Alexandria University

Faculty of Engineering

Electrical Engineering Department

Power Section

2020/2021

Power System Protection II

Assignment 1

Report Done by:

1- **Ahmed Al-Sayed Ali Morsi** **(ID:8)**
2- **Ahmed Adel Mohamed El-Damanhori** **(ID:19)**
3- **Ahmed Mohammed Hamdan Mahmoud** **(ID:29)**
4- **Mohamed Magdy El-Sayed El-Wardani** **(ID:160)**

Objective:

Calculating the values of rms current and voltage values and angles and impedance values and angles from samples by the equations illustrated in these codes, and checking the error values and the overcurrent while exceeding 120% of the current value.

1. MATLAP CODE 1 (m-file):

```
clc;
clear;
Ireal=400/sqrt(2); %rms normal current
Fs=10^4; %sampling frequency
Fp=50; %power frequency
Ik=[118.21 155.77 191.77 225.86 257.7 286.94]; %current samples
Vk=[2933.6  3190.17  3414.94  3605.64  3760.4  3877.65];  %voltage samples
NoSamples=6; %number of samples

if NoSamples==length(Ik) && NoSamples==length(Vk) %number of current and voltage samples must be equal
      %constructing the general matrices
      Irms(1,NoSamples)=0;
      Vrms(1,NoSamples)=0;
      Z(1,NoSamples)=0;
      theta_i(1,NoSamples)=0;
      theta_v(1,NoSamples)=0;
      theta_z(1,NoSamples)=0;
      error(1,NoSamples)=0;
      Overcurrent(1,NoSamples)=false;

      x=input('Enter The Number of Method >> ');

      switch x
          case 1
              for k=2:NoSamples %counter for calculation
                  %calculating magnitude and angle of i & v & z
                  Irms(k)=sqrt(Ik(k)^2+(((Fs*(Ik(k)-Ik(k-1)))^2)/((2*pi*Fp)^2)))/sqrt(2);
                  Vrms(k)=sqrt(Vk(k)^2+(((Fs*(Vk(k)-Vk(k-1)))^2)/((2*pi*Fp)^2)))/sqrt(2);
                  Z(k)=Vrms(k)/Irms(k);
                  theta_i(k)=atan((2*pi*Ik(k))/(Fs)*(Ik(k)-Ik(k-1)));
                  theta_v(k)=atan((2*pi*Vk(k))/(Fs)*(Vk(k)-Vk(k-1)));
                  theta_z(k)=theta_v(k)-theta_i(k);
                  error(k)=Irms(k)-Ireal;
                  Overcurrent(k)=logical((120/100)*Irms(k));
```

```
            end

        case 2

            %constructing the matrices for this method only
            Vd1(1,NoSamples)=0;
            Vd2(1,NoSamples)=0;
            Id1(1,NoSamples)=0;
            Id2(1,NoSamples)=0;

            for k=2:NoSamples-1
                %calculating magnitude and angle of i & v & z
                Id1(k)=(Fs/2)*(Ik(k+1)-Ik(k-1));
                Vd1(k)=(Fs/2)*(Vk(k+1)-Vk(k-1));
                Id2(k)=(Fs^2)*(Ik(k+1)-2*Ik(k)+Ik(k-1));
                Vd2(k)=(Fs^2)*(Vk(k+1)-2*Vk(k)+Vk(k-1));
                Irms(k)=sqrt(((1/(2*pi*Fp))^2)*((Id1(k))^2+(Id2(k)/(2*pi*Fp))^2))/sqrt(2);
                Vrms(k)=sqrt(((1/(2*pi*Fp))^2)*((Vd1(k))^2+(Vd2(k)/(2*pi*Fp))^2))/sqrt(2);
                Z(k)=Vrms(k)/Irms(k);
                theta_i(k)=-atan(Id2(k)/(2*pi*Fp*Id1(k)));
                theta_v(k)=-atan(Vd2(k)/(2*pi*Fp*Vd1(k)));
                theta_z(k)=theta_v(k)-theta_i(k);
                error(k)= Irms(k)-Ireal;
                Overcurrent(k)=logical((120/100)*Irms(k));
            end

        case 3
            %constructing the matrices for this method only
            theta(1,NoSamples)=0;

            for k=1:NoSamples-1
                %calculating magnitude and angle of i & v & z
                Irms(k)=(sqrt(((Ik(1,k)^2+Ik(1,k+1)^2-2*Ik(1,k)*Ik(1,k+1)*cos(2*pi*Fp/Fs)))/(sin(2*pi*Fp/Fs))^2))/sqrt(2);
                Vrms(k)=(sqrt(((Vk(1,k)^2+Vk(1,k+1)^2-2*Vk(1,k)*Vk(1,k+1)*cos(2*pi*50/Fs)))/(sin(2*pi*Fp/Fs))^2))/sqrt(2);
                Z(k)=Vrms(k)/Irms(k);
                theta(k)=acos((Vk(1,k)*Ik(1,k)+Vk(1,k+1)*Ik(1,k+1)-(Vk(1,k)*Ik(1,k+1)+Vk(1,k+1)*Ik(1,k))*cos(2*pi*Fp/Fs))/(Vrms(1,k)*Irms(1,k)*2*sin(2*pi*Fp/Fs)^2));
                theta_i(k)=theta(k);
                theta_z(k)=-theta(k);
                error(k)=Irms(k)-Ireal;
                Overcurrent(k)=logical((120/100)*Irms(k));
            end
```

```
        otherwise
            msgbox('Invalid Number of Method', 'Error','error'); %showing an error message box
    end

T=table((1:NoSamples)',Irms',theta_i',Vrms',theta_v',Z',theta_z',error',Overcurrent'); %construct a table for results
T.Properties.VariableNames={'Sample','Irms','theta_i','Vrms','theta_v','Z','theta_z','error','Overcurrent'} %row names of the table

    subplot(2,4,1) %plotting the magnitude of rms current signal at all samples
    bar(1:NoSamples,Irms);
    line(xlim,[Ireal,Ireal],'Color','r','LineWidth',1);
    xlabel('No. of Samples');
    ylabel('Estimated Magnitude of rms Current');

    subplot(2,4,5) %plotting the angle of rms current signal at all samples
    bar(1:NoSamples,theta_i);
    xlabel('No. of Samples');
    ylabel('Estimated Angle of rms Current');

    subplot(2,4,2) %plotting the magnitude of rms voltage signal at all samples
    bar(1:NoSamples,Vrms);
    xlabel('No. of Samples');
    ylabel('Estimated Magnitude of rms Voltage');

    subplot(2,4,6) %plotting the angle of rms voltage signal at all samples
    bar(1:NoSamples,theta_v);
    xlabel('No. of Samples');
    ylabel('Estimated Angle of rms Voltage');

    subplot(2,4,3) %plotting the magnitude of rms impedance signal at all samples
    bar(1:NoSamples,Z);
    xlabel('No. of Samples');
    ylabel('Estimated Magnitude of rms Impedance');

    subplot(2,4,7) %plotting the angles of rms impedance signal at all samples
    bar(1:NoSamples,theta_z);
    xlabel('No. of Samples');
    ylabel('Estimated Angle of rms Impedance');

    subplot(2,4,[4 8]) %plotting the error at all samples
```

```
        bar(1:NoSamples,error);
        xlabel('No. of Samples');
        ylabel('Error');

else
        msgbox('Invalid Symmetry for Samples', 'Error','error'); %showing an error message box

end
```

Results:

With choosing the method by choosing x, the graphs will appear in one figure and table appears in the command window showing all results.

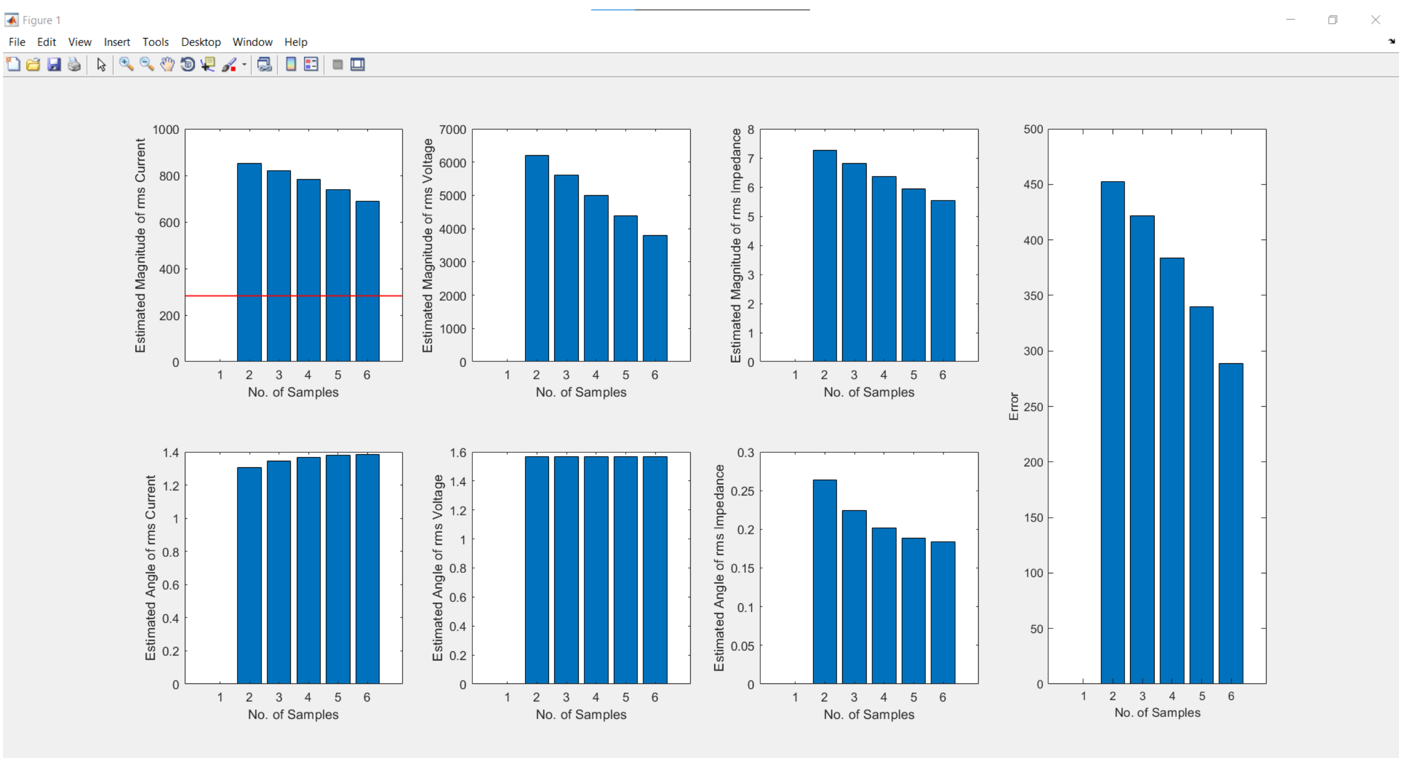
1- at x=1:

Command Window:

Command Window

Sample	Ik	I_theta	Vk	V_theta	Zk	Z_theta	error	Overcurrent
1	0	0	0	0	0	0	0	false
2	852.54	1.3052	6199.8	1.5689	7.2721	0.26365	452.54	true
3	821.55	1.3442	5605.8	1.5687	6.8235	0.2245	421.55	true
4	783.74	1.367	4992.4	1.5685	6.3699	0.20152	383.74	true
5	739.46	1.3792	4382.2	1.5681	5.9263	0.18886	339.46	true
6	688.7	1.3833	3805.6	1.5673	5.5258	0.18397	288.7	true

Plot:



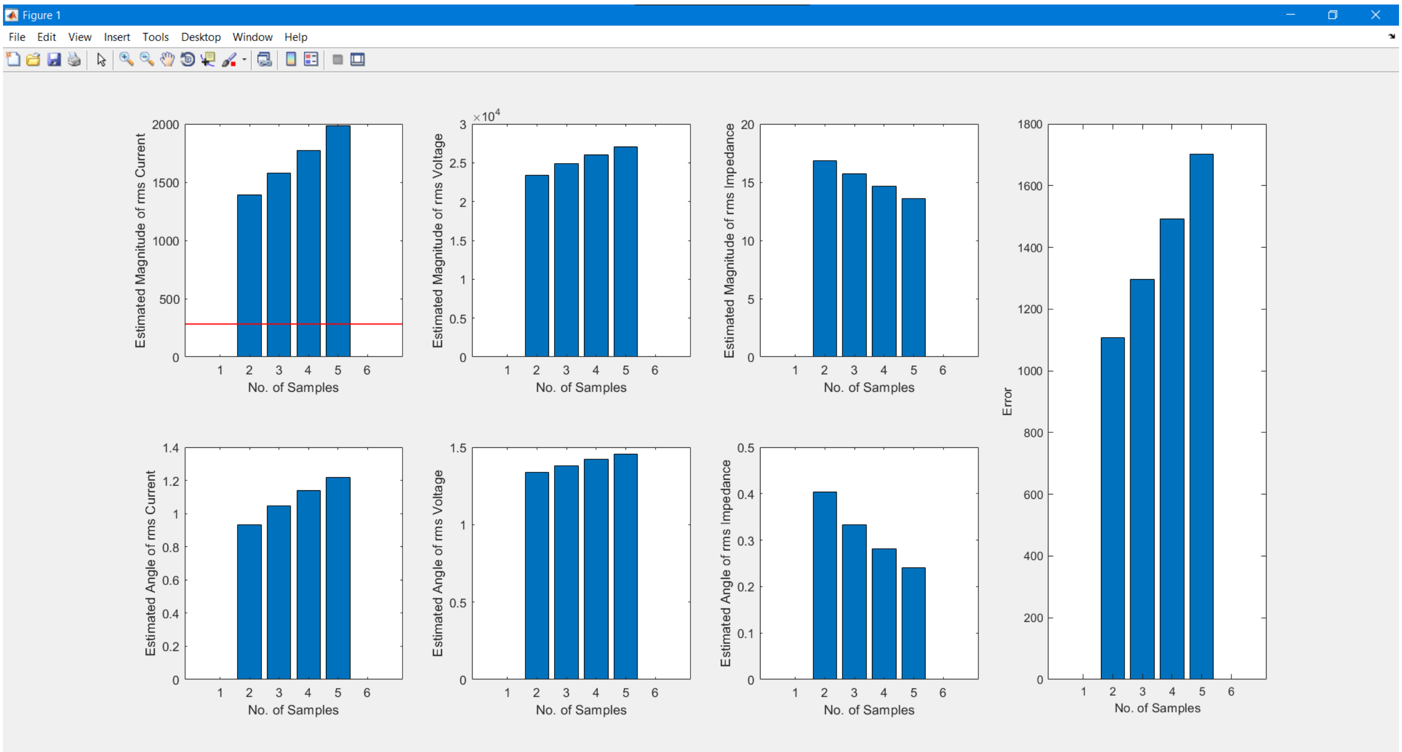
2- at x=2:

Command Window:

Command Window

Sample	Irms	theta_i	Vrms	theta_v	Z	theta_z	error	Overcurrent
1	0	0	0	0	0	0	0	false
2	1390.9	0.93328	23418	1.3374	16.837	0.40409	1108	true
3	1579.5	1.0479	24853	1.3815	15.735	0.33364	1296.6	true
4	1774.6	1.1394	26041	1.4209	14.675	0.28151	1491.7	true
5	1985.6	1.2173	27048	1.4574	13.622	0.24009	1702.7	true
6	0	0	0	0	0	0	0	false

Plot:



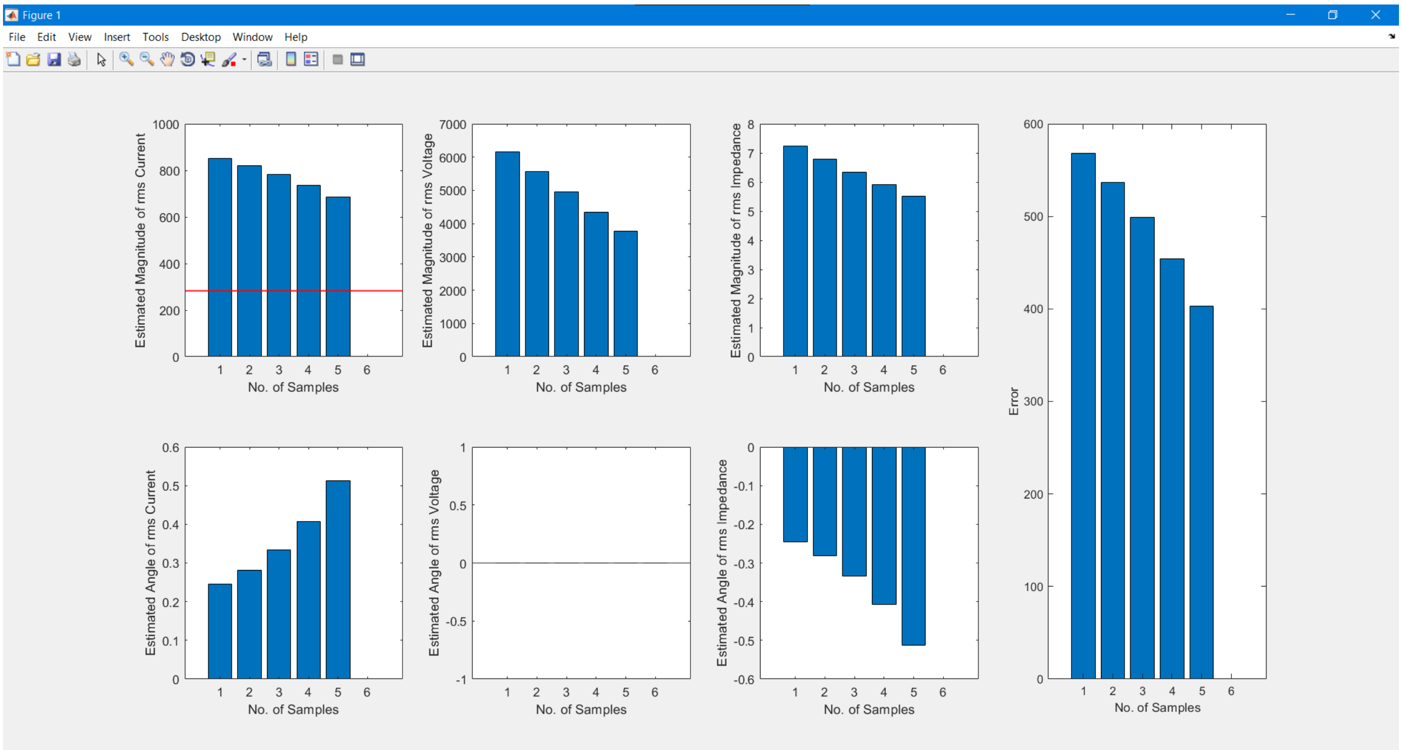
3- at x=3:

Command Window:

Command Window

Sample	Irms	theta_i	Vrms	theta_v	Z	theta_z	error	Overcurrent
1	850.96	0.24463	6167.7	0	7.2479	-0.24463	568.12	true
2	819.58	0.28198	5572.4	0	6.799	-0.28198	536.74	true
3	781.41	0.33421	4958.6	0	6.3457	-0.33421	498.56	true
4	736.8	0.40783	4349.5	0	5.9033	-0.40783	453.95	true
5	685.76	0.51228	3776.2	0	5.5066	-0.51228	402.91	true
6	0	0	0	0	0	0	0	false

Plot:



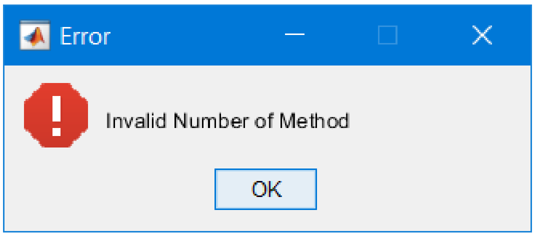
4- at x= any other number:

Command Window:

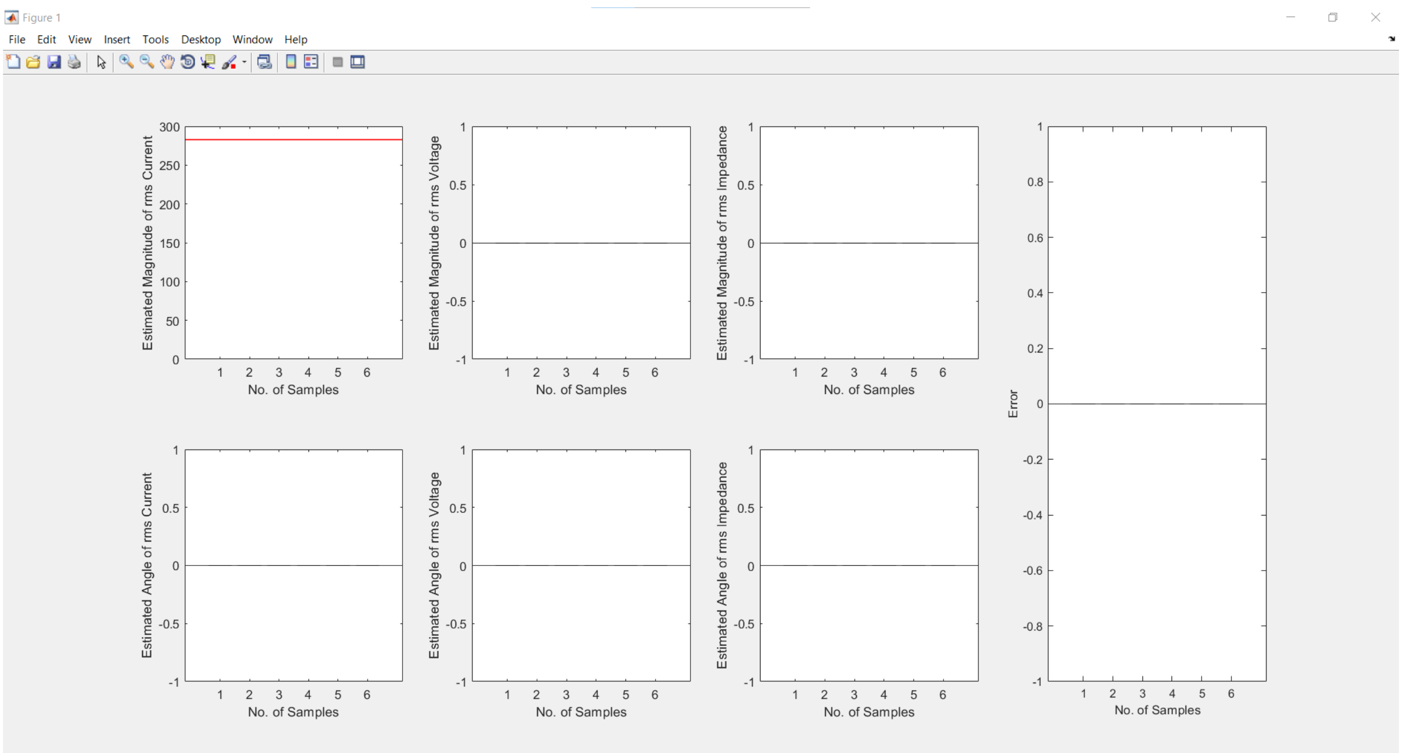
Command Window

Sample	Irms	theta_i	Vrms	theta_v	Z	theta_z	error	Overcurrent
1	0	0	0	0	0	0	0	false
2	0	0	0	0	0	0	0	false
3	0	0	0	0	0	0	0	false
4	0	0	0	0	0	0	0	false
5	0	0	0	0	0	0	0	false
6	0	0	0	0	0	0	0	false

Check Box:



Plot:



2. MATLAP CODE 2 (fig file & m-file):

```
function varargout = Assignment_1_Complete_GUI(varargin)
% ASSIGNMENT_1_COMPLETE_GUI MATLAB code for Assignment_1_Complete_GUI.fig
%      ASSIGNMENT_1_COMPLETE_GUI, by itself, creates a new ASSIGNMENT_1_COMPLETE_GUI or raises the existing
%      singleton*.
%
%      H = ASSIGNMENT_1_COMPLETE_GUI returns the handle to a new ASSIGNMENT_1_COMPLETE_GUI or the handle to
%      the existing singleton*.
%
%      ASSIGNMENT_1_COMPLETE_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ASSIGNMENT_1_COMPLETE_GUI.M with the given input arguments.
%
%      ASSIGNMENT_1_COMPLETE_GUI('Property','Value',...) creates a new ASSIGNMENT_1_COMPLETE_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Assignment_1_Complete_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Assignment_1_Complete_GUI_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 Assignment_1_Complete_GUI

% Last Modified by GUIDE v2.5 13-Dec-2020 02:40:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Assignment_1_Complete_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Assignment_1_Complete_GUI_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 Assignment_1_Complete_GUI is made visible.
function Assignment_1_Complete_GUI_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 Assignment_1_Complete_GUI (see VARARGIN)

% Choose default command line output for Assignment_1_Complete_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Assignment_1_Complete_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Assignment_1_Complete_GUI_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 radiobutton1.
```

```
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (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 radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (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 radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (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 radiobutton3


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% 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)
Ireal=400/sqrt(2); %rms normal current
Fs=10^4; %sampling frequency
Fp=50; %power frequency
Ik=[118.21 155.77 191.77 225.86 257.7 286.94]; %current samples
Vk=[2933.6 3190.17 3414.94 3605.64 3760.4 3877.65]; %voltage samples
NoSamples=6; %number of samples

if NoSamples==length(Ik) && NoSamples==length(Vk) %number of current and voltage samples must be equal
    %constructing the general matrices
    Irms(1,NoSamples)=0;
    Vrms(1,NoSamples)=0;
    Z(1,NoSamples)=0;
```

```
    theta_i(1,NoSamples)=0;
    theta_v(1,NoSamples)=0;
    theta_z(1,NoSamples)=0;
    error(1,NoSamples)=0;
    Overcurrent(1,NoSamples)=0;

    if get(handles.radiobutton1,'value')==1 %method 1
        disp('1. Sample and First Derivative Method :');
        for k=2:NoSamples %counter for calculation
            %calculating magnitude and angle of i & v & z
            Irms(k)=sqrt(Ik(k)^2+(((Fs*(Ik(k)-Ik(k-1)))^2)/((2*pi*Fp)^2)))/sqrt(2);
            Vrms(k)=sqrt(Vk(k)^2+(((Fs*(Vk(k)-Vk(k-1)))^2)/((2*pi*Fp)^2)))/sqrt(2);
            Z(k)=Vrms(k)/Irms(k);
            theta_i(k)=atan((2*pi*Ik(k))/(Fs)*(Ik(k)-Ik(k-1)));
            theta_v(k)=atan((2*pi*Vk(k))/(Fs)*(Vk(k)-Vk(k-1)));
            theta_z(k)=theta_v(k)-theta_i(k);
            error(k)=Irms(k)-Ireal;
            Overcurrent(k)=logical((120/100)*Irms(k));
        end

    elseif get(handles.radiobutton2,'value')==1 %method 2
        %constructing the matrices for this method only
        Vd1(1,NoSamples)=0;
        Vd2(1,NoSamples)=0;
        Id1(1,NoSamples)=0;
        Id2(1,NoSamples)=0;
        disp('2. First and Decond Derivative Method :');

        for k=2:NoSamples-1
            %calculating magnitude and angle of i & v & z
            Id1(k)=(Fs/2)*(Ik(k+1)-Ik(k-1));
            Vd1(k)=(Fs/2)*(Vk(k+1)-Vk(k-1));
            Id2(k)=(Fs^2)*(Ik(k+1)-2*Ik(k)+Ik(k-1));
            Vd2(k)=(Fs^2)*(Vk(k+1)-2*Vk(k)+Vk(k-1));
            Irms(k)=sqrt(((1/(2*pi*Fp))^2)*((Id1(k))^2+(Id2(k)/(2*pi*Fp))^2))/sqrt(2);
            Vrms(k)=sqrt(((1/(2*pi*Fp))^2)*((Vd1(k))^2+(Vd2(k)/(2*pi*Fp))^2))/sqrt(2);
            Z(k)=Vrms(k)/Irms(k);
            theta_i(k)=-atan(Id2(k)/(2*pi*Fp*Id1(k)));
            theta_v(k)=-atan(Vd2(k)/(2*pi*Fp*Vd1(k)));
            theta_z(k)=theta_v(k)-theta_i(k);
            error(k)= Irms(k)-Ireal;
            Overcurrent(k)=logical((120/100)*Irms(k));
        end

```

```
    elseif get(handles.radiobutton3,'value')==1 %method 3
        %constructing the matrices for this method only
        theta(1,NoSamples)=0;
        disp('3. Two-Sample Technique :');

        for k=1:NoSamples-1
            %calculating magnitude and angle of i & v & z
            Irms(k)=(sqrt(((Ik(1,k)^2+Ik(1,k+1)^2-2*Ik(1,k)*Ik(1,k+1)*cos(2*pi*Fp/Fs)))/(sin(2*pi*Fp/Fs))^2))/sqrt(2);
            Vrms(k)=(sqrt(((Vk(1,k)^2+Vk(1,k+1)^2-2*Vk(1,k)*Vk(1,k+1)*cos(2*pi*50/Fs)))/(sin(2*pi*Fp/Fs))^2))/sqrt(2);
            Z(k)=Vrms(k)/Irms(k);
theta(k)=acos((Vk(1,k)*Ik(1,k)+Vk(1,k+1)*Ik(1,k+1)-(Vk(1,k)*Ik(1,k+1)+Vk(1,k+1)*Ik(1,k))*cos(2*pi*Fp/Fs))/(Vrms(1,k)*Irms(1,k)*2*sin(2*pi*Fp/Fs)^2));
            theta_i(k)=theta(k);
            theta_z(k)=-theta(k);
            error(k)=Irms(k)-Ireal;
            Overcurrent(k)=logical((120/100)*Irms(k));
        end
    end


T=table((1:NoSamples)',Irms',theta_i',Vrms',theta_v',Z',theta_z',error',Overcurrent'); %construct a table for results
T.Properties.VariableNames={'Sample','Irms','theta_i','Vrms','theta_v','Z','theta_z','error','Overcurrent'} %row names of the table

    %plotting the magnitude of rms current signal at all samples
    axes(handles.axes1)
    bar(1:NoSamples,Irms);
    line(xlim,[Ireal,Ireal],'Color','r','LineWidth',1);

    %plotting the angle of rms current signal at all samples
    axes(handles.axes2)
    bar(1:NoSamples,theta_i);

    %plotting the magnitude of rms voltage signal at all samples
    axes(handles.axes3)
    bar(1:NoSamples,Vrms);

    %plotting the angle of rms voltage signal at all samples
    axes(handles.axes4)
    bar(1:NoSamples,theta_v);

    %plotting the magnitude of rms impedance signal at all samples
```

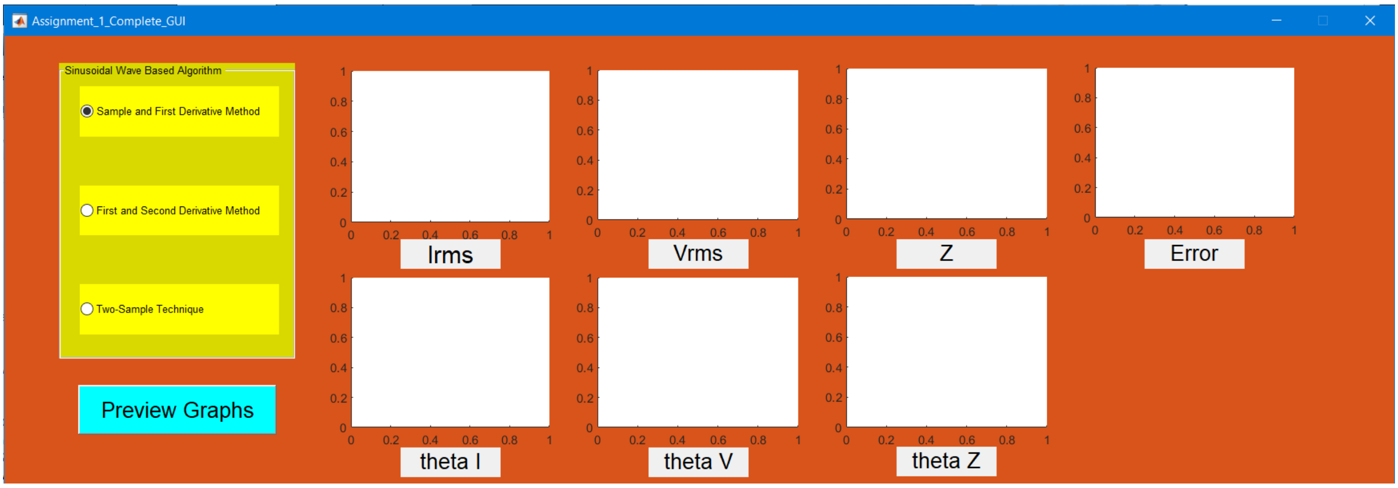
```
    axes(handles.axes5)
    bar(1:NoSamples,Z);

    %plotting the angles of rms impedance signal at all samples
    axes(handles.axes6)
    bar(1:NoSamples,theta_z);

    %plotting the error at all samples
    axes(handles.axes7)
    bar(1:NoSamples,error);

end
```

GUI Interface:



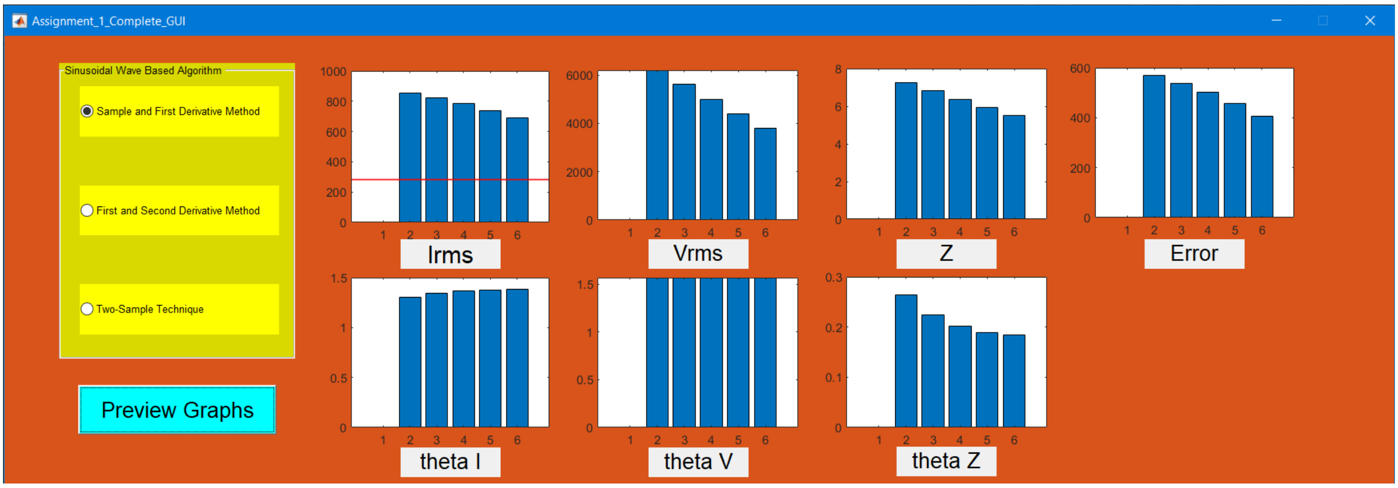
<u>**Results:**</u>

The same as previous with these code and GUI with this difference:

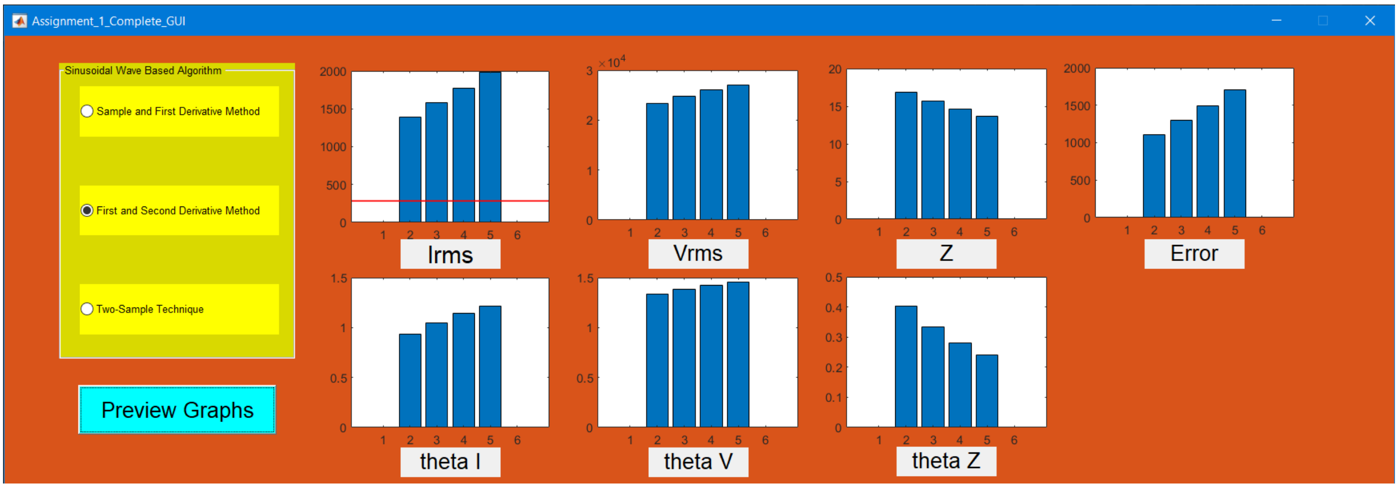
With choosing any of the three radiobuttons and pressing “Preview Graphs” pushbutton, the previous graphs will appear in the seven GUI axes and the table appears in the command window.

And the GUI states is illustrated here:

1- Method 1:



2- Method 2:



3- Method 3: