MATLAB EXPO 2018

KOREA

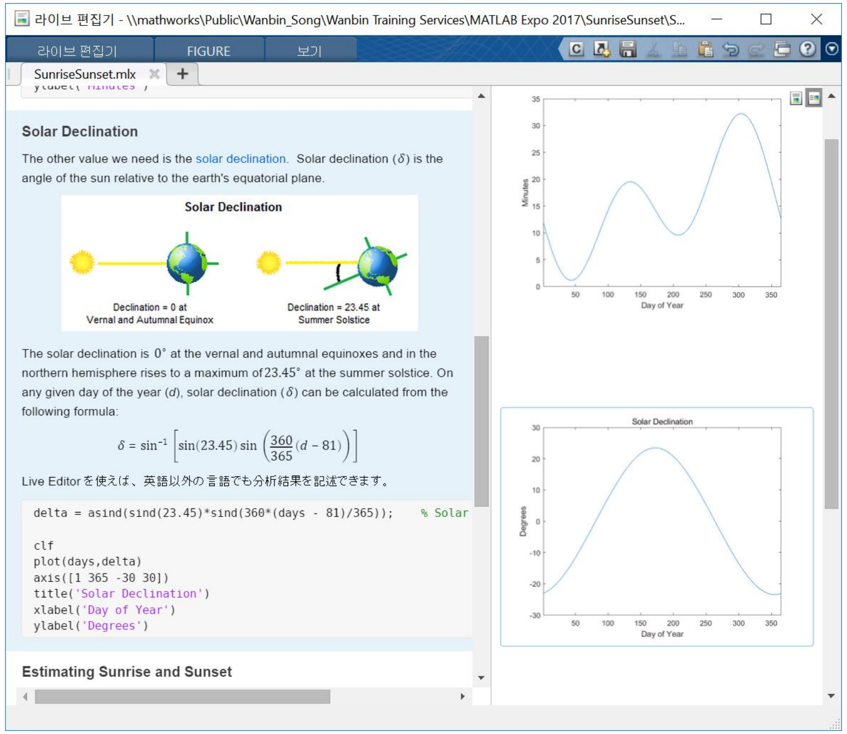
MATLAB EXPO 2018

Interactive programming and enhanced GUI

Hyunuk Ha

Table of contents

- ✓ Live editor
  - ✓ Short introduction to live editor
  - ✓ Key features till R2017b
  - ✓ Feature highlights in R2018a
- ✓ App designer
  - ✓ (Not so) Short introduction to app designer
  - ✓ Key features till R2017b
  - ✓ Feature highlights in R2018a



라이브 편집기 - \\mathworks\Public\Wanbin_Song\Wanbin Training Services\MATLAB Expo 2017\SunriseSunset\S...
라이브 편집기
FIGURE
보기
SunriseSunset.mlx
Solar Declination
Declination = 0 at Vernal and Autumnal Equinox
Declination = 23.45 at Summer Solstice
Minutes
Day of Year
Degrees

Solar Declination

The other value we need is the solar declination. Solar declination (δ) is the angle of the sun relative to the earth's equatorial plane.

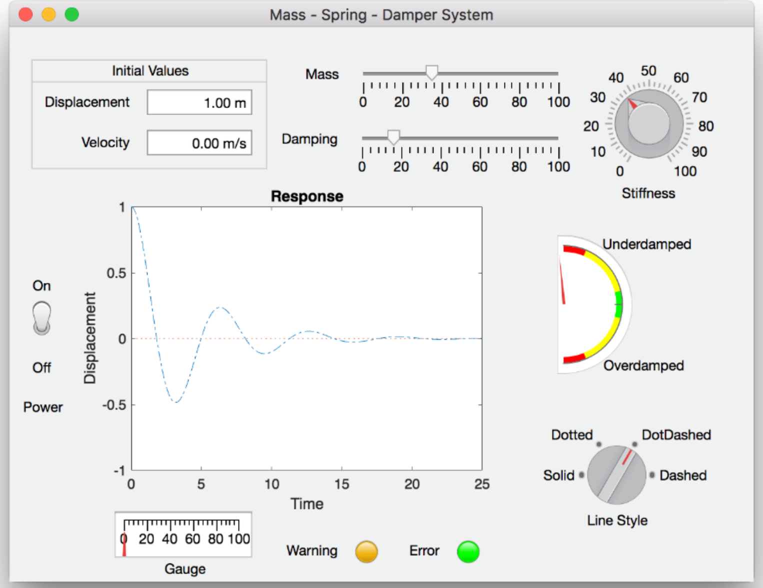
The solar declination is 0° at the vernal and autumnal equinoxes and in the northern hemisphere rises to a maximum of 23.45° at the summer solstice. On any given day of the year (d), solar declination (δ) can be calculated from the following formula:

$$\delta = \sin^{-1}\left[\sin(23.45)\sin\left(\frac{360}{365}(d-81)\right)\right]$$

Live Editor を使えば、英語以外の言語でも分析結果を記述できます。

```
delta = asind(sind(23.45)*sind(360*(days - 81)/365));    % Solar

clf
plot(days,delta)
axis([1 365 -30 30])
title('Solar Declination')
xlabel('Day of Year')
ylabel('Degrees')
```

Estimating Sunrise and Sunset

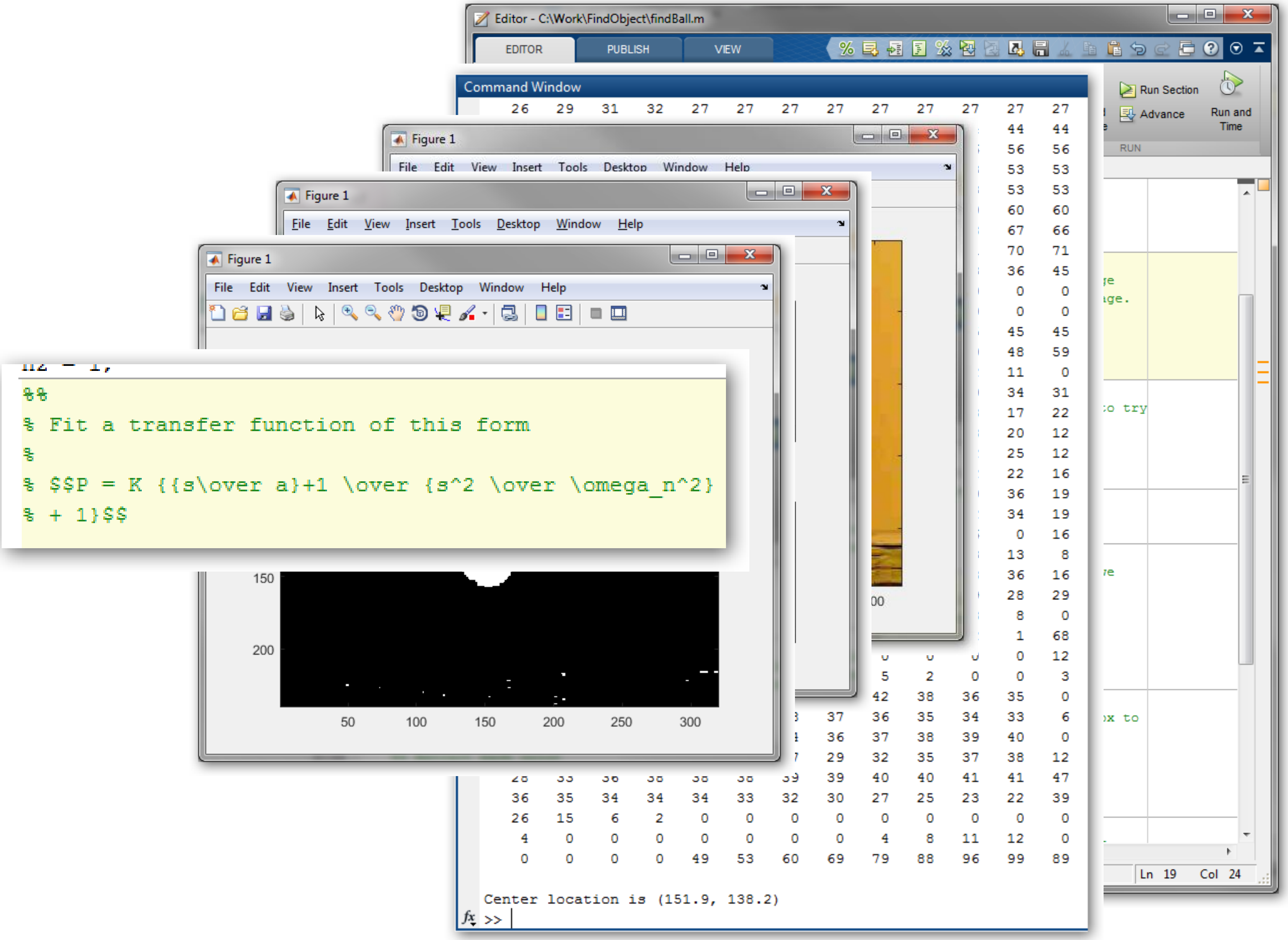


Displacement 1.00 m

Velocity 0.00 m/s

Conventional plain text script

- ✓ Plain-text editing
- ✓ Output goes to Command Window
- ✓ Multiple figure windows appear
- ✓ Equations, images, and hyperlinks only appear if published



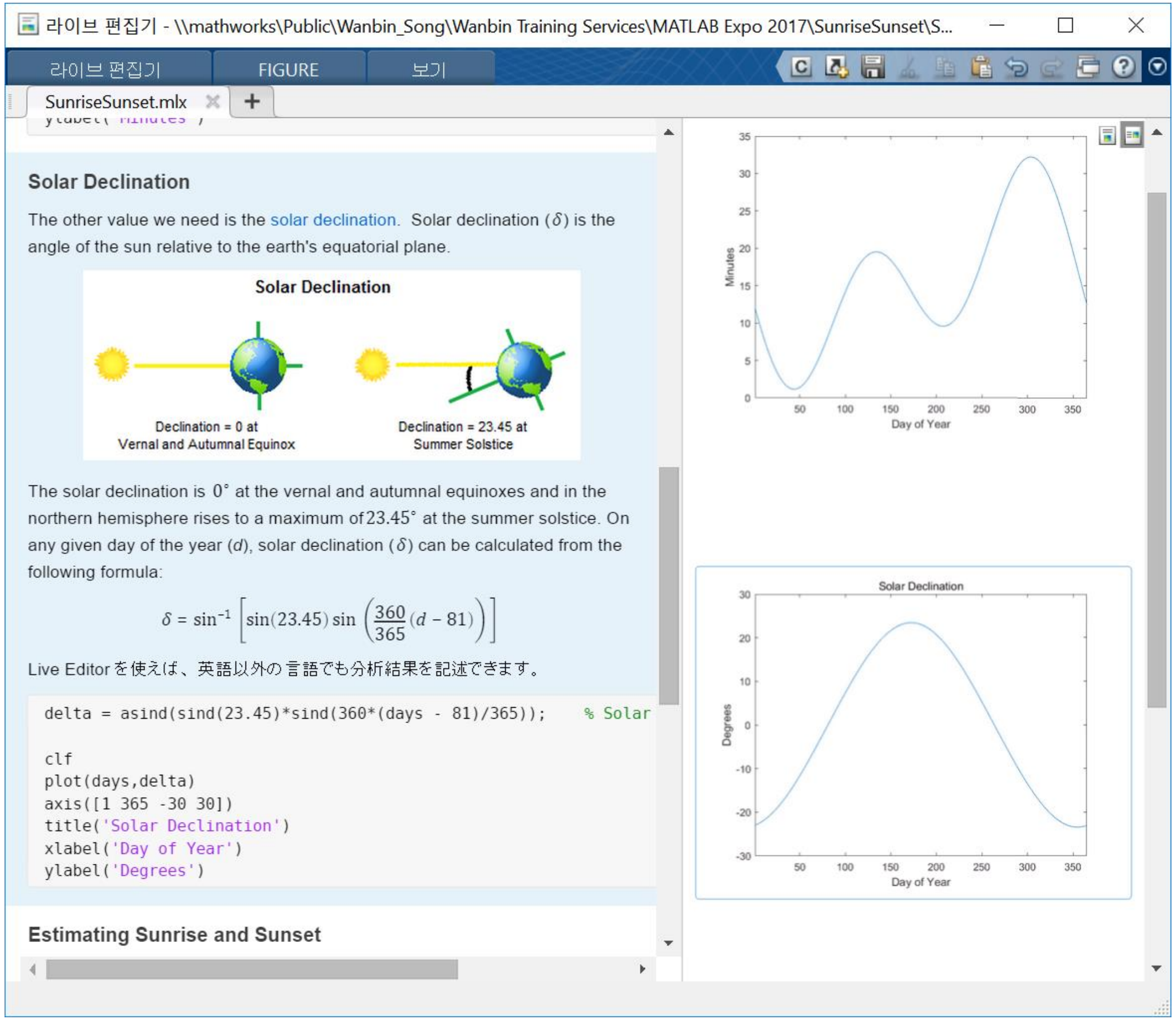
```
%%
% Fit a transfer function of this form
%
% $$P = K {{s\over a}+1 \over {s^2 \over \omega_n^2}
% + 1}$$
```

```
Center location is (151.9, 138.2)
>>
```

What is the Live Editor?

The Live Editor provides **a new way to create, edit and run MATLAB code**.

- ✓ Live editor = script + (fully formatted)text + result
- ✓ Write, edit(debug), and run code in a single interactive environment
- ✓ Generate results and graphics within the integrated developing environment
- ✓ Include (WYSWYG)images, (LaTeX)equations, hyperlinks and table of contents to create an interactive narrative
- ✓ Share your script as a richly formatted and executable document with code and its results



Solar Declination

The other value we need is the solar declination. Solar declination (δ) is the angle of the sun relative to the earth's equatorial plane.

The solar declination is 0° at the vernal and autumnal equinoxes and in the northern hemisphere rises to a maximum of 23.45° at the summer solstice. On any given day of the year (d), solar declination (δ) can be calculated from the following formula:

$$\delta = \sin^{-1}\left[\sin(23.45)\sin\left(\frac{360}{365}(d-81)\right)\right]$$

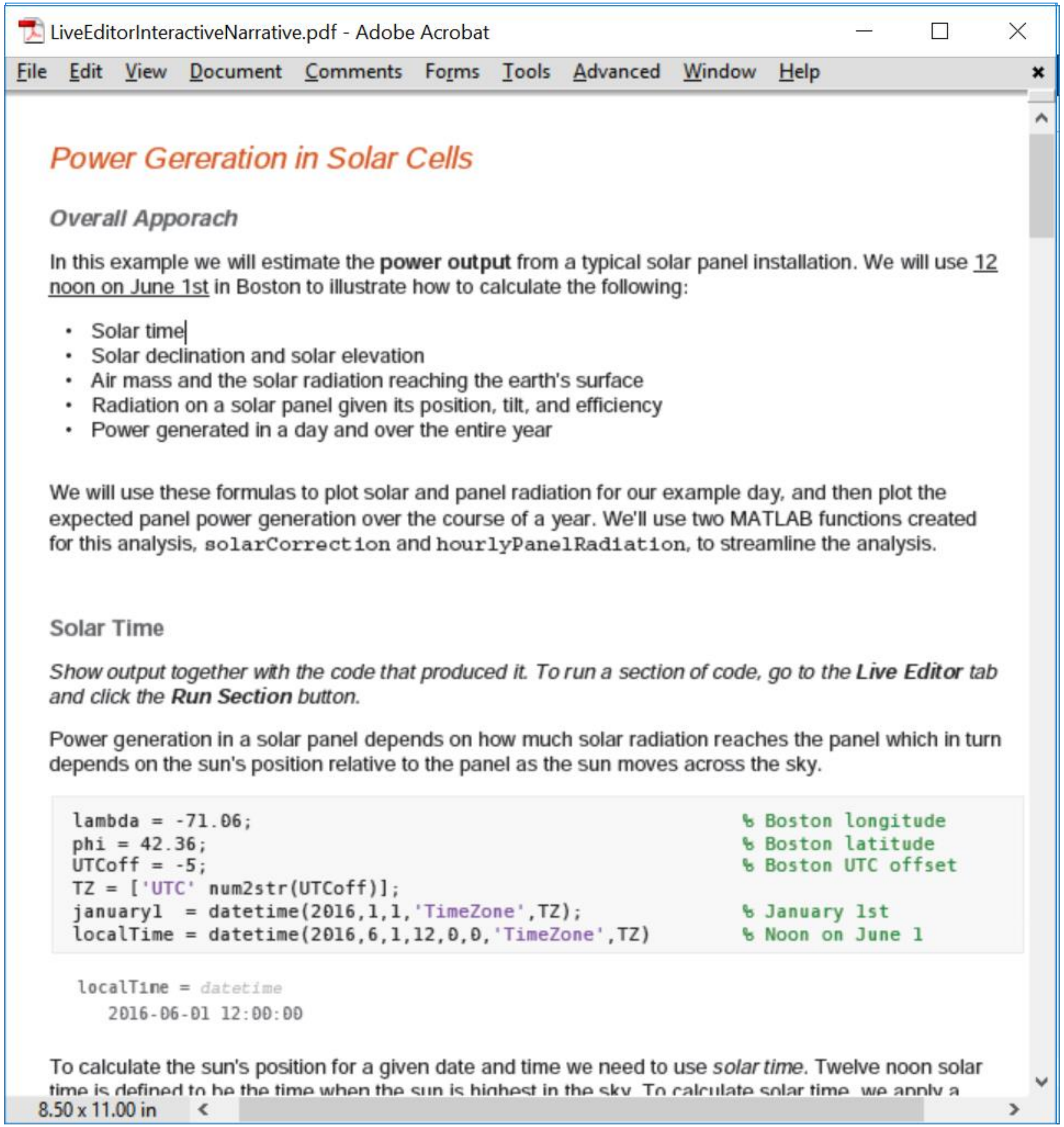
Live Editor を使えば、英語以外の言語でも分析結果を記述できます。

```
delta = asind(sind(23.45)*sind(360*(days - 81)/365));    % Solar

clf
plot(days,delta)
axis([1 365 -30 30])
title('Solar Declination')
xlabel('Day of Year')
ylabel('Degrees')
```

