Kirchhoff-Insitut für Physik
Physikalisches Insitut

Winter semester 2013-14
KIP CIP Pool (1.401)

Exercises for Statistical Methods in Particle Physics

http://www.physi.uni-heidelberg.de/~nberger/teaching/ws13/statistics/statistics.php

Dr. Niklaus Berger (nberger@physi.uni-heidelberg.de)
Dr. Oleg Brandt (obrandt@kip.uni-heidelberg.de)

Exercise 4: The Central Limit Theorem and applied MC simulation

4. November 2013
Hand-in solutions by 14:00, 11. November 2013

Please send your solutions to obrandt@kip.uni-heidelberg.de by 11.11.2013, 14:00. Make sure that you use *SMIPP:Exercise04* as subject line. Please put each macro into one separate .C or .py file, which can easily be tested (i.e. executable via e.g. `root -l my\_code.C`. Note that for this to work, the main method of the macro has to be called with the same name as the macro, i.e. `void my_code( <some optional arguments> )`). If plots are requested, please include print statements to produce pdf files in your code, and provide the plots separately. Please add comments to your source code explaining the steps – try to think of somebody who is not familiar with the course should be able to understand easily from your source code what each part of it is there for. Test macros and programs before sending them off...

1 Central Limit Theorem

Measurements can acquire uncertainties form many different sources. Usually, it is assumed that the uncertainties follow a Gaussian distribution. This is justified by the Central Limit Theorem. It states that the sum of N independent variables taken from the same distribution converges to a Gaussian distribution when $N \to \infty$. Note that the variance has to be defined for the single distribution.

1.1 The uniform distribution

In this part of the exercise, we study the Central Limit Theorem using a sample of uniformly distributed random numbers. Write a programme that:

- generates uniform random numbers on the interval $[0, 5]$,
- plots the distribution of the average of k random numbers with $2 \leq k \leq 20$ (10,000 times for each k),
- fits a Gaussian to the distribution for each k,
- and plots the mean and the sigma parameter of the Gaussian versus k.

For how many averaged random numbers does the distribution look like a genuine Gaussian?
Note:
A Gaussian fit to a distribution can be performed in ROOT by defining the TF1 object with the formula string parameter, i.e. the second parameter, set to `"GAUS"`, and calling the `Fit(<name of TF1>,<parameters>)` method of the object that one wants to fit, i.e. in this case a `TH1`.
(Please hand in your code called ex_4_1_1.C/.py and provide plots titled ex_4_1_1.pdf)

1.2 The Breit-Wigner distribution

The Breit-Wigner distribution is often used in particle physics, as it describes the invariant mass of a decaying particle. In mathematical terms, the Breit-Wigner distribution is a special case of the Cauchy PDF given by:

$$f(x) = \frac{1}{\pi}\frac{1}{1+x^2}\,.$$

- Show (analytically) that for a uniformly distrubted r in $[0,1]$

  $$x(r) = \tan[\pi(r - \frac{1}{2})]$$

  follows the Cauchy distribution.

- Using the above result write a macro to generate 10,000 Cauchy-distributed random numbers and plot them in a histogram.

- Modify your macro from before to generate repeated experiments each consisting of k independent Cauchy distributed values for e.g. $k = 10$. For each sample, compute the sample mean $\bar{x} = \frac{1}{k}\sum_{i=1}^{n} x_i$. Compare the histogram of $\bar{x}$ with the original histogram of x. Does it agree with what you would expect from the Central Limit Theorem? Interpret the result!

(Please hand in your code called ex_4_1_2.C/.py and provide plots titled ex_4_1_2.pdf)

2 MC simulation of a muon beam

The Paul Scherrer Institute (PSI) in Switzerland provides the most intense continuous muon beams in the world. These beams are created by shooting over 2 mA of 590 MeV/c protons at a 4 cm thick carbon target. In the $p-C$ interactions, many pions are generated and then stopped in the target. Charged pions decay mostly to a muon and a muon (anti)neutrino, $\pi^+ \to \mu^+\nu_\mu$ and $\pi^- \to \mu^-\bar{\nu}_\mu$. These are the muons used in the experiments. In the coordinate frame where the pion is at rest, the muons will have a fixed momentum of $p_{max} = 29.79$ MeV/c defined by the two-body pion decay. Muons traversing the target material will lose some energy, thus the highest momentum muons tend to come from the target surface. In fact, the muon intensity behaves as

$$I(p) = \begin{cases} I_0 \cdot p^{3.5} & \text{if } 0 < p < p_{\max} \\ 0 & \text{otherwise} \end{cases}\,, \tag{1}$$

where I_0 is an (arbitrary) normalization factor.

2.1 The ideal muon beam

For the simulation of the future $\mu \to eee$ experiment at PSI, we would like to generate muons with the momentum distribution given by Eq. 1. Generate 100'000 muon momenta using the transformation method and fill them into a histogram with appropriate binning. (Please hand in your code called ex_4_2_1.C/.py and provide plots titled ex_4_2_1.pdf)

2.2 Towards a realistic muon beam

In reality, the sharp edge at $p_{\max} = 29.79$ MeV/c is washed out because many pions are not perfectly at rest in the target. The resulting distribution is the intensity from the above problem (I_{ideal}) convoluted with a Gaussian distribution

$$I_{\text{real}}(p) = \int_{-\infty}^{+\infty} I_{\text{ideal}}(\tau)\cdot g(p-\tau)d\tau = \int_{-\infty}^{+\infty} g(\tau)\cdot I_{\text{ideal}}(p-\tau)d\tau \tag{2}$$

where $g(x)$ is the Gaussian or normal distribution,

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In our example, the width σ of the distribution happens to be a convenient 1 MeV/c and the center μ is conveniently at 0. The integral in Equation 2 has no analytical solution, so we will approximate it numerically by a sum (using the fact that the normal distribution falls off rather rapidly):

$$\int_{-\infty}^{+\infty} g(\tau) \cdot f(x-\tau) d\tau \approx \frac{\sum_{\tau=-a}^{a} g(\tau) f(x-\tau)}{\sum_{\tau=-a}^{a} g(\tau)},$$

where a is chosen as a few units of σ of the normal distribution (typically, 3 is a reasonably good choice) and the steps of τ are chosen to be small enough. Write a function that implements this convolution starting from a function implementing a normal distribution with a σ of 1 MeV/c and a mean of 0 MeV/c (does not need to be normalised) and a function implementing I_{ideal}. In native (C++) ROOT, these functions should have prototypes of the form

```
double myFunction(double* x, double* par) {
    p = x[0] ;
    parameter0 = par[0] ;
    parameter1 = par[1] ;
    ...
    return result ;
}
```

where `x` is an array of the running variable (of length 1 our in case of a onedimensional function) and `par` is an array of the function parameters. In Python, everything is somewhat simpler due to implicit typing:

```
def myFunction(x, par):
    p = x[0]
    parameter0 = par[0]
    parameter1 = par[1]
          ...
          return result
```

These functions can then be used in TF1 objects to draw the function.

```
TF1 * rootfunction = new TF1("myFunction",myFunction,0,30,2) ;
rootfunction->SetParameter(0,1.3) ;
rootfunction->SetParameter(1,2.7) ;
rootfunction->Draw() ;
```

where the arguments to the constructor are name, the actual function, the lower and upper edges of the function range and the number of parameters, which can then be set via `SetParameter(index, value)`. Determine the number of steps needed for τ by drawing the function and increasing the number until you obtain a smooth behaviour.
(Please hand in your code called ex_4_2_2.C/.py and provide plots titled ex_4_2_2.pdf)

2.3 The realistic muon beam

Use the hit-and-miss method to generate 100'000 muons with the distribution obtained in the last problem in the range from 0 to 35 MeV/c. Plot the resulting distribution. Count how many random numbers you need per generated muon. Could this be made more efficient? How (without implementation)?
(Please hand in your code called ex_4_2_3.C/.py and provide plots titled ex_4_2_3.pdf)