Lecture 9: Dynamic Graphical User Interface
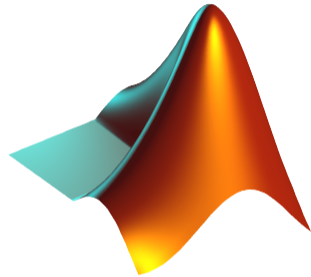## BE0B17MTB – Matlab

Miloslav Čapek, Viktor Adler, Pavel Valtr, Michal Mašek, and Vít Losenický

Department of Electromagnetic Field
Czech Technical University in Prague
Czech Republic
matlab@elmag.org

April 23, 2020
Summer semester 2019/20

Outline

1. Dynamic GUI
2. Excercises

Callback Function

- There are user-evoked events related to each object (button click, list selection, slider move, . . .).
- These events are served by so called callback functions.
  - In other words, when user pushes button, move slider, . . . , callback function of this event is activated (if defined).
- When GUI is not to be static, it has to contain at least one callback function.
- Callback function is stored as an object property -– it is possible to change it, delete it, copy it etc.
- If a `Enable` property of an object is set to `'off'`, the callback function is not called.

GUI

List of Callback Functions I.

ButtonPushedFcn	uibutton
ButtonDownFcn	uifigure
CellEditCallback	uitable
CellSelectionCallback	uitable
CloseRequestFcn	uifigure
ContextMenuOpeningFcn	uicontextmenu
CreateFcn	all UI components and containers
DeleteFcn	all UI components and containers
DisplayDataChangedFcn	uitable
ImageClickedFcn	uiimage
KeyPressFcn	uifigure
KeyReleaseFcn	uifigure
MenuSelectedFcn	uimenu
NodeCollapsedFcn	uitree
NodeExpandedFcn	uitree
NodeTextChangedFcn	uitree

GUI

List of Callback Functions II.

SelectionChangedFcn	uibuttongroup, uitree, uitabgroup
SizeChangedFcn	uiaxes, uibuttongroup, uipanel, uitab, uitabgroup, uifigure
ValueChangedFcn	uicheckbox, uidatepicker, uidropdown, uieditfield, uilistbox, uislider, uispinner, uibutton('state'), uitextarea, uiknob, uiswitch
ValueChangingFcn	uislider, uispinner, uiknob('continuous')
WindowButtonDownFcn	uifigure
WindowButtonMotionFcn	uifigure
WindowButtonUpFcn	uifigure
WindowKeyPressFcn	uifigure
WindowKeyReleaseFcn	uifigure
WindowScrollWheelFcn	uifigure

GUI

Evaluation of Callback Function I.

▶ Callback function defined as function handle without input arguments:

```
clear;
hFig = uifigure();
hButt = uibutton(hFig, 'Text', 'Plot line');
% set callback function:
hButt.ButtonPushedFcn = @plotLine;


function plotLine(hButt, eventData)
% callback function has by default two input arguments
hButt % handle to the object raising the callback
eventData % description of the event
end
```

GUI

Evaluation of Callback Function II.

▶ Callback function defined as function handle with specified input arguments:

```
clear;
hFig = uifigure();
hButt = uibutton(hFig, 'Text', 'Plot line');
myVar = pi;
% set callback functions:
hButt.ButtonPushedFcn = @(src, event)plotLine(myVar);


function plotLine(myVar)
% the only input argument is the one defined by user
myVar
end
```

GUI

Evaluation of Callback Function III.

- ▶ Callback function defined as `cell` variable.
  - ▶ First element is function handle without input arguments.
  - ▶ Next elements are additional input arguments.

```matlab
clear;
hFig = uifigure();
hButt = uibutton(hFig, 'Text', 'Plot line');
myVar1 = pi;
myVar2 = 2*pi;
% set callback functions:
hButt.ButtonPushedFcn = {@plotLine, myVar1, myVar2};


function plotLine(hButt, eventData, myVar1, myVar2)
% Basic inputs added to first positions again
hButt % handle to the object raising the callback
eventData % description of the event
myVar1 % additional input
myVar2 % additional input
end
```

GUI

Evaluation of Callback Function IV.

▶ Callback function defined as character vector.

```
clear;
hFig = uifigure();
hButt = uibutton(hFig, 'Text', 'Plot line');
% set callback functions:
hButt.ButtonPushedFcn = 'plot(rand(20, 3))';
```
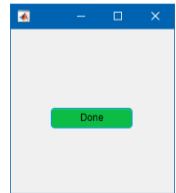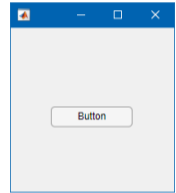
▶ Very limited possibilities.
  ▶ The string can contain variables as well.
  ▶ Only the variables from base Workspace are evaluated correctly.
  ▶ Handle of source object is not available.
  ▶ It is possible to call just scripts or main functions with predefined inputs.
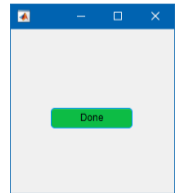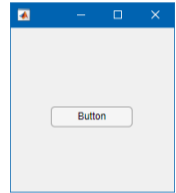▶ Not recommended!

GUI

Evaluation of Callback Function V.

▶ Change background color of push button and change its label to 'Done' when clicked.

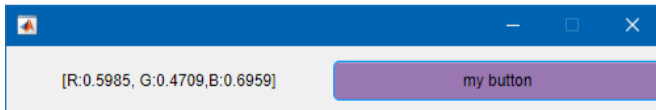▶ Two default input arguments of callback function utilized.

```matlab
function GUI()
hFig = uifigure('Position', 200*ones(1, 4));
hButt = uibutton(hFig, 'Position', [50, 50, 100, 25], ...
    'Text', 'Button');
hButt.ButtonPushedFcn = @pressButton;
end

function pressButton(src, eventData)
% src and event are default input arguments
% src  - handle to the object raising the callback
% event - description of the event

set(src, 'Text', 'Done', 'BackgroundColor', rand(1, 3));
end
```


GUI

Evaluation of Callback Function VI.

▶ Change background color of push button and change its label to 'Done' when clicked.

▶ The useless input argument suppressed.

```matlab
function GUI()
hFig = uifigure('Position', 200*ones(1, 4));
hButt = uibutton(hFig, 'Position', [50, 50, 100, 25], ...
    'Text', 'Button');
hButt.ButtonPushedFcn = @(src, event)pressButton(hButt);
end

function pressButton(hButt)
% hButt  - handle to the object raising the callback

set(hButt, 'Text', 'Done', 'BackgroundColor', rand(1, 3));
end
```

Button Callback

- Create figure with button and label box (`uilabel`).
- When clicking on button background color of button changes to random and displays individual RGB components in text box.
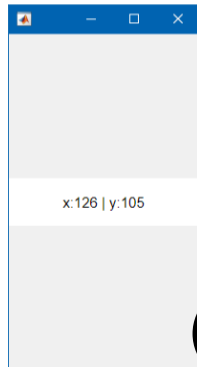- Hint: `>> sprintf('\%f, \%f, \%f', [1 2 3]).`
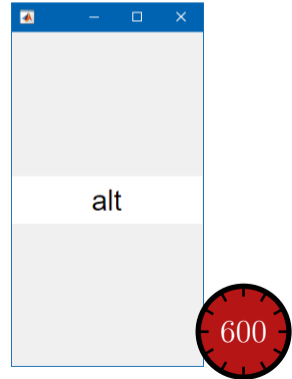
Button Callback

Mouse Position

▶ Create a text array in a figure showing mouse position over the figure.
  ▶ Figure's callback for mouse movement is `WindowButtonMotionFcn`.
  ▶ Mouse position can be found in figure's property `CurrentPoint`.

x:126 | y:105

600

Keyboard Scan

- ▶ Create a text array that displays last key pressed.
  - ▶ Information on the key pressed is to be found in `event` parameter.
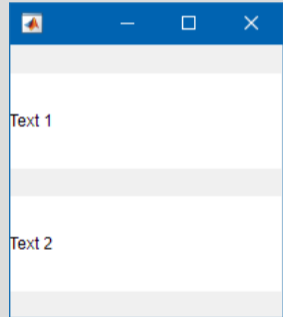 - ▶ Figure's callback for pressing key is `WindowKeyPressFcn`.