Estimating Sunrise and Sunset

Key features till R2017b

- ✓ Write, execute, and test code in a single interactive environment
- ✓ Generate results and graphics in the Live Editor alongside the code that produced them
- ✓ Find errors at the location in the file where they occur
- ✓ Suggests corrections for mistyped commands and variables
- ✓ Edit a figure interactively
- ✓ Add images, and hyperlinks as supporting material
- ✓ Export report in pdf, html, LaTeX fomat



Power Gereration in Solar Cells

Overall Apporach

In this example we will estimate the **power output** from a typical solar panel installation. We will use 12 noon on June 1st in Boston to illustrate how to calculate the following:

- Solar time
- Solar declination and solar elevation
- Air mass and the solar radiation reaching the earth's surface
- Radiation on a solar panel given its position, tilt, and efficiency
- Power generated in a day and over the entire year

We will use these formulas to plot solar and panel radiation for our example day, and then plot the expected panel power generation over the course of a year. We'll use two MATLAB functions created for this analysis, `solarCorrection` and `hourlyPanelRadiation`, to streamline the analysis.

Solar Time

*Show output together with the code that produced it. To run a section of code, go to the **Live Editor** tab and click the **Run Section** button.*

Power generation in a solar panel depends on how much solar radiation reaches the panel which in turn depends on the sun's position relative to the panel as the sun moves across the sky.

```
lambda = -71.06;                                              % Boston longitude
phi = 42.36;                                                  % Boston latitude
UTCoff = -5;                                                  % Boston UTC offset
TZ = ['UTC' num2str(UTCoff)];
january1  = datetime(2016,1,1,'TimeZone',TZ);                 % January 1st
localTime = datetime(2016,6,1,12,0,0,'TimeZone',TZ)           % Noon on June 1
```

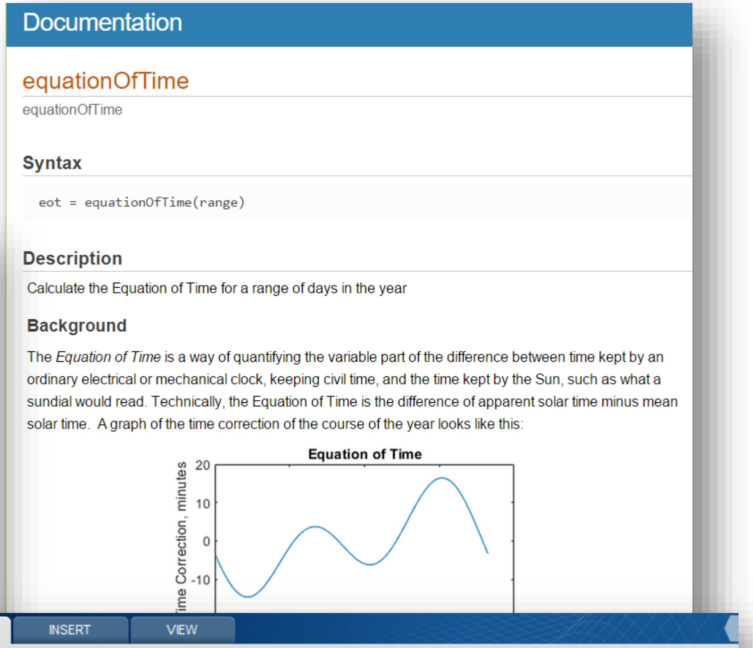
```
localTime = datetime
   2016-06-01 12:00:00
```

To calculate the sun's position for a given date and time we need to use *solar time*. Twelve noon solar time is defined to be the time when the sun is highest in the sky. To calculate solar time, we apply a

MathWorks®

Key features of Live Editor in R2018a

- ✓ Create functions with formatted documentation
  - ✓ Use the Help Browser to view function documentation
- ✓ Debug functions and scripts
  - ✓ Run to here
  - ✓ Set breakpoints
  - ✓ Step into functions
- ✓ Use interactive controls to control values
  - ✓ Sliders and combo boxes
  - ✓ Easy insertion of annotation to figure



equationOfTime

equationOfTime

Syntax

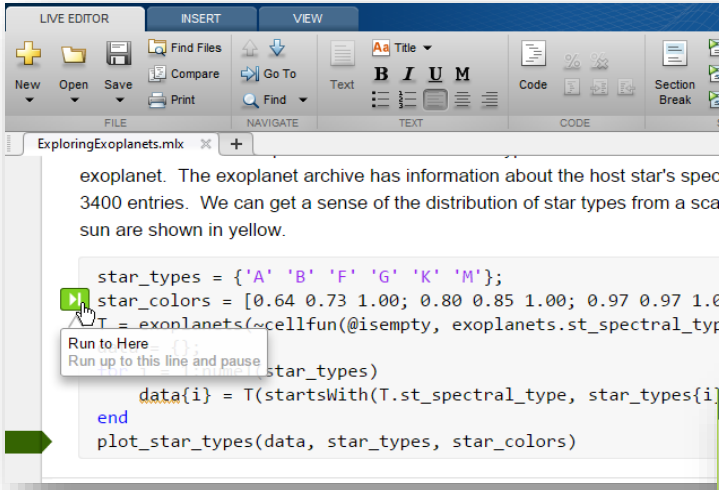
```
eot = equationOfTime(range)
```

Description

Calculate the Equation of Time for a range of days in the year

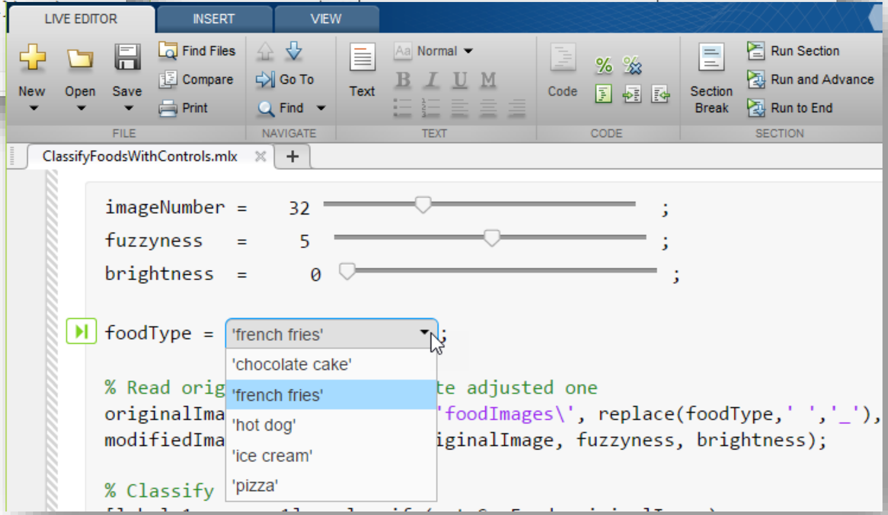
Background

The *Equation of Time* is a way of quantifying the variable part of the difference between time kept by an ordinary electrical or mechanical clock, keeping civil time, and the time kept by the Sun, such as what a sundial would read. Technically, the Equation of Time is the difference of apparent solar time minus mean solar time. A graph of the time correction of the course of the year looks like this:



exoplanet. The exoplanet archive has information about the host star's spectral type for ab
3400 entries. We can get a sense of the distribution of star types from a scatter plot. G-ty
sun are shown in yellow.

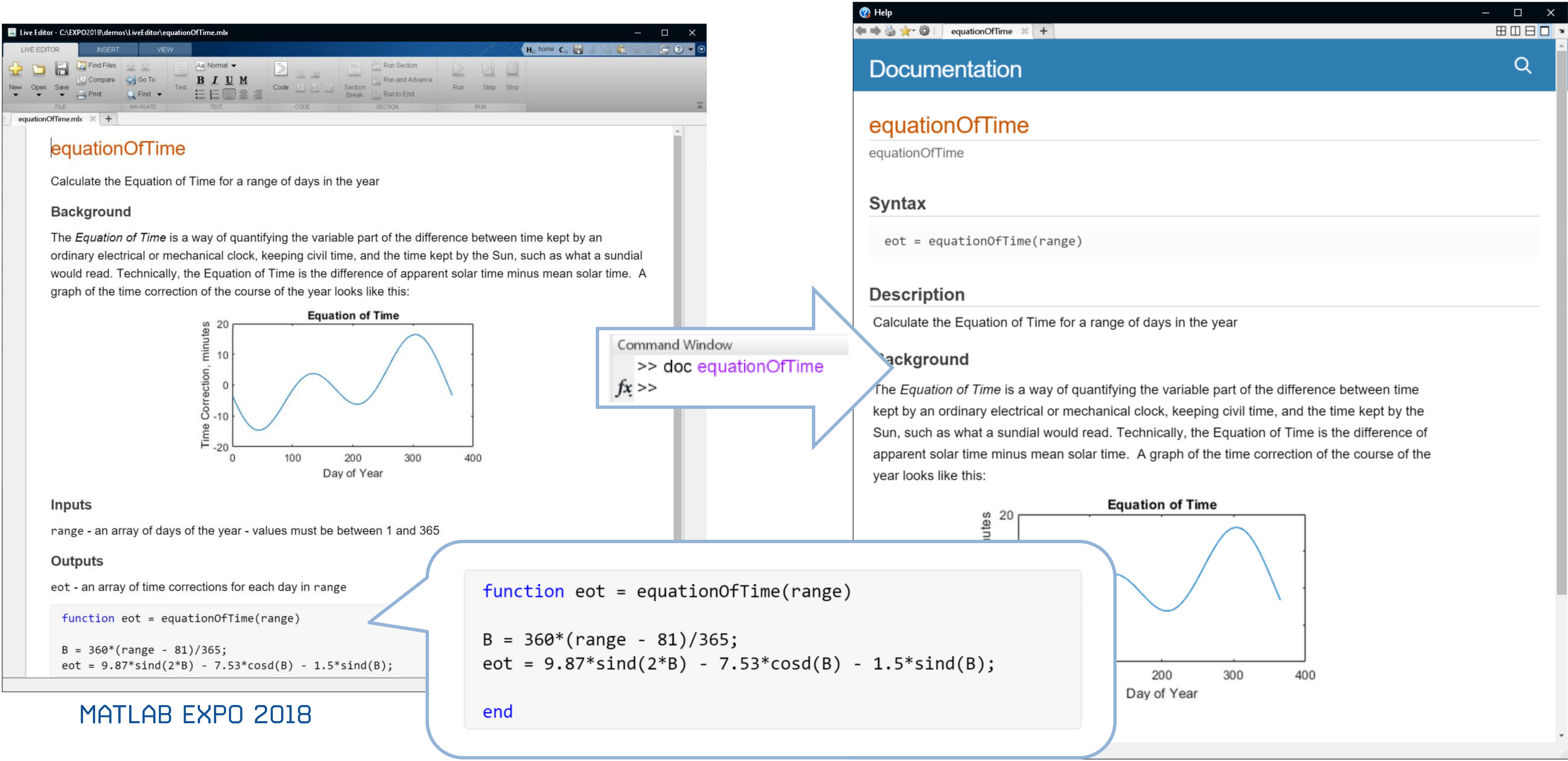
```
star_types = {'A' 'B' 'F' 'G' 'K' 'M'};
star_colors = [0.64 0.73 1.00; 0.80 0.85 1.00; 0.97 0.97 1.00; 1.00 1.0
T = exoplanets(~cellfun(@isempty, exoplanets.st_spectral_type),:);
                         star_types)
    data{i} = T(startsWith(T.st_spectral_type, star_types{i]
end
plot_star_types(data, star_types, star_colors)
```



```
imageNumber =    32  ;
fuzzyness   =     5  ;
brightness  =     0  ;

foodType = 'french fries' ;
           'chocolate cake'
% Read orig'french fries'  te adjusted one
originalIma'hot dog'       'foodImages\', replace(foodType,' ','_'),
modifiedIma'ice cream'     iginalImage, fuzzyness, brightness);
           'pizza'
% Classify
```

MATLAB EXPO 2018

Create functions with formatted documentation



equationOfTime

Calculate the Equation of Time for a range of days in the year

Background

The *Equation of Time* is a way of quantifying the variable part of the difference between time kept by an ordinary electrical or mechanical clock, keeping civil time, and the time kept by the Sun, such as what a sundial would read. Technically, the Equation of Time is the difference of apparent solar time minus mean solar time. A graph of the time correction of the course of the year looks like this:

Inputs

`range` - an array of days of the year - values must be between 1 and 365

Outputs

`eot` - an array of time corrections for each day in `range`

```
function eot = equationOfTime(range)

B = 360*(range - 81)/365;
eot = 9.87*sind(2*B) - 7.53*cosd(B) - 1.5*sind(B);
```

Command Window

```
>> doc equationOfTime
fx >>
```

equationOfTime

equationOfTime

Syntax

```
eot = equationOfTime(range)
```

Description

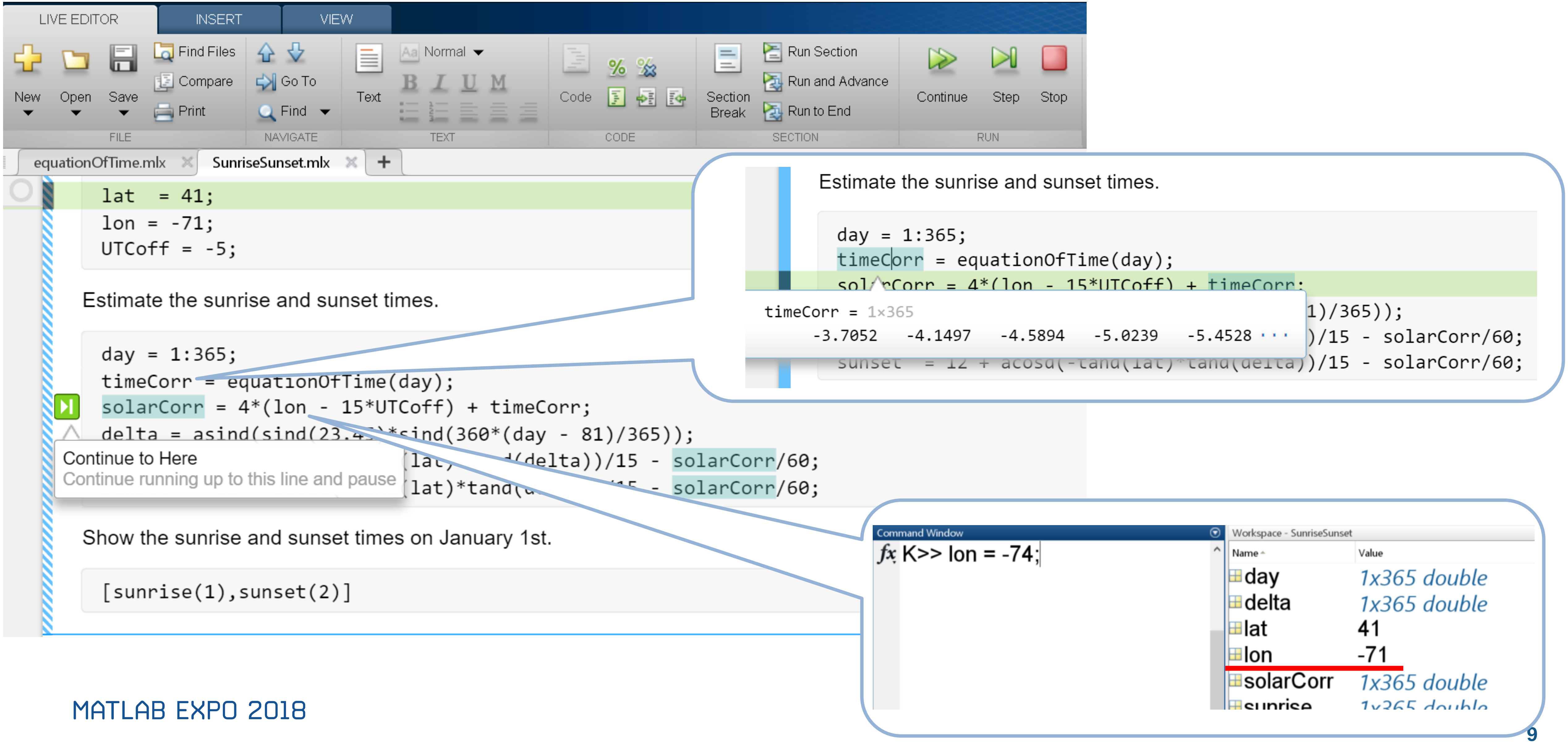
Calculate the Equation of Time for a range of days in the year

