Traditional Monte-Carlo Algorithm

Task. We know:

- the function $f(x_1, \ldots, x_n)$ used for data processing,
- the results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of measuring the values $x_1, \ldots, x_n$, and
- the standard deviations σ_i of measurement errors $\Delta x_i = \widetilde{x}_i - x_i$; we assume that the mean values of measurement errors are 0s (i.e., in statistical terms, that there is "no bias").

We have computed the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of data processing, and we want to find the standard deviation σ of the resulting uncertainty of data processing, i.e., of the difference

$$\Delta y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n).$$

Algorithm. Several (N) times, for $k = 1, \ldots, N$, do the following:

- simulate n variables r_{ki} which are normally distributed with 0 mean and standard deviation 1 (in the following text, we explain how to do it);
- compute the difference

$$\Delta y_k = \widetilde{y} - f(\widetilde{x}_1 - \sigma_1 \cdot r_{k1}, \ldots, \widetilde{x}_n - \sigma_n \cdot r_{kn}).$$

After that, we estimate σ as follows:

$$\sigma = \sqrt{\frac{1}{N} \cdot \sum_{k=1}^{N} (\Delta y_k)^2}.$$

Comment. The relative accuracy of this estimate is $\approx \dfrac{1}{\sqrt{N}}$. So:

- if we want to find σ with accuracy 20%, we should take $N = 50$;
- if we want to find σ with accuracy 10%, we should take $N = 100$, etc.

How to simulate normal distribution. Most programming languages have a method for simulating random numbers that are uniformly distributed in the interval $[0, 1]$.

To simulate a normal distribution with 0 mean and standard deviation 1, we can:

- call this uniform random number generator 12 times, getting 12 simulated values $u_1, \ldots, u_{12}$, and then
- take
$$r = (u_1 - 0.5) + \ldots + (u_{12} - 0.5).$$