Final Project – Computers for Physicists

Your final project will involve the practices you learned in the course, as well as some more self-learning of useful Python features. You will implement a fitting program, equivalent to Lab A's 'fitgui' for linear functions. Your code will be based on the technique used in fitgui as well.
Your program will receive a name of a file containing the data points and axis titles, compute the minimal χ^2 for a linear fit and plot the results using the Python package **PyPlot**.

1. Input file handling

The input file will be a table with data points x, y and their uncertainties dx, dy in two possible forms. One with the data within columns and the other with the data within rows. The number of points is unknown, the titles can be in capital letters or in lowercase letters and the order of the data is not fixed, meaning the data can be (x,DX,Y,dy) or (y,dx,DY,y) or any other configuration. Under the table will the names of the x and y axes.
Example for the input files were added to this instruction file.

Your program should check whether the input file is valid:

1. Check if all the data rows / columns have the same length. If not (for example, if x has 3 numbers and dx has 4), print "Input file error: Data lists are not the same length.".
2. The program should also check that all the uncertainties bigger than zero (dy_i, dx_i>0). If not, print: "Input file error: Not all uncertainties are positive."

If the file is not valid, stop the program from running. Otherwise, the program continues as described in the next sections.

2. Fitting function and definitions

Let's recall the definition of χ^2 :

$$(1) \quad \chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i;a)}{\sqrt{{dy_i}^2 + (f(x_i+dx_i;a) - f(x_i-dx_i;a))^2}} \right)^2$$

where x_i, y_i, dx_i, dy_i is a set of data points and their errors, N is the number of data point and a is a set of the fitted function's parameters.
Say we want to fit our data to the linear function $ax + b$.
We want to find a and b that bring our χ^2 function to a minimum. We can do this by differentiating equation (1) with respect to a and b:

$$\frac{\partial\chi^2}{\partial a} = 0, \ \frac{\partial\chi^2}{\partial b} = 0$$

Assuming $adx_i \ll dy_i$, we neglect dx_i s contribution to χ^2, and only minimize:

$$(2) \quad \chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (ax_i + b)}{dy_i} \right)^2$$

After some algebra, we are left with:

$$(3) \quad a = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2}, \qquad da^2 = \frac{\overline{dy^2}}{N(\overline{x^2} - \bar{x}^2)}$$

$$b = \bar{y} - a\bar{x}, \qquad db^2 = \frac{\overline{dy^2}\ \overline{x^2}}{N(\overline{x^2} - \bar{x}^2)}$$

where

$$\bar{z} = \frac{\sum \frac{z_i}{dy_i^2}}{\sum \frac{1}{dy_i^2}}$$

We now normalize our result for χ^2 by dividing it by the number of degrees of freedom:

$$(4) \quad \chi^2_{reduced} = \frac{\chi^2}{N - 2}$$

3. Output format

Your program should print to screen the following text:
Evaluated fitting parameters:
a = <a> +- <da>
b = <b> +- <db>
chi2 = <χ^2>
chi2_reduced = <χ^2_reduced>
Where <x> should be replaced by the computed value of x.

4. The PyPlot package

You will need to plot your results on a graph. For this you will use the matplotlib.PyPlot package.
A python package is a collection of modules, and Python has a lot of open-source packages that can be used for various applications - mathematical packages, data analysis packages, web design packages… Here we will focus on the most popular graph plotting package, PyPlot which is a part of the bigger matplotlib package.
If you want to use matplotlib, it is not enough to *import* it, you need to *install* it. You can do it in one of the following ways:

1. (recommended) through PyCharm,

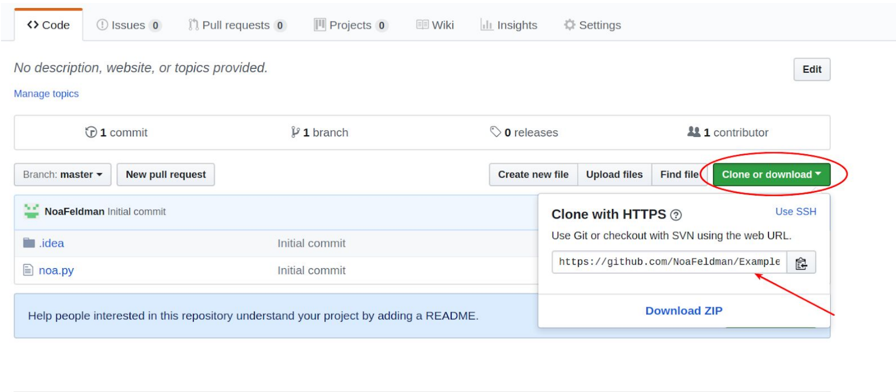
2. **open File >> Settings >> Project:<*your_project_name*>> project interpreter >> install** (the little '+' sign on the top right) **>>** search **'matplotlib' >>** mark 'matplotlib' on the left panel and press **'install package'**
   Note that for Mac computers, instead of
   open File >> Settings
   You need to choose
   PyCharm >> Preferences
3. Through linux command line / Windows cmd -
   type:
   `python -m pip install -U pip`
   and then:
   `python -m pip install -U matplotlib`
   (This works for Wing, too)

After you installed the package, in order to use it you need to add an import:
`from matplotlib import pyplot`
Now you can use the plotting functions.
Here are some important functions you need to know for the project:
pyplot.show
pyplot.plot
pyplot.errorbar
pyplot.savefig
It is also recommended to look and test some examples, like the ones on this link for plotting and this link for errorbars.

The plot requirements are the following:
1. The fitted function (line defined by the parameters *a* and *b*) should be plotted in red.
2. The data points should be plotted as an error bar. The line for the data points should not be seen, and the data points and their errors should be plotted in blue.
   Note: Step one should be executed before step 2, so that the error bars appear in front of the fitted plot.
3. Add axis titles based on the titles extracted from the input file.
4. Save the figure under the name "linear_fit" in an SVG format (=file type). The rest of the formatting parameters in savefig should remain in their default values.

5. Git Repository

Your program should be submitted as a link to a git repository with the name 'computersProject_<your_name>'. It is recommended that you also use git while writing your code for backup and version handling. The instructions for opening a github account and using it with pyCharm are in the Moodle (week 8).
The link for a repository can be taken from the github.com repository page:



For using git with the Wing IDE, follow [this] tutorial. For using git through Linux command line or Windows cmd (though we stress that PyCharm or Wing is recommended), you can follow [this] tutorial.

6. General Instructions

1. The main module of your project should be called *main.py*, and it will contain a function
   fit_linear(filename)
   which will be the head function of the project. If you implement the bonus, the bonus function will be in this file too.
2. Your functions names, output prints and figure should match the instructions **exactly**.
3. Your code will be examined for the use of code styling as taught in week 5 (follow the slides in the course Moodle website).

7. Bonus - 15 points

This fit is only valid for linear functions. In other types of functions, we are not so lucky, and the way to minimize χ^2 is to numerically search for the best values for a and b.
In this stage, you will write a (very simple) searching program:

search_best_parameter(filename)

The program scans all possible values for some parameter, computes χ^2 (using equation (1)) for it and prints the value that minimizes χ^2. You will also plot χ^2 as a function of the parameters you tried.

The input file format will be as follows:

[data points, axis titles]

a <a_initial> <a_step_size> <a_final>
b <b_initial> <b_step_size> <b_final>

An example input file is added to this instruction file. The data points and axis titles are read as in section 1. The lower lines indicate the initial values, step sizes and final values for a and for b. For example, if the input file contains:
a 9 0.01 12

$b \; 7 \; 0.1 \; 20$,

you will need to compute χ^2 for *a = 9, 9.01, 9.02 … 12*, and for *b = 7, 7.1, 7.2 … 20*.
The program then chooses the best *(a, b)* pair out of the ones chosen. It then performs two actions:

1. Prints the values of a, b and χ^2 to screen as in section 3, choosing da and db as the step sizes for the sampling data, and plots and saves the function and data points as in section 4.
2. For the best value of *b*, plots a graph of χ^2 as a function of *a*. The x axis title will be ‘a’ and y axis title will be chi2(a, b = <*best_b*>), when <*best_b*> will be replaced by the value of b you used.
   The plot will have a blue line, and all other plotting options will be the default for the pyplot.plot function.
   Save the figure as ‘numeric_sampling’ in the svg format.