Simulation of ALICE Project

Coli Simone

November 24th, 2021

1 Introduction

ALICE_Simulation is a computer program that simulates the ALICE experiment conducted at CERN since 1993, which consists of analyzing the collisions occurring between particles at very high energies (1). Those would create different particles following a probability distribution (1).

Particle Type	Probability
π^+	40%
π^-	40%
k^+	5%
k^-	5%
P^+	4.5%
P^-	4.5%
k^*	1%

Table 1: *The table shows the probability of obtaining a specific particle from a collision of high energy particles*

In the simulation, we generated a finite amount of particles resulting from the collision of the flux in the particle accelerator, each with a proper momentum, mass, and resulting energy to maintain the conservation of those properties.The goal of the experiment is to detect the existence of the Kaon 0 (k^*), a very rare particle that decays into either a Pion plus (π^+) and Kaon minus (k^-) or a Pion minus (π^-) and Kaon plus (k^+), after only $5.2 \times 10^{-8} s$. We so stored the data of momentum, energy, charge, and invariant mass of all the particles to detect the presence of differences that could indicate the existence of the Kaon.

The program we implemented stores the information about the particles and generates histograms from which we studied the system.

2 Code Structure

The code's division into different files and folders has the background idea of making it more orderly. There are eight files for the simulation program (one main file: `mainE.cpp`, one libraries file: `library.hpp`, three header files: `ParticleType.hpp`, `ResonanceType.hpp`, `Particle.hpp`, and three source files: `ParticleType.cpp`, `ResonanceType.cpp`, `Particle.cpp`) and one for the data analysis (`analysis.C`). The header

files contain the classes implemented for the proper functioning of the simulation. Meanwhile, the source files contain the implementation of the methods defined in the headers.

ParticleType Class

The class `ParticleType` creates a homonymous type that contains the name, the mass, and the charge of a particular particle, respectively as a `std::string`, a `double`, and an `integer`. This class has five methods, two of which are virtual:

- `GetParticleName()`;
- `GetParticleMass()`;
- `GetParticleCharge()`;
- `GetParticleWidth()` (Virtual);
- `Print()` (Virtual);

All the "GetParticle" functions are getter functions so that return the value of the variable for which called. Meanwhile, the Print function prints a particle's characteristics.

ResonanceType Class

The `ResonanceType` class is a derived class from `Particle- Type` and, in addition to the base class items, it contains the information about the width of the particle as a double. The type defined with the name of this class creates a particle with a width. Contrarily to what happens for a `ParticleType` object, in which the width of the particle is always zero. This class has two methods, both of them are the override of the already existing virtual methods in the base class:

- `GetParticleWidth()`;
- `Print()`;

Particle Class

The class `Particle` is the one that allows creating a particle giving it momentum and making it decay if necessary. It contains the three momentum components (`fPx`, `fPy`, `fPz`), a pointer to an array of ParticleType and its dimension (`fParticleType`, `fMaxNumParticleType`), an index of particle type (`fNParticleType`), and a code proper of each particle type (`fIndex`). This class has several methods, some of which are static, which means they are accessible from the main without defining an object:

- `FindParticle()` (Static);
  (This function returns the value of the code of a particle type)
- `Boost()`;
  (This function Boost the Lorenz vector)
- `AddParticle()` (Static);
  (This function allows to allocate a new particle type in the vector)
- `Decay2body()`;
  (This function decay the particle for which called)

- `ParticleFeatures()` (Static);
  (This fucntion has in memory the characteristics of some particle type)

  Getter Functions:
  (All of this function return the value of the variable for which called)

- `GetIndex()`;
- `GetCharge()`;
- `GetName()`;
- `GetPx()`;
- `GetPy()`;
- `GetPz()`;
- `GetMass()`;
- `GetMomentum()`;
- `GetTotalEnergy()`;
- `GetInvMass()`;
- `GetTrasMomentum()`;

  Setter Functions:
  (All of this functions set the value of the variable for which called)

- `SetIndex()`;
- `SetMomentum()`;

  Printer Functions:
  (All of this functions give a standard output of the information for which called)

- `PrintTable()` (Static);
- `PrintParticle()`;

3 Generation

Work Citacion

1. Alice Experiment. CERTN. `https://home.cern/science/experiments/alice`. November 29^{th}, 2021.