Background

The *Equation of Time* is a way of quantifying the variable part of the difference between time kept by an ordinary electrical or mechanical clock, keeping civil time, and the time kept by the Sun, such as what a sundial would read. Technically, the Equation of Time is the difference of apparent solar time minus mean solar time. A graph of the time correction of the course of the year looks like this:

```
function eot = equationOfTime(range)

B = 360*(range - 81)/365;
eot = 9.87*sind(2*B) - 7.53*cosd(B) - 1.5*sind(B);

end
```

Debug function and script in live editor



```
lat  = 41;
lon = -71;
UTCoff = -5;
```

Estimate the sunrise and sunset times.

```
day = 1:365;
timeCorr = equationOfTime(day);
solarCorr = 4*(lon - 15*UTCoff) + timeCorr;
delta = asind(sind(23.4   *sind(360*(day - 81)/365));
                     (lat)     (delta))/15 - solarCorr/60;
                     (lat)*tand(               - solarCorr/60;
```

Continue to Here
Continue running up to this line and pause

Show the sunrise and sunset times on January 1st.

```
[sunrise(1),sunset(2)]
```

Estimate the sunrise and sunset times.

```
day = 1:365;
timeCorr = equationOfTime(day);
sol rCorr = 4*(lon - 15*UTCoff) + timeCorr;
                                                1)/365));
                                                )/15 - solarCorr/60;
sunset  = 12 + acosd(-tand(lat)*tand(delta))/15 - solarCorr/60;
```

```
timeCorr = 1×365
     -3.7052    -4.1497    -4.5894    -5.0239    -5.4528 ···
```

Command Window

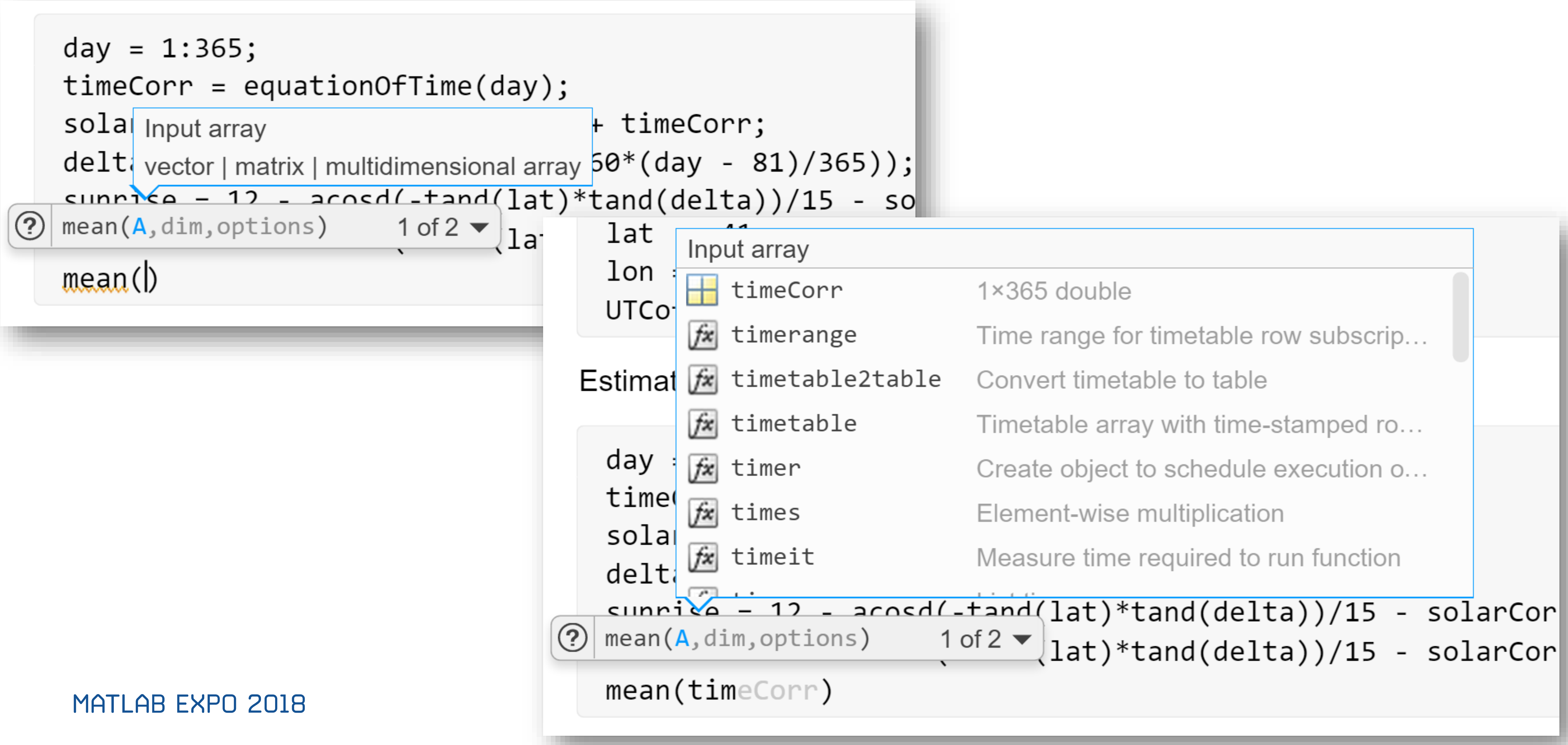
```
K>> lon = -74;
```

Workspace - SunriseSunset

Name	Value
day	1x365 double
delta	1x365 double
lat	41
lon	-71
solarCorr	1x365 double
sunrise	1x365 double

Help function and variable’s Contextual hints

R2018a



```
day = 1:365;
timeCorr = equationOfTime(day);
sola                                          + timeCorr;
delt                                          60*(day - 81)/365));
sunrise = 12 - acosd(-tand(lat)*tand(delta))/15 - so
                                              (la
mean(|)
```

Input array

vector | matrix | multidimensional array

? mean(A,dim,options) 1 of 2 ▾

```
lat
lon
UTCo

Estimat

day
time
sola
delt
sunrise = 12 - acosd(-tand(lat)*tand(delta))/15 - solarCor
                                 lat)*tand(delta))/15 - solarCor
mean(timeCorr)
```

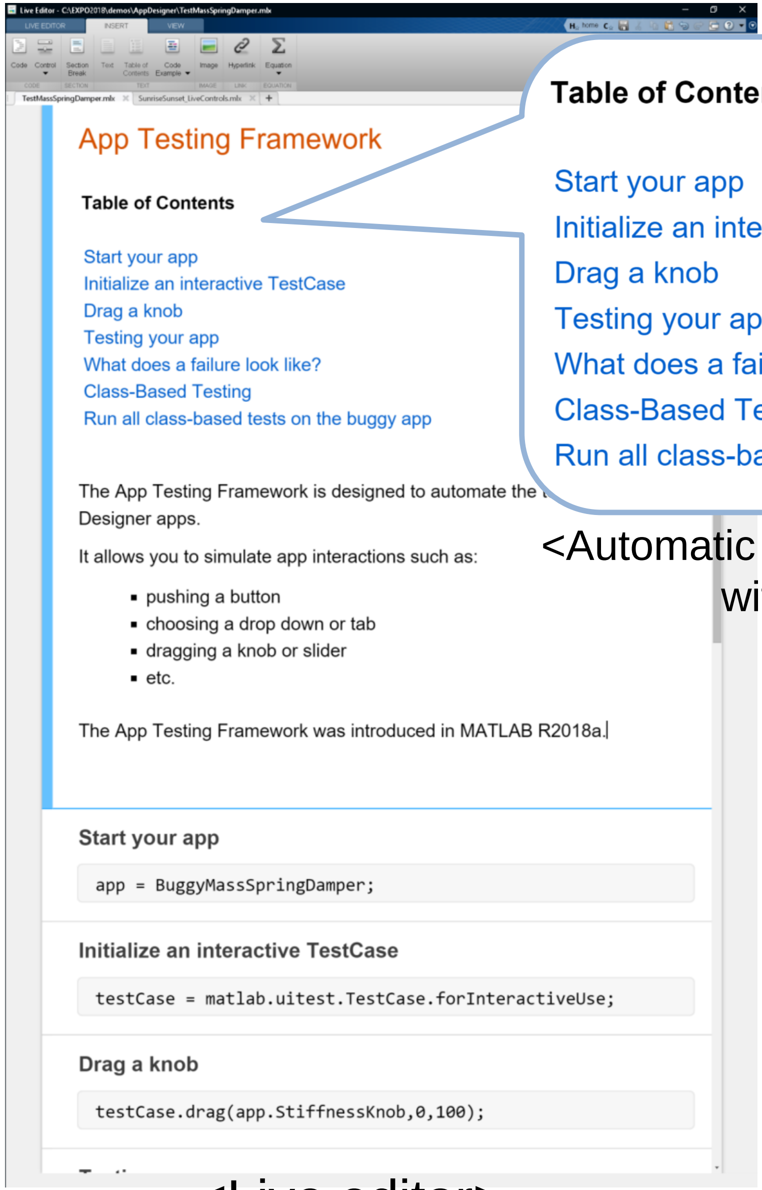
Input array

▦	timeCorr	1×365 double
fx	timerange	Time range for timetable row subscrip…
fx	timetable2table	Convert timetable to table
fx	timetable	Timetable array with time-stamped ro…
fx	timer	Create object to schedule execution o…
fx	times	Element-wise multiplication
fx	timeit	Measure time required to run function

? mean(A,dim,options) 1 of 2 ▾

MATLAB EXPO 2018

Report generation to pdf, html, and LaTeX



App Testing Framework

Table of Contents

The App Testing Framework is designed to automate the Designer apps.

It allows you to simulate app interactions such as:

- pushing a button
- choosing a drop down or tab
- dragging a knob or slider
- etc.

The App Testing Framework was introduced in MATLAB R2018a.

Start your app

```
app = BuggyMassSpringDamper;
```

Initialize an interactive TestCase

```
testCase = matlab.uitest.TestCase.forInteractiveUse;
```

Drag a knob

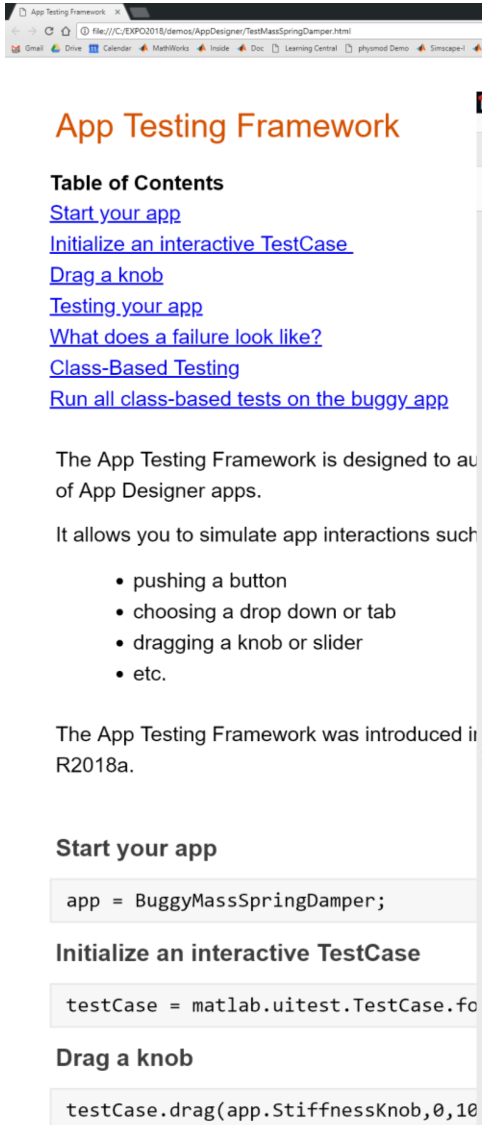
```
testCase.drag(app.StiffnessKnob,0,100);
```

Table of Contents

<Automatic contents generation> with section title

<Live editor>

Export to pdf, html, LaTeX



App Testing Framework

Table of Contents

The App Testing Framework is designed to au of App Designer apps.

It allows you to simulate app interactions such

- pushing a button
- choosing a drop down or tab
- dragging a knob or slider
- etc.

The App Testing Framework was introduced i R2018a.

Start your app

```
app = BuggyMassSpringDamper;
```

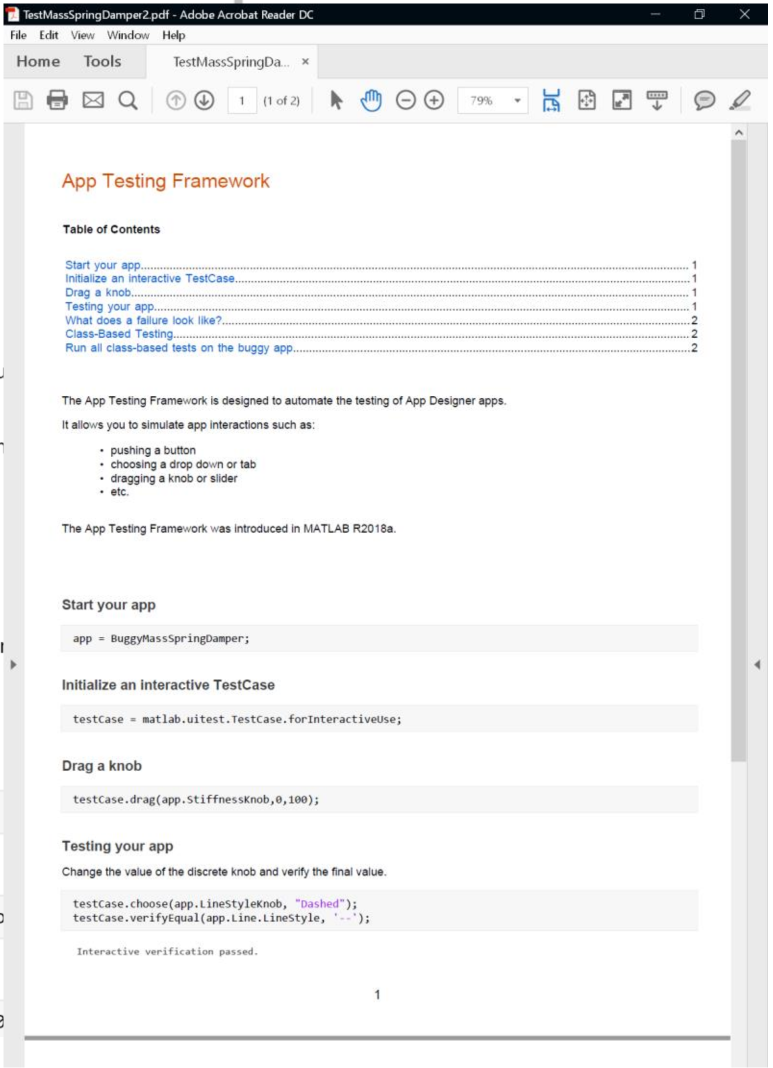
Initialize an interactive TestCase

```
testCase = matlab.uitest.TestCase.fo
```

Drag a knob

```
testCase.drag(app.StiffnessKnob,0,10
```

<html for web sharing>



TestMassSpringDamper2.pdf - Adobe Acrobat Reader DC

App Testing Framework

Table of Contents

The App Testing Framework is designed to automate the testing of App Designer apps.

