Bangladesh University of Engineering and Technology

Department of Electrical and Electronic Engineering

Course no: EEE 212

Course Title: Numerical Technique Laboratory

Report on Project

Project Title:

Transient Analysis:

Determining a Circuit and Circuit Parameters

From a set of given Data Points

Submitted to:

I.K.M. Reaz Rahman

Lecturer,Department of EEE,BUET

Rajat Chakraborty

Lecturer,Department of EEE,BUET

Submitted By:

Name: Saleh Ahmed Khan Student ID: 1706053

Name: Tiasa Mondal Student ID: 1706054

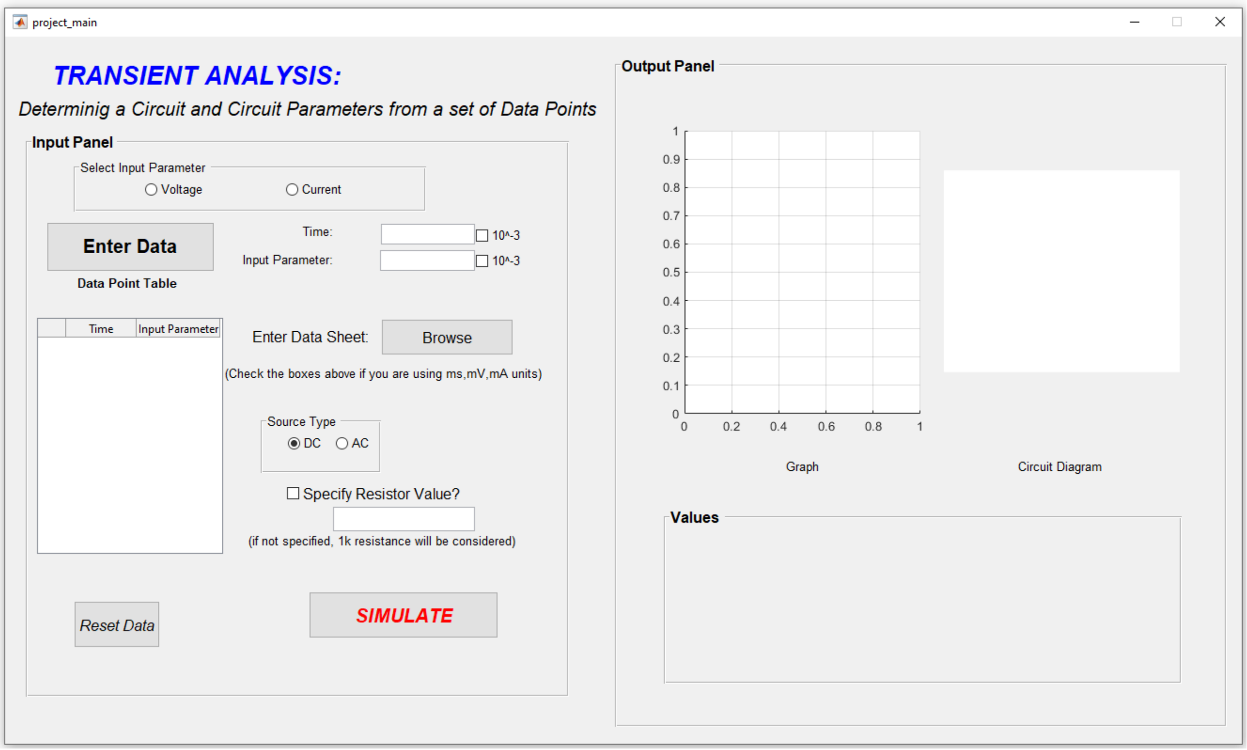
Section: A2

Date of Submission: 18/9/2019

The main goals of the project:

- Plot a voltage/current vs time graph from the given data points using proper methods (curve fitting, spline, cubic interpolation, user-defined functions)
- Determine specific circuit case (like source free dc RLC, ac RL etc.)
- Find the value of circuit elements
- Show a picture of that type of circuit
- We have taken input manually or via text files (as per choice) using GUI and shows the plot and circuit diagram on the same GUI window.

Graphical User Interface for the Project:



TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

(Check the boxes above if you are using ms,mV,mA units)

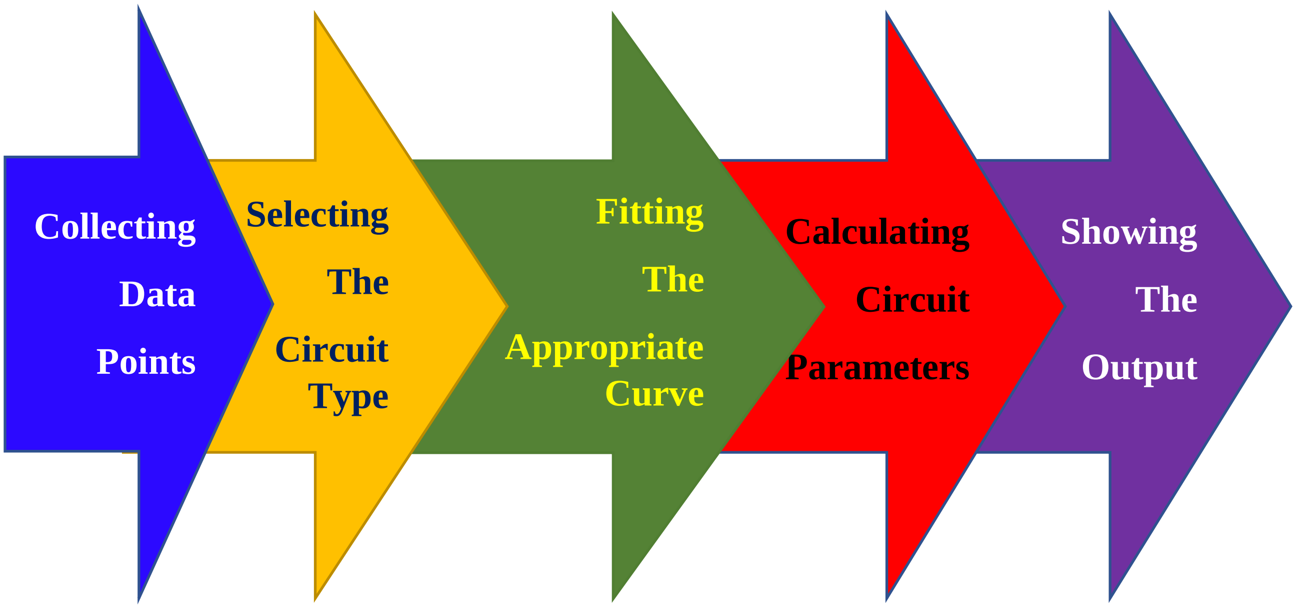
Source Type: ● DC ○ AC

☐ Specify Resistor Value?

