

# MATLAB EXPO 2018

What's New in MATLAB  
and Simulink **R2017b** **R2018a**

Cynthia Cudicini  
Daniel Martins

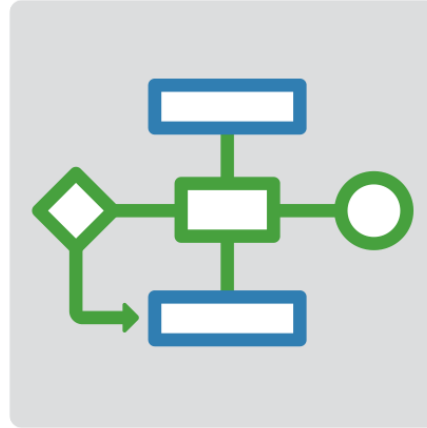


## Platform Productivity



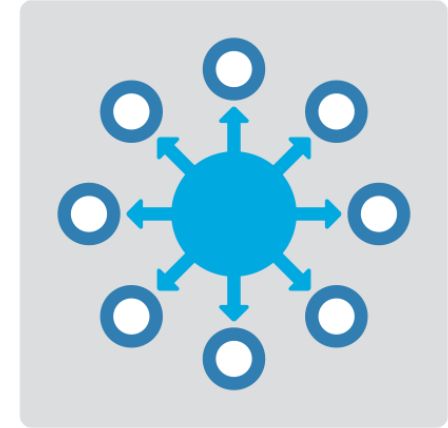
**Getting your work  
done faster**

## Workflow Depth



**Support for your  
entire workflow**

## Application Breadth

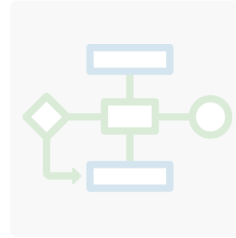


**Products for the  
work you do**

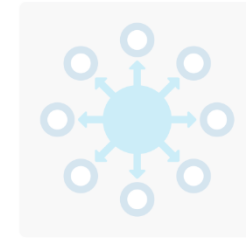
## Platform Productivity



## Workflow Depth



## Application Breadth



- **Create Your Designs Faster**
- **Simplify Analysis**
- **Simulate Faster and Scale Your Work**
- **Collaborate**

# Create Your Designs Faster – Live Editor

Live Editor - C:\Demos\ExploreEvents.mlx

**Explore and Analyze Storm Events**

**Frequency of Events**

Explore the frequency of various storm events and locations and the associated damage costs.

```
clear
load prepEvents
data = timetable2table(data);
head(data)
```

**Visualize with a Heatmap**

This is helpful in exploring patterns across categories like the events and locations.

```
bigFigure;
heatmap(data, 'state', 'weathercats');
xlabel('State')
ylabel('Storm Event')
title('Frequency of Events by Location')
```

Live Editor - C:\Users\gha\Downloads\ExploringExoplanets.mlx

**Exploring Exoplanets**

In this example we will explore some data on exoplanets - planets outside our own solar system. The data used here is a subset of data from the NASA Exoplanet Archive. We will start by using the data to answer some questions about the set of exoplanets in the archive. Then we will do some calculations to try to identify planets in the archive that might be capable of supporting life.

```
exoplanets = readtable('exoplanets.xlsx', 'Sheet', 'Database');
exoplanets(1:8,:)
```

HOME PLOTS APPS LIVE EDITOR

New Open Save Find Files Compare Go To Text

Save Ctrl+S Save As... Export to PDF... Export to HTML... Export to LaTeX...

Current Folder: gha \ Downloads \ ExploringExoplanets.mlx

the Exoplanet Archive

Now that we have an equilibrium temperature for the

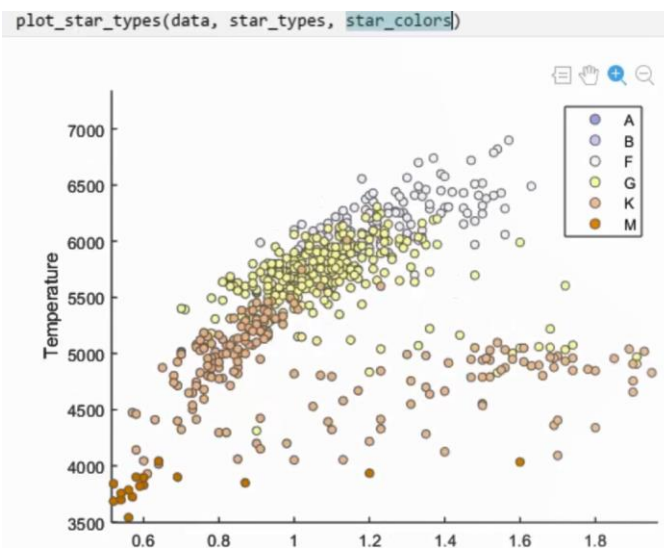
minTemp = 273;

We can see the characteristics of the planets found by the top four methods above.

```
methods = {'Transit' 'Radial Velocity' 'Microlensing' 'Imaging'};
colors = {'black' 'blue' 'red' 'green'};
data = {};
for i = 1:numel(methods)
    data{i} = exoplanets(strcmp(exoplanets(:, 'pl_discovery_method'), methods{i}));
end
plot_discoveries(data, methods, colors)
```

It Step into plot\_discoveries reflected light are large planets far from their host stars and the transit

Run current line and step into plot\_discoveries



Live Editor - C:\Users\gha\Downloads\ExploringExoplanets.mlx

```
star_colors = [0.64 0.73 1.00; 0.80 0.85 1.00; 0.97 0.97 1.00; 1.00 1.00 1.00];
T = exoplanets(~cellfun(@isempty, exoplanets.st_spectral_type), :);
data = {};
```

```
1:numel(star_types)
for i = 1:numel(star_types)
    T(i) = T(startsWith(T.st_spectral_type, star_types{i}), :);
end
plot_star_types(data, star_types, star_colors)
```

Code ^

xlim([0.51 1.97]) ylim([3496 7344])

Update Code Copy

star\_colors

star\_types

startsWith

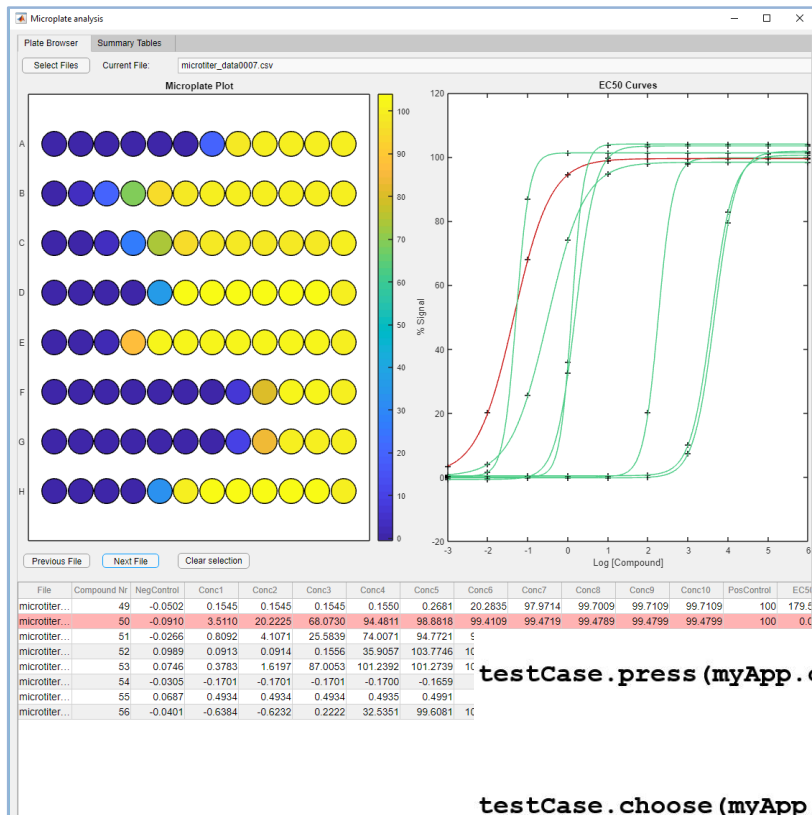
start\_simulink

starepository

startDeploytoolPackage

startFileSystemMonitor

# Create Your Designs Faster – App Designer

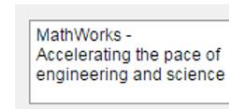
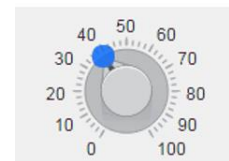
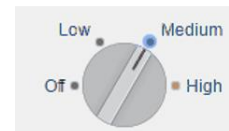
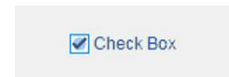


```
testCase.press(myApp.checkbox)
```

```
testCase.choose(myApp.discreteKnob, "Medium")
```

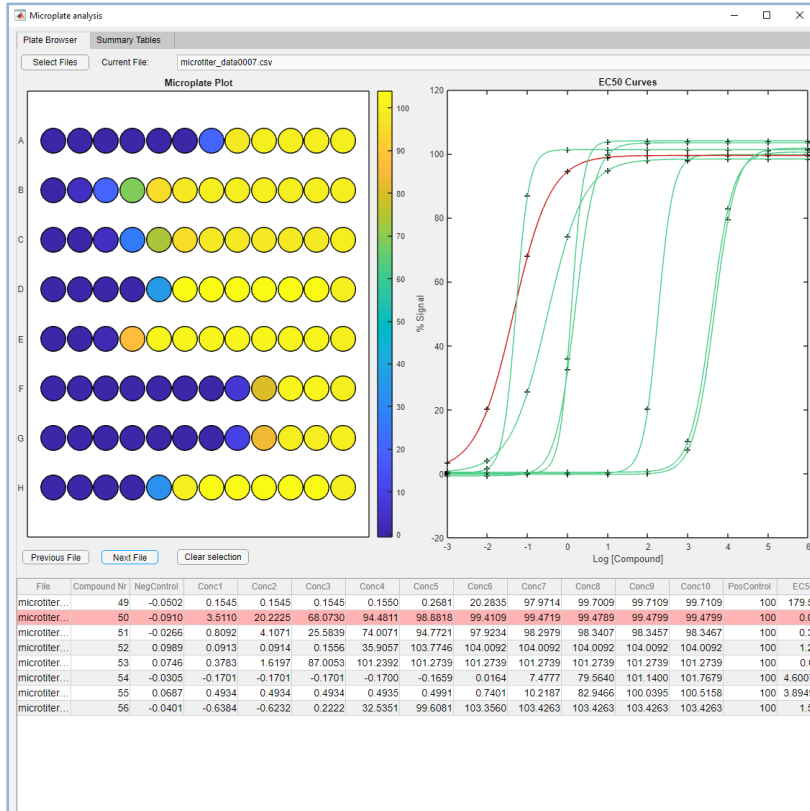
```
testCase.drag(myApp.continuousKnob, 10, 90)
```

```
testCase.type(myApp.editfield, myTextVar)
```

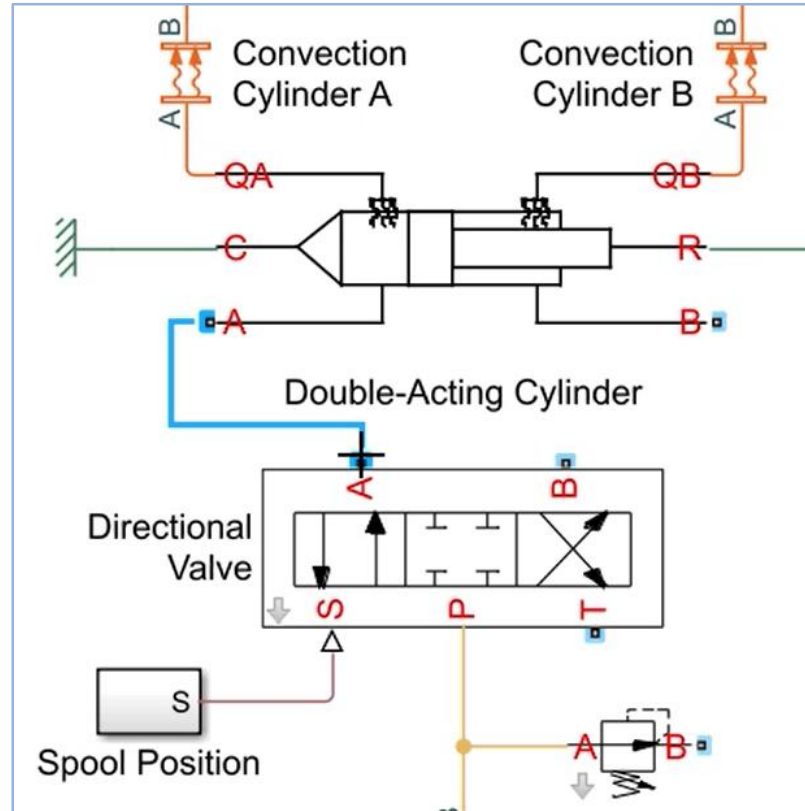


The screenshot shows the 'File Exchange' section of the App Designer interface. It features the MathWorks logo at the top, a search bar, and a list of available migration guides. The main text reads: 'GUIDE to App Designer Migration Tool for MATLAB version 1.0 (15.1 KB) by MathWorks App Designer Team'. Below this, it says: 'Use the GUIDE to App Designer Migration tool to help transition your GUIDE apps to App Designer.'

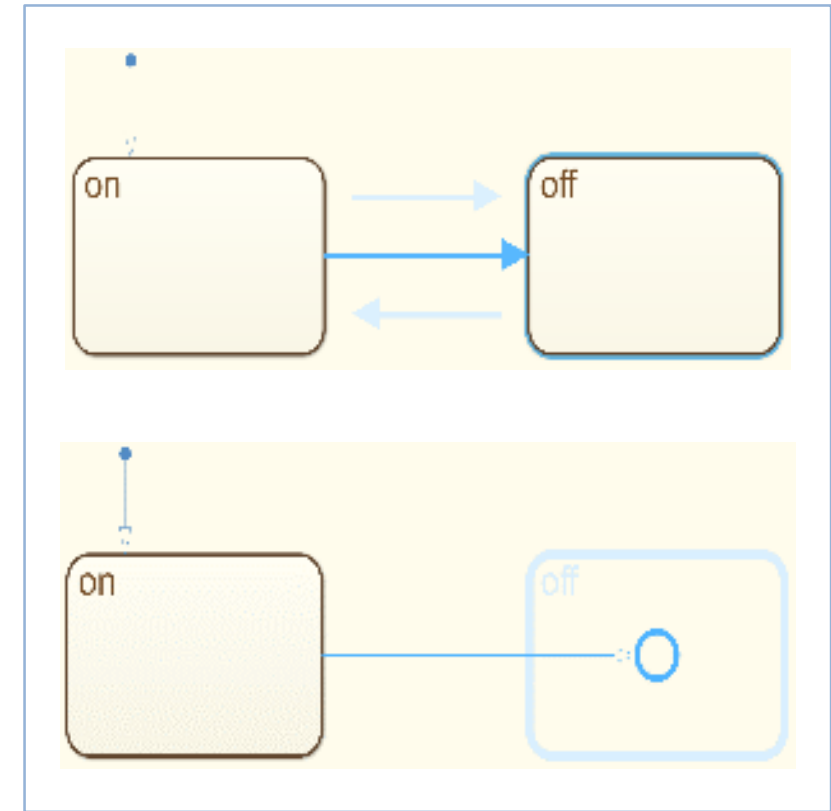
# Create Your Designs Faster – Simulink and Stateflow



**MATLAB**



**Simulink**

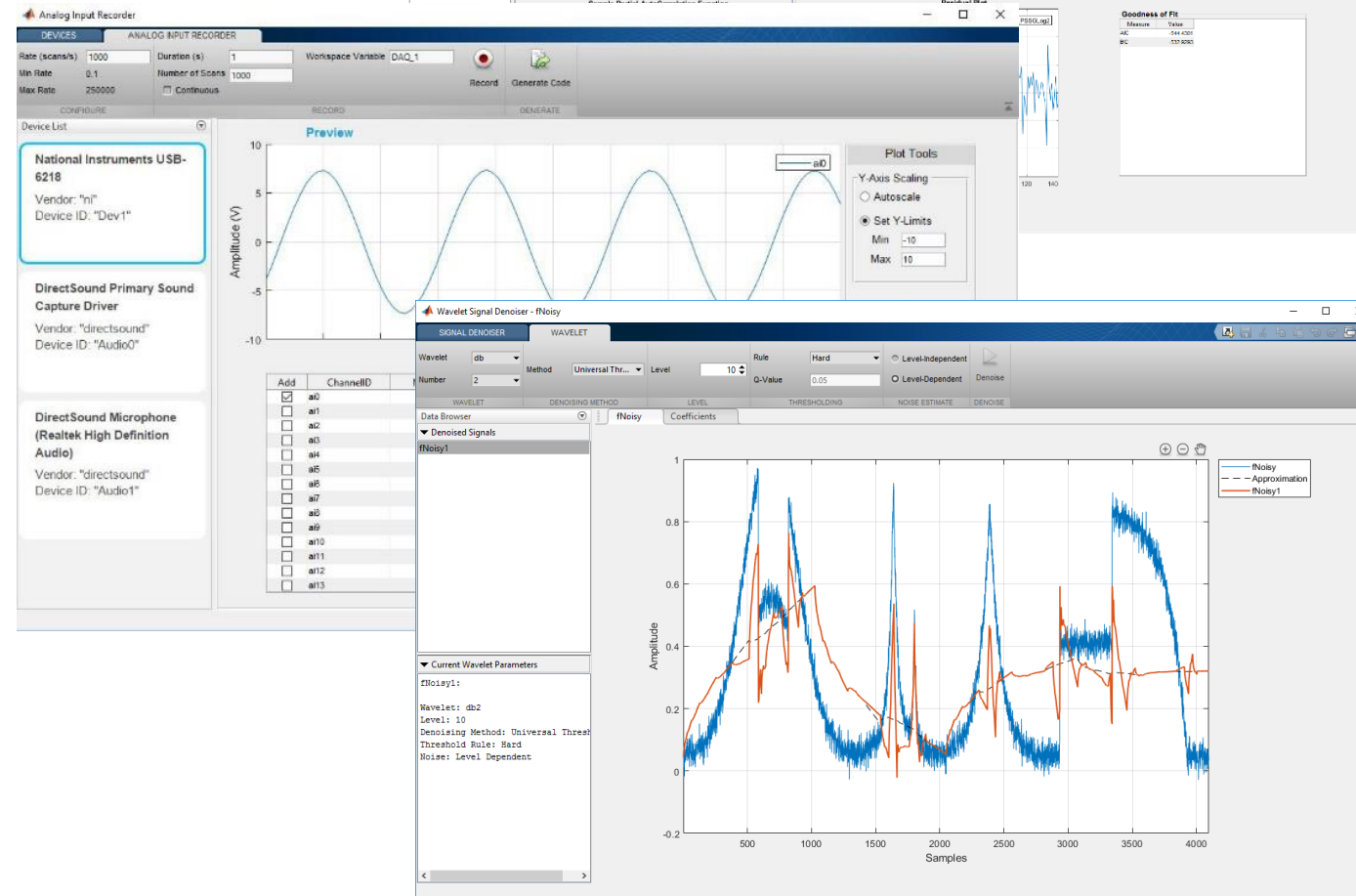


**Stateflow**

# Simplify Analysis with Apps

These interactive applications automate common technical computing tasks

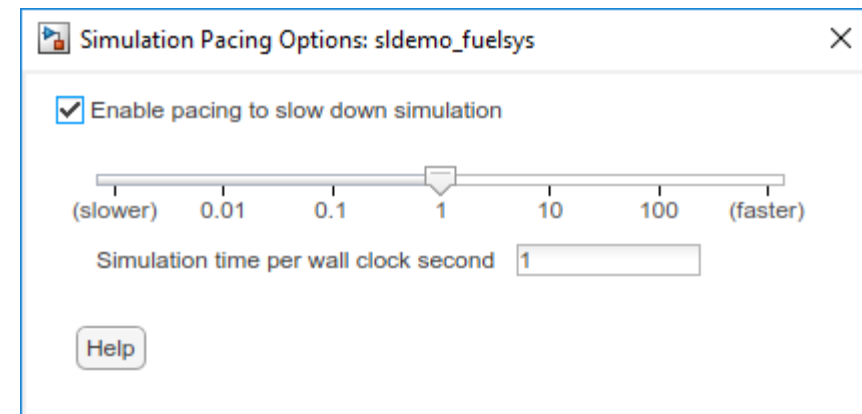
- **Econometric Modeler app**
  - Perform time series analysis, specification testing, modeling, and diagnostics
  
- **Analog Input Recorder app**
  - Acquire and visualize analog input signals
  
- **Wavelet Signal Denoiser app**
  - Visualize and denoise time series data



# Simplify Analysis by Simulating at Wall Clock Speed

## Slow down the simulation for easier model interactivity

- Especially for models controlled and monitored via Dashboard blocks and other displays
- Useful when model is connected to hardware

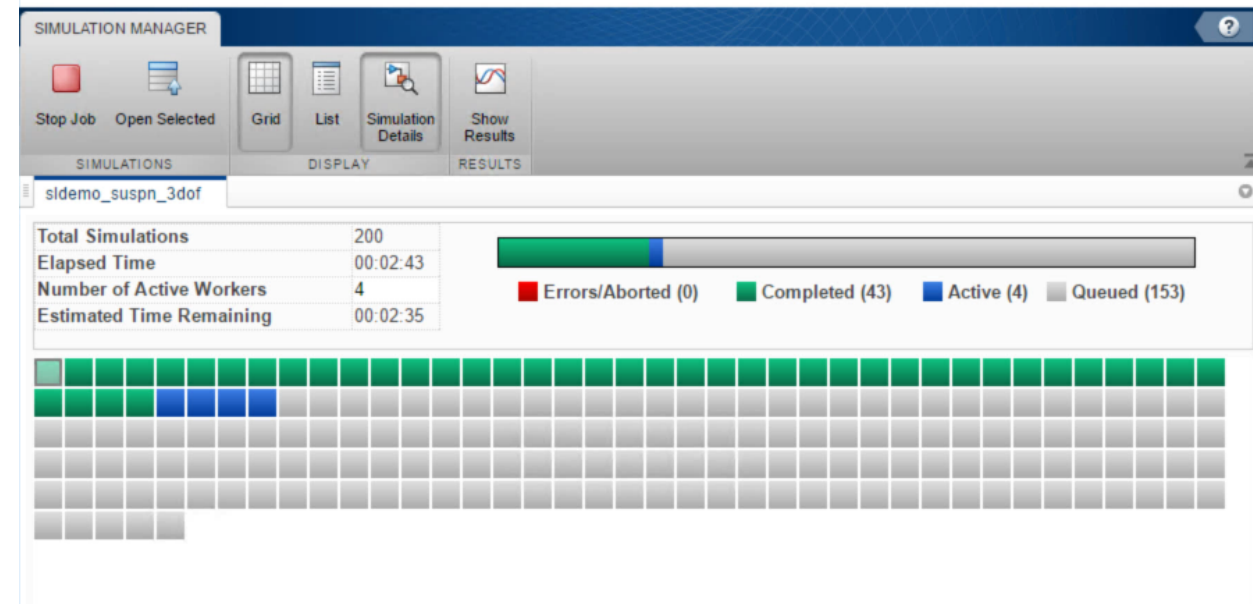
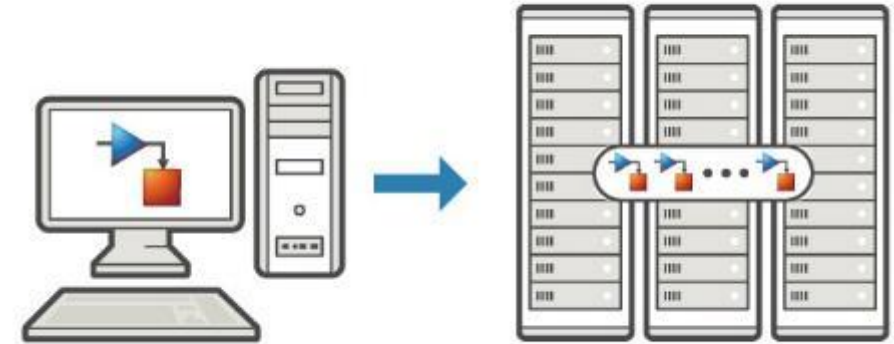




# Scale Your Work

Use parallel computing to run multiple simulations faster

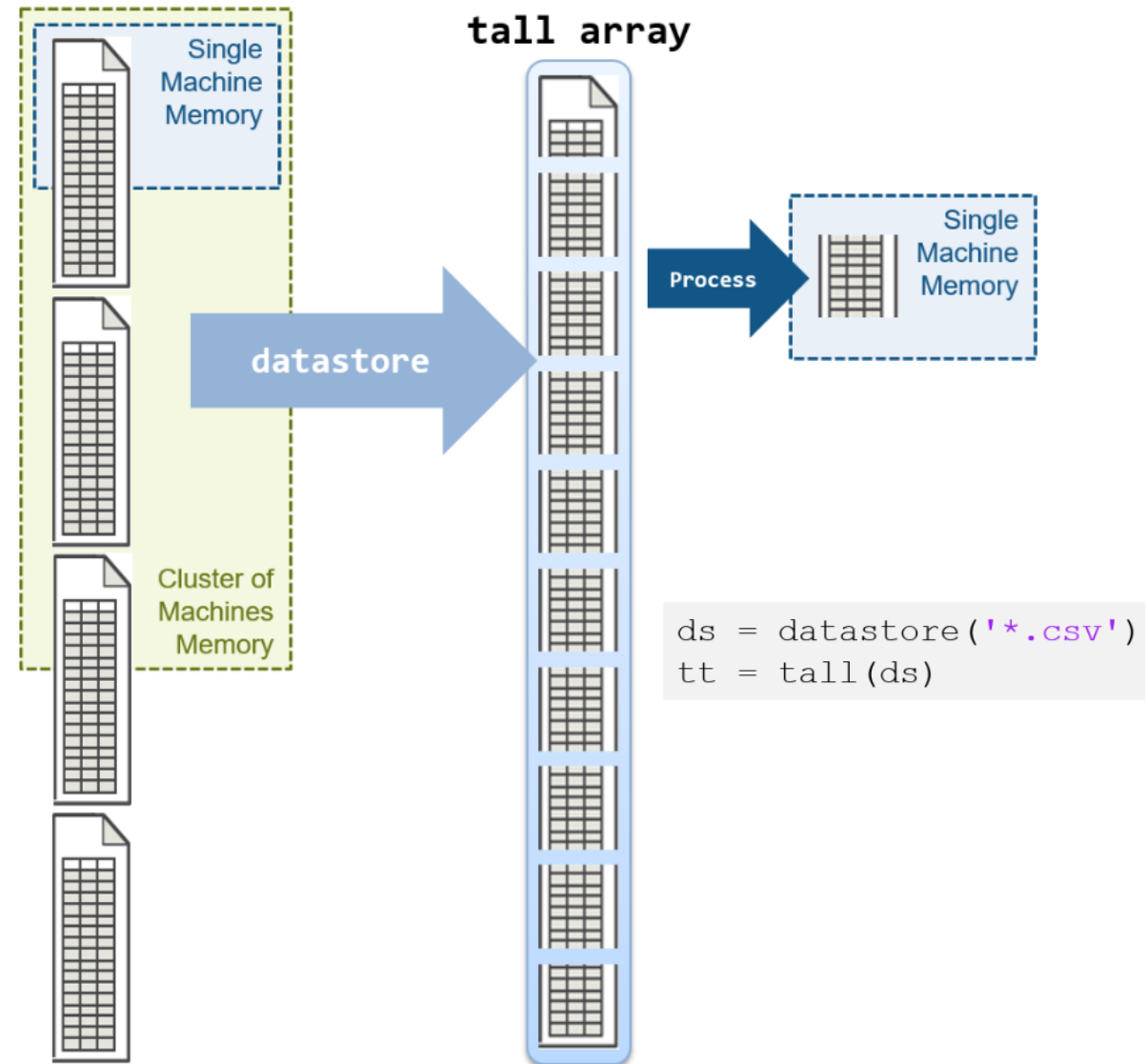
- Run multiple parallel simulations with `parsim`
- Monitor simulation status and progress in the Simulation Manager



# Scale Your Work

Use tall arrays to manipulate and analyze data that is too big to fit in memory

- Use familiar MATLAB functions and syntax
- Support for hundreds of functions
- Works with Spark + Hadoop clusters

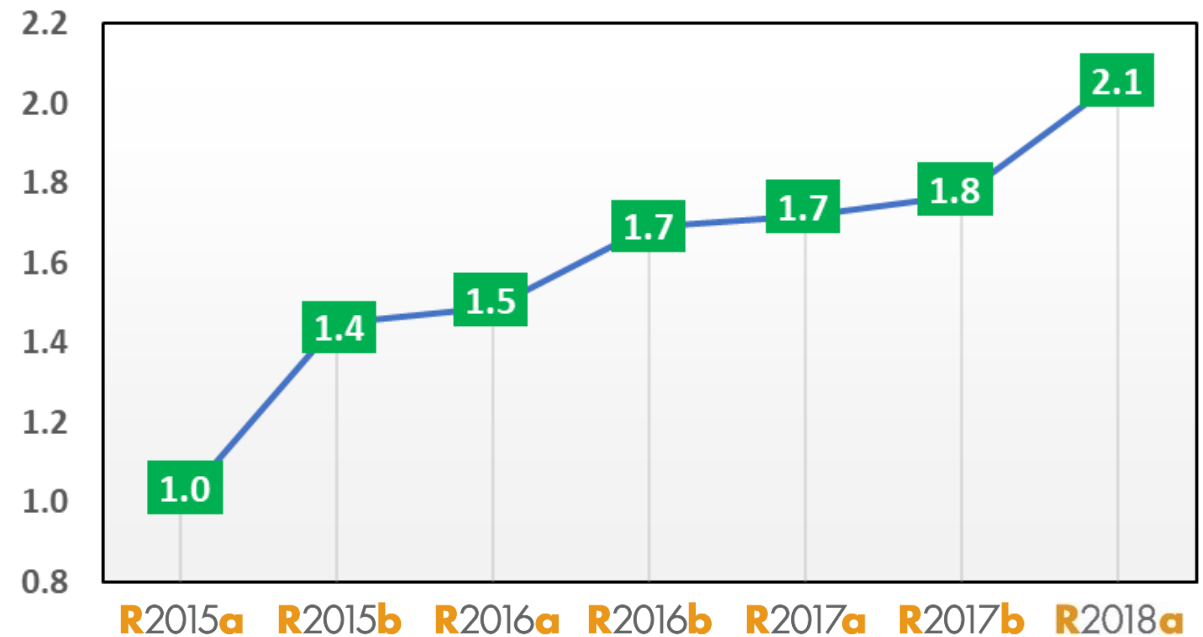


# Simulate Faster

## Redesigned execution engine runs MATLAB code faster

- All MATLAB code can now be JIT compiled
- MATLAB runs your code over twice as fast as it did just three years ago
- No need to change a single line of your code
- Increased speed of MATLAB startup in R2018a

Average Speedup in Customer Workflows



# Team Collaboration

Use advanced software development features to manage, test, and integrate MATLAB code

Current Folder

Name	SVN
Profile Summary	
Created: 09-Dec-2014 14:25:00	
Test Class Definition	
<pre>classdef MyComponentTest &lt; matlab.unittest.TestCase</pre>	

## MATLAB® Test Report

Timestamp: 04-Jan-2017 13:28:06  
 Host: AH-SDE  
 Platform: win64  
 MATLAB Version: 9.1.0.441655 (R2016b)

Number of Tests: 17  
 Testing Time: 0.4516 seconds

Overall Result: PASSED

17 passed

### Overview

Test Name	Duration
BlipTests.BlipSizeLengthTests	0.1403 seconds
BlipTests.BlipSubsasnTests	0.1542 seconds
BlipTests.BlipSubsreffTests	0.1572 seconds

### Details

**BlipTests.BlipSizeLengthTests**

- scalarBlipSize: The test passed. Duration: 0.0863 seconds
- vectorBlipSize: The test passed. Duration: 0.0027 seconds
- scalarBlipLength: The test passed. Duration: 0.0044 seconds

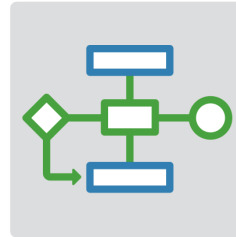
Identify differences between model elements, Stateflow charts, and MATLAB Function blocks

TYPE	UNRESOLVED	RESOLVED
Conflict	1	0
Conflicted manual merge	0	0
Manual merge	0	0
Automatic	0	4
<b>Total</b>	<b>1</b>	<b>4</b>

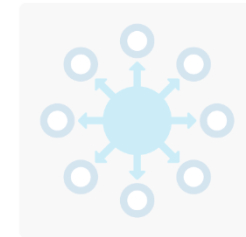
## Platform Productivity



## Workflow Depth



## Application Breadth



- **Deployment of MATLAB Algorithms and Applications**
- **Code Generation from Simulink Models**
- **Verification and Validation**

# Deploy MATLAB Algorithms and Applications

## Access Data



Sensors



Files



Databases

## Analyze Data



Data exploration



Preprocessing



Domain-specific algorithms

## Develop



AI model



Algorithm development



Modeling & simulation

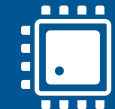
## Deploy



Desktop apps

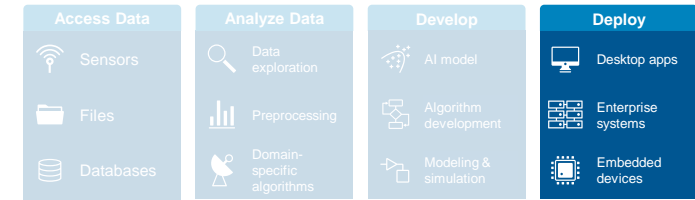


Enterprise systems



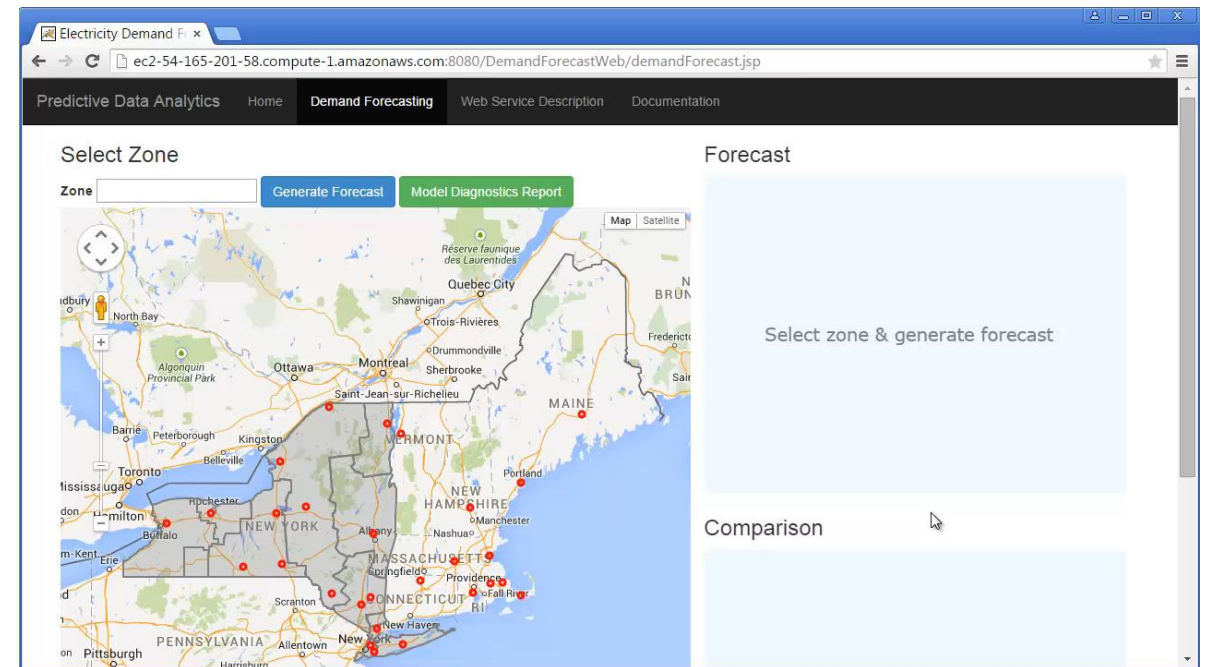
Embedded devices

# Deploy MATLAB Algorithms and Applications

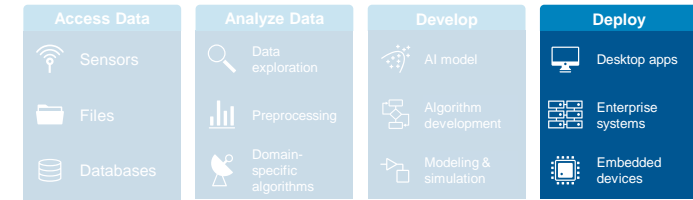


Share your work outside of MATLAB without having to recode your algorithms

- Standalone desktop applications
- Add-ins for Microsoft Excel
- Software components to integrate with other languages (*C/C++*, *.NET*, *Python*, *Java*)
- Software components for web and enterprise applications

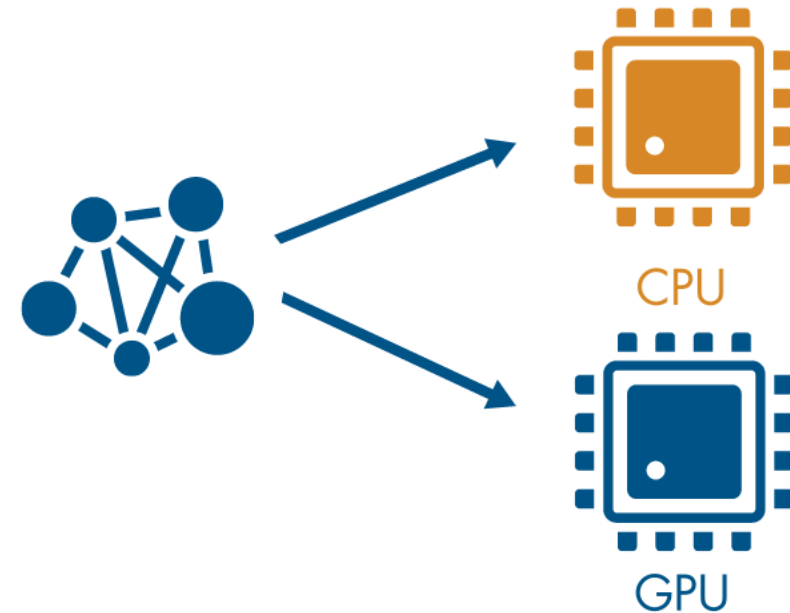


# Deploy MATLAB Algorithms



## Deploy machine learning and deep learning models using automatically generated code

- Generate C code for predictive machine learning and deep learning models
- Generate optimized CUDA code for deep learning, embedded vision, and autonomous systems

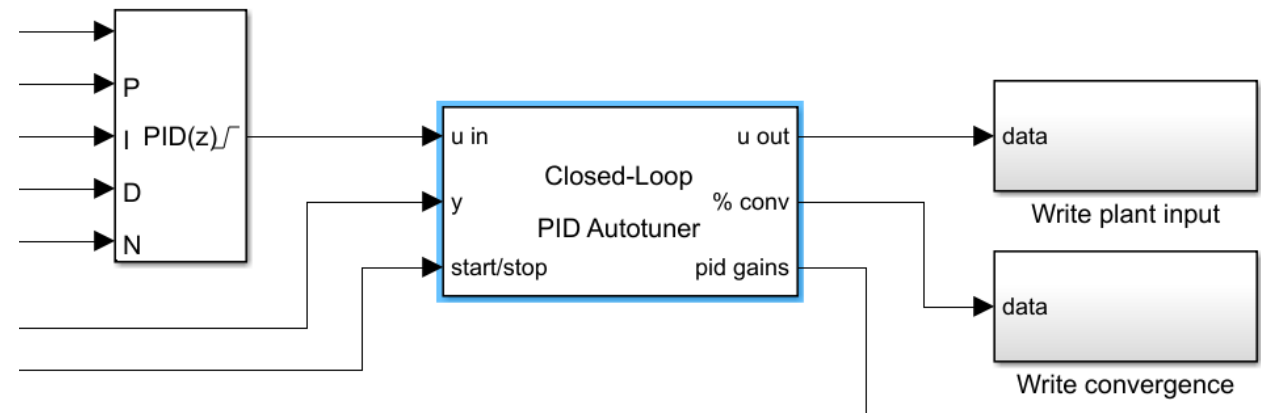




# PID Control Tuning

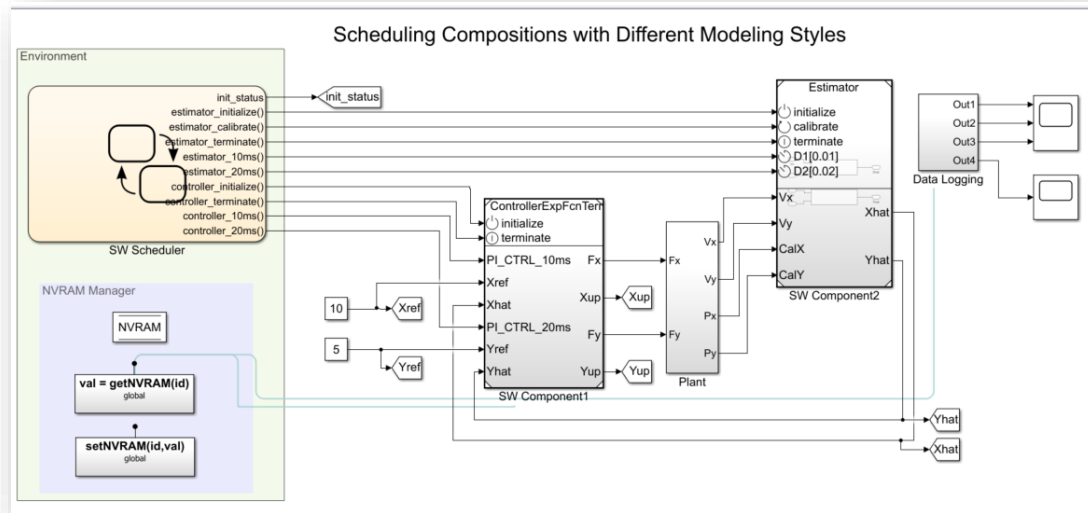
## Implement an embedded PID auto-tuning algorithm

- Automatically tune PID controller gains in real time against a physical plant
- No model of plant dynamics required
- Deploy the auto-tuning algorithm to embedded software using automatic code generation



# Prepare Your Model for Code Generation

Prepare model components for code generation

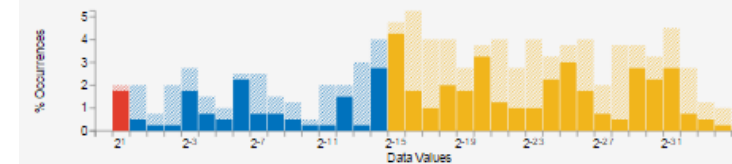


Prepare model data for code generation

The screenshot shows the FIXED-POINT TOOL interface. The 'Simulation Ranges' dropdown is selected, and the 'Propose Data Types' button is highlighted with a blue box. A green arrow points from the 'Propose Data Types' button to the 'Compare Results' button. The 'MODEL HIERARCHY' pane shows a tree structure with 'Simulink Root' and 'Data Objects'. The 'Results' table is visible below.

Name	Run	CompiledDT	SpecifiedDT	ProposedDT	Accept Sim
SRP Subsystem...	Ranges(Double)	double	fixdt(1,13,11)	locked	0
SRP Subsystem...	Ranges(Double)	double	fixdt(1,13,11)	locked	0
SRP Subsystem...	Ranges(Double)	double	fixdt(1,17,15)	fixdt(0,17,22)	0

Visualization of Simulation Data



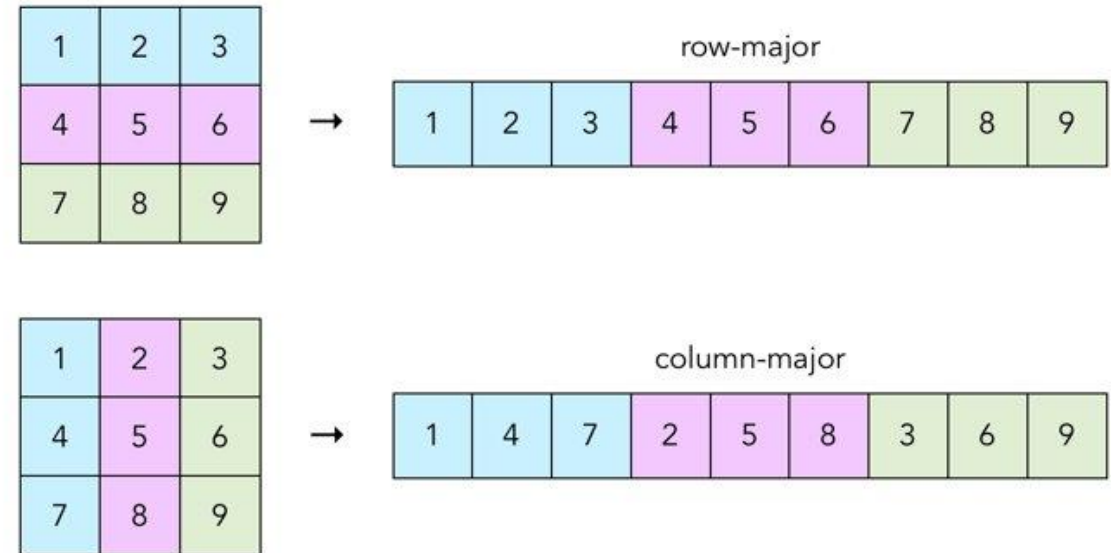
	Potential Overflows	In-Range	Potential Underflows
Positive Values	7	53	139
Negative Values	1	64	135

Number of times zero occurred: 0

# Generate Code from Simulink Models

## Access and define all the information in your model related to code generation

- View and define implementation data in one place
- View implementation details without model details
- Improve code performance and ease integration with other C code



## Row-major memory layout option

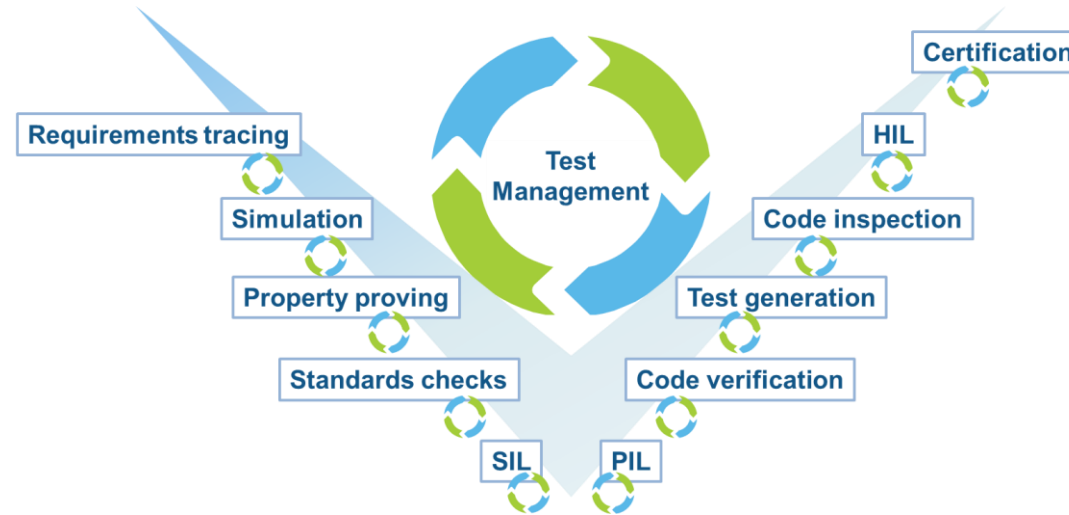
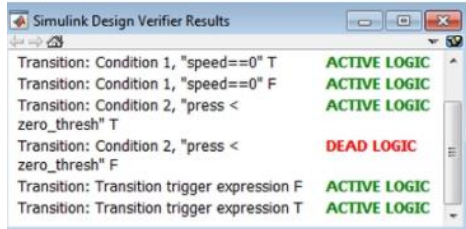
# Verification and Validation

## Products for the entire workflow

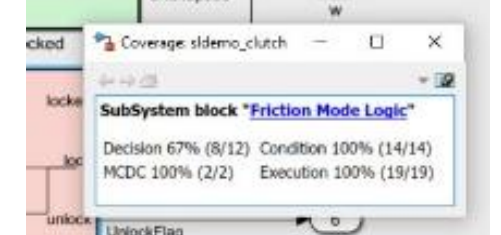
### Simulink Requirements R2017b



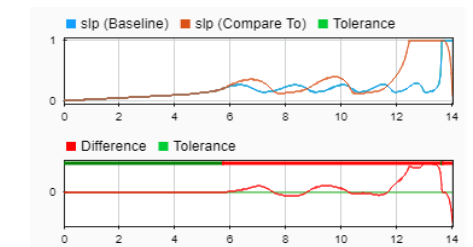
### Simulink Design Verifier



### Simulink Coverage R2017b



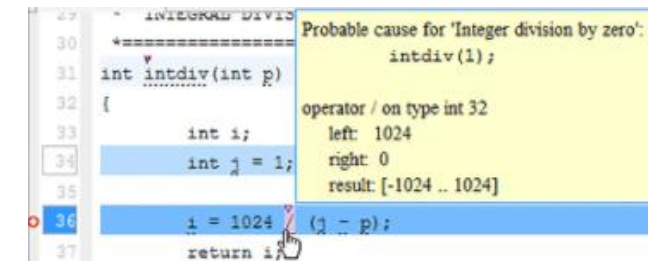
### Simulink Test



### Simulink Check R2017b

- Modeling Standards for Secure Coding (CERT C, CWE, ISO/IEC TS 17961)
  - Check configuration parameters for secure coding standards
  - Check for blocks not recommended for C/C++ production code deployment
  - Check for blocks not recommended for secure coding standards
  - Check usage of Assignment blocks
  - Check for switch case expressions without a default case
  - Check for bitwise operations on signed integers
  - Check for equality and inequality operations on floating-point values
  - Check integer word lengths
  - Detect Dead Logic

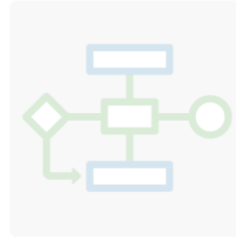
### Polyspace



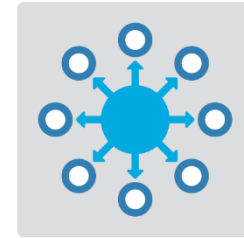
## Platform Productivity



## Workflow Depth

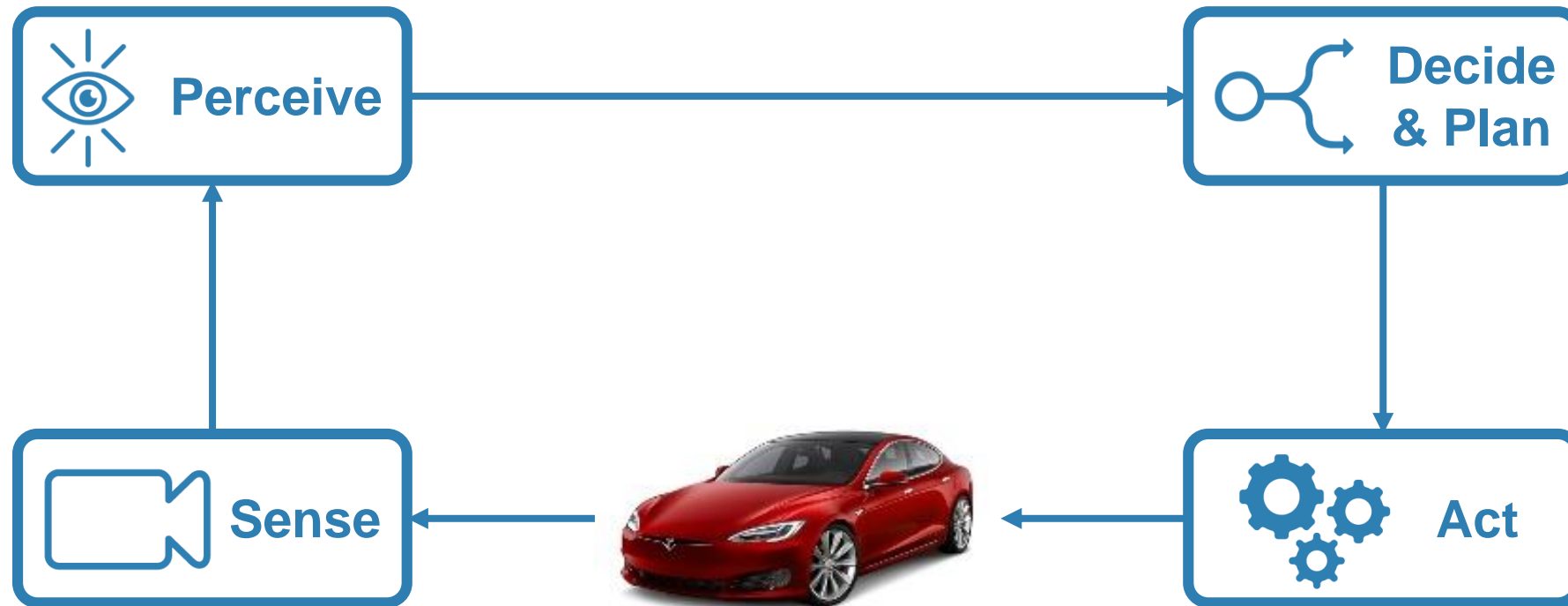


## Application Breadth



- **Autonomous Systems**
- **Wireless Communications**
- **Artificial Intelligence (AI)**

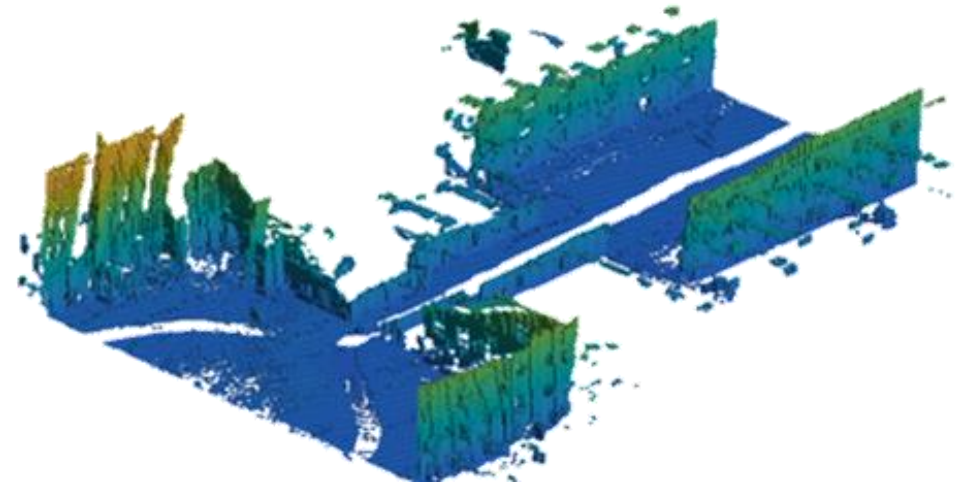
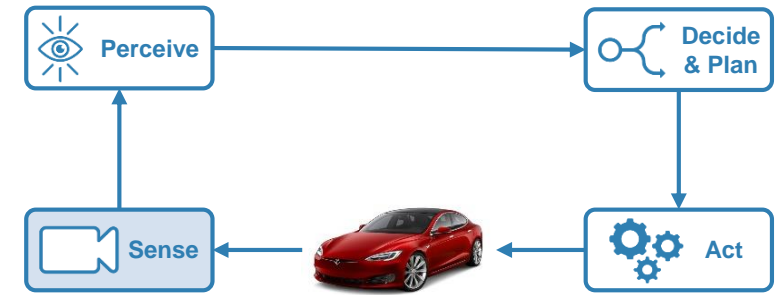
# Designing Autonomous Systems



# Designing Autonomous Systems

## Mapping of environments using sensor data

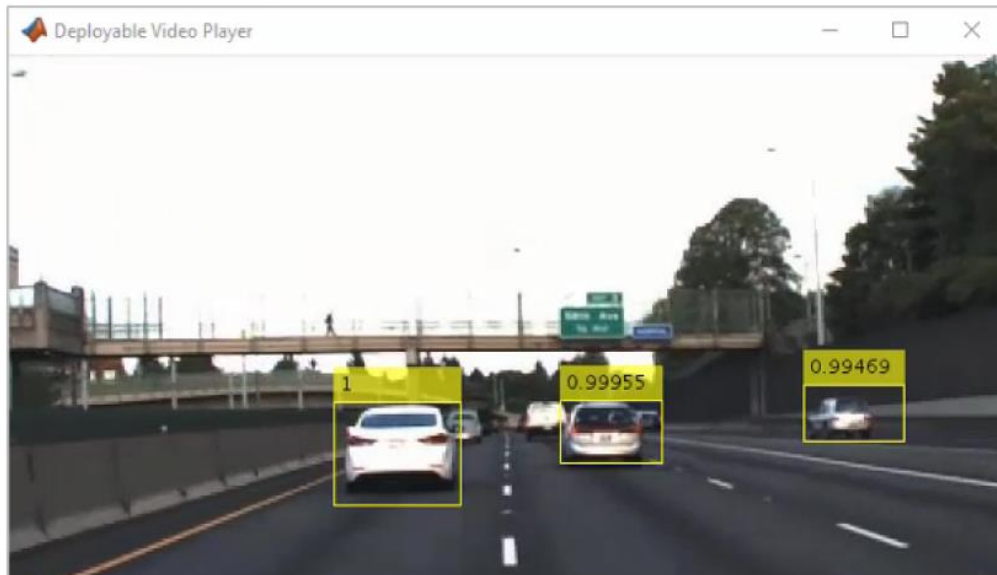
- Segment and register lidar point clouds
- Lidar-Based SLAM: Localize robots and build map environments using lidar sensors



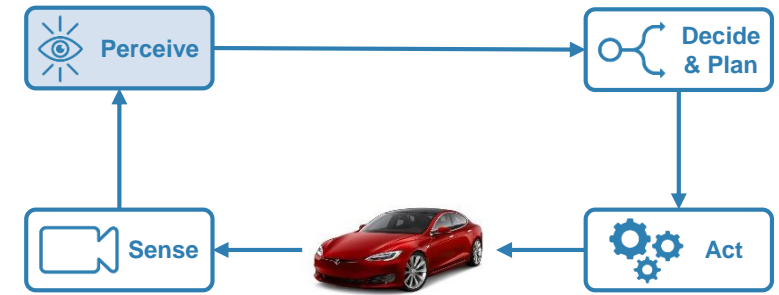
# Designing Autonomous Systems

## Understanding the environment using computer vision and deep learning techniques

- Object detection and tracking
- Semantic segmentation using deep learning



Neural Network Toolbox  
 Computer Vision System Toolbox  
 Automated Driving System Toolbox



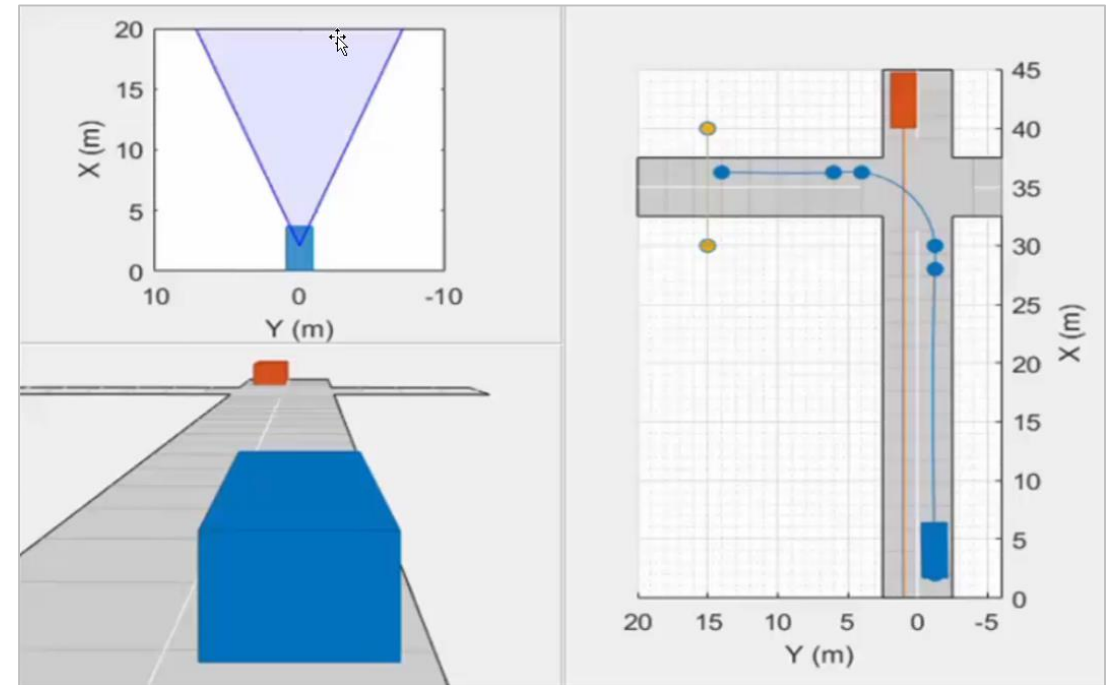
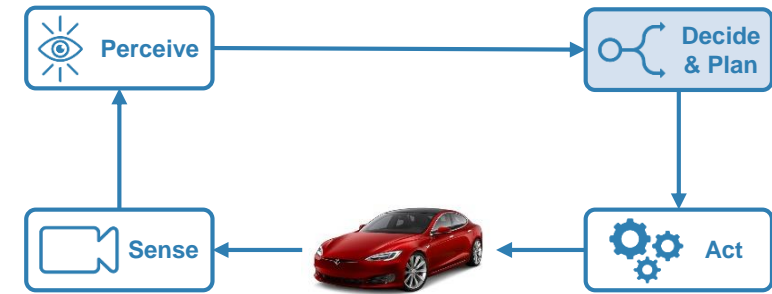
CamVid Database: Brostow, Gabriel J., Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database." *Pattern Recognition Letters* Vol 30, Issue 2, 2009, pp 88-97.



# Designing Autonomous Systems

Design synthetic driving scenarios to test controllers and sensor fusion algorithms

- Interactively design synthetic driving scenarios composed of roads and actors (*vehicles, pedestrians, etc.*)
- Generate visual and radar detections of actors

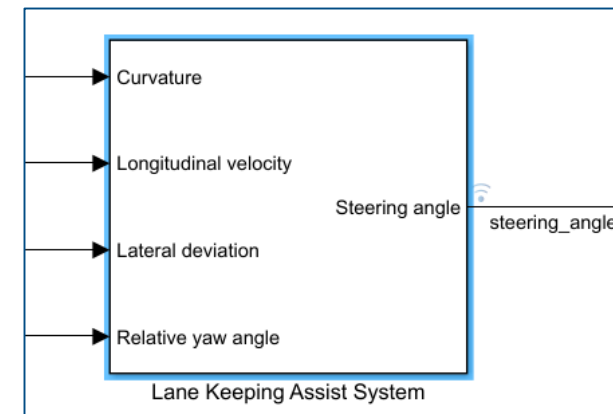
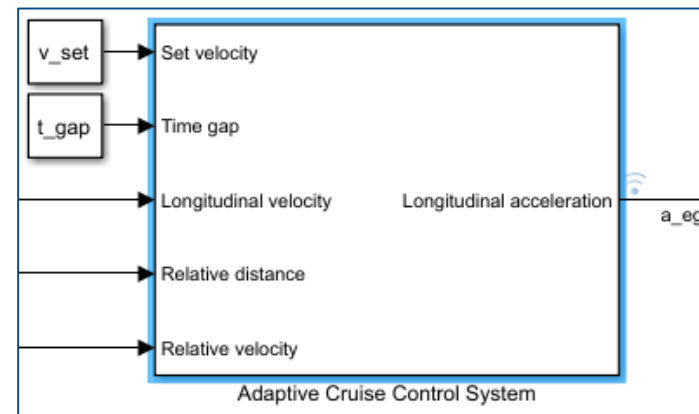
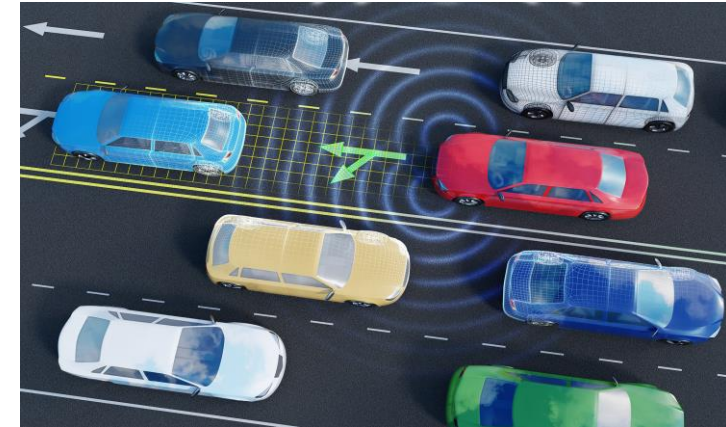
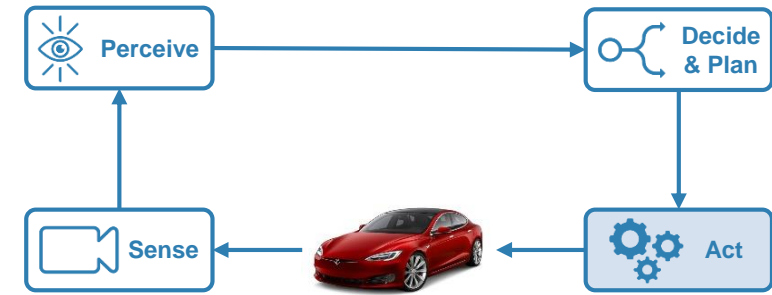


Driving Scenario Designer App

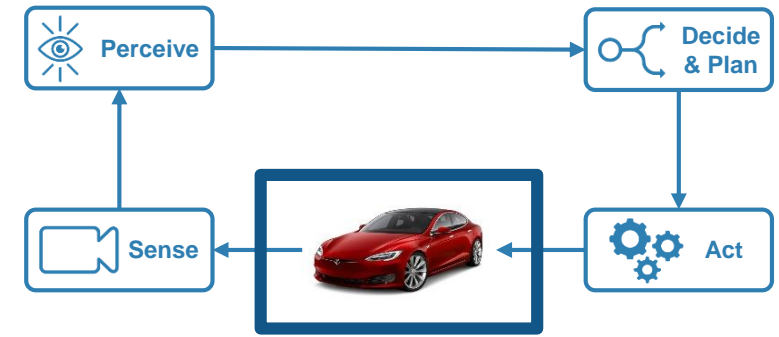
# Designing Autonomous Systems

## Model predictive control for adaptive cruise control and lane-keeping algorithms

- Use prebuilt blocks instead of starting from scratch
- Simplified application-specific interfaces for configuring model predictive controllers
- Flexibility to customize for your application



# Full Vehicle Simulation



Ride & handling



Chassis controls



Automated Driving

# Design with the Latest Wireless Standards



**Lte**™  
Advanced  
Pro



**5G**™



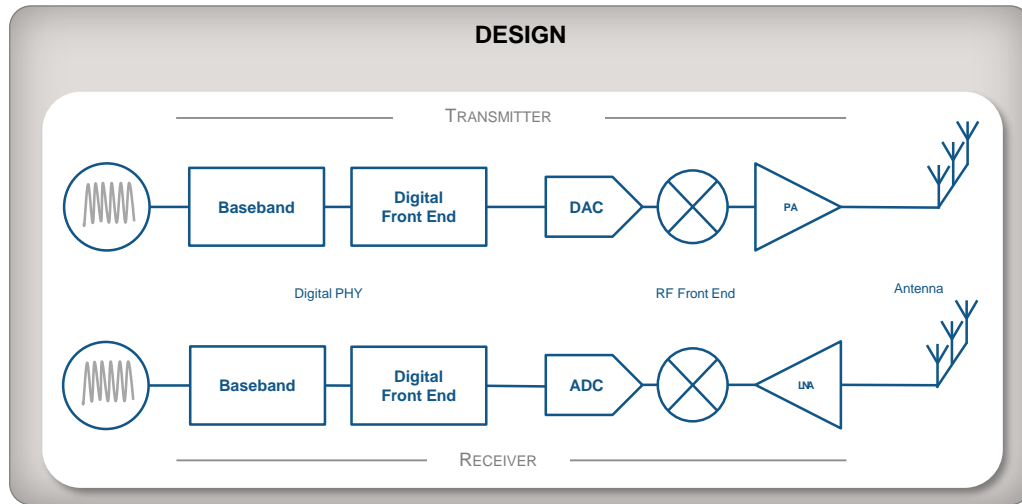
**WiFi**™  
802.11ax



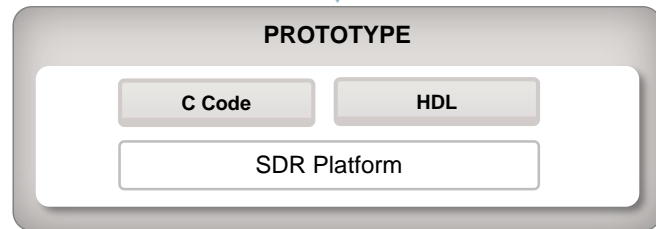
**ZigBee**®

**NB-IoT**

# Model-Based Design for Wireless Communications



- Algorithm Design and Verification
- RF, Digital and Antenna Co-Design
- System Verification and Testing
- Rapid Prototyping and Production



**Code Generation and Verification**

Fixed-Point Designer

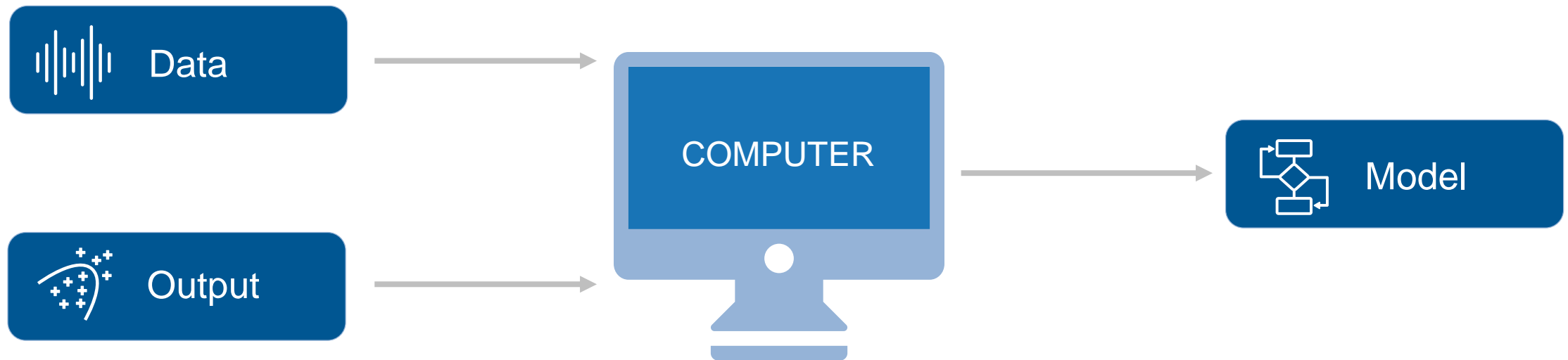
HDL Coder

HDL Verifier

LTE HDL Toolbox **R2017b**

Embedded Coder

# Artificial Intelligence



# Text Analytics

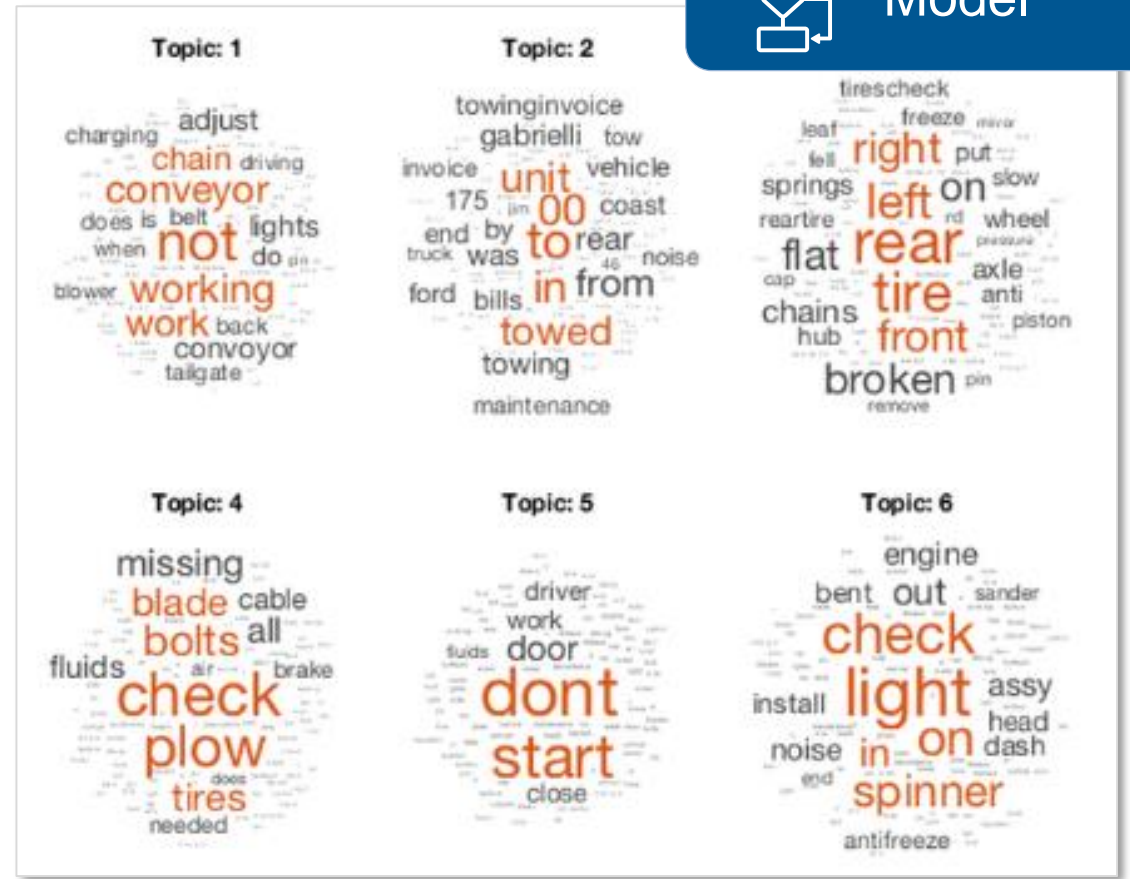
Data

```
repairNotes = 617x1 string array
"PM SERVICE, CHECK TURN SIGNAL, CLUNKING NOISE"
"SERVICEROB,EXT,5604"
"NEED 4 PLOW PINS"
"INSTALL SPINNER ASSY"
"DONT START"
"DOG BONE PIN BROKEN"
"NEED SERVICE, CHECK BRAKES"
"HYD CAP CHECK ENGINE LIGHT ON"
"TARP VALVE STICKINGRIGHT SIDE MIRROR BRACKET B"
"HANDLES IN CAB LOOSE"
"NO PLOW LIGHTS"
"WILL NOT START"
```

Output



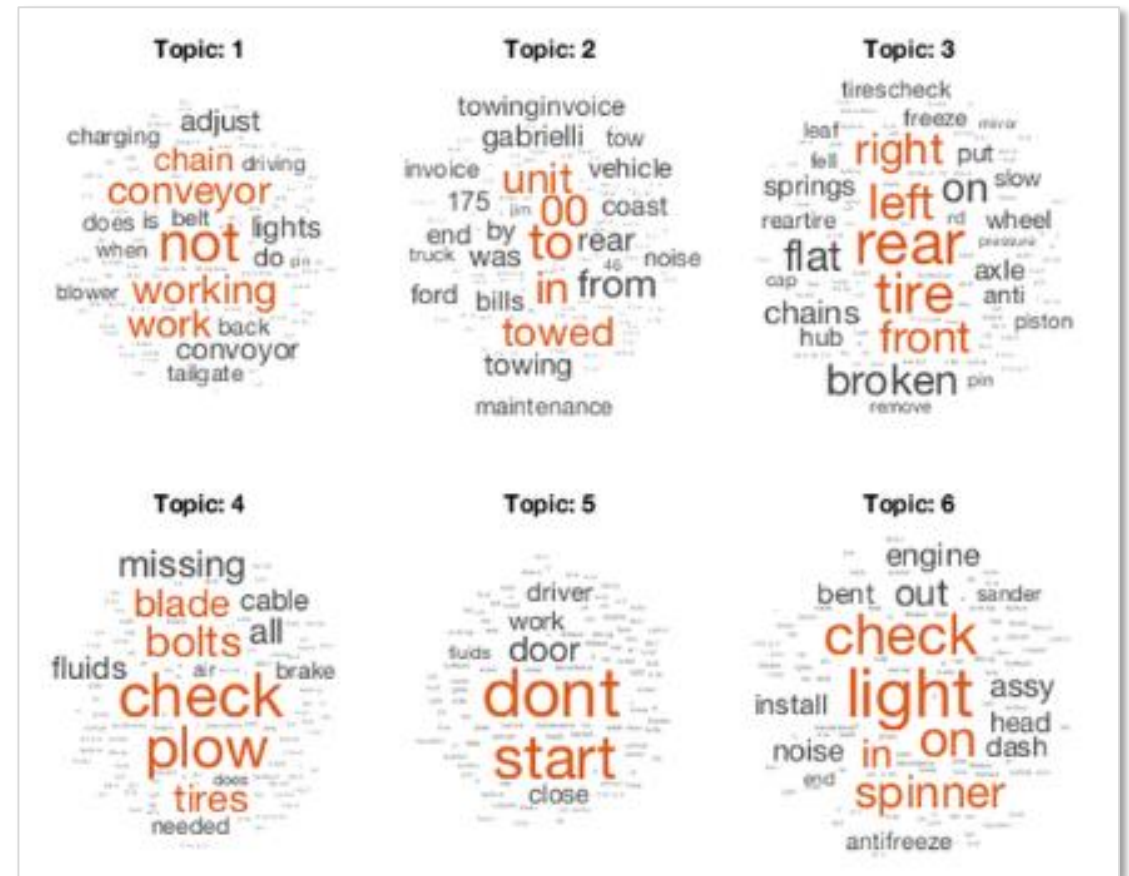
Model



# Text Analytics

## Work with text from equipment logs and operator reports

- **Preprocess** raw text data by extracting, filtering, and splitting
- **Visualize** text using word clouds and text scatter plots
- **Develop** predictive models using built-in machine learning algorithms (LDA, LSA, word2vec)



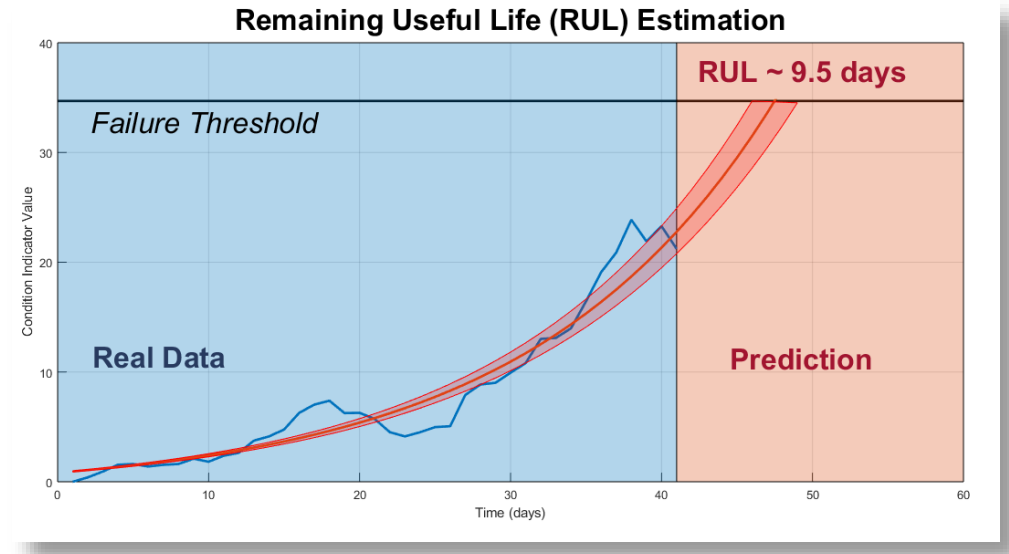
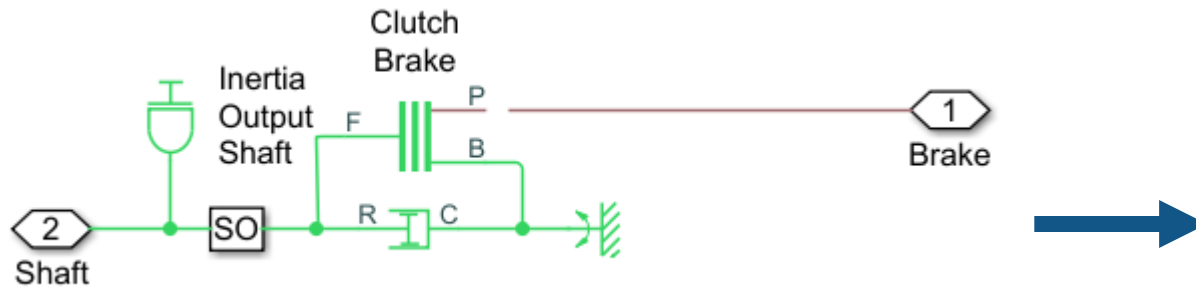


# Predictive Maintenance

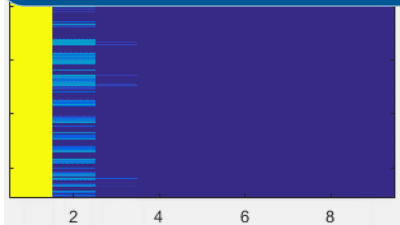
 Data

 Sensors

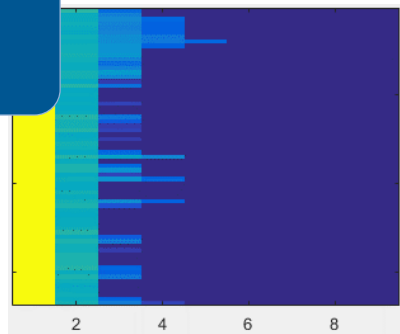
 Model



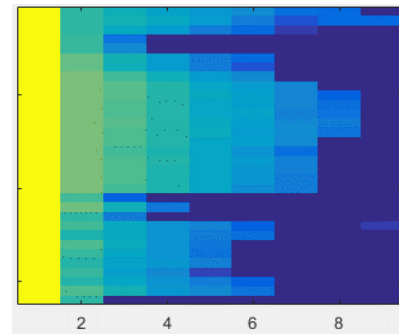
 Output



Normal Operation



Monitor Closely

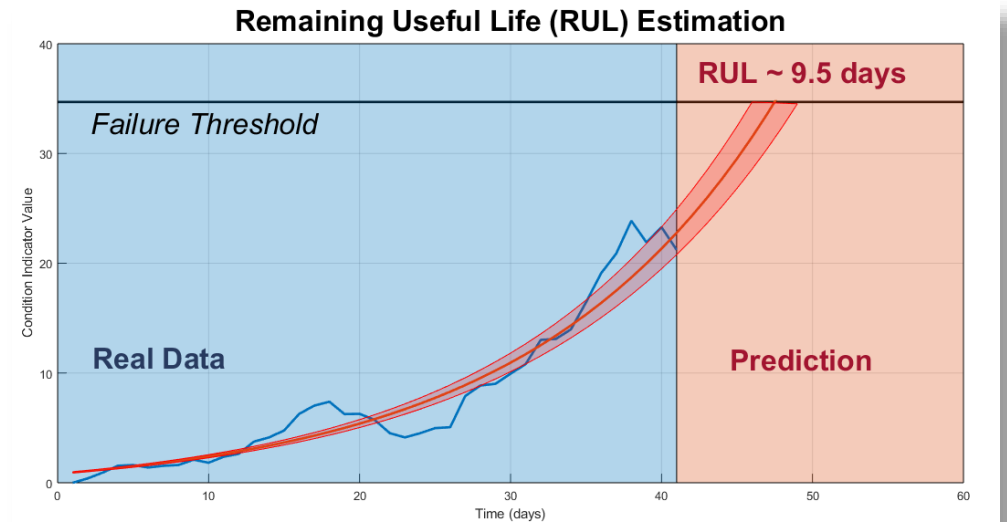


Maintenance Needed

# Predictive Maintenance

## Design and test condition monitoring and predictive maintenance algorithms

- Import sensor data from local files and cloud storage (*Amazon S3, Windows Azure Blob Storage, and Hadoop HDFS*)
- Use simulated failure data from Simulink models
- Estimate remaining useful life (RUL)
- Get started with examples (*motors, gearboxes, batteries, and other machines*)

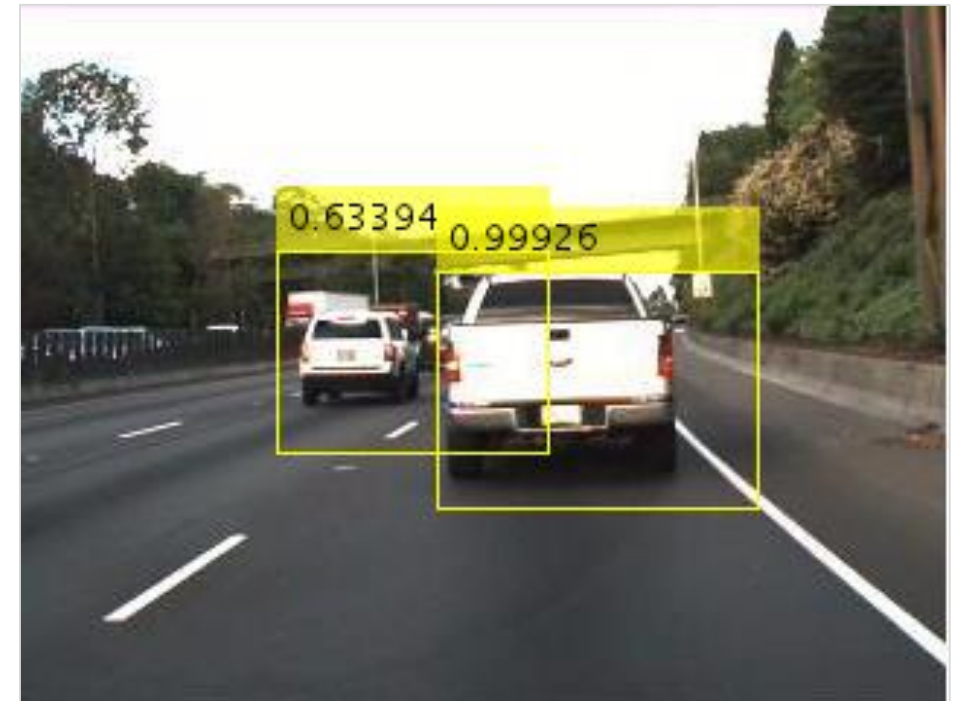


# Deep Learning

Data



Model



Output

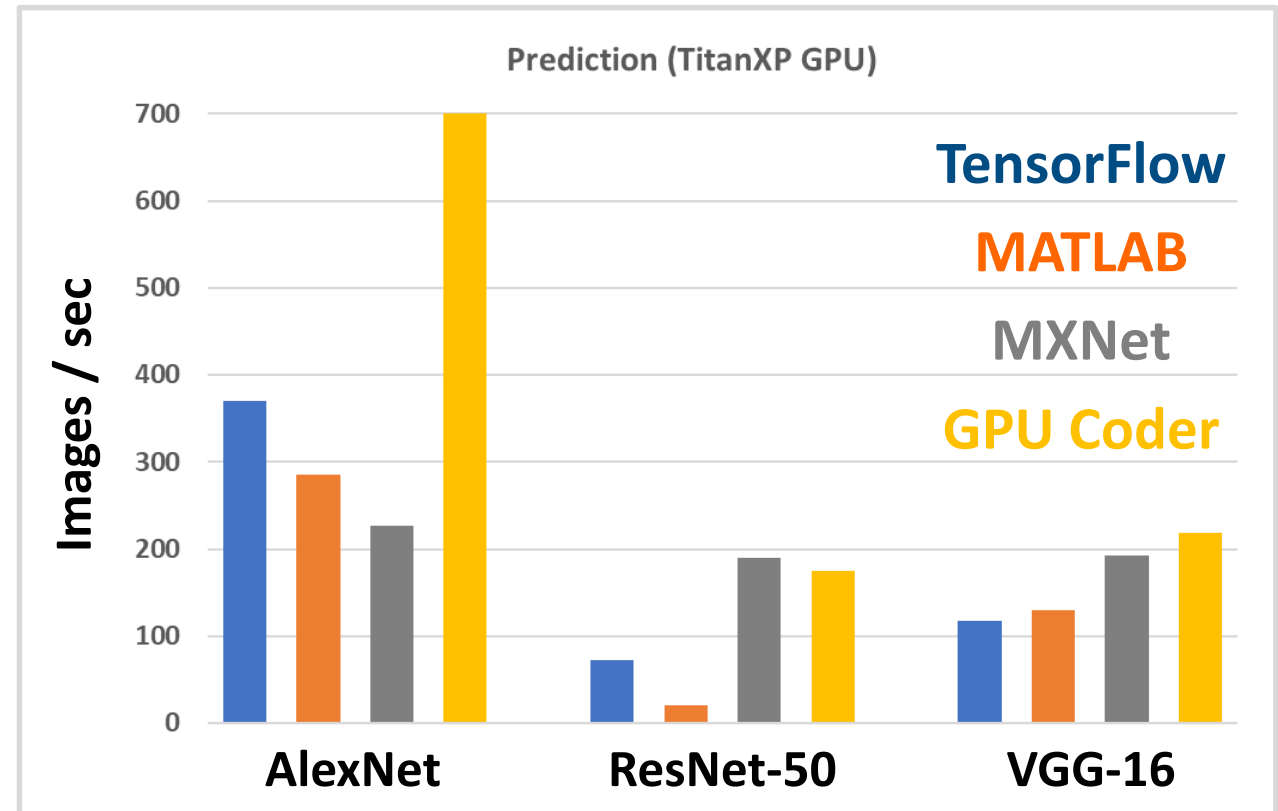


Neural Network Toolbox  
 Computer Vision System Toolbox  
 GPU Coder

# Deep Learning

## Design, build, and visualize convolutional neural networks

- Access the latest models
- Import pretrained models and use transfer learning
- Automate ground-truth labeling using apps
- Design and build your own models
- Use NVIDIA GPUs to train your models
- Automatically generate high-performance CUDA code for embedded deployment



FREE

# Learn to Use MATLAB for Deep Learning in 2 Hours

Launch Deep Learning Onramp

The screenshot shows the MATLAB Deep Learning Onramp interface. The top navigation bar includes "My Courses", "Deep Learning Onramp" (51% complete), and user information "Chal Chitale". The main content area is titled "2.2 Making Predictions: (1/2) Make a prediction".

**Task 2** (Reset):

**Info:** You can use the `classify` function to make a prediction on an image.

```
pred = classify(net,img);
```

Use the `classify` function with the pretrained AlexNet network to predict the subject of the image stored in the variable `img1`. Store the network's prediction in a variable called `pred1`.

You may want to leave off the semicolon to see the result.

Buttons: Submit, Hint, See Solution, Next task

**Correct!**

**Test Suite**

- ✓ Is `pred1` created correctly?
  - Show test suite details

**Task 3**

**Further Practice**

The **LIVE EDITOR** shows the following code:

```
deepnet = alexnet;
```

**Import, view, and classify an image**

Import and display the image in `file01.jpg`.

```
img1 = imread('file01.jpg');
imshow(img1)
```

**Task 2: Classify the image in the variable `img1`.**

```
pred1 = classify(deepnet,img1)
```

**Classify further images**

**Task 3: Classify the images in `file02.jpg` and `file03.jpg`.**

```
img2 = imread('file02.jpg');
```

The **WORKSPACE** shows the result: `pred1 = categorical seashore`. A preview of the image (a beach scene) is shown in the workspace area.

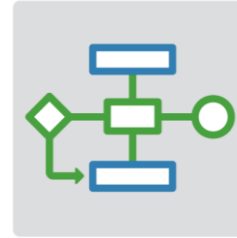
# What's New in MATLAB and Simulink?

## Platform Productivity



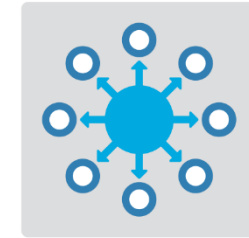
- Design Creation
- Analysis
- Simulation, Scaling
- Collaboration

## Workflow Depth



- Deployment
- Code Generation
- Verification and Validation

## Application Breadth



- Autonomous Systems
- Wireless Communications
- Artificial Intelligence (AI)

# Upgrade your MATLAB Code and Simulink Models

**Upgrade Advisor - sf\_climate\_control**

File Edit Run Settings Help

Find:     Disable Upgrade Notifications

- Check model for SB2SL blocks
- Check that the model is saved in SLX format
- Identify configurable subsystem blocks for converting
- Identify Variant Model blocks and convert those to V
- Check and update masked blocks in library to use pr
- Check and update mask image display commands wi
- Check rapid accelerator signal logging
- Check get\_param calls for block CompiledSampleTim
- Check Model History properties
- Identify Model Info blocks that can interact with exte
- Check and update mask to affirm icon drawing comm
- Check model for Aerospace Blockset legacy 3DoF or r
- Check model for Aerospace Blockset navigation block
- Check and update model to use toolchain approach t
- Check virtual bus inputs to blocks

**Web Browser - (3 Errors) Code Compatibility Report**

(3 Errors) Code Compatibility Report

**Code Compatibility Report** Top 3 Errors 1 Warning 304 Checks 2 Files

Check Name	Result
Check model settings for migration to simplified initialization mode	Passed
Check that the model is saved in SLX format	Passed
Check usage of function-call connections	Passed
Check and set embedded target model to use ert.tlc system target file	Passed
Check and update masked blocks in library to use promoted parameters	Passed
Check and update mask image display commands with unnecessary imread() function calls	Passed
Check and update mask to affirm icon drawing commands dependency on mask workspace	Passed
Check and update model to use toolchain approach to build generated code	Passed

**Upgrade Project Report**

100% Passed

Passed  
 Need attention

	Models	Libraries	MATLAB Code
Passed	7	1	8
Need attention	-	-	-

Show: All Files All Results

File	Check Name	Result
AnalogControl.mdl	AnalogControl.mdl	Passed 27 checks
analyzeModelFiles.m	<a href="#">analyzeModelFiles.m</a>	Passed with fixes 3 checks
billOfMaterials.m	<a href="#">billOfMaterials.m</a>	Need attention -
checkCodeProblems.m	<a href="#">checkCodeProblems.m</a>	
DigitalControl.slx	Check Name	
f14_airframe.slx	Check model settings for migration to simplified initialization mode	Passed
f14_airframe_test.m	Check that the model is saved in SLX format	Passed
find_top_models.m	Check usage of function-call connections	Passed
LinearActuator.slx	Check and set embedded target model to use ert.tlc system target file	Passed
NonLinearActuator.mdl	Check and update masked blocks in library to use promoted parameters	Passed
rebuild_s_functions.m	Check and update mask image display commands with unnecessary imread() function calls	Passed
runUnitTest.m	Check and update mask to affirm icon drawing commands dependency on mask workspace	Passed
slproject_f14.slx	Check and update model to use toolchain approach to build generated code	Passed
upgrade_project.m		
vertical_channel.slx		
wind_gust_lib.slx		

Checks run on 02/01/2018 10:44

MATLAB EXPO 2018

