Matlab Laboratory 6. Graphical User Interfaces in Matlab

Aim of the laboratory

- 2D and 3D graphical representations
- Customization of Matlab plots

Necessary equipment

- Workstations with LTSpice and MATLAB

Theoretical Approach

The first step in designing a graphical user interface (GUI) in Matlab is to create a new window. For this purpose one will use the ***figure*** function. The syntax of the ***figure*** function is

```
h = figure('PropertyName',propertyvalue,...);
```

where *h* is the handle to the figure, and the '*PropertyName*', *propertyvalue* pairs customize the figure. The full list of figure properties can be consulted in the Matlab help.

To be noticed is that a numeric handle can also be assigned to the figure using

```
figure(h);
```

where *h* is an integer. The figure handle can be used later in the Matlab code so switch back to the desired window. Consider for exemplification:

```
Fig1=figure('Name','Lowpass Filter','NumberTitle','off');  % opens a window
                             % named 'Lowpass Filter' assigned to handle Fig1
…
Fig2=figure('Name','Highpass Filter','NumberTitle','off');  % opens a window
                             % named 'Highpass Filter' assigned to handle Fig2
…
figure(Fig1)                 % activates the window named 'Lowpass Filter'
```

Importation and display of images to Matlab

Consider the passive RC lowpass and highpass filters from figure 1.

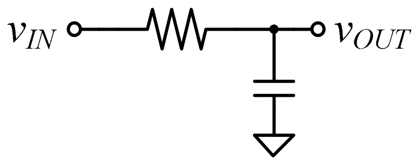


Figure 1

The electrical schematics is saved as a *.png* file, namely *lpf.png*.

Importation of an image file to Matlab is performed with the ***imread*** function. The syntax of the ***imread*** function is

```
A = imread(filename, fmt)
```

where *filename* specifies the image file name, provided it is saved in the current folder, or the complete path to the image file. Next *fmt* specifies the file format. The complete list of supportade image formats can be consulted in the Matlab help. To be noticed is that the *fmt* argument can be omitted in the ***imread*** call. Nevertheless, filename must contain the file extension.

In order to display an image in Matlab one will use the ***imshow*** function. The syntax of the ***imshow*** function is

```
imshow(A [, param, val])
```

where *A* is the image imported with the ***imread*** function or string stating the path and filename of the image to be displayed. Next, the [*param*, *val*] pairs specify a list of parameters and values to customize the image display [Matlab help] as follows:

Table 1

Parameter	Value
'Border'	`'tight'`, `'loose'`.
'Colormap'	m-by-3 matrix
'DisplayRange'	[low high]
'InitialMagnification'	numeric scalar, `'fit'`
'Parent'	axis handle
'Reduce'	logical value, only valid for TIFF images
'XData'	two-element vector
'YData'	two-element vector

Consider for exemplification:

```
function f = lowpassFilter

Fig=figure('Name','Lowpass Filter',...
'NumberTitle','off');

w = imread('lpf.png');
imshow(w, 'InitialMagnification', 120);
```

To be noticed is that, although the newly created window is used for image display, it behaves as a new image for function plots. Accordingly, it accepts a title, axis labels, etc. For exemplification, the image display window can further be customized with the addition of a title and some text.

```
title('Lowpass Filter');
xlabel('Cutoff frequency: fo = 1 / 2*\pi*R*C');
```

To be noticed is that the '*\pi*' string operates towards printing the π character.

The result displayed after calling the *lowpassFilter* function from the command line is illustrated in figure 2.

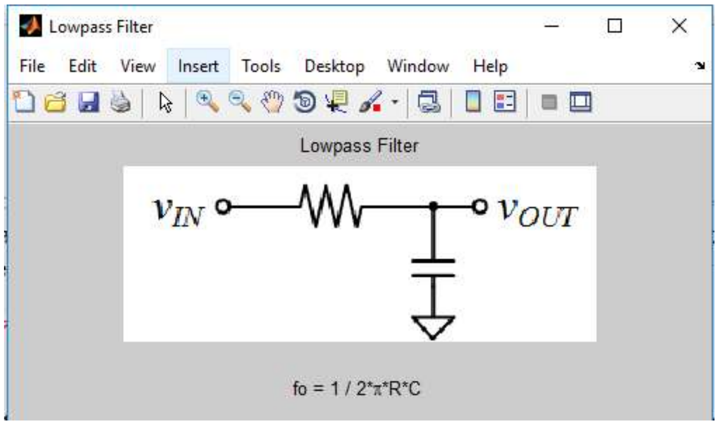


Figure 2

In this example, the schematic of the lowpass filter was saved with a 200x200 pixel resolution. This is rather small to be displayed in Matlab and consequently a 120% magnification was applied.

Finally, if closing the image is desired one will use the ***close*** function indicating the figure handle as argument, as for example

```
close(Fig);
```

Calling the ***close()*** function without any arguments will close the currently active window.

Importation and display of text files to Matlab

Matlab allows the importation of a series of text file formats, e.g.: *.txt*, *.doc*, *.docx*, etc. as well as web page formats, e.g. *.html*, etc. One will chose the format of the textfile or web page depending in the visual requirements of the project. Importation and display of a *.docx* file is considered further on for illustration.

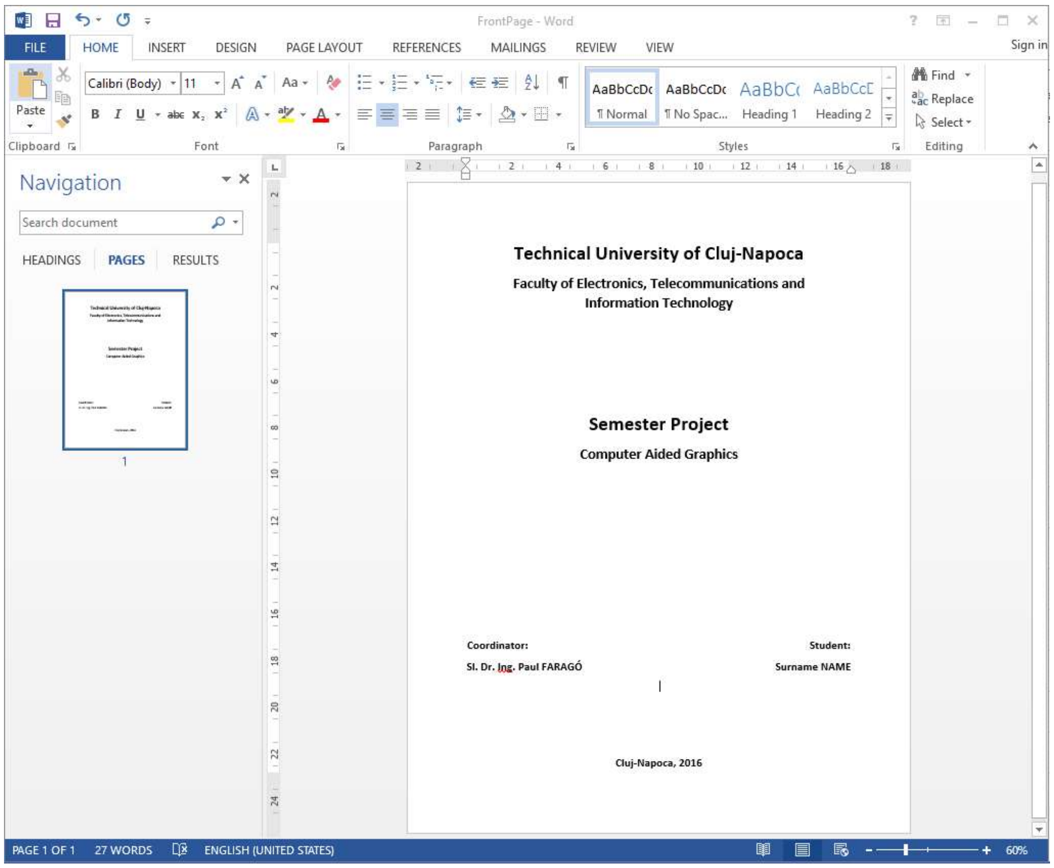
A project documentation must contain the following sections:

- Front page
- Table of contents
- Abstract
- Introduction
- Theoretical backgrounds

The documentation front page must contain the following elements:

- University name,
- Faculty name,
- Type of project (semester, annual, biannual, diploma thesys, etc.)
- Subject / Title
- Name and function of the coordinator
- Author’s name,
- Place and date

An example of a project front page is illustrated in figure 3.



Technical University of Cluj-Napoca

Faculty of Electronics, Telecommunications and Information Technology

Semester Project

Computer Aided Graphics

Coordinator:
Sl. Dr. Ing. Paul FARAGÓ

Student:
Surname NAME

Cluj-Napoca, 2016

Figure 3

Importation of a text file to Matlab is performed with the ***open*** function. The syntax of the ***open*** function is

```
T = open(name)
```

where *name* is a string indicating the text document filename, and is the document is not in the current folder *filename* also indicates the file path. In a similar manner, the ***open*** function can open several file types. The complete list of file types which can be opened with the ***open*** function can be consulted in the Matlab help.

Consider for exemplification:

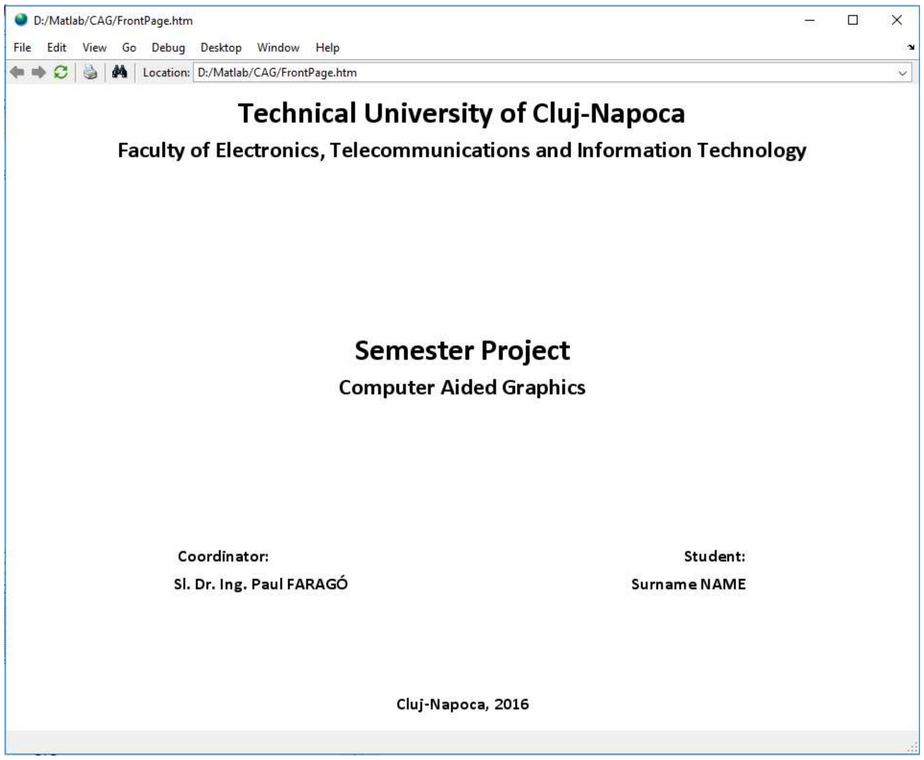
```
function frontPage
open('FrontPage.docx');
```

This function will open the *FrontPage.docx* file in MS Word, as illustrated in Figure 3. However, since the *.docx* file is opened in a different environment than Matlab, the ***close*** function will not close the text document.