(if not specified, 1k resistance will be considered)

Figure 1: Outlook of the GUI Window

Process of Calculation


Collecting Data Points
Selecting The Circuit Type
Fitting The Appropriate Curve
Calculating Circuit Parameters
Showing The Output

<u>Case 1:</u> DC, Discharging Capacitor/Inductor

Capacitor and Inductor discharges exponentially, therefore simply fitting an exponential curve to the data points yields the values of initial condition (V_0/I_0) and time constant (τ). From them the circuit parameters can be calculated.

$v_c(t)=V_0\exp(-t/\tau),\ \tau=RC$

$i_L(t)=I_0\exp(-t/\tau),\ \tau=L/R$

<u>Case 2:</u> DC, Charging Capacitor/Inductor

Capacitor and Inductor charges exponentially as well, but can't be directly fitted using an exponential curve.

$v_c(t)=V_{inf} + (V_0-V_{inf})\exp(-t/\tau),\ \tau=RC$

$i_L(t)= I_{inf} + (I_0-I_{inf})\exp(-t/\tau),\ \tau=L/R$

Only the second portion has an exponential part so the data must be manipulated to match that curve, this is done by

1. Subtracting the steady-state part from data points.

2. Multiplying with a minus to make the graph positive.

Now the data points can be fitting to an exponential model to obtain (V_0-V_{inf}) and time constant and other values can be calculated accordingly.

Case 3: AC, Simple RC/RL circuit

AC transient equations have two parts, one is an exponential part and the other is a forced response. If a graph can be obtained using any of the curve fitting or interpolation methods, it is possible obtain the coefficients of the exponential and steady state parameters from it. The program uses Fourier curve fitting and in some cases, Cubic Spline method to obtain sinusoidal waveforms.

$$i_L(t)= \frac{Em}{\sqrt{R^2+X^2}} \sin(wt+\lambda-\theta) - \frac{Em}{\sqrt{R^2+X^2}}\sin(\lambda-\theta)\ e^{-Rt/L} \quad \text{(AC RL)}$$

$$v_c(t)= -\frac{Em}{w\sqrt{R^2+X^2}} \cos(wt+\lambda+\theta) + \frac{Em}{w\sqrt{R^2+X^2}}\cos(\lambda+\theta)\ e^{-t/RC} \quad \text{(AC RC)}$$

Once a graph has been fitted, we can take the positions of the maxima and minima from the graph and figure out their midpoint which is actually data points for the exponential plot, curve fitting them with an exponential model gives us with the coefficients of the exponential part of the equation.

Also, by using the peaks we can calculate the Amplitude and frequency of the periodic part of the equation.

Once these values have been obtained, we can calculate the circuit parameters.

Case 4: DC, RLC series circuit

1. Underdamped Case:

   The Underdamped equation has a sinusoidal part multiplied to an exponential part. First, the graph is obtained using cubic spline interpolation.

   $i(t)= \exp(-s_1t)(A_1\sin(wt) + A_2\cos(wt))$

   It is possible to determine w using the maxima and minima of the fitted graph, also we can also determine the exponential part be curve fitting either the maxima or the minima of the graph.

   Once s_1, w has been found other circuit parameters can be determined from them.

2. Overdamped Case:

   The equation is the linear combinationof two exponential curves.

   $i(t)= A_1\exp(-s_1t) + A_2\exp(-s_2t)$

   Using the MATLAB fit function for two exponential terms, we can determine s_1, s_2 and from them it is possible to calculate the circuit parameters.

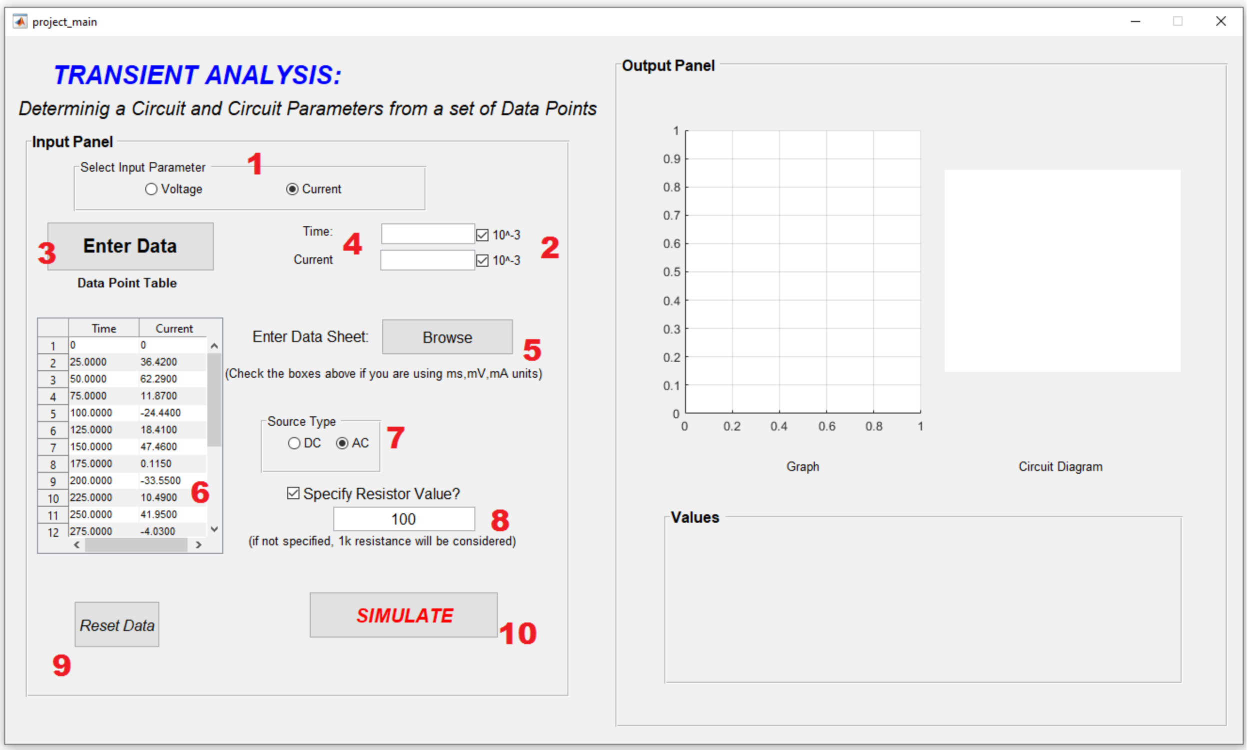
3. Critically Damped Case:

   $i(t)= A_1\exp(-s_1t) + A_2t\exp(-s_1t)$

   In critically damped case, only s_1 is required to calculate the circuit parameters, it is also fitted using the fit function.

Instruction for the Users

INPUT:



TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

Input Panel

Select Input Parameter: ○ Voltage ● Current

Time: ☑ 10^-3

Current ☑ 10^-3

Data Point Table

	Time	Current
1	0	0
2	25.0000	36.4200
3	50.0000	62.2900
4	75.0000	11.8700
5	100.0000	-24.4400
6	125.0000	18.4100
7	150.0000	47.4600
8	175.0000	0.1150
9	200.0000	-33.5500
10	225.0000	10.4900
11	250.0000	41.9500
12	275.0000	-4.0300

Enter Data Sheet: Browse

(Check the boxes above if you are using ms,mV,mA units)

Source Type: ○ DC ● AC

☑ Specify Resistor Value?