It allows you to simulate app interactions such as:

- pushing a button
- choosing a drop down or tab
- dragging a knob or slider
- etc.

The App Testing Framework was introduced in MATLAB R2018a.

Start your app

```
app = BuggyMassSpringDamper;
```

Initialize an interactive TestCase

```
testCase = matlab.uitest.TestCase.forInteractiveUse;
```

Drag a knob

```
testCase.drag(app.StiffnessKnob,0,100);
```

Testing your app

Change the value of the discrete knob and verify the final value.

```
testCase.choose(app.LineStyleKnob, "Dashed");
testCase.verifyEqual(app.Line.LineStyle, '--');
```

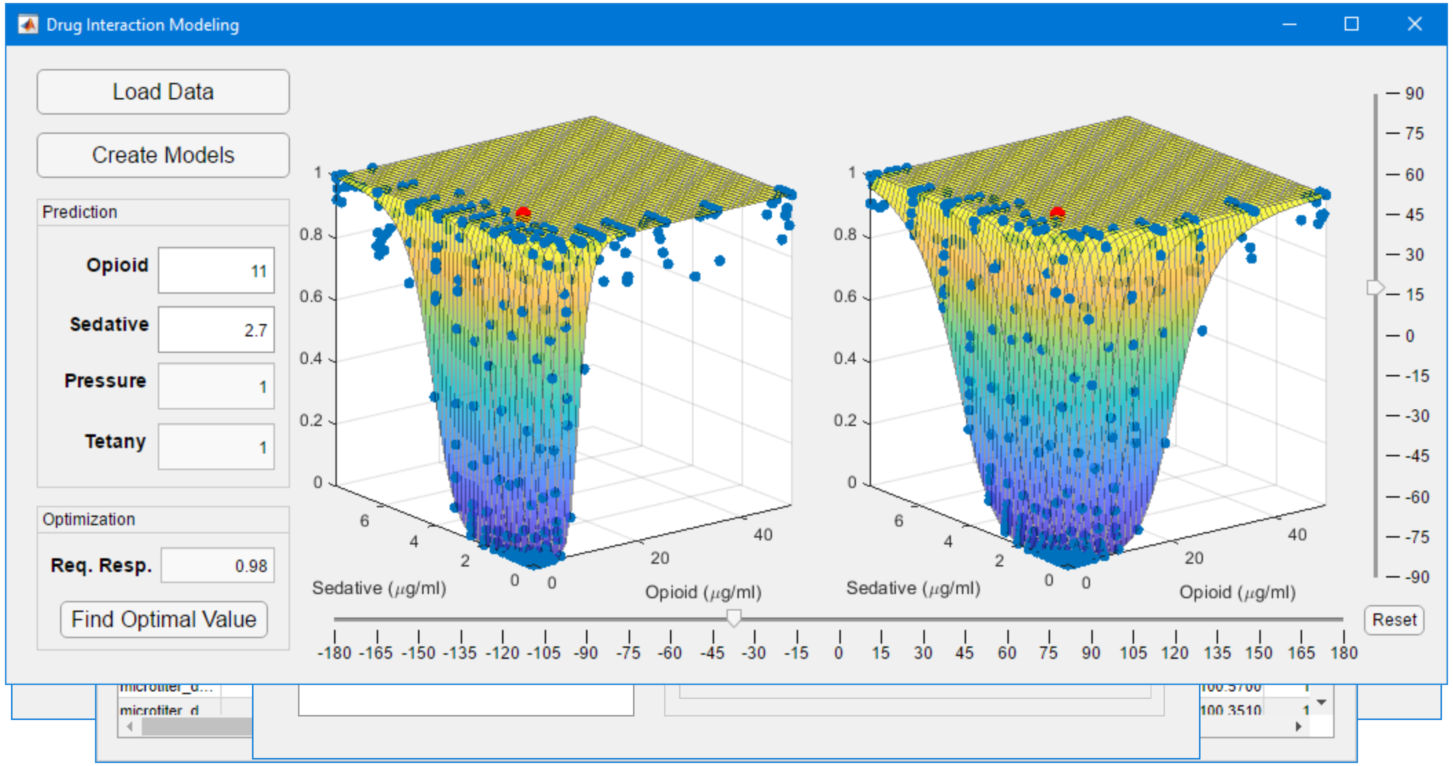
```
Interactive verification passed.
```

1

<pdf for report>

What is App Designer?

- ✓ A new environment for building MATLAB Apps
- ✓ Broad set of UI components including instrumentation controls
- ✓ Integrates the two primary tasks of app building
  - ✓ laying out visual components
  - ✓ programming app behavior
- ✓ Generates code as a MATLAB class



Drug Interaction Modeling

Prediction

Opioid: 11

Sedative: 2.7

Pressure: 1

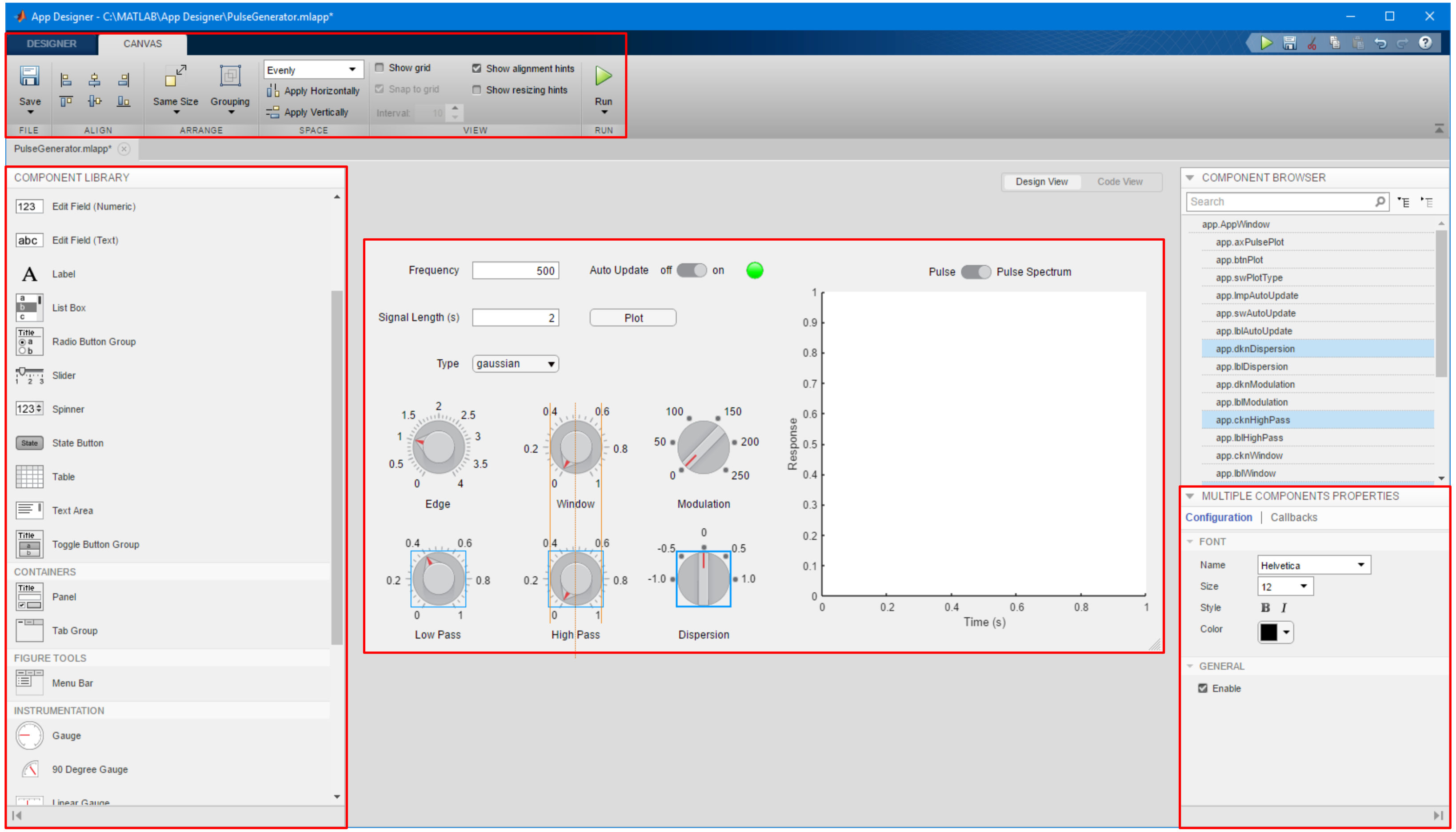
Tetany: 1

Optimization

Req. Resp.: 0.98

The App Designer Desktop – Design View

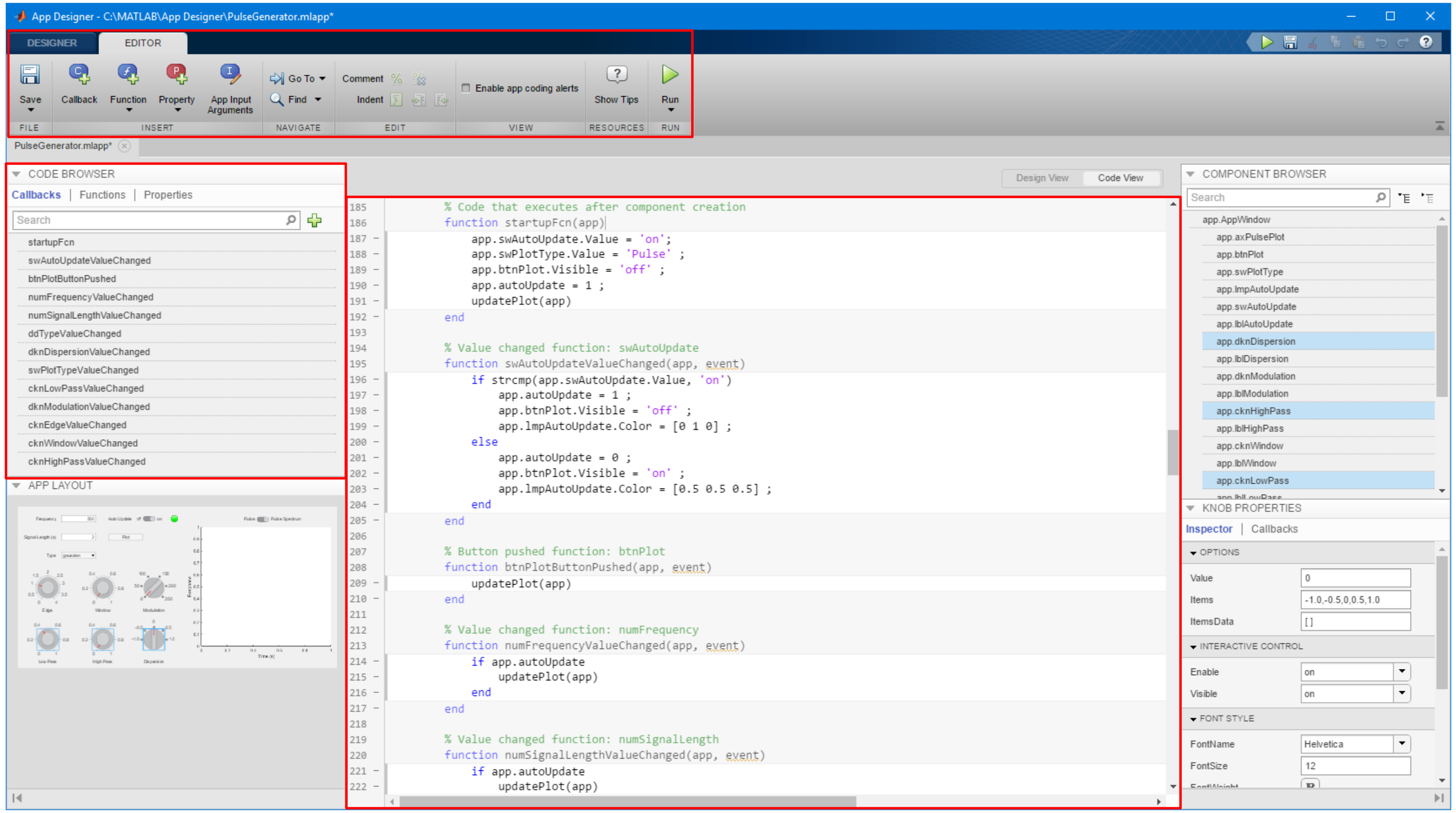
- ✓ Design and layout the app's interface
- ✓ Component Library
  - ✓ Select components and add them to the canvas
- ✓ Design Canvas
  - ✓ Layout components
- ✓ Toolstrip
  - ✓ Align, space, and group components
- ✓ Properties panel
  - ✓ Set common component properties



MATLAB EXPO 2018

The App Designer Desktop – Code View

- ✓ Write code to control the app's behavior
- ✓ Editor
  - ✓ Write code for callbacks and other functions
- ✓ Code Browser
  - ✓ Navigate to callbacks and app properties
- ✓ Toolstrip
  - ✓ Add new code elements – properties, callbacks, and functions

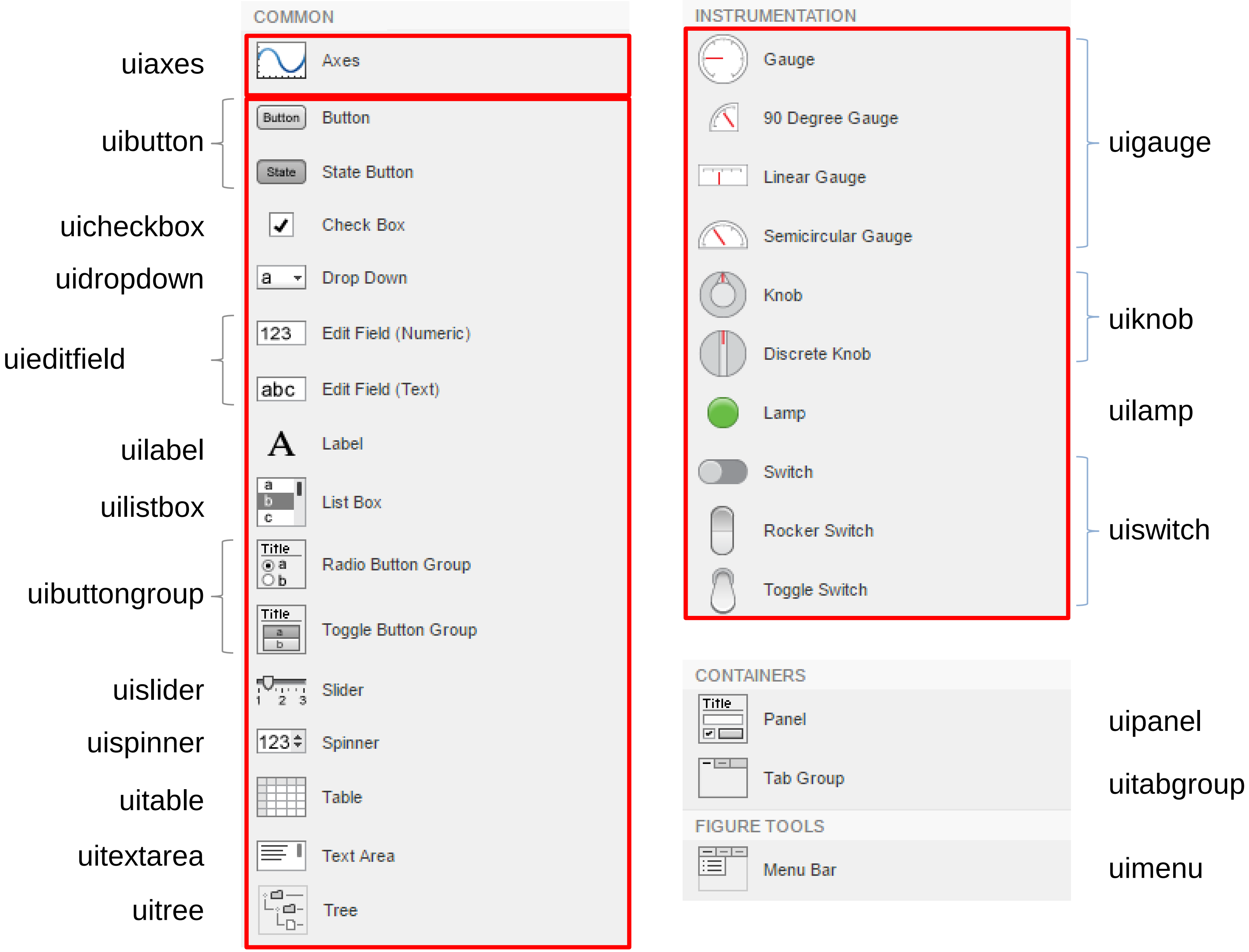


```
        % Code that executes after component creation
        function startupFcn(app)
            app.swAutoUpdate.Value = 'on';
            app.swPlotType.Value = 'Pulse' ;
            app.btnPlot.Visible = 'off' ;
            app.autoUpdate = 1 ;
            updatePlot(app)
        end

        % Value changed function: swAutoUpdate
        function swAutoUpdateValueChanged(app, event)
            if strcmp(app.swAutoUpdate.Value, 'on')
                app.autoUpdate = 1 ;
                app.btnPlot.Visible = 'off' ;
                app.lmpAutoUpdate.Color = [0 1 0] ;
            else
                app.autoUpdate = 0 ;
                app.btnPlot.Visible = 'on' ;
                app.lmpAutoUpdate.Color = [0.5 0.5 0.5] ;
            end
        end

        % Button pushed function: btnPlot
        function btnPlotButtonPushed(app, event)
            updatePlot(app)
        end

        % Value changed function: numFrequency
        function numFrequencyValueChanged(app, event)
            if app.autoUpdate
                updatePlot(app)
            end
        end

        % Value changed function: numSignalLength
        function numSignalLengthValueChanged(app, event)
            if app.autoUpdate
                updatePlot(app)
```

Value: 0
Items: -1.0,-0.5,0,0.5,1.0
ItemsData: [ ]
Enable: on
Visible: on
FontName: Helvetica
FontSize: 12

App Designer Components

- ✓ New javascript `uifigure` component
- ✓ New `uiaxes` component for web graphics
- ✓ Expanded set of standard components
- ✓ New instrumentation components
- ✓ Other components being considered
  - ✓ file picker
  - ✓ date picker
  - ✓ toolbar



COMMON

uiaxes	Axes
uibutton	Button State Button
uicheckbox	Check Box
uidropdown	Drop Down
uieditfield	Edit Field (Numeric) Edit Field (Text)
uilabel	Label
uilistbox	List Box
uibuttongroup	Radio Button Group Toggle Button Group
uislider	Slider
uispinner	Spinner
uitable	Table
uitextarea	Text Area
uitree	Tree

INSTRUMENTATION

Gauge 90 Degree Gauge Linear Gauge Semicircular Gauge	uigauge
Knob Discrete Knob	uiknob
Lamp	uilamp
Switch Rocker Switch Toggle Switch	uiswitch

CONTAINERS

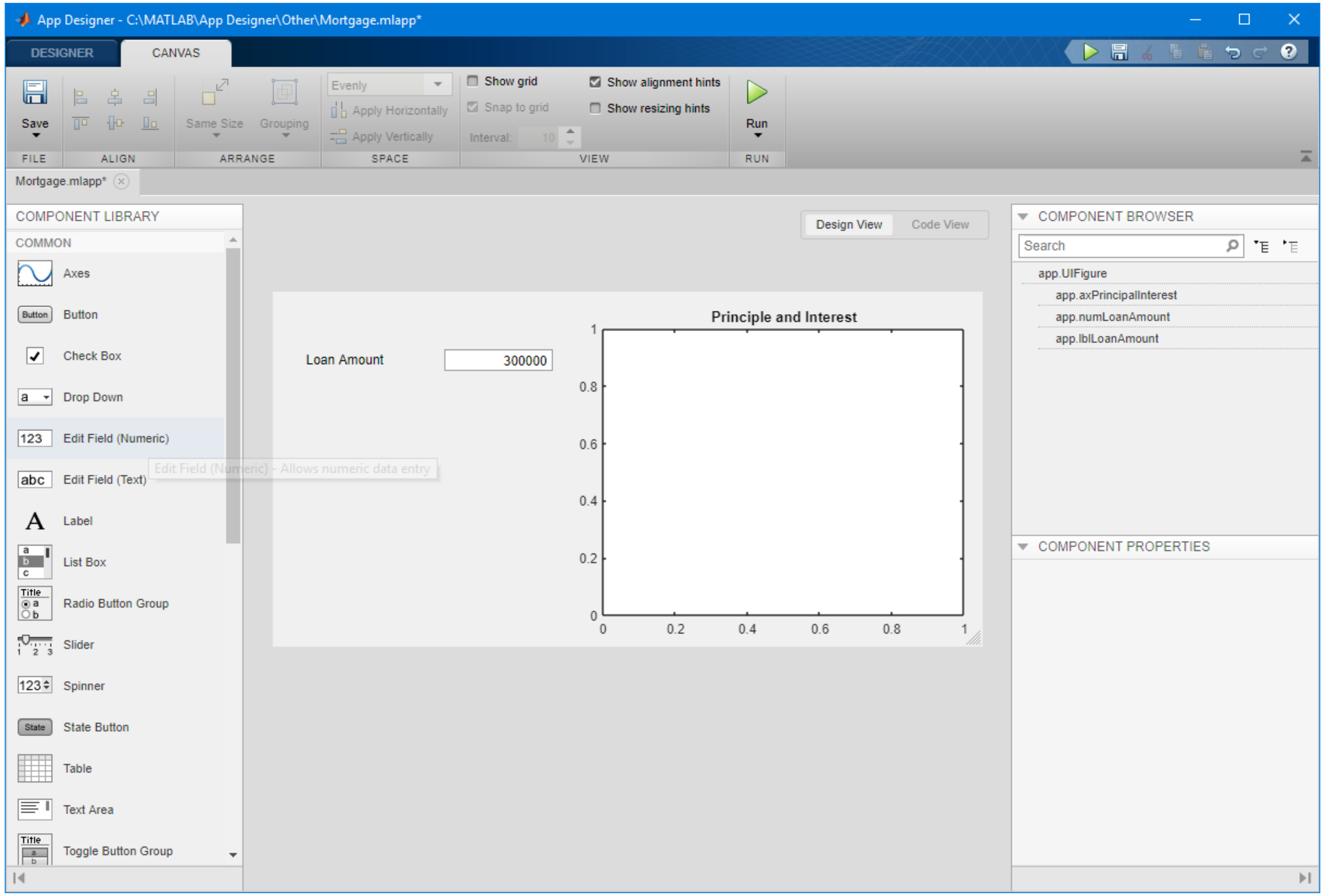
Panel	uipanel
Tab Group	uitabgroup

FIGURE TOOLS

Menu Bar	uimenu

Design and Layout – Basic Steps

- ✓ Select a component from the library and drag it to the canvas
- ✓ Name the component
- ✓ Set the component properties
- ✓ Position manually or align with other components


App Designer - C:\MATLAB\App Designer\Other\Mortgage.mlapp*

☐ Show grid
☑ Snap to grid
Interval: 10
☑ Show alignment hints
☐ Show resizing hints

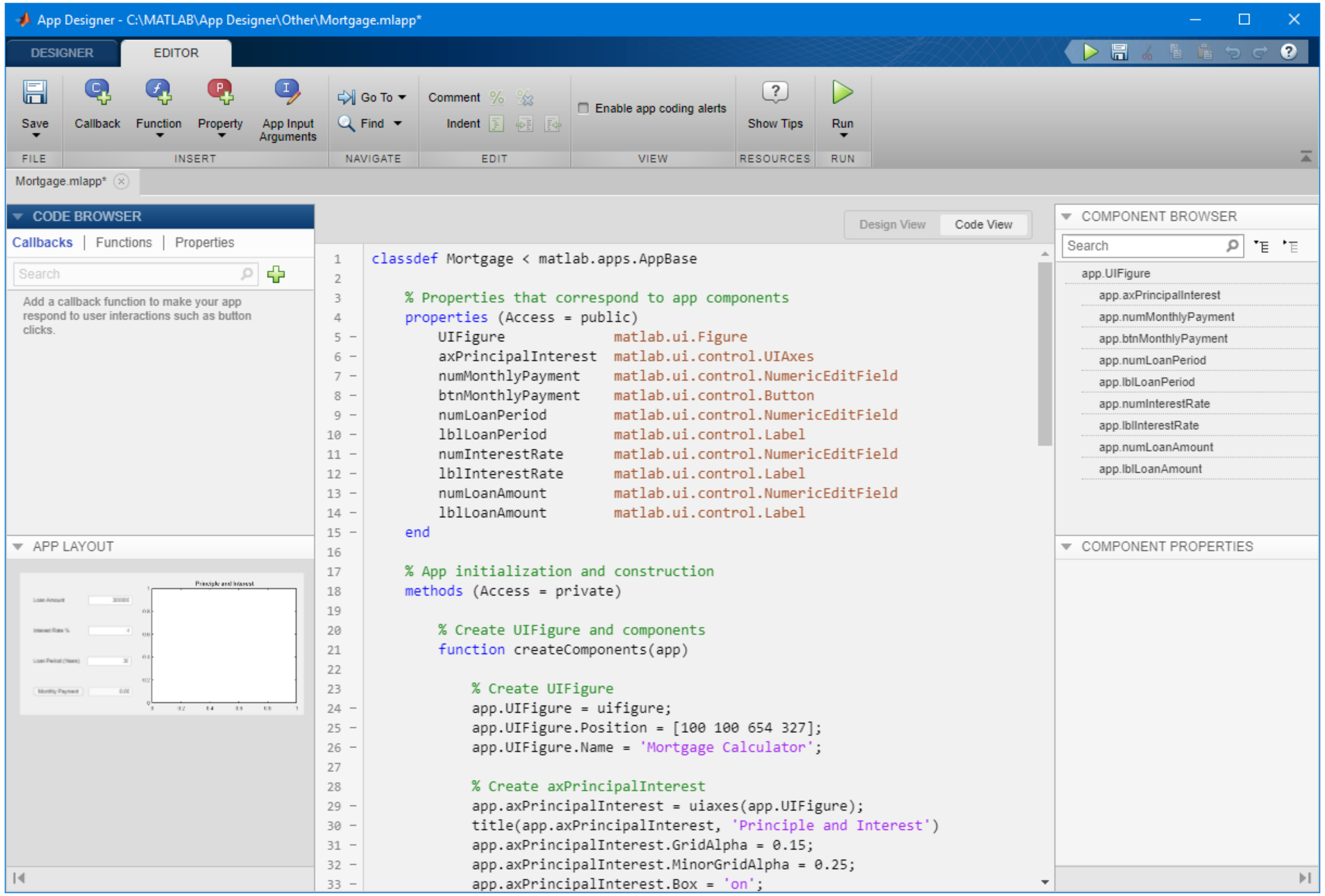
Loan Amount 300000

Edit Field (Numeric) - Allows numeric data entry

app.UIFigure
app.axPrincipalInterest
app.numLoanAmount
app.lblLoanAmount

Coding App Behavior – Basic Steps

- ✓ Select a component
- ✓ Create a callback
- ✓ Add callback code
- ✓ Use hints to avoid common programming errors

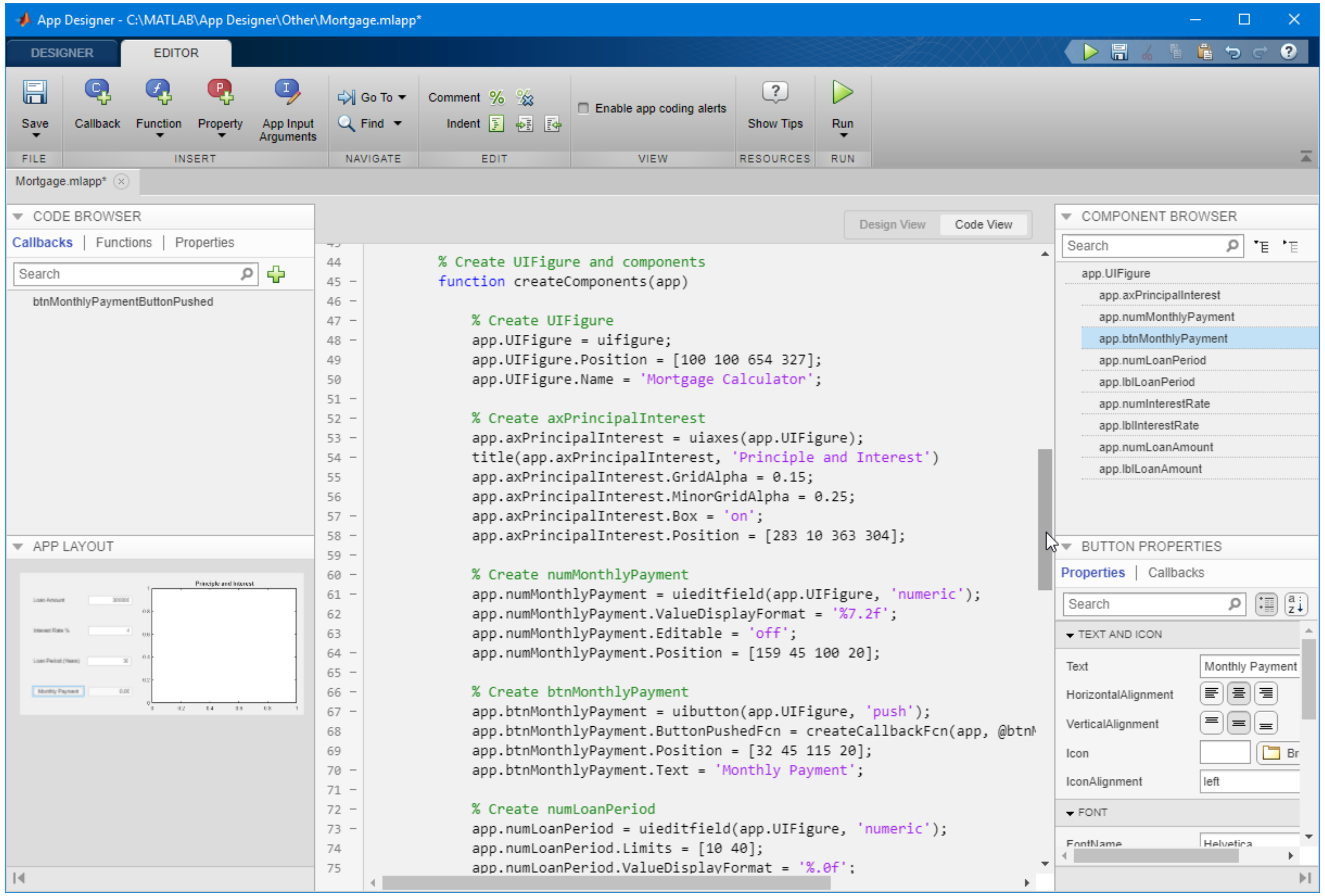


Add a callback function to make your app respond to user interactions such as button clicks.

```matlab
classdef Mortgage < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                 matlab.ui.Figure
        axPrincipalInterest  matlab.ui.control.UIAxes
        numMonthlyPayment    matlab.ui.control.NumericEditField
        btnMonthlyPayment    matlab.ui.control.Button
        numLoanPeriod        matlab.ui.control.NumericEditField
        lblLoanPeriod        matlab.ui.control.Label
        numInterestRate      matlab.ui.control.NumericEditField
        lblInterestRate      matlab.ui.control.Label
        numLoanAmount        matlab.ui.control.NumericEditField
        lblLoanAmount        matlab.ui.control.Label
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 654 327];
            app.UIFigure.Name = 'Mortgage Calculator';

            % Create axPrincipalInterest
            app.axPrincipalInterest = uiaxes(app.UIFigure);
            title(app.axPrincipalInterest, 'Principle and Interest')
            app.axPrincipalInterest.GridAlpha = 0.15;
            app.axPrincipalInterest.MinorGridAlpha = 0.25;
            app.axPrincipalInterest.Box = 'on';
```

