Enhanced Knowledge in Sciences and Technology Vol. 3 No. 2 (2023) 302-312

EKST

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/ekst
e-ISSN: 2773-6385

A Customized Python Code for Laser Induced Breakdown Spectroscopy Data Analysis

Edwin Liew Kai Yuan[1], Syed Zuhaib[1*]

[1]Department of Physics and Chemistry,
Faculty of Applied Sciences and Technology,
Universiti Tun Hussein Onn Malaysia (Pagoh Campus),
84600 Pagoh, Muar, Johor, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/ekst.2023.03.02.036
Received 17 January 2023; Accepted 14 February 2023; Available online 30 November 2023

Abstract: Laser-induced breakdown spectroscopy (LIBS) is a powerful atomic emission spectroscopic technique for studying the elemental composition of a material. The major spectral analyses include the estimation of self-absorption and calculation of plasma parameters, i.e., electron density and plasma temperature. Getting subscription or purchasing license spectral analysis software is awfully expensive for students and individuals. It motivated this project to write codes for common and spectroscopic analysis of significant importance in Python and make it available for free on GitHub. All coding routines are implemented in Python programming language using Jupyter Notebook as IDE for Python. Coding sections for plotting of LIBS spectra, estimation of self-absorption, calculation of electron density by Stark Broadening method and plasma temperature by Boltzmann Plot method are developed. The output results in Jupyter Notebook are compared with the results obtained from OriginLab software for same calculations. The coding has been uploaded on GitHub and hence available for free to be accessed and used by other researchers. The instruction and the data that require user to change for meet their need have mention in each part of coding and in an instruction file in GitHub.

Keywords: LIBS, Python, Self-Absorption, Plasma Temperature, Electron Density

1. Introduction

Laser-induced breakdown spectroscopy (LIBS) is a relatively recent, effective, and cost-efficient analytical detection technique for measuring elemental composition based on atomic emission spectroscopy [1]. Only when the focused laser reaches a certain threshold for optical breakdown, which varies depending on the environment and the target material, does plasma begin to form [2]. Statistical analysis techniques and the application of chemometrics to broadband LIBS spectra have benefited LIBS applications in recent years [3], allowing all information from the complete LIBS spectrum to be taken into account rather than just that for a narrow spectral section.

**Corresponding author: syedzuhaib@uthm.edu.my*

LIBS has two types of analysis, there are qualitative analysis and quantitative analysis. Plasma investigation of quantitative analysis is the main part in this research. Meanwhile, LIBS existing constraints prevent it from being used consistently for quantification. As a result, there is a sizable body of research describing the categorization of diverse materials using LIBS. In this project, Python is applied to analyse data of the LIBS from previous experiment. The Python IDEs selected is Jupyter Notebook, it's the most up-to-date web-based interactive development platform for notebooks, code, and data. Its flexible interface allows users to create and arrange workflows in scientific computing, machine learning, computational journalism, and data science. A modular architecture encourages extensions to extend and enrich functionality. Jupyter Notebook is great for showcasing the work, able to view the coding and result, and can run the coding cell by cell to better understand what the codes did.

2. Materials and Methods

The materials and methods section included the procedures of collection of LIBS spectral data from metallic samples, followed implementing a Python code for filtering the functional spectra lines and calculating the plasma parameters such as electron density, and plasma temperature.

2.1 Import data in Python

From previous experiment, the wavelength and intensity data are exported in a .txt format file. A .txt file is a standard text document that contains plain text. It can be opened and edited in any text-editing or word-processing program. The .txt files are compiled and converted into a CSV file which is in the form of comma-separated values. The CSV files are imported into Python by Pandas library, read and converted to CSV by a simple ".to_CSV" command with the location that wants to save in the computer.

2.2 Self-absorption calculation

From the data, choosing an emission line for plasma parameter computation or LIBS system calibration, make sure to check for self-absorption. On the x-axis, the scale is set from -2000 to 2000. The variables of wavelength and transition probability were declared in the name of variables "wave" and "$A_k g_k$", respectively. The first and second variables were indicated by the subscripts 1 and 2. Depending on the computation, it serves as input for any user to alter to the initialized value. The right side of Eq. 1 is used to calculate the theoretical intensity ratio of a multiplet [4].

$$\frac{\boldsymbol{I_1}}{\boldsymbol{I_2}} = \frac{\boldsymbol{A_1 g_1 \lambda_2}}{\boldsymbol{A_2 g_2 \lambda_1}} \qquad \textit{Eq.}\ 1$$

The calculated value is stored in an array called "theoretical"; it is plotted as a straight line on the graph. The different intensity has been declared in an array; the values act as input and can be changed by any user; the subscripts 1 and 2 refer to first and second spectral line. The left side of Eq. 1 is used to calculate the intensity ratio. The calculated values of intensity ratio from the experimental data presented as scatter graph with the theoretical value as a straight line. "fig.update_layout" is used to set the range of the x and y-axis, for the graph show specifically range of the difference between the intensity and theoretical value.

2.3 Electron density calculation

From the previous step of importing data, a line graph to be ploted in the output window; the library used to plot the graph is Plotly Python Graphic Library. Select lines from the graph, determine which of the lines is suitable to proceed for the calculation of electron density, and record the start and end wavelength of the x-axis for the range selected. The data of "wavelength" and "intensity" between the range have been read out by using the slicing method. The range selected has been plotted in a scatter

graph and displayed in the output window. Variable "selectx" is the range of the x-axis selected; variable "selecty" is the range of the y-axis selected.

The formula of non-linear curve fitting profiles has been declared in a function; the fitting profiles are Lorentz, Voigt and Gaussian. Library `scipy.optimize` is used in each of the formulas to determine the optimal value to achieve the perfect R-squared value with initialized values. SciPy optimize library provides functions for maximizing objective functions, possibly subject to constraints. It includes solvers for non-linear problems curve fitting. The first part of non-linear fitting optimizes the value and calculates the R-squared value; the second part uses the optimized value from the first part to plot a line graph; and the third part calculates 1000 values to smooth the curve; the variation between each value is the last value minus the first value of "wavelength" and divided by 1000. The 1000 values from the previous phase are used to compute the full width half maximum (FWHM) in the third portion.

The formula for the Lorentz non-linear fitting profile is shown in Eq. 2; y_0 denotes intensity, and x denotes wavelength; both terms are taken from the spectrum.

$$y = y_0 + \left(2 * \frac{A}{\pi}\right) * \left(\frac{w}{4*(x-x_c)^2+w^2}\right) \quad Eq.\ 2$$

Eq.3 is the formula of Voigt non-linear fitting profile, y_0 is intensity and x is wavelength, both are from the spectrum.

$$y = y_0 + A * \left[mu \times \left(\frac{2}{\pi}\right) * \left(\frac{w}{4*(x-x_c)^2+w^2}\right) + (1-mu) \times \left(\sqrt{4*\frac{\ln(2)}{\sqrt{\pi}\times w}}\right) * e^{\left(-\left(4\times\frac{\ln(2)}{w^2}\times(x-x_c)^2\right)\right)}\right] \quad Eq.3$$

Eq. 4 is the formula of Gaussian non-linear fitting profile, y_0 is intensity and x are wavelength, both are from the spectrum.

$$y = y_0 + A * e^{\left((-0.5*\left(\frac{x-xc}{w}\right)^2\right)} \quad Eq.\ 4$$

All the formulas are declared in a different class before optimizing the value by using library of `scipy.optimize`. R-squared value shows how well the data fit the regression model and R-squared value determine by regression analysis, Eq. 6 is used to calculate the R-squared value.

$$R - squared = \left(\frac{\sum(fitted\ y\ value\ -\ mean\ of\ y)^2}{\sum(y\ -\ mean\ of\ y)^2}\right) \quad Eq.\ 5$$

Determine the maximum value of the y-axis of the fitting value and divide by 2; half-peak is determined. Find the nearest value of the y-axis to the half peak and the corresponding x-axis determined. The non-linear fitting curve has separated to left and right to determine the two nearest values to the half peak of the curve. FWHM is determined by the nearest from the right minus the nearest value from the left.

The electron density is represented by n_e, which measures the Stark broadening of ionic emission lines or non-self-absorption atomic. The density of electron, n_e, is obtained by the $H\alpha$, for the analysis the element's spectra, according to the relation, where $\Delta\lambda h$ is the FWHMs of the Hα line, $\alpha_{1/2}$ is the half-width of the reduced Stark profiles in Angstrom (Å) [5].

$$n_e = N_e(H_\alpha) = 8.02 \times 10^{12} \left(\frac{\Delta\lambda_h}{\alpha_{\frac{1}{2}}}\right) cm^{-3} \quad Eq.\ 6$$

2.4 Plasma temperature calculation

The most essential plasma parameter is plasma temperature, plasma temperature is determined from the relative intensity of the spectral lines. Temperature of plasma was calculated by using Boltzmann plot from spectroscopic parameters of 5 spectral lines of the element.

Where $\boldsymbol{K}$ is the Boltzmann constant, $\boldsymbol{T}$ denote the highest temperature monitored in this experiment in the unit of K, $\boldsymbol{I}$ represent the intensity in counts, $\boldsymbol{\lambda}$ represent the spectra line wavelength in nm, $\boldsymbol{Ag}$ denote the level of upper energy's degeneracy, and $\boldsymbol{E}$ represents Energy of the upper state. The following equation may be expressed by taking the natural logarithm of both sides:

$$\boldsymbol{ln\left(\frac{I\lambda}{Ag}\right) = -\frac{E}{KT} + ln\left(-\frac{hcN}{4nu}\right)} \qquad \textit{Eq.}\ 7$$

The selected wavelength, transition probability and the upper energy level are declared as variables "`selectLines`", "$A_k g_k$", "`boltz_x`", respectively. These spectroscopic data is obtained from the NIST database for the line emission. The calculation of y-axis has shown in.

$$\boldsymbol{y = ln\left(\frac{I\lambda}{A_k g_k}\right)} \qquad \textit{Eq.}\ 8$$

The wavelength, λ and I represented intensity, intensity selected is the local maxima of each of the selected wavelengths. The slope of the line is obtained by a general equation of a straight line:

$$\boldsymbol{y = mx + C} \qquad \textit{Eq.}9$$

From Eq. 7, the final element in the preceding equation is constant for a particular species, plotting a graph ln(Iλ/Ag) versus E for a straight slope line will result from the number of spectral emission lines $\frac{-1}{KT}$,and intercept $ln\left(\frac{hcN}{4nU}\right)$. As a result, measuring slope may be used to infer plasma temperature. The gradient of the line is represented in m, and used in the equation to determine plasma temperature. The formula of plasma temperature is given as:

$$\boldsymbol{K_B = -\frac{1}{slope}} \qquad \textit{Eq.}10$$

K_B is Boltzmann Constant, and m is the slope of the straight line.

3. Results and Discussion

The results are shared and screenshot from the Jupyter notebook output window as in Figure 1. The study was conducted to estimate of self-absorption (SA) and calculation of plasma parameter i.e., plasma temperature and electron density from the spectrum. The coding used LIBS spectral data obtained in previous experiments.

3.1 Import Data into Python

This is the data that import and read from local computer in data frame form by using Pandas Library in Pyhton. There were 3648 rows of data and 2 columns which are wavelength and intensity, the number from 0 to 3647 is the index number for rows.

```
      wavelength  intensity
0         195.28          9
1         195.55          9
2         195.82          9
3         196.09          1
4         196.36         -1
...          ...        ...
3643     1118.05          7
3644     1118.29          5
3645     1118.53         19
3646     1118.77         10
3647     1119.01          4

[3648 rows x 2 columns]
```

Figure 1: Data import from local file

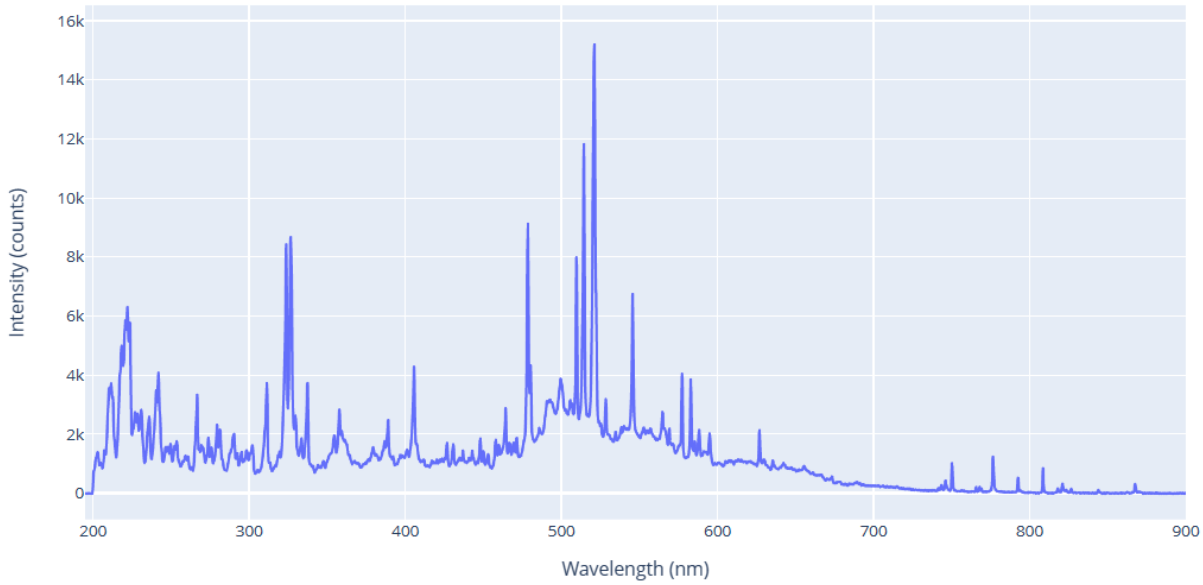


Figure 2: LIBS Spectrum

The output of the coding is the whole spectral data imported and plotted as a line graph having "wavelength" label on x-axis, "intensity" on y-axis (**Figure 2**). The range of x-axis was set to 200nm-900nm because there was no useful information outside this spectral range. However, it can be easily modified according to specific requirement of the data.

3.2 Estimation self-absorption

On the self-absorption graph, a straight-line labelled as "Theoretical", it is a theoretical line representing the ratio of spectral lines which are free of self-absorption. On y-axis, the element and wavelength selected have label and the x-axis label as "pressure (mbar)". Figure 3, shows the graph of self-absorption by Python.

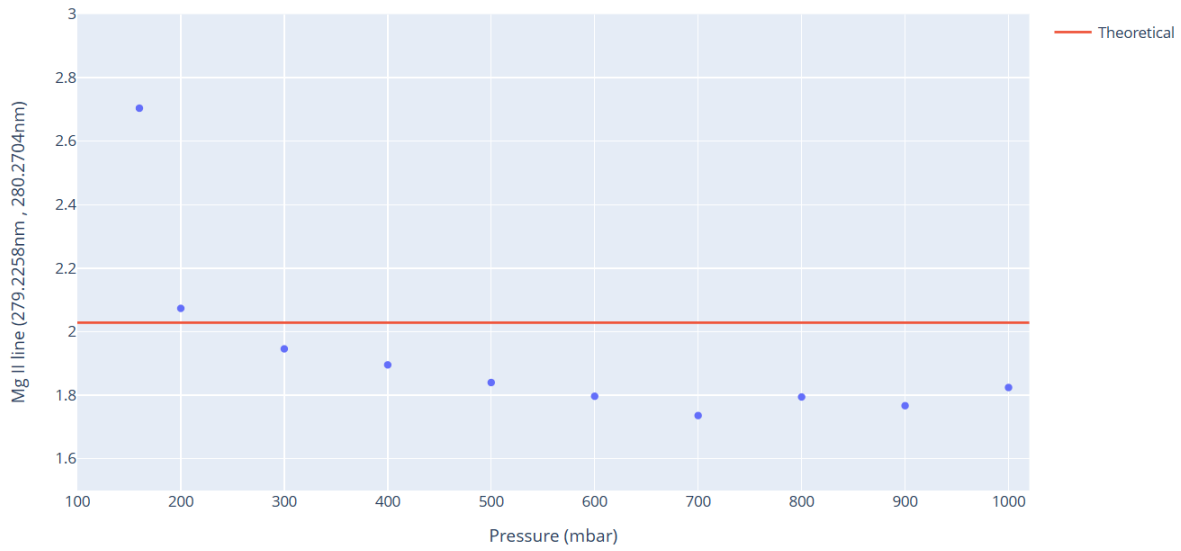


Figure 3: Self-absorption graph

3.3 Electron density calculation

To determine the electron density, selection of an appropriate isolated line is the 1st step, Plotly library provides the feature to magnify the spectrum (**Figure 2**) to the spectral region of interest i.e., an isolated spectral line. Note the 1st and last points on x-axis of the as selected range as shown in **Figure 4**.

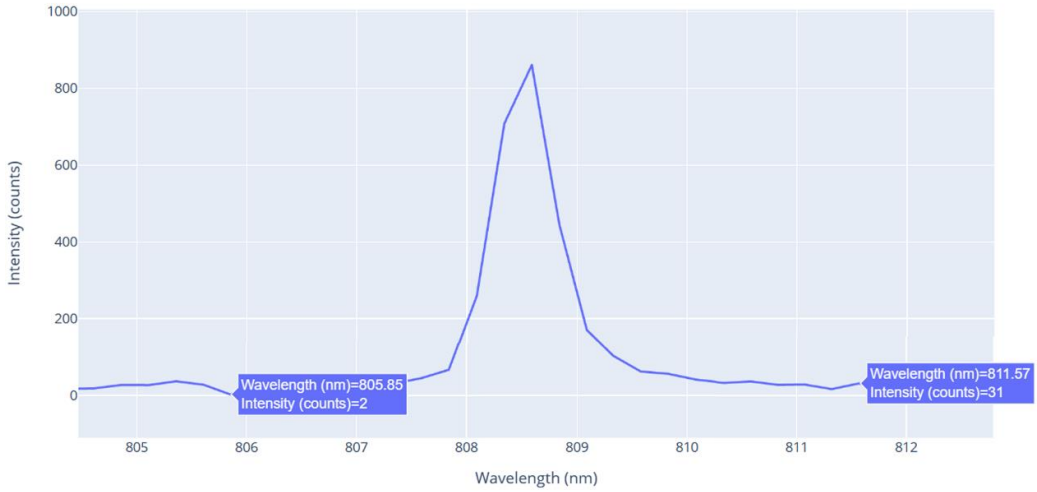


Figure 4: The 1st points and the last point on x-axis of the selected range

The result displays the index of 1st point and the last point is display in the output between symbol of ":", and the index of 1st point and the last point is 2362 : 2388.

The data between these 2 points had used to plot a scatter graph to show the pattern of the spectrum that selected.

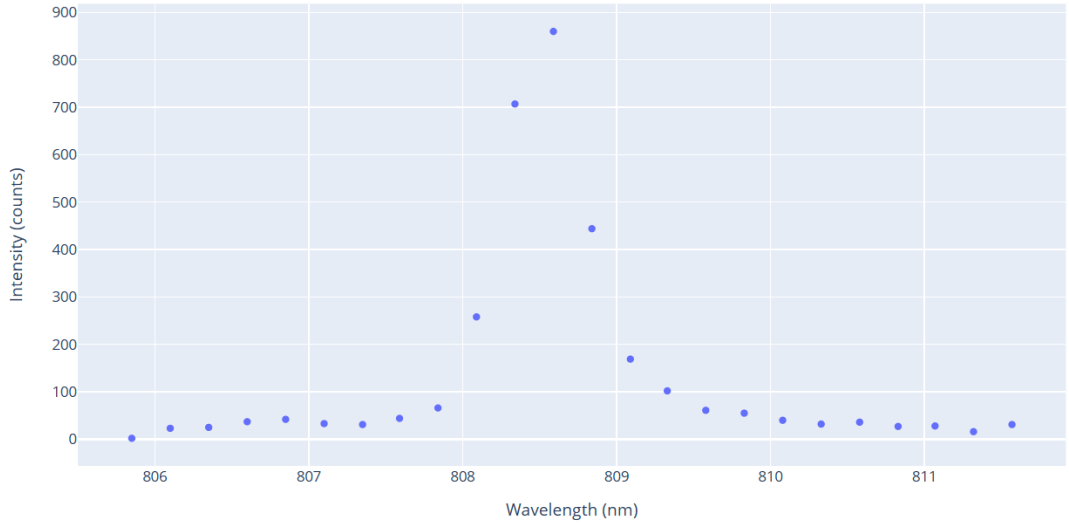
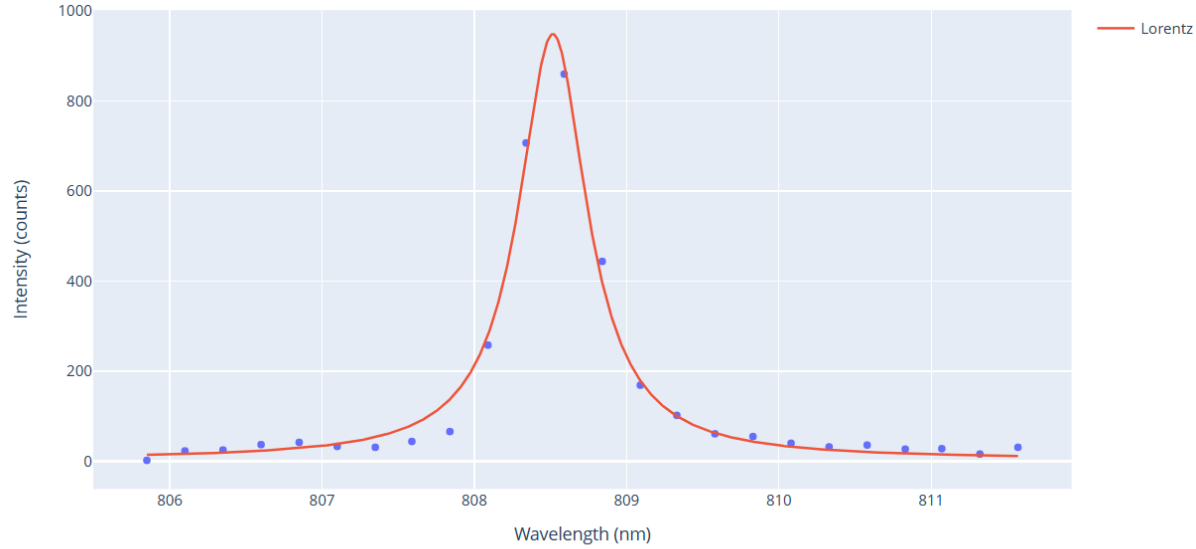


Figure 5: Selected range in scatter graph

The different fitting profile have applied and compare the R-squared value between these fitting profiles.



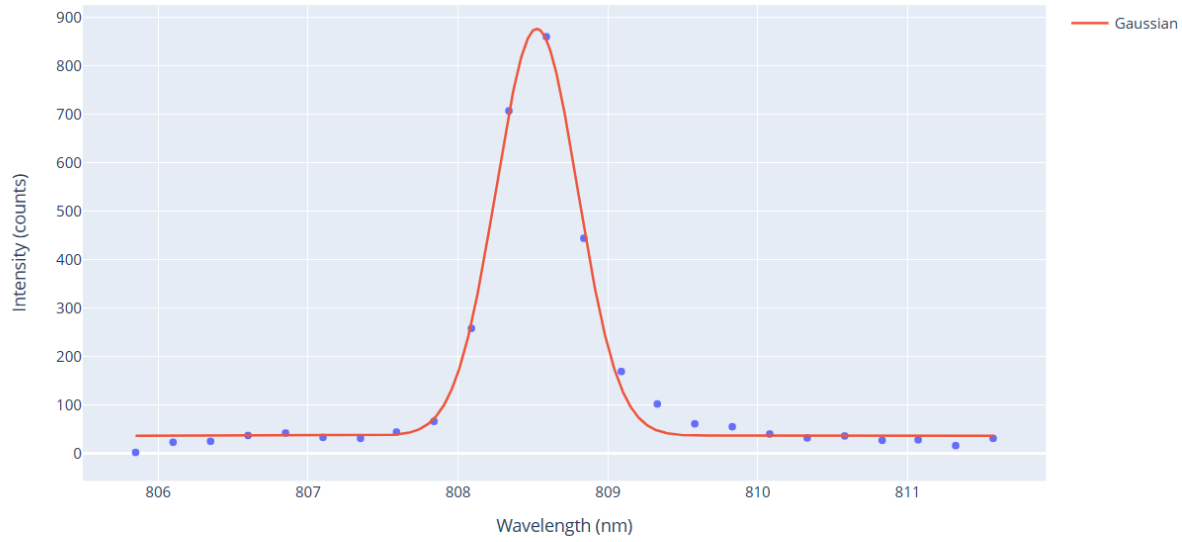

```
R-Squared : 0.98854
```

Figure 6: Lorentz non-linear fitting profile with r-squared value

Figure 6 is Lorentz non-linear fitting profile, the red fitting line is added on the scatter graph and the legend states the fitting function besides the graph. The r-squared value calculated from the coding is 0.98854 is same as the r-squared value calculated in OriginLab.

Table 1: R-squared value of Lorentz fitting in OriginLab

Model	Lorentz
R-Square	0.98854



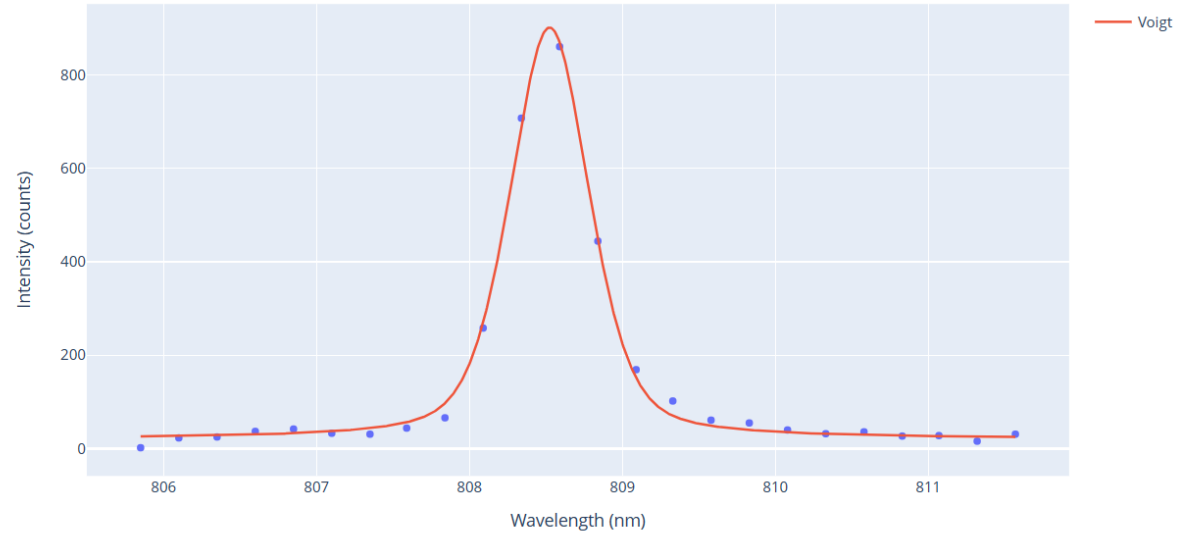
R-Squared : 0.99289

Figure 7: Gaussian non-linear fitting profile with r-squared value

Table 2: R-squared value of Gaussian fitting in OriginLab

Model	Gaussian
R-Square	0.99289

Figure 7 is Gaussian non-linear fitting profile, the red fitting line is added on the scatter graph, and the name of the fitting stated by the legend besides the graph. The r-squared value calculated from the coding is 0.99289 and is same as the r-squared value obtained from the Gaussian fitting of the same spectral line in OriginLab.



R-Squared : 0.99616

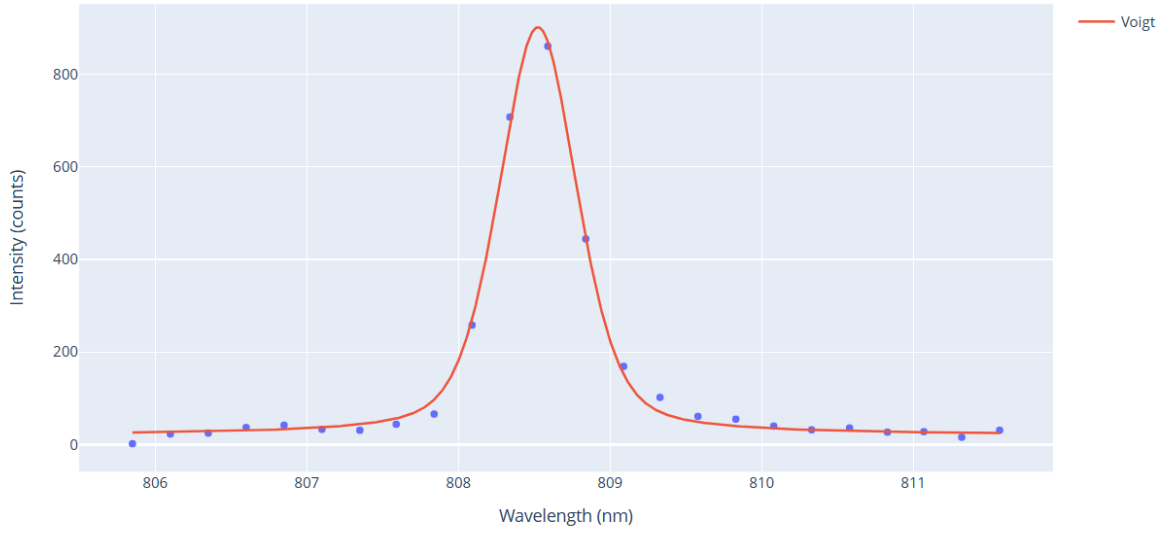
Figure 9: Voigt non-linear fitting profile with r-squared value

Figure is Voigt non-linear fitting profile, the red fitting line is added to the scatter plot, and the name of the fitting function/profile is stated in the legend besides the graph. The r-squared value calculated from the coding is 0.99616 is same as the r-squared value obtained of OriginLab.

Table 3: R-squared value of Voigt fitting in OriginLab

Model	Voigt
R-Square	0.99616

After choosing the best fitting profile i.e., the one with best R-squared value, the FWHM value will be used from that fitting function. The r-squared value and electron density had round off to 5 decimal places with the unit cm^{-3} is display below the graph.

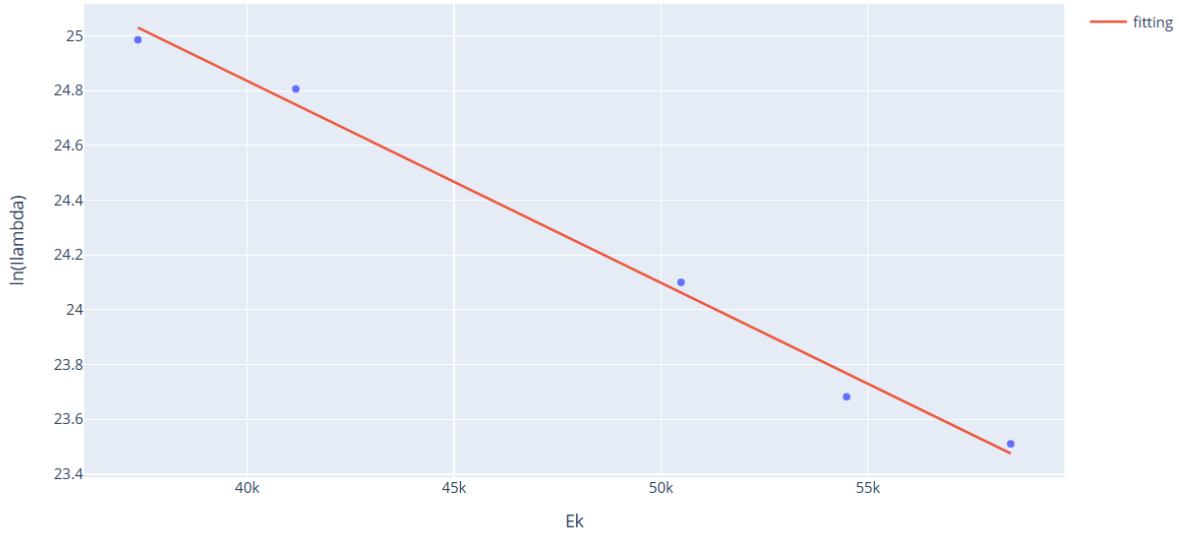



```
R-Squared : 0.99616
Electron density =  2.81972e+18 cm³
```

Figure 8: Best non-linear fitting graph with R-squared and electron density

3.4 Plasma Temperature

The plasma temperature is calculated by Boltzmann plot method. The Boltzmann plot is shown in **Figure 9**, the y-axis is ln(Iλ/gA) and x-axis is label as E_k, the linear fitting line is the red line and stated in the legend besides the graph. The calculated plasma temperature is using slope of the Boltzmann plot (**Figure 9**) is displayed below the graph and with the unit of "K", the value has rounded off to 5 decimal places. The value of slope calculated via Python is -7.3731×10^{-5}.




```
Plasma Temperature = 13562.90268K
```

Figure 9: Boltzmann plot graph and plasma temperature

Figure 9 shows the graph of Boltzmann plot with plasma temperature is calculated from Python coding.

The full coding has been uploaded to GitHub platform in 3 different parts, these are self-absorption, electron density and plasma temperature with the title name of LIBS spectral data analysis. This is the link to the code on GitHub is **edwinliew10/LIBS-spectroscopic-analysis (github.com)**. The instruction and the data that require user to change for meet their need have mention in each part of coding and in an instruction file in GitHub. Hence, user can download it and import their data into the coding with ease.

4. Conclusion

LIBS is a highly capable and powerful elemental analysis technique that is rapid and easy for quantitative analysis. The self-absorption and plasma parameters, such as electron density and plasma temperature, were calculated by Python programming.

The data from the .txt format was converted into a CSV file to organize large amounts of data better, and this data was imported into Python by Pandas library. The useful spectral lines were filtered and extracted to isolate data for further calculation of plasma parameters, i.e., electron density and plasma temperature.

Electron density and plasma temperature were calculated from data imported in Python. Electron density was determined by the Stark broadening method, applying the different non-linear curve fitting profiles to the selected spectral line and obtaining the FWHM from the fitting parameters to calculate the density. Plasma temperature was determined by the Boltzmann plot method, the slope from the linear fitting was obtained for the calculation of plasma temperature. Lastly, all the coding routine developed for the spectral analysis has been uploaded to GitHub. Hence, it is available for free to be accessed and used by other researchers.

Acknowledgement

This research was support by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (vot H845).

References

[1] R. Noll, C. Fricke-Begemann, S. Connemann, C. Meinhardt, and V. Sturm, "LIBS analyses for industrial applications – an overview of developments from 2014 to 2018," *J. Anal. At. Spectrom.*, vol. 33, no. 6, pp. 945–956, 2018, doi: 10.1039/C8JA00076J.

[2] *Laser Induced Breakdown Spectroscopy*. Cambridge: Cambridge University Press, 2006. doi: DOI: 10.1017/CBO9780511541261.

[3] V. Lazic, F. Colao, R. Fantoni, and V. Spizzichino, "Laser-induced breakdown spectroscopy in water: Improvement of the detection threshold by signal processing," *Spectrochim. Acta Part B At. Spectrosc.*, vol. 60, p. 1002, Aug. 2005, doi: 10.1016/j.sab.2005.06.007.

[4] H. Hegazy, H. A. Abd El-Ghany, S. H. Allam, and T. M. El-Sherbini, "Spectral evolution of nano-second laser interaction with Ti target in Air," *Appl. Phys. B*, vol. 110, no. 4, pp. 509–518, 2013, doi: 10.1007/s00340-012-5287-z.

[5] Y. Yang, X. Hao, and L. Ren, "Correction of self-absorption effect in calibration-free laser-induced breakdown spectroscopy(CF-LIBS) by considering plasma temperature and electron density," *Optik (Stuttg).*, vol. 208, p. 163702, 2020, doi: https://doi.org/10.1016/j.ijleo.2019.163702.