# Unit IX
# Least Squares Fitting of Data

# Unit IX
# Least Squares Fitting of Data

In this unit, we will investigate the basic techniques of fitting functions to real data. This is akin to the interpolation techniques we learned earlier, with the following distinction. In interpolation, we make the assumption that each point is exactly correct, and we are trying to simply extend the function smoothly between points. We are often using the interpolation technique because it is too painful (either calculationally or experimentally) to obtain the intermediate points; we may not even know the functional form. However, for least squares fitting of data, we assume that we know what the functional form is. What we typically lack is perfect data. The data points have some random error included in them, and we wish to then estimate the most likely parameters for the function meant to describe the data.

The method of least squares fitting of linear and higher order polynomial functions is so straightforward that it is included in many computer graphing packages and on most scientific calculators. Not surprisingly, the ease of use often leads to misuse. The resulting functional fit is only as good as the data we are trying to fit, and is only as good as our understanding of the function underlying the data. It is completely misleading to find a linear fit to data that is in fact exponential. But this warning aside, the technique of least squares is extremely powerful.

The basic assumption of the least squares method is that we choose the parameters of the function (for a line, the slope and intercept) that minimize the average distance away from the line as measured by

$$\sum \left( y_i - y_{line} \right)^2 .$$

This is the parameter that maximizes the probability of correct fit, if we assume that the y values are distributed "normally," that is according to the normal error, or gaussian, distribution.

*Guidebook Entry IX.1: Simple Linear Fit*

Let's make the assumption that our data is fit by a line, where the parameters are defined implicitly in its form

$$y = mx + b.$$

The quantity we wish to minimize is the sum of the squares of the discrepancies from this line, that is

$$\chi^2 = \sum_i \left( y_i - y_{line} \right)^2$$

where the left-hand side is referred to as chi-squared. Substitute the expression for our line of $mx_i + b$ for $y_{line}$ in this expression above, and expand the expression, bringing everything you can out of the summations.

Now, to find the actual parameters m and b, you need to minimize chi-squared. This is simply a calculus problem; you need to take the derivatives of chi-squared with respect to m and b, and set them equal to zero. First, do that for the derivative with respect to m.

Now take the derivative of chi squared with respect to b, and solve for b.

Substitute your result for b into your first expression, and thereby solve the two simultaneously for m.

Finally, solve for b.

You should have results that can be rearranged to look like

$$m = \frac{1}{\Delta}\left(N\sum x_i y_i - \sum x_i \sum y_i\right)$$

$$b = \frac{1}{\Delta}\left(\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i\right), \text{ where}$$

$$\Delta = N\sum x_i^2 - \left(\sum x_i\right)^2$$

Make a spreadsheet that calculates the best fit line parameters for a set of data. Do this by calculating the following sums in separate cells:

$$\sum x_i$$

$$\sum y_i$$

$$\sum x_i^2$$

$$\sum x_i y_i$$

$N$ (total number of points)

and then calculating m and b (you may also wish to calculate $\Delta$ separately). Use the following set of data

| x | y |
|---|---|
| 0 | 19.2 |
| 1 | 16.0 |
| 2 | 14.4 |
| 3 | 12.0 |
| 4 | 7.5 |
| 5 | 5.9 |
| 6 | 2.1 |
| 7 | -1.4 |
| 8 | -4.0 |
| 9 | -6.4 |
| 10 | -11.5 |

Make sure that your fit looks reasonable by creating a third column of mx+b values for each x value, and then plot them all together.

Now let's do this the easy way, first in Excel, and then in Mathematica. Select your data in your Excel sheet, and create a new x-y (scatter) graph. Choose the format that has only data points, and no connecting lines.

Next, double click anywhere on the resulting chart. The edges of the chart will become bold. Now click *one* time on any of the data points in the chart. They data points will all become bold. Now, in the "Insert" menu, select the "Trendline" item. This will bring up a dialog box. If it isn't already selected, choose the linear type. Before clicking ok and dispatching the box, choose the options tab along the top of the box by clicking on it, and then select the "Display Equation on Chart" option. Click the "OK" and your chart should now include a best fit line with the equation displayed. Does the fit line agree with your result above?

Now let's do the same thing in Mathematica. We must first give Mathematica the table of data. The easiest way to do this is to give the data set a name, something descriptive, such as lindata. Use lower case so Mathematica easilty distinguishes it from its own functions. Entering the set of data goes like:

```
lindata = {{0,19.2},{1,16.0},{2,14.4},{3,12.0},{4,7.5},{5,5.9},
{6,2.1},{7,-1.4},{8,-4.0},{9,-6.4},{10,-11.5}}
```

You can make a graph of this data set using the ListPlot command. We will do more than that by giving a name to the output, such as

```
gdata = ListPlot[lindata]
```

You then can make a linear fit with the command

```
Fit[lindata,{1,x},x}
```

which tells Mathematica to make a best fit using a linear combination of the basis functions 1 and x. If we had written instead {1,x,x^2} it would have done a quadratic fit. In fact, you can include any functions that Mathematica knows, even ones you define for it. But for this example, stick with the linear case. Do you again get the same fit as the previous two times?

You can also plot the previous result, using simply

```
fitplot=Plot[%,{x,0,10}]
```

If you had intervening commands, add the appropriate line number following the %. You could also have named the fit, and then plotted it by name. Finally, you probably would like to have the data and the fit on the same graph. This is done using the Show command, as in:

```
Show[gdata,fitplot]
```

This best-fit line made the assumption that we treat all points equally. In other words, we assumed that the error associated with each point was the same as that of any other point. We also assumed that all the error was all in the y values, and not in the x values. If errors are not the same for all y values, we need a procedure that weights the better points more heavily. If there are significant errors in the x values, the fix is very messy, but rarely makes a significant difference in the resulting line, and is almost universally ignored; we will follow that time-honored path.

*Guidebook Entry IX.2: Weighted Linear Fit*

How might we have significantly smaller errors on one point than another? We might have made many measurements at one x value, then averaged the y values to give a single, more accurate, data point. This suggests an obvious way of weighting the point. Let's say the fifth point was really the average of four measurements. Show that measuring y values at a given x value four times (and including those four separate values as four distinct points) is equivalent to using the average $y_5$ value with a weighting factor of four in each of the sums.

Convince yourself that an overall weighting factor (i.e. a scaling constant multiplying each summation) has no effect on the values for m and b (don't forget to treat N like a sum as well).

It is easy to show that averaging n points with equal errors reduces the error in the average by the square root of n (you can try that propagation of error problem on your own, or consult an error analysis reference like *Data Reduction and Error Analysis for the Physical Sciences* by Philip Bevington). Given this, construct a weighting factor system that depends only on the size of the errors of each point.

While it is possible to construct methods of calculating the coefficients of polynomials or even more complicated functions using the least squares method, let us introduce the more general technique of minimizing chi-squared.

*Guidebook Entry IX.3: Minimizing Chi-Squared*

Let's consider the following data

| x | y | error |
|---|---|---|
| -3 | .29 | .05 |
| -2 | .56 | .13 |
| -1 | 1.18 | .24 |
| 0 | 1.72 | .35 |
| 1 | 1.75 | .40 |
| 2 | 1.63 | .35 |
| 3 | 1.17 | .24 |
| 4 | .52 | .13 |
| 5 | .25 | .05 |
| 6 | .10 | .02 |
| 7 | .02 | .004 |

We will guess that this might be fit by a gaussian function, something of the form

$$y(x) = Ae^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

You are literally going to guess some values for the amplitude A, width σ, and centroid μ, and vary them around to minimize chi-squared, but before doing so, look at the data, and make some reasonable guesses for these parameters. Check with your instructor if you are at a loss as to how to do this. Write your guesses below, and use those for starting values.

Following on our weighting factor method, what we want to minimize is what is more properly (than what we introduced earlier) called chi-squared:

$$\chi^2 = \sum \left(\frac{y_i - y(x_i)}{\delta_i}\right)^2$$

where $\delta_i$ is the error associated with the i'th point. Produce a spreadsheet with columns for x, y, $\delta_i$, a test gaussian function, and the contribution of

each point to chi-squared. Sum this last column to produce chi-squared. Make sure to keep the parameters of amplitude A, width $\sigma$, and centroid $\mu$ in their own cells so you can vary them.

Once you have this sheet, manipulate the three parameters "by hand" until you minimize chi-squared. What are your best parameter values?

You probably began wondering, how hard should you try to minimize chi-squared? If each point is off by an amount that is comparable to the error, how much does each point typically contribute to chi-squared?

How large should chi-squared be for a respectably good fit given that we have eleven points?

This final point is often rephrased into the so-called reduced chi-squared, where chi-squared is divided by the number of "degrees of freedom," which is the number of points minus the number of parameters being determined in the fit. Typically, a good fit will give a reduced chi-squared of about one. If reduced chi-squared is much larger than this (and unfortunately the definition of "much" depends also on the number of points), then you probably don't have the right fitting function, or your estimates of errors were too conservative. If on the other hand, if the reduced chi-squared is much less than one, you can be virtually assured that you have been too generous in your error estimates.