Department of Physics

Statistical Methods of Data Analysis

Exercise Sheet 04

Annsofie Tappe, Helen Thews, Rene-Marcel Lehner

SuSe 2023

Assignment N° #04

E9: Angle distribution

Task:

For the simulation of scattering processes, particles at an interaction point must receive a new, random direction. For this purpose, an angular distribution must be assumed from which deflection angles are drawn at random. In this task said angular distribution is to be created and implemented. The angular change $\Delta\Psi$ is given by the angular probability distribution

$$\mathrm{PDF}(\Delta\Psi) = \begin{cases} N \cdot \exp(-|\Delta\Psi| \cdot k), & \text{if } \Delta\Psi \in [-\pi, \pi) \\ 0, & \text{otherwise} \end{cases}$$

in which N is the normalisation constant. With the help of the free parameter k, it is possible to set how much the particle changes its direction as a result of an interaction. The higher k, the narrower the peak around $\Delta\Psi = 0$. A deflection angle of $\Delta\Psi = 0$ means that the particle is not deflected.
In the following subtasks, add the methods in the file `simulation/detector/exp_angle_dist.py`. To run and test your implementation use the script `exercises/angle_pdf.py` with the command:

```
# (replace <groupname> with the name of your group)
python exercises/angle_pdf.py project_<groupname> -d plots
```

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

a) Calculate the normalisation constant N, in order to create a correct probability density function (`PDF`) with (1). Then implement the PDF method in the given field in the file `simulation/detector/exp_angle_dist.py`.

b) Calculate the cumulative density function (`CDF`) and implement it in the given field.

c) Then calculate the inverse cumulative distribution (`PPF`) and implement it in the given field.

d) Now use the previously implemented `PPF` method to draw random values from the angle distribution by inversion. To do this, implement the `rvs` method in the given field.

Include the overview `PDF` you created in your submission. Interpret the results.

a) We calculate the normalisation constant N to create the correct probability density function.

$$\begin{aligned} 1 &= \int_{-\pi}^{\pi} N \cdot \exp\{-|\Delta\Psi| \cdot k\} \, d\Delta\Psi \\ 1 &= \int_{-\pi}^{0} N \cdot \exp\{\Delta\Psi \cdot k\} \, d\Delta\Psi + \int_{0}^{\pi} N \cdot \exp\{-\Delta\Psi \cdot k\} \, d\Delta\Psi \\ N &= \frac{k}{2 \cdot (1 - \exp\{-\pi \cdot k\})} \end{aligned}$$

Implementation of the PDF is done in the provided Python script.

b) Then we calculate the cumulative density distribution with the following equation:

$$\mathrm{CDF} = \int_{-\infty}^{\Delta\Psi} \mathrm{PDF} \, d\Delta\Psi$$

Case differentiation:

$$\Delta\Psi \leq 0:$$

$$\begin{aligned}\mathrm{CDF} &= N \cdot \int_{-\pi}^{\Delta\Psi} \exp\{\Delta\Psi' k\}\, d\Delta\Psi' \\ &= \frac{N}{k} \cdot (\exp\{\Delta\Psi k\} - \exp\{-\pi k\})\end{aligned}$$

$$\mathrm{CDF}(-\pi) = 0 \text{ and } \mathrm{CDF} = \frac{1}{2}$$

$$\Delta\Psi > 0:$$

$$\begin{aligned}\mathrm{CDF} &= \frac{1}{2} + N \cdot \int_{0}^{\Delta\Psi} N \cdot \exp\{-\Delta\Psi' k\}\, d\Delta\Psi' \\ &= \frac{N}{k} \cdot (2 - \exp\{-\pi k\} - \exp\{-\Delta\Psi k\})\end{aligned}$$

$$\mathrm{CDF}(\Delta\Psi) = \frac{1}{2(1-\exp(-\pi k)} \begin{cases} (\exp(\Delta\Psi \cdot k) - \exp(-\pi \cdot k)), & \text{if } \Delta\Psi \leq 0 \\ (2 - \exp(-\pi \cdot k) - \exp(-\Delta\Psi \cdot k), & \text{if } \Delta\Psi > 0 \end{cases}$$

The implementation of the CDF is also done in the provided Python script.

c) In the following we calculate the inverse cumulative distribution PPF with

$$\mathrm{PPF}(q) = \mathrm{CDF}^{-1}(q).$$

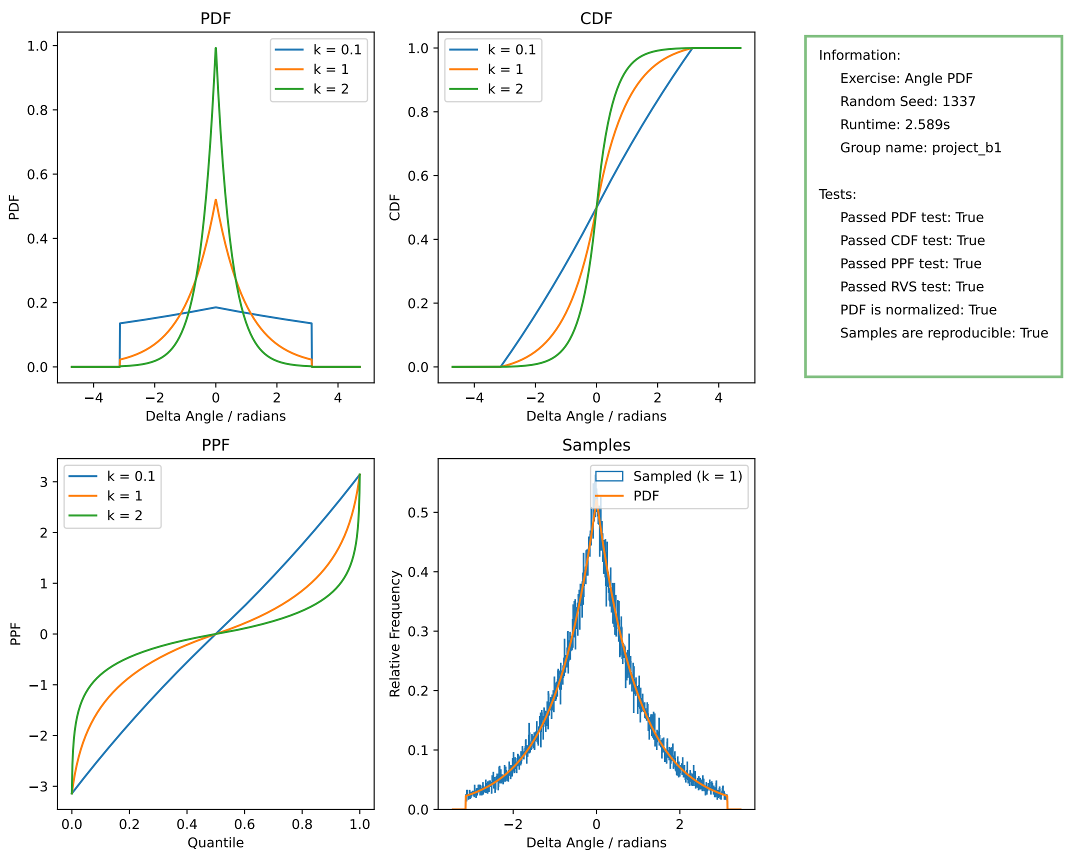
$$\Delta\Psi \leq 0 \text{ and } 0 < u \leq \frac{1}{2}:$$

$$\begin{aligned}u = \mathrm{CDF}(\Delta\Psi) &= \frac{N}{k} \cdot (\exp(\Delta\Psi \cdot k) - \exp(-\pi \cdot k)) \\ \Delta\Psi &= \frac{1}{k} \cdot \ln\left(\frac{k}{N} \cdot u + \exp(-\pi \cdot k)\right)\end{aligned}$$

$$\Delta\Psi > 0 \text{ and } \frac{1}{2} < u < 1:$$

$$\begin{aligned}u = \mathrm{CDF}(\Delta\Psi) &= \frac{N}{k}(2 - \exp(-\pi \cdot k) - \exp(-\Delta\Psi \cdot k)) \\ \Delta\Psi &= -\frac{1}{k} \ln\left(2 - \frac{k}{N} \cdot u - \exp(-\pi \cdot k)\right)\end{aligned}$$

d) To generate random angle distributions the rvs method was implemented in the python script. The used random generator is from the method random state. The generated angles are passed to the ppf method. The figure in 1.1 shows the results from this exercise. The expected PDF, CDF and PPF can be seen. The random generated angles from the rvs method have the same curve like the PDF curve with k=1. Therefore a correct implementation is assumed.



```
Information:
    Exercise: Angle PDF
    Random Seed: 1337
    Runtime: 2.589s
    Group name: project_b1

Tests:
    Passed PDF test: True
    Passed CDF test: True
    Passed PPF test: True
    Passed RVS test: True
    PDF is normalized: True
    Samples are reproducible: True
```

Figure 1.1: CDF, PDF, PPF

E10: Multiple scattering

Task:

In this task, a particle with multiple scattering in $2\mathrm{D}$ is to be implemented. The scattering processes are completely elastic so that the particle does not lose any energy at the points of interaction and only changes its direction. At an absorption point, however, the particle loses and deposits its energy completely. Implement the `propagate()`-method in the file `simulation/particle/multiple_scattering.py` in the given field. The `propagate()` method of a particle propagates a particle and creates energy losses at the interaction points, which are then returned as a list. The energy loss consists of a tuple of:

$$(\text{x-position}, y\text{-position, deposited energy } E_{\text{dep}}\,,\ \text{angle direction } \Psi),$$

where Ψ is the direction of the particle at the interaction point **before it has changed its direction**. Your implementation should add an energy loss with deposited energy $E_{\text{dep}} = 0$ for each scattering interaction point. The first added energy loss should be the starting point of the particle (with $E_{\text{dep}} = 0)$ and the last added energy loss is the point at which the particle is absorbed. The deposited energy should be the total energy of the particle at this point. The distance between two scattering points as well as the total propagated distance to absorption are to be exponentially distributed and given by the mean ranges $L_{\text{scattering}}$ and $L_{\text{absorption}}$. The scattering and absorption lengths, $L_{\text{scattering}}$ and $L_{\text{absorption}}$, respectively, are set in the `__init__()` method and are available as member variables via self. `self.scattering_length` and `self.absorption_length`. The same applies to the initial parameters of the particle: `self.x`, `self.y`, `self.energy`, `self.direction`.
Proceed with the simulation as follows:

a) First, draw the distance at which the particle is absorbed. To do this, use the `random_state` instance, which corresponds to the `default_rng()` of `numpy`.

b) Then perform scattering processes in a loop until the particle reaches the absorption point. To perform a scattering process, first draw the distance to the scattering point. Then update the location of the particle. Create an energy loss with the current position and direction. Then use the `rvs` method of the angle distribution `self.angle_distribution` (The angle distribution is implemented in the file `simulation/detector/angle_dist.py`), to draw a deflection angle $\Delta\Psi$. The new direction of the particle is then calculated as

$$\Psi_{\text{new}} = \Psi_{\text{old}} + \Delta\Psi$$

Repeat this process until the particle is absorbed.

c) The resulting distances between two scattering points deviate slightly from the exponential distribution. There are distances shorter than expected. Why does this happen?

To run and test your implementation you can use the script `exercises/mc_multiple_scattering.py` with the command:

```
# (replace <groupname > with the name of your group)
python exercises/mc_multiple_scattering .py project_ <groupname > -d plots
```

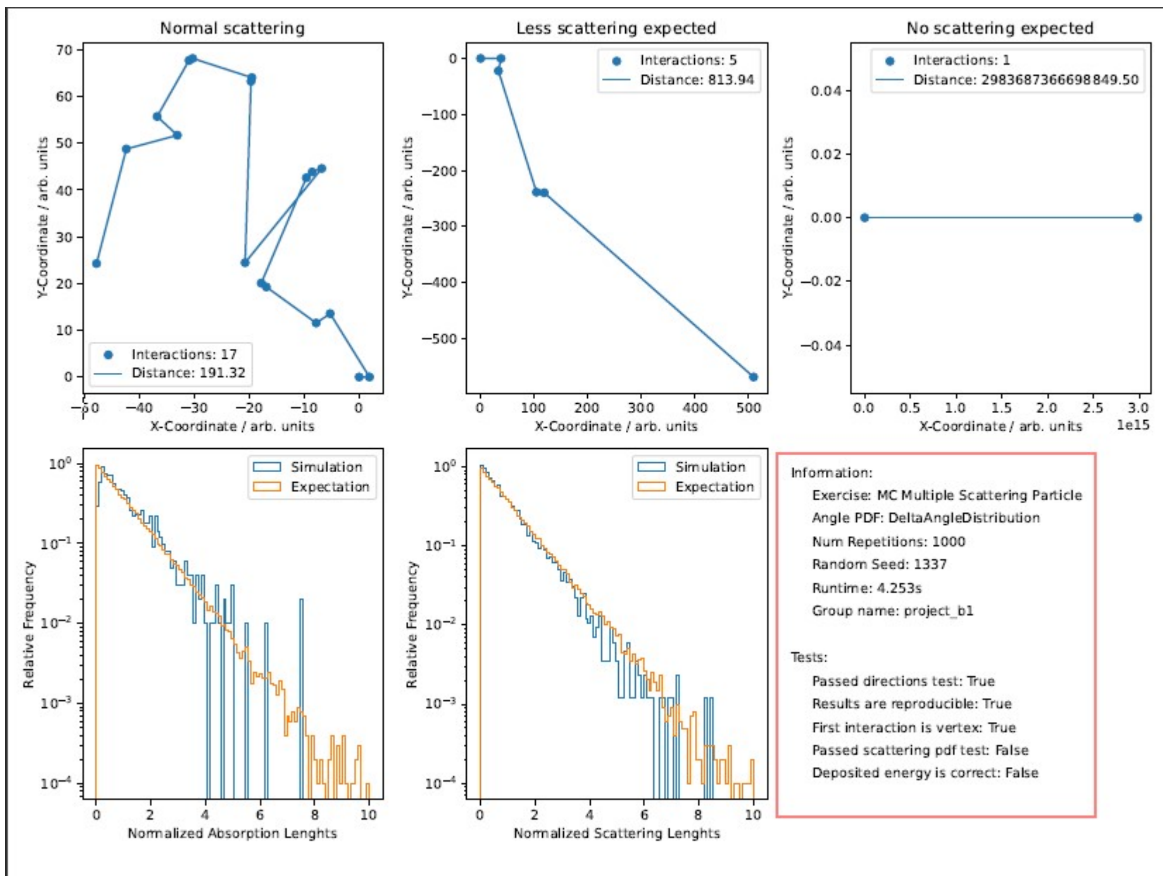
If edited correctly, all tests should pass successfully. Keep in mind, however, that successfully passed tests do not equate to everything being correct!

Include the overview PDF you created in your submission. Interpret the results.
For those interested: Instead of the default angular distribution, you can also test your previously implemented angular distribution via the flag -k <parameter-K>. Furthermore, it is possible to change the random seed via -s <Random Seed>. Also have a look at other options via -h e l p:

```
python exercises/mc_multiple_scattering .py --help
```

a) We add a line to draw the absorption distance in the propagate function.

b) Our implementation for the particle path is added in the python script.



Information:
Exercise: MC Multiple Scattering Particle
Angle PDF: DeltaAngleDistribution
Num Repetitions: 1000
Random Seed: 1337
Runtime: 4.253s
Group name: project_b1

Tests:
Passed directions test: True
Results are reproducible: True
First interaction is vertex: True
Passed scattering pdf test: False
Deposited energy is correct: False

Figure 1.2: The Simulation of the particle path.

c) The observed distances between two scattering points might occasionally be shorter than expected from the exponential distribution. This discrepancy arises due to our implementation. In the condition check, if the scattering point is greater than the difference between the absorption point and the current particle distance, we redefine the scattering point to be this difference. Therefore, we effectively force the scattering length to sometimes be shorter, causing deviations from the ideal exponential distribution.