Detector simulations

Lab group 3:
Giuseppe A. Brischetto, Tomas Klinavičius,
Majd Ghrear, Jose A. Pavon-Rodriguez

Table of contents

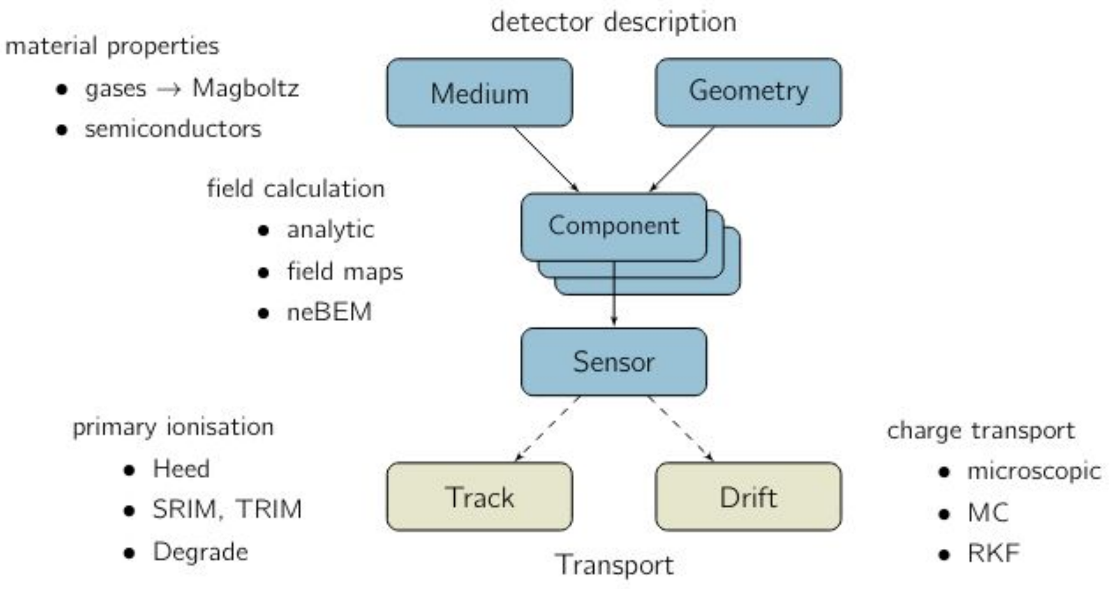
- Description of the simulation toolkit and computing environment
- Exercise 2: Gas ionization and cluster evaluation
- Energy loss evaluation of muons
- Conclusions and feedback

Garfield++

Garfield++ is an open source toolkit for detailed simulation of charge transport and signals in particle detectors.

It can track electrons, simulate their drift, amplification and signal creation.

It also provides analytic solutions for electric field in 2D geometries.

