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
Conventional plain text script

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

MATLAB EXPO 2018
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
Create functions with formatted documentation

```
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

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

Estimate the sunrise and sunset times.

day = 1:365;
timeCorr = equationOfTime(day);
solarCorr = 4*(lon - 15*UTCoff) + timeCorr;

solarCorr = 4*(lon - 15*UTCoff) + timeCorr;

delta = asind(sind(23.45)*sind(360*(day - 81)/365));

Continued running up to this line and pause.

Show the sunrise and sunset times on January 1st.

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

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

```
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

day = 1:365;
timeCorr = equationOfTime(day);

solarCorr = 50*(day - 81)/365);
sunrise = 12 - acosd(stand(lat)*tand(delta))/15 - solarCorr;

mean(A, dim, options)
Report generation to pdf, html, and LaTeX

App Testing Framework

Table of Contents

Start your app
Initialize an interactive TestCase
Drag a knob
Testing your app
What does a failure look like?
Class-Based Testing
Run all class-based tests on the buggy app

The App Testing Framework is designed to automate 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.

Export to pdf, html, LaTeX

<Live editor>

<Automatic contents generation> with section title

<Table of Contents>

Start your app
Initialize an interactive TestCase
Drag a knob
Testing your app
What does a failure look like?
Class-Based Testing
Run all class-based tests on the buggy app

The App Testing Framework was introduced in MATLAB R2018a.

Start your app
app = BuggyMassSpringDamper;

Initialize an interactive TestCase
testCase = matlab.uittest.TestCase.ForInteractiveUse;

Drag a knob
testCase.drag(app.StiffnessKnob,0,100);

The App Testing Framework was introduced in MATLAB R2018a.

Start your app
app = BuggyMassSpringDamper;

Initialize an interactive TestCase
testCase = matlab.uittest.TestCase.ForInteractiveUse;

Drag a knob
testCase.drag(app.StiffnessKnob,0,100);

App Testing Framework

<html for web sharing>  <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
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
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
**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

MATLAB EXPO 2018
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
Coding App Behavior – Basic Steps

✓ Select a component
✓ Create a callback
✓ Add callback code
✓ Use hints to avoid common programming errors
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
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
## Differences between GUIDE and App Designer

<table>
<thead>
<tr>
<th>GUIDE</th>
<th>App designer</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="GUIDE Interface" /></td>
<td><img src="image2" alt="App Designer Interface" /></td>
</tr>
<tr>
<td>Use figure functions and figure properties</td>
<td>Use uifigure functions and UI Figure properties</td>
</tr>
<tr>
<td>Essential components for GUI</td>
<td>More HMI friendly components</td>
</tr>
<tr>
<td>Standalone deployable</td>
<td>Standalone deployable + Web deployable</td>
</tr>
<tr>
<td>Use get, set functions</td>
<td>Use dot notation using class</td>
</tr>
</tbody>
</table>
GUIDE to App Designer Migration Tool for MATLAB

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.

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 a web browser.
Standalone desktop app & Web Deploy

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

app.exe
https:\www.app.com
Sharing apps before R2018a

MATLAB User

LAB compiler

App

Installer.exe

MATLAB Runtime

Standalone App

End-Users

Yesterday

MATLAB EXPO 2018
Sharing apps after R2018a

Today with R2018a
+ 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

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.miapp');
Any questions on interactive programming in MATLAB?