Winter Quarter 2019 – UCSB Physics 129L
Homework 7
Due Friday, March 8, 5 pm

Read carefully the instructions on the website on how to prepare your homework for turning it in to the TAs.
If your last name starts with A through N, send the homework to Jenny. Otherwise send it to Francesco.
The emails of the TAs are on the website.
Put the instructor in cc to the email.

Make sure to have your rpi updated to pick up the latest/greatest example programs from class.

Exercise 1

The experimental apparatus in the figure consists of four thin charged particle detectors at $x = 2, 3, 5$ and 7 cm. Each detector is segmented with many 50 μm wide strips that measure the y-coordinates of charged particles going through. (Note: if the width of the strip is w, the y-coordinate is measured with an uncertainty $w/\sqrt{12}$ [1]).
Particles traveling along the y-axis in the positive x direction decay somewhere in the vicinity of ($x = 0$ $y = 0$) into two charged particles that leave hits in the detectors. You are interested in finding the x coordinate of the decay for a given pair of tracks. A simulation of many such particle decays has been carried out. The simulated data are in
`/home/pi/physrpi/campagnari/python/straightTracks.txt`
Each line of this file has information for one pair of tracks as follows:

```
X0 Y0 y00 y01 y02 y03 y10 y11 y12 y13
where
X0, Y0 = true common point of origin of pair (Y0 always 0)
yij = digitized coordinate of track i at detector j
All lengths in cm
```

Your job now is to write a program using the `yij` and the known x positions

[1] If you do not believe this, calculate for yourself the variance of a set of numbers picked uniformly at random betwee 0 and w.

of the four detectors to estimate `X0`. To do this, you need to fit the coordinates of each track to a straight line. You can use either `np.polyfit` or `curve_fit` from `scipy.optimize`. From the straight line fit, for each pair, you can extract the intercepts of the two tracks with the x-axis: (`X1` and `X2`). Then, to check how well you are doing, make histograms of

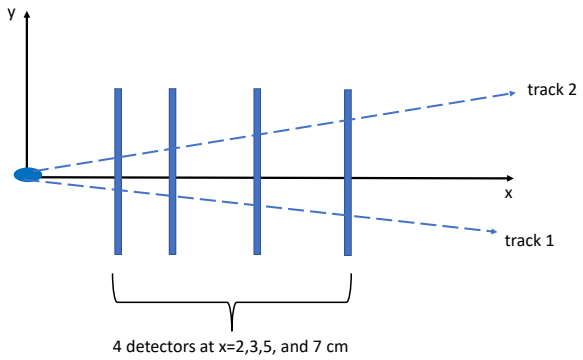
- `X1-X0` and `X2-X0` (put both quantities into the same histogram).
- `Xav-X0` where `Xav` is the average of `X1` and `X2`

The histograms should have 100 bins between -500 and 500 μm. Include a statistics box using `ccHistStuff.py` (it is interesting to see how many overflows and underflows you have!!). You should find (not surprisingly) that `Xav` is usually a better estimated of `X0` than `X1` or `X2`.

This last part is not for credit, try it if you feel like it:

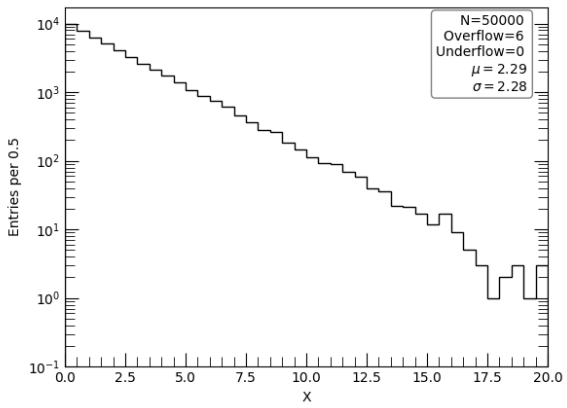
Keep track of the uncertainties on `X1` and `X2`. To do this you need access to the covariance matrix. Because (as discussed in class) the version of `np.polyfit` installed on the rpi has problems with covariance matrices, you need to use `curve_fit`. Then for each intercept `Xi` extract its uncertainty and histogram the pull, defined as `Xi-X0` divided by the uncertainty. If everything is well, the pull should be close to a Gaussian of mean 0 and sigma one. Then, instead of calculating the average of `X1` and `X2`, calculate the **weighted** average `Xav_w`, and plot `Xav_w - X0`.

Finally: you probably have developed a sense by now that some intercepts are more precise than other. Think about why that might be, and "explore" the data to verify your theory.



Exercise 2

In Homework 4 Exercise 1 you were asked to plot some data from a npy file. The data were exponentially distributed, and they look best when displayed on a log scale such as in the Figure below, with 40 bins, from 0 to 20.



Perform a χ^2 fit of the histogram as defined above to an exponential function $f(x) = p_0 e^{-p_1 x}$. Construct the χ^2 as a sum over bins of $(N - f(x))^2/N$, where N is the number of entries in the bin and x is the center of the bin. Setting the denominator to N assumes that N is large enough to be in the Gaussian regime[2]. For this reason limit your fit to the first 25 bins of the histogram. Superimpose your fitted function on the histogram and output the fitted parameters and the fitted χ^2 to the screen. You can recycle your solution to Homework 4 Exercise 1 as a starting point (even better: you can start from the solution provided on the website, since it sets up the appropriate binning for you).

Do not use any prepackaged fitting routines. Use the method described in class. Lecture notes are available on the website. Examples can be found in:

`/home/pi/physrpi/campagnari/python/simpleFit_v1.py`
`/home/pi/physrpi/campagnari/python/simpleFit_v2.py`

[2]Strictly speaking the denominator should be $f(x)$, but this is a detail that is usually overlooked.

Exercise 3

We can improve on the determination of `X0` in Exercise 1 by performing a constrained fit, i.e, instead of fitting the two tracks separately, we fit them simultaneously, imposing the requirement that they have the same intercept with the x-axis. So instead of having two fits with two free parameters each, we have one fit with three parameters in total. (It takes three parameters to describe two lines that intersect the x-axis at the same point). The single χ^2 is then constructed from the 8 measurements and the predicted track hits. Perform this constrained fit using the method described in class. Plot `Xf - X0` in a histogram with 100 bins between -500 and 500 μm, where `Xf` is the fitted value of the intercept. Plot also the pull, i.e., `Xf - X0` divided by the uncertainty in `Xf`.

Note: this is a semi-realistic problem in experimental high energy physics. In the real world, the tracks are in 3D (not 2D), they bend in a magnetic field, they are constrained to come from the same 3D point (not simply $y = 0$), there may be more than 2 tracks in the vertex, the detector is more complicated, etc. etc.