Time	Group	Submission in Moodle; Mails with subject: [SMD2023]
Th. 12:00–13:00	A	lukas.beiske@udo.edu and tristan.gradetzke@udo.edu
Fr. 08:45–09:45	B	jonas.hackfeld@ruhr-uni-bochum.de and ludwig.neste@udo.edu
Fr. 10:00–11:00	C	stefan.froese@udo.edu and vincent.latko@udo.edu

Exercise 11 *Simulation chain for neutrino detector* **10 p.**

For the following tasks, use the Python library `pandas`. Write the results of all individual subtasks, except the last one into a `DataFrame` and save this DataFrame as a `hdf5`-file called `NeutrinoMC.hdf5` with the key `Signal`. Then save the results of the last subtask in a separate `DataFrame` and save it in the same `hdf5` file with the key `Background`. To write a `hdf5` file the `DataFrame` has the method `to_hdf()`.

Important: The created `hdf5` file should **not** be handed in! The finished program must work without the already existing `hdf5` file.

(a) Signal MC

The flow of neutrinos is given by:

$$\Phi(E) = \Phi_0 \cdot \left(\frac{E}{\mathrm{TeV}}\right)^{-\gamma}.$$

Here, the spectral index is $\gamma = 2.7$, the lower energy limit is 1 TeV and the upper energy limit infinite.

Using the transformation method, simulate 10^5 signal events and save them in the `DataFrame` with the key `Energy`.

(b) Acceptance

The probability of detecting an event is energy dependent. This must be taken into account in the simulation. The detection probability can be described by the following equation:

$$P(E) = (1 - \mathrm{e}^{-E/2})^3.$$

Use the Neumann rejection method to consider detector acceptance for the signal events simulated in part a). Save the results in form of a *mask* (`True-False` sequence) under the key `AcceptanceMask` in the `DataFrame`.

Histogram the result of a) and b) using a log-log representation.

(c) Polar method

Implement the polar method known from the lecture in your project repository. To do this, add the `standard_normal` method of the `generator` class in the `project_<groupname>/random.py` file. To obtain random numbers of arbitrary Gaussian distributions the drawn numbers of the standard normal distribution must be shifted and scaled. To do this, implement the `normal` method. Test your `Polar method` by executing

```
python exercises/polar.py project_<groupname> -o polar.pdf .
```

Replace `<groupname>` with the name of your group.

If implemented correctly, all tests should pass successfully. Keep in mind, however, that successfully passed tests do not equate to everything being correct!

Include the overview PDF you created in your submission.

(d) Energy measurement

A realistic detector has a finite energy resolution. Moreover, the energy is not measured directly, but reconstructed with the help of energy-correlated observables. One such observable is, for example, the number of photomultipliers that have responded (hereafter referred to as hits).

The number of hits can be drawn from a normal distribution with the following properties:

$$\mathcal{N}(10 \cdot E/\mathrm{TeV}, 2 \cdot E/\mathrm{TeV}).$$

Here $\mathcal{N}(10E/\mathrm{TeV}, 2E/\mathrm{TeV})$ denotes the normal distribution with mean $10E/\mathrm{TeV}$ and standard deviation $2E/\mathrm{TeV}$. Convert the number of hits to a whole number, since only whole numbers of hits can occur. Also make sure that only values above zero can be considered for the number of hits N. Draw a new random number if if this condition is violated. Simulate the number of hits depending on the previously simulated energy and store the number under the key `NumberOfHits`. Plot the `NumberOfHits` in a log-log histogram. For this purpose, use the polar method you implemented to realize the normal distribution. You can import this with the following command:

Listing 1: Example of importing and usin the polar method (Replace `<groupname>` with the name of your group):

```
from project_<groupname>.random import Generator

gen = Generator(seed=0)

# generate 10000 random numbers with mu=0 and sigma=1
values = gen.standard_normal(size=10000)

# generate 10000 random numbers with mu=5 and sigma=3
scaled_values = gen.normal(loc=5, scale=3, size=10000)
```

(e) Spatial measurement

In the following, consider a square area detector with an edge length of 10 length units. The signal hits the detector at point (7,3). The spatial resolution is again energy dependent. Simulate the locations of the previously generated events by calculating a normal distribution for both x- as well as for the y-direction . Here σ is energy dependent and given by

$$\sigma = \frac{1}{\log_{10}(N+1)},$$

where N is the previously determined number of hits. Again, make sure that the drawn events are within the detector (draw new random numbers if necessary). Save the coordinates under the keys `x` and `y`. Plot the obtained locations in a two-dimensional histogram.

(f) Underground MC

The number of expected underground events is large in relation to the expected signal. Create a new `DataFrame` with the keys `NumberOfHits`, `x` and `y`.

Simulate 10^7 underground events with the following properties:

- The logarithm base ten of the number of hits follows a normal distribution with $\mu = 2$ and $\sigma = 1$.

- The x and y coordinates of the events are normally distributed around the center of the detector and $\sigma = 3$.
- There is a correlation of $\rho = 0.5$ between the x and y coordinates.

Plot the locations of the background events in a two-dimensional histogram and the number of hits in a one-dimensional log-log histogram.

If one draws standard normally distributed random numbers $x^\star$ or $y^\star$, then the normally distributed random numbers x and y with arbitrary μ, σ and correlation coefficient ρ result from the transformation:

$$x = \sqrt{1-\rho^2} \cdot \sigma \cdot x^\star + \rho \cdot \sigma \cdot y^\star + \mu$$
$$y = \sigma \cdot y^\star + \mu.$$