Simulation based on Monte-Carlo methods

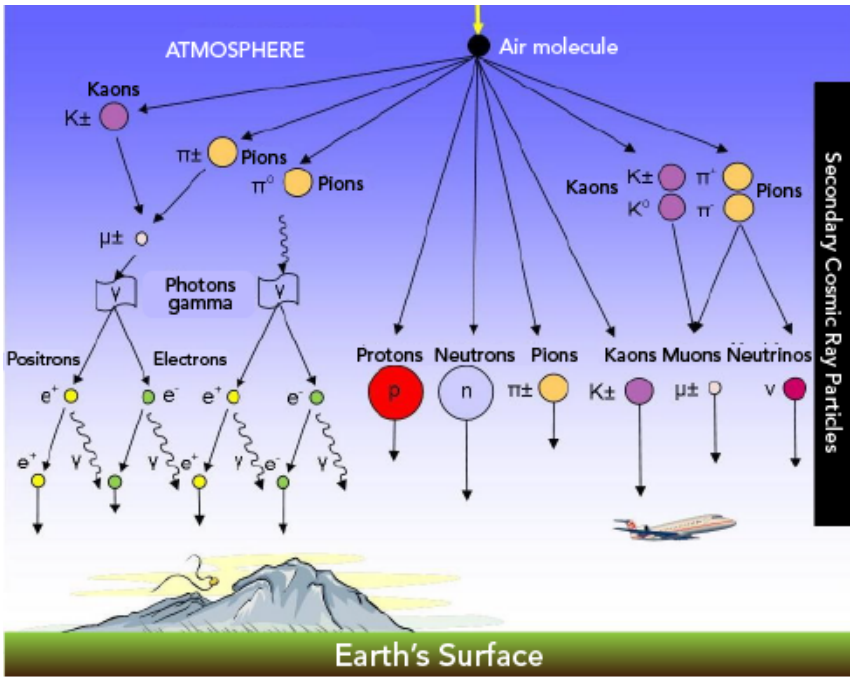
Mohamed-Rami Ahmane, Sacha Rejai

Faculté de Physique et Ingénierie
Université de Strasbourg

24^{th} May 2023

Motivation

Cosmic radiation is a stream of particles that propagates through space and is generated by extraterrestrial sources, such as supernovas and black holes. The detection of these particles is an important area of research in particle physics, as it provides a better understanding of the composition of the universe and the physical processes that take place in it.



Introduction

Project Objective:

- Simulate the passage of cosmic particles through detectors.
- We consider horizontal detectors to simulate the detection of cosmic particles.
- Retrace the path taken by the detected particle. This is similar to what is done in particle accelerator such as CERN.
- Create an impulse spectrometer, which allows to measure the impulse of particles. Determining the impulses can allow us to know the origin of cosmic rays and to study them.

Our choice

We made some choices regarding the project:

- We have adapted the subject to the detection of cosmic particles
- 3D work for more realism, in particular for the detection of cosmic particles
- We work with charged particles that are deflected by a magnetic field
- We neglect the air and the volume of the particles
- We introduced a distribution of incidence angle of the particles

The steps of the program

The simulation consists of three main steps :

- Generation of cosmic particles
  Cosmic particles are generated using a probability distribution based on the experimental data.
- Simulation of their passage through the detectors
- Measurement of the detector response to the particles

Monte Carlo simulation

We use a Monte-Carlo simulation to generate our particles

The basic principle of the Monte Carlo method is to generate a large number of random samples from a specified distribution and use these samples to estimate a numerical value or solve a problem.

