Simple Tool for Importing Characteristic Curves for MATLAB

The tool converts graphical data into numerical data and can be used to generate fit functions from the data to facilitate optimization routines or comparisons between similar components. Support for graphs with logarithmic axes is also provided. The tool can be used either as a Matlab function or a Matlab App for those who prefer working with a graphical user interface.

Functionalities

- Converts image data into numerical data
- Handles linear and logarithmic axes
- Has configurable axes limits
- Creates fit functions from the data
- Can be used with a GUI and as a command line function
- Can be scripted inside a Matlab m-file

Prerequisites

- Matlab
- Curve Fitting Toolbox

Tested on:

- Matlab 2017b (Windows)
- Matlab 2019a (MacOS)

Installing and usage

Command line functions

Installing

Copy the two .m -files in to the desired Matlab working folder.

Usage

`fitting` is used to generate a fitted curve from the datapoints that have been imported using `image_digitizer`.

The function call is executed in the Matlab command window:

```
fo = fitting(filename, n_curve, Xlim, Ylim, fit_type)
```

The definitions of the input parameters are:

```
filename:   Location of the image file as a string e.g 'C:\image.bmp'
n_curve:    Total number of curves in the image file. The function
            searches for n number of curves that have different
            RGB values.
Xlim:       X-axis limits in vector format e.g [0 1000]. If the minimum
            value is '0', the user can also omit the minimum value
            e.g Xlim = 1000
Ylim:       Y-axis limits in vector format e.g [0 1000]. If the minimum
            value is '0', the user can also omit the minimum value
            e.g Ylim = 1000
fit_type:   Fit type passed through to the function FIT. Supports all the
            same function names as FIT (e.g 'poly1', 'poly2' etc.).
            Additionally one can use 'foster' to fit a 4th order
            foster function.
```

The default RGB values for the search are:

```
    CURVE{1} = [0 255 0]   % Green
    CURVE{2} = [255 0 0]   % Red
    CURVE{3} = [0 0 255]   % Blue
    CURVE{4} = [255 255 0] % Yellow
    CURVE{5} = [0 255 255] % Cyan
```

and for the axes calibration points:

```
    Calibdot = [255 0 255] % Magenta
```

The above default RGB values can be changed inside the function file. Additionally the user can define optional input parameters as follows:

```
fo = fitting(..., 'OptParam1', value1, 'OptParam2', value2, ...)
```

Name-value pairs are used to determine the optional input parameters. These parameters use their default values if not given as an input.

The definitions of the optional input parameters are:

```
Plot:       Set to '1' to plot the results in a figure. The default
            value is '1'.
Xlog:       Set to '1' if the image has a logarithmic X-axis and to '0'
            if the axis is linear. The default value is '0'.
Ylog:       Set to '1' if the image has a logarithmic Y-axis and to '0'
            if the axis is linear. The default value is '0'.
```

```
Invert:     Set to '1' to invert the X and Y axes before fitting a curve.
            The default value is '0'.
FigNum:     Number of the figure (if the function is called several times
            the user can plot the results in separate figures)
FigLegend:  Figure legend defined as a string cell
            e.g {'Curve 1', 'Curve 2'}.
            Defaults to 'Curve n', where n = 1,2,3...n_curve
```

`image_digitizer` searches pixels from an image file based on user defined RGB values. The pixel x and y indexes are scaled based on user given axes limits to convert a graph to numerical data. The function has support for both linear and logarithmic axes.

The function call is executed in the Matlab command window:

```
[outX, outY] = image_digitizer(filename, linecol, calibpoint, Xlim, Ylim)
```

The definitions of the input parameters are:

```
filename:    Location of the image file as a string
             e.g 'C:\image.bmp'
linecol:     RGB value of the curve points the user wants to
             find e.g [0 255 255]
calibpoint:  RGB value of the calibration points that define
             the origin and axis limits
             e.g [255 0 255]
Xlim:        X-axis limits in vector format e.g [0 1000]
Ylim:        Y-axis limits vector format e.g [0 1000]
```

Additionally the user can define optional input parameters as follows:

```
[outX, outY] = image_digitizer(..., 'Xlog', value1, 'Ylog', value2)
```

Name-value pairs are used to determine the optional input parameters. These parameters default to 0 if not given as an input.

The definitions of the optional input parameters are:

```
Xlog:        Set to '1' if the image has a logarithmic X-axis and to '0'
             if the axis is linear
Ylog:        Set to '1' if the image has a logarithmic Y-axis and to '0'
             if the axis is linear
```

Example of the command line function call

Examples of the typical function calls for `fitting` are provided with the repository. The examples can be executed by running the following file in Matlab:

```
function_call_example_paper.m
```

Please make sure to store the example image files provided with the repository in the same path with the above .m file.

App

Installing

If you have Matlab installed then you can simply download the repository and double click the .mlapp file and follow the on-screen instructions.

If you run in to any issues please refer to the Matlab documentation on Matlab Apps.

Usage

After the App has been installed the user can begin the data importing process by opening the App. The App requires the user to take a screenshot of the datasheet graph in question and to save it as an image file. The image file can be opened in the App by pressing the *Open image file* button in the App GUI and by selecting the image file from the file browser. The user can proceed by selecting the origin of the graph and a point for the maximum x and y values (buttons *Define Origin* and *Define max X/Y*). Next, the user can begin defining the curve by pressing the *Define Curve* button. A new Matlab figure is automatically opened and the curve points can be selected from the figure using the cursor. If necessary, delete points by pressing Backspace or Delete. The last point can be added by shift-, right-, or double-click, which also ends the selection mode. The selection can also be finalized by pressing Enter or Return without selecting a final point. The external figure closes automatically after the selections have been carried out, whereas the selections remain plotted in the figure located inside the GUI.

The App only selects pixel coordinates from the image file and therefore, the user has to define what the actual values in the graph are. The values of the axis limits in the original datasheet graph can be defined in the fields *Origin* and *Max X/Y*. The axis type can be selected separately for both the x and y axes in the drop-down menus *X-axis* and *Y-axis*, whereas the values of the curve points are scaled automatically depending on the axis type.

The App also provides an opportunity to fit a curve to the imported data. The desired fit type can be selected from a drop-down menu *Fit type*, which offers a selection of the typical polynomial, exponential, and power fit functions. If the selectable fit functions are not sufficient, the user can define the desired function by selecting custom from the drop-down menu and then enter a function in the editable field *Custom*. The user can enter any of the Matlab-supported fit function names as the input. After all of the fit options have been defined, the fitting procedure can be started by pressing the button *Start fitting*.

The fitted curve and original data points are automatically plotted in an external figure for visual inspection, whereas numerical indicators of the fit quality such as the sum of squares due to error (SSE) and R-square can be viewed in the main App window under *Goodness of fit*. The curve data can be saved to the Matlab workspace by pressing the button *Save Data*. The output uses the same Matlab struct data type as the function `fitting` described in the previous subsection and contains the fit object *cfit*, the numerical values of the user selected curve data points, the fit coefficients, and the goodness of fit data. When the user presses the *Save Data* button, the App prompts for curve and output variable names, whereas a default naming system is used if no user input is given. The prompt fields are prefilled using the default naming system, whereas the last user inputted output variable name is saved for convenience if the user wishes to save several curves from a single image file. Each curve can be saved consecutively under the same output variable name, and the user can for example try out several different fit functions before saving each curve. The output variable is saved in an internal cache of the App and is cleared when a new image file is loaded.

Licensing

When resusing this work please cite the following publication:

A. Mattsson, V. Tikka, P. Silventoinen, and J. Partanen, “Working with Datasheet Information—A Handy Tool for Importing Characteristic Curves,” in IEEE Trans. Power Electron., vol. , no. pp, January 2020.