Function `findobj`

▶ Finds an object(s) withinn UI container with required property.
▶ Returns handles of the object (or an array of handles).

```matlab
hFig = uifigure('Position', 200*ones(1, 4));
uilabel(hFig, 'Position', [0 110 200 70], ...
  'Text', 'Text 1', 'Tag', 'first text', ...
  'BackgroundColor', 'w');
uilabel(hFig, 'Position', [0 20 200 70], ...
  'Text', 'Text 2', 'Tag', 'second text', ...
  'BackgroundColor', 'w');
```

```matlab
hBothLabels = findobj(hFig, 'Type', 'uilabel')
% or
hBothLabels = findobj(hFig, 'Tag', 'first text', ...
  '-or', 'Tag', 'second text')
```

```matlab
hFirstLabel.HandleVisibility = 'off';
% first label can't be reached by findobj:
hSecondLabel = findobj(hFig, 'Type', 'uilabel')
```
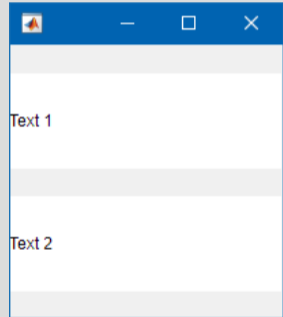
Functions `findall`, `allchild`

▶ `findall` finds all graphic objects (including hidden).

▶ `allchild` finds all children of selected object (including hidden).

```matlab
hFig = uifigure('Position', 200*ones(1, 4));
uilabel(hFig, 'Position', [0 110 200 70], ...
   'Text', 'Text 1', 'Tag', 'first text', ...
   'BackgroundColor', 'w', ...
   'HandleVisibility', 'off');
uilabel(hFig, 'Position', [0 20 200 70], ...
   'Text', 'Text 2', 'Tag', 'second text', ...
   'BackgroundColor', 'w', ...
   'HandleVisibility', 'off');
```

```matlab
hAllObjs = findall(hFig)
hBothLabels = allchild(hFig)
```
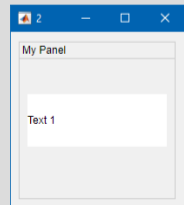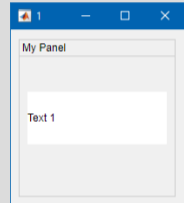
Function `copyobj`

- Creates copies of graphics objects and assigns the objects to the new parent.
  - Copies object and its descendants.

```matlab
hFig1 = uifigure('Position', 200*ones(1, 4), 'Name', '1');
hPanel = uipanel(hFig1, ...
    'Position', [10, 10, 180, 180], ...
    'Title', 'My Panel');
hText = uilabel(hPanel, ...
    'Position', [10, 60, 160, 60], ...
    'Text', 'Text 1', 'Tag', 'first text', ...
    'BackgroundColor', 'w');
```

```matlab
hFig2 = uifigure('Position', 200*ones(1, 4), 'Name', '2');
hPanelCopy = copyobj(hPanel, hFig2)
```
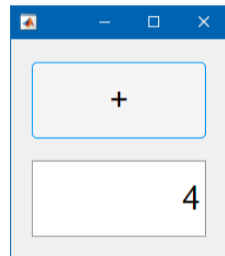
Storing Data in GUI I.

- ▶ How to store data in GUI?
  - ▶ Share variables from main function with nested functions.
    - ▶ Ideal for small applications.
  - ▶ Using property `UserData`.
    - ▶ Acceptable even for large applications.
  - ▶ Using functions `setappdata` and `getappdata`.
    - ▶ Suitable.
  - ▶ Fully OOP access (including functional part of the code).
    - ▶ Ideal, but challenging.
  - ▶ Global variables (extreme case, keyword `global`).
    - ▶ Unacceptable. Forbidden in this course!

GUI

Storing Data in GUI II.

► Sharing variables between main and nested functions.
  ► Shared variables must be defined within the main function!

```matlab
function nestedGUI()
% constants
result = 0;
% GUI building
hFig = uifigure('Position', 200*ones(1, 4));
hEdit = uieditfield(hFig, 'numeric', ...
  'Position', [20, 20, 160, 70], 'Value', result);
hButt = uibutton(hFig, 'Text', '+', ...
  'Position', [20, 110, 160, 70]);
set([hEdit, hButt], 'FontSize', 30);
% set callback
hButt.ButtonPushedFcn = @(src, event)increment();
  function increment()
    % callback function definition
    result = result + 1;
    hEdit.Value = result;
  end
end
```
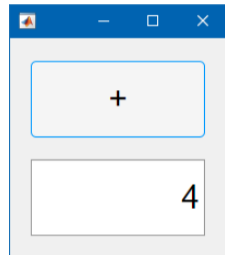
Storing Data in GUI III.