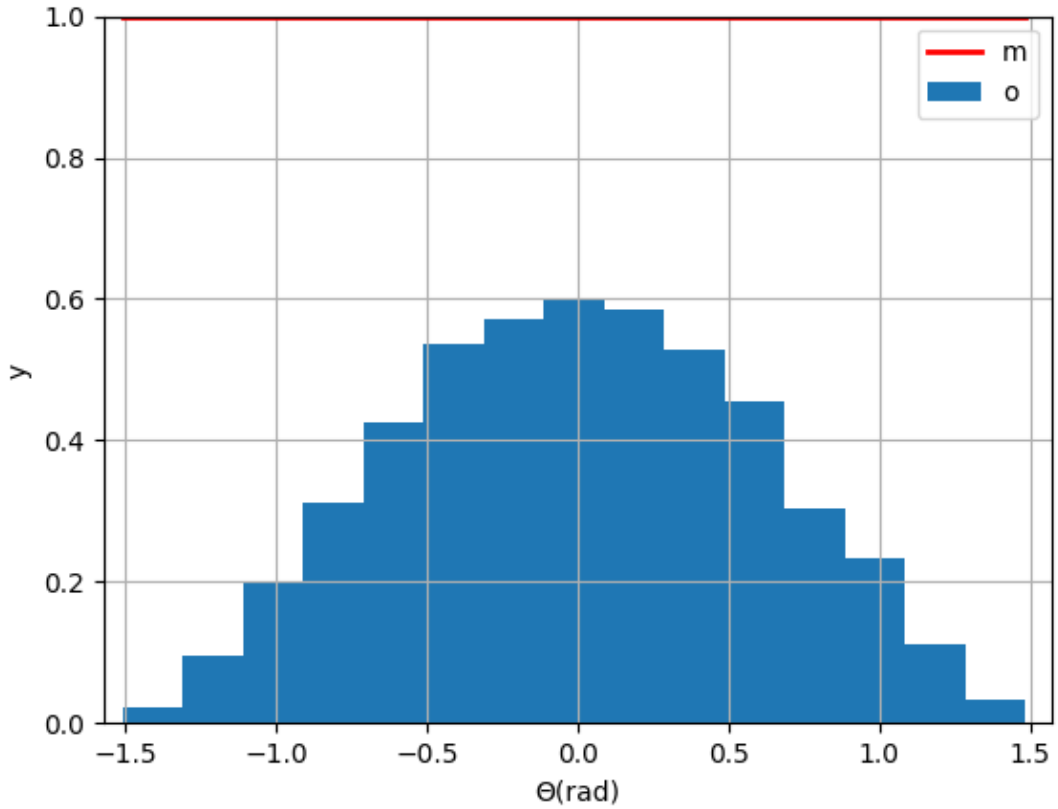
We generate the coordinates, the angle of incidence, the energy of the particles.

```
#FUNCTIONS

def distr_theta(): #monte carlo for theta distribution
    theta=m.pi/2
    y=1
    while y>m.pow(np.cos(theta),1.85) :
        theta = rand.uniform(-m.pi/2,m.pi/2)  #pitch on the bounds here 0 a pi/2
        y = rand.uniform(0,1)
    return theta
```

First step

The code has several classes to represent the particles with their properties and the detectors
We generate the particles with an angular distribution that follows a law in cosine power 1,85



Detector class

Class definition

```
#DEF DETECTOR CLASS

class detecteur():
    def __init__(self):  #creator
        self.px = 0 #variable that stores the pixel according to x that is activated by the particle
        self.py = 0 #variable that stores the pixel according to y that is activated by the particle

    def check(self):  #function that checks the activated pixels in our detector
        print("la particule a activé selon x le pixel numéro :",self.px)
        print("la particule a activé selon y le pixel numéro :",self.py)
```

```
#DEF COORDINATES CLASS

class coor():
    def __init__(self):  #creator
        self.x=0  #list that stores the x-position of the particle
        self.y=0  #list that stores the y-position of the particle
        self.z=0  #list that stores the z-position of the particle

    def check(self):  #function that checks the parameters of our coordinates
        print(self.x)
        print(self.y)
        print(self.z)
```

Particle class

```python
#DEF PARTICULE CLASS

class part():
    def __init__(self):  #creator
        self.x = rand.uniform(-dim[0],dim[0]) #initial position at the source in x
        self.y = rand.uniform(-dim[1],dim[1]) #initial position at the source in y
        self.z = 0 #initial position at the source in z
        self.phi = 0 #phi angle of the particle
        self.theta = 0 #theta angle of the particle
        self.trace = [coor() for i in range(8)] #list that contains the positions of the particle for a number of detecto
        self.E = np.random.normal(2e6,10) #the energy of the particle ditribution normal centered in 2 MeV
        self.qe = -1 #rand.randint(-1,1)
        self.q = False  #if the particle is out in our system
        self.n = 0  # to which plan the share goes out if it goes out

    def check(self):  #function that checks the parameters of our particle
        print("x=",self.x)
        print("y=",self.y)
        print("z=",self.z)
        print("phi=",self.phi)
        print("theta=",self.theta)
        print("q=",self.q)
        print("e=",self.e)
        for i in range(len(self.trace)):
            print("pour le detecteur n° en x",i ,self.trace[i].x)
            print("pour le detecteur n° en y",i ,self.trace[i].y)
            print("pour le detecteur n° en z",i ,self.trace[i].z)

    def restart(self):
        self.E = np.random.normal(2e6,200)
```

Second step

The simulation of particle passage through the detectors :

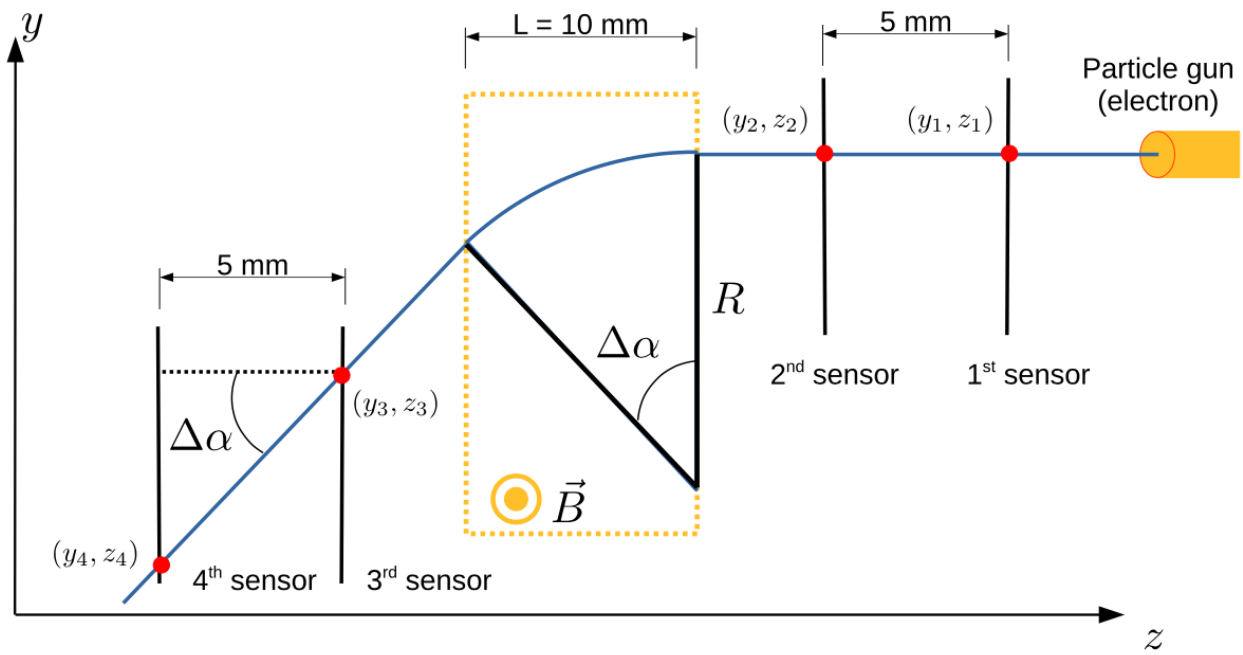
We have defined detectors composed of pixels of dimension $1\mu m$ with an uncertainty of $\frac{1\mu m}{\sqrt{12}}$

When the particle passes through a pixel of the detector, it is activated and we recover the coordinates of the particle on each detector.

Then we use an interpolation to connect these points and recreate the trajectory of the particle

Scheme of the device

Illustration of the deflection induced by the magnetic field :



$$\sigma_\alpha = \sqrt{df_x^2\sigma_x^2 + df_y^2\sigma_y^2}$$

Uncertainties and formulas

The following uncertainty propagation formula is used :

$$\delta f = \sqrt{\left(\frac{\partial f}{\partial x_1}\right)\delta x_1 + \left(\frac{\partial f}{\partial x_2}\right)\delta x_2 + ... + \left(\frac{\partial f}{\partial x_n}\right)\delta x_n}$$

We use this formula to determine the uncertainties on each quantity.

$$\alpha_{deviated} = \theta_{initial} - \theta_{final}$$

$$\sigma_{\alpha_{deviated}} = \sqrt{\delta\theta_i^2 + \delta\theta_f^2}$$

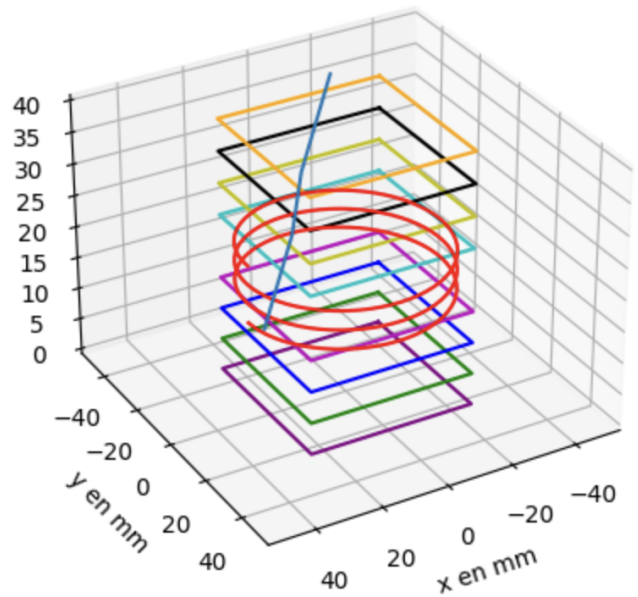
$$p = qBR = \frac{qBL}{sin(\alpha)} = f(\alpha)$$

Schematic of the assembly

Example of particle trajectory through the detectors :

The squares represent the detectors and the red circles represent the coil responsible for the magnetic field

The blue line represents the trajectory of the particle.



Influence of the angle of entry of the particles into the detectors

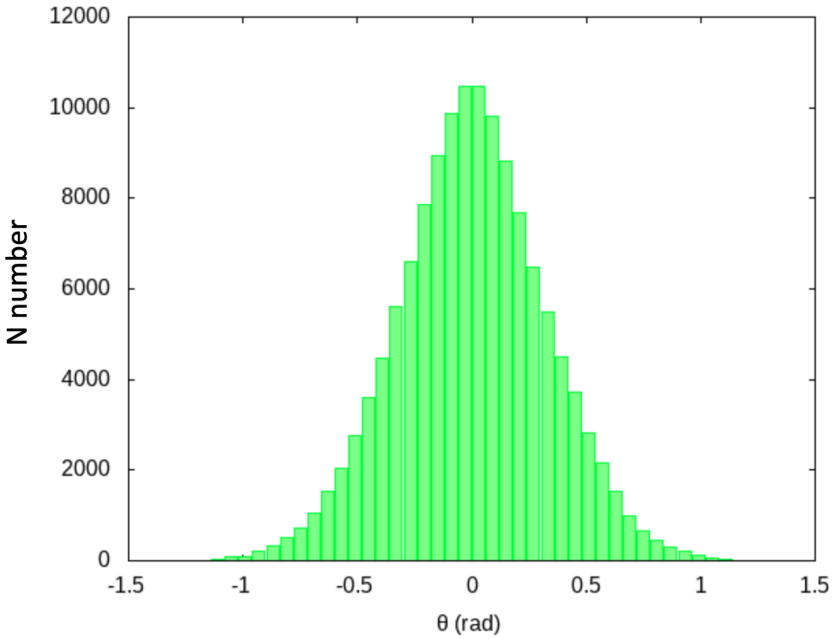


Figure: zenithal angle of 0°

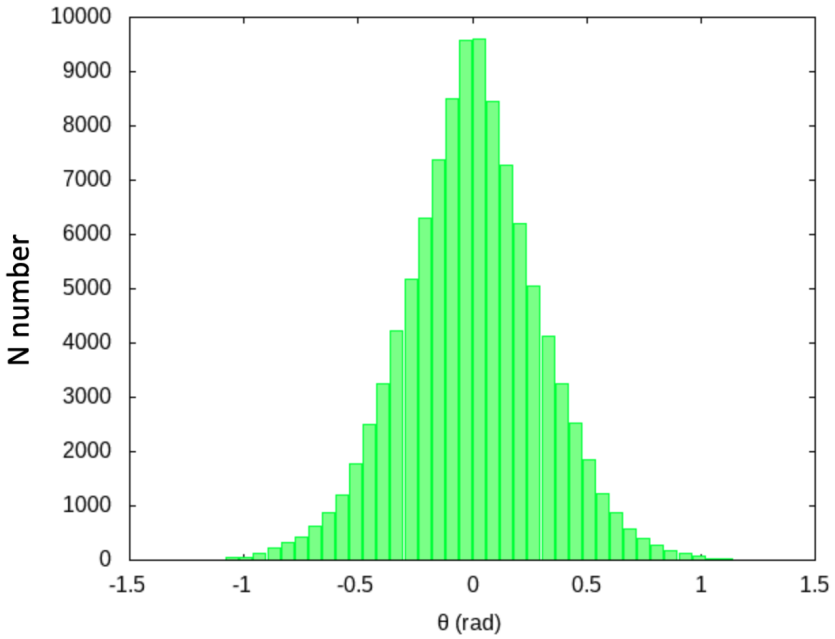


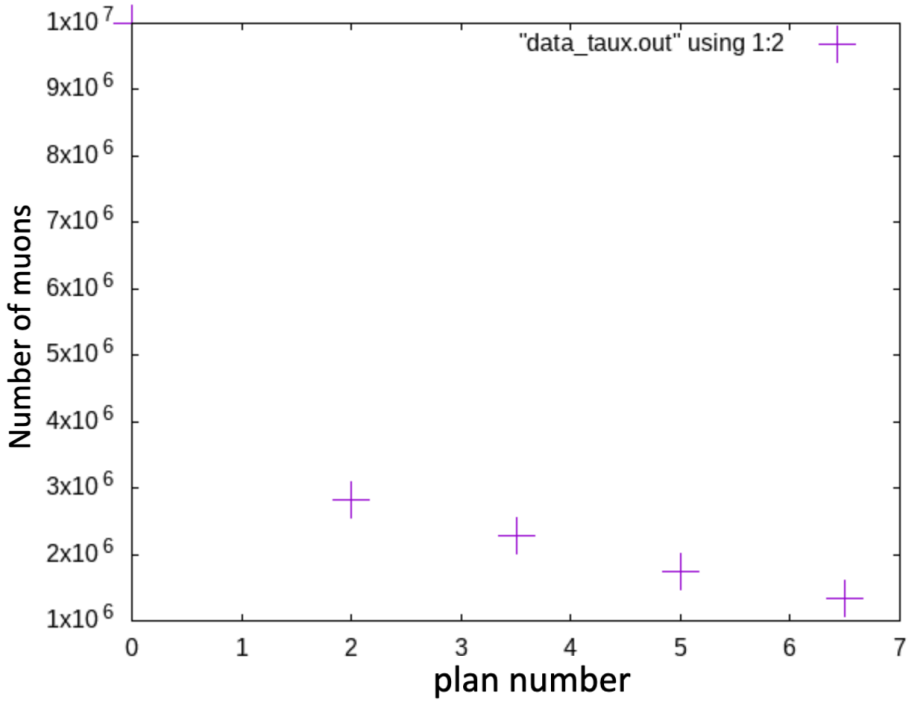
Figure: zenithal angle of 45°

The inclination of the detectors reduces the detection area so only small angles will pass through our detectors.

Count rate

Not all particles pass through all detectors. So we can see the evolution of the number of particles depending on the detector.

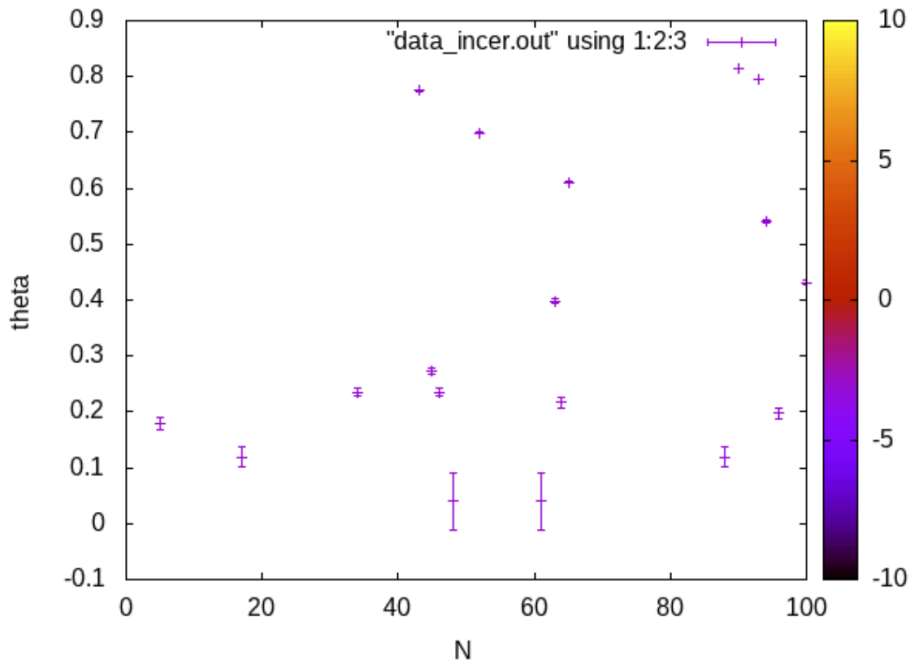
We tighten the angle distribution, so less particles entering and less particles on the last detectors



Angular distribution

Here we have the theta angle distribution as a function of the number of particles.

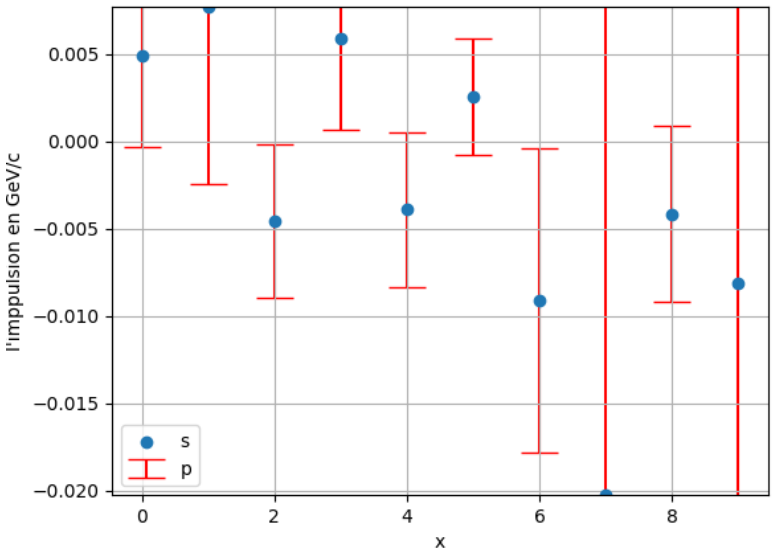
We see that close to 0, the uncertainty is large because we will have a variation of 1 pixel between two detectors so with an uncertainty of about 1 pixel, the uncertainty on the angle is very large.



Pulse spectrometer

We finally deduce the properties of the particle from the trajectory

The deflection of charged particles in a magnetic field between the different detectors is used to find the value of their impulse from the angle of deflection. Thus, we can create a pulse spectrometer



$$p = \frac{qBL}{sin(\alpha)}$$

and uncertainty propagation formula for σ_p

Conclusion

So we have a program that allows us to determine the trajectory and impulse of cosmic particles with an angle of incidence.

Nevertheless, we have adapted here the study to muons and it is necessary to modify parameters from experimental data if we want to study other particles.

Moreover the impulse is deduced from the angle of deflection due to the magnetic field, we cannot determine the impulse for particles of zero charge.

This code can be used to study the formation of muons in the atmosphere.