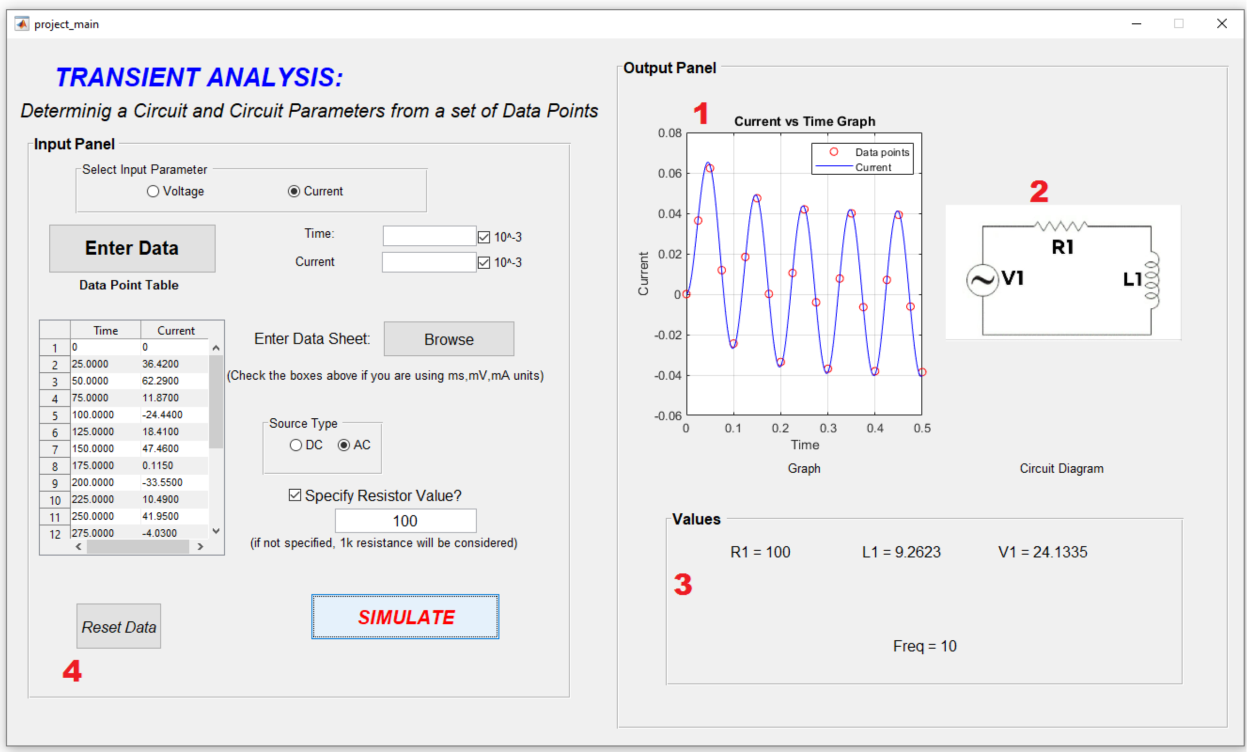
100

(if not specified, 1k resistance will be considered)

Figure 2: Input Panel

1. Select the type of you input variable. (Voltage/Current)
2. Check the boxes if you are entering values in millisecond, millivolt or milliampere Units.
3. Use the Enter Data button to input data into the Table.
4. Type in the Data Point you want to use in the Program
5. Alternatively, you can also enter a text file containing your previously collected Data. Select the Browse Option and select the .txt file.
6. Your Data will be displayed in the Table.
7. Select the type of your source and therefore the type of your analysis. (AC/DC)
8. Specific value of at least one circuit element is required to calculate others. Provide a suitable value for the resistor for your circuit. (Otherwise the program will calculate using a 1 k ohm resistor
9. Use the Reset Data button to erase previous calculations or to Clear the table is you entered a wrong value.
10. Use the “Simulate” button to see the results.

OUTPUT:



project_main

TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

Input Panel

Select Input Parameter: ○ Voltage ● Current

Time: ☑ 10^-3

Current ☑ 10^-3

Data Point Table

	Time	Current
1	0	0
2	25.0000	36.4200
3	50.0000	62.2900
4	75.0000	11.8700
5	100.0000	-24.4400
6	125.0000	18.4100
7	150.0000	47.4600
8	175.0000	0.1150
9	200.0000	-33.5500
10	225.0000	10.4900
11	250.0000	41.9500
12	275.0000	-4.0300

Enter Data Sheet: Browse

(Check the boxes above if you are using ms,mV,mA units)

Source Type: ○ DC ● AC

☑ Specify Resistor Value? 100

(if not specified, 1k resistance will be considered)

Output Panel

Values

R1 = 100 L1 = 9.2623 V1 = 24.1335

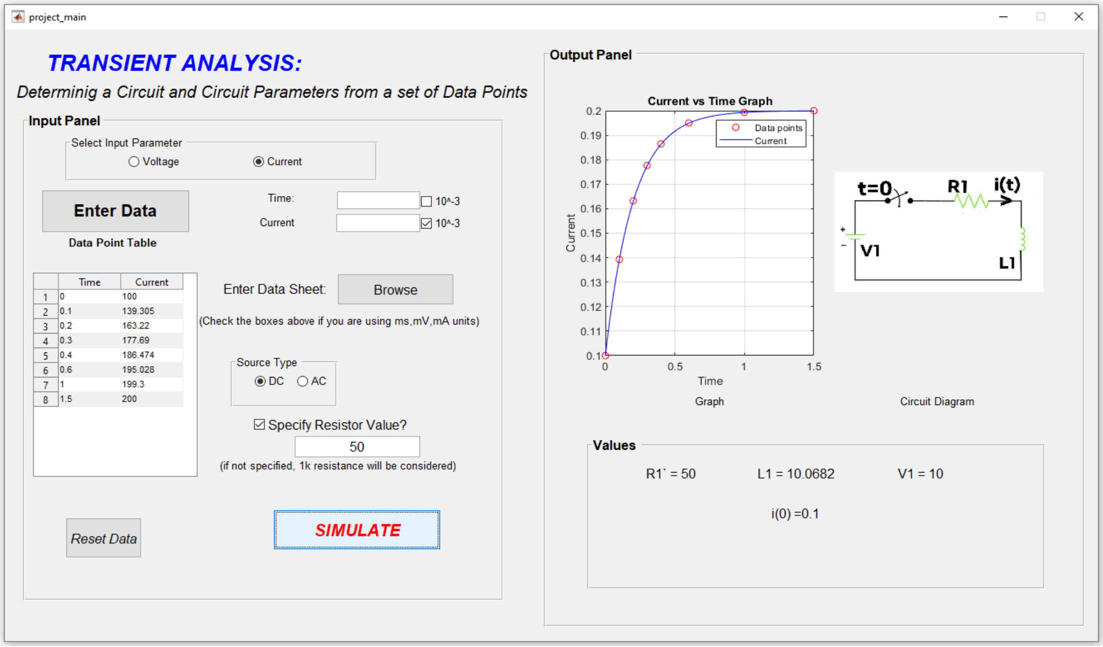
Freq = 10

Figure 3: Output Panel

1. A suitable graph plotted using your data point will be shown here.
2. A suitable circuit corresponding to your data will be shown here.
3. Calculated values of the Circuit parameters will be shown in the box.
4. Use reset data button to clear all previous calculations.

Examples

1. DC first order circuit (RC/RL):



TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

Input Panel

Select Input Parameter: ○ Voltage ● Current

Time: □ 10^-3

Current: ☑ 10^-3

Data Point Table

	Time	Current
1	0	100
2	0.1	139.305
3	0.2	163.22
4	0.3	177.69
5	0.4	186.474
6	0.6	195.028
7	1	199.3
8	1.5	200

Enter Data Sheet:

(Check the boxes above if you are using ms,mV,mA units)

Source Type: ● DC ○ AC

☑ Specify Resistor Value?

50

(if not specified, 1k resistance will be considered)

Output Panel

Graph

Circuit Diagram

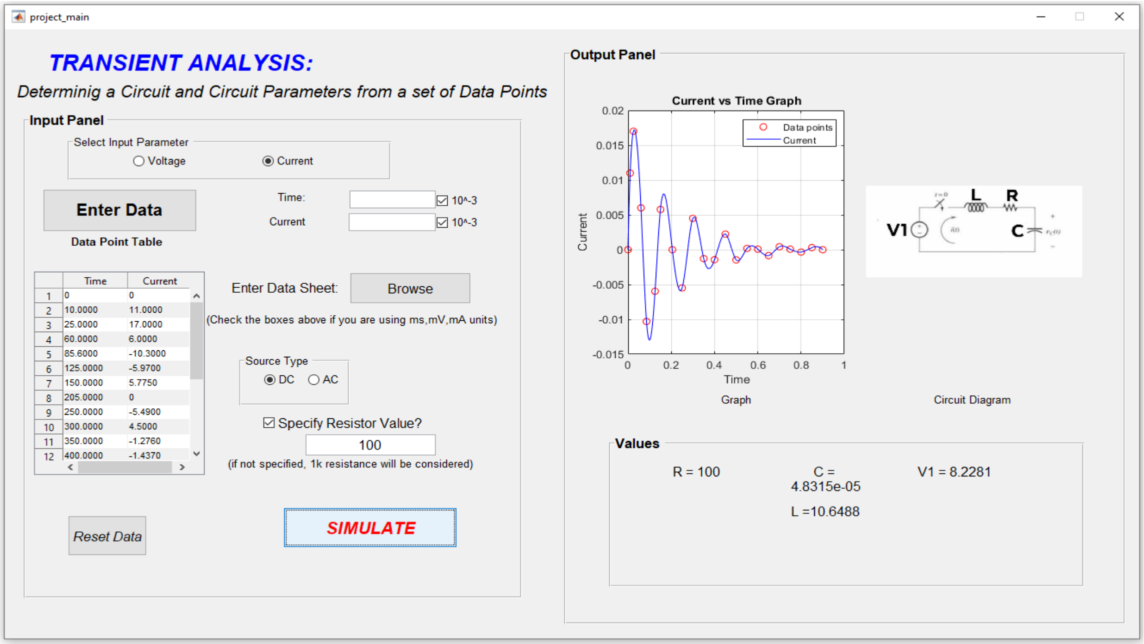
Values

R1` = 50 L1 = 10.0682 V1 = 10

i(0) =0.1

Figure 4

2. DC second order circuit (RLC):



TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

Input Panel

Select Input Parameter: ○ Voltage ● Current

Time: ☑ 10^-3

Current: ☑ 10^-3

Data Point Table

	Time	Current
1	0	0
2	10.0000	11.0000
3	25.0000	17.0000
4	60.0000	6.0000
5	85.6000	-10.3000
6	125.0000	-5.9700
7	150.0000	5.7750
8	205.0000	0
9	250.0000	-5.4900
10	300.0000	4.5000
11	350.0000	-1.2760
12	400.0000	-1.4370

Enter Data Sheet:

(Check the boxes above if you are using ms,mV,mA units)

Source Type: ● DC ○ AC

☑ Specify Resistor Value?

100

(if not specified, 1k resistance will be considered)

Output Panel

Graph

Circuit Diagram

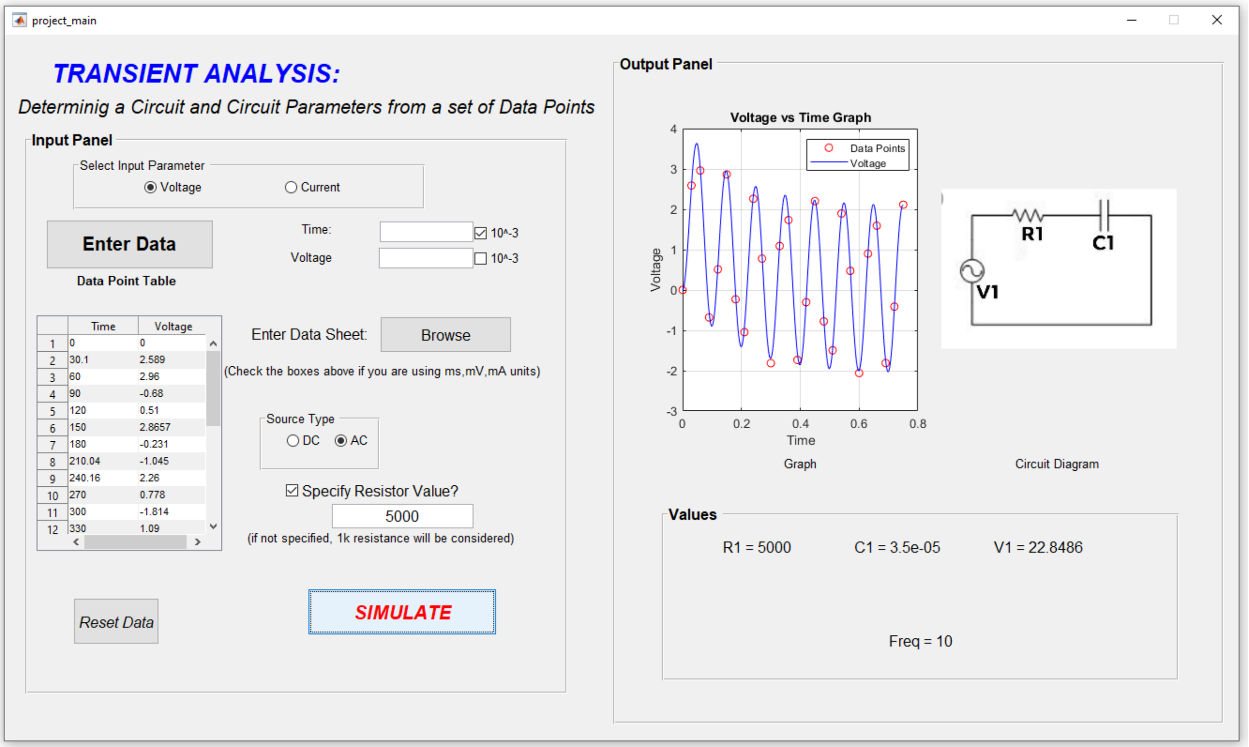
Values

R = 100 C = 4.8315e-05 V1 = 8.2281

L =10.6488

Figure 5

3. AC transient circuit (RC/RL):



TRANSIENT ANALYSIS:

Determinig a Circuit and Circuit Parameters from a set of Data Points

Input Panel

Select Input Parameter: ● Voltage ○ Current

Time: ☑ 10^-3

Voltage ☐ 10^-3

Data Point Table

	Time	Voltage
1	0	0
2	30.1	2.589
3	60	2.96
4	90	-0.68
5	120	0.51
6	150	2.8657
7	180	-0.231
8	210.04	-1.045
9	240.16	2.26
10	270	0.778
11	300	-1.814
12	330	1.09

Enter Data Sheet:

(Check the boxes above if you are using ms,mV,mA units)

Source Type: ○ DC ● AC

☑ Specify Resistor Value?

5000

(if not specified, 1k resistance will be considered)

Values

R1 = 5000 C1 = 3.5e-05 V1 = 22.8486

Freq = 10

Figure 6

The Data points provided here are taken from Pspice plot.

The Results match with the circuit parameters used in Pspice Simulation

Appendix

Main Function and G.U.I script:

```
function varargout = project_main(varargin)

% PROJECT_MAIN MATLAB code for project_main.fig
%      PROJECT_MAIN, by itself, creates a new PROJECT_MAIN or raises the
existing
%      singleton*.
%
%      H = PROJECT_MAIN returns the handle to a new PROJECT_MAIN or the
handle to
%      the existing singleton*.
%
%      PROJECT_MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJECT_MAIN.M with the given input
arguments.
%
%      PROJECT_MAIN('Property','Value',...) creates a new PROJECT_MAIN or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before project_main_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to project_main_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_main

% Last Modified by GUIDE v2.5 12-Sep-2019 10:01:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @project_main_OpeningFcn, ...
    'gui_OutputFcn',  @project_main_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_main is made visible.
function project_main_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_main (see VARARGIN)
set(handles.r1_val,'string','');
set(handles.r2_val,'string','');
set(handles.lc1_val,'string','');
set(handles.lc2_val,'string','');
set(handles.v1_val,'string','');
set(handles.v2_val,'string','');
set(handles.freq_val,'string','');
grid on

global dat
dat.info=[];
set(handles.dc_check,'value',0);
set(handles.voltage_check,'value',0);

axes(handles.ckt_diagrm);
a=imread('blank.png');
imshow(a);



set(handles.data_table,'data','');
% Choose default command line output for project_main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes project_main wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = project_main_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 on button press in enter_data.
function enter_data_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to enter_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global dat
time = get(handles.time_data,'String');
vc = get(handles.vc_data,'String');

dat.info= [dat.info; [ {time} {vc}]];

set(handles.data_table,'data',dat.info);

set(handles.time_data,'string','');
set(handles.vc_data,'string','');

function time_data_Callback(hObject, eventdata, handles)
% hObject    handle to time_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of time_data as text
%        str2double(get(hObject,'String')) returns contents of time_data as a
double

%time=str2double(get(hObject,'String'))
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function time_data_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function vc_data_Callback(hObject, eventdata, handles)
% hObject    handle to vc_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vc_data as text
%        str2double(get(hObject,'String')) returns contents of vc_data as a
double

%vc=str2double(get(hObject,'String'))
guidata(hObject, handles);


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

```
function vc_data_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vc_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in ac_check.
function ac_check_Callback(hObject, eventdata, handles)
% hObject    handle to ac_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ac_check



% --- Executes on button press in dc_check.
function dc_check_Callback(hObject, eventdata, handles)
% hObject    handle to dc_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dc_check



% --- Executes on button press in start_button.
function start_button_Callback(hObject, eventdata, handles)
% hObject    handle to start_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

DATA=str2double(get(handles.data_table,'data'));
if(get(handles.time_mil,'value')==1)
    DATA(:,1)=DATA(:,1)*10^-3;
end
if(get(handles.vc_mil,'value')==1)
    DATA(:,2)=DATA(:,2)*10^-3;
end
axes(handles.transient_plot);
plot(DATA(:,1),DATA(:,2),'ro');
hold on;
source_type=get(handles.dc_check,'value');
input_type=get(handles.voltage_check,'value');
r_val=get(handles.resistor_value,'string');
r_check=get(handles.r_value,'value');
[t_out,vc_out,type,r1,r2,lc1,lc2,v1,v2,freq]=sorter(DATA,source_type,input_ty
pe,r_check,r_val);
plot(t_out,vc_out,'b'),grid on;
xlabel('Time');
```

```
if(input_type==1)
    ylabel('Voltage');
    legend('Data Points','Voltage');
    title('Voltage vs Time Graph');
else
    ylabel('Current');
    legend('Data points','Current');
    title('Current vs Time Graph');
end
hold off;
if(type==1)
    axes(handles.ckt_diagrm);
    a=imread('ac rc.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
    set(handles.freq_val,'string',['Freq = ' num2str(freq)]);
end
if(type==2)
    axes(handles.ckt_diagrm);
    a=imread('ac rl.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['L1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
    set(handles.freq_val,'string',['Freq = ' num2str(freq)]);
end
if(type==3)
    axes(handles.ckt_diagrm);
    a=imread('source free dc rc.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==4)
    axes(handles.ckt_diagrm);
    a=imread('source free dc rl.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['L1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==5)
    axes(handles.ckt_diagrm);
    a=imread('dc rc with source.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==6)
    axes(handles.ckt_diagrm);
    a=imread('dc rl with source.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['L1 = ' num2str(lc1)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==7)
    axes(handles.ckt_diagrm);
    a=imread('dc rlc series.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C = ' num2str(lc1)]);
    set(handles.lc2_val,'string',['L =' num2str(lc2)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==8)
    axes(handles.ckt_diagrm);
    a=imread('dc rlc series.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C = ' num2str(lc1)]);
    set(handles.lc2_val,'string',['L =' num2str(lc2)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==9)
    axes(handles.ckt_diagrm);
    a=imread('dc rlc series.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C = ' num2str(lc1)]);
    set(handles.lc2_val,'string',['L =' num2str(lc2)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==10)
    axes(handles.ckt_diagrm);
    a=imread('dc rc special.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1 = ' num2str(r1)]);
    set(handles.lc1_val,'string',['C1 = ' num2str(lc1)]);
    set(handles.lc2_val,'string',['Vc(0) =' num2str(v2)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end
if(type==11)
    axes(handles.ckt_diagrm);
    a=imread('dc rl special.jpg');
    imshow(a);
    set(handles.r1_val,'string',['R1` = ' num2str(r1)]);
    set(handles.lc1_val,'string',['L1 = ' num2str(lc1)]);
    set(handles.lc2_val,'string',['i(0) =' num2str(v2)]);
    set(handles.v1_val,'string',['V1 = ' num2str(v1)]);
end


