Scientific Computing: Basic Methods
J. Bürki, M. Block and R. Barniol Duran

Homework 7: Histograms and Function Fitting in Python

For the following exercises, write your code in one `.py` *file for each problem. Please, zip them all in a single file using your last name as in "barniol_hw07.zip."*

Make sure that your scripts run without error in order to get credit. Do not hesitate to ask for help if needed!

Problem 1 Normally distributed random numbers (5 pts)

Check that the random numbers generated by `numpy` for a normal distribution are indeed normally distributed; that is, the distribution is the Gaussian function:

$$d(x) = d_0 e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

Do this by plotting a histogram from a large number of normally distributed random numbers of average $\mu = 5$ and standard deviation value $\sigma = 1.25$, and try fitting this distribution to a Gaussian, Eq. (1), using its height d_0, mean value μ and width σ as fitting parameters. You'll need to find the midpoints of the edges as described in the notes to use as the x values in the call to `curve_fit`.

Add the midpoints as blue circles and the fit as a dashed, red line to your plot. Play around with the number of bins you use.

Optional practice (not required): If you do this process with `density=True`, it should be the case that the integral of your fitted function (from $-\infty$ and $+\infty$) should be close to 1. The integral of Eq. (1) above works out to be:

$$\sqrt{2\pi} d_0 \sigma \tag{2}$$

Plug in your fitted values for d_0 and σ in (2) and see how close it is to 1 (do this within your `Python` script and print your results to the screen).

Problem 2 Linear fit (5 pts)

Import the data stored in the file `HW07-linearFit.dat` and fit it to several polynomials of order n

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{3}$$

for various values of the polynomial order n.

Plot the data and fits for various values of $n = 1, 2, 3, \ldots$ (stop at the value of n when you find a reasonable fit) together on the same graph (make sure to include a legend that indicates the value of n used for each fit).

What minimum value of n do you need to have in the polynomial to obtain a reasonable fit?

Problem 3 Nonlinear fit (10 pts)

The data stored in `HW07-nonLinearFit.dat` was generated from a combination of 2 sinusoidal functions of similar frequencies and different amplitudes,

$$f(t) = A\sin(\omega_1 t) + B\sin(\omega_2 t) \tag{4}$$

Find the amplitudes A, B and frequencies ω_1, ω_2 using `curve_fit`.

Note: If you just apply `curve_fit` without giving it any guidance, you're likely to get a terrible fit! Remember that I mentioned that nonlinear fitting is sort of an art? Well, this problem is one of those cases, and will require some care if you want to get even a decent fit. The goal of this exercise is to guide you through this case to show you what can be done when fitting is no so easy...

You will have to suggest initial guesses for the parameters by using the `p0` argument to specify a tuple of initial guesses in the order the arguments occur in the fit function. You can often times get a rough estimate of what these initial guesses should be from a plot of the data. Note however that it is not always enough to do that, and unless your "guesses" are very good, it will probably not be sufficient in this case.

Try the following procedure:

(i) First fit the data to a single sine function $f_1(t) = A\sin(\omega_1 t)$. (You'll probably need to give initial guesses for the parameters A and ω_1);

(ii) Add a second sine function $B\sin(\omega_2 t)$ and fit for its parameters B and ω_2 while keeping the parameters A and ω_1 obtained in (i) for the first sine function fixed; you will also need to provide initial guesses for this second set of parameters, *keeping in mind* that the amplitude B and frequency ω_2 should be similar to the first set;
Another way to say this is you now want to fit to the whole Eq. (4), but keeping the values of A and ω_1 fixed.

(iii) Finally, fit for all parameters, starting with the values obtained in (i) and (ii) as initial guesses for A, B, ω_1, and ω_2.

You'll want to create a single plot showing the data, and the various steps of your fit (various fits) described above (don't forget the legend).

Problem 4 Data import & Histogram (5 pts)

In this problem, you'll analyze some data that, while made up for this problem, was inspired by experimental results that observed quantization of conductance in metallic nanowires.

Use the data contained in the attached file `HW07-doubleGaussian.dat`. Let's call the data G, which is the symbol commonly used for conductance (the reciprocal of the resistance) and will be assumed to be in units of $2e^2/h$ (the unit of quantized conductance).

(a) Start by importing the data and plot it as a scatter plot of G vs. the point number; don't forget to label your axes properly.

You should see data that seem to follow a curve that overall increases while moving along the horizontal axis, but seems to flatten a couple of times. These flatter parts, or *plateaus*, mean that some values of G are more likely than others to be observed. In order to get a better idea of the relative probabilities of observing any value of G, we need to "count" how many times each value of G occurs, or in other words, make a histogram of the G values.

(b) Make such a histogram of the G data. Try different number of bins until you get a "good-looking" histogram.
Make sure to obtain the midpoints of the edges, as described in the notes. Make a new figure that includes both the histogram and the midpoints. You should observe a double-peaked distribution that corresponds to the two plateaus in the initial plot.

(c) Try to fit the histogram data to a sum of two Gaussians such as that of Eq. (1) in Problem 1 above.
Depending on how many bins you used (feel free to experiment with that number), `curve_fit` may give a pretty good fit without any further work. If not, try and get estimates of the positions, height, and half-width of the peaks and provide these as initial guesses to `curve_fit`.
Add your fit to the plot.

Optional practice (not required): Another way to identify the two peaks is to separate the initial data into two sub-arrays and calculate the mean and standard deviation of each set of data.

A little trial and error shows that roughly the first 1100 points of data correspond to the first peak, while the rest correspond to the second peak.

(d) Separate the array (remember how to do slicing in `Python`? If you have an array called `data`, you need to use `data[:1100]` and `data[1100:]`), and make a new figure (or subplot) showing histogram plots for each of the two sets of data on the same axes (you'll need to call `plt.hist` twice; consider adjusting the transparency with the alpha argument).

(e) Calculate the mean and standard deviation for each peak using `numpy` functions on the data.
Compare those results to the values obtained with the fit from part (c).