▶ Using property `UserData`.

```matlab
function userDataGUI()
% constants
defResult = 0;
% GUI building
hFig = uifigure('Position', 200*ones(1, 4));
hEdit = uieditfield(hFig, 'numeric', ...
    'Position', [20, 20 160 70], 'Value', defResult);
hButt = uibutton(hFig, 'Text', '+', ...
    'Position', [20, 110, 160, 70]);
set([hEdit, hButt], 'FontSize', 30);
% save data
hEdit.UserData = defResult;
% set callback
hButt.ButtonPushedFcn = @(src, event)increment(hEdit);
end
```

```matlab
function increment(hEdit)
% callback function definition
oldResult = hEdit.UserData;
newResult = oldResult + 1;
hEdit.UserData = newResult;
hEdit.Value = newResult;
end
```

Storing Data in GUI IV.

- ▶ Functions `setappdata` and `getappdata`.
  - ▶ Enables to store or get data.
  - ▶ Data is related to a handle that exist during whole lifetime of GUI.
    - ▶ Data is actually saved in object's parent figure!
    - ▶ Arbitrary descendant of the figure object can be used to retrieve the data.
  - ▶ The procedure is as follows:
    - ▶ Save the data: `setappdata(hObj, 'identifier', data)`.
    - ▶ Get data copy where needed: `data = getappdata(hObj, 'identifier')`.
    - ▶ Carry out data modification / calculation required.
    - ▶ If the data is changed, store it: `setappdata(hObj, 'identifier', data)`.
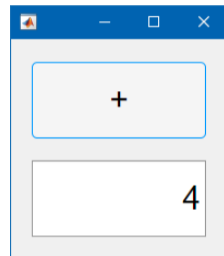 - ▶ It is possible to save different sets of data identified by `'identifier'` argument.

GUI

Storing Data in GUI V.

▶ Functions `setappdata` and `getappdata`.

```matlab
function setGetAppDataGUI()
% constants
defResult = 0;
% GUI building
hFig = uifigure('Position', 200*ones(1, 4));
hEdit = uieditfield(hFig, 'numeric', ...
    'Position', [20, 20 160 70], 'Value', defResult);
hButt = uibutton(hFig, 'Text', '+', ...
    'Position', [20, 110, 160, 70]);
set([hEdit, hButt], 'FontSize', 30);
% save data
setappdata(hEdit, 'myResult', defResult);
% set callback
hButt.ButtonPushedFcn = @(src, event)increment(hEdit);
end
```

```matlab
function increment(hEdit)
% callback function definition
oldResult = getappdata(hEdit, 'myResult');
newResult = oldResult + 1;
setappdata(hEdit, 'myResult', newResult);
hEdit.Value = newResult;
end
```
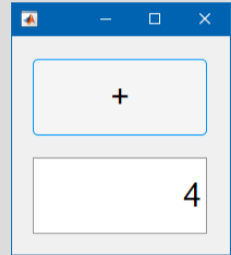

GUI

Storing Data in GUI VI.

- ▶ Object oriented programming.
  - ▶ Run it by `app = myApp`.

```matlab
classdef myApp < handle
    properties (Access = private)
        hFig
        hEdit
        hButt
        result
    end

    methods
        function obj = myApp()
            obj.result = 0;
            obj.hFig = uifigure('Position', 200*ones(1, 4));
            obj.hEdit = uieditfield(obj.hFig, 'numeric', ...
                'Position', [20, 20 160 70], 'Value', obj.result);
            obj.hButt = uibutton(obj.hFig, 'Text', '+', ...
                'Position', [20, 110, 160, 70]);
            set([obj.hEdit, obj.hButt], 'FontSize', 30);
            obj.hButt.ButtonPushedFcn = @(src, event)obj.increment();
        end
    end

    methods (Access = private)
        function increment(obj)
            obj.result = obj.result + 1;
            obj.hEdit.Value = obj.result;
        end
    end
end
```
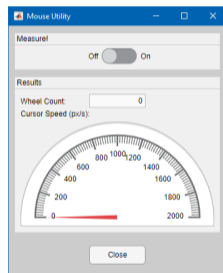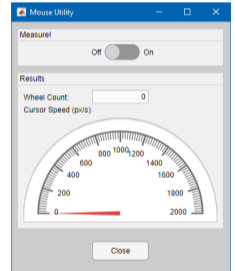
Closing a Figure

▶ Write callback function for Close button which closes the application figure.
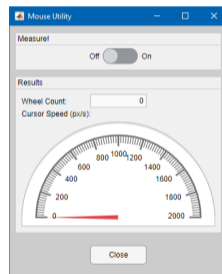 ▶ The figure will close also when a key Esc is pushed.

