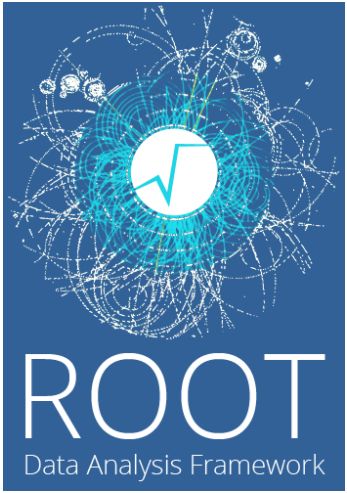
Знайомство з ROOT: реконструкція інваріантної маси частинок

Олександр Зенаєв

ROOT – це програмний пакет для зберігання, аналізу та візуалізації великих обсягів даних, який широко використовується у фізиці високих енергій

- відкрите програмне забезпечення (фреймворк), розробляється у CERN (Європейській організації з ядерних досліджень) 1995–...
- написаний на C++, має інтерактивний інтерпретатор, підтримує інтеграцію з іншими мовами, такими як Python (через інтерфейс PyROOT)
- основне застосування:
  - зберігання та зчитування великих обсягів даних
  - статистичний аналіз даних
  - візуалізація
- підтримує паралельні обчислення
- має широку підтримку наукової спільноти (документація, приклади, форум https://root-forum.cern.ch/)



Початок роботи з ROOT: інтерпретатор, компіляція

ROOT інтерпретатор	C++ компілятор із ROOT бібліотекою

`// hello.cpp`
 	
`void hello() {`
`  printf("Hello world\n");`
`  TH1F hist("hist", "My histo", 300, 0., 3.);`
`  hist.Fill(1.2);`
`  hist.Print();`
`}`
	`// hello_main.cpp`
	
	`#include <cstdio>`
	`#include <TH1F.h>`
	`int main() {`
	`  printf("Hello world\n");`
	`  TH1F hist("hist", "My histo", 300, 0., 3.);`
	`  hist.Fill(1.2);`
	`  hist.Print();`
	`  return 0;`
	`}`

`root -l -q hello.cpp`	`g++ -o hello_main` `` `root-config --cflags --libs` `` `hello_main.cpp` `./hello_main`
Processing hello.cxx... Hello world TH1.Print Name = hist, Entries= 1, Total sum= 1	Hello world TH1.Print Name = hist, Entries= 1, Total sum= 1

Початок роботи з ROOT: PyROOT

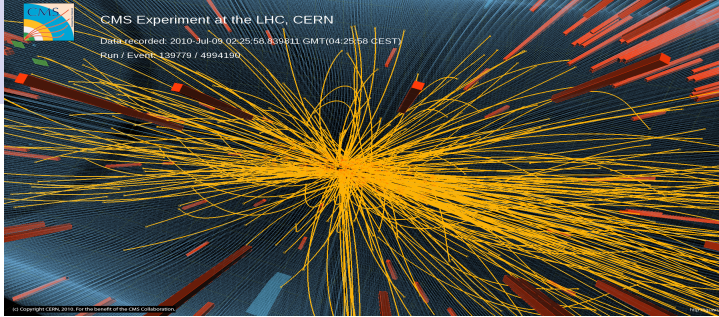
- PyROOT: Python інтерфейс до ROOT
  - дуже зручний для прототипування (дослідницький код: швидкість розробки, внесення змін)
  - багато корисних Python модулів (numpy, scipy, pandas, matplotlib ...) + документація і дуже велика спільнота

```
# hello.py

import ROOT

if __name__ == '__main__':
  print("Hello world")
  hist = ROOT.TH1F("hist", "My histo", 300, 0., 3.)
  hist.Fill(1.2)
  hist.Print()
```

```
python hello.py
```

```
Hello world
TH1.Print Name  = hist, Entries= 1, Total sum= 1
```

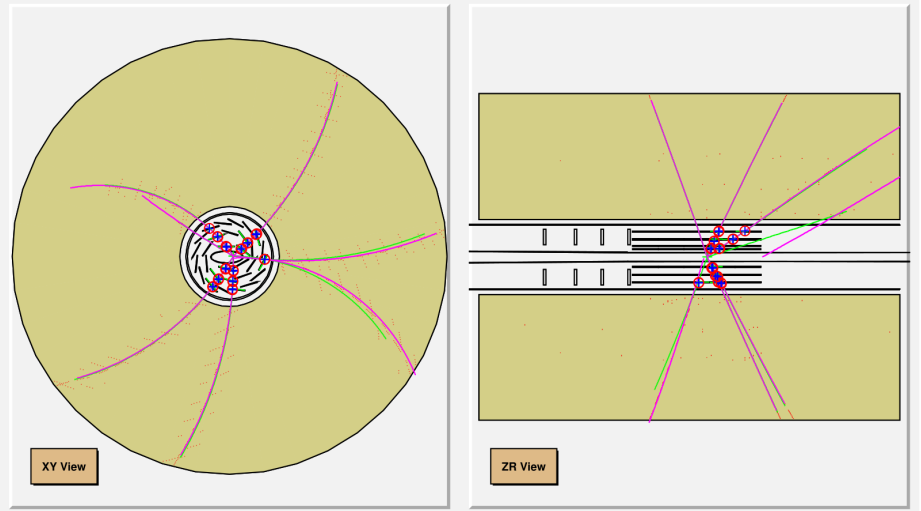
Реконструкція інваріантної маси (системи) частинок



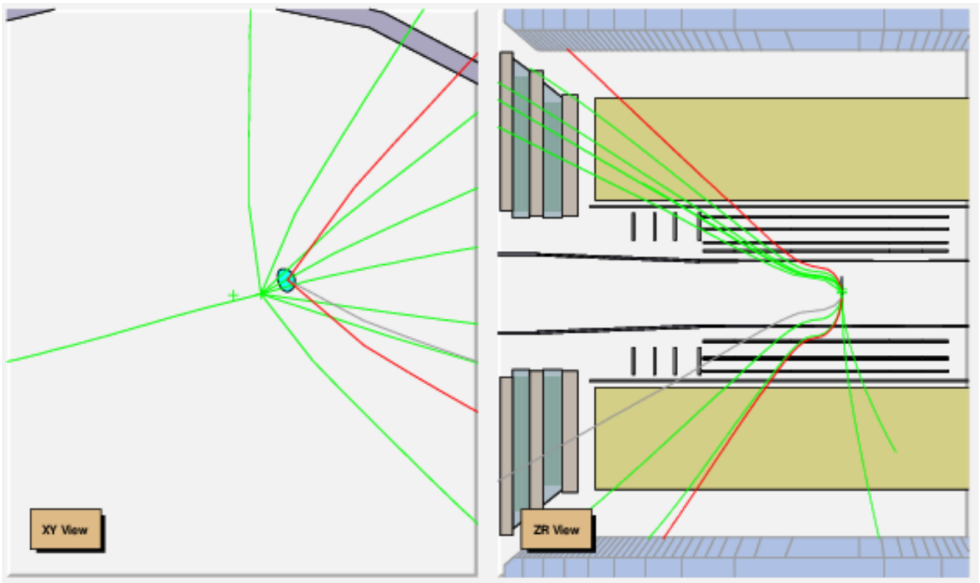
- Реконструкція інваріантної маси – важлива складова аналізу даних експериментів у HEP
  - дослідження властивостей частинок (вимірювання маси, ширини розпаду тощо)
  - вимірювання перерізів народження частинок
- Інваріантна маса $M = \sqrt{E^2 - P^2} = \sqrt{(E_1 + E_2)^2 - (P_1 + P_2)^2}$ не змінюється при перетворенні системи відліку
- Експериментальні дані складаються з числених подій (одне зіткнення частинок)
- Дані про кожну подію містять енергії та/чи імпульси зареєстрованих детектором (стабільних чи довгоживучих) частинок
  - трекова система в магнітному полі рекенструює треки (tracks) (імпульси для заряджених частинок)
  - система ідентифікації частинок (Particle Identification, PID)свизначає тип частинки (маса)
    → це дозволяє реконструювати (тобто обчислити енергію та імпульс) батьківської нестабільної частинки, яку хочемо досліджувати і яка не може бути зареєстрована детектором
- Аналіз даних потребує фільтрацію шуму (фон), корекцію детекторних ефектів тощо
  - зазвичай народжується багато нестабільних частинок → десятки, сотні чи навіть тисячі треків у кожній події → тисячі чи мільони комбінацій (комбінаторний фон)

Реконструкція інваріантної маси $D^+ \to K^-\pi^+\pi^+$

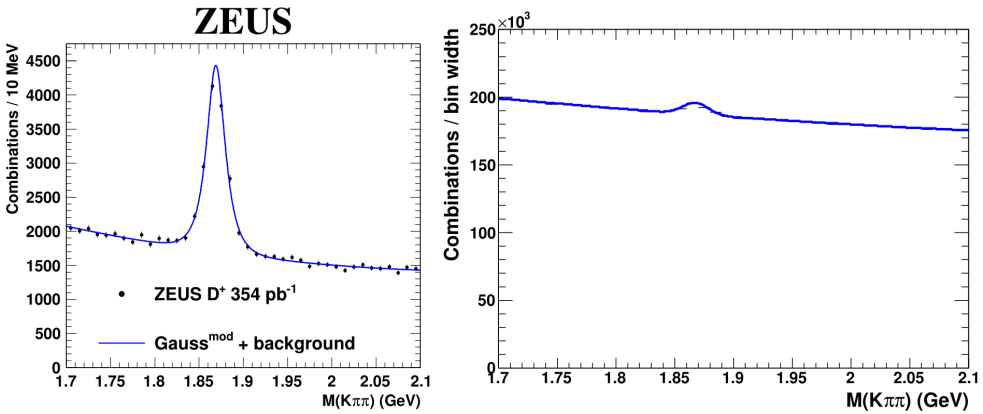
ZEUS Coll., H. Abramowicz et al., "Measurement of $D^{\pm}$ production in deep inelastic ep scattering with the ZEUS detector at HERA", JHEP05 (2013) 023 [arXiv:1302.5058]



Реконструкція інваріантної маси $D^+ \to K^-\pi^+\pi^+$



Реконструкція інваріантної маси $D^+ \to K^-\pi^+\pi^+$



- D^+ складається з $c\bar{d}$ кварків
- $M(D^+) = 1869.66 \pm 0.05$ ГеВ [Particle Data Group (PDG) https://pdg.lbl.gov/]
- $\tau(D^+) = (1.033 \pm 0.005) \times 10^{-12}$ с: розпадається за рахунок слабкої взаємодії
- Маючи імпульс $\sim$ ГеВ, D^+ мезон пролітає $L = \frac{p}{M} c\tau \sim$ декілька мм $\to$ його точка розпаду (вершина) може бути реконструйована $\to$ це дозволяє суттєво зменшити комбінаторний фон

Практичне заняття

- Згенерувати $N = 1000$ подій
- У кожній події згенерувати дві частинки: K^0 і D^0 мезони
- Згенерувати розпади $K^0 \to \pi^+\pi^-$, $D^0 \to \pi^+\pi^-$
- Накласти ефект роздільної здатності детектора
- Реконструювати інваріанту масу кожної пари дочірних частинок
- Зафітувати (fit: апроксимація, регресія) розподіл інваріантної маси, визначити маси і кількість реконструйованих частинок
- Намалювати графік інваріантної маси
- Зберегти згенеровані дані в ROOT файл і зчитати їх із нього
- (Домашнє) завдання: переписати код для виконання завдання на C++

Github:
https://github.com/zenaiev/hep/blob/main/invmass/invmass.py

Google Colab:
https://colab.research.google.com/github/zenaiev/hep/blob/main/invmass/invmass.ipynb