App Code Structure

- ✓ Code created for the App is a MATLAB class
- ✓ Controls and shared data are stored as properties of the class
- ✓ Callbacks and helper functions are stored as methods of the class
- ✓ App Designer generates the code for all the app components



App Designer - C:\MATLAB\App Designer\Other\Mortgage.mlapp*

```
        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 654 327];
            app.UIFigure.Name = 'Mortgage Calculator';

            % Create axPrincipalInterest
            app.axPrincipalInterest = uiaxes(app.UIFigure);
            title(app.axPrincipalInterest, 'Principle and Interest')
            app.axPrincipalInterest.GridAlpha = 0.15;
            app.axPrincipalInterest.MinorGridAlpha = 0.25;
            app.axPrincipalInterest.Box = 'on';
            app.axPrincipalInterest.Position = [283 10 363 304];

            % Create numMonthlyPayment
            app.numMonthlyPayment = uieditfield(app.UIFigure, 'numeric');
            app.numMonthlyPayment.ValueDisplayFormat = '%7.2f';
            app.numMonthlyPayment.Editable = 'off';
            app.numMonthlyPayment.Position = [159 45 100 20];

            % Create btnMonthlyPayment
            app.btnMonthlyPayment = uibutton(app.UIFigure, 'push');
            app.btnMonthlyPayment.ButtonPushedFcn = createCallbackFcn(app, @btn
            app.btnMonthlyPayment.Position = [32 45 115 20];
            app.btnMonthlyPayment.Text = 'Monthly Payment';

            % Create numLoanPeriod
            app.numLoanPeriod = uieditfield(app.UIFigure, 'numeric');
            app.numLoanPeriod.Limits = [10 40];
            app.numLoanPeriod.ValueDisplayFormat = '%.0f';
```

CODE BROWSER: Callbacks | Functions | Properties

btnMonthlyPaymentButtonPushed

COMPONENT BROWSER

- app.UIFigure
  - app.axPrincipalInterest
  - app.numMonthlyPayment
  - app.btnMonthlyPayment
  - app.numLoanPeriod
  - app.lblLoanPeriod
  - app.numInterestRate
  - app.lblInterestRate
  - app.numLoanAmount
  - app.lblLoanAmount

BUTTON PROPERTIES: Properties | Callbacks

TEXT AND ICON

Text: Monthly Payment

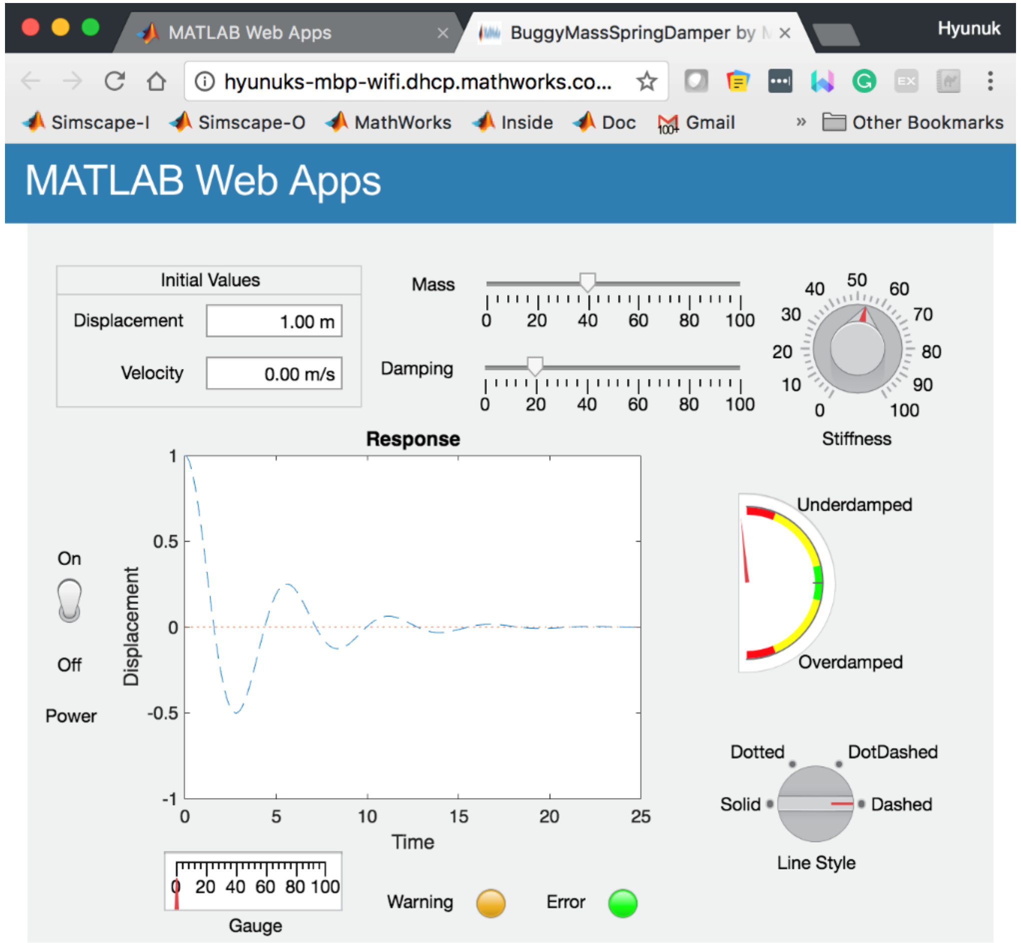
IconAlignment: left

FONT

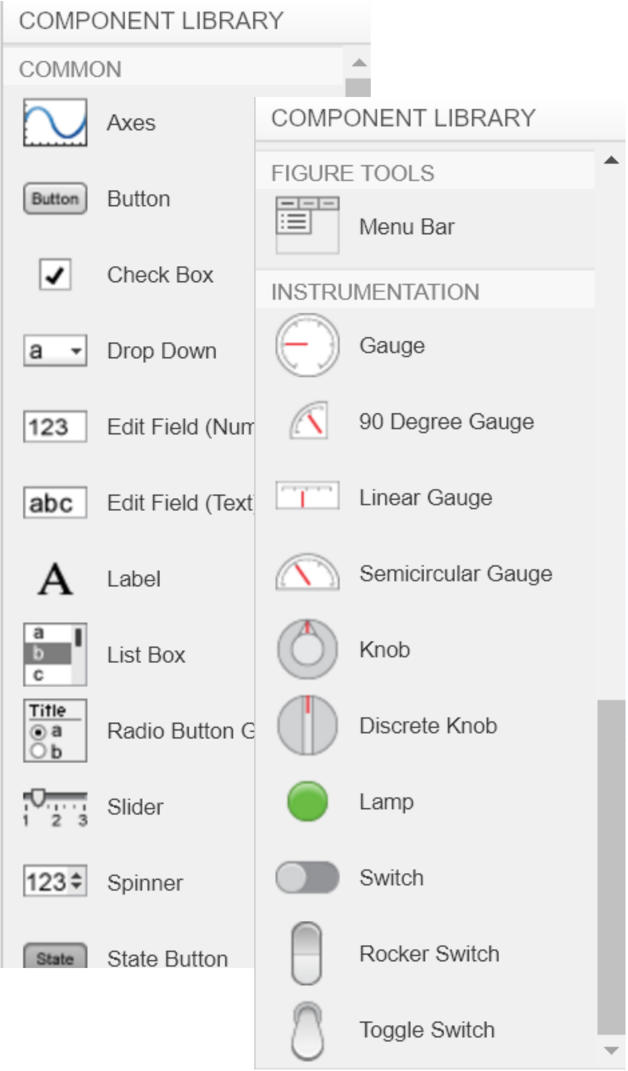
FontName: Helvetica

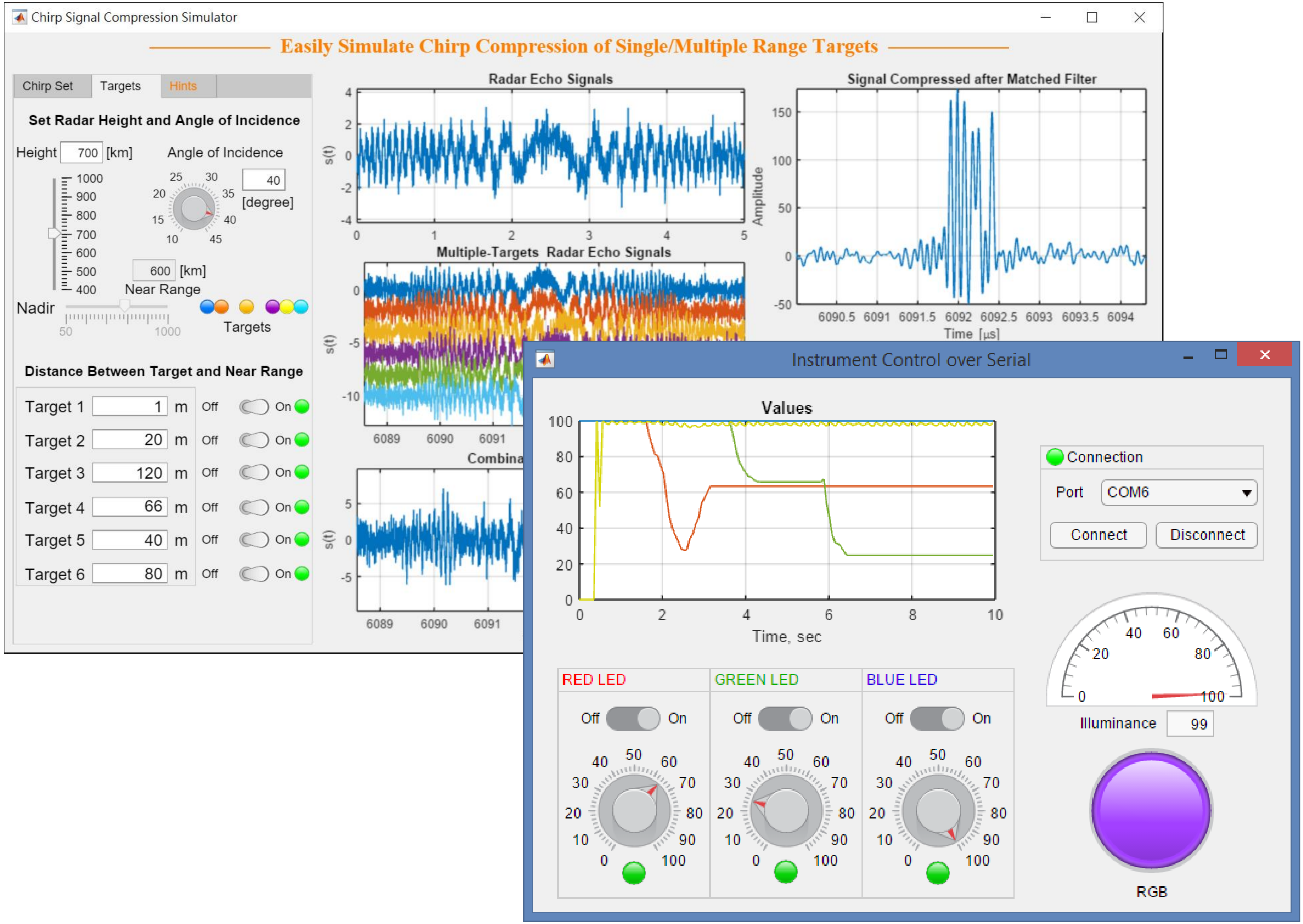
Key features of App designer in R2018a

- ✓ More HMI friendly components
- ✓ Web deploy
- ✓ GUIDE to App designer tool
- ✓ + append : Integrating with Simulink
  (Not the latest feature)



More HMI-friendly components




Chirp Signal Compression Simulator
Easily Simulate Chirp Compression of Single/Multiple Range Targets
Chirp Set
Targets
Hints
Set Radar Height and Angle of Incidence
Nadir
Targets
Distance Between Target and Near Range
Radar Echo Signals
Multiple-Targets Radar Echo Signals
Combina
Signal Compressed after Matched Filter
s(t)
Amplitude
Time [μs]
Instrument Control over Serial
Values
Time, sec
RED LED
GREEN LED
BLUE LED
Off
On
Connection
Connect
Disconnect
RGB

Height 700 [km]

Angle of Incidence 40 [degree]

600 [km] Near Range

Target 1 1 m

Target 2 20 m

Target 3 120 m

Target 4 66 m

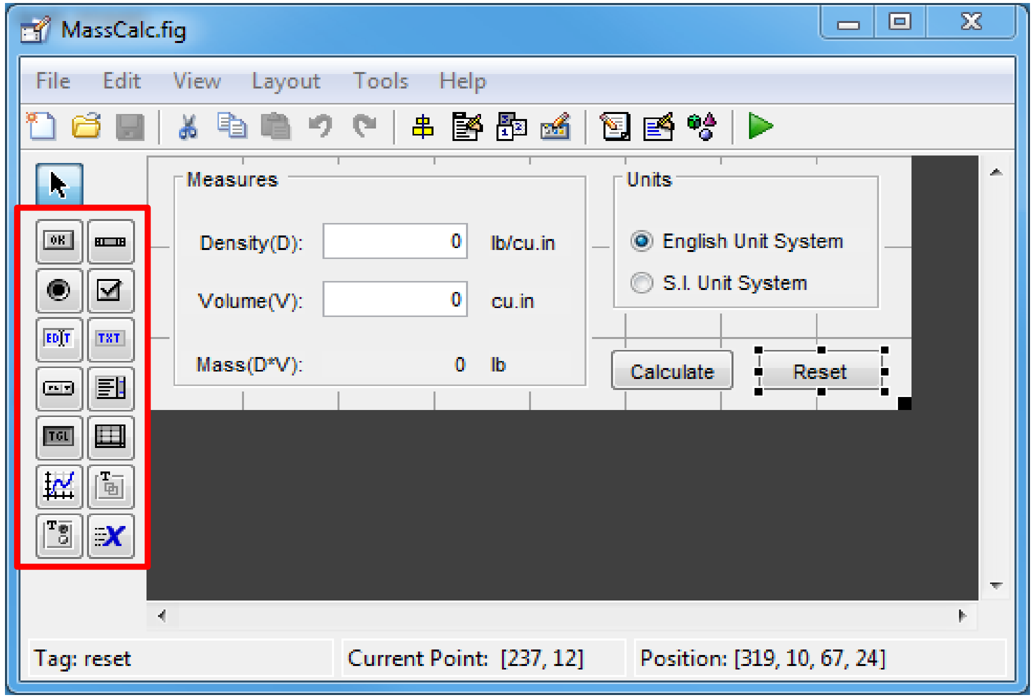
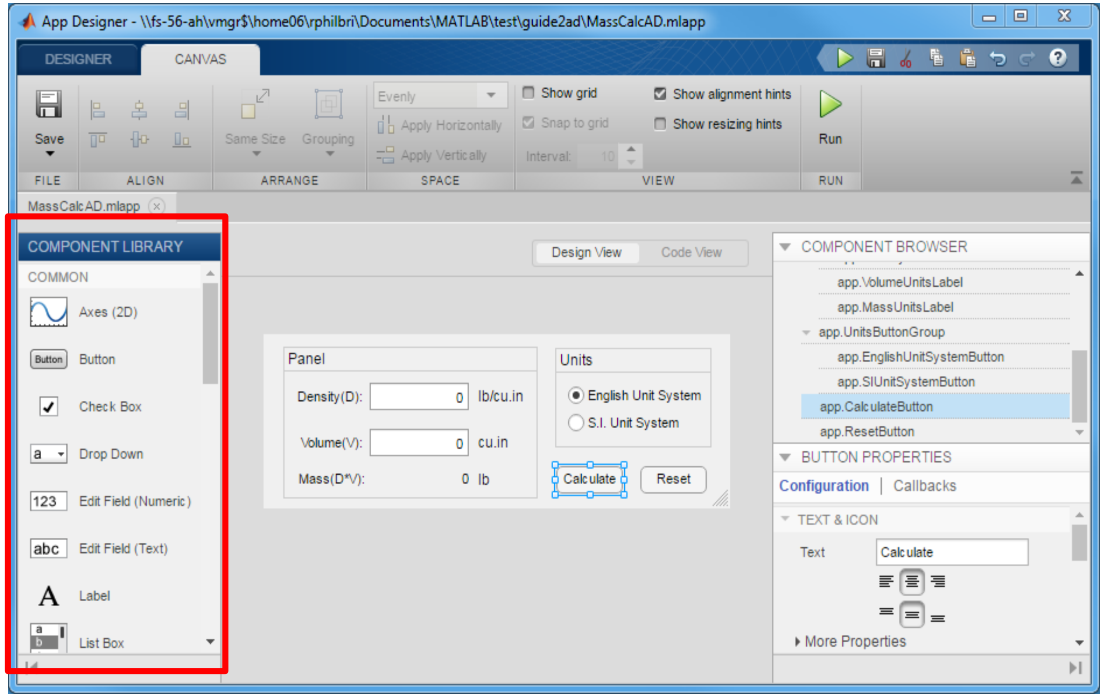
Target 5 40 m

Target 6 80 m

Port COM6

Illuminance 99

Differences between GUIDE and App Designer

GUIDE	App designer
 MassCalc.fig File Edit View Layout Tools Help Measures Density(D): 0 lb/cu.in Volume(V): 0 cu.in Mass(D*V): 0 lb Units English Unit System S.I. Unit System Calculate Reset Tag: reset Current Point: [237, 12] Position: [319, 10, 67, 24]	 App Designer - \\fs-56-ah\vmgr$\home06\rphilbri\Documents\MATLAB\test\guide2ad\MassCalcAD.mlapp DESIGNER CANVAS Save Same Size Grouping Evenly Apply Horizontally Apply Vertically Show grid Snap to grid Interval 10 Show alignment hints Show resizing hints Run FILE ALIGN ARRANGE SPACE VIEW RUN MassCalcAD.mlapp COMPONENT LIBRARY COMMON Axes (2D) Button Check Box Drop Down Edit Field (Numeric) Edit Field (Text) Label List Box Design View Code View Panel Density(D): 0 lb/cu.in Volume(V): 0 cu.in Mass(D*V): 0 lb Units English Unit System S.I. Unit System Calculate Reset COMPONENT BROWSER app.VolumeUnitsLabel app.MassUnitsLabel app.UnitsButtonGroup app.EnglishUnitSystemButton app.SIUnitSystemButton app.CalculateButton app.ResetButton BUTTON PROPERTIES Configuration Callbacks TEXT & ICON Text Calculate More Properties
Use figure functions and figure properties	Use uifigure functions and UI Figure properties
Essential components for GUI	More HMI friendly components
Standalone deployable	Standalone deployable + Web deployable
Use get, set functions	Use dot notation using class

GUIDE to App Designer Migration Tool for MATLAB

★★★★☆ 5 Ratings

107 Downloads

Updated 26 Mar 2018

version 1.0 (15.1 KB) by MathWorks App Designer Team

Use the GUIDE to App Designer Migration tool to help transition your GUIDE apps to App Designer.

Add to Watchlist | Download

Overview

App Designer is a new environment for building MATLAB apps. There are many advantages to migrating existing GUIDE apps to App Designer including:
- An improved design canvas, and a new generated code structure that makes it easier to share data across the app.
- An expanded component set with a full set of standard user interface components, new components such as a tree, date picker, and an enhanced table, as well as components to create control panels and human-machine interfaces.
- Ability to deploy to the web, so you can share your app with anyone in your organization, or run it in

MATLAB Release

MATLAB 9.4 (R2018a)

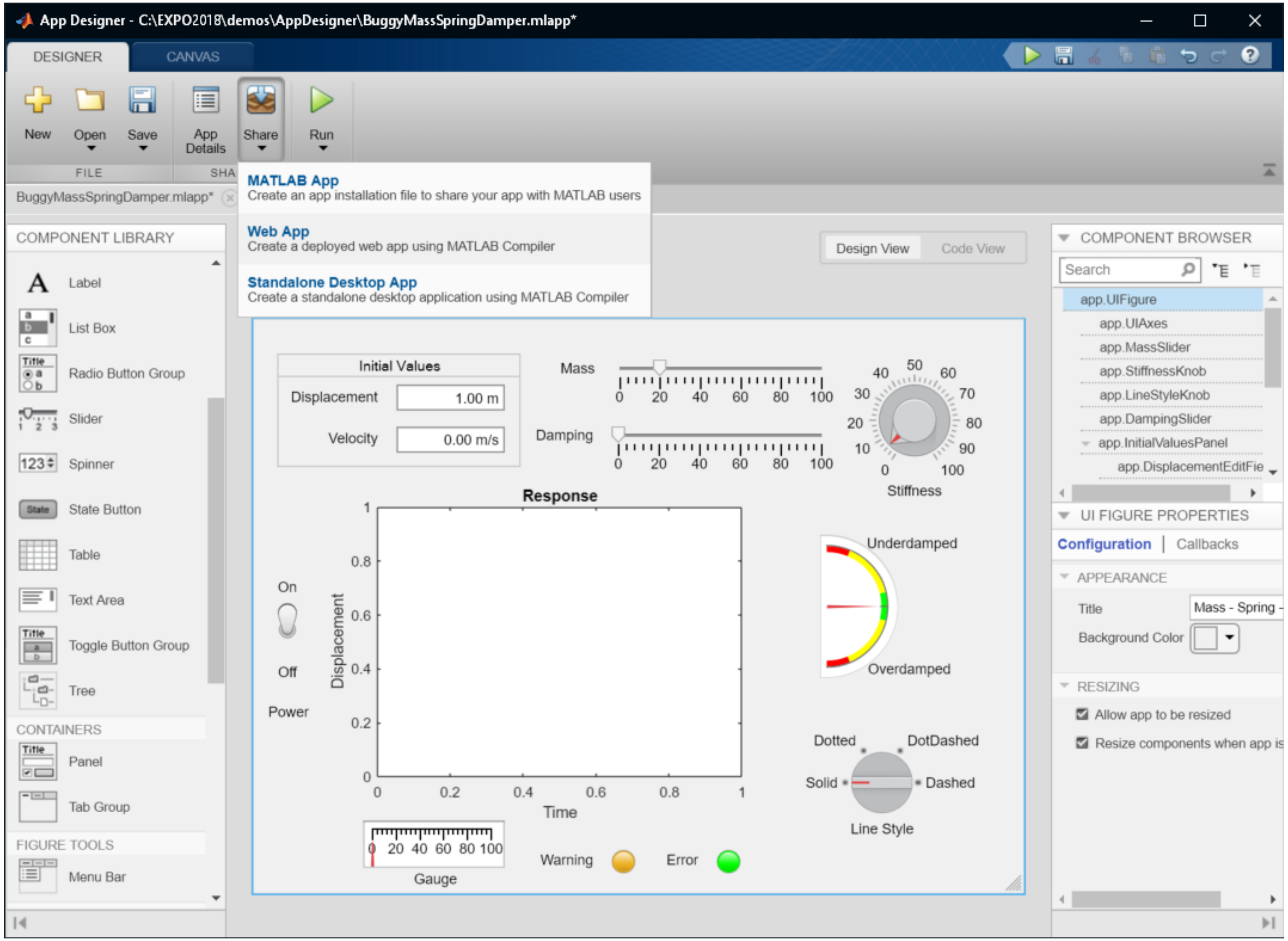
Tags

Add Tags

app designer

Standalone desktop app & Web Deploy

R2018a

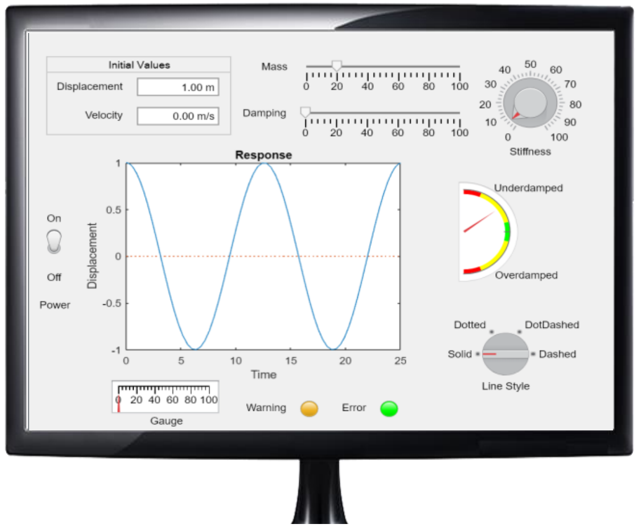


MATLAB App
Create an app installation file to share your app with MATLAB users

Web App
Create a deployed web app using MATLAB Compiler

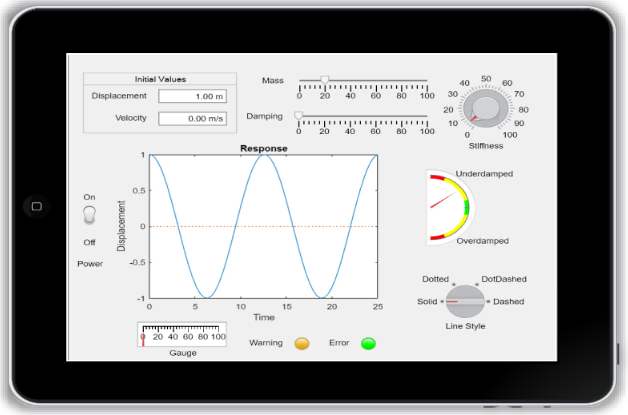
Standalone Desktop App
Create a standalone desktop application using MATLAB Compiler

Stand alone



app.exe

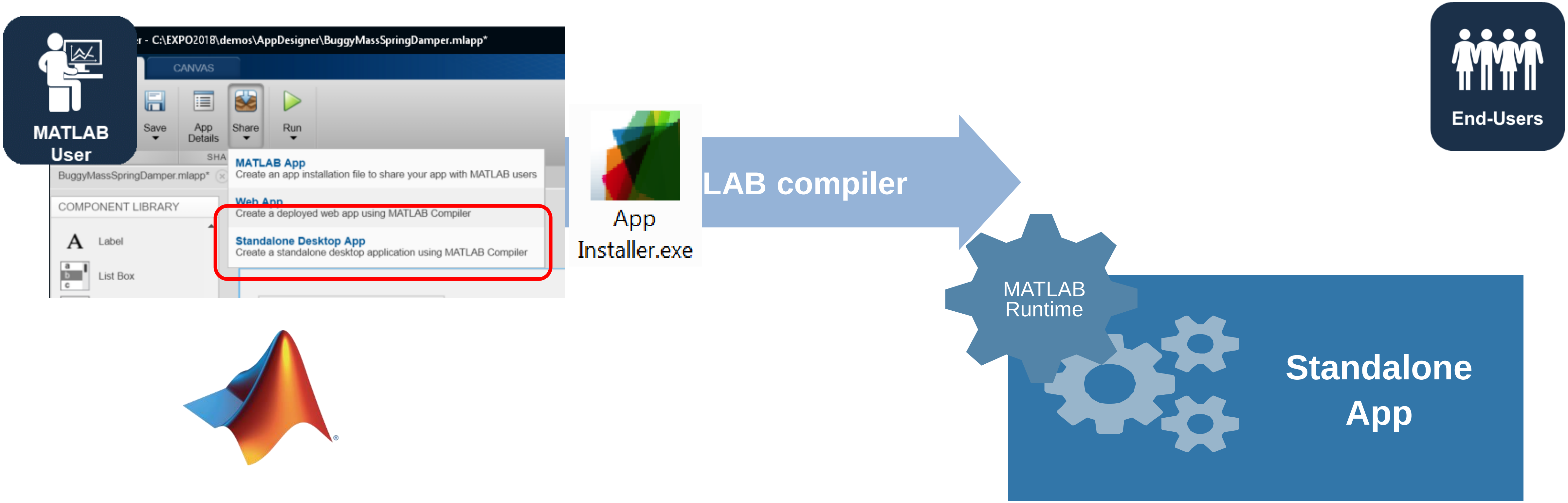
Web deploy



https:\\www.app.com

MATLAB EXPO 2018

Sharing apps before R2018a



r - C:\EXPO2018\demos\AppDesigner\BuggyMassSpringDamper.mlapp*

MATLAB App
Create an app installation file to share your app with MATLAB users

Web App
Create a deployed web app using MATLAB Compiler

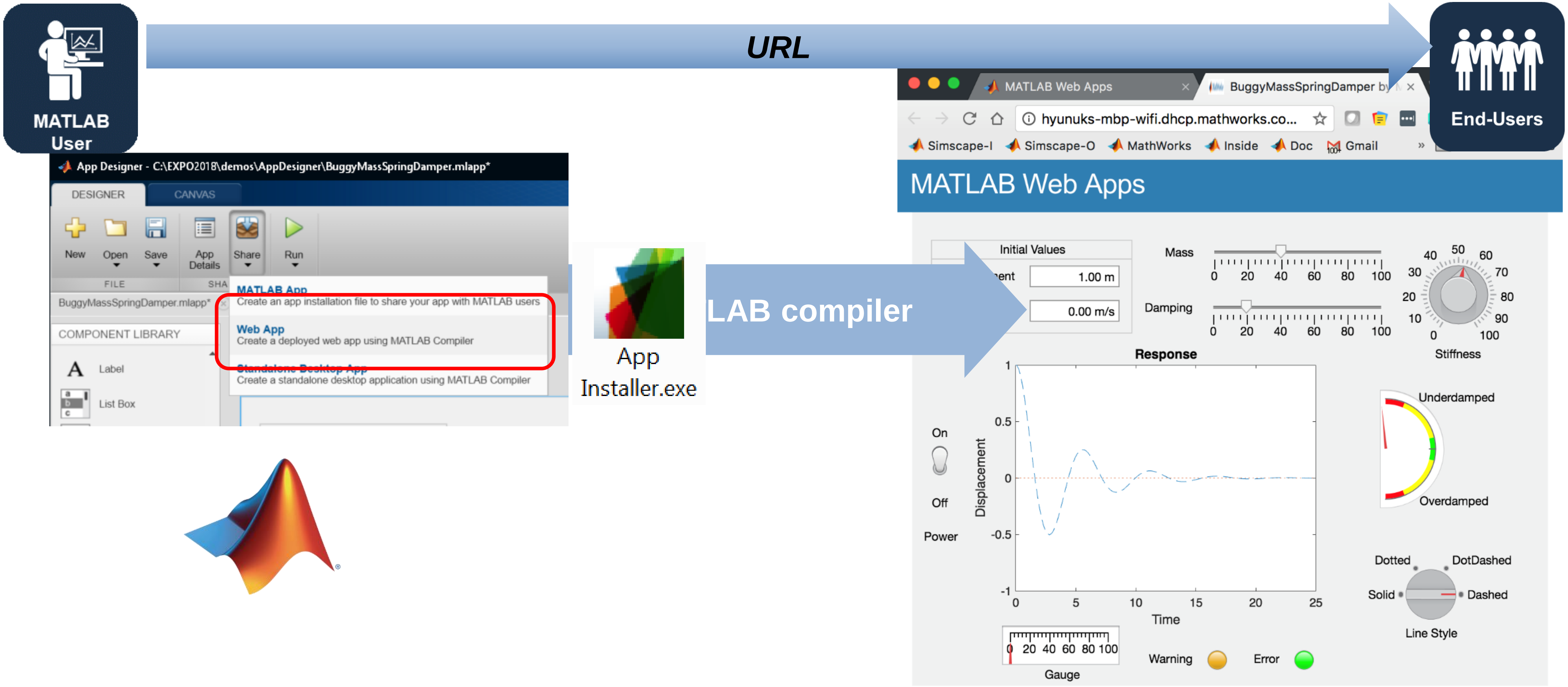
Standalone Desktop App
Create a standalone desktop application using MATLAB Compiler

BuggyMassSpringDamper.mlapp*

COMPONENT LIBRARY

Yesterday

Sharing apps after R2018a



MATLAB App
Create an app installation file to share your app with MATLAB users

Web App
Create a deployed web app using MATLAB Compiler

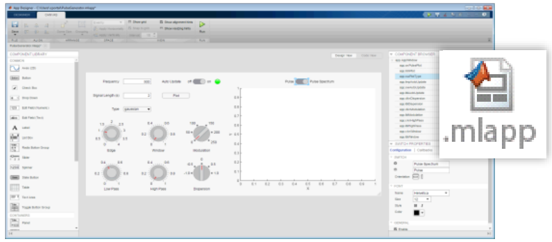
Standalone Desktop App
Create a standalone desktop application using MATLAB Compiler