Count Wheel Steps

▶ Write a callback function for counting a number of mouse wheel steps.
▶ Hint: figure's callback `WindowScrollWheelFcn`, event's property `VerticalScrollCount`.
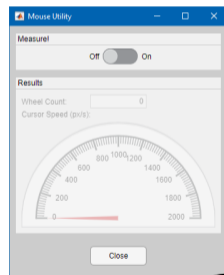
Mouse Cursor Speed

▶ Compute speed of a mouse cursor when moving over app figure.

    ▶ Hints: figure's properties `CurrentPoint` and `WindowButtonMotionFcn`. Functions `clock` and `etime`.

    ▶ Advanced: for smooth speed data compute it from several last positions and times.

Enable Measurement with Switch

▶ When the switch is in Off position set property `Enable` of all components within Results panel to `'off'`.

   ▶ Ensure this state also when starting the application.
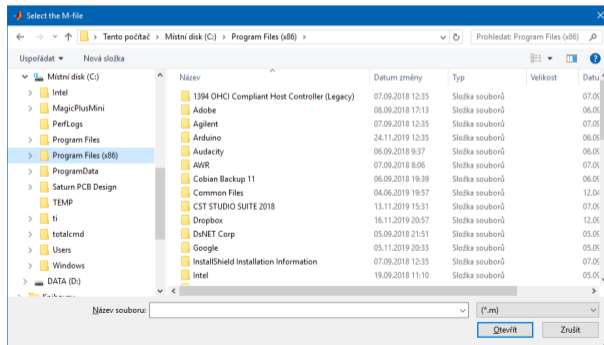   ▶ Do not call callback functions when the components are disabled.

600

Function `uigetfile`

▶ User can select file(s) from file dialog box.
  ▶ Files can be filtered by their suffix.
  ▶ It is necessary to treat the case when Cancel/Storno is pushed.

```
>> [FileName, PathName] = uigetfile('*.m', 'Select the M-file');
```
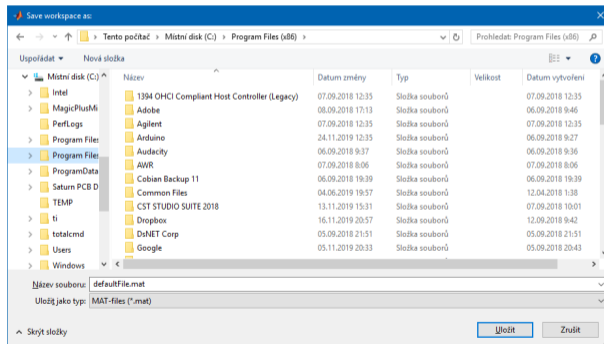

GUI

Function `uiputfile`

▶ Opens dialog for file saving.
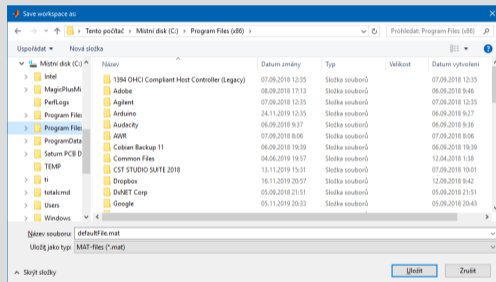 ▶ Files can be filtered by their suffix.

```
>> [FileName, PathName] = uiputfile('*.mat', 'Save workspace as:', ...
   'defaultFile.mat')
```


GUI

Saving into File

► Save variable `data` from Workspace in a file using dialog box.
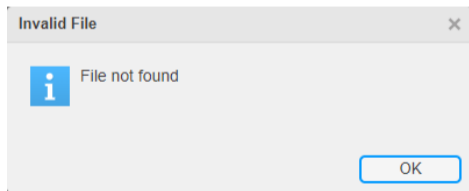 ► Treat the case when Cancel/Storno is pushed.

300

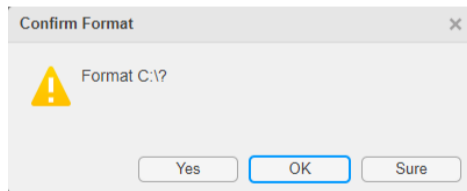# Predefined Dialog Windows – `uialert`, `uiconfirm`

▶ Useful for common alerts and confirmations in application.