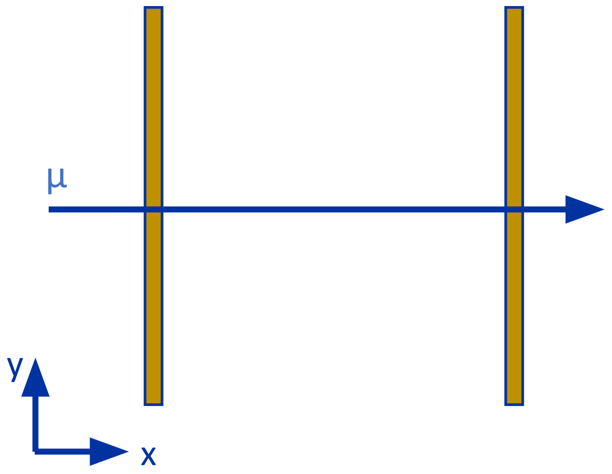


https://garfieldpp.web.cern.ch/garfieldpp/

Exercise 2: Gas ionization by simulation of a charged particle traversing it

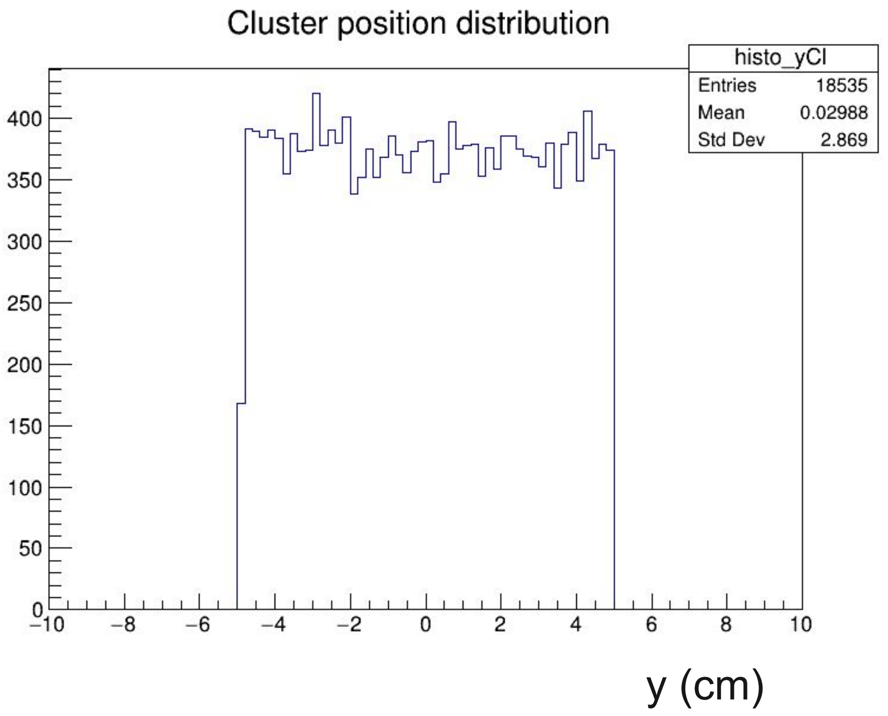
- Gas mixture: Ar (70%) – CO_2 (30%)
- Electric field: 2 plane electrodes at 10 cm distance with ΔV = 1000 V
- Particle type: Muon (μ^-)
- Momentum: 170 GeV/c

```
# Set up the gas medium.
gas = ROOT.Garfield.MediumMagboltz()
gas.SetTemperature(293.15)
gas.SetPressure(740.)
gas.SetComposition("ar", 70., "co2", 30.)
gas.EnableDrift()
```



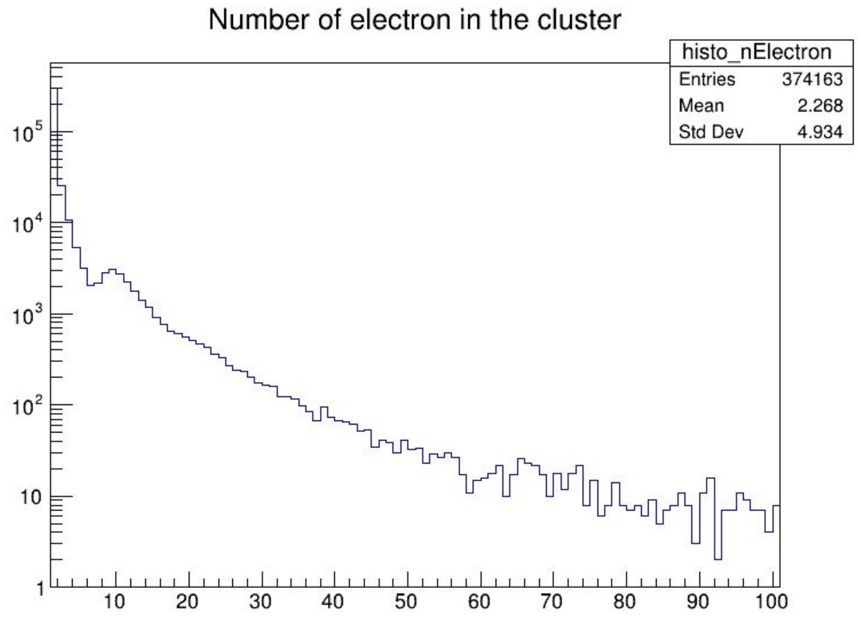
Gas ionization and cluster evaluation

```
#Define a muon of 170GeV
track = ROOT.Garfield.TrackHeed()
track.SetParticle("muon");
track.SetEnergy(170.e9);
track.SetSensor(sensor);
#Define the impact point and direction
x0 = 0.0
y0 = -4.9
z0 = 0.0
t0 = 0.0
dx0 = 0.0
dy0 = 1.000001
dz0 = 0.0
#Get the primary ionization cluster
track.NewTrack(x0, y0, z0, t0, dx0, dy0,
dz0)
cluster = track.GetClusters()
```



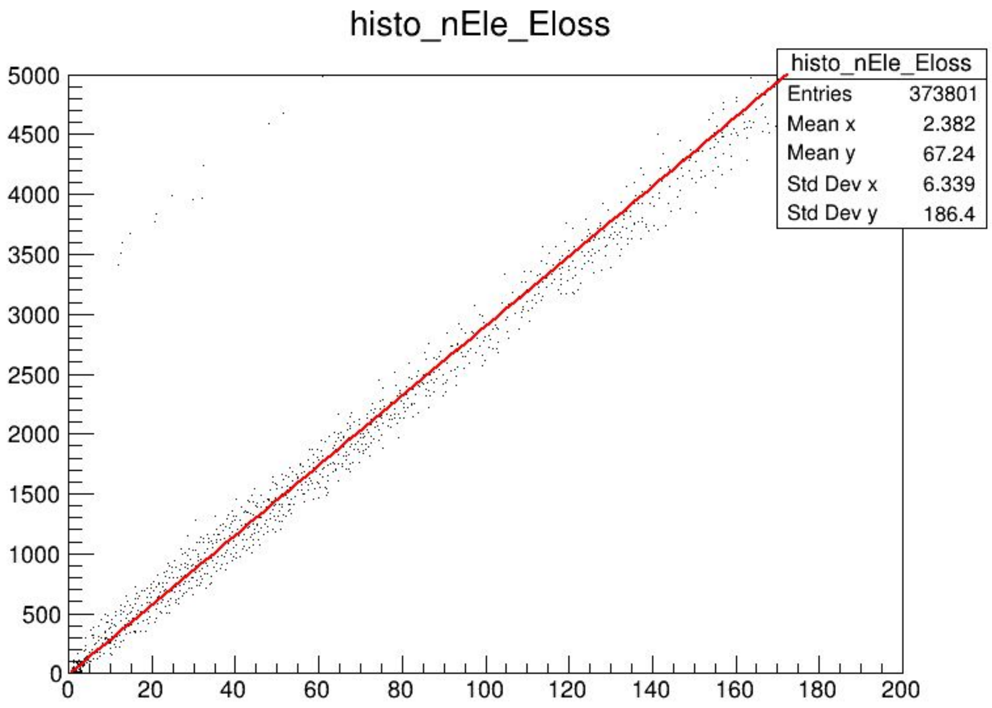
Gas ionization and cluster evaluation

```
for i in range(1000):
 track.NewTrack(x0, y0, z0, dx0, dy0, dz0, 0)
 for cluster in track.GetClusters():
   histo_nElectron.Fill(cluster.electrons.size())
```



Gas ionization and cluster evaluation

```
#Compare the energy loss and the number of
electron per cluster
histo_nEle_Eloss =
ROOT.TH2F("histo_nEle_Eloss","histo_nEle_Eloss",
200,0,200,100,0,5000)
for i in range(1000):
 track.NewTrack(x0, y0, z0, dx0, dy0, dz0, 0)
 for cluster in track.GetClusters():
   histo_nEle_Eloss.Fill(
      cluster.electrons.size(),cluster.energy)
```