Similarly, the front page can be saved in MS Word as a webpage with *.htm* extension. The Matlab function to open the web page is

```
function frontPage
open('FrontPage.htm');
```

This function will open the *FrontPage.htm* webpage in a new window in Matlab, as illustrated in Figure 4. Since the webpage was opened in Matlab, closing the file in this case is performed with the ***close*** function. Calling the ***close()*** function without any arguments will close the currently active window.


D:/Matlab/CAG/FrontPage.htm
File Edit View Go Debug Desktop Window Help
Location: D:/Matlab/CAG/FrontPage.htm

Technical University of Cluj-Napoca

Faculty of Electronics, Telecommunications and Information Technology

Semester Project

Computer Aided Graphics

Coordinator:

Sl. Dr. Ing. Paul FARAGÓ

Student:

Surname NAME

Cluj-Napoca, 2016

Figure 4

Creating user interface control elements

Control elements in Matlab are created using the ***uicontrol*** function. The syntax of the ***uicontrol*** function is

```
h = uicontrol(Name,Value,...)
```

where *h* is the handle to the control element, and the [Name, Value] pairs specify the properties of the control element. The complete list of control element properties can be consulted in the Matlab help. Nevertheless, the properties of interest in this laboratory are listed in Table 2.

Table 2

Name	Value
'Style'	string, type of control element
'Units'	string, interpretation of 'Position vector'
'Position'	four-element vector, coordinates of the control element
'String'	string, text to be displayed
'Callback'	string, action

The types of control elements which can be added on a Matlab GUI are:

- checkbox, edit, frame, listbox, popupmenu, pushbutton, radiobutton slider, text, togglebutton.

For ease of interpreting the control elements of the coordinates, the units will be specified 'normalized'. In this case, both horizontal and vertical dimensions of the display window will be considered unity, as illustrated in figure 6.

The four-element position vector [*hor vert dim_hor dim_vert*] indicates the position and dimensions of the control element as follows:

- *hor* and *vert* indicate the horizontal and vertical coordinates of the control element origin, i.e. the bottom-left corner of the control element,
- *dim_hor* and *dim_vert* indicate the horizontal and vertical dimensions of the control element.

Since the units are normalized, the four elements are given with subunit values, as illustrated in figure 6.

Finally, the callback consists of either Matlab code which states the actions to be taken when the control element is operated. Nevertheless, it is preferable to have the actions described in either a Matlab function or a Matlab script, and the callback should call the respective *.m* file.

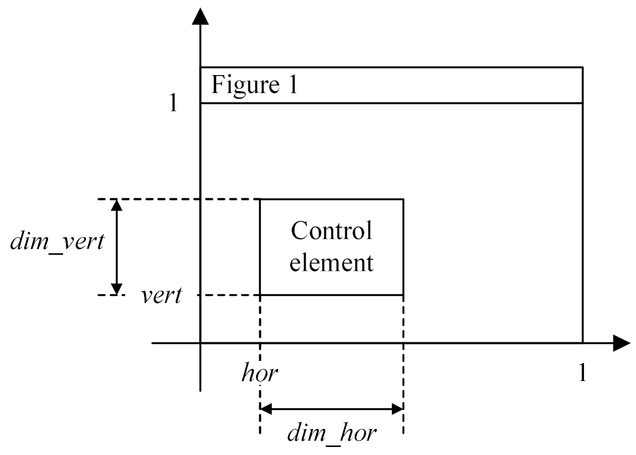


Figure 6

For exemplification, consider designing an application with a GUI to simulate the passive RC lowpass filter from figure 1. The simulator should have the following functionalities:

- a button to display the documentation front page,
- a button to display the electrical schematic of the filter,
- a button to display the time-domain response of the filter (to evaluate the time-domain response of the RC lowpass filter see Laboratory…),
- a button to display the frequency-domain response of the filter (to evaluate the frequency-domain response of the RC lowpass filter see Laboratory…),
- a button to close the GUI.