▶ Predefined icons settable via `Icon` property:

  ▶ `'error'` – 🛑, `'warning'` – ⚠, `'info'` – ℹ, `'success'` – ✅, `'question'` – ❓

```
hFig = uifigure();
uialert(hFig, 'File not found', ...
  'Invalid File', 'icon', 'info');
```

```
hFig = uifigure();
sel = uiconfirm(hFig, 'Format C:\?', ...
  'Confirm Format',...
  'Options', {'Yes', 'OK', 'Sure'}, ...
  'Icon', 'warning', 'DefaultOption', 2);
```
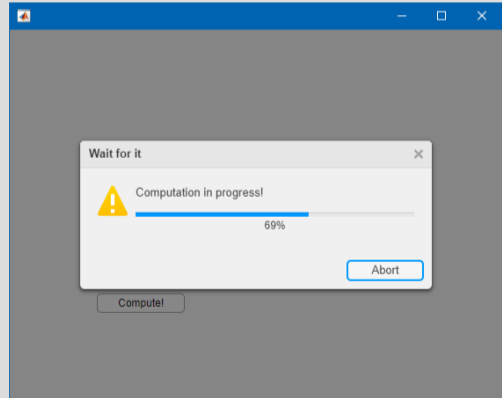
GUI

Function `uiprogressdlg`

- Displays a determinate progress dialog box in figure.
- Aborting the process is possible via `CancelRequested` property.

```matlab
function compWithProgress()
hFig = uifigure();
uibutton(hFig, 'Text', 'Compute!', ...
    'ButtonPushedFcn', @(a, b)compute);

    function compute()
        hDiag = uiprogressdlg(hFig, ...
            'Message', 'Computation in progress!', ...
            'Title', 'Wait for it', 'ShowPercentage', 'on', ...
            'Cancelable', 'on', ...
            'CancelText', 'Abort');
        nSteps = 1e2;
        for iStep = 1:nSteps
            hDiag.Value = iStep/nSteps;
            pause(0.02);
            if hDiag.CancelRequested
                pause(1);
                hDiag.close();
                break;
            end
        end
        hDiag.close();
    end
end
```

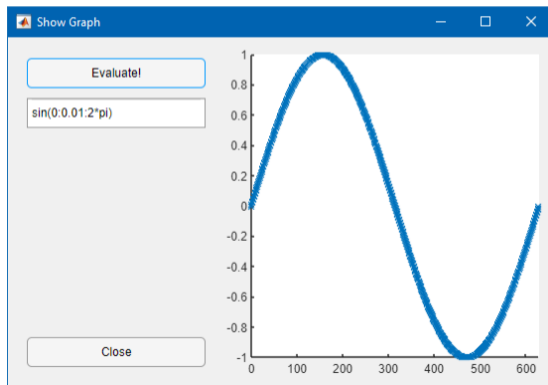# Design of a Simple GUI – Recommended Procedure

- ▶ Consider:
  - ▶ what the GUI should do (detailed description of functionality),
  - ▶ what are the user inputs,
  - ▶ required outputs,
  - ▶ objects used (scheme of GUI, list of elements, available properties),
  - ▶ callback functions, dynamic elements,
  - ▶ required outputs,
  - ▶ saving of handles and data in GUI,
  - ▶ programming style,
  - ▶ implementation of individual parts.
- ▶ Getting it to work, testing, debugging, ...
- ▶ Proceed in small steps.

GUI

Exercises

Exercise I.a Displaying Graph of a Function

▶ Create a GUI which shows a graph of a function defined by a user.
  ▶ Use `try-catch` to eliminate erroneous inputs.
  ▶ Indicate somehow the erroneous input (color of edit field, `uialert`, ...)
  ▶ What function do you use to evaluate the text input?

Exercise I.b Displaying Graph of a Function

Exercise II.a Paint

▶ Create an application allowing simple drawing by a mouse movements over a figure (axes).

  ▶ Allow change of drawing color and clearing of the canvas.
  ▶ Adding other functionality is up to you.

▶ Hints:

  ▶ draw individual lines as `line` objects in `uiaxes` object,
  ▶ `TightInset` property of `uiaxes` object specify border size around `uiaxes`,
  ▶ it is necessary to transform `CurrentPoint` value of the `uifigure` to coordinates of the `uiaxes`,
  ▶ color picker is `uisetcolor` (treat Cancel button),
  ▶ read `CurrentPoint` property of the `uifigure` just after pushing the mouse left button.

1200

Exercise II.b Paint

Questions?

BE0B17MTB – Matlab
matlab@elmag.org

April 23, 2020
Summer semester 2019/20