The Bethe – Bloch formula

$$-\left\langle\frac{dE}{dx}\right\rangle = \frac{4\pi}{m_e c^2}\cdot\frac{nz^2}{\beta^2}\cdot\left(\frac{e^2}{4\pi\varepsilon_0}\right)^2\left[\ln\left(\frac{2m_e c^2\beta^2}{\mathrm{I}-(1-\beta^2)}\right)-\beta^2\right]$$

c – speed of light in vacuum

m_e – electron mass

e – electron charge

z – charge of the impacting particle (in multiples of electron charge)

$\beta = \frac{v}{c}$ – ratio of the velocity of the impacting particle and the speed of light in vacuum

I – mean excitation energy of the material

ρ – density of the material

Z – atomic number of the material

A – relative atomic mass of the material

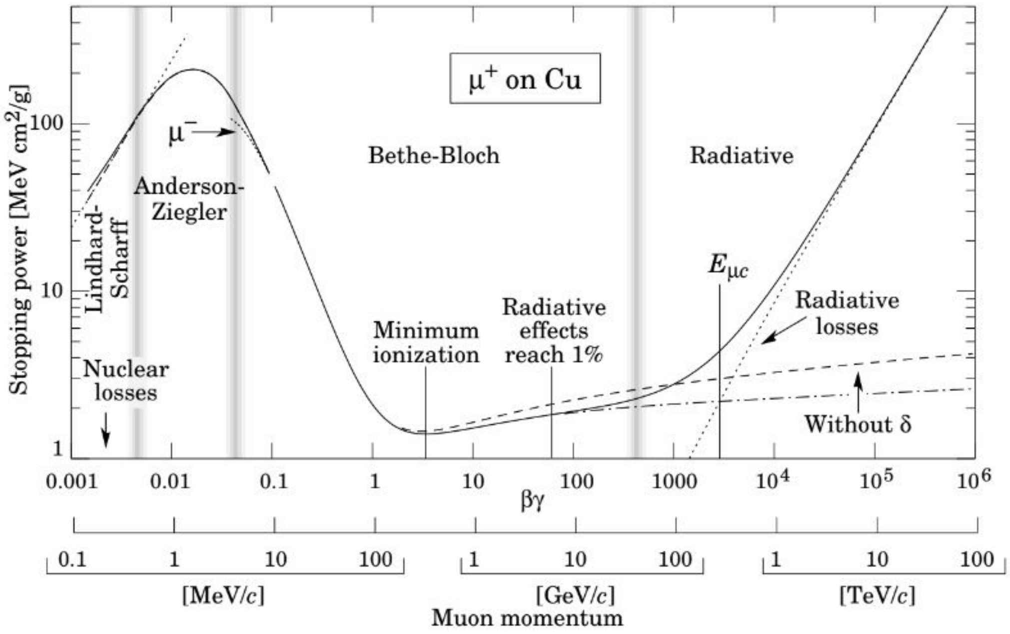
N_A – Avogadro constant

M_u – Molar mass constant

$n = \frac{N_A Z\rho}{AM_u}$ – electron density of the material

The Bethe – Bloch formula

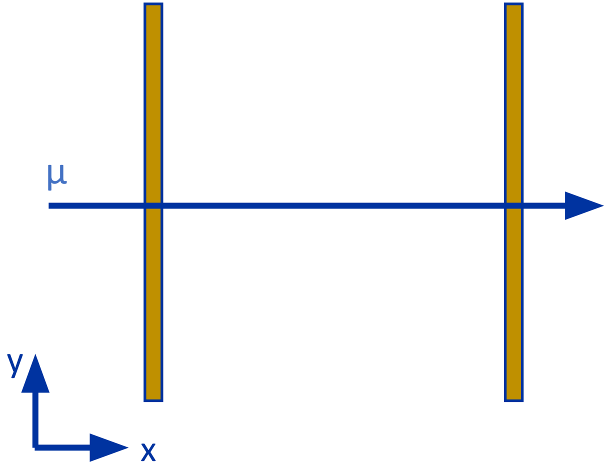
$$-\left\langle \frac{dE}{dx} \right\rangle = \frac{4\pi}{m_e c^2} \cdot \frac{nz^2}{\beta^2} \cdot \left(\frac{e^2}{4\pi\varepsilon_0} \right)^2 \left[\ln \left(\frac{2m_e c^2 \beta^2}{I - (1-\beta^2)} \right) - \beta^2 \right]$$



