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 3: Pseudo random number generators

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

Please send your solutions to `obrandt@kip.uni-heidelberg.de` by 4.11.2013, 14:00. Make sure that you use *SMIPP:Exercise03* 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 Monte Carlo Integration

Note that this question is Exercise 2.4 reposed, as the last exercise sheet was rather on the long side. If you already handed in 2.4 last time but would like to polish your solution further, you are invited to re-submit.

Calculate the following integral

$$\int_0^{100} \cos(2\pi x)\mathrm{d}x \tag{1}$$

- analytically;
- numerically by the approximation:

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=1}^{N} f(x_i)\Delta x \tag{2}$$

  where the function f is evaluated at N equidistant points with a constant step size $\Delta x = (b-a)/N,\ x_i = a + \Delta x \cdot (i - 1/2)$. Graph how the error, i.e. the signed difference between your calculation and the true result changes as you calculate f for an increasing number of sampling points N = 1, 2, ..., 150. You can either (ab)use a histogram for this (with `SetBinContent()` you can set bin contents to arbitrary values) or, more elegantly, use a `TGraph` in conjunction with the draw option "`AP`". Note that in the latter case a `TGraph` will not display just by itself unless you draw it with the "`A`" option;
- by Monte Carlo integration using pseudo random numbers

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{N}\sum_{j=1}^{N} f(x_j) = (b-a)\cdot\langle f\rangle \tag{3}$$

where the x_j are random values in the interval $[a, b]$, and $\langle f \rangle$ is the mean of the function value over the interval $[a, b]$. The squared *expected numerical error*, i.e. variance of the estimate of the integral is given by

$$\sigma_i^2 = (b-a)^2 \cdot \frac{\sigma_N^2}{N},$$

where σ_N^2 is the sample variance of the function f for N sampling points

$$\sigma_N^2 = \frac{1}{N}\sum_{j=1}^{N}(f(x_j) - \langle f \rangle)^2,$$

Again show the estimate (this time with its expected numerical error) as a function of $N = 1, 2, \ldots 150$. How does the convergence compare to the numerical approximation above? By the way, you can get 2π (in double precision) via `TMath::TwoPi()`.

2 Generating non-uniformly distributed random number distributions

Write a macro which generates a random number distribution according to $f(x) = 1 + x^2$ in the interval $[1, 1]$. As input, use the random numbers r_i in the interval $[0, 1]$ from a uniform random number generator.
We use the decomposition method where $f(x)$ is split into two parts: $f_a(x) = 1$ and $f_b(x) = x^2$. Thus, a certain fraction of the events will be generated according to f_a and the remaining events according to f_b, i.e. in reality we generate two random number sequences which are combined to obtain the final result. This fraction is determined by calculating the integral of both function on the interval [1, 1]. Since

$$\int_{-1}^{1} f_a(x)\mathrm{d}x = 2 \qquad \int_{-1}^{1} f_b(x)\mathrm{d}x = \frac{2}{3},$$

we have to generate in 3/4 of the cases a number according to $f_a(x)$, and in the remaining 1/4 of the cases according to $f_b(x)$.
First, a test value r_1 is generated. If this r_1 is less than 3/4, we generate a number according to f_a, and according to f_b otherwise. For $r_1 \geq 0.75$, there are two ways to generate random numbers distributed according f_b:

- **"Hit-and-miss" method**:
    1. Generate a pair of values r_j and $r_k \in [0, 1]$.
    2. Transform both random numbers to the considered sampling and result intervals, respectively, i.e. in our case $r_{j,\mathrm{trans}} \in [1, 1]$ and $r_{k,\mathrm{trans}} \in [0, f_{b,\max}]$. Note that $f_{b,\max} = 1$ in our case).
    3. Now, if $r_{k,\mathrm{trans}} \leq f_b(r_{j,\mathrm{trans}})$, fill the histogram with $r_{j,\mathrm{trans}}$ ("hit"). Otherwise, return to step 1 ("miss").
- **Transformation method**:
    - We have random numbers r_j distributed according to a uniform distribution $g(r)$ and want to generate random numbers x_i according to a probability density function (p.d.f.) $f(x)$ in the interval $[p, q]$;
    - From the equation $f(x)\mathrm{d}x = g(r)\mathrm{d}r$ we obtain the cumulative distribution function (c.d.f.) $F(X)$ and thus $r = F(X) = \int_{-\infty}^{x} f(x')\mathrm{d}x'$, which we solve as $x = F^{-1}(r)$;

- If r_j are uniformly distributed random numbers between $F(p)$ and $F(q)$, the x_i are following the p.d.f. $f(x)$;
- The method works well if $F(x)$ is analytical and can be easily inverted.

Note that here, $x_i = (3r_j - 1)^{1/3},\ r_j \in [0, \frac{2}{3}]$.

Provide the source code and two histograms with random numbers distributed according to $f(x)$ using the two methods. Do you obtain consistent results?
(Hint: x^y in C++: `std::pow(x,y)`, from `#include <cmath>`.)

3 Generating random numbers according to the exponential distribution

Generate random numbers according to an exponential distribution e^{-x} for $x > 0$. Take uniformly distributed random numbers in $[0, 1]$ and apply the transformation method. Why would it be computationally very expensive to apply the hit-and-miss method for e^{-x}?
Provide the source code and show the histogram of your result.