Grid Search Method: Advanced Lab

John Francis

March 7, 2024

1 Introduction

Fitting data to a curve is not always simple. With linear functions, we can use the least squares method. With exponential functions, we can always linearize them by taking a logarithm. Some functions, can not be easily linearized and we must utilize different method of fitting the data to the curve.

A helpful indicator for how good our function matches our data is the chi-squared test. This test tells how how similar the data is to our function, or how well the function fits the data. The formula for calculating the chi-squared value is as follows:

$$\chi^2 = \sum_i^n \frac{(yi - y(xi))^2}{\sigma_i} \tag{1}$$

The chi-squared test gives us a number greater than zero; the bigger the number, the less similar the data is to the function. With all linearizing methods, we try to minimize the chi-squared value, ideally getting a value of 1. If we get less than one, our uncertainty in our measurements was too high. If we get greater than one, our data does not fit the function well.

To calculate our chi-squared value, we need a few things. First, we need data. Second, we need a function that we think should fit the data. Third, we need parameters for that function. In our curve fitting methods discussed, we will be assuming a function and data, and adjusting the parameters until the resulting chi-squared value is sufficiently small.

The data and function we are using is found in problem 7.2 from "Data Reduction and Error Analysis for the Physical Sciences" by Philip R. Bevington. The problem states:

7.2 "The tabulated data (see table 1) represents the lower bin limit x and the bin contents y of a histogram of data that fall into 2 peaks.

Use the method of least squares to find the amplitudes a_1 and a_2 and their uncertainties by fitting to the data the function

$$y(x) = a_1 L(x; \mu_1, \Gamma_1) + a_2 L(x; \mu_2, \Gamma_2)$$

i	1	2	3	4	5	6	7	8	9	10
x_i	50	60	70	80	90	100	110	120	130	140
y_i	5	7	11	13	21	43	30	16	15	10
i	11	12	13	14	15	16	17	18	19	20
x_i	150	160	170	180	190	200	210	220	230	240
y_i	13	42	90	75	29	13	8	4	6	3

Table 1: Data table from the exercise

with $\mu_1 = 102.1, \Gamma_1 = 30, \mu_2 = 177.9, \Gamma_2 = 20$. The function $L(x; \mu, \Gamma)$ is the Lorentzian function of equation (2.32). Assume statistical uncertainties of $(\sigma_i = \sqrt{y_i})$. Find χ^2 for the fit and the full error matrix."

The formula for the Lorentzian distribution (equation 2.32) is as follows:

$$L(x; \mu, \Gamma) = \frac{1}{\pi} \frac{\Gamma/2}{(x-\mu)^2 + (\Gamma/2)^2}$$

2 Method: Grid Search

The grid search method was used to calculate the optimal parameters. The idea behind the grid search method is outlined in Bevington's book as follows:

1. Select starting values a_j and step sizes Δa_j for each parameter, and calculate χ^2 with these starting params.
2. Increment one parameter a_j by Δa_j and make sure that χ^2 decreases with the new value
3. Repeat step 2 until χ^2 is minimized (aka it stops decreasing and starts increasing)
4. Use the minimum value and the point on either side (3 datapoints) to fit a parabola. Then find the minimum of that parabola. The minimum of this parabola is our accepted value of a_j.
5. Repeat for all the other parameters.
6. Continue until the total χ^2 value is minimized to what we perfer.

To find the optimal parameters, I developed a python program that uses the grid search method for finding the lowest chi-squared value. It uses the numpy library for arrays, the sympy library for matrices, and the matplotlib library for plotting.

To reduce computational time and resources, initial conditions for all the parameters were gathered from the same problem's least-squares solution. Then the grid search method starts with the initial guess minus 3, and steps in the positive direction with a step size of .01.

3 Results

The original graph that we generated using the least squares method had a decent, but not great fit. It is shown here in figure 1.

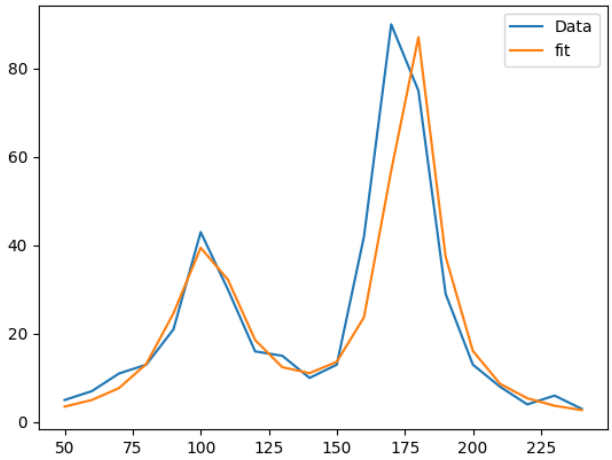

Figure 1: The resulting curve fit using the least squares method.

After using grid search method, a much better fit was generated, as shown in figure 2.

The program also prints out the chi-squared value generated by the fit. The output is shown in figure 3. The chi-squared value after doing the grid search value was 0.83. This suggests that our program did an excellent job of finding the best chi-squared value, but our data does not have accurate enough uncertainty. In other words, our curve is over-fitted, or it fits the data "too well."

Aside from the low chi-squared value, the program outputs the optimal values for each parameter. These parameters define our function that fits the data with the most precision. The parameters, compared with similar parameters calculated with the least squares fit, are displayed on table 2.

parameter	least-squares	grid search
a_1	1826	1826.217
a_2	2812	2812.496
μ_1	102.1	101.65
μ_2	177.9	174.915
Γ_1	30	31.63
Γ_2	20	18.22

Table 2: Caption

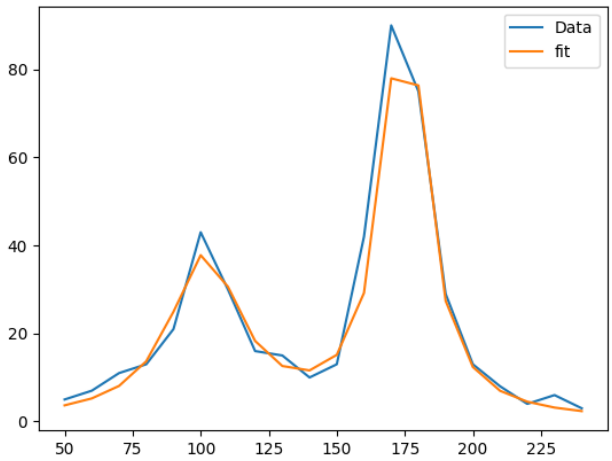

Figure 2: The resulting curve fit using the grid search method

```
The optimal value of a1 is 1826.21652301914 with a chi squared value of: 30.7852368714068
The optimal value of a2 is 2812.49599233190 with a chi squared value of: 30.7852296866077
The optimal value of mu1 is 101.650329801921 with a chi squared value of: 30.7015312231249
The optimal value of mu2 is 174.914960100043 with a chi squared value of: 11.6857126951218
The optimal value of gamma1 is 31.6303410231948 with a chi squared value of: 12.3591867204869
The optimal value of gamma2 is 18.2208858849177 with a chi squared value of: 11.6194972898590
Final chi squared: 0.829964092132788
```

Figure 3: Outputs of the grid-search run. Note: the chi squared values are not reduced, until the last line calculates the reduced chi-squared value.

4 Conclusion

The grid search is incredibly slow and inefficient. It is a "brute force" method of finding the curve fit's parameters. The reason this python program didn't take too long was because of some educated initial guesses for the parameters. If we had no idea about any of the parameters, it could have taken hours to compute.

There are many more efficient methods that are worth trying out. Some of these more efficient methods are gradient search, expansion, simplex, and genetic algorithm. Each employs a different method to find the lowest chi-squared value.

Another issue with the grid search method is that it stops when it hits a minimum value of chi-squared. This could mistakenly stop at a local minimum instead of an absolute minimum. To overcome this, multiple runs of the grid search method could be completed with different initial values of the parameters. Any method that checks values sequentially will run into this local minimum

error, so to avoid it a different approach can be taken like the genetic algorithm method.

In spite of it's drawbacks, the grid search method did a better job fitting the data than the least squares method. It calculated more accurate values for all of our parameters and produced an over-fit function for our data. This tells us that the uncertainty of $\sigma_i = \sqrt{y_i}$ is not sufficiently accurate.

5 Appendix A

The python program for calculating the parameters based on a grid search method can be found at this github url: https://github.com/john9francis/fitting-methods