85/100

Experiment 1. Programming And Data Analysis In Lab Windows.

Yi-Chun Hung[1, *] and Mostafa Alkady[1, 2]

[1]*Department of Physics, Northeastern University, Boston, Massachusetts,* 02115, *USA*

[2]*My partner*

(Dated: January 23, 2023)

In this section, we perform the analysis of a set of data by using lab windows. We read the given data from a .txt file and subtracted it with a given background value before normalizing it. After plotting the analyzed data, we store our results in another .txt file. The centroid of the data was found to be 75 with 0 background value. Compared to the results obtained from Matlab, it is confident that the data analysis done on lab windows is correct. Eventually, a comparison between using lab windows and using other languages, such as Matlab, to do the data analysis is discussed.

Keywords: *lab windows,C,GUI,Matlab,efficiency*

I. INTRODUCTION

This section of the Principle of Experimental Physics is designed to learn how to analyze data using lab windows [1]. First, we had to realize our data analysis technique, which we had mastered long ago, using a graphical user interface (GUI) generated by lab windows. More specifically, we had to link every process in data analysis into a button such that the data analysis could not be done quickly by just running the program but also needed to click on buttons to proceed with the analysis. Second, instead of using intuitive programming languages that are easy to analyze data, lab windows is based on C, requiring more non-trivial tricks to analyze the same data. Therefore, we had to recall the knowledge about C we learned in high school.

In this section, we use lab windows to create a GUI that contains buttons to do the analysis, including reading the data, subtracting the background, normalizing the data, plotting the data, and saving the analyzed data in a specific format. We read the two-column data from a .txt file into lab windows and perform the data analysis by clicking the buttons on the GUI, which the buttons are linked to the function that conducts each step in data analysis. After plotting the data, we saved the analyzed data into a specific format by clicking another button on the GUI. By comparing the results obtained from using five minutes doing the same procedure in Matlab, the analyzed data obtained from lab windows is confident to be the same as expected, which is an oscillatory line with a peak. Eventually, a comparison between lab windows and other languages, such as Matlab, to do the data analysis is discussed, along with the advantages and disadvantages of each way. The methods used in this work are introduced in chapter II. The results of this work and its discussion along with the comparison with other programming languages are given in chapter III and chapter IV, respectively. Last but not least, a summary is made in chapter V.

excellent introduction!

* hung.yi@northeastern.edu

II. METHODS

We use the lines in Fig. 7 to Fig. 12 to do the data analysis using GUI generated by lab windows. This chapter contains sub-chapters that briefly introduce the **functions** and important *variables* used in this section in an ordered sequence. Nevertheless, within seconds, more detailed and completed descriptions of these functions can be easily found in [2]. Due to the limit of the page, the images of the codes are present in Appendix A to have better quality.

A. main

The main program, in which the GUI is generated through **LoadPanel**, stored as *panelHandle* and displayed through **DisplayPanel**. After running through **RunUserInterface**, it is discarded through **DiscardPanel** (see Fig. 7).

B. non_sensegauge and grab_st

The call-back function assesses the input value and displays it in a gauge, which we use to input and display the background value (see Fig. 8), and the call-back function generates a button that uses **FileSelectPopup** to open the directory to choose the file to be opened stored as *pn*, which is opened through **OpenFile**, respectively. Then, to know the size of the data, a while loop to keep reading the data line by line is used. It counts the number of blanks *num_loop*, which is equivalent to the number of rows of the data. After that, we use **FileToArray** to read the data into a one-column array *dataxy*, of which the odd elements are the elements in the first column of the data. In contrast, the even elements are the elements in the second column of the data. Then, we use a for loop to sort the data from this one-column array into two equal-sized arrays, *x-axis* and *y-axis* (see Fig. 9).

background chosen to be zero - your data clearly is a peak with offset value. because you did not subtract an appr

C. subtract_st and normalize_st

The call-back function generates a button to subtract a background value from the data, and the call-back function generates a button to normalize the data, respectively. In the former case, the background value input in the gauge is extracted by the **GetCtrVal** and stored as x. Then, a for loop is used to subtract this value for every element in *y-axis*. In the latter case, the data is normalized so that the area between the normalized data and the x-axis is 1. To do that, we use a for loop to calculate the *area* between the data and the x-axis by adding each element in *y-axis* since the bin width equals 1 in this case. Although the boundary points should have a slight change in the bin width, it is still a good approximation for calculating the area. Another for loop is used to update each element in *y-axis* by dividing each element in *y-axis* by *area* (see Fig. 10).

D. Plot_st

The call-back function generates a button to plot the analyzed data (see Fig. 11) on the canvas *PANEL_GRAPH_2*. It uses **PlotXY** to plot *x-axis*, *y-axis* into the x-axis and the y-axis, respectively.

E. centroid_of_st

The call-back function generates a button calculating the centroid (see Fig. 12), an x-axis value that divides the data into two equal-area data sets. The centroid is calculated by using a for loop, in which the area *centroid_area* is calculated to the j^{th} element of the data at the j^{th} iteration. When the area is bigger than $\frac{1}{2}$, the if command breaks the for loop and returns j to *centroid*. Then, it is displayed through **SetCtrlVal** on a block *PANEL_NUMERIC*.

F. save_st

The call-back function generates a button saving the analyzed data (see Fig. 12), *saved_array*, as a two-column data into a .txt file *1d_st.txt* through **ArrayToFile**. the *saved_array* is a array with the first half be *x-axis* and the latter half be *y-axis*, which is constructed through two for loops.

III. RESULTS

The results of this section are shown in Fig. 1 to Fig. 4. In Fig. 1, no buttons are clicked. The title of the canvas is named "HUNG-MOSTAFA LAW," the horizontal axis is labeled as "time(s)," and the vertical axis is labeled as "A(m)." The functions of the buttons are the same as their labels. After clicking, the *GRAB DATA* button reads the data from the chosen .txt file into lab windows. The *subtract* button subtracts the data from a given background value entered in the box on its right after clicking (see Fig. 2); in this case, the background value is chosen to be 0 because we want to study the original data. The *Normalize* button normalizes the data after clicking (see Fig. 3). The *Plot_graph* button plots the analyzed data on the canvas (see Fig. 2 and Fig. 3). The *calculate centroid* button calculates the centroid, the x-value that separates the data into two equal-area data sets, and displays it in the box on its right after clicking (see Fig. 4). In this case, it is 75. The *save* button saves the analyzed data as a two-column data into a .txt file named *1d_st.txt* after clicking.

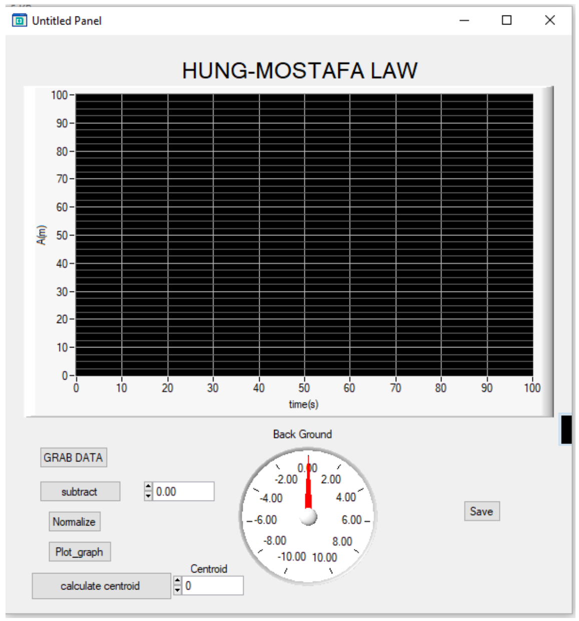


FIG. 1. The GUI before clicking any button, which takes more than six hours of coding.

IV. DISCUSSION

To benchmark our results obtained from lab windows, we spent a few minutes coding in another language that requires fewer non-trivial tricks to do the same analysis. Here, instead of Excel, I choose Matlab to do the trick because it can accomplish the same task as Excel without dramatically increasing time cost, but easier to make a more beautiful graph and get more practice in coding. The results of the data analysis through Matlab are shown in Fig. 5, where the source code is shown in Fig. 6.

With the comparison between Fig. 3 and Fig. 5, it is confident that the data analysis accomplished in lab windows is correct since the data in both graphs shows the same feature and the same data value. The centroid

6 hours to code - not a good comparison. this lab can be done in less than an hour if you know labwindows. I a

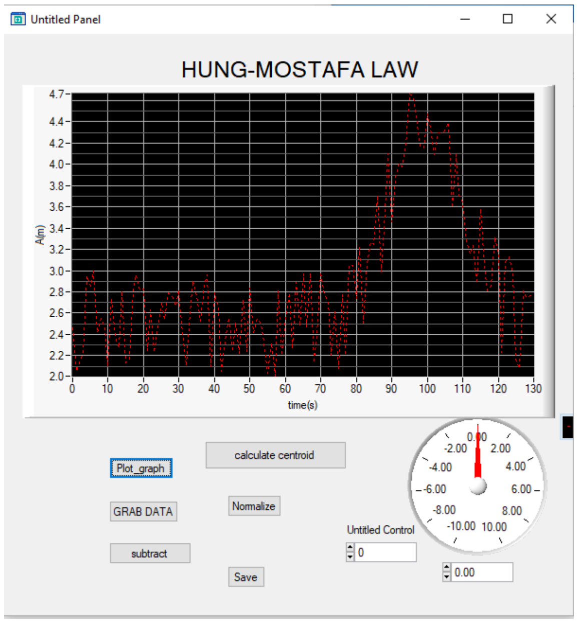


FIG. 2. The GUI after clicking on *subtract* and *Plot_graph* in sequence.

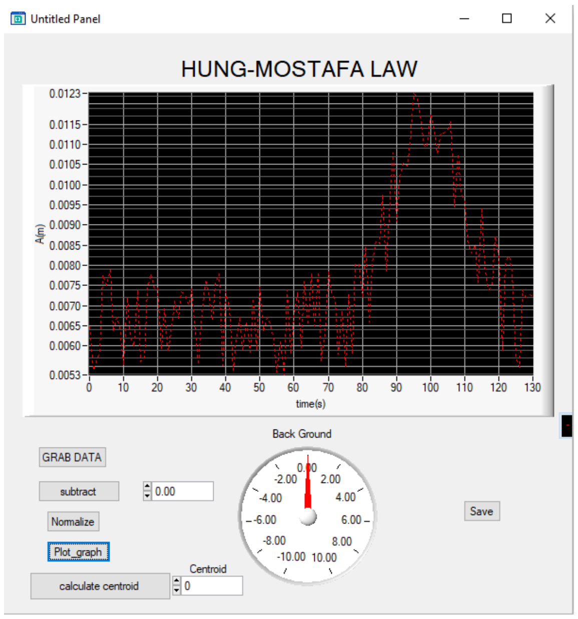


FIG. 3. The GUI after clicking on *subtract*,*Normalize*, and *Plot_graph* in sequence.

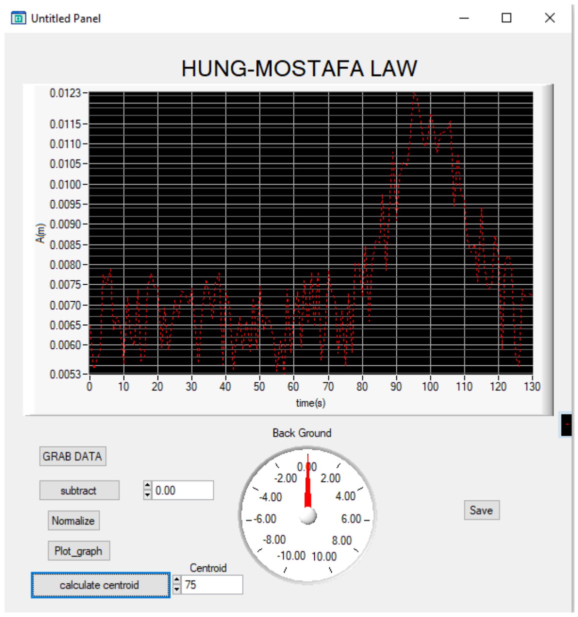


FIG. 4. The GUI from Fig. 3 after clicking on *calculate centroid.*

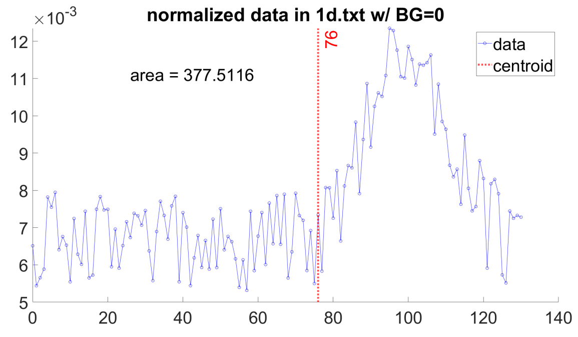


FIG. 5. The result obtained from Matlab, which takes less than five minutes from writing the first line of the code to get the final results. It's more efficient and easier than using a GUI generated by lab windows to do the data analysis.

makes sense because it needs more data in the ”flat region” to have the same area as the data around the peak. The rough approximation causes the difference between the centroid for the numerical integration in lab windows. In contrast, the centroid obtained from Matlab is correct since the numerical integration is more refined. Details about the approximation made for numerical integration in our lab windows codes can be found in chapter II.

Although lab windows and Matlab can both perform the same analysis, the time spent on and the efficiency of the process are very different. The lab windows takes more than six hours to code, while Matlab takes less than five minutes to get the final results. In this connection, it is necessary to analyze the advantages and disadvantages of analyzing data in academic natural science with lab windows and other languages. Compared to other languages, lab windows use C, which can use memory better than other languages. However, C is an out-of-date language for data analysis in academic natural science subjects such as physics, especially for experimental high-energy physics and condensed matter physics, since other modern languages have more intuitive coding grammar and more well-developed functions and packages for data analysis faster. Most of the universities in the U.S. usually have access to those modern programming languages. Further, some good and popular modern lanyou should discuss the values of the baseline and centr

guages, such as Python and Julia, are free and have a large community compared to lab windows.

On the other hand, analyzing data through lab windows is less efficient because one needs to write the analyzing codes and the codes generating GUI, which not only takes much more time but also costs unnecessary time spent on clicking buttons. GUI is indeed user-friendly; however, writing the data analysis codes as a function and calling the function through a command is also very convenient and user-friendly. It takes a little time to adjust a script into a function for most programming languages, which is much faster than generating a functional GUI. In short, for people who already can do data analysis with modern programming languages, such as a Ph.D. student in physics, doing data analysis through a GUI based on C, such as lab windows, is neither a fast nor an efficient choice. It is a method that is more suitable for people knowing nothing about coding and whose program is bought from or sponsored by a software company. For people who have already mastered some programming languages, there are much better ways to analyze data.

V. SUMMARY

In this section, we realized our mastered data analysis technique using a GUI generated by lab windows. Each step of data analysis is linked to a button such that the data analysis has to proceed with clicking buttons instead of just running the program, which includes reading the data from a two-column .txt file, subtracting the background, normalizing the data, plotting the data, and saving the analyzed results into a different two-column .txt file. Comparing the results obtained from five-minute coding on Matlab, the results obtained from six-hour coding on the lab windows is confident to be correct and the same as expected, which is an oscillatory line with a peak with a centroid located around 75 with 0 background value. Eventually, a comparison between lab windows and using other languages, such as Matlab, to do the data analysis is given. Using the GUI generated by lab windows to analyze data is neither fast nor efficient.

ACKNOWLEDGEMENT

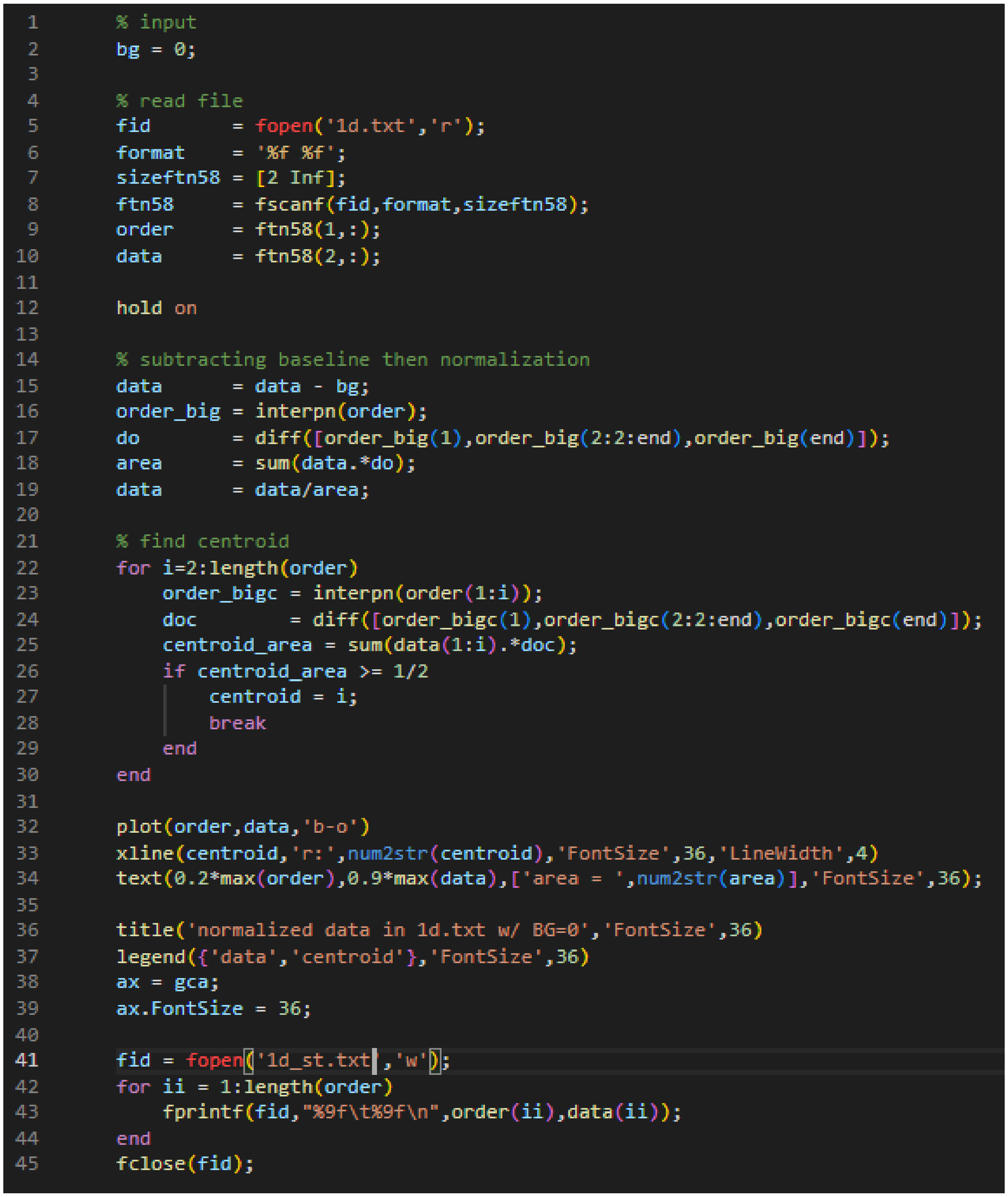
The authors thank **Professor Mark Williams**, **Ben Cashen** and **Joha Joshi**, the teaching assistant in this class, who helped us to understand some basic operations in lab windows and were patient in answering our problems. The authors also appreciate themselves and each other for sustaining considerable hours doing something they are and will not be interested in since it is never useful for theorists.

Appendix A: Codes Used In This Work

Although the link to the GitHub storing my codes is not required, I'll just put it here like normal people usually do in a journal paper `https://github.com/lengentyh/ST_Lab-Windows/tree/main` because normally people who really care about the codes want to see the codes in their editor instead of figures. Still, the screenshots of the codes I used in this section are listed below.

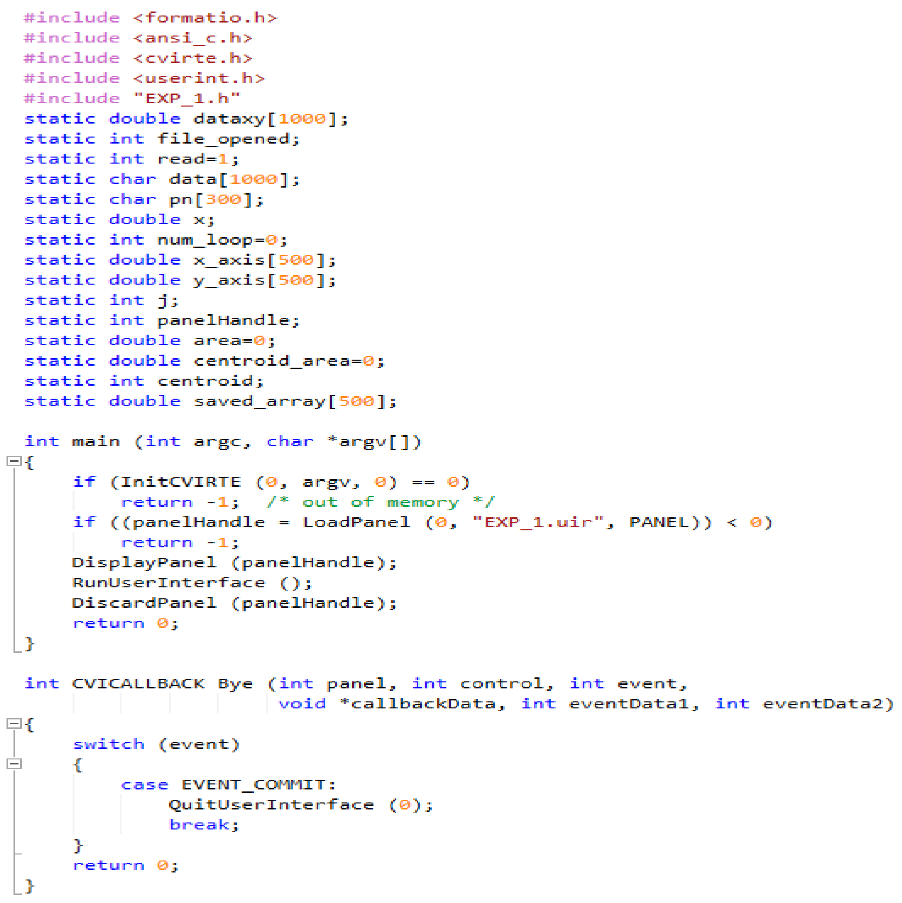
[1] N. U. Phys5318, Experiment 1: Programming and data analysis in labwindows/cvi (unpublished) (2023).
[2] https://www.ni.com/docs/en-us/bundle/labwindowscvi/page/cvi/intro.htm.

I understand that you are disappointed in the time it took to do this lab and that you will probably not directly app



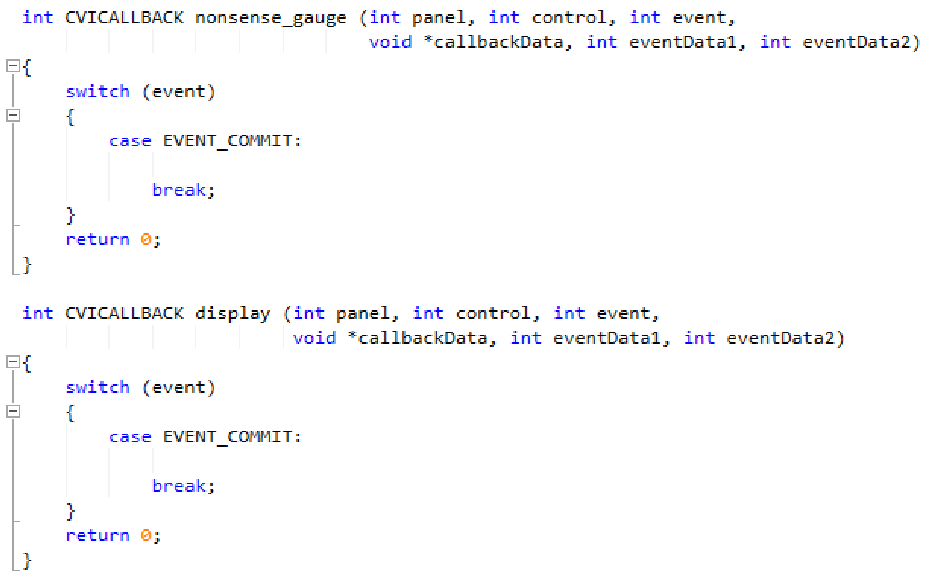
```
% input
bg = 0;

% read file
fid        = fopen('1d.txt','r');
format     = '%f %f';
sizeftn58 = [2 Inf];
ftn58      = fscanf(fid,format,sizeftn58);
order      = ftn58(1,:);
data       = ftn58(2,:);

hold on

% subtracting baseline then normalization
data       = data - bg;
order_big = interpn(order);
do         = diff([order_big(1),order_big(2:2:end),order_big(end)]);
area       = sum(data.*do);
data       = data/area;

% find centroid
for i=2:length(order)
    order_bigc = interpn(order(1:i));
    doc         = diff([order_bigc(1),order_bigc(2:2:end),order_bigc(end)]);
    centroid_area = sum(data(1:i).*doc);
    if centroid_area >= 1/2
        centroid = i;
        break
    end
end

plot(order,data,'b-o')
xline(centroid,'r:',num2str(centroid),'FontSize',36,'LineWidth',4)
text(0.2*max(order),0.9*max(data),['area = ',num2str(area)],'FontSize',36);

title('normalized data in 1d.txt w/ BG=0','FontSize',36)
legend({'data','centroid'},'FontSize',36)
ax = gca;
ax.FontSize = 36;

fid = fopen('1d_st.txt','w');
for ii = 1:length(order)
    fprintf(fid,"%9f\t%9f\n",order(ii),data(ii));
end
fclose(fid);
```

FIG. 6. The Matlab Code that generates 5, which does the same thing as the one from lab windows. However, writing the first line to get the final results takes less than five minutes, which is obviously more efficient and easier than using lab windows to create GUI to analyze data.



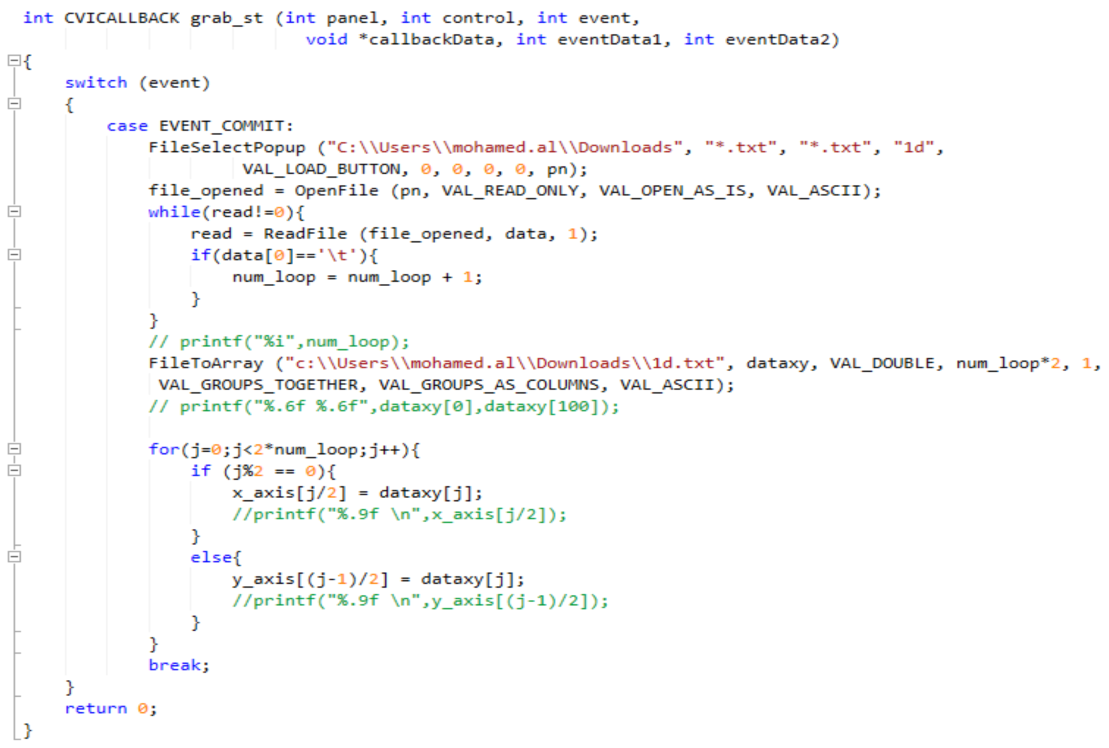
```c
#include <formatio.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "EXP_1.h"
static double dataxy[1000];
static int file_opened;
static int read=1;
static char data[1000];
static char pn[300];
static double x;
static int num_loop=0;
static double x_axis[500];
static double y_axis[500];
static int j;
static int panelHandle;
static double area=0;
static double centroid_area=0;
static int centroid;
static double saved_array[500];

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "EXP_1.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    DiscardPanel (panelHandle);
    return 0;
}

int CVICALLBACK Bye (int panel, int control, int event,
                     void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}
```

FIG. 7. The first paragraph of our lab windows source code, which contains the lines for the practice section.



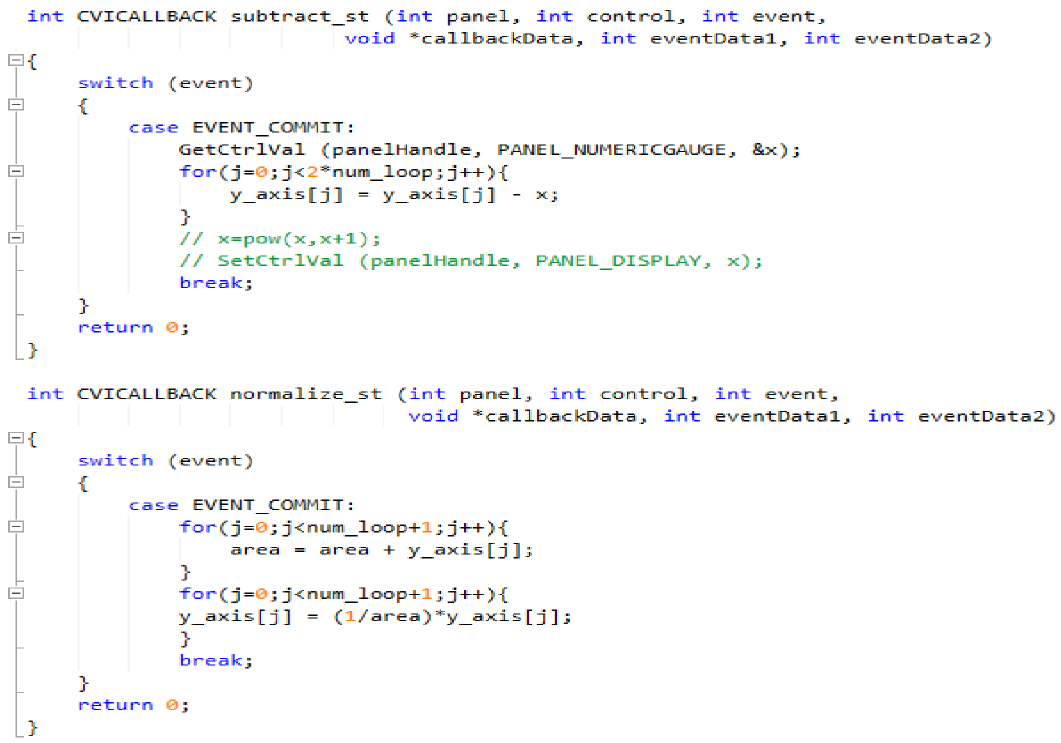
```c
int CVICALLBACK nonsense_gauge (int panel, int control, int event,
                                void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;
    }
    return 0;
}

int CVICALLBACK display (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;
    }
    return 0;
}
```

FIG. 8. The second paragraph of our lab windows source code, which contains the lines for the practice section.



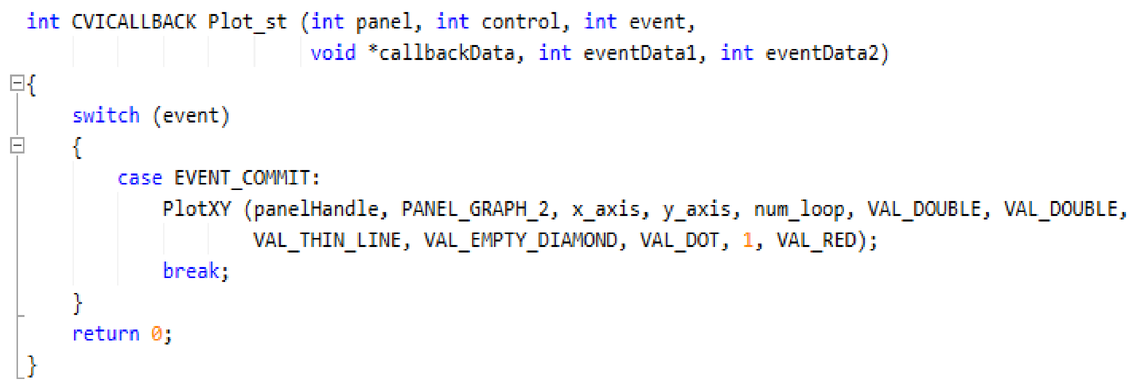
```
int CVICALLBACK grab_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            FileSelectPopup ("C:\\Users\\mohamed.al\\Downloads", "*.txt", "*.txt", "1d",
                     VAL_LOAD_BUTTON, 0, 0, 0, 0, pn);
            file_opened = OpenFile (pn, VAL_READ_ONLY, VAL_OPEN_AS_IS, VAL_ASCII);
            while(read!=0){
                read = ReadFile (file_opened, data, 1);
                if(data[0]=='\t'){
                    num_loop = num_loop + 1;
                }
            }
            // printf("%i",num_loop);
            FileToArray ("c:\\Users\\mohamed.al\\Downloads\\1d.txt", dataxy, VAL_DOUBLE, num_loop*2, 1,
             VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS, VAL_ASCII);
            // printf("%.6f %.6f",dataxy[0],dataxy[100]);

            for(j=0;j<2*num_loop;j++){
                if (j%2 == 0){
                    x_axis[j/2] = dataxy[j];
                    //printf("%.9f \n",x_axis[j/2]);
                }
                else{
                    y_axis[(j-1)/2] = dataxy[j];
                    //printf("%.9f \n",y_axis[(j-1)/2]);
                }
            }
            break;
    }
    return 0;
}
```

FIG. 9. The third paragraph of our lab windows source code, which contains the lines for the practice section.



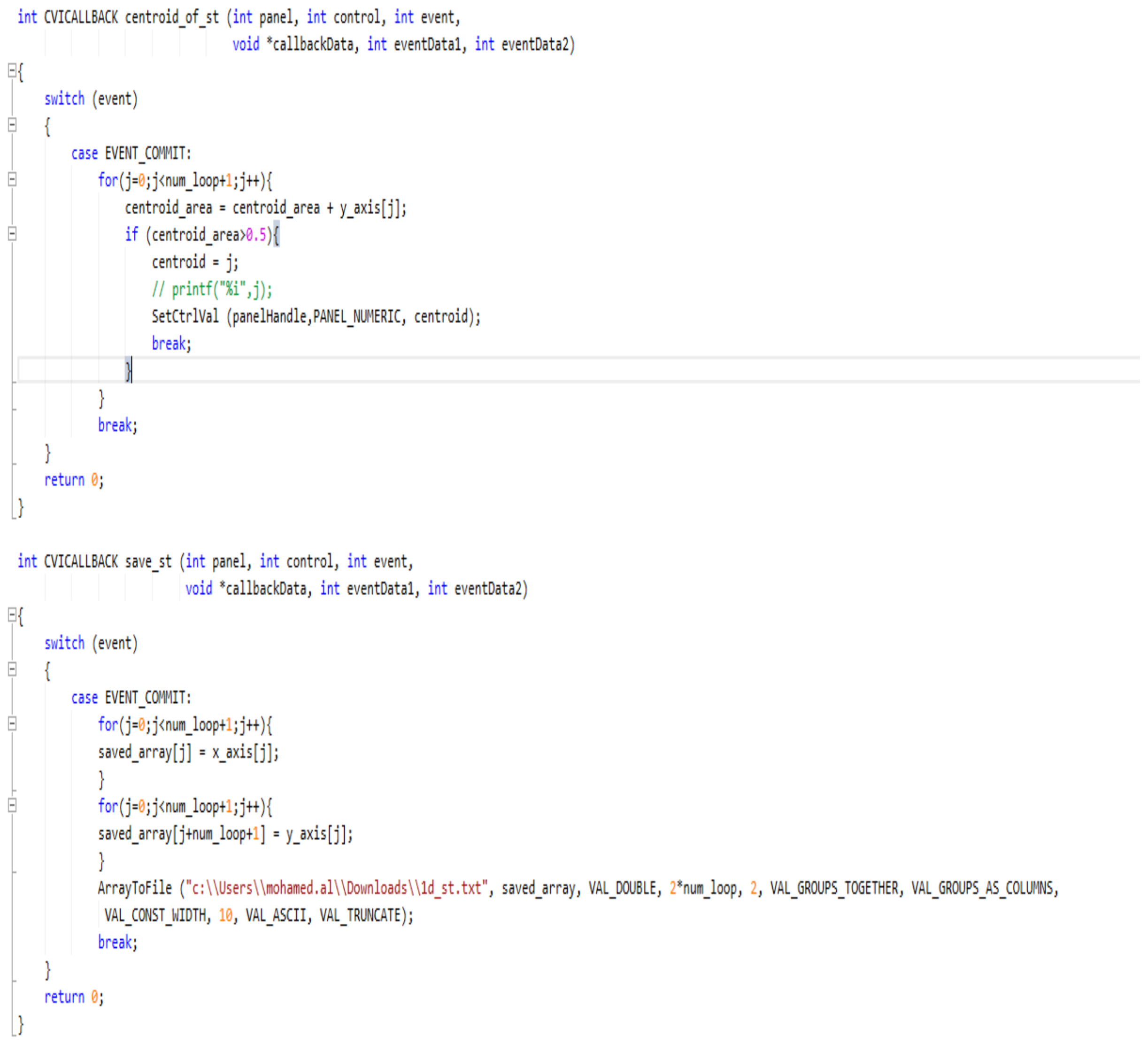
```
int CVICALLBACK subtract_st (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal (panelHandle, PANEL_NUMERICGAUGE, &x);
            for(j=0;j<2*num_loop;j++){
                y_axis[j] = y_axis[j] - x;
            }
            // x=pow(x,x+1);
            // SetCtrlVal (panelHandle, PANEL_DISPLAY, x);
            break;
    }
    return 0;
}

int CVICALLBACK normalize_st (int panel, int control, int event,
                              void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            for(j=0;j<num_loop+1;j++){
                area = area + y_axis[j];
            }
            for(j=0;j<num_loop+1;j++){
            y_axis[j] = (1/area)*y_axis[j];
            }
            break;
    }
    return 0;
}
```

FIG. 10. The fourth paragraph of our lab windows source code, which contains the lines for the practice section.



```
int CVICALLBACK Plot_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            PlotXY (panelHandle, PANEL_GRAPH_2, x_axis, y_axis, num_loop, VAL_DOUBLE, VAL_DOUBLE,
                    VAL_THIN_LINE, VAL_EMPTY_DIAMOND, VAL_DOT, 1, VAL_RED);
            break;
    }
    return 0;
}
```

FIG. 11. The fifth paragraph of our lab windows source code, which contains the lines for the practice section.



```
int CVICALLBACK centroid_of_st (int panel, int control, int event,
                                void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            for(j=0;j<num_loop+1;j++){
                centroid_area = centroid_area + y_axis[j];
                if (centroid_area>0.5){
                    centroid = j;
                    // printf("%i",j);
                    SetCtrlVal (panelHandle,PANEL_NUMERIC, centroid);
                    break;
                }
            }
            break;
    }
    return 0;
}

int CVICALLBACK save_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            for(j=0;j<num_loop+1;j++){
            saved_array[j] = x_axis[j];
            }
            for(j=0;j<num_loop+1;j++){
            saved_array[j+num_loop+1] = y_axis[j];
            }
            ArrayToFile ("c:\\Users\\mohamed.al\\Downloads\\1d_st.txt", saved_array, VAL_DOUBLE, 2*num_loop, 2, VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,
             VAL_CONST_WIDTH, 10, VAL_ASCII, VAL_TRUNCATE);
            break;
    }
    return 0;
}
```

FIG. 12. The sixth paragraph of our lab windows source code, which contains the lines for the practice section.