Université de Genève Autumn term 2022

Statistical Methods in Physics (14P058)

Prof. Federico Sánchez (federico.sancheznieto@unige.ch)
Dr. Hepeng Yao (hepeng.yao@unige.ch)
Dr. Knut Zoch (knut.zoch@unige.ch)

Exercise VII – Goodness of fit

15 December 2022, 9:15, room: SCI–202

In this exercise, we will look at the Crystal Ball function: a function that combines Gaussian behaviour with a power-law low-end tail. The function and its first derivative are continuous despite this stitching. The function is given by:

$$f(x;\beta,m,\mu,\sigma) \propto \begin{cases} -\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) & \text{for } \left(\frac{x-\mu}{\sigma}\right) > -\beta \\ A\cdot\left(B-\frac{x-\mu}{\sigma}\right)^m & \text{for } \left(\frac{x-\mu}{\sigma}\right) \leq -\beta \end{cases} \tag{1}$$

where A, B are parameters that can be expressed through β and m. β defines where the definition switches from a Gaussian to a power-law distribution, and m is the power of the power-law tail. μ and σ are the central value and the standard deviation of the Gaussian part of the Crystal Ball function, respectively. The function can be implemented in Python using `scipy.stats`:

```python
import scipy.stats as stats
def CrystalBall(x, beta, m, mu, sigma, norm):
    """Takes x values and evaluates Crystal Ball at that point."""
    return norm * stats.crystalball.pdf(x, beta=beta, m=m, loc=mu, scale=sigma)
```

You perform a HEP experiment, but you are not entirely sure if the underlying probability distribution function is a Crystal Ball or a simple Gaussian function. After histogramming the results in 7 equidistant bins between $x \in [-5, 2]$, these are the bin centres, y values and uncertainties on y you obtain:

```python
x     = np.array([-4.5, -3.5, -2.5, -1.5, -0.5,  0.5, 1.5])
y     = np.array([ 0.3,  1.0,  0.9,  3.1,  6.9,  6.5, 1.7])
yerrs = np.array([ 0.2,  0.3,  0.4,  0.4,  0.5,  0.3, 0.2])
```

Your idea is to test both the Crystal Ball and the Gaussian distribution for compatibility with the obtained data. For that, perform the following tasks:

1. Implement functions to evaluate the Crystal Ball and Gaussian distribution. For the former, you can simply take the definition above. *Hint:* for the Gaussian function, there is the equivalent module called `scipy.stats.norm`.
2. Implement a function to calculate the χ^2 goodness of fit according to the following formula:

$$\chi^2 = \sum_{i=0}^{N} \left(\frac{y_i - f(x_i; p_1, p_2, \dots)}{\sigma[y_i]}\right)^2 \tag{2}$$

where x_i, y_i are the obtained measurement results and $\sigma[y_i]$ is the uncertainty on y_i. f is the probability density function with parameters, $p_1, p_2, \ldots$ against which the results should be compared. *Hint:* the function should have the following signature:

```python
def calc_chi2(x, y, yerrs, func, params):
"""Calculates chi2 of x, y dataset w.r.t. to given function."""
```

3. Perform a fit to the measured values with a Gaussian function. You can use the module `scipy.optimize.fit` and the Gaussian function you implemented in Task (1). Make sure the uncertainties on y are included into the fit through the `sigma` keyword. Plot the Gaussian function using the "best fit" parameters and compare it with the measured values.

4. Repeat the previous task, but this time use the Crystal Ball function you implemented. Plot the Crystal Ball function using the "best fit" parameters and compare it with the measured values.

5. Use the χ^2 implementation of Task (2) and calculate the χ^2 value for the two fitted functions.

6. Compare the χ^2 values with the corresponding two χ^2 distributions. How many degrees of freedom does each of the two fits have? In what quantile do the obtained χ^2 values lie? What does this comparison mean for the "goodness of fit"?