TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

FACULTY OF ELECTRONICS, TELECOMMUNICATIONS AND INFORMATION TECHNOLOGY

VOLTAGE AND CURRENT DIVIDERS IN RLC CIRCUITS

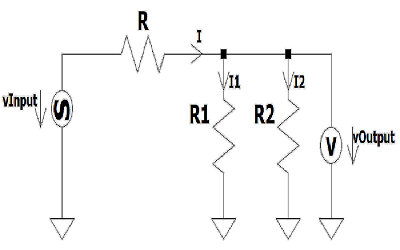
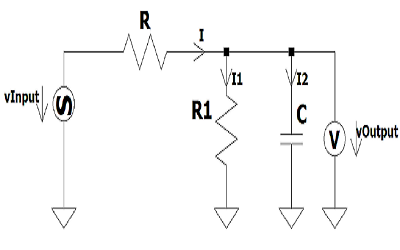
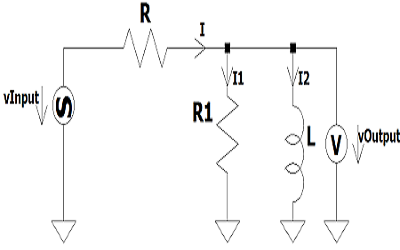
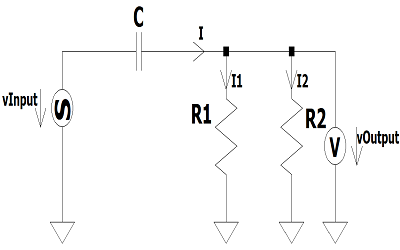
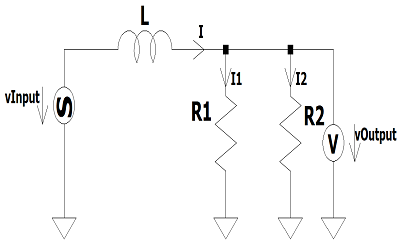
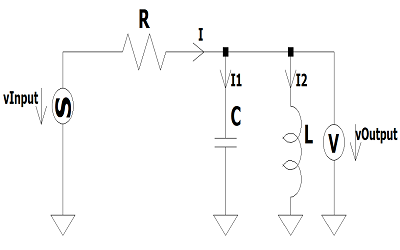
Mircică Tudor Coordinators:

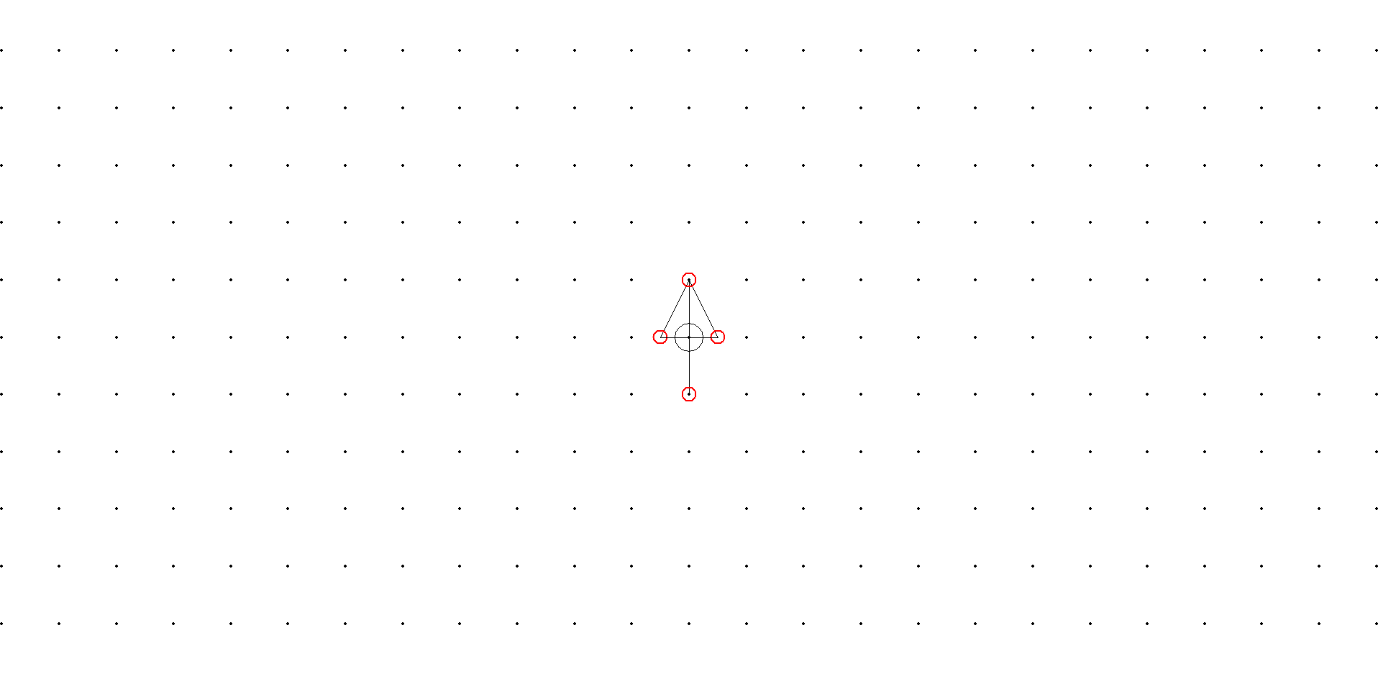
Group: e\_2024 Cirlugea Mihaela Terhes Diana

Description of the theme:

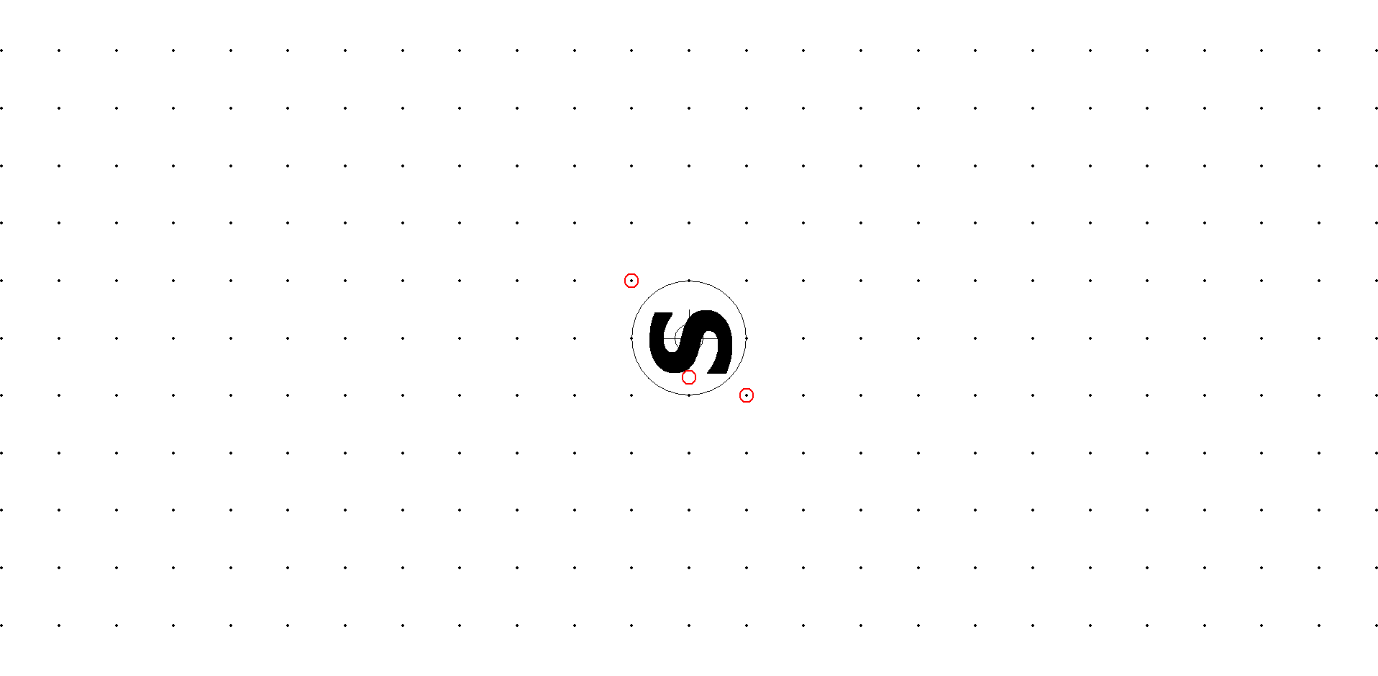
The MatLab application simulates the behavior of different circuits containing resistors, as well as capacitors and coils. The user can size the components of the chosen circuit and compute, using custom scripts, the currents and voltages from it by a single push of a button.

List of figures:

* Figure 1: Circuit with 3 resistors (output voltage on R2)
* Figure 2: Circuit with 2 resistors and a capacitor (output voltage on C)
* Figure 3: Circuit with 2 resistors and a coil (output voltage on L)
* Figure 4: Circuit with a capacitor and 2 resistors (output voltage on R2)
* Figure 5: Circuit with a coil and 2 resistors (output voltage on R2)
* Figure 6: Circuit with a resistor, a capacitor and a coil (output voltage on L)
* Arrow symbol – designed in LtSpice (used for specifying directions of voltages and currents)



* Alternating voltage source – designed in LtSpice



* Voltmeter – designed in LtSpice (used to show to output voltage)

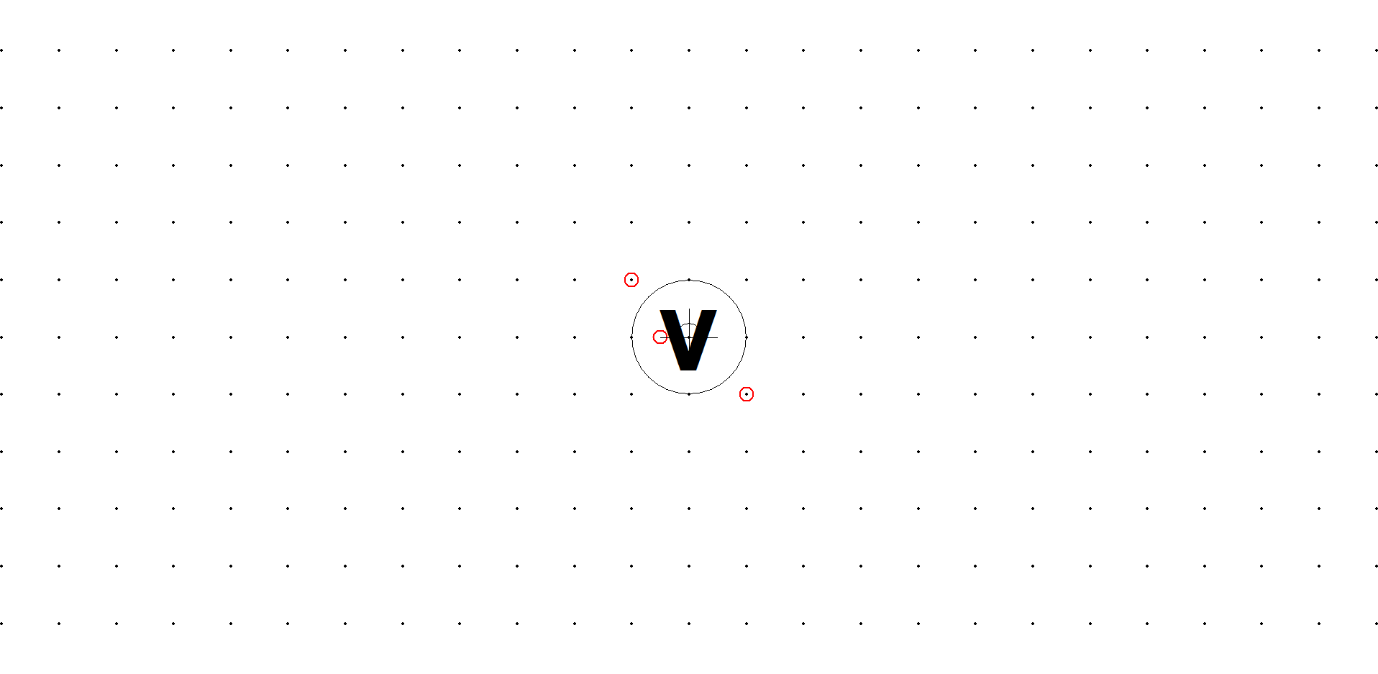


Table of contents:

1. Abstract page / Keywords
2. Introduction
   1. History

2.2. What was already done and discoverd

2.3. Where my project is situated

3. Theoretical approach

4. Practical results

4.1. My procedures

4.2. Methods used

4.3 My calculus

4.4. My results

5. Conclusion

6. References/Bibliography

7. Appendix

Goal:

* Design various RLC circuits and calculate the voltages and currents through different components, after sizing them;

Personal report:

I had to create an application in MatLab representing a graphical user interface. I chose to draw a few simple circuits which contain simple electronic components, such as resistors, capacitors and inductors in LtSpice. I managed to compute the output voltages and currents on both parallel branches of each cirtuit, when the input voltage was given. This results were calculated using the formulas for current and voltage dividers, known from other subjects studied before. I personally consider that the goal of this project was fulfilled, but there is still room for improvement, given the simplicity of the theme.

Keywords:

* Resisitor
* Capacitor
* Coil/Inductor
* Voltage
* Input/Output
* Circuit

Introduction

* Histoty:

The idea of finding the output voltages from such circuits is inspired by the Signals and Systems seminaries, but most of the figures are personal creations. I chose this theme because its diversity and as I found it easy to implement .

* What was already done and discoverd

At this point, the formulas for voltage and current dividers are already well-known, but need to be personalized for each specific circuit. Moreover, I have been familiarised with the MatLab Graphical User Interface from previous laboratories.

* Where my project is situated

When I started working on this project, all the theoretical aspects had been already presented to me. I just had to apply the known formulas for each individual circuit, depending on its components. I consider this project to be an application of what we have learnt during this semester.

Theoretical approach

* Description of all the elements in the project:
* MatLab - MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.
* GUI - The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.
* Voltage source - A voltage source is a two-terminal device which can maintain a fixed voltage. An ideal voltage source can maintain the fixed voltage independent of the load resistance or the output current. However, a real-world voltage source cannot supply unlimited current. A voltage source is the dual of a current source.
* Voltmeter - A voltmeter is an instrument used for measuring electrical potential difference between two points in an electric circuit.
* Resistor - A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses.
* Capacitor – A capacitor is a device that stores electrical energy in an electric field. It is a passive electronic component with two terminals. The effect of a capacitor is known as capacitance.
* Coil/inductor -An electromagnetic coil is an electrical [conductor](https://en.wikipedia.org/wiki/Electrical_conductivity) such as a [wire](https://en.wikipedia.org/wiki/Wire" \o "Wire) in the shape of a [coil](https://en.wiktionary.org/wiki/coil" \o "wiktionary:coil), [spiral](https://en.wikipedia.org/wiki/Spiral) or [helix](https://en.wikipedia.org/wiki/Helix).

Practical results

* My procedures

Project\_Mircica.m

Fig6.m

Fig1.m

Fig5.m

Fig2.m

Fig3.m

Fig4.m

* + Methods used:

1. Voltage divider
2. Current diver
3. Ohm’s law
4. Kirchhoff’s laws

- My calculus

1.

a = sin(w\*t);

vI= A\*a;

Rp=(R1\*R2)/(R1+R2);

I = vI/(R+Rp);

i1 = (R2/(R1+R2))\*I;

i2 = (R1/(R1+R2))\*I;

vo = (Rp/(R+Rp))\*vI;

vO = vo/a;

I1 = i1/a\*1000;

I2 = i2/a\*1000;

2.

z1 = R1/(1+R1^2\*C^2\*w^2);

z2 = -(R1^2\*C\*w)/(1+R1^2\*C^2\*w^2);

Z = complex(z1,z2);

vo = (Z/(R+Z))\*vI;

vO = vo/a;

I = vO/(R+Z)\*1000;

I1 = vO/R1\*1000;

I2 = I-I1;

3.

z1 = (R1\*L^2\*w^2)/(R1^2+w^2\*L^2);

z2 = (R1^2\*w\*L)/(R1^2+w^2\*L^2);

Z = complex(z1,z2);

vo = (Z/(R+Z))\*vI;

vO = vo/a;

I = vO/(R+Z)\*1000;

I1 = vO/R1\*1000;

I2 = I-I1;

4.

Z = (R1\*R2)/(R1+R2);

z1 = (w^2\*C^2\*Z^2)/(1+w^2\*Z^2\*C^2);

z2 = (w\*Z\*C)/(1+w^2\*Z^2\*C^2);

Zech = complex(z1,z2);

vo = Zech\*vI;

vO = vo/a;

I1 = (vO/R1)\*1000;

I2 = (vO/R2)\*1000;

5.

Z = (R1\*R2)/(R1+R2);

z1 = (Z^2)/(Z^2+w^2\*L^2);

z2 = (w\*L\*Z)/(Z^2+w^2\*L^2);

Zech = complex(z1,z2);

vo = Zech\*vI;

vO = vo/a;

I1 = (vO/R1)\*1000;

I2 = (vO/R2)\*1000;

6.

Z = (w\*L)/(1-w^2\*L\*C);

Z = complex(0,z);

vo = z/(R+Z)\*vI;

vO = vo/a;

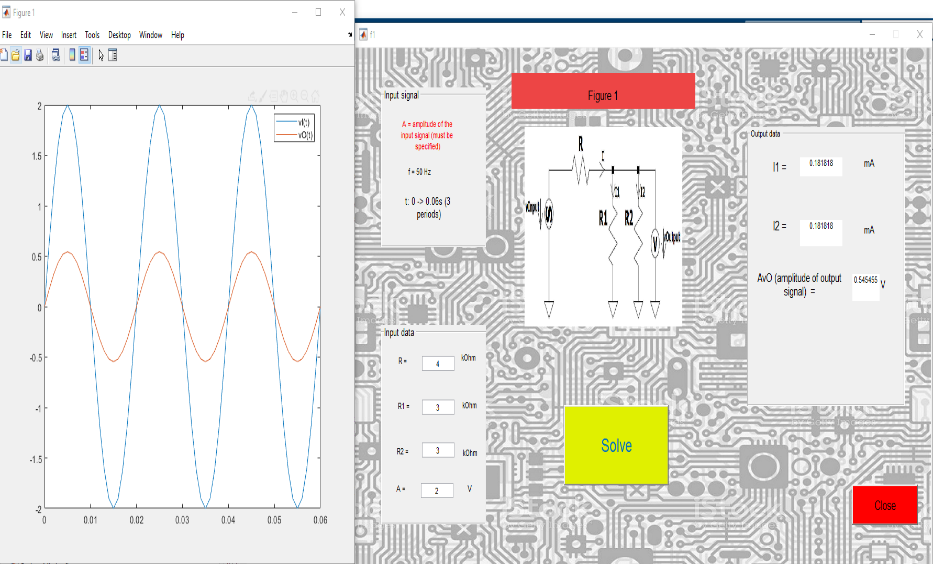
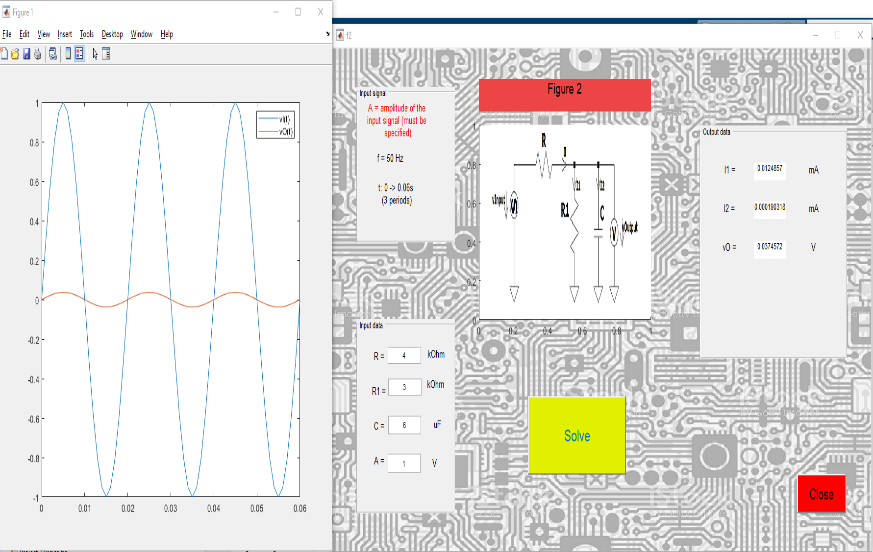
i = vo/(R+Z);

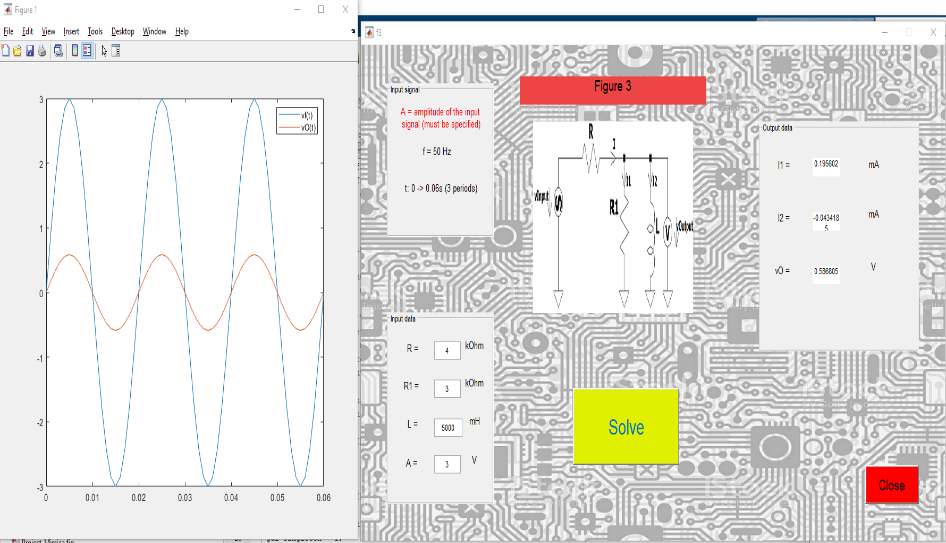
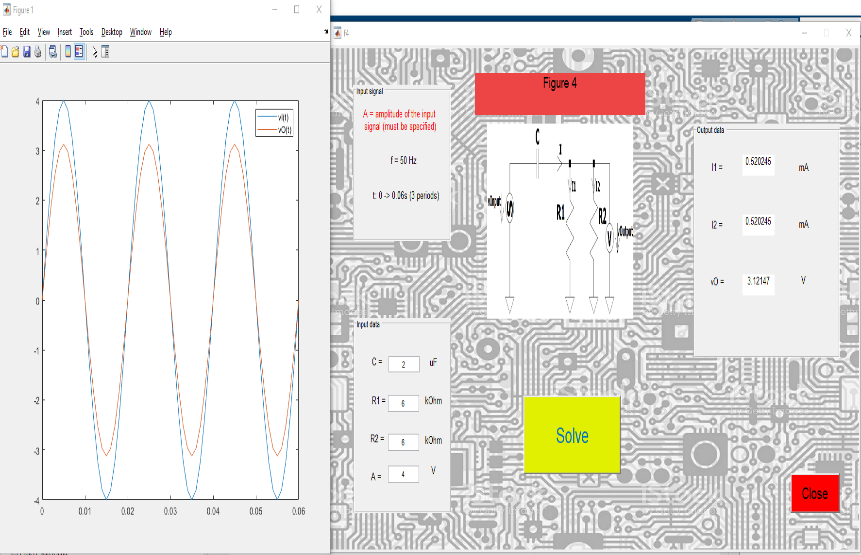
I = i/a;

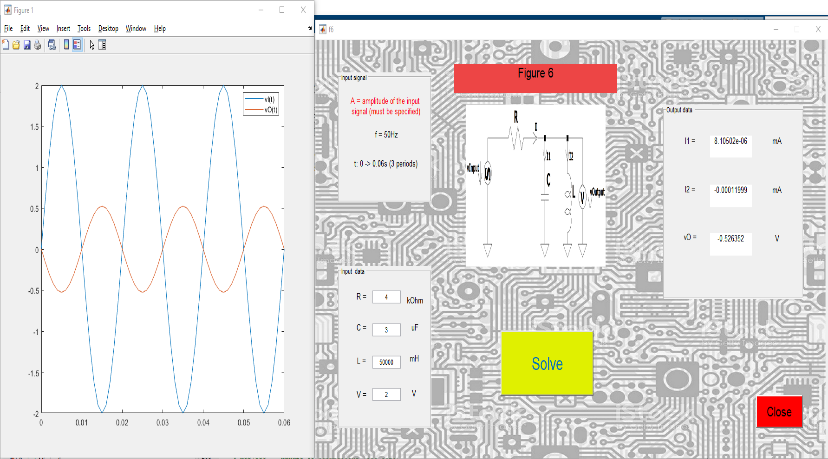
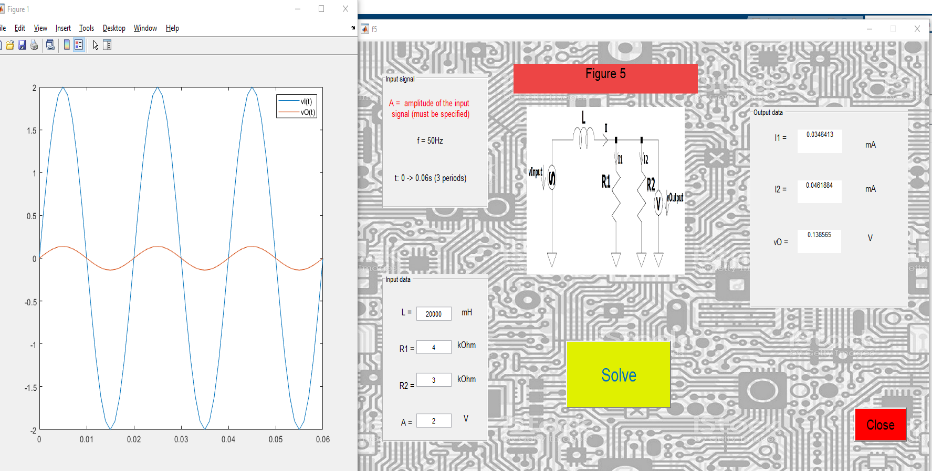
I1 = 1/(1-w^2\*L\*C)\*I;

I2 = I-I1;

- My results







Conclusion

I personally consider that this project reached its goal. Even though at first, I found it difficult to familiriaze with the MatLab GUI, I managed to create an application based on what we have learned throughout the semester.

* Results obtained:
* Mostly good
* At some point, a couple of errors have occured, but I was able to get rid of them
* Future work:
* Plots for currents through each branch
* More complex circuits

References/Bibliography

* MatLab help/documentation

<https://www.mathworks.com/help/matlab/>

* CAG laboratories

<http://www.bel.utcluj.ro/ci/eng/cag/index.html?fbclid=IwAR3-mEYFdl_DEM_WOPy92JQrW39dE7h9MHJ0eu1VBkwTdIlssQ3GEYt9oRA>

* Random Indian dudes on Youtube

Appendix – all programming code

1.Project\_Mircica.m

function varargout = Project\_Mircica(varargin)

% PROJECT\_MIRCICA MATLAB code for Project\_Mircica.fig

% PROJECT\_MIRCICA, by itself, creates a new PROJECT\_MIRCICA or raises the existing

% singleton\*.

%

% H = PROJECT\_MIRCICA returns the handle to a new PROJECT\_MIRCICA or the handle to

% the existing singleton\*.

%

% PROJECT\_MIRCICA('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in PROJECT\_MIRCICA.M with the given input arguments.

%

% PROJECT\_MIRCICA('Property','Value',...) creates a new PROJECT\_MIRCICA or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before Project\_Mircica\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to Project\_Mircica\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Project\_Mircica

% Last Modified by GUIDE v2.5 03-Jan-2020 13:11:33

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Project\_Mircica\_OpeningFcn, ...

'gui\_OutputFcn', @Project\_Mircica\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before Project\_Mircica is made visible.

function Project\_Mircica\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to Project\_Mircica (see VARARGIN)

% Choose default command line output for Project\_Mircica

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes Project\_Mircica wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = Project\_Mircica\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.

function axes1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ1.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes1

% --- Executes during object creation, after setting all properties.

function axes2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ2.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes2

% --- Executes during object creation, after setting all properties.

function axes3\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ3.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes3

% --- Executes during object creation, after setting all properties.

function axes4\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ4.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes4

% --- Executes during object creation, after setting all properties.

function axes5\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ5.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes5

% --- Executes during object creation, after setting all properties.

function axes6\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes6 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('circ6.png');

imshow(w);

% Hint: place code in OpeningFcn to populate axes6

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

f1();

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

f2();

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

f3();

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

f4();

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

f5();

% hObject handle to pushbutton5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton4.

function pushbutton6\_Callback(hObject, eventdata, handles)

f6();

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hint: place code in OpeningFcn to populate axes8

% --- Executes on button press in pushbutton7.

function pushbutton7\_Callback(hObject, eventdata, handles)

close();

% hObject handle to pushbutton7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.

function axes7\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

w = imread('back.jpg');

imshow(w);

% Hint: place code in OpeningFcn to populate axes7

% --- Executes on button press in pushbutton8.

function pushbutton8\_Callback(hObject, eventdata, handles)

open('documentation\_mircica.docx');

% hObject handle to pushbutton8 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)