Chapter 3

The Monte Carlo Method

For these exercises you will need a random number generator to produce random values uniformly distributed between zero and one. A simple FORTRAN example is given in the file `random.f`. This routine is mainly for pedagogical purposes and simple applications. More sophisticated routines such as `RANMAR` or `RANLUX` can be found in the CERN Program Library.

Exercise 3.1: (a) Using `random.f` or another random number generator, write a short program to generate 10000 random values uniformly distributed between zero and one, and display the result as a histogram with 100 bins.

Exercise 3.2: Modify the histogram from Exercise 3.1 to have only 5 bins and $N = 100$ entries. The generated histogram can be regarded as an observation of a multinomially distributed vector $(n_1, \ldots, n_5)$, with parameters $N = 100$ and $p_i = 0.2$ for $i = 1, \ldots, 5$.

(a) By placing the code to generate the histogram in a loop, modify the program to repeat the Monte Carlo experiment 100 times, each time with a different seed value. (As long as the program is not terminated after each experiment, a new seed will be used automatically.) Produce a histogram of the value of any bin n_i (e.g. for $i = 3$) after each experiment. This should follow a binomial distribution with mean $Np_i = 20$ and standard deviation $\sqrt{Np_i(1-p_i)} = 4$.

(b) Produce a scatter plot (two-dimensional histogram) for the values of any two bins n_i and n_j. These should have a covariance $\mathrm{cov}[n_i, n_j] = -Np_ip_j = -4$ or a correlation coefficient $\rho = -4/4^2 = -0.25$.

If you are using the HBOOK histogram package from the CERN Program Library, you can use the routine `HUNPAK` to unpack the values $(n_1, \ldots, n_5)$ after each experiment.

Exercise 3.3: Consider a sawtooth p.d.f.,

$$f(x) = \begin{cases} \frac{2x}{x_{\max}^2} & 0 < x < x_{\max} \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{3.1}$$

(a) Use the transformation method to find the function $x(r)$ to generate random numbers according $f(x)$, cf. Section 3.2. Implement the method in a short computer program and make a histogram of the results. (Use e.g. $x_{\max} = 1$.)

(b) Write a program to generate random numbers according to the sawtooth p.d.f. using the acceptance-rejection technique, cf. Section 3.3. Plot a histogram of the results.

Exercise 3.4: The purpose of this exercise is to generate random numbers according to a Gaussian p.d.f. A number of algorithms exist for this purpose, implemented, for example, in the routine `RNORMX` from the CERN library. A simple algorithm suitable for pedagogical purposes is based on the central limit theorem: a sum of random variables becomes Gaussian in the limit that the number of terms in the sum is large, as long as none of the terms make up a significant fraction of the sum (cf. SDA Chapter 10).

(a) Suppose x is uniformly distributed in $[0, 1]$ and consider the sum of n independent x values,

$$y = \sum_{i=1}^{n} x_i \,. \tag{3.2}$$

Show that the expectation value of y is $n/2$ and the variance is $n/12$. Show that, as a consequence, the variable

$$z = \frac{\sum_{i=1}^{n} x_i - \frac{n}{2}}{\sqrt{n/12}} \,. \tag{3.3}$$

has a mean of zero and unit standard deviation.

(b) Write a computer program to generate values z as defined in (a) for arbitrary values of n. Make histograms of 10000 values of z for $n = 0, \ldots, 20$. At what point does the distribution appear approximately Gaussian? A convenient choice for a simple Gaussian generator is $n = 12$. Comment on the limitations of such an algorithm. Optional: Derive the explicit form of the p.d.f. of z for $n = 2$.

Exercise 3.5: The variable t follows an exponential distribution with mean $\tau = 1$ and x is Gaussian distributed with mean $\mu = 0$ and standard deviation $\sigma = 0.5$. Write a Monte Carlo program to generate values of

$$y = t + x \,. \tag{3.4}$$

Here the value t could represent the true decay time of an unstable particle, and the value x the measurement error, so that y represents the measured decay time. Make a histogram of the values. Modify the program to investigate the cases $\tau \ll \sigma$ and $\tau \gg \sigma$.

3.6: Consider a random variable x distributed according to the Cauchy (Breit-Wigner) p.d.f.,

$$f(x) = \frac{1}{\pi} \frac{1}{1 + x^2} \,. \tag{3.5}$$

(a) Show that if r is uniformly distributed in $[0, 1]$, then

$$x(r) = \tan[\pi(r - \tfrac{1}{2})] \tag{3.6}$$

follows the Cauchy p.d.f.

(b) Using the result from (a), write a computer program to generate Cauchy distributed random numbers. Generate 10000 values and display the result as a histogram.

(c) Modify the program in (b) to generate repeated experiments each consisting of n independent Cauchy distributed values for e.g. $n = 10$. For each sample, compute the sample mean $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Compare a histogram of $\overline{x}$ with the original histogram of x. (See also Exercise 10.6.)

