ELEC431, ASSIGNMENT 1

Hague, Benjamin

NOVEMBER 15, 2019
THE UNIVERSITY OF LIVERPOOL

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 OBJECTIVES

The objective of this assignment is to produce a GUI used explicitly for analysing the characteristics of a Plasma discharge.

1.2 SAMPLE INPUT

The input expected is two text files with two columns, each dictated by tab delimiter. This data then features timestamps in the first column and voltages in the second one file must be converted to represent charge in coulombs.

1.3 SAMPLE OUTPUT

There are a few outputs expected from this system. Firstly, a visualisation of each of the input signals, and an analysis, (RMS, P-P Voltage and current, Max and Min voltages, etc.). Next, it is imperative to calculate power from the discharge; this can is done using a Lissajous figure.

2 SPECIFICATION ANALYSIS

The specification list is as follows outlines the expected outputs

- Acquire input from tab delimited text file (Uapp.txt and Uc.txt)
- Process Data Relating to charge (Uc.txt) 1000:1 probe and 22nF capacitor
- Plot signal data in a visual format
- Calculate Maximum, Minimum, Peak to Peak and RMS for each of the input signals
- Smooth the input signals
- Plot the average Lissajous Figure to show the power
- Calculate the average power of the Discharge (Lissajous Area)
- Display all outputs on unified GUI

The Specification of Visualisation for the results dictate that this will be a GUI driven application. This is the most appropriate output as it is easiest for the user to evaluate the data and results. A text based interface would not be as reliable for accurately evaluating the calculated results or input waveforms.

3 PROGRAM DESIGN

This section will discuss the basic principals for the design of the program outline the Call-backs linking them to the program, and a list of notable variables, Types and how data is passed within the system.

3.1 MODULES

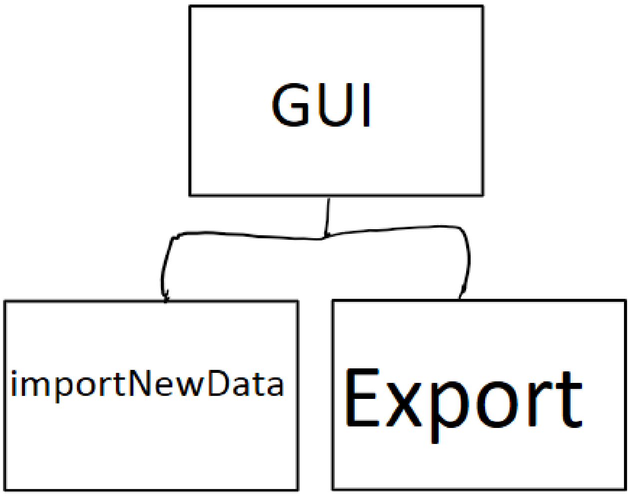

FIGURE 1, HIERARCHY CHART

The Hierarchy chart, Figure 1, shows how the program interacts between modules. The following section introduces each module and explains inputs, outputs and how they interface.

3.1.1 IMPORTNEWDATA

The Process of importing the data into the program should be a trivial task. As the software has been written in MATLAB, commands such as uigetfile can be used to acquire the file path. The file can then be saved into an array and returned to the main program. Table 1 shows the input and outputs for this module.

TABLE 1, DATA TABLE FOR IMPORTNEWDATA

Variable Name	DataType	Comment
Data	Matrix	Output, Data imported from file
filename	String	Output, The Filename data is imported from
fFormat	String	Input, The name format for input file.
File	File	Input, from uigetfile, gets the import location path and filename

The function, importNewData (7.2), allows the program to dictate the format for the file to import. This allows the program to look for a default file name, in our case this is Uapp.txt or Uc.txt. The if statement ensures that a file has been selected, and if not returns 0 for data and no filename. If the next line declares the delimiter, tab, and the number of variables, 2, this data is then read into a matrix and returned to the main program.

3.1.2 EXPORT

The process of exporting data is just as trivial as to import, simple commands (UIPutFile) can be used to get a file path from the user which can then be populated with the data in a relevant format. Table 2 shows the inputs and outputs for this module.

TABLE 2, DATA TABLE FOR EXPORT

Variable Name	DataType	Comment
Data	Matrix	Input, Data passed from program
filename	String	Output, The Filename data is exported to

File	File	Input, From uiputfile, gets the export location path and filename

The function, Export (7.3), is an open-ended exporting function. It is written to be very flexible and allow easy exporting of data. The variable 'filter' allows the possible output formats to be written for the GUI. It is important to ensure that the options available here do not contradict the possible input arguments for the writematrix command. The if statement ensures that we do not attempt to write our data to a non-existent file.

3.1.3 GUI

The GUI has a set of functions which act as a callback for a button object or as an initialiser. Any data needed across multiple methods is stored within the handles data structure, this is outlined below in Table 3.

TABLE 3, VARIABLES WITHIN THE HANDLES DATA STRUCTURE

Variable Name	DataType	Comment
Data_Uapp/Data_Uc	2x200000 matrix	From File Selected using inputNewData
Data_UappSmooth/Data_UcSmooth	2x200000 matrix	Mathematical smoothing operation on Data_Uapp/DataUc respectively
Data_UappFile/DataUcFile	String	From import new data, Contains the working file name for input data
Data_UappFreq/Data_UcFreq	Int	Holds the fundamental frequency for the input signal at all points
Data_UappAnalytics/Data_UcAnalytics	5x1 Matrix	Holds the analytic data for each signal once calculated

3.1.3.1 CALLBACKS

3.1.3.1.1 BUTTON_UAPP_CHOOSE_CALLBACK & BUTTON_UC_CHOOSE_CALLBACK

This callback processes the button press for choosing the respective input file. This works by calling the ImportNewData function, and storing the data, in a smoothed and unsmoothed format, within the handles.

3.1.3.1.2 BUTTON_UAPP_REDRAW_CALLBACK & BUTTON_UC_REDRAW_CALLBACK

This callback draws the graph on the corresponding axes when the respective button is clicked. No further calculations are done within this function, and no data other than the plots is modified. The syntax of plotting is simple, see 3.2.7.

3.1.3.1.3 BUTTON_UAPP_CALCULATE_CALLBACK & BUTTON_UC_CALCULATE_CALLBACK

This callback conducts all the basic signal analysis, Max, Min, Peak to Peak and RMS is all calculated in this function. See 3.2 to understand how the characteristics are determined. This callback executes when its respective button is pressed. The analytics is then stored within the handles data Structure.

3.1.3.1.4 BUTTON_DRAW_LISSAJOUS_CALLBACK

This callback plots the values of Uapp and Uc to the Lissajous axis after processing the input as discussed in 3.2.8, it also updates the dissipated power measurement as explained in 3.2.6.

3.1.3.1.5 EXPORT_CALLBACK

This callback combines the analytics matrices and exports the result to a user specified file type using the Export module.

3.2 SUMMARY OF MATHEMETICAL METHODS.

The methodology for processing the data is relatively simple and mostly uses built-in commands

3.2.1 MAX

To find the maximum value for each signal, the max command will be used. This returns the highest value in the array.

3.2.2 MIN

To find the minimum value for each signal, the min command will be used. This returns the lowest value in the array.

3.2.3 PEAK TO PEAK

To calculate the peak to peak voltage of the array, the minimum will be subtracted from the maximum, using the max and min commands as indicated in 3.2.1 and 3.2.2.

3.2.4 RMS

To find the RMS value for each signal, the rms command will be used. This returns the RMS value of the signal.

3.2.5 FREQUENCY

While outside of the current scope of the system. So far, the implementation of the system enables a Fourier transform to be substituted to calculate the frequency. The frequency is currently held at a static 35khz.

3.2.6 POWER

For the power, we use the Lissajous figure discussed in section 3.2.8 and use the trapz command to calculate the area of the trapezium using the smoothed average for the data.

3.2.7 PLOTTING GRAPHS

Plotting graphs within MATLAB is a trivial affair. There is a simple process for this. Select axis, Give Title and Axis names, plot the data and then set the limits if appropriate. The basic syntax for this is plot(X,Y, ‘options’) where options dictates the colour, line style and other variable options.

3.2.8 LISSAJOUS

To draw the Lissajous, the average of all the complete wavelengths is needed. The signal must be partitioned into full wavelengths and the mean calculated.

First, we find the period closest to a full wavelength. We do this by subtracting the period from each time value and finding the minimum. The index of this is the number of data points within a wavelength. The data is then reshaped to each column responding to a wavelength. The mean waveform can then be calculated for each signal. This can be smoothed using the MATLAB moving mean smooth algorithm and then the figure of charge against voltage can be plotted.

3.3 GUI DESIGN

The Design of the GUI is simple but functional. The graphic, Figure 2, labels the sets of visual elements which are explained in Table 4.

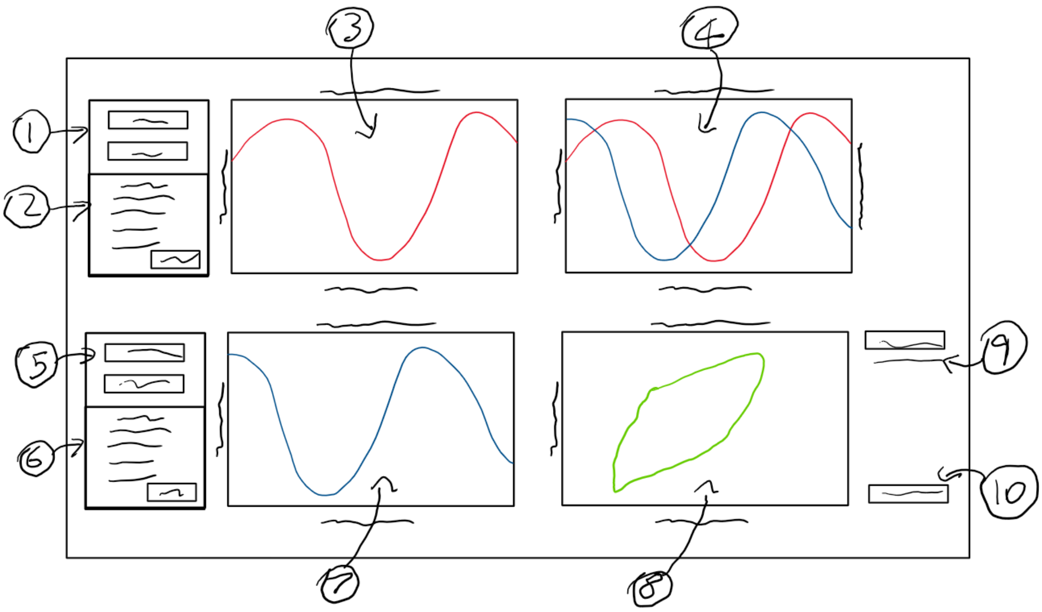

FIGURE 2, GUI DESIGN ELEMENTS

TABLE 4, GUI DESIGN ELEMENTS

Element	Description
1	This is the control panel for the Uapp input. It features 2 Buttons and an additional Section, element 2, containing the signal input features, the first button allows choosing of an input file, the second redrawing the signal
2	This is the Uapp input Features panel, it has a list of signal characteristics shown here, Vmax, Vmin, Vp-p, Vrms and Frequency are all here, also here is the button to recalculate this data.
3	This is the graph showing the high voltage signal, Axis labels are Voltage and Time and the Graph Title is "High Voltage Signal (Uapp)"
4	This is a combinational axis featuring both the charge and the voltage plotted together, the graph shown on this axis will the smoothed for easier analysis. The axis labels will be Voltage, Charge, and time. The graph title will be "Combined Smoothed Graph" the Lines will be different colours corresponding to the element 3 and 7 line colours.
5	This is the control panel for the Uc input. It features 2 Buttons and an additional Section, element 2, containing the signal input features, the first button allows choosing of an input file, the second redrawing the signal
6	This is the Uc input Features panel, it has a list of signal characteristics shown here, Qmax, Qmin, Qp-p, Qrms and Frequency are all here, also here is the button to recalculate this data.
7	This is the graph showing the capacitor charge, Axis labels are Charge and Time and the Graph Title is "Capacitor Charge (Uc)"
8	This is the graph showing a smoothed Lissajous figure, the axis labels are charge and voltage and the graph title is "Q-Uapp"
9	This is the button to Draw the Lissajous figure, Below this is the power dissipated, displayed in watts, this will be updated with the draw Lissajous figure button
10	Export Analytics Button, this will export the calculated signal features and power to an excel or CSV format for later analysis.

4 TESTING

4.1 DATA IMPORT TEST

The Data import function is simple and easy to test. The function takes input of a chosen file type or name, the Function then shows a GUI for the user to select their file. The data outputted in the event of a no file selection is 0. This is tested and shown in Figure 3

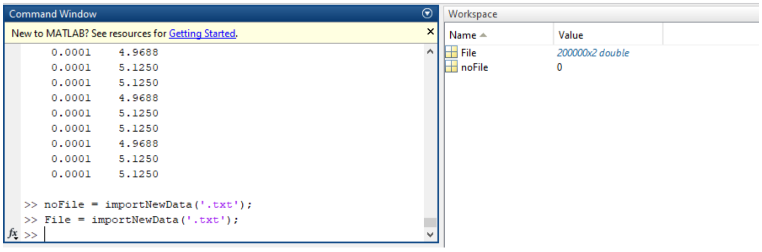

FIGURE 3, IMPORT DATA TEST

4.2 DATA EXPORT TEST

The Data Export function is simple and easy to test the function takes an input of data and shows a GUI for the user to choose their output file. The data is then exported to that file. Figure 4 shows when a file fails to be selected, and each of the export options is satisfied.

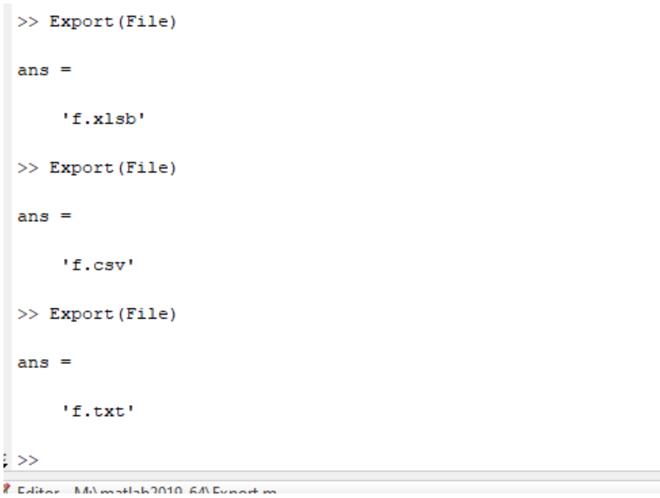

FIGURE 4, EXPORTING DATA

Each of the files has then been opened and checked for completeness: Figure 5 shows the data in the Excel file, Figure 6 shows data in the text file and Figure 7 shows data in the CSV file.

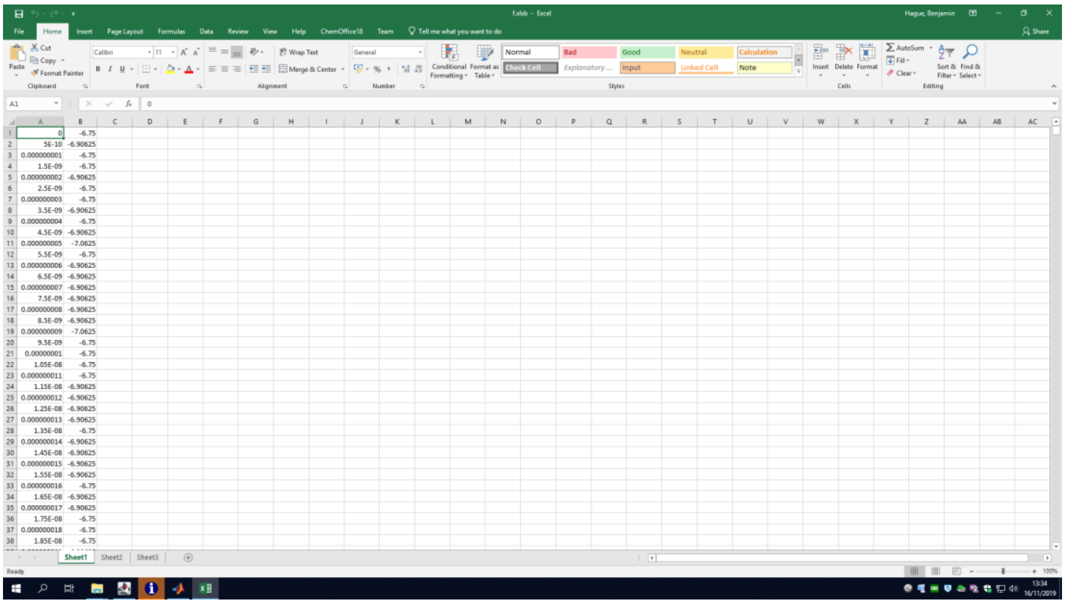

FIGURE 5, DATA EXPORT EXCEL FILE

FIGURE 6, DATA EXPORT TEXT FILE

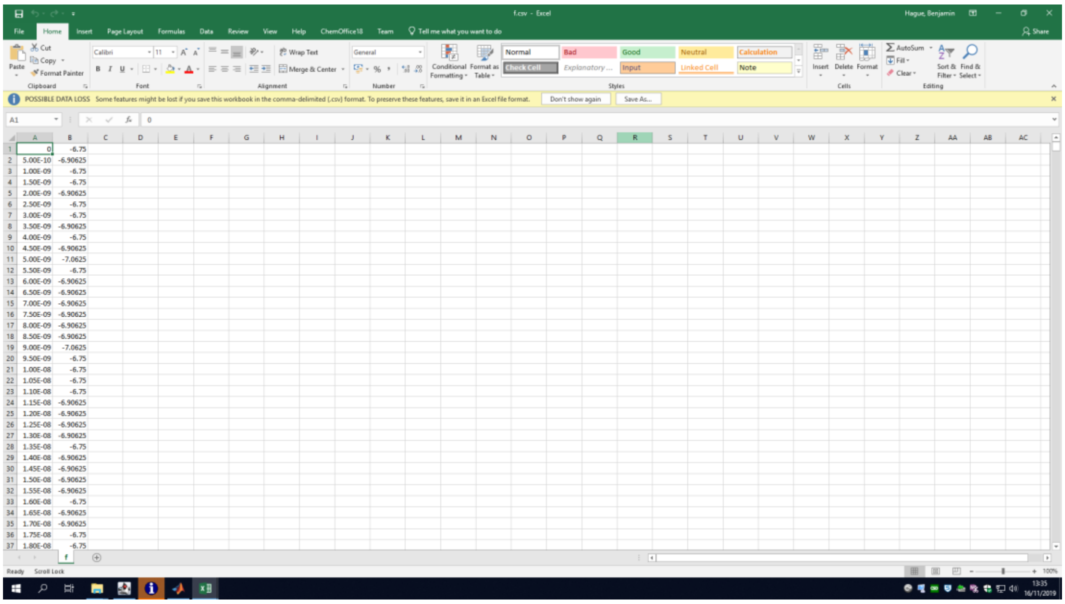

FIGURE 7, DATA EXPORT CSV FILE

4.3 PRE-DATA IMPORT TEST.

The difficult time for the processing within the code to be executed is before we have acquired any sample data. To prevent this from being an issue, the beginning placeholder for the data is constructed with a small array of zeros and a set of blank file names. Thus preventing unexpected errors from occurring, Figure 8 shows how the code is able to function reliably when no data is provided.

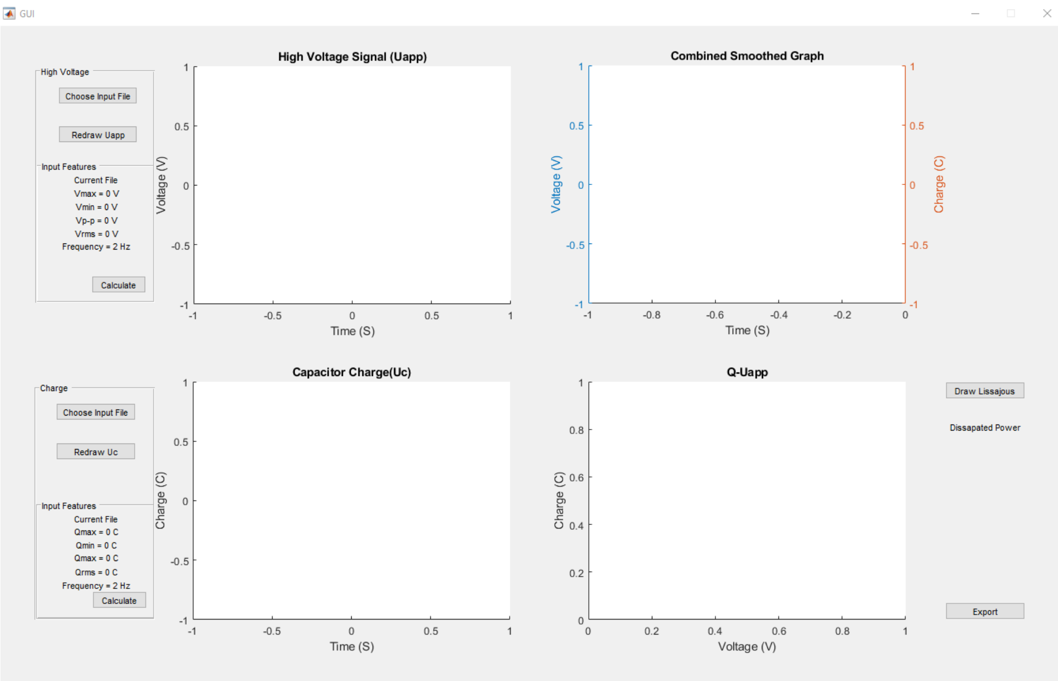

FIGURE 8, NO SPECIFIED INPUT FILES

4.4 FUNCTIONALITY TEST

We can see in Figure 9 that the output from the program is as we expect with the example input data, The power 61W and the appearance of the graphs and Lissajous figure is as we have expected.

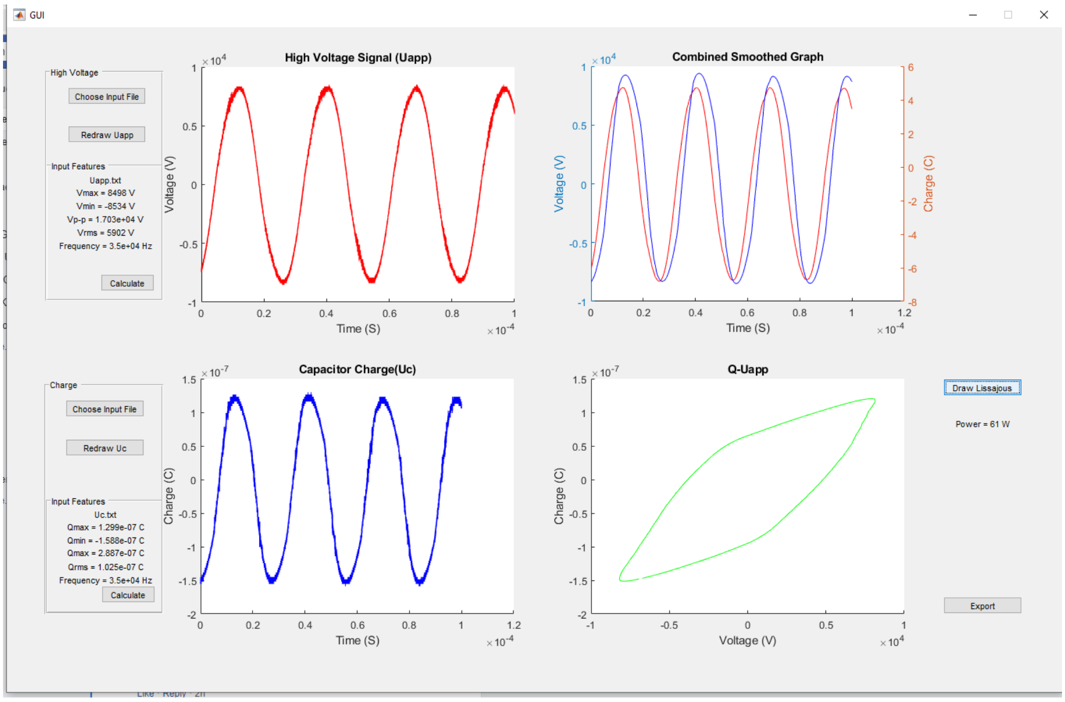

FIGURE 9, FUNCTIONAL DEMONSTRATION

5 MODIFICATIONS

Below Discusses any modifications that could be added, how they could be added and what additional functionality that will be achieved by having them

5.1 LARGER INPUT FILE SIZE

The application whilst able to handle larger input files would not currently be able to extract data from them. To extend this functionality each of the callbacks for choosing the input file would have to be modified to provide the data in a usable way to the rest of the application. This is unfortunately outside of the scope of the assignment but stands to be a possible simple program addition in the future.

5.2 FOURIER TRANSFORM TO CALCULATE FREQUENCY

The codes have specifically been written to hold variables for holding the frequency (Set in one place) this is important as it means that if a simple function to calculate the major frequencies is written than this can be added with minimal work.

6 USER MANUAL

6.1 OVERVIEW

Figure 10 Shows numbered elements listed with their function explained below in Table 5

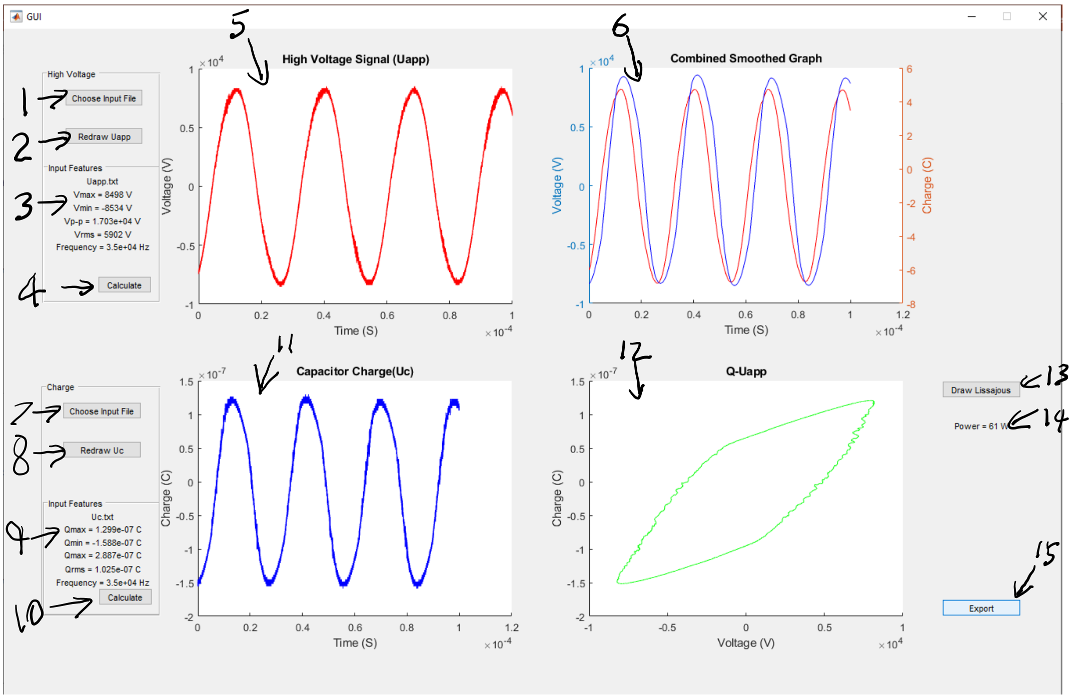

FIGURE 10, OVERVIEW OF FUNCTIONALITY

TABLE 5, EXPLANATION OF GUI ELEMENTS

Item	Classification	Function
1	Button	When this button is clicked the input selection menu appears for the High Voltage Signal. The Graphs are then plotted, and the Characteristics calculated
2	Button	This button redraws the graphic element relating to the high voltage signal on the graphs numbered 5 and 6
3	Display	This displays the characteristics related to the input signal selected from button 1
4	Button	This updates the displayed data in 3 from the input signal imported with button 1
5	Graph	This displays the raw data imported for the high voltage signal
6	Graph	This displays all data, Uapp and Uc after being smoothed
7	Button	When this button is clicked the input selection menu appears for the Capacitor Voltage the charge is then calculated with the assumption of a 22nF capacitor. The Graphs are then plotted, and the Characteristics calculated
8	Button	This button redraws the graphic element relating to the Charge signal on the graphs numbered 11 and 6
9	Display	This displays the characteristics related to the input signal selected from button 7
10	Button	This updates the displayed data in 9 from the input signal imported with button 7
11	Graph	This displays the Charge signal calculated from the imported data (Uc)

12	Graph	This displays the Lissajous figure of Uapp Voltage against Uc Charge
13	Button	This Button uses the data for Uc charge and Uapp V to plot the Lissajous figure. This also calculates power.
14	Display	This displays the power when the Lissajous figure is drawn
15	Button	This gives the option to export the analytics data for the signal inputs.

6.2 BASIC OPERATION

The Section below Covers the basic operation of the application, detailing how to do all the necessary tasks for simple signal analysis.

6.2.1 IMPORTING DATA

To import data to for each of the signals you must use Choose the input file using the buttons as shown in Figure 11, the window then appears asking you to select which data to choose Figure 12, the default file name for each section is already filled in. It does not matter what order in which the signals are imported provided both signals are imported.

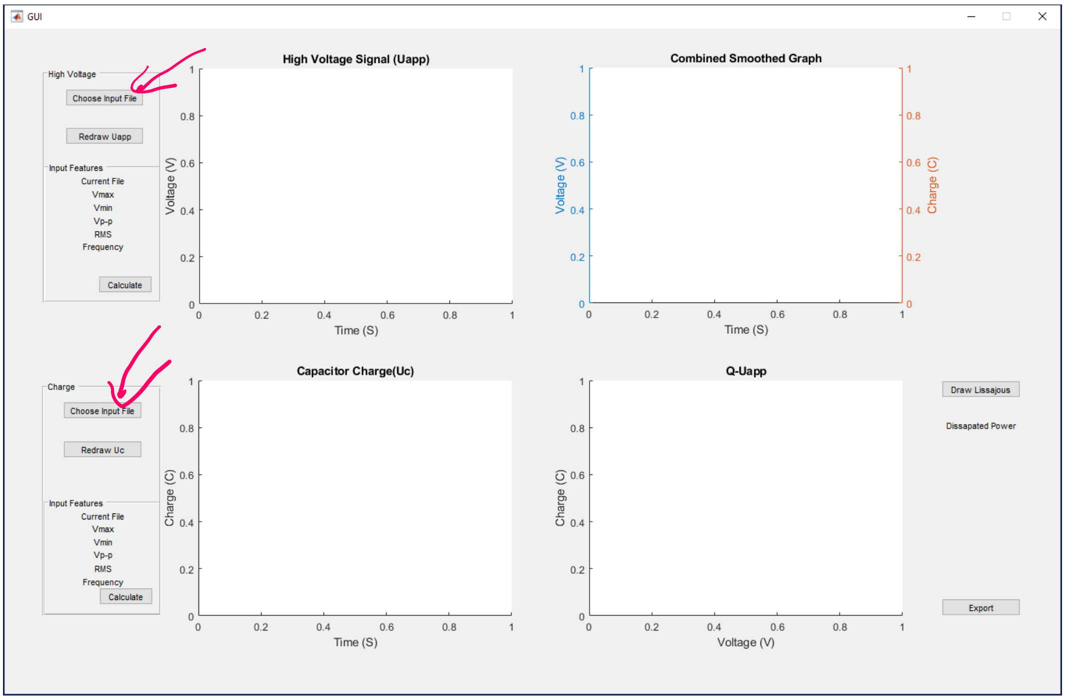

FIGURE 11, CHOOSE FILE BUTTONS

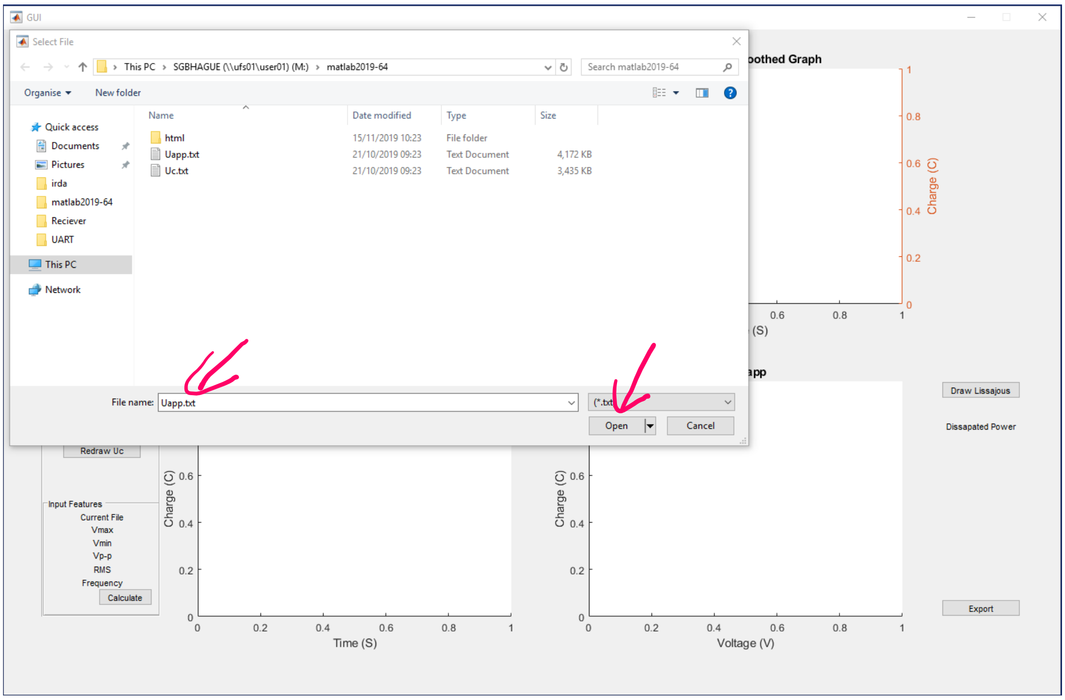

FIGURE 12, CHOOSE FILE DIALOG

6.2.2 DISPLAY GRAPHS

Upon importing the signals, the graphs corresponding to that signal should already be displayed on the screen, Figure 13 shows the buttons that can be pressed to redraw the graphs from the imported data should there be an error.

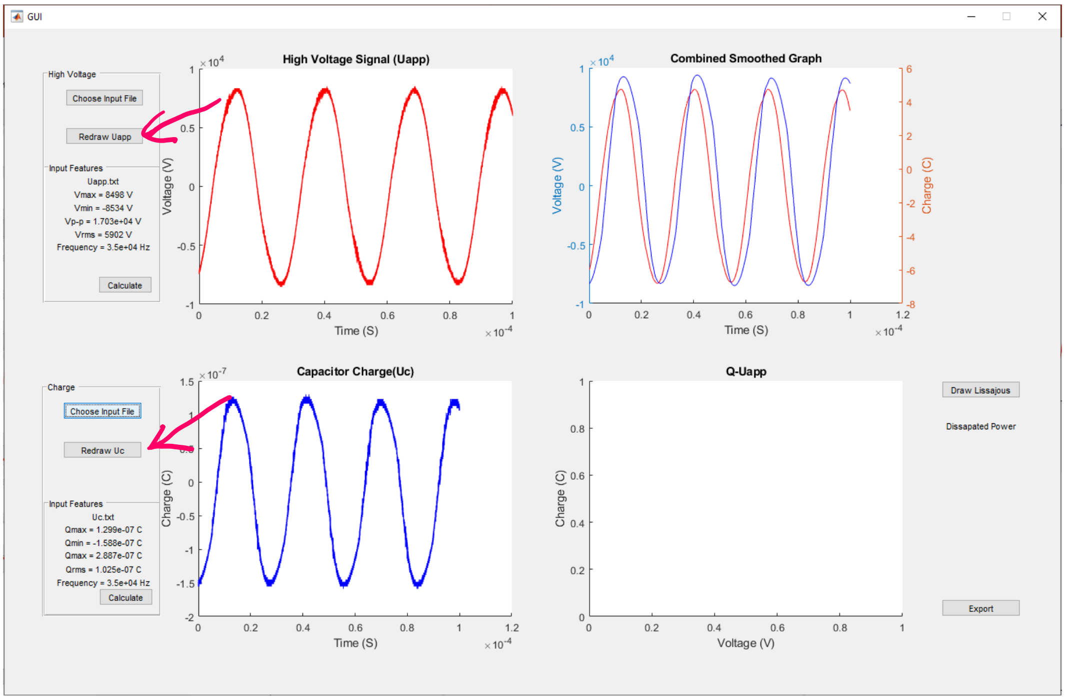

FIGURE 13, REDRAW GRAPHS BUTTONS

The Lissajous figure does not automatically plot, Figure 14 shows the button which must be pressed to plot the Lissajous figure. Both Uapp and Uc must have been chosen for the Lissajous figure to be drawn. The Lissajous graph has been drawn in Figure 15, The Power is also displayed at this point.

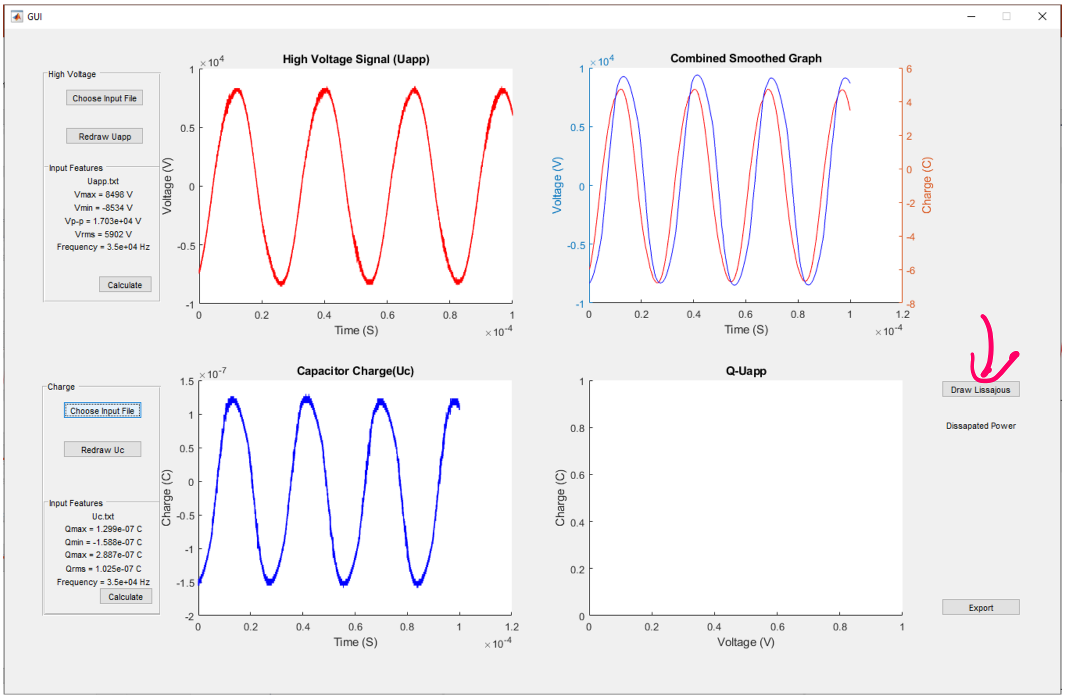

FIGURE 14, DRAW LISSAJOUS BUTTON

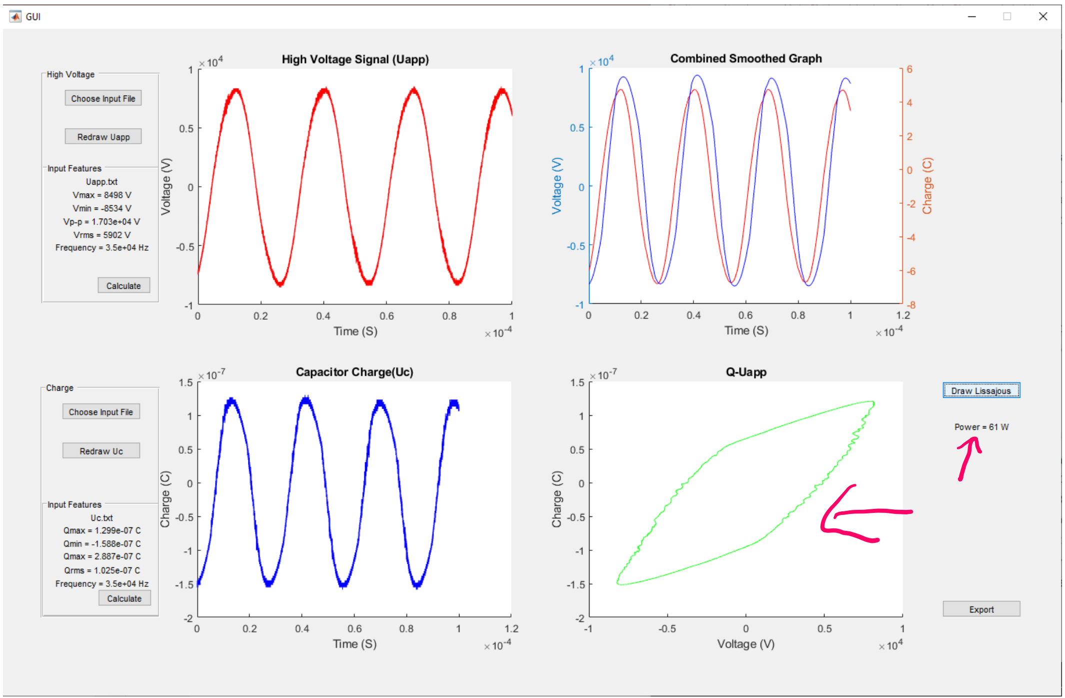

FIGURE 15, DRAWN LISSAJOUS GRAPH

6.2.3 DISPLAY PARAMETERS

Upon importing the signals, the features for each signal should already be displayed. Figure 16 shows the buttons which can be used to recalculate the signal features. This allows the analytics data to be recalculated.

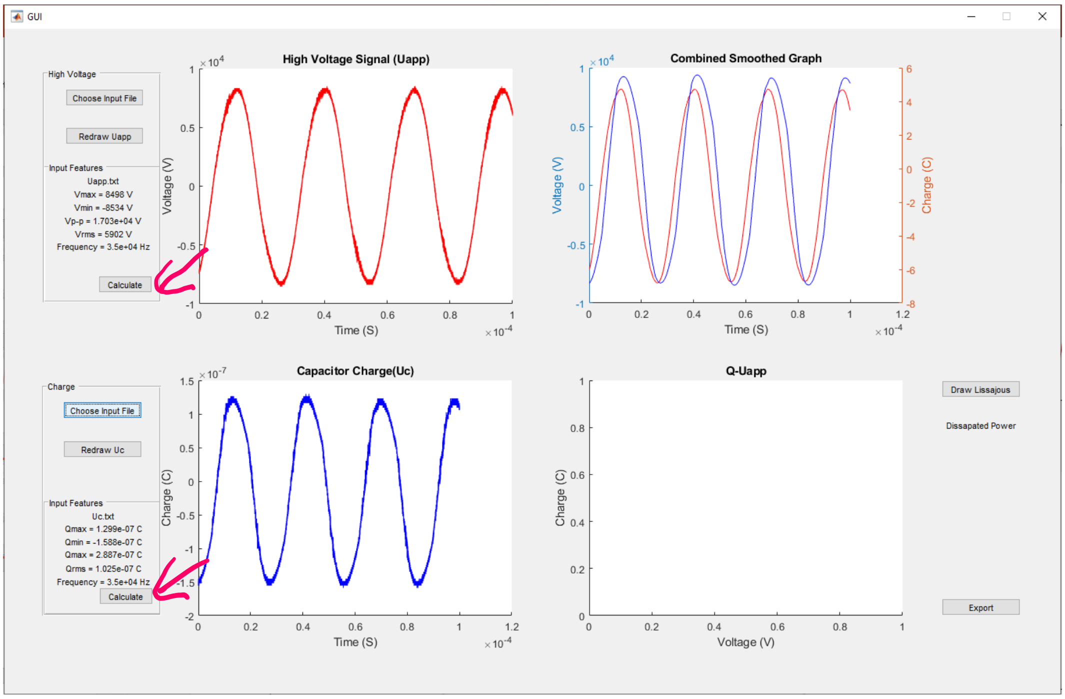

FIGURE 16, CALCULATE SIGNAL PARAMETERS BUTTONS

The Dissipated Power does not update with the import of data or recalculate buttons, This is due to both signals needing to be present to calculate the power, to resolve this, the power is calculated when the Lissajous graph is plotted as shown by Figure 17.

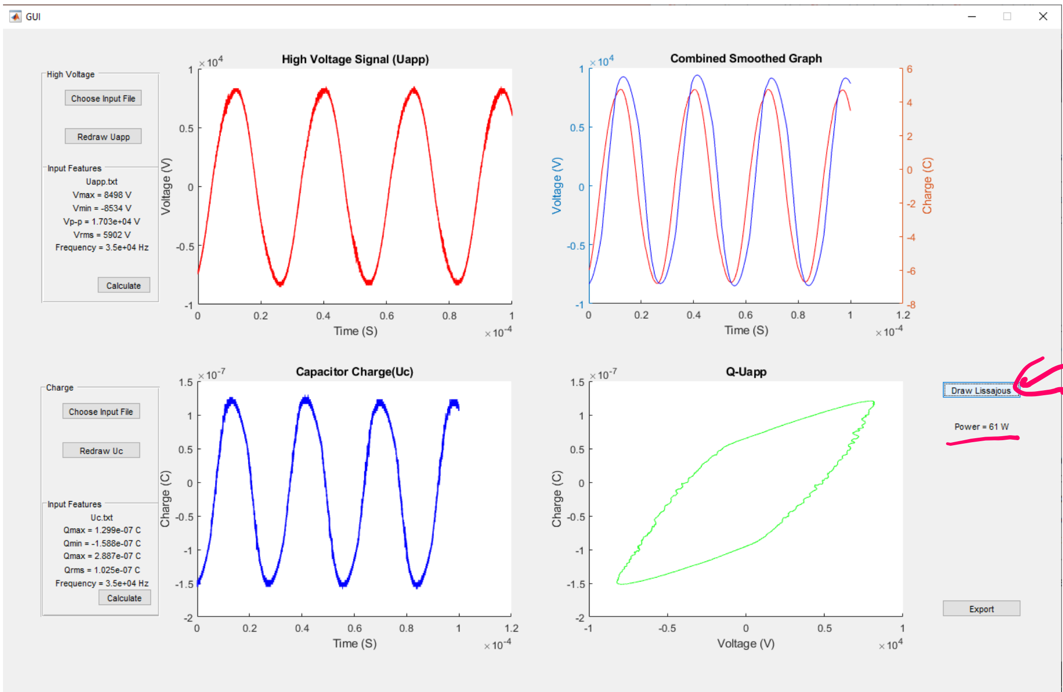

FIGURE 17, CALCULATE POWER

6.2.4 EXPORT ANALYTICS

Exporting the analytics from the application is simple using the export button Figure 18, this pops up with a display Figure 19 where the file can be named. The system currently supports export to an

Excel Document, CSV file, or a .TXT file These can be chosen using the drop-down menu shown in Figure 20.

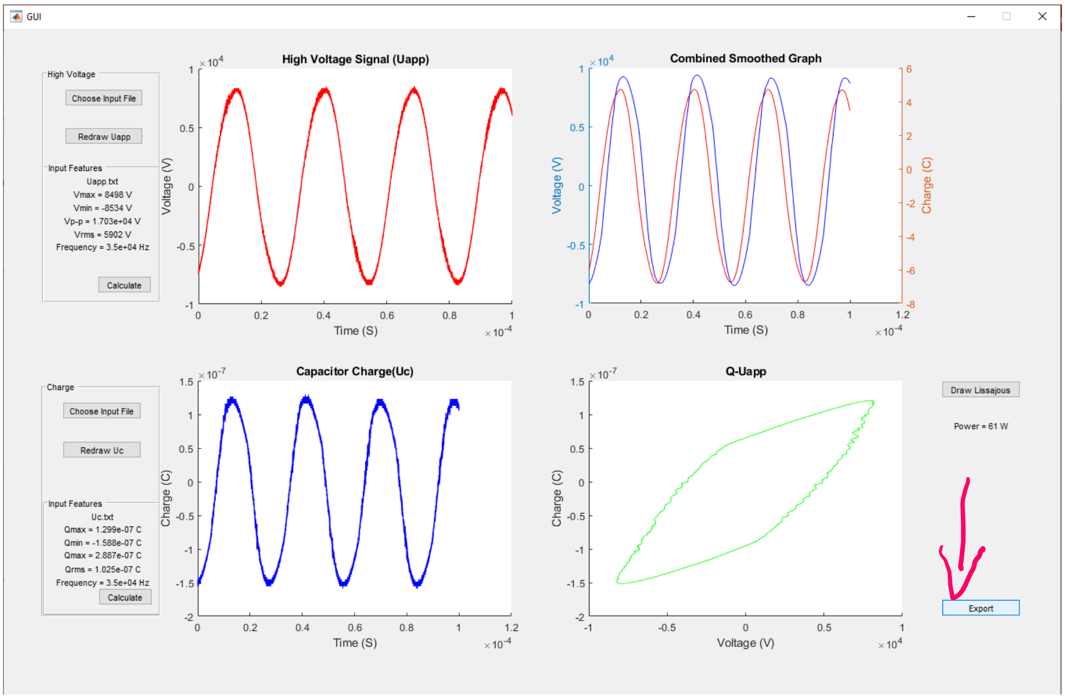

FIGURE 18, EXPORT BUTTON

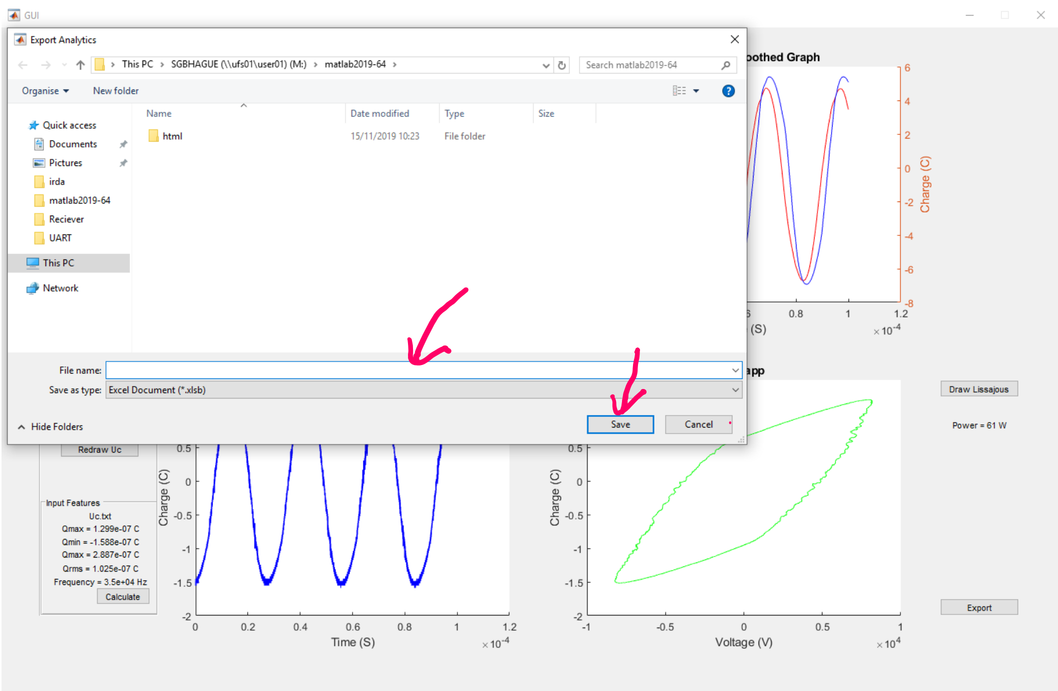

FIGURE 19, EXPORT DIALOG

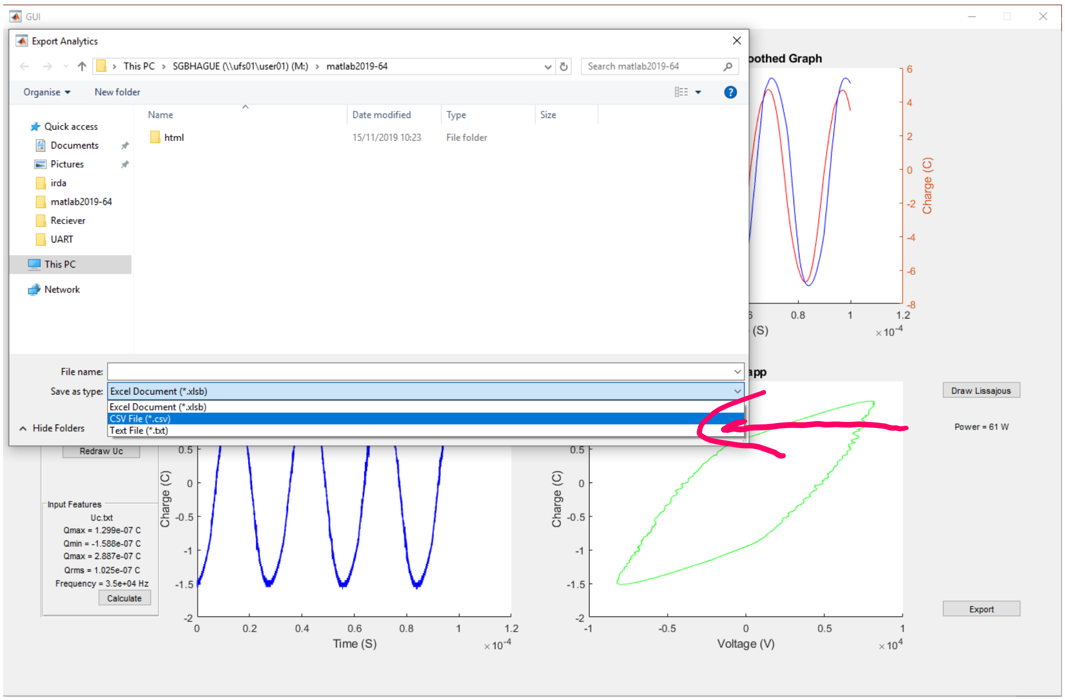

FIGURE 20, EXPORT FILE TYPE

7 APPENDIX

7.1 GUI.M

```
function varargout = GUI(varargin)
%GUI MATLAB code file for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('Property','Value',...) creates a new GUI using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to GUI_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      GUI('CALLBACK') and GUI('CALLBACK',hObject,...) call the
%      local function named CALLBACK in GUI.M with the given input
%      arguments.
%
%      *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 GUI

% Last Modified by GUIDE v2.5 16-Nov-2019 10:40:40

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @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 GUI is made visible.
function 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   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for GUI
handles.output = hObject;
% set properties for Uapp axis
axes(handles.AxisUapp);
title('High Voltage Signal (Uapp)');
```

```
xlabel('Time (S)');
ylabel('Voltage (V)');
% set properties for Uc axis
axes(handles.AxisUc);
title('Capacitor Charge(Uc)');
xlabel('Time (S)');
ylabel('Charge (C)');
% set properties for Combined axis
axes(handles.Combined);
title('Combined Smoothed Graph');
xlabel('Time (S)');
yyaxis left;
ylabel('Voltage (V)');
yyaxis right;
ylabel('Charge (C)');
% set properties for Lissajous axis
axes(handles.Lissajous);
title('Q-Uapp');
ylabel('Charge (C)');
xlabel('Voltage (V)');
% Placeholders to prevent errors
    %Default Data Value of zeros
    def = zeros(5,2);
    %Uapp Defaults
    handles.Data_Uapp = def;
    handles.Data_Uapp_smooth = def;
    handles.Data_Uappfile = ' ';
    handles.Data_Uappfreq=2;
    %Uc Defaults
    handles.Data_Uc = def;
    handles.Data_Uc_smooth = def;
    handles.Data_Ucfile = ' ';
    handles.Data_Ucfreq=2;
% Update handles structure
guidata(hObject, handles);

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

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

%%Functions for Uapp input
% --- Executes on button press in Button_Uapp_Choose.
% --- Chooses Uapp Data with GUI, Smooths it, Updates file name text box
% --- Stores data in handles for later use, Calls the Redraw and Calculate
% --- function
function Button_Uapp_Choose_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uapp_Choose (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    % import Data
    [Data,Name] = importNewData('Uapp.txt');
    % Store Data and Smoothed Data
```

```
    handles.Data_Uapp = Data;
    handles.Data_Uapp_smooth = smoothdata(Data, 'movmean',1000);
    handles.Data_Uappfile = Name;
    handles.Data_Uappfreq=35000;
    % Update File Name Text
    set(handles.Text_UappFile, 'String', sprintf(handles.Data_Uappfile));
    % Call Redraw and Calculate
    Button_Uapp_Redraw_Callback(hObject, eventdata, handles);
    Button_Uapp_Calculate_Callback(hObject, eventdata, handles);
    guidata(hObject,handles);

% --- Executes on button press in Button_Uapp_Redraw.
function Button_Uapp_Redraw_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uapp_Redraw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
    % Select Uapp Axis
    axes(handles.AxisUapp);
    % hold all labels on axis, Clear Data
    hold on
    cla;
    % Plot Data
    plot(handles.Data_Uapp(:,1),handles.Data_Uapp(:,2), '-r');
    hold off
    % Select Left Combined axis.
    axes(handles.Combined);
    yyaxis left;
    % hold all labels on axis, Clear Data
    hold on
    cla
    %Plot Data
    plot(handles.Data_Uapp(:,1),handles.Data_Uapp_smooth(:,2), '-r');
    hold off
    guidata(hObject,handles);

% --- Executes on button press in Button_Uapp_Calculate.
function Button_Uapp_Calculate_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uapp_Calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    % Calculate all the stuff with local variables
    dataMax = max(handles.Data_Uapp(:,2));
    dataMin = min(handles.Data_Uapp(:,2));
    dataPtoP = dataMax - dataMin;
    dataRMS = rms(handles.Data_Uapp(:,2));
    dataFreq = handles.Data_Uappfreq;
    % Update the Text Displaying each Variable
    set(handles.Text_UappMax, 'String',sprintf('Vmax = %0.4g V',dataMax))
    set(handles.Text_UappMin, 'String',sprintf('Vmin = %0.4g V',dataMin))
    set(handles.Text_UappPtoP, 'String',sprintf('Vp-p = %0.4g V',dataPtoP))
    set(handles.Text_UappRMS, 'String',sprintf('Vrms = %0.4g V',dataRMS))
    set(handles.Text_Uappfreq, 'String',sprintf('Frequency = %0.4g
Hz',dataFreq))
    % Store the data in handles to allow later access
    handles.Data_UappAnalytics = [dataMax, dataMin,dataPtoP,
dataRMS,dataFreq]';
    guidata(hObject,handles);


%%Functions for Uc input
```

```
% --- Executes on button press in Button_Uc_choose.
% --- Chooses Uc Data with GUI, Smooths it, Updates file name text box
% --- Stores data in handles for later use, Calls the Redraw and Calculate
% --- function
function Button_Uc_choose_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uc_choose (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    % import Data
    [Data,Name] = importNewData('Uc.txt');
    % Store Data and Smoothed Data
    handles.Data_Uc = Data;
    handles.Data_Uc(:,2) = Data(:,2).*(22e-9);
    handles.Data_Uc_smooth = smoothdata(Data, 'movmean',1000);
    handles.Data_Ucfile = Name;
    handles.Data_Ucfreq=35000;
    % Update File name Text
    set(handles.Text_UcFile, 'String', sprintf(handles.Data_Ucfile));
    % Call Redraw and Calculate
    Button_Uc_Redraw_Callback(hObject, eventdata, handles);
    Button_Uc_Calculate_Callback(hObject, eventdata, handles);
    guidata(hObject,handles);

% --- Executes on button press in Button_Uc_Redraw.
function Button_Uc_Redraw_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uc_Redraw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    %Select Uc Axis
    axes(handles.AxisUc);
    % hold all labels on axis, Clear Data
    hold on
    cla
    %Plot Data
    plot(handles.Data_Uc(:,1),handles.Data_Uc(:,2), '-b');
    hold off
    % Select Right Combined axis.
    axes(handles.Combined);
    yyaxis right;
    % hold all labels on axis, Clear Data
    hold on
    cla
    %Plot Data
    plot(handles.Data_Uc(:,1),handles.Data_Uc_smooth(:,2), '-b');
    hold off
    guidata(hObject, handles);

% --- Executes on button press in Button_Uc_Calculate.
function Button_Uc_Calculate_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Uc_Calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    % Calculate all the stuff with local variables
    dataMax = max(handles.Data_Uc(:,2));
    dataMin = min(handles.Data_Uc(:,2));
    dataPtoP = dataMax - dataMin;
    dataRMS = rms(handles.Data_Uc(:,2));
    dataFreq = handles.Data_Ucfreq;
    % Update the Text Displaying each Variable
    set(handles.Text_UcMax, 'String',sprintf('Qmax = %0.4g C',dataMax));
    set(handles.Text_UcMin, 'String',sprintf('Qmin = %0.4g C',dataMin));
```

```
    set(handles.Text_UcPtoP, 'String',sprintf('Qmax = %0.4g C',dataPtoP));
    set(handles.Text_UcRMS, 'String',sprintf('Qrms = %0.4g C',dataRMS));
    set(handles.Text_Ucfreq, 'String',sprintf('Frequency = %0.4g Hz',
dataFreq));
    % Store the data in handles to allow later access
    handles.Data_UcAnalytics = [dataMax, dataMin,dataPtoP,
dataRMS,dataFreq]';
    guidata(hObject,handles);

% Other Tasks
% --- Executes on button press in Button_Draw_Lissajous.
% --- Draws Lissajous Figure to the respective axis and Calculates power
function Button_Draw_Lissajous_Callback(hObject, eventdata, handles)
% hObject    handle to Button_Draw_Lissajous (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    %Select Lissajous Axis
     axes(handles.Lissajous);
     % Calculate the period of of the signal
     period = 1/handles.Data_Uappfreq;
     % Split the data
     data = handles.Data_Uapp(:,2);
     time = handles.Data_Uapp(:,1);
     % Find the length of 1 period of data
     [~,I] = min(abs(time-period));
     % rearrange the data to be of 1 period length
     datasample= data(1:length(data)-mod(length(data), I));
     refact = reshape(datasample,[I,length(datasample)/I]);
     % average data for each period
     X = mean(refact,2);
     % select Uc Data
     data = handles.Data_Uc(:,2);
     % Reshape to match the Average size
     datasample= data(1:length(data)-mod(length(data), I));
     refact = reshape(datasample,[I,length(datasample)/I]);
     % average the data sample
     Y = mean(refact,2);
     % keep the axis labels and title, not data
     hold on
     cla
      % Smooth and Plot Data
     plot(smoothdata(X, 'movmean',2000),smoothdata(Y, 'movmean',2000),'g');
     hold off
     %Calculate Power
     area = abs(trapz(X,Y));
     Power = area*handles.Data_Uappfreq;
     % Set dissapated Power String
     set(handles.Text_Power, 'String',sprintf('Power = %0.4g W',Power));
     guidata(hObject,handles);

% --- Executes on button press in Export.
% --- Exports Data to File using Gui
function Export_Callback(hObject, eventdata, handles)
% hObject    handle to Export (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Select Data for export
Analytics(:,1) = handles.Data_UcAnalytics;
Analytics(:,2) = handles.Data_UappAnalytics;
% Export Data
Export(Analytics);
```

7.2 IMPORTNEWDATA.M

```
function [Data, fileName] = importNewData(fFormat)
%     Select a Valid File using the File Selector GUI
    [fileName, pathName] =  uigetfile({fFormat}, 'Select File');
    if ~ischar(fileName)
        Data = 0;
        return;
    end
%     Convert the file to a readable Format
    File = fullfile(pathName,fileName);
%     Specify Specific inport options for the file
    opts = delimitedTextImportOptions("NumVariables", 2);
    opts.Delimiter = "\t";
    opts.VariableTypes = ["double", "double"];

%     Import the data as a table
    Data = readmatrix(File, opts);
    return;
```

7.3 EXPORT.M

```
function fileName = Export(data)
    %Fliter valid output file formats
    filter = {'*.xlsb', 'Excel Document';'*.csv', 'CSV File'; '*.txt',
'Text File'};
    % Get new File path
    [fileName, pathName] = uiputfile(filter, 'Export Analytics');
    % Check Filename/path is valid
    if ~ischar(fileName)
        return;
    end
    % combine path and write to file
    File = fullfile(pathName,fileName);
    writematrix(data, File)
```