Draft version February 24, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

A Program to Find The Best-Fit Parameters for a Least-Squares Fit using the Levenberg-Marquardt Algorithm

Akshat S. Chaturvedi[1]

[1]*Department of Astronomy & Astrophysics, The Pennsylvania State University, University Park, PA 16802, USA*

ABSTRACT

I created a program that takes in user input initial guesses to find the best-fit parameters for a Lorentzian fit and a Gaussian fit to the input data, using the Levenberg-Marquardt Algorithm.

1. INTRODUCTION

The Levenberg-Marquardt algorithm is an algorithm which uses a mix of the Gauss-Newton and steepest descent algorithms to find the least square fit parameters for a given dataset in an efficient manner. The algorithm was developed by Kenneth Levenberg and then further worked upon by Donald Marquardt (Kelley (1999)).

For this assignment, I developed a `python` script which uses the Levenberg-Marquardt algorithm to determine the best fit parameters for a a dataset containing frequencies and line strengths of those frequencies. I assume from that limited information that the data corresponds to emission spectra, but that itself is not relevant to the scope of this assignment.

2. METHODS

The `python` script calls in the data from the dataset, which is shown in the appendix of this paper, and tries to find the best parameters to fit the data to two models: the *Lorentzian* fit and the *Gaussian* fit.

The *Lorentzian* fit is given by equation 1 below,

$$\phi_L(\nu) = \frac{1}{\pi}\frac{\alpha_L}{(\nu-\nu_0)^2+\alpha_L^2} \tag{1}$$

The *Gaussian* fit is given by equation 2 below,

$$\phi_G(\nu) = \frac{1}{\alpha_G}\sqrt{\frac{\ln 2}{\pi}}e^{\frac{-\ln 2(\nu-\nu_0)^2}{\alpha_G^2}} \tag{2}$$

The ϕ in both of these equations is the line strength as a function of frequency, ν. The terms α_L, α_G and ν_0 are free parameters.

The Levenberg-Marquardt function that I developed works to determine the free parameters of each respective model which produces the best possible fit. This "goodness" of fit is determined using the χ^2 measurement of each fit, which is shown in equation 3 below.

$$\chi^2 = \sum_{i=1}^{n}\frac{(y_i - f_i)^2}{\sigma_i^2} \tag{3}$$

The algorithm functions by iteratively changing the alpha matrix containing the free parameters, shown in equation 4 below (where α_f is the corresponding α term for each fit), by a value, `lambda`, and solving the equation $\bar{\alpha}\mathbf{X} = \bar{\beta}$, where $\bar{\beta}$ is the beta vector, as shown in equation 5 below.

$$\bar{\alpha} = \frac{1}{2}\cdot\begin{bmatrix}\frac{\partial^2\chi^2}{\partial\alpha_f^2}(1+\texttt{lambda}) & \frac{\partial^2\chi^2}{\partial\alpha_f\nu_0} \\ \frac{\partial^2\chi^2}{\partial\alpha_f\nu_0} & \frac{\partial^2\chi^2}{\partial\nu_0^2}(1+\texttt{lambda})\end{bmatrix} \tag{4}$$

$$\bar{\beta} = -\frac{1}{2}\cdot\begin{bmatrix}\frac{\partial\chi^2}{\partial\alpha_f} \\ \frac{\partial\chi^2}{\partial\nu_0}\end{bmatrix} \tag{5}$$

The solutions to the equation $\bar{\alpha}\mathbf{X} = \bar{\beta}$ are the amounts added to the free parameters of α_f and ν_0 respectively. Using these new parameters, the χ^2 is re-calculated, and then the algorithm tests whether it is lower than the previous χ^2 value.

If the new χ^2 value is higher than the previous χ^2, the algorithm increases the `lambda` value by a factor of 10.

If the new χ^2 is lower, then the algorithm reduces the `lambda` value by a factor of 10. This process repeats itself until the lowest possible χ^2 value is achieved.

The function then outputs the resultant free parameters, which are used to produce plots of the data with the *Lorentzian* and *Gaussian* fits respectively.

3. RESULTS

The script was run using initial guesses of **45** and **10** for α_L and ν_0 respectively for the *Lorentzian* fit, and initial guesses of **40** and **10** for α_G and ν_0 respectively for the *Gaussian* fit. These initial guesses, once run through the algorithm, produce final χ^2 values of **24641.660** and **102.499**.

Plots of the raw data, as well as the two fits are shown in figure 1 below.

As is clearly visible from the plots as well as the χ^2 values, we can see that the *Gaussian* fit is the better

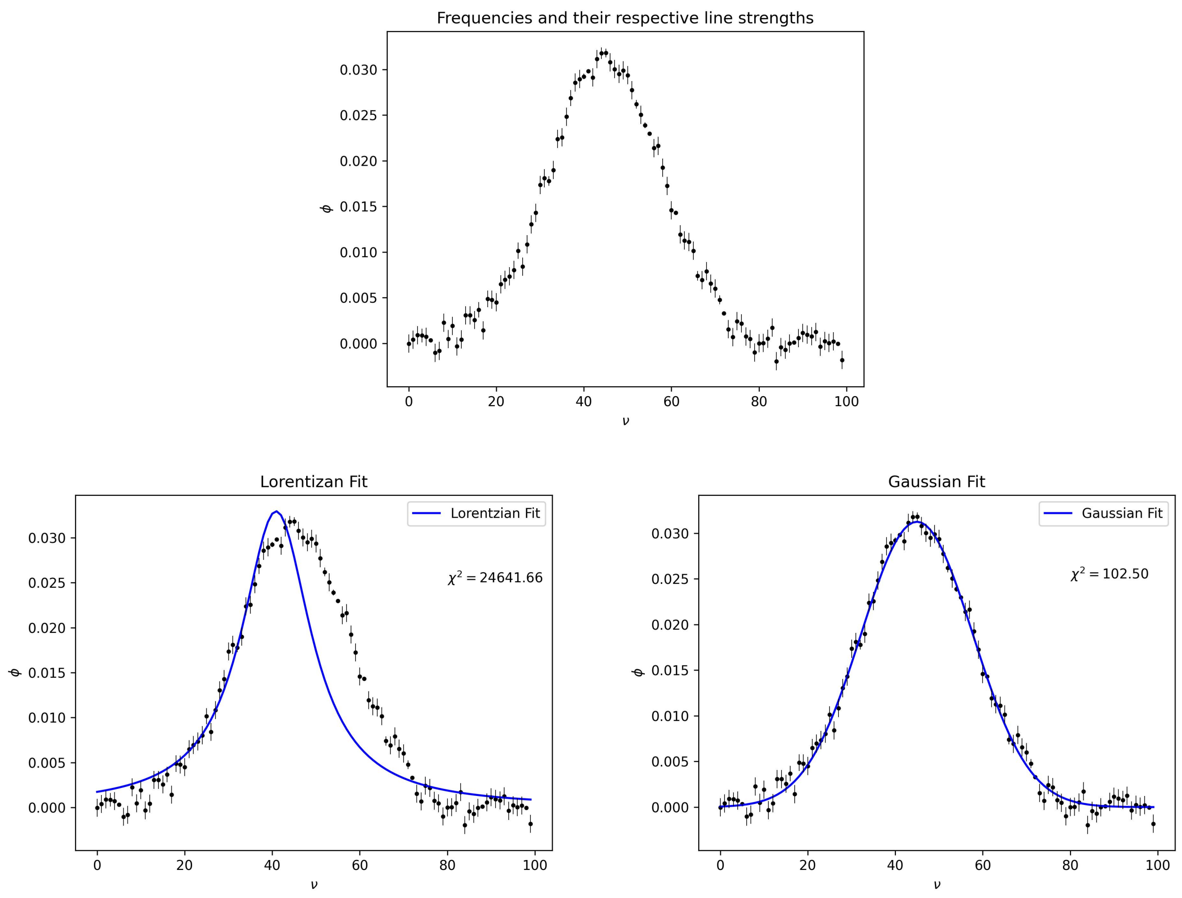

Figure 1: Raw data, the Lorentzian fit and the Gaussian fit

fit to the data. I believe that this is due to the fact that most data, given a sufficiently large sample size, tends to follow a normal, or *Gaussian* distribution, as stated by the Central Limit Theorem: *"random fluctuations that are a result of the sum of many independent random components, tend to be distributed normally, independent of the type of distribution sampled by each component."* (Berendsen (2011))

4. DRAWBACKS

There are a few shortcomings to the Levenberg-Marquardt Algorithm. One of them being that the algorithm requires that initial values are provided by the user. These values need to be within a certain degree of the actual value of those parameters otherwise the algorithm does not converge, or returns the wrong values.

Moreover, there is also another drawback with my `python` script itself. My code takes derivatives using the `sympy` library, which does the diffrentiation symbolically. However, to input the values of the dataset into those derivatives requires the data to be entered one at a time using various `for` loops. This makes the script slow to run, taking ≈ 30 seconds for both fits to be output.

REFERENCES

Berendsen, H. J. 2011, A student's guide to data and error analysis (Cambridge University Press)

Kelley, C. T. 1999, Iterative methods for optimization (SIAM)