% --- Executes on button press in voltage_check.
function voltage_check_Callback(hObject, eventdata, handles)
% hObject    handle to voltage_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.vc_input,'String','Voltage');
set(handles.data_table,'ColumnName',{'Time';'Voltage'});
% Hint: get(hObject,'Value') returns toggle state of voltage_check


% --- Executes on button press in current_check.
function current_check_Callback(hObject, eventdata, handles)
% hObject    handle to current_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.vc_input,'String','Current');
set(handles.data_table,'ColumnName',{'Time';'Current'});
% Hint: get(hObject,'Value') returns toggle state of current_check


% --- Executes during object creation, after setting all properties.
function choose_vc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to choose_vc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function data_table_CreateFcn(hObject, eventdata, handles)
% hObject    handle to data_table (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in r_value.
function r_value_Callback(hObject, eventdata, handles)
% hObject    handle to r_value (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of r_value



function resistor_value_Callback(hObject, eventdata, handles)
% hObject    handle to resistor_value (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of resistor_value as text
%        str2double(get(hObject,'String')) returns contents of resistor_value
as a double


% --- Executes during object creation, after setting all properties.
function resistor_value_CreateFcn(hObject, eventdata, handles)
% hObject    handle to resistor_value (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in re_set.
function re_set_Callback(hObject, eventdata, handles)
% hObject    handle to re_set (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.r1_val,'string','');
set(handles.r2_val,'string','');
set(handles.lc1_val,'string','');
set(handles.lc2_val,'string','');
set(handles.v1_val,'string','');
set(handles.v2_val,'string','');
set(handles.freq_val,'string','');

set(handles.time_data,'string','');
set(handles.vc_data,'string','');
set(handles.resistor_value,'string','');
set(handles.data_table,'data','');
set(handles.r_value,'value',0);
set(handles.time_mil,'value',0);
set(handles.vc_mil,'value',0);
axes(handles.transient_plot);
plot([1 0],[0 1],'w');
grid on;
axes(handles.ckt_diagrm);
a=imread('blank.png');
imshow(a);
global dat
dat.info=[];


% --- Executes on button press in time_mil.
function time_mil_Callback(hObject, eventdata, handles)
% hObject    handle to time_mil (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of time_mil


% --- Executes on button press in vc_mil.
function vc_mil_Callback(hObject, eventdata, handles)
% hObject    handle to vc_mil (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of vc_mil
% --- Executes on button press in Browse_button.
function Browse_button_Callback(hObject, eventdata, handles)
% hObject    handle to Browse_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global dat
[filename pathname]=uigetfile({'*.txt'},'File Selector');
fullpathname=strcat(pathname,filename);
text=fileread(fullpathname);
file_value=strsplit(text);
for i=1:2:length(file_value)
    dat.info= [dat.info; [ file_value(i) file_value(i+1) ]];
    set(handles.data_table,'data',dat.info);
end
```

Caller Function (named as Sorter)

```
function
[t_out,vc_out,type,r,r2,lc1,lc2,v1,v2,freq]=sorter(data,source_type,input_typ
e,r_check,r_val)

%r1 is already defined
r2=0;
lc1=0;
lc2=0;
v1=0;
v2=0;
freq=0;

ud_flag=0;

data=sortrows(data);

if(r_check==0)
    r=10^3;
else
    r=str2double(r_val);
end

time=data(:,1)';
vc=data(:,2)';

if(source_type==0)
    %type='AC';
    if(input_type==1)
        type=1;
        [r,lc1,v1,freq,t_out,vc_out]=ac_rc(time,vc,r);
    else
        type=2;
        [r,lc1,v1,freq,t_out,vc_out]=ac_rl(time,vc,r);
    end
    %ac done
else
    %type='DC';
    for i=1:length(vc)
        if(vc(i)<0)
            ud_flag=1;
            break;
        end
    end
    %%initial cond case
    if(vc(1)~=0 && vc(length(vc))~=0)
        if(input_type==1)
            type=10;
            [r,lc1,v1,v2,t_out,vc_out]=dc_sp_C_dis(time,vc,r);
        else
            type=11;
            [r,lc1,v1,v2,t_out,vc_out]=dc_sp_L_dis(time,vc,r);
        end
        return;
    end
    %%initial cond case
    if(issorted(flip(vc))==1)
        if(input_type==1)
            type=3;
            [r,lc1,v1,t_out,vc_out]=source_free_rc(time,vc,r);
        else
            type=4;
            [r,lc1,v1,t_out,vc_out]=source_free_rl(time,vc,r);
        end
        %source free done
    elseif(issorted(vc)==1)
        if(input_type==1)
            type=5;
            [r,lc1,v1,t_out,vc_out]=dcrc_with_source(time,vc,r);
        else
            type=6;
            [r,lc1,v1,t_out,vc_out]=dcrl_with_source(time,vc,r);
        end
        %with source done
    elseif(ud_flag==1)
        type=7;
        [r,lc1,lc2,v1,t_out,vc_out]=rlc_underdamped(time,vc,r);
    elseif(vc(length(vc))==0)
        type=8;
        [r,lc1,lc2,v1,t_out,vc_out]=rlc_critdamped(time,vc,r);
    else
        type=9;
        [r,lc1,lc2,v1,t_out,vc_out]=rlc_overdamped(time,vc,r);
    end
end
end
```

Case 1: Source Free DC RC

```
function [r,c,v0,t_out,vc_out]=source_free_rc(t,v,r)

t_out=t(1):0.01:t(length(t));

t(find(t==0))=0.000001;
v(find(v==0))=0.000001;

t(1)=[];
v(1)=[];
t(end)=[];
v(end)=[];

v=log(v);

res=lin_reg(t,v);

tau=(-1/res(1));
v0=round(exp(res(2)),2);

c=tau/r;

vc_out=v0*exp(-1*t_out/tau);
```

Case 2: Source Free DC RL

```
function [r,l,v0,t_out,vc_out]=source_free_rl(t,i,r)

t_out=t(1):0.001:t(length(t));

t(find(t==0))=0.000001;
i(find(i==0))=0.000001;
t(end)=[];
i(end)=[];

i=log(i);

res=lin_reg(t,i);

tau=(-1/res(1));
i0=round(exp(res(2)),2);

v0=r*i0;
l=tau*r;
vc_out=i0*exp(-1*t_out/tau);
```

Case 3: DC RC with Source

```
function [r,c,v1,t_out,vc_out]=dcrc_with_source(t,v,r)

t_out=t(1):0.001:t(length(t));
vinf=v(length(v));

t(find(t==0))=0.000001;
v(find(v==0))=0.000001;

t(end)=[];
v(end)=[];

v=log(vinf-v);

res=lin_reg(t,v);

tau=(-1/res(1));
v0=round(vinf-exp(res(2)),2);

vc_out=vinf+((v0-vinf)*exp(-1*t_out/tau));

c=tau/r;

v1=vinf;
```

Case 4: DC RL with Source

```
function [r,l,v1,t_out,vc_out]=dcrl_with_source(t,i,r)

t_out=t(1):0.001:t(length(t));
Iinf=i(length(i));

t(find(t==0))=0.000001;
i(find(i==0))=0.000001;

t(end)=[];
i(end)=[];

i=log(Iinf-i);
res=lin_reg(t,i);
tau=(-1/res(1));
i0=round(Iinf-exp(res(2)),2);
vc_out=Iinf+((i0-Iinf)*exp(-1*t_out/tau));

l=tau*r;
v1=Iinf*r;
```

Case 5: DC RC with Initial Conditions

```
function [r,c,v1,v2,t_out,vc_out]=dc_sp_C_dis(t,v,r)
v2=v(1);
v1=v(length(v));
if(issorted(v)==0)
    v=v-v(length(v));
    t_out=t(1):0.001:t(length(t));
    t(find(t==0))=0.000001;
    v(find(v==0))=0.000001;

    t(1)=[];
    v(1)=[];
    t(end)=[];
    v(end)=[];

    v=log(v);

    res=lin_reg(t,v);

    tau=(-1/res(1));
    v0=round(exp(res(2)),2);
    c=tau/r;

    vc_out=v0*exp(-1*t_out/tau)+v1;
else
    v=v-v(length(0));
    t_out=t(1):0.001:t(length(t));
    vinf=v(length(v));

    t(find(t==0))=0.000001;
    v(find(v==0))=0.000001;

    t(end)=[];
    v(end)=[];

    v=log(vinf-v);

    res=lin_reg(t,v);

    tau=(-1/res(1));
    v0=round(vinf-exp(res(2)),2);

    vc_out=vinf+((v0-vinf)*exp(-1*t_out/tau))+v2;

    c=tau/r;

    %v1=vinf+v(length(v));
end
```

Case 6: DC RL with Initial Conditions

```
function [r,l,v1,v2,t_out,vc_out]=dc_sp_L_dis(t,i,r)
v2=i(1);
i1=i(length(i));
v1=i1*r;

if(issorted(i)==0)
    i=i-i(length(i));
    t_out=t(1):0.001:t(length(t));
    t(find(t==0))=0.000001;
    i(find(i==0))=0.000001;

    t(1)=[];
    i(1)=[];
    t(end)=[];
    i(end)=[];

    i=log(i);

    res=lin_reg(t,i);

    tau=(-1/res(1));
    i0=round(exp(res(2)),2);
    l=tau*r;

    vc_out=i0*exp(-1*t_out/tau)+i1;
else
    i=i-i(length(0));
    t_out=t(1):0.001:t(length(t));
    vinf=i(length(i));

    t(find(t==0))=0.000001;
    i(find(i==0))=0.000001;

    t(end)=[];
    i(end)=[];

    i=log(vinf-i);

    res=lin_reg(t,i);

    tau=(-1/res(1));
    i0=round(vinf-exp(res(2)),2);

    vc_out=vinf+((i0-vinf)*exp(-1*t_out/tau))+v2;

    l=tau*r;

    %v1=vinf+v(length(v));
end
```

Case 7: DC RLC overdamped

```
function [r,c,l,v1,t_out,vc_out]=rlc_overdamped(t,vc,r)

f=fit(t',vc','exp2');
t_out=t(1):0.001:2*t(length(t));
vc_out=f(t_out);

s1=f.b;
s2=f.d;

a=-1*(s1+s2)/2;
w=sqrt((a*a)-((a+s1)*(a+s1)));
l=r/(2*a);
c=1/(l*w*w);

t_c=t(1):0.001:5*t(length(t));
v1=@(t_c) (1/c)*(((f.a/f.b)*exp(f.b*t_c))+((f.c/f.d)*exp(f.d*t_c)));
v_c=v1(t_c)-v1(0);
v1=v_c(length(v_c));
```

Case 8: DC RLC critically damped

```
function [r,c,l,v1,t_out,vc_out]=rlc_critdamped(t,vc,r)

t_out=t(1):0.001:t(length(t));
%cftool
[f, gof] = createFit_cdamp(t,vc);
vc_out=f(t_out);

l=r/(2*f.b);
c=1/(l*f.b*f.b);
v1=(1/c)*euler_imp_val(vc_out,t_out,0);
v1=v1(length(v1));
```

Case 9: DC RLC underdamped

```
function [r,c,l,v1,t_out,vc_out]=rlc_underdamped(t,vc,r)


t_out=t(1):0.001:t(length(t));
vc_out=spline(t,vc,t_out);

%plot(t,vc,'r.',t_out,vc_out,'b');

[maxi,locmax]=findpeaks(vc_out,t_out);
[mini,locmin]=findpeaks(-1*vc_out,t_out);
mini=-1*mini;

%plot(locmax,maxi,'r+',locmin,mini,'r+');

if(length(maxi)>length(mini))
    maxi=maxi(1:length(mini));
    locmax=locmax(1:length(locmin));
elseif(length(maxi)<length(mini))
    mini=mini(1:length(maxi));
    locmin=locmin(1:length(locmax));
end

E=(maxi+mini)/2;
loc=locmax+(abs(locmax-locmin));

loc(E<0)=[];
E(E<0)=[];

%plot(loc,E,'bo');

sorter=[loc' E'];
sorter=sortrows(sorter);
E=sorter(:,2);
loc=sorter(:,1);

[a,b]=linearization_2_pro(loc,E);

y_=a*exp(-b*t_out);

%plot(t_out,y_);

T=abs(mean(locmax-locmin)*2);
freq=1/T;
wd=2*pi*freq;
w=sqrt(wd^2+b^2);
l=r/(2*b);
c=1/(w*w*l);
v1=(1/c)*euler_imp_val(vc_out,t_out,0);
v1=v1(length(v1));
```

Case 10: AC RC

```
function[r,c,v1,freq,t_out,vc_out]=ac_rc(t,v,r)

%hold on;
%grid on;
%plot(t,v,'ro');

[fitresult, gof] = createFit(t, v);
t_out=t(1):0.001:t(length(t));

vc_out=fitresult(t_out);
%vc_out=triginterp(t_out,t,v);
%vc_out=spline(t,v,t_out);
%plot(t_out,vc_out,'b');