A possible main window of the LPF simulator is illustrated in figure 7.

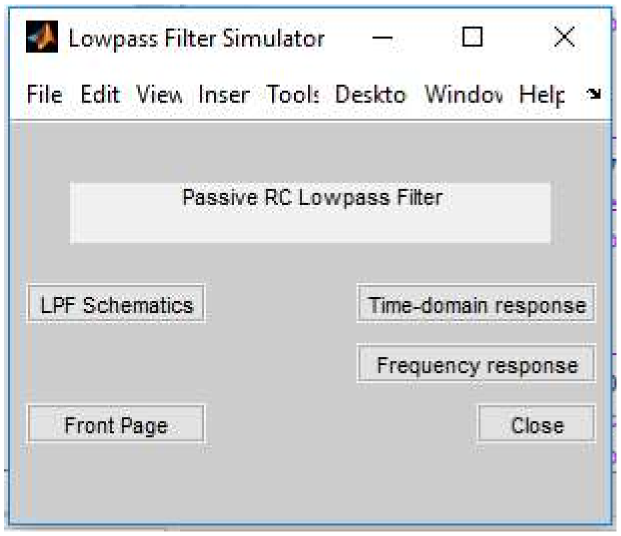


Figure 7

For illustration purpose, the Matlab code to create the GUI window, the application title and the pushbutton for the illustration of the RC filter schematics is listed as follows.

```
close all;
clear all;
clc;

Fig=figure('Name','Lowpass Filter Simulator',...
    'Position',[300 300 300 200],...
    'NumberTitle','off');

uicontrol('Style','text',...
    'Units','normalized',...
    'Position',[0.1 0.8 0.8 0.15],...
    'String','Passive RC Lowpass Filter');

uicontrol('Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.025 0.6 0.3 0.1],...
    'String','LPF Schematics',...
    'Callback','lowpassFilter;');
```

In a similar fashion, create the pushbuttons for illustration of the documentation front page, as well as the time-domain and frequency responses respectively. Additionally, consider adding a 'Close' button to each newly opened widow. For exemplification, a 'Close' button is added to the electrical schematics window in the *lowpassFilter.m* function with the following Matlab code:

```
uicontrol('Style','pushbutton',...
    'Units','normalized',...
    'Position',[0.775 0.05 0.2 0.1],...
    'String','Close',...
    'Callback','close();');
```

This changes the schematics illustration window as shown in figure 8.

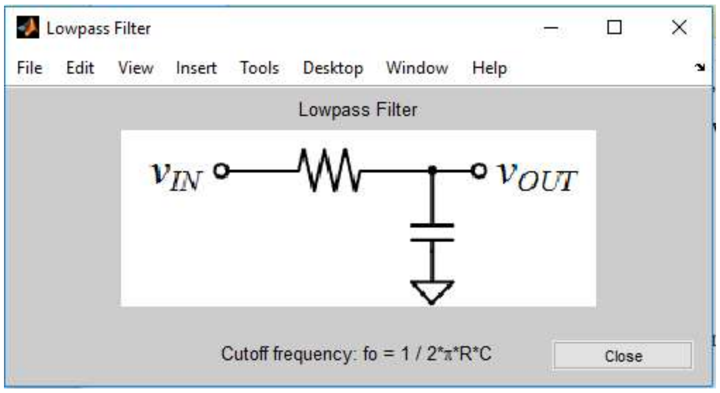

Lowpass Filter
File Edit View Insert Tools Desktop Window Help
Lowpass Filter
v_{IN}
v_{OUT}
Cutoff frequency: fo = 1 / 2*π*R*C
Close

Figure 8

Creating user interface menus

A new menu item in the figure window are created using the ***uimenu*** function. The syntax of the ***uimenu*** function is

```
h = uimenu([parent,]'Name',Value,...);
```

where *h* is the handle to the menu item, *parent* is the handle to a previously defined menu item, and the [*Name*, *Value*] pairs specify the properties of the menu item. The complete list of menu item properties can be consulted in the Matlab help. The properties of interest in this laboratory are listed in Table 1.

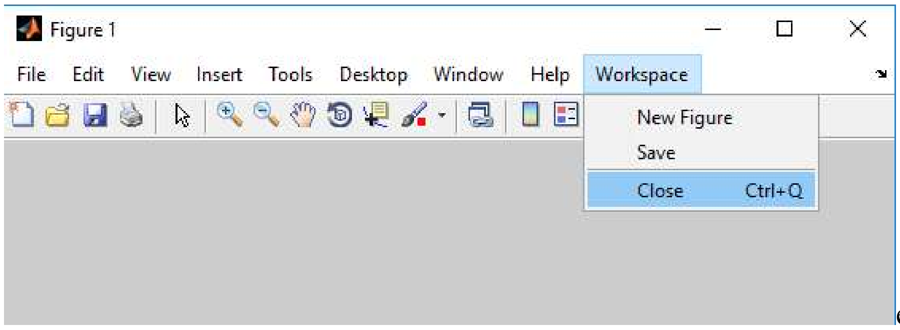
Table 1

Name	Value
'Label'	string, name of the menu item
'Separator'	string: 'on'/'off', separation line within the menu
'Accelerator'	character, keyboard shortcut for the menu item
'Callback'	string, action

If the ***uimenu*** function is called without the *parent* argument, it creates a new menu hierarchy in the menu bar of the window. The handle to the new menu is assigned to variable *h*, which is later used for creating menus and submenus in the menu hierarchy. Consider for exemplification:

```
figure();
h = uimenu('Label','Workspace');
uimenu(h,'Label','New Figure','Callback','figure');
uimenu(h,'Label','Save','Callback','save');
uimenu(h,'Label','Close','Callback','close',...
    'Separator','on','Accelerator','Q');
```

This code sequence adds a new menu item, namely "Workspace", to the Matlab figure as illustrated in figure 1.



e

Figure 1

The menu handle is assigned to variable *h*, which is later used to create the "New Figure", "Save" and "Close" menu elements. A horizontal line is drawn above the "Close" menu item by activating the *Separator* property. Additionally, the “Q” character is added as keyboard shortcut to the “Close” menu item within the *Accelerator* property. Finally, action of the menu elements is added in the *Callback* property as follows:

- ***figure*** – creates and opens a new Matlab widow,
- ***save*** – saves workspace variables to a file,
- ***close*** – closes the currently opened window.

A full description of the ***figure***, ***save*** and ***close*** functions can be found in the Matlab help, and thus the functionality of the code sequence listed above can be customized according to the GUI designer’s needs.

In order to create submenus, the menus require dedicated handles. Consider for exemplification changing the definition of the “Save” menu item as follows:

```
h1 = uimenu(h,'Label','Save');
uimenu(h1,'Label','Save R','Callback','save(''R.mat'',''R'')');
uimenu(h1,'Label','Save L','Callback','save(''R.mat'',''R'')');
uimenu(h1,'Label','Save All','Callback','save');
```

Accordingly, the Menu bar changes as illustrated in figure 2

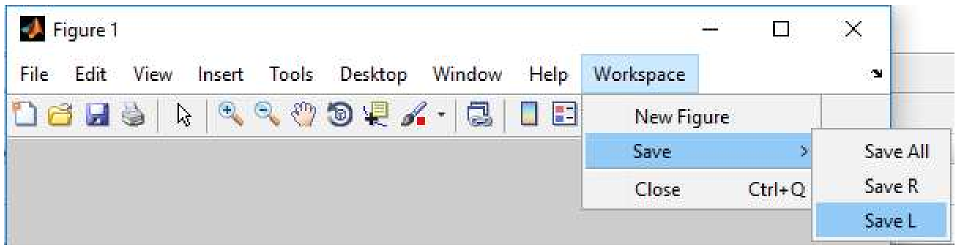


Figure 2

Applications

Exercise 1. Design an application with a GUI to simulate the passive RC highpass filter from figure 9. The simulator should have the following functionalities:

- a button to display the documentation front page,
- a button to display the electrical schematic of the filter,
- a button to display the time-domain response of the filter,
- a button to display the frequency-domain response of the filter,
- a button to close the GUI.

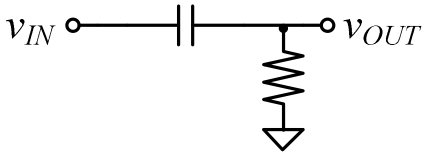


Figure 9

Exercise 2. Design an application with a GUI to simulate the passive RL lowpass filter from figure 3. The simulator should have the following functionalities:

- two plots on the same window: one for the frequency response and one for the time-domain response,
- a new menu "Documentation" which opens each document section separately: Front page, Table of contents, Abstract, Introduction, Theoretical Backgrounds, etc.,
- a new menu "Parameters" which:
    - reads the RL filter parameter values in a new window (use *Edit* type control elements using the ***uicontrol*** function, as studied in laboratory …),
    - saves the frequency and time-domain responses in *.mat* files,
    - opens previously saved frequency and time-domain responses,
- a button to display the electrical schematic of the filter,
- a button to plot the RL filter frequency and time-domain responses with the newly read parameter values.

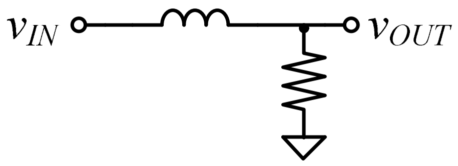


Figure 3

A possible main window of the LPF simulator is illustrated in figure 4.

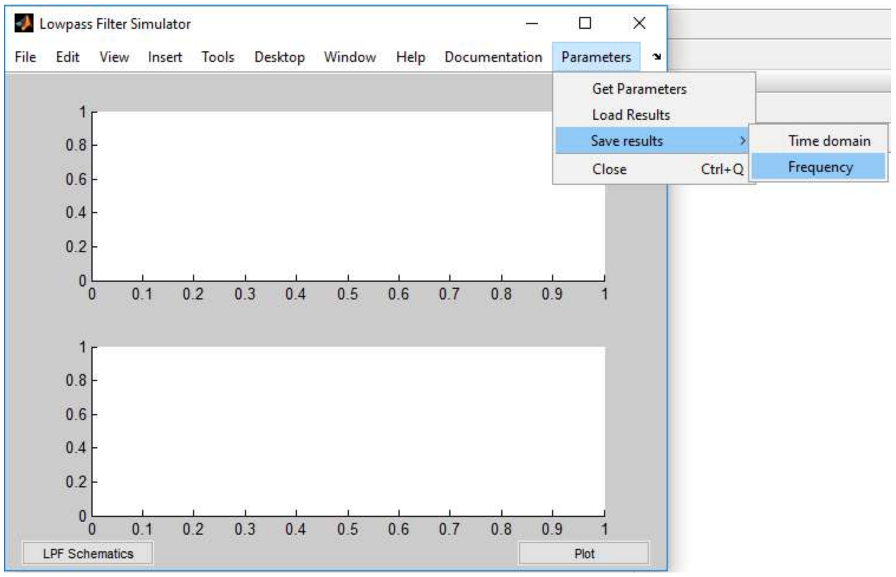


Figure 4