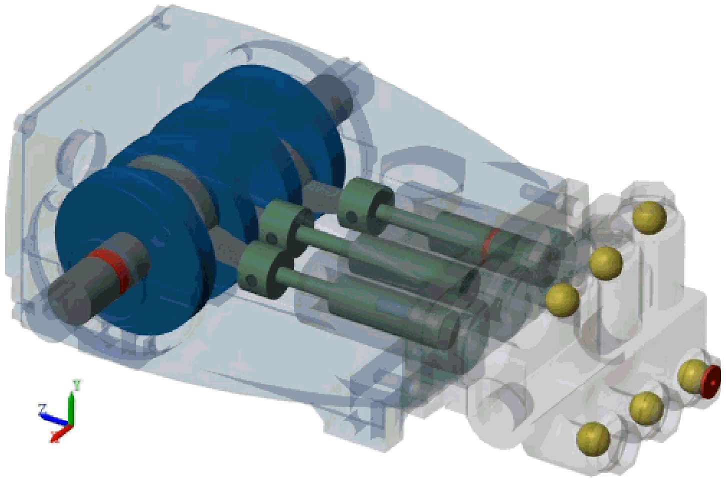
Figure 3
File Edit View Insert Tools Desktop Window Help
Leakage = 1e-09, blockage = 0.8, friction = 0.00053333
Pressure
7.32
7.3
7.28
7.26
7.24
7.22
7.2
7.18
7.16
7.14
50
100
150
200
250
300
350
400
450
500
Time (ms)



Why?!
When data is hard to get through lab test or otherwise

How?
Repurpose existing simulation models
Use Digital Twins to *generate data* to train better AI models for application deployment

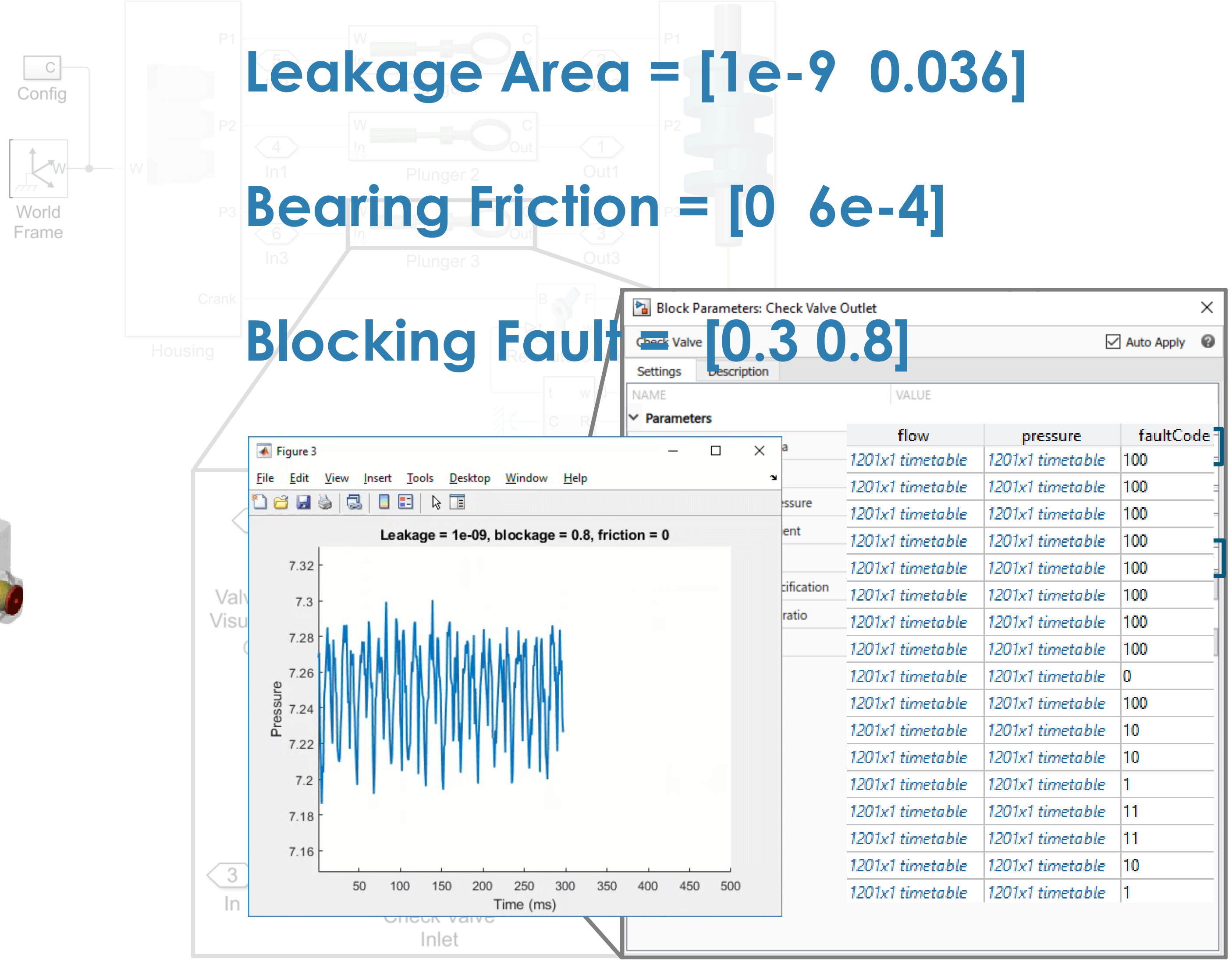
Simulate fault data with Digital Twins



Leakage Area = [1e-9 0.036]

Bearing Friction = [0 6e-4]

Blocking Fault = [0.3 0.8]



flow	pressure	faultCode
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	0
1201x1 timetable	1201x1 timetable	100
1201x1 timetable	1201x1 timetable	10
1201x1 timetable	1201x1 timetable	10
1201x1 timetable	1201x1 timetable	1
1201x1 timetable	1201x1 timetable	11
1201x1 timetable	1201x1 timetable	11
1201x1 timetable	1201x1 timetable	10
1201x1 timetable	1201x1 timetable	1

Enhance datasets for AI using Digital Twins

Challenge

Increase the performance of an automated beverage-packaging system by incorporating a dynamic tripod robot into the design

Solution

Use Simulink and Simscape Multibody to create an accurate digital twin that supports design optimization, fault testing, and predictive maintenance

Results

- Robot performance increased
- Product development time shortened
- Testing time significantly reduced

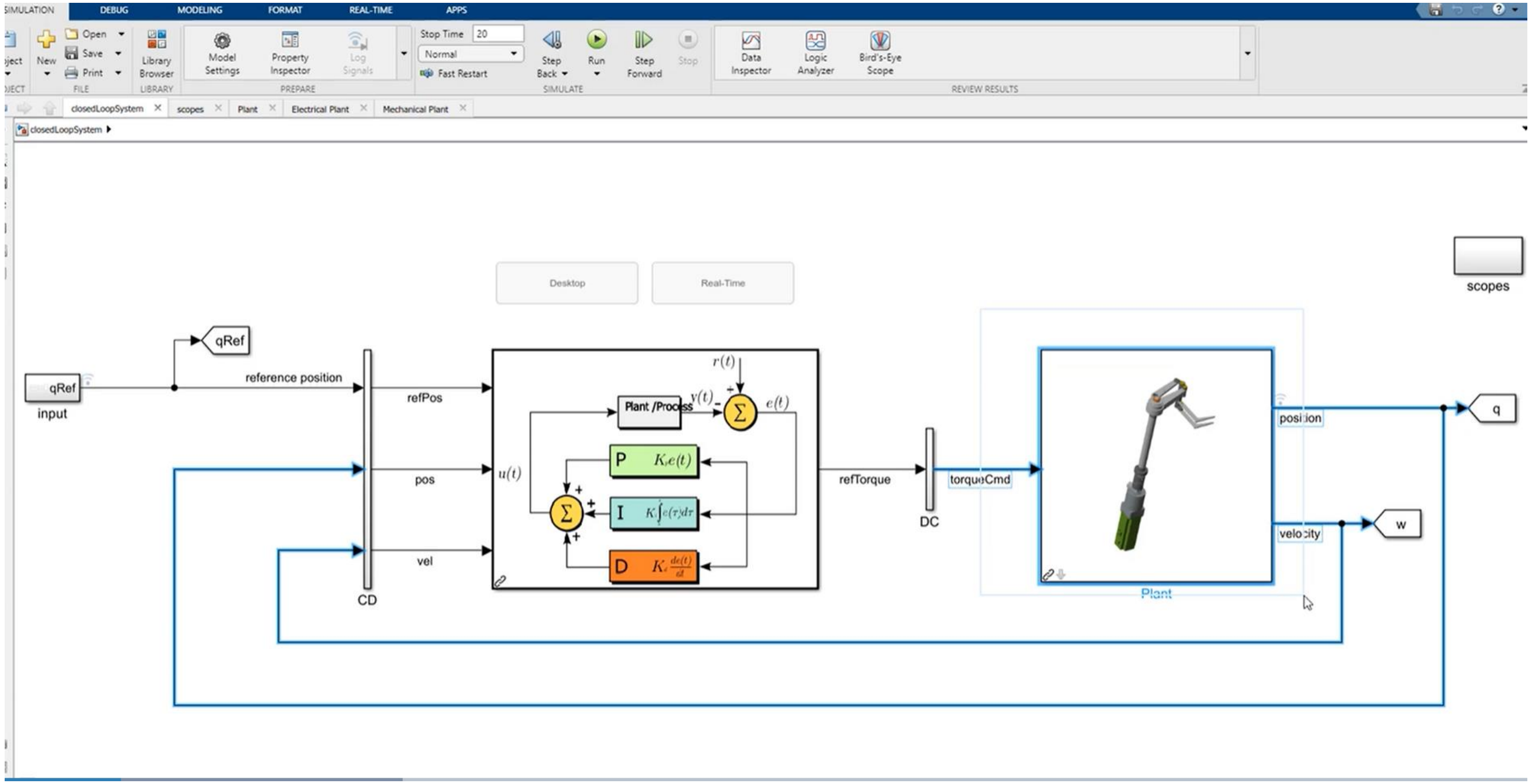
The Krones Robobox T-GM package-handling robot.