[maxi,locmax]=findpeaks(vc_out,t_out);
[mini,locmin]=findpeaks(-1*vc_out,t_out);
mini=-1*mini;

%plot(locmax,maxi,'r+',locmin,mini,'r+');

if(length(maxi)>length(mini))
    maxi=maxi(1:length(mini));
    locmax=locmax(1:length(locmin));
elseif(length(maxi)<length(mini))
    mini=mini(1:length(maxi));
    locmin=locmin(1:length(locmax));
end

E=(maxi+mini)/2;
loc=locmax+(abs(locmax-locmin)/2);

loc(E<0)=[];
E(E<0)=[];
E(end+1)=maxi(1)/2;
loc(end+1)=locmax(1)/2;

E(end+1)=abs(min(maxi-mini)/2);
loc(end+1)=0.0001;

%plot(loc,E,'bo');

sorter=[loc' E];
sorter=sortrows(sorter);
E=sorter(:,2);
loc=sorter(:,1);

[a,b]=linearization_2_pro(loc,E);

y_=a*exp(-b*t_out);
```

```
%plot(t_out,y_);

tau=1/b;

c=tau/r;


T=round(abs(mean(locmax-locmin)*2),2);

freq=(1/T);

w=2*pi*freq;

ampli=abs(mean(maxi-mini)/2);

theta=atan(1/(w*r*c));

v1=ampli*sqrt(r^2+(1/(w*c))^2)*w*c;

vc_out=-1*ampli*cos(w*t_out+theta)+ampli*cos(theta)*exp(-1*t_out/tau);

%plot(t_out,vc_out,'r');
```

Case 11: AC RC

```
function[r,l,v1,freq,t_out,vc_out]=ac_rl(t,i,r)

%hold on;
%grid on;
%plot(t,i,'ro');

f=fit(t',i','fourier8');

t_out=t(1):0.001:t(length(t));

vc_out=f(t_out);
%vc_out=triginterp(t_out,t,i);
%vc_out=spline(t,i,t_out);
%plot(t_out,vc_out,'b');

[maxi,locmax]=findpeaks(vc_out,t_out);
[mini,locmin]=findpeaks(-1*vc_out,t_out);
mini=-1*mini;

if(length(maxi)>length(mini))
    maxi=maxi(1:length(mini));
    locmax=locmax(1:length(locmin));
elseif(length(maxi)<length(mini))
    mini=mini(1:length(maxi));
    locmin=locmin(1:length(locmax));
end

%plot(locmax,maxi,'r+',locmin,mini,'r+');

E=(maxi+mini)/2;
loc=locmax+(abs(locmax-locmin)/2);

loc(E<0)=[];
E(E<0)=[];
E(end+1)=maxi(1)/2;
loc(end+1)=locmax(1)/2;

%plot(loc,E,'bo');

sorter=[loc' E];
sorter=sortrows(sorter);
E=sorter(:,2);
loc=sorter(:,1);

[a,b]=linearization_2_pro(loc,E);

%y_=a*exp(-b*t_out);

%plot(t_out,y_);
```

```
tau=1/b
l=tau*r;
T=round(abs(mean(locmax-locmin)*2),3);
freq=round(1/T);
w=2*pi*freq;
%ampli=abs(mean(maxi-mini)/2);
theta=atan(w*l/r);
ampli=abs(mean(maxi-mini)/2);
v1=ampli*sqrt(r^2+(l*w)^2);
vc_out=ampli*sin(w*t_out-theta)+ampli*sin(theta)*exp(-1*t_out/tau);
%plot(t_out,vc_out,'r');
```

Other Functions

```
function equ=lin_reg(x,y)

A=[length(x), sum(x)
    sum(x) sum(x.^2)];
B=[sum(y);sum(x.*y)];

equ=flip(A\B);
```

```
function equ=gen_reg(x,y,deg)

n=deg+1;
A=zeros(n,n);
B=zeros(n,1);

for i=1:n
    for j=1:n
        A(i,j)=sum(x.^(i+j-2));
    end

    B(i)=sum((x.^(i-1)).*y);

end
equ=flip(A\B);
end
```

```
function [r c]=linearization_1_pro(x,y)
vinf=y(length(y));
x(length(x))=[];
y(length(y))=[];
n=length(x);


y=log(vinf-y);
a1=(n*sum(x.*y)-(sum(x).*sum(y)))/(n*sum(x.^2)-(sum(x))^2);
a0=(sum(y)/n)-(a1*(sum(x)/n));
tau=-1/a1;
vo=vinf-exp(a0);
r=1000;
c=tau/r;

end
```

```
function [a b]=linearization_2_pro(x,y)


n=length(x);


y=log(y);
a1=(n*sum(x.*y)-(sum(x).*sum(y)))/(n*sum(x.^2)-(sum(x))^2);
a0=(sum(y)/n)-(a1*(sum(x)/n));
a=exp(a0);
b=-a1;
% tau=-1/a1;
% vo=exp(a0);
% r=1000;
% c=tau/r;

end
```

```
function [r1 r2 c]=linearization_3_pro(x,y,vs)
vinf=y(length(y));
x(length(x))=[];
y(length(y))=[];
n=length(x);


y=log(vinf-y);
a1=(n*sum(x.*y)-(sum(x).*sum(y)))/(n*sum(x.^2)-(sum(x))^2);
a0=(sum(y)/n)-(a1*(sum(x)/n));
tau=-1/a1;
vo=vinf-exp(a0);

r2=10000;
r1=r2*((vs/vinf)-1);
c=tau/r2;

end
```

```
function [r1 r2 c]=linearization_4_pro(x,y,v1,v2)
vinf=v1;
x(length(x))=[];
y(length(y))=[];
n=length(x);


y=log(v1-y);
a1=(n*sum(x.*y)-(sum(x).*sum(y)))/(n*sum(x.^2)-(sum(x))^2);
a0=(sum(y)/n)-(a1*(sum(x)/n));
tau=-1/a1;
vo=v1-exp(a0);
r1=10000;
r2=1000;
c=tau/r1;
end
```

```
function [fitresult, gof] = createFit(t, v)

[xData, yData] = prepareCurveData( t, v );
ft = fittype( 'fourier8' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8.5679799643358];
[fitresult, gof] = fit( xData, yData, ft, opts );

end
```

```
function [fitresult, gof] = createFit_cdamp(t, i)

[xData, yData] = prepareCurveData( t, i );

ft = fittype( 'a*exp(-b*x)+c*x*exp(-b*x)', 'independent', 'x', 'dependent',
'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
%opts.Display = 'Off';
opts.StartPoint = [0.197117046466842 0.374638887225915 0.960456314540204];
[fitresult, gof] = fit( xData, yData, ft, opts );
```

```
function res=euler_imp_val(fd,x,y0)

n=length(x);
h=abs(x(2)-x(1));
res=zeros(1,n);

res(1)=y0;

for i=2:n
    res(i)=res(i-1) + (h/2)* (fd(i-1) + fd(i));
end
```