M. Shchedrolosiev, MSc Thesis
https://agenda.linearcollider.org/event/8107/attachments/34048/55608/main_eng.pdf

Basic preperation

In [709…
```
from ROOT import TFile, gROOT, TGraphErrors, TH1F, TGraph
from ROOT import TCanvas, TPad, TF1, gStyle, TH2F, TLegend
from ROOT import TVector3, TMarker, TLine, TSpline3

from array import array
import numpy as np
import math
import re, os
from os import listdir
import argparse
```

In [710…
```
# to generate merged histograms

# to hadd root files

#!hadd electrons1BX.root /home/gleb/Documents/JUPITER/luxe_signal_165gev_al2_cv


#!hadd 1.57e9_electrons1BX.root /afs/desy.de/group/flc/luxe/bgdata/bh/luxe_sign

#!hadd 1.48e9_electrons1BX.root /afs/desy.de/group/flc/luxe/background/ob/lx_1a
```

In [711…
```
base_dir = './' # change to your directory or just './' if you did previus step
# filename = f'{base_dir}1.48e9_electrons1BX.root'
filename = f'{base_dir}luxe_signal_165gev_al2_cv7emstd_1umstep_tv31_hv1_15_t0.r

background_file = TFile.Open(filename)
```

In [712…
```
background_file.ls()
```

```
TFile**         ./luxe_signal_165gev_al2_cv7emstd_1umstep_tv31_hv1_15_t0.roo
t
 TFile*         ./luxe_signal_165gev_al2_cv7emstd_1umstep_tv31_hv1_15_t0.roo
t
  KEY: TTree    lxtsim;1        Bremsstrahlung photons at IP
  KEY: TTree    Tracks;1        Tracks hitting volumes marked for track inte
rception
  KEY: TTree    Hits;1  Hits in sensitive detectors
  KEY: TTree    HitTracks;1     Tracks which produced hits in sensitive dete
ctors
  KEY: TTree    DetSettings;1   Sensitive detector settings
```

In [713…
```
hit_tree = background_file.Get('Hits')
detset_tree = background_file.Get('DetSettings')

print("hits variables: \n", [g.GetName() for g in hit_tree.GetListOfBranches()]
print("det variables: \n", [g.GetName() for g in detset_tree.GetListOfBranches(
```

```
hits variables:
 ['eventid', 'detid', 'layerid', 'cellx', 'celly', 'edep', 'hitid', 'track_l
ist', 'trackx', 'tracky', 'trackz', 'weight']
det variables:
 ['detid', 'det_name', 'layerid', 'n_cell_x', 'n_cell_y', 'size_x', 'size_
y', 'size_z', 'translation_x', 'translation_y', 'translation_z', 'e_phi', 'e
_theta', 'e_psi']
```

Geometry part (having a look on the basic detector variables)

In [716…

```
for i in range(10,52):
    detset_tree.GetEntry(i)
    print(detset_tree.detid, ' ',
          detset_tree.det_name, ' ',detset_tree.n_cell_x, ' ', detset_tree.tran
          detset_tree.size_x,        detset_tree.n_cell_y, ' ', detset_tree.tra
          detset_tree.size_y,detset_tree.layerid, ' ', detset_tree.translation_
```

```
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 0   4258.54   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 1   4263.04199999
99995   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 2   4267.544   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 3   4272.04599999
9999   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 4   4276.548   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 5   4281.05   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 6   4285.552   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 7   4290.054   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 8   4294.556   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 9   4299.058   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 10   4303.5599999
999995   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 11   4308.062
0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 12   4312.5639999
99999   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 13   4317.066
0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 14   4321.5679999
99999   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 15   4326.07   0.
0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 16   4330.572
0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 17   4335.074
0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 18   4339.576
0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 19   4344.0779999
999995   0.0
2000   ECalSensorve   110   304.13   550.0 11   0.0   55.0 20   4348.58   0.
0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 0   4258.54   0.
0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 1   4263.0419999
999995   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 2   4267.544
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 3   4272.0459999
99999   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 4   4276.548
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 5   4281.05   0.
0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 6   4285.552
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 7   4290.054
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 8   4294.556
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 9   4299.058
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 10   4303.559999
9999995   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 11   4308.062
```

```
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 12   4312.563999
999999   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 13   4317.066
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 14   4321.567999
999999   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 15   4326.07
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 16   4330.572
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 17   4335.074
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 18   4339.576
0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 19   4344.077999
9999995   0.0
2001   ECalSensorve   110   -304.13   550.0 11   0.0   55.0 20   4348.58
0.0
```

In [717…

```
layer_zpos = []
for i in range(60):
    detset_tree.GetEntry(i)
    if detset_tree.detid==2000:
        layer_zpos.append(detset_tree.translation_z)
print("len: ",len(layer_zpos))
print("Z positions [mm]: ",layer_zpos)
hit_tree.GetEntry(11)
celly=hit_tree.GetListOfBranches()
print("celly:", celly[4])
```

```
len:  21
Z positions [mm]:  [4258.54, 4263.0419999999995, 4267.544, 4272.04599999999
9, 4276.548, 4281.05, 4285.552, 4290.054, 4294.556, 4299.058, 4303.559999999
9995, 4308.062, 4312.563999999999, 4317.066, 4321.567999999999, 4326.07, 433
0.572, 4335.074, 4339.576, 4344.0779999999995, 4348.58]
celly: Name: celly Title: Hits
```

In [718…

```
detset_tree.GetEntry(11)
edge_dist_x = detset_tree.translation_x - detset_tree.size_x/2.0
print("Distance from center to the edge in x [mm]: ", edge_dist_x)
```

```
Distance from center to the edge in x [mm]:  29.129999999999995
```

In [719…

```
magnet_center_z = 2050.0 #mm
magnet_length = 1029.0 #mm

cell_size_x = 5 #mm
```

In [ ]:

This part is just calculation of the Energy of the particle after going through the magnet

$$R_g = \frac{1}{ecB}\sqrt{E^2-m_e^2c^2}$$$$tg(\theta)=\frac{\sqrt{R_g^2-l^2}}{l}$$$$x = R_g - \sqrt{R_g^2-l^2} + \frac{z-m-l/2}{tg(\theta)} $$

where, $ R_g $ is the radius of curvature of the trajectory of the electron in the magnet, B is the magnetic field inside homogeneous magnet, $ e $ is the charge of the electron, $ c $ is the speed of light, $ E, m_e $ is the energy and mass of the electron, $ tg(\theta) $ is the angle at which the particle exits the magnet, l is the thickness of the magnet, $ m $ is the distance from the center of the magnet to the point of interaction, $ z $ is the distance from the point of interaction to the point where we count position of the particle $ x $.

In [720…

```
import numpy as np
```

```
import math

# simulation parameters:
cm = 0.01 # cm in m
fMagnetFieldValue = 1.4 #Tesla
fMagnetSizeZ = 100.0*cm
fMagnetZPos = 100.0*cm + 150.0*cm/2.0

#constants:
mass_e = 9.11E-31 # kg
e = 1.60217662E-19 # Coulombs
mc_gev = 0.511 # MeV
c = 299792458

# direct function (vectorized)
def calculating_x(E,z0): # signature -> (array [MeV], scalar [mm])
    z0 = z0/1000 # to meters
    R = mass_e * c / (e * fMagnetFieldValue ) * np.sqrt((E/mc_gev)**2 - 1.0)
    dx = R - np.sqrt( R**2 - fMagnetSizeZ**2)
    tan = np.sqrt(R**2 - fMagnetSizeZ**2)/fMagnetSizeZ
    return ( dx + ( z0 - fMagnetZPos - fMagnetSizeZ / 2)/ tan ) * 1000

# inverse function (numerical)
from scipy.optimize import fsolve
def calculating_E(x,z0): # signature -> (scalar [mm],scalar [mm])
    x = x/1000 # to meters
    z0 = z0/1000 # to meters
    l = fMagnetSizeZ
    A = z0 - fMagnetZPos - fMagnetSizeZ/2.0
    func = lambda R : (x-R)*math.sqrt(R**2-l**2)+R**2-l**2-l*A
    R_i = 10
    R = fsolve(func, R_i)
    return math.sqrt( (R*e*fMagnetFieldValue/(mass_e*c))**2 + 1)*mc_gev

# Inverse function (analytical) obtained trying to evaluate real part of the co
# (I used Wolfram-Mathematica to evaluate this part from the 3rd power equation
'''
def calculating_E_wolf(x,z0): # signature -> (scalar [mm],scalar [mm])
    x = x/1000 # to meters
    z0 = z0/1000 # to meters
    l = fMagnetSizeZ
    A = z0 - fMagnetZPos - fMagnetSizeZ/2.0
    R = (1/(6*x))*(2*A*l + l**2 + x**2 + \
    2*np.sqrt(4*A**2*l**2 + l**4 + 14*l**2*x**2 + x**4 + 4*A*l*(l**2 + x**2))*
    np.cos((1/3)*np.arctan2(l**3*(2*A + l)**3 - 3*l**2*(14*A**2 + 20*A*l + 11*l
    6*l*x*np.sqrt(3*l*(2*A + l)**3 + 3*(A**2 + 10*A*l - 2*l**2)*x**2 + 3*x**4)*
    return np.sqrt( (R*e*fMagnetFieldValue/(mass_e*c))**2 + 1)*mc_gev

'''
```

Out[720…

```
'\ndef calculating_E_wolf(x,z0): # signature -> (scalar [mm],scalar [mm])\n
x = x/1000 # to meters\n    z0 = z0/1000 # to meters\n    l = fMagnetSizeZ\n
A = z0 - fMagnetZPos - fMagnetSizeZ/2.0\n    R = (1/(6*x))*(2*A*l + l**2 + x
**2 +     2*np.sqrt(4*A**2*l**2 + l**4 + 14*l**2*x**2 + x**4 + 4*A*l*(l**2 +
x**2))*     np.cos((1/3)*np.arctan2(l**3*(2*A + l)**3 - 3*l**2*(14*A**2 + 20
*A*l + 11*l**2)*x**2 + 3*(2*A - 11*l)*l*x**4 + x**6,     6*l*x*np.sqrt(3*l*
(2*A + l)**3 + 3*(A**2 + 10*A*l - 2*l**2)*x**2 + 3*x**4)*np.abs((-l)*(A + l)
+ x**2)))) # Loss of percission is somewhere in functon\n    return np.sqrt(
(R*e*fMagnetFieldValue/(mass_e*c))**2 + 1)*mc_gev\n\n'
```

Counting the number of the particles

For this purpose we used the calculation of the particle flux by the cells, algorithm that allows you to count the number of particles very quickly, its essence is:

1) Knowing the coordinates of the cell center of the calorimeter, find the particle energy that is expected in that cell $E_{exp}(x_i,z_i)$.

2) Calculate the concentration of particles in this cell knowing the energy absorbed in it as: $N=\frac{E_i}{E_{exp}}$.

3) Sum up for all cells $N=\sum{\frac{E_i}{E_{exp}}}$ and find the total number of the particles.
\n

Expected energy in cells (after magnet)

In [721…
```
#import magnet_equesion #inverse function (numerical)
# signature of calculating_E -> (scalar [mm],scalar [mm])
```

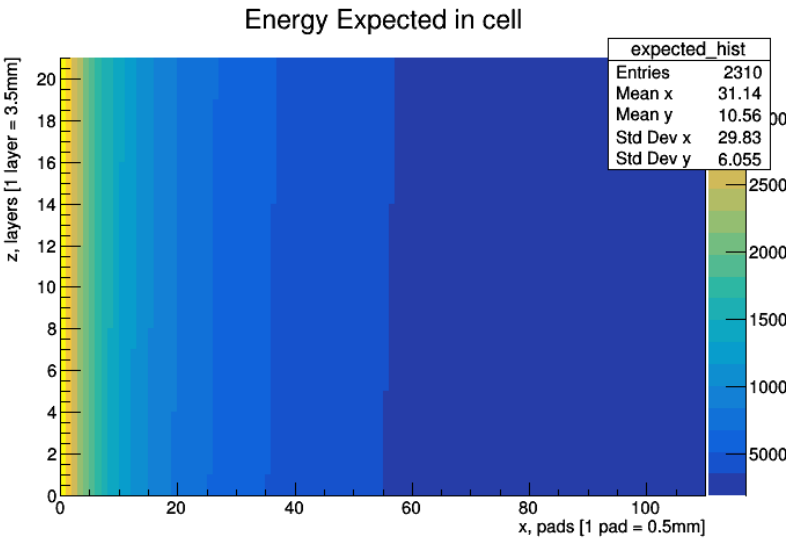
In [722…
```
# simulation parameters:
#magnet_equesion.fMagnetFieldValue = 1.0 #Tesla
#magnet_equesion.fMagnetSizeZ = magnet_length / 1000.0 # mm to m
#magnet_equesion.fMagnetZPos = magnet_center_z / 1000.0 # mm to m

#print("test expected [MeV] at (x,z)point: ",magnet_equesion.calculating_E(200.
```

In [723…
```
expected_E = np.zeros((21, 110, 11)) #Expected energy in cell in [MeV]
for i_z in range(21):
    for i_x in range(110):
        for i_y in range(11):
            expected_E[i_z,i_x,i_y] = calculating_E(i_x * cell_size_x + cell_si
                                                   layer_zpos[i_z])
expected_E.shape
```

Out[723… (21, 110, 11)

In [724…
```
# Just for illustration
#expected_Ezy = np.sum(expected_E,axis = 2)
#print(expected_Ezy)
expected_Ezy = expected_E[:,:,5]
print(expected_Ezy.shape)

def fill_hist(h, a):
    [[h.SetBinContent(j+1,i+1,a[i][j]) for i in range(a.shape[0])] for j in ran

expected_hist = TH2F('expected_hist', 'Energy Expected in cell', 110, 0, 110, 2
fill_hist(expected_hist,expected_Ezy)

cv = TCanvas()
expected_hist.SetTitle("Energy Expected in cell;x, pads [1 pad = 0.5mm];z, laye

expected_hist.Draw("colz")
# cv.SetLogz(True)
cv.Draw()
```

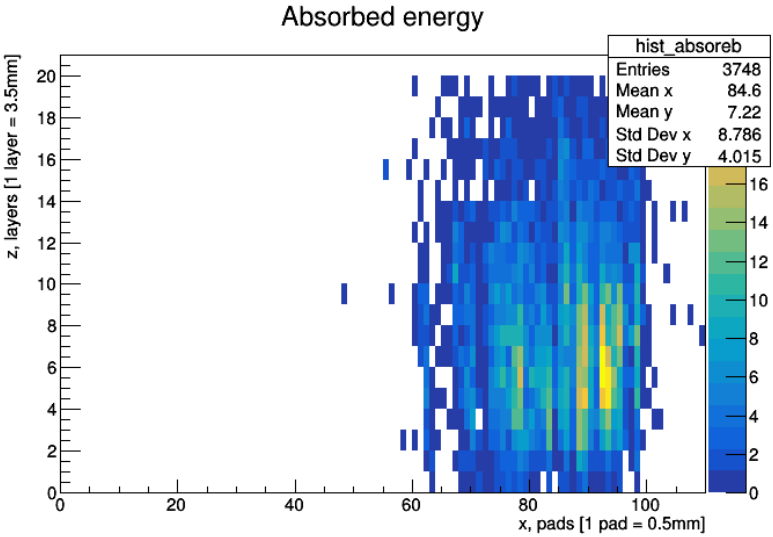
```
(21, 110)
```



Read Root files and collect hits

In [725…
```
hist_absoreb = TH2F('hist_absoreb', '', 110, 0, 110, 21, 0, 21)
```

In [726…
```
cut = '(detid==2001&&(celly>=2&&celly<=11)&&layerid<20)'
#cut='(detid==2000||detid==2001)&&(celly>=2&&celly<=9)&&layerid<21'
#nocut = '(detid==2000)'
cv2 = TCanvas()
# hit_tree.Project('layerid:cellx', 'layerid:cellx')
hit_tree.Draw('layerid:cellx>>hist_absoreb', cut+'*1000*edep',"goff")
hist_absoreb.SetTitle("Absorbed energy;x, pads [1 pad = 0.5mm];z, layers [1 lay

hist_absoreb.Draw("colz")
cv2.Draw()
```



Number of particles in absorbed energy

In [727…
```
callibr = 86.44
```

In [728…
```
counts = 0
absE = 0
y_to_take = 6
for i in range(expected_Ezy.shape[0]):
    for j in range(expected_Ezy.shape[1]):
        counts += hist_absoreb.GetBinContent(j+1,i+1)*callibr/expected_E[i][j][
        absE += hist_absoreb.GetBinContent(j+1,i+1)*callibr
print("number of counted particles: ",counts)
print("absorbed energy [MeV]: ", absE)
```

```
number of counted particles:  89.99839387799786
absorbed energy [MeV]:  214207.08979565918
```

Spectral density

Consider the procedure for reproducing the spectrum of particles detected in a calorimetric system.

1) We have cell with coordinates (x_i,y_i,z_i) and corresponding absorbed energy in cell E_{dep}

2) We know that the particle that went through the magnet is expected to have Energy $E_{expected}$ (it isalready calculated analytically in previous section)

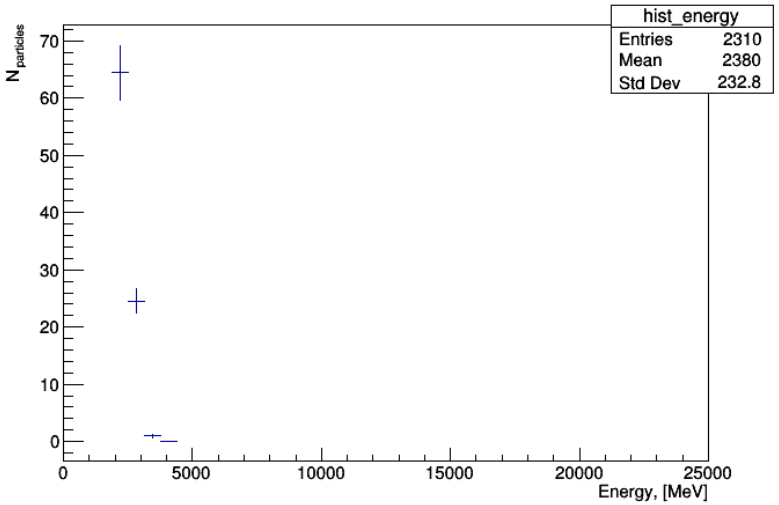
3) We fill the histogram with $E_{expected}$ (putting this values into a bin which corresponds to such energy) with weight $W=N=E_{dep}/E_{expected}$

4) Repeat for all cells

In [729…
```
hist_energy = TH1F('hist_energy', '', 40, 0, 25000)
```

In [730…
```
for i in range(expected_Ezy.shape[0]):
    for j in range(expected_Ezy.shape[1]):
        hist_energy.Fill(expected_E[i][j][y_to_take],hist_absoreb.GetBinContent
```

In [731…
```
cv3 = TCanvas()
#hit_tree.Project('layerid:cellx', 'layerid:cellx')
hist_energy.SetTitle(";Energy, [MeV];N_{ particles}")
hist_energy.Draw()
cv3.Draw()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: