

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. MATLAB CODE 1 (m-file):

```
1) clc;
2) clear;
3) Ireal=400/sqrt(2); %rms normal current
4) Fs=10^4; %sampling frequency
5) Fp=50; %power frequency
6) Ik=[118.21 155.77 191.77 225.86 257.7 286.94]; %current samples
7) Vk=[2933.6 3190.17 3414.94 3605.64 3760.4 3877.65]; %voltage samples
8) NoSamples=6; %number of samples
9)
10) if NoSamples==length(Ik) && NoSamples==length(Vk) %number of
    current and voltage samples must be equal
11)     %constructing the general matrices
12)     Irms(1,NoSamples)=0;
13)     Vrms(1,NoSamples)=0;
14)     Z(1,NoSamples)=0;
15)     theta_i(1,NoSamples)=0;
16)     theta_v(1,NoSamples)=0;
17)     theta_z(1,NoSamples)=0;
18)     error(1,NoSamples)=0;
19)     Overcurrent(1,NoSamples)=false;
20)
21)     x=input('Enter The Number of Method >> ');
22)
23)     switch x
24)         case 1
25)             for k=2:NoSamples %counter for calculation
26)                 %calculating magnitude and angle of i & v & z
27)                 Irms(k)=sqrt(Ik(k)^2+((Fs*(Ik(k)-Ik(k-
                    1))))^2)/((2*pi*Fp)^2))/sqrt(2);
28)                 Vrms(k)=sqrt(Vk(k)^2+((Fs*(Vk(k)-Vk(k-
                    1))))^2)/((2*pi*Fp)^2))/sqrt(2);
29)                 Z(k)=Vrms(k)/Irms(k);
30)                 theta_i(k)=atan((2*pi*Ik(k))/(Fs)*(Ik(k)-Ik(k-
                    1))));
31)                 theta_v(k)=atan((2*pi*Vk(k))/(Fs)*(Vk(k)-Vk(k-
                    1))));
32)                 theta_z(k)=theta_v(k)-theta_i(k);
33)                 error(k)=Irms(k)-Ireal;
34)                 Overcurrent(k)=logical((120/100)*Irms(k));
```

```

35)                 end
36)
37)         case 2
38)
39)             %constructing the matrices for this method only
40)             Vd1(1,NoSamples)=0;
41)             Vd2(1,NoSamples)=0;
42)             Id1(1,NoSamples)=0;
43)             Id2(1,NoSamples)=0;
44)
45)             for k=2:NoSamples-1
46)                 %calculating magnitude and angle of i & v & z
47)                 Id1(k)=(Fs/2)*(Ik(k+1)-Ik(k-1));
48)                 Vd1(k)=(Fs/2)*(Vk(k+1)-Vk(k-1));
49)                 Id2(k)=(Fs^2)*(Ik(k+1)-2*Ic(k)+Ik(k-1));
50)                 Vd2(k)=(Fs^2)*(Vk(k+1)-2*Vk(k)+Vk(k-1));
51)
52)                 Irms(k)=sqrt(((1/(2*pi*Fp))^2)*((Id1(k))^2+(Id2(k)/(2*pi*Fp))^2)
53)                 )/sqrt(2);
54)                 Vrms(k)=sqrt(((1/(2*pi*Fp))^2)*((Vd1(k))^2+(Vd2(k)/(2*pi*Fp))^2)
55)                 )/sqrt(2);
56)                 Z(k)=Vrms(k)/Irms(k);
57)                 theta_i(k)=-atan(Id2(k)/(2*pi*Fp*Id1(k)));
58)                 theta_v(k)=-atan(Vd2(k)/(2*pi*Fp*Vd1(k)));
59)                 theta_z(k)=theta_v(k)-theta_i(k);
60)                 error(k)= Irms(k)-Ireal;
61)                 Overcurrent(k)=logical((120/100)*Irms(k));
62)             end
63)
64)         case 3
65)
66)             %constructing the matrices for this method only
67)             theta(1,NoSamples)=0;
68)
69)             for k=1:NoSamples-1
70)                 %calculating magnitude and angle of i & v & z
71)                 Irms(k)=(sqrt(((Ik(1,k)^2+Ik(1,k+1)^2-
72)                 2*Ic(1,k)*Ik(1,k+1)*cos(2*pi*Fp/Fs)))/(sin(2*pi*Fp/Fs)^2))/sqrt
73)                 (2);
74)                 Vrms(k)=(sqrt(((Vk(1,k)^2+Vk(1,k+1)^2-
75)                 2*Vk(1,k)*Vk(1,k+1)*cos(2*pi*50/Fs)))/(sin(2*pi*Fp/Fs)^2))/sqrt
76)                 (2);
77)                 Z(k)=Vrms(k)/Irms(k);
78)
79)                 theta(k)=acos((Vk(1,k)*Ik(1,k)+Vk(1,k+1)*Ik(1,k+1)-
80)                 (Vk(1,k)*Ik(1,k+1)+Vk(1,k+1)*Ik(1,k))*cos(2*pi*Fp/Fs))/(Vrms(1,k)
81)                 )*Irms(1,k)*2*sin(2*pi*Fp/Fs)^2));
82)                 theta_i(k)=theta(k);
83)                 theta_z(k)=-theta(k);
84)                 error(k)=Irms(k)-Ireal;
85)                 Overcurrent(k)=logical((120/100)*Irms(k));
86)             end

```

```

76)
77)         otherwise
78)             msgbox('Invalid          Number          of          Method',
'Error','error'); %showing an error message box
79)         end
80)
81)
T=table((1:NoSamples)',Irms',theta_i',Vrms',theta_v',Z',theta_z'
,error',Overcurrent'); %construct a table for results
82)
T.Properties.VariableNames={'Sample','Irms','theta_i','Vrms','th
eta_v','Z','theta_z','error','Overcurrent'} %row names of the
table
83)
84)     subplot(2,4,1) %plotting the magnitude of rms current signal
at all samples
85)     bar(1:NoSamples,Irms);
86)     line(xlim,[Ireal,Ireal],'Color','r','LineWidth',1);
87)     xlabel('No. of Samples');
88)     ylabel('Estimated Magnitude of rms Current');
89)
90)     subplot(2,4,5) %plotting the angle of rms current signal at
all samples
91)     bar(1:NoSamples,theta_i);
92)     xlabel('No. of Samples');
93)     ylabel('Estimated Angle of rms Current');
94)
95)     subplot(2,4,2) %plotting the magnitude of rms voltage signal
at all samples
96)     bar(1:NoSamples,Vrms);
97)     xlabel('No. of Samples');
98)     ylabel('Estimated Magnitude of rms Voltage');
99)
100)    subplot(2,4,6) %plotting the angle of rms voltage signal at
all samples
101)    bar(1:NoSamples,theta_v);
102)    xlabel('No. of Samples');
103)    ylabel('Estimated Angle of rms Voltage');
104)
105)    subplot(2,4,3) %plotting the magnitude of rms impedance
signal at all samples
106)    bar(1:NoSamples,Z);
107)    xlabel('No. of Samples');
108)    ylabel('Estimated Magnitude of rms Impedance');
109)
110)    subplot(2,4,7) %plotting the angles of rms impedance signal
at all samples
111)    bar(1:NoSamples,theta_z);
112)    xlabel('No. of Samples');
113)    ylabel('Estimated Angle of rms Impedance');
114)
115)    subplot(2,4,[4 8]) %plotting the error at all samples

```

```

116)     bar(1:NoSamples,error);
117)     xlabel('No. of Samples');
118)     ylabel('Error');
119)
120) else
121)     msgbox('Invalid Symmetry for Samples', 'Error','error');
    %showing an error message box
122)
123) end

```

Results:

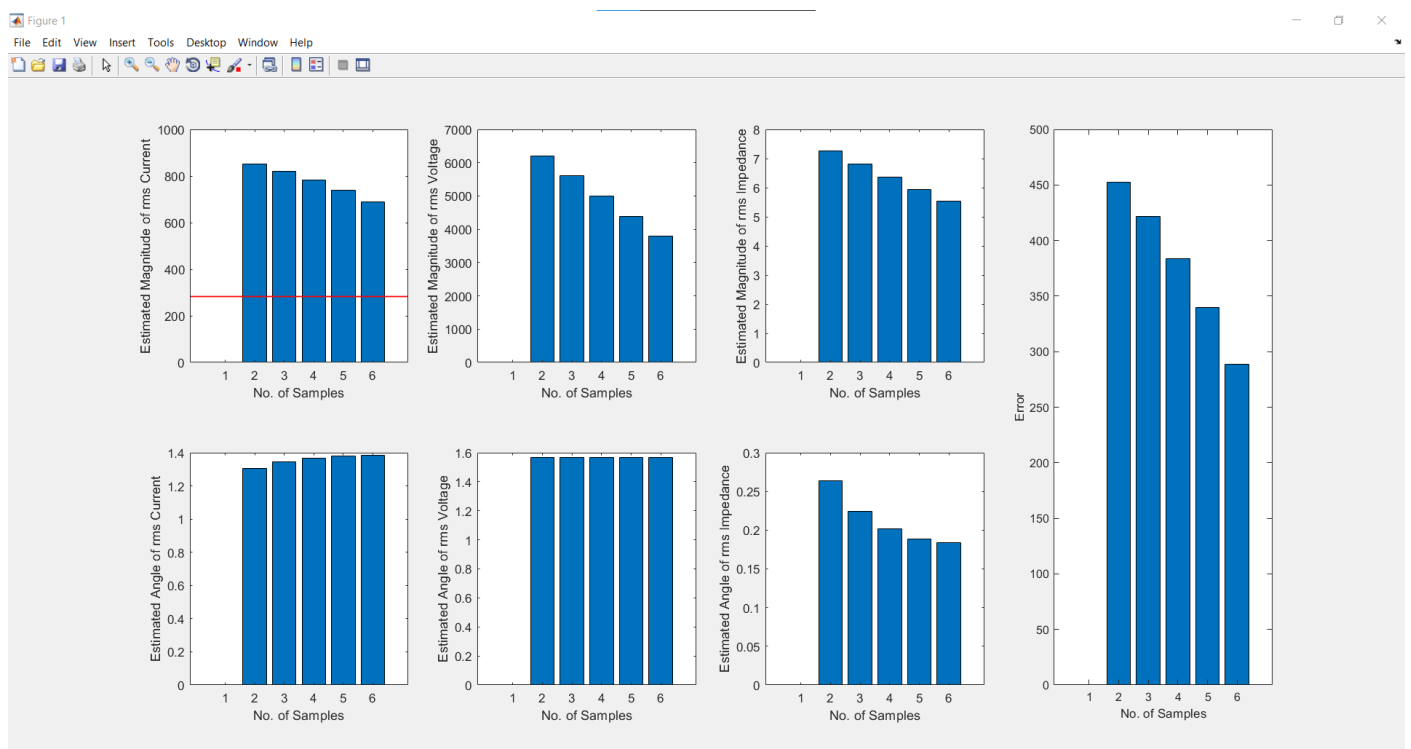
With choosing the method by choosing x, the graphs will appear in one figure and table appears in the command window showing all results.

1- at x=1:

Command Window:

Command Window								
Sample	I _k	I _{theta}	V _k	V _{theta}	Z _k	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:

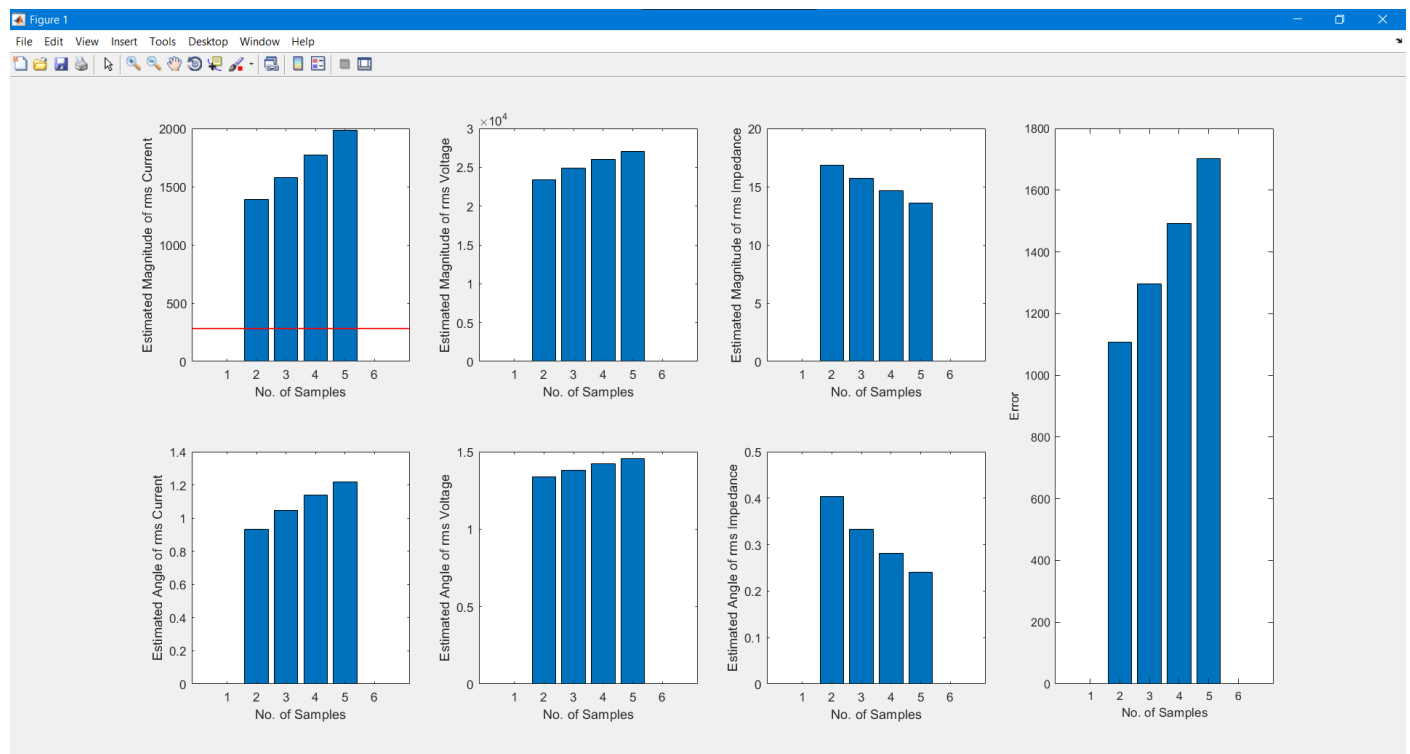


2- at x=2:

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:



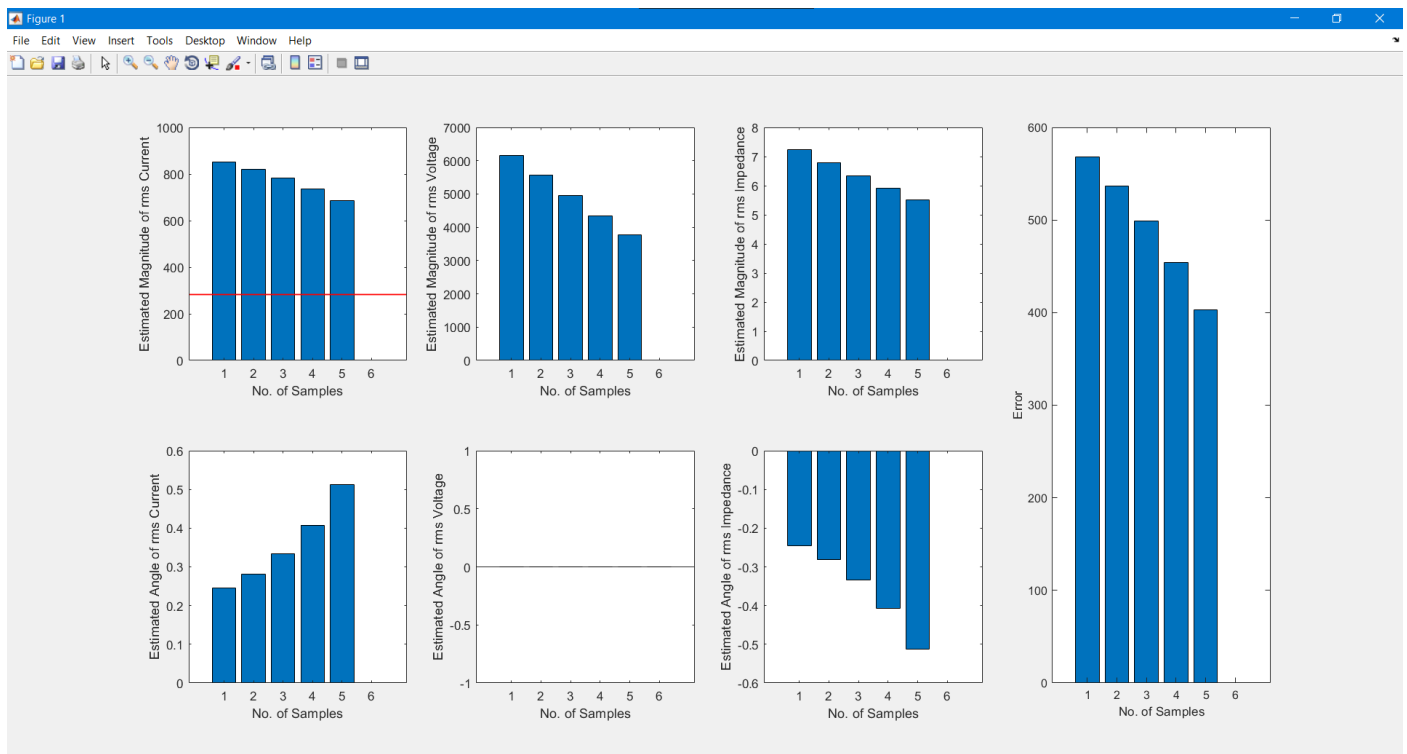
3- at x=3:

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:

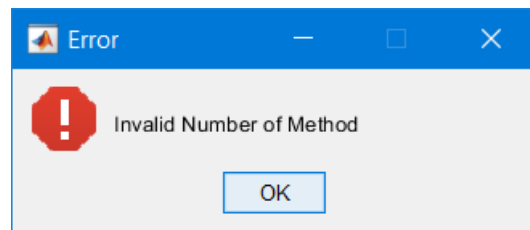


4- at x= any other number:

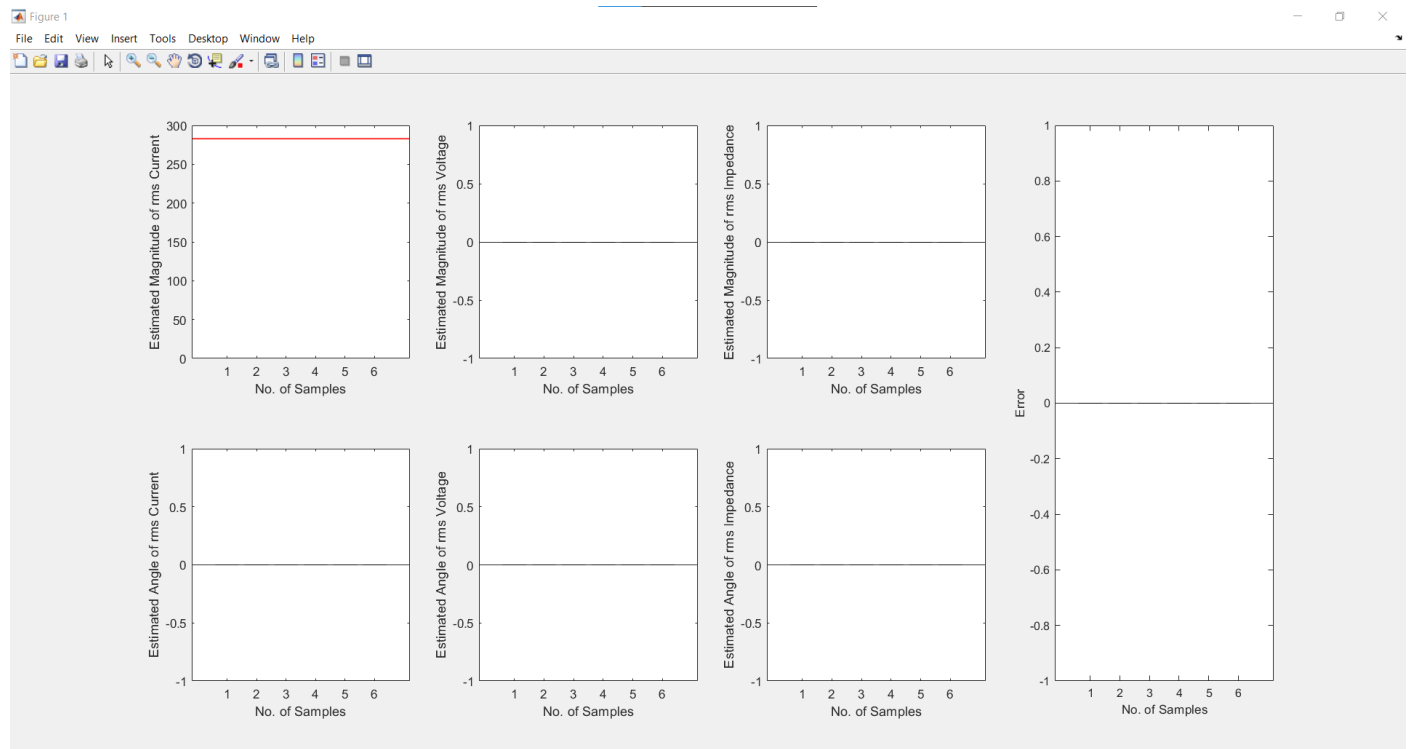
Command Window:

Command Window								
Sample	I _{rms}	theta _i	V _{rms}	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. MATLAB CODE 2 (fig file & m-file):

```
1) function varargout = Assignment_1_Complete_GUI(varargin)
2) %      ASSIGNMENT_1_COMPLETE_GUI      MATLAB      code      for
   Assignment_1_Complete_GUI.fig
3) %      ASSIGNMENT_1_COMPLETE_GUI, by itself, creates a new
   ASSIGNMENT_1_COMPLETE_GUI or raises the existing
4) %      singleton*.
5) %
6) %      H = ASSIGNMENT_1_COMPLETE_GUI returns the handle to a new
   ASSIGNMENT_1_COMPLETE_GUI or the handle to
7) %      the existing singleton*.
8) %
9) %
   ASSIGNMENT_1_COMPLETE_GUI('CALLBACK',hObject,eventData,handles,.
   ..) calls the local
10) %      function named CALLBACK in ASSIGNMENT_1_COMPLETE_GUI.M
   with the given input arguments.
11) %
12) %      ASSIGNMENT_1_COMPLETE_GUI('Property','Value',...) creates
   a new ASSIGNMENT_1_COMPLETE_GUI or raises the
13) %      existing singleton*. Starting from the left, property
   value pairs are
14) %      applied to the GUI before
   Assignment_1_Complete_GUI_OpeningFcn gets called. An
15) %      unrecognized property name or invalid value makes property
   application
16) %      stop. All inputs are passed to
   Assignment_1_Complete_GUI_OpeningFcn via varargin.
17) %
18) %      *See GUI Options on GUIDE's Tools menu. Choose "GUI
   allows only one
19) %      instance to run (singleton)".
20) %
21) % See also: GUIDE, GUIDATA, GUIHANDLES
22)
23) % Edit the above text to modify the response to help
   Assignment_1_Complete_GUI
24)
25) % Last Modified by GUIDE v2.5 13-Dec-2020 02:40:34
26)
27) % Begin initialization code - DO NOT EDIT
28) gui_Singleton = 1;
29) gui_State = struct('gui_Name',      mfilename, ...
30)                  'gui_Singleton',  gui_Singleton, ...
31)                  'gui_OpeningFcn',
   @Assignment_1_Complete_GUI_OpeningFcn, ...
32)                  'gui_OutputFcn',
   @Assignment_1_Complete_GUI_OutputFcn, ...
33)                  'gui_LayoutFcn',  [] , ...
34)                  'gui_Callback',   []);
35) if nargin && ischar(varargin{1})
```

```

36)     gui_State.gui_Callback = str2func(varargin{1});
37) end
38)
39) if nargout
40)     [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
41) else
42)     gui_mainfcn(gui_State, varargin{:});
43) end
44) % End initialization code - DO NOT EDIT
45)
46)
47) % --- Executes just before Assignment_1_Complete_GUI is made
visible.
48) function Assignment_1_Complete_GUI_OpeningFcn(hObject,
eventdata, handles, varargin)
49) % This function has no output args, see OutputFcn.
50) % hObject handle to figure
51) % eventdata reserved - to be defined in a future version of
MATLAB
52) % handles structure with handles and user data (see GUIDATA)
53) % varargin command line arguments to Assignment_1_Complete_GUI
(see VARARGIN)
54)
55) % Choose default command line output for
Assignment_1_Complete_GUI
56) handles.output = hObject;
57)
58) % Update handles structure
59) guidata(hObject, handles);
60)
61) % UIWAIT makes Assignment_1_Complete_GUI wait for user response
(see UIRESUME)
62) % uiwait(handles.figure1);
63)
64)
65) % --- Outputs from this function are returned to the command
line.
66) function varargout =
Assignment_1_Complete_GUI_OutputFcn(hObject, eventdata, handles)
67) % varargout cell array for returning output args (see
VARARGOUT);
68) % hObject handle to figure
69) % eventdata reserved - to be defined in a future version of
MATLAB
70) % handles structure with handles and user data (see GUIDATA)
71)
72) % Get default command line output from handles structure
73) varargout{1} = handles.output;
74)
75)
76) % --- Executes on button press in radiobutton1.

```

```

77) function radiobutton1_Callback(hObject, eventdata, handles)
78) % hObject      handle to radiobutton1 (see GCBO)
79) % eventdata    reserved - to be defined in a future version of
    MATLAB
80) % handles      structure with handles and user data (see GUIDATA)
81)
82) % Hint:  get(hObject,'Value') returns toggle state of
    radiobutton1
83)
84)
85) % --- Executes on button press in radiobutton2.
86) function radiobutton2_Callback(hObject, eventdata, handles)
87) % hObject      handle to radiobutton2 (see GCBO)
88) % eventdata    reserved - to be defined in a future version of
    MATLAB
89) % handles      structure with handles and user data (see GUIDATA)
90)
91) % Hint:  get(hObject,'Value') returns toggle state of
    radiobutton2
92)
93)
94) % --- Executes on button press in radiobutton3.
95) function radiobutton3_Callback(hObject, eventdata, handles)
96) % hObject      handle to radiobutton3 (see GCBO)
97) % eventdata    reserved - to be defined in a future version of
    MATLAB
98) % handles      structure with handles and user data (see GUIDATA)
99)
100) % Hint:  get(hObject,'Value') returns toggle state of
    radiobutton3
101)
102)
103) % --- Executes on button press in pushbutton1.
104) function pushbutton1_Callback(hObject, eventdata, handles)
105) % hObject      handle to pushbutton1 (see GCBO)
106) % eventdata    reserved - to be defined in a future version of
    MATLAB
107) % handles      structure with handles and user data (see GUIDATA)
108) Ireal=400/sqrt(2); %rms normal current
109) Fs=10^4; %sampling frequency
110) Fp=50; %power frequency
111) Ik=[118.21 155.77 191.77 225.86 257.7 286.94]; %current samples
112) Vk=[2933.6 3190.17 3414.94 3605.64 3760.4 3877.65]; %voltage
    samples
113) NoSamples=6; %number of samples
114)
115) if NoSamples==length(Ik) && NoSamples==length(Vk) %number of
    current and voltage samples must be equal
116)     %constructing the general matrices
117)     Irms(1,NoSamples)=0;
118)     Vrms(1,NoSamples)=0;
119)     Z(1,NoSamples)=0;

```

```

120)     theta_i(1,NoSamples)=0;
121)     theta_v(1,NoSamples)=0;
122)     theta_z(1,NoSamples)=0;
123)     error(1,NoSamples)=0;
124)     Overcurrent(1,NoSamples)=0;
125)
126)     if get(handles radiobutton1,'value')==1 %method 1
127)         disp('1. Sample and First Derivative Method :');
128)         for k=2:NoSamples %counter for calculation
129)             %calculating magnitude and angle of i & v & z
130)             Irms(k)=sqrt(Ik(k)^2+(((Fs*(Ik(k)-Ik(k-
131)                 1)))^2)/((2*pi*Fp)^2))/sqrt(2);
132)             Vrms(k)=sqrt(Vk(k)^2+(((Fs*(Vk(k)-Vk(k-
133)                 1)))^2)/((2*pi*Fp)^2))/sqrt(2);
134)             Z(k)=Vrms(k)/Irms(k);
135)             theta_i(k)=atan((2*pi*Ic(k))/(Fs)*(Ik(k)-Ik(k-
136)                 1)));
137)             theta_v(k)=atan((2*pi*Vk(k))/(Fs)*(Vk(k)-Vk(k-
138)                 1)));
139)             theta_z(k)=theta_v(k)-theta_i(k);
140)             error(k)=Irms(k)-Ireal;
141)             Overcurrent(k)=logical((120/100)*Irms(k));
142)         end
143)
144)     elseif get(handles radiobutton2,'value')==1 %method 2
145)         %constructing the matrices for this method only
146)         Vd1(1,NoSamples)=0;
147)         Vd2(1,NoSamples)=0;
148)         Id1(1,NoSamples)=0;
149)         Id2(1,NoSamples)=0;
150)         disp('2. First and Decond Derivative Method :');
151)
152)         for k=2:NoSamples-1
153)             %calculating magnitude and angle of i & v & z
154)             Id1(k)=(Fs/2)*(Ik(k+1)-Ik(k-1));
155)             Vd1(k)=(Fs/2)*(Vk(k+1)-Vk(k-1));
156)             Id2(k)=(Fs^2)*(Ik(k+1)-2*Ic(k)+Ik(k-1));
157)             Vd2(k)=(Fs^2)*(Vk(k+1)-2*Vk(k)+Vk(k-1));
158)
159)             Irms(k)=sqrt(((1/(2*pi*Fp))^2)*((Id1(k))^2+(Id2(k)/(2*pi*Fp))^2)
160)                 )/sqrt(2);
161)             Vrms(k)=sqrt(((1/(2*pi*Fp))^2)*((Vd1(k))^2+(Vd2(k)/(2*pi*Fp))^2)
162)                 )/sqrt(2);
163)             Z(k)=Vrms(k)/Irms(k);
164)             theta_i(k)=-atan(Id2(k)/(2*pi*Fp*Id1(k)));
165)             theta_v(k)=-atan(Vd2(k)/(2*pi*Fp*Vd1(k)));
166)             theta_z(k)=theta_v(k)-theta_i(k);
167)             error(k)= Irms(k)-Ireal;
168)             Overcurrent(k)=logical((120/100)*Irms(k));
169)         end
170)

```

```

164)         elseif get(handles.radiobutton3,'value')==1 %method 3
165)             %constructing the matrices for this method only
166)             theta(1,NoSamples)=0;
167)             disp('3. Two-Sample Technique :');
168)
169)             for k=1:NoSamples-1
170)                 %calculating magnitude and angle of i & v & z
171)                 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);
172)                 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);
173)                 Z(k)=Vrms(k)/Irms(k);
174)
175)                 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));
176)                 theta_i(k)=theta(k);
177)                 theta_z(k)=-theta(k);
178)                 error(k)=Irms(k)-Ireal;
179)                 Overcurrent(k)=logical((120/100)*Irms(k));
180)             end
181)         end
182)
183)         T=table((1:NoSamples)',Irms',theta_i',Vrms',theta_v',Z',theta_z'
,error',Overcurrent'); %construct a table for results
184)
185)         T.Properties.VariableNames={'Sample','Irms','theta_i','Vrms','th
eta_v','Z','theta_z','error','Overcurrent'} %row names of the
table
186)
187)         %plotting the magnitude of rms current signal at all samples
188)         axes(handles.axes1)
189)         bar(1:NoSamples,Irms);
190)         line(xlim,[Ireal,Ireal],'Color','r','LineWidth',1);
191)
192)         %plotting the angle of rms current signal at all samples
193)         axes(handles.axes2)
194)         bar(1:NoSamples,theta_i);
195)
196)         %plotting the magnitude of rms voltage signal at all samples
197)         axes(handles.axes3)
198)         bar(1:NoSamples,Vrms);
199)
200)         %plotting the angle of rms voltage signal at all samples
201)         axes(handles.axes4)
202)         bar(1:NoSamples,theta_v);
203)
204)         %plotting the magnitude of rms impedance signal at all
samples

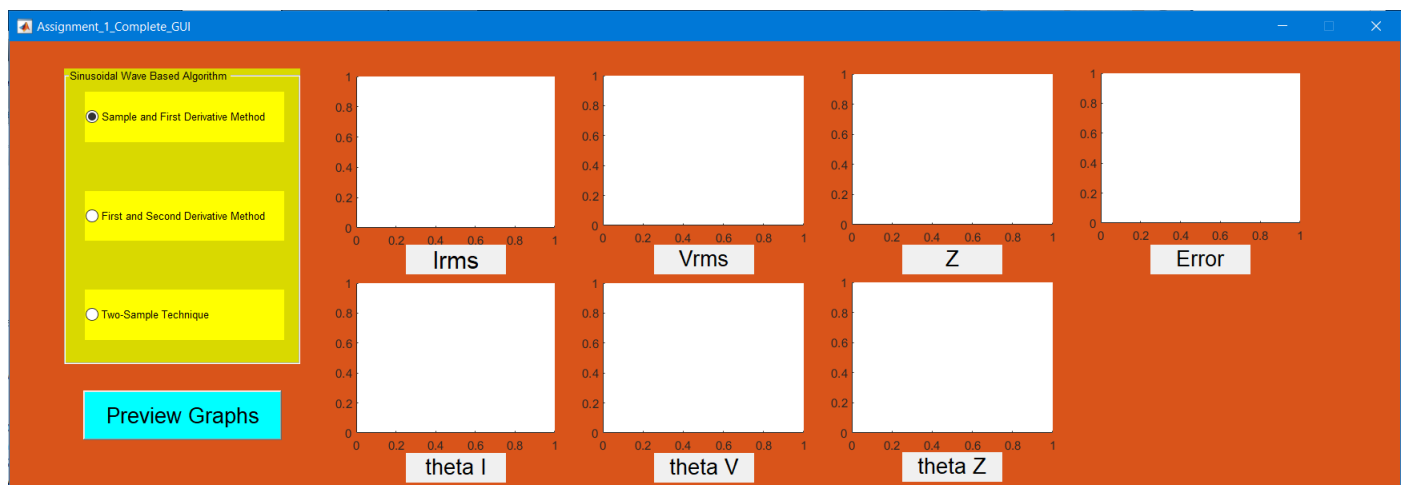
```

```

203)     axes(handles.axes5)
204)     bar(1:NoSamples,Z);
205)
206)     %plotting the angles of rms impedance signal at all samples
207)     axes(handles.axes6)
208)     bar(1:NoSamples,theta_z);
209)
210)     %plotting the error at all samples
211)     axes(handles.axes7)
212)     bar(1:NoSamples,error);
213)
214) end

```

GUI Interface:



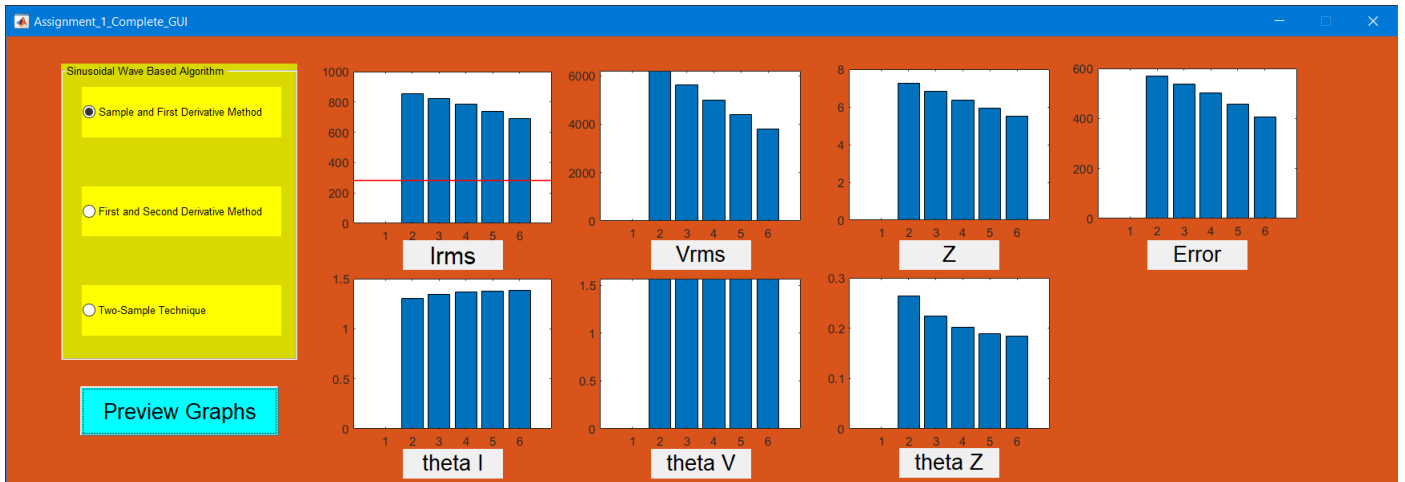
Results:

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

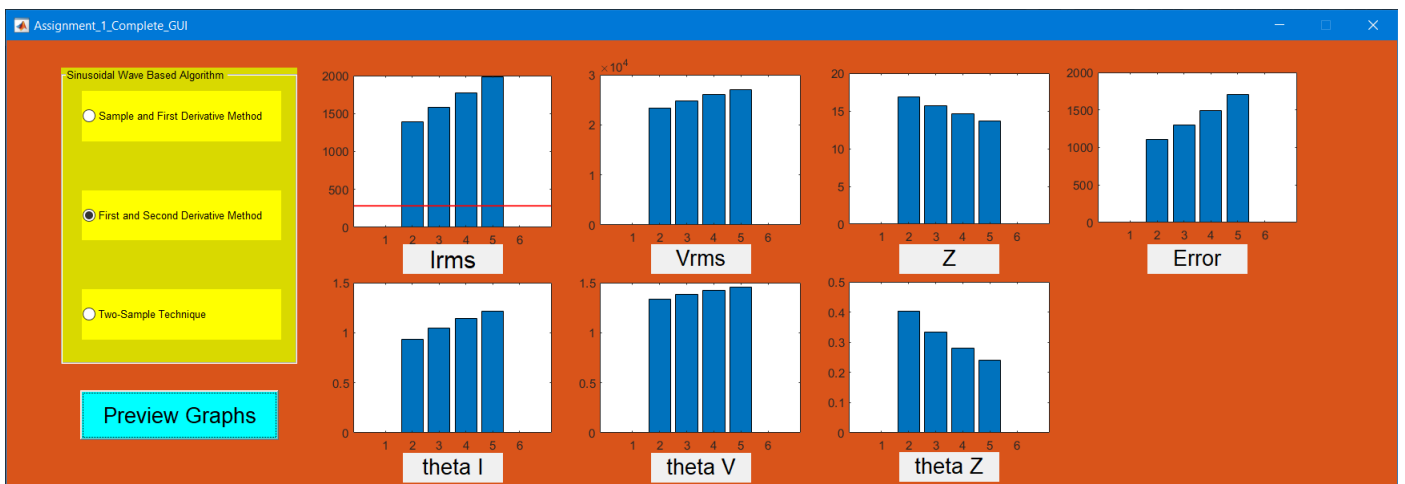
With choosing any of the three radiobuttons and pressing “Preview Graphs” pushbutton, the previous graphs will appear in the seven GUI axes and the table appears in the command window.

And the GUI states is illustrated here:

1- Method 1:



2- Method 2:



3- Method 3:

