Gnuplot tutorial #2, Physics 209

Linear and Non-linear fitting

Our goal in this tutorial is to learn how to use gnuplot to find a least-squares fit to experimental data. One key requirement is that our fitting routine must do a weighted least-squares fit.

The general problem we would like to solve is: given a function, $f(x)$, that depends on some parameters a_i, and a set of data $\{x_j, y_j, \delta y_j\}$, for what values of the a_i does the function best fit our data? Expressed mathematically, we want to minimize the function χ^2 with respect to the a_i, where

$$\chi^2 = \sum_j \left(\frac{f(x_j) - y_j}{\delta y_j} \right)^2$$

in which the sum includes all of the data points. χ is the greek letter chi (pronounced "kai").

If $f(x)$ is a straight line, then you can analytically find the values of the slope and intercept from sums involving the x_j and y_j. For more complicated functions, there is usually not a simple analytic solution. A number of algorithms have been developed to find the best-fit of a function (ie. minimize χ^2). One well-known algorithm, called Marquardt-Levenberg, has been implemented in gnuplot. While you don't need to understand all of the details of this algorithm in order to use it, having a basic understanding of how it works will help you use it much better.

χ^2 is a function of the a_i's. If we imagine an example with two such parameters, then χ^2 can be thought of as a surface in a three-dimensional space. To use the Marquardt-Levenberg algorithm, we must supply initial guesses for the parameters. The algorithm then calculates the gradient of the surface at the point of our initial guess, and steps down the gradient till the minimum is found. This works very well, as long as the initial guesses aren't "too far" off of the best fit values. If the guesses aren't good enough, the algorithm can get trapped in local minima of the χ^2 function. The other thing to be aware of, is that each step of the algorithm involves the inversion of a matrix. If the parameters are dramatically different in size (eg. one is 1.4×10^{-4} and the other is 7.9×10^{12}, then the matrix inversion becomes very troublesome. If you are dealing with parameters that are more than about 3 orders of magnitude different, it is best to rescale them in the function so that they are more similar.

Let's try this with a simple linear function.

- Get logged into one of the computers in Henn 205.
- Grab a sample data set with the command:
  `wget http://www.phas.ubc.ca/~phys209/files/fit_data1.txt`
  (wget is a very convenient command-line tool to download a file from a web site.) Look inside the file to see what's there with: `more fit_data1.txt` (if the file is too big to all fit on screen, you can use the space bar to page through it).
- Now start up gnuplot, and plot the data with: `plot 'fit_data1.txt'`
- Now we will define our function:
  `f(x) = m * x +b`
  You can see the functions that are defined with `show functions`.

- Next, we need to set the parameters m and b to initial values. Looking at the data, we try:
  ```
  m=5
  b=5
  ```
  the command: `show variables` (as with most gnuplot commands, this can be shortened: `sh var` is good enough) will show you all of the variables that have been defined.

- Let's see how good a job we did with our guess:
  ```
  plot f(x),'fit_data1.txt'
  ```
  Not bad for a first guess.

- Our plot needs error bars. Inside the file we're told that the error in the y values is 1% of the reading plus 1% of the range (100 mA). So let's add the correct error bars:
  ```
  plot f(x),'fit_data1.txt' us 1:2:($2*.01+1) w err
  ```

- Now let's do the fit:
  ```
  fit f(x) 'fit_data1.txt' us 1:2:($2*.01+1) via b,m
  ```
  gnuplot will run the fitting algorithm and provide lots of information about how it is progressing. If everything went well, you should see that the fit has converged, and you should see the final sum of squares of residuals (that is χ^2). A good fit has $\chi^2 \sim N - M$, where N is the number of measurements and M is the number of parameters in the function. The quantity $N - M$ is known as the number of degrees of freedom. If χ^2 is much smaller than this, you probably overestimated your errors, and if its much bigger, that indicates that your model doesn't fit the data well. For more information, read the help under: `help fit` and `help fit error`. Some other references are given at the end of this tutorial.

  gnuplot provides you with the best fit values of the parameters as well as estimates of the standard error in those parameters. For a fit to a linear function, when the errors in the measurements have a normal distribution, the standard errors have the expected meaning. If you do a fit to a non-linear function, the standard errors quoted are only very rough approximations to the true errors in the parameters. See the references at the end for more detail. Finding meaningful error estimates from a non-linear fit involves much more work, and we will explore this issue in a future assignment.

  Finally gnuplot gives you a correlation matrix that tells you to what degree the different parameters are distinguishable from each other. If you see a number close to 1 or -1 in an off-diagonal element of the correlation matrix, that indicates that the fitting routine is not able to distinguish between those two parameters. A trivial example is if you defined a function: $y = m{*}x{+}a{+}b$ and tried to fit the same data set with this new function with three parameters m, a, and b. The correlation coefficient between a and b is -1, indicating that any change made to a can be undone by a compensating change in b. A nearly as trivial example would be something like: $f(x) = ae^b + mx$ where again a and b would have a correlation coefficient of -1.

- To see what the fit looks like, we plot it and the data (with errors) together:
  ```
  plot f(x), 'fit_data1.txt' us 1:2:($2*.01+1) w err
  ```

- Ok, now lets add some labels and make this pretty enough to print out:
  ```
  set title 'Current versus voltage for unknown resistance'
  set xlabel 'Voltage (V)'
  set ylabel 'Current (mA)'
  ```
  `set nokey` - removes the key from the top right corner

- Next: `save 'ohms_law.plt'` will save our work so we can reload it later.

- If you created a `printit` file last time, you can load it and create the printable postscript file, or you can do the following:

  - in your gnuplot window, type: `!xterm &` to get a new window
  - in the new window, type: `pico ohms_law.plt`
  - scroll down to the bottom of the file. The best fit parameter values, the function definition, and even the last fit command we used are stored in the file.
  - down below the plot command, but before the EOF line, add these lines:
    ```
    set term post eps enh 24
    set out 'ohms_law.eps'
    set size 1.4,1.4
    replot
    set out
    set term x11
    ```
    and then save the file.
  - Now back in gnuplot, type: `load 'ohms_law.plt'` This will plot the fit and the data again, but this time it will also generate the postscript file for you to print.
  - at the unix command prompt, you can type: `lpr ohms_law.eps`

References:

1) Numerical Recipes in C, W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Cambridge University Press, Cambridge, 1988.
2) Data Reduction and Eror Analysis for the Physical Sciences, P.R. Bevington, McGraw-Hill, New York, 1969.