https://pdg.lbl.gov/

The simulation setup

- Gas mixture: Ar (70%) – CO_2 (30%)
- Electric field: 2 plane electrodes at 10 cm distance with ΔV = 5000 V
- Particle type: Muon (μ^-)
- Momentum: 6 MeV/c - 10 GeV/c



Calculation the Bethe-Bloch curve

```
histo_momentum_Eloss = ROOT.TH2F("histo_momentum_Eloss","Bethe-Blochformula",100,0.01,10,100,1,100)
histo_momentum_Eloss.GetXaxis().SetTitle("p (GeV/c)")
histo_momentum_Eloss.GetYaxis().SetTitle("dE/dx (keV/cm)")

tot_eloss = 0
for i in range(1000):
    track.SetMomentum(6.e7+i*1.e7)
    for j in range(10):
        track.NewTrack(x0, y0, z0, dx0, dy0, dz0, 0)
        for cluster in track.GetClusters():
            tot_eloss = tot_eloss + cluster.energy

    tot_eloss = tot_eloss/1.e4
    histo_momentum_Eloss.Fill(track.GetMomentum()/1.e9,tot_eloss)

canvas20 = ROOT.TCanvas("canvas20","",800,600)
canvas20.cd()
histo_momentum_Eloss.Draw()
canvas20.Draw()
```

Calculation the Bethe-Bloch curve

```
histo_momentum_Eloss = ROOT.TH2F("histo_momentum_Eloss","Bethe-Blochformula",100,0.01,10,100,1,100)
histo_momentum_Eloss.GetXaxis().SetTitle("p (GeV/c)") # Describe horizontal (x) axis in histogram
histo_momentum_Eloss.GetYaxis().SetTitle("dE/dx (keV/cm)") # Describe vertical (y) axis in histogram
```

```
tot_eloss = 0
for i in range(1000):
    track.SetMomentum(6.e7+i*1.e7)
    for j in range(10):
        track.NewTrack(x0, y0, z0, dx0, dy0, dz0, 0)
        for cluster in track.GetClusters():
            tot_eloss = tot_eloss + cluster.energy

    tot_eloss = tot_eloss/1.e4
    histo_momentum_Eloss.Fill(track.GetMomentum()/1.e9,tot_eloss)

canvas20 = ROOT.TCanvas("canvas20","",800,600)
canvas20.cd()
histo_momentum_Eloss.Draw()
canvas20.Draw()
```

Calculation the Bethe-Bloch curve

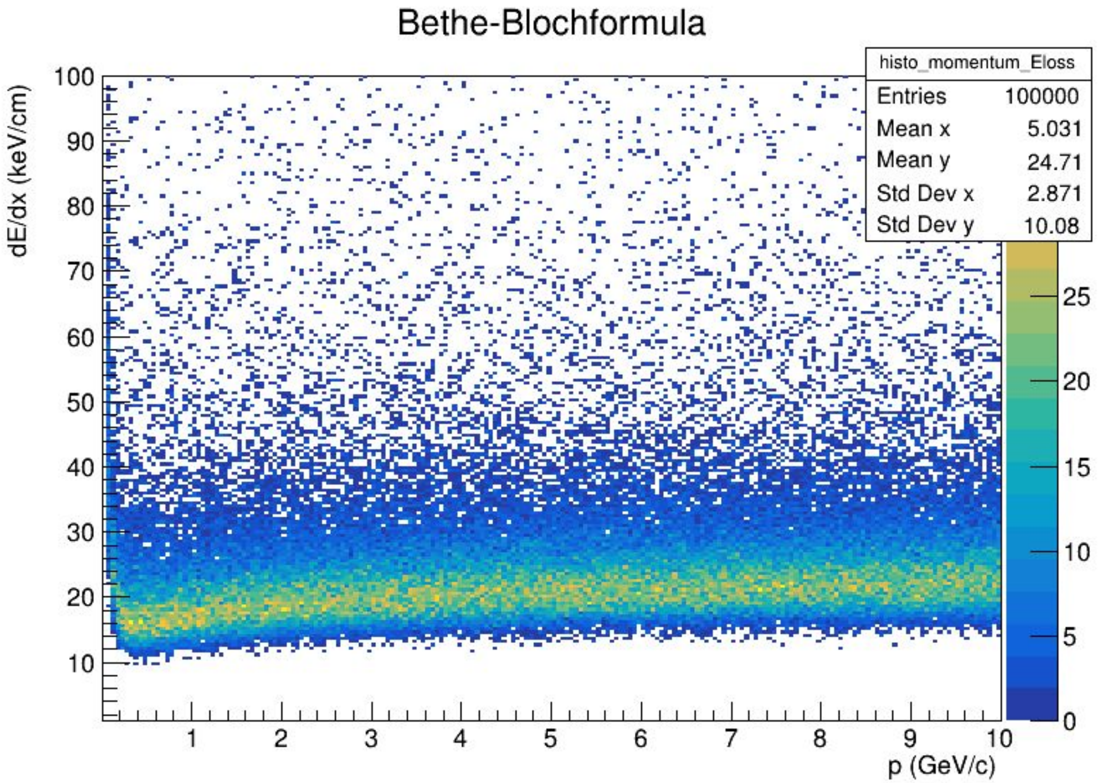
```
histo_momentum_Eloss = ROOT.TH2F("histo_momentum_Eloss","Bethe-Blochformula",100,0.01,10,100,1,100)
histo_momentum_Eloss.GetXaxis().SetTitle("p (GeV/c)")
histo_momentum_Eloss.GetYaxis().SetTitle("dE/dx (keV/cm)")
```

```
tot_eloss = 0 # Initialize total  energy loss as 0 for summation
for i in range(1000): # Iterate the entire procedure 1000 times
    track.SetMomentum(6.e7+i*1.e7) # Set momentum of i-th track in unit of eV
    for j in range(10): # Generate 10 tracks, all with the same initial momentum
        track.NewTrack(x0, y0, z0, dx0, dy0, dz0, 0) # Generate j-th track
        for cluster in track.GetClusters(): # For-loop on every cluster produced by the j-th track
            tot_eloss = tot_eloss + cluster.energy # Add up energy losses for every cluster

    tot_eloss = tot_eloss/1.e4 # Convert the energy loss in keV/cm
    histo_momentum_Eloss.Fill(track.GetMomentum()/1.e9,tot_eloss) # Fill the histogram with the momentum of
the muon (in GeV) and the energy loss
```

```
canvas20 = ROOT.TCanvas("canvas20","",800,600)
canvas20.cd()
histo_momentum_Eloss.Draw()
canvas20.Draw()
```

Energy loss of muons



Conclusions and feedback

Garfield proves to be a very useful tool for detailed simulation of gaseous detectors. Moreover, its recent upgrade to C++, including ROOT classes and allowing a coupling with GEANT4 allow more elaborated and realistic simulations.

The framework used in this lab to introduce Garfield has been very practical, using a python wrap and the use of jupyter.ntbs and Google Collab, avoiding the usual problems of a local installation and env setup, thus allowing a hands-on experience from the start.

The introduction into Garfield++ might be a bit step if no previous experience with object oriented programming.

Thank you for your attention

Do you have any questions?