MATLAB EXPO 2019

What’s New in MATLAB and Simulink
Using MATLAB & Simulink to Build Algorithms in Everything

Simplifying your work...

...often at higher levels of abstraction.
Using MATLAB & Simulink to Build Algorithms in Everything

Inputs → Design → Outputs

MATLAB & SIMULINK®
Artificial Intelligence

The capability of a machine to match or exceed intelligent human behavior by *training a machine to learn the desired behavior*
There are two ways to get a computer to do what you want

Traditional Programming

Data → COMPUTER → Output
Program → COMPUTER
There are two ways to get a computer to do what you want

Machine Learning

Data → COMPUTER → Model

Output
Artificial Intelligence

Data → Machine Learning → Deep Learning → Model
Using MATLAB and Simulink to Build Deep Learning Models

Data → Machine Learning → Deep Learning → Model

Inputs → Design → Outputs

MATLAB & SIMULINK®
Using Apps for Ground Truth Labeling
Image and Video Data

Computer Vision Toolbox
Using Apps for Ground Truth Labeling
Signal Data

Signal Processing Toolbox
Using Apps for Ground Truth Labeling
Audio Data
Using Apps for Designing Deep Learning Networks

Deep Learning Toolbox
Using Transfer Learning with Pre-trained Models

- Inception-v3
- ResNet-101
- VGG-16
- Inception-ResNet-v2

- ResNet-18
- GoogLeNet
- DenseNet-201
- VGG-19

- SqueezeNet
- AlexNet
- ResNet-50
Using Models from Other Frameworks

MATLAB

Keras-Tensorflow

Caffe

ONNX

PyTorch

CNTK

(...)

Caffe2

MXNet

Core ML

Deep Learning Toolbox
Deploying Deep Learning Applications

- MATLAB Coder
- GPU Coder
- Pre-processing
- Deep Learning Networks
- Post-processing
- Coder Products
  - Intel MKL-DNN Library
  - NVIDIA TensorRT & cuDNN Libraries
  - ARM Compute Library
Using MATLAB and Simulink for Reinforcement Learning
Using MATLAB and Simulink for Reinforcement Learning
Using MATLAB and Simulink for Reinforcement Learning

Inputs → Machine Learning → Outputs

Data → Deep Learning → Model
Using MATLAB and Simulink for Reinforcement Learning

Generate Data
- Scenario Design
- Simulation-based data generation

Inputs

Machine Learning
Deep Learning

Design

Outputs

Model

MATLAB & SIMULINK

Simulink
Reinforcement Learning Toolbox
Using MATLAB and Simulink for Reinforcement Learning
Using MATLAB & Simulink to Build Algorithms in Everything

Inputs → Design → Outputs

MATLAB & SIMULINK®
## Working with Text Data

In this section, we will explore how to work with text data in MATLAB. The data provided is a list of vehicle repair records, which can be used to demonstrate various text processing techniques.

### Vehicle Repairs Log

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>VehicleID</th>
<th>UnitNo</th>
<th>Reason</th>
<th>Notes</th>
<th>CostParts</th>
<th>CostLabor</th>
<th>CostTotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-01-01</td>
<td>12:00</td>
<td>1234</td>
<td>567</td>
<td>Traction</td>
<td>Issue with wheel bearings</td>
<td>90</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>2015-01-02</td>
<td>08:30</td>
<td>789</td>
<td>111</td>
<td>Engine</td>
<td>Overheating</td>
<td>120</td>
<td>15</td>
<td>135</td>
</tr>
</tbody>
</table>

### Example Usage

```matlab
% Read the vehicle repair log
repairs = readtable('Vehicle_Repair.csv');

% Display the first few rows of the data table
disp(repairs(1:5, :));
```

This code reads the vehicle repair log from a CSV file and displays the first few rows of the data table. The data can then be processed further using various text processing functions available in MATLAB.
t = readtable(filename,'TextType','string');
disp(t(1:20,6:7))

<table>
<thead>
<tr>
<th>Reason</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;04 DRIVER'S REPORT&quot;</td>
<td>&quot;PM SERVICE, CHECK TURN SIGNAL, CLUNKING NOISE WHEN DRIVING&quot;</td>
</tr>
<tr>
<td>&quot;08 PM SERVICE&quot;</td>
<td>&quot;SERVICEROB,EXT,5604&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;NEED 4 PLOW PINS&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;INSTALL SPINNER ASSY&quot;</td>
</tr>
<tr>
<td>&quot;13 SNOW BREAKDOWN&quot;</td>
<td>&quot;DON'T START&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;DOG BONE PIN BROKEN&quot;</td>
</tr>
<tr>
<td>&quot;08 PM SERVICE&quot;</td>
<td>&quot;NEED SERVICE, CHECK BRAKES&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;HYD CAP CHECK ENGINE LIGHT ON&quot;</td>
</tr>
<tr>
<td>&quot;40 NEGLIGENCE&quot;</td>
<td>&quot;TARP VALVE STICKINGRIGHT SIDE MIRROR BRACKET BROKEN&quot;</td>
</tr>
<tr>
<td>&quot;13 SNOW BREAKDOWN&quot;</td>
<td>&quot;HANDLES IN CAB LOOSE&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;NO PLOW LIGHTS&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;WILL NOT START&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;WILL NOT START&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;WILL NOT START&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;WILL NOT START&quot;</td>
</tr>
<tr>
<td>&quot;04 DRIVER’S REPORT&quot;</td>
<td>&quot;CONVEYOR NOT WORKING&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;DON'T START&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;DON'T START&quot;</td>
</tr>
<tr>
<td>&quot;10 ROADCALL&quot;</td>
<td>&quot;DON'T START&quot;</td>
</tr>
</tbody>
</table>
Working with Text Data

Deep Learning Toolbox
Statistics and Machine Learning Toolbox
Text Analytics Toolbox
MATLAB
Working with Text Data
Creating Your Own Data
Identifying the Useful Data

Acquire Data -> Preprocess Data -> Identify Condition Indicators -> Train Model -> Deploy & Integrate

- Visualize data
- Extract Features
- Select the most useful features

Machine Learning
Identifying the Useful Data

![Signal trace plot for Vibration/Data in focus.](image)
Identifying the Useful Data

Signal Features
- Generate statistics features from signals

Rotating Machinery Features
- Generate features from rotating machinery signals

Nonlinear Features
- Generate nonlinear features from signals

Predictive Maintenance Toolbox
Identifying the Useful Data
Designing Decision Logic with Stateflow in MATLAB

```matlab
inNormalRegion = true;
counter = 0;
for i=1:length(inData)
    if(inNormalRegion)
        if(inData(i)<t1)
            counter = counter+1;
            if(counter>=N1)
                inNormalRegion = false;
            end
        else
            counter = 0;
        end
    else
        if(inData(i)>=t2)
            counter = counter+1;
            if(counter>=N2)
                inNormalRegion = true;
            end
        else
            counter = 0;
        end
    end
    if(inNormalRegion)
        outData(i) = inData(i);
    else
        outData(i) = 0;
    end
end
```
Using Stateflow in MATLAB

% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    app.lanternLogic = blink.lanternLogic('app',app);
end

% Button pushed function: POWERButton
function POWERButtonPushed(app, event)
    app.lanternLogic.powerButton();
end

% Button pushed function: COLORButton
function COLORButtonPushed(app, event)
    app.lanternLogic.colorButton();
end

% Close request function: UIFigure
function UIFigureCloseRequest(app, event)
    delete(app.lanternLogic);
    delete(app);
end

% Button pushed function: BLINKButton
function BLINKButtonPushed(app, event)
    app.lanternLogic.blinkButton();
end

end
Editing at the Speed of Thought

[Diagram showing a Simulink model with blocks labeled 'Sources', 'Operator', 'Environment', 'Controller data type = single', 'Mechanical System', and 'x_sensor']
Editing at the Speed of Thought
Editing at the Speed of Thought
Editing at the Speed of Thought
Editing at the Speed of Thought
Controlling the Execution of Model Components
Controlling the Execution of Model Components
Simplifying Integration with External C/C++ Code

```
#include "rtwdemo_rowlutcol2row_workflow_rowrow.h"

/* Block parameters (default storage) */
Param = {
    /* Variable: Tbl_1 */
    /* Referenced by: '<Root>/2-D Lookup Table' */
    {
        [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0]};
```
Simplifying Integration with External C/C++ Code

Row-Major
Viewing Generated Code Alongside the Model
Viewing Generated Code Alongside the Model

Fuel Rate Control Subsystem

- Sensors
- EngSensors D2
- Control Logic
- Fuel Mode
- Est Airflow (gs)
- Est Airflow (gs)
- Single D2 (g/s)
- Single D2 (g/s)
- Single D2 (g/s)
- Single D2 (g/s)
- Single D2 (g/s)
- Fuel Rate (g/s)
- Fuel Rate (g/s)
- Fuel Rate (g/s)
- Fuel Rate (g/s)
Estimating Sunrise and Sunset

Using the latitude ($\phi$), the sun's declination ($\delta$) and the solar time correction ($SC$) we can calculate sunrise and sunset times.

\[
\text{sunrise} = 12 - \cos^{-1}\left(-\tan \phi \tan \delta\right) - \frac{SC}{60} \quad \text{sunrise} = 12 + \cos^{-1}\left(-\tan \phi \tan \delta\right) - \frac{SC}{60}
\]

Refer to this page for background and details on the equations used.
Sharing Live Scripts

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.

```matlab
exoplanets = readtable('exoplanets.xls');
exoplanets(X18M);
```

**How Far Away Are these Planets?**

There are 90 exoplanets within 50 light-years of earth and 460 exoplanets within 200 light-years.

```matlab
histogram1(50*exoplanets.st_distance,'BinWidth', 50); axis([0 1000])
ylabel('Number of Planets')
xlabel('Light Years From Earth')
```

**Where is the nearest exoplanet?**

```matlab
idx = find(exoplanets.st_distance == min(exoplanets.st_distance));
name = char(exoplanets(idx, st_name));
```
Sharing Live Scripts
Creating Apps
Deploying Web Apps
Using MATLAB & Simulink to Build Algorithms in Everything
Evaluating Architectures

Inputs → Architecture → Design → Outputs

MATLAB & SIMULINK®
Evaluating Architectures
Designing System and Software Architectures
Designing System and Software Architectures
Designing Beyond System and Software Architectures

Systems and Software

SoC Hardware and Software

AUTOSAR Software

System Composer

SoC Blockset

AUTOSAR Blockset
Using MATLAB & Simulink to Build Algorithms in Everything

Inputs → Architecture → Design → Outputs

- Test & Verification
- Collaboration
- Scaling

© 2019 The MathWorks, Inc.
Using MATLAB & Simulink to Build Algorithms in Everything
Using MATLAB & Simulink to Build Algorithms in Everything
Integrating with Third-party Requirements Tools

External Requirements
- .doc
- .xls
- Requirements Management Tools

Simulink Requirements
- External Requirements
- Authored Requirements

ReqIF
- Import
- Edit
- Export

R2019a
Include Custom Code in Test & Verification

Simulink

C/C++

Stateflow

C/C++

Simulink Design Verifier

- X

- ✓
Using the MATLAB Unit Test Framework

```matlab
>> result.table
ans =
    2×6 table
    Name       Passed     Failed    Incomplete  Duration   Details
'\text{test\_Predictions/Test\_ModelType}'  true        false       false       0.12241     [1×1 struct]
'\text{test\_Predictions/Test\_Prediction}' false       true        true        0.11542     [1×1 struct]
```
Using the MATLAB App Testing Framework

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

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

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

testCase.type(myApp.editfield, myTextVar)
```
Using the MATLAB Performance Testing Framework
Using Continuous Integration

Plugins Index

Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project.

Browse categories
- Platforms
- User interface
- Administration
- Source code management

New Plugins
- OREbel
- MATLAB
- MISRA Compliance Report
- Zoom
- VectorCAST Execution
- Klocwork Community
- JQuery
- Analysis Model API

MATLAB

https://plugins.jenkins.io/
Using Continuous Integration

The Jenkins plugin for MATLAB® enables you to easily run your MATLAB tests and generate test artifacts in formats such as JUnit, TAP, and Cobertura code coverage reports.

Features

- Support to run MATLAB tests, present in the Jenkins workspace automatically. (This also includes the tests present in .prj files)
- Generate tests artifacts in JUnit, TAP & Cobertura code coverage formats.
- Support to run tests, using custom MATLAB command or custom MATLAB script file.
### Using Projects in MATLAB

#### Project Explorer

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Git</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Test</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Test</td>
</tr>
<tr>
<td>ACI</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>Dashboard</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>Documents</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>MachineLearning</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>MATLAB_KafkaProducer_java</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>mps_stream</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>SimExecutable</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td></td>
</tr>
<tr>
<td>DocExample_MultiClassFaultDetectionUsing...</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>genPumpData.m</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>javasetup.m</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>Main_ExampleWorkflow.mlx</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>MLModels.mat</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>rawdata.mat</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
<tr>
<td>README.md</td>
<td>✔️</td>
<td><img src="repository" alt="Repository" /></td>
<td>Design</td>
</tr>
</tbody>
</table>
Parallel Simulations in Simulink

Simulation Manager

MATLAB Desktop

Simulation Jobs

Simulation Results

Worker

Worker

Worker

Head Worker

Simulation Manager

batchsim

Simulink
Parallel Computing Toolbox
Scaling Computations on Clusters and Clouds

MATLAB

Parallel Computing Toolbox

MATLAB Parallel Server

Cloud

GPU

Multi-core CPU
Using MATLAB & Simulink to Build Algorithms in Everything

Inputs ➔ Architecture ➔ Design ➔ Outputs

Test & Verification ➔ Collaboration ➔ Scaling
Specialized Tools for Building Algorithms in Everything

Communications

Physical interconnects

Analog Mixed-Signal

5G Toolbox

SerDes Toolbox

Mixed-Signal Blockset
Developing Autonomous Systems

Perception

Planning

Control
Evaluate Sensor Fusion Architectures
Simulate Path Planning Algorithms
Design Lane-following and Spacing Control Algorithms
Developing Autonomous Systems

Lidar Processing & Tracking

HERE HD Maps & OpenDRIVE Roads

UAV Algorithms

Computer Vision Toolbox

Automated Driving Toolbox

Robotics System Toolbox
Using MATLAB & Simulink to Build Algorithms in Everything

- Inputs
- Architecture
- Design
- Outputs

Test & Verification
Collaboration
Scaling

MATLAB & SIMULINK
Read the Release Notes

Explore What's New
Get more out of MATLAB and Simulink by downloading the latest release.

Download release now

Release Highlights

Deep Learning
Develop controllers and decision making systems using reinforcement learning, train deep learning models on NVIDIA DGX and cloud platforms, and apply deep learning to 3-D data.

» Learn more

Automotive
Design and simulate AUTOSAR software, interface with HERE HD maps, and generate energy balance reports.

» Learn more

Systems Engineering
Design and analyze system and software architectures with System Composer.

» Learn more

Projects
Use projects in MATLAB and Simulink to organize, manage, and share your work.

» Learn more
Get Started

MATLAB Onramp
Quickly learn the essentials of MATLAB.

Simulink Onramp
Learn to create, edit, and troubleshoot Simulink models.

Deep Learning Onramp
Learn to use deep learning techniques in MATLAB for image recognition.