TANGLATEC SCHOOLSTRAAT 8 9883,
BELLEM, BELGIUM +32486257049
info@tanglatec.com www.tanglatec.com
VAT: BE0644.780.972

DETAILED WEEKLY REPORT MACHINE LEARNING AND DATA SCIENCE WEEK 4

Submitted by Nyuysemo Calglain

Under the supervision of

TANGLATECH

09/08/2024

Task

Develop a C++ function that returns the peak position of the gaussian plot for a given data set irrespective of its format.

Contents

Table of figures

1 Introduction

This report contains a detailed description of how I was able to build python modules that are able to detect the peak position of a gaussian plot.

The report also consists of the various difficulties faced, screenshots of various instances of the implementation, lessons learnt and comparisons of the values gotten from the different functions.

2 Problem statement

- Develop C++ functions to determine the peak position of a Gaussian distribution from given x, y data.
- Functions should be versatile to handle data from various sources (CSV, TXT, etc.).

3 Procedure (plan of work)

- Implement a function that will handle reading of data in any format
- Describe the data
- Implement two methods: direct maximum search and Gaussian fitting.
- Apply functions to multiple datasets provided

4 Implementation procedure

I aim to build to implement two functions that will both return the maximum position and the max value at this position in c++ and to do this, I firstly created two libraries **calcMax.h** and **gaussianFitMax.h** that calculate the maximum value directly and using the guassian fit method respectively. In each library the necessary libraries were imported based on the operations to be carried out by the functions of the library.

5 Data importation

To do any calculation I have firstly import my data in any desired format as seen below **(this is just a recap of what was done in week 2 skip the next 2 pages if you already know this).**

I created a folder called ETL then

proceeded by creating **the ETL class** This class will permit us to read data from a csv file.

In each file we create we will import the necessary packages we are going to use some of them being **iostream, string,**

boost: used for input and output functions,

Eigen: it's a library for matrices, linear algebra and vectors,

Vectors: permits us to declare and use vector arrays that is arrays where we don't know the size yet

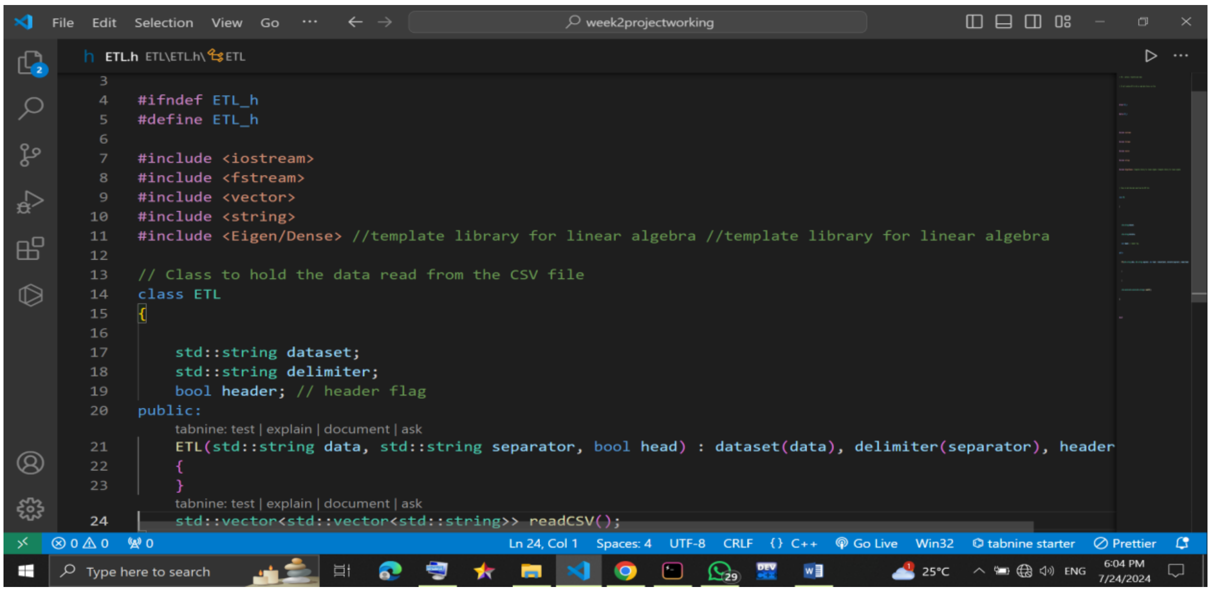
Fstream: used for file input and output.

Our ETL class has 3 attributes. The dataset that will collect our dataset in csv format, the delimiter or simply the separator of the x and y components and the header (header flag) to determine whether our data has a header or not. **We then create a constructor for this class**

In this class we also declare an instance of our function readCSV ()

That will read the csv file. as shown on the screenshot below.

Screenshot:



```
#ifndef ETL_h
#define ETL_h

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <Eigen/Dense> //template library for linear algebra //template library for linear algebra

// Class to hold the data read from the CSV file
class ETL
{

    std::string dataset;
    std::string delimiter;
    bool header; // header flag
public:
    tabnine: test | explain | document | ask
    ETL(std::string data, std::string separator, bool head) : dataset(data), delimiter(separator), header
    {
    }
    tabnine: test | explain | document | ask
    std::vector<std::vector<std::string>> readCSV();
```

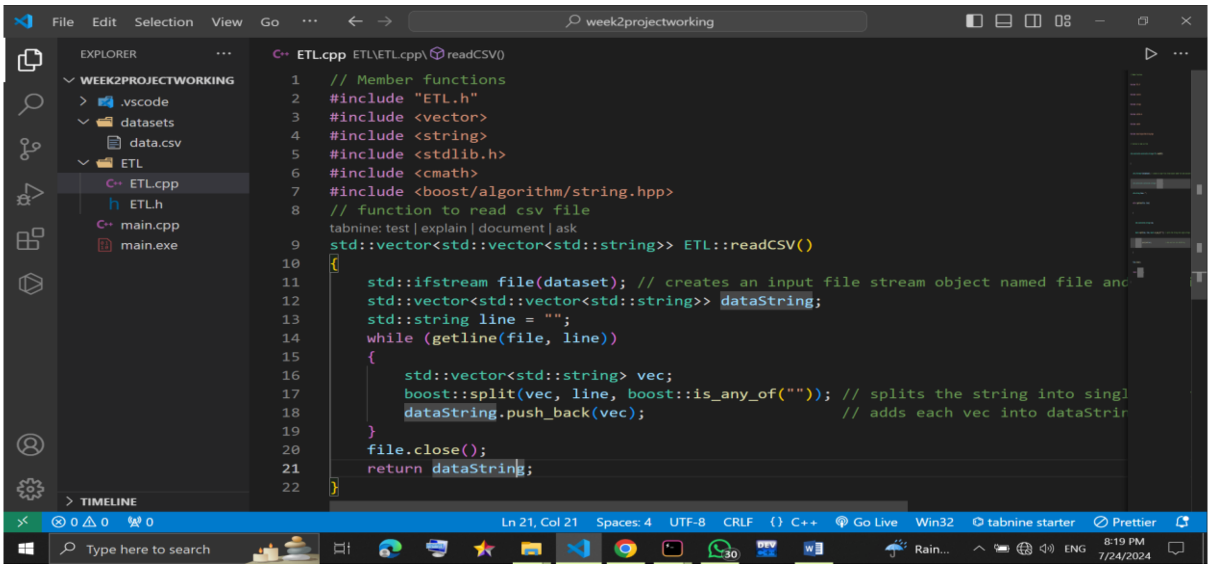
Figure 1 ETL.h file day 3

We proceed by creating the file ETL.cpp in which we will build the functions for our ETL member class.

We import "ETL.h "header file and all the necessary libraries. Then we call an instance on the readCSV() function from the ETL library which returns a 2D vector string (data set) in csv format.

In this function we declare the various parameters that we will use such as dataString parameter which returns the actual data set in csv format. The code implementation is seen in the implementation below.

Screenshot:



```
// Member functions
#include "ETL.h"
#include <vector>
#include <string>
#include <stdlib.h>
#include <cmath>
#include <boost/algorithm/string.hpp>
// function to read csv file
std::vector<std::vector<std::string>> ETL::readCSV()
{
    std::ifstream file(dataset); // creates an input file stream object named file and
    std::vector<std::vector<std::string>> dataString;
    std::string line = "";
    while (getline(file, line))
    {
        std::vector<std::string> vec;
        boost::split(vec, line, boost::is_any_of("")); // splits the string into singl
        dataString.push_back(vec);                     // adds each vec into dataStrir
    }
    file.close();
    return dataString;
}
```

Figure 2 ETL.cpp file day 3

I created another folder called datasets and pasted my data set in the csv format as can be seen on the screenshot above.

Disclaimer: I tested this function for data only with the .csv, and .tsv file formats for now.

6 Data visualization

I used gnuplot to display the particular data set I decided to work with so I can already guess what maximum value will look like and the position where it's found so when I get my calculated values I will be able to know whether these values are close to accurate.

The plot of this dataset is

Screenshot:

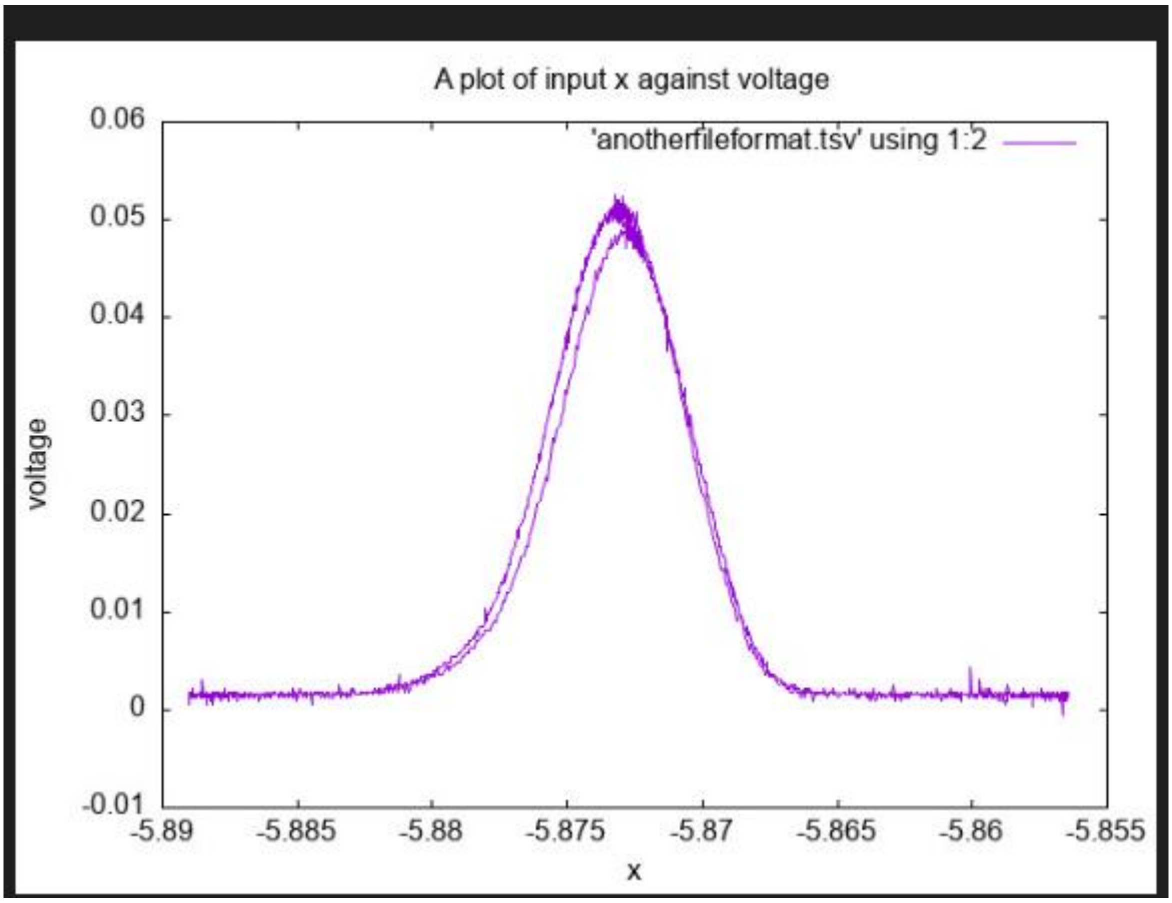


Figure 3 gaussian plot

the graph we can already guess that the x value for which the voltage is maximum should be between -5.875 and -5.870. This is our main aim of trying to visualize the data.

7 Determining the peak position

Here I'll implement the functions that will determine the peak position.

We will do this using two methods and at the end we see what the results will look like.

7.1 Direct maximum determination:

7.1.1 procedure

Here I created a function called getMaxValue (const std::vector<std::vector<std::string>>& dataset)

that will take the dataset as input. This function has a condition that checks whether the dataset is empty and returns a message if it's the case otherwise the function

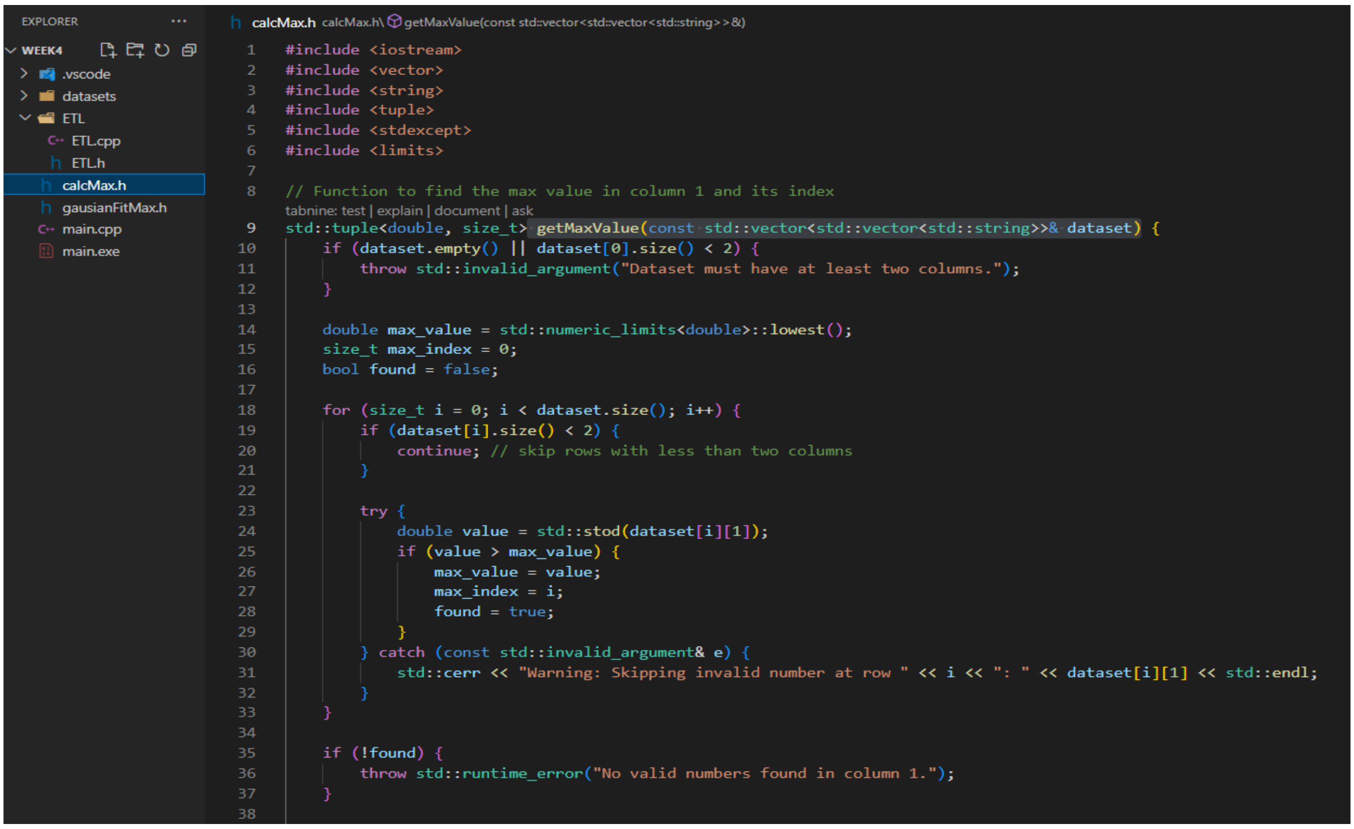
tries {to determine the maximum value by declaring two variables max_value and max_index and give them initial values of the lowest value from the numeric library to max_value and 0 to the index of the max value.

The function then iterates through the second column of the dataset where our voltage is found and compares the value of the max_value with each value of the dataset and then updating the max_value to the highest value while incrementing the index. This iteration only stops when we have reached the bottom of the list and our function then returns a tuple containing the max_value and the index.}

And if there is an

except it returns the exception.

Screenshot:



EXPLORER
WEEK4
.vscode
datasets
ETL
ETL.cpp
ETL.h
calcMax.h
gausianFitMax.h
main.cpp
main.exe
calcMax.h calcMax.h\ getMaxValue(const std::vector<std::vector<std::string>>&)
tabnine: test | explain | document | ask

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <tuple>
#include <stdexcept>
#include <limits>

// Function to find the max value in column 1 and its index
std::tuple<double, size_t> getMaxValue(const std::vector<std::vector<std::string>>& dataset) {
    if (dataset.empty() || dataset[0].size() < 2) {
        throw std::invalid_argument("Dataset must have at least two columns.");
    }

    double max_value = std::numeric_limits<double>::lowest();
    size_t max_index = 0;
    bool found = false;

    for (size_t i = 0; i < dataset.size(); i++) {
        if (dataset[i].size() < 2) {
            continue; // skip rows with less than two columns
        }

        try {
            double value = std::stod(dataset[i][1]);
            if (value > max_value) {
                max_value = value;
                max_index = i;
                found = true;
            }
        } catch (const std::invalid_argument& e) {
            std::cerr << "Warning: Skipping invalid number at row " << i << ": " << dataset[i][1] << std::endl;
        }
    }

    if (!found) {
        throw std::runtime_error("No valid numbers found in column 1.");
    }

```

Figure 4 direct maximum implementation

7.2 Gaussian Fit method:

7.2.1 Procedure:

I followed the algorithm below to implement the Gaussian Fit method.

I started by defining the function **gaussian_pdf** that takes 4 parameters which are the x_value, the amplitude, the mean and the standard deviation and calculate the peak position based on the equation below.

7.2.1.1 Gaussian Equation

f(x) = amplitude * exp (-(x - mean) ^2 / (2 * std^2))

I then created another function called

calculateMeanAndSigma (const std::vector<double>& values)

That takes a double vector as input and then uses its values to calculate the mean and the standard deviation and returns a tuple consisting the two values.

Another function is the

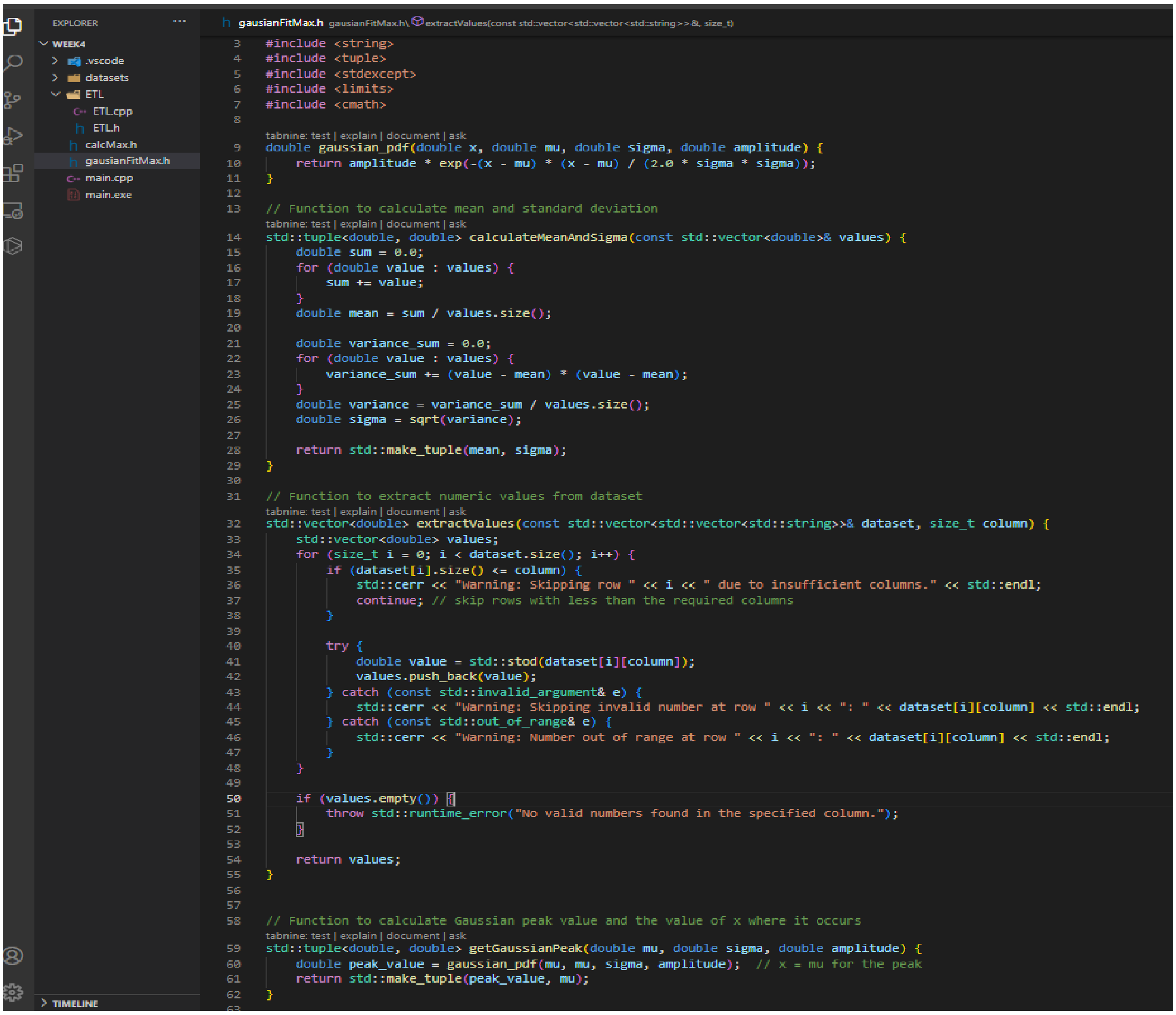
extractValues (const std::vector<std::vector<std::string>>& dataset, size_t column) function that takes the data set and column we are working with as in put and extracts numeric values from the dataset for which we can use to parse to the function that calculates the mean and the standard deviation.

Finally, I created the function

getGaussianPeak(double mu, double sigma, double amplitude) that collects the mean, standard deviation and the amplitude as parameters. **This function is responsible for the fitting of the data. It calls the gaussian_pdf we created above and fits the data in it then returns the maximum value and index as a tuple.**

This implementation can be seen on the screenshot below

Screenshot:



```cpp
#include <string>
#include <tuple>
#include <stdexcept>
#include <limits>
#include <cmath>

double gaussian_pdf(double x, double mu, double sigma, double amplitude) {
    return amplitude * exp(-(x - mu) * (x - mu) / (2.0 * sigma * sigma));
}

// Function to calculate mean and standard deviation
std::tuple<double, double> calculateMeanAndSigma(const std::vector<double>& values) {
    double sum = 0.0;
    for (double value : values) {
        sum += value;
    }
    double mean = sum / values.size();

    double variance_sum = 0.0;
    for (double value : values) {
        variance_sum += (value - mean) * (value - mean);
    }
    double variance = variance_sum / values.size();
    double sigma = sqrt(variance);

    return std::make_tuple(mean, sigma);
}

// Function to extract numeric values from dataset
std::vector<double> extractValues(const std::vector<std::vector<std::string>>& dataset, size_t column) {
    std::vector<double> values;
    for (size_t i = 0; i < dataset.size(); i++) {
        if (dataset[i].size() <= column) {
            std::cerr << "Warning: Skipping row " << i << " due to insufficient columns." << std::endl;
            continue; // skip rows with less than the required columns
        }

        try {
            double value = std::stod(dataset[i][column]);
            values.push_back(value);
        } catch (const std::invalid_argument& e) {
            std::cerr << "Warning: Skipping invalid number at row " << i << ": " << dataset[i][column] << std::endl;
        } catch (const std::out_of_range& e) {
            std::cerr << "Warning: Number out of range at row " << i << ": " << dataset[i][column] << std::endl;
        }
    }

    if (values.empty()) {
        throw std::runtime_error("No valid numbers found in the specified column.");
    }

    return values;
}


// Function to calculate Gaussian peak value and the value of x where it occurs
std::tuple<double, double> getGaussianPeak(double mu, double sigma, double amplitude) {
    double peak_value = gaussian_pdf(mu, mu, sigma, amplitude);  // x = mu for the peak
    return std::make_tuple(peak_value, mu);
}
```

Figure 5 gaussianfit method

8 Code execution

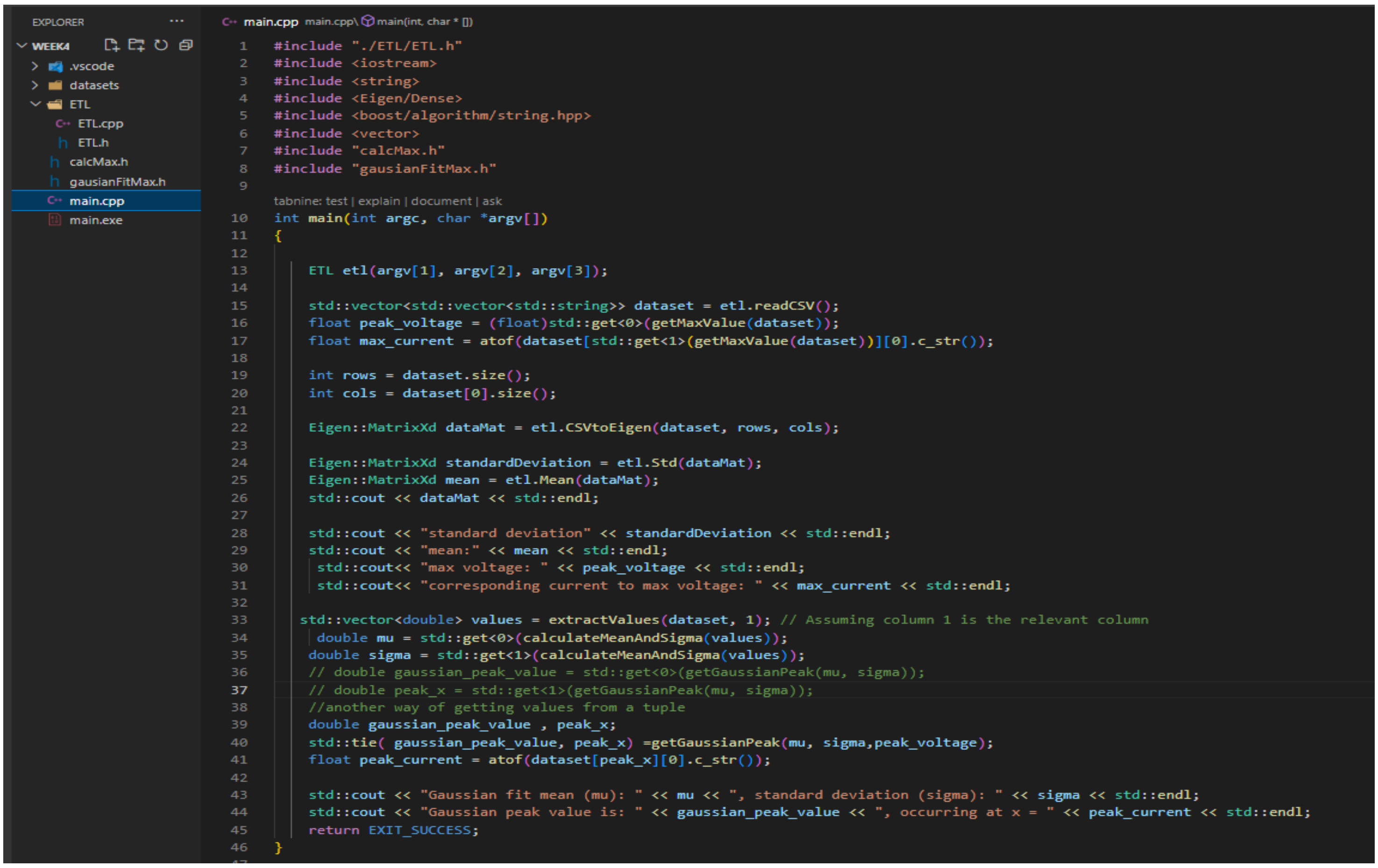
After implementing the libraries, functions and classes above I created the main.cpp file that will call and execute the implementation of these functions. Our main function will take 3 input parameters at run time from the terminal ant pass it to **elt** which is an implementation of the **ETL class**. To run our code, we open the terminal and type ***g++ -std=c++11 ETL/ETL.cpp main.cpp -o main*.** This command seeks to use g++ compiler to execute using c++11 the ETL.cpp and main.cpp files and save it's output to an executable file called main.

After running this command, we then run the executable main file and pass our dataset, delimiter and whether the dataset has a header or not as seen below

```
./main datasets/anotherfileformat.tsv " " header=False
```

The contents of the main function and its libraries are;

Screenshot:



```cpp
#include "./ETL/ETL.h"
#include <iostream>
#include <string>
#include <Eigen/Dense>
#include <boost/algorithm/string.hpp>
#include <vector>
#include "calcMax.h"
#include "gausianFitMax.h"

int main(int argc, char *argv[])
{

    ETL etl(argv[1], argv[2], argv[3]);

    std::vector<std::vector<std::string>> dataset = etl.readCSV();
    float peak_voltage = (float)std::get<0>(getMaxValue(dataset));
    float max_current = atof(dataset[std::get<1>(getMaxValue(dataset))][0].c_str());

    int rows = dataset.size();
    int cols = dataset[0].size();

    Eigen::MatrixXd dataMat = etl.CSVtoEigen(dataset, rows, cols);

    Eigen::MatrixXd standardDeviation = etl.Std(dataMat);
    Eigen::MatrixXd mean = etl.Mean(dataMat);
    std::cout << dataMat << std::endl;

    std::cout << "standard deviation" << standardDeviation << std::endl;
    std::cout << "mean:" << mean << std::endl;
     std::cout<< "max voltage: " << peak_voltage << std::endl;
     std::cout<< "corresponding current to max voltage: " << max_current << std::endl;

   std::vector<double> values = extractValues(dataset, 1); // Assuming column 1 is the relevant column
     double mu = std::get<0>(calculateMeanAndSigma(values));
    double sigma = std::get<1>(calculateMeanAndSigma(values));
    // double gaussian_peak_value = std::get<0>(getGaussianPeak(mu, sigma));
    // double peak_x = std::get<1>(getGaussianPeak(mu, sigma));
    //another way of getting values from a tuple
    double gaussian_peak_value , peak_x;
    std::tie( gaussian_peak_value, peak_x) =getGaussianPeak(mu, sigma,peak_voltage);
    float peak_current = atof(dataset[peak_x][0].c_str());

    std::cout << "Gaussian fit mean (mu): " << mu << ", standard deviation (sigma): " << sigma << std::endl;
    std::cout << "Gaussian peak value is: " << gaussian_peak_value << ", occurring at x = " << peak_current << std::endl;
    return EXIT_SUCCESS;
}
```

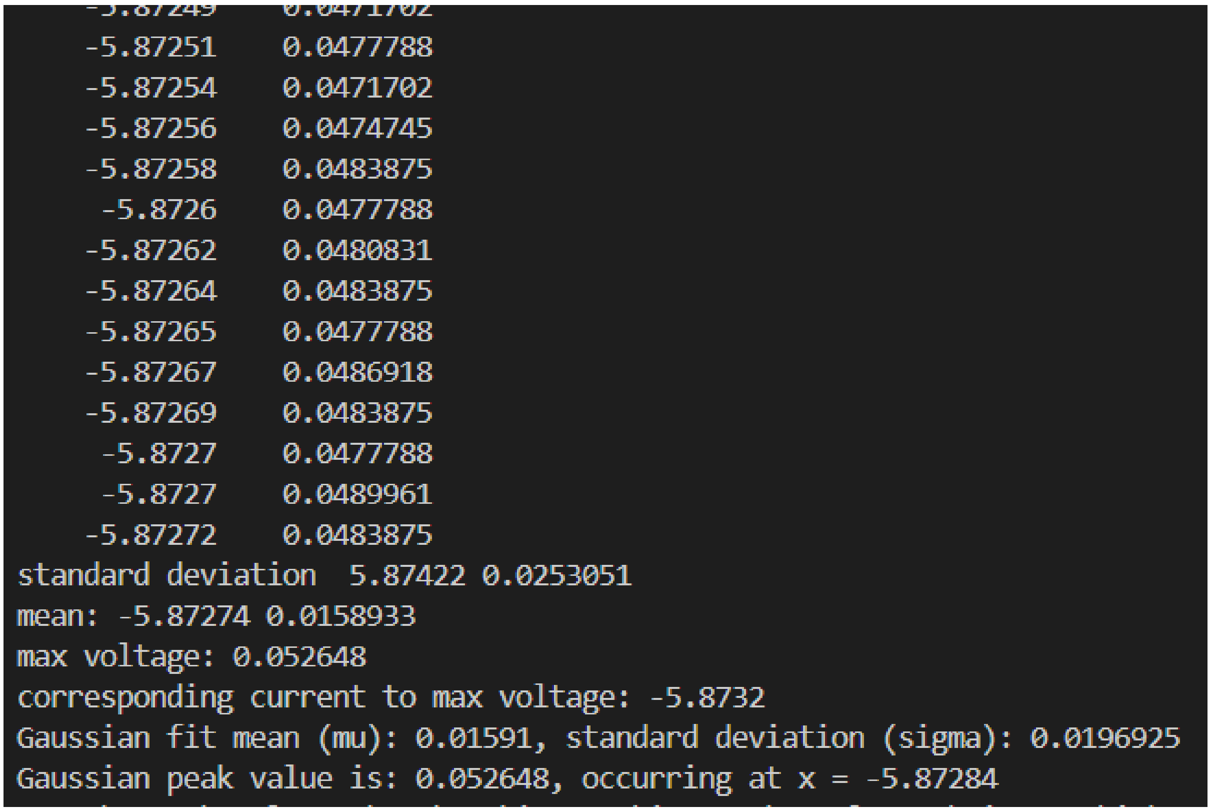
Figure 6 main functiion

9 Output

W e seek to display the dataset and the calculations we made on it .

After execution we get the following output

Screenshot:



```
    -5.87249    0.0471702
    -5.87251    0.0477788
    -5.87254    0.0471702
    -5.87256    0.0474745
    -5.87258    0.0483875
     -5.8726    0.0477788
    -5.87262    0.0480831
    -5.87264    0.0483875
    -5.87265    0.0477788
    -5.87267    0.0486918
    -5.87269    0.0483875
     -5.8727    0.0477788
     -5.8727    0.0489961
    -5.87272    0.0483875
standard deviation  5.87422 0.0253051
mean: -5.87274 0.0158933
max voltage: 0.052648
corresponding current to max voltage: -5.8732
Gaussian fit mean (mu): 0.01591, standard deviation (sigma): 0.0196925
Gaussian peak value is: 0.052648, occurring at x = -5.87284
```

Figure 7 output

10 Still to do

- Analise the relationship between the gaussian fit method and the direct method
- Test the function with different datasets
- Test the functions with datasets in excel and word format
- Fine tune the code

11 Conclusion

We have worked through the library importation, reading data in any format, describing and visualizing our data and finally determining the maximum position or peak value using a direct method and using the gaussian fit approach.

12 Difficulties

- I was not able to implement all the possible data formats that exist
- Building the module that calculate the max position using the gaussian is still not in its best form as I still need to finetune the code to make it more efficient

13 References

- Google
- Artificial intelligence
- Friends
- YouTube