Exercise 3.7: A photomultiplier is a device capable of detecting individual photons as illustrated in Fig. 3.1.[1] A photon strikes the photocathode, where there is a certain probability for it to eject an electron (called a photoelectron). The photoelectron is accelerated in an electric field towards an electrode (called a dynode). In the collision with the first dynode, the photoelectron can liberate further electrons. These are accelerated towards the second dynode, where more electrons are produced. This continues through a series of stages until the electrons produced at the final dynode are collected.

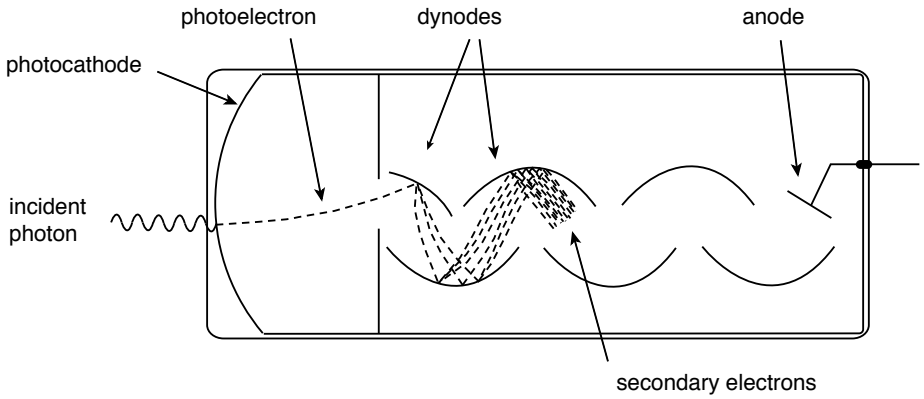

Figure 3.1: Schematic drawing of a photomultiplier tube.

The number of electrons produced at the ith dynode for each incoming electron can be modeled as a Poisson variable n_i with mean value ν_i, which in general can be different for each stage. Suppose the photomultiplier has N dynodes. The number of electrons n_{out} produced at the final stage for a single incident photoelectron has an expectation value (the *gain* of the photomultiplier),

$$\nu_{\text{out}} = E[n_{\text{out}}] = \prod_{i=1}^{N} \nu_i \tag{3.7}$$

(a) Write a Monte Carlo program to determine the distribution of the number of electrons n_{out} at the end of $N = 6$ dynodes produced by a single initial photoelectron, with $\nu = 3.0$ for each dynode. (Poisson random numbers can be generated with the routine `RNPSSN` from the CERN program library. A partial solution is given in the program `pmt.f`.) Run the program to simulate $M = 1000$ initial photoelectrons and make a histogram of n_{out}. Estimate the mean ν_{out} and variance $V[n_{\text{out}}] = \sigma^2_{\text{out}}$ by calculating the sample mean,

$$\overline{n}_{\text{out}} = \frac{1}{M}\sum_{i=1}^{M} n_{\text{out},i} \tag{3.8}$$

[1] For a more detailed description see e.g. C. Grupen, A. Böhrer and L. Smolik, *Particle Detectors*, Cambridge University Press, Cambridge, 1996.

and sample variance

$$s^2_{\rm out} = \frac{1}{M-1}\sum_{i=1}^{M}(n_{{\rm out},i} - \overline{n}_{\rm out})^2 \,. \tag{3.9}$$

(The sample mean and variance are described further in SDA Chapter 5.) Compare the sample mean to the value from equation (3.7). Compare the sample variance (or standard deviation) to the value that one would obtain from a Poisson variable of mean $\nu_{\rm out}$. Explain qualitatively why the standard deviation of $n_{\rm out}$ is much larger than in the Poisson case.

(b) One would like the standard deviation of $n_{\rm out}$ to be small in order to be able to determine as accurately as possible the number of initial photoelectrons (and thus estimate the number of incident photons). In some applications one would like to have the standard deviation small enough to distinguish between 1 and 2 photoelectrons; hence one tries to have a relative resolution, i.e. the ratio of standard deviation to mean, less than unity. One way of achieving this is to increase the mean number of electrons produced at the first dynode. This can be done by increasing the voltage so that the photoelectrons collide with a higher energy, and also by using a metal with a low work function, i.e. a high probability for secondary electron emission.

Modify the program from (a) so that the mean for the first dynode is larger, e.g. $\nu_1 = 6$. Run the program and estimate the ratio of standard deviation to mean of $n_{\rm out}$. Explain qualitatively why this gives a better resolution than in the case with all ν_i equal. Why does it not help much to increase the gain of the dynodes in the later stages of the photomultiplier?

(c) Try to extend the program to simulate $N = 12$ dynodes. You will quickly find out that it requires too much computing time to simulate the collision of each electron with each dynode. Instead, run the program for $N = 6$ with enough events to obtain a good estimate of the distribution of $n_{\rm out}$ (e.g. at least $M \approx 10^4$ events in a histogram with 50 bins from $0 \leq n_{\rm out} \leq 5000$). Generate numbers which follow this distribution using any method, e.g. acceptance-rejection. For each electron obtained after the first six dynodes, generate in a similar way the number of electrons that it produces in the next six. For the first six stages, use a distribution of $n_{\rm out}$ based on $\nu_1 = 6$ and the rest of the $\nu_i = 3$; for the last six, take all $\nu_i = 3$.