During simulations the team injected faults, such as extremely high friction, to analyze system behavior under fault conditions.

They then used the tripod robot model to train a machine learning classification algorithm for predictive maintenance.

Link to user story

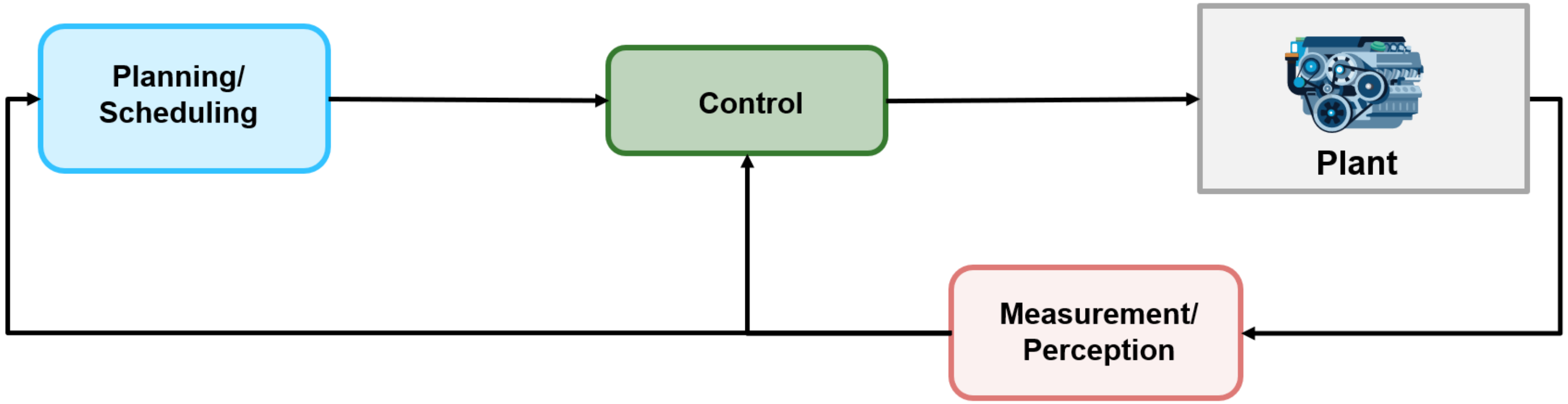
Learn more??



[Video - Design for Predictive Maintenance: Data Generation]

Conclusions?

- Many promising applications in the intersection between AI and Simulation
  - AI models can be used to enhance simulation models
  - Simulation models can be used to enhance AI models
- In MATLAB and Simulink you can use one toolchain to do both AI and Simulation with seamless interaction in-between



Thank You!

- Questions? aloytyno@mathworks.com