Today with R2018a

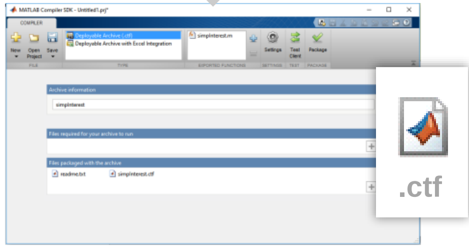
MATLAB User

URL

End-Users

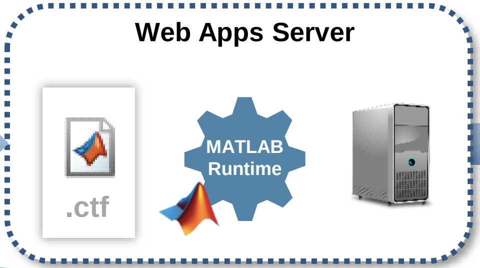


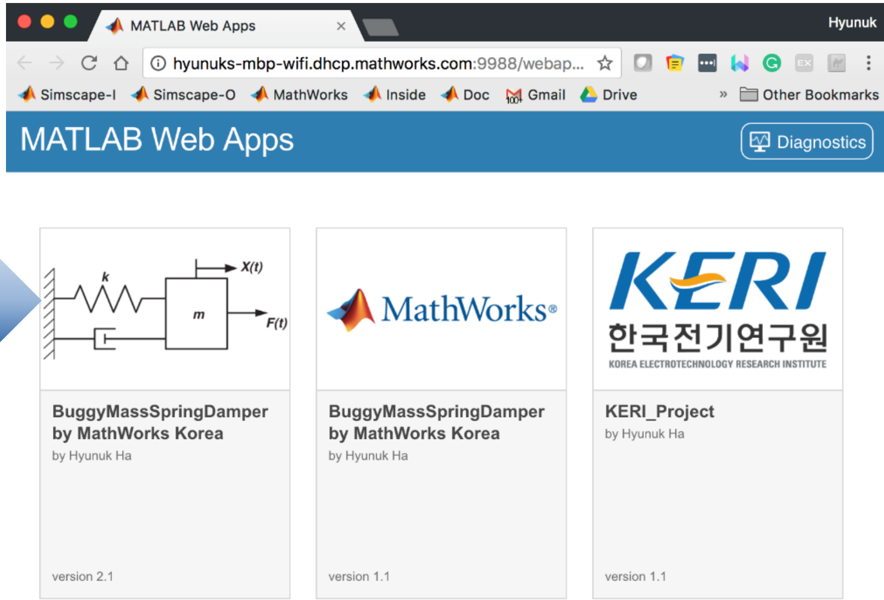
App Designer


MATLAB Compiler SDK - Untitled1.prj*
COMPILER
Deployable Archive (.ctf)
Deployable Archive with Excel Integration
simpleInterest.m
Settings
Test Client
Package
Archive information
simpleInterest
Files required for your archive to run
Files packaged with the archive
readme.txt
simpleInterest.ctf
.ctf

MATLAB Compiler

App





BuggyMassSpringDamper by MathWorks Korea
by Hyunuk Ha
version 2.1

BuggyMassSpringDamper by MathWorks Korea
by Hyunuk Ha
version 1.1

KERI_Project
by Hyunuk Ha
version 1.1

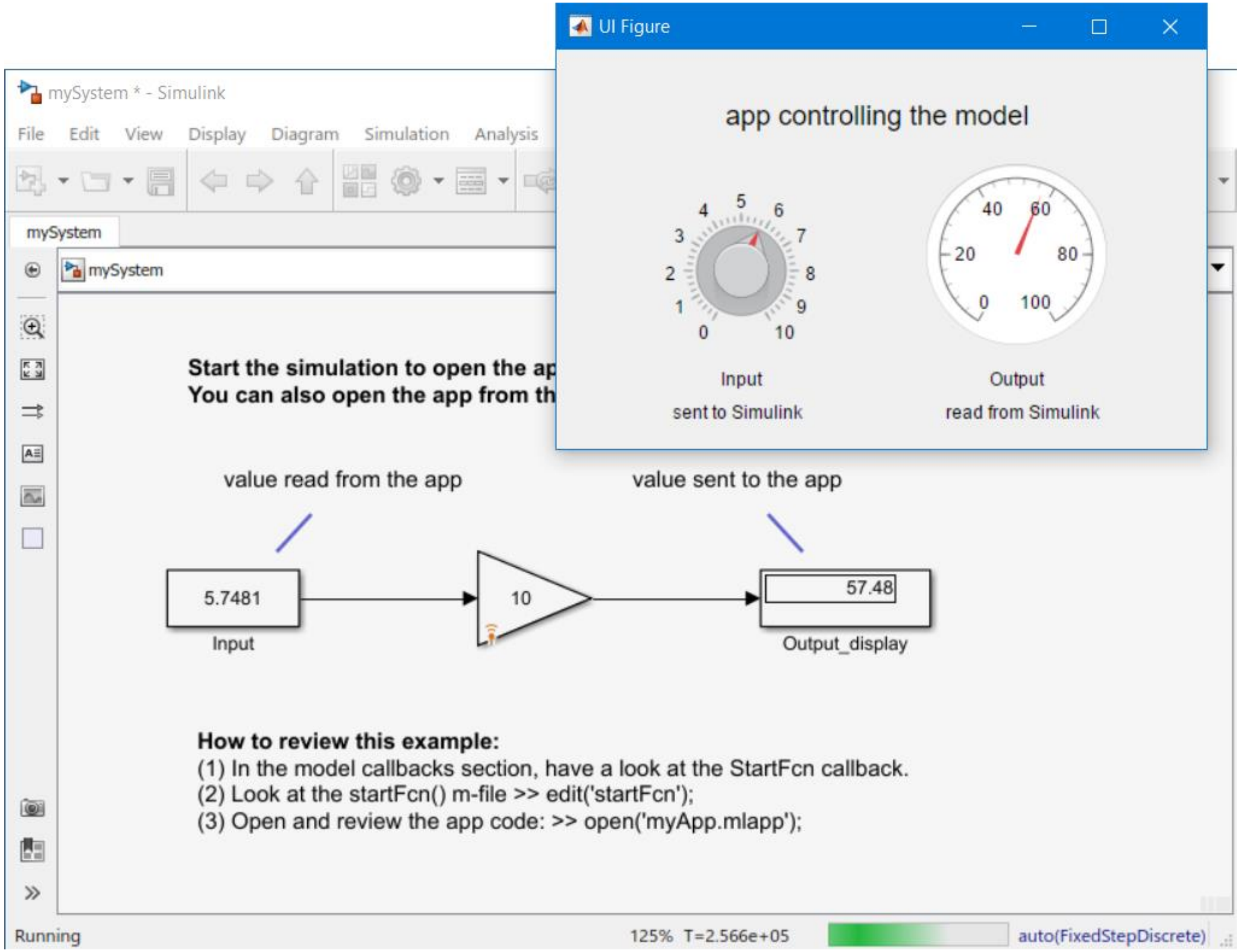
MATLAB_Deployed_Web_Apps Installer
Connection Settings
MATLAB_Deployed_Web_Apps 1.0
MATLAB & SIMULINK
R2017a
< Back
Next >
Cancel

Installer

+ Additional feature for HMI : Integrating with Simulink

- Objective: Use an app to write and read block values in a Simulink model

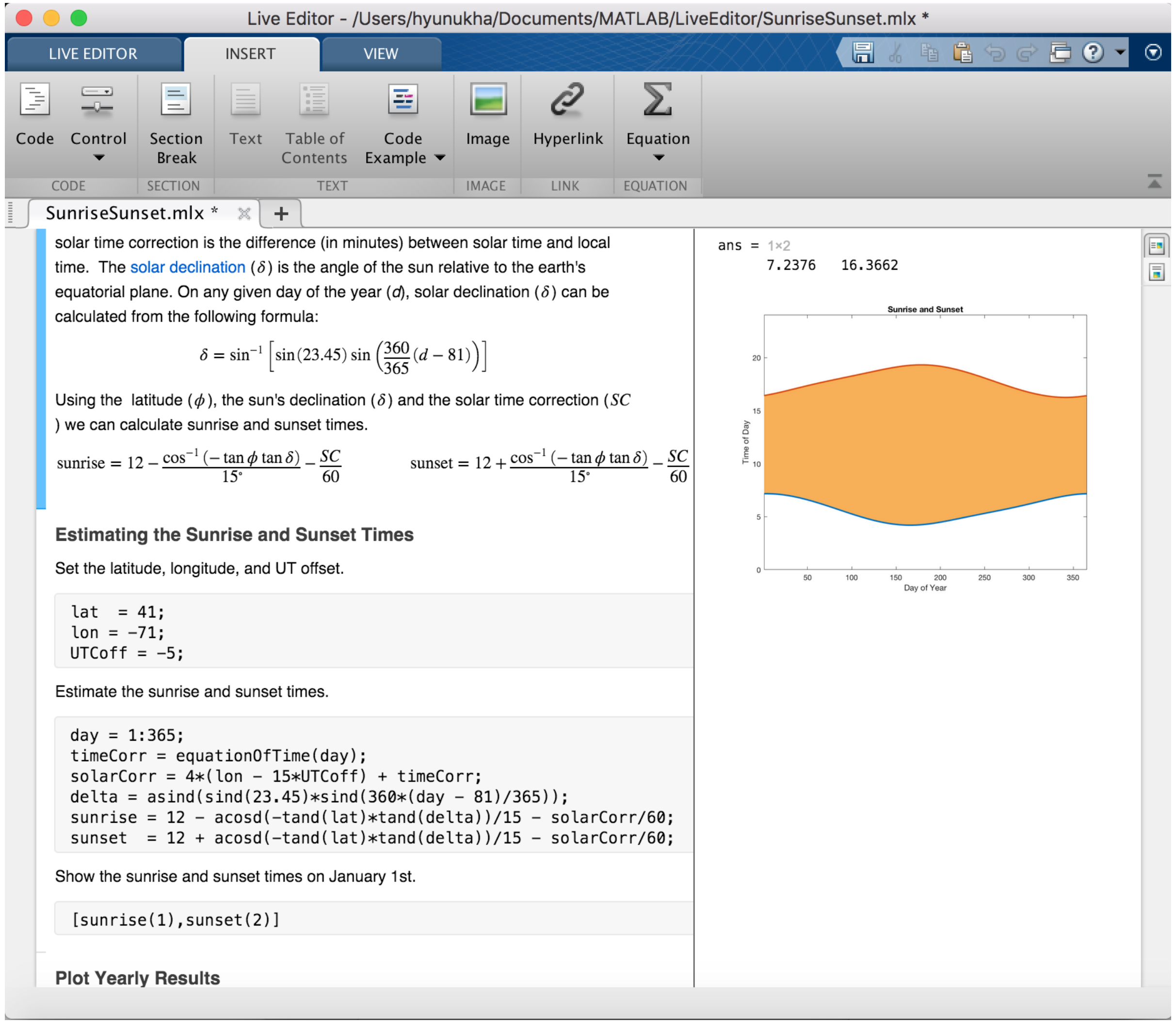
- What we'll see
  - Open an app from a model
  - Set a model parameter from the app
  - Display a value from the model in the app



Start the simulation to open the ap
You can also open the app from th

How to review this example:
(1) In the model callbacks section, have a look at the StartFcn callback.
(2) Look at the startFcn() m-file >> edit('startFcn');
(3) Open and review the app code: >> open('myApp.mlapp');

Any questions on interactive programming in MATLAB?



solar time correction is the difference (in minutes) between solar time and local time. The solar declination (δ) is the angle of the sun relative to the earth's equatorial plane. On any given day of the year (d), solar declination (δ) can be calculated from the following formula:

$$\delta = \sin^{-1}\left[\sin(23.45)\sin\left(\frac{360}{365}(d-81)\right)\right]$$

Using the latitude (ϕ), the sun's declination (δ) and the solar time correction (SC) we can calculate sunrise and sunset times.

$$\text{sunrise} = 12 - \frac{\cos^{-1}(-\tan\phi\tan\delta)}{15^{\circ}} - \frac{SC}{60} \qquad \text{sunset} = 12 + \frac{\cos^{-1}(-\tan\phi\tan\delta)}{15^{\circ}} - \frac{SC}{60}$$

Estimating the Sunrise and Sunset Times

Set the latitude, longitude, and UT offset.

```
lat  = 41;
lon  = -71;
UTCoff = -5;
```

Estimate the sunrise and sunset times.

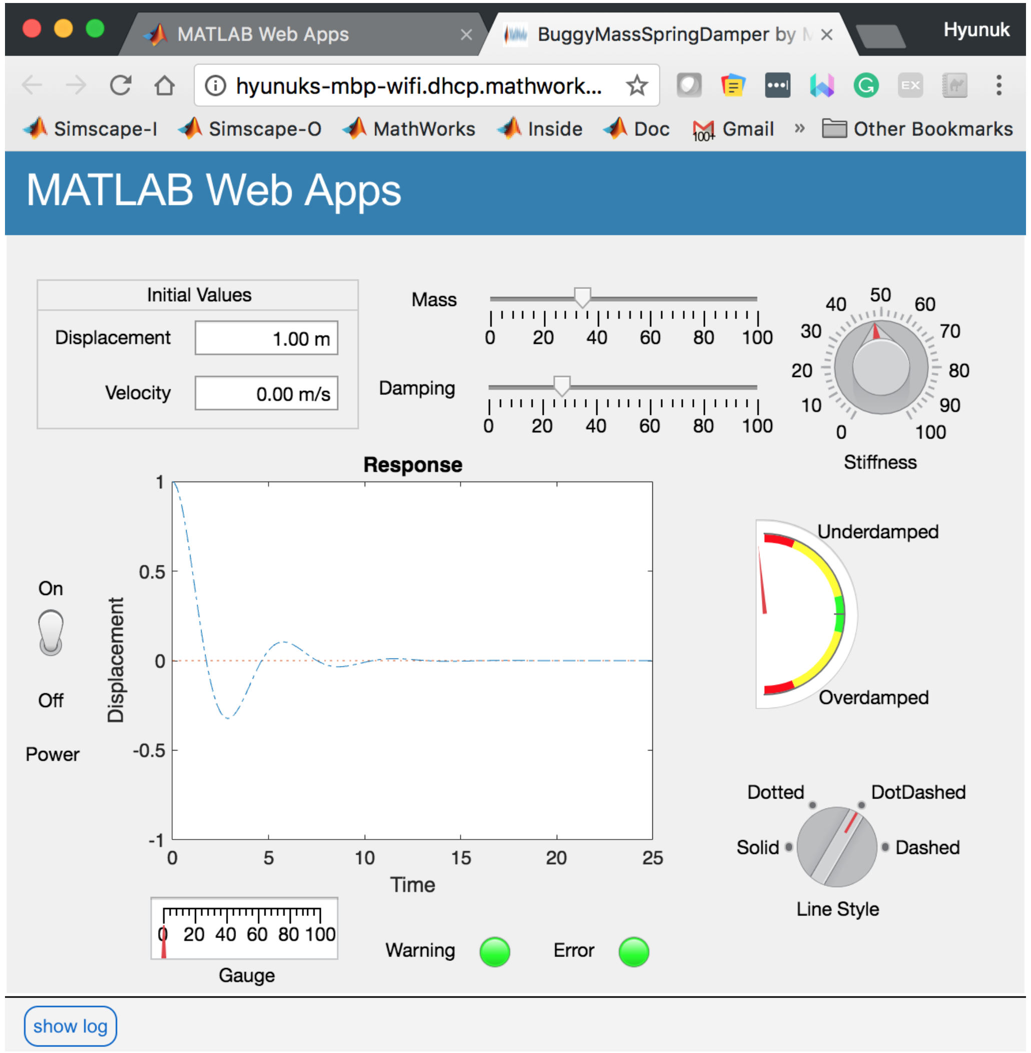
```
day = 1:365;
timeCorr = equationOfTime(day);
solarCorr = 4*(lon - 15*UTCoff) + timeCorr;
delta = asind(sind(23.45)*sind(360*(day - 81)/365));
sunrise = 12 - acosd(-tand(lat)*tand(delta))/15 - solarCorr/60;
sunset  = 12 + acosd(-tand(lat)*tand(delta))/15 - solarCorr/60;
```

Show the sunrise and sunset times on January 1st.

```
[sunrise(1),sunset(2)]
```

```
ans = 1×2
      7.2376   16.3662
```

Plot Yearly Results



Initial Values

Displacement 1.00 m

Velocity 0.00 m/s