

Praktikum zu Moderne Methoden der Datenanalyse

Exercise 2: Random Numbers

Monte Carlo simulations heavily rely on random numbers. In principle random physics processes like e.g. the radioactive decay of nuclei can be used to obtain random numbers. However for practical purposes algorithms are used that produce pseudo random numbers. It is very important that the numbers from these algorithms do not contain any correlations and behave like real random numbers. Many tests have been developed to check for a (hidden) structure in a sequence of random numbers.

- **Exercise 2.1:**

Write a random number generator for uniformly distributed numbers u_j between 0 and 1 by using the formula for a linear congruent generator:

$$I_j = (a \cdot I_{j-1} + c) \bmod m \quad u_j = I_j/m$$

a , c and m are integer numbers. Choose a value for each of the three parameters which seems appropriate for a random number generator with a high periodicity, but be careful to avoid an overflow of the integer value range.

Optional exercise: Invent your own random number generation algorithm.

- **Exercise 2.2:**

Check the randomness of your random number generator from exercise 2.1, of a linear congruent generator with $a = 205$, $c = 29573$ and $m = 139968$ and of the default random number generator implemented in root (`gRandom->Rndm()`):

- Fill a one-dimensional histogram of k bins from 0 to 1 with $N \gg k$ random numbers. Then calculate the value

$$\chi^2 = \sum_{i=1}^k \frac{(N_i - N/k)^2}{N/k}$$

where N_i is the number of entries in bin i . The obtained value should follow a χ^2 distribution with $k - 1$ degrees of freedom. So on average $\chi^2/(k - 1)$ should be 1.

- Make a two-dimensional plot using N pairs of subsequent random numbers. Use the class `TGraph` and its method `SetPoint` for this. Draw it with the option “AP”. The displayed points should be equally distributed in the square $[0, 1] \times [0, 1]$ without showing a structure.
- Apply the sequence or up-down test: Compare subsequent random numbers and assign to the comparison the bit 0 if the successor u_{j+1} is smaller than the previous random number u_j , assign the bit 1 if $u_{j+1} > u_j$. Now look at sequences of bits with the same value. Determine the length k of each sequence and count how many sequences of length k were produced. This number $N(k)$ should on average have the following value:

$$N(k) = \frac{2[(k^2 + 3k + 1)N - (k^3 + 3k^2 - k - 4)]}{(k + 3)!}$$

Here $N + 1$ is the total amount of random numbers. Also check that the relation $\sum_{k=1}^N k \cdot N(k) = N$ holds.

Make a histogram of the obtained and of the expected $N(k)$ and plot both of them together (Draw option “SAME”). Use the method `SetBinContent` for the histogram of expected values. The method `TMath::Factorial` can be used to calculate the factorial.

• **Exercise 2.3:**

Generate random numbers according to an exponential distribution $\exp(-x)$ for $x > 0$. Take the uniformly distributed random numbers (`gRandom->Rndm()`) and apply the transformation method. Write 100 000 exponentially distributed random numbers to an ntuple (root class `TNtuple`).

Hint (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 p.d.f. $f(x)$
- From equation $f(x)dx = g(r)dr$ we get $r = F(x) = \int_{-\infty}^x f(x')dx'$, which we solve for $x = F^{-1}(r)$
- If r_j are uniformly distributed random numbers between 0 and 1, the x_j are following the p.d.f $f(x)$.
- The method works well if $F(x)$ is analytical and can be easily inverted.

• **Exercise 2.4:**

Calculate the integral

$$\int_0^{100} \cos(2\pi x) dx$$

- analytically

- numerically by the approximation

$$\int_a^b f(x) dx \approx \sum_{i=1}^N f(x_i) \Delta x$$

where the integrand f is evaluated at N values with constant step size $\Delta x = (b - a)/N$, $x_i = a + \Delta x \cdot (i - 1/2)$

- by Monte Carlo integration using N uniformly distributed random numbers x_i between a and b .

Plot the result of the integration as a function of N for $N = 1$ to $N = 150$. Add the expected error of the Monte Carlo integration to the plot. The variance V of the Monte Carlo integration is given by $V = (b - a)^2/N \cdot